ELDERLY CARE AND ASSISTANCE BOOKING PLATFORM

HEW ZI XUAN

UNIVERSITI TUNKU ABDUL RAHMAN

Elderly Care and Assistance Booking Platform

HEW ZI XUAN

A project report submitted in partial fulfilment of the requirements for the award of Bachelor of Science (Honours) Software Engineering

Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

October 2024

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature	:	[illegible]
Name	:	Hew Zi Xuan
ID No.	:	2005764
Date	:	25-2-2024

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"TITLE TO BE THE SAME AS FRONT COVER, CAPITAL LETTER, BOLD"** was prepared by **STUDENT'S NAME** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Software Engineering with Honours at Universiti Tunku Abdul Rahman.

Approved by,

Signature	:	[illegible]
Supervisor	:	Nawaf Hassan Mohammed Mohsen Shrifan
Date	:	1/10/2024

Signature	:	
Co-Supervisor	:	
Date	:	

The copyright of this report belongs to the author under the terms of the copyright Act 1987 as qualified by Intellectual Property Policy of Universiti Tunku Abdul Rahman. Due acknowledgement shall always be made of the use of any material contained in, or derived from, this report.

© 2024, Hew Zi Xuan. All right reserved.

ACKNOWLEDGEMENTS

I would like to thank everyone who helped get this project done. Special thanks to my supervisor, Dr. Nawaf Hassan Mohammed Mohsen Shrifan, who provided clear direction and assistance during the research. Furthermore, I am grateful to my friends and family for their support and understanding. This project would not have been achieved without their collaborative efforts.

ABSTRACT

The Elderly Care and Assistance Booking Platform project aims to address the urgent need in Malaysia for an interactive platform that enables customers to access care services and facilitates direct interaction with caregivers. Currently, there is a large gap in such platforms, making this project highly relevant and impactful. The project focuses on developing a platform that enables registration, strong communication channels, and comprehensive search capabilities for customized care services for clients and caregivers. The project adopts a Rapid Application Development (RAD) approach to ensure agility and adaptability to meet the changing needs of the care sector. This approach helps to quickly adjust and respond to user preferences and emerging needs. By leveraging insights gained from the existing platform, the project aims to enhance the well-being of clients and caregivers throughout Malaysia. To ensure the platform meets the highest standards of quality and functionality, four types of testing will be conducted. The testing includes unit testing, feature testing, black-box testing, and user acceptance testing (UAT). These tests will cover various aspects of the platform from individual components to the overall user experience. An attractive feature of the platform is its AI recommendation system. This system is designed to personalize care service suggestions based on user preferences and needs. This system provides tailored recommendations and enhances the matching process between clients and caregivers. Additionally, the platform includes a GPT-powered description function that generates detailed and context-aware descriptions of services. This further improves the user experience and facilitates informed decision-making. For this project, Laravel serves as the project's fundamental framework. Laravel ensures safe data management, user authentication, and scalability via its Model-View-Controller (MVC) architecture. This strong architecture facilitates the platform's expansion and adaptability which gives a safe and scalable response to the changing needs of elderly care service.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS / ABBREVIATIONS

DOSM	Department of Statistic Malaysia
MVC	Model-View-Controller
MYSQL	My Structured Query Language
RAD	Rapid Application Development
PHP	Hypertext Preprocessor
DDoS	Distributed denial-of-service
BCrypt	BlowFish &Crypt
CSS	Cascading Style Sheets
JSX	JavaScript XML
ES6	ECMAScript 6
HTML	Hypertext Markup Language
VS Code	Visual Studio Code
IDE	Integrated Programming Environment
RAM	Random Access Memory
PCI DSS	Payment Card Industry Data Security Standard
ERD	Entity Relationship Diagram
API	Application Programming Interface
ORM	Object-Relational Mapping
WBS	Work Breakdown Structure
UML	Unified Modeling Language
UUID	Universally Unique Identifier
PDF	Portable Document Format
CSV	Comma Separated Value
FK	Foreign Key
SMLP	Simple Mail Transfer Protocol
TLS	Transport Layer Security
AJAX	Asynchronous JavaScript and XML
TF-IDF	Term Frequency-Inverse Document Frequency
NLP	Natural Language Processing
UAT	User Acceptance Test

LIST OF APPENDICES

CHAPTER 1
INTRODUCTION

1.1 Project Background

Malaysia’s population is aging, the most recent statistic from DOSM indicate that the percentage of Malaysia’s population that is 65 years of age or over climbed from 7.2% to 7.4% with roughly 2.5 million people (Columnist, 2023). It is expected that by 2023, Malaysia’s aging population will account for 15% of the total population, highlighting the urgent need for a comprehensive elderly care platform to maintain the dignity of the elderly and improve their quality of life (Meera Murugesan, 2021).

Elderly individuals may experience difficulties with daily activities such as clothing, grooming, bathing, and meal preparation as they get older. Sometimes, as the caregiver, we might handle every single one of these tasks (Scnova, 2024). However, due to work and study commitments, providing constant care and attention to our elderly family members becomes challenging, especially when we have other responsibilities demanding our time and energy. Therefore, we might think to have a caregiver to help with these tasks but finding a reliable caregiver for these tasks is also not easy. The process of searching for suitable caregivers can be time-consuming. So that, this is why an elderly care and assistance booking platform is extremely important.

The elderly care and assistance booking platform offer a range of benefits for both the elderly and their families. The first benefit is accessibility and convenience, enabling users to easily access the platform by using their tablets or computers. This platform allows users to choose from different kind of services that best meet their needs. Additionally, users can communicate directly with their caregiver through chat features in this platform.

The second benefit of this platform is affordability and flexibility. This platform reduces the cost of care services since user is direct pay to the caregiver then eliminating the need for intermediary agencies that charge commissions.

In addition, users are free to cancel the services whenever they want since there is not bound with the long-term contract.

By performing background check and letting users to access the caregivers' performance and feedback by other users, this platform further guarantees the reliability and quality of the caregiver. Moreover, this platform provides customer support that allow user to communicate with the administrator in case of any issues or problems therefore enhancing user satisfaction.

Furthermore, the platform offers a wide range of care services, including medical care, and food care etc. Users can easily find and compare the caregiver based on price, reviews, and services offered. This streamlining the process of finding quality care and reducing uncertainty.

By developing such a platform, family members can focus on their work or studies while efficiently managing their time. Simultaneously, elderly individuals receive a quality care and leading to an improved quality of life. Thus, this platform benefits multiple parties, including elderly individuals, caregivers, and family members (FasterCapital, 2024).

1.2 Problem Statement

After conducting a review on existing elderly care and assistance booking platforms, three main problems have been identified in the Malaysia elderly care platform.

1.2.1 Lack of a Booking Platform that Allow Registration for Both Clients and Caregivers

One of the significant problems identified in the platform is the absence of a comprehensive booking platform that allows registration for both users and caregivers. This platform lacks a registration process for both clients and caregivers. Some of the existing platforms focus on booking consultations only. This restriction makes it more difficult for caregivers to advertise their services and highlight to users their credentials, experience, and availability. This problem restricts consumers' options for choosing a caregiver as well as caregivers' chances to connect with a larger user base.

1.2.2 The Lack of Communication Channel Between Families and Caregiver

The lack of a communication channel between families and caregivers is another issue observed in Malaysia elderly care platforms. Many platforms in Malaysia do not provide chat features that enable direct communication between users and caregivers. In some of the existing platforms, users can only communicate with the platform assistance by using the live chat box. This restricts the direct communication between the client and caregiver. Without a chat feature, clients may face difficulties in discussing specific care requirements with the caregiver. This will result in misunderstandings or delays in service delivery. Thus, clients might be dissatisfied and become not confident about the services that provided in this platform.

1.2.3 The Lack of a Search Function with Filtering Capabilities

The lack of a search function with filtering capabilities is also one of the problems in many Malaysia elderly care platform. Without this feature, users are unable to specify their exact service requirements and preferences. This will be leading to a lack of caregiver matching list. Due to this problem, users may struggle to find a caregiver who meet their needs then users will waste more time to browsing irrelevant profiles and become frustration. This problem will affect both users and caregivers since users may have to navigate through many profiles manually, while caregivers may struggle to attract the right clients who match their expertise and availability.

1.3 Project Objectives

The objective of this project is to develop an elderly care and assistance booking platform that improving the overall quality of care provided to elderly people. The three main objectives are:

1. To develop an integrated booking platform that allow registration for both clients and caregiver so that clients can find the services they need, and caregiver can promote their services.
2. To implement a chat channel that allows clients to communicate directly with caregivers for information exchange.
3. To design a platform with search function with filtering capabilities to generate caregiver matching lists based on user preferences and requirements.

1.4 Project Solution

To develop an elderly care and assistance booking platform that meets the project objectives and solves the project problem described in the previous section, the Laravel platform was chosen as the development framework due to its blade templating engine, complete authentication system, and rapid development. Model-View-Controller (MVC) architecture is implemented by Laravel. This framework allows for both front-end and back-end system development. The details of the system architecture design will be listed in Section 5.1.

1.5 Project Approach

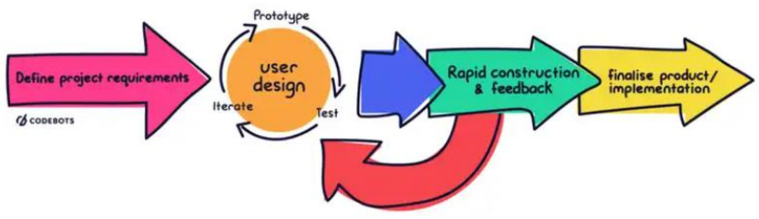

Define project requirements
CODEBOTS
Prototype
user design
Iterate
Test
Rapid construction & feedback
finalise product/ implementation

Figure 1.1: Step of Rapid Application Development methodology (Chien, 2020).

This project will adopt Rapid Application Development (RAD) methodology to develop an elderly care and assistance booking platform. This methodology's emphasis on rapid prototyping and iterative cycles aligns perfectly with creating a user-friendly platform for elderly individuals, family members, and caregivers. By swiftly prototyping and refining key features like caregiver matching and communication tools, the project can incorporate valuable user feedback early on, ensuring a seamless user experience (Chien, 2020). Since this is an individual approach, it allows for focused decision-making and flexibility in addressing project requirements, leading to a timely launch of the platform with iterative updates based on real-world usage and evolving user needs.

1.6 Scope and Limitation of the Study

1.6.1 Target Users

1.6.1.1 Elderly Individuals

This platform is mostly used by elderly individuals who require various care services to assist them with daily tasks and improve their quality of life. They can use this platform to search for and book caregivers who suite their specific needs.

1.6.1.2 Family Members

Another important user group consists of family members who oversee and manage the care of their elderly family. This covers adult children, partners, siblings, and other family members who contribute significantly to the caregiving process. They may search and book caregiver, communicate with caregivers, and provide feedback or ratings based on their experience.

1.6.1.3 Caregivers

Caregivers are the one of the essential users of this platform. They create their own profile by filling up their qualifications, experience, availability, service offered to promote their services. They can also use this platform to connect with the care receiver by using the chat channel.

1.6.1.4 Administrators

The administrator oversees the overall functioning of the platform. They manage user account including account of elderly individuals, family members as well as caregiver account. Administrator can also communicate with the users.

1.6.2 Modules Covered

Users in this section consider as elderly individuals and family members.

1.6.2.1 User and Caregiver Registration and Login

This platform allows users and caregivers to register their account on the platform and login to the platform using their criteria.

1.6.2.2 Profile Management

Users and caregivers can manage their profiles by updating their personal information such as name, email, contact numbers, etc. The users can view the caregiver's profile as same as the caregivers.

1.6.2.3 Transaction History Module

Users can view the transaction history they made before, and caregivers can view the payment received history. Users and caregivers can generate invoice and export CSV of any transaction.

1.6.2.4 Service Seach and Filtering

Users can search for care services based on specific criteria. Advanced filtering features help users find the most suitable caregivers. Users can also ask for service recommendations by using the recommendation chatbot.

1.6.2.5 Booking Management

Users can book for a service directly through this platform. They can manage the booking later including view the booking, update the booking and cancel the booking. After the user makes a booking, the caregiver can accept or decline the booking. If the caregiver has accepted the booking, the users may proceed to make the payment.

1.6.2.6 Care Services Management

Caregiver can promote their services on this platform. They can manage their services later including view the service, update the service and delete the service. Besides, the administrator can delete a service that was created by the caregiver.

1.6.2.7 Account Management

Administrator can manage the user account by viewing and deleting the user and caregiver’s account.

1.6.2.8 Communication Channel

This platform establishes chat features that allows direct communication between users and caregiver. Administrator can also use this tool to communicate to users and caregivers.

1.6.2.9 Feedback System

This platform allows users to provide feedback to caregivers after conducting the care services based on their experiences and will enable caregivers to view this feedback. This module helps accountability within the platform.

1.6.3 Modules Not Covered

Due to the limitation of time and technologies, payment processing is not covered in this project.

1.6.4 Assumptions of this project

i. The users and caregivers are assumed to have a basic level of digital literacy necessary to navigate the platform.

ii. Assumed that the web application function seamlessly across different operating systems such as Windows, macOS and Linux, enabling users to access it from any device with an internet connection.

CHAPTER 2
LITERATURE REVIEW

2.1 Introduction

This chapter has four main sections, which are Section 2.2, Section 2.3, Section 2.4, and Section 2.5. Section 2.2 discusses the background of the existing elderly care platforms and their strengths and weaknesses. Next, Section 2.3 discusses the software development methodologies with their strengths and weaknesses. Other than that, Section 2.4 discusses web development tools such as development framework, code editors, database along with their advantages and disadvantages. Section 2.5 is about the conclusion of each session.

2.2 Elderly Care platforms

Due to the limited number of elderly care and assistance booking platforms that have developed in Malaysia, I reviewed two platforms from other countries and two from Malaysia.

2.2.1 Care.com

Care.com was founded in 2006 and launched in 2007. In 2012, Care.com expanded its reach by launching in the United Kingdom and Canada while also acquiring the Berlin-based Betreut.de. This strategic move broadened its footprint and solidified its position as a leading online resource connecting families with caregivers across different regions (Weingus, 2024).

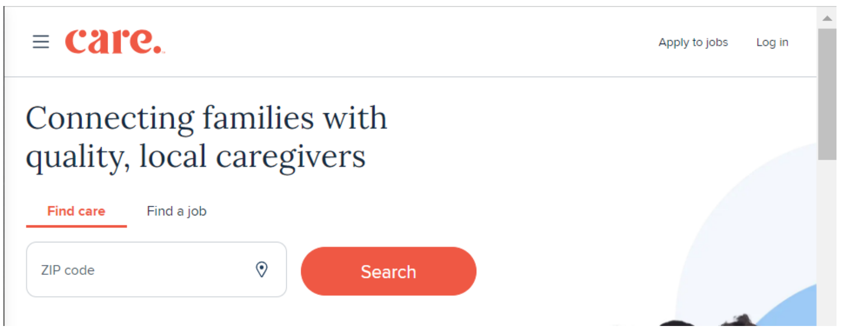
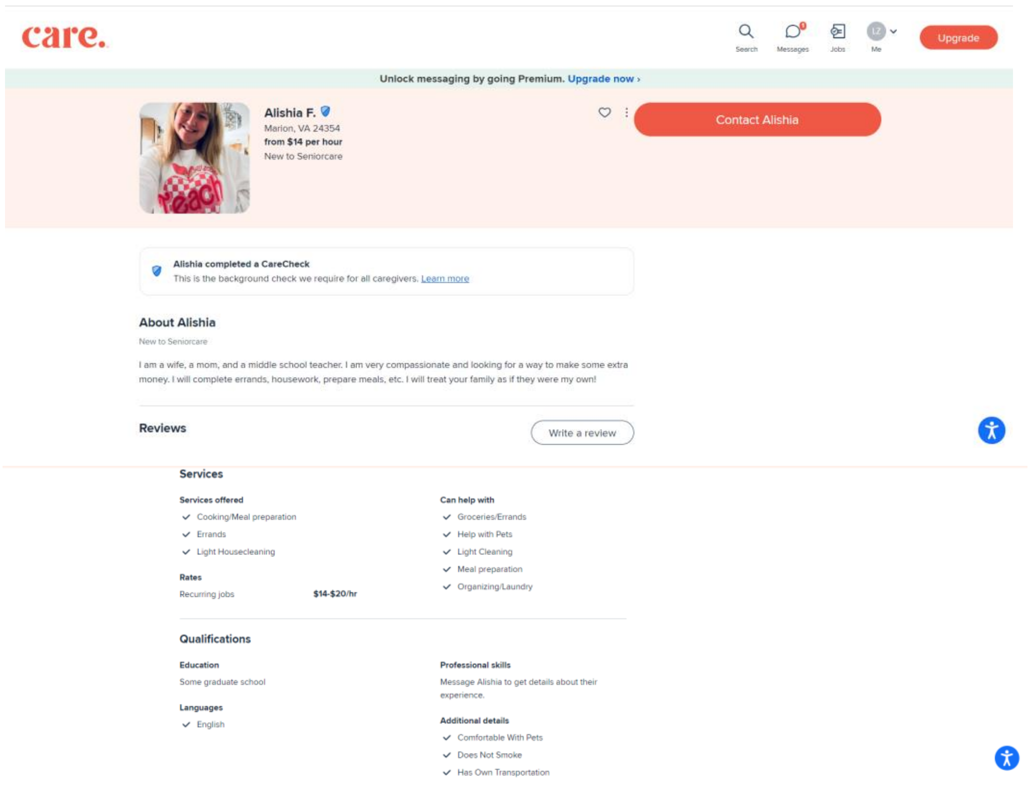


Figure 2.1: Home Page of Care.com

The figure above shows that Care.com is a platform that allow users to apply to jobs or log in to find a care service. This boots the interaction between clients and caregivers.



Unlock messaging by going Premium. Upgrade now ›

Alishia F.
Marion, VA 24354
from $14 per hour
New to Seniorcare

Alishia completed a CareCheck
This is the background check we require for all caregivers. Learn more

About Alishia
New to Seniorcare
I am a wife, a mom, and a middle school teacher. I am very compassionate and looking for a way to make some extra money. I will complete errands, housework, prepare meals, etc. I will treat your family as if they were my own!

Reviews

Services

Services offered
- ✓ Cooking/Meal preparation
- ✓ Errands
- ✓ Light Housecleaning

Can help with
- ✓ Groceries/Errands
- ✓ Help with Pets
- ✓ Light Cleaning
- ✓ Meal preparation
- ✓ Organizing/Laundry

Rates

Recurring jobs	$14-$20/hr

Qualifications

Education
Some graduate school

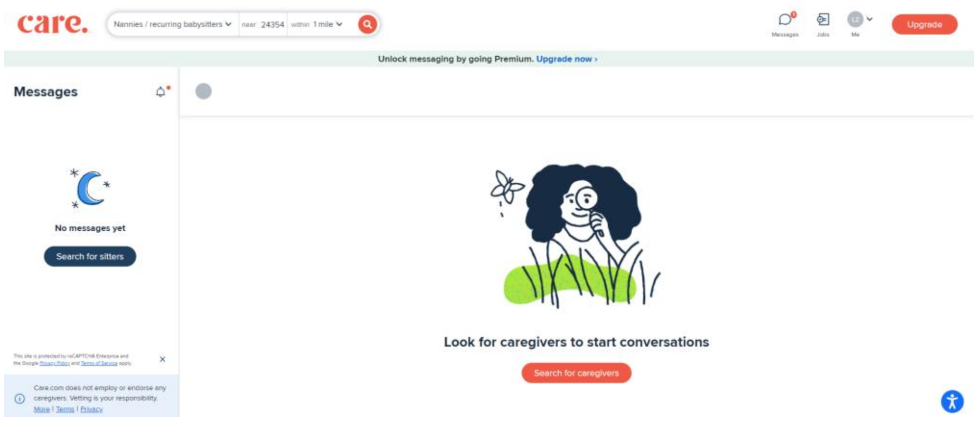
Languages
- ✓ English

Professional skills
Message Alishia to get details about their experience.

Additional details
- ✓ Comfortable With Pets
- ✓ Does Not Smoke
- ✓ Has Own Transportation

Figure 2.2: Caregiver' profile page

Care.com has various strengths that dramatically improve the care experience for the users. One significant aspect is the detailed caregiver profiles. Care.com provides detailed information on caregivers' backgrounds, including the services provided, rates, qualifications, languages spoken, and professional skills. These profiles provide users confidence by offering comprehensive data, which empowers them to make well-informed decisions based on their individual needs and preferences. To guarantee the safety and security of elderly individuals or family members who use the platform, Care.com also thoroughly examines the backgrounds of carers. Care.com also allows users to provide a review after they use the services. This review may help clients make decisions when booking a service.

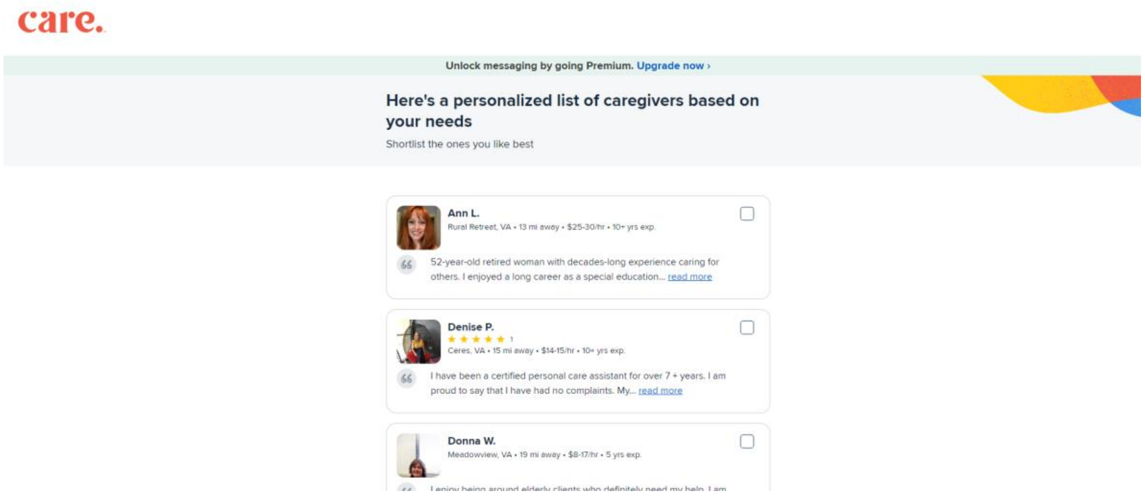

care.
Nannies / recurring babysitters
near 24354
within 1 mile
Messages
Jobs
Me
Upgrade
Unlock messaging by going Premium. Upgrade now ›
Messages
No messages yet
Search for sitters
Look for caregivers to start conversations
Search for caregivers

This site is protected by reCAPTCHA Enterprise and the Google Privacy Policy and Terms of Service apply.

Care.com does not employ or endorse any caregivers. Vetting is your responsibility. More | Terms | Privacy

Figure 2.3: Messaging System Page

Furthermore, Care.com has a message system that allows communication between clients and caregivers. This seamless communication platform will enable them to communicate with each other on time. This system also helps exchange critical information and discuss individual care requirements while maintaining privacy and security.



Here's a personalized list of caregivers based on your needs

Shortlist the ones you like best

Ann L.
Rural Retreat, VA • 13 mi away • $25-30/hr • 10+ yrs exp.
52-year-old retired woman with decades-long experience caring for others. I enjoyed a long career as a special education... read more

Denise P.
★★★★★ 1
Ceres, VA • 15 mi away • $14-15/hr • 10+ yrs exp.
I have been a certified personal care assistant for over 7 + years. I am proud to say that I have had no complaints. My... read more

Donna W.
Meadowview, VA • 19 mi away • $8-17/hr • 5 yrs exp.

Figure 2.4: Matching Caregivers' Page

Moreover, Care.com offers a personalized list of caregivers based on the client's needs. This customised approach helps the client to select a caregiver more easily and quickly. This also ensures that caregivers can align their services with the right user base.

However, Care.com has its weaknesses too. The platform is subscription-based so clients must pay membership fees to access the full range of features. This may reduce user usage since not all users might use a platform that requires a fee. They may use another free app instead of this.

2.2.2 Pillar

Pillar is Malaysia's No.1 home caregiver provider platform. Users can use this platform without registering an account. In this case, this platform eliminates the need for registration for both clients and caregivers.

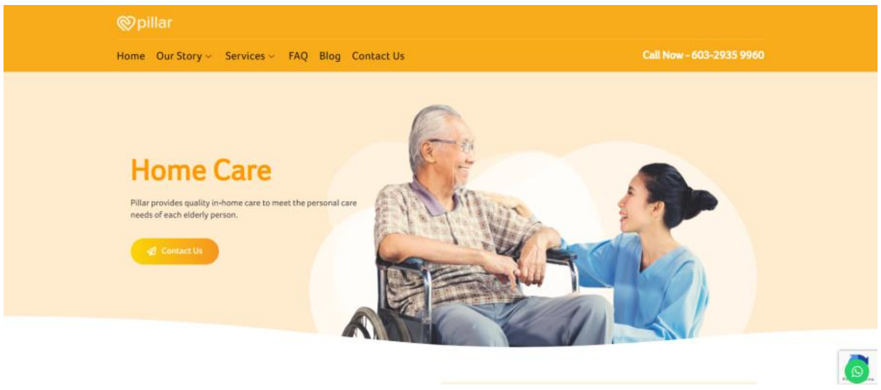

pillar
Home
Our Story
Services
FAQ
Blog
Contact Us
Call Now - 603-2935 9960
Home Care
Pillar provides quality in-home care to meet the personal care needs of each elderly person.
Contact Us

Figure 2.5: Home Page of Pillar

One of Pillar's most significant features is its user-friendly interface. This can be proven by its ease of use. Users may simply explore the platform and find important information about its diverse range of services. This is because its page has a clever design and straightforward navigation. This emphasis on user experience enhances interaction and engagement.

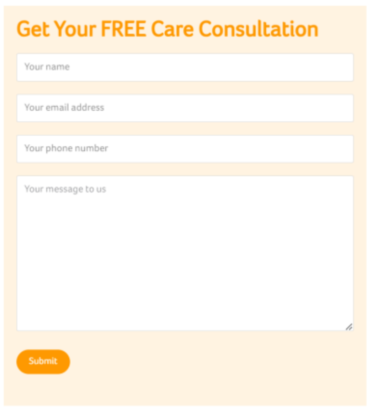
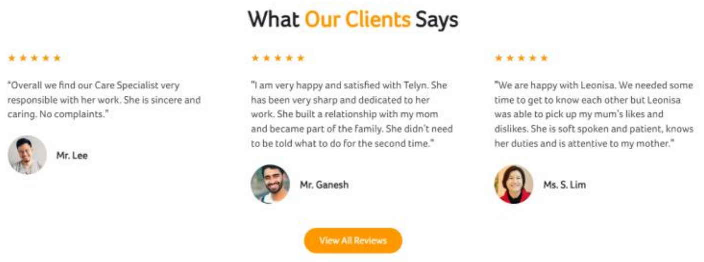


Figure 2.6: Care Consultation Page

Additionally, users can submit a Contact Us form by entering their detailed information and the message to the platform. This client-centric approach strategy ensures that users’ needs and requirements are clear from the start. These free consultations make customers feel respected and supported.


What Our Clients Says

★★★★★
"Overall we find our Care Specialist very responsible with her work. She is sincere and caring. No complaints."
Mr. Lee

★★★★★
"I am very happy and satisfied with Telyn. She has been very sharp and dedicated to her work. She built a relationship with my mom and became part of the family. She didn't need to be told what to do for the second time."
Mr. Ganesh

★★★★★
"We are happy with Leonisa. We needed some time to get to know each other but Leonisa was able to pick up my mum's likes and dislikes. She is soft spoken and patient, knows her duties and is attentive to my mother."
Ms. S. Lim

View All Reviews

Figure 2.7: Review and Rating System Page

Furthermore, users can view ratings and feedback from other users for any service they have used. This feedback system will help users understand the actual experiences of other users. This can help them make an informed decision when they want to book a service.

However, Pillar's platform exhibits several weaknesses. One of them is a lack of communication channels that allow users to directly communicate with caregivers. This lack of direct communication channels can lead to delays and misunderstandings between the client and caregiver. This will impact the overall quality of care delivery and the user experience.

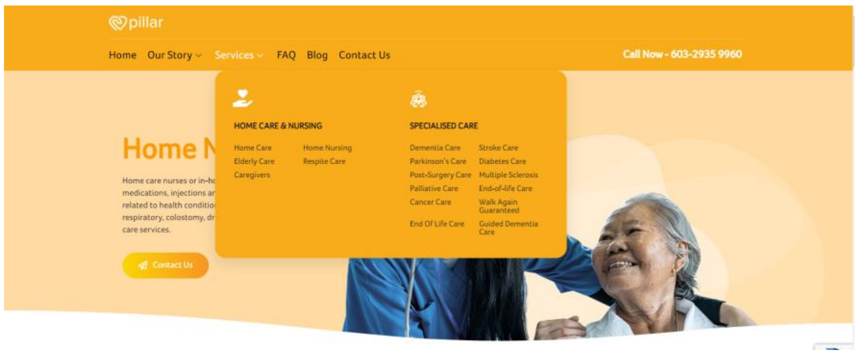
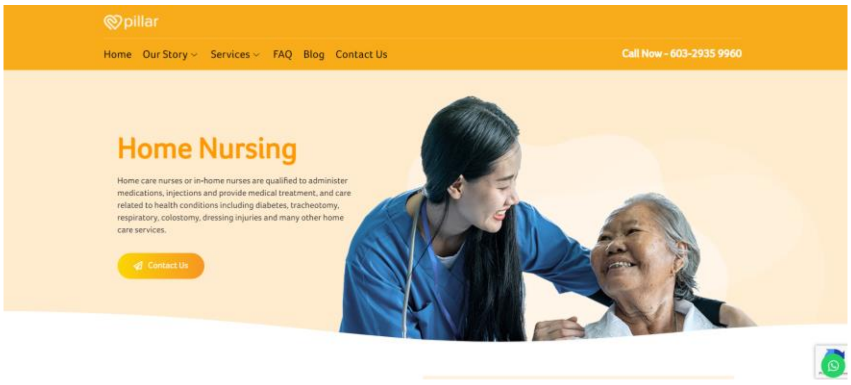


Figure 2.8: List of Services Provided



Home Nursing

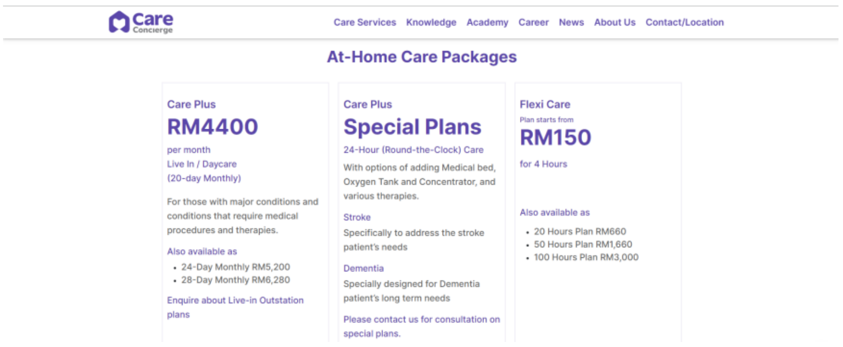
Home care nurses or in-home nurses are qualified to administer medications, injections and provide medical treatment, and care related to health conditions including diabetes, tracheotomy, respiratory, colostomy, dressing injuries and many other home care services.

Figure 2.9: Description of the Services Provided

In addition, Pillar's platform focuses on showcasing care services rather than providing profiles of caregivers. Users can access a list of services provided by the platform. For example, a brief description of the scope of these services is provided. In this case, the platform may lack detailed information about the caregiver's details and information. This limitation may make it more difficult for clients to select the best caregiver for their needs and make a fully informed choice.

2.2.3 CareConcierge

CareConcierge is a platform that offers home care services for the elderly. This platform meets the varied needs of the elderly who require assistance in their own homes.



At-Home Care Packages

Care Plus
RM4400
per month
Live In / Daycare
(20-day Monthly)
For those with major conditions and conditions that require medical procedures and therapies.
Also available as
- 24-Day Monthly RM5,200
- 28-Day Monthly RM6,280

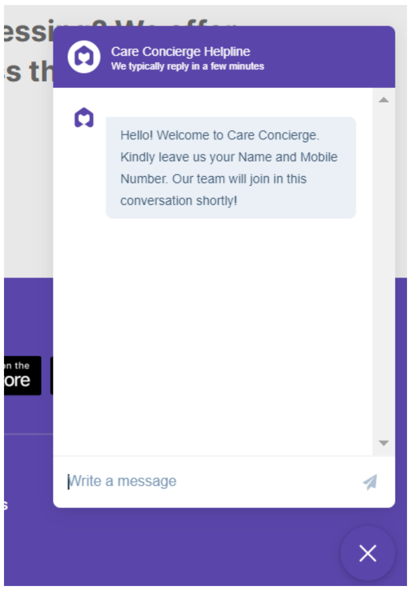
Enquire about Live-in Outstation plans

Care Plus
Special Plans
24-Hour (Round-the-Clock) Care
With options of adding Medical bed, Oxygen Tank and Concentrator, and various therapies.
Stroke
Specifically to address the stroke patient's needs
Dementia
Specially designed for Dementia patient's long term needs
Please contact us for consultation on special plans.

Flexi Care
Plan starts from
RM150
for 4 Hours
Also available as
- 20 Hours Plan RM660
- 50 Hours Plan RM1,660
- 100 Hours Plan RM3,000

Figure 2.10: Care Packages Page

CareConcierge provides different types of care package plans. With these care plans, CareConcierge makes the process of selecting care services more effective. This allows users to choose plans that align with their requirements without the need for extensive comparisons of many services.


Care Concierge Helpline
We typically reply in a few minutes

Hello! Welcome to Care Concierge. Kindly leave us your Name and Mobile Number. Our team will join in this conversation shortly!

Figure 2.11: Live Chat Support Page

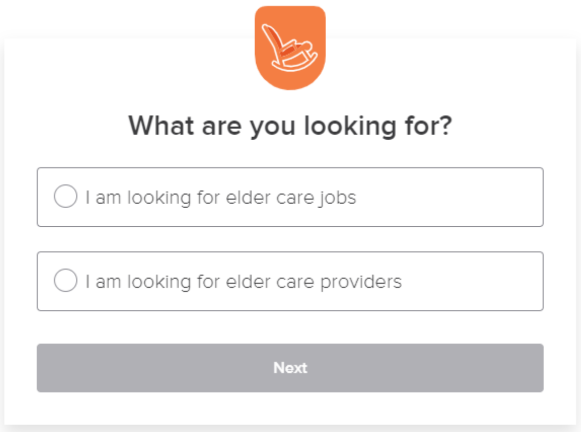
Additionally, CareConcierge enhances the customer experience with a live chat support option. The live chat feature on the platform allows users to ask and receive questions. This live chat will allow users to resolve their issues instantly, thereby enhancing the trust and confidence of users when using the platform.

However, there are various weaknesses on CareConcierge platform. This platform does not provide the caregiver's details and information for user view purposes. This will lead to user frustration since there is not sufficient information available regarding a caregiver's details. Users may find it difficult to evaluate the caregiver's eligibility. This restriction may prevent users from picking a caregiver who best meets their requirements and desires. Another weakness of CareConcierge is the lack of user reviews or feedback. Users may struggle to assess the quality and dependability of the platform's care services in the absence of feedback from past users.

Finally, CareConcierge’s lack of a direct booking feature may be problematic for some users. The platform allows users to make bookings via WhatsApp by navigating to it by clicking on the WhatsApp logo. However, some users may prefer to book directly through the website. Implementing a direct booking feature may improve platform customer satisfaction.

2.2.4 ElderCare

ElderCare is a platform that focuses on supporting elderly individuals in the Canada, United States, and the United Kingdom. This platform is designed specifically for elderly individuals by offering a range of services and resources to help them stay healthy and independent.

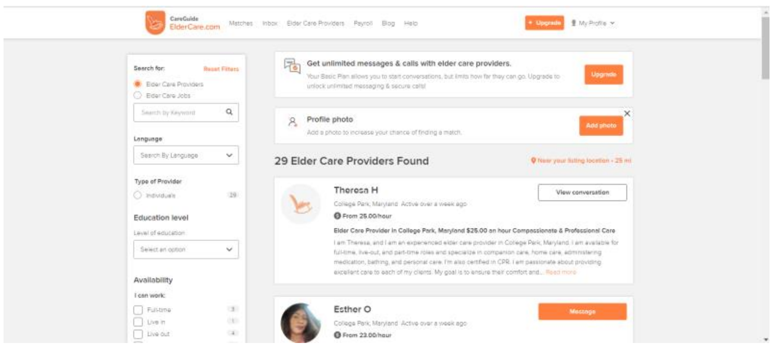

What are you looking for?

I am looking for elder care jobs

I am looking for elder care providers

Figure 2.12: Registration Page

On the Registration page, users are presented with two options as shown in Figure 2.12. By offering this registration choice, ElderCare facilitates a streamlined process for both caregivers and clients to engage with the platform.



Get unlimited messages & calls with elder care providers.

Your Basic Plan allows you to start conversations, but limits how far they can go. Upgrade to unlock unlimited messaging & secure calls!

Profile photo

Add a photo to increase your chance of finding a match.

29 Elder Care Providers Found

Theresa H

College Park, Maryland Active over a week ago

From 25.00/hour

Elder Care Provider in College Park, Maryland $25.00 an hour Compassionate & Professional Care

I am Theresa, and I am an experienced elder care provider in College Park, Maryland. I am available for full-time, live-out, and part-time roles and specialize in companion care, home care, administering medication, bathing, and personal care. I'm also certified in CPR. I am passionate about providing excellent care to each of my clients. My goal is to ensure their comfort and... Read more

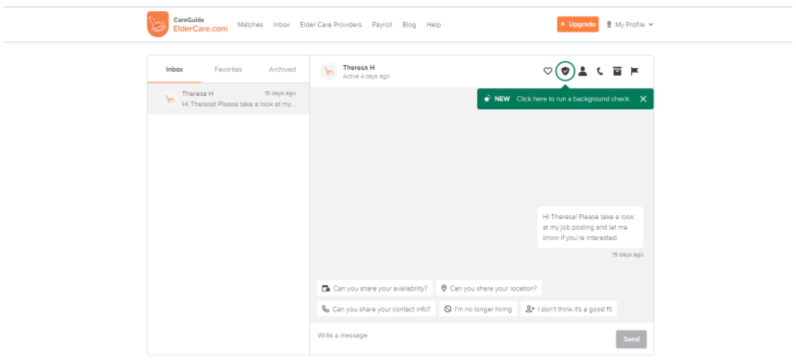
Esther O

College Park, Maryland Active over a week ago

From 23.00/hour

Figure 2.13: Matching Caregivers Page

ElderCare allows users to search for services by filtering criteria such as language, education level, availability, qualifications, and responsibilities. After applying these filters, the platform will show a list of caregivers that meet their needs. This feature significantly reduces the time and effort required for users to find suitable services. Therefore, users can quickly identify carers that align with their needs and preferences.



Theresa H 15 days ago
Hi Theresa! Please take a look at my...

Hi Theresa! Please take a look at my job posting and let me know if you're interested.
15 days ago

Figure 2.14: Live Chat Page

ElderCare has established a chat feature that facilitates direct communication between users and carers. This helps improve the quality of care and support provided. This secure messaging enables users to connect directly with carers to discuss care plans and ask questions.

ElderCare offers a more user-friendly booking mechanism. This booking mechanism allows customers to request services and pay caregivers directly through the platform. This guarantees that the reservation process runs smoothly and efficiently. ElderCare improves the transaction process by incorporating payment capabilities.

Despite its many strengths, ElderCare also has some specific weaknesses. One notable weakness is the costs associated with eldercare services. For some individuals and families, the cost of using these services can be too expensive. This limits their ability to fully utilize ElderCare resources.

ElderCare doesn't offer feedback or ratings for each caregiver. This means that users cannot review the ratings or feedback from other users about a specific carer's services. This makes it challenging for users to find quality and reliable services.

2.2.5 Comparison of Existing Elderly Care and Assistance Booking Platform

Table 2.1: Comparison of Existing Elderly Care and Assistance Booking Platform

Features	Care.com	Pillar	CareConcierge	ElderCare
Strengths				
Registration process for clients and caregivers	Yes	No	No	Yes
Comprehensive caregiver profiles	Yes	No	No	Yes
Secure messaging with caregivers	Yes	No	No	Yes
User-friendly interface	Yes	Yes	Yes	Yes
Transparent reviews and ratings	No	Yes	Yes	No
Caregivers Matching List	Yes	No	No	Yes
Convenient booking process	Yes	No	No	Yes
Live Chat Feature	Yes	No	No	Yes
Weaknesses				
Lack of direct booking process	No	Yes	Yes	No

Lack of caregiver details	No	Yes	Yes	No
Lack of user reviews	Yes	No	Yes	Yes

Based on the table comparing the features of various elderly care and assistance booking platforms that I reviewed, it's evident that the platforms developed in Malaysia share a common weakness.

First, these platforms do not allow users and caregivers to register directly on the platform, preventing a seamless connection between those seeking services and caregivers providing expertise. This restriction restricts users from easily finding the services they need and prevents caregivers from displaying their profiles and proposing their services within the platform. This led to the lack of a direct booking process on these sites contributes to the disconnect between users and caregivers. Users cannot directly schedule services they require, and caregivers cannot receive bookings directly through the platform, resulting in a fragmented and inefficient booking experience.

Moreover, the reviewed platforms lack communication channels that allow for direct communication between users and caregivers. Without secured messaging system, users and caregivers struggle to communicate effectively, which is critical for discussing care plans, asking questions, and providing real-time updates.

Additionally, the existing platforms in Malaysia lack search function with filter features. Users might need to search for their services by filtering some criteria. After applying the filter, this list should be provided on the elderly care and assistance booking platform so that users can easily access the services they need. By lacking this functionality, users might spend more time to find or compare various services until find a service that meets their requirements.

In conclusion, the common weakness of existing elderly care and assistance booking platform that developed in Malaysia are lacking platform that allow registration of both client and caregiver, lack of a communication channel and lack of a search function with filter capabilities. Enhancing features such as direct registration, robust communication channels, and search function with filter capabilities can significantly improve the user experience and the quality of caregiving services offered through these platforms.

2.3 Software development methodologies

2.3.1 Agile Development

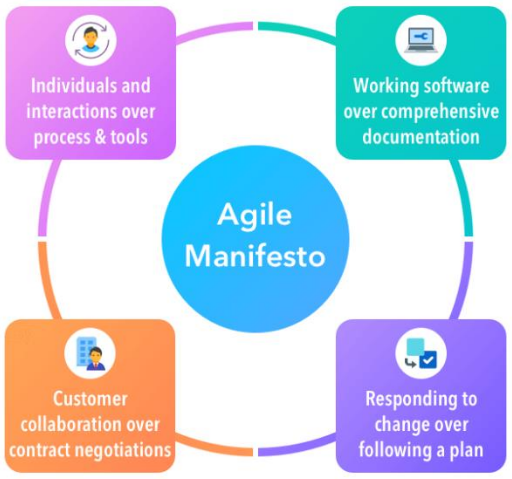

Individuals and interactions over process & tools
Working software over comprehensive documentation
Agile Manifesto
Customer collaboration over contract negotiations
Responding to change over following a plan

Figure 2.15: Agile Manifesto (Infinity, 2024).

Agile methodology is popular in the software development business due to its ability to adapt to changing needs and deliver value progressively. Its advantages lie from its adaptability and collaborative, which enables teams to react swiftly to feedback and market developments. Agile project management divides work into manageable segments, promoting ongoing improvement and ensuring that the final product meets customer requirements (Atlassian, 2024).

Small-to-medium sized firms benefit from agile because it speeds up decision-making, which helps teams adjust to change more successfully (Olic, 2020). Its emphasis on adaptability and flexibility also guarantees that the finished product satisfies client expectations by modifying methods in response to changing specifications and input.

However, Agile has several limitations. One major limitation is the lack of certainty in development schedules. This is because Agile is based on adaptability and iterative development so that project deadlines can be difficult to anticipate effectively. This uncertainty may cause dissatisfied among stakeholders who want a more consistent delivery schedule.

Another limitation of Agile is its emphasis on delivering small part of the software solution frequently. Although this strategy encourages gradual enhancement, it may lead to a restricted scope and complicate the delivery of a complete solution on schedule. This constraint might force teams to abandon some features and disappoint some stakeholders (Tutorialspoint, 2024).

Overall, Agile methodology provides benefits in flexibility, adaptability, and customer satisfaction. However, challenges arise from its iterative development and unpredictable deadline, impacting project management and scope.

2.3.2 Waterfall Development

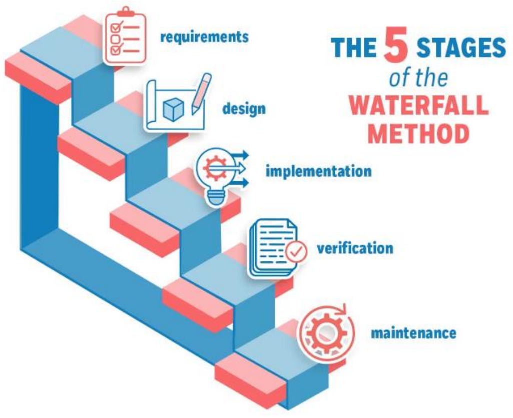


Figure 2.16: The five stages of Waterfall Development (Abraham, 2023).

The Waterfall methodology introduced by Winston W. Royce in 1970, is a step-by-step project management technique with five consecutive phases. To move forward, each step relies on the previous phase's deliverable. It is ideal for projects that have clearly defined end goals and require predictability (Hoory, 2022).

The strength of the Waterfall approach is its focus on systematic information exchange at each stage, providing complete documentation throughout the project cycle. This facilitates seamless phases and helps quickly onboard new team members when needed (Lucidchart, 2018).

Nevertheless, the Waterfall methodology has its weaknesses. One of it is the challenge of making changes after the project has entered the testing phase. This is because Waterfall is a linear process that requires each phase to be finished before proceeding to the next phase. After a phase is finished, backtracking and integrating changes becomes challenging and costly.

Additionally, the waterfall approach might not be suitable for complex or object-oriented projects due to its inflexible that could hinder adaptability. It is also not ideal for long-term projects or those with medium to high risks projects because of the changing requirements, as it lacks the capability to effectively manage such changes.

In conclusion, the Waterfall approach provides benefits in organized data exchange and predictability but is limited in its flexibility, adaptability, and responsiveness to change. Thus, this approach making it less appropriate for certain projects, particularly those with changing requirements or high complexity levels (Dutta, 2024).

2.3.3 Rapid Application Development (RAD)

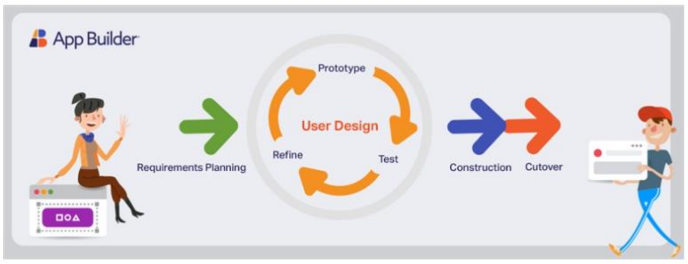


Figure 2.17: Flow of Rapid Application Development

Rapid Application Development is a flexible software development method known for its emphasis on prototyping, fast feedback cycles, and reduced focus on detailed planning. RAD promotes iterative refinement of the software through development and prototype building, enabling quick iterations and updates to meet user requirements (Kissflow, Inc, 2024).

One of the advantages of RAD is flexibility. RAD’s flexibility comes from its iterative nature. This allows developers to quickly adapt to changing needs and incorporate user feedback. Thus, resulting in a more customer-centric development process (Kissflow, Inc, 2024). The software is made to be both value-driven and functional through a continual feedback loop. Unlike sequential, waterfall-driven approaches, this methodology ensures quality from the beginning of product development.

However, RAD may be less suitable for broad or complex projects that lack clear boundaries. When faced with complex requirements or complex systems, RAD's collaborative and flexible approach can become unmanageable. In this case, waterfall or agile methodologies will be considered better options (Sharma, 2024).

In short, RAD is beneficial for small to medium-sized projects with clear scopes due to its flexibility and responsiveness to user feedback but might not be the best option for larger or complicated projects that require more structured methodologies.

2.3.4 Comparison of Software Development Methodology

Table 2.2: Comparison of Software Development Methodology

Aspect	Agile Development	Waterfall Development	Rapid Application Development (RAD)
Flexibility	Excellent	Poor	Good
Predictability	Poor	Excellent	Good
Collaboration	Good	Poor	Good
Responsiveness to Change	Excellent	Poor	Good
Suitability	Good	Excellent	Good

The table above outlines that Agile Development excels in flexibility and responsiveness to change, making it best suited for dynamic project but less predictable. Waterfall Development offers excellent predictability but lacks flexibility and responsiveness. This makes it well for stable, well-defined projects. RAD combines good flexibility, predictability, and collaboration, making it suitable for time-sensitive projects.

2.4 Web Development Tools

2.4.1 Development Frameworks

2.4.1.1 Laravel

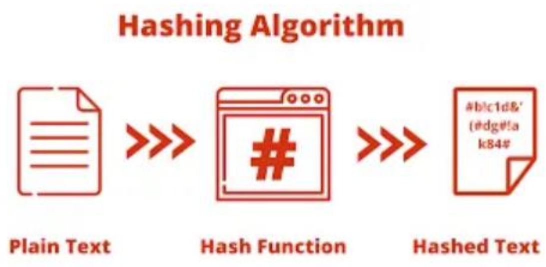


Figure 2.18: Hashing Algorithm (Blog, 2021).

Laravel is a PHP framework that offers a powerful collection of tools for web development and security. Laravel has strong security which will protect the website from DDoS attacks. Besides, the Bcrypt algorithm keeps passwords safe by encrypting them instead of directly saving them in the database.

Laravel also facilitates seamless database migration. This migration process allows developers to track changes made to the database over time. It also allows developers to roll back or update the database schema. Additionally, developers may create migration files and can easily manage the tables and columns without directly writing SQL queries. (ICStudio,2022).

However, Laravel still has some limitations. One notable disadvantage is its lack of direct support for payment processing. Although Laravel provides robust tools for web development, it lacks built-in functionality for managing payment transactions. Therefore, developers need to utilize external libraries and connect with widely used payment gateways like Stripe, PayPal, and Braintree. (Sharma, 2024).

2.4.1.2 React

React is a UI development library for JavaScript that manages CSS file prefixes and uses Webpack to assemble React, JSX, and ES6 code automatically. Since its release in May 2013, it has grown to rank among the frontend libraries for web development that are most frequently used (Deshpande, 2023).

React using a component-based architecture by breaking down the user interface into reusable components. Reusing components saves developers time and ensure that changes made to one section of the application don't impact other sections. Furthermore, compared to other frontend frameworks, React's modular nature makes code maintenance simpler and more adaptable, which helps organizations save a lot of time and money (Modan, 2024).

However, React is not a full-featured framework as it focuses on the view component of the MVC architecture. Developers require additional libraries and tools for the Controller and Model components, which may result in a less structured codebase and patterns.

Another issue is the lack of documentation caused by Reacts rapid expansion, which includes the addition of new tools and patterns on a regular basis. This can make it difficult for new developers to begin working with React and may result in delayed development, particularly in teams with inexperienced developers (Ragala, 2023).

2.4.1.3 Vue.js

Vue.js is a JavaScript framework for building user interfaces, leveraging standard HTML, CSS, and JavaScript. It offers a declarative, component-based programming model for efficient development of user interfaces (Vue.js, 2024).

The strength of Vue.js is its detailed documentation, which is crucial for both beginners and experienced developers. It provides clear explanations, comparisons to other frameworks, and is regularly updated, making it a reliable resource for troubleshooting and learning.

Another advantage of Vue.js is the reusability achieved through its component-based approach, allowing the creation of reusable single-file components for cohesive, maintainable UI elements, thereby increasing development efficiency (Patel and Patel, 2024).

However, compared to mature frameworks like React, Vue.js faces challenges such as a lack of plugins, which may hinder full dependency and require switching to other languages to implement certain features.

Additionally, Vue.js may not offer robust support for large-scale projects due to its smaller community and development team size, which could impact stability and quick issue resolution required for enterprise-level projects (Editor, 2022).

2.4.1.4 Comparison of Development Frameworks

Table 2.3: Comparison of Development Frameworks

Aspect	Laravel	React	Vue.js
Backend Framework	Yes	No	No
Frontend Library	No	Yes	No
Built-in Security	Yes	No	Yes
Built-in Authentication	Yes	No	No
Seamless Database Migration	Yes	No	No
Direct payment processing	No	No	No
High Reusability	No	Yes	Yes
Detailed documentation	Yes	No	Yes
Strong community support	Yes	Yes	Yes
Scalability	Yes	Yes	Yes

From the table above, it can be said that Laravel stands out as a backend framework with built-in security, authentication, seamless database management, and detailed documentation. React is a frontend library known for high reusability of components, strong community support, and scalability in building interactive user interfaces. Vue.js shares similarities with React in terms of high reusability and strong community support, and offering scalability for developing large-scale applications, but it lacks built-in authentication and database migration features.

2.4.2 Code Editors

2.4.2.1 VS Code

Visual Studio Code is Microsoft's adaptive integrated programming environment (IDE). It is known for its speed, usability, and comprehensive feature set. This cross-platform code editor is widely appreciated for its extensive feature set and user-friendly interface (WebHostingMonkey, 2024).

A significant advantage of Visual Studio Code is its extensive debugging assistance. Developers can improve debugging and accelerate application error resolution using tools such as analyzing variables, stepping through code, managing exceptions, and creating breakpoints (Pedamkar, 2023).

Visual Studio Code is the best solution for developers working with various programming languages such as Python, C, Java, and JavaScript. This is because it provides important language support such as syntax highlighting, code completion, and language-specific features. This broad compatibility improves development processes and increases productivity on multilingual projects (Mir, 2023).

Although VS Code has many benefits, new users may encounter difficulties. The huge feature set and adaptable options may be too much for less experienced programmers who may not need all of its features. For those with no coding experience, it can be difficult to navigate and use the editor effectively without spending some time and effort (Mir, 2023).

2.4.2.2 NotePad++

For Microsoft Windows, Notepad++ is a free text and source code editor. With its tabbed editing feature, users may work on many open files from a single window (UniversityOfKent, 2024).

One feature of Notepad++ is its ability for multiple tabs, which makes it easier to work on many files at once. Users may simply switch between tabs, and each tab preserves its own settings such as font size, color scheme, and line numbers, which help with organization and navigation across various sections of code.

Another significant feature is Notepad++'s syntax highlighting. It enhances coding by using different colours to highlight distinct parts of the code according to their functions. This facilitates reading and comprehending complicated code files, which boosts productivity and code comprehension (Centro and Centro, 2023).

However, Notepad++ has certain limitations. One limitation is that it relies on plugins to increase functionality, which might cause dependence problems or compatibility issues with newer versions of Notepad++ or the plugins themselves (Schaferhoff, 2022).

Furthermore, Notepad++ is mainly a text editor and lacks some complex Integrated Development Environment (IDE) features available in more complete IDEs, such as code debugging, project management, and built-in compilers. This constraint may affect developers who need a more integrated development environment for complex projects (Smithaydon, 2023).

2.4.2.3 Phpstrom

PhpStorm is an integrated development environment for PHP developers and designed with the purpose of boosting development productivity. With the help of this software, developers on Linux, Windows, and macOS may easily write, edit, analyse, restructure, test, and debug PHP code (Phpstorm, 2021).

One of the PhpStrom strength is its strong support of different PHP frameworks, making it a great option for developers who working with frameworks like Laravel, Drupal, WordPress, CakePHP, Symfony, and others. Its integrated features simplify development processes and improve productivity with widely used frameworks (Monovm, 2021).

Additionally, PhpStorm stands out for its strong database support, offering various tools for smooth integration with SQL and diverse databases. Users can manage the database directly within the IDE. This extensive support contributes to improved code assistance and simplifies database management for projects (Pedamkar, 2023).

However, since PhpStorm is a paid integrated development environment that costs 200 euros per year, some developers with limited funds will not hesitate to use it. Moreover, PhpStorm needs a substantial amount of random-access memory (RAM) to function optimally, with a minimum of 16GB recommended. This demand for resources can cause difficulties for developers working with constrained hardware or environments, impacting performance and availability (Phpstorm, 2021).

2.3.3.4 Comparison of Code Editors

Table 2.4: Comparison of Development Frameworks

Aspect	Vs Code	NotePad++	PhpStorm
Integrated Development Environment (IDE)	Yes	No	Yes
Cross-platform	Yes	No	Yes
Free to use	Yes	Yes	No
Extensive debugging support	Yes	No	Yes
Wide range of programming languages	Yes	No	Yes
Advanced PHP support	Yes	No	Yes
Minimal resource usage	No	No	No

VS Code and PhpStorm are IDEs suitable for various programming languages, offering extensive debugging support, and provides advanced PHP-specific features. Vs Code is free to use while PhpStorm requiring a paid license. In contrast, NotePad++ is a lightweight text editor that lack of IDE functionalities and PHP support, primarily suited for basic text editing tasks.

2.4.3 Databases

2.4.3.1 Oracle

Oracle is a relational database management system that developed by Oracle Corparation. It is known for its database engine which excel in data organization, storage and retrieval (javaTpoint, 2024).

One of Oracle's strengths is its emphasis on performance, which includes approaches for achieving high performance. Performance tuning strategies can be used within the database to retrieve and modify data more quickly, reducing query execution time and optimizing application operations.

Another advantage is Oracle's robust data security measures. It makes use of IP blocklists, multiple authentication methods, and encryption as data security mechanisms. Additionally, it complies with globally recognised security standards like PCI DSS, which guarantees the security of company data (Nguyen and Nguyen, 2024).

However, there are several limitations to using an Oracle database. One of it is its complexity, which can be difficult for users who are not technically adept or lack the necessary technical skills to operate with Oracle. This complexity may make adoption difficult, particularly among users who prefer simpler database management systems (javaTpoint, 2024).

2.4.3.2 MySQL

MySQL is available for free and enables users to manage structured data. It is widely implemented in a variety of applications, including medium-scale projects, enterprise-level solutions, and large-scale websites (Domantas G., 2024).

One advantage of MySQL is that it is open-source and free, making it a popular choice for entrepreneurs and developers on a tight budget. It provides almost all the functionality required by a database server without affecting application performance or consistency (blueclaw, 2021).

MySQL well know about its speed, scalability, and flexibility. It is considered one of the fastest databases to access due to its ability to enable multi-threading to improve performance. Additionally, MySQL is suitable for a variety of use cases as it can handle embedded applications (blueclaw, 2021).

However, MySQL has disadvantages. Certain use cases, such as auditing, or transactions may cause stability difficulties and data corruption. Furthermore, MySQL's speed may slow down under heavy stress, making it less suitable for large organizations with millions of records and transactions compare to other databases such as Oracle or SQL Server (blueclaw, 2021).

2.4.3.3 SQLite

SQLite is a free-to-use database management system that runs without a server. It requires zero configuration and does not need installation, making it highly convenient with its compact size of less than 500kb, significantly smaller than other database management systems (S, 2023).

One advantage of SQLite is its better performance compared to traditional file systems, offering fast reading, and writing operations. It achieves this by only loading the necessary data and overwriting specific parts of the file when edits are made, leading to efficient use of memory (javaTpoint, 2024).

SQLite is also portable across all 32-bit and 64-bit operating systems and various architectures, allowing multiple processes to attach to the same application file without interference. It is compatible with all programming languages and may be integrated with them without any problems (javaTpoint, 2024).

However, SQLite has limitations, such as lacking support for features like stored procedures and user-defined functions found in traditional RDBMS. It also lacks user management features, making it unsuitable for applications requiring user authentication and authorization (Wong, 2023).

Another limitation of SQLite is its file-based nature, which limits centralized control of the database and can be challenging to manage in large-scale applications. Additional tools or systems may be necessary for effective database management (Wong, 2023).

2.4.3.4 Comparison of databases

Table 2.5: Comparison of databases

Aspect	Oracle Database	MySQL	SQLite
Performance tuning	Yes	Yes	Yes
Robust data security	Yes	Yes	Limited
Complexity	Yes	Moderate	No
Open-source and free	No	Yes	Yes
Speed and scalability	Yes	Yes	Limited
Stability	Yes	Moderate	Moderate
Suitable for large-scale applications	Yes	Yes	Limited
Support for stored procedures and user-defined functions	Yes	Yes	Limited
Centralized control over the database	Yes	Yes	No
Suitable for user authentication and authorization	Yes	Yes	Limited

By comparing the table above, it is note that Oracle Database and MySQL excel in performance tuning, robust data security, scalability for large-scale applications, and support for stored procedures and user-defined functions. However, Oracle Database is complex and not open-source, while MySQL offers open-source availability. SQLite is simpler, open-source, and free, suitable for smaller applications with limited security features, scalability, and centralized control over the database compared to Oracle Database and MySQL.

2.5 Conclusion

In conclusion, after reviewing existing elderly care and assistance booking platforms in Malaysia, it's evident that these platforms lack a centralized platform for users and caregivers to register and interact, efficient booking systems within the platform, communication channels between them, and a search function with filter capabilities.

By comparing the software development methodology, it is noted that RAD is more suitable for this project compared to other methodology. This is because RAD focus on quick prototyping, iterative development, allow for rapid iterations, and frequent updates based on user input.

For the development framework, Laravel stands out of its full-stack framework. Laravel's backend capabilities ensure efficient development, security, and scalability. Laravel’s Blade templating engine allow developers to create dynamic fronted views using PHP. By using Blade templates, developers can reuse the components, extend layouts, and pass data from the backend to the frontend seamlessly.

For the code editor, VS Code as the IDE ensures coding, debugging, and collaboration across frontend and backend components. Lastly, MySQL provides scalability, robust data security, and complex query support crucial for managing user and caregiver data.

CHAPTER 3
METHODOLOGY AND WORK PLAN

3.1 Introduction

The objective of this chapter is to provide specifics on the project's methodology and workplan. There are four sections in this chapter. Based on the study presented in Chapter 2, Section 3.2 describes the selected software development methodology. The development and prototype tools used in system development are covered in Section 3.3. The Gantt Chart and WBS are presented in Section 3.4. Finally, this chapter is concluded in Section 3.5.

3.2 Software Development Methodology

Rapid Application Development (RAD) was chosen for the elderly care and assistance booking platform project due to its iterative approach. The RAD methodology involves four main phases: Planning and Analysis: where requirements are gathered and scoped; Design: focusing on system architecture and interface layout; Development and Testing: featuring rapid prototyping and iterative development with continuous feedback; and Closing: encompassing finalizing the system, user acceptance testing, bug fixing, and deployment preparation.

3.2.1 Planning and Analysis

Determining the project's direction and comprehending the needs of stakeholders are essential tasks for the Planning and Analysis phase.

3.2.1.1 Identify Stakeholders

Identifying the proper stakeholders is essential to making sure that the project's development considers all relevant parties. The stakeholder in this project is elderly individuals in need of care, family members in charge of scheduling and administering care services, and platform administrators supervising system operations.

3.2.1.2 Gather Requirements

A thorough literature analysis was carried out to examine existing elderly care and assistance booking platform offerings to efficiently gather requirements. Through this review, I gained insight on the feasibility and functionality analysis, user input, and industry best practices. Knowledge on user expectations and system capabilities was obtained through this review. Thus, this eliminated the need for extra surveys or questionnaires and added significant data and insights to the requirement-gathering phase.

3.2.1.3 Define Project Scope

In this phase, the project scope is identified. As stated in Chapter 1, the project's scope is broad and includes a variety of modules and features necessary for a productive and intuitive eldercare platform. These consist of systems enabling both users and carers to register and manage their profiles securely, search and booking capabilities for services and carers, a communication channel that allows users to communicate directly with carers, and more features to improve the overall user experience.

3.2.2. Design

In the Design phase, the Use Case Diagram, Class Diagram, Activity Diagram, and Entity Relationship Diagram (ERD) are developed to define user interactions and database structure for the elderly care platform.

3.2.2.1 Use Case Diagram Creation

A use case diagram will be drawn in this design stage. The purpose of this diagram is to visually represent the interaction between users and the system. In this project, this diagram will illustrate three users in the platform which are the client, caregiver, and admin, and how each interacts with the system.

3.2.2.2 Construction of the Class Diagram

The class diagram will be drawn based on all the models of the platform to illustrate the key models of the Elder Care and Assistance Booking Platform and their relationships. This class diagram provides a comprehensive view of the system structure.

3.2.2.2 Construction of the Activity Diagram

Activity diagrams will be drawn based on all the use cases in the Use Case Diagram. This diagram provides a clear and detailed visualization of the specific flow of activities and actions involved in the use case.

3.2.2.2 Construction of the Entity Relationship Diagram (ERD)

The database structure of the platforms will be specified using an entity relationship diagram (ERD). The Entity Relationship Diagram (ERD) will demonstrate the relationships between various entities, including users, notifications, ch_favorites, ch_meesages, services, service_dates, service_timeslots, and bookings.

3.2.3 Development and Testing

The Development and Testing phase will use an iterative methodology to ensure speedy development and continual testing for project success. This phase will include several iterations, each focused on improving the platform's capabilities to fulfill the following goals:

3.2.3.1 First Iteration

The first iteration of this project will focus on developing the Client Module. All the features of the Client Module will be developed. After developing all the features of the Client Module, unit tests, and feature tests will be carried out to ensure that all functions are working well.

3.2.3.2 Second Iteration

The second iteration of this project will focus on developing the Caregiver Module. All the features of the Caregiver Module will be developed. After the development of all the features of the Caregiver Module, unit tests, and feature tests will be carried out to ensure that all functions are working well.

3.2.3.3 Third Iteration

The third iteration of this project will focus on developing the Admin Module. All the features of the Admin Module will be developed. After the development of all the features of the Admin Module, unit tests, and feature tests will be carried out to ensure that all functions are working well. Lastly, the User Acceptance Test will be carried out to ensure user satisfaction on the platform.

3.2.4 Closing

In the closing state, all development and testing codes will be finalized. If any bugs are found, they will be resolved. Once the platform has no further issues, the report documentation will begin.

3.3 Development Tools

3.3.1 Markup Languages

3.3.1.1 HyperText Markup Language (HTML)

HTML is required to develop web pages and defining its content and layout structure. It provides the structure needed to display forms, graphics, text, and other types of components online.

3.3.1.2 Cascading Style Sheets (CSS)

CSS enables developers to format and style HTML elements while also changing the visual appearance of the platform. It includes colours, fonts, layout, and responsive design features to provide a visually appealing and user-friendly experience.

3.3.2 Programming Languages

Using AJAX, JavaScript can interact with PHP to allow dynamic content changes without the need for page reloads. This connection allows for the creation of a search function with filter options.

3.3.2.1 JavaScript

JavaScript can be used to create dynamic and interactive web elements. It is essential when integrating third-party libraries and APIs, controlling user interactions, form validations, animations, and client-side scripting.

3.3.2.2 PHP

The server-side programming language PHP is used to create the backend of web-based applications. It handles form data processing, database interactions, user authentication control, and dynamic content generation based on user input or system logic.

3.3.3 Framework

3.3.3.1 Laravel

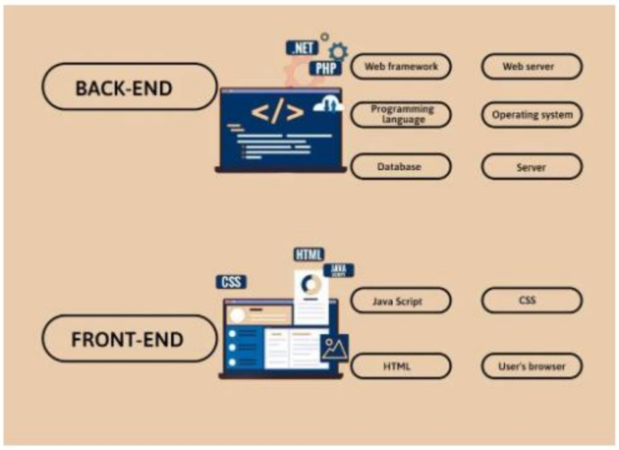


Figure 3.1: Laravel as a full-stack framework (Tis, 2023)

In this project, Laravel will be used for both frontend and backend development. Laravel combines HTML and Blade templates for creating the user interface while JavaScript adds interactivity and CSS style to the pages. This setup allows developers to develop the platform smoothly and effectively without needing extra fronted frameworks.

3.3.4 Runtime Environments

3.3.4.1 Node.js

In this project, Node.js will be used as a runtime environment that allows JavaScript to be executed server-side. It will be utilized in the service recommendation system by enabling efficient handling of server-side logic and asynchronous events to provide fast and scalable recommendations to users.

3.3.5 Integrated Development Environment (IDE)

3.3.5.1 VS Code

Visual Studio Code is the chosen integrated development environment for this project. The developer can open the project and edit its code by using VS Code. Its integrated terminal helps developers to execute any command more efficiently.

3.3.6 Database System

3.3.6.1 MySQL

MySQL was selected as the database management system for this platform. This is because MYSQL is quite compatible with the Laravel framework. Laravel includes support for Object-Relational Mapping (ORM) via Eloquent ORM. This will ease database interactions and querying using MYSQL. Besides, its features like data indexing, querying, transactions, and user management make it an excellent option for organizing and storing data.

3.3.7 Hosting Server

3.3.7.1 WampServer

WampServer will serve as the hosting server for this platform. WampServer meets the system requirements for running Laravel, such as PHP, MYSQL and Apache. This setup ensures that Laravel application runs smoothly without compatibility issues and saving time and effort in setting up a development environment.

3.3.8 Prototyping Tool

3.3.8.1 Axure RP 9

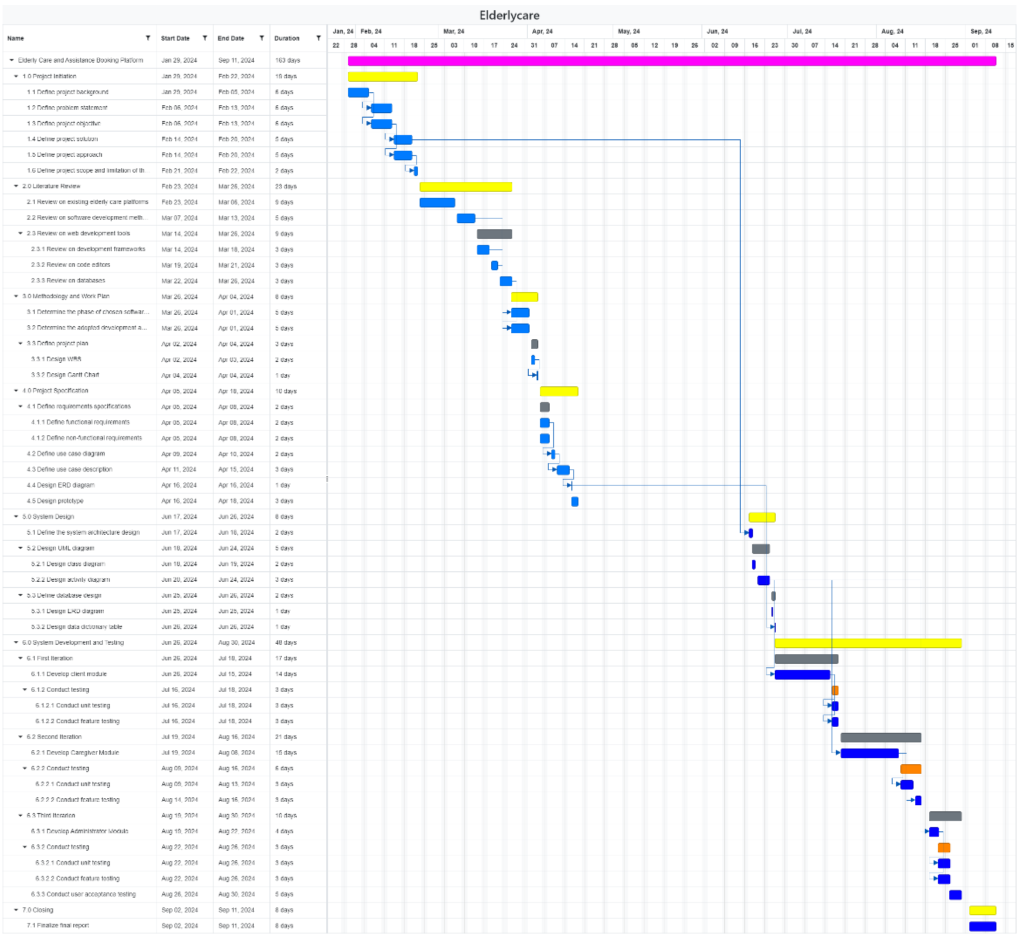
Axure RP 9 is a UI tool for creating functional prototypes. Axure RP 9 was selected for developing and testing the user interface and functionality of the platform. The prototype will be design and develop using Axure RP.

3.4 Project Plan

3.4.1 Work Breakdown Structure

WBS is attached in Appendix A.

3.4.2 Gantt Chart



Name	Start Date	End Date	Duration
Elderly Care and Assistance Booking Platform	Jan 29, 2024	Sep 11, 2024	163 days
1.0 Project Initiation	Jan 29, 2024	Feb 22, 2024	19 days
1.1 Define project background	Jan 29, 2024	Feb 05, 2024	6 days
1.2 Define problem statement	Feb 06, 2024	Feb 13, 2024	6 days
1.3 Define project objective	Feb 06, 2024	Feb 13, 2024	6 days
1.4 Define project solution	Feb 14, 2024	Feb 20, 2024	5 days
1.5 Define project approach	Feb 14, 2024	Feb 20, 2024	5 days
1.6 Define project scope and limitation of th...	Feb 21, 2024	Feb 22, 2024	2 days
2.0 Literature Review	Feb 23, 2024	Mar 26, 2024	23 days
2.1 Review on existing elderly care platforms	Feb 23, 2024	Mar 06, 2024	9 days
2.2 Review on software development meth...	Mar 07, 2024	Mar 13, 2024	5 days
2.3 Review on web development tools	Mar 14, 2024	Mar 26, 2024	9 days
2.3.1 Review on development frameworks	Mar 14, 2024	Mar 18, 2024	3 days
2.3.2 Review on code editors	Mar 19, 2024	Mar 21, 2024	3 days
2.3.3 Review on databases	Mar 22, 2024	Mar 26, 2024	3 days
3.0 Methodology and Work Plan	Mar 26, 2024	Apr 04, 2024	8 days
3.1 Determine the phase of chosen softwar...	Mar 26, 2024	Apr 01, 2024	5 days
3.2 Determine the adopted development a...	Mar 26, 2024	Apr 01, 2024	5 days
3.3 Define project plan	Apr 02, 2024	Apr 04, 2024	3 days
3.3.1 Design WBS	Apr 02, 2024	Apr 03, 2024	2 days
3.3.2 Design Gantt Chart	Apr 04, 2024	Apr 04, 2024	1 day
4.0 Project Specification	Apr 05, 2024	Apr 18, 2024	10 days
4.1 Define requirements specifications	Apr 05, 2024	Apr 08, 2024	2 days
4.1.1 Define functional requirements	Apr 05, 2024	Apr 08, 2024	2 days
4.1.2 Define non-functional requirements	Apr 05, 2024	Apr 08, 2024	2 days
4.2 Define use case diagram	Apr 09, 2024	Apr 10, 2024	2 days
4.3 Define use case description	Apr 11, 2024	Apr 15, 2024	3 days
4.4 Design ERD diagram	Apr 16, 2024	Apr 16, 2024	1 day
4.5 Design prototype	Apr 16, 2024	Apr 18, 2024	3 days
5.0 System Design	Jun 17, 2024	Jun 26, 2024	8 days
5.1 Define the system architecture design	Jun 17, 2024	Jun 18, 2024	2 days
5.2 Design UML diagram	Jun 18, 2024	Jun 24, 2024	5 days
5.2.1 Design class diagram	Jun 18, 2024	Jun 19, 2024	2 days
5.2.2 Design activity diagram	Jun 20, 2024	Jun 24, 2024	3 days
5.3 Define database design	Jun 25, 2024	Jun 26, 2024	2 days
5.3.1 Design ERD diagram	Jun 25, 2024	Jun 25, 2024	1 day
5.3.2 Design data dictionary table	Jun 26, 2024	Jun 26, 2024	1 day
6.0 System Development and Testing	Jun 26, 2024	Aug 30, 2024	48 days
6.1 First Iteration	Jun 26, 2024	Jul 18, 2024	17 days
6.1.1 Develop client module	Jun 26, 2024	Jul 15, 2024	14 days
6.1.2 Conduct testing	Jul 16, 2024	Jul 18, 2024	3 days
6.1.2.1 Conduct unit testing	Jul 16, 2024	Jul 18, 2024	3 days
6.1.2.2 Conduct feature testing	Jul 16, 2024	Jul 18, 2024	3 days
6.2 Second Iteration	Jul 19, 2024	Aug 16, 2024	21 days
6.2.1 Develop Caregiver Module	Jul 19, 2024	Aug 08, 2024	15 days
6.2.2 Conduct testing	Aug 09, 2024	Aug 16, 2024	6 days
6.2.2.1 Conduct unit testing	Aug 09, 2024	Aug 13, 2024	3 days
6.2.2.2 Conduct feature testing	Aug 14, 2024	Aug 16, 2024	3 days
6.3 Third Iteration	Aug 19, 2024	Aug 30, 2024	10 days
6.3.1 Develop Administrator Module	Aug 19, 2024	Aug 22, 2024	4 days
6.3.2 Conduct testing	Aug 22, 2024	Aug 26, 2024	3 days
6.3.2.1 Conduct unit testing	Aug 22, 2024	Aug 26, 2024	3 days
6.3.2.2 Conduct feature testing	Aug 22, 2024	Aug 26, 2024	3 days
6.3.3 Conduct user acceptance testing	Aug 26, 2024	Aug 30, 2024	5 days
7.0 Closing	Sep 02, 2024	Sep 11, 2024	8 days
7.1 Finalize final report	Sep 02, 2024	Sep 11, 2024	8 days

Figure 3.2: Gantt Chart

3.5 Summary

In Summary, this chapter highlights how crucial it is to choose a suitable software development methodology based on research findings, to improve system development by using efficient development and prototyping tools, and to manage projects effectively by creating a thorough project plan that includes a Gantt chart and Work Breakdown Structure (WBS). This chapter provides a clear road plan for the project's successful completion, laying the foundation for the following chapter.

CHAPTER 4
PROJECT SPECIFICATION

4.1 Introduction

This chapter will focus on the details of specification in this project. This chapter is divided into six sections. Section 4.2 contains a list of requirements specifications which encompass functional requirements for the client, caregiver, and administrator, as well as non-functional requirements. Section 4.3 is to show the use case diagram of this project. Section 4.4 is to list out all use case descriptions based on the use case diagram. Section 4.5 is to provide preliminary user interface design for the following iterations. Section 4.6 is to summarize this chapter.

4.2 Requirements Specifications

This section discusses on functional requirements for client, caregiver, and administrator, and non-functional requirements of this project.

4.2.1 Functional Requirements

4.2.1.1 Functional Requirements for Clients

Table 4.1: Functional Requirements for Clients

Requirement ID	**Functional Requirement**
FR001	The platform should allow clients to register an account using a valid email address and password, with validation rules for email format and password strength.
nFR002	The platform should allow clients to login using their registered account.
FR003	The platform should allow clients to logout.
FR004	The platform should allow clients to manage their profile including updating personal information.

FR005	The platform should allow clients to search for care service by applying the filter function and view the caregiver matching list based on search criteria.
FR006	The platform should allow clients to get a service recommendation by describing the service they need.
FR007	The platform should allow clients to view detailed information about caregiver profiles.
FR008	The platform should allow clients to create a booking for a care service.
FR009	The platform should allow clients to view for their bookings list.
FR010	The platform should allow clients to update their booking.
FR011	The platform should allow clients to cancel their booking with confirmation prompts.
FR012	The platform should allow clients to make a payment by selecting a payment method after the booking has been accepted.
FR013	The platform should allow clients to view the transaction history, generate invoice, or export CSV for any transactions.
FR014	The platform should allow clients to communicate with their caregiver and the administrator through messaging features.
FR015	The platform should allow clients to provide feedback after conducting a service.

4.2.1.2 Functional Requirements for Caregivers

Table 4.2: Functional Requirements for Caregivers

Requirement ID	**Functional Requirement**
FR001	The platform should allow caregivers to register an account using a valid email address and password, with validation rules for email format and password strength.
nFR002	The platform should allow caregivers to login using their registered account.
FR003	The platform should allow caregivers to logout.
FR004	The platform should allow caregivers to update their profile including updating personal information.
FR005	The platform should allow caregivers to create care services.
FR006	The platform should allow caregivers to view their own care services.
FR007	The platform should allow caregivers to update their care services.
FR008	The platform should allow caregivers to delete their care services.
FR009	The platform should allow caregivers to view of their bookings with user information.
FR010	The platform should allow caregivers to approve or decline booking requests.
FR011	The platform should allow caregivers to view their feedback from clients.
FR012	The platform should allow caregivers to view the payment received history.

FR013	The platform should allow caregivers to view all the appointments in the appointment calendar.
FR014	The platform should allow caregivers to communicate with users and administrators through messaging features.

4.2.1.3 Functional Requirements for Administrators

Table 4.3: Functional Requirements for Administrators

Requirement ID	**Functional Requirement**
FR001	The platform should allow admin to view the details of user account including profile information.
FR002	The platform should allow admin to delete the user account with confirmation prompts.
FR003	The platform should allow admin to delete the service created by caregiver with confirmation prompts.
FR004	The platform should allow admin to communicate with clients and caregivers.

4.2.2 Non-functional Requirements

Table 4.4: Non-functional Requirements

Requirement ID	**Non-functional Requirement**
NFR001	The platform should be user-friendly with clear navigation and intuitive interfaces.
NFR002	The platform should be reliable by ensuring that bookings and communication are processed accurately and on time.

NFR003	The platform should implement robust security measures including encryption for data protection to protect user data.
NFR004	The platform should support major browsers such as Chrome, Firefox, and Edge.
NFR005	The platform should have efficient error handling which provides error messages and informative messages for troubleshooting.

4.3 Use Case Diagram

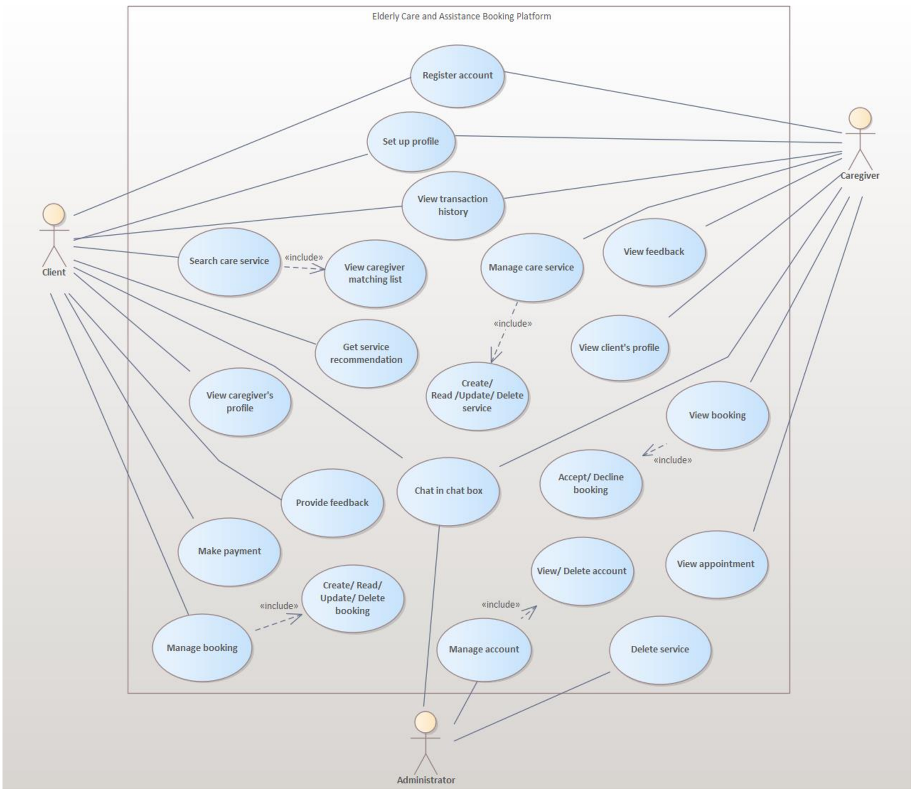


Figure 4.1: Use Case Diagram

4.4 Use Case Description

Table 4.5: Use Case Description of Register Account

Use Case Name: Register Account	ID: UC01	Importance Level: High
Primary Actor: Client, Caregiver	Use Case Type: Detail, Essential	

Stakeholders and Interests: Client: Client wants to register an account to search for care service. Caregiver: Caregiver wants to register account to provide care services.
Brief Description: This use case describes how the client and caregiver register a new account on the platform.
Trigger: Client wants to seek for care service and Caregiver wants to provide care service.
Relationships: Association : Client, Caregiver Include : - Extend : - Generalization: -
Normal Flow of Events: 1. Client and caregiver access the registration page of the platform. 2. The client and caregiver enter their name, email address, password, and confirmed password and choose their role to create an account. 3. The system validates the entered email address is valid. Continue to E3: Reenter Email Address. 4. The system validates the entered password is compliance with password policies. Continue to E4: Reenter Password. 5. The system verifies that the confirmed password matches the entered password. Continue to E5: Reenter Confirmed Password. 6. If both email address and password are valid, the system will send an email verification link to the email registered. 7. After the email is verified, the system proceeds to create a new user account. 8. The client and caregiver will be redirected to the Login Page.
Sub-flows:

Alternate/Exceptional Flows: E3: Reenter Email Address 1. If the client and caregiver enter an email address that is already registered in the system, the system will display an error message indicating that the email address is already exist. 2. The client and caregiver are prompted to use another email address for registration. E4: Reenter Password 1. If the client and caregiver enter a password that is not compliance with password policies, the system will display an error message indicating that the password does not meet the requirements. 2. The client and caregiver are prompted to use another password for registration. E5: Reenter Confirmed Password 1. If the client and caregiver enter the confirmed password that is not matches the entered password, the system will display an error message indicating that the password does not match the entered password. 2. The client and caregiver are prompted to reenter the confirmed password.

Table 4.6: Use Case Description of Search Care Services

Use Case Name: Search Care Services	ID: UC02	Importance Level: High
Primary Actor: Client	Use Case Type: Detail, Essential	
Stakeholders and Interests: Client: Client wants to search for care services that meet their requirements		
Brief Description: This use case describes how the client search for available care services based on their specific needs and		

preferences. The system will display a list of caregivers that match their search criteria for care services.
Trigger: Client wants to find suitable care services.
Relationships: Association : Client Include : View Caregiver Matching List Extend : - Generalization: -
Normal Flow of Events: 1. The client search for care services by applying filter to filter the service type, duration of service, price, location, and provider. 2. The system processes the search query and retrieves matching care services from the database. Continue to E2: No Matching Care Service. 3. The system presents a list of caregivers that match the user's search criteria along with their services and details.
Sub-flows:
Alternate/Exceptional Flows: E2: No Matching Care Service 1. If no matching care services are found, the system will display an empty list. 2. The client can clear the filter to reset to the default state, showing all available services

Table 4.7: Use Case Description of Get Service Recommendation

Use Case Name: Get Service Recommendation	ID: UC03	Importance Level: High
Primary Actor: Client	Use Case Type: Detail, Essential	

Stakeholders and Interests: Client: The client wants to get a service recommendation from the system.
Brief Description: This use case describes how the client gets a service recommendation from the system. The client will use a chat feature to ask for a service by describing the service they need.
Trigger: The client wants to get a recommended service.
Relationships: Association : Client Include : Extend : - Generalization: -
Normal Flow of Events: 1. The client navigates to the Search Service Page. 2. The client clicks on the “Click here for suggestion” button. 3. The client types the description of the service they want in the chat box and sends it. 4. If the service is found, a recommended service with the caregiver's details will be returned to the client. 5. If not, the chat box will return a message indicating that no service found.
Sub-flows:
Alternate/Exceptional Flows:

Table 4.8: Use Case Description of View Caregiver's Profile

Use Case Name: View Caregiver’s Profile	ID: UC04	Importance Level: High
Primary Actor: Client	Use Case Type: Detail, Essential	

Stakeholders and Interests: Client: They want to view caregiver’s detail information.	
Brief Description: This use case describes how the client view detailed information about a caregiver’s profile.	
Trigger: Client wants to gather more information about caregiver through their profile.	
Relationships: Association : Client Include : - Extend : - Generalization: -	
Normal Flow of Events: 1. The client navigates to the caregiver’s profile page. 2. The system displays the detailed profile of the selected caregiver that include caregiver’s name, photo, email, phone number, qualifications, experience etc. 3. The client reviews the details information provided in the profile to assess suitability.	
Sub-flows:	
Alternate/Exceptional Flows:	

Table 4.9: Use Case Description of Manage Booking

Use Case Name: Manage Booking	ID: UC05	Importance Level: High
Primary Actor: Client	Use Case Type: Detail, Essential	
Stakeholders and Interests:		

Client: Client wants to manage their own booking.
Brief Description: This use case describes how the client managing booking, including creating, viewing, updating, cancel booking.
Trigger: Client intends to manage booking on the platform.
Relationships: Association : Client Include : Create/ Read/ Update/ Delete Booking Extend : - Generalization: -
Normal Flow of Events: 1 If the client wants to create the booking, S1, Create Booking is performed. 2 If the client wants to view the booking, S2: View Booking is performed. 3 If the client wants to edit the booking, S3: Edit Booking is performed. 4 If t the client wants to cancel the booking, S4: Cancel Booking is performed.
Sub-flows: S1: Create Booking 1. The client navigates to Caregiver List Page and select a service to view its details. These details include the caregiver's name, service type, service description and feedback etc. from other clients. 2. In the Service Detail Page, the client can make the booking by choosing the date and time they wish to have the service. 3. The system saves the details into the database and confirms the successful booking with a success message. S2: View Booking

1. The client accesses their Booking List Page to view a list of their existing bookings. 2. The client selects a specific booking to view detailed information. 3. The system displays details about the booking, including caregiver name, service type, service details, booking date and time etc. S3: Edit Booking 1. The client accesses their Booking List Page and selects a booking to do modification. 2. The system redirects the client to an Update Booking Page to allow client to modify the existing booking. 3. After making changes, client can click on the Update button, and the system updates the booking information accordingly. 4. The system shows the success message to indicating that the booking updated successfully. S4: Cancel Booking 1. The client accesses their Booking List Page to select a booking to cancel. 2. The system prompts them to confirm the cancellation. 3. After confirmation, the system removes it from the booking list. 4. The system displays the success message to indicating that the booking deleted successfully.
Alternate/Exceptional Flows:

Table 4.10: Use Case Description of Make Payment

Use Case Name: Make Payment	ID: UC06	Importance Level: High
Primary Actor: Client	Use Case Type: Detail, Essential	

Stakeholders and Interests: Client: They want to make payment after a booking has been accepted.	
Brief Description: This use case describes how the client makes a booking.	
Trigger: Client wants to make a booking after their pending booking has been accepted.	
Relationships: Association : Client Include : - Extend : - Generalization: -	
Normal Flow of Events: 1. The client navigates to the Accepted Booking page. 2. The client selects a booking and clicks on the "Pay" button. 3. The payment modal will drop down then the client can select a payment method to pay. 4. The system stores the payment method in the database and shows a success message.	
Sub-flows:	
Alternate/Exceptional Flows:	

Table 4.11: Use Case Description of Provide Feedback

Use Case Name: Provide Feedback	ID: UC07	Importance Level: High
Primary Actor: Client	Use Case Type: Detail, Essential	
Stakeholders and Interests: Client: They want to sharing feedback after conducted a care service.		

Brief Description: This use case describes how the client share their feedback after a care service has been provided.
Trigger: Client wants to share their feedback to assists other users in making informed decisions.
Relationships: Association : Client Include : - Extend : - Generalization: -
Normal Flow of Events: 1. The client accesses the Booking List Page and select the Approved state and choose a booking to give some comments. 2. The feedback modal will pop out and allow the client to rank and submit feedback. 3. The system stores the feedback in the database with success message and linked to the respective caregiver.
Sub-flows:
Alternate/Exceptional Flows:

Table 4.12: Use Case Description of View Transaction History

Use Case Name: View Transaction History	ID: UC08	Importance Level: High
Primary Actor: Client, Caregiver	Use Case Type: Detail, Essential	
Stakeholders and Interests: Client: Client wants to view the transaction history after they pay for a booking.		

Caregiver: Caregiver wants to view the payment received history to ensure that the client has paid for the service.
Brief Description: This use case describes how the clients and caregivers view the transaction history. They can generate invoices and export CSV for a specific transaction.
Trigger: The client and caregiver want to view the transaction history, generate an invoice, or export CSV for a transaction.
Relationships: Association : Client, Caregiver Include : - Extend : - Generalization: -
Normal Flow of Events: 1. The client and caregiver access the Transaction History Page to view all the transactions they made before. 2. The client and caregiver can select any of the transactions to generate an invoice or export CSV. The Invoice in pdf format or CSV file will be downloaded automatically.
Sub-flows:
Alternate/Exceptional Flows:

Table 4.13: Use Case Description of Chat in Chat Box

Use Case Name: Chat in Chat Box	ID: UC09	Importance Level: High
Primary Actor: Client, Caregiver, Administrator	Use Case Type: Detail, Essential	
Stakeholders and Interests:		

Client: Client wants to chat with caregiver or administrator in the chat group. Caregiver: Caregiver wants to chat with client or administrator in the chat group. Administrator: Administrator wants to chat with the client or caregiver in the chat group.
Brief Description: This use case describes how the client, caregiver and administrator communicate with each other.
Trigger: Client, caregiver, or administrator initiates a chat session.
Relationships: Association : Client, Caregiver, Administrator Include : - Extend : - Generalization: -
Normal Flow of Events: 1. User navigates to the Chat Page within the platform. 2. User can search for conversation by name. 3. In the conversation, user can type any message in the chat box and sends it by clicking the Send button.
Sub-flows:
Alternate/Exceptional Flows:

Table 4.14: Use Case Description of Set Up Profile

Use Case Name: Set Up Profile	ID: UC10	Importance Level: High
Primary Actor: Caregiver, Client	Use Case Type: Detail, Essential	
Stakeholders and Interests: Caregiver: He/She want to set up their profile information on the platform.		

Client: He/She want to set up their profile information on the platform.
Brief Description: This use case describes how the caregiver and client setting up their profile on the platform. For client, name, email, phone number, gender and location need to be fill up. For caregiver, name, email, phone number, gender, location, availability, qualifications, experience, and about me need to be fill up.
Trigger: Caregiver and client want to update their profile information on the platform after register an account.
Relationships: Association : Caregiver, Client Include : - Extend : - Generalization: -
Normal Flow of Events: 1. The caregiver or client navigates to the Profile Page. 2. The system displays the initial profile interface with fields for entering personal details. 3. The caregiver or client enters personal information. 4. The system saves the profile information in the database. 5. The system displays a confirmation message to caregiver or client to indicate that the profile setup successfully.
Sub-flows:
Alternate/Exceptional Flows:

Table 4.15: Use Case Description of View Booking

Use Case Name: View Booking	ID: UC11	Importance Level: High
Primary Actor: Caregiver	Use Case Type: Detail, Essential	

Stakeholders and Interests: Caregiver: He/She want to view their booking list which are book from the client.	
Brief Description: This use case describes how the caregiver reviewing bookings made by the client for care services on the platform.	
Trigger: Caregivers want to view the booking and accept or decline the booking.	
Relationships: Association : Caregiver Include : Accept/Decline Booking Extend : - Generalization: -	
Normal Flow of Events: 1. The caregiver navigates to the Booking List page. 2. The system searches for care service bookings assigned to the caregiver, made by the client. Continue to E2: No Booking Found. 3. The system displays the care service bookings lists. 4. The caregiver selects a specific booking from the list and decide on the booking. Continue to S4-1: Accept Booking or S4-2: Decline Booking.	
Sub-flows: S4-1: Accept Booking 4a. If the caregiver decides to accept the booking, the system processes the caregiver's acceptance request. 4b. The system updates the booking status to "accepted" in the database. S4-2: Decline Booking 4a. If the caregiver decides to decline the booking, the system processes the caregiver's decline request.	

4b. The system updates the booking status to "declined" in the database.
Alternate/Exceptional Flows: E2: No Booking found. 1. If the system does not find any bookings assigned to the caregiver, the system will display a message "No Booking Found".

Table 4.16: Use Case Description of View Feedback

Use Case Name: View Feedback		ID: UC12	Importance Level: High
Primary Actor: Caregiver	Use Case Type: Detail, Essential		
Stakeholders and Interests: Caregiver: He/She want to review feedback provided by the client.			
Brief Description: This use case describes how the caregiver reviewing feedback provided by the client after receiving care services.			
Trigger: Caregivers want to check the feedback given by elderly individuals or family members to make improvement.			
Relationships: Association : Caregiver Include : - Extend : - Generalization: -			
Normal Flow of Events: 1. The caregiver navigates to the Feedback List page. 2. The system displays a list of feedback provided by elderly individuals or family members. Continue to E2: No Feedback Found.			
Sub-flows:			

Alternate/Exceptional Flows: E2: No Feedback Found 1. If the system does not find any feedback, the system will display an error message like "No Feedback Found" to indicating there is no feedback from elderly individual and family member after using the service.

Table 4.17: Use Case Description of View Client's Profile

Use Case Name: View Client's Profile		ID: UC13	Importance Level: High
Primary Actor: Caregiver	Use Case Type: Detail, Essential		
Stakeholders and Interests: Caregiver: They want to view client's detail information.			
Brief Description: This use case describes how the caregiver view detailed information about a client's profile.			
Trigger: Client wants to gather more information about client through their profile.			
Relationships: Association : Caregiver Include : - Extend : - Generalization: -			
Normal Flow of Events: 1. The caregiver navigates to the client's profile page. 2. The system displays the detailed profile of the selected client that includes the client's name, photo, email, phone number, gender, about me, and location.			

3. The caregiver reviews the details information provided in the profile.
Sub-flows:
Alternate/Exceptional Flows:

Table 4.18: Use Case Description of Manage Care Service

Use Case Name: Manage Care Service	ID: UC14	Importance Level: High
Primary Actor: Caregiver	Use Case Type: Detail, Essential	
Stakeholders and Interests: Caregiver: Caregiver wants to manage their own care services.		
Brief Description: This use case describes how the caregiver managing care service, including creating, viewing, updating, deleting service.		
Trigger: Caregiver intends to manage service on the platform.		
Relationships: Association : Caregiver Include : Create/ Read/ Update/ Delete Service Extend : - Generalization: -		
Normal Flow of Events: 1. If the caregiver wants to create the service, S1, Create Service is performed. 2. If the caregiver wants to view the service, S2: View Service is performed. 3. If the caregiver wants to edit the service, S3: Edit Service is performed.		

4. If the caregiver wants to delete the service, S4: Delete Service is performed.
Sub-flows: S1: Create Service 1. The caregiver navigates to Service List Page and click on the Add Service button to create a care service. 2. The caregiver will be redirected to the Add Service Page and required to enter the service type, service description, etc. to create a new care service. 3. The system saves the details into the database and confirms the successful creation of a service with a success message. S2: View Service 1. The caregiver navigates to Service List Page to view a list of their existing services. 2. The caregiver selects a specific service to view detailed information. 3. The system displays details about the services, including service name, descriptions, etc. S3: Edit Service 1. The caregiver navigates to Service List Page to view a list of their existing services. 2. The caregiver selects a service to do modification. 3. The system redirects the caregiver to an Update Service Page to allow caregiver to modify the existing service. 4. After making changes, they can click on the Update button, and the system updates the service information accordingly. 5. The system shows the success message to indicating that the service updated successfully. S4: Delete Service 1. The caregiver navigates to Service List Page to view a list of their existing services.

2. The caregiver selects a service to delete. 3. The system prompts them to confirm the deletion. 4. After confirmation, the system removes it from the service list. 5. The system displays the success message to indicating that the service deleted successfully.
Alternate/Exceptional Flows:

Table 4.19: Use Case Description of View Appointment

Use Case Name: View Appointment	ID: UC15	Importance Level: High
Primary Actor: Caregiver	Use Case Type: Detail, Essential	
Stakeholders and Interests: Caregiver: Caregiver wants to view all the appointment dates easily.		
Brief Description: This use case describes how the caregiver views the appointment date on the Appointment Calendar Page.		
Trigger: The caregiver wants to check all their appointment quickly and easily.		
Relationships: Association : Care Include : - Extend : - Generalization: -		
Normal Flow of Events: 1. The caregiver accesses the Appointment Calendar Page. 2. This page shows all the appointments. 3. The caregiver can click on the date link to navigate to the booking details page.		

Sub-flows:
Alternate/Exceptional Flows:

Table 4.20: Use Case Description of Manage Account

Use Case Name: Manage Account	ID: UC16	Importance Level: High
Primary Actor: Administrator	Use Case Type: Detail, Essential	
Stakeholders and Interests: Administrator: Administrator wants to manage client and caregiver account.		
Brief Description: This use case describes how the administrator managing user accounts on the platform. This includes viewing or deleting a user account.		
Trigger: Administrator wants to perform account management activities on the platform.		
Relationships: Association : Administrator Include : View/ Detele Account Extend : - Generalization: -		
Normal Flow of Events: 1. The administrator navigates to the Manage User Account Page. 2. The administrator can perform various actions related to the user accounts: 2.1 If the administrator wants to view the details of the existing account, S1: View Account will perform. 2.2 If the administrator wants to delete existing account, S2: Delete Account will perform.		

Sub-flows: S1: View Account 1. The admin selects a user account to view its detail. 2. The system displays details information about the user account, including name, email, contact number etc. S2: Delete Account 1. The admin selects a user account to delete. 2. The system prompts admin to confirm the deletion. 3. After confirmation, the system removes the account and all associated data from database. 4. The system displays the success message to indicating that the booking deleted successfully.
Alternate/Exceptional Flows:

Table 4.21: Use Case Description of Delete Service

Use Case Name: Delete Service	ID: UC17	Importance Level: High
Primary Actor: Administrator	Use Case Type: Detail, Essential	
Stakeholders and Interests: Administrator: Administrator wants to delete the service that was created by the caregiver.		
Brief Description: This use case describes how the administrator deletes the services in the platform.		
Trigger: Administrator wants to delete a service that does not meet the standard.		
Relationships: Association : Administrator Include :		

Extend : - Generalization: -
Normal Flow of Events: 1. The admin navigates to the Manage Service Page. 2. The admin selects a service and clicks on the "Delete" button. 3. An alert message will show to ask for the confirmation of the deletion. 4. Upon confirmation is confirmed, the service will be deleted and removed from the database. 5. A success message will show to indicate the service has been deleted successfully.
Sub-flows:
Alternate/Exceptional Flows:

4.5 Prototype

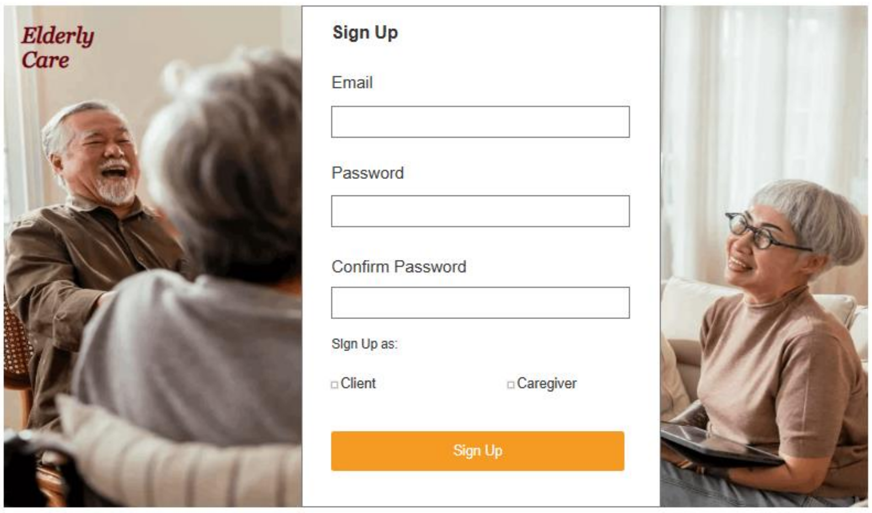


Figure 4.2: Sign Up Page

In Sign Up Page, users are required to enter their email, password, confirm password, and select the role as client or caregiver to sign up an account.

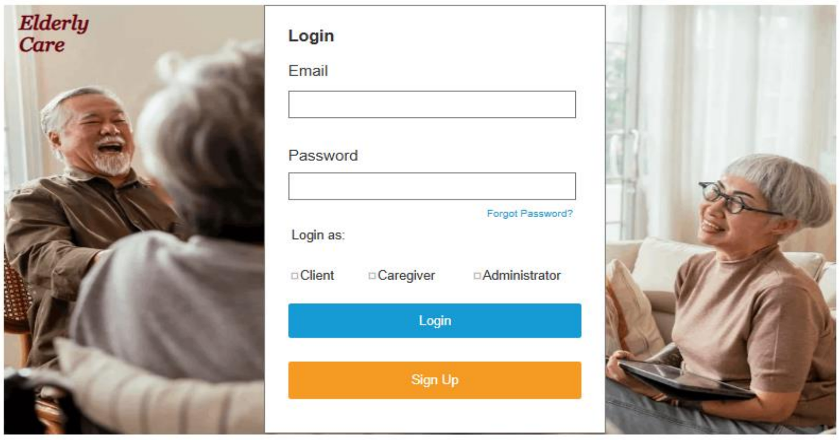


Figure 4.3: Login Page

In Login Page, users are required to enter their email, password, and choose their role to login the platform.

4.5.1 Prototype of Client

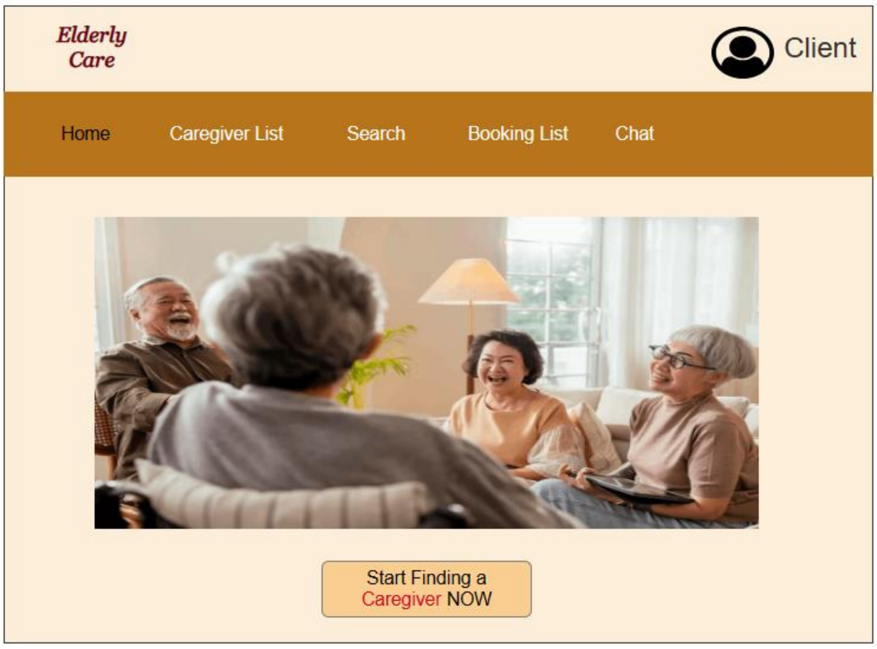


Figure 4.4: Home Page

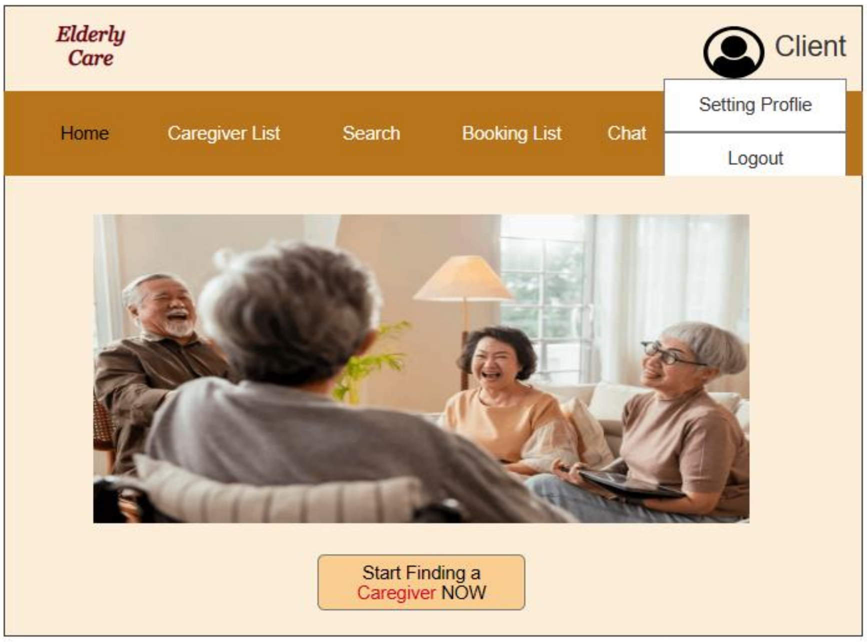


Figure 4.5: Manage Account Page

Client can click on the account logo to setting profile or logout.

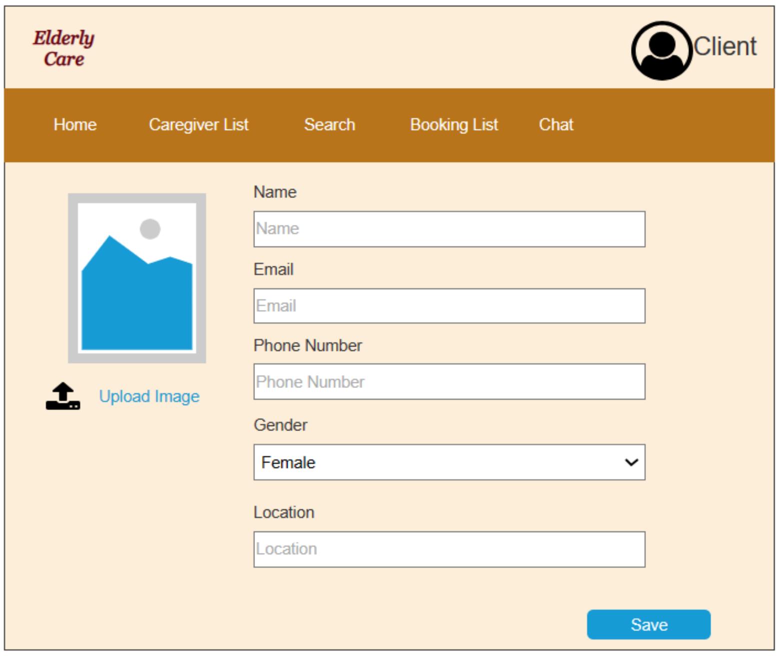
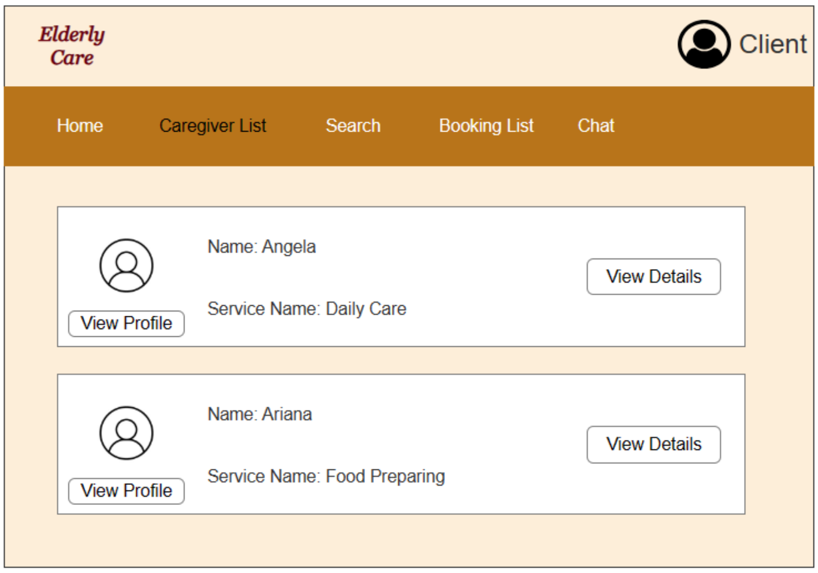


Figure 4.6: Setting Profile Page



Name: Angela

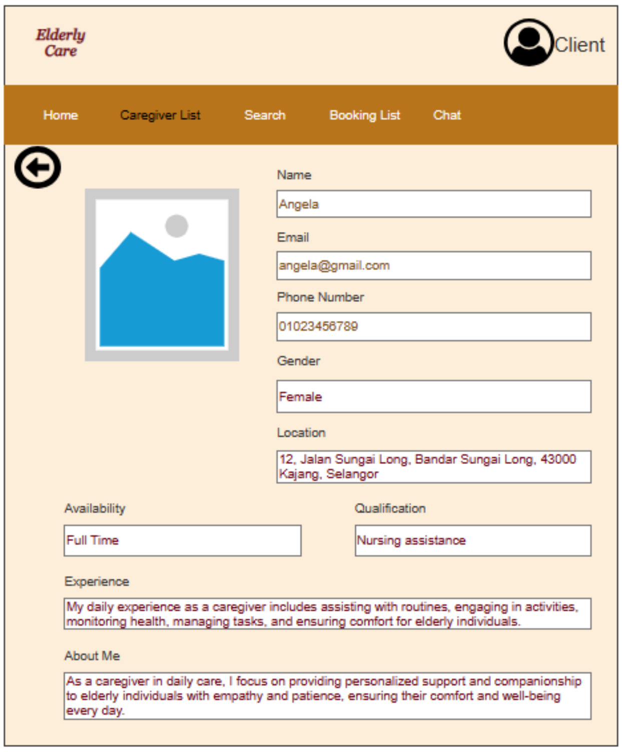
Service Name: Daily Care

Name: Ariana

Service Name: Food Preparing

Figure 4.7: Caregiver List Page

Client will be redirected to View Caregiver Profile Page when they click on the "View Profile" button and redirected to View Service Detail Page once they click on the "View Details" button.



Name: Angela

Email: angela@gmail.com

Phone Number: 01023456789

Gender: Female

Location: 12, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

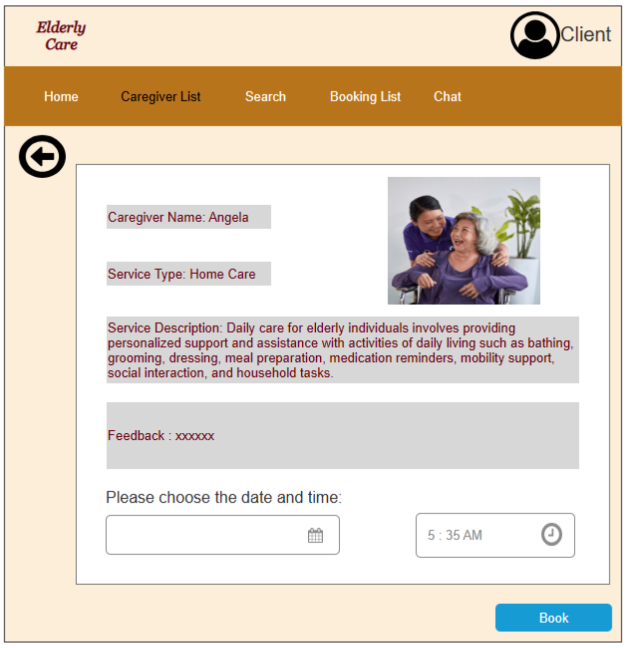
Availability: Full Time

Qualification: Nursing assistance

Experience: My daily experience as a caregiver includes assisting with routines, engaging in activities, monitoring health, managing tasks, and ensuring comfort for elderly individuals.

About Me: As a caregiver in daily care, I focus on providing personalized support and companionship to elderly individuals with empathy and patience, ensuring their comfort and well-being every day.

Figure 4.8: View Caregiver Profile Page



Caregiver Name: Angela

Service Type: Home Care

Service Description: Daily care for elderly individuals involves providing personalized support and assistance with activities of daily living such as bathing, grooming, dressing, meal preparation, medication reminders, mobility support, social interaction, and household tasks.

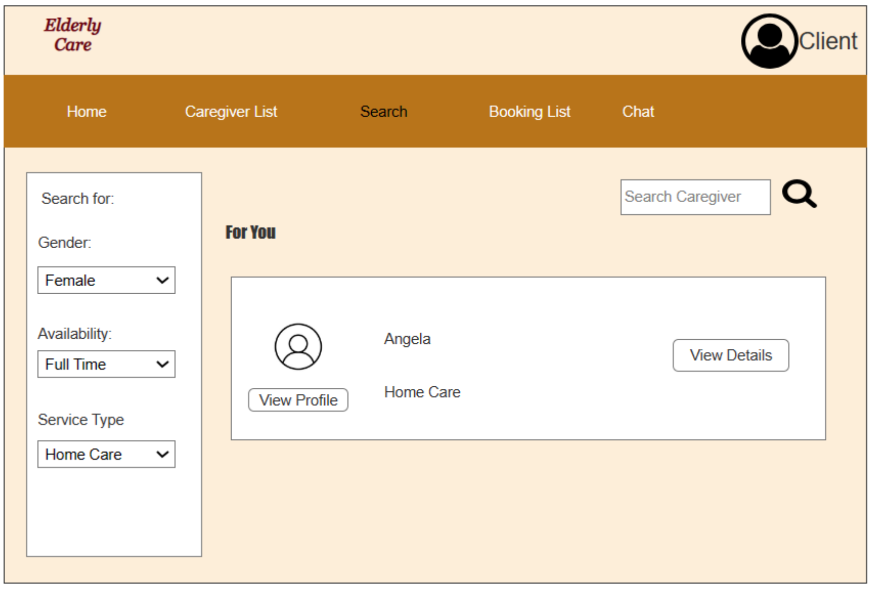
Feedback : xxxxxx

Please choose the date and time:

5 : 35 AM

Figure 4.9: View Service Detail Page

Client may book for the service by choosing the date and time in this page.



Search for:

Gender: Female

Availability: Full Time

Service Type: Home Care

Angela

Home Care

Figure 4.10: Search Service Page

Client may search for the service they want by applying the filter. After applying the filter, the caregiver matching list will be shown.

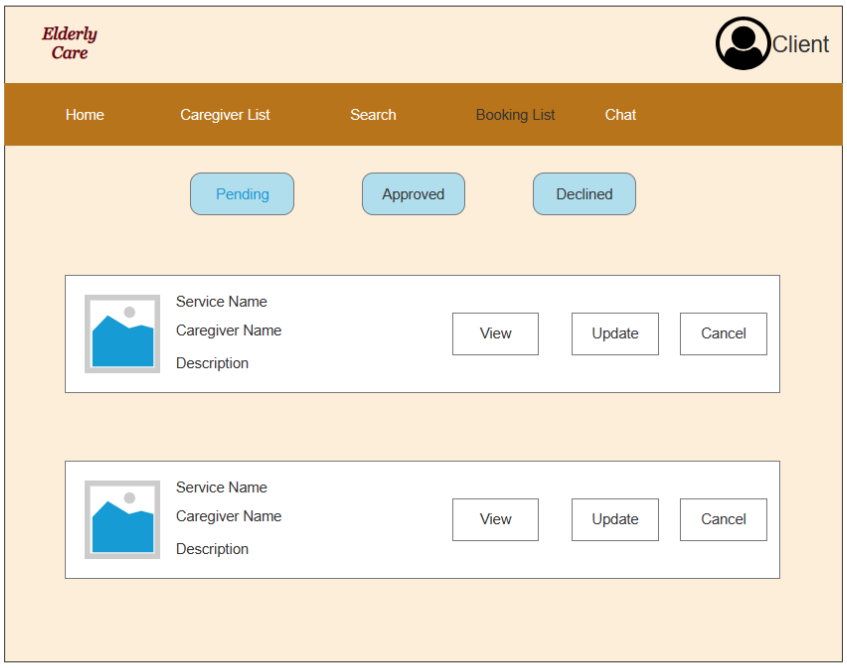
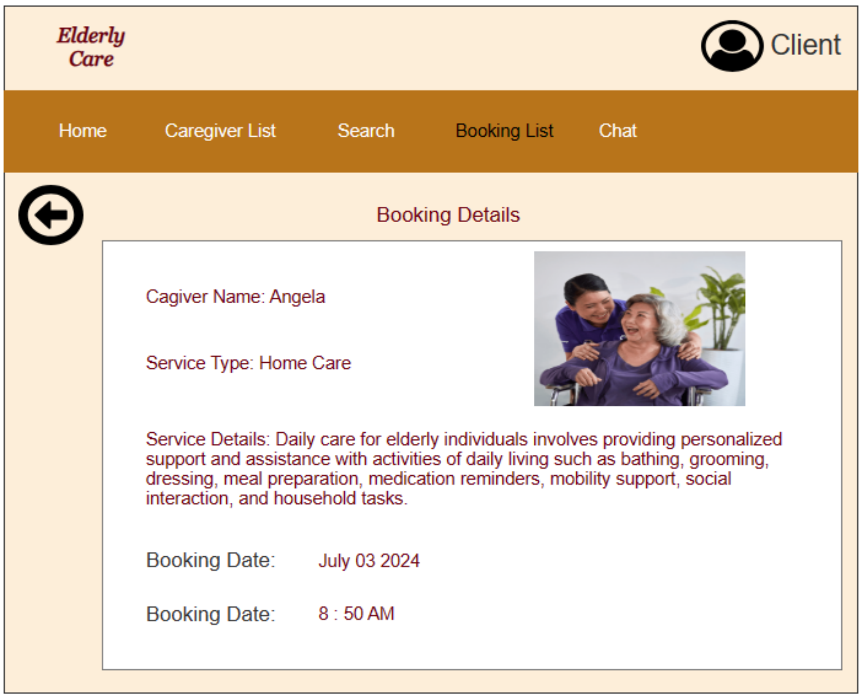


Figure 4.11: Booking List Page (Pending state)

This page shows all the booking that are pending for approval and declination. Client is allowed to view, update and cancel their booking in this page.



Booking Details

Cagiver Name: Angela

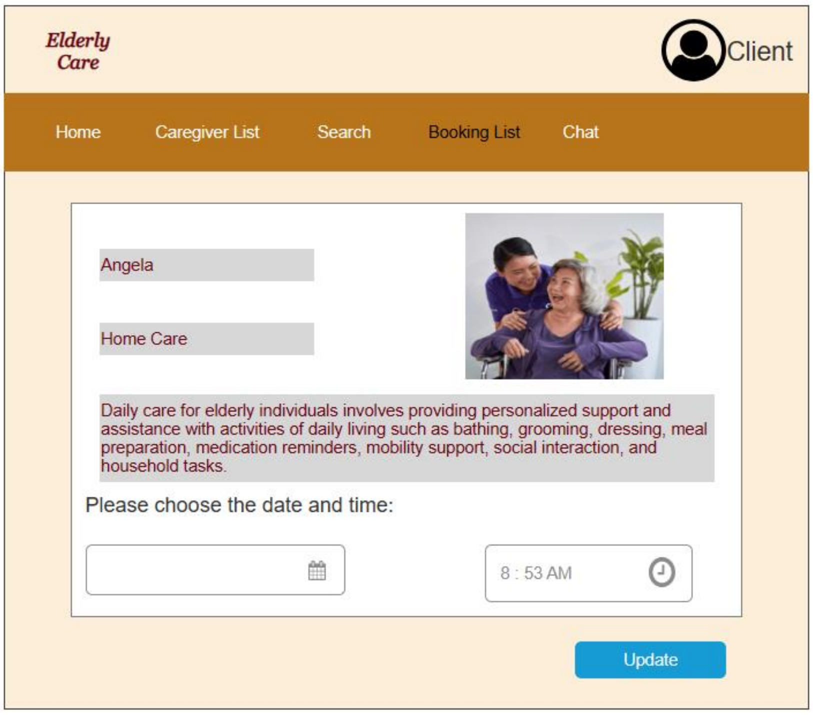
Service Type: Home Care

Service Details: Daily care for elderly individuals involves providing personalized support and assistance with activities of daily living such as bathing, grooming, dressing, meal preparation, medication reminders, mobility support, social interaction, and household tasks.

Booking Date: July 03 2024

Booking Date: 8 : 50 AM

Figure 4.12: View Booking Page



Angela

Home Care

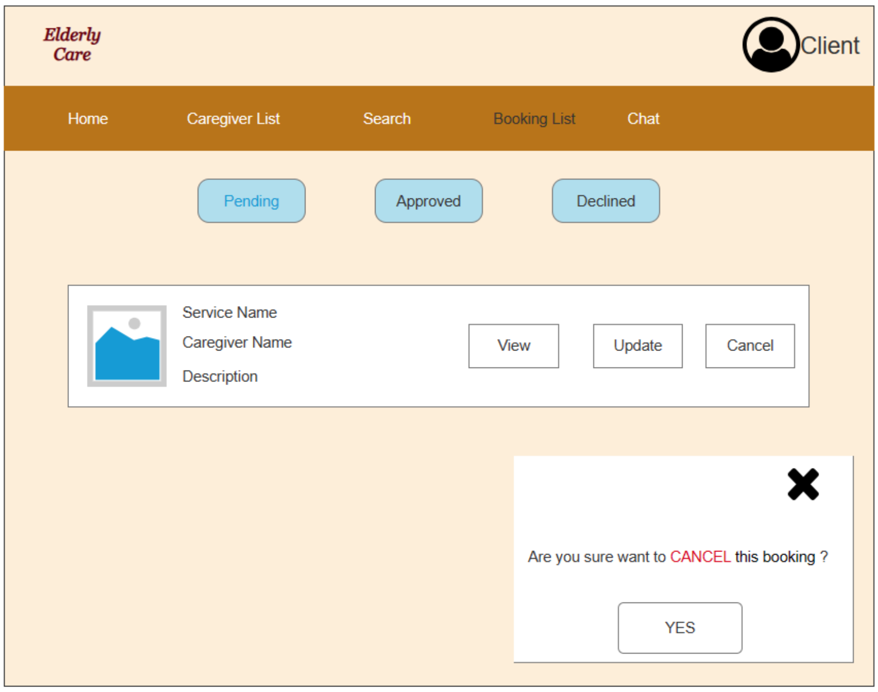
Daily care for elderly individuals involves providing personalized support and assistance with activities of daily living such as bathing, grooming, dressing, meal preparation, medication reminders, mobility support, social interaction, and household tasks.

Please choose the date and time:

8 : 53 AM

Figure 4.13: Update Booking Page

Client may renter the date and time and click on the Update button to update their booking.



Are you sure want to CANCEL this booking ?

Figure 4.14: Delete Booking Page

Once the client clicks on the Cancel Button, a confirmation will pop out to make sure if the client wants to cancel the booking.

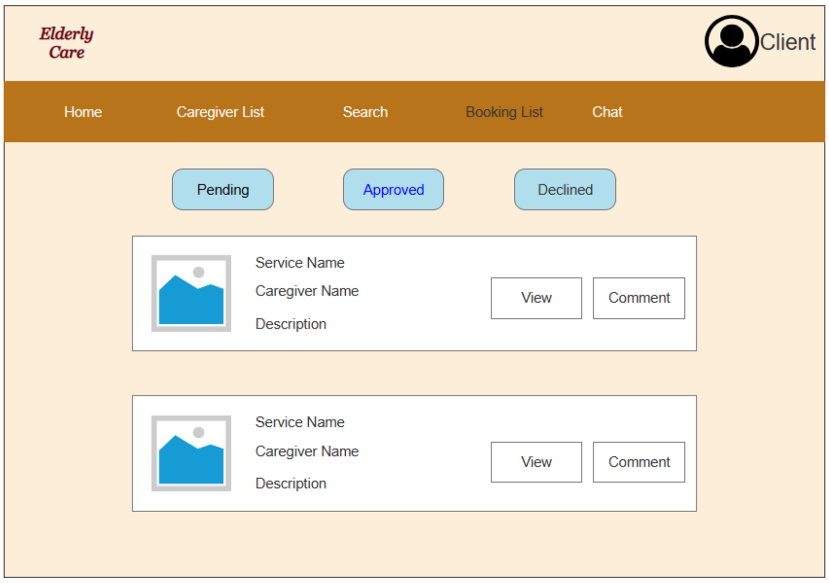


Figure 4.15: Booking List Page (Approved state)

In this page, all the approved booking is shown, and clients are only allowed to view the booking and give comment after they conduct the booking.

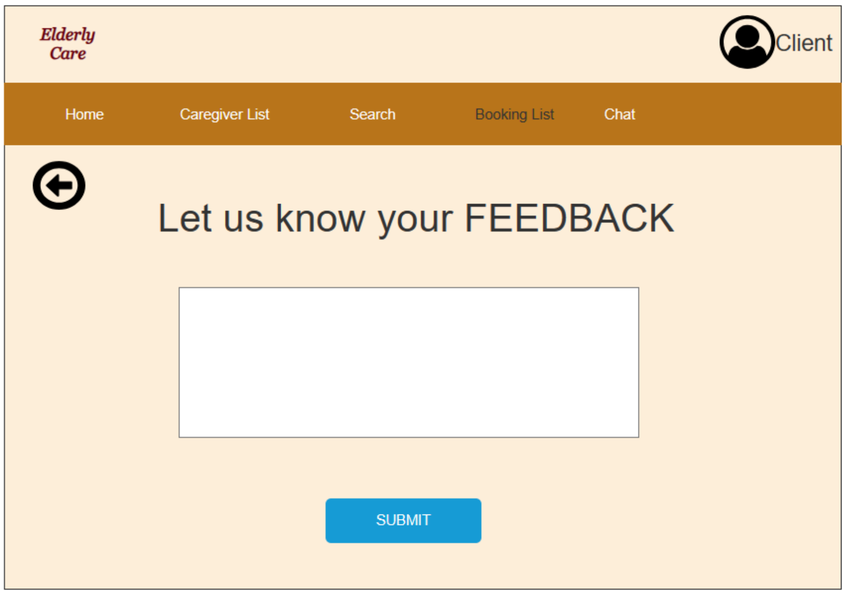


Figure 4.16: Feedback Page

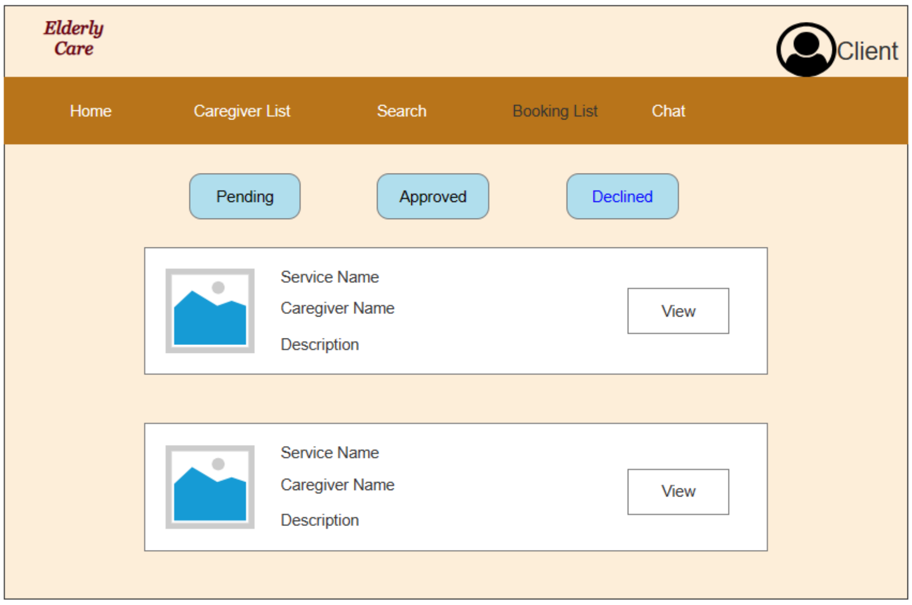


Figure 4.17: Booking List Page (Decline state)

For the Declined bookings, clients are only allowed to view their bookings.

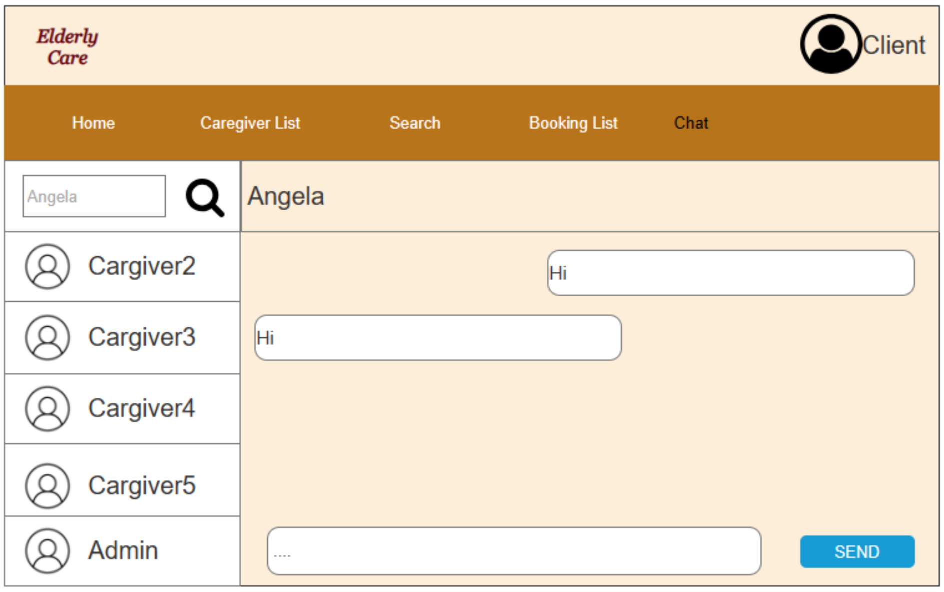


Figure 4.18: Chat Page

Client can search for the name of the caregivers or administrators to start for a conversation.

4.5.2 Prototype of Caregiver

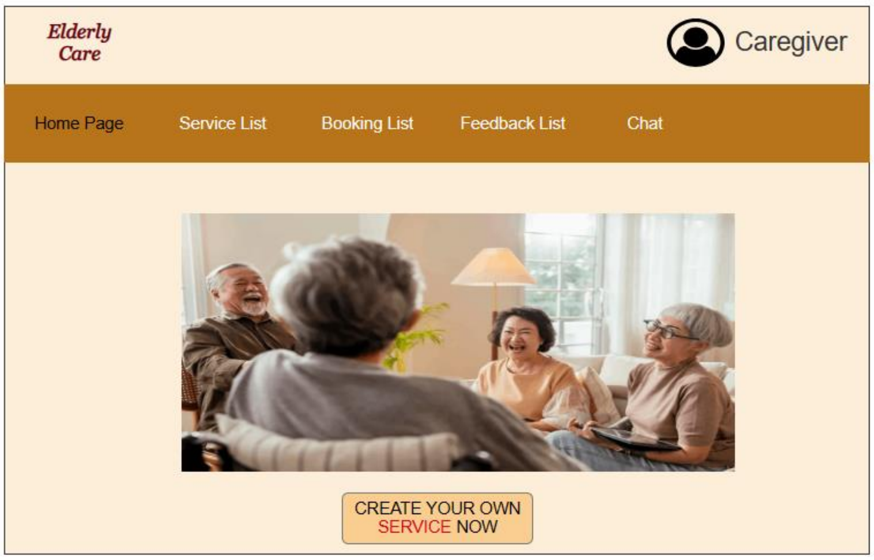


Figure 4.19: Home Page

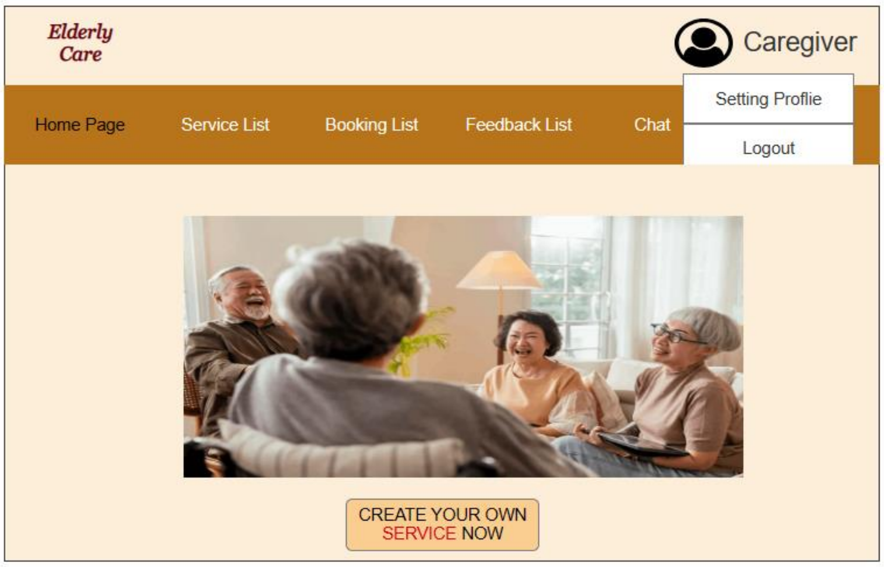


Figure 4.20: Manage Account Page

Caregiver can click on the account logo to setting profile or logout.

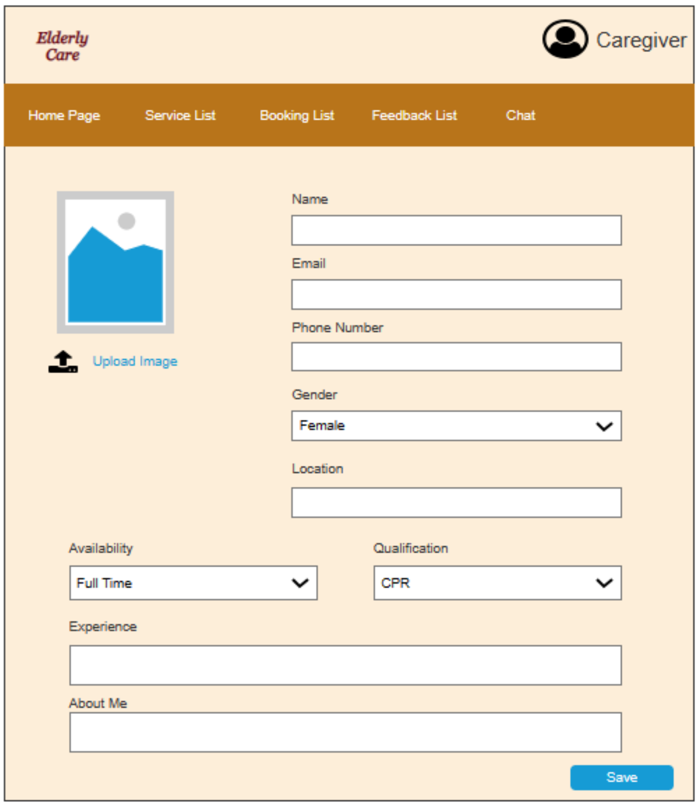


Figure 4.21: Setting Profile Page

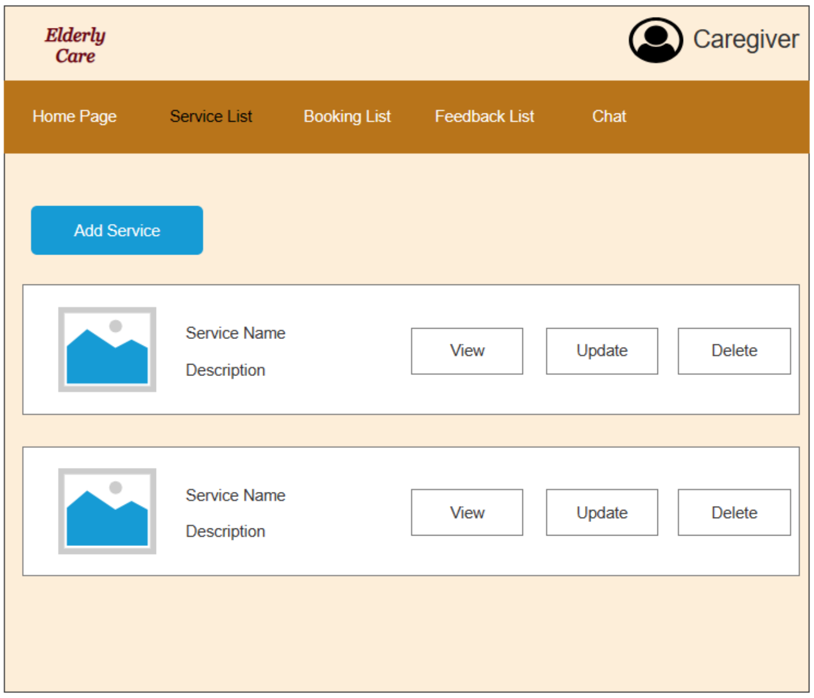
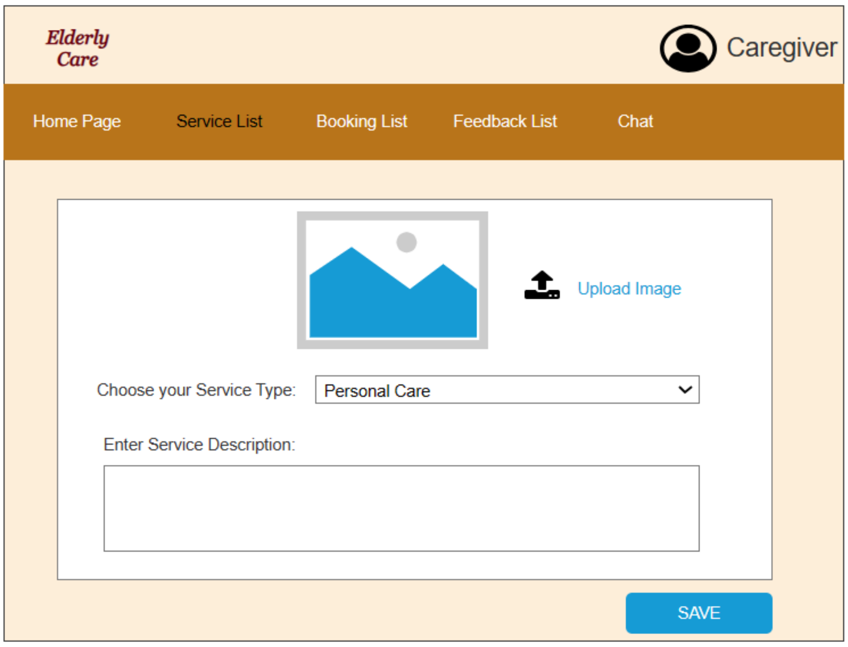


Figure 4.22: Service List Page

In this page, caregiver can choose to add a new service and view/update/delete their existing services.

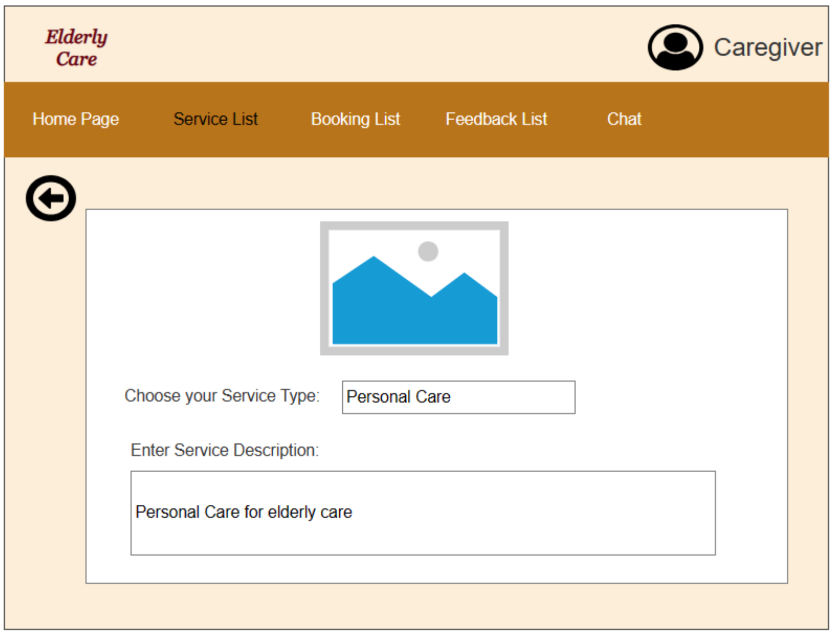


Choose your Service Type: Personal Care

Enter Service Description:

Figure 4.23: Create Service Page

In this page, caregiver is required to choose their service type and enter service description to create a new service.



Choose your Service Type: Personal Care

Enter Service Description: Personal Care for elderly care

Figure 4.24: View Service Page

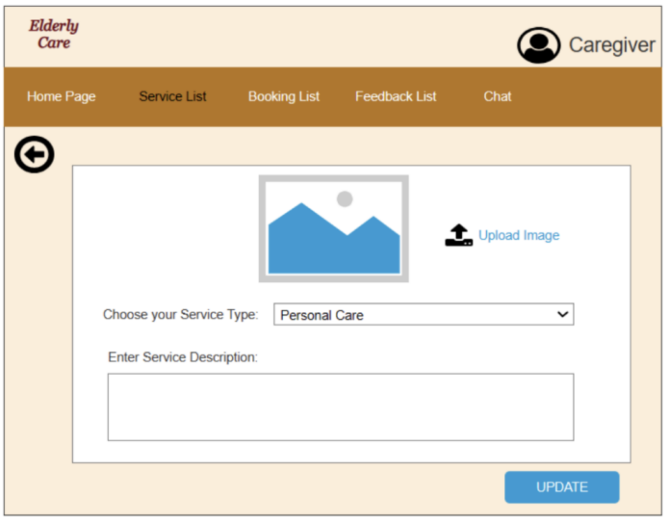


Figure 4.25: Update Service Page

In this page, caregiver is required to rechoose their service type and re-enter service description to update the selected service.

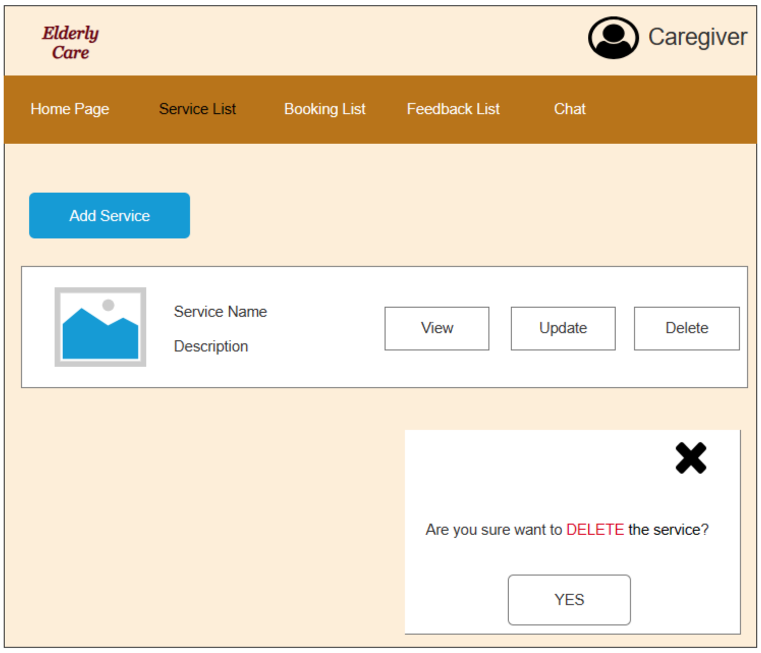


Figure 4.26: Delete Service Page

Once the caregiver clicks on the Delete Button, a confirmation will pop out to make sure if the client wants to delete their service.

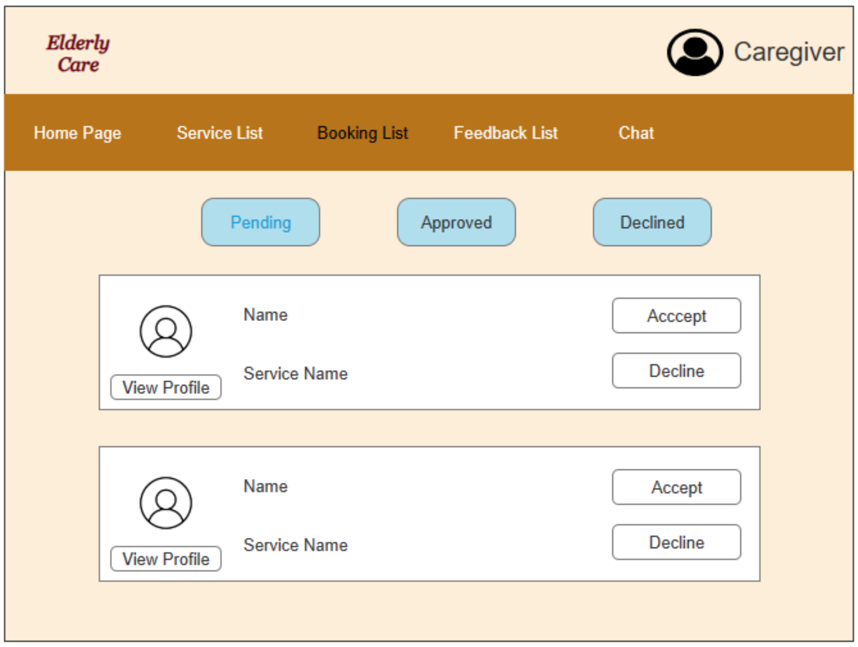
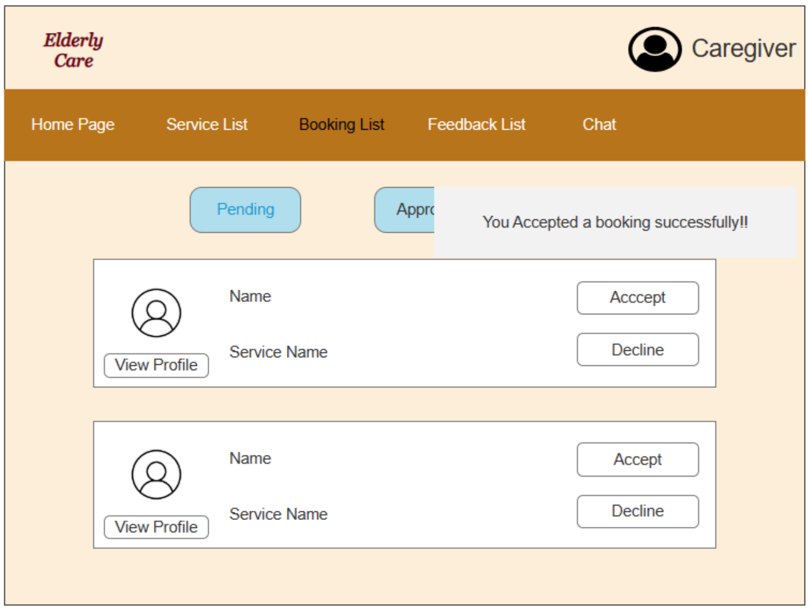


Figure 4.27: Booking List Page (Pending)

For the pending bookings from clients, caregiver can choose to accept or decline the bookings. Caregiver is also allowed to view the client's profile.



You Accepted a booking successfully!!

Figure 4.28: Message shown when click on Accept Button

If the caregiver clicks on the Accept button, a confirmation message will appear to indicate that the booking is accepted successfully.

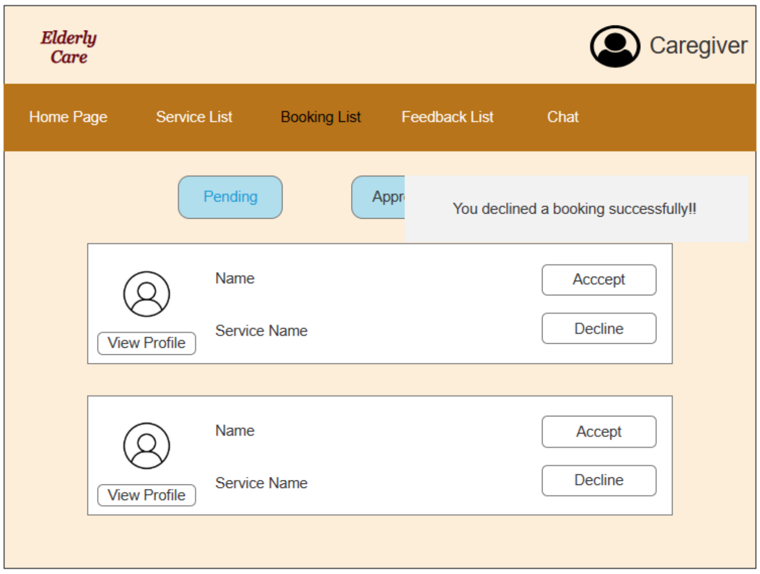


Figure 4.29: Message shown when click on Decline Button

If the caregiver clicks on the Decline button, a confirmation message will appear to indicate that the booking is declined successfully.

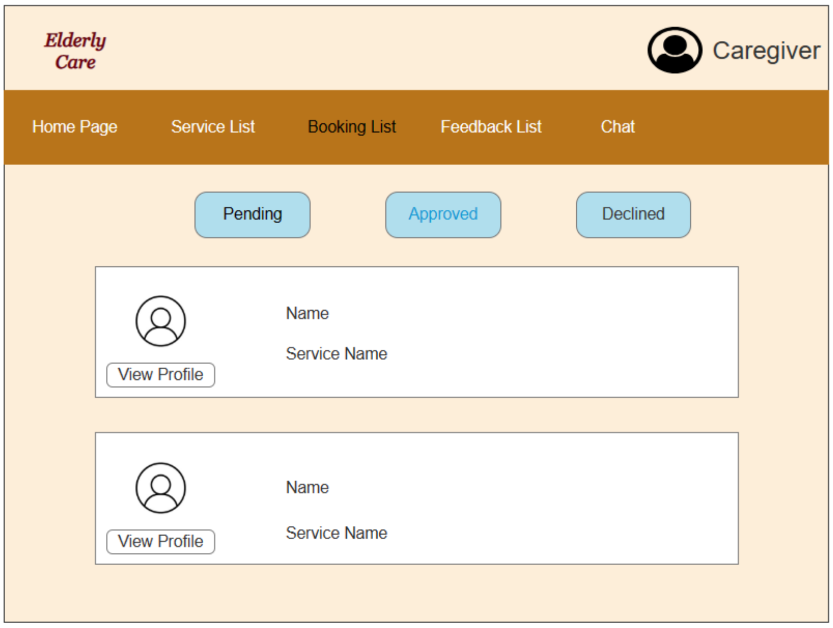


Figure 4.30: Booking List Page (Approved)

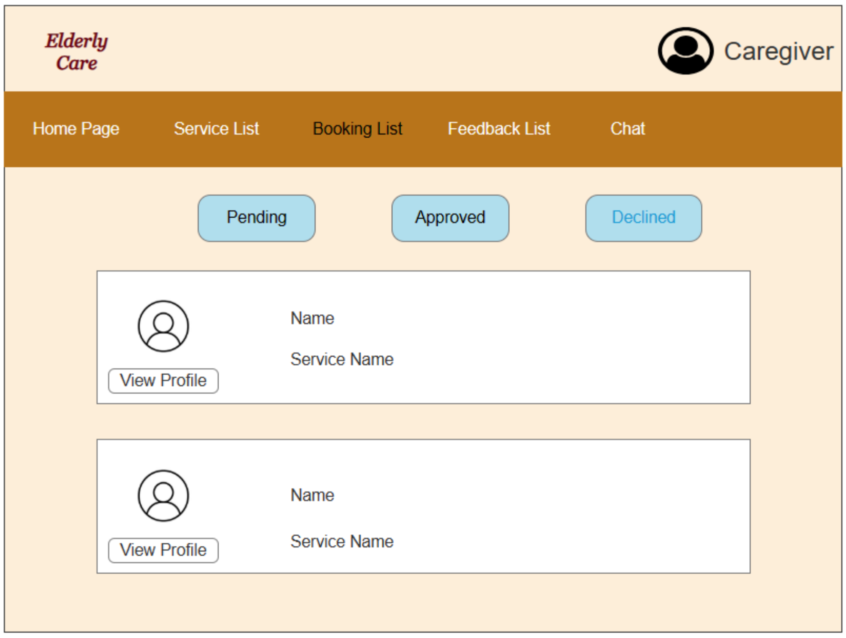


Figure 4.31: Booking List Page (Declined)

Caregiver is only allowed to view the client's profile in these two pages.

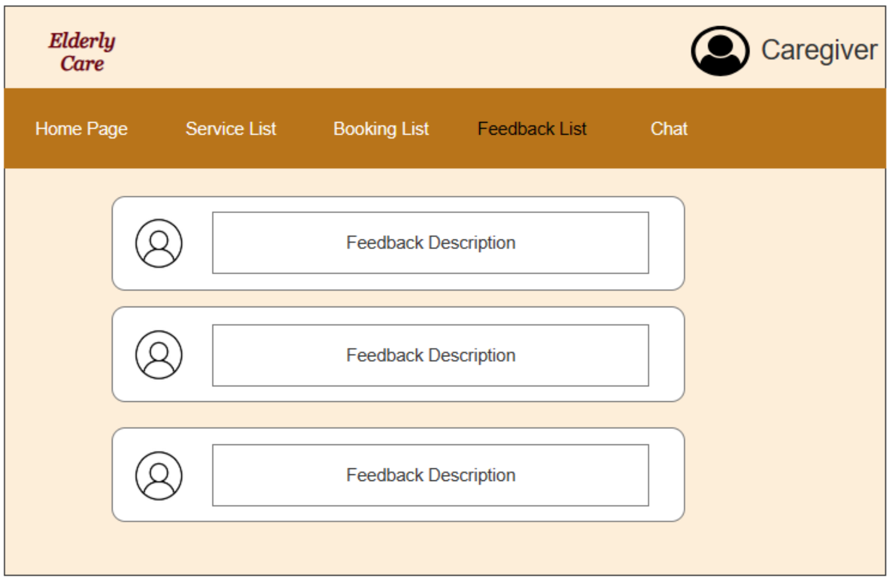


Figure 4.32: Feedback List Page

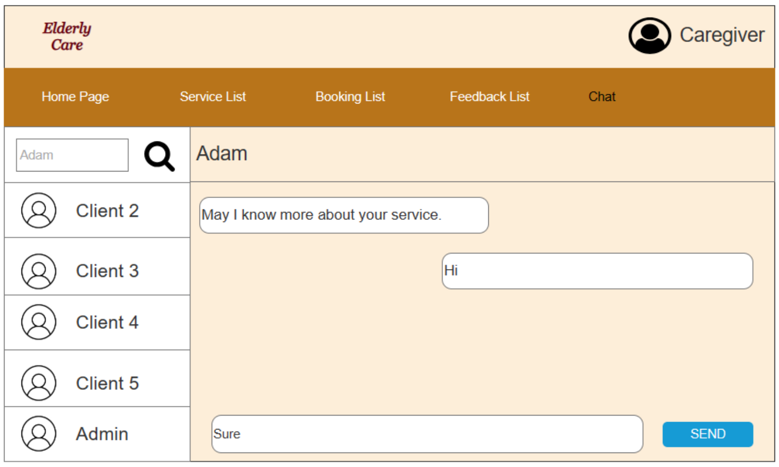
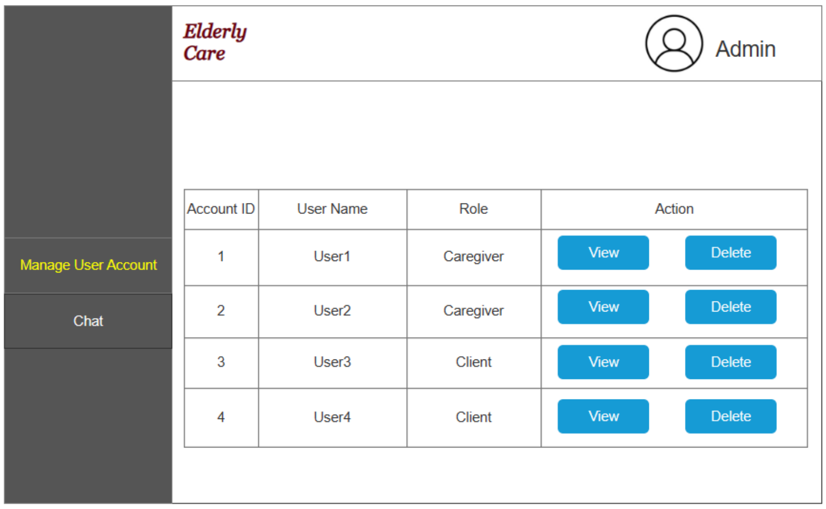


Figure 4.33: Chat Page

Caregiver can search for the name of the clients or administrators to start for a conversation.

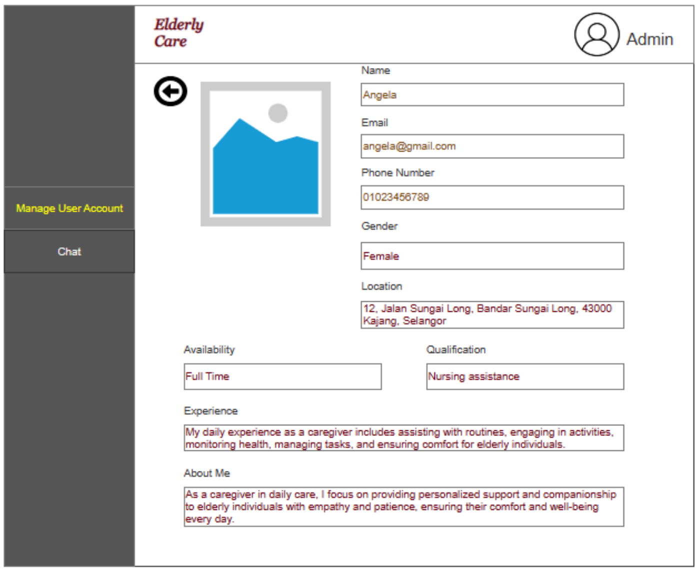
4.5.3 Prototype of Administrator



Account ID	User Name	Role	Action	
1	User1	Caregiver	View	Delete
2	User2	Caregiver	View	Delete
3	User3	Client	View	Delete
4	User4	Client	View	Delete

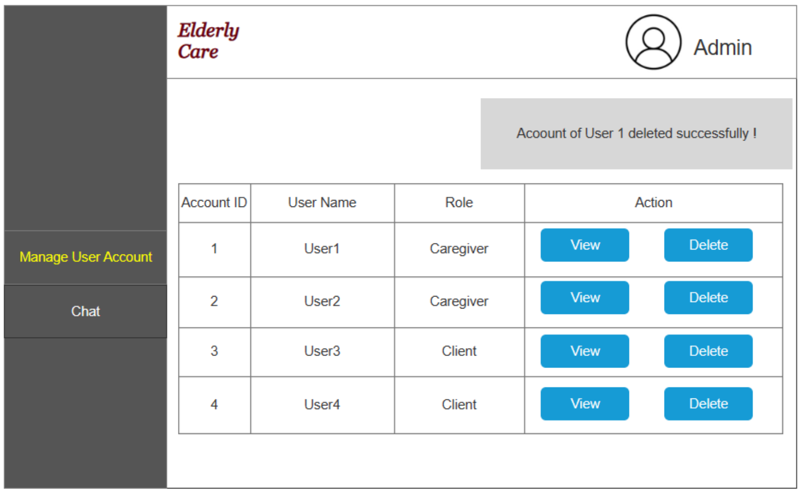
Figure 4.34: Manage User Account Page

In this page, admin is allowed to view the profile of the client or caregiver and delete their account.



Name	Angela
Email	angela@gmail.com
Phone Number	01023456789
Gender	Female
Location	12, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor
Availability	Full Time
Qualification	Nursing assistance
Experience	My daily experience as a caregiver includes assisting with routines, engaging in activities, monitoring health, managing tasks, and ensuring comfort for elderly individuals.
About Me	As a caregiver in daily care, I focus on providing personalized support and companionship to elderly individuals with empathy and patience, ensuring their comfort and well-being every day.

Figure 4.35: View Profile Page

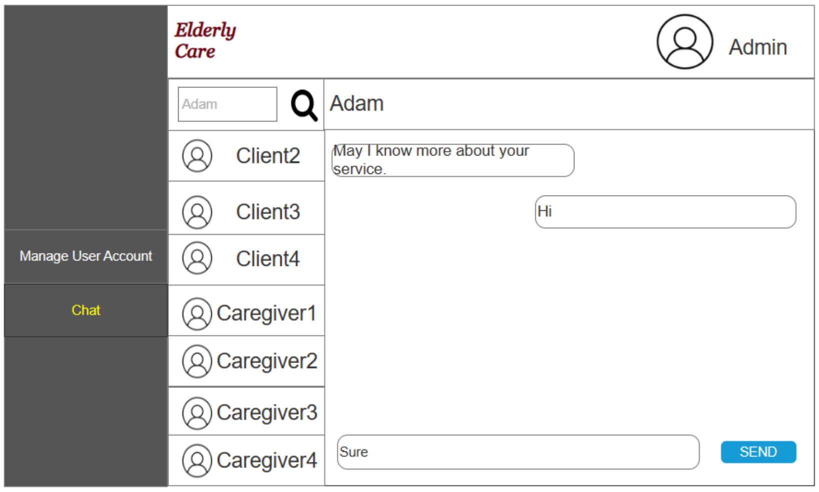


Account of User 1 deleted successfully !

Account ID	User Name	Role	Action	
1	User1	Caregiver	View	Delete
2	User2	Caregiver	View	Delete
3	User3	Client	View	Delete
4	User4	Client	View	Delete

Figure 4.36: Message shown when clicking Delete Button

Once admin clicked on the Delete button, a successful message will be shown to indicate that the account has been deleted successfully.



May I know more about your service.

Hi

Sure

Figure 4.37: Chat Page

Admin can search for the name of the clients or caregivers to start for a conversation.

4.6 Summary

In Summary, the necessary requirements had been obtained by conducting the literature review. There are 12 functional requirements for both the client and caregiver and 5 functional requirements for the administrator as well as 5 non-functional requirements. A use case diagram has been drawn and correctly outlined with 12 use case descriptions. The ERD diagram is also created to show the connection among tables. Finally, a prototype had been developed to demonstrate the interface and for the purpose of further system implementation.

CHAPTER 5
SYSTEM DESIGN

5.1 Introduction

This chapter will focus on the system design of the project. This chapter is divided into three sections. Section 5.2 will focus on the system architecture design of the platform. Section 5.3 will show all the designed UML diagrams which include activity diagrams and class diagrams. The last section 5.4 will show the database design including the ERD diagram and data dictionary of the database.

5.2 System Architecture Design

As stated in Section 1.4, Laravel MVC architecture was used throughout the entire development process.

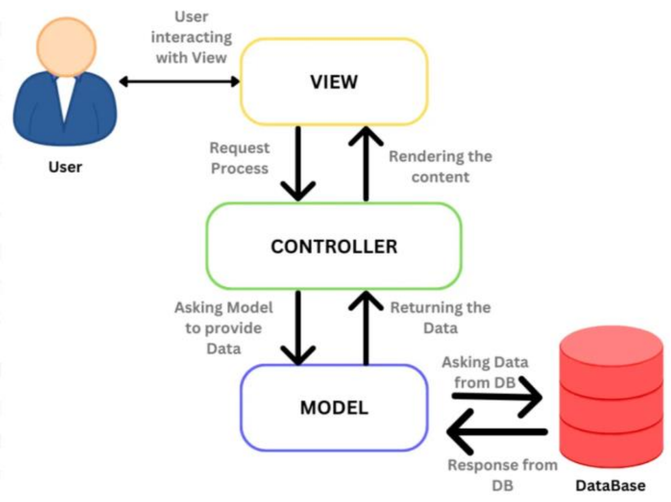


Figure 5.1: Model-View-Controller Architectural Pattern (Sadika, 2023).

From this architecture, Model can used to interact with the database and perform data manipulation. Between the Model and View, the Controller serves as a mediator. It handles the user's request, asks the Model to provide data, and updates the view. View is used to display data from the Model to users and handles the request process (Sadika, 2023).

Additionally, MySQL was utilized as the database management system while WampServer was used as a web development environment which allow developers to create web applications with Apache2, PHP, and a MySQL database. PhpMyAdmin in WampServer was utilized for easy and efficient database management (F and F, 2024).

5.3 Designed UML Diagram

5.3.1 Class Diagram

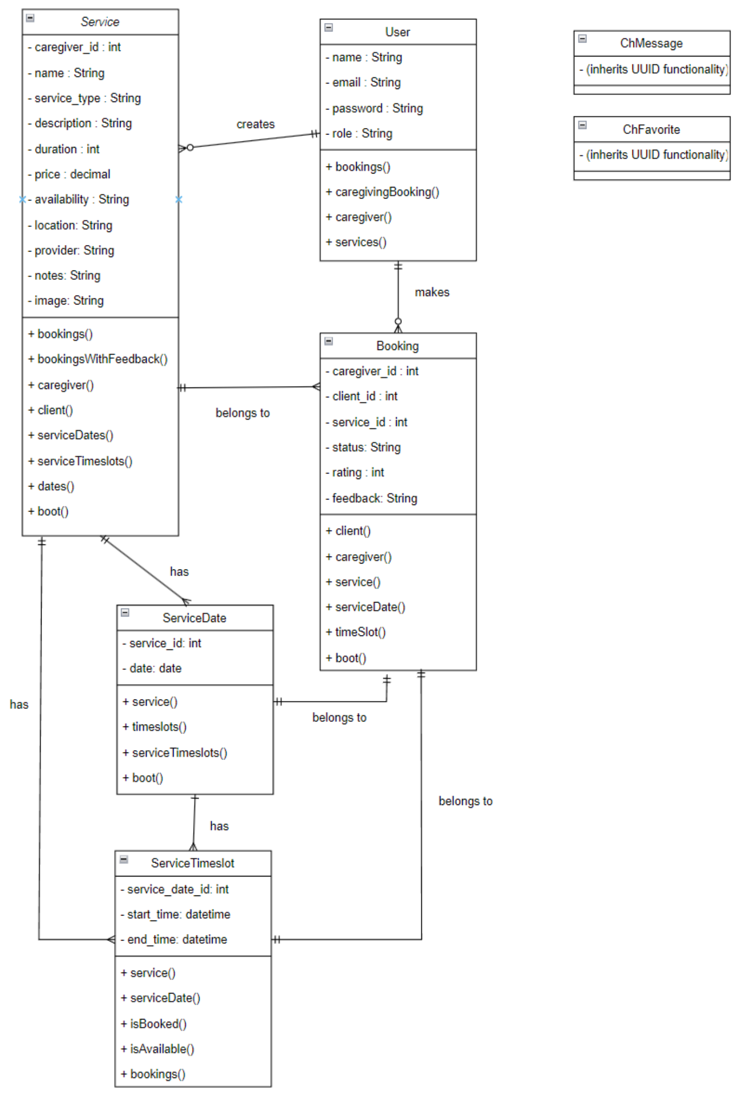


Figure 5.2: Class Diagram

The above diagram illustrates the key models of the Elder Care and Assistance Booking Platform and their relationships. The diagram provides a comprehensive view of the system structure. The User model represents different users by distinguishing roles. The Service model details the services provided by the caregiver. The Booking model connects customers with caregivers and services. Bookings are associated with specific service dates and time slots which ensure that the service is available at a specific time. The ServiceDate model indicates the available date of a specific service while the ServiceTimeslot model further refines the availability by specifying the start and end time of each date. In addition, the ChMessage and ChFavorite models combine UUID functionality for unique identification. These models may facilitate communication between customers and caregivers and enable them to mark other users as favorites.

5.3.2 Activity Diagram

Figure 5.3 to Figure 5.19 shows the Activity Diagram based on each use case

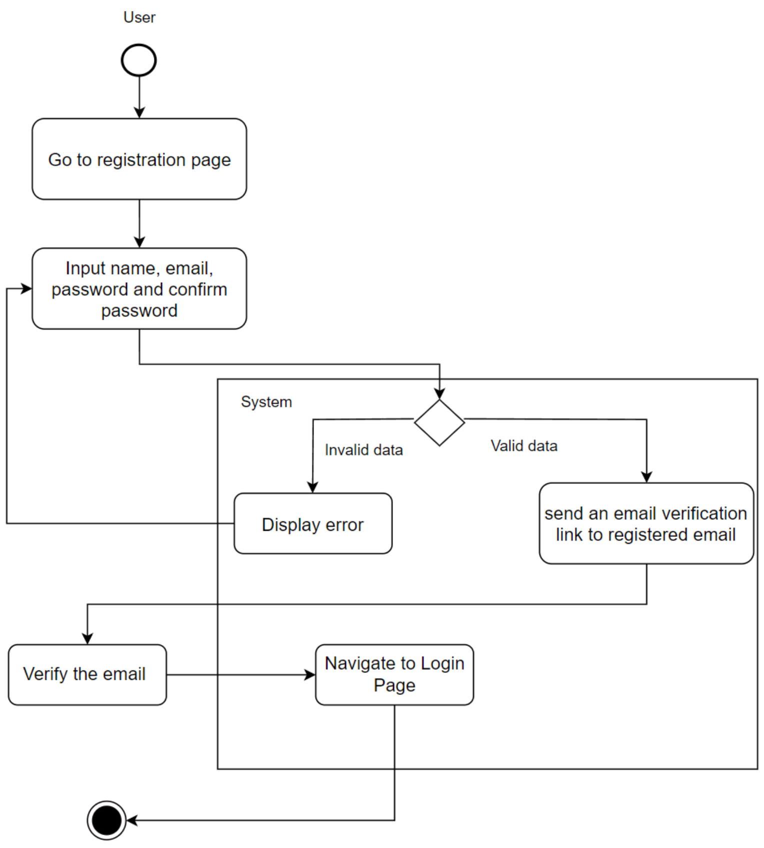


Figure 5.3: Activity Diagram for Register Account

For registration, the user may go to the registration page and input their name, email, password, and confirm password. After this, the system will validate the input. If the input has invalid data, the system will display an error, else the system will send an email verification link to the registered email. Once the user verifies the email, the system will redirect the page to the Login page.

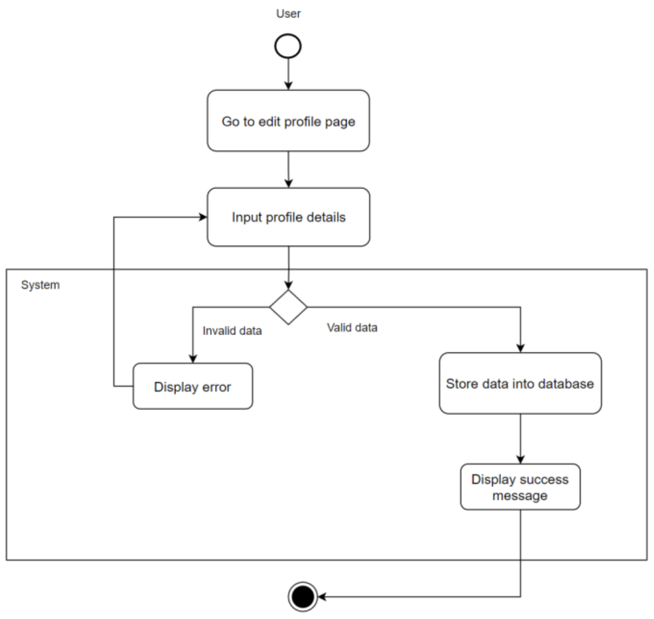


Figure 5.4: Activity Diagram for Setup Profile

To set up a profile, users may go to the edit profile page and input their profile details. After this, the system will validate the input. If the input has invalid data, the system will display an error. Otherwise, the system will store data in the database and display a success message.

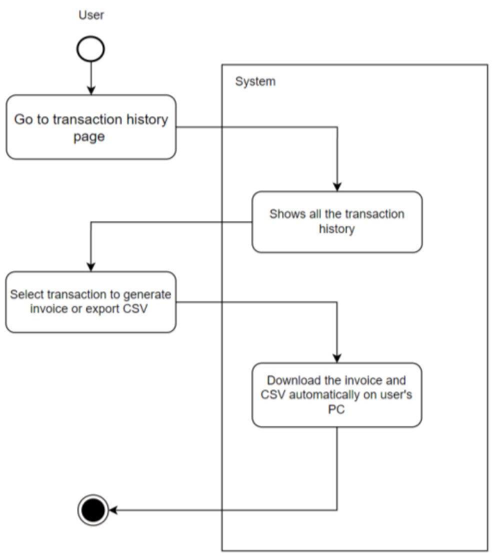

User
System
Go to transaction history page
Shows all the transaction history
Select transaction to generate invoice or export CSV
Download the invoice and CSV automatically on user's PC

Figure 5.5: Activity Diagram for View Transaction History

To view the transaction history, the users may go to the transaction history page. The system will then show all the transaction history. After that, the user can select any of the transactions to generate an invoice or export CSV. If the user does so, the system will download the invoice and CSV automatically on their PC.

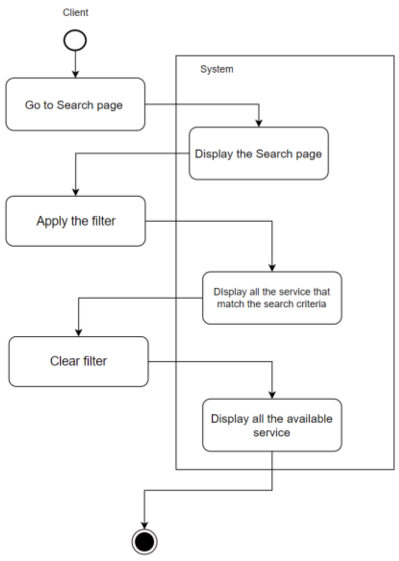

Client
System
Go to Search page
Display the Search page
Apply the filter
Display all the service that match the search criteria
Clear filter
Display all the available service

Figure 5.6: Activity Diagram for Search Care Service

To search for a care service, the client needs to go to the Search page. Then, the system will display the search page with all services. The client can apply the filter to find a specific service. After doing so, the system will display the service that matches the search criteria. If the user clears the filter, the system will then display all available service.

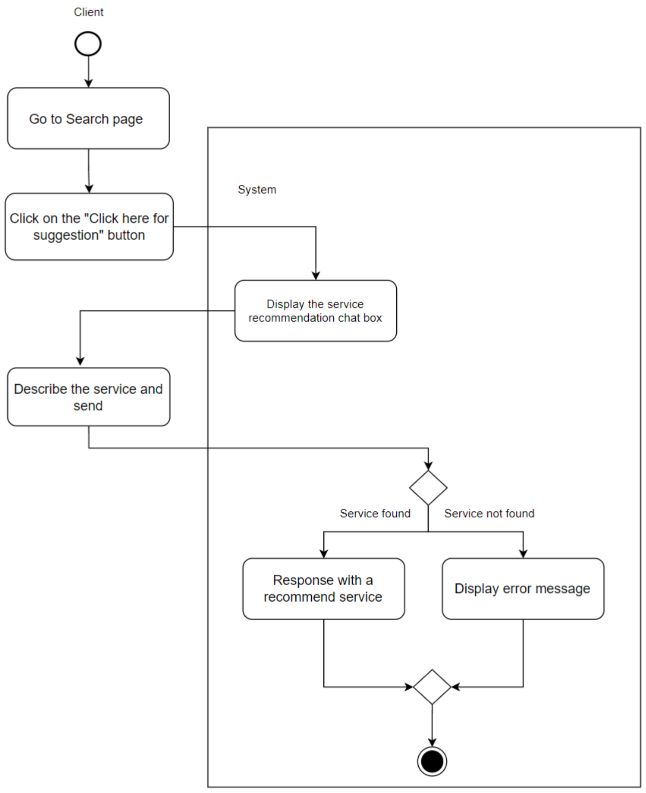

Client
Go to Search page
Click on the "Click here for suggestion" button
System
Display the service recommendation chat box
Describe the service and send
Service found
Service not found
Response with a recommend service
Display error message

Figure 5.7: Activity Diagram for Get Service Recommendation

To get service recommendations, customers need to go to the Search page. They can click on the "Click here to get recommendations" button. After clicking on this button, the system will display the service recommendation chat box so that the customer can describe the desired service and send it. After sending the message, the system will analyze the input and respond with a recommended service if a service is found, and if not, an error message will be displayed.

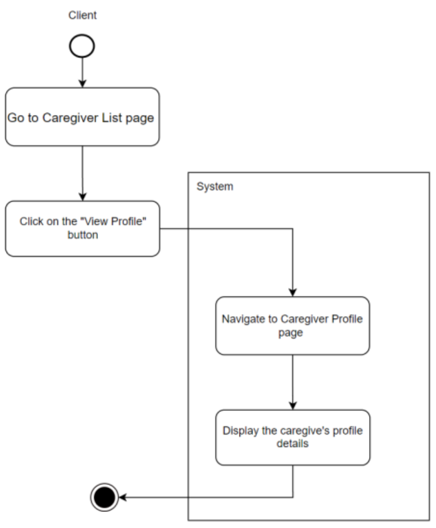

Client
Go to Caregiver List page
Click on the "View Profile" button
System
Navigate to Caregiver Profile page
Display the caregive's profile details

Figure 5.8: Activity Diagram for View Caregiver’s Profile

To view the caregiver's profile, the client needs to go to the caregiver list page and click on the "View Profile" button. After clicking on this button, the system will navigate to the caregiver's profile page and display the caregiver's profile details.

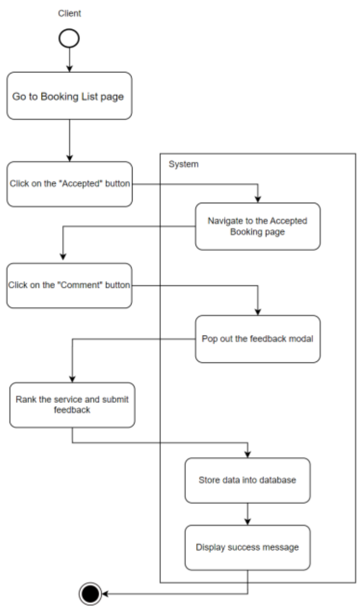

Client
Go to Booking List page
Click on the "Accepted" button
System
Navigate to the Accepted Booking page
Click on the "Comment" button
Pop out the feedback modal
Rank the service and submit feedback
Store data into database
Display success message

Figure 5.9: Activity Diagram for Provide Feedback

To provide feedback, the client needs to go to the Booking List page and click on the "Accepted" button. The system will navigate to the Accepted Booking page. On this page, the customer can click on the "Review" button. After clicking on this button, the system will pop up the feedback modal. The customer can then rank the service and submit feedback. The system will store the data into the database and display a success message.

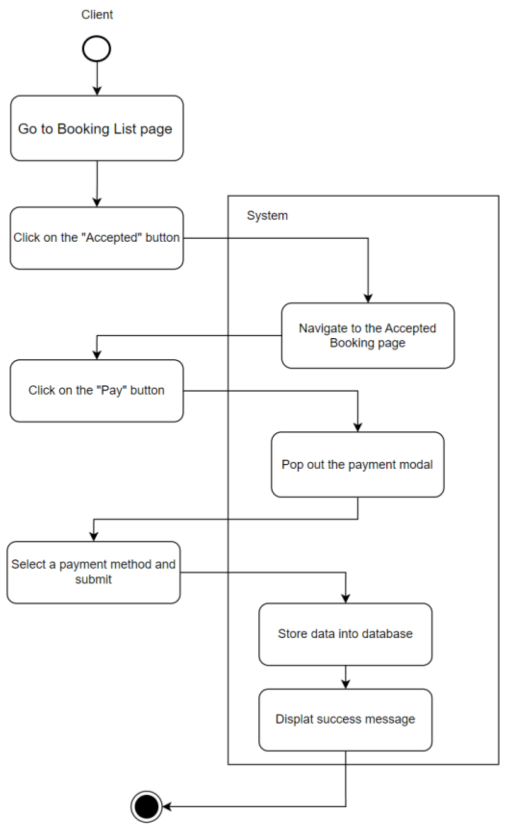

Client
Go to Booking List page
Click on the "Accepted" button
System
Navigate to the Accepted Booking page
Click on the "Pay" button
Pop out the payment modal
Select a payment method and submit
Store data into database
Displat success message

Figure 5.10: Activity Diagram for Make Payment

To make a payment, the customer needs to go to the booking Listing page and click on the "Accepted" button. The system will navigate to the Accepted Booking page. On this page, the customer can click on the "Pay" button. After clicking on this button, the system will pop up the payment modal. The customer can then select a payment method and submit. The system will store the data into the database and display a success message.

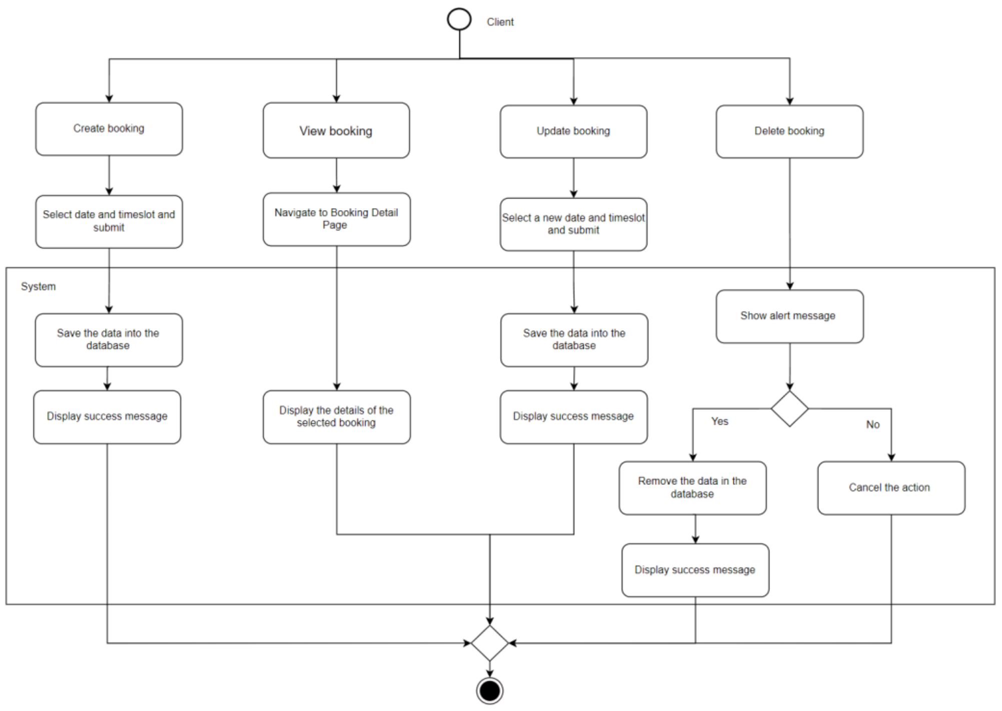

Client
Create booking
View booking
Update booking
Delete booking
Select date and timeslot and submit
Navigate to Booking Detail Page
Select a new date and timeslot and submit
System
Save the data into the database
Show alert message
Save the data into the database
Display success message
Display the details of the selected booking
Display success message
Yes
No
Remove the data in the database
Cancel the action
Display success message

Figure 5.11: Activity Diagram for Manage Booking

When a client creates a booking, they need to select a date and time slot and submit it. After submitting the form, the system saves the data to the database and displays a success message. To view the booking, the client may need to navigate to the Booking Details page. This page will display the details of the selected booking. To update the booking, the client can select a new date and time slot and submit it. After that, the system saves the data to the database and displays a success message. When a customer deletes a booking, the system displays an alert message. If the response is confirmed to delete, the system will delete the data from the database and display a success message, otherwise, the operation will be canceled.

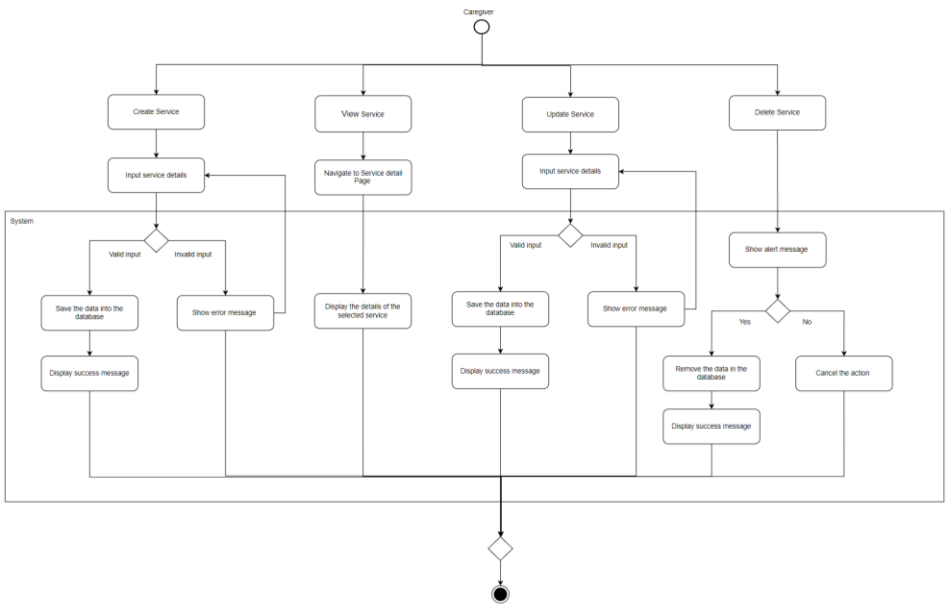


Figure 5.12: Activity Diagram for Manage Care Service

When a caregiver creates a service, they need to enter the service details. If the data entered has invalid input, the system displays an error message, otherwise, the system saves the data to the database and displays a success message. To view a service, the caregiver may need to navigate to the Service Details page. This page will display the details of the selected service. To update a service, the caregiver can enter new service details. If the data entered has invalid input, the system displays an error message, otherwise, the system saves the data to the database and displays a success message. When a caregiver deletes a service, the system displays an alert message. If the deletion is confirmed, the system deletes the data from the database and displays a success message, otherwise, the operation is cancelled.

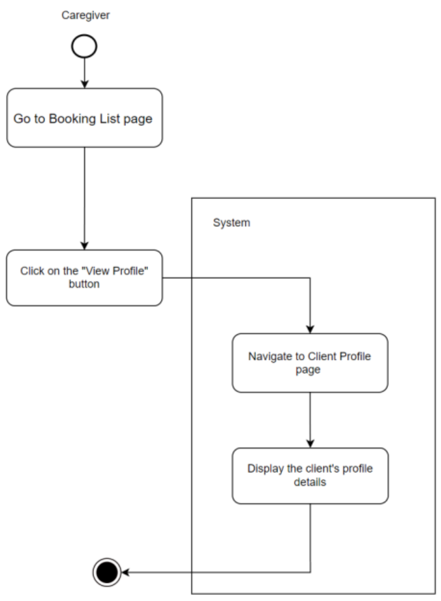

Caregiver
Go to Booking List page
System
Click on the "View Profile" button
Navigate to Client Profile page
Display the client's profile details

Figure 5.13: Activity Diagram for View Client’s Profile

To view the client's profile, the client needs to go to the Booking List page and click on the "View Profile" button. After clicking on this button, the system will navigate to the client's profile page and display the client’s profile details.

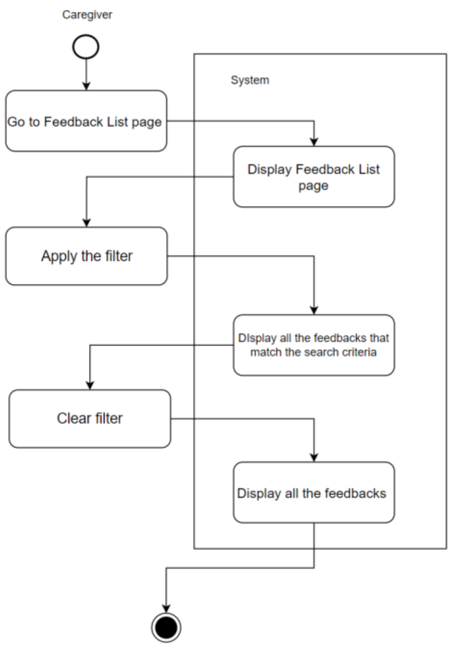

Caregiver
System
Go to Feedback List page
Display Feedback List page
Apply the filter
Display all the feedbacks that match the search criteria
Clear filter
Display all the feedbacks

Figure 5.14: Activity Diagram for View Feedback

To view the feedback, the caregiver needs to go to the Feedback List page. Then, the system will display the Feedback List page with all feedback. The client can apply the filter to find specific feedback. After doing so, the system will display all the feedback that match the search criteria. If the caregiver clears the filter, the system will then display all feedback again.

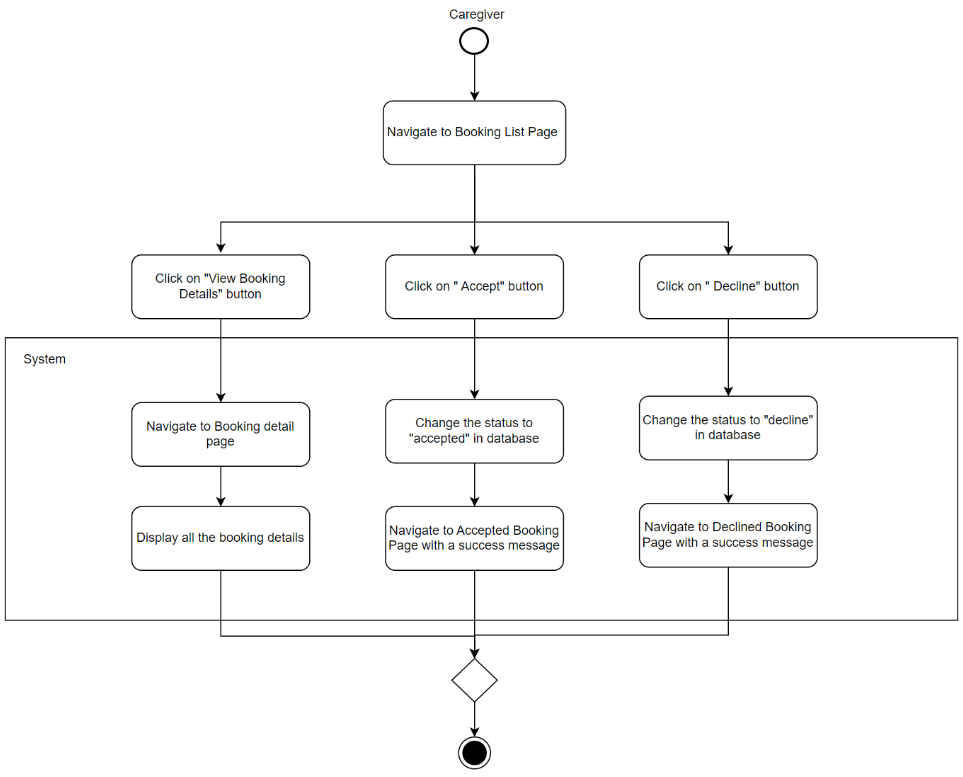

Caregiver
Navigate to Booking List Page
Click on "View Booking Details" button
Click on " Accept" button
Click on " Decline" button
System
Navigate to Booking detail page
Change the status to "accepted" in database
Change the status to "decline" in database
Display all the booking details
Navigate to Accepted Booking Page with a success message
Navigate to Declined Booking Page with a success message

Figure 5.15: Activity Diagram for View Booking

Caregiver first navigates to the Booking List page. To view booking details, the caregiver can click on the “View Booking Details” button. After this, the system will navigate to the booking detail page and display all the booking details. To accept a booking, the caregiver can click on the “Accept” button. Once this button is clicked, the system will change the status of the booking to “accepted” in the database and navigate to the Accepted Booking page with a success message. To decline a booking, the caregiver can click on the “Decline” button. Once this button is clicked, the system will change the status of the booking to “declined” in the database and navigate to the Declined Booking page with a success message.

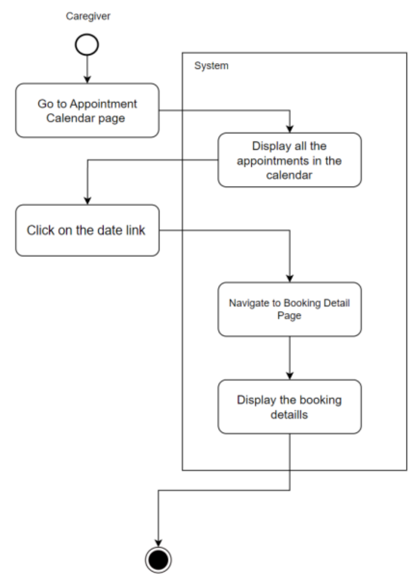

Caregiver
System
Go to Appointment Calendar page
Display all the appointments in the calendar
Click on the date link
Navigate to Booking Detail Page
Display the booking detaills

Figure 5.16: Activity Diagram for View Appointment

To view the appointment, the caregiver will need to navigate to the Appointment Calendar page. The system will then display all the appointments in the calendar. The caregiver can click on the date link in the calendar. After clicking on it, the system will navigate to the Booking Detail page and display the booking details of the booking.

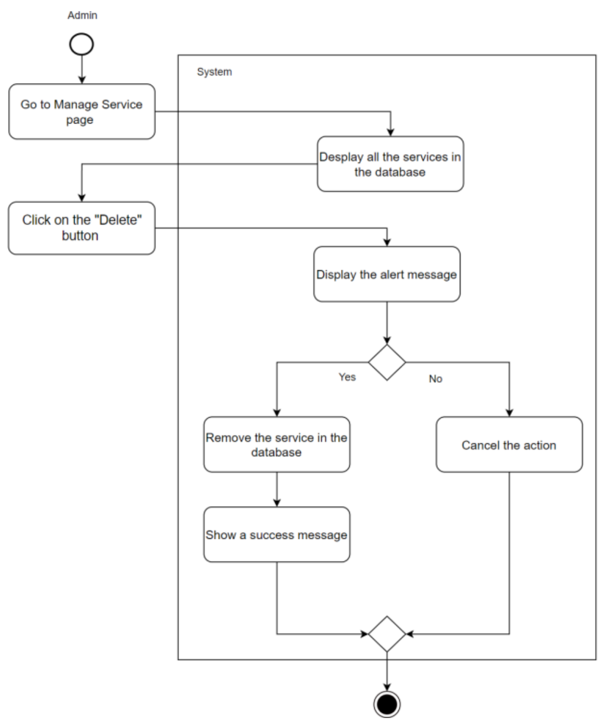

Admin
System
Go to Manage Service page
Desplay all the services in the database
Click on the "Delete" button
Display the alert message
Yes
No
Remove the service in the database
Cancel the action
Show a success message

Figure 5.17: Activity Diagram for Delete Service

To delete a service, the admin needs to go to the Manage Service page. The system will then display all the services in the database. Admin can select a service and click on the "Delete" button. Once the button is clicked, the system will display the alert message. If the response is confirmed to delete, the system will remove the service from the database and show a success message. Otherwise, the action will be cancelled.

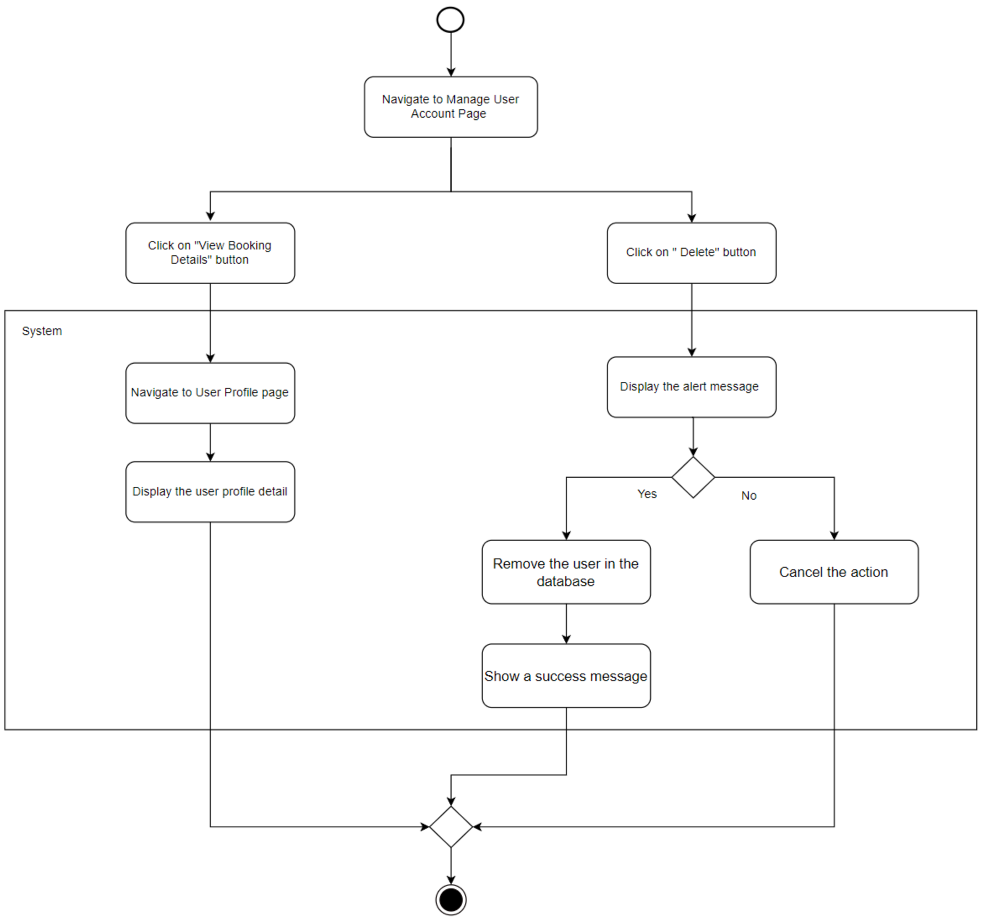

Navigate to Manage User Account Page
Click on "View Booking Details" button
Click on " Delete" button
System
Navigate to User Profile page
Display the user profile detail
Display the alert message
Yes
No
Remove the user in the database
Cancel the action
Show a success message

Figure 5.18: Activity Diagram for Manage Account

To manage the account, the admin may navigate to the Manage User Account Page. On this page, the admin can click on the "View Booking Details" button. Once the button is clicked, the system will navigate to the User Profile page and display the user profile details. Admin may also click on the "Delete" button on the Manage User Account Page. Once the button is clicked, the system will display an alert message, if the response is confirmed to delete, the system will remove the user from the database and show a success message. Otherwise, the action will be cancelled.

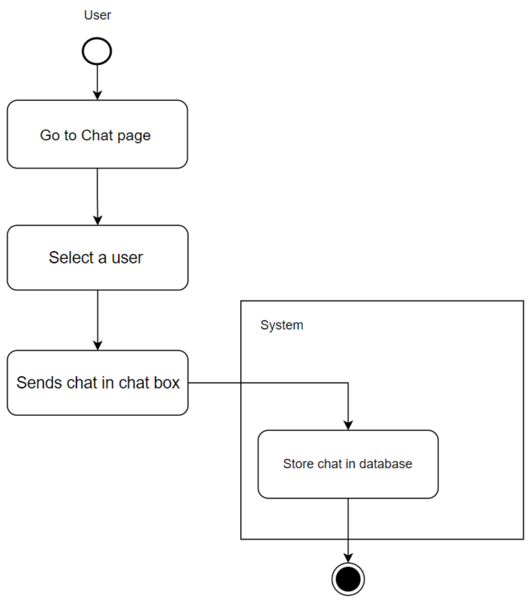


Figure 5.19: Activity Diagram for Chat in Chat Box

To chat in the chat box, the user may go to the Chat page and select a user to send a chat in the chat box. After that, the chat data will be stored in the database.

5.4 Database Design

5.4.1 Entity Relationship Diagram (ERD)

The Entity Relationship Diagram below represents an Elderly Care and Assistance Booking Platform that consists of several entities, including users, bookings, services, service_dates, service_timeslots, notifications, ch_favorites and ch_messages. Each entity has its own attributes and relationships.

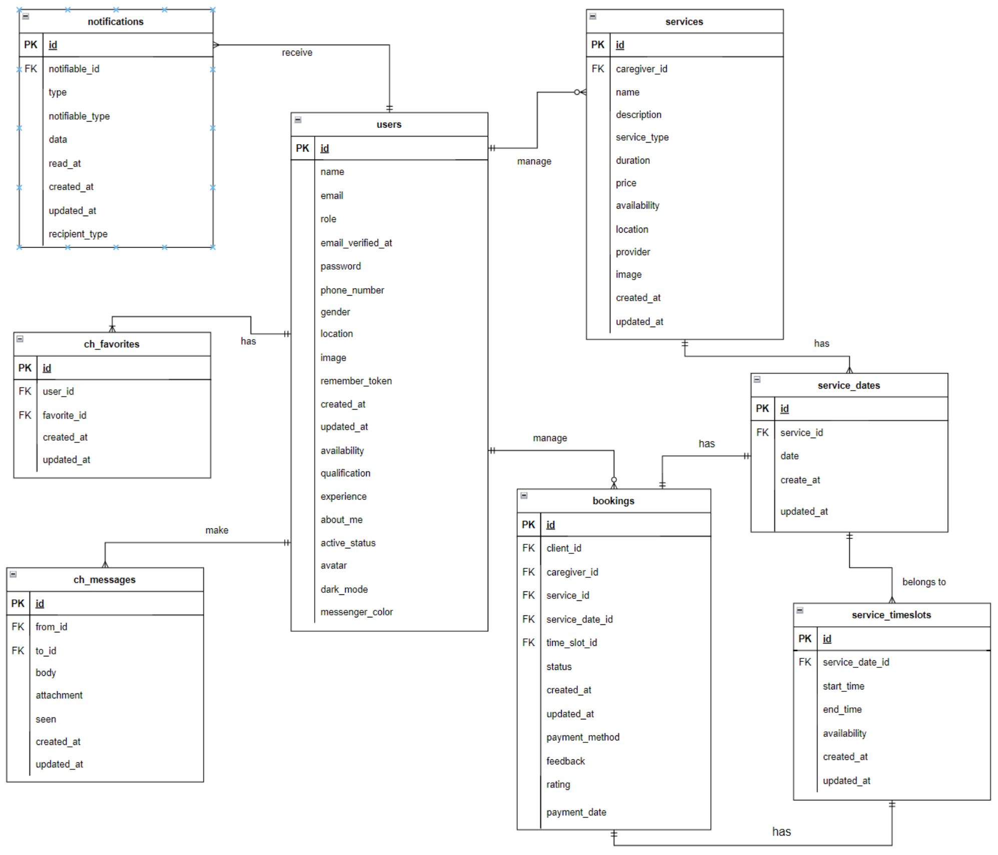


Figure 5.20: Entity Relationship Diagram

The Users and Services entities have a one-to-zero/many relationship. This means the user can manage no or many services. This is because only caregivers are allowed to manage services while other users cannot.

The Services and Service Dates (service_dates) entities have a one-to-many relationship. This means a service can have many associated service dates.

The Service Dates and Service Time Slots (service_timeslots) entities also have a one-to-many relationship. This means multiple time slots can belong to a single service date.

For Users and Bookings entities, there is a one-to-zero/many relationship. This means a user can manage no bookings or many bookings, but each booking is linked to only one user. This is because only client can manage the bookings while other users cannot.

The Bookings and Service Dates (service_dates) entities have a one-to-one relationship. This means each booking is associated with exactly one service date. Similarly, the Bookings and Service Time Slots (service_timeslots) entities have a one-to-one relationship. This means each booking is linked to one time slot.

Additionally, the Users and Notifications entities have a one-to-many relationship. This means a user can receive many notifications.

The Users and Chat Favorites (ch_favorites) entities also have a one-to-many relationship. This means a user can have multiple chat favorites.

Finally, the Users and Chat Messages (ch_messages) entities have a one-to-many relationship. This means a user can send and receive many chat messages.

5.4.2 Data Dictionary

Table 5.1: Data dictionary for the table "bookings"

Field Name	Data Type	Field Length	Constraint	Description
id	int	-	PRIMARY KEY	Unique ID of booking
client_id	int	-	FK, NOT NULL	ID of client
caregiver_id	int	-	FK, NOT NULL	ID of caregiver
service_id	int	-	FK, NOT NULL	ID of service
service_date_id	int	-	FK, NOT NULL	ID of service date
time_slot_id	int	-	FK, NOT NULL	ID of time slot

status	enum	-	-	Status of booking
created_at	timestamp	-	-	Creation timestamp
updated_at	timestamp	-	-	Updated timestamp
payment_method	varchar	191	-	Payment method for a booking
feedback	text	-	-	Feedback on a booking
rating	int	-	-	Rating of a booking
payment_date	timestamp	-	-	Payment date of a booking

Table 5.2: Data dictionary for the table "ch_favorites"

Field Name	Data Type	Field Length	Constraint	Description
id	int	-	PRIMARY KEY	Unique ID of chat favorite
user_id	int	-	FK, NOT NULL	ID of user who added the favorite
favorite_id	int	-	FK, NOT NULL	ID of user marked as a favorite
created_at	timestamp	-	-	Creation timestamp
updated_at	timestamp	-	-	Updated timestamp

Table 5.3: Data dictionary for the table “ch_messages”

Field Name	Data Type	Field Length	Constraint	Description
id	int	-	PRIMARY KEY	Unique ID of chat message
from_id	int	-	FK, NOT NULL	ID of user who sent the message
to_id	int	-	FK, NOT NULL	ID of user received the message
body	varchar	5000	-	Content of the message
attachment	varchar	191	-	Attachment associated with the message
seen	boolean	-	-	Indicates if the message has been seen
created_at	timestamp	-	-	Creation timestamp
updated_at	timestamp	-	-	Updated timestamp

Table 5.4: Data dictionary for the table “notifications”

Field Name	Data Type	Field Length	Constraint	Description
id	int	-	PRIMARY KEY	Unique ID of notification

notifiable_id	int	-	FK, NOT NULL	ID of related entity
type	varchar	191	NOT NULL	Type of the notification
notifiable_type	int	-	NOT NULL	Type of related entity
data	text	5000	NOT NULL	Content of the notification
read_at	timestamp	191	-	Timestamp when read
created_at	timestamp	-	-	Creation timestamp
updated_at	timestamp	-	-	Updated timestamp

Table 5.5: Data dictionary for the table “services”

Field Name	Data Type	Field Length	Constraint	Description
id	int	-	PRIMARY KEY	Unique ID of service
caregiver_id	int	-	FK, NOT NULL	ID of caregiver
name	varchar	191	NOT NULL	Name of the service
description	text	-	NOT NULL	Description of service
service_type	varchar	191	NOT NULL	Type of service
duration	varchar	191	NOT NULL	Duration of service

price	decimal	-	NOT NULL	Price of service
availability	varchar	191	NOT NULL	Availability of service
location	varchar	191	NOT NULL	Location where the service provider
image	varchar	191	-	Image of service
created_at	timestamp	-	-	Creation timestamp
updated_at	timestamp	-	-	Updated timestamp

Table 5.6: Data dictionary for the table “services_dates”

Field Name	Data Type	Field Length	Constraint	Description
id	int	-	PRIMARY KEY	Unique ID of service date
service_id	int	-	FK, NOT NULL	ID of service
date	int	-	NOT NULL	Date of the service
created_at	timestamp	-	-	Creation timestamp
updated_at	timestamp	-	-	Updated timestamp

Table 5.7: Data dictionary for the table “service_timeslots”

Field Name	Data Type	Field Length	Constraint	Description

id	int	-	PRIMARY KEY	Unique ID of service time slot
service_date_id	int	-	FK, NOT NULL	ID of service date
start_time	time	-	NOT NULL	Start time of the service
end_time	time	-	NOT NULL	End time of the service
availability	varchar	191		Availability of the service
created_at	timestamp	-	-	Creation timestamp
updated_at	timestamp	-	-	Updated timestamp

Table 5.8: Data dictionary for the table "users"

Field Name	Data Type	Field Length	Constraint	Description
id	int	-	PRIMARY KEY	Unique ID of user
name	varchar	191	NOT NULL	User's name
email	varchar	191	NOT NULL	User's email
role	varchar	191	NOT NULL	User's role
email_verified_at	timestamp	-	-	Timestamp when email was verified
password	varchar	191	NOT NULL	User's hashed password

phone_number	varchar	191	-	User’s phone number
gender	enum	-	-	User’s gender
loacation	varchar	191	-	Users’ address
image	varchar	191	-	User’s profile image
remember_token	varchar	100	-	Token for "remember me" functionality
created_at	timestamp	-	-	Creation timestamp
updated_at	timestamp	-	-	Updated timestamp
availability	varchar	191	-	Payment date of a booking
qualification	varchar	191	-	User’s qualification
experience	text	-	-	User’s experience details
about_me	text	-	-	User’s personal description
active_status	boolean	-	NOT NULL	Indicates if the user is active

avatar	varchar	191	NOT NULL	Avatar image URL
dark_mode	boolean	-	NOT NULL	Indicates if dark mode is enabled
messenger_color	varchar	191	-	User's preferred messenger color

CHAPTER 6
SYSTEM IMPLEMENTATION

6.1 Introduction

This chapter will focus on the system implementation details of this project. This chapter is divided into four sections. Section 5.2 outlines the basic software tools required to develop this project and the settings of this software. Section 5.3 will explain how to configure the necessary settings in the "env" file to ensure that the platform has the appropriate environment to run properly. Section 5.4 will provide a visual demonstration of the main functions of the platform. Section 5.5 will express the challenges I encountered in the process of developing the elderly care and assistance booking platform.

6.2 Software Setup

In this project, three essential software applications need to be installed to develop the Elderly Care and Assistance Booking Platform. The software applications are Visual Studio Code (VS Code), WampServer, and Node.js. These tools provide the necessary environment for coding, testing, and running the platform locally before deploying it to a live server.

6.2.1 Visual Studio Code

Visual Studio Code will be used as the primary code editor during the entire development process. It supports multiple programming languages which include PHP, JavaScript, HTML, and CSS, which are crucial for this project. Besides that, the built-in terminal in VS codes allows developers to run commands and scripts directly within the editor. Last, VSCode integrates seamlessly with GitHub. This enables developers to perform version control operations efficiently. Developers commit, push, pull, and manage branches directly from the editor. This is the link for VSCode installation: https://code.visualstudio.com/download.

6.2.2 WampServer

WampServer is used to create a local server environment which is essential to the development of this platform. Since web development requires Apache, MySQL, and PHP, WampServer comes with a complete configuration of these. To make managing the MySQL database simple, WampServer also comes with phpMyAdmin. Before installing the platform, this configuration enables effective development and testing on your local computer. Download links for WampServer are available at https://wampserver.aviatechno.net/.

6.2.3 Node.js

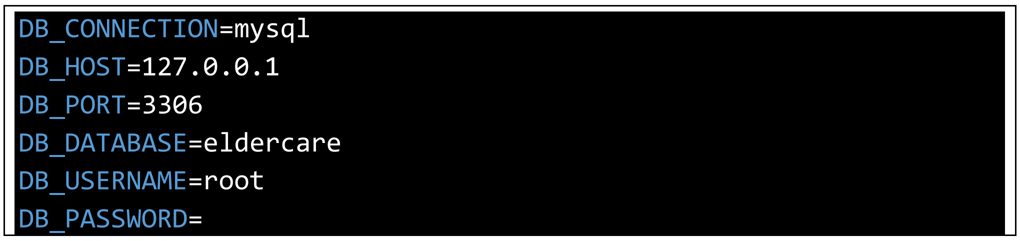
Node.js will be used to handle the platform's service recommendation functionality. It allows server-side applications to provide recommendations based on client requests. This is the link for Node.js installation: https://nodejs.org/en/download/.

6.3 Setting and Configuration

To ensure that the platform operates correctly, the specific environment settings need to be configured correctly in the '. env' file.

6.3.1 Database Configuration

Table 6.1: Database Configuration



```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=eldercare
DB_USERNAME=root
DB_PASSWORD=
```

With these configurations, the program is guaranteed to establish a connection with the local MySQL server on the same system. Replace "DB_DATABASE" with the name of the MySQL database. The database credentials were left as default for development purposes.

6.3.2 Mail Server Configuration

Table 6.2: Mail Server Configuration

```
MAIL_MAILER=smtp
MAIL_HOST=smtp.gmail.com
MAIL_PORT=587
MAIL_USERNAME=zixuan@1utar.my
MAIL_PASSWORD=qxyecvkpngqeaytr
MAIL_ENCRYPTION=tls
MAIL_FROM_ADDRESS=zixuan@1utar.my
MAIL_FROM_NAME="${APP_NAME}"
```

After an account is registered, this platform will utilize the mail server to send an email for email verification. It will also use the mail server to send a link for password resets. The mail server must be configured to allow Gmail's SMTP server with TLS encryption for the platform to deliver emails safely. The sender's name and email address are indicated by Mail_FROM_NAME and Mail_FROM_ADDRESS respectively.

6.3.3 Pusher API Configuration

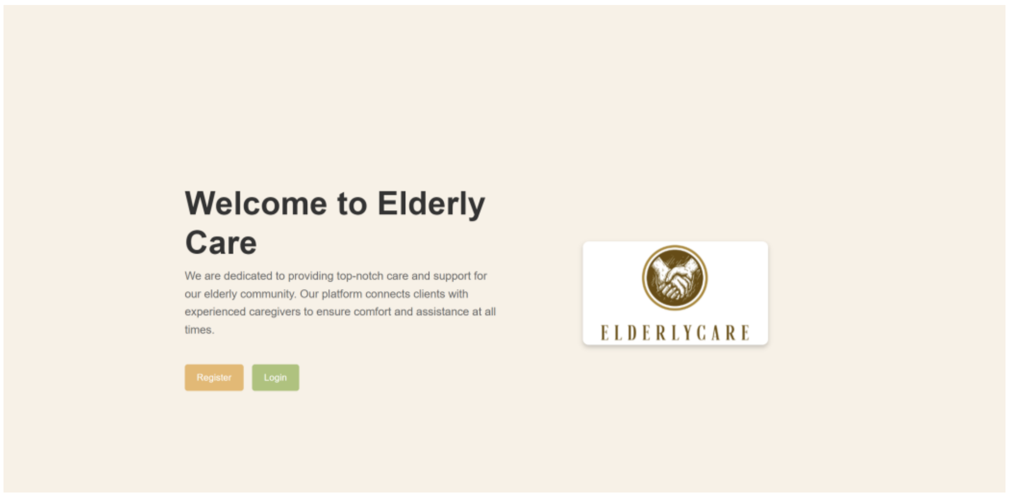
Table 6.3: Pusher API Configuration

```
PUSHER_APP_ID=1835973
PUSHER_APP_KEY=00eca1ea162231d33ffe
PUSHER_APP_SECRET=b96de7f5bdee1a63777a
PUSHER_APP_CLUSTER=ap1
```

To enable real-time chat functionality with Chatify, the PUSHER API should be configured in the '. env' file. After configuring this, Chatify can use Pusher for instant messaging and notifications.

6.4 System Operation with Screenshots

6.4.1 Home Page


Welcome to Elderly Care

We are dedicated to providing top-notch care and support for our elderly community. Our platform connects clients with experienced caregivers to ensure comfort and assistance at all times.

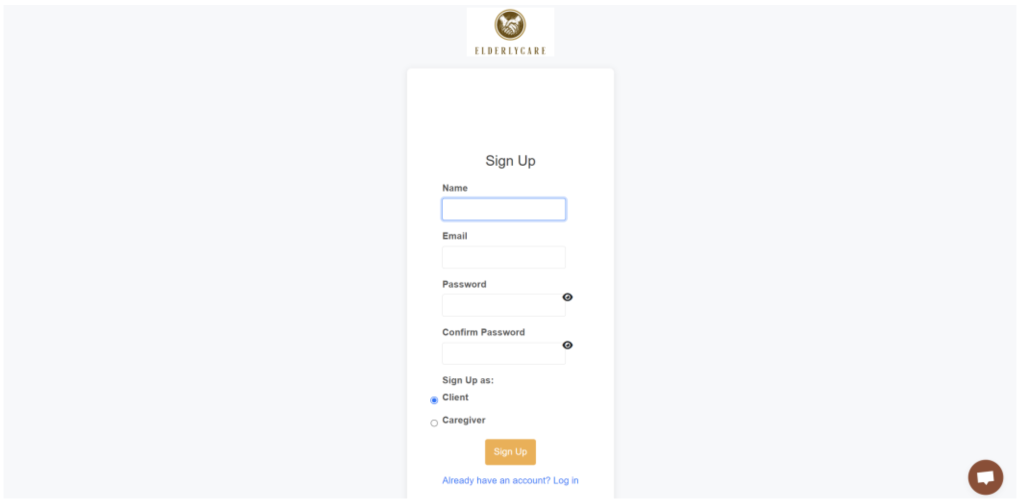
Register Login

ELDERLYCARE

Figure 6.1: Home Page

Users will first be redirected to the home page when they go to this link: http://127.0.0.1:8000.

6.4.2 Sign Up Page


ELDERLYCARE

Sign Up

Name

Email

Password

Confirm Password

Sign Up as:

● Client

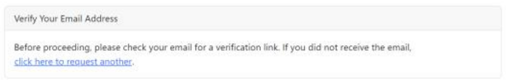
○ Caregiver

Sign Up

Already have an account? Log in

Figure 6.2: Sign Up page

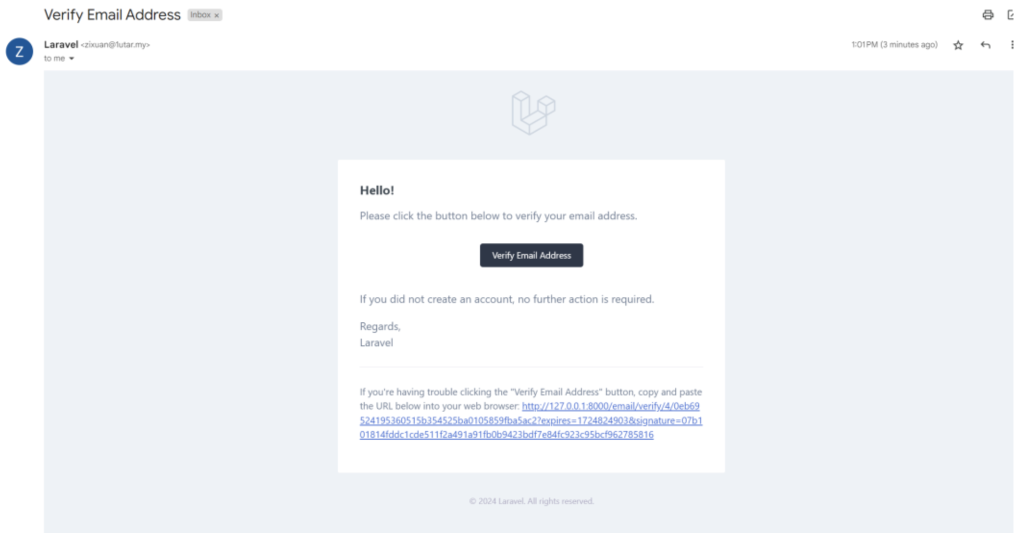
In the Sign Up page, users will need to enter their information, including name, email, and password, and whether they want to register as a client or caregiver. The user may click on the eye icon to view their password for confirmation of the correct password.


Verify Your Email Address

Before proceeding, please check your email for a verification link. If you did not receive the email, click here to request another.

Figure 6.3: Email Verification Message

After registering an account, a verification link will be sent to the registered email, and a message will be shown to notify the users.


Verify Email Address Inbox ×

Laravel <zixuan@1utar.my>
to me

1:01PM (3 minutes ago)

Hello!

Please click the button below to verify your email address.

If you did not create an account, no further action is required.

Regards,
Laravel

If you're having trouble clicking the "Verify Email Address" button, copy and paste the URL below into your web browser: http://127.0.0.1:8000/email/verify/4/0eb69524195360515b354525ba0105859fba5ac2?expires=1724824903&signature=07b101814fddc1cde511f2a491a91fb0b9423bdf7e84fc923c95bcf962785816

© 2024 Laravel. All rights reserved.

Figure 6.4: Email of the Email Verification

The figure above shows the email of the email verification. User may click on the verify link in the email to verify their email.

6.4.3 Login Page

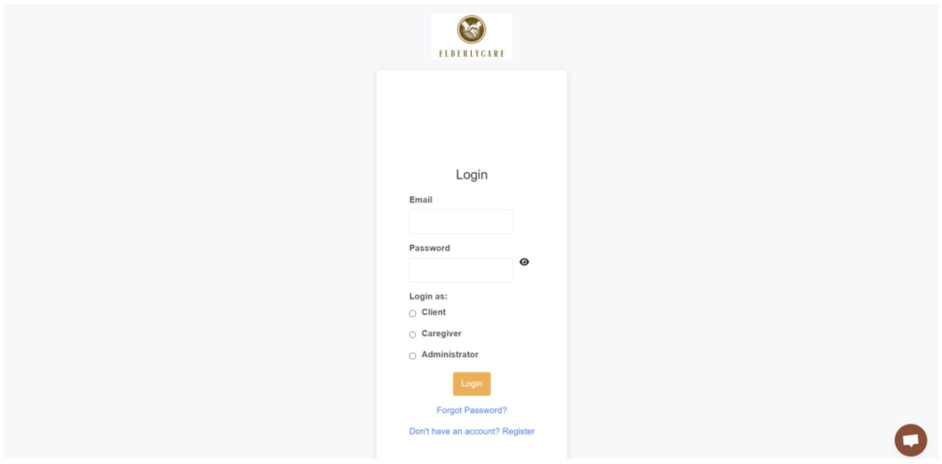


Figure 6.5: Login Page

After verifying the email address, the user can now login to the platform. The user is requiring to enter their email, password, and their role to login. The user may also click on the eye icon to view their password for confirmation of the correct password.

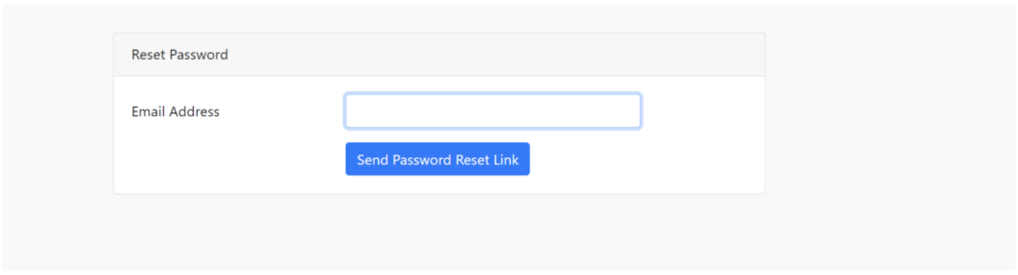
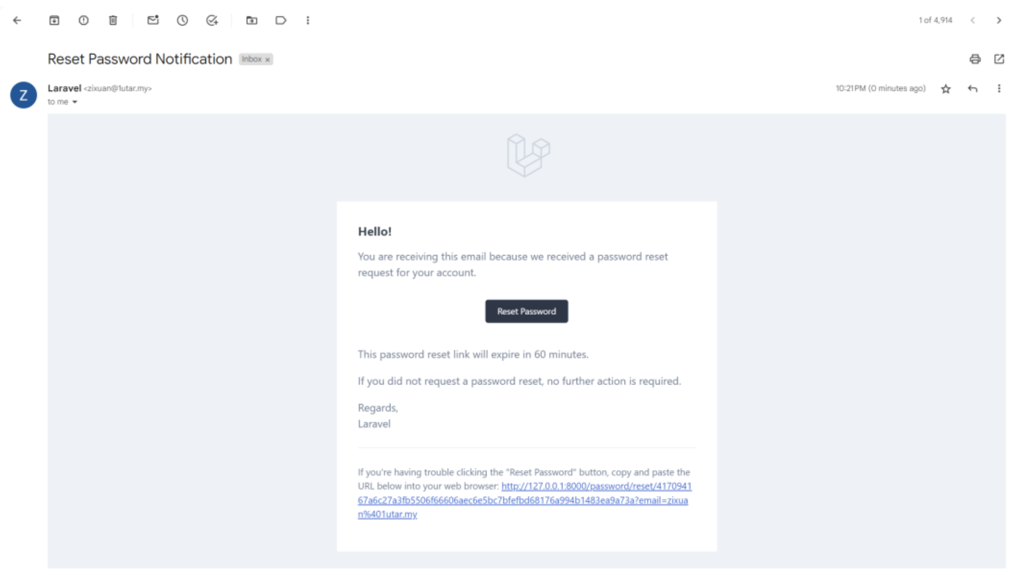


Figure 6.6: Send Password Reset Page

If the user forgets their password, they may click the "Forgot Password?" link on the login page, and then they will be redirected to this Reset Password Page. To get the password reset link, the user must enter their email address again and click the blue button. This reset password link will be sent to the email address that the user entered.



Reset Password Notification

Laravel <zixuan@1utar.my>

Hello!

You are receiving this email because we received a password reset request for your account.

This password reset link will expire in 60 minutes.

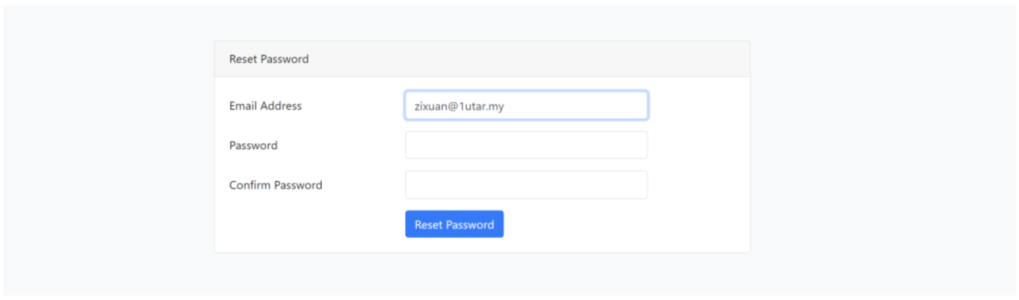
If you did not request a password reset, no further action is required.

Regards,
Laravel

If you're having trouble clicking the "Reset Password" button, copy and paste the URL below into your web browser: http://127.0.0.1:8000/password/reset/417094167a6c27a3fb5506f66606aec6e5bc7bfefbd68176a994b1483ea9a73a?email=zixuan%401utar.my

Figure 6.7: Email of the Reset Password Notification

The figure above shows the email that the user will receive after requesting a password reset link.



Reset Password

Email Address: zixuan@1utar.my

Password:

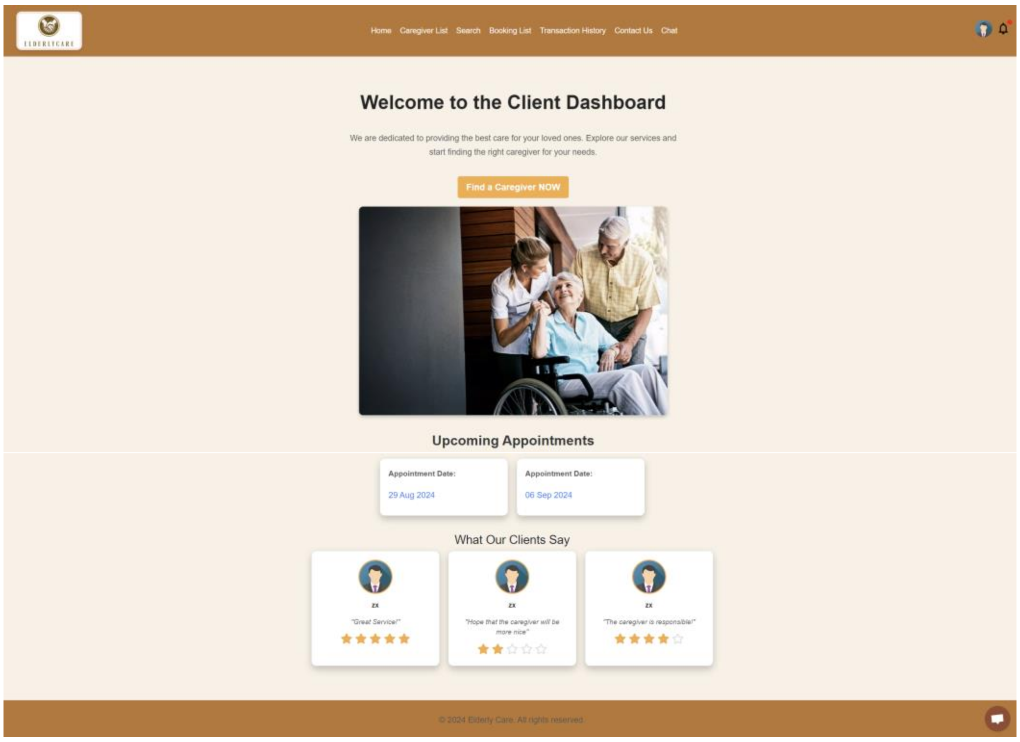
Confirm Password:

Figure 6.8: Reset Password Page

The user will be redirected to this page after clicking on the reset password button in the email. The user can now reset their password on this page by entering their email and the new password.

6.4.4 Client

6.4.4.1 Home Page



Welcome to the Client Dashboard

We are dedicated to providing the best care for your loved ones. Explore our services and start finding the right caregiver for your needs.

Upcoming Appointments

Appointment Date: 29 Aug 2024

Appointment Date: 06 Sep 2024

What Our Clients Say

zx
"Great Service!"
★★★★★

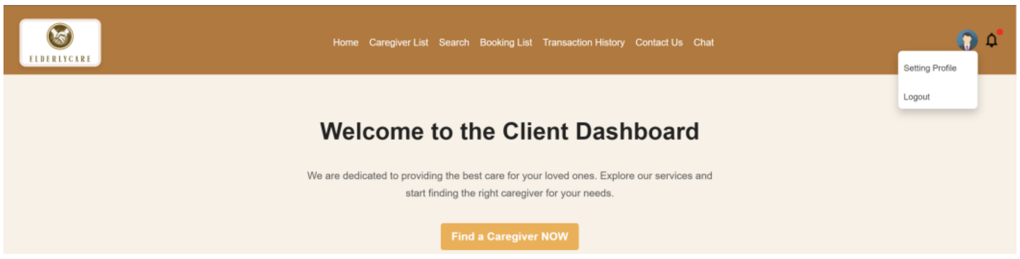
zx
"Hope that the caregiver will be more nice"
★★☆☆☆

zx
"The caregiver is responsible!"
★★★★☆

© 2024 Elderly Care. All rights reserved.

Figure 6.9: Home Page of Client

The client can view the upcoming appointments on this Home Page. Once the client clicks on the date link of the upcoming appointments, they will be redirected to the booking details page. Clients can also view all the feedback and ratings that were provided by other clients on this page.

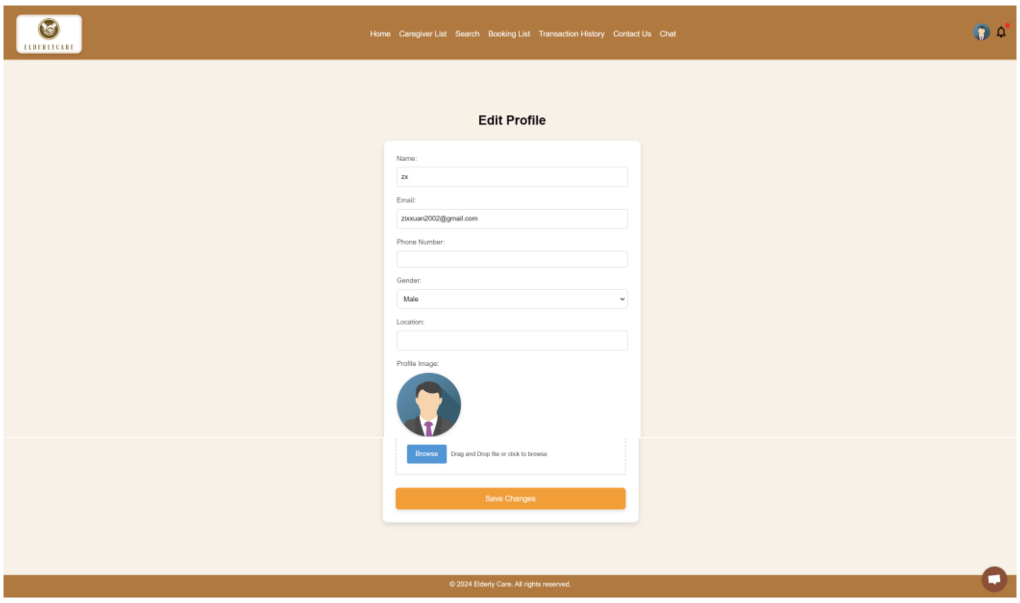


Welcome to the Client Dashboard

We are dedicated to providing the best care for your loved ones. Explore our services and start finding the right caregiver for your needs.

Figure 6.10: Profile Icon

The setting profile and logout dropdown will show when the client clicks on the profile icon on the header.



Edit Profile

Name: zx

Email: zixxuan2002@gmail.com

Phone Number:

Gender: Male

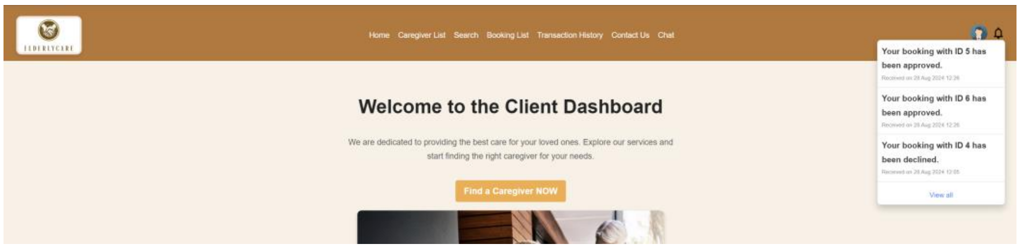
Location:

Profile Image:

© 2024 Elderly Care. All rights reserved.

Figure 6.11: Setting Profile Page

Once the client clicks on the “Setting Profile” on the profile dropdown, they will be redirected to this page. The client can update their profile details on this page by uploading their personal information and image profile.



Welcome to the Client Dashboard

We are dedicated to providing the best care for your loved ones. Explore our services and start finding the right caregiver for your needs.

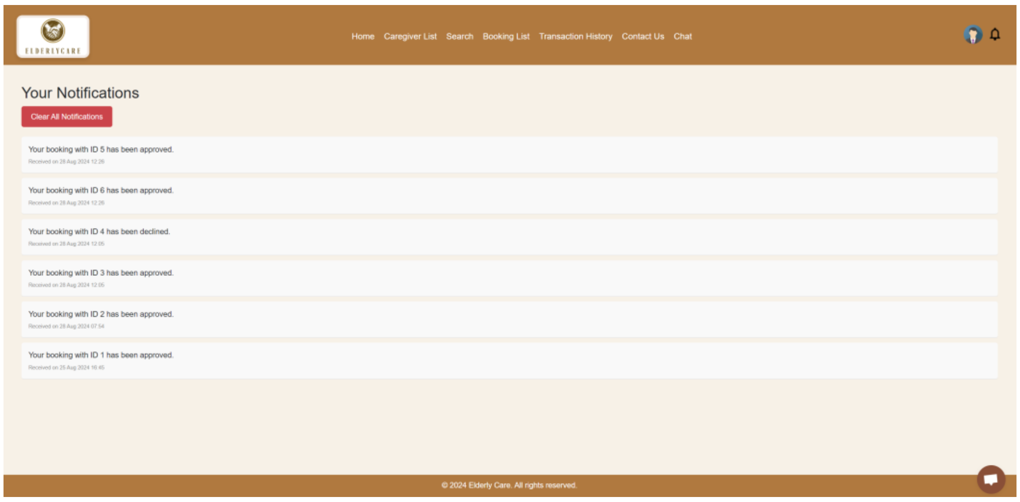
Your booking with ID 5 has been approved.
Received on 28 Aug 2024 12:26

Your booking with ID 6 has been approved.
Received on 28 Aug 2024 12:26

Your booking with ID 4 has been declined.
Received on 28 Aug 2024 12:05

Figure 6.12: Notification dropdown

The client can check for the new notification by clicking on the notification icon on the header. The red dot on the notification icon indicates that there is an unread notification. Once the client clicks on it, the red dot will disappear, indicating that there is no unread notification anymore.



Your Notifications

Your booking with ID 5 has been approved.
Received on 28 Aug 2024 12:26

Your booking with ID 6 has been approved.
Received on 28 Aug 2024 12:26

Your booking with ID 4 has been declined.
Received on 28 Aug 2024 12:05

Your booking with ID 3 has been approved.
Received on 28 Aug 2024 12:05

Your booking with ID 2 has been approved.
Received on 28 Aug 2024 07:54

Your booking with ID 1 has been approved.
Received on 25 Aug 2024 16:45

© 2024 Elderly Care. All rights reserved.

Figure 6.13: Notification Page

When the client clicks on the "view all" blue word on the notification dropdown, they will be redirected to this Notification Page. This page shows all the notifications received by the client. The client may clear all the notifications by clicking on the red clear notification button.

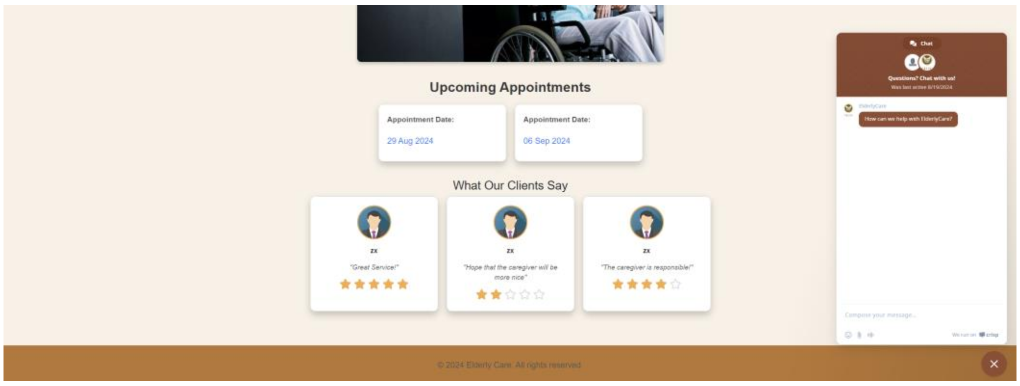


Figure 6.14: Live Chat

The client can use the live chat that lies on the footer to have a chat with the platform assistance directly. This live chat feature allows clients to get immediate assistance without having to wait for email. Therefore, enhances the user experience by resolving issues and answering questions quickly.

6.4.4.2 Caregiver List Page

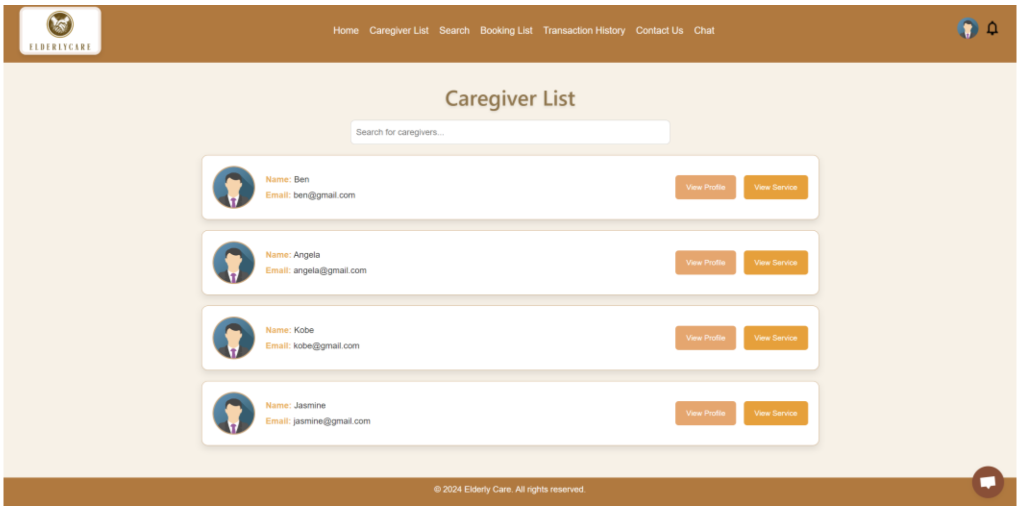
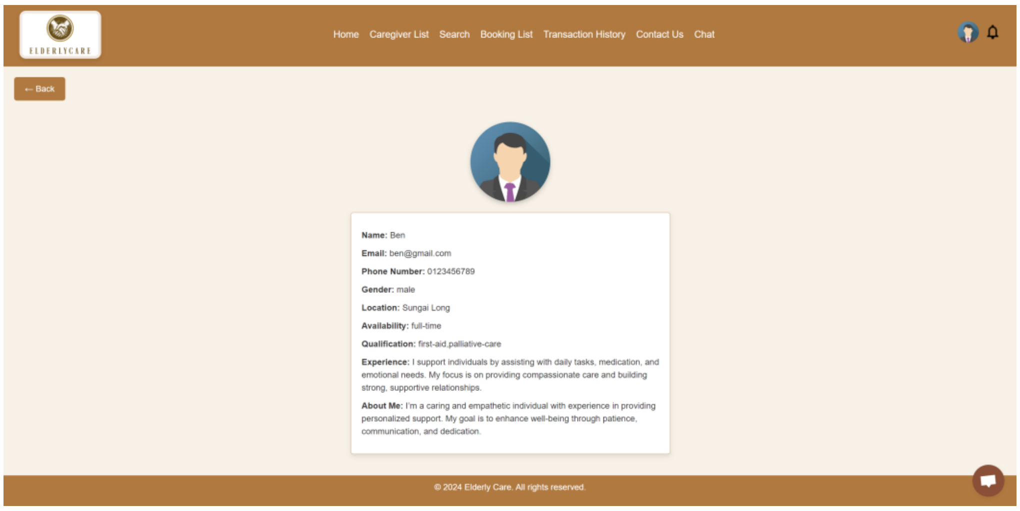


Figure 6.15: Caregiver List Page

The client can view all the caregivers on this Caregiver List page. They can also search for a caregiver by entering the caregiver's name in the search box. As the client types in the search box, the page dynamically updates to show matching caregivers without needing to reload the page by using the AJAX.



Name: Ben

Email: ben@gmail.com

Phone Number: 0123456789

Gender: male

Location: Sungai Long

Availability: full-time

Qualification: first-aid,palliative-care

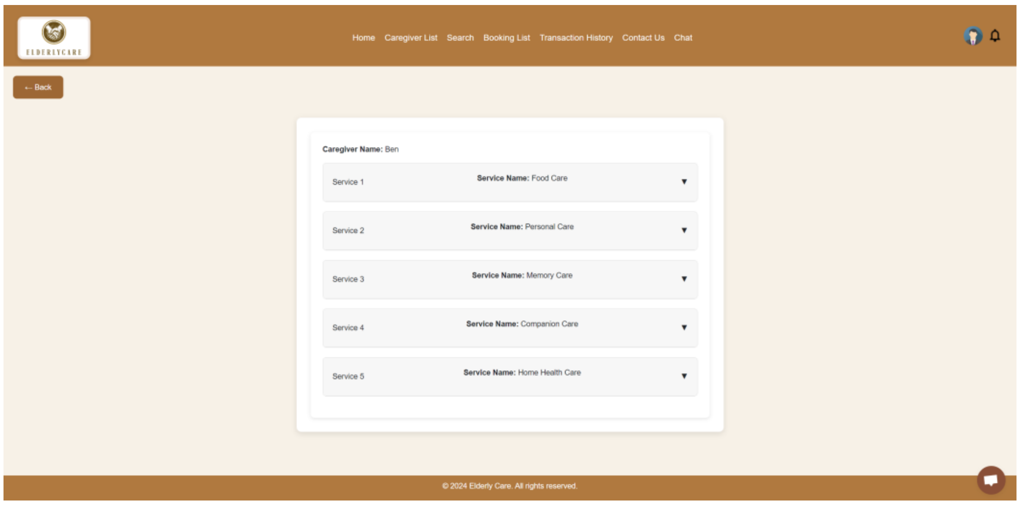
Experience: I support individuals by assisting with daily tasks, medication, and emotional needs. My focus is on providing compassionate care and building strong, supportive relationships.

About Me: I'm a caring and empathetic individual with experience in providing personalized support. My goal is to enhance well-being through patience, communication, and dedication.

© 2024 Elderly Care. All rights reserved.

Figure 6.16: Caregiver’s Profile Page

The client will be navigated to the Caregiver's Profile Page if they click on the “View Profile” button. This page shows the detailed information of the caregiver.



Caregiver Name: Ben

Service 1 **Service Name:** Food Care ▼

Service 2 **Service Name:** Personal Care ▼

Service 3 **Service Name:** Memory Care ▼

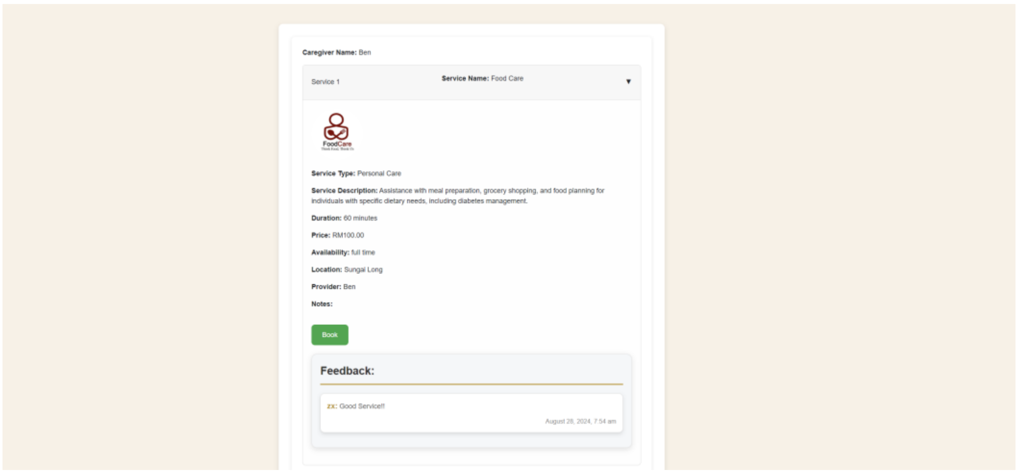
Service 4 **Service Name:** Companion Care ▼

Service 5 **Service Name:** Home Health Care ▼

© 2024 Elderly Care. All rights reserved.

Figure 6.17: Care Service Page

After clicking on the “View Service” button on the Caregiver List Page, the client will be redirected to this Care Service Page. This page shows all the services provided by the selected caregiver.


Caregiver Name: Ben

Service 1 | Service Name: Food Care

Service Type: Personal Care

Service Description: Assistance with meal preparation, grocery shopping, and food planning for individuals with specific dietary needs, including diabetes management.

Duration: 60 minutes

Price: RM100.00

Availability: full time

Location: Sungai Long

Provider: Ben

Notes:

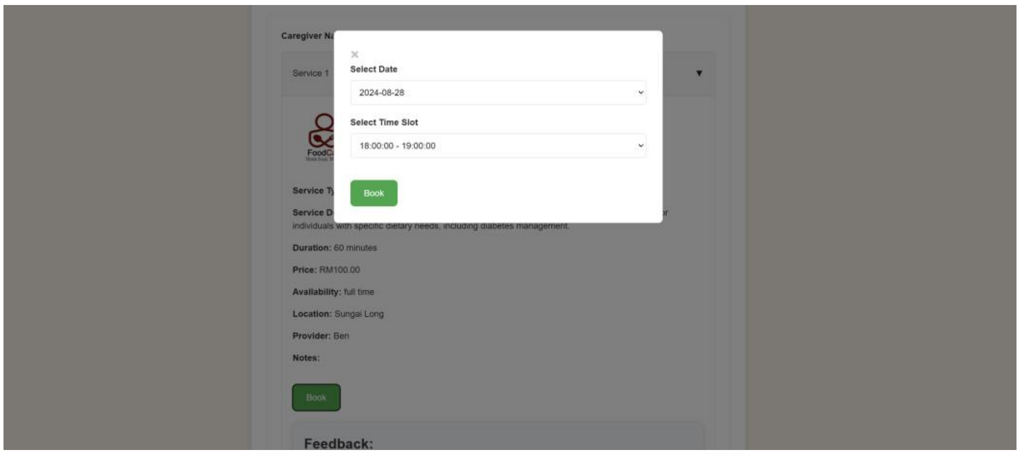
Feedback:

zx: Good Service!!

August 28, 2024, 7:54 am

Figure 6.18: View of Service Details

The service details will be listed when the client expands any of the services. On this page, the client can view the details and the feedback of the service. After reviewing the details and the feedback on the service, they can book the service by clicking on the book button.


Select Date

2024-08-28

Select Time Slot

18:00:00 - 19:00:00

Duration: 60 minutes

Price: RM100.00

Availability: full time

Location: Sungai Long

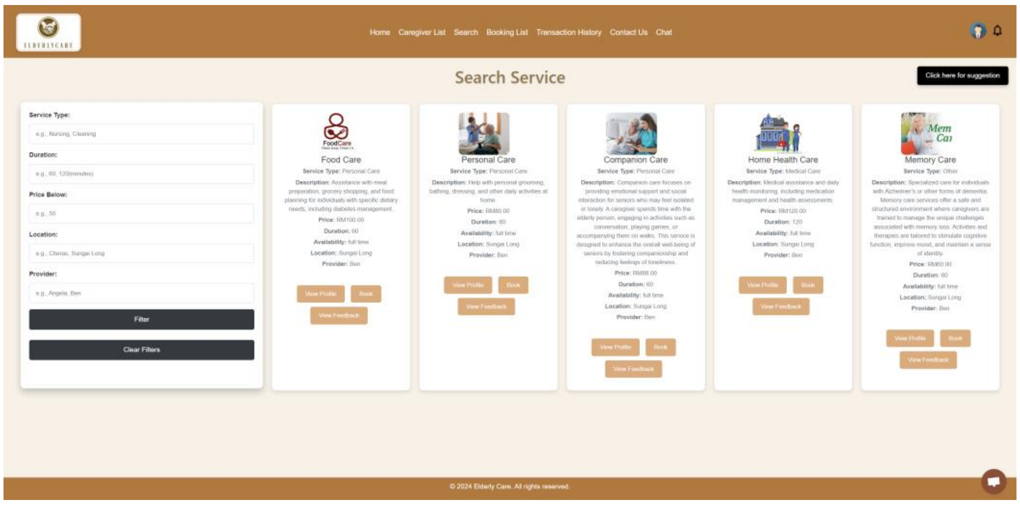
Provider: Ben

Notes:

Figure 6.19: Booking Modal

The booking modal will pop out and ask the client to choose the booking date and timeslot when the client clicks on the "Book" button. After the booking has been made, a notification will be sent to the caregiver to notify them that the service has been booked.

6.4.4.3 Search Page



Food Care
Service Type: Personal Care
Description: Assistance with meal preparation, grocery shopping, and food planning for individuals with specific dietary needs, including diabetes management.
Price: RM100.00
Duration: 60
Availability: full time
Location: Sungai Long
Provider: Ben

Personal Care
Service Type: Personal Care
Description: Help with personal grooming, bathing, dressing, and other daily activities at home
Price: RM80.00
Duration: 60
Availability: full time
Location: Sungai Long
Provider: Ben

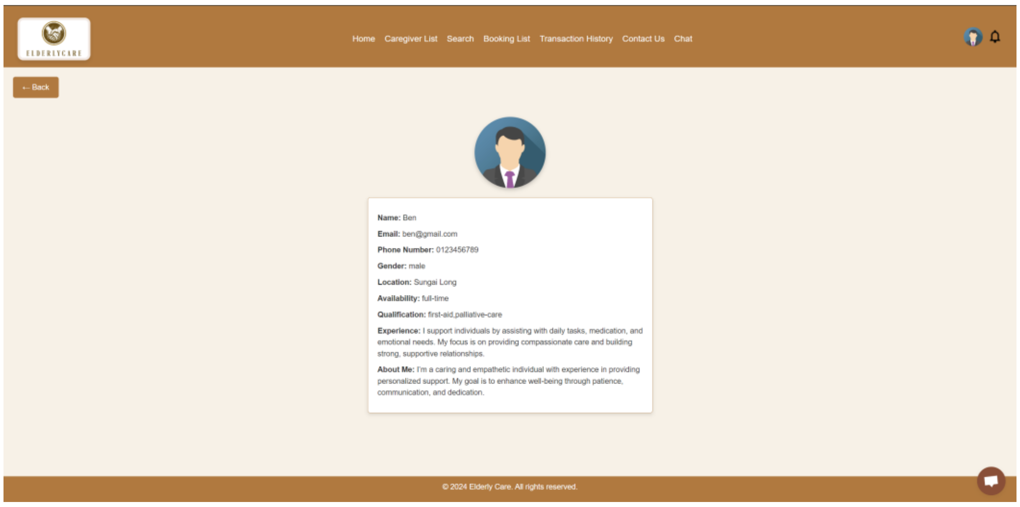
Companion Care
Service Type: Personal Care
Description: Companion care focuses on providing emotional support and social interaction for seniors who may feel isolated or lonely. A caregiver spends time with the elderly person, engaging in activities such as conversation, playing games, or accompanying them on walks. This service is designed to enhance the overall well-being of seniors by fostering companionship and reducing feelings of loneliness.
Price: RM88.00
Duration: 60
Availability: full time
Location: Sungai Long
Provider: Ben

Home Health Care
Service Type: Medical Care
Description: Medical assistance and daily health monitoring, including medication management and health assessments.
Price: RM120.00
Duration: 120
Availability: full time
Location: Sungai Long
Provider: Ben

Memory Care
Service Type: Other
Description: Specialized care for individuals with Alzheimer's or other forms of dementia. Memory care services offer a safe and structured environment where caregivers are trained to manage the unique challenges associated with memory loss. Activities and therapies are tailored to stimulate cognitive function, improve mood, and maintain a sense of identity.
Price: RM60.00
Duration: 60
Availability: full time
Location: Sungai Long
Provider: Ben

Figure 6.20: Search Page

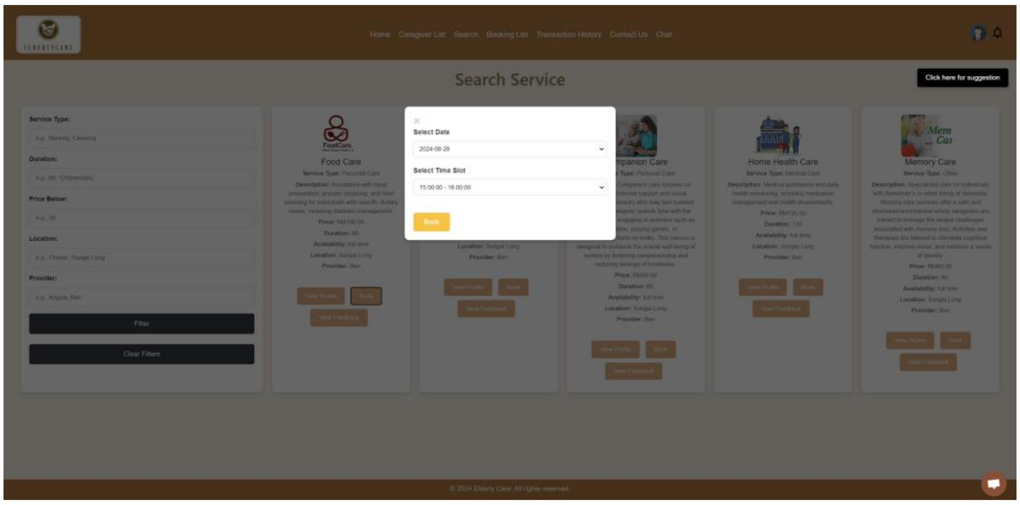
On this search page, clients can view all the services created by the caregivers. The client can filter the service by entering the service type, duration, price, location, and the provider. The clear filter button will clear all the filters applied and return all the services to this page. After that, the client can perform actions like view the caregiver's profile, book the service, and view the feedback on this service.



Name: Ben
Email: ben@gmail.com
Phone Number: 0123456789
Gender: male
Location: Sungai Long
Availability: full-time
Qualification: first-aid,palliative-care
Experience: I support individuals by assisting with daily tasks, medication, and emotional needs. My focus is on providing compassionate care and building strong, supportive relationships.
About Me: I'm a caring and empathetic individual with experience in providing personalized support. My goal is to enhance well-being through patience, communication, and dedication.

Figure 6.21: Caregiver’s Profile Page

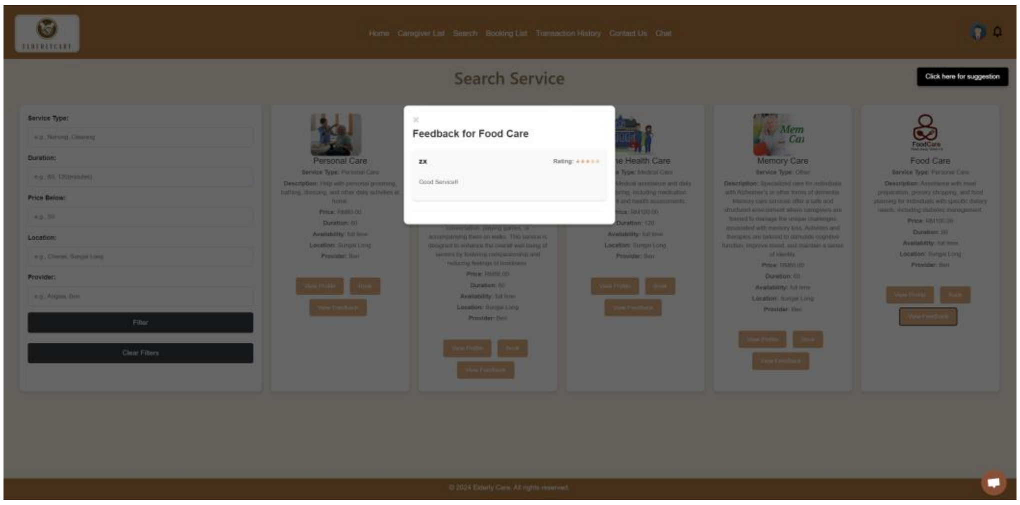
The client can view all the caregiver’s details and information on this page after clicking on the “View Profile” button in the Search Page.


Home Caregiver List Search Booking List Transaction History Contact Us Chat
Search Service
Click here for suggestion
Filter
Clear Filters
Book

Select Date
2024-06-28
Select Time Slot
15:00:00 - 16:00:00

Figure 6.22: Booking Modal

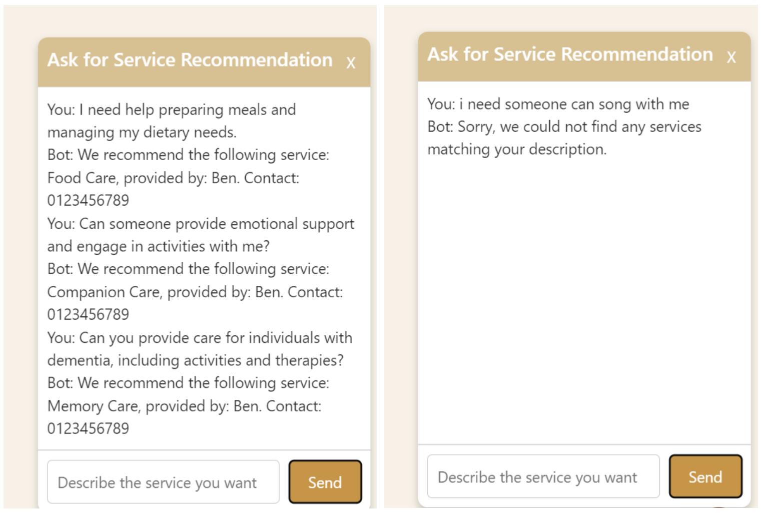
If the client clicks the “Book” button on the Search Page, a booking modal will pop up and ask them to choose the booking date and time slot. The client can select the booking date and time slot to make a booking.


Home Caregiver List Search Booking List Transaction History Contact Us Chat
Search Service
Click here for suggestion
Filter
Clear Filters

Feedback for Food Care
zx Rating: ★★★★★
Good Service!!

Figure 6.23: Feedback Modal

If the client clicks on the view feedback button, a feedback modal will pop out and show all the feedback and ratings of the service.


Ask for Service Recommendation x

You: I need help preparing meals and managing my dietary needs.
Bot: We recommend the following service: Food Care, provided by: Ben. Contact: 0123456789
You: Can someone provide emotional support and engage in activities with me?
Bot: We recommend the following service: Companion Care, provided by: Ben. Contact: 0123456789
You: Can you provide care for individuals with dementia, including activities and therapies?
Bot: We recommend the following service: Memory Care, provided by: Ben. Contact: 0123456789

Ask for Service Recommendation x

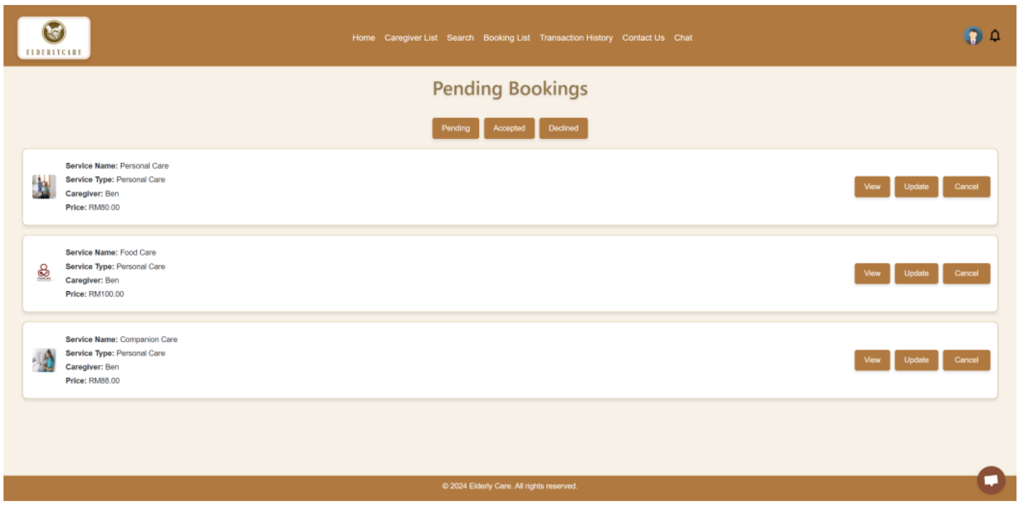
You: i need someone can song with me
Bot: Sorry, we could not find any services matching your description.

Figure 6.24: AI Service Recommendation Chatbot

The client may click on the "Click Here for Suggestion" black button to request a service recommendation. When the client clicks the button, they will be prompted to describe the service they need. Once the description is submitted, the server processes the request using natural language processing techniques. Specifically, the server compares the client's description with the descriptions of services in the database using a TF-IDF model. It identifies the service with the highest similarity score to the client's request and returns this as the recommended service along with the caregiver details.

If the client's description does not match any of the platform's services, the server will return a message stating that no matching services were discovered. This guarantees that clients receive appropriate recommendations or are notified if their request cannot be addressed by the present service options. Overall, this process aims to provide clients with tailored service suggestions based on their specific needs or inform them when their request is outside the platform's capabilities.

6.4.4.4 Booking List Page



Pending Bookings

Service Name: Personal Care
Service Type: Personal Care
Caregiver: Ben
Price: RM80.00

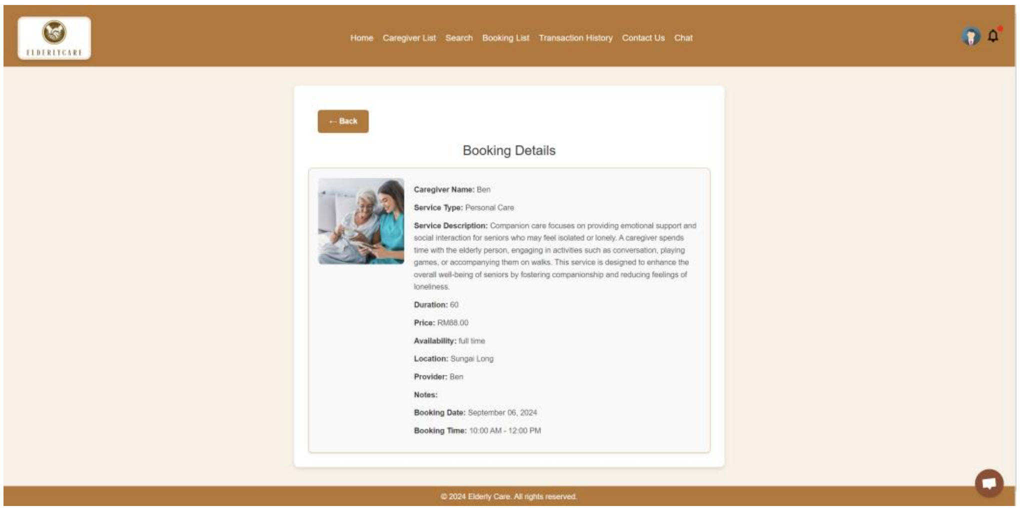
Service Name: Food Care
Service Type: Personal Care
Caregiver: Ben
Price: RM100.00

Service Name: Companion Care
Service Type: Personal Care
Caregiver: Ben
Price: RM88.00

© 2024 Elderly Care. All rights reserved.

Figure 6.25: Pending Booking Page

The Pending Booking Page shows all the services that are still pending from the caregiver.



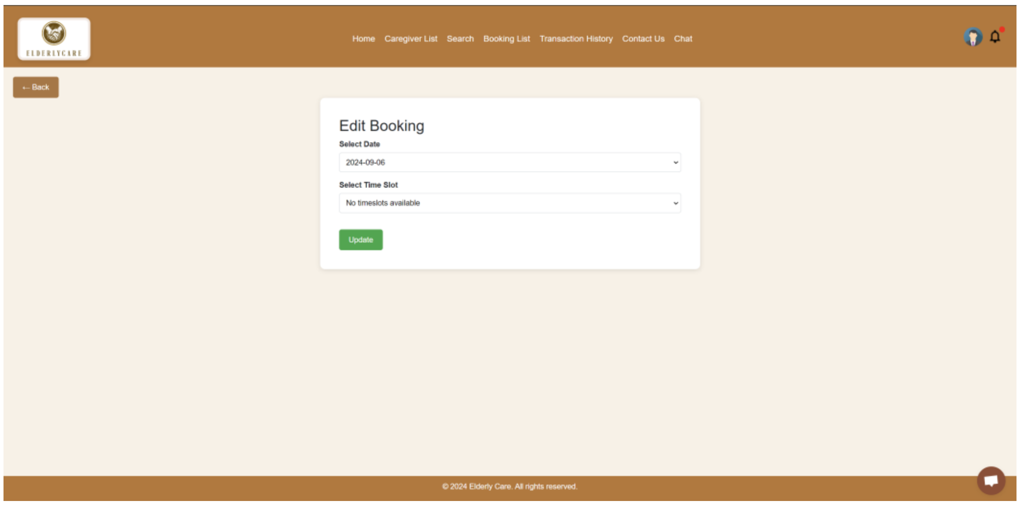
Booking Details

Caregiver Name: Ben
Service Type: Personal Care
Service Description: Companion care focuses on providing emotional support and social interaction for seniors who may feel isolated or lonely. A caregiver spends time with the elderly person, engaging in activities such as conversation, playing games, or accompanying them on walks. This service is designed to enhance the overall well-being of seniors by fostering companionship and reducing feelings of loneliness.
Duration: 60
Price: RM88.00
Availability: full time
Location: Sungai Long
Provider: Ben
Notes:
Booking Date: September 06, 2024
Booking Time: 10:00 AM - 12:00 PM

© 2024 Elderly Care. All rights reserved.

Figure 6.26: Booking Details Page

After clicking the "View" button on the Pending Booking Page, the client will be redirected to this booking details page. This page will show the service details and the booking date and time.



Edit Booking

Select Date

2024-09-06

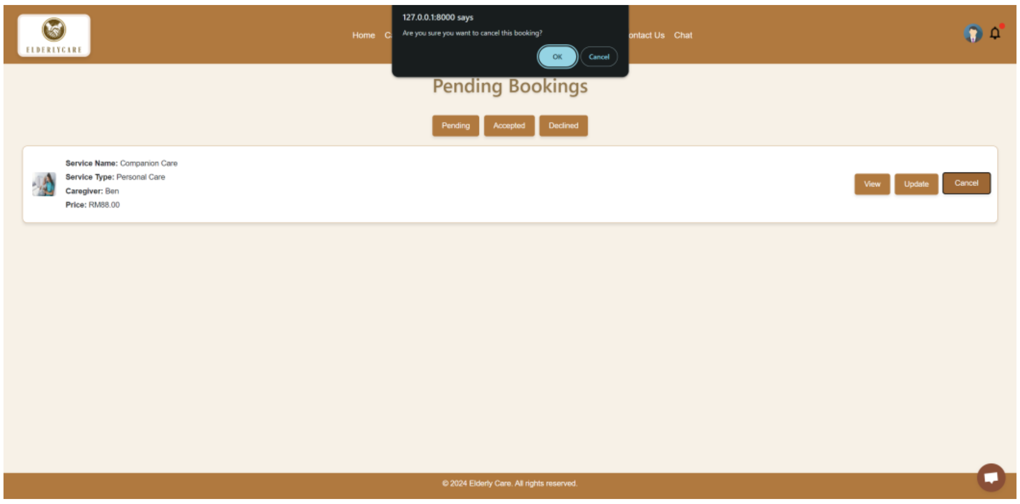
Select Time Slot

No timeslots available

© 2024 Elderly Care. All rights reserved.

Figure 6.27: Edit Booking Page

The client can also update the service by selecting a new date and timeslot and updating it.



127.0.0.1:8000 says

Are you sure you want to cancel this booking?

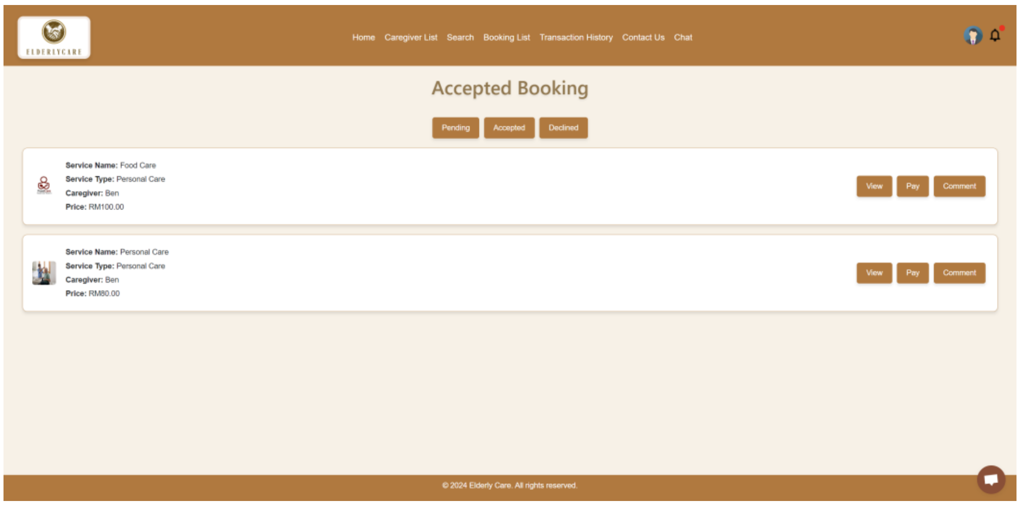
Pending Bookings

Service Name: Companion Care
Service Type: Personal Care
Caregiver: Ben
Price: RM88.00

© 2024 Elderly Care. All rights reserved.

Figure 6.28: Alert Message of Deletion of a Booking

If the client clicks on the “Cancel” button, an alert message will show. Once the client confirms the deletion, the booking will be deleted.



Accepted Booking

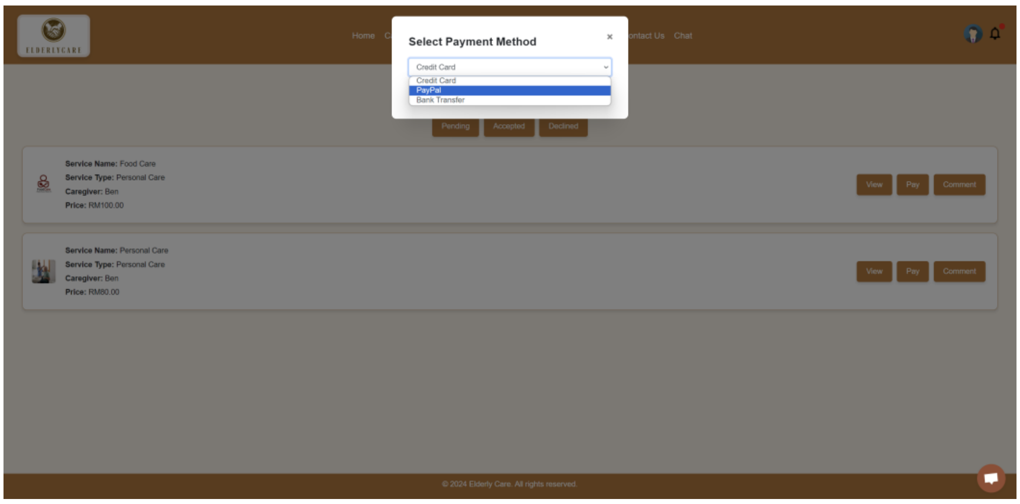
Service Name: Food Care
Service Type: Personal Care
Caregiver: Ben
Price: RM100.00

Service Name: Personal Care
Service Type: Personal Care
Caregiver: Ben
Price: RM80.00

© 2024 Elderly Care. All rights reserved.

Figure 6.29: Accepted Booking List Page

This page will show all the bookings that were accepted by the caregiver. On this page, the client can view the bookings, pay the bill, and provide a comment.



Select Payment Method

Credit Card
- Credit Card
- PayPal
- Bank Transfer

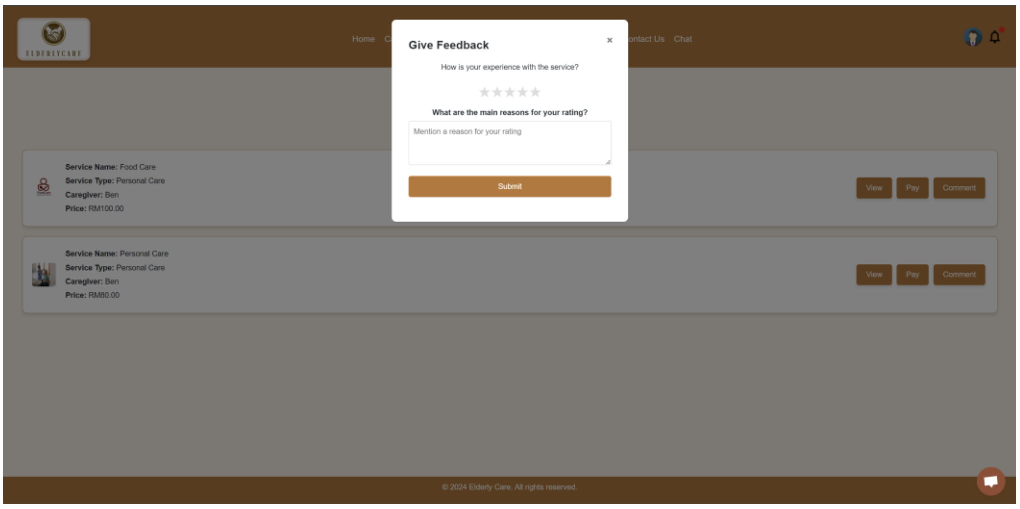
Service Name: Food Care
Service Type: Personal Care
Caregiver: Ben
Price: RM100.00

Service Name: Personal Care
Service Type: Personal Care
Caregiver: Ben
Price: RM80.00

© 2024 Elderly Care. All rights reserved.

Figure 6.30: Payment Modal

Once the client clicks on the "Pay" button, the Select Payment Method modal will be shown, and ask the caregiver to select a payment method. After the payment has been made, a notification will be sent to the caregiver to notify them that the payment of the service has been paid.



Give Feedback

How is your experience with the service?

★★★★★

What are the main reasons for your rating?

Mention a reason for your rating

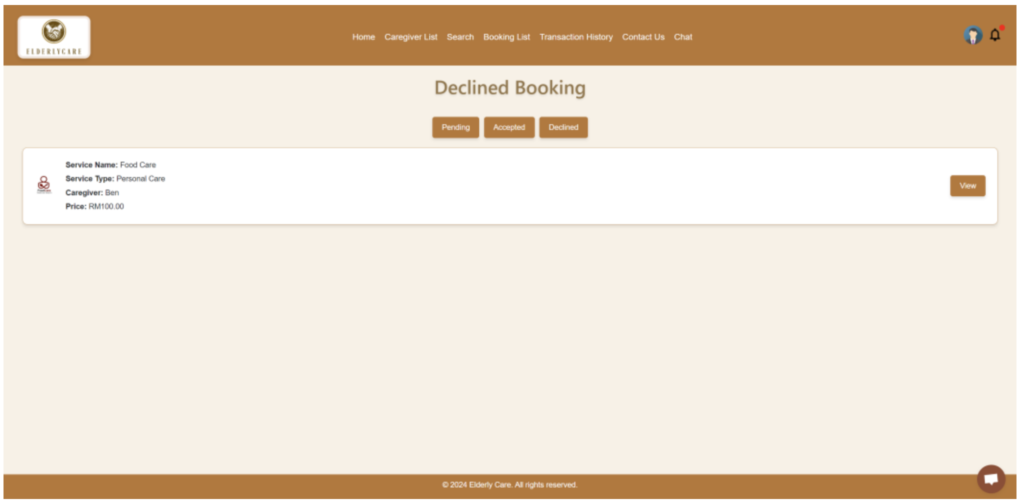
Submit

Service Name: Food Care
Service Type: Personal Care
Caregiver: Ben
Price: RM100.00

Service Name: Personal Care
Service Type: Personal Care
Caregiver: Ben
Price: RM80.00

Figure 6.31: Feedback Modal

Same as the "Pay" button, once the client clicks on the "Comment" button, the feedback modal will be shown. The client can rank the service and provide some comments on the service.



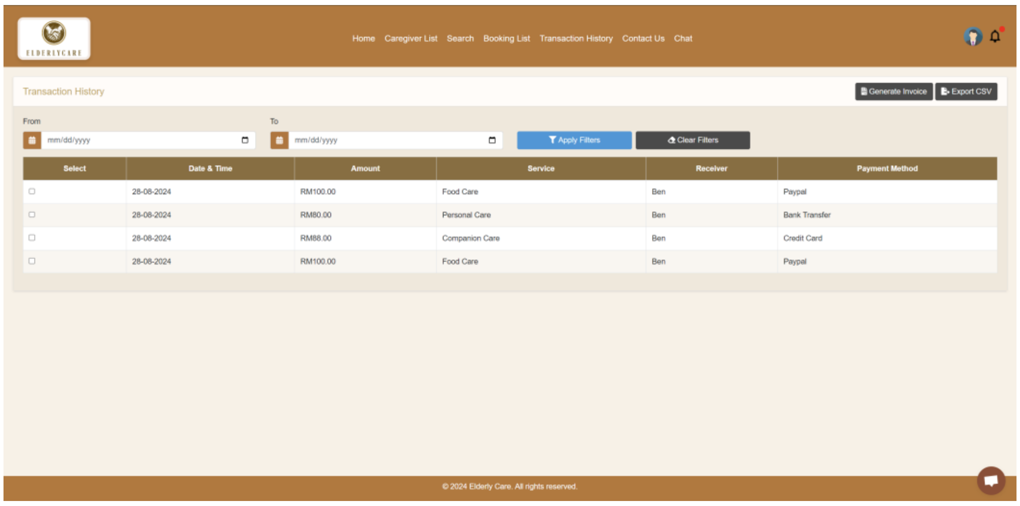
Declined Booking

Service Name: Food Care
Service Type: Personal Care
Caregiver: Ben
Price: RM100.00

Figure 6.32: Declined Booking List Page

The Declined Booking Page shows all the services that have been declined by the caregiver. The client can view the booking details on this page.

6.4.4.5 Transaction History Page

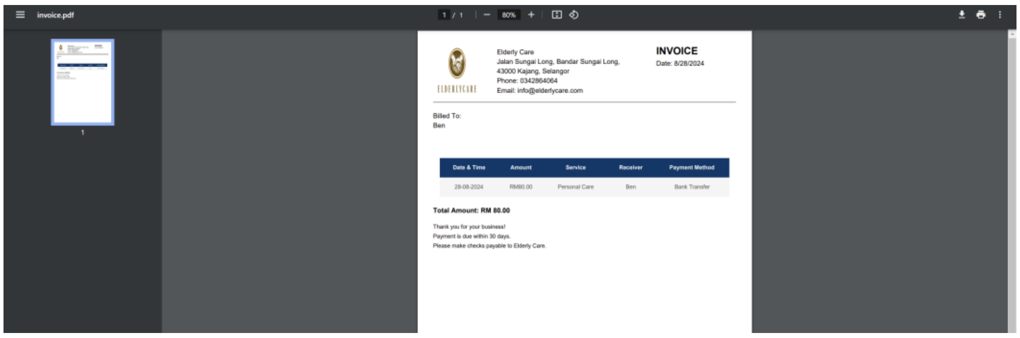


Select	Date & Time	Amount	Service	Receiver	Payment Method
☐	28-08-2024	RM100.00	Food Care	Ben	Paypal
☐	28-08-2024	RM80.00	Personal Care	Ben	Bank Transfer
☐	28-08-2024	RM88.00	Companion Care	Ben	Credit Card
☐	28-08-2024	RM100.00	Food Care	Ben	Paypal

© 2024 Elderly Care. All rights reserved.

Figure 6.33: Transaction History Page

This Transaction History Page will show all the transactions made by the log-in client for record-keeping purposes. On this page, clients can search for the transaction history during a specific date by applying the date filter. The client may also select one or more transactions to generate an invoice or export to CSV.



Elderly Care
Jalan Sungai Long, Bandar Sungai Long,
43000 Kajang, Selangor
Phone: 0342864064
Email: info@elderlycare.com

INVOICE
Date: 8/28/2024

Billed To:
Ben

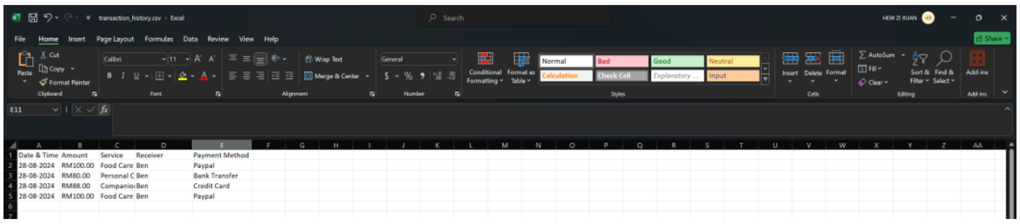
Date & Time	Amount	Service	Receiver	Payment Method
28-08-2024	RM80.00	Personal Care	Ben	Bank Transfer

Total Amount: RM 80.00

Thank you for your business!
Payment is due within 30 days.
Please make checks payable to Elderly Care.

Figure 6.34: Invoice of a Transaction

The client can select one or more transactions to generate an invoice by clicking on the "Generate Invoice" button. After clicking on the button, the invoice pdf will download automatically to the client's PC. The figure above shows the invoice of a transaction.

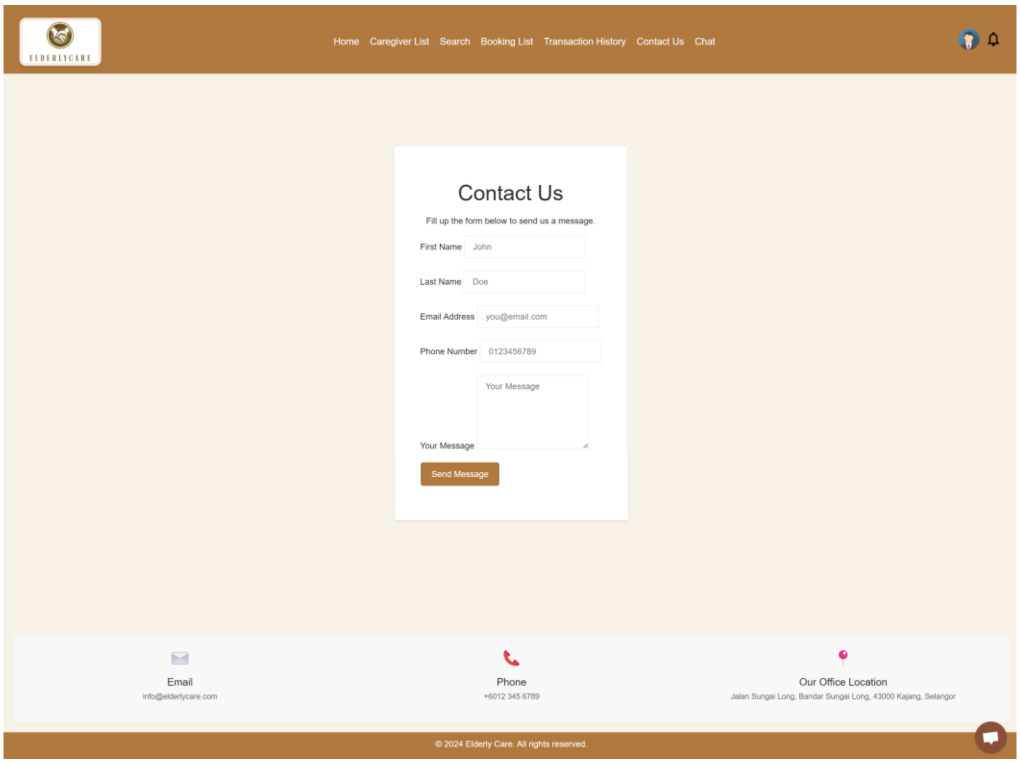


Date & Time	Amount	Service	Receiver	Payment Method
28-08-2024	RM100.00	Food Care	Ben	Paypal
28-08-2024	RM80.00	Personal C	Ben	Bank Transfer
28-08-2024	RM88.00	Companio	Ben	Credit Card
28-08-2024	RM100.00	Food Care	Ben	Paypal

Figure 6.35: CSV file of a Transaction

The client can also select one or more transactions to export to CSV by clicking on the "Export CSV" button. After clicking on the button, the CSV file will download automatically to the client's PC. The figure above shows the CSV file of a transaction.

6.4.4.6 Contact Us Page



Contact Us

Fill up the form below to send us a message.

First Name	John
Last Name	Doe
Email Address	you@email.com
Phone Number	0123456789
Your Message	Your Message

Email
info@elderlycare.com

Phone
+6012 345 6789

Our Office Location
Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

© 2024 Elderly Care. All rights reserved.

Figure 6.36: Contact Us Page

The client can fill up the Contact Us form to send a message or ask any questions. The client will receive the response in the email later.

6.4.4.7 Chat Page

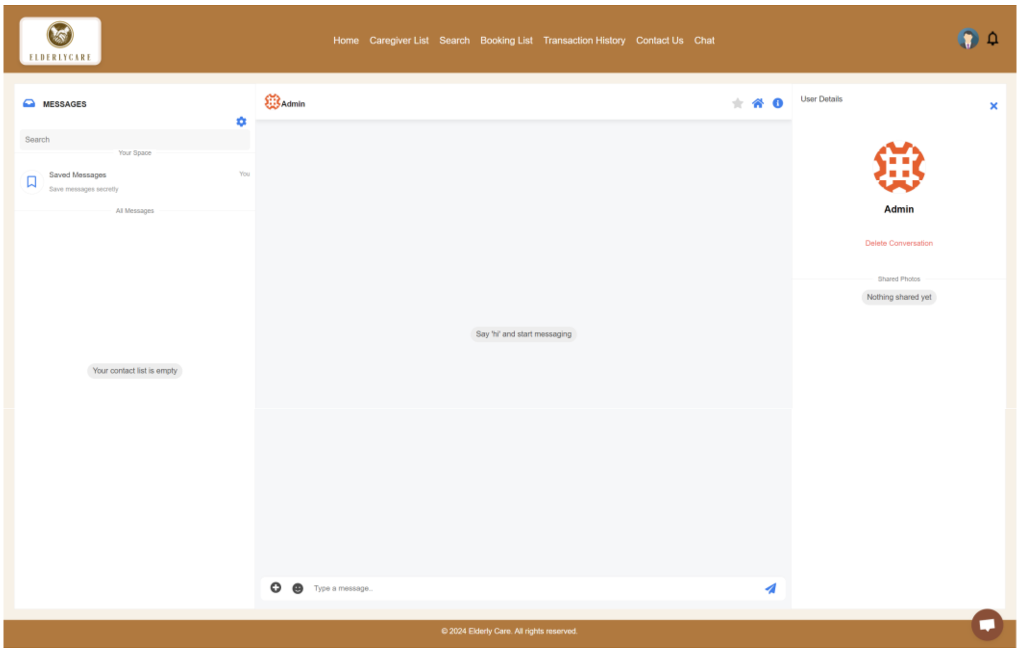
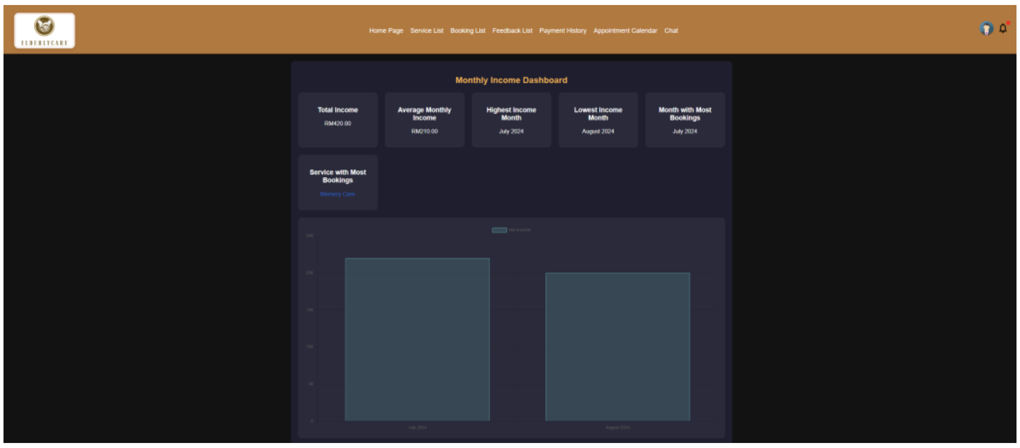


Figure 6.37: Chat Page

The client can engage in direct communication with all platform users through the integrated chat channel. Additionally, the client has the option to mark specific users as favourites for easier access and streamlined interactions in the future. The client is also allowed to send images or emojis using this chat channel.

6.4.5 Caregiver

6.4.5.1 Home Page



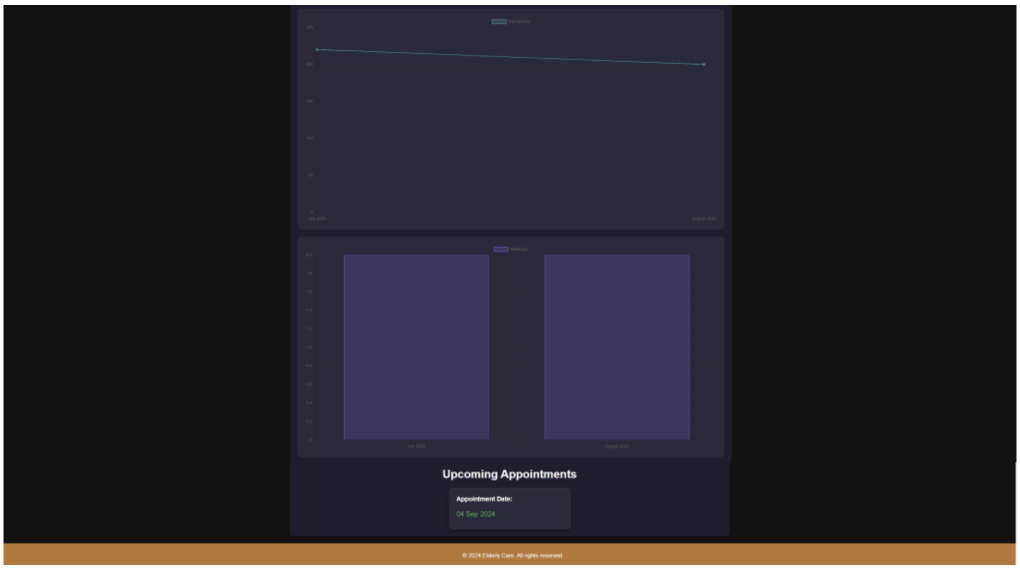


Figure 6.38: Caregiver Monthly Dashboard Page

The caregiver can check for their monthly income dashboard and upcoming appointments on the Home Page. Some financial information, such as total income, average monthly income, highest income month, lowest income month, month with most bookings, and service with most bookings, will be listed on this page. Apart from that, some graphs also show a better evaluation of the monthly income. The first and second graphs are for the net income, and the third graph is for the net booking for each month. For the upcoming appointments section, the caregiver can click on the date link, and they will be redirected to the booking details page.

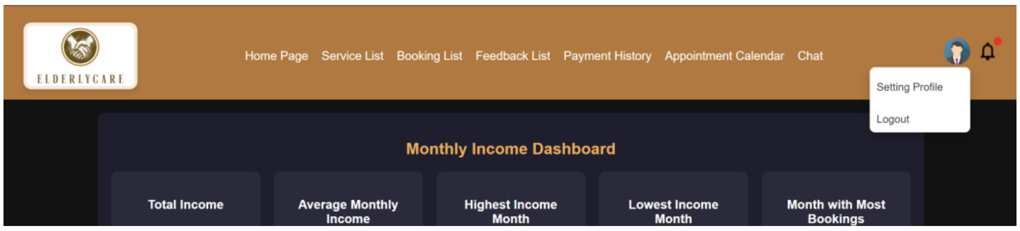
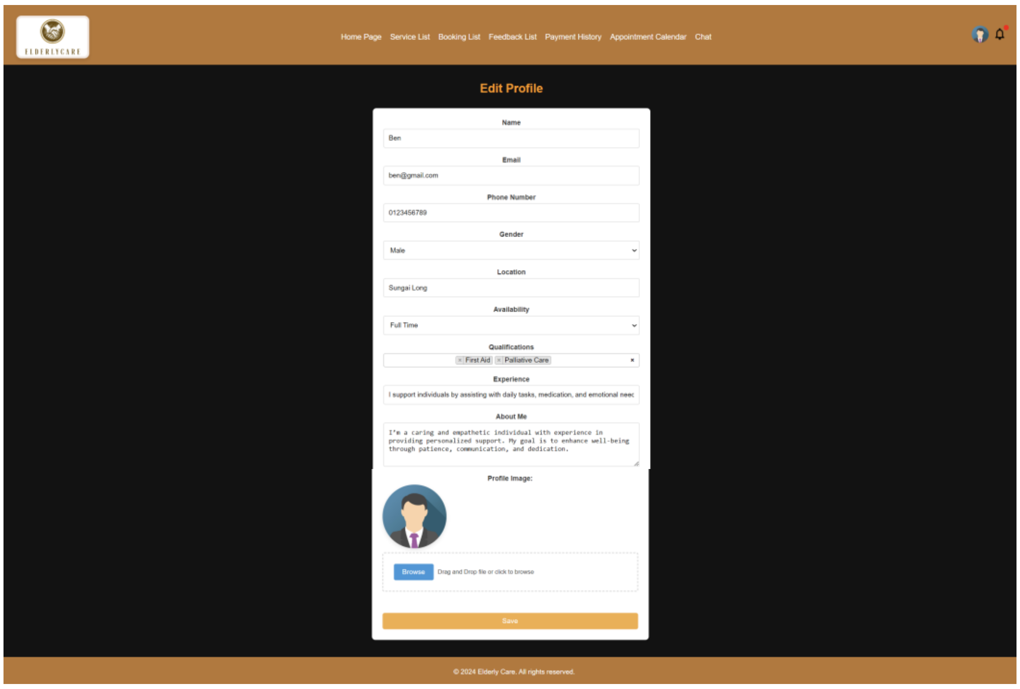


Figure 6.39: Profile Icon Dropdown

The setting profile and logout dropdown will show when the caregiver clicks on the profile icon on the header.



Edit Profile

Name: Ben

Email: ben@gmail.com

Phone Number: 0123456789

Gender: Male

Location: Sungai Long

Availability: Full Time

Qualifications: First Aid, Palliative Care

Experience: I support individuals by assisting with daily tasks, medication, and emotional neec

About Me: I'm a caring and empathetic individual with experience in providing personalized support. My goal is to enhance well-being through patience, communication, and dedication.

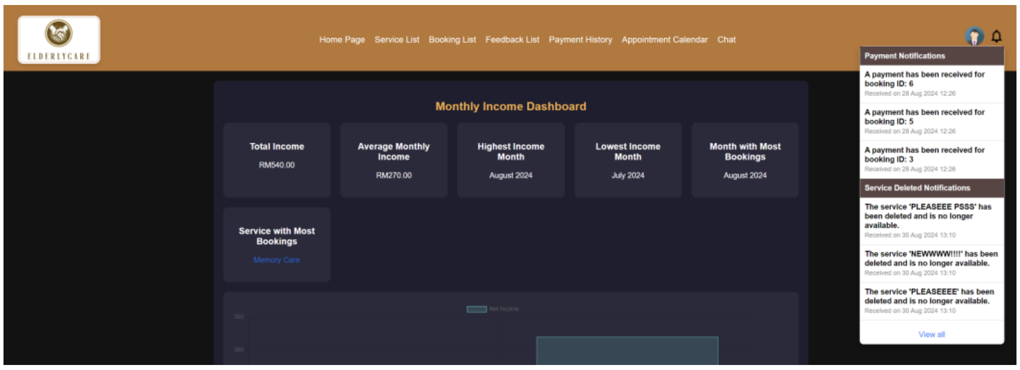
Profile Image:

Drag and Drop file or click to browse

© 2024 Elderly Care. All rights reserved.

Figure 6.40: Edit Profile Page

The caregiver can update their profile details on this page by entering their personal information and profile image.



Monthly Income Dashboard

Total Income: RM540.00

Average Monthly Income: RM270.00

Highest Income Month: August 2024

Lowest Income Month: July 2024

Month with Most Bookings: August 2024

Service with Most Bookings: Memory Care

Payment Notifications

A payment has been received for booking ID: 6
Received on 28 Aug 2024 12:26

A payment has been received for booking ID: 5
Received on 28 Aug 2024 12:26

A payment has been received for booking ID: 3
Received on 28 Aug 2024 12:26

Service Deleted Notifications

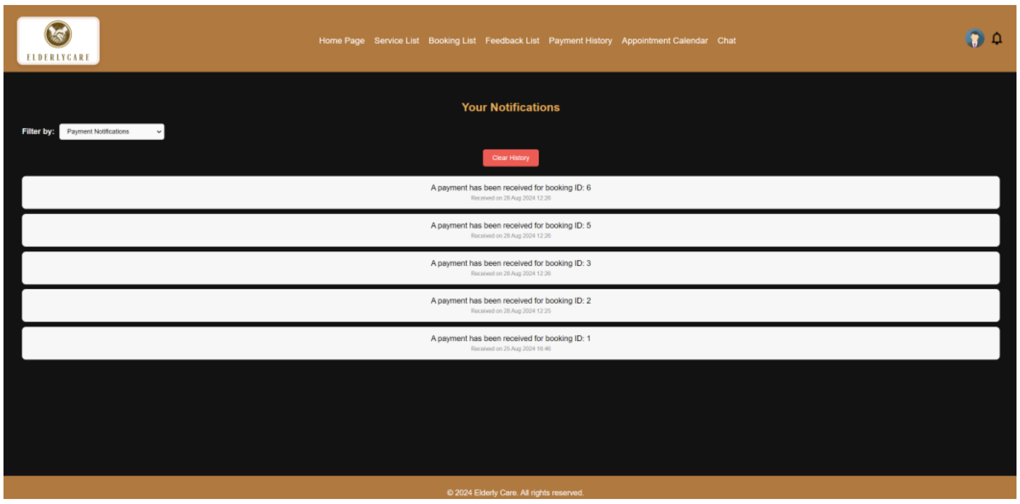
The service 'PLEASEEE PSSS' has been deleted and is no longer available.
Received on 30 Aug 2024 13:10

The service 'NEWWWW!!!!' has been deleted and is no longer available.
Received on 30 Aug 2024 13:10

The service 'PLEASEEEE' has been deleted and is no longer available.
Received on 30 Aug 2024 13:10

Figure 6.41: Notification Dropdown

The caregiver can check for the new notification by clicking on the notification icon on the header. The red dot on the notification icon indicates that there is an unread notification. Once the caregiver clicks on it, the red dot will disappear, indicating that there is no unread notification anymore.



Your Notifications

A payment has been received for booking ID: 6
Received on 28 Aug 2024 12:26

A payment has been received for booking ID: 5
Received on 28 Aug 2024 12:26

A payment has been received for booking ID: 3
Received on 28 Aug 2024 12:26

A payment has been received for booking ID: 2
Received on 28 Aug 2024 12:25

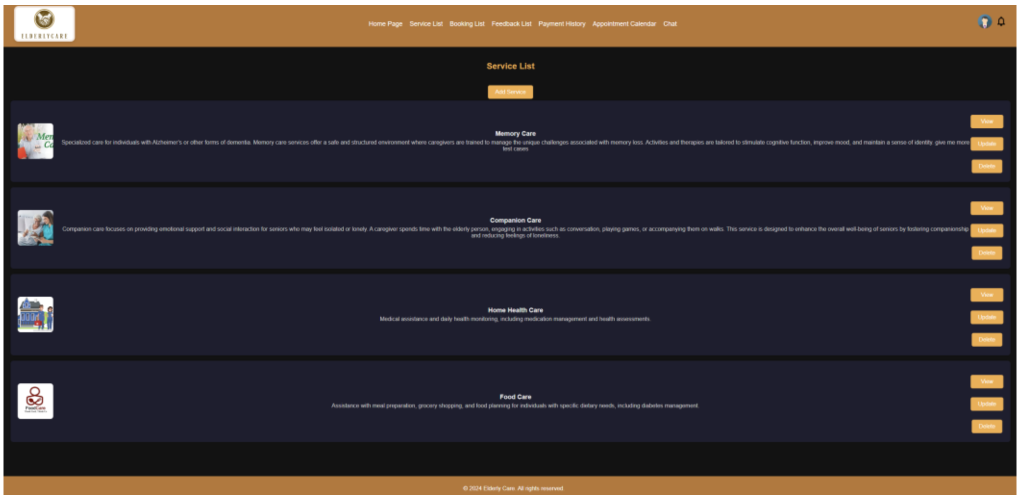
A payment has been received for booking ID: 1
Received on 25 Aug 2024 16:46

© 2024 Elderly Care. All rights reserved.

Figure 6.42: Notification Page

When the caregiver clicks on the "View all" blue word on the notification dropdown, they will be redirected to this Notification Page which shows all the notifications received by the caregiver. The caregiver can filter the notification by applying the notification filter. The caregiver may also clear all the notifications by clicking on the red clear notification button.

6.4.5.2 Service List Page



Service List

Memory Care
Specialized care for individuals with Alzheimer's or other forms of dementia. Memory care services offer a safe and structured environment where caregivers are trained to manage the unique challenges associated with memory loss. Activities and therapies are tailored to stimulate cognitive function, improve mood, and maintain a sense of identity. give me more test cases

Companion Care
Companion care focuses on providing emotional support and social interaction for seniors who may feel isolated or lonely. A caregiver spends time with the elderly person, engaging in activities such as conversation, playing games, or accompanying them on walks. This service is designed to enhance the overall well-being of seniors by fostering companionship and reducing feelings of loneliness.

Home Health Care
Medical assistance and daily health monitoring, including medication management and health assessments.

Food Care
Assistance with meal preparation, grocery shopping, and food planning for individuals with specific dietary needs, including diabetes management.

© 2024 Elderly Care. All rights reserved.

Figure 6.43: Service List Page

In this service list page, caregivers are allowed to add, view, update, and delete a service.

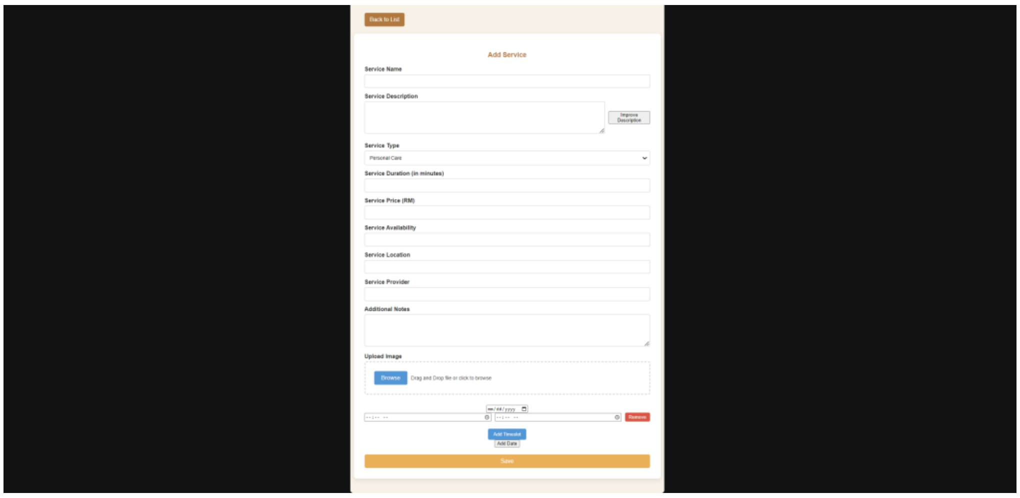


Figure 6.44: Create Service Page

When the caregiver clicks on the “Add” button on the Service List Page, they will be redirected to this Create Service Page. The caregiver needs to fill up the service details, the date, and timeslot to create a new service. If there is any incorrect data input, such as the end time being earlier than the start time, the error message will be shown.

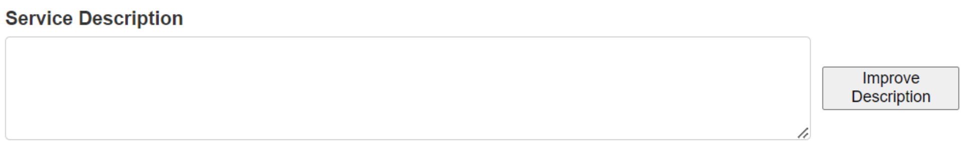
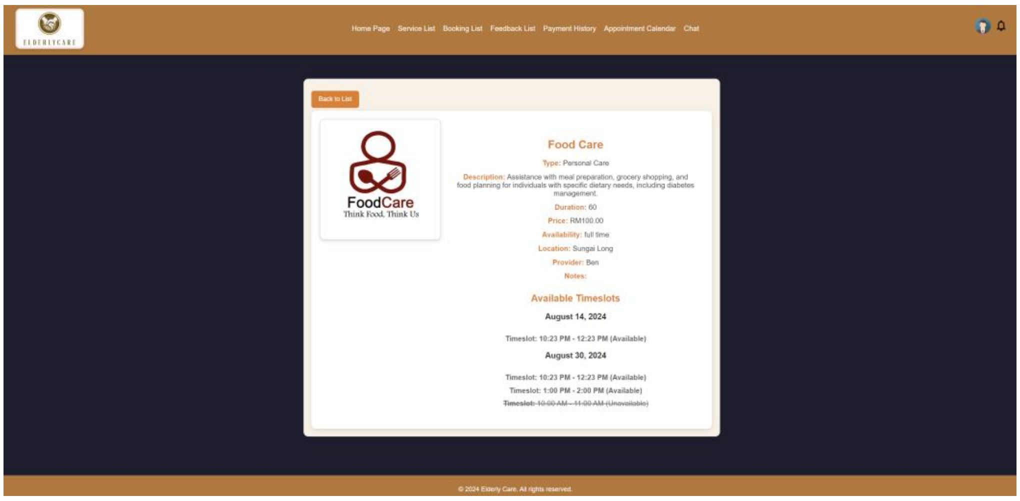


Figure 6.45: AI Improve Description

The caregiver can enter a brief description of their service into the service description text box and click on the "Improve Description" button to enhance the content. When this button is clicked, the system first checks if the caregiver has described by validating the input field. If the text box is empty, an alert prompts the user to input a description before proceeding. Once a description is provided, an API call is triggered using the `fetch` function. This call sends a POST request to the ChatGPT API via RapidAPI which containing the caregiver's description wrapped in a structured JSON body.

The API request sends the message in the format: "Improve this description: [caregiver's description]," along with parameters such as max tokens, and authentication headers like the API key. The API processes this request and responds with an improved version of the caregiver's description. Once the response is received, the code checks if the API returned a valid result. If successful, the improved description automatically replaces the original text in the input field. This allows the caregiver to immediately view the enhanced version. If there is an error with the API call or the response structure is unexpected, an error message is logged and the user is notified. This process allows caregivers to refine and optimize their service descriptions effortlessly by ensuring they present more attractive and engaging descriptions for potential clients.



Food Care

Type: Personal Care

Description: Assistance with meal preparation, grocery shopping, and food planning for individuals with specific dietary needs, including diabetes management.

Duration: 60

Price: RM100.00

Availability: full time

Location: Sungai Long

Provider: Ben

Notes:

Available Timeslots

August 14, 2024

Timeslot: 10:23 PM - 12:23 PM (Available)

August 30, 2024

Timeslot: 10:23 PM - 12:23 PM (Available)

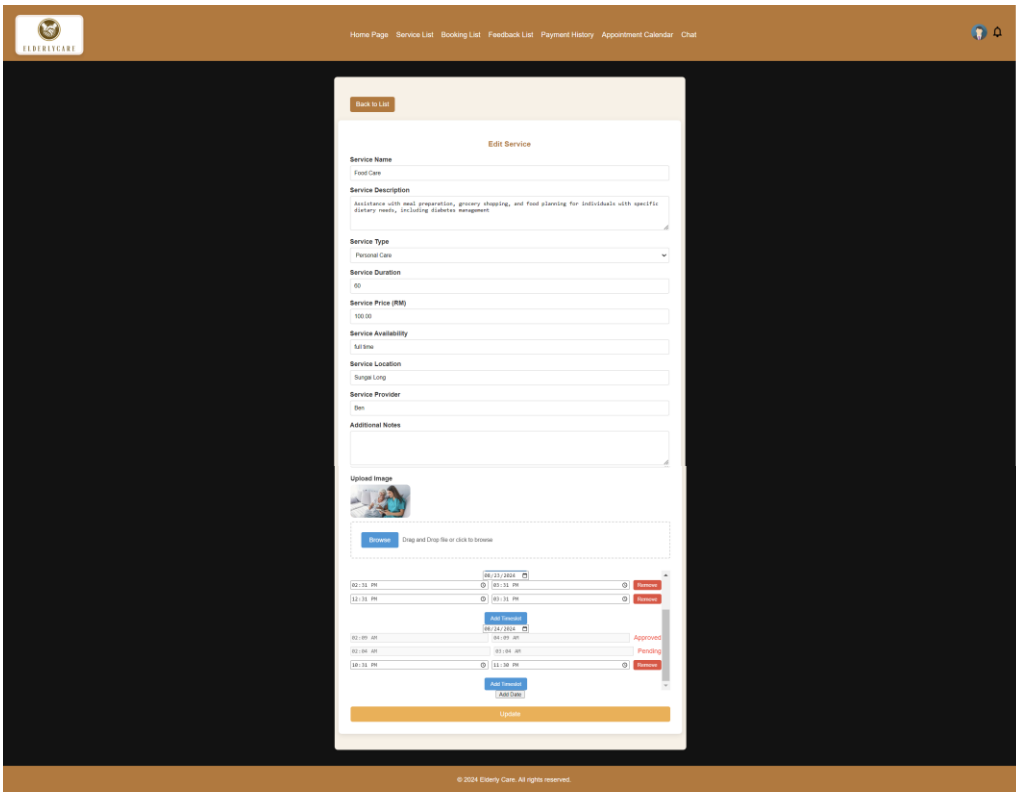
Timeslot: 1:00 PM - 2:00 PM (Available)

~~Timeslot: 10:00 AM - 11:00 AM (Unavailable)~~

© 2024 Elderly Care. All rights reserved.

Figure 6.46: Service Details Page

After clicking on the “View” button on the Service List Page, the caregiver will be redirected to this page. On this page, the caregiver can view the service details and the current availability of the timeslots.



Edit Service

Service Name: Food Care

Service Description: Assistance with meal preparation, grocery shopping, and food planning for individuals with specific dietary needs, including diabetes management

Service Type: Personal Care

Service Duration: 60

Service Price (RM): 100.00

Service Availability: full time

Service Location: Sungai Long

Service Provider: Ben

Additional Notes:

Upload Image

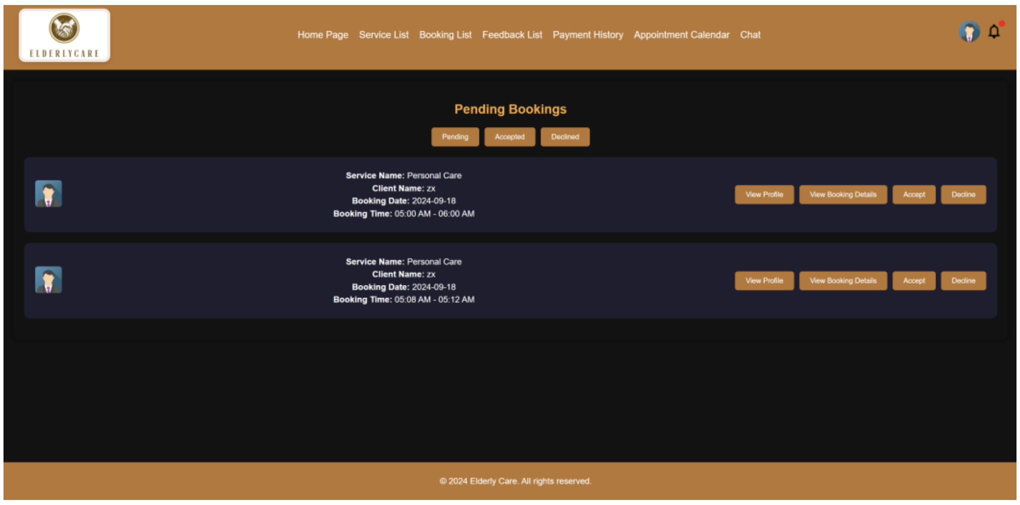
Drag and Drop file or click to browse

08/23/2024		
02:31 PM	03:31 PM	Remove
12:31 PM	03:31 PM	Remove
08/24/2024		
02:09 AM	04:09 AM	Approved
02:04 AM	03:04 AM	Pending
10:31 PM	11:30 PM	Remove

Figure 6.47: Update Service Page

After selecting a service and clicking the "Update" button, the caregiver will be redirected to the Update Service Page. On this page, the caregiver can update their service by entering new data. As shown in Figure 6.47, this page displays the timeslots associated with the selected service, including both pending and approved slots. Caregivers are not permitted to modify these pending or approved timeslots. If any incorrect data is entered, such as an end time that precedes the start time, an error message will be displayed to guide the caregiver in correcting the issue.

6.4.5.3 Booking List Page



Pending Bookings

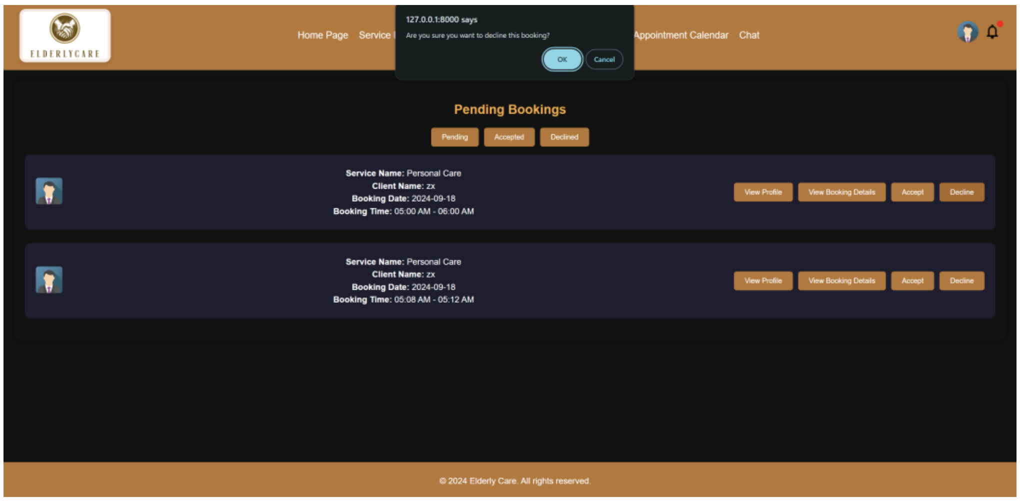
Service Name: Personal Care
Client Name: zx
Booking Date: 2024-09-18
Booking Time: 05:00 AM - 06:00 AM

Service Name: Personal Care
Client Name: zx
Booking Date: 2024-09-18
Booking Time: 05:08 AM - 05:12 AM

© 2024 Elderly Care. All rights reserved.

Figure 6.48: Pending Booking Page

The figure above is the Pending Booking Page, which shows the booking that was booked by the client and is still pending action. The caregiver can view the client profile, view booking details, and accept or decline the service on this page. After the booking has been approved or declined, a notification will be sent to the caregiver to notify them that the service has been approved or declined.



127.0.0.1:8000 says

Are you sure you want to decline this booking?

Pending Bookings

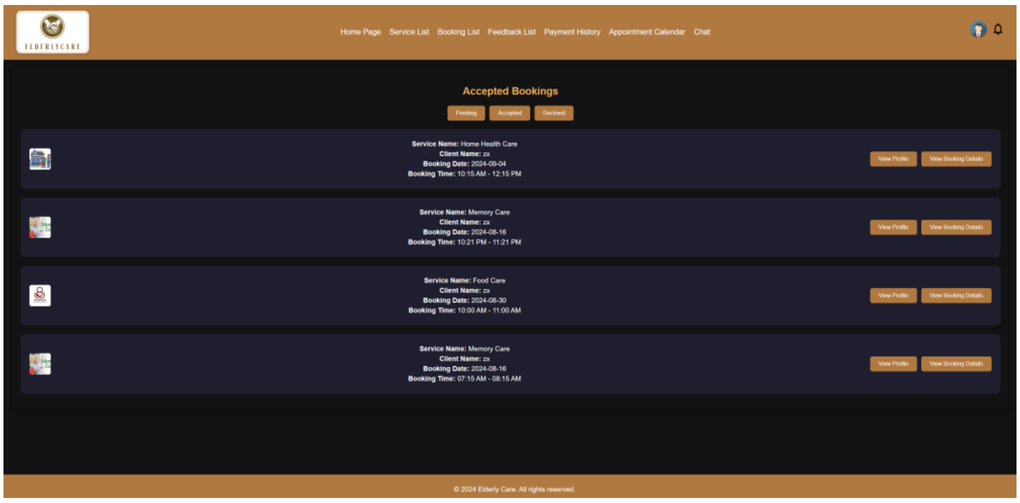
Service Name: Personal Care
Client Name: zx
Booking Date: 2024-09-18
Booking Time: 05:00 AM - 06:00 AM

Service Name: Personal Care
Client Name: zx
Booking Date: 2024-09-18
Booking Time: 05:08 AM - 05:12 AM

© 2024 Elderly Care. All rights reserved.

Figure 6.49: Alert Message When Decline the Booking

If the caregiver clicks on the "Decline" button, an alert message will show. Once the caregiver confirms the deletion, the booking will be deleted.



Accepted Bookings

Service Name: Home Health Care
Client Name: zx
Booking Date: 2024-09-04
Booking Time: 10:15 AM - 12:15 PM

Service Name: Memory Care
Client Name: zx
Booking Date: 2024-08-16
Booking Time: 10:21 PM - 11:21 PM

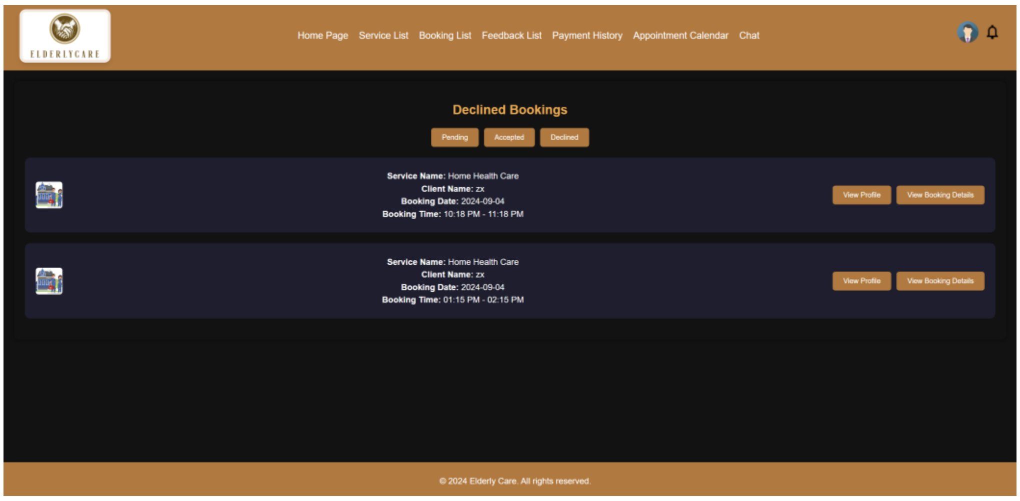
Service Name: Food Care
Client Name: zx
Booking Date: 2024-08-30
Booking Time: 10:00 AM - 11:00 AM

Service Name: Memory Care
Client Name: zx
Booking Date: 2024-08-16
Booking Time: 07:15 AM - 08:15 AM

© 2024 Elderly Care. All rights reserved.

Figure 6.50: Accepted Booking Page

After the caregiver accepts a booking from the Pending Booking Page, the status of the booking will now become accepted and listed in the Accepted Booking Page.



Declined Bookings

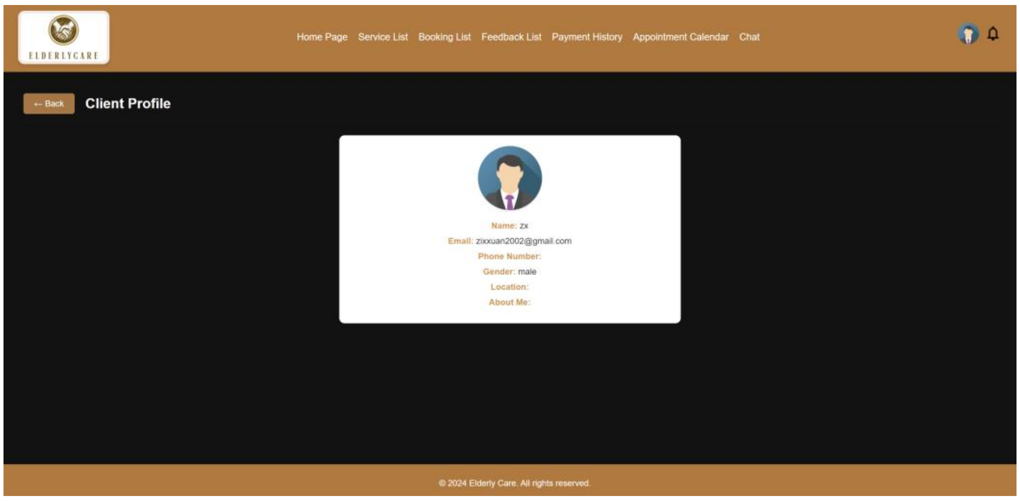
Service Name: Home Health Care
Client Name: zx
Booking Date: 2024-09-04
Booking Time: 10:18 PM - 11:18 PM

Service Name: Home Health Care
Client Name: zx
Booking Date: 2024-09-04
Booking Time: 01:15 PM - 02:15 PM

© 2024 Elderly Care. All rights reserved.

Figure 6.51: Declined Booking Page

If the caregiver declines a pending booking, the status of the booking now becomes declined and will be listed on this Declined Booking Page.



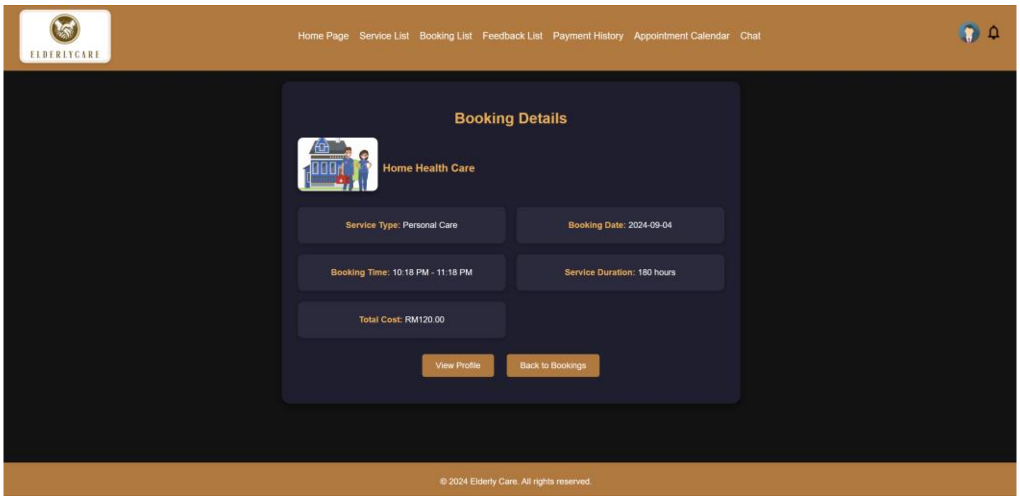
Client Profile

Name: zx
Email: zixxuan2002@gmail.com
Phone Number:
Gender: male
Location:
About Me:

© 2024 Elderly Care. All rights reserved.

Figure 6.52: Client's Profile Page

The caregiver will be redirected to this page when they click on the "View" profile button. This page shows all the client profile details including name, email, phone number, gender, location, and about me.



Booking Details

Home Health Care

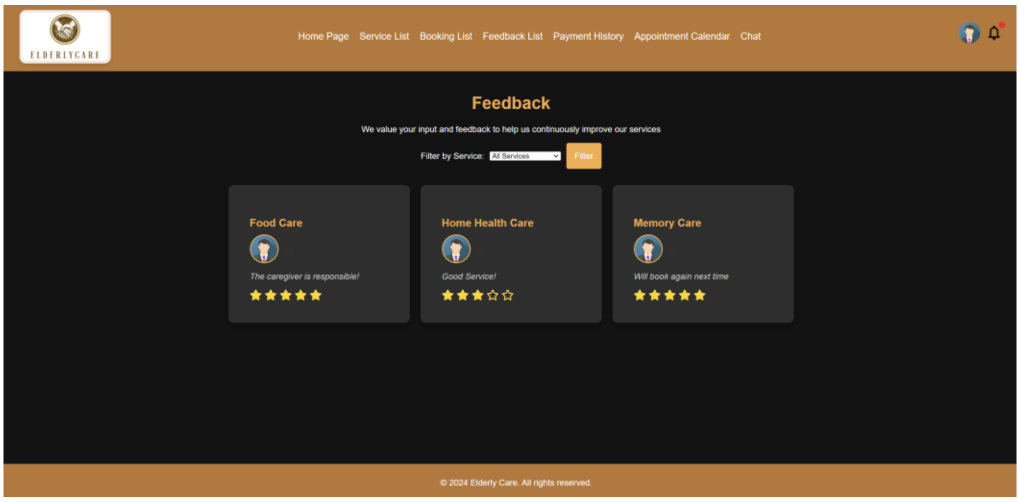
Service Type: Personal Care
Booking Date: 2024-09-04
Booking Time: 10:18 PM - 11:18 PM
Service Duration: 180 hours
Total Cost: RM120.00

© 2024 Elderly Care. All rights reserved.

Figure 6.53: Booking Details Page

After the caregiver chooses a booking and clicks on the "View Booking Details" button, they will be redirected to this page. This page shows the booking details including service type, booking date and time, service duration, client address, and total cost.

6.4.5.4 Feedback List Page



Feedback

We value your input and feedback to help us continuously improve our services

Food Care

The caregiver is responsible!

★★★★★

Home Health Care

Good Service!

★★★☆☆

Memory Care

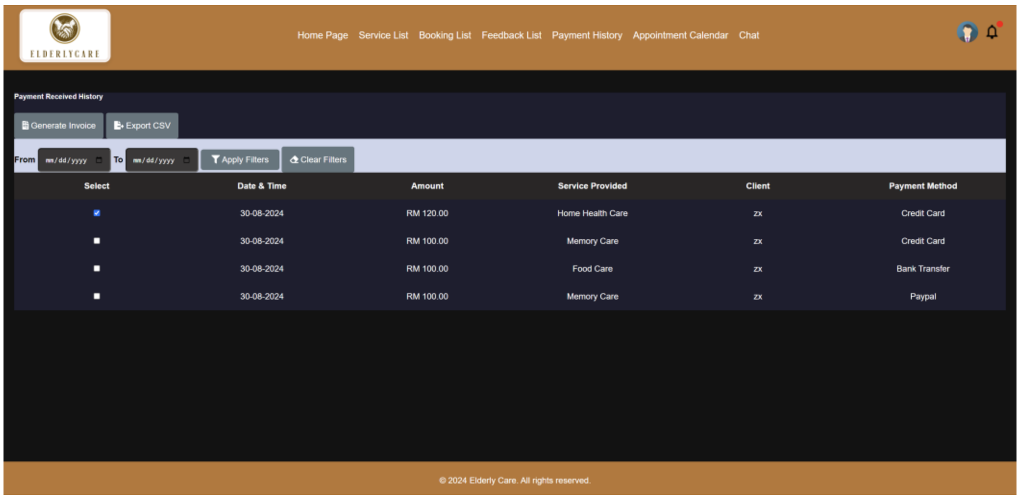
Will book again next time

★★★★★

Figure 6.54: Feedback List Page

On the Feedback Page, the caregiver can view all feedback provided by clients. Additionally, the caregiver has the option to filter the feedback by applying a service-specific filter. This allows them to view comments related to a particular service. This feature helps caregivers easily manage and review feedback relevant to their offerings.

6.4.5.5 Payment History Page

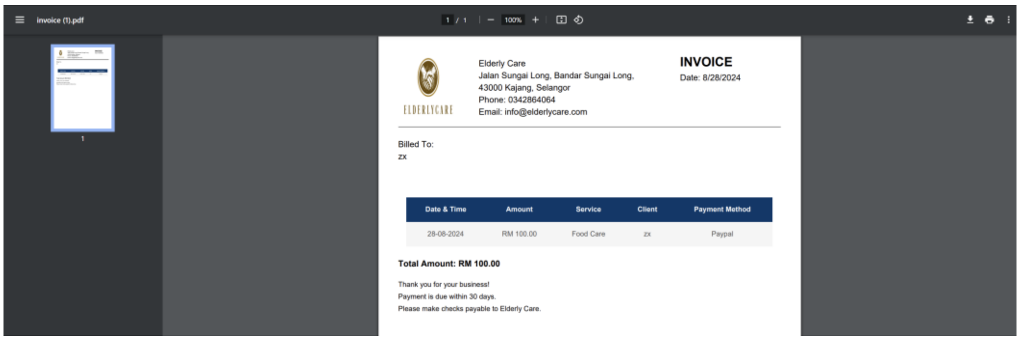


Payment Received History

Select	Date & Time	Amount	Service Provided	Client	Payment Method
✓	30-08-2024	RM 120.00	Home Health Care	zx	Credit Card
☐	30-08-2024	RM 100.00	Memory Care	zx	Credit Card
☐	30-08-2024	RM 100.00	Food Care	zx	Bank Transfer
☐	30-08-2024	RM 100.00	Memory Care	zx	Paypal

Figure 6.55: Payment History Page

This Payment History Page will show all the transactions made by the client for record-keeping purposes. On this page, the caregiver can search for the transaction history during a specific date by applying the date filter. The caregiver may also select one or more transactions to generate an invoice or export to CSV.


Elderly Care
Jalan Sungai Long, Bandar Sungai Long,
43000 Kajang, Selangor
Phone: 0342864064
Email: info@elderlycare.com

INVOICE
Date: 8/28/2024

Billed To:
zx

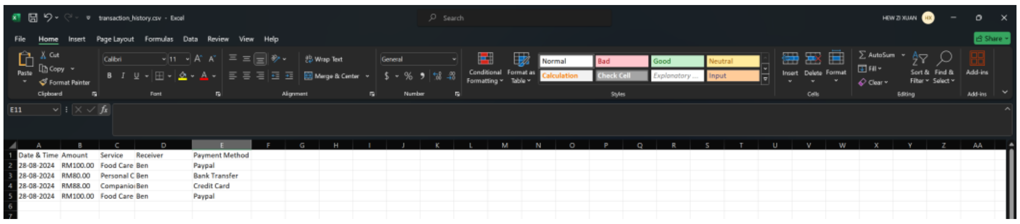
Date & Time	Amount	Service	Client	Payment Method
28-08-2024	RM 100.00	Food Care	zx	Paypal

Total Amount: RM 100.00

Thank you for your business!
Payment is due within 30 days.
Please make checks payable to Elderly Care.

Figure 6.56: Invoice of a Transaction

The caregiver can select one or more transactions and generate an invoice by clicking the "Generate Invoice" button. Upon clicking the button, a PDF invoice will be automatically downloaded, and formatted as shown in Figure 6.56 provided above.

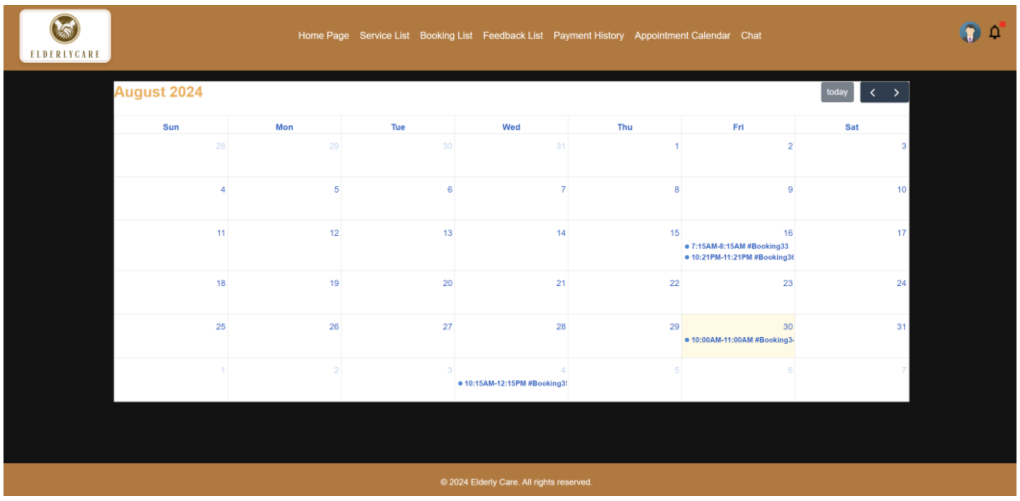


	A	B	C	D	E
1	Date & Time	Amount	Service	Receiver	Payment Method
2	28-08-2024	RM100.00	Food Care	Ben	Paypal
3	28-08-2024	RM80.00	Personal C	Ben	Bank Transfer
4	28-08-2024	RM88.00	Companio	Ben	Credit Card
5	28-08-2024	RM100.00	Food Care	Ben	Paypal

Figure 6.57: CSV File of a Transaction

The client can select one or more transactions and export them to a CSV file by clicking the "Export CSV" button. Once the button is clicked, a CSV file will be automatically downloaded, and formatted as illustrated in Figure 6.57 provided above.

6.4.5.6 Appointment Calendar Page

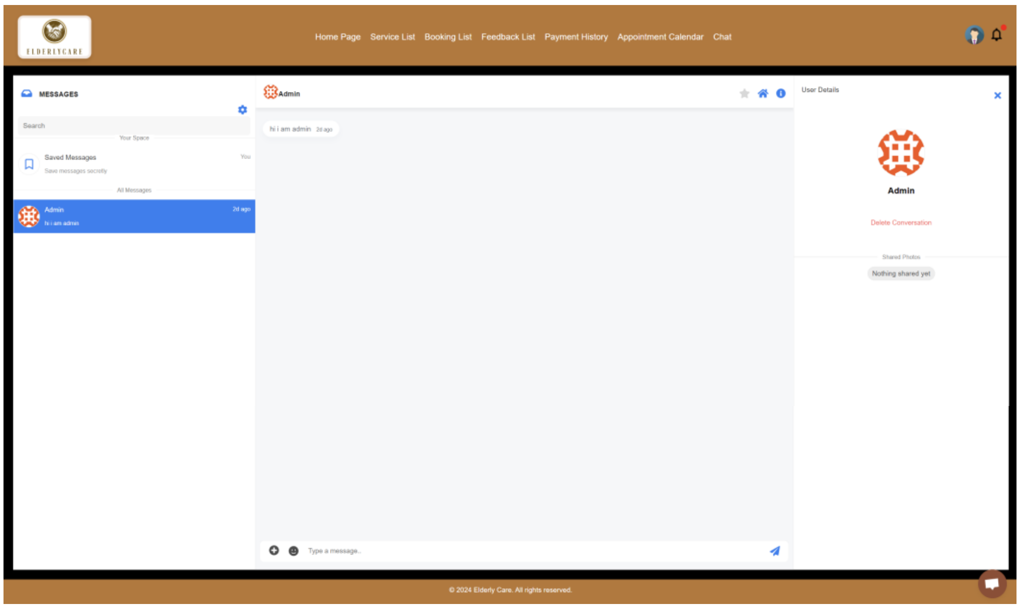


Sun	Mon	Tue	Wed	Thu	Fri	Sat
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16 ● 7:15AM-8:15AM #Booking33 ● 10:21PM-11:21PM #Booking3	17
18	19	20	21	22	23	24
25	26	27	28	29	30 ● 10:00AM-11:00AM #Booking3	31
1	2	3	4 ● 10:15AM-12:15PM #Booking3	5	6	7

Figure 6.58: Appointment Calendar Page

On the Appointment Calendar Page, caregivers can view all their scheduled appointment dates and times for added convenience. Each appointment is represented as a clickable link. When a caregiver clicks on it, they will be navigated to the booking details page where they can access a more detailed view of the specific booking.

6.4.5.7 Chat Page



Admin 2d ago
hi i am admin

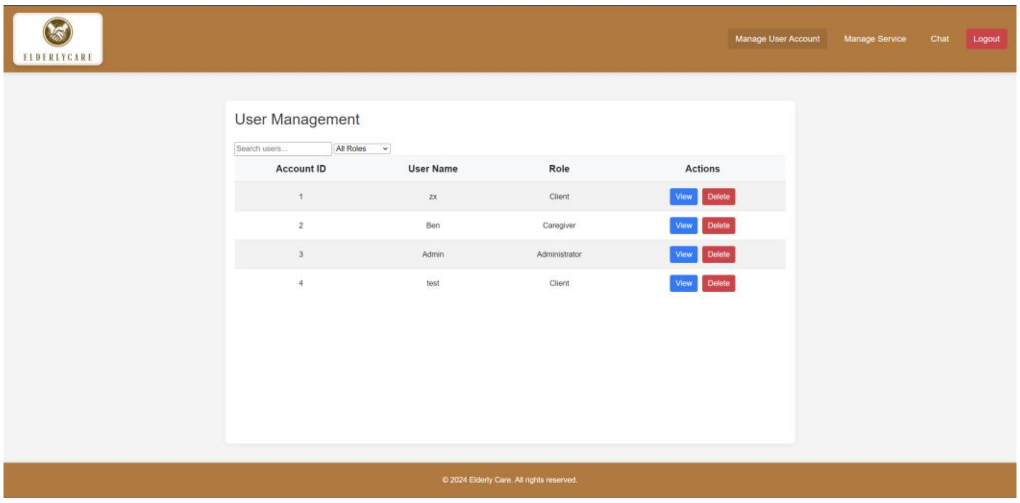
hi i am admin 2d ago

Figure 6.59: Chat Page

The caregiver can communicate with all the clients and the admin by using this chat channel. The caregiver is also allowed to send images or emojis using this chat channel.

6.4.6 Administrator

6.4.6.1 Manage User Account Page



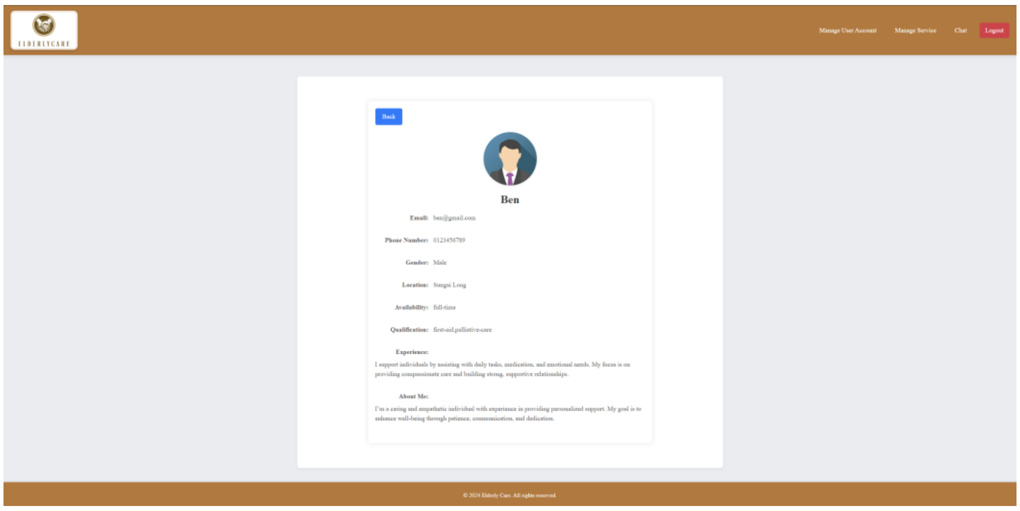
User Management

Account ID	User Name	Role	Actions
1	zx	Client	View Delete
2	Ben	Caregiver	View Delete
3	Admin	Administrator	View Delete
4	test	Client	View Delete

© 2024 Elderly Care. All rights reserved.

Figure 6.60: Manage User Account Page

After admins login to the platform, they will be redirected to this page. On this page, admins can manage user accounts by viewing the user details and deleting the user account. Admin can also search the users by entering the user's name in the "search users" search box and filtering the users by roles. When an admin searches for users by entering a name into the "search users" box or applies filters based on user roles, AJAX automatically sends the search and filter requests to the server. The server processes these requests and responds with the relevant data. The page then updates dynamically to display the filtered user list and search results without requiring a full page reload.



Ben

Email: ben@gmail.com

Phone Number: 0123456789

Gender: Male

Location: Sungai Long

Availability: full-time

Qualification: first-aid,palliative-care

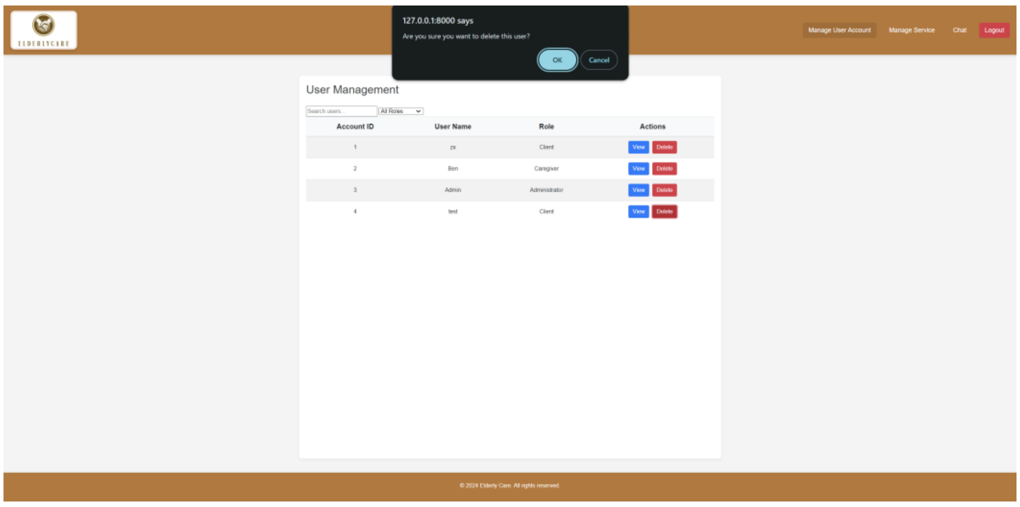
Experience:
I support individuals by assisting with daily tasks, medication, and emotional needs. My focus is on providing compassionate care and building strong, supportive relationships.

About Me:
I'm a caring and empathetic individual with experience in providing personalized support. My goal is to enhance well-being through patience, communication, and dedication.

© 2024 Elderly Care. All rights reserved.

Figure 6.61: User's Details Page

After clicking on the "View" button on the Manage User Account Page, the admin will be redirected to this User's Details Page. This page will show the details of the user.



127.0.0.1:8000 says

Are you sure you want to delete this user?

User Management

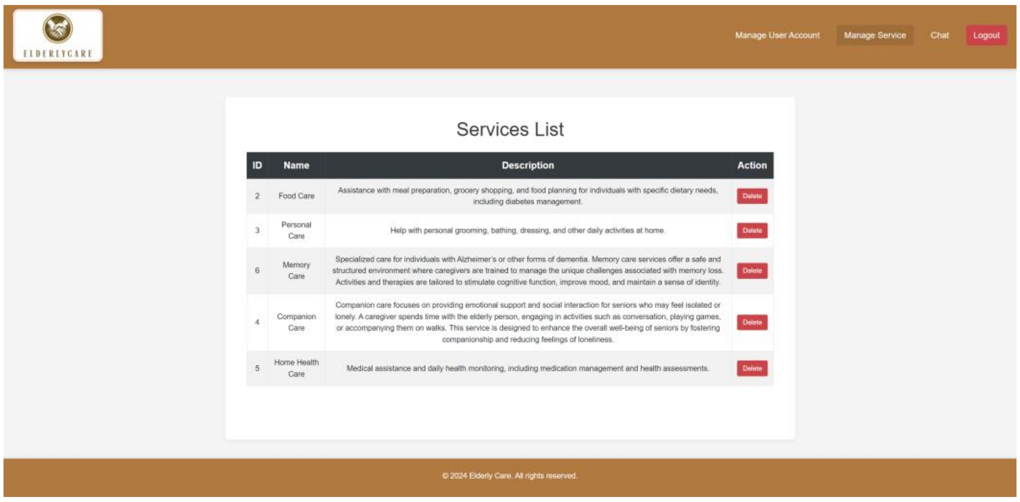
Account ID	User Name	Role	Actions
1	zx	Client	View Delete
2	Ben	Caregiver	View Delete
3	Admin	Administrator	View Delete
4	test	Client	View Delete

© 2024 Elderly Care. All rights reserved.

Figure 6.62: Alert Message When Delete a User Account

An alert message will show when the user clicks on the "Delete" button. After users confirm the deletion, the user account will be removed from the database and a success message will show.

6.4.6.2 Manage Service Page



Services List

ID	Name	Description	Action
2	Food Care	Assistance with meal preparation, grocery shopping, and food planning for individuals with specific dietary needs, including diabetes management.	Delete
3	Personal Care	Help with personal grooming, bathing, dressing, and other daily activities at home.	Delete
6	Memory Care	Specialized care for individuals with Alzheimer's or other forms of dementia. Memory care services offer a safe and structured environment where caregivers are trained to manage the unique challenges associated with memory loss. Activities and therapies are tailored to stimulate cognitive function, improve mood, and maintain a sense of identity.	Delete
4	Companion Care	Companion care focuses on providing emotional support and social interaction for seniors who may feel isolated or lonely. A caregiver spends time with the elderly person, engaging in activities such as conversation, playing games, or accompanying them on walks. This service is designed to enhance the overall well-being of seniors by fostering companionship and reducing feelings of loneliness.	Delete
5	Home Health Care	Medical assistance and daily health monitoring, including medication management and health assessments.	Delete

© 2024 Elderly Care. All rights reserved.

Figure 6.63: Manage Service Page

Admin can view all the services and perform deletion on the service on this page. If the service has been deleted, a notification will be sent to the caregiver to notify them that the service has been deleted.

6.4.6.3 Chat Page

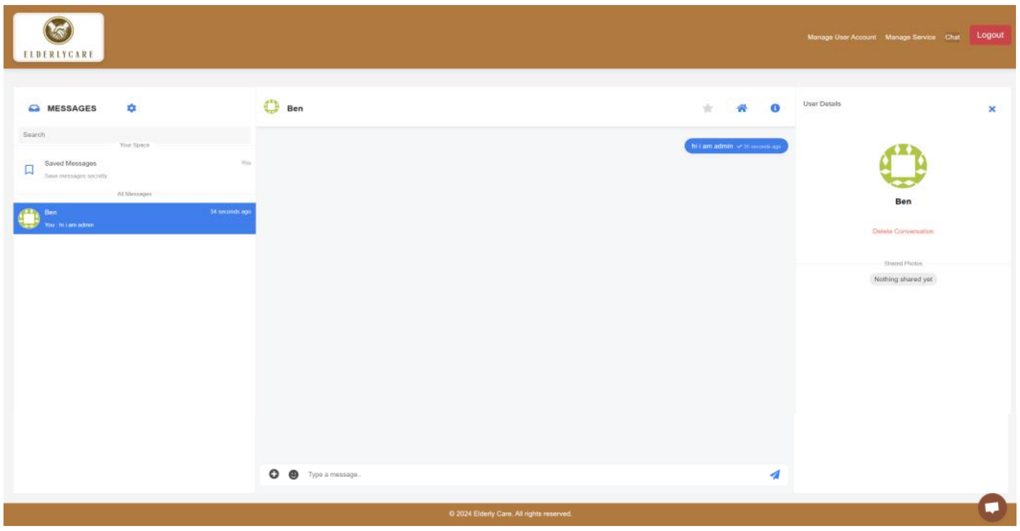


Figure 6.64: Chat Page

Admin can communicate with all the clients and caregivers by using this chat channel. The admin is also allowed to send images or emojis using this chat channel.

6.5 Implementation Issues and Challenges

During the development of this project, I encountered several problems and challenges. The first problem I encountered was the creation of service dates and time slots in the Add Service function. First, the Add Service function I designed just let the customer select the date and time to book the service. Later, I felt that this was illogical because it just let the customer select the date and time slot they wanted to use the service without considering whether the caregiver was available. After consideration, I modified the logic to let the caregivers declare their service duration first and then I create time slots for each caregiver based on this duration. For example, if caregiver A declares her/his service time to be 2 hours, then I will set the time slots to 12 pm to 2 pm, 2 pm to 4 pm, and so on. After implementation, I still felt that it was not very convenient because the caregivers could not manage their time slots themselves. Finally, I thought of a better idea, which is to let the caregivers create the service time slots themselves. Caregivers can first select a service date, and then they can add time slots for that date. This makes the creation of time slots completely managed by the caregivers themselves, which is the most convenient. After I implemented this, there were still some minor errors, that is, after the customer booked the time slot, the caregiver could still update the time slot, which made the data uncontrollable. So, I modified the update service to check if the time slot was already booked, making it impossible to modify it. Finally, this time slot issue was solved, but it took me a long time because I tried many different ways to create the time slot.

The second challenge was the implementation of AI service recommendations. Since this was not planned for FYP1 but recommended by my supervisor, I spent a lot of time researching and learning how to apply AI to recommendation systems. To avoid spending more time testing the model, I found a better way, which is to use NLP methods to process text and make intelligent recommendations.

CHAPTER 7
SYSTEM TESTING

7.1 Introduction

This chapter is divided into five sections. Section 7.2 provides an overview of the types of testing that will be performed in this project. Section 7.3 will explain how to plan and execute unit testing. Section 7.4 will outline how to plan and execute feature testing. Section 7.5 will discuss how to plan and execute black-box testing. The final section, 7.6, discusses the planning and execution process for UAT.

7.2 Testing Types

Once the project is implemented, four types of testing will be performed. Unit and Feature testing will be automatically performed using Laravel's built-in tools to ensure that individual components and integrations function properly. For Unit and Feature tests, they will be automatically run in GitHub Actions after they are written. Additionally, manual black-box testing will be performed to evaluate the general functionality of the application from the user's perspective without considering the internal code structure. Finally, end users themselves conduct User Acceptance Testing (UAT) to verify that the program meets their needs and operates as expected. By combining automated and manual testing methods, the approach ensures thorough coverage and reliability which addressing both technical performance and user satisfaction.

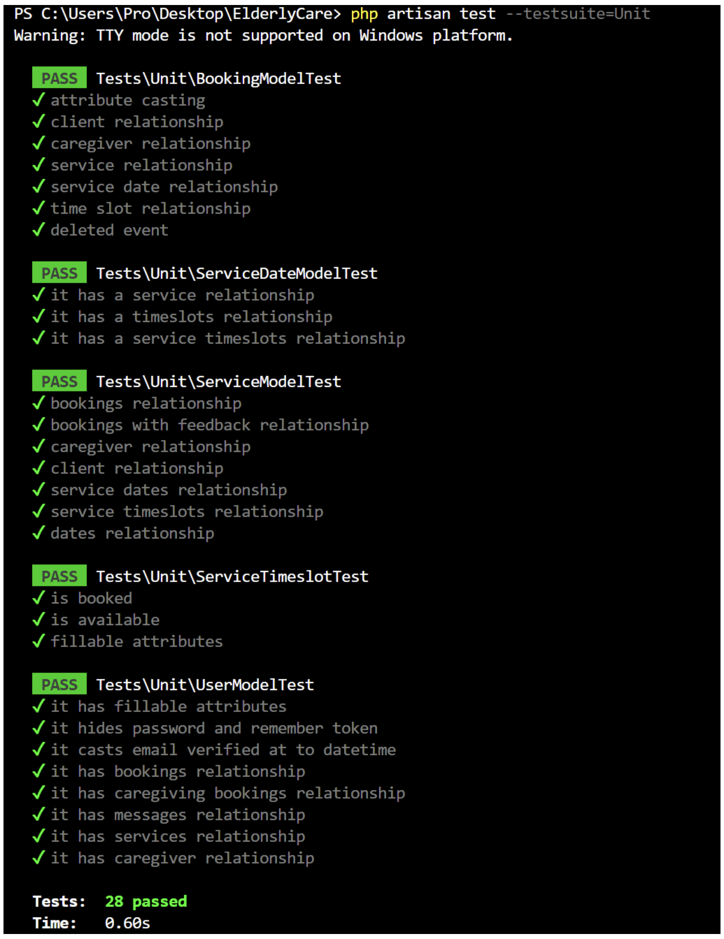
7.3 Unit Test

Unit testing is an important part of software testing where each component or functionality of a software program is tested independently. This strategy ensures that each software unit functions properly. In this unit testing, all models will be tested through unit testing. This ensures that individual logic in the model such as data operations and relationships work properly in isolation.

In this unit testing, 5 unit modules with a total of 28 unit test cases were conducted. All 28 test cases are passed during the unit testing. The testing code of this unit test will be attached in Appendix B.

Table 7.1: Unit Test Result

Unit Test Module Name	Number of unit test cases in the module	Number of passed unit test cases in the module
BookingModelTest	7	7
ServiceDateModelTest	3	3
ServiceModelTest	7	7
ServiceTimeslotModelTest	3	3
UserModelTest	8	8
Total	28	28



```
PS C:\Users\Pro\Desktop\ElderlyCare> php artisan test --testsuite=Unit
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Unit\BookingModelTest
  ✓ attribute casting
  ✓ client relationship
  ✓ caregiver relationship
  ✓ service relationship
  ✓ service date relationship
  ✓ time slot relationship
  ✓ deleted event

  PASS  Tests\Unit\ServiceDateModelTest
  ✓ it has a service relationship
  ✓ it has a timeslots relationship
  ✓ it has a service timeslots relationship

  PASS  Tests\Unit\ServiceModelTest
  ✓ bookings relationship
  ✓ bookings with feedback relationship
  ✓ caregiver relationship
  ✓ client relationship
  ✓ service dates relationship
  ✓ service timeslots relationship
  ✓ dates relationship

  PASS  Tests\Unit\ServiceTimeslotTest
  ✓ is booked
  ✓ is available
  ✓ fillable attributes

  PASS  Tests\Unit\UserModelTest
  ✓ it has fillable attributes
  ✓ it hides password and remember token
  ✓ it casts email verified at to datetime
  ✓ it has bookings relationship
  ✓ it has caregiving bookings relationship
  ✓ it has messages relationship
  ✓ it has services relationship
  ✓ it has caregiver relationship

  Tests:  28 passed
  Time:   0.60s
```

Figure 7.1: Unit Test Result

The figure above shows that after executing the 'php artisan test --testsuite=Unit' command, all unit tests passed.

7.4 Feature Test

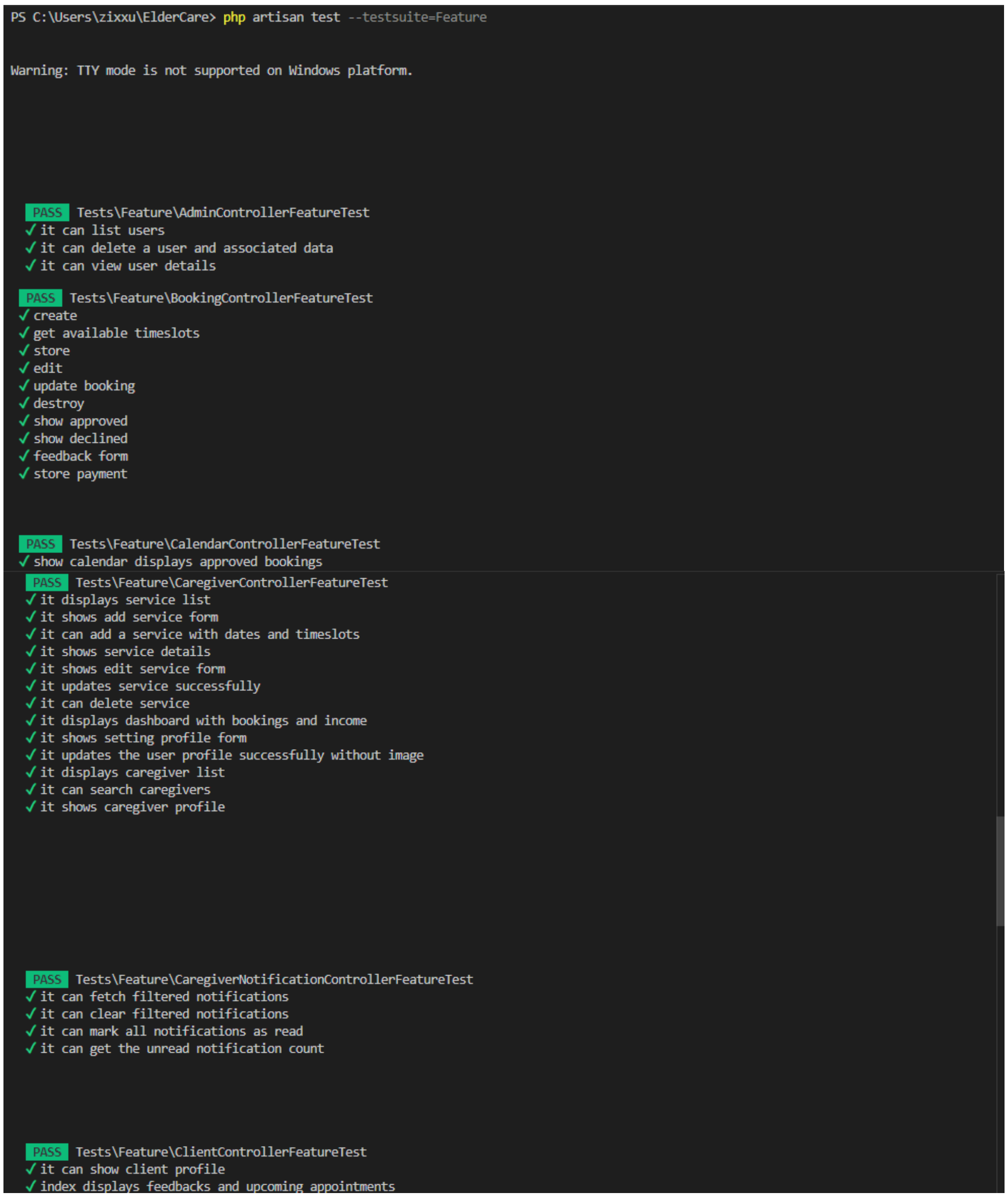
Feature test involves testing large blocks of code that typically complete HTTP requests and responses which may include multiple units of code that work together to perform a task. For example, handling a form submission or returning a view. In this case, all controllers will be tested with this Feature test. This will verify that the application's endpoints, routes, and user interactions function as expected, covering the flow from request to response.

In this feature test, 13 feature modules with a total of 56 feature test cases were conducted. All 56 test cases are passed during the feature testing. The testing code of this feature test will be attached in Appendix B.

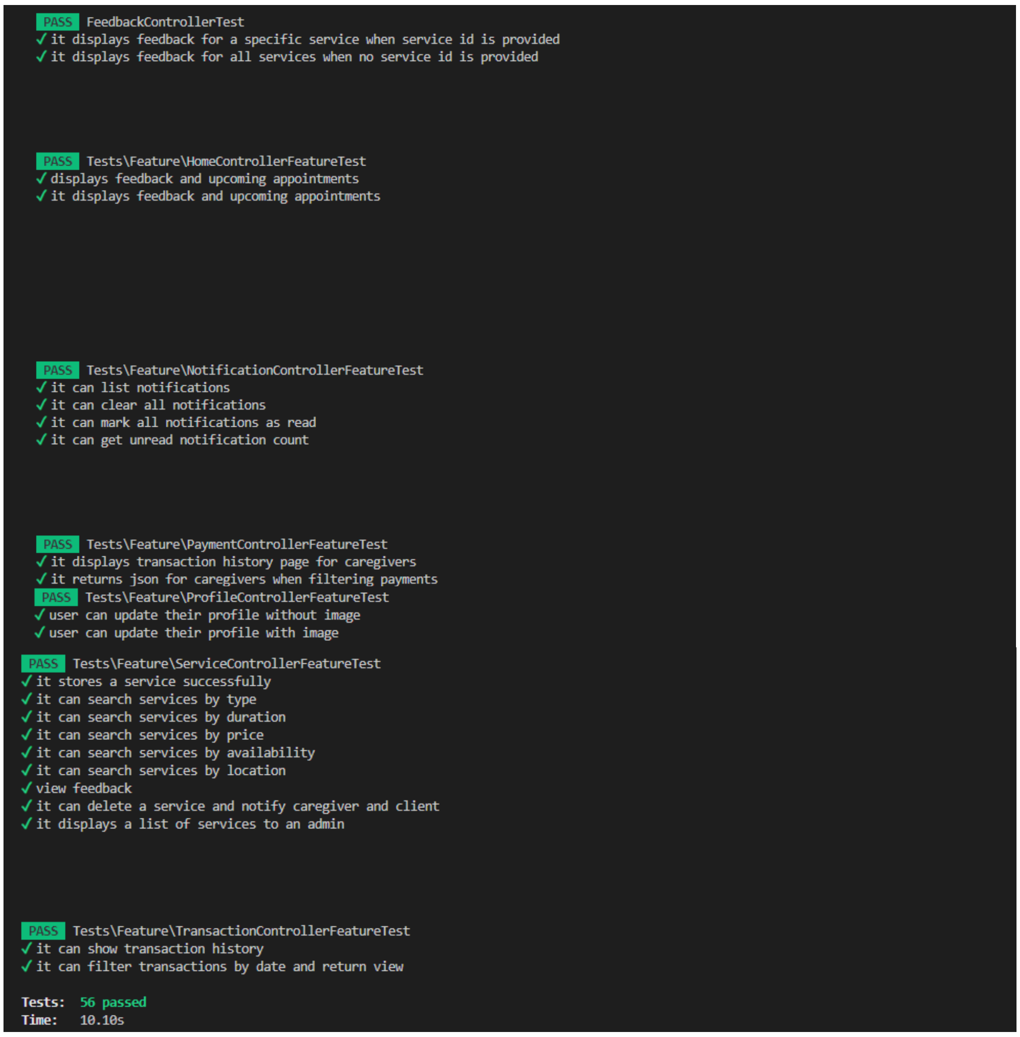
Table 7.2: Feature Test Result

Feature Test Module Name	Number of feature test cases in the module	Number of passed feature test cases in the module
AdminControllerFeatureTest	3	3
BookingControllerFeatureTest	10	10
CalendarControllerFeatureTest	1	1
CaregiverControllerFeatureTest	13	13
CaregiverNotificationControllerFeatureTest	4	4
ClientControllerFeatureTest	2	2
FeedbackControllerFeatureTest	2	2
HomeControllerFeatureTest	2	2
NotificationControllerFeatureTest	4	4
PaymentControllerFeatureTest	2	2
ProfileControllerFeatureTest	2	2
ServiceControllerFeatureTest	9	9

TransactionControllerFeatureTest	2	2
Total	56	56



```
PS C:\Users\zixxu\ElderCare> php artisan test --testsuite=Feature


Warning: TTY mode is not supported on Windows platform.




   PASS  Tests\Feature\AdminControllerFeatureTest
  ✓ it can list users
  ✓ it can delete a user and associated data
  ✓ it can view user details

  PASS  Tests\Feature\BookingControllerFeatureTest
  ✓ create
  ✓ get available timeslots
  ✓ store
  ✓ edit
  ✓ update booking
  ✓ destroy
  ✓ show approved
  ✓ show declined
  ✓ feedback form
  ✓ store payment


  PASS  Tests\Feature\CalendarControllerFeatureTest
  ✓ show calendar displays approved bookings
   PASS  Tests\Feature\CaregiverControllerFeatureTest
   ✓ it displays service list
   ✓ it shows add service form
   ✓ it can add a service with dates and timeslots
   ✓ it shows service details
   ✓ it shows edit service form
   ✓ it updates service successfully
   ✓ it can delete service
   ✓ it displays dashboard with bookings and income
   ✓ it shows setting profile form
   ✓ it updates the user profile successfully without image
   ✓ it displays caregiver list
   ✓ it can search caregivers
   ✓ it shows caregiver profile




   PASS  Tests\Feature\CaregiverNotificationControllerFeatureTest
   ✓ it can fetch filtered notifications
   ✓ it can clear filtered notifications
   ✓ it can mark all notifications as read
   ✓ it can get the unread notification count


   PASS  Tests\Feature\ClientControllerFeatureTest
   ✓ it can show client profile
   ✓ index displays feedbacks and upcoming appointments
```



```
 PASS  FeedbackControllerTest
✓ it displays feedback for a specific service when service id is provided
✓ it displays feedback for all services when no service id is provided



 PASS  Tests\Feature\HomeControllerFeatureTest
✓ displays feedback and upcoming appointments
✓ it displays feedback and upcoming appointments



 PASS  Tests\Feature\NotificationControllerFeatureTest
✓ it can list notifications
✓ it can clear all notifications
✓ it can mark all notifications as read
✓ it can get unread notification count



 PASS  Tests\Feature\PaymentControllerFeatureTest
✓ it displays transaction history page for caregivers
✓ it returns json for caregivers when filtering payments
 PASS  Tests\Feature\ProfileControllerFeatureTest
✓ user can update their profile without image
✓ user can update their profile with image
 PASS  Tests\Feature\ServiceControllerFeatureTest
✓ it stores a service successfully
✓ it can search services by type
✓ it can search services by duration
✓ it can search services by price
✓ it can search services by availability
✓ it can search services by location
✓ view feedback
✓ it can delete a service and notify caregiver and client
✓ it displays a list of services to an admin



 PASS  Tests\Feature\TransactionControllerFeatureTest
✓ it can show transaction history
✓ it can filter transactions by date and return view

Tests:  56 passed
Time:   10.10s
```

Figure 7.2: Feature Test Result

The figure shows that all unit tests passed after executing the 'php artisan test --testsuite=Feature' command.

7.5 Black Box Test

7.5.1 Black Box Test Cases for Login and Registration

Table 7.3: Login and Registration Test Cases

Test Case No	Test Scenario	Input	Expected Result	Actual Result	Pass / Fail
TC001	Registration with valid data	Input valid credentials	An email verification will	As expected	Pass

			send to registered email		
TC002	Registration with already-used email	Input invalid credentials	Error message should be displayed and the registration is not completed	As expected	Pass
TC003	Login with valid data	Input valid credentials	User will be redirect to home page	As expected	Pass
TC004	Login with invalid data	Input invalid credentials	Error message should be displayed and the user remains on the login page	As expected	Pass

7.5.2 Black Box Test Cases for Client Perspective

Table 7.4: Client Module Test Cases

Test Case No	Test Scenario	Input	Expected Result	Actual Result	Pass / Fail
TC001	Book a Service	Select a date and timeslot then click the "Book" button	Booking is created and client is redirected to the Pending Booking page	As expected	Pass
TC002	View booking	Select a booking and click on the “View” button	Client will be redirected to the booking detail page	As expected	Pass
TC003	Update Booking	Select a new date and timeslot then	Client will be redirected to the Pending Booking	As expected	Pass

		click on the “Update” button	Page with a success message		
TC004	Delete Booking	Select a booking from Pending Booking Page and click on “Cancel” button	Alert message shown first and once confirm to delete, a success message shown	As expected	Pass
TC005	Make payment	Select a booking from the Accepted Booking Page and click on the “Pay” button	The payment modal drops down. Client can select a payment method and click the "Proceed" button. After that, a success message will be displayed	As expected	Pass
TC006	Provide comment	Select a booking from the Accepted Booking Page and click on the “Comment” button	The comment modal drops down. Client can rank and comment on the service and click on the "Submit" button. After that, a success message will be displayed	As expected	Pass
TC007	Search Service	Apply filter criteria	The services that match the applied filters are shown in the Search Page	As expected	Pass
TC008	Clear Search Filter	Apply a filter and then click	Search results are reset to the default	As expected	Pass

		the "Clear Filter" button	state, showing all available services		
TC009	Get Service Suggestion	Click on the “Click here for suggestion” button and send a message describing the service you want	If the platform has a matching service, a recommended service is sent back to the user. If no, an error message is sent indicating that no service matches the description	As expected	Pass
TC010	View Feedback	Select a service in Search Page and click on “View Feedback” button	A feedback modal will pops out and displaying all feedbacks related to the selected service	As expected	Pass
TC011	Apply Date Filter in Transaction History Page	Select a date range using the "From" and "To" date pickers then click on the “Apply Filters” button	The transaction history list is filtered to show only transactions that occurred within the selected date range	As expected	Pass
TC012	Apply Clear Filter on Transaction	Apply any filter date then click on the “Clear Filter” button	The transaction history page resets to its default state, displaying all transactions	As expected	Pass

	History Page		without any filters applied		
TC013	Generate Invoice	Select a transaction history and click on the “Generate Invoice” button	An invoice PDF is automatically downloaded to the user's PC	As expected	Pass
TC014	Export CSV	Select a transaction history and click on the “Export CSV” button	A CSV file is automatically downloaded to the user's PC	As expected	Pass
TC015	Send a message	Fill out the "Contact Us" form and click the "Send Message" button	A success message is shown indicating that the message was sent successfully	As expected	Pass
TC016	Edit profile with valid data	Input valid credentials and click on the “Save Changes” button	A success message is shown and the profile is updated	As expected	Pass
TC017	Edit profile with invalid data	Input invalid credentials and click on the “Save Changes” button	Error message shown indicating the profile is not updated	As expected	Pass
TC018	Chat in the Chat channel	Send a message to any user in the chat channel	A double tick appears next to the message	As expected	Pass

			indicating that it has been successfully sent		

7.5.3 Black Box Test Cases for Caregiver Perspective

Table 7.5: Caregiver Module Test Cases

Test Case No	Test Scenario	Input	Expected Result	Actual Result	Pass / Fail
TC001	Add service with valid data	Input valid credentials	Service added and caregiver is redirected to the service list page with a success message	As expected	Pass
TC002	Add service with invalid data	Input invalid credentials	Error message shown	As expected	Pass
TC003	View service	Select a service and click on the “View” button	The caregiver is redirected to the service detail page and all details of the service are displayed	As expected	Pass
TC004	Update service with valid data	Input valid credentials	Service updated and caregiver is redirected to the service list page with a success message	As expected	Pass
TC005	Update service with invalid data	Input invalid credentials	Error message shown	As expected	Pass

TC006	Delete service	Select a service and click on the “Delete” button	A alert message shown and upon confirmation, successful message shown	As expected	Pass
TC007	View client profile	Select a booking from the Booking List Page and click on the “View Profile” button	Caregiver is redirected to the client detail page and all the details of the client shown	As expected	Pass
TC008	View booking details	Select a booking from the Booking List Page and click on the “View Booking Details” button	Caregiver is redirected to the booking detail page and all the details about the booking shown	As expected	Pass
TC009	Accept booking	Select a booking from Pending Booking Page and	The booking is accepted, and the success message shown	As expected	Pass

		click on the “Accept” button			
TC010	Decline booking	Select a booking from Pending Booking Page and click on the “Decline” button	The booking is declined, and the success message shown	As expected	Pass
TC011	Filter feedback	Apply a filter by service	Feedback related to the selected service is displayed	As expected	Pass
TC012	Apply Date Filter in Payment History Page	Select a date range using the "From" and "To" date pickers then click on the “Apply Filters” button	The payment received list is filtered to show only history that occurred within the selected date range	As expected	Pass
TC013	Apply Clear Filter on Payment History Page	Apply any filter date then click on the	The payment received history page resets to its default state, displaying all	As expected	Pass

		“Clear Filter” button	history without any filters applied		
TC014	Generate Invoice	Select a payment received history and click on the “Generate Invoice” button	An invoice PDF is automatically downloaded to the user's PC	As expected	Pass
TC015	Export CSV	Select a payment received history and click on the “Export CSV” button	A CSV file is automatically downloaded to the user's PC	As expected	Pass
TC016	Edit profile with valid data	Input valid credentials and click on the “Save” button	A success message is shown and the profile is updated	As expected	Pass
TC017	Edit profile with invalid data	Input invalid credentials and click on the	Error message shown indicating the profile is not updated	As expected	Pass

		“Save” button			
TC018	Chat in the Chat channel	Send a message to any user in the chat channel	A double tick appears next to the message indicating that it has been successfully sent	As expected	Pass

7.5.4 Black Box Test Cases for Admin Perspective

Table 7.6: Admin Module Test Case

Test Case No	Test Scenario	Input	Expected Result	Actual Result	Pass / Fail
TC001	Search user by name	Enter a name in to the ‘Search users’ text box	The user list updates to display only the users whose name matches the input. If no users match, the list should be empty	As expected	Pass
TC002	Filter user list by role	Select a role from the dropdown filter	The user list updates to display only the users associated with the selected role. If no users match, the list should be empty	As expected	Pass
TC003	View user details	Select a user from the list and click the	Admin is redirected to user detail page and the detail of the user are shown	As expected	Pass

		"View" button			
TC004	Delete user account	Select a user from the list and click the "Delete" button	An alert message prompts for confirmation. Upon confirming, the user account is deleted, and a success message is displayed	As expected	Pass
TC005	Delete Service	Select a service from the list and click the "Delete" button,	An alert message prompts for confirmation. Upon confirming, the service is deleted, and a success message is displayed	As expected	Pass
TC006	Chat in the Chat channel	Send a message to any user in the chat channel	A double tick appears next to the message indicating that it has been successfully sent	As expected	Pass

7.6 User Acceptance Test (UAT)

For User Acceptance Testing (UAT), I will use Google Forms to adapt questions to four key areas of the platform. This encompasses usability, functionality, visual design, and overall satisfaction. This systematic approach enables a full review of the platform by collecting focused feedback on key aspects. By focusing on these precise areas, I can accurately assess the platform's user experience, operational efficiency, and aesthetics. This will help me to assess which aspects of the platform are functioning well and which require further work or enhancement to satisfy user expectations and improve overall performance. For this test, 12 testers were selected to assess the platform's usability, functionality, and visual design, and overall satisfaction. Their responses have been analyzed and are presented in the following sections.

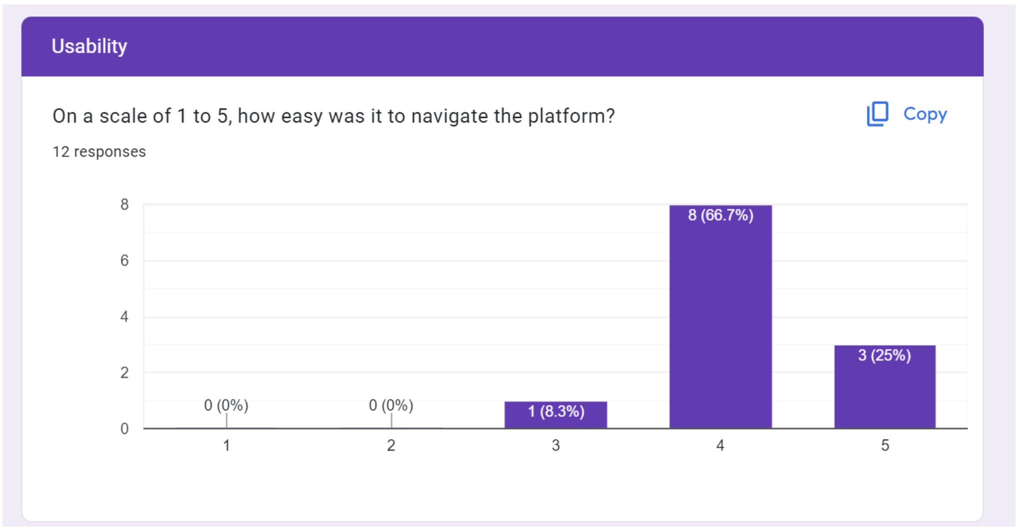


Figure 7.3: UAT Result of Question 1

On a scale of 1 to 5, the majority of users (66.7%) thought the platform's navigation was "4". This means it was reasonably straightforward to use. A quarter of customers gave it a "5" rating, indicating that they thought it to be extremely user-friendly. Only one person (8.3%) rated the navigation as "3 which implies it was moderately easy. No one rated it lower than 3 which suggests overall positive feedback on navigation.

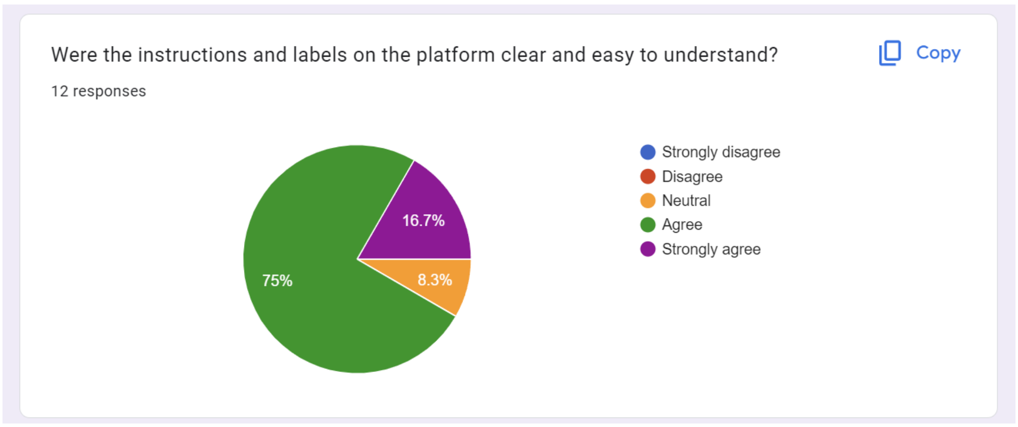


Figure 7.4: UAT Result of Question 2

The platform's instructions and labels were deemed straightforward and easy to comprehend by 91.7% of users who either "Agreed" (75%) or "Strongly Agreed" (16.7%). Although there was some slight space for improvement, just 8.3% of respondents were uncertain indicating that the clarity was well-received.

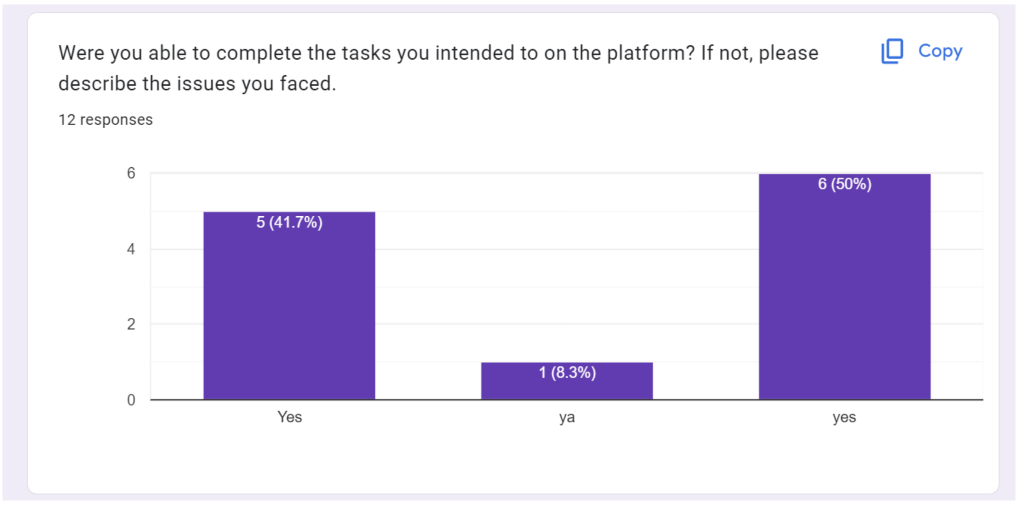


Figure 7.5: UAT Result of Question 3

100% of users confirmed they were able to complete the tasks they intended on the platform. This indicates that although all tasks were completed, there could have been minor issues that users faced.

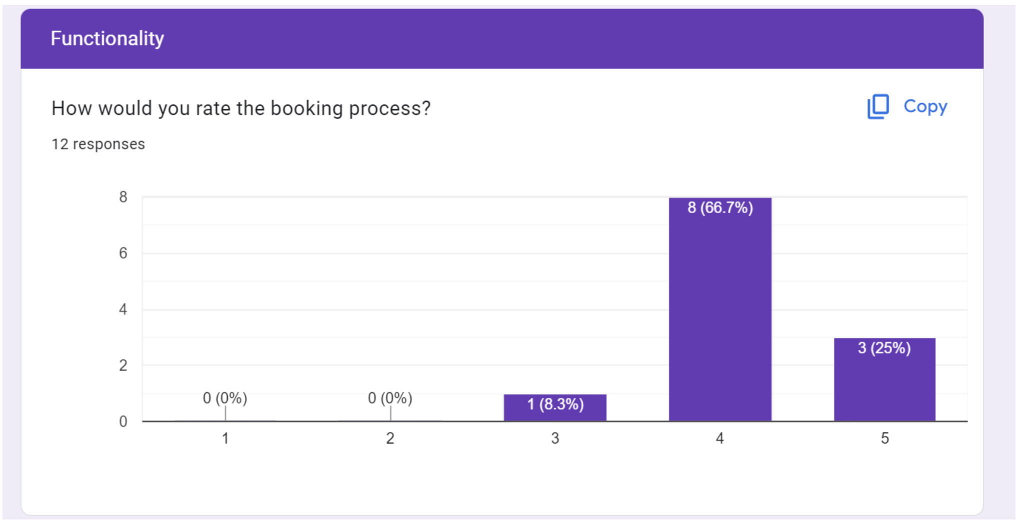


Figure 7.6: UAT Result of Question 4

The majority of users (66.7%) rated the process of booking a "4" which indicates overall satisfaction. 25% of the users rated it a "5". Only one user (8.3%) rated it a "3" suggesting that the booking process was moderately effective for them. This can conclude that the process of booking is functioning well.

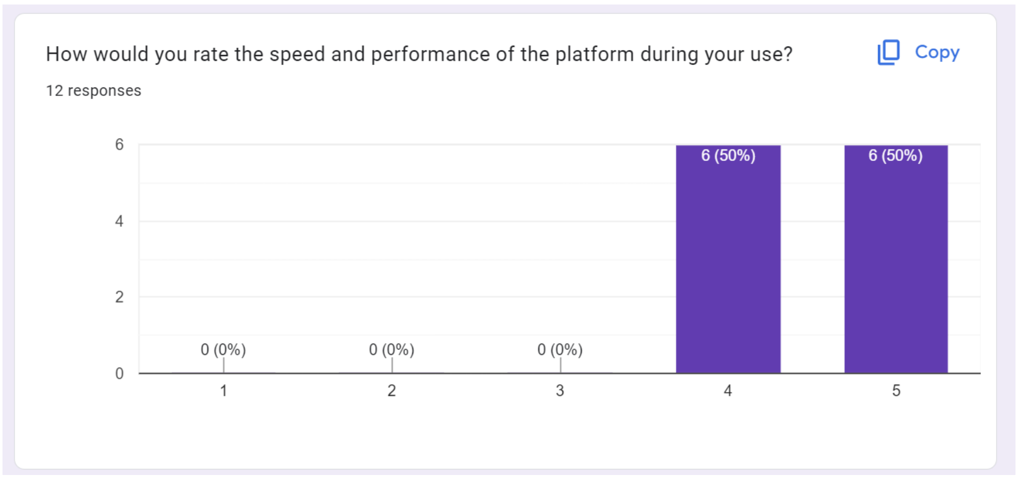
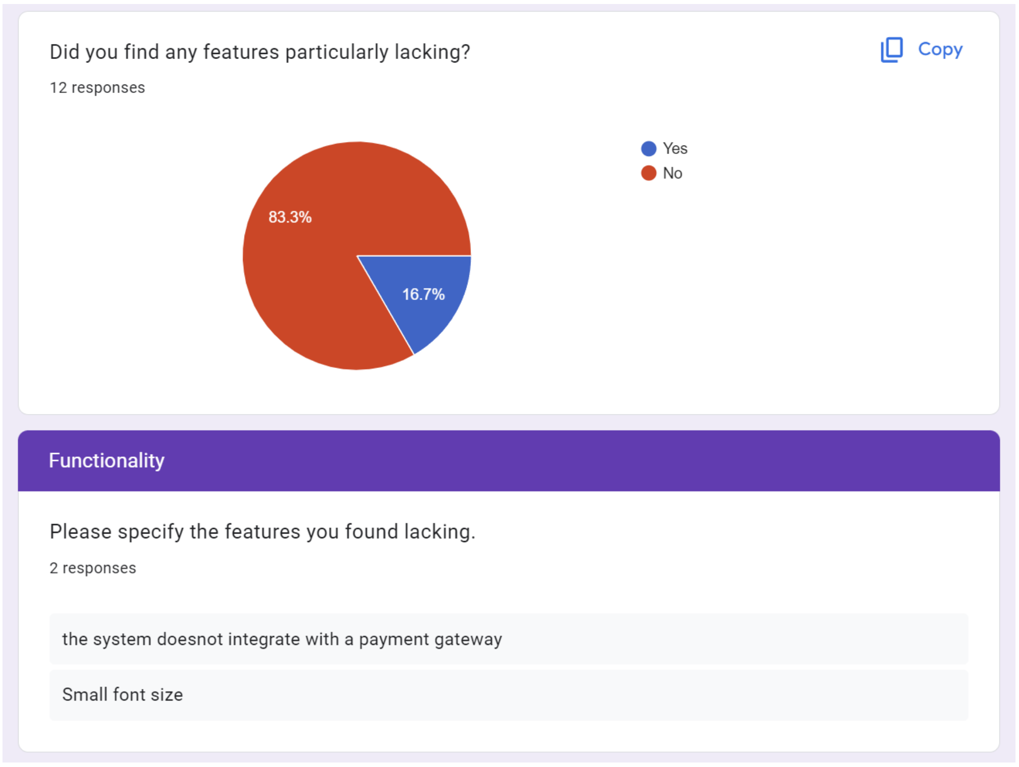


Figure 7.7: UAT Result of Question 5

Most users were satisfied with the platform’s performance. This can be proved with 50% rating it a “4” and 50% rating it a “5.” There were no ratings lower than a "4" which indicates that the platform operates with good speed and dependability.


Did you find any features particularly lacking?

12 responses

Functionality

Please specify the features you found lacking.

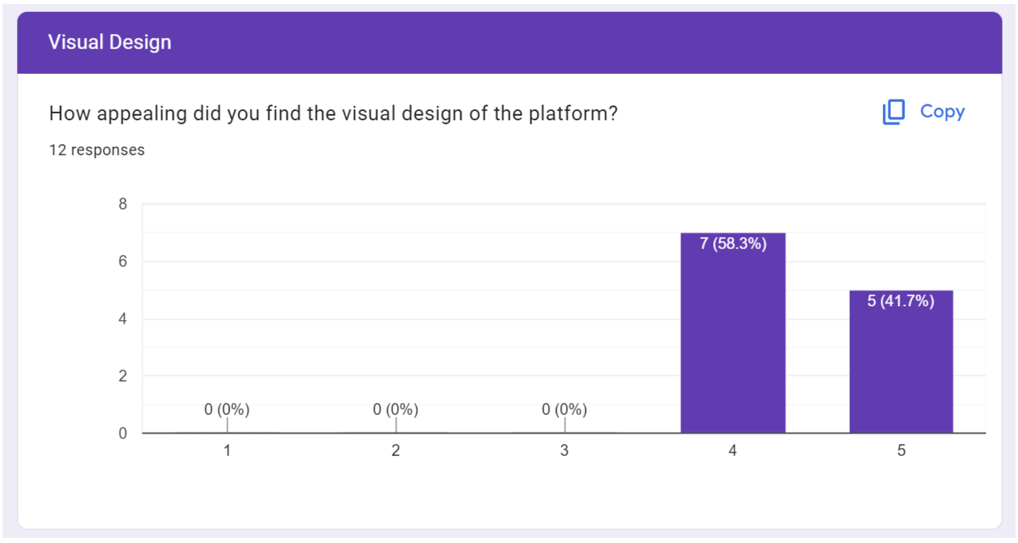
2 responses

the system doesnot integrate with a payment gateway

Small font size

Figure 7.8: UAT Result of Question 6

83.3% of users stated that they did not find any features lacking while 16.7% did mention a few missing aspects. One respondent pointed out the lack of integration with a payment gateway. This is crucial if transactions are part of the platform. Another respondent mentioned that the font size is small which might impact the platform's usability for some users. Even though the majority of users reported no missing functionality, implementing a payment gateway and changing the font size could improve the user experience.

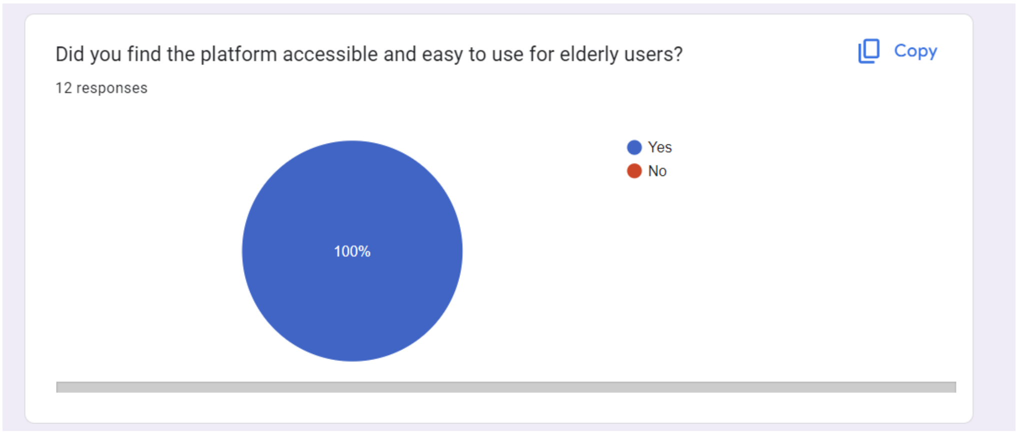

Visual Design

How appealing did you find the visual design of the platform?

12 responses

Figure 7.9: UAT Result of Question 7

58.3% of users rated the visual design of the platform as a "4" out of 5, while 41.7% rated it a "5." This indicates that users generally found the design appealing, with no ratings below "4." Although the platform's design has garnered positive feedback, there might be opportunities for minor improvements to further elevate its visual appeal.


Did you find the platform accessible and easy to use for elderly users?

12 responses

Figure 7.10: UAT Result of Question 8

All respondents thought the platform was user-friendly and accessible to elderly people. This is a positive result, showing that your platform is inclusive and user-friendly for different age groups, particularly elderly users.

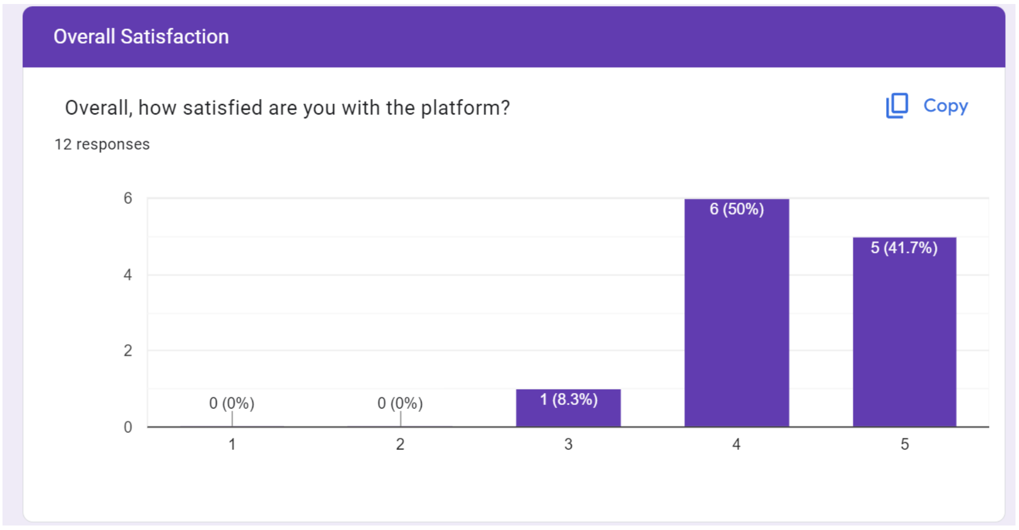


Figure 7.11: UAT Result of Question 9

50% of users rated their satisfaction with the platform as a "4" while 41.7% rated it a "5". Only one user (8.3%) gave a rating of "3" suggesting a moderate level of satisfaction for that user. Most users are quite satisfied with the platform, with only a small minority expressing moderate satisfaction.

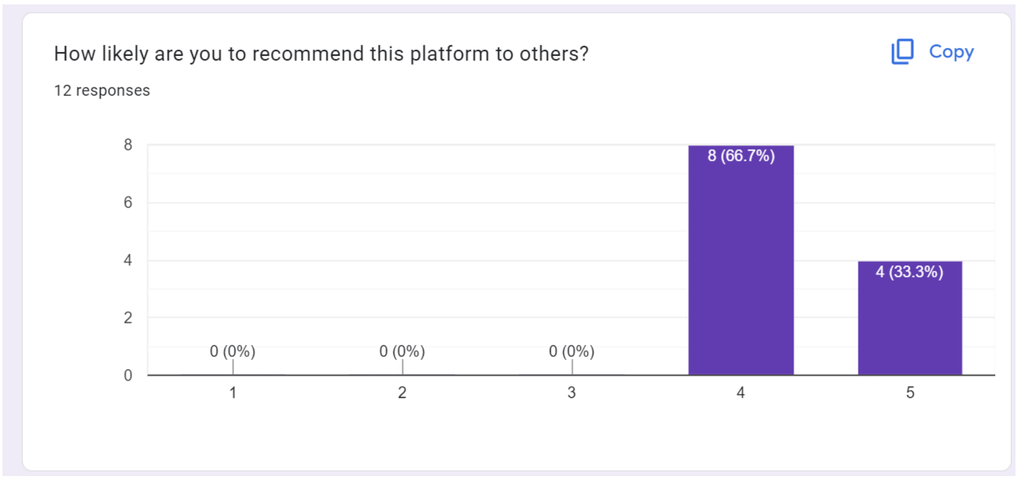
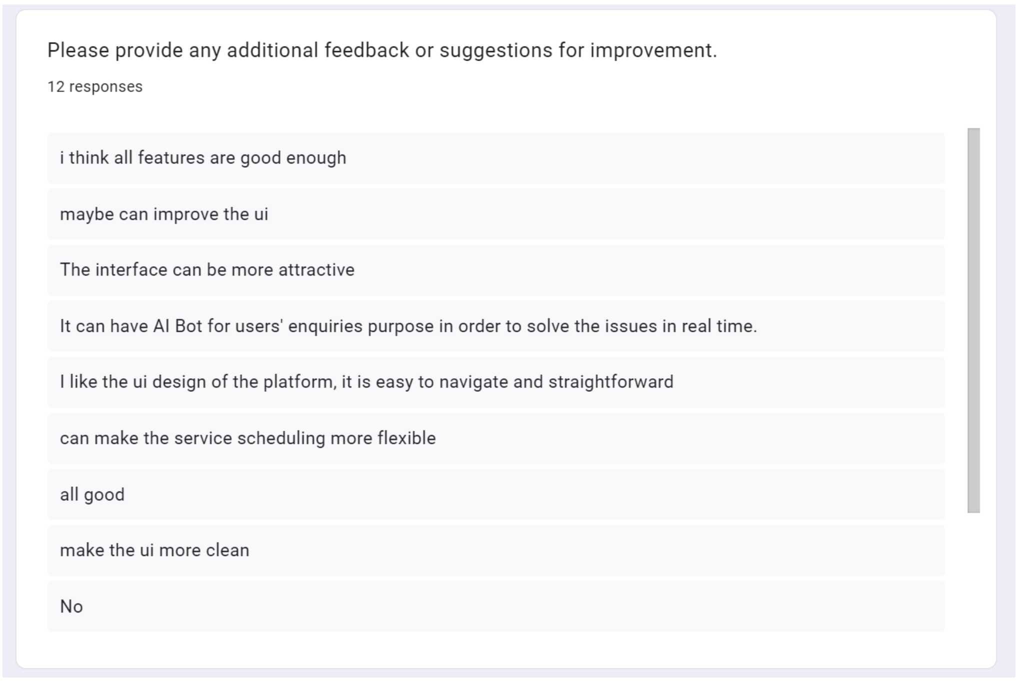


Figure 7.12: UAT Result of Question 10

66.7% of users rated their likelihood to recommend the platform to others as a "4" and 33.3% rated it a "5". No users rated it below "4" which indicates a strong endorsement from users. Users are likely to recommend the platform to others, demonstrating a positive overall experience.


Please provide any additional feedback or suggestions for improvement.

12 responses

i think all features are good enough

maybe can improve the ui

The interface can be more attractive

It can have AI Bot for users' enquiries purpose in order to solve the issues in real time.

I like the ui design of the platform, it is easy to navigate and straightforward

can make the service scheduling more flexible

all good

make the ui more clean

No

Figure 7.13: UAT Result of Question 11

Many users felt that "all features are good enough" and liked the platform's ease of navigation and straightforwardness. Some appreciated the UI design and felt the system could be made "more flexible." Some suggested enhancing the UI for a more modern look or making the UI more attractive. One user recommended adding a bot for real-time issue resolution while another mentioned making the scheduling system more flexible. There were also suggestions to make the UI cleaner and possibly enhance the overall appearance. While most feedback is positive, focusing on UI improvements and adding more advanced features like real-time support could further enhance user satisfaction.

CHAPTER 8
CONCLUSION AND RECOMMENDATION

8.1 Conclusion

In conclusion, this project successfully achieved the objectives outlined in Chapter 1 Section 1.3. These objectives were to:

1. To develop an integrated booking platform that allow registration for both clients and caregiver so that clients can find the services they need, and caregiver can promote their services.
2. To implement a chat channel that allows clients to communicate directly with caregivers for information exchange.
3. To design a platform with search function with filtering capabilities to generate caregiver matching lists based on user preferences and requirements.

The first objective of develop an integrated booking platform that allow registration for both clients and caregiver so that clients can find the services they need, and caregiver can promote their services has been successfully achieved. Users can choose their role either as a client or a caregiver. This role-based registration system ensures that clients can easily search for services while caregivers can promote their service to the client.

The second objective was accomplished through the implementation of a fully functional chat channel. This chat channel enables all the users in this platform to communicate with each other. Additionally, the users may use the “favorite user” function to mark a user as a favorite. This makes it easier to find and communicate with them quickly in the future.

For the third objective, a robust search functionality with filtering capabilities was integrated into the platform. This feature allows users to apply filters based on specific criteria such as location, service type, etc. After applying the filter, client can get all the match services. Moreover, the newly added service recommendation feature further enhances the platform. Clients can use this feature by describe their service requirements. The platform will then find the most suitable service in the database and return it to the client. This feature is particularly beneficial for clients who may not be familiar with all available services and require guidance.

8.2 Limitations

These are some of the limitations of this platform:

1. Limited Flexibility for Service Scheduling

   The platform requires caregivers to manually enter dates and timeslots for their services. For caregivers with complicated schedules, this process may be time-consuming. Scheduling problems and confusion may arise from overlapping appointments caused by a lack of automated scheduling and conflict detection.

2. Does Not Integrate with Payment Gateway

   The platform does not support direct payment processing through a payment gateway. This absence limits the ability to handle online transactions securely and efficiently, potentially leading to a fragmented user experience and reliance on less convenient alternative payment methods.

3. No Multilingual Support

   Accessibility for non-native English speakers is limited because the platform only supports English currently. This limitation affects user participation and inclusivity, particularly for users from diverse language backgrounds.

8.3 Recommendations

Based on the Section 8.2 Limitation, these are the recommendations and enhancement for future work:

1. Enhanced Scheduling Tools

   Implement automated or recurring scheduling options and conflict detection features to simplify service management for caregivers. This will reduce manual entry, prevent overlapping appointments, and improve scheduling accuracy.

2. Integrate Payment Gateway

   Add integration with a reliable payment gateway to enable secure and seamless online payment processing. This will enhance user experience by allowing direct transactions and ensuring compliance with industry security standards.

3. Implement Multilingual Support

   Include multilingual features to serve a larger audience. In order to increase accessibility and user happiness, start with important languages depending on user demographics and market expansion objectives.

REFERENCES

Columnist, 2023. *Malaysian aging society at crossroads* [Online]. Available at: <https://www.astroawani.com/berita-malaysia/columnist-malaysian-aging-society-crossroads-438187> [Accessed 2 Februaty 2024].

Murugesan, M., 2021. Extending care for the elderly. *NST Online*. Available at: <https://www.nst.com.my/lifestyle/heal/2021/05/688938/extending-care-elderly> [Accessed 2 Februaty 2024].

Scnova, 2024.*Major challenges to expect when caring for the elderly* [Online]. Available at: <https://scnova.org/major-challenges-to-expect-when-caring-for-the-elderly/> [Accessed 1 Februaty 2024].

FasterCapital, 2024, *What are elderly care platforms and why are they important - FasterCapital* [Online]. Available at: <https://fastercapital.com/topics/what-are-elderly-care-platforms-and-why-are-they-important.html> [Accessed 2 Februaty 2024].

Care Concierge Malaysia, 2023, Care Concierge - Home care and nursing services for elderly [Online]. Available at: <https://mycareconcierge.com/> [Accessed 16 April 2024].

Sadika, 2023. The MVC Architecture - Sadika - Medium. Medium. Available at: <https://medium.com/@sadikarahmantanisha/the-mvc-architecture-97d47e071eb2> [Accessed 17 April 2024].

F, H. and F, H., 2024, WAMP vs XAMPP: Which is the Best Suitable Local Server for Web Development? [Online]. Available at: <https://www.temok.com/blog/wamp-vs-xampp/> [Accessed 17 April 2024].

Chien, C., 2020, What is Rapid Application Development (RAD)? [Online]. Available at: <https://codebots.com/app-development/what-is-rapid-application-development-rad> [Accessed 17 April 2024].

Weingus, L., 2024. Care.com review: Tried and tested (2024). *Forbes Health*. Available at: <https://www.forbes.com/health/family/care-com-review/> [Accessed 3 Februaty 2024].

Infinity, Agile Methodology: Better organization for your team [Online]. Available at: <https://startinfinity.com/project-management-methodologies/agile> [Accessed 3 Februaty 2024].

Atlassian, 2024. *What is Agile? | Atlassian* [Online]. Available at: <https://www.atlassian.com/agile#:~:text=The%20Agile%20methodology%20is%20a,READ%20ON%20BELOW> [Accessed 7 March 2024].

Olic, A., 2020. Advantages and Disadvantages of Agile Project Management [Checklist]. *ActiveCollab*. Available at: <https://activecollab.com/blog/project-management/agile-project-management-advantages-disadvantages> [Accessed 7 March 2024].

Tutorialspoint, 2024. *What are the Advantages and Disadvantages of Agile?* [Online]. Available at: <https://www.tutorialspoint.com/what-are-the-advantages-and-disadvantages-of-agile> [Accessed 7 March 2024].

Abraham, M., 2023. Waterfall Methodology – Ultimate Guide. Management.Org. Available at: <https://management.org/waterfall-methodology> [Accessed 17 April 2024].

Hoory, L., 2022. What is waterfall methodology? Here's how it can help your project management strategy. *Forbes Advisor*. Available at: <https://www.forbes.com/advisor/business/what-is-waterfall-methodology/> [Accessed 9 March 2024].

Lucidchart, 2018, *The pros and cons of waterfall methodology* [Online]. Available at: <https://www.lucidchart.com/blog/pros-and-cons-of-waterfall-methodology> [Accessed 9 March 2024].

Dutta, B., 2024. *Waterfall Methodology: working, Advantages & Disadvantages | Analytics Steps* [Online]. Available at: <https://www.analyticssteps.com/blogs/waterfall-methodology-working-advantages-disadvantages#google_vignette> [Accessed 9 March 2024].

Kissflow, Inc, 2024, *Rapid Application Development (RAD) | Definition, Steps & Full Guide* [Online]. Available at: <https://kissflow.com/application-development/rad/rapid-application-development/> [Accessed 11 March 2024].

Sharma, R., 2024. Top 12 Commerce Project Topics & Ideas in 2023 [For Freshers]. *upGrad blog*. Available at: <https://www.upgrad.com/blog/rad-models-overview/> [Accessed 11 March 2024].

Blog, Https.I., 2021, Everything about Secure Hashing Algorithm (SHA) - Security Boulevard [Online]. Available at: <https://securityboulevard.com/2021/09/everything-about-secure-hashing-algorithm-sha/> [Accessed 11 March 2024].

ICStudio, 2022, *17 Benefits of LaRavel Framework* [Online]. Available at: <https://icstudio.online/en/post/17-benefits-laravel-framework> [Accessed 14 March 2024].

Deshpande, C., 2023, *The best guide to know what is react* [Online]. Available at: <https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs> [Accessed 16 March 2024].

Modan, S., 2024. *Benefits and reasons to choose ReactJS for your project* [Online]. Available at: <https://www.peerbits.com/blog/reasons-to-choose-reactjs-for-your-web-development-project.html> [Accessed 16 March 2024].

Ragala, B. K., 2023, What are the pros and cons of React [Online]. Available at: <https://www.knowledgehut.com/blog/web-development/pros-and-cons-of-react#> [Accessed 16 March 2024].

Vue,js, 2024. *Vue.js* [Online]. Available at: <https://vuejs.org/guide/introduction> [Accessed 18 March 2024].

Patel, J. and Patel, J., 2024, *10 Outstanding Advantages of Vue js You Should Know About* [Online]. Available at: <https://www.monocubed.com/blog/advantages-of-vue-js/> [Accessed 18 March 2024].

Editor, 2022. The good and the bad of Vue.js framework programming. *AltexSoft*. Available at: <https://www.altexsoft.com/blog/pros-and-cons-of-vue-js/> [Accessed 18 March 2024].

WebHostingMonkey, 2024, *What is Visual Studio Code (VS Code)* [Online]. Available at: <https://webhostingmonkey.com/vs-code/> [Accessed 19 March 2024].

Mir, M.A., 2023. What are the advantages and disadvantages of using Visual Studio Code or Atom? *Medium*. Available at: <https://medium.com/@ssc.ahmed.926748/what-are-the-advantages-and-disadvantages-of-using-visual-studio-code-or-atom-d3132bf1af85> [Accessed 19 March 2024].

UniversityOfKent, 2024. *Notepad++ - Software finder - University of Kent* [Online]. Available at: <https://www.kent.ac.uk/software/notepad> [Accessed 20 March 2024].

Centro and Centro, 2023. The power of NotePad++: leveraging its advantages for efficient coding and text editing – a comprehensive guide on how to make the most - Madrid Centro.

Madrid Centro - Lettera Trattoria Moderna. Available at: <https://www.letteramadrid.com/centro/2023/03/22/the-power-of-notepad-leveraging-its-advantages-for/> [Accessed 20 March 2024].

Schaferhoff, N., 2022. *NotePad++ Review – A powerful, free code editor packed with features* [Online]. Available at: <https://www.elegantthemes.com/blog/resources/notepad-review-a-powerful-free-code-editor-packed-with-features#notepad-cons-and-turnoffs> [Accessed 20 March 2024].

Smithaydon, 2023. Is NotePad++ an IDE? - Smithaydon - Medium. *Medium*. Available at: <https://medium.com/@smithaydon2/is-notepad-an-ide-b34e61903e02> [Accessed 20 March 2024].

PhpStorm, 2021. *Explore PhpStorm features | PhpStorm* [Online]. Available at: <https://www.jetbrains.com/help/phpstorm/quick-start-guide-phpstorm.html> [Accessed 21March 2024].

Monovm, 2021, *What is PhpStorm?* [Online]. Available at: <https://monovm.com/blog/what-is-phpstorm/#Benefits-of-PHPStorm> [Accessed 21 March 2024].

Pedamkar, P., 2023, *PhPStorm* [Online]. Available at: <https://www.educba.com/phpstorm/> [Accessed 21 March 2024].

JavaTpoint, 2024. *What is Oracle - javatpoint* [Online]. Available at: < https://www.javatpoint.com/what-is-oracle> [Accessed 23 March 2024].

Nguyen, S. and Nguyen, S., 2024. The benefits of Oracle DBMS for your organization. *DreamFactory Software- Blog - API Management, Enterprise Integrations, Data Security and More*. Available at: <https://blog.dreamfactory.com/the-benefits-of-oracle-dbms-for-your-organization/#2> [Accessed 23 March 2024].

Domantas G., 2024, *What is MySQL and how does it work* [Online]. Available at: <https://www.hostinger.com/tutorials/what-is-mysql> [Accessed 25 March 2024].

blueclaw, 2021, *MySQL Advantages and Disadvantages - Blue Claw Database Developer Resource* [Online]. Available at: <https://blueclawdb.com/mysql/advantages-disadvantages-mysql/> [Accessed 25 March 2024].

S, R.A., 2023, *What is SQLite? Everything You Need to Know* [Online]. Available at: <https://www.simplilearn.com/tutorials/sql-tutorial/what-is-sqlite> [Accessed 26 March 2024].

javaTpoint, 2024. SQLite Advantages and Disadvantages - javatpoint [Online]. Available at: <https://www.javatpoint.com/sqlite-advantages-and-disadvantages> [Accessed 26 March 2024].

Wong, C., 2023. Advantages and Disadvantages of using SQLite - Christopher Wong - Medium. *Medium*. Available at: <https://medium.com/@cw30355/advantages-and-disadvantages-of-using-sqlite-2f490fa467bd> [Accessed 26 March 2024].

Tis, T., 2023. LaRavel: Benefits for outstanding PHP Web Development. TIS. Available at: <https://www.tisdigitech.com/blog/laravel-framework-best-choice-for-php-web-development/> [Accessed 18 April 2024].

APPENDICES

APPENDIX A: Work Breakdown Structure

1.0 Project Initiation
1.1 Identify project background
1.2 Identify problem statement
1.3 Identify project objective
1.4 Identify project solution
1.5 Identify project approach
1.6 Identify project scope and limitation of the study
2.0 Literature Review
2.1 Review on existing elderly care platforms
2.2 Review on software development methodologies
2.3 Review on web development tools
2.3.1 Review on development frameworks
2.3.2 Review on code editors
2.3.3 Review on databases
3.0 Methodology and Work Plan
3.1 Determine the phase of chosen software development methodology
3.2 Determine the adopted development and prototyping tools
3.3 Define project plan
3.3.1 Construct WBS
3.3.2 Construct Gantt Chart
4.0 Project Specification
4.1 Define requirements specifications
4.1.1 Define functional requirements
4.1.2 Define non-functional requirements
4.2 Construct use case diagram
4.3 Outline use case description

4.4 Develop prototype
5.0 System Design
5.1 Define the system architecture design
5.2 Design UML diagram
5.2.1 Design Class diagram
5.2.2 Design Activity diagram
5.3 Define the database design
5.3.1 Design ERD diagram
5.3.2 Design data dictionary table
6.0 System Development and Testing
6.1 First Iteration
6.1.1 Develop Client Module
6.1.2 Conduct Testing
6.1.2.1 Conduct Unit Testing
6.1.2.2 Conduct Feature Testing
6.2 Second Iteration
6.2.1 Develop Caregiver Module
6.2.2 Conduct Testing
6.2.2.1 Conduct Unit Testing
6.2.2.2 Conduct Feature Testing
6.3 Third Iteration
6.3.1 Develop Administrator Module
6.3.2 Conduct Testing
6.3.2.1 Conduct Unit Testing
6.3.2.2 Conduct Feature Testing
6.3.3 Conduct Use Acceptance Testing
7.0 Closing
7.1 Finalize final report

APPENDIX B: Unit Test code

```php
<?php
namespace Tests\Unit;

use Tests\TestCase;
use App\Models\Booking;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use Mockery;
use Carbon\Carbon;

class BookingModelTest extends TestCase
{
    // Test attribute casting without involving database
    public function testAttributeCasting()
    {
        $booking = new Booking();
        $booking->payment_date = '2024-01-01 10:00:00';

        // Test if casting works
        $this->assertInstanceOf(Carbon::class, $booking->payment_date);
        $this->assertEquals('2024-01-01 10:00:00',
$booking->payment_date->format('Y-m-d H:i:s'));
    }

    // Mock relationships and verify expected behavior
    public function testClientRelationship()
    {
        $booking = Mockery::mock(Booking::class)->makePartial();
        $clientRelation = Mockery::mock(BelongsTo::class);

        // Set up the mock to return a relation instance
        $booking->shouldReceive('client')->andReturn($clientRelation);
        $clientRelation->shouldReceive('getResults')->andReturn(Mockery::mock('Ap
p\Models\User'));

        $this->assertInstanceOf(BelongsTo::class, $booking->client());
    }

    public function testCaregiverRelationship()
    {
        $booking = Mockery::mock(Booking::class)->makePartial();
        $caregiverRelation = Mockery::mock(BelongsTo::class);
        // Set up the mock to return a relation instance
        $booking->shouldReceive('caregiver')->andReturn($caregiverRelation);
        $caregiverRelation->shouldReceive('getResults')->andReturn(Mockery::mock(
'App\Models\User'));

        $this->assertInstanceOf(BelongsTo::class, $booking->caregiver());
    }

    public function testServiceRelationship()
    {
        $booking = Mockery::mock(Booking::class)->makePartial();
        $serviceRelation = Mockery::mock(BelongsTo::class);

        // Set up the mock to return a relation instance
        $booking->shouldReceive('service')->andReturn($serviceRelation);
        $serviceRelation->shouldReceive('getResults')->andReturn(Mockery::mock('A
pp\Models\Service'));

        $this->assertInstanceOf(BelongsTo::class, $booking->service());
    }

    public function testServiceDateRelationship()
    {
        $booking = Mockery::mock(Booking::class)->makePartial();
        $serviceDateRelation = Mockery::mock(BelongsTo::class);

        // Set up the mock to return a relation instance
        $booking->shouldReceive('serviceDate')->andReturn($serviceDateRelation);
        $serviceDateRelation->shouldReceive('getResults')->andReturn(Mockery::moc
k('App\Models\ServiceDate'));

        $this->assertInstanceOf(BelongsTo::class, $booking->serviceDate());
    }

    public function testTimeSlotRelationship()
    {
        $booking = Mockery::mock(Booking::class)->makePartial();
        $timeSlotRelation = Mockery::mock(BelongsTo::class);

        // Set up the mock to return a relation instance
        $booking->shouldReceive('timeSlot')->andReturn($timeSlotRelation);
        $timeSlotRelation->shouldReceive('getResults')->andReturn(Mockery::mock('
App\Models\ServiceTimeslot'));

        $this->assertInstanceOf(BelongsTo::class, $booking->timeSlot());
    }

    // Test that the deleted event is triggered, mocking any side effects
    public function testDeletedEvent()
    {
        $booking = Mockery::mock(Booking::class)->makePartial();

        // Mock the delete method and verify it's called
        $booking->shouldReceive('delete')->once();
        $booking->delete();

        // Verify that delete was called
        $this->assertTrue(true);
    }

    protected function tearDown(): void
    {
        Mockery::close();
        parent::tearDown();
    }
}
```

```
<?php
namespace Tests\Unit;

use Tests\TestCase;
use App\Models\Service;
use Illuminate\Database\Eloquent\Relations\HasMany;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use Mockery;

class ServiceModelTest extends TestCase
{
    public function testBookingsRelationship()
    {
        // Create a partial mock of the Service model
        $service = Mockery::mock(Service::class)->makePartial();

        // Create a mock of the HasMany relationship
        $bookingsRelation = Mockery::mock(HasMany::class);

        // Set up the expectation that the bookings method will return the
HasMany mock
```

```
        $service->shouldReceive('bookings')->andReturn($bookingsRelation);

        // Verify that the bookings method returns an instance of HasMany
        $this->assertInstanceOf(HasMany::class, $service->bookings());
    }

    public function testBookingsWithFeedbackRelationship()
    {
        // Create a partial mock of the Service model
        $service = Mockery::mock(Service::class)->makePartial();

        // Create a mock of the HasMany relationship
        $bookingsWithFeedbackRelation = Mockery::mock(HasMany::class);

        // Set up the expectation that the bookingsWithFeedback method will
return the HasMany mock
        $service->shouldReceive('bookingsWithFeedback')->andReturn($bookingsWithF
eedbackRelation);

        // Verify that the bookingsWithFeedback method returns an instance of
HasMany
        $this->assertInstanceOf(HasMany::class,
$service->bookingsWithFeedback());
    }

    public function testCaregiverRelationship()
    {
        // Create a partial mock of the Service model
        $service = Mockery::mock(Service::class)->makePartial();

        // Create a mock of the BelongsTo relationship
        $caregiverRelation = Mockery::mock(BelongsTo::class);

        // Set up the expectation that the caregiver method will return the
BelongsTo mock
        $service->shouldReceive('caregiver')->andReturn($caregiverRelation);

        // Verify that the caregiver method returns an instance of BelongsTo
        $this->assertInstanceOf(BelongsTo::class, $service->caregiver());
    }

    public function testClientRelationship()
    {
        // Create a partial mock of the Service model
        $service = Mockery::mock(Service::class)->makePartial();
        // Create a mock of the BelongsTo relationship
        $clientRelation = Mockery::mock(BelongsTo::class);

        // Set up the expectation that the client method will return the
BelongsTo mock
        $service->shouldReceive('client')->andReturn($clientRelation);

        // Verify that the client method returns an instance of BelongsTo
        $this->assertInstanceOf(BelongsTo::class, $service->client());
    }

    public function testServiceDatesRelationship()
    {
        // Create a partial mock of the Service model
        $service = Mockery::mock(Service::class)->makePartial();

        // Create a mock of the HasMany relationship
        $serviceDatesRelation = Mockery::mock(HasMany::class);

        // Set up the expectation that the serviceDates method will return the
HasMany mock
        $service->shouldReceive('serviceDates')->andReturn($serviceDatesRelation)
;

        // Verify that the serviceDates method returns an instance of HasMany
        $this->assertInstanceOf(HasMany::class, $service->serviceDates());
    }

    public function testServiceTimeslotsRelationship()
    {
        // Create a partial mock of the Service model
        $service = Mockery::mock(Service::class)->makePartial();

        // Create a mock of the HasMany relationship
        $serviceTimeslotsRelation = Mockery::mock(HasMany::class);

        // Set up the expectation that the serviceTimeslots method will return
the HasMany mock
        $service->shouldReceive('serviceTimeslots')->andReturn($serviceTimeslotsR
elation);

        // Verify that the serviceTimeslots method returns an instance of HasMany
        $this->assertInstanceOf(HasMany::class, $service->serviceTimeslots());
    }
```

```php
    public function testDatesRelationship()
    {
        // Create a partial mock of the Service model
        $service = Mockery::mock(Service::class)->makePartial();

        // Create a mock of the HasMany relationship
        $datesRelation = Mockery::mock(HasMany::class);

        // Set up the expectation that the dates method will return the HasMany
mock
        $service->shouldReceive('dates')->andReturn($datesRelation);

        // Verify that the dates method returns an instance of HasMany
        $this->assertInstanceOf(HasMany::class, $service->dates());
    }

    // Clean up Mockery after each test
    protected function tearDown(): void
    {
        Mockery::close();
        parent::tearDown();
    }
}
```

```php
<?php

namespace Tests\Unit;

use App\Models\ServiceDate;
use App\Models\Service;
use App\Models\ServiceTimeslot;
use Illuminate\Database\Eloquent\Collection;
use Mockery;
use PHPUnit\Framework\TestCase;

class ServiceDateModelTest extends TestCase
{
    protected function tearDown(): void
    {
        // Close Mockery after each test to ensure no lingering expectations
        Mockery::close();
        parent::tearDown();
    }

    /** @test */
    public function it_has_a_service_relationship()
    {
        $serviceDate = Mockery::mock(ServiceDate::class)->makePartial();

        $service = Mockery::mock(Service::class);
        $serviceDate->shouldReceive('service')
            ->once()
            ->andReturn($service);

        $this->assertInstanceOf(Service::class, $serviceDate->service());
    }

    /** @test */
    public function it_has_a_timeslots_relationship()
    {
        $serviceDate = Mockery::mock(ServiceDate::class)->makePartial();

        $timeslots = Mockery::mock(Collection::class);
        $serviceDate->shouldReceive('timeslots')
            ->once()
            ->andReturn($timeslots);

        $this->assertInstanceOf(Collection::class, $serviceDate->timeslots());
    }

    /** @test */
    public function it_has_a_service_timeslots_relationship()
    {
        $serviceDate = Mockery::mock(ServiceDate::class)->makePartial();

        $serviceTimeslots = Mockery::mock(Collection::class);
        $serviceDate->shouldReceive('serviceTimeslots')
            ->once()
            ->andReturn($serviceTimeslots);

        $this->assertInstanceOf(Collection::class,
$serviceDate->serviceTimeslots());
    }


}
```

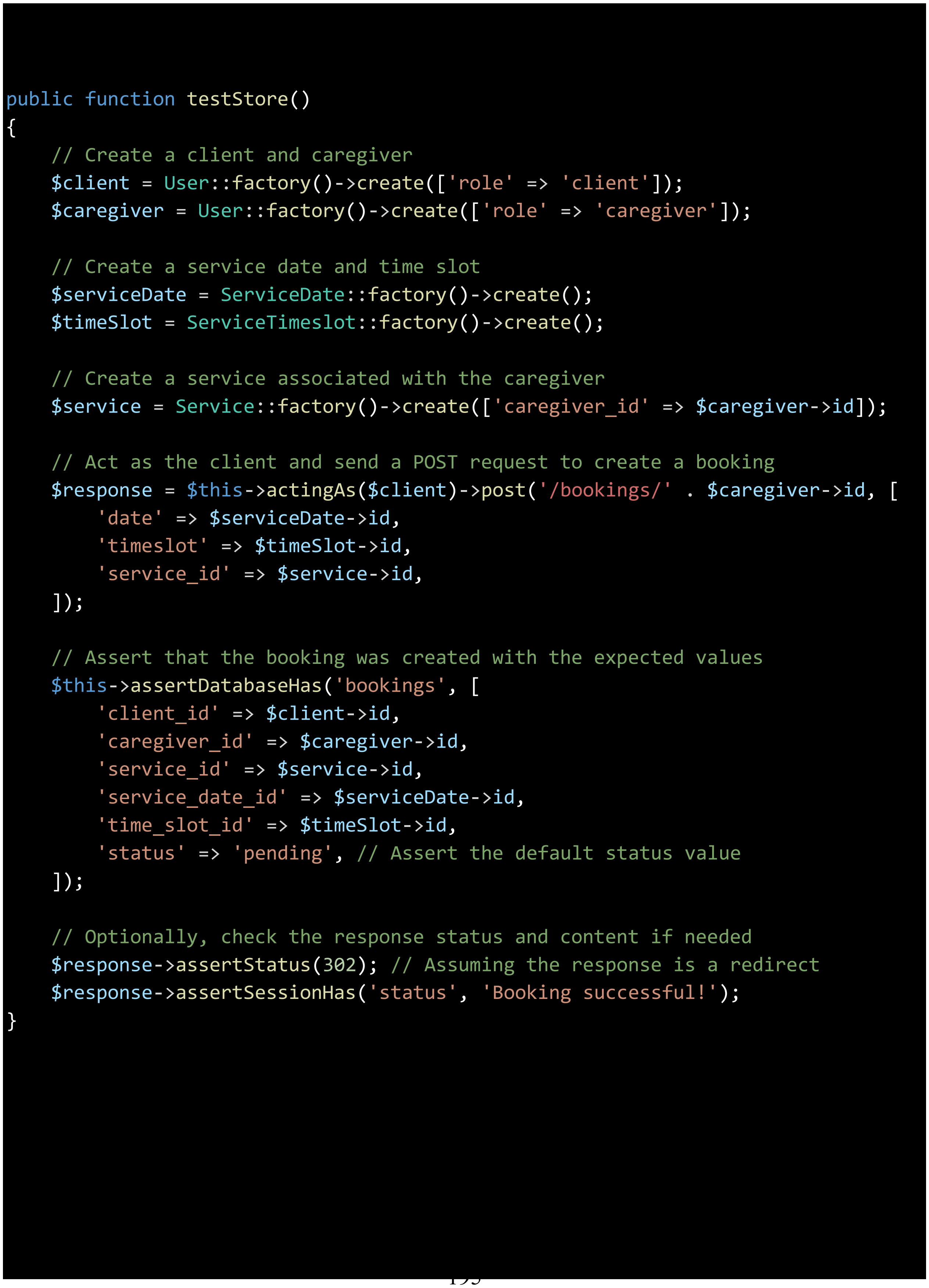
```
<?php

namespace Tests\Unit;

use App\Models\ServiceTimeslot;
use PHPUnit\Framework\TestCase;

class ServiceTimeslotTest extends TestCase
{
    public function testIsBooked()
    {
        $timeslot = new ServiceTimeslot();
        $timeslot->availability = 1; // Simulate a booked state

        $this->assertTrue($timeslot->isBooked(), 'The timeslot should be
booked.');
        $this->assertFalse($timeslot->isAvailable(), 'The timeslot should not be
available.');
    }


    public function testIsAvailable()
    {
        // Create an instance of ServiceTimeslot with the availability set to
available (0)
        $timeslot = new ServiceTimeslot(['availability' => 0]);

        // Assert that isAvailable returns true
        $this->assertTrue($timeslot->isAvailable());

        // Assert that isBooked returns false
        $this->assertFalse($timeslot->isBooked());
    }

    public function testFillableAttributes()
    {
        $fillable = (new ServiceTimeslot())->getFillable();
        $expected = [
            'service_date_id',
            'start_time',
            'end_time',
        ];

        // Assert that the fillable attributes match the expected ones
        $this->assertEquals($expected, $fillable);
    }
}
```

```php
<?php

namespace Tests\Unit;

use Tests\TestCase;
use Mockery;
use App\Models\User;
use Illuminate\Database\Eloquent\Collection;

class UserModelTest extends TestCase
{
    /** @test */
    public function it_has_fillable_attributes()
    {
        $user = new User();

        $this->assertEquals([
            'name',
            'email',
            'password',
            'role',
        ], $user->getFillable());
    }

    /** @test */
    public function it_hides_password_and_remember_token()
    {
        $user = new User();

        $this->assertEquals([
            'password',
            'remember_token',
        ], $user->getHidden());
    }

    /** @test */
    public function it_casts_email_verified_at_to_datetime()
    {
        $user = new User();
        $this->assertArrayHasKey('email_verified_at', $user->getCasts());
        $this->assertEquals('datetime', $user->getCasts()['email_verified_at']);
    }

    /** @test */
    public function it_has_bookings_relationship()
    {
        $mockedCollection = Mockery::mock(Collection::class);

        $user = Mockery::mock(User::class)->makePartial();
        $user->shouldReceive('bookings')->andReturn($mockedCollection);

        $this->assertInstanceOf(Collection::class, $user->bookings());
    }

    /** @test */
    public function it_has_caregiving_bookings_relationship()
    {
        $mockedCollection = Mockery::mock(Collection::class);

        $user = Mockery::mock(User::class)->makePartial();
        $user->shouldReceive('caregivingBookings')->andReturn($mockedCollection);

        $this->assertInstanceOf(Collection::class, $user->caregivingBookings());
    }

    /** @test */
    public function it_has_messages_relationship()
    {
        $mockedCollection = Mockery::mock(Collection::class);

        $user = Mockery::mock(User::class)->makePartial();
        $user->shouldReceive('messages')->andReturn($mockedCollection);

        $this->assertInstanceOf(Collection::class, $user->messages());
    }

    /** @test */
    public function it_has_services_relationship()
    {
        $mockedCollection = Mockery::mock(Collection::class);

        $user = Mockery::mock(User::class)->makePartial();
        $user->shouldReceive('services')->andReturn($mockedCollection);
        $this->assertInstanceOf(Collection::class, $user->services());
    }

    /** @test */
    public function it_has_caregiver_relationship()
    {
        $mockedCaregiver = Mockery::mock(User::class);

        $user = Mockery::mock(User::class)->makePartial();
        $user->shouldReceive('caregiver')->andReturn($mockedCaregiver);

        $this->assertInstanceOf(User::class, $user->caregiver());
    }

    protected function tearDown(): void
    {
        Mockery::close();
        parent::tearDown();
    }
}
```

APPENDIX C: Feature Test code

```php
<?php

namespace Tests\Feature;

use App\Models\User;
use Illuminate\Foundation\Testing\RefreshDatabase;
use Tests\TestCase;

class AdminControllerFeatureTest extends TestCase
{
    use RefreshDatabase;

    protected function setUp(): void
    {
        parent::setUp();
        // Create a user with admin role if needed
        $this->admin = User::factory()->create(['role' => 'administrator']);
        $this->actingAs($this->admin);
    }

    /** @test */
    public function it_can_list_users()
    {
        // Create some users
        User::factory()->count(20)->create();

        // Make a GET request to the listUsers route
        $response = $this->get(route('admin.users'));

        // Check the response
        $response->assertStatus(200);
        $response->assertViewIs('admin.users');
        $response->assertViewHas('users');
    }

    /** @test */
    public function it_can_delete_a_user_and_associated_data()
    {
        // Create a user with bookings
        $user = User::factory()->create(['role' => 'client']);
        $user->bookings()->create([
            'caregiver_id' => User::factory()->create(['role' =>
'caregiver'])->id,
            'service_id' => 1,
            'service_date_id' => 1,
            'status' => 'pending',
        ]);

        // Ensure the user and their bookings exist
        $this->assertDatabaseHas('users', ['id' => $user->id]);
        $this->assertDatabaseHas('bookings', ['client_id' => $user->id]);

        // Make a DELETE request to the deleteUser route
        $response = $this->delete(route('admin.deleteUser', ['id' =>
$user->id]));

        // Check the response
        $response->assertStatus(302);
        $response->assertRedirect(route('admin.users'));
        $response->assertSessionHas('success', 'User and associated data deleted
successfully.');

        // Assert that the user and their bookings were deleted
        $this->assertDatabaseMissing('users', ['id' => $user->id]);
        $this->assertDatabaseMissing('bookings', ['client_id' => $user->id]);
    }

    /** @test */
    public function it_can_view_user_details()
    {
        // Create a user
        $user = User::factory()->create([
            'name' => 'John Doe',
            'email' => 'john.doe@example.com',
        ]);

        // Make a GET request to the viewUser route
        $response = $this->get(route('admin.viewUser', ['id' => $user->id]));

        // Check the response
        $response->assertStatus(200);
        $response->assertViewIs('admin.user_details');
        $response->assertViewHas('user', function ($viewUser) use ($user) {
            return $viewUser->id === $user->id;
        });
    }
}
```

```php
<?php

namespace Tests\Feature;

use App\Models\Client;
use App\Models\Caregiver;
use App\Models\Service;
use App\Models\ServiceDate;
use App\Models\ServiceTimeslot;
use App\Models\Booking;
use Tests\TestCase;
use Illuminate\Testing\TestResponse;
use App\Http\Controllers\BookingController;
use App\Models\User;
use Illuminate\Foundation\Testing\RefreshDatabase;
use Illuminate\Foundation\Testing\WithoutMiddleware;
use Illuminate\Support\Facades\Notification;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Log;

class BookingControllerFeatureTest extends TestCase
{
    use RefreshDatabase, WithoutMiddleware;

    protected $user;
    protected $caregiver;
    protected $service;
    protected $serviceDate;
    protected $timeSlot;

    protected function setUp(): void
    {
        parent::setUp();
        $this->withoutMiddleware(\App\Http\Middleware\VerifyCsrfToken::class);

        // Create test data
        $this->user = User::factory()->create();
        $this->caregiver = User::factory()->create(['role' => 'caregiver']);
        $this->service = Service::factory()->create(['caregiver_id' =>
$this->caregiver->id]);
        $this->serviceDate = ServiceDate::factory()->create();
        $this->timeSlot = ServiceTimeslot::factory()->create(['availability' =>
1]); // Availability 1 means available
    }
    public function testCreate()
    {
        $controller = new BookingController();
        $response = $controller->create($this->caregiver->id);

        $this->assertEquals('booking', $response->getName());
        $this->assertArrayHasKey('caregiver', $response->getData());
        $this->assertArrayHasKey('services', $response->getData());
    }

    public function testGetAvailableTimeslots()
{
    // Create a ServiceDate and a ServiceTimeslot with known data
    $serviceDate = ServiceDate::factory()->create();
    $timeslot = ServiceTimeslot::factory()->create([
        'service_date_id' => $serviceDate->id,
        'availability' => 0, // Assuming 0 means available
        'start_time' => '06:49:24',
        'end_time' => '02:03:44'
    ]);

    // Create a request with the service_date_id
    $request = Request::create('/timeslots', 'GET', ['service_date_id' =>
$serviceDate->id]);

    // Initialize the controller and get the response
    $controller = new BookingController();
    $response = $controller->getAvailableTimeslots($request);

    // Decode the JSON response into an array
    $responseData = json_decode($response->getContent(), true);

    // Debug: Output response data for inspection
    // dd($responseData);

    // Assert that the specific timeslot is in the response data
    $this->assertContains([
        'id' => $timeslot->id,
        'start_time' => '06:49:24', // Ensure this matches the data in the
response
        'end_time' => '02:03:44',   // Ensure this matches the data in the
response
    ], $responseData);
}
```



```php
public function testStore()
{
    // Create a client and caregiver
    $client = User::factory()->create(['role' => 'client']);
    $caregiver = User::factory()->create(['role' => 'caregiver']);

    // Create a service date and time slot
    $serviceDate = ServiceDate::factory()->create();
    $timeSlot = ServiceTimeslot::factory()->create();

    // Create a service associated with the caregiver
    $service = Service::factory()->create(['caregiver_id' => $caregiver->id]);

    // Act as the client and send a POST request to create a booking
    $response = $this->actingAs($client)->post('/bookings/' . $caregiver->id, [
        'date' => $serviceDate->id,
        'timeslot' => $timeSlot->id,
        'service_id' => $service->id,
    ]);

    // Assert that the booking was created with the expected values
    $this->assertDatabaseHas('bookings', [
        'client_id' => $client->id,
        'caregiver_id' => $caregiver->id,
        'service_id' => $service->id,
        'service_date_id' => $serviceDate->id,
        'time_slot_id' => $timeSlot->id,
        'status' => 'pending', // Assert the default status value
    ]);

    // Optionally, check the response status and content if needed
    $response->assertStatus(302); // Assuming the response is a redirect
    $response->assertSessionHas('status', 'Booking successful!');
}
```

```
    public function testEdit()
    {
        $serviceDate = ServiceDate::create([
            'service_id' => 1, // Ensure this ID exists in the 'services' table
            'date' => '2025-03-25',
        ]);

        $this->assertDatabaseHas('service_dates', [
            'service_id' => 1,
            'date' => '2025-03-25',
        ]);
    }

    public function testUpdateBooking()
{
    // Create required records
    $client = User::factory()->create(['role' => 'client']);
    $caregiver = User::factory()->create(['role' => 'caregiver']);
    $service = Service::factory()->create(['caregiver_id' => $caregiver->id]);
    $serviceDate = ServiceDate::factory()->create(['service_id' =>
$service->id]);
    $timeSlot = ServiceTimeslot::factory()->create(['service_date_id' =>
$serviceDate->id, 'availability' => 0]);

    // Create a booking with existing IDs
    $booking = Booking::factory()->create([
        'client_id' => $client->id,
        'caregiver_id' => $caregiver->id,
        'service_id' => $service->id,
        'service_date_id' => $serviceDate->id,
        'time_slot_id' => $timeSlot->id,
        'status' => 'pending',
    ]);

    // Create new records for update
    $newDate = ServiceDate::factory()->create(['service_id' => $service->id]);
    $newTimeslot = ServiceTimeslot::factory()->create(['service_date_id' =>
$newDate->id]);

    // Update the booking
    $response = $this->putJson(route('booking.update', $booking->id), [
        'date' => $newDate->id,
        'timeslot' => $newTimeslot->id,
    ]);
    // Assert the response
    $response->assertStatus(302);
    $response->assertSessionHas('success', 'Booking updated successfully.');

    // Assert database changes
    $this->assertDatabaseHas('bookings', [
        'id' => $booking->id,
        'service_date_id' => $newDate->id,
        'time_slot_id' => $newTimeslot->id,
    ]);

    // Check availability
    $this->assertDatabaseHas('service_timeslots', [
        'id' => $newTimeslot->id,
        'availability' => 1, // Should be available after selection
    ]);
    $this->assertDatabaseHas('service_timeslots', [
        'id' => $timeSlot->id,
        'availability' => 0, // Should remain unavailable
    ]);
}

    public function testDestroy()
{
    // Disable CSRF protection for testing
    $this->withoutMiddleware(\App\Http\Middleware\VerifyCsrfToken::class);

    // Create a time slot with availability set to 1
    $timeSlot = ServiceTimeslot::factory()->create(['availability' => 1]);

    // Create a booking associated with the time slot
    $booking = Booking::factory()->create([
        'time_slot_id' => $timeSlot->id,
        'client_id' => User::factory()->create(['role' => 'client'])->id,
        'caregiver_id' => User::factory()->create(['role' => 'caregiver'])->id,
        'service_id' => Service::factory()->create()->id,
        'service_date_id' => ServiceDate::factory()->create()->id,
    ]);

    // Perform the DELETE request to destroy the booking
    $response = $this->delete(route('booking.destroy', ['id' => $booking->id]));

    // Assert the response status and redirection
    $response->assertStatus(302);
    $response->assertRedirect(route('booking-list'));
    // Assert that the booking has been deleted from the database
    $this->assertDatabaseMissing('bookings', ['id' => $booking->id]);

    // Refresh the time slot and assert its availability is updated
    $timeSlot->refresh();
    $this->assertEquals(0, $timeSlot->availability);
}


public function testShowApproved()
{
    // Create a user and act as that user
    $user = User::factory()->create();
    $this->actingAs($user);

    // Create necessary related records
    $caregiver = User::factory()->create(['role' => 'caregiver']);
    $service = Service::factory()->create(['caregiver_id' => $caregiver->id]);
    $serviceDate = ServiceDate::factory()->create();
    $timeSlot = ServiceTimeslot::factory()->create(['availability' => 0]);

    // Create a booking with 'approved' status and all related records
    $booking = Booking::factory()->create([
        'client_id' => $user->id,
        'caregiver_id' => $caregiver->id,
        'service_id' => $service->id,
        'status' => 'approved',
        'service_date_id' => $serviceDate->id,
        'time_slot_id' => $timeSlot->id
    ]);

    // Perform a GET request to the route that should return approved bookings
    $response = $this->get(route('approved'));

    // Check if the status code is 200 (OK)
    $response->assertStatus(200);

    // Verify that the response contains the booking ID
    $response->assertSee($booking->id);
}


public function testShowDeclined()
{
    // Create a user and act as that user
    $user = User::factory()->create();
    $this->actingAs($user);

    // Create necessary related records
    $caregiver = User::factory()->create(['role' => 'caregiver']);
    $service = Service::factory()->create(['caregiver_id' => $caregiver->id]);
    $serviceDate = ServiceDate::factory()->create();
    $timeSlot = ServiceTimeslot::factory()->create(['availability' => 0]);

    // Create a booking with 'approved' status and all related records
    $booking = Booking::factory()->create([
        'client_id' => $user->id,
        'caregiver_id' => $caregiver->id,
        'service_id' => $service->id,
        'status' => 'declined',
        'service_date_id' => $serviceDate->id,
        'time_slot_id' => $timeSlot->id
    ]);

    // Perform a GET request to the route that should return approved bookings
    $response = $this->get(route('declined'));

    // Check if the status code is 200 (OK)
    $response->assertStatus(200);

    // Verify that the response contains the booking ID
    $response->assertSee($booking->id);
}


    public function testFeedbackForm()
{
    // Create related records
    $client = User::factory()->create(['role' => 'client']);
    $caregiver = User::factory()->create(['role' => 'caregiver']);
    $service = Service::factory()->create(['caregiver_id' => $caregiver->id]);
    $serviceDate = ServiceDate::factory()->create();
    $timeSlot = ServiceTimeslot::factory()->create(['availability' => 0]);

    // Create a booking with all necessary related records
    $booking = Booking::factory()->create([
        'client_id' => $client->id,
        'caregiver_id' => $caregiver->id,
        'service_id' => $service->id,
        'service_date_id' => $serviceDate->id,
        'time_slot_id' => $timeSlot->id,
        'status' => 'approved'
    ]);

    // Create an instance of the controller and call the method
    $controller = new BookingController();
    $response = $controller->feedbackForm($booking->id);

    // Check that the view name and data are as expected
    $this->assertEquals('feedback', $response->getName());
    $this->assertArrayHasKey('booking', $response->getData());
}


public function testStorePayment()
{
    // Create related records
    $client = User::factory()->create(['role' => 'client']);
    $caregiver = User::factory()->create(['role' => 'caregiver']);
    $service = Service::factory()->create(['caregiver_id' => $caregiver->id]);
    $serviceDate = ServiceDate::factory()->create();
    $timeSlot = ServiceTimeslot::factory()->create(['availability' => 0]);

    // Create a booking with all necessary related records
    $booking = Booking::factory()->create([
        'client_id' => $client->id,
        'caregiver_id' => $caregiver->id,
        'service_id' => $service->id,
        'service_date_id' => $serviceDate->id,
        'time_slot_id' => $timeSlot->id,
        'status' => 'completed'
    ]);

    $paymentMethod = 'Credit Card';

    $response =
$this->withoutMiddleware(\App\Http\Middleware\VerifyCsrfToken::class)
                    ->postJson(route('storePayment', ['id' => $booking->id]), [
                        'payment_method' => $paymentMethod
                    ]);

    $response->assertStatus(200)
             ->assertJson(['message' => 'Payment method saved successfully.']);
```

```php
    $booking->refresh();
    $this->assertEquals($paymentMethod, $booking->payment_method);
    $this->assertNotNull($booking->payment_date);
}

}
```

```php
<?php

namespace Tests\Feature;

use Illuminate\Foundation\Testing\RefreshDatabase;
use Illuminate\Support\Facades\Auth;
use Tests\TestCase;
use App\Models\User;
use App\Models\Booking;
use App\Models\ServiceDate;
use App\Models\ServiceTimeslot;
use Carbon\Carbon;

class CalendarControllerFeatureTest extends TestCase
{
    use RefreshDatabase;

    /**
     * Test that the calendar shows the caregiver's approved bookings.
     *
     * @return void
     */
    public function test_show_calendar_displays_approved_bookings()
    {
        // Create a user and log them in as the caregiver
        $caregiver = User::factory()->create();
        Auth::login($caregiver);

        // Create service dates and timeslots
        $serviceDate = ServiceDate::factory()->create([
            'date' => '2025-01-25', // Set a specific date for consistency
        ]);
        $timeSlot = ServiceTimeslot::factory()->create([
            'start_time' => '09:00:00',
            'end_time' => '10:00:00',
        ]);
        // Create a client (required for foreign key reference)
        $client = User::factory()->create(); // Assuming User is used as a client
as well

        // Create an approved booking
        $booking = Booking::factory()->create([
            'caregiver_id' => $caregiver->id,
            'client_id' => $client->id, // Ensure client_id is valid
            'service_date_id' => $serviceDate->id,
            'time_slot_id' => $timeSlot->id,
            'status' => 'approved',
        ]);

        // Create a booking with a different status (should not be displayed)
        Booking::factory()->create([
            'caregiver_id' => $caregiver->id,
            'client_id' => $client->id, // Ensure client_id is valid
            'service_date_id' => $serviceDate->id,
            'time_slot_id' => $timeSlot->id,
            'status' => 'pending',
        ]);

        // Send a GET request to the showCalendar route
        $response = $this->get(route('caregiver.calendar'));

        // Assert the response is successful
        $response->assertStatus(200);

        // Extract the start and end times for assertion
        $startDateTime = Carbon::parse($serviceDate->date . ' ' .
$timeSlot->start_time);
        $endDateTime = Carbon::parse($serviceDate->date . ' ' .
$timeSlot->end_time);

        $startTime = $startDateTime->format('Y-m-d\TH:i:s');
        $endTime = $endDateTime->format('Y-m-d\TH:i:s');

        // Format the title
        $title = $startDateTime->format('g:iA') . '-' .
$endDateTime->format('g:iA') . ' #Booking' . $booking->id;

        // Assert that the event title contains the booking ID and times
        $response->assertSee($title);
        $response->assertSee($startTime);
        $response->assertSee($endTime);
    }
}
```

```
<?php
namespace Tests\Feature;

use Tests\TestCase;
use Illuminate\Foundation\Testing\RefreshDatabase;
use Illuminate\Foundation\Testing\WithFaker;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Storage;
use App\Models\Service;
use App\Models\User;
use App\Models\ServiceTimeslot;
use App\Models\Booking;
use App\Models\ServiceDate;
use Illuminate\Support\Facades\Log;

use App\Models\Caregiver;
use Illuminate\Http\UploadedFile;

use Illuminate\Support\Facades\DB;
use Carbon\Carbon;

class CaregiverControllerFeatureTest extends TestCase
{
    use RefreshDatabase;

    protected $caregiver;

    protected function setUp(): void
    {
        parent::setUp();
        $this->caregiver = User::factory()->create(['role' => 'caregiver']);
        Auth::login($this->caregiver);
    }

    /** @test */
    public function it_displays_service_list()
    {
        // Create a caregiver user
        $caregiver = User::factory()->create();
        // Log in as the caregiver
        Auth::login($caregiver);

        // Create a service associated with the caregiver
        $service = Service::factory()->create(['caregiver_id' =>
$caregiver->id]);

        // Send a request to the service list route
        $response = $this->get(route('service-list'));

        // Assert the response status is OK
        $response->assertStatus(200);

        // Assert that the response contains the service name or any other
expected content
        $response->assertSee($service->name);
    }


    /** @test */
    public function it_shows_add_service_form()
    {
        $response = $this->get(route('add-service'));

        $response->assertStatus(200);
        $response->assertViewIs('caregiver.add-service');
    }

    /** @test */
    public function it_can_add_a_service_with_dates_and_timeslots()
    {
        // Fake the storage for image uploads
        Storage::fake('public');

        // Prepare the request data
        $data = [
            'name' => 'Test Service',
            'description' => 'A description of the service',
            'service_type' => 'Type A',
            'duration' => '60',
            'price' => 100.0,
            'availability' => 'Monday to Friday',
            'location' => 'Location A',
            'provider' => 'Provider A',
            'notes' => 'Additional notes',
            'dates' => ['2024-08-01', '2024-08-02'],
            'timeslots' => [
                [
                    ['start' => '09:00', 'end' => '12:00'],
                ],
                [
                    ['start' => '13:00', 'end' => '15:00'],
                ],
            ],
            // 'image' => UploadedFile::fake()->image('service.jpg'),
        ];

        // Perform the request
        $response = $this->post(route('save-service'), $data);

        // Assert the service was added
        $response->assertRedirect(route('service-list'));
        $this->assertDatabaseHas('services', [
            'name' => 'Test Service',
            'description' => 'A description of the service',
        ]);

        // Assert the image was stored
        $service = Service::first();
        // Storage::disk('public')->assertExists($service->image);

        // Assert the dates and timeslots were added
        $this->assertCount(2, ServiceDate::where('service_id',
$service->id)->get());
        $this->assertDatabaseHas('service_timeslots', [
            'start_time' => '09:00',
            'end_time' => '12:00',
        ]);
        $this->assertDatabaseHas('service_timeslots', [
            'start_time' => '13:00',
            'end_time' => '15:00',
        ]);
    }


    /** @test */
    public function it_shows_service_details()
    {
        $service = Service::factory()->create();
        $response = $this->get(route('service.show', ['service' =>
$service->id]));

        $response->assertStatus(200);
        $response->assertViewHas('service');
        $response->assertSee($service->name);
    }

    /** @test */
    public function it_shows_edit_service_form()
    {
        $service = Service::factory()->create();
        $response = $this->get(route('service.edit', ['service' =>
$service->id]));

        $response->assertStatus(200);
        $response->assertViewHas('service');
    }

/** @test */
public function it_updates_service_successfully()
    {
        // Arrange
        $user = User::factory()->create();
        $this->actingAs($user);

        $service = Service::factory()->create();

        // Define valid request data
        $data = [
            'name' => 'New Service',
            'description' => 'Updated description',
            'service_type' => 'Type A',
            'duration' => 60,
            'price' => 100.00,
            'availability' => 'Available',
            'location' => 'Location X',
            'provider' => 'Provider Y',
            'notes' => 'Some notes',
            'dates' => ['2024-08-24'],
            'timeslots' => [
                [
                    ['start' => '09:00:00', 'end' => '10:00:00']
                ]
            ],
        ];

        // Act
        $response = $this->put(route('service.update', ['id' => $service->id]),
$data);

        // Assert
        $response->assertRedirect(route('service-list'));
        $this->assertDatabaseHas('services', [
            'id' => $service->id,
            'name' => 'New Service',
            'description' => 'Updated description',
            'price' => 100.00,
        ]);
    }


    /** @test */
    public function it_can_delete_service()
    {
        $service = Service::factory()->create();

        $this->withoutMiddleware(\App\Http\Middleware\VerifyCsrfToken::class);

        $response = $this->delete(route('service.destroy', ['service' =>
$service->id]));

        $response->assertRedirect(route('service-list'));
        $response->assertSessionHas('success', 'Service and all related data have
been deleted successfully.');
        $this->assertDatabaseMissing('services', ['id' => $service->id]);
    }

    /** @test */
    public function it_displays_dashboard_with_bookings_and_income()
{
    // Arrange: Create related records
    $client = User::factory()->create();
    $service = Service::factory()->create();
    $serviceDate = ServiceDate::factory()->create(['service_id' =>
$service->id]);

    // Ensure the caregiver is also created
    $caregiver = $this->caregiver;
    // Create a booking that references the related records
    $booking = Booking::factory()->create([
        'client_id' => $client->id,
        'caregiver_id' => $caregiver->id,
        'service_id' => $service->id,
        'service_date_id' => $serviceDate->id,
        'status' => 'approved'
    ]);

    // Act: Make a GET request to the dashboard route
    $response = $this->get(route('caregiver-dashboard'));

    // Assert: Check that the response status is 200 and that the view has the
expected data
    $response->assertStatus(200);
    $response->assertViewHas('incomePerMonth');
    $response->assertViewHas('bookingsPerMonth');
    $response->assertViewHas('highestBookingMonth');
    $response->assertViewHas('upcomingAppointments');
}


    /** @test */
    public function it_shows_setting_profile_form()
    {
        $response = $this->get(route('caregiver.profile-edit'));

        $response->assertStatus(200);
        $response->assertViewIs('caregiver.setting-profile');
    }

    /** @test */
    public function it_updates_the_user_profile_successfully_without_image()
    {
        // Create a user and log them in
        $user = User::factory()->create();
        $this->actingAs($user);

        // Prepare the data for the request
        $data = [
            'name' => 'Updated Name',
            'email' => 'zixxuan2002@gmail.com',
            'phone' => '1234567890',
            'gender' => 'male',
            'location' => 'Updated Location',
            'availability' => 'Weekdays',
            'qualification' => ['Updated Qualification'], // Adjusted to array
format
            'experience' => '5 years',
            'about_me' => 'Updated about me',
            // No image provided
        ];

        // Make a PUT request to the profile update route
        $response = $this->put(route('profile.update'), $data);

        // Assertions
        $response->assertStatus(302); // Redirect status
        $response->assertRedirect(route('caregiver.profile-edit')); // Ensure
this matches your actual route
        $response->assertSessionHas('success', 'Profile updated successfully.');

        // Check that the data was updated in the database
        $this->assertDatabaseHas('users', [
            'id' => $user->id,
            'name' => 'Updated Name',
            'email' => 'zixxuan2002@gmail.com',
            'phone_number' => '1234567890',
            'gender' => 'male',
            'location' => 'Updated Location',
            'availability' => 'Weekdays',
            'qualification' => 'Updated Qualification',
            'experience' => '5 years',
            'about_me' => 'Updated about me',
        ]);

        // Check that no image was uploaded
        $this->assertNull($user->fresh()->image);
    }



    /** @test */
    public function it_displays_caregiver_list()
    {
        $caregiver = User::factory()->create(['role' => 'caregiver']);
        $response = $this->get(route('caregiver-list'));

        $response->assertStatus(200);
        $response->assertViewHas('caregivers');
        $response->assertSee($caregiver->name);
    }

    /** @test */
    public function it_can_search_caregivers()
    {
        $caregiver = User::factory()->create(['role' => 'caregiver', 'name' =>
'John Doe']);
        $response = $this->json('GET', route('caregiver.search'), ['query' =>
'John']);

        $response->assertStatus(200);
        $response->assertJsonFragment(['name' => 'John Doe']);
    }

    /** @test */
    public function it_shows_caregiver_profile()
    {
        $caregiver = User::factory()->create(['role' => 'caregiver']);
        $response = $this->get(route('caregiver-profile', ['id' =>
$caregiver->id]));

        $response->assertStatus(200);
        $response->assertViewHas('caregiver');
    }
}
```

```
<?php

namespace Tests\Feature;

use Tests\TestCase;
use Illuminate\Foundation\Testing\RefreshDatabase;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Notification;
use App\Models\User;
use App\Models\Service;
use App\Notifications\CaregiverNotification;
use App\Notifications\PaymentReceivedNotification;
use App\Notifications\ServiceDeleted;

class CaregiverNotificationControllerFeatureTest extends TestCase
{
    use RefreshDatabase;

    protected $user;

    protected function setUp(): void
    {
        parent::setUp();
        $this->user = User::factory()->create(); // Create a user for testing
        Auth::login($this->user);
    }

    /** @test */
public function it_can_fetch_filtered_notifications()
{
    // Create notifications for the user
    $this->user->notify(new CaregiverNotification((object)['id' => 1, 'client' =>
(object)['name' => 'Client A']]));
    $this->user->notify(new PaymentReceivedNotification((object)['id' => 2,
'payment_method' => 'Credit Card', 'payment_date' => now()]));
    $this->user->notify(new ServiceDeleted(Service::factory()->create()));

    // Fetch caregiver notifications
    $response = $this->get(route('caregiver.notifications', ['filter' =>
'caregiver']));
    $response->assertStatus(200);
    $response->assertViewHas('notifications');
    $response->assertSee('You have a new booking from Client A');

    // Fetch payment notifications
    $response = $this->get(route('caregiver.notifications', ['filter' =>
'payment']));
    $response->assertStatus(200);
    $response->assertViewHas('notifications');
    $response->assertSee('A payment has been received for booking ID: 2');

    // Fetch service deleted notifications
    $response = $this->get(route('caregiver.notifications', ['filter' =>
'service_deleted']));
    $response->assertStatus(200);
    $response->assertViewHas('notifications');
    $response->assertSee('The service');
}

/** @test */
public function it_can_clear_filtered_notifications()
```

```
{
    // Create notifications for the user
    $this->user->notify(new CaregiverNotification((object)['id' => 1, 'client' =>
(object)['name' => 'Client A']]));
    $this->user->notify(new PaymentReceivedNotification((object)['id' => 2,
'payment_method' => 'Credit Card', 'payment_date' => now()]));
    $this->user->notify(new ServiceDeleted(Service::factory()->create()));

    // Clear caregiver notifications
    $response = $this->post(route('notifications.clear'), ['filter' =>
'caregiver']);
    $response->assertRedirect(route('caregiver.notifications'));
    $response->assertSessionHas('success', 'Notifications have been cleared.');
    $this->assertDatabaseMissing('notifications', ['type' =>
CaregiverNotification::class]);

    // Clear payment notifications
    $response = $this->post(route('notifications.clear'), ['filter' =>
'payment']);
    $response->assertRedirect(route('caregiver.notifications'));
    $response->assertSessionHas('success', 'Notifications have been cleared.');
    $this->assertDatabaseMissing('notifications', ['type' =>
PaymentReceivedNotification::class]);

    // Clear all notifications
    $response = $this->post(route('notifications.clear'), ['filter' => '']);
    $response->assertRedirect(route('caregiver.notifications'));
    $response->assertSessionHas('success', 'Notifications have been cleared.');
    $this->assertDatabaseMissing('notifications', ['user_id' =>
$this->user->id]);
}

/** @test */
public function it_can_mark_all_notifications_as_read()
{
    // Create unread notifications
    $this->user->notify(new CaregiverNotification((object)['id' => 1, 'client' =>
(object)['name' => 'Client A']]));
    $this->user->notify(new PaymentReceivedNotification((object)['id' => 2,
'payment_method' => 'Credit Card', 'payment_date' => now()]));

    // Mark all notifications as read
    $response = $this->post(route('caregiver.notifications.markRead'));
    $response->assertJson(['status' => 'success']);
    // Fetch the notifications from the database and check that the `read_at`
field is not null
    $notifications = $this->user->notifications()->get();
    foreach ($notifications as $notification) {
        $this->assertNotNull($notification->read_at, "The notification's read_at
timestamp should not be null");
    }
}



/** @test */
public function it_can_get_the_unread_notification_count()
{
    // Create unread notifications
    $this->user->notify(new CaregiverNotification((object)['id' => 1, 'client' =>
(object)['name' => 'Client A']]));
    $this->user->notify(new PaymentReceivedNotification((object)['id' => 2,
'payment_method' => 'Credit Card', 'payment_date' => now()]));

    // Mark all notifications as read
    $response = $this->post(route('caregiver.notifications.markRead'));
    $response->assertJson(['status' => 'success']);

    // Force fresh retrieval from the database
    $unreadNotifications = $this->user->unreadNotifications()->count();

    // Assert that there are no unread notifications left
    $this->assertEquals(0, $unreadNotifications, "There should be no unread
notifications after marking them as read.");
}







}
```

```
<?php

namespace Tests\Feature;
```

```php
use App\Models\Service;
use App\Models\ServiceTimeSlot;
use App\Models\ServiceDate;
use App\Models\Booking;
use Carbon\Carbon;
use Tests\TestCase;
use Illuminate\Foundation\Testing\RefreshDatabase;
use Illuminate\Support\Facades\Auth;
use App\Models\User;

class ClientControllerFeatureTest extends TestCase
{
    use RefreshDatabase;

    protected function setUp(): void
    {
        parent::setUp();

    }

    /** @test */
    public function it_can_show_client_profile()
    {
        // Create a client user
        $client = User::factory()->create(['role' => 'client']);

        // Authenticate as the client
        $this->actingAs($client);

        // Call the route that displays the client profile
        $response = $this->get(route('client-profile', ['id' => $client->id]));

        // Assert that the response status is 200
        $response->assertStatus(200);

        // Assert that the view name is correct
        $response->assertViewIs('caregiver.client-profile');

        // Assert that the view has the correct data
        $response->assertViewHas('client', function ($viewClient) use ($client) {
            return $viewClient->id === $client->id && $viewClient->name ===
$client->name;
        });
    }
```

```
        /** @test */
    public function testIndexDisplaysFeedbacksAndUpcomingAppointments()
    {
        // Create a client user
        $client = User::factory()->create(['role' => 'client']);

        // Create a caregiver user
        $caregiver = User::factory()->create(['role' => 'caregiver']);

        // Create a service
        $service = Service::factory()->create(['caregiver_id' =>
$caregiver->id]);

        // Create service dates
        $futureDate1 = Carbon::now()->addDays(5)->format('Y-m-d');
        $futureDate2 = Carbon::now()->addDays(10)->format('Y-m-d');
        $serviceDate1 = ServiceDate::factory()->create(['date' => $futureDate1]);
        $serviceDate2 = ServiceDate::factory()->create(['date' => $futureDate2]);

        // Create time slots
        $timeSlot1 = ServiceTimeSlot::factory()->create();
        $timeSlot2 = ServiceTimeSlot::factory()->create();

        // Create a booking with feedback
        Booking::factory()->create([
            'client_id' => $client->id,
            'caregiver_id' => $caregiver->id,
            'service_id' => $service->id,
            'service_date_id' => $serviceDate1->id,
            'time_slot_id' => $timeSlot1->id,
            'feedback' => 'Great service!',
            'rating' => 5,
            'status' => 'approved',
        ]);

        // Create an upcoming appointment
        Booking::factory()->create([
            'client_id' => $client->id,
            'caregiver_id' => $caregiver->id,
            'service_id' => $service->id,
            'service_date_id' => $serviceDate2->id,
            'time_slot_id' => $timeSlot2->id,
            'status' => 'accepted',
```

```php
        ]);

        // Authenticate as the client
        $this->actingAs($client);

        // Call the index method
        $response = $this->get('/client-dashboard');

        // Assert that the view is returned
        $response->assertStatus(200);
        $response->assertViewIs('client-dashboard');

        // Retrieve the view data
        $data = $response->original->getData();

        // Debug information for failing assertions
        // dd($data);

        // Assert the feedback is present
        $feedbacks = $data['feedbacks']->toArray();
        $this->assertNotEmpty($feedbacks, 'Feedbacks should not be empty.');
        $this->assertContains('Great service!', array_column($feedbacks,
'feedback'));

        // Assert the upcoming appointment is present
        $appointments = $data['upcomingAppointments']->toArray();
        $this->assertNotEmpty($appointments, 'Upcoming appointments should not be
empty.');

        // Convert expected date to Carbon instance for comparison
        $expectedDate = Carbon::parse($futureDate2)->format('Y-m-d');

        // Assert the future date is present
        $appointmentDates = array_map(function($appointment) {
            return Carbon::parse($appointment['appointment_date'])->format('Y-m-
d');
        }, $appointments);


    }
}
```

```php
<?php
```

```php
use Illuminate\Foundation\Testing\RefreshDatabase;
use Illuminate\Support\Facades\Auth;
use App\Models\User;
use App\Models\Service;
use App\Models\ServiceDate;
use App\Models\Booking;
use Tests\TestCase;

class FeedbackControllerTest extends TestCase
{
    use RefreshDatabase;

    /** @test */
    public function
it_displays_feedback_for_a_specific_service_when_service_id_is_provided()
    {
        // Create a user and authenticate
        $caregiver = User::factory()->create();
        Auth::login($caregiver);

        // Create a client
        $client = User::factory()->create();

        // Create a service
        $service = Service::factory()->create(['caregiver_id' =>
$caregiver->id]);

        // Create a service date
        $serviceDate = ServiceDate::factory()->create();

        // Create a booking with feedback
        Booking::factory()->create([
            'client_id' => $client->id,
            'caregiver_id' => $caregiver->id,
            'service_id' => $service->id,
            'status' => 'approved',
            'feedback' => 'Great service!',
            'rating' => 4,
            'service_date_id' => $serviceDate->id,
        ]);

        $response = $this->get(route('feedback-list', ['service_id' =>
$service->id]));

        $response->assertStatus(200)
```

```
                 ->assertViewHas('feedbacks', function ($feedbacks) use
($service) {
                     return $feedbacks->contains(fn($feedback) =>
$feedback->service_id === $service->id);
                 })
                 ->assertViewHas('services', function ($services) use ($service)
{
                     return $services->contains($service);
                 })
                 ->assertViewHas('serviceId', $service->id);
    }

    /** @test */
    public function
it_displays_feedback_for_all_services_when_no_service_id_is_provided()
    {
        // Create a user and authenticate
        $caregiver = User::factory()->create();
        Auth::login($caregiver);

        // Create a client
        $client = User::factory()->create();

        // Create services
        $service1 = Service::factory()->create(['caregiver_id' =>
$caregiver->id]);
        $service2 = Service::factory()->create(['caregiver_id' =>
$caregiver->id]);

        // Create a service date
        $serviceDate = ServiceDate::factory()->create();

        // Create bookings with feedback
        Booking::factory()->create([
            'client_id' => $client->id,
            'caregiver_id' => $caregiver->id,
            'service_id' => $service1->id,
            'status' => 'approved',
            'feedback' => 'Good job!',
            'rating' => 4,
            'service_date_id' => $serviceDate->id,
        ]);

        Booking::factory()->create([
            'client_id' => $client->id,
            'caregiver_id' => $caregiver->id,
            'service_id' => $service2->id,
            'status' => 'approved',
            'feedback' => 'Excellent service!',
            'rating' => 5,
            'service_date_id' => $serviceDate->id,
        ]);

        $response = $this->get(route('feedback-list'));

        $response->assertStatus(200)
                 ->assertViewHas('feedbacks', function ($feedbacks) use
($service1, $service2) {
                     return $feedbacks->contains(fn($feedback) =>
$feedback->service_id === $service1->id || $feedback->service_id ===
$service2->id);
                 })
                 ->assertViewHas('services', function ($services) use ($service1,
$service2) {
                     return $services->contains($service1) &&
$services->contains($service2);
                 })
                 ->assertViewHas('serviceId', null);
    }
}
```

```
<?php

namespace Tests\Feature;

use Illuminate\Foundation\Testing\RefreshDatabase;
use Illuminate\Foundation\Testing\WithoutMiddleware;
use Tests\TestCase;
use App\Models\User;
use App\Models\Booking;
use App\Models\Service;
use App\Models\ServiceDate;
use Carbon\Carbon;

class HomeControllerFeatureTest extends TestCase
{
    use RefreshDatabase, WithoutMiddleware;
    /** @test */
    public function testDisplaysFeedbackAndUpcomingAppointments()
{
    $response = $this->get('/client-dashboard');

    $response->assertStatus(200); // Ensure the page loads successfully

    // Check for specific content in the HTML
    $response->assertSee('Feedback'); // Adjust the content to what should be
present
    $response->assertSee('Upcoming Appointments'); // Adjust the content to what
should be present
}

/** @test */
public function it_displays_feedback_and_upcoming_appointments()
    {
        // Create necessary data
        $user = User::factory()->create();
        $caregiver = User::factory()->create();
        $service = Service::factory()->create();
        $serviceDate = ServiceDate::factory()->create(['date' =>
Carbon::now()->addDays(1)]);

        // Create a booking with all necessary foreign keys
        $booking = Booking::factory()->create([
            'client_id' => $user->id,
            'caregiver_id' => $caregiver->id,
            'service_id' => $service->id,
            'service_date_id' => $serviceDate->id,
            'status' => 'approved',
            'feedback' => 'Great service!',
            'rating' => 5
        ]);

        // Create additional data for feedback
        Booking::factory()->create([
            'client_id' => $user->id,
            'caregiver_id' => $caregiver->id,
            'service_id' => $service->id,
            'service_date_id' => $serviceDate->id,
            'status' => 'approved',
            'feedback' => 'Another feedback!',
            'rating' => 4
        ]);
        // Make a GET request to the client dashboard
        $response = $this->actingAs($user)->get('/client-dashboard');

        // Assert the response status is 200
        $response->assertStatus(200);

        // Assert the view contains the expected content
        $response->assertSee('Upcoming Appointments');
        $response->assertSee('Great service!');
        $response->assertSee($serviceDate->date->format('d M Y'));

        // Check that the feedback is displayed with the correct rating
        $response->assertSee('<i class="fas fa-star"></i>', false); // Ensure
rating star is present
        $response->assertSee('<i class="far fa-star"></i>', false); // Ensure
empty star is present
    }
}
```

```
<?php

namespace Tests\Feature;

use Tests\TestCase;
use Illuminate\Foundation\Testing\RefreshDatabase;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Notification;
use App\Models\User;
use Illuminate\Notifications\DatabaseNotification;

class NotificationControllerFeatureTest extends TestCase
{
    use RefreshDatabase;

    protected $user;

    protected function setUp(): void
    {
        parent::setUp();
        $this->user = User::factory()->create(); // Create a user for testing
        Auth::login($this->user);
    }

    /** @test */
    public function it_can_list_notifications()
    {
        Notification::fake();
        $notification = $this->user->notify(new
\App\Notifications\BookingStatusUpdated((object)['id' => 1, 'status' =>
'confirmed']));

        $response = $this->get(route('notifications'));

        $response->assertStatus(200);
        $response->assertViewHas('notifications');
    }

    /** @test */
    public function it_can_clear_all_notifications()
    {
        Notification::fake();
        $this->user->notify(new
\App\Notifications\BookingStatusUpdated((object)['id' => 1, 'status' =>
'confirmed']));
        $this->user->notify(new
\App\Notifications\BookingStatusUpdated((object)['id' => 2, 'status' =>
'pending']));

        $response = $this->post(route('client-notifications.clear'));

        $response->assertRedirect(route('notifications'));
        $response->assertSessionHas('success', 'All notifications have been
cleared.');
        $this->assertDatabaseMissing('notifications', ['notifiable_id' =>
$this->user->id]);
    }

    /** @test */
    public function it_can_mark_all_notifications_as_read()
    {
        // Create notifications directly
        $this->user->notify(new
\App\Notifications\BookingStatusUpdated((object)['id' => 1, 'status' =>
'confirmed']));
        $this->user->notify(new
\App\Notifications\BookingStatusUpdated((object)['id' => 2, 'status' =>
'pending']));

        // Fetch notifications from the database
        $notifications = $this->user->notifications;

        // Mark all notifications as read
        $response = $this->post(route('notifications.markRead'));

        $response->assertJson(['status' => 'success']);

        // Assert that all notifications are marked as read
        foreach ($notifications as $notification) {
            $this->assertNotNull($notification->fresh()->read_at);
        }
    }

    /** @test */
    public function it_can_get_unread_notification_count()
    {
        // Create notifications directly
        $this->user->notify(new  \App\Notifications\BookingStatusUpdated((object)
['id' => 1, 'status' => 'confirmed']));
        $this->user->notify(new  \App\Notifications\BookingStatusUpdated((object)
['id' => 2, 'status' => 'pending']));

        // Fetch notifications and mark one as read
        $notifications = $this->user->notifications;
        if ($notifications->count() > 0) {
            $notifications->first()->markAsRead(); // Mark the first notification
as read
        }

        $response = $this->getJson(route('notifications.unreadCount'));

        $response->assertStatus(200);
        $response->assertJson(['unread_count' =>
$this->user->unreadNotifications()->count()]);
    }
}
```

```
<?php
namespace Tests\Feature;

use Illuminate\Foundation\Testing\RefreshDatabase;
use Tests\TestCase;
use App\Models\Booking;
use App\Models\Service;
use App\Models\User;
use Illuminate\Support\Facades\Auth;

class PaymentControllerFeatureTest extends TestCase
{
    use RefreshDatabase;

  /** @test */
public function it_displays_transaction_history_page_for_caregivers()
{
    // Simulate authenticated caregiver
    $caregiver = User::factory()->create();
    Auth::login($caregiver);

    // Create necessary data
    $service = Service::factory()->create(['price' => 100.00]);
    $client = User::factory()->create();
    $booking = Booking::factory()->create([
        'caregiver_id' => $caregiver->id,
        'payment_date' => now()->subDays(5),
        'service_id' => $service->id,
        'client_id' => $client->id,
        'payment_method' => 'credit_card',
        'status' => 'approved',
    ]);

    // Regular request to the correct URL
    $response = $this->get('/transaction-history?from_date=' .
now()->subDays(10)->toDateString() . '&to_date=' . now()->toDateString());

    $response->assertStatus(200)
             ->assertViewIs('caregiver.transaction-history')
             ->assertViewHas('history', function ($history) use ($booking,
$service, $client) {
                 return $history[0]['date'] ===
$booking->payment_date->format('d-m-Y') &&
                        $history[0]['amount'] === 'RM ' .
number_format($service->price, 2) &&
                        $history[0]['client'] === $client->name &&
                        $history[0]['service'] === $service->name &&
                        $history[0]['payment_method'] ===
ucfirst($booking->payment_method);
             })
             ->assertViewHas('userName', $caregiver->name);
```

```php
}


/** @test */
public function it_returns_json_for_caregivers_when_filtering_payments()
{
    // Simulate authenticated caregiver
    $caregiver = User::factory()->create();
    Auth::login($caregiver);

    // Create necessary data
    $service = Service::factory()->create(['price' => 100.00]);
    $client = User::factory()->create();
    $booking = Booking::factory()->create([
        'caregiver_id' => $caregiver->id,
        'payment_date' => now()->subDays(5),
        'service_id' => $service->id,
        'client_id' => $client->id,
        'payment_method' => 'credit_card',
        'status' => 'approved',
    ]);

    // Simulate AJAX request to the correct URL
    $response = $this->json('GET', '/filter-payments', [
        'from_date' => now()->subDays(10)->toDateString(),
        'to_date' => now()->toDateString()
    ]);

    $response->assertStatus(200)
             ->assertHeader('Content-Type', 'application/json')
             ->assertJsonFragment([
                 'date' => $booking->payment_date->format('d-m-Y'),
                 'amount' => 'RM ' . number_format($service->price, 2),
                 'client' => $client->name,
                 'service' => $service->name,
                 'payment_method' => ucfirst($booking->payment_method),
             ]);
}

}
```

```php
<?php
```

```
namespace Tests\Feature;

use Illuminate\Foundation\Testing\RefreshDatabase;
use Illuminate\Http\UploadedFile;
use Illuminate\Support\Facades\Storage;
use Illuminate\Support\Facades\Auth;
use Tests\TestCase;
use App\Models\User;

class ProfileControllerFeatureTest extends TestCase
{
    use RefreshDatabase;

/** @test */
public function user_can_update_their_profile_without_image()
{
    // Set up a fake storage disk
    Storage::fake('public');

    // Create a user and log in
    $user = User::factory()->create();
    $this->actingAs($user);

    // Define profile update data
    $data = [
        'name' => 'Updated Name', // Ensure this matches the value you expect
        'email' => 'zixuan@1utar.my',
        'phone' => '0123456789',
        'gender' => 'female',
        'location' => 'Updated Location',
    ];

    // Send a PUT request to update the profile
    $response = $this->put(route('profile-update'), $data);

    // Refresh the user instance
    $user->refresh();

    // Log the updated user details
    \Log::info('User after update:', $user->toArray());

    // Assert the user details have been updated
    $this->assertEquals('Updated Name', $user->name);
    $this->assertEquals('zixuan@1utar.my', $user->email);
    $this->assertEquals('0123456789', $user->phone_number);
    $this->assertEquals('female', $user->gender);
    $this->assertEquals('Updated Location', $user->location);

    // Assert redirection and success message
    $response->assertRedirect(route('profile.edit'));
    $response->assertSessionHas('success', 'Profile updated successfully.');
}


    /** @test */
    public function user_can_update_their_profile_with_image()
    {
        // Set up a fake storage disk
        Storage::fake('public');

        // Create a user and log in
        $user = User::factory()->create();
        $this->actingAs($user);

        // Define profile update data with an image
        $image = UploadedFile::fake()->image('profile.jpg');
        $data = [
            'name' => 'Updated Name',
            'email' => 'zixuan@1utar.my',
            'phone' => '0123456789',
            'gender' => 'female',
            'location' => 'Updated Location',
            // 'image' => $image,
        ];

        // Send a PUT request to update the profile
        $response = $this->put(route('profile.update'), $data);

        // Refresh the user instance
        $user->refresh();

        // Assert the user details have been updated
        $this->assertEquals('Updated Name', $user->name);
        $this->assertEquals('zixuan@1utar.my', $user->email);
        $this->assertEquals('0123456789', $user->phone_number);
        $this->assertEquals('female', $user->gender);
        $this->assertEquals('Updated Location', $user->location);
        // $this->assertNotNull($user->image);
        // Storage::disk('public')->assertExists('images/' . $user->image);
        // Assert redirection and success message
        $response->assertRedirect(route('caregiver.profile-edit'));
        $response->assertSessionHas('success', 'Profile updated successfully.');
    }
}
```

```
<?php

namespace Tests\Feature;

use Illuminate\Foundation\Testing\RefreshDatabase;
use Illuminate\Support\Facades\Notification;
use Illuminate\Support\Facades\Storage;
use Illuminate\Http\UploadedFile;
use Tests\TestCase;
use App\Models\Service;
use App\Models\ServiceDate;
use App\Models\User;
use App\Models\ServiceTimeslot;
use App\Models\Booking;
use App\Notifications\ServiceDeleted;

use Illuminate\Foundation\Testing\TestResponse;


class ServiceControllerFeatureTest extends TestCase
{
    use RefreshDatabase;
    /**
     * A basic unit test example.
     *
     * @return void
     */

     public function setUp(): void
{
    parent::setUp();

    // Disable CSRF protection for tests
    $this->withoutMiddleware([\App\Http\Middleware\VerifyCsrfToken::class]);
}
```

```
/** @test */
public function it_stores_a_service_successfully()
{
    // Arrange
    Storage::fake('public'); // For testing file uploads
    $user = \App\Models\User::factory()->create(); // Assuming you have a User
factory

    $requestData = [
        'name' => 'Test Service',
        'description' => 'Service description',
        'service_type' => 'Type',
        'duration' => 60,
        'price' => 100,
        'availability' => 'Available',
        'location' => 'Location',
        'provider' => 'Provider',
        'notes' => 'Some notes',
        // 'image' => UploadedFile::fake()->image('image.jpg'), // Fake image for
upload
        'dates' => ['2024-08-25', '2024-08-26'],
        'timeslots' => [
            [
                ['start' => '08:00', 'end' => '12:00'],
                ['start' => '13:00', 'end' => '17:00'],
            ],
            [
                ['start' => '09:00', 'end' => '11:00'],
            ],
        ],
    ];

    // Act
    $response = $this->actingAs($user)->post(route('save-service'),
$requestData);

    // Assert
    $response->assertRedirect(route('service-list'));
    $response->assertSessionHas('success', 'Service added successfully!');

    $this->assertDatabaseHas('services', [
        'name' => 'Test Service',
        'description' => 'Service description',
        'service_type' => 'Type',
        'duration' => 60,
        'price' => 100,
        'availability' => 'Available',
        'location' => 'Location',
        'provider' => 'Provider',
        'notes' => 'Some notes',
    ]);

    $service = Service::first();

    $this->assertNotNull($service);
    $this->assertDatabaseHas('service_dates', ['service_id' => $service->id,
'date' => '2024-08-25']);
    $this->assertDatabaseHas('service_dates', ['service_id' => $service->id,
'date' => '2024-08-26']);
    $this->assertDatabaseHas('service_timeslots', ['start_time' => '08:00',
'end_time' => '12:00']);
    $this->assertDatabaseHas('service_timeslots', ['start_time' => '13:00',
'end_time' => '17:00']);
    $this->assertDatabaseHas('service_timeslots', ['start_time' => '09:00',
'end_time' => '11:00']);

    // Check if the image was uploaded
    // Storage::disk('public')->assertExists('images/' . $service->image);
}

/** @test */
public function it_can_search_services_by_type()
{
    // Arrange: Create a service with a specific type
    $service = Service::factory()->create([
        'service_type' => 'Cleaning',
        'duration' => '2 hours',
        'price' => 50,
        'availability' => 'available',
        'location' => 'New York',
        'provider' => 'John Doe'
    ]);

    // Act: Perform the search with a query parameter
    $response = $this->get('/search?service_type=Cleaning');

    // Assert: The service should be in the response
    $response->assertStatus(200); // Directly use status code
    $response->assertSee($service->service_type);
}
```

```
/** @test */
public function it_can_search_services_by_duration()
{
    // Arrange: Create a service with a specific duration
    $service = Service::factory()->create([
        'service_type' => 'Cleaning',
        'duration' => '60',
        'price' => 50,
        'availability' => 'available',
        'location' => 'New York',
        'provider' => 'John Doe'
    ]);

    // Act: Perform the search with a query parameter
    $response = $this->get('/search?duration=60');

    // Assert: The service should be in the response
    $response->assertStatus(200);
    $response->assertSee($service->duration);
}

/** @test */
public function it_can_search_services_by_price()
{
    // Arrange: Create a service with a specific price
    $service = Service::factory()->create([
        'service_type' => 'Cleaning',
        'duration' => '2 hours',
        'price' => 50,
        'availability' => 'available',
        'location' => 'New York',
        'provider' => 'John Doe'
    ]);

    // Act: Perform the search with a query parameter
    $response = $this->get('/search?price=50');

    // Assert: The service should be in the response
    $response->assertStatus(200);
    $response->assertSee($service->price);
}

/** @test */
public function it_can_search_services_by_availability()
```

```
{
    // Arrange: Create a service with a specific availability
    $service = Service::factory()->create([
        'service_type' => 'Cleaning',
        'duration' => '2 hours',
        'price' => 50,
        'availability' => 'available',
        'location' => 'New York',
        'provider' => 'John Doe'
    ]);

    // Arrange: Create a service date and timeslot with availability
    $serviceDate = ServiceDate::factory()->create(['service_id' =>
$service->id]);
    ServiceTimeslot::factory()->create([
        'service_date_id' => $serviceDate->id,
        'availability' => 0, // 0 means available
    ]);

    // Act: Perform the search with a query parameter
    $response = $this->get('/search?availability=available');

    // Assert: The service should be in the response
    $response->assertStatus(200);
    $response->assertSee($service->availability);
}

/** @test */
public function it_can_search_services_by_location()
{
    // Arrange: Create a service with a specific location
    $service = Service::factory()->create([
        'service_type' => 'Cleaning',
        'duration' => '2 hours',
        'price' => 50,
        'availability' => 'available',
        'location' => 'Cheras',
        'provider' => 'John Doe'
    ]);

    // Act: Perform the search with a query parameter
    $response = $this->get('/search?location=Cheras');

    // Assert: The service should be in the response
    $response->assertStatus(200);
    $response->assertSee($service->location);
}




public function test_view_feedback()
{
    // Create necessary related models first
    $service = Service::factory()->create();
    $client = User::factory()->create();
    $caregiver = User::factory()->create();
    $serviceDate = ServiceDate::factory()->create(['service_id' =>
$service->id]);

    // Now create the booking
    $booking = Booking::factory()->create([
        'service_id' => $service->id,
        'client_id' => $client->id,
        'caregiver_id' => $caregiver->id,
        'service_date_id' => $serviceDate->id,
        'feedback' => 'Great service!'
    ]);

    // Run the test
    $response = $this->get(route('service.feedback', ['id' => $service->id]));

    $response->assertStatus(200);
    $response->assertViewHas('feedback', function ($feedbacks) use ($booking) {
        return $feedbacks->contains($booking);
    });
}


    /** @test */
    public function it_can_delete_a_service_and_notify_caregiver_and_client()
{
    // Set up fake notifications
    Notification::fake();

    // Create a caregiver and a client
    $caregiver = User::factory()->create();
    $client = User::factory()->create();

    // Create a service and a service date
    $service = Service::factory()->create(['caregiver_id' => $caregiver->id]);
    $serviceDate = ServiceDate::factory()->create(['service_id' =>
$service->id]);

    // Create a booking for that service
    $booking = Booking::factory()->create([
        'service_id' => $service->id,
        'client_id' => $client->id,
        'caregiver_id' => $caregiver->id,
        'service_date_id' => $serviceDate->id,
        'status' => 'pending'
    ]);

    // Act as an admin user (or whoever has the permissions to delete a service)
    $admin = User::factory()->create(['role' => 'administrator']);
    $this->actingAs($admin);

    // Send the DELETE request to delete the service
    $response = $this->delete(route('services.destroy', $service->id));

    // Assert the service is deleted from the database
    $this->assertDatabaseMissing('services', ['id' => $service->id]);

    // Assert that notifications were sent
    Notification::assertSentTo(
        $caregiver,
        ServiceDeleted::class
    );
    Notification::assertSentTo(
        $client,
        ServiceDeleted::class
    );

    // Assert redirect and success message
    $response->assertRedirect(route('admin.services'));
    $response->assertSessionHas('success', 'Service deleted successfully.');
}




    /** @test */
    public function it_displays_a_list_of_services_to_an_admin()
    {
        // Create some services
        $services = Service::factory()->count(3)->create();

        // Create an admin user
        $admin = User::factory()->create(['role' => 'administrator']);

        // Act as the admin user
        $this->actingAs($admin);

        // Send a GET request to the index route
        $response = $this->get(route('admin.services'));

        // Assert the response status
        $response->assertStatus(200);

        // Assert that the view is correct
        $response->assertViewIs('admin.services-list');

        // Assert that the view receives the expected data
        $response->assertViewHas('services', function ($viewServices) use
($services) {
            // Ensure the view services match the expected services
            foreach ($services as $service) {
                $this->assertTrue($viewServices->contains('id', $service->id));
                $this->assertTrue($viewServices->contains('name',
$service->name));
                $this->assertTrue($viewServices->contains('description',
$service->description));
                // Add assertions for other attributes if needed
            }
            return true;
        });

        // Optionally, assert that the service data is present in the response
        foreach ($services as $service) {
            $response->assertSee($service->name);
            $response->assertSee($service->description);
            // Add assertions for other attributes if needed
        }
    }




}
```

```php
<?php

namespace Tests\Feature;

use Tests\TestCase;
use Illuminate\Foundation\Testing\RefreshDatabase;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Notification;
use Illuminate\Support\Carbon;
use App\Models\Booking;
use App\Models\Service;
use App\Models\User;
use App\Models\Caregiver;

class TransactionControllerFeatureTest extends TestCase
{
    use RefreshDatabase;

    protected function setUp(): void
    {
        parent::setUp();

        // Create and authenticate a user
        $this->user = User::factory()->create();
        $this->actingAs($this->user);
    }

    /** @test */
    public function it_can_show_transaction_history()
    {
        // Arrange
        $caregiver = User::factory()->create(['role' => 'caregiver']);
        $service = Service::factory()->create(['name' => 'Service 1', 'price' =>
100.00]);
        $booking = Booking::factory()->create([
            'client_id' => $this->user->id,
            'caregiver_id' => $caregiver->id,
            'payment_method' => 'credit_card',
            'payment_date' => Carbon::now()->toDateString(),
            'service_id' => $service->id,
            'status' => 'approved',
        ]);

        // Act
        $response = $this->get(route('transaction_history'));
        // Assert
        $response->assertStatus(200);
        $response->assertViewIs('transaction_history');
        $response->assertViewHas('history', function ($history) use ($booking,
$caregiver) {
            return $history->contains(function ($item) use ($booking, $caregiver)
{
                return $item['date'] === $booking->payment_date->format('d-m-Y')
&&
                       $item['amount'] === $booking->service->price &&
                       $item['caregiver'] === $caregiver->name &&
                       $item['service'] === $booking->service->name &&
                       $item['payment_method'] ===
ucfirst($booking->payment_method);
            });
        });
    }

    /** @test */
    public function it_can_filter_transactions_by_date_and_return_view()
    {
        // Arrange
        $user = User::factory()->create();
        Auth::login($user);
        $service = Service::factory()->create(['price' => 100.00]);
        $caregiver = User::factory()->create(['role' => 'caregiver']);
        $client = User::factory()->create(['role' => 'client']);
        $booking = Booking::factory()->create([
            'client_id' => $client->id,
            'caregiver_id' => $caregiver->id,
            'service_id' => $service->id,
            'service_date_id' => 1,
            'status' => 'approved',
            'feedback' => 'Great service!',
            'rating' => 5,
            'payment_date' => now()->subDays(5),
            'payment_method' => 'credit_card',
        ]);

        // Act
        $response = $this->get('/filter-transactions?from_date=' .
now()->subDays(10)->toDateString() . '&to_date=' . now()->toDateString());

        // Assert
        $response->assertStatus(200);
        $response->assertViewIs('transaction_history');
        $response->assertViewHas('history', function ($history) use ($booking,
$service, $caregiver) {
            return $history[0]['date'] === $booking->payment_date->format('d-m-
Y') &&
                   $history[0]['amount'] === 'RM ' .
number_format($service->price, 2) &&
                   $history[0]['caregiver'] === $caregiver->name &&
                   $history[0]['service'] === $service->name &&
                   $history[0]['payment_method'] ===
ucfirst($booking->payment_method);
        });
        $response->assertViewHas('userName', $user->name);
    }
}
```