PERSONAL FINANCE MANAGEMENT APPLICATION FOR UNIVERSITY STUDENTS

BY

NG SIAN HONG

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF INFORMATION SYSTEMS (HONOURS) INFORMATION SYSTEM ENGINNEERING

Faculty of Information and Communication Technology

(Kampar Campus)

JAN 2022

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: Personal Finance Management Application for University Students

Academic Session: Jan 2022

I NG SIAN HONG

(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,

(Author's signature) (Supervisor's signature)

Address:
NO 387, LORONG 16,
42600 JENJAROM,
SELANGOR.

Ts Dr Chan Lee Kwun
Supervisor's name

Date: 22nd April 2022 **Date**: 22nd April 2022

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

UNIVERSITI TUNKU ABDUL RAHMAN

Date: 22nd April 2022

SUBMISSION OF FINAL YEAR PROJECT

It is hereby certified that __Ng Sian Hong___ (ID No: __19ACB04788__) has completed this final year project entitled "___Personal Finance Management Application for University Students ___" under the supervision of __Ts Dr Chan Lee Kwun_ __ (Supervisor) from the Department of ___Information Systems___, Faculty of __Information and Communication Technology_____ , and ____Ms Yap Seok Gee____ (Co-Supervisor) from the Department of Information Systems, Faculty of _____Information and Communication Technology__.

I understand that University will upload softcopy of my final year project in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

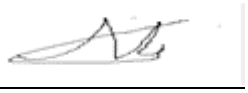
Yours truly,

_____

Ng Sian Hong

DECLARATION OF ORIGINALITY

I declare that this report entitled "**PERSONAL FINANCE MANAGEMETN APPLICATION FOR UNIVERSITY STUDENTS**" is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature : 

Name : NG SIAN HONG

Date : 22nd April 2022

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and very great appreciation to my supervisors, Ts Dr Chan Lee Kwun and moderator, Ms Yap Seok Gee, who has given me this brilliant opportunity to engage in an application project. Both of them all inspired me and provided me with this once-in-a-lifetime opportunity to participate in a real-life application project that I have always been passionate about. Ts Dr Chan supported me enormously when I had new ideas and helped me solve doubts and issues when needed with their constructive feedback. In addition, thanks to Ts Dr Chan for her advice and assistance in keeping and helping my progress on schedule and suggestions. It is my first step to establish a career in the application development field. A million thanks to both of you.

For the family who has always supported me, the gratitude to everyone cannot be expressed in words. Thank you for supporting me all the time in the same boat. Thank you for showing up when I need a listener. Thank you for always believing in me and encouraging me to achieve my goals. Without your spiritual support, this project would be impossible to complete.

Thank you to a very special person in my life who shall remain anonymous and thank her for her patience, unconditional support and advice, and for standing with me in difficult times. Finally, I must say thanks to my parents and my family for their love, support, and continuous encouragement throughout the course.

I would also like to thank everyone who helped me turn imaginative ideas far beyond simple into concrete things. Thanks to everyone who supported me in my university life.

ABSTRACT

Personal finance management is very important for everyone in this world, especially for university students. By definition, personal finance management not only focuses and involves the handling of investments but is also used to keep track of transactions and to do budgeting. University students have to learn about how to manage their money carefully since they are almost working adults and need to learn to become independent. In addition, each of the transactions records involves a different situation and location. Therefore, it is not easy to keep track of the transactions. Despite this, most university students manage their personal finance by memorizing the transactions using their brains. Obviously, this is not an effective way to handle personal finance management.

Currently, there are several personal finance applications in the market that are used to help people manage personal finance. However, most of them are fairly complicated and hard to grasp, especially when dealing with interest rates and loans. In fact, university students are better off without the involvement of interest rates and loans as they are more likely to only keep track of transactions and perform budgeting transactions. Therefore, the personal finance applications that are currently available in the market are limited to specific aspects of personal finance and thus, are not suitable for university students.

The objective of this project is to address the problems above by developing a useful and secure desktop-based personal finance management application. This application allows users to maintain and keep track of their personal finance such as to manage incomes and expenses as well as to do budgeting to avoid unnecessary transactions finance. Lastly, clear, and easy user interfaces are provided to reduces the complexity involved in managing their personal finance.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

CRUD	Create, Read, Update, Delete
ERD	Entity Relationship Diagram

CHAPTER 1

INTRODUCTION

In this chapter, the background information of this project, the problem statement and motivation of this project, the aims and contributions of this projects as well as the outline of this project's report will be discussed in detail.

1.1 Background Information

In this materialistic era, money is arguably the most essential aspect of one's life. Money is used in transactions, businesses, or even investments. These activities require a large sum of money, usually paid in one lump sum. To ensure the security and convenience of these activities, bank organisations exist to help the society in these aspects. However, in real life, private everyday activities that involve money would not be managed by the bank. Examples of these private activities include monthly budgeting for food expenses, daily-use supplies and et cetera. Based on the article written by Jeff Smith [1], learning to manage money early will prepare university students for the stress-free financial future they deserve. Hence, university students do not only need to increase knowledge in university, but also need to manage finances in adult life, save money and avoid future debts as well.

Traditionally, due to the lack of adequate technologies, people used the manual pen-and-paper method to jot down these private activities. However, with the advancement in technologies today, people are more reliant on software and technologies to benefit themselves. Therefore, the creation of this personal finance management application aims to aid university student in managing the debits and credits of their everyday activities.

1.2 Problem Statement and Motivation

Everyone has a critical growth time in their lives – this typically takes place when one steps foot into his or her university life. For many young adults, studying in a university is their first experience of managing money because they must learn to live alone and manage their own expenses by themselves. Two of the crucial fields that frequently lead to difficulties for university students are budget management and personal finance management. The reason behind this is that university students live a freer live-in terms of handling money as the wills of their parents no longer bound them. Moreover, students usually never take full responsibility for their personal finances when they begin their tertiary education.

First and foremost, many university students use the manual pen-and-paper method to record their daily expenses and income. The method behind this is similar to how accountants use the double-entry concept to keep a record of their financial needs manually. The phrase "double entry concept" refers to a method of transactions in terms of debit(income) and credits(expenses). However, there are several problems associated with this method of keeping track of one's personal finance. For example, one of the problems is that the papers used in the financial records are vulnerable to physical wear and tear. Moreover, the records on the papers might be lost entirely if the paper comes into contact with any sort of liquid, such as water.

Secondly, the security of using physical papers to keep track of one's personal finance is questionable. Papers are one of the everyday things that people will use; therefore, some students might unintentionally leave these pieces of papers all over the place and forget about them afterwards. On a more serious note, these papers could be picked up by random strangers, who will instantly get a glimpse of the paper's contents. This exposure of personal information puts a student's financial details at high risk.

Thirdly, most university students take a dislike towards long lists of records and words, since they have to deal with assignments all day. Also, the messy user interface of existing applications will adversely affect the university students in managing their personal finance as they will be discouraged to use complicated applications. For example, after a long day of doing assignments, university students might not feel encouraged to face even more confusing figures and records when trying to keep track of their income and expenses records. Due to the unappealing and monotonous

appearance of manual personal finance management methods, university students might give up on managing their personal finance altogether.

Last but not least, one common mistake that university students will make in their personal finance management is that they do not know where their money is going. Normally, university students spend their money without doing any records. Therefore, after a long time, they will realise that their money is slowly slipping away without their consent. For example, the university student spends RM8 on his or her lunch and thinks that he or she might be able to remember this spending. However, after a few days, the memory of the spending is lost and at the end of the month, the student might be confused as to where majority of the money went. Once universities students run out of money, they might ask for money from their parents, which creates a burden for their family and indirectly causes their parents to lose money.

This personal finance management application covers all aspects of one's money management, savings and budgeting. Next, this application also helps one to meet long-term and short-term personal financial goals. Examples of this include the gauging of whether one has enough sufficient short-term financial needs, managing retiring plans and even saving up money for future goals. The application analyses one's income, expenses, life requirements, and personal goals and effectively develop a plan that is able to meet these needs within one's financial constraints. Also, this application can help university students avoid some financial mistakes in real life. With the help of this application, university students can review their previous income and expenses records in order to find out unnecessary mistakes. Therefore, the application delivered in this project not only provides a useful way of managing personal finance but also help the university students know more about personal financial knowledge

1.3 Project Scope and Direction

In order to provide an adequate personal finance management application to university students, the outcome of this project is a novel model of a desktop-based application that aids university students in managing their personal finance effectively. The project is expected to provide a simple-to-use platform that helps university students manage personal finance and budgeting. The coverage of this project includes students who are currently pursuing their degrees in universities. Although this project

may be suitable for secondary school students around 17 or 18 years old on the verge of entering universities, the primary targeted audience of this personal finance management is students between the ages of 19 to 22. As a result, this project aims to tackle the lack of a convenient platform for university students to keep track of their income and expenses. The works in this project are based on the assumption that the majority of university students are lazy to manage their personal finance due to the need for manual recording methods and budget memorization.

1.4 Objectives

The primary aim of this project is to develop the use of personal finance management and budgeting system into a desktop application to safely track personal finances. Listed below are the main objectives of this project:

1. To identify the problems of manual paper-and-pen transaction management associated with personal finance.
2. To design a prototype that is useful to improve the personal finance management of university students.
3. To develop a software prototype for university students with personal budgeting and transaction tracking functionalities.

1.5 Impact, Significance and Contributions

With this personal finance management application, university students will have a better understanding of how to manage their personal finance. Managing money and sticking to pre-planned budgets may be daunting to university students. However, this project allows university students to track budgets, expenses and income easily, without the need for proficient knowledge in financial education. Equipped with the fundamental finance managing functions such as add, delete and update, this project allows students to customise their own personal financial plans with ease. Taking into consideration students' dislikes towards plain digital data forms, this project integrates the option to generate graphical representations of overall views of personal finance. Thus, this project emphasises heavily on implementing finance management features conveniently and effectively for university students. Furthermore, this project helps

university students gain confidence in making financial decisions. With this personal finance management application's financial report generation feature, students will always have an idea of their remaining budget and can make important financial decisions confidently in the future.

1.6 Report Organization

The details of this project can be comprehended through the following chapters. In Chapter 2, the methods of personal finance management for university students, existing systems in the market, and comparison tables to compare the existing systems are described in detail. Furthermore, in Chapter 3, the method and approach of this application are illustrated in detail. Next, Chapter 4 is mainly discussed the system design by including the block diagram of this application, system components interaction operations and user interface design. Moreover, Chapter 5 mainly describes the completed work that has been completed and the corresponding results. Moving on, Chapter 6 is discussed the application testing result and performance metrics of this application. Lastly, the conclusion of this entire project will be discussed in Chapter 7.

CHAPTER 2

Literature Review

In this chapter, the basic background information is presented by including the existing methodologies of some personal finance managements. Next, the existing desktop-based and web-based personal finance management applications are reviewed and compared.

2.1 Personal Finance Management Methodologies

Personal finance management applications can help university students manage their personal finance efficiently since they able to control their finance and keep track of expenses and incomes. For example, university students can plan their transaction budgets so that they always know how much savings they currently have. Generally, personal finance applications are commonly found through online app stores and even the World Wide Web. There are many people who use these kinds of applications to manage their personal finance. Hence, most of the existing personal finance management applications had incorporated some useful functions such as managing income, expenses, investments, loans, and etcetera which can consider all aspects of personal finance. In this section, the fundamental methods involved in personal finance management are discussed.

First of all, the common way to manage personal finance is through the use of basic money management with manual data entry [2]. In this method, normally people have to manually key in the balance, transactions as well as income to track their current financial situation. This way of personal finance management is commonly selected because of its ease of use and simple calculations. By far, this is the most traditional method used to keep track of personal finance details.

Next, another method of personal finance management is money investment management [2]. Money investment management uses the starting investment amount to calculate the future money by using interest rate(return rate), compound as well as additional contribution amount. Basically, this method is all about assisting people to

manage their investments involving complex calculations. because the cash flow of this method normally involved a large sum of money, it often requires accurate calculations to manage.

The third method to manage personal finance is loan management. Examples of loans include credit cards, household loans, car loans and etcetera. Normally, loans have to be paid back monthly. In order to ensure that people have enough money to pay their loans, loan management methods are often used to calculate the amount that needs to be paid in the future so that people can estimate the money that they have to spend. In addition, loan management methods also involve complex calculations to calculate the monthly payment, such as the initial loan amount, interest rates, total loan amount and etcetera. Hence, this personal finance management method is more suitable for people with multiple loans to pay back every month.

Furthermore, the fourth method used to manage personal finance is daily and monthly budgeting [2]. Basically, this method allows people to make monthly and daily budget plans and view the current balance distributions. Using this method, people can reduce the risk of wasting money at a large expense. The reason is because this method allows people to plan where the money should be spent. For example, with the use of budget planning, people will not try to spend money exceeding the budget planned because if the budget is exceeded, people would know the money to-be-spent is not part of the budget and might be for unnecessary items. Additionally, monthly budget planning can also allow people to maintain their personal finance in an average line. This way, people can save their money and use it in the future instead of spending money in the wrong places.

However, for university students, the features of managing investments and loans might not be necessary at this point of time. The reason is because university students need to focus on increasing their personal finance knowledge instead of having to know how to manage loans and investments. Instead, university students have a more significant need for the use of functions such as managing income and expenses. Hence, in this literature review, various personal finance management applications will be reviewed in terms of the features that are beneficial to university students – basic income and expenses management, graph and chart generation, expenses filtering and etcetera.

2.2 Review of Existing Personal Finance Management Application

2.2.1 Goodbudget [3]

The first application to be reviewed is called Goodbudget. Goodbudget is a budgeting system that is used to track the expenses and income using virtual envelopes. A virtual envelope is used to manage the expenses. There consists of two parts in the virtual envelopes, which are monthly expenses and annual expenses. Basically, in monthly expenses, the user was allowed to create several sub-virtual envelopes, and these would be used to keep track of the daily expenses such as rent, groceries and food expenses. Furthermore, annual expenses were quite similar to monthly expenses, but instead, the former was used to record huge amounts of expenses and substantial loans to pay off. Examples of annual expenses included car instalments, insurances, and etcetera. Figure 2.1.1 shows an example of virtual envelopes to manage the expenses records.

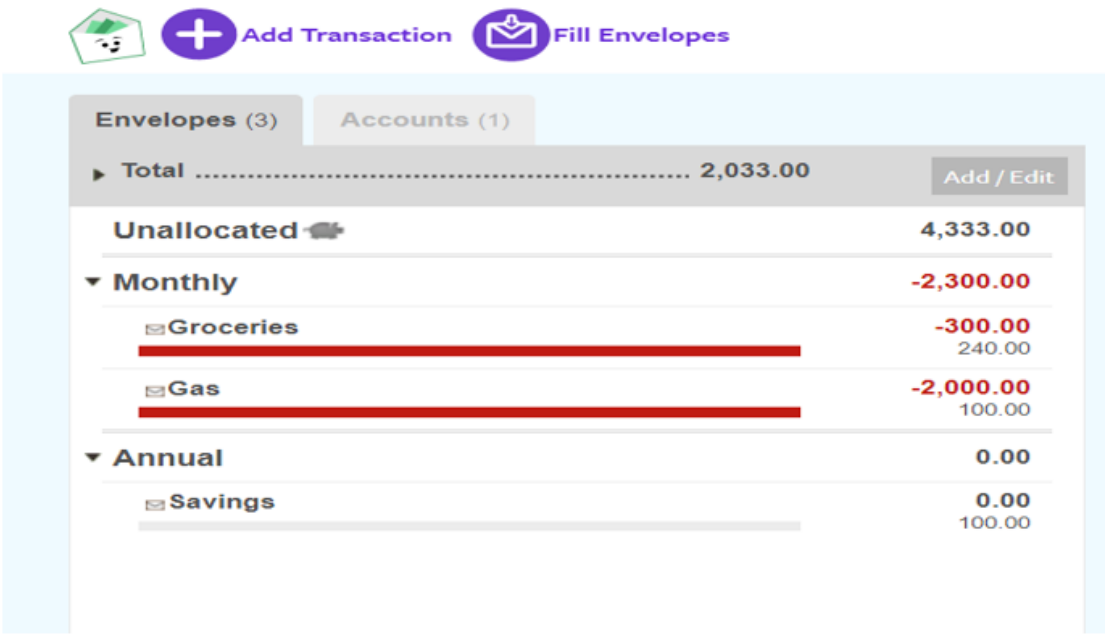


Figure 2.1.1 Virtual envelopes for expenses management

As for the income management module, it was not divided into different types of income. All income records that were added would be merged into a single list which given the user an apparent view, as an example is shown in figure 2.1.2.

Additionally, if the user enters the wrong amount, Goodbudget allowed the user to modify entered values.

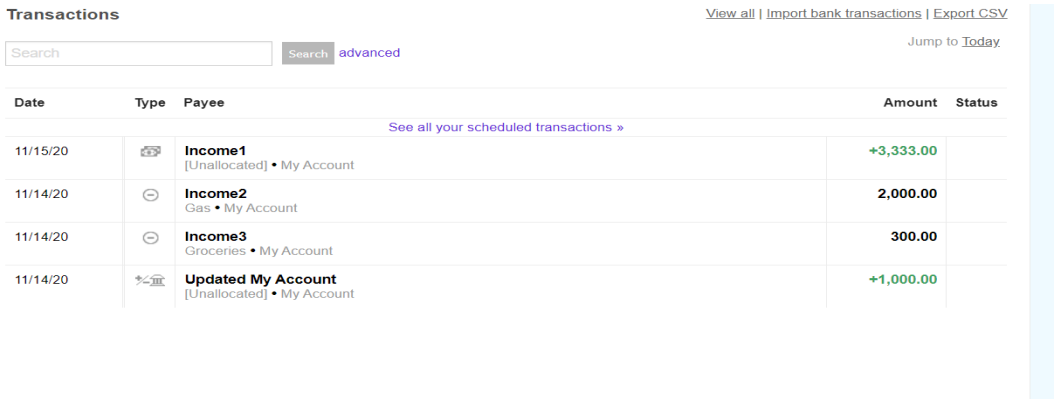


Figure 2.1.2 Screenshot of Goodbudget's income list

Goodbudget had one major advantage and one major disadvantage in the implementation of its budgeting system.

As for the advantage, Goodbudget had a powerful feature – the ability to generate different types of reports or charts based on the user's incomes and expenses. As Goodbudget allowed users to add an unlimited number of financial records, Goodbudget came up with the idea of a feature that was able to put tons of financial records into simple, concise and eye-catching graphs for the user's convenience. There were many options for the report generated, such as spending by month, income vs spending, spending vs budget and etcetera. Examples of graphs that Goodbudget can generate include bar charts, line graphs, and so on. Figure 2.1.3 shows an example of a report between the comparison of income and expenses in Goodbudget. Moreover, Goodbudget was able to display the charts and reports of the previous month or year in comparison to the current month or year. This feature gave users the details they need to make drastic financial decisions in the future. Without this feature, users had to manually go through all the records and calculate the statistics that they need, such as a percentage increase in spending compared to income. If users used this budgeting system without this significant feature, it defied the very purpose of the introduction of a personal budgeting system – to bring convenience to users. Therefore, this advantageous feature of Goodbudget helped users tremendously in summarising their monthly or annual incomes and expenses.

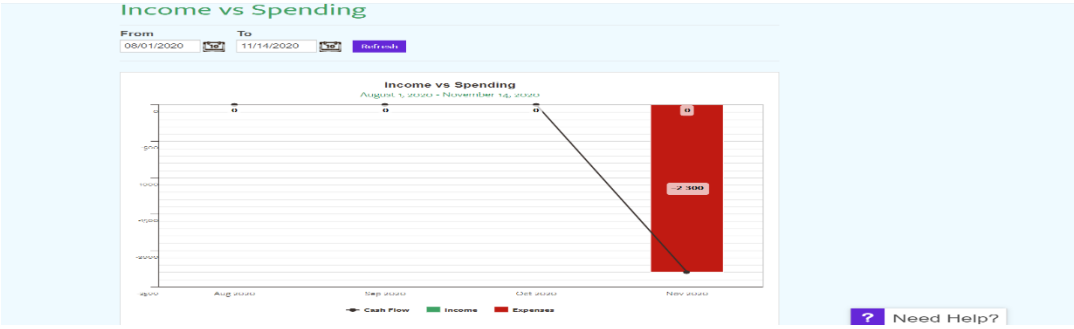


Figure 2.1.3 Income vs Spending Report

On the other hand, the disadvantage of Goodbudget was that the system did not alert the user when the user's expenses were more than his or her income. Instead, Goodbudget only showed red colours in charts and graphs when the user had exceeded his budget or had run out of income and balance. Although Goodbudget did notify the users through the striking red colour on graphs, there was no formal notification sent to the user. There were also no severe alerts given to the user when the user decided to add an excessive amount of expenses records to the system. To remind users of their potential financial issues, Goodbudget should include obvious alerts whenever the user's finances were in the negative zone. As for this project, this disadvantage will be conquered by including automated notifications to the user when a personal finance crisis is detected.

2.2.2 Mobills [4]

The second application to be reviewed is Mobills. The Mobills application functions as a personal finance manager. It helped the user to create a budget plan and keep track of his or her expenses and incomes. One of the powerful functions that Mobills provided is its function to display the overall income, expenses and account all in one place. This allowed the user to manage his or her personal finance conveniently, without having to navigate through complicated menus. Besides that, the essential functions of income and expenses management were also implemented effectively in Mobills . Examples of these essential functions include add, delete and modify income

and expenses. Figure 2.2.1 shows how Mobills display the overall income, expenses and current balancing to the user.

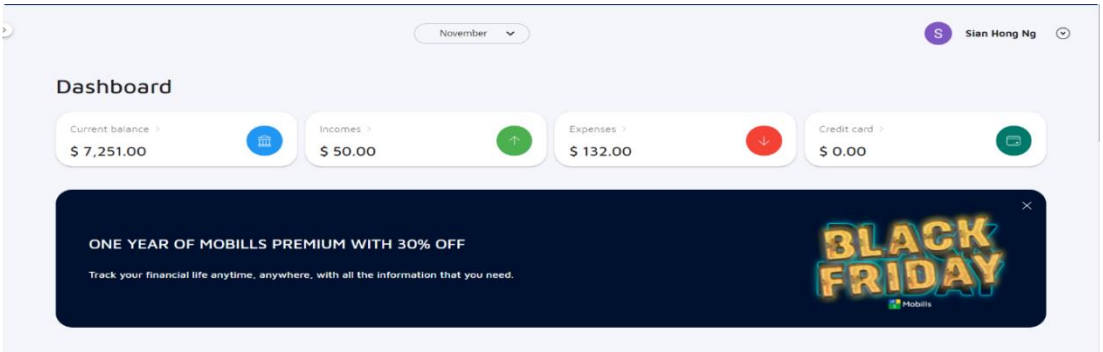


Figure 2.2.1 Overall Income and Expenses

Moreover, the user could create categories to clarify the source of the expenses and income. Additionally, Mobills also provided a list of graphs and charts for the user to choose from. These include incomes by category, expenses by category, monthly income and etcetera. All the graphs and charts would be updated dynamically along with the user's action. In this page, users could also view their overall income and expenses. Figure 2.2.2 shows one of the example graphs in Mobills.

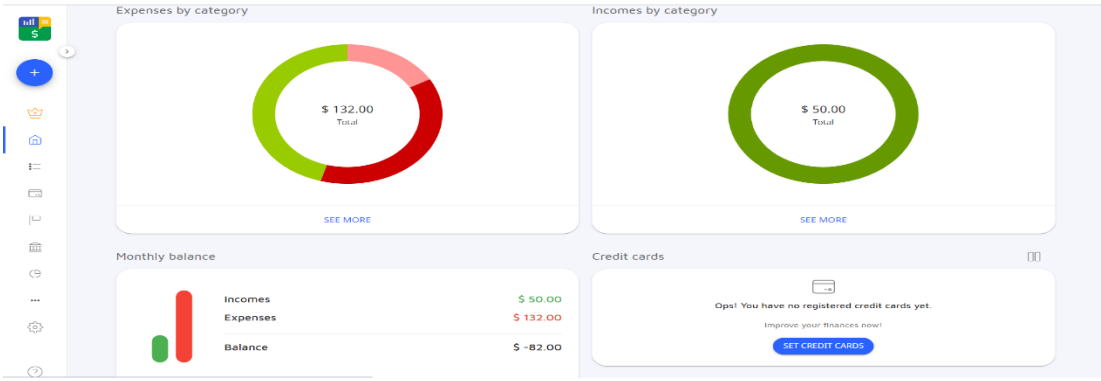


Figure 2.2.2 Income and Expenses Graph

In addition, users were allowed to export the income and expenses records to a file and saved it for back up or reference purposes. Users could also choose to export the records by selecting the types of categories, accounts, start dates and end dates of the report to be generated. Last but not least, users could set a budget for each expense category and get reminders when the expense is getting dangerously close to the

budgeted amount. Figure 2.2.3 shows the export feature that user could be used.

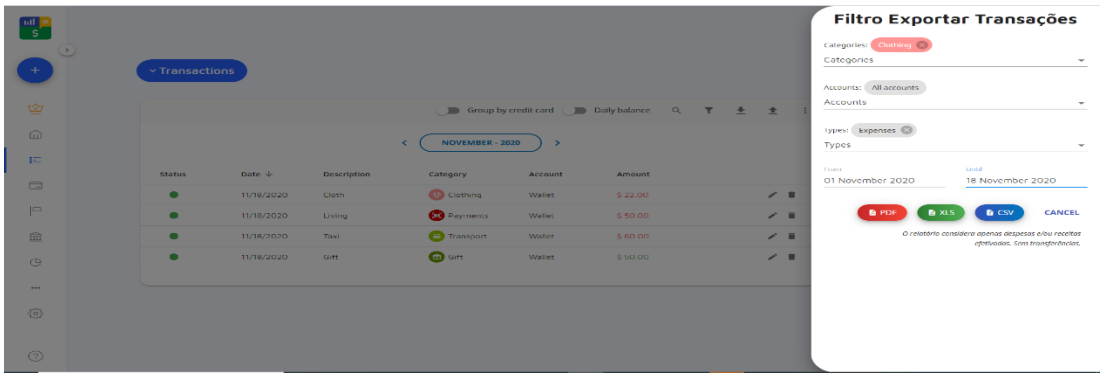


Figure 2.2.3 Export Feature

Mobills had one major advantage and one major disadvantage in the implementation of its personal finance management system.

Mobills had a powerful feature to help the user manage the income and expenses effortlessly, and that was the ability to filter the income or expenses records based on the user's requirement. For example, if the user wanted to find the expenses under the living category, all the users had to do is select the filter feature and key in the requirements of the filter. Then, the application would only display the filtered records, and the other records would be hidden. This feature is handy in this personal finance management application since the user might have added hundreds of records in the income or expenses list. Without this filter feature, the user would have a hard time keeping track of the income and expenses records by the different categories. Moreover, the filter function aided the user to search for a particular category of income or expenses records. Hence, this filter feature brought convenience to users and helped them save time in managing their personal finance. Figure 2.2.4 shows an outline of the filter feature.

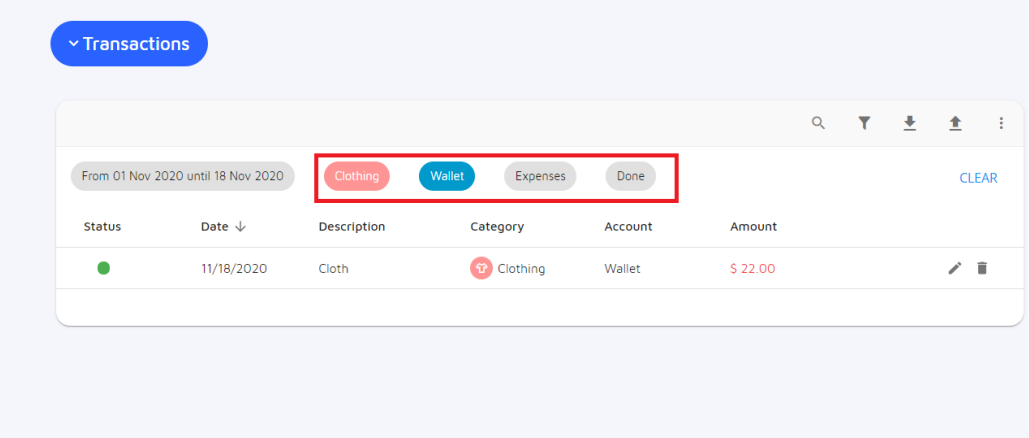


Figure 2.2.4 Filter Feature

On the other hand, the disadvantage of Mobill was that it was unable to let users delete an account. In real life, people would use many different accounts to save money. Sometimes, people would switch in-between accounts for their convenience. In Mobill, users were allowed to create multiple virtual accounts to manage their incomes and expenses separately. However, if the user decided that the account was no longer worth managing and wanted to delete it, he would realise that he was unable to do so because Mobill did not provide this function. Thus, the redundant accounts would stay in the user's homepage and create unnecessary clutter. Besides being an eye-sore, these unused accounts also made it difficult for the user to locate the important accounts in the list of accounts. Therefore, to counter this disadvantage, this project would include a delete account function for users to get rid of their unused accounts. Figure 2.2.5 shows an example of account management.

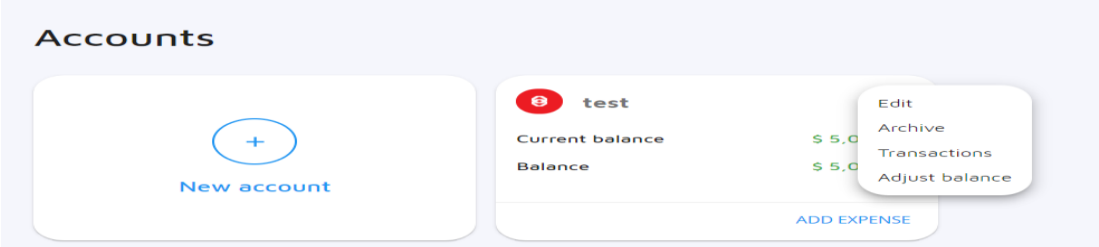


Figure 2.2.5 Account management

2.2.3 EveryDollar [5]

The third application reviewed was EveryDollar. EveryDollar was a web-based application that allowed the user to manage personal budgeting. The overall budgeting system in EveryDollar was similar to Mobills(web.mobillsapp.com, 2013), as reviewed in the earlier section. In EveryDollar, crucial basic functions such as add, delete and modify income and expenses records were implemented. Besides that, one of the features provided by EveryDollar was allowed the user to set due dates on the expenses such as loans or utility bills. These due dates would remind the user to pay their bills on time. Figure 2.3.1 shows the example of how user to set the due date of the expense.

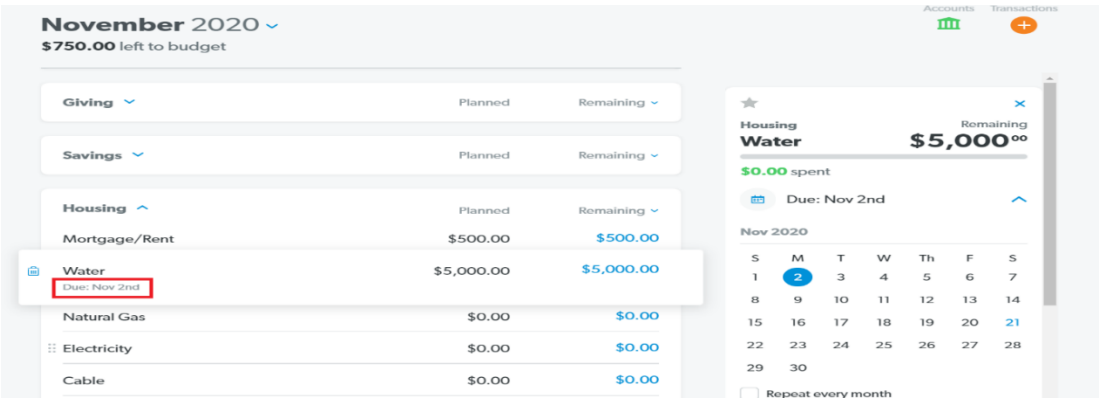


Figure 2.3.1 Due Date of The Expense

Moreover, EveryDollar's management of budget was divided into three categories, which are planned, spent and remaining. For the planned category, the user was allowed to plan future budget amounts by categories. Examples of categories included giving, living, transportation, health and etcetera. Figure 2.3.2 shows an example of planned features. After that, once the user made payment using the add transaction function for the expenses listed under the planned category, the system would automatically record the expenses into the spent category and deduct the paid amount from the planned category according to the transaction record. Figure 2.3.3 shows the budget spent in EveryDollar. Finally, for the remaining category, it showed the remaining budget amount, which was calculated using the planned amount minus the spent amount. Figure 2.3.4 shows an example of the remaining features.

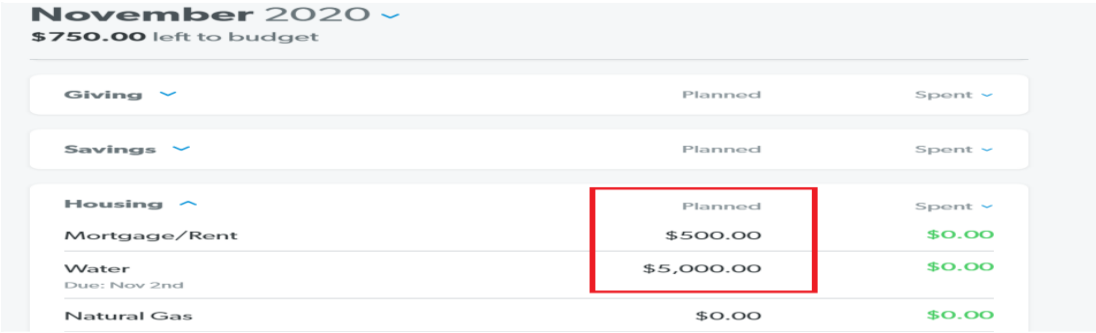


Figure 2.3.2 Budget Planned

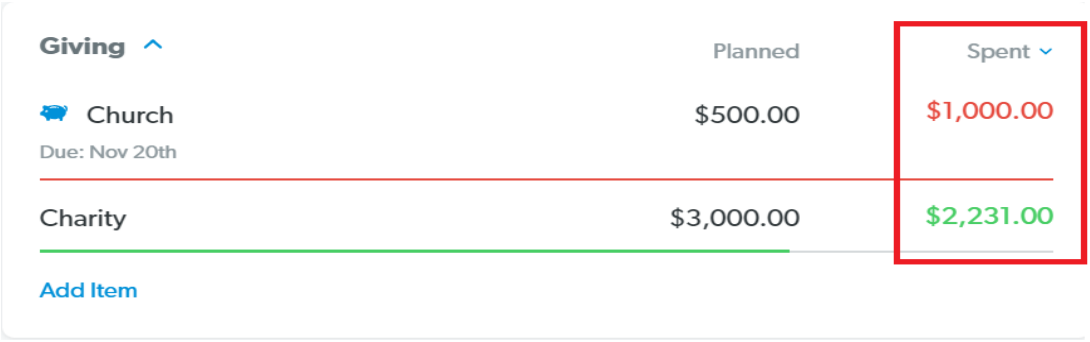


Figure 2.3.3 Budget Spent

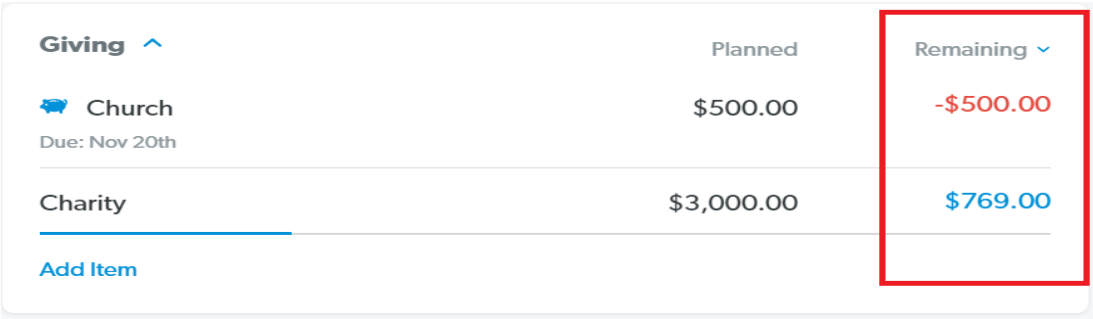


Figure 2.3.4 Budget Remaining

In addition, EveryDollar had a feature which allowed the user to manage and keep track of his or her expenses and income transactions. Upon choosing the option to add a new transaction, the user would be prompted to enter the transaction amount, transaction date, the source of the amount and well as the choice of the budget item. Figure 2.3.5 shows a screenshot of the budget planned. Under the budget items, a list of commonly used tags such as rent, groceries, water, and car loan were shown to the user. The user could choose whichever category that best suited the current transaction, and the amount would be totalled up into the said category. Here, the remaining balance of the categories was shown for the user's reference too. This feature brings

convenience to users as later on, they could easily search for their added transaction in a specific category. In EveryDollar's main menu, the transactions tab split the transaction records into tracked and deleted. The tracked tab showed the latest transactions that the user added previously, and the deleted tab showed the transaction records that were deleted by the user. In the deleted tab, the user was allowed to restore the deleted transaction records, which would bring them back to life inside the tracked tab. Figure 2.3.6 shows an example of how the user restores the deleted records.

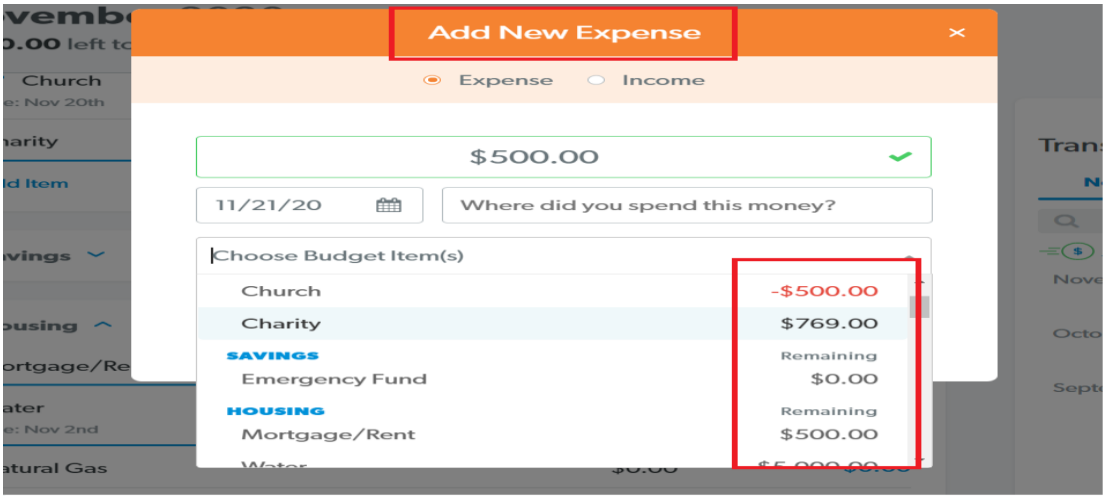


Figure 2.3.5 Budget Planned

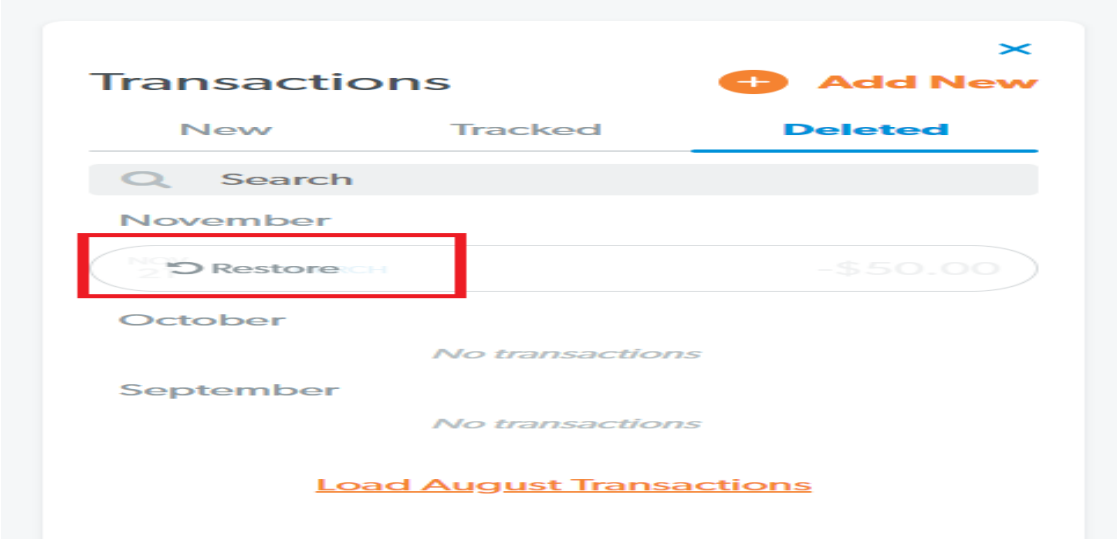


Figure 2.3.6 Restore the Deleted Transaction

The next feature of EveryDollar was its ability to provide charts to display a comparison of a user's income and expenses. The user was able to switch between planned, spent and remaining, where were the three categories of budget management

that EveryDollar provided. In the event that the expenses exceeded the planned budget amount, EveryDollar would display the balance in a striking red colour. In short, these were the main features of the EveryDollar personal budgeting application. Figure 2.3.7 shows a screenshot of the overall chart statistic.

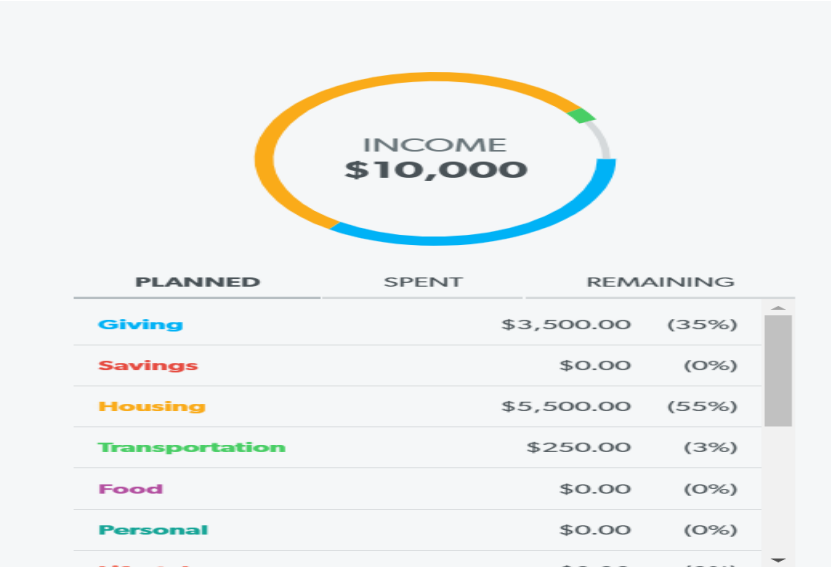


Figure 2.3.7 Chart Statistic

EveryDollar had one major advantage and one major disadvantage in the implementation of its budgeting system.

The advantage of EveryDollar was its ability to filter the income and automatically expenses according to their respective categories. In the other personal budgeting applications reviewed earlier, all the income and expenses records were concatenated into one central list. If a user wanted to view records of a specific category, one had to use the application's built-in filter function. However, EveryDollar already arranged its records according to categories, omitting the need for an extra filter function. In each of the categories shown in EveryDollar's main menu, users were allowed to add in new records straight away. Moreover, the user was able to add in new, custom-named categories if the user felt the need for more detailed categories. This advantage not only brought convenience to users but also saved precious time needed to search for records by categories. Figure 2.3.8 shows the example of categories generated by EveryDollar.

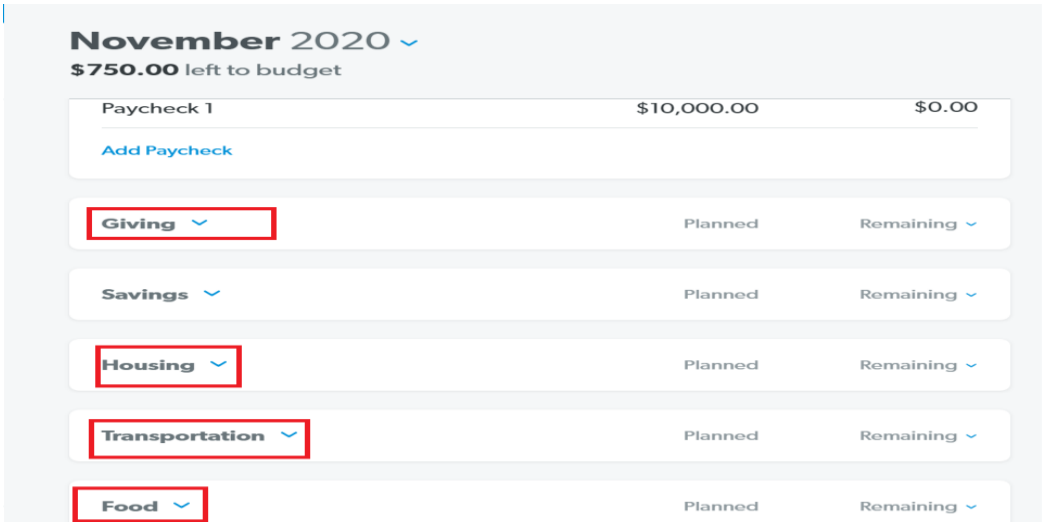


Figure 2.3.8 Expenses Categories

On the other hand, the advantage mentioned previously also yielded a disadvantage. Even though the user was allowed to add new categories into the application, the user was not allowed to delete unnecessary categories. The inability to delete unused categories made EveryDollar vulnerable to clutter, which would make searching for specific categories a tedious process for the user. The only option provided to the user to get rid of all the added categories was the Reset Budget button, which would delete all the added categories as well as wipe out all the user's past transactions in the current month. Hence, without any option to delete accidental categories, the user would have to be cautious not to add unnecessary categories, which could stay unused for a lifetime. As figure 2.3.9 shows, there were a lot of "untitled" categories could not be deleted.

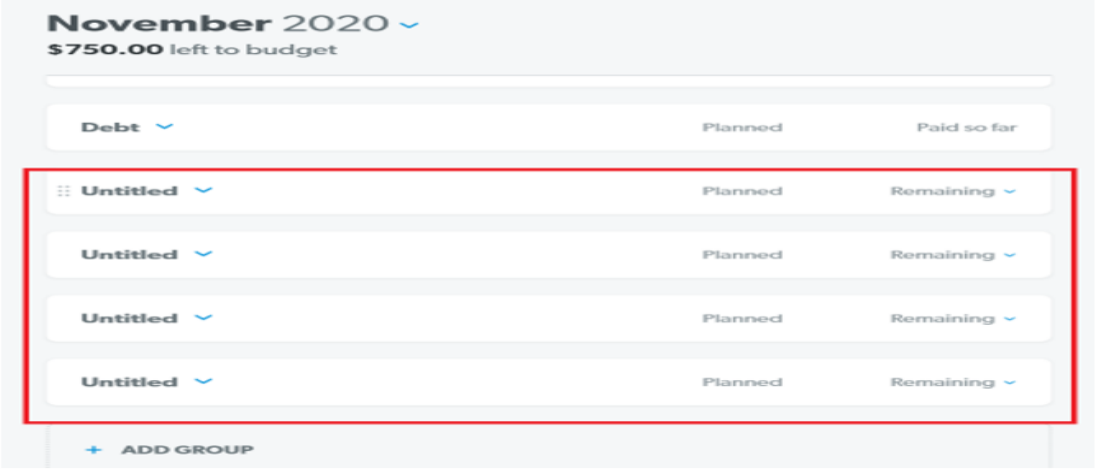


Figure 2.3.9 Untitled Categories

2.2.4 MoneyLine [6]

Last but not least, the last application reviewed was MoneyLine. MoneyLine was an application that allowed the user to manage personal finance. Similar to the applications reviewed in the earlier section, all the basic create, add and delete functions were implemented in MoneyLine. However, one of the different features that MoneyLine provided was that it allowed users to generate the report and that the report was displayed in a statement format instead of attractive, colourful graphs and charts. Examples of reports included Income and Expenses by Category, Expenses by Payee, Account Activity and Budget. Figure 2.4.1 shows the example of reports could be generated in MoneyLine. Figure 2.4.2 shows an example of the income/ expenses of the report by category. With the statement report format, users could clearly see the records listed in the statement.

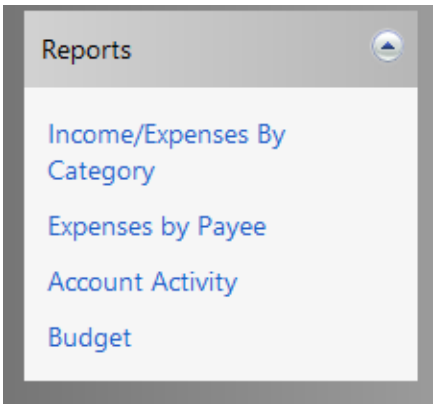


Figure 2.4.1 Example of Reports

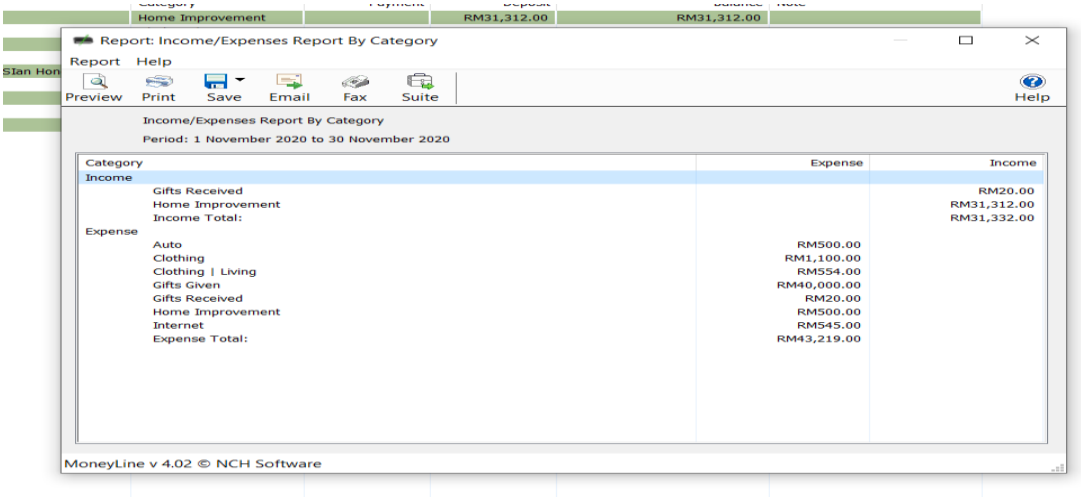


Figure 2.4.2 Income/ Expenses of Report by Category

Moreover, MoneyLine not only allowed the user to manage personal finances but also create budgets. In the budget feature, once the user had paid certain expenses, the system would automatically check the paid expenses against the budget list. If the expenses were budgeted in the budget list before this, MoneyLine would deduct the amount from the budget. This feature was similar to the EveryDollar(EveryDollar.com,2019) budget system. Figure 2.4.3 shows a screenshot of the report according to the budget planned.

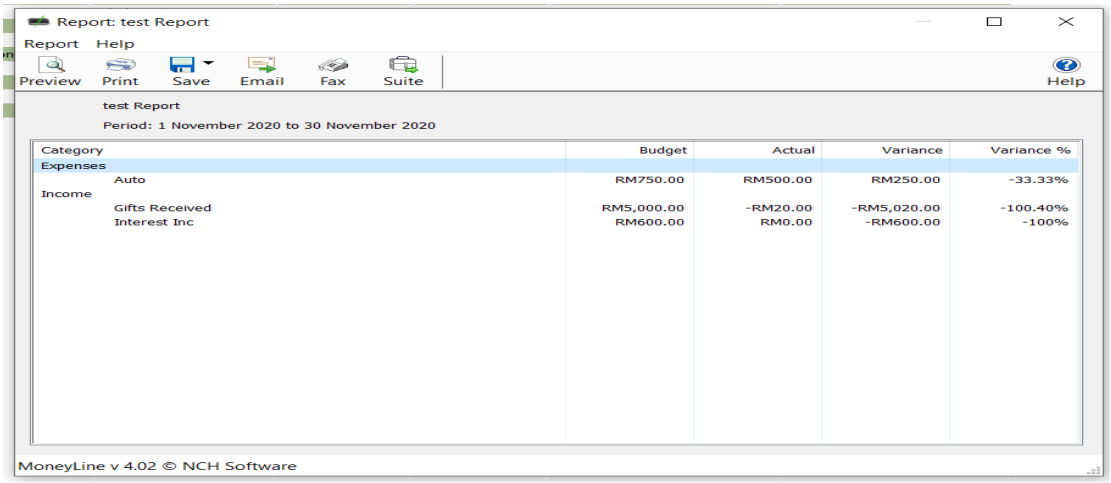


Figure 2.4.3 Budget Report

In addition, this system provided a manage transaction feature which allowed users to add income or expenses records. Inside the manage transaction feature, users were allowed to add a transaction record by selecting the transaction date, amount, payee as well as category. Figure 2.4.4 shows a screenshot on how the user adds a

transaction. After adding a transaction, MoneyLine would automatically display it into the record list. Figure 2.4.5 shows a screenshot of the record list. Moreover, if the user deemed a record unnecessary or wrong, the user could delete or edit the newly added transaction record.

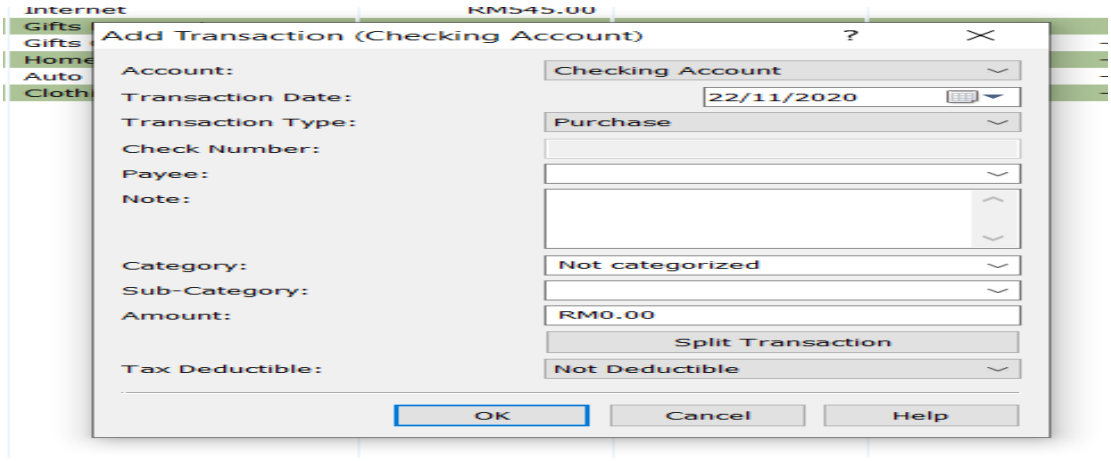


Figure 2.4.4 Add Transaction

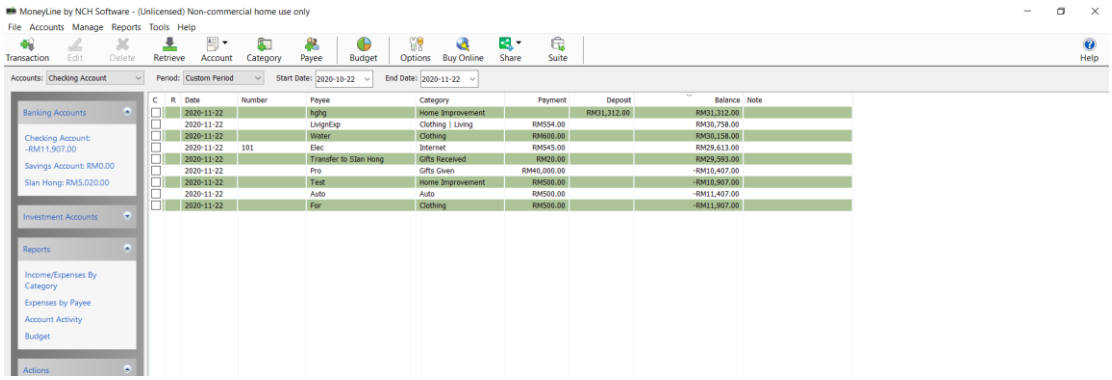


Figure 2.4.5 List of Records

Furthermore, one of the useful features provided by MoneyLine was that users could rearrange the records by clicking the record header. By default, the income and expenses records were arranged according to the newest added record on top. Suppose users wanted to view the records arranged by category. In that case, they could click on the "Category" column header and the records would be arranged in alphabetical order according to the category name. Upon clicking the same column header again, the names would be rearranged in reverse alphabetical order. The same concept applied for the other column headers such as date, payment, deposit and balance. Figure 2.4.6

shows a screenshot of the records arranged by category. Also, figure 2.4.7 shows an example of records arranged based on balance.

C	R	Date	Number	Payee	Category	Payment	Deposit	Balance	Note
☐		2020-11-22	101	Elec	Internet	RM545.00		RM29,613.00	
☐		2020-11-22		hghg	Home Improvement		RM31,312.00	RM31,312.00	
☐		2020-11-22		Test	Home Improvement	RM500.00		-RM10,907.00	
☐		2020-11-22		Transfer to SIan Hong	Gifts Received	RM20.00		RM29,593.00	
☐		2020-11-22		Pro	Gifts Given	RM40,000.00		-RM10,407.00	
☐		2020-11-22		LivignExp	Clothing | Living	RM554.00		RM30,758.00	
☐		2020-11-22		Water	Clothing	RM600.00		RM30,158.00	
☐		2020-11-22		For	Clothing	RM500.00		-RM11,907.00	
☐		2020-11-22		Auto	Auto	RM500.00		-RM11,407.00	

Figure 2.4.6 Arranged by Category

C	R	Date	Number	Payee	Category	Payment	Deposit	Balance	Note
☐		2020-11-22		hghg	Home Improvement		RM31,312.00	RM31,312.00	
☐		2020-11-22		LivignExp	Clothing | Living	RM554.00		RM30,758.00	
☐		2020-11-22		Water	Clothing	RM600.00		RM30,158.00	
☐		2020-11-22	101	Elec	Internet	RM545.00		RM29,613.00	
☐		2020-11-22		Transfer to SIan Hong	Gifts Received	RM20.00		RM29,593.00	
☐		2020-11-22		Pro	Gifts Given	RM40,000.00		-RM10,407.00	
☐		2020-11-22		Test	Home Improvement	RM500.00		-RM10,907.00	
☐		2020-11-22		Auto	Auto	RM500.00		-RM11,407.00	
☐		2020-11-22		For	Clothing	RM500.00		-RM11,907.00	

Figure 2.4.7 Arranged by Balance

Next, MoneyLine provided a functional accounts management feature for users. In personal finance management, account management was considered one of the more crucial components. This feature allowed users to create different types of accounts to manage income and expenses records. Moreover, users were also allowed to transfer the income and expenses between the multiple accounts created. Additionally, the current account's balance would always be displayed at the side menu in order to allow the user to keep track of the balance easily. Figure 2.4.8 shows an example of the account balance in the side menu. MoneyLine also gave users the option to delete unnecessary accounts or modify the current account's balance. Figure 2.4.9 shows a screenshot of the account management interface. In short, these were the main features of the MoneyLine personal finance management application.

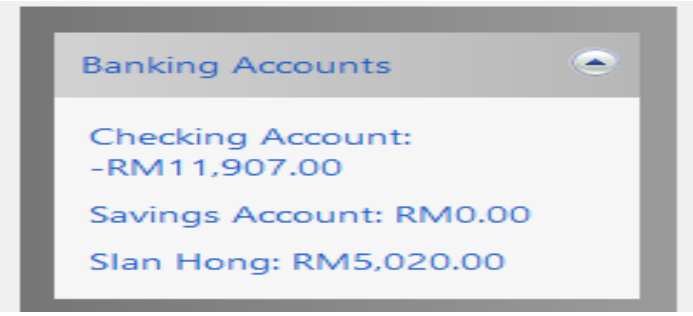


Figure 2.4.8 Account Balance in Side Menu

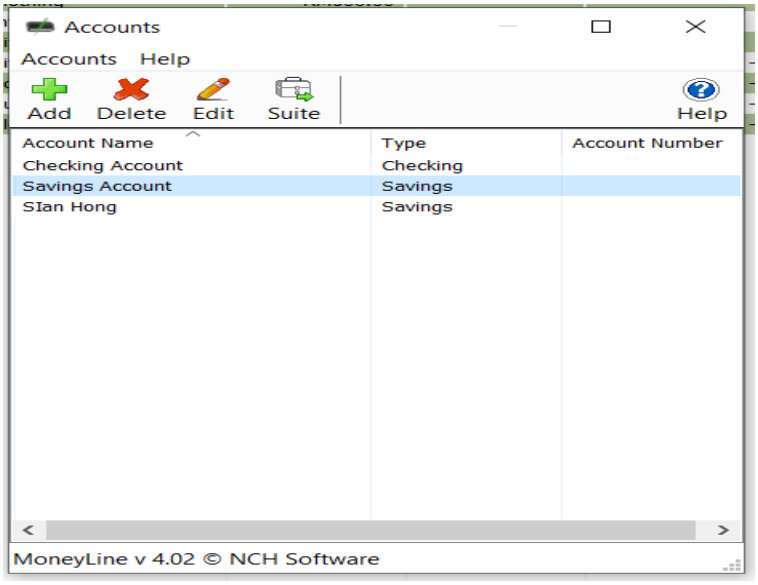


Figure 2.4.9 Account Management

MoneyLine had one significant advantage and one major disadvantage in the implementation of its personal finance management system.

The advantage of MoneyLine was its ability to manage the income and expenses category efficiently. Generally, every personal finance application would create some common finance categories for the convenience of the users. Examples of these categories included utility bills, loans, donations and etcetera. However, as reviewed in the earlier section, most of the applications did not allow users to delete the default categories. With the wide variety of categories provided by the applications previously, users might not use them all, and this results in a cluttered category list. However, in MoneyLine, the user could perform a delete action to remove the unwanted default categories. When the user accessed this category list in the future, the user would not have to go through a lot of unnecessary category names, which saved time for the user. Figure 2.4.10 shows a screenshot of categories management.

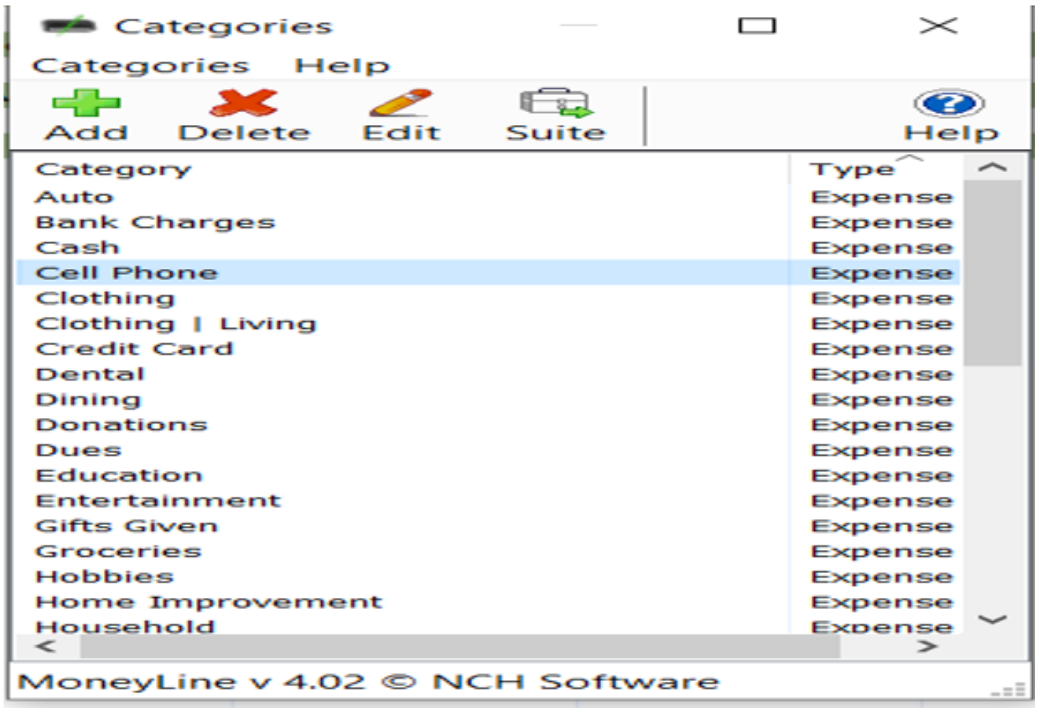


Figure 2.4.10 Categories Management

On the other hand, the disadvantage of MoneyLine was that no alerts or notification messages were displayed when the user's balance has reached a negative amount. The lack of notifications here was a disadvantage because the user might not be aware that the expenses' amount has exceeded the income's amount. When not notified, the user might continue adding expenses records into the account, which would only further enlarge the negative value's gap. This disadvantage was similar to that in Goodbudget (Goodbudget.com, 2020), as reviewed in the earlier section. However, in a comparison between the two, MoneyLine was in a worse state compared to Goodbudget because Goodbudget still attempted to notify the user through red colours in the charts and graphs shown. However, MoneyLine did not provide any form of warning to the user in this case. Figure 2.4.11 shows a screenshot of the negative balance without any notifications. Therefore, an improvement that should be made to MoneyLine was to include these kinds of alert messages to the user when the user's balance hits a negative value. This was to aid the user in redoing the budgeting process to avoid future financial problems.

C	R	Date	Number	Payee	Category	Payment	Deposit	Balance	Note
☐		2020-11-22		hghg	Home Improvement		RM31,312.00	RM31,312.00	
☐		2020-11-22		LivignExp	Clothing | Living	RM554.00		RM30,758.00	
☐		2020-11-22		Water	Clothing	RM600.00		RM30,158.00	
☐		2020-11-22	101	Elec	Internet	RM545.00		RM29,613.00	
☐		2020-11-22		Transfer to SIan Hong	Gifts Received	RM20.00		RM29,593.00	
☐		2020-11-22		Pro	Gifts Given	RM40,000.00		-RM10,407.00	
☐		2020-11-22		Test	Home Improvement	RM500.00		-RM10,907.00	
☐		2020-11-22		Auto	Auto	RM500.00		-RM11,407.00	
☐		2020-11-22		For	Clothing	RM500.00		-RM11,907.00	

Figure 2.4.11 Negative Balance

2.3 Summary of Strength and Weakness

Table 2.5 summarizes the advantages and disadvantages of the four personal finance applications discussed above.

Table 2.5 Types of Personal Finance Applications

Application	Advantages	Disadvantages
Goodbudget	1. Used envelope budgeting method to track expenses 2. Money could be transferred between envelopes to cover additional costs 3. Clearly displayed the overall income and expenses using graphs and charts	1. Did not alert user when the expenses exceeded the income 2. Time-consuming to enter purchase records. 3. Transactions had to be categorised manually.
Mobills	1. Able to export the income and expenses for back-up purposes. 2.Able to filter the income and expenses records based on criteria	1. Unable to delete an account 2. Did not alert user when expenses exceeded income
EveryDollar	1. Provided comparison charts to display a user's income and expenses. 2. Filtered the income and expenses automatically	1. Unable to delete unused categories
MoneyLine	1. Clear and concise report format 2. Easy to use	1. No alerts or notification messages displayed when user's balance has reached a negative amount.

	3. Able to create many accounts 4. Provided a clear overview of one's financial status 5. Able to manage the categories easily	2. All records displayed in a single list

2.4 Comparison Between Existing System

Table 2.6 Comparison of this application and other applications

	Goodbudget	Mobills	EveryDollar	MoneyLine	This Application
Basic Income and Expenses Management	✓	✓	✓	✓	✓
Personal Budgeting	✓	✗	✓	✓	✓
Graph and Chart generation	✓	✓	✓	✓	✓
Report Generation	✓	✓	✓	✓	✓
Basic CRUD functions in Account and User Management	✓	Lack of delete function	✓	✓	✓
Alert user when negative balance	✗	✗	✗	✗	✓

CHAPTER 3

SYSTEM METHODOLOGY / APPROACH

In terms of project development, the processes can be divided into different phases which are functions development, graphical user interface design and development, database, backend and frontend implementation. In this chapter, the methodology for each module is described in detail as well as the system design specification and requirement (UML diagram).

3.1 System Modules Specifications

In this section, the methodologies and workflow of this application used will be described in detail. The detail will be discussed in different modules to be developed and the technology in each module.

3.1.1 Methodologies and General Work Procedures

In this project, the PHP programming language will be implemented to develop the personal finance management application for university students. The reason for the selected programming language is that PHP programming language is one of the easiest ways to develop a desktop-based application. It allows the developer to easily convert an existing PHP website into a desktop application without many modifications.

Besides, XMAPP will be used to act as a server to test the process of development. It is a free and open-source cross-platform web server solution stack package. In terms of the database, there are many database sources in online such as Firebase and MySQL. For firebase, although it is real-time database, but it is a no-relationship database. If firebase was used to manage relationships database, it will be quite hard to implement it. In addition, most desktop application will not use the online database due to the fact that desktop application's data is quite confidential for each individual user. As such, MySQL will be used to manage the database of this project. The reason for choosing MySQL is because it can perform an excellent interaction between the data and users.

Moreover, it also provides an effective way to retrieve data from database. For example, using SQL queries to retrieve some values. Therefore, MySQL is the selected backend database for this project. This project includes six modules that work cohesively to achieve the project's objectives:

- Income and expenses management
- Report management
- Graph and chart generation
- User management
- Personal budgeting
- Account management

Module 1- Income and expenses management:

The first module of this personal finance management application is income and expenses management. In this module, users are provided with basic CRUD functions that allow them to keep track of their income and expenses. These functions include add, delete and modify, which can all be performed on the income and expenses records. Additionally, when the user's expenses are more than the income, the application will immediately notify the user.

Module 2- Report management :

The second module to be implemented is report management. This module helps the user to generate a report regarding the user's income and expenses records. The user is allowed to select the type of report to be generated – overall account activity, income report or expenses report. Not only that, the user is able to select a particular month as the time range of the generated report.

Module 3- Graph and chart generation:

The third module is the graph and chart generation. This module dynamically generates the graph or chart based on the user's current income and expenses records. Upon changes made by the users in their income and expenses records, the graph and chart module automatically update the chart to display the latest data.

Module 4- User management:

The fourth module to be used in this project is user management. This module is used to protect the user's personal finance details. Before the user is permitted to use the application, the user is prompted to register a new account or log in to an existing one. The use of different accounts for different users is to keep their personal information private and secure. Moreover, users are allowed to modify their user account settings through functions such as a password reset.

Module 5- Personal budgeting:

The fifth module is personal budgeting. This module allows the user to budget their income and expenses. Users can create many types of categorised income and expenses to perform the budgeting. This module allows pre-budgeting to be done for the upcoming months.

Module 6- Account management:

The last module of this project is account management. This module allows users to create multiple accounts to manage their income and expenses. This module is not to be confused with the user management module. The relation between these two modules is that one user account may have multiple accounts. Additionally, the individual income and expenses records belong to a specific account, and the amounts will be recorded into the said account accordingly. The purpose of this module is to accommodate users that might choose to manage their finance according to the different financial accounts that they own in real life.

3.1.2 Hardware Requirement

In this project, the hardware used to implement and run the application is a laptop or desktop. Besides, when the final product is complete, the user only needs to download the 'exe' file to exe the installation by using their own laptop or desktop. Table 4.1 shows the recommended specifications for the hardware used:

Table 3.1 Hardware required:

Device	Specification	Recommended
	Operating System	Windows 10 (32-bit)

Laptop	Memory	8GB
	Graphic Card	NVIDIA GeForce 940MX
	Storage	2-5 GB available space

3.1.3 Software Requirement

The software that needed for this project to develop is listed below.

1. **Visual Studio Code**

   Visual Studio code is an excellent coding editor that supports various of languages including PHP. The reason of choosing visual studio code is because it has PHP libraries and those syntax that support PHP. In addition, visual studio code has a clear and simple user interface that allow programmers to easily write and debug code.

2. **XAMPP**

   As mentioned in the methodology section, XMAPP will be used as a server to test the process of development. XAMPP also hosts the MySQL server to provide database access to this project. It helps to store the user, account,

   transaction, category as well as budget planning data.

3. **PHP-Desktop File**

   Since php is mostly used to develop a web application, therefore, the php-desktop file is used to convert the web version to a desktop application. When all the php files are put in the right place inside the PHP-Desktop File, the user can just simply click on the .exe to use the application.

4. **Firebase**

   Firebase is a developer toolset created by Google that provides a wide variety of services. The Firebase features utilized in this project is authentication . This feature is used to allow users to log in to the personal finance desktop

application, registration user account as well as resetting their password by using email verification.

3.2 Modules Concept Design Diagram

In this section, 6 modules are elaborated more details in terms of the activity diagram, use case diagram, ERD diagram, sequence diagram as well as database design diagram.

3.2.1 Use Case Diagram

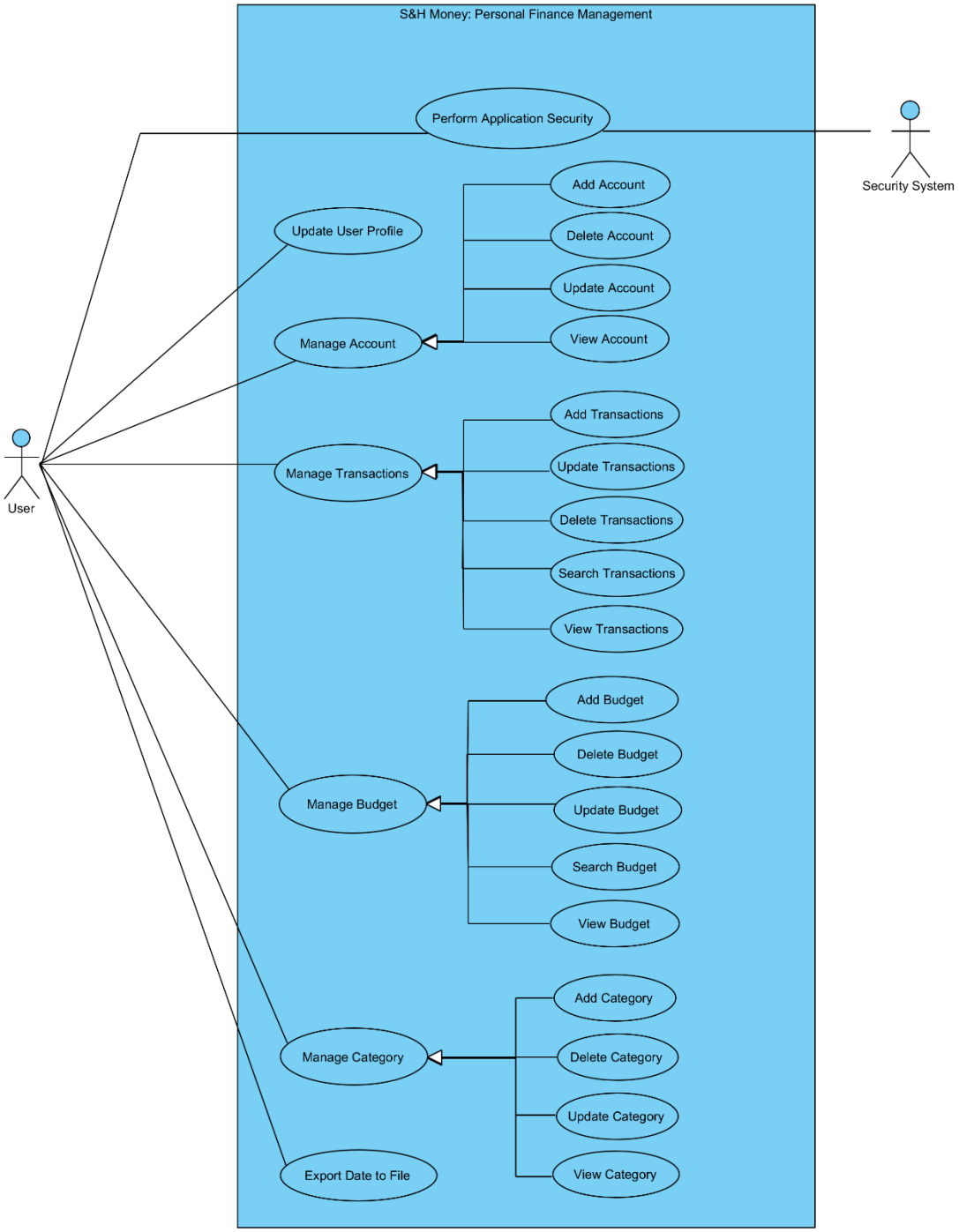


Figure 3.2.1.1 Use Case Diagram

Application Security- Before university students manage their personal finance, they need to login so that they can use the functions and see the existing transactions and records. The reason is because all the personal finance data can be considered as

confidential data. If university students do not have any user account to login, they can click on the “register” hyperlink to proceed with the registration. In addition, when the university student feels that his or her user account’s password is insecure, the university is allowed to change new password by using old password.

Update User Profile- Once the university student creates a user, the university student is allowed to change the user’s information after login, such as username, email and phone number. The system will validate the updated information whether it is valid. If the updated information is not valid, the system will not allow the university student to update the profile.

Manage Account- Normally, one user can have multiple accounts to manage their personal finance. For example, one user may have Public Bank account, Maybank account and etcetera. Without creating any accounts, the user is not able to do the personal finance management because the system will automatically detect whether the user has any existing accounts. If not, the system will prompt a message that will guide the user to create an account and do the personal finance management. Once the user has created an account, the user also can go to the account list to view the existing account to do some modification, such as update and delete.

Manage Transaction- After creating an account, the user is able to use the account to add the transaction. The transaction can be divided into two types, which are expenses and income. The user needs to go to the right navigation in order to add a correct transaction type. After that, the user is able to perform add, delete, update and search transaction based on the account that user has selected. For every new transaction, the system will validate it whether it exists in the database. If there is any duplication transaction or error format of transaction information, the system will prompt an error message to notify the user about the error. Last but not least, the actions of update and delete transaction record can be performed in the transaction list.

Manage Budget- The user is able to perform budget planning in this application. Budget planning can be divided into two parts which are account budget planning and category budget planning. Each of the parts has its own basic CRUD functions. However, before adding a category budget planning, the user must add an account planning first before adding the category budget planning because the category budget planning will be created based on the account budget planning. In other words, the

account budget planning is the “parent”, and the category budget is the “child”. The system will automatically detect whether there are any account budget planning records before adding a category budget planning. If there are no account budget planning records, the system will notify the user to create one before proceeding to add a category budget planning record.

Manage Category- Each of the transaction must be under a category so that they can be managed effectively. Before adding a new transaction, the user needs to add a category. No matter whether it is income or expense, the category needs to be created beforehand. Users can perform basic actions, such as add, delete, update to manage category. Since the transaction is divided into two types to manage, the user just needs to go to either the expense or income part to manage the category. After that, the system will automatically set the category under which types of transaction. In addition, different users will have different categories, and all the added categories will be based on the user’s account to be displayed to the relevant users.

Export Data to File- Users are allowed to export their transactions to an external file such as pdf file. This function gives user an overall view of their personal finance. The reason is because users can export a specific month of transaction records to an external file and keep it as a copy rather than search some records in the application, which is time-consuming.

3.2.2 Activity Diagram

In this section, the primary use cases are illustrated using activity diagrams. The use cases included in this section are manage account, manage transaction, manage budget, perform application security. The details are shown below in Figure 3.2.2.1.

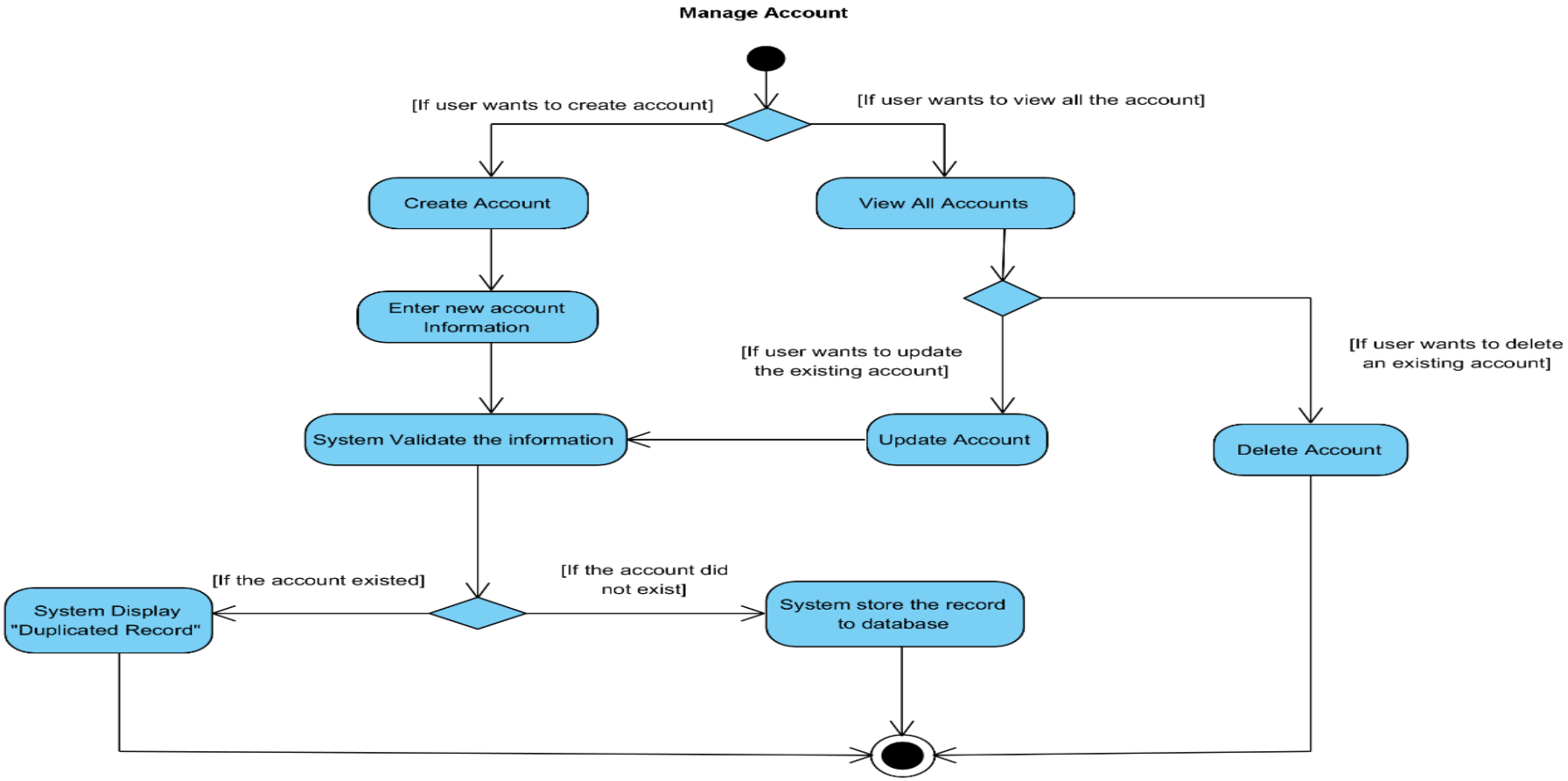


Figure 3.2.2.1 Manage Account Activity Diagram

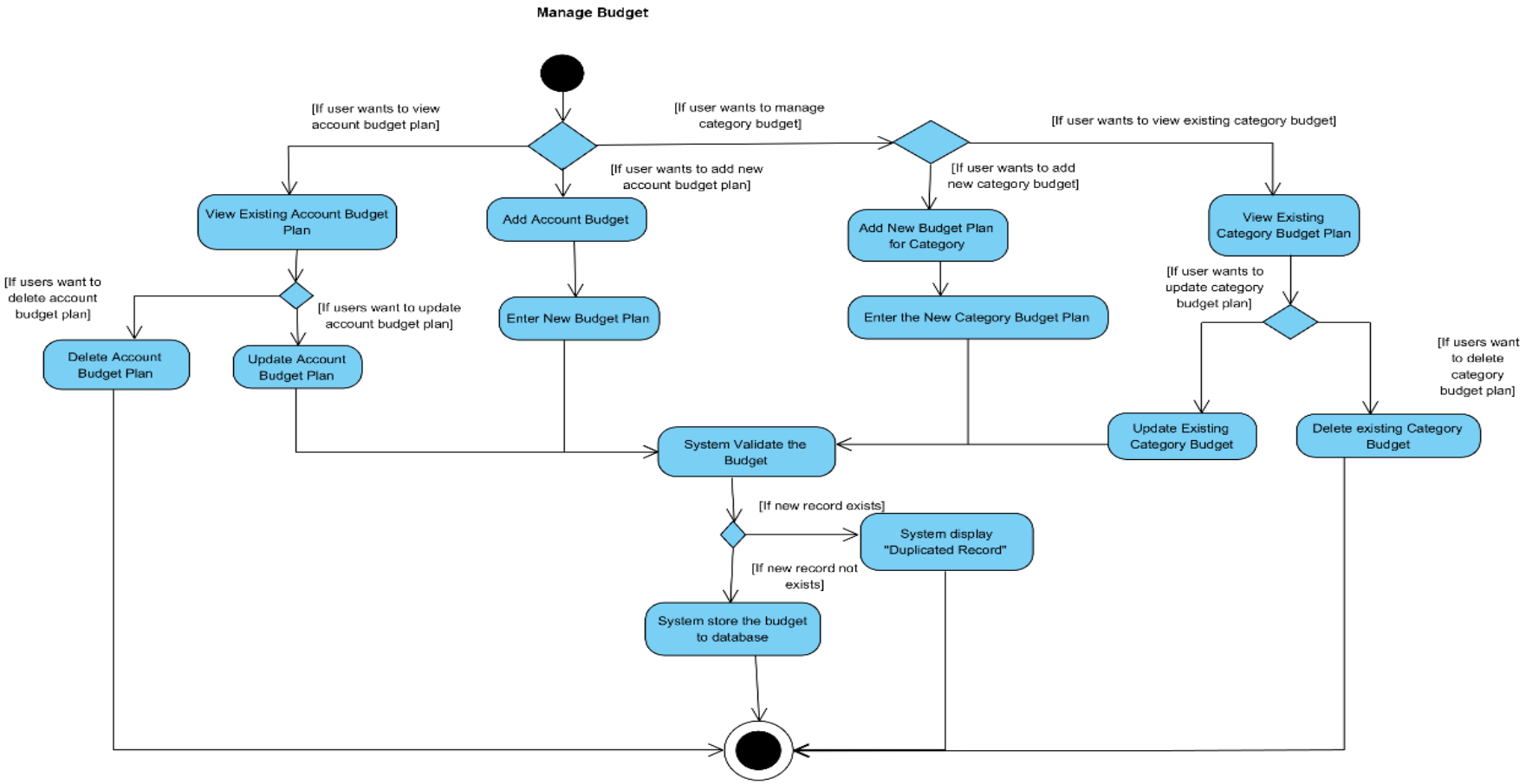


Figure 3.2.2.2 Manage Budget Activity Diagram

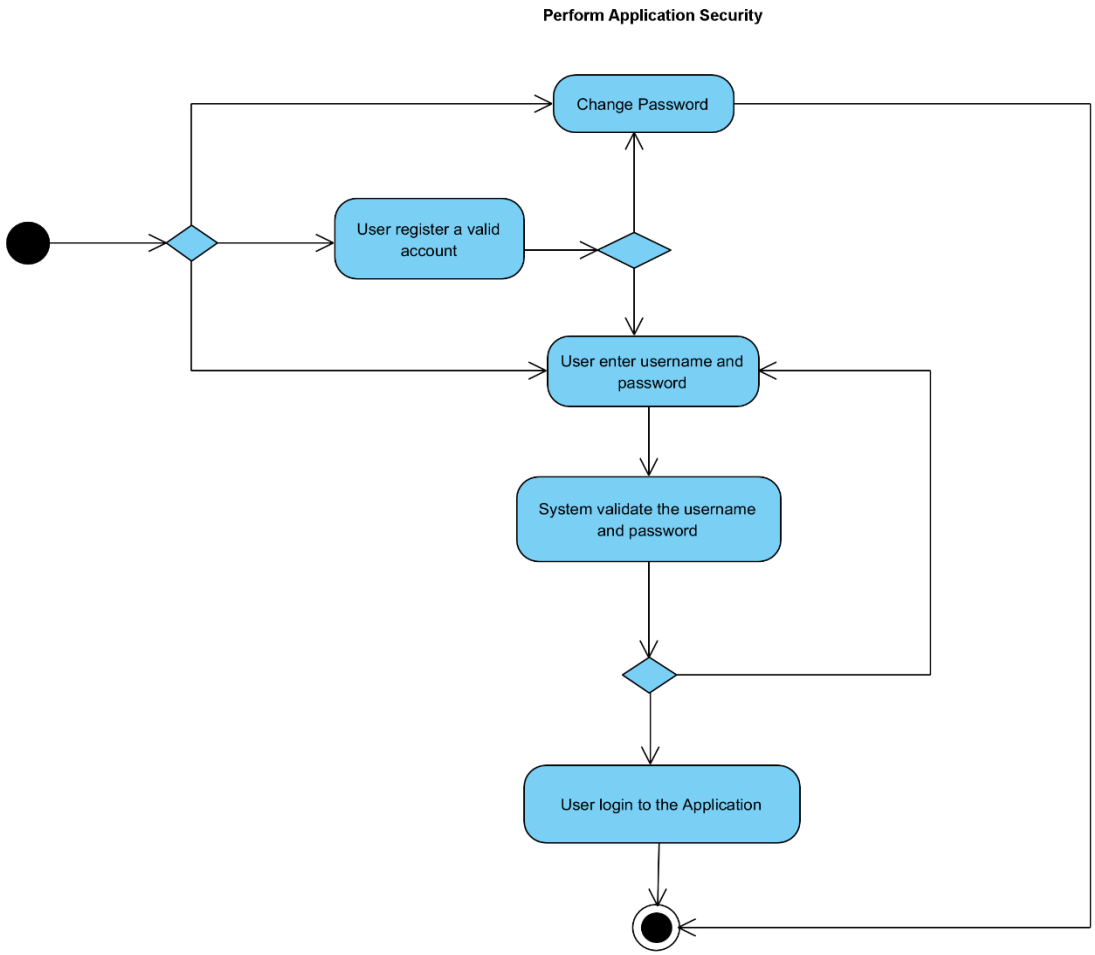


Figure 3.2.2.3 Perform Application Security Activity Diagram

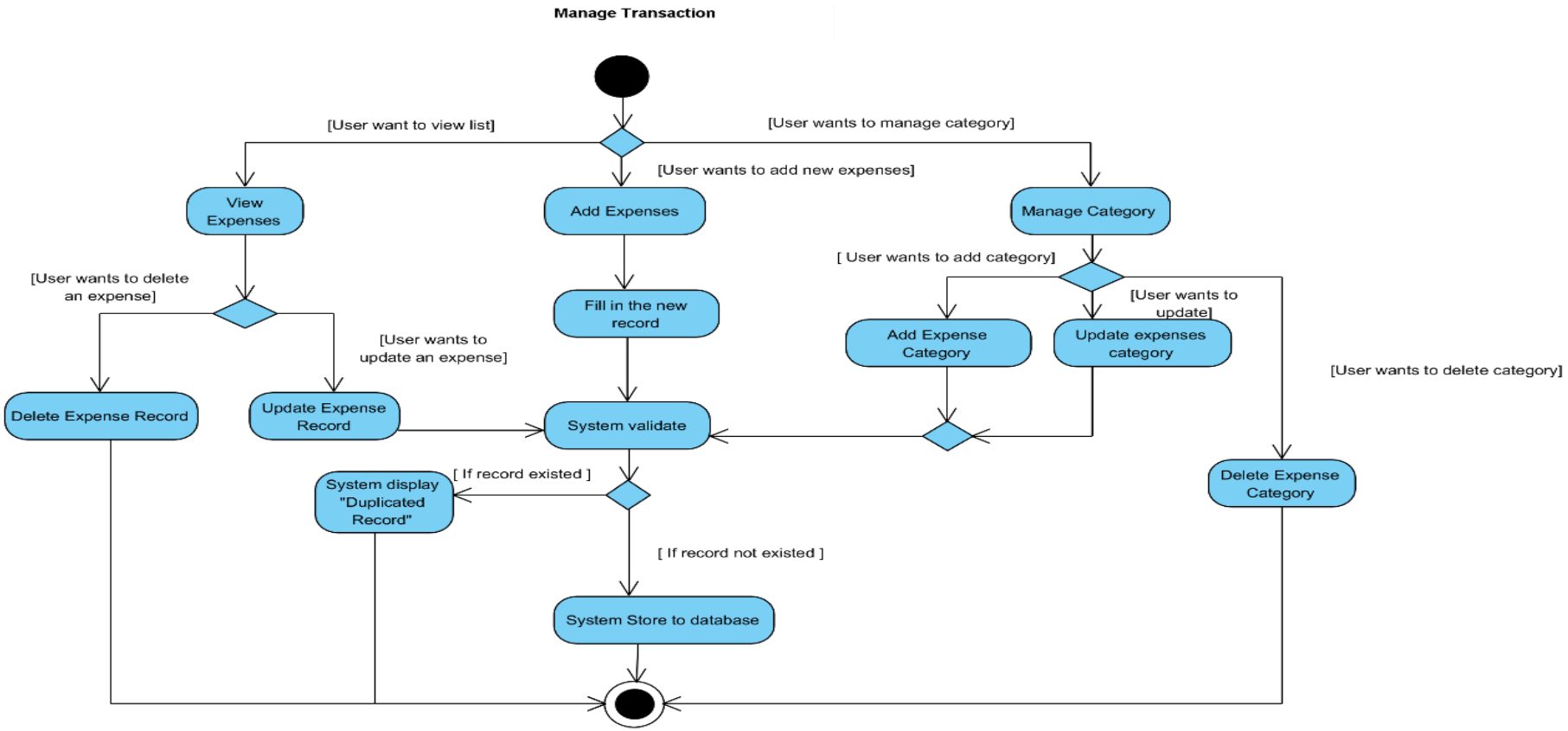


Figure 3.2.2.4 Manage Transaction Activity Diagram

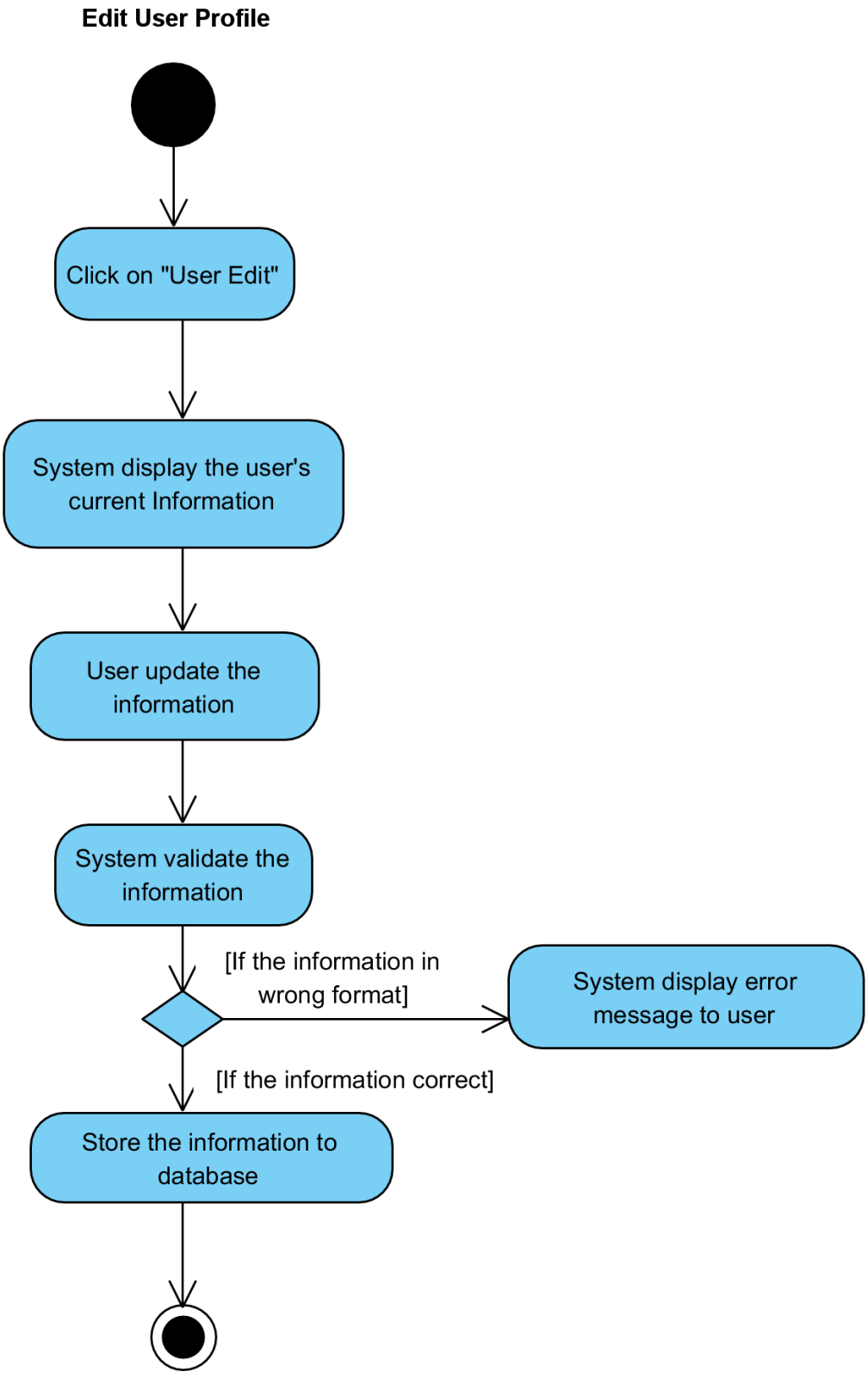


Figure 3.2.2.5 Edit User Profile Activity Diagram

3.2.3 Sequence Diagram

In this section, the generate chart and report module are illustrated using sequence diagrams to show the flow of the process. Basically, the common flow of process is user clicks on the generate button, after that, the application will receive the user action and perform the user request to the user. The details are shown below in Figure 3.2.3.1.

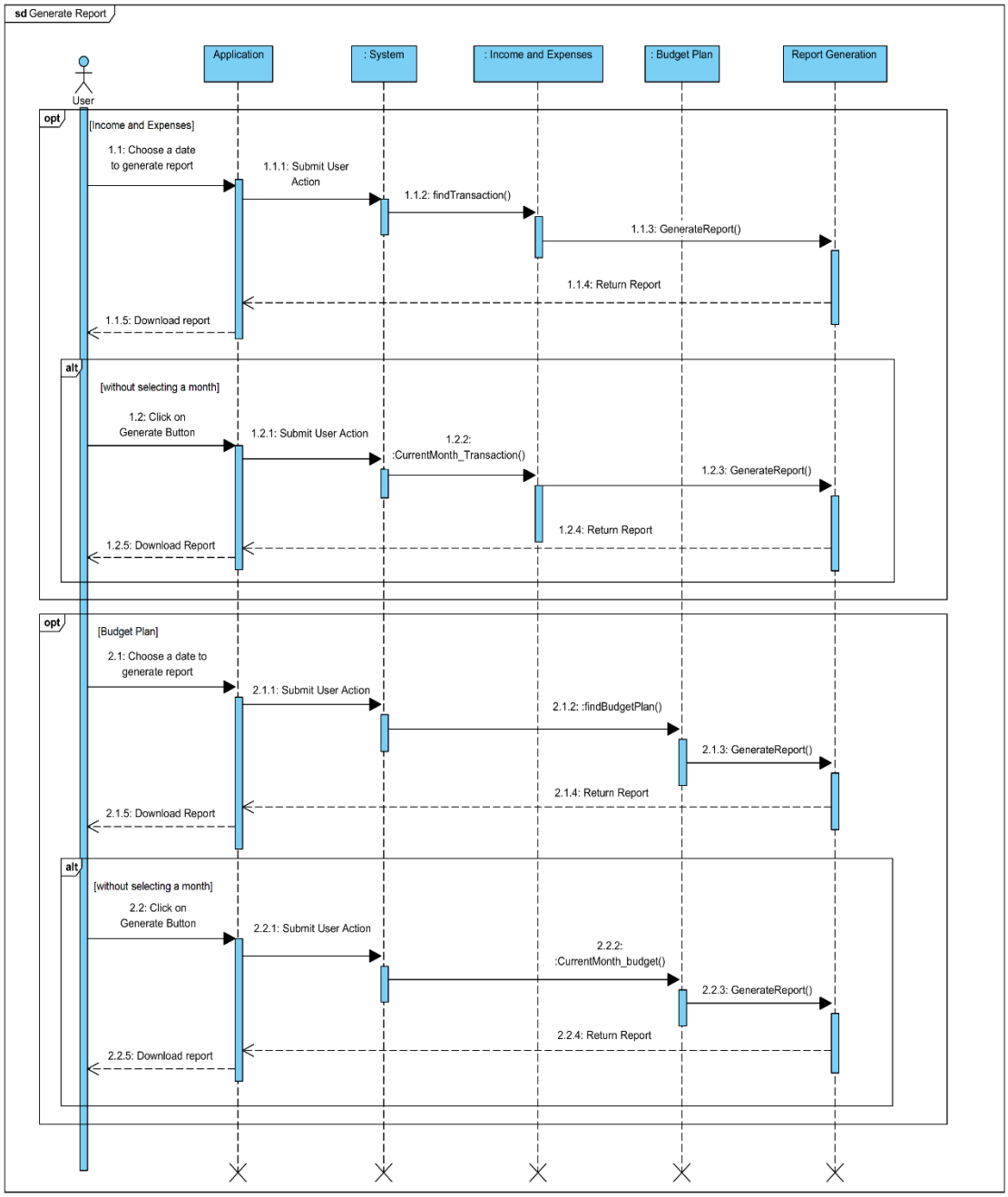


Figure 3.2.3.1 Generate Report Sequence Diagram

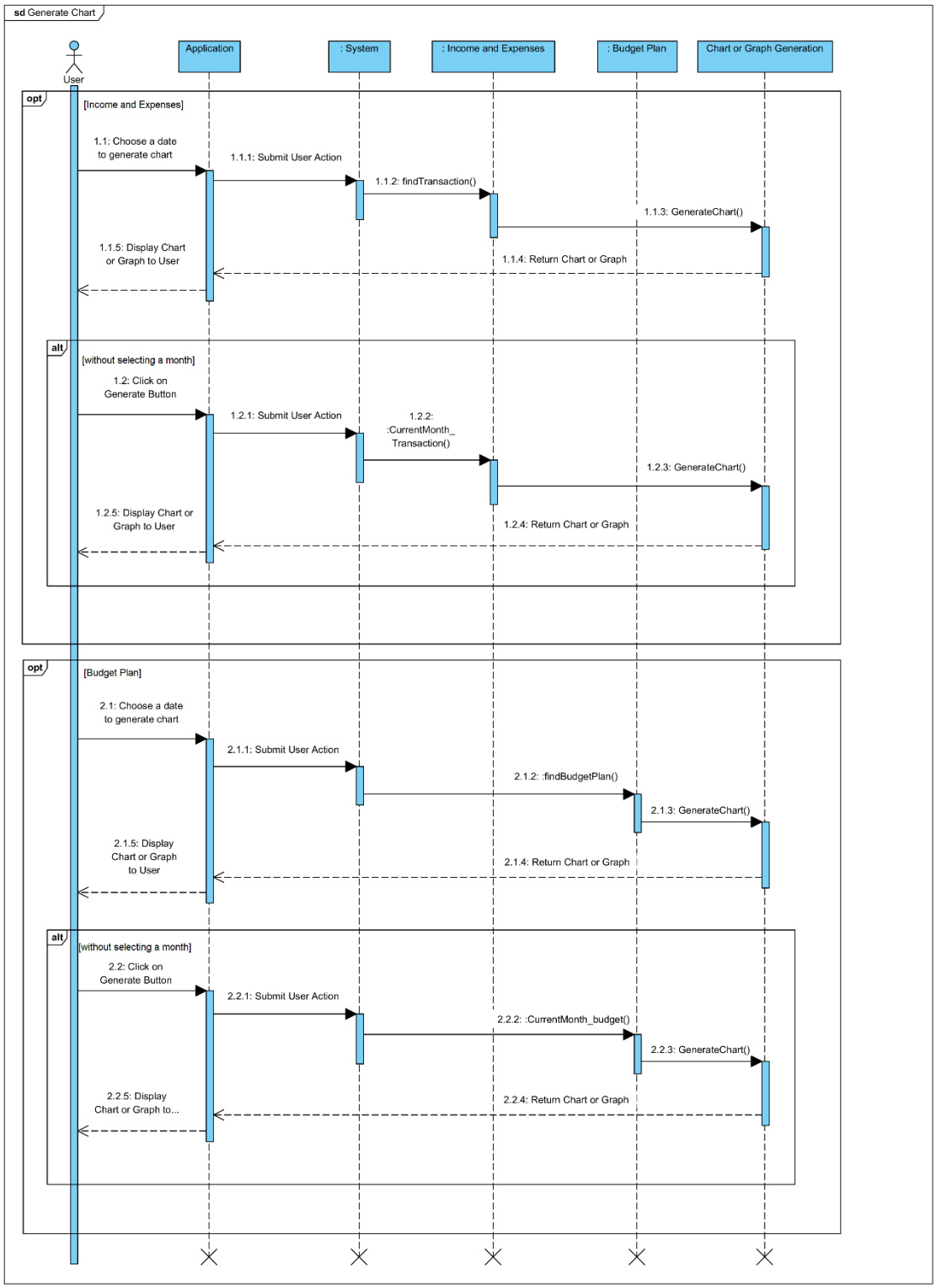


Figure 3.2.3.2 Generate Chart Sequence Diagram

3.2.4 Database Design Diagram

In this section, all the database design will be shown. Basically, this personal finance application included 7 tables to store the information. The tables included User, Account, Balance, Category, Transaction, Budget, Cat_Budget. The overall relationship for this database:

- One user(User) can have multiple categories(Category) and accounts(Account)
- An account(Account) can have multiple transactions(Transaction) and budgets(Budget) as well as single balance(Balance) amount
- A budget(Budget) can have multiple category's budget planning(cat_budget).

The 3NF normalisation of this database as shown below, the word with line and bold is primary key and the word with * is foreign key:

User (**id**, name, phone, email, password)

Account (**a_id**, acc_id, acc_name, user_id*)

Balance (**b_id**, b_amount, month, account_id*)

Category(**c_id**, name,type, user_id*)

Transaction(**t_id**, t_amount, t_desc, t_date, t_type, account_id, c_id*)

Budget(**bud_id**, amount, bud_month, account_id*)

Cat_Budget(**cb_id**, cb_amount, budget_id*, c_id*)

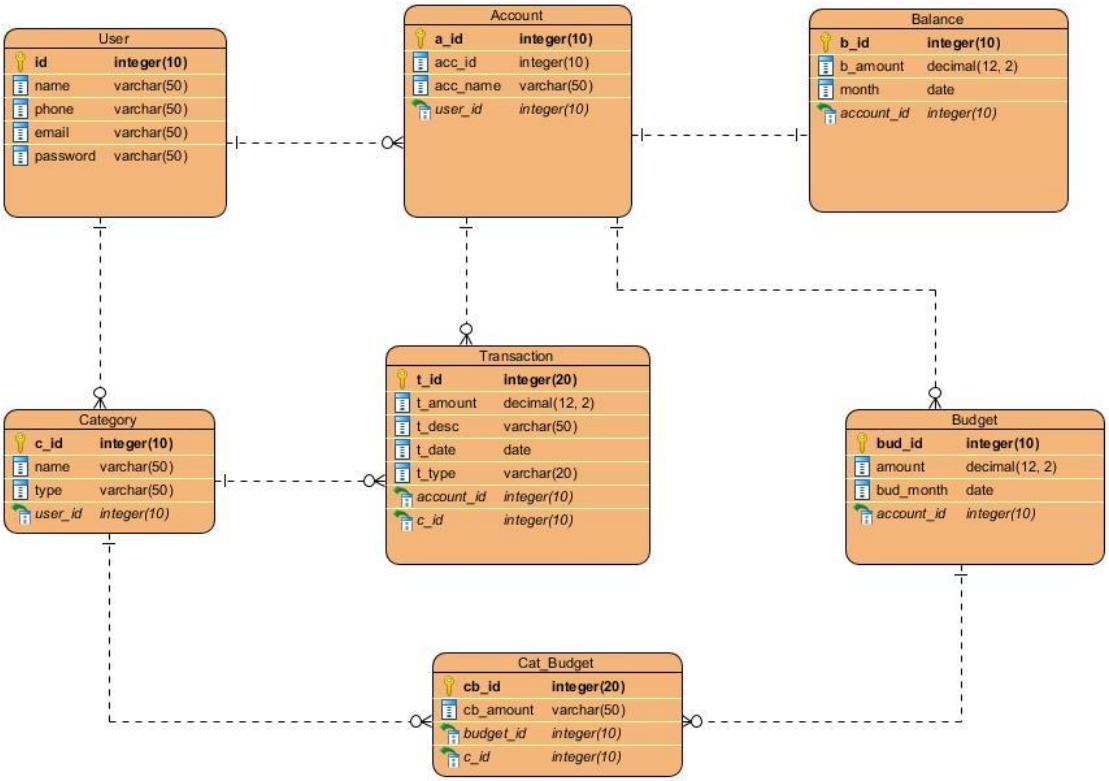


Figure 3.2.4.1 ERD Diagram

In the figure above, ERD diagram has been illustrated by including the 7 tables mentioned earlier. This diagram shows the communication and relationship between these 7 tables. The data dictionary for each table is shown below.

*PK= Primary Key, *FK= Foreign Key

Table 3.2 Data Dictionary of User Table

User						
Column Name	Data Type	Length	Nulls	PK/FK	Reference to	Example
Id	Int	11	No	PK	-	1
Name	Varchar	50	No	-	-	SianHong
Phone	Varchar	50	No	-	-	019-4914144
Email	Varchar	50	No	-	-	Ngsianhong1@gmail.com
password	Varchar	50	No	-	-	1313

Table 3.3 Data Dictionary of Account Table

Account						
Column Name	Data Type	Length	Nulls	PK/FK	Reference to	Example
a_id	Int	10	No	PK	-	1
acc_id	Int	10	No	-	-	6455646
acc_name	Varchar	50	No	-	-	Public Bank
user_id	Int	10	No	FK	User	2

Table 3.4 Data Dictionary of Balance Table

Balance						
Column Name	Data Type	Length	Nulls	PK/FK	Reference to	Example
B_id	Int	10	No	PK	-	1
b_amount	Decimal	12,2	No	-	-	500.00
Month	Date	-	No	-	-	2021-05-15
Acc_id	Int	10	No	FK	Account	2

Table 3.5 Data Dictionary of Category Table

Category						
Column Name	Data Type	Length	Nulls	PK/FK	Reference to	Example
C_id	Int	10	No	PK	-	1
Name	Varchar	50	No	-	-	Daily
Type	Varchar	20	No	-	-	expenses
User_id	Int	10	No	FK	User	2

Table 3.6 Data Dictionary of Budget Table

Budget						
Column Name	Data Type	Length	Nulls	PK/FK	Reference to	Example
bud_id	Int	10	No	PK	-	1
Amount	Decimal	12,2	No	-	-	500.00
Month	Date	-	No	-	-	2021-06-20
Account_id	Int	10	No	FK	Account	2

Table 3.7 Data Dictionary of Transaction Table

Transaction						
Column Name	Data Type	Length	Nulls	PK/FK	Reference to	Example
t_id	Int	20	No	PK	-	1
T_amount	Decimal	12,2	No	-	-	50.00
T_desc	Varchar	50	No	-	-	Bus ticket Fee
T_date	Date	-	No	-	-	2021-05-18
T_type	Varchar	20	No	-	-	expenses
Account_id	Int	10	No	FK	Account	2
C_id	Int	10	No	FK	Category	2

Table 3.8 Data Dictionary of Cat_budget Table

Cat_budget						
Column Name	Data Type	Length	Nulls	PK/FK	Reference to	Example
Cb_id	Int	20	No	PK	-	1
Cb_amount	Decimal	12,2	No	-	-	600.00
Budget_id	Int	10	No	FK	Budget	1
C_id	Int	10	No	FK	Category	2

CHAPTER 4

SYSTEM DESIGN

In this chapter, the system design of this application is explained in detail by including the system block diagram, system components interaction operations and the function and user interface of the six modules. The six modules including User Management, Account Management, Income and Expenses Management, Personal Budgeting, Generate Report and Chart. The completion of these modules can be further subdivided into UI development and function development.

4.1 System Block Diagram

In this section, 6 modules are elaborated more details in terms of the general workflow with the block diagram.

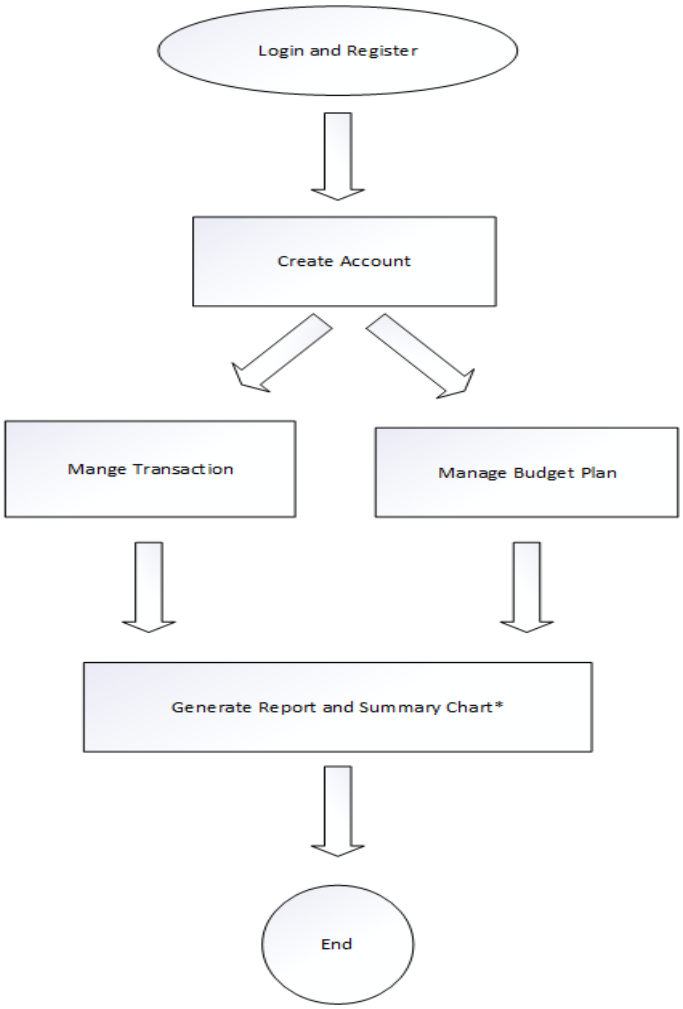


Figure 4.1.1 Block Diagram

Figure 41.1 shows the six modules in the general flow of this developed application for university students. First of all, the university student needs to create a valid user so that he or she can access to the application. This module is used to protect the university student personal finance transaction records. Next, once the user has completed the user registration, the user is able to create multiple accounts to manage the personal finance. After that, each account is able to manage different transactions and budget plans. Furthermore, after adding some transactions, user is able to generate reports based on the transactions or budget plan. Additionally, the system will be based on the account's transaction and budget planning to automatically show summary charts to display to the user their overall status of personal finance. Both generate report and generate summary charts are in different module because both of these are in charge of different thing.

In the following section, the system components interaction operation, actual function and user interface in this application will be discussed in detail.

4.2 System Components Interaction Operations

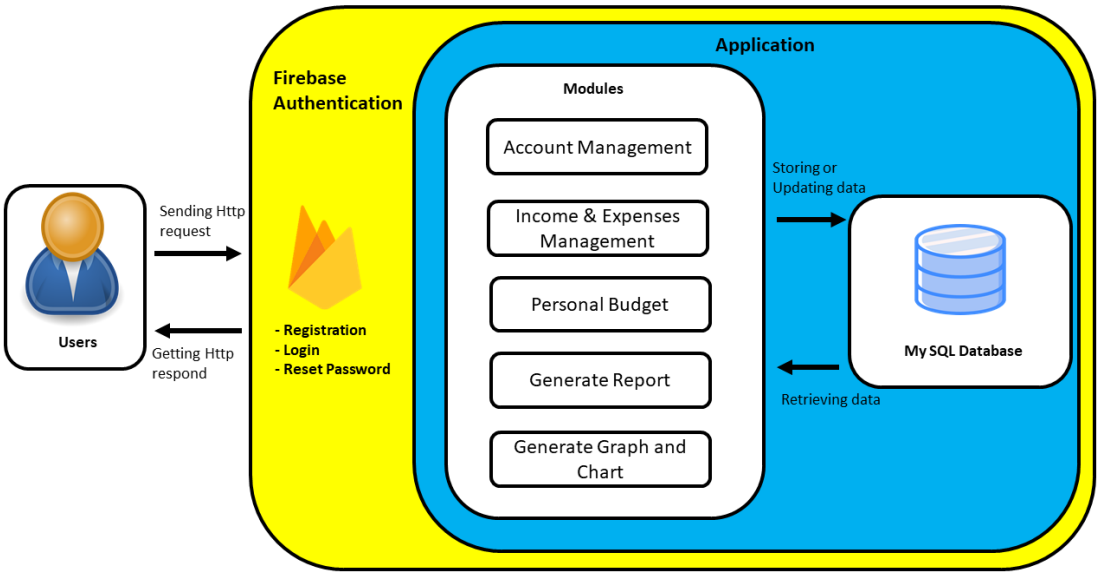


Figure 4.2.1 System Components Interaction Diagram

In the figure above, the diagram illustrates how this application created by component and component in the backside. First, if the user wants to access this application, the user has to perform registration and then proceed to login. All of this authentication is done by the external service, which is firebase. In addition, this firebase is also able to help user to reset password when the user has forgotten their login password. Once the user has passed all the authentication, the user can just perform the http request to use this application. Therefore, in the application component level, the application is ready to receive the user http request and then perform the application's module to the user. After that, any data that required to store, the application is able to communication with the database within the application component level. The communication between application and database included storing or updating data and retrieving data. Last but not least, every http request that made by the user is able to get a respond from the application once the application is done the request with the database.

4.3 System Function and User Interface Design

In this section, the system function and user interface of this application used will be described in detail. The detail will be discussed in different modules to be developed and the technology in each module.

4.3.1 User Management

This module is used to perform the login procedure of this application. University students can first register as valid users and then proceed to the login page. The functionalities of the user management module are listed and described as below:

1. Register User – University students are allowed to register as new users in order to manage their personal finance. All the personal finance will be based on the user account created by the university student. The university student needs to provide an email, username as well as password to register as a valid user. In addition, the system will avoid any duplicated records in the database. If the user exists in the database, the university student is not allowed to register.

2. Forgot or Change Password – This function allows a valid user to change their existing password by using an email verification.

3. Login – Once the university student has a valid user, the university student can log into the personal finance management system.

4. Edit User Profile – This function allows the university student to change the user information such as username and email.

The functions of this module can be accessed when the university student executes the application. Without performing through this module, a university student is not able to perform any personal finance management functions. The interface design for the login page is shown below in figure 4.3.1.

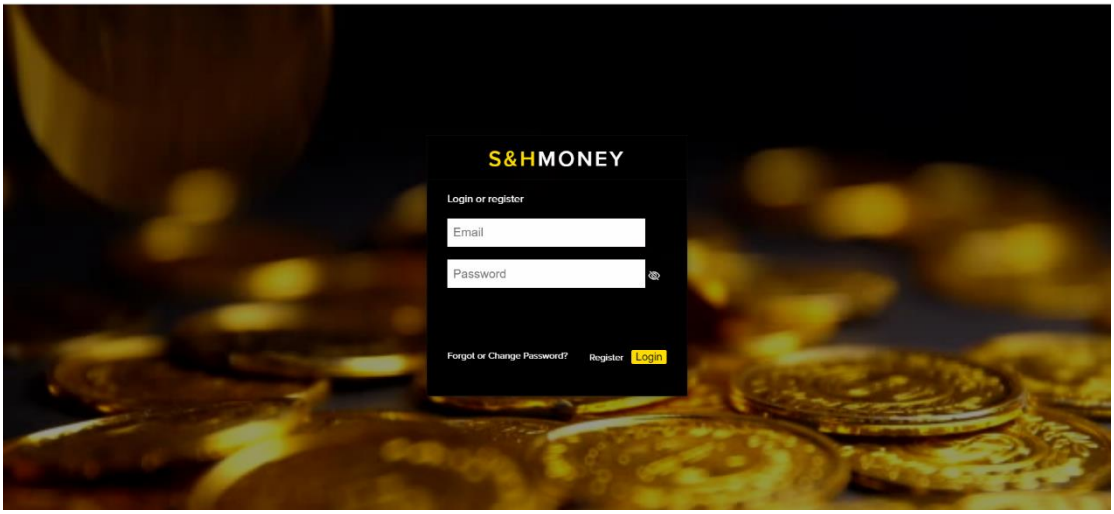


Figure 4.3.1 Login Page

The *Register* button above allows the interface in which a university student can register as a valid user. Besides, the *Login* button in the figure above enables the university to perform the login validation after filling in the email and password. Next, the *Forgot or Change Password* button above allows the university student to change their existing password. Figure 4.3.2 below shows the interface design for the Register functionality.

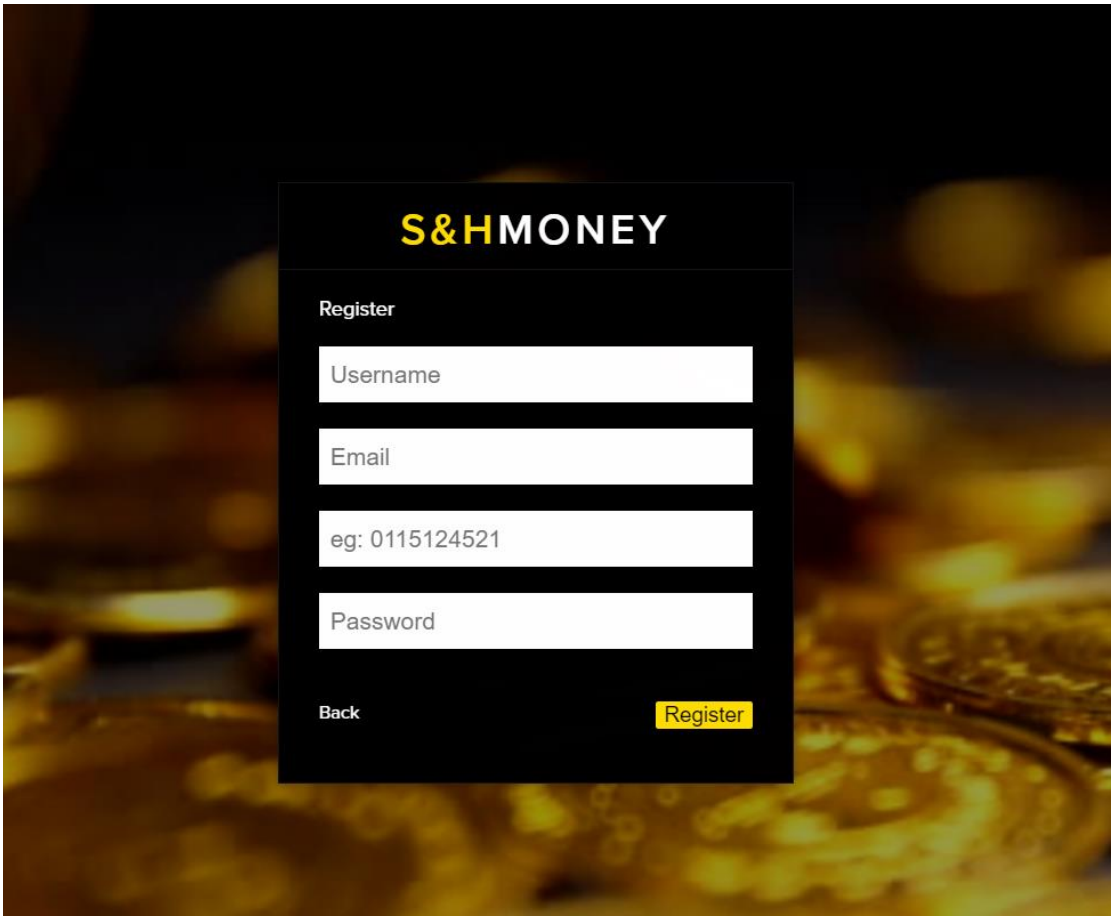


Figure 4.3.2 User Registration Page

On the user registration page, the university student needs to fill in the details in order to register as a valid user. Basic validation such as blank input will be implemented in this registration form to avoid unnecessary records in the database. All the input fields need to be keyed in the textboxes provided. After the university student entered all the input fields, the university student needs to click the *Register* button with yellow background to validate the information and store it into the database. Once all the validation and process finish, the system will switch the page to the login page. However, if the university student wants to cancel the registration process, the university student can click on the Back button to switch to the login page.

After that, if the university student thinks that their password is unsecure, the university student can click the *Forgot or Change Password* button on the login page. The interface design for the change password page is shown below in figure 4.3.3.

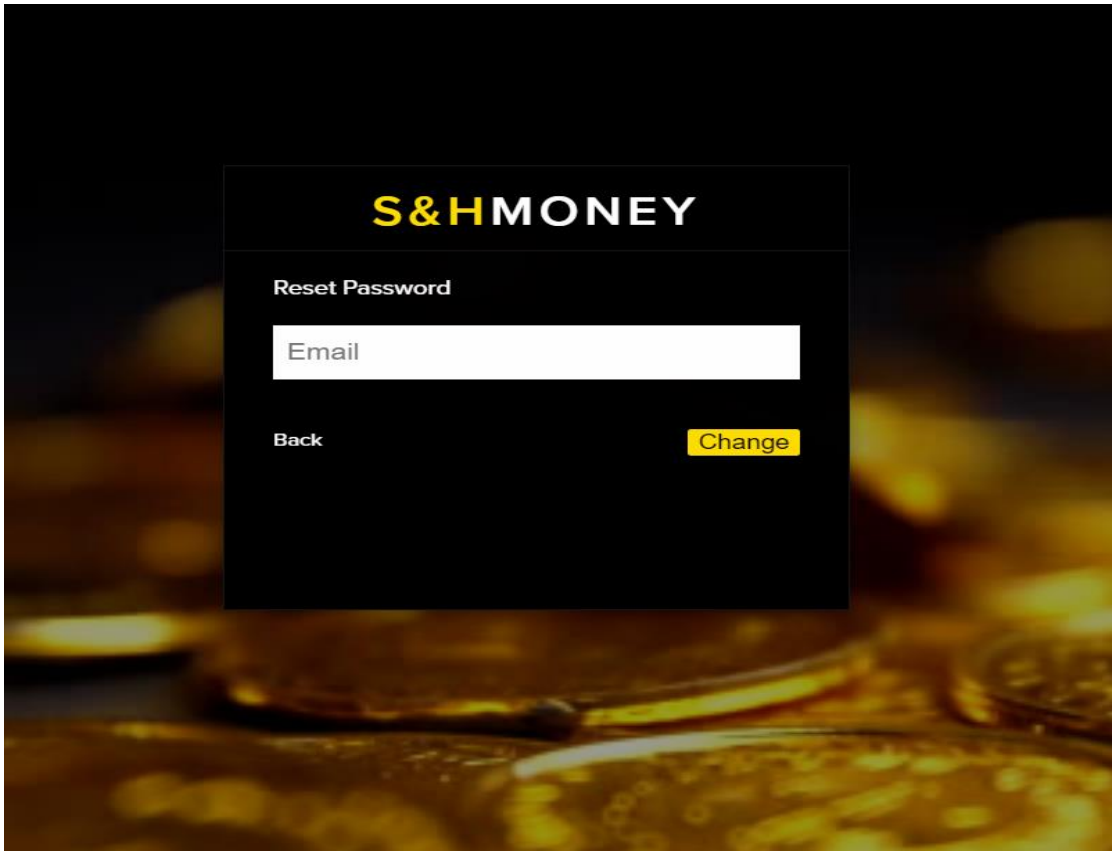


Figure 4.3.3 Forgot and Change Password Page

In the figure above, changing a password is similar to the registration process, where the input field needs to be entered. The validation of the email is required. If the email cannot be found in the database, the university student is not allowed to change and reset the old password. Once the university student entered the input field, the university student is allowed to click on the *Change* button with yellow background to let the system to validate the information and perform the functions required.

Once the user has successfully logged in, the system will display the home page of this application. The interface of the home page is shown below in figure 4.3.4.

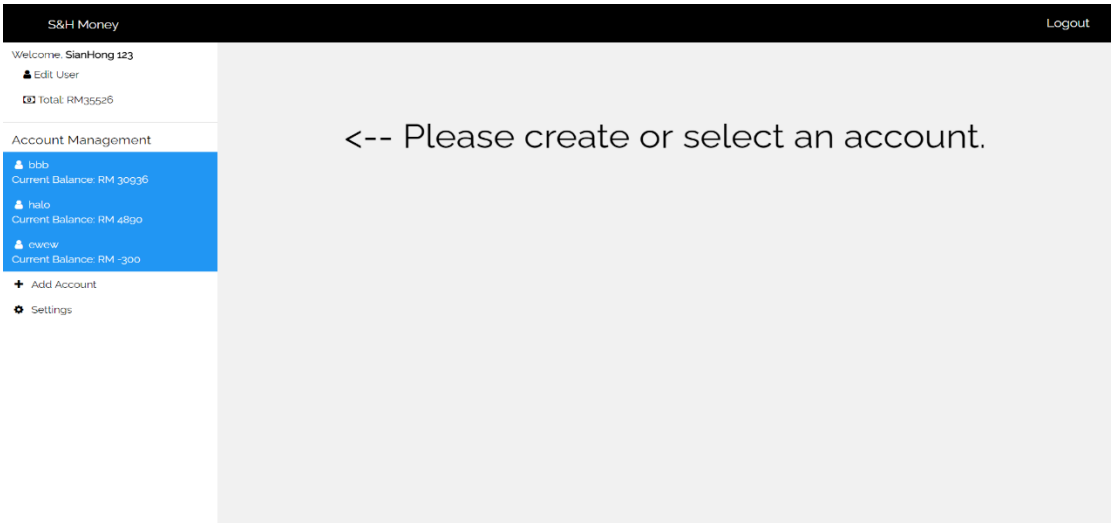


Figure 4.3.4 Home Page

In the figure above, the system will display a welcome message with the username on the left sidebar. In addition, below the welcome message, there is a clickable *Edit User* button. The purpose of this button is to let the user modify the user profile. Moving on, the system will always show the total balance of this user in the left sidebar as well.

Once the user clicks on the *Edit Use* button, the user will be prompted to the edit user page to change the user profile's information. The interface design for the user profile is shown below in figure 4.3.5.

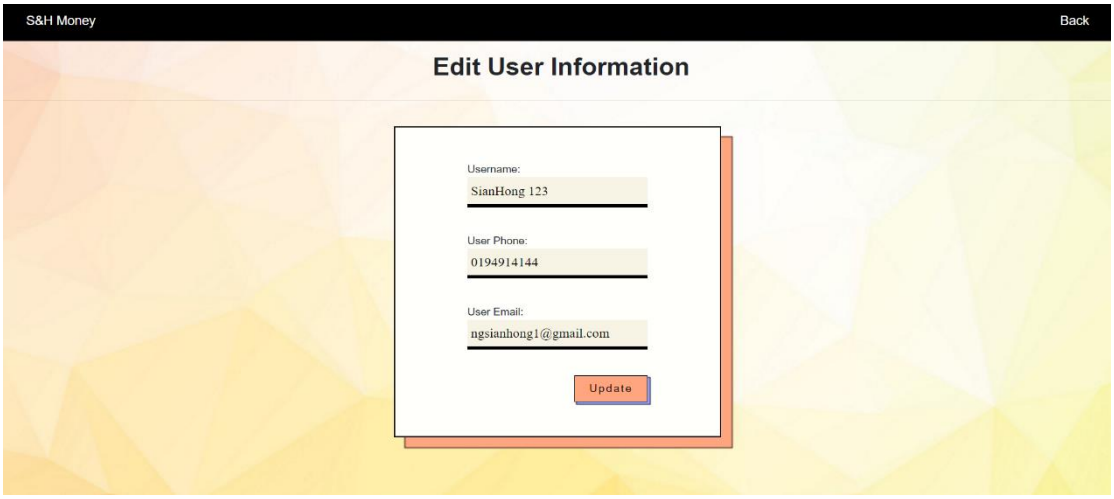


Figure 4.3.5 Edit User Page

In the figure above, the system will display the current user profile in the form for the user to refer to. The user is allowed to modify the user profile based on the

changes that the user intends to make. However, leaving the text fields in the form blank is strictly not allowed. Once the user has done the modification, the user can click on the *Update* button to perform the changes in the system

All in all, this module is the first step of this application. Allowing each user to register and log in with their own account ensures that the personal data of each user is not shared amongst users of the system.

4.3.2 Account Management

The Account Management module allows each user to create multiple accounts to perform personal finance management functions. Account management is crucial because all the transaction and budget planning will be based on the account. The functionalities of the Account management module are listed and described as below:

1. Add Account- A user is allowed to add an account into this application's database. This functionality consists of one activity. The activity prompts the user to enter the account's basic information such as account ID, account name, starting balance, and the balance date.
2. View Account- Since a user can have multiple accounts in this application, the user is allowed to view the existing accounts created by the user before. After that, the user is also allowed to perform the update and delete functions, which can be accessed from the view account page by clicking on the related buttons.
3. Update Account- Once the user clicks the Update button in the view account, the user can perform the update function by modifying the existing information in the update page.
4. Delete Account- The user is allowed to delete an existing account. Once the account is deleted, all the other related information, such as transaction and budget planning, will be deleted as well.

The functionalities above can be accessed after performing the login function. The system will show the dashboard to the user and the user can access these functions by clicking on the related buttons. The interface design for the account menu is shown below in figure 4.3.6.

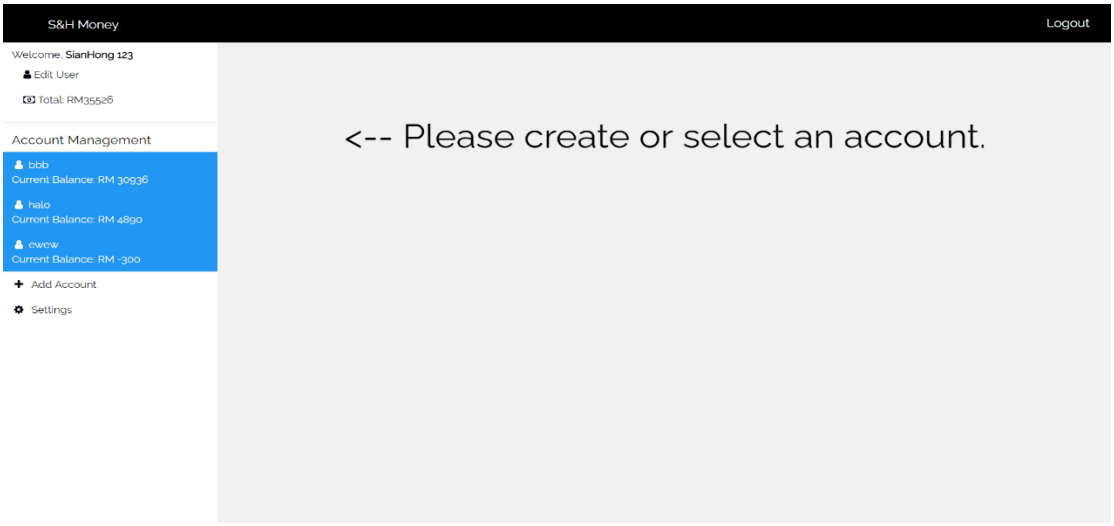


Figure 4.3.6 Account Menu

In the above figure, the user needs to create or select an account before performing personal finance management functions. The account menu is located in the left sidebar. Upon clicking the *Add Account* button, the user is prompted to add an account to this application. The interface design of the add account function is shown below in figure 4.3.7.

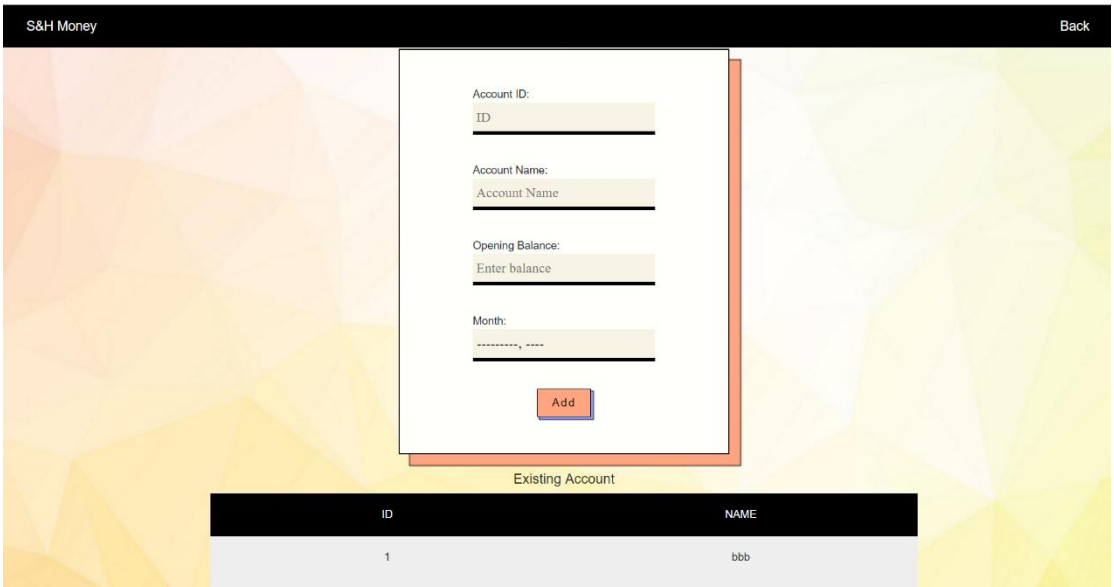


Figure 4.3.7 Add Account Page

In the figure above, the user is allowed to add the details of the new account by entering a unique Account ID, Account Name, Opening balance, and the Month (year and month of creation). Basic validations will be implemented in this form. All the

input fields have to be filled in by the user in the textboxes except for the Month, in which the user will be prompted to select from a small calendar. Once these details have been validated successfully and added to the database, the system will switch the current page to the home page. In addition, at the bottom of the page, there is a table showing the existing accounts. The purpose of this table is to prevent users from adding duplicated accounts as the user is allowed to refer to the table to see whether the new account to be added already exists in the table.

Once the user has added the new account, the user can view the account list by clicking the *Setting* button on the account menu. The interface design for the list of accounts is shown below in figure 4.3.8.

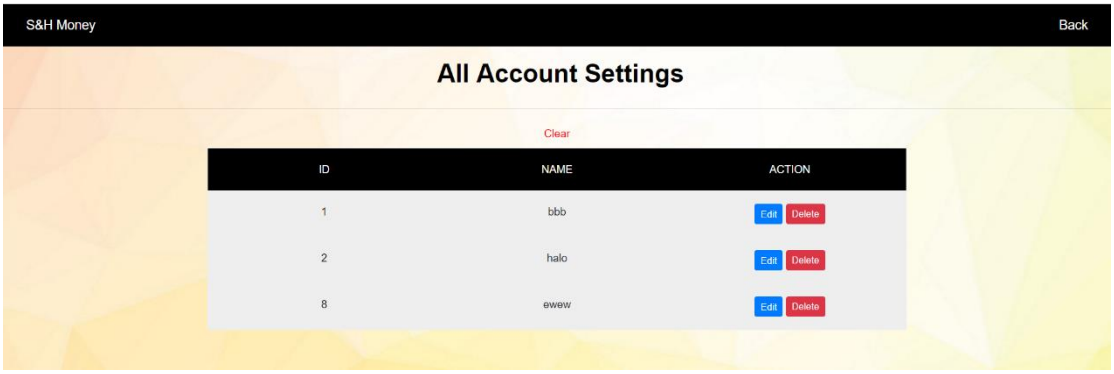


Figure 4.3.8 All Account List

In the figure above, the list of accounts is displayed as a table to the user. In the table, only the account ID and account name are displayed along with the two buttons which are *Edit* and *Delete*. If the user wants to delete an account, the user can click on the *Delete* button on that particular row. When the *Delete* button is clicked, the system will delete the particular account and update it to the database. If there are any transactions and budget planning in this account, the corresponding records will be deleted as well.

In addition, once the user clicks on the *Edit* button, the system will switch the page to the edit account page. The interface design of the edit account page is shown below in figure 4.3.9.

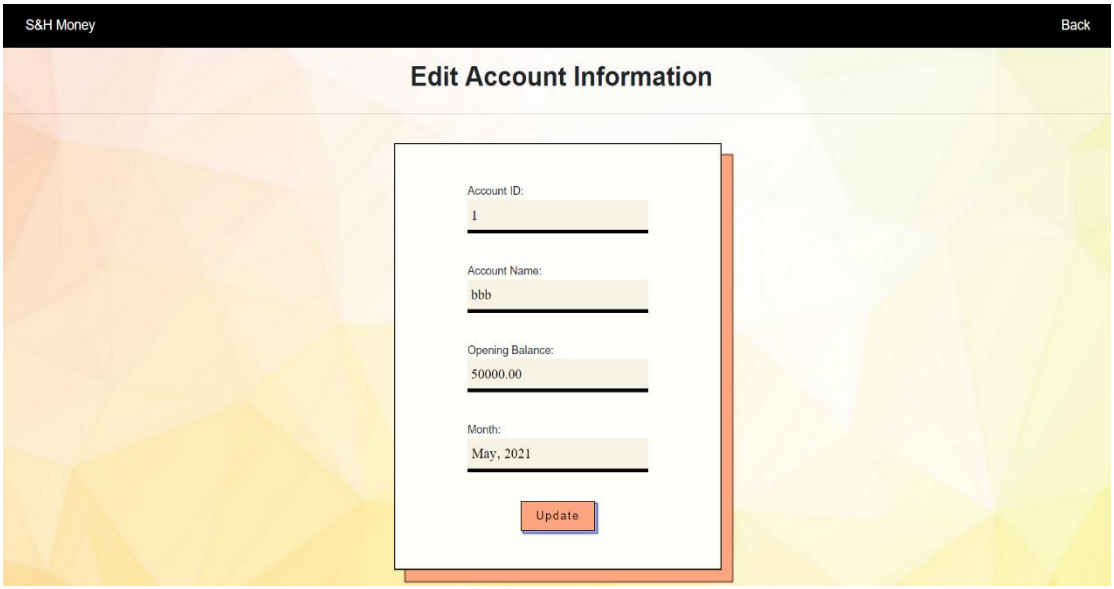


Figure 4.3.9 Edit Account Page

In the figure above, once the user clicked on a particular account's *Edit* button in the account list, the system will display the account information in the form for the user to refer. The user is allowed to modify the information based on the changes that the user intends to make. However, leaving the text fields in the form blank is strictly not allowed. Once the user has finished the modifications, the user is allowed to click on the *Update* button to commit the changes.

After performing all the account management, if the user wants to do the personal finance management, the user can click on any account in the account menu list.

4.3.3 Income and Expenses Management

This Income and Expenses Management module allows user to add their daily income or expenses into this application. Before adding an income or expense, the user needs to select an account first. After that, the user interface will display all the information related to the account, such as basic navigation, the latest income or expenses, and a summary chart. The functionalities of the Income and Expenses Management module are listed and described as below:

1. Add Income and Expenses- A user is allowed to add an income or expenses into this application's database. This functionality consists of two activities. The first activity prompts the user to add a category first. The second activity prompts the user to enter the income or expenses basic information such as description, select category, amount as well as the date. Basic validation will be implemented to all the input fields.
2. View Income and Expenses - The user is allowed to view the existing expenses or income that are created. After that, the user also allows to perform the update and delete function in this view income and expenses function by clicking on the related button.
3. Update Income and Expenses - Once the user clicks the Update button in view income and expenses, the user can perform the update function by modifying the existing information in the update page.
4. Delete Income and Expenses - The user is allowed to delete an existing income or expense. Once the account is deleted, all the other related information, such as transaction and budget planning, will be deleted as well.
5. Search Income and Expenses – The user is allowed to select a particular month and year and click on the *Search* button to prompt the system to find the relevant results. Once the system finds the related result in the database, the system will update the result.
6. Add Category- The user is allowed to add a new category name for tracking the income and expenses. Both income and expenses will have different categories. All the categories to be added will be based on the type of transaction.
7. View Category- The user is allowed to view all the existing category names in the category list. After that, the user is also allowed to perform the update and delete function in this view category function by clicking on the related button.

8. Update Category – The user is allowed to update the category name after clicking the *Update* button in the view category list. The user can perform the update function by modifying the existing information in the update page.
9. Delete Category - The user is allowed to delete an existing category name. If the system finds that other transactions or budget records are using the category name, then the user is not allowed to delete the category.

The functionaries above can be accessed from the dashboard by clicking on the *Income* or *Expenses* button. After that, the system will switch the page to the income or expenses menu. In the menu, there will have options to manage categories, add transactions and manage transactions. The interface design after the user selected an account is shown below in figure 4.3.10.

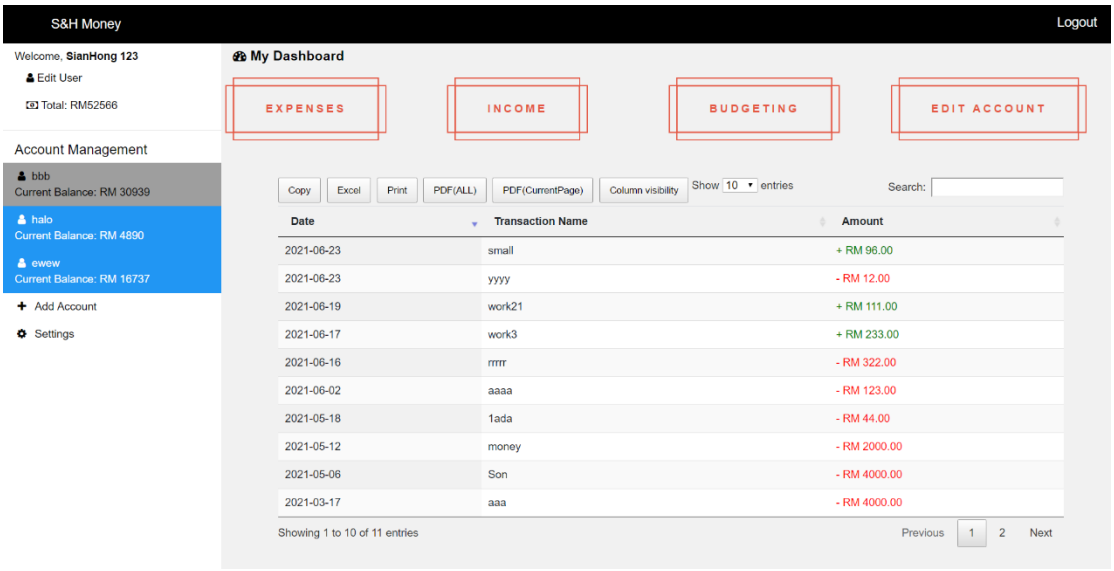


Figure 4.3.10 The Dashboard and Main Menu

In the figure above, there are four navigation buttons in the dashboard. Each of the buttons will navigate the user to the different pages once the user clicks on it. After that, the section below the dashboard's navigation bar shows the recent transactions for the current account. Since there might be many transactions, the user can perform the search function to search for a particular keyword or date and also use the pagination function to see the other transactions. Other than that, user is allowed to select how many records need to be display in one page.

Upon clicking the *Expenses* or *Income* button in the dashboard, the user is prompted to the expenses or income menu. The user interface expenses menu is shown below in figure 4.3.11.

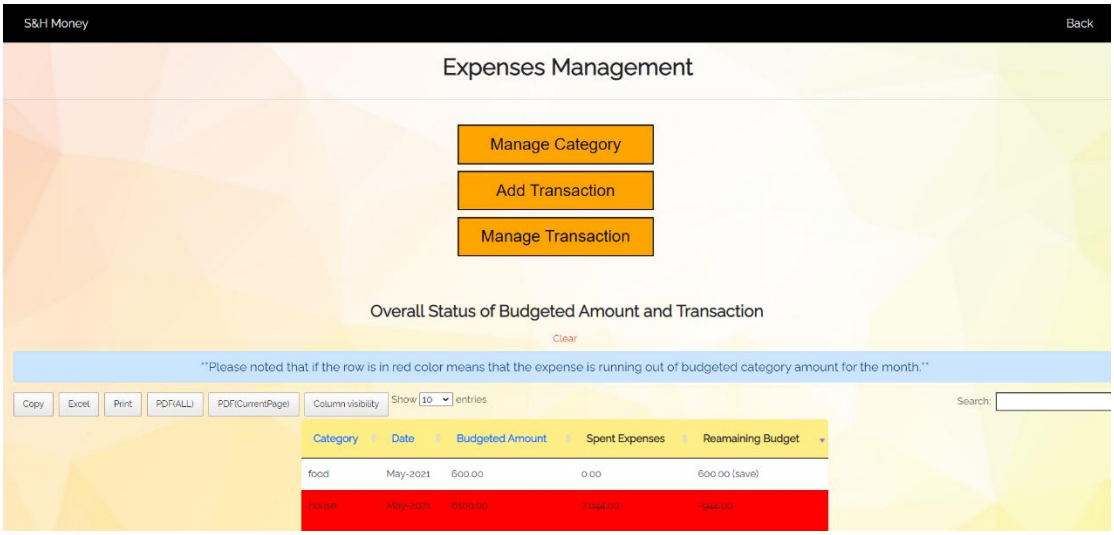


Figure 4.3.11 Expenses Menu Interface

In the figure above, the overall status of the budgeted amount and transaction will be shown in the table at the bottom of the page. This table allows the user to keep track of the budget plan easily. However, for the income menu, there is no table to display the budget plan. The reason is that the budget plan is only allowed to be used in expenses. The user interface of the income menu is shown below in figure 4.3.12.

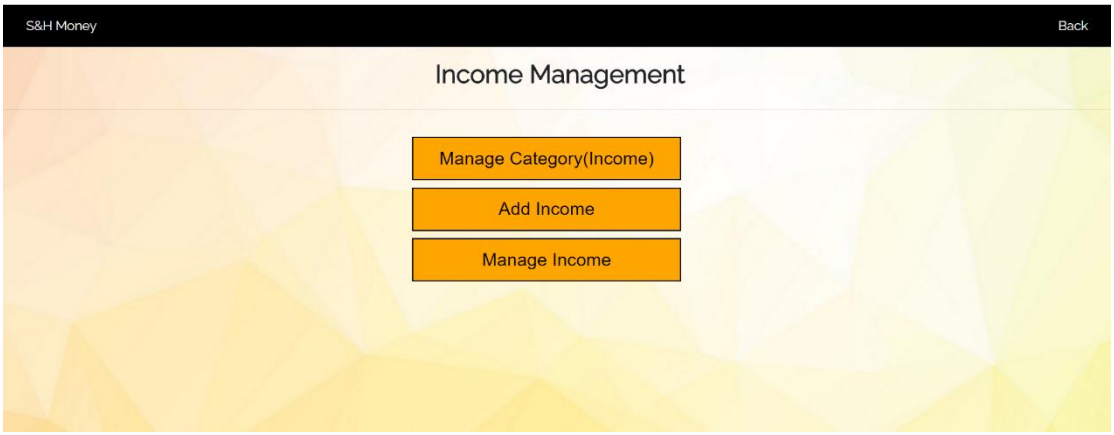


Figure 4.3.12 Income Menu Interface

Upon clicking the *Manage Category* button, the system will switch the page to the manage category page. The user interface of the manage category page is shown below in figure 4.3.13.

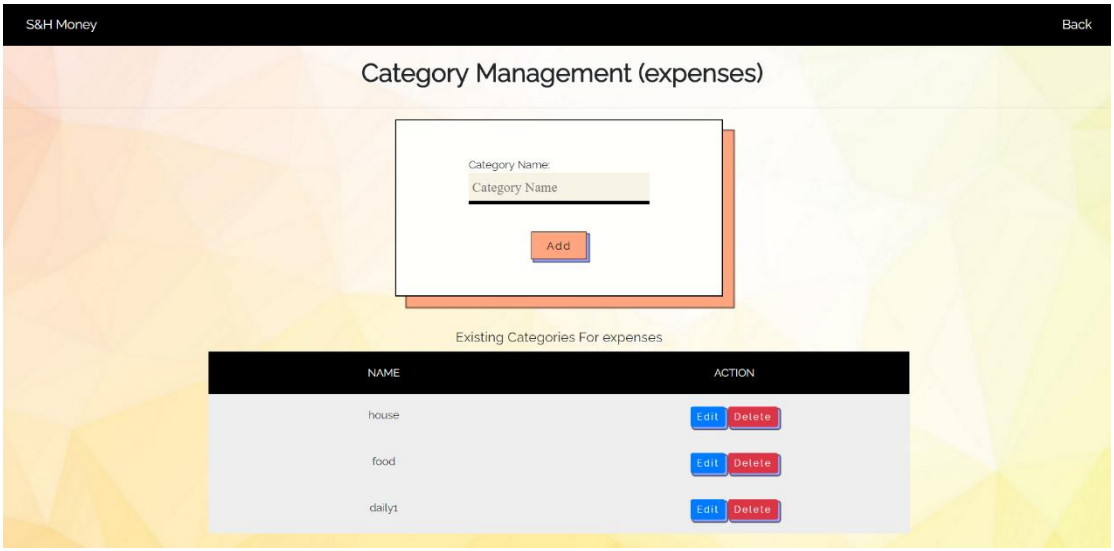


Figure 4.3.13 Category Management Page

In the figure above, it is shown that the category page is applicable for both the income and expenses. If the user arrives at this page from the expenses menu, this page will automatically detect the transaction type to be expenses. On the other hand, if the user navigates to this page from the income menu, the type of category will be changed to income. In this page, the user is allowed to perform the add, update and delete category functions. Once the new record has been added, the table will dynamically update the data and display the latest list of categories.

After the user has done the category management, the user can go to add the new transaction. The user interface of add new transaction is shown below in figure 4.3.14 and 4.3.15.

Figure 4.3.14 Add New Transaction(Expenses)

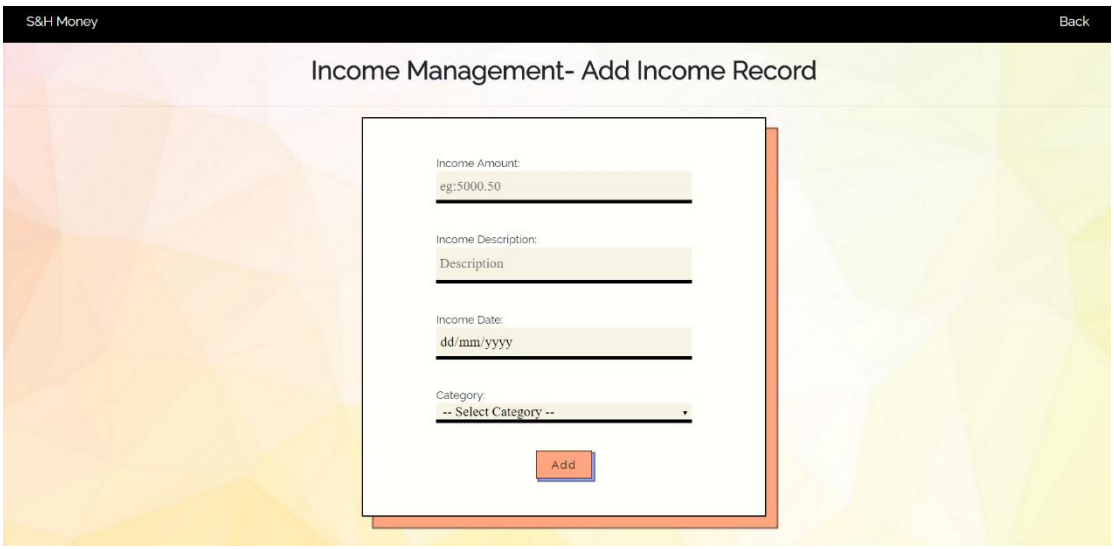


Figure 4.3.15 Add New Transaction(Income)

In the figure above, the user is allowed to add a new transaction by filling in the required information. Proper validation will be implemented for all the input fields to prevent the invalid format of data store into the database. All the input fields need to be filled in by the user in the textboxes or except for the category fields. This field will use a drop-down list to show all the existing categories. After that, the user can choose the category accordingly. Once all the input fields have been entered, the user can check the existence of the new record by referring to the table. If the new record does not exist in the table, the user can click on the *Add* button to store the data into the database.

Once the user has added a new transaction, the user can view the transaction details by clicking on the *Manage Transaction* button on the menu. The interface design for the list of transactions is shown in figure 4.3.16 and figure 4.3.17.

Figure 4.3.16 List of Transactions (Expenses)

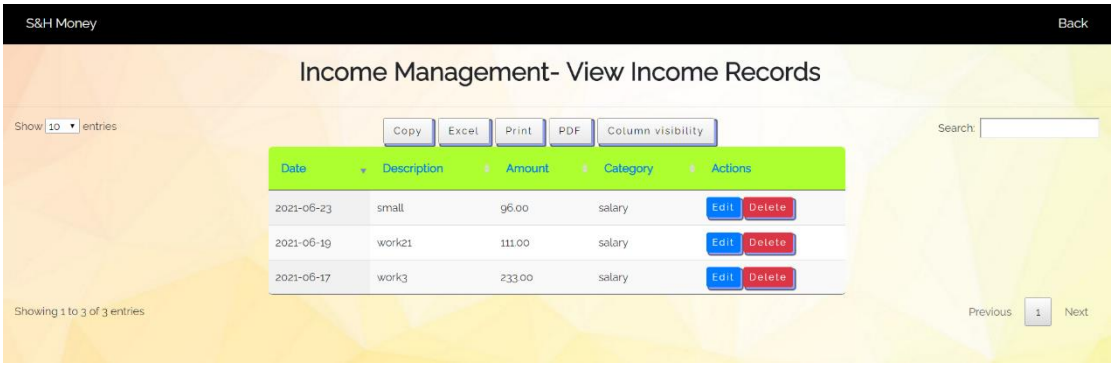


Figure 4.3.17 List of Transactions (Income)

The two figures above display all the existing transactions to the user. In the list, the date, description, amount, and category will be displayed to the user. In figure 4.2.16, if the row in the list has a red background, it means that the expenses have already run out of this category's budget. Furthermore, the user can perform the sorting function by clicking on the column header to sort the list accordingly. Next, all the pagination and search functions will be implemented in this view transaction lists. In addition, if the user wants to delete a transaction, the user can click on the *Delete* button on that particular row. When the *Delete* button is clicked, the system will delete the particular transaction and update it to the database.

Furthermore, once the user clicks on the *Edit* button, the system will switch the page to the edit transaction page. The interface design of the edit transaction page is shown in figure 4.3.18 and figure 4.3.19.

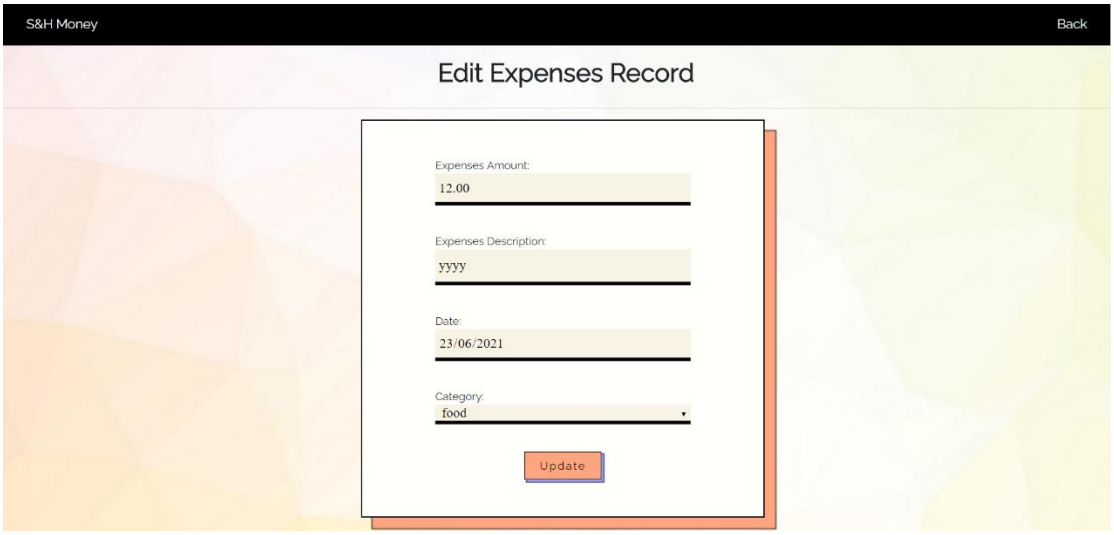


Figure 4.3.18 Edit Transaction (Expenses)

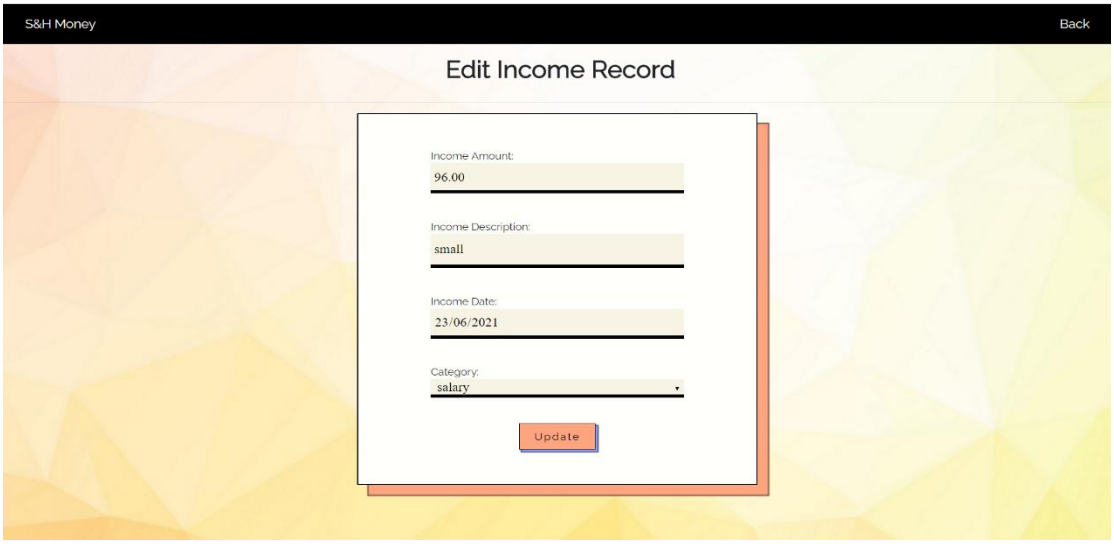


Figure 4.3.19 Edit Transaction (Income)

In the figure above, once the user clicked on a particular transaction's *Edit* button in the transaction list, the system will display the transaction information in the form for the user to refer to. The user is allowed to modify and change the information based on the changes that the user wants to make. However, leaving the text fields in the form blank is strictly not allowed. Once the user has finished making the modifications, the user is allowed to click on the *Update* button to update it to the database.

4.3.4 Personal Budgeting

The personal budgeting module allows users to add budget planning to this developed application. Before using this personal budgeting module, the user has to go through all the previous modules such as, user management module, account management module as well as income and expenses module. The functionalities of this personal budgeting module are listed below:

1. Add Account Budget Plan- A user is allowed to add an account budget plan into this application's database. This functionality consists of only one activity, which prompts the user to enter the account budget plan information such as the budget amount and the date. Basic validation will be implemented to all the input fields.
2. View Account Budget Plan - The user is allowed to view the existing account budget plan that has been created before. After that, the user is allowed to perform the update and delete functions, which can be accessed from the view account budget plan page by clicking on the related buttons.
3. Update Account Budget Plan- Once the user clicks the *Update* button in the view account budget plan, the user is allowed to perform the update function by modifying the existing information in the update page.
4. Delete Account Budget Plan - The user is allowed to delete an existing account budget plan. Once the account is deleted, all the other related information, such as category budget plan, will be deleted as well.
5. Search Account Budget Plan – The user is allowed to search a particular result by selecting the Month and Year. After that, the user can click on the *Search* button to search the result from the database.
6. Add Category Budget Plan- Before adding a category budget plan, the user must add an account budget plan. After that, the user is allowed to add a new category budget plan for managing their personal finance. This functionality consists of only one activity. The user needs to enter the category budget plan information, such as the budget amount in this activity. After that, the user must choose an existing account budget plan and category name from the drop-down list to add a new category budget plan.
7. View Category Budget Plan - The user is allowed to view all the existing category budget plans from a list. After that, the user is allowed to perform the update and

delete functions, which can be accessed from the view category budget plan page by clicking on the related buttons.

8. Update Category Budget Plan – The user is allowed to update the category budget plan after clicking the *Update* button in the view function. The user can perform the update function by modifying the existing information in the update page.
9. Delete Category Budget Plan - The user is allowed to delete an existing category budget plan by clicking the *Delete* button in the view function.

The functionalities above can be accessed from the main dashboard by clicking on the *Budget* button. After that, the application will display the budget menu, which includes the options to manage the account budget plan, category budget plan, and edit account budget plan. The interface design for the attendance menu is shown below in figure 4.3.20.

Figure 4.3.20 Budget Menu Interface

Upon clicking the *Account Budget Planning* button, the application will display a form to prompt the user to enter the account budget plan information. The interface design of the account budget plan activity is shown in figure 4.3.21.

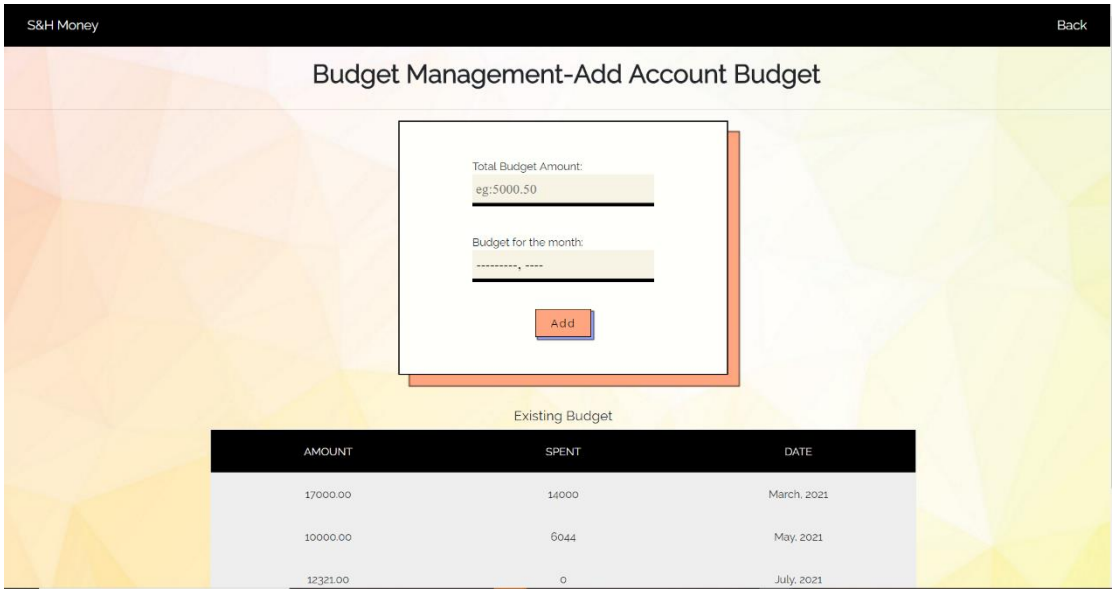


Figure 4.3.21 Add Account Budget Plan

In the figure above, the user is allowed to add a new account budget plan by filling in the details into the form given by the application. These details included two input fields: the total budget amount and the Month of the budget plan. The first input field requires the user to enter a total budget amount into the textbox. The second input field is a small calendar to let the user choose the Month and year. The basic validations such as empty input, duplicated record, and invalid format will be implemented in this form. If the details have been entered, the user can click on the *Add* button to add the record. Once these details have been validated without any problem, the application will store the new record into the database. After that, the table below will also update the existing budget plan dynamically to let users view the existing budget plan.

Once the user has created an account budget plan, the user is allowed to add category budget plans by clicking on the *Expenses(Category) Budget* in the budget menu. The user will then be prompted to add, delete or update a category budget plan. The user interface design is as illustrated in figure 4.3.22.

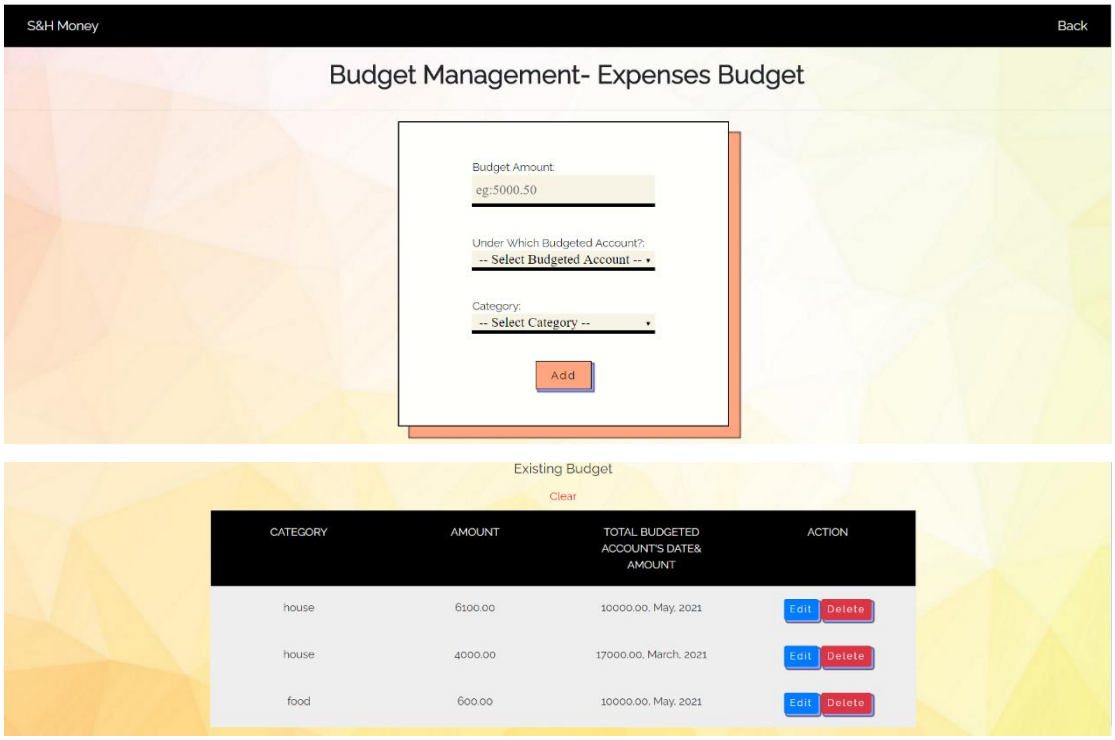


Figure 4.3.22 Category Budget Plan Management Interface

In the figure above, the user will first see a form that allows them to enter a category budget plan details into the input fields. The details in the form include the budget amount, from which account budget plan and which category. For the budget amount, the user needs to enter a valid number into the input field. Basic validations will be implemented for this input field. As for the two remaining input fields, the application will use a drop-down list to list all the existing account budget plans and category names to the user. The user is allowed to select one of the options to perform the category budgeting. If there are no account budget plans or categories, the application will prompt an error message to notify users to add the information. Once all the input fields have been filled in, the user can click on the *Add* button to submit the new category budget plan. The application will validate the new record after the user clicks on the *Add* button. If the category budget plan exceeded the account budget plan, the application will prompt an error message to the user. After that, if the application validates the new record without error or duplication, the application will store the new category budget plan into the database.

Furthermore, the existing category budget plan will be listed in the table at the bottom of the page. In this table, the user can perform the edit and delete function by

clicking on the related button and at a particular row. Once the user clicks on the *Delete* button, the particular category budget plan will be deleted immediately.

Moreover, the user can click on the *Edit* button to update a particular category plan. Once the user clicks on the *Edit* button, the user will be redirected to the update form to modify the budget plan. The user interface design is designed as illustrated in figure 4.3.23.

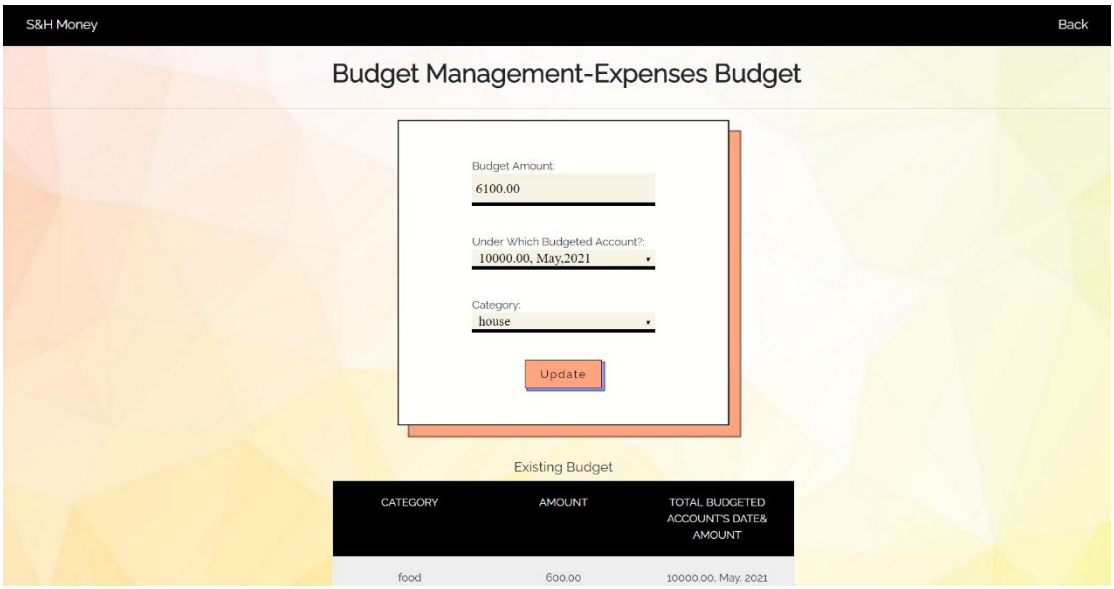


Figure 4.3.23 Update Category Budget Plan Interface

In the figure above, once the user clicked on a particular category budget plan's *Edit* button in the table list, the system will display the information in the form for the user to refer to and modify. The user is allowed to modify and change the information based on the changes that user intends to make. However, leaving any tex fields in the form blank is strictly not allowed. All the validation in the adding part will also be implemented in this update form. Once the user has finished the modifications, the user is allowed to click on the *Update* button to update it to the database.

Upon clicking the *Edit Account Budget Planning* button in the budget menu, the user will be redirected to the view account budget plan interface. The user interface for the budget management is shown below in figure 4.3.24.

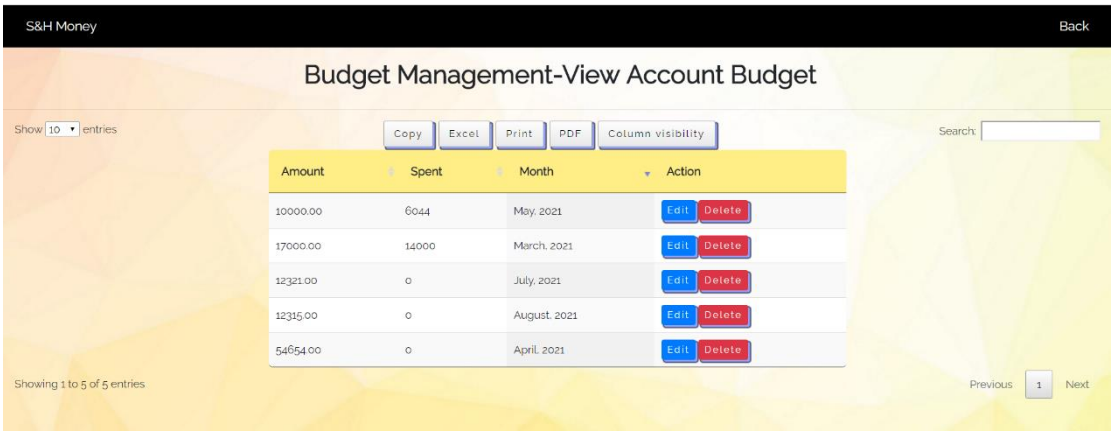


Figure 4.3.24 View Account Budget Plan Interface

In the figure above, the existing account budget plan will be listed in the table. This table shows the budget plan amount, spent amount, and the Month of the budget plan. Users can refer to the table to see the current status of each of the account budget plans. Next, user is allowed to perform sorting, searching and pagination in this table. In addition, the user can perform the edit and delete function by clicking on the related button in a particular row. Once the user clicks on the *Delete* button, a particular account budget plan will be deleted immediately.

Furthermore, the user can click on the *Edit* button to update a particular account budget plan. Once the user clicks on the *Edit* button, the user will be redirected to the update form to modify the budget plan. The user interface design of the update form is designed as in figure 4.3.25.

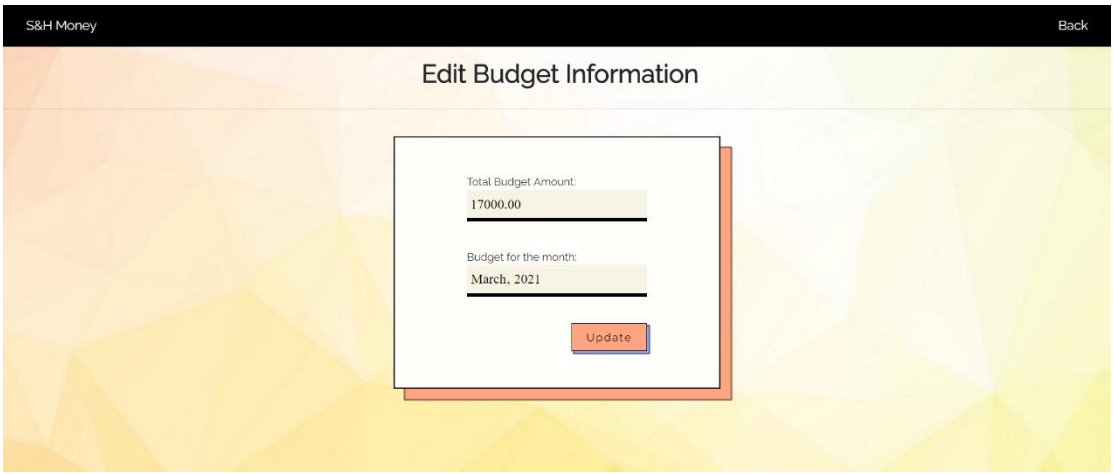


Figure 4.3.25 Edit Account Budget Plan Interface

In the figure above, once the user clicked on a particular account budget plan's *Edit* button in the table list, the system will display the information in the form for the user to modify. The user is allowed to modify and change the information based on the changes that the user is wanted. The input fields will be the same as the adding part, which only two input fields. The two input fields are the budget amount and budget date. However, leaving text fields in the form blank is strictly not allowed. All the validation in the adding part will also be implemented in this update form. Once the user has done the modifications, the user is allowed to click on the *Update* button to update it to the database.

4.3.5 Generate Report

The Generate Report module allows users to export the income and expenses transactions to an external data file such as pdf and CSV. Not only that, but the user is also allowed to generate reports based on the budget plan. The functionalities of the generate report module are listed below:

1. Generate Income and Expenses Report- The user is allowed to generate a report based on the income and expenses. By default, if the user does not specify any month or year of the report, this function will generate the report based on the current month and year income and transaction. In addition, the user is allowed to choose a specific month and year to generate the income and expenses report.
2. Generate Budget Plan Report- This function allows users to export the previous budget plan to a report. The user is allowed to click on the *Generate Report* button to generate the budget plan report. The report will show the initial amount of the budget plan, the amount spent, and the amount remaining.

The functionalities above can be accessed from the view list of each module. In the view list, there were some buttons including pdf, csv, and print to allow users to generate reports. The interface design for the generate report is shown below in Figures 4.3.26 and 4.3.27.

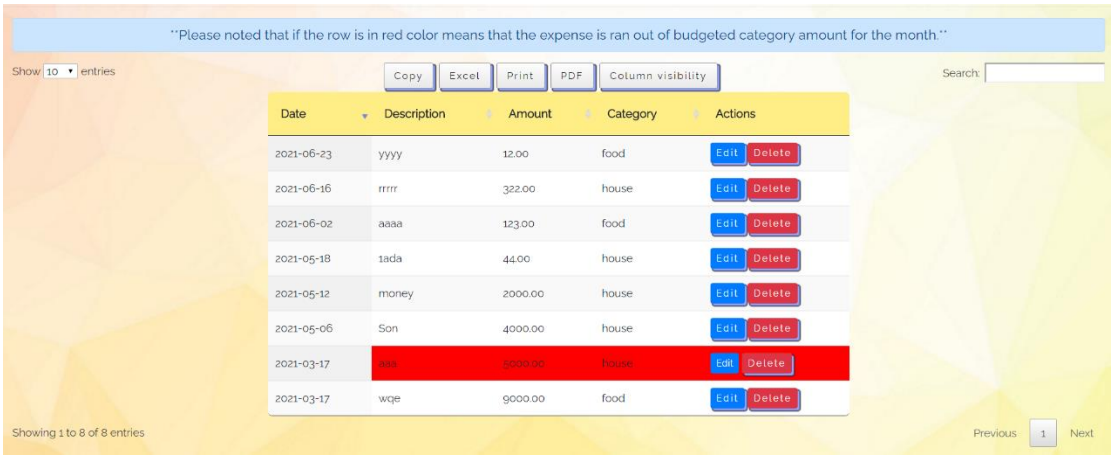


Figure 4.3.26 Generate Report Interface (Expenses & Income)

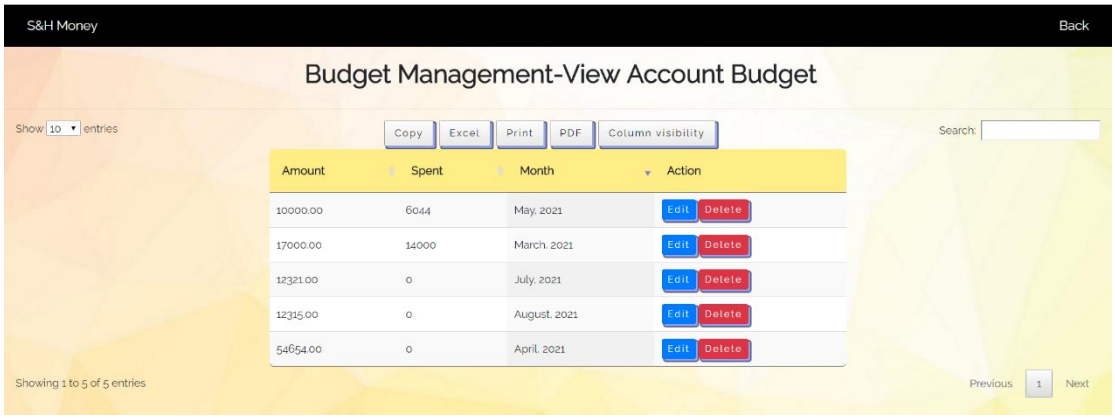


Figure 4.3.27 Generate Report Interface (Budget)

In the figure above, the user can click on the *PDF*, *Excel* buttons to generate different types of reports. The user is able to generate a report and can customize the view list as well as copy the records in the clipboard. Once the user has clicked on the generate report button, the system will prompt the user to choose the file path and create a file name to download the report. The user interface design is as illustrated in figure 4.3.28.

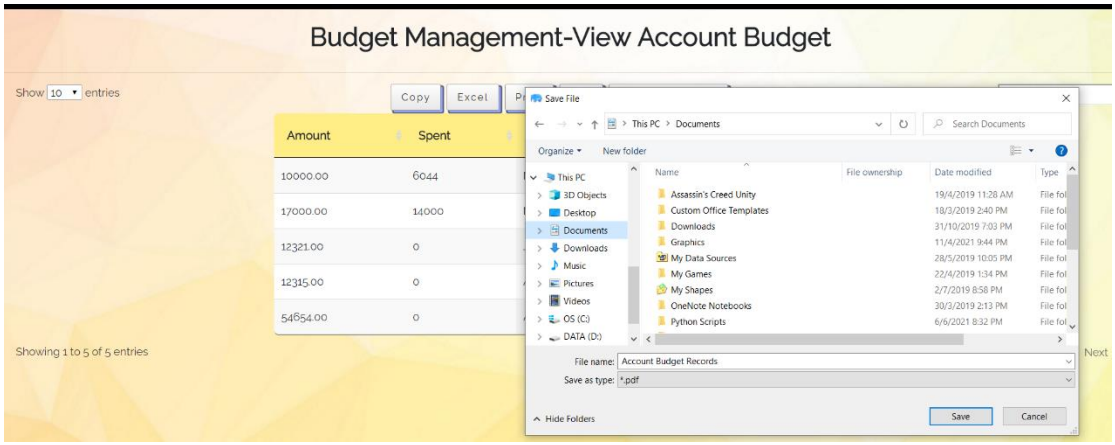


Figure 4.3.28 File Path to Generate Report

Once the user has selected a file path and modified the file name, the user can click on the *Save* button to generate the report. After that, the report will be generated in the related file path. The report format design is shown below in Figures 4.3.29 and 4.3.30.

Account Budget Records

Amount	Spent	Month
10000.00	6044	May, 2021
17000.00	14000	March, 2021
12321.00	0	July, 2021
12315.00	0	August, 2021
54654.00	0	April, 2021

Figure 4.3.29 Report Format(Budget)

Expense Records

Date	Description	Amount	Category
2021-06-23	yyyy	12.00	food
2021-06-16	rrrrr	322.00	house
2021-06-02	aaaa	123.00	food
2021-05-18	1ada	44.00	house
2021-05-12	money	2000.00	house
2021-05-06	Son	4000.00	house
2021-03-17	aaa	5000.00	house
2021-03-17	wqe	9000.00	food

Figure 4.3.30 Report Format(Expenses)

In the figure above, all records have been exported to a pdf file in the user's device. Users can go to the file path that was selected earlier to find the exported file. In addition, the user also is allowed to key in a specific date such as "2021/06" to search the records and export to a data file by clicking on the *PDF(CurrentPage)* button. The example of searching specific dates is shown in figure 4.3.31.

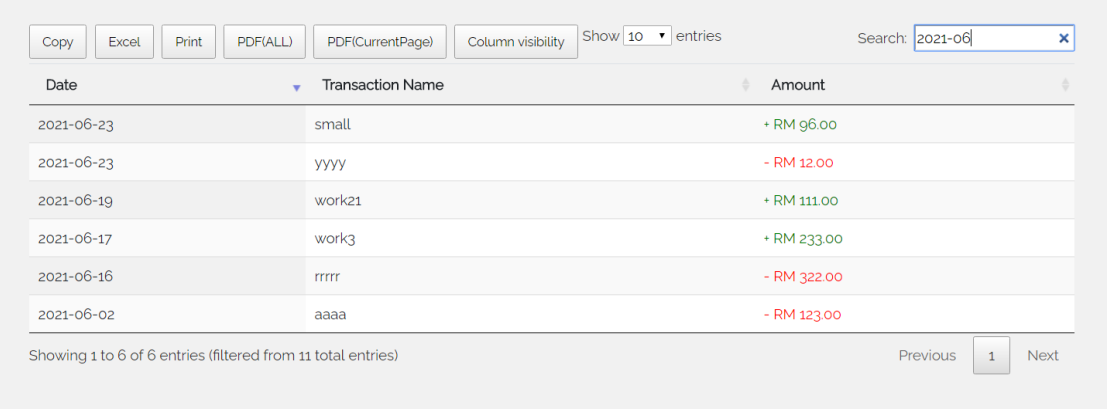


Figure 4.3.31 Search specific date to generate report

In the figure above, once the user clicks the related button, the system will proceed to the process same as the previous figure, such as choosing the file location and naming the file.

4.3.6 Generate Graph and Chart

The generate graph and chart module allows the user to see the summary of their personal finance. Basically, all the graphs and charts will change dynamically upon the user has done some action such as add transaction or update transaction. In addition, a basic graph and chart will be displayed based on the budget plan to show the percentages of the budget plan that is being used. The functionalities of this generate report module are listed below:

1. Generate Graph and Chart for income and expenses- The user is able to see the summary chart and graph after adding some income and expenses. In addition, if the user wants to see the previous year's summary charts and graphs, the user is allowed to select a specific year in a drop-down list to display. By default, the application will display the current year summary chart.
2. Generate Graph and Chart for all account usage- The user is able to see a summary chart and graph to display the percentage for each account's balance.
3. Generate Graph or Chart for Budget Plan- A summary graph or chart will be displayed to show the amount of the budget plan that has already been saved.

The functionalities above can be accessed from the home page by scrolling down until the bottom of the overall transaction table. The layout of this user interface are shown in figure 4.3.32 and 4.3.33.

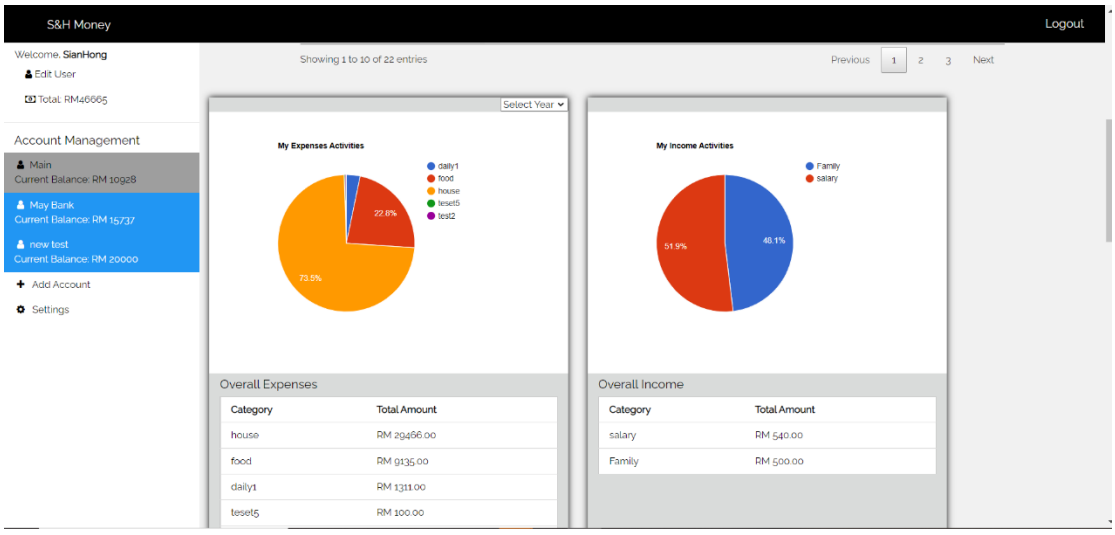


Figure 4.3.32 Income and Expense Pie Chart

Figure 4.3.33 Income and Expense Line Chart

In the figure above, the user can perform filter function to choose which year's income and expense to summarise into the chart. If there is no data for the particular year, the chart will display no data to the user. After that, an account status summary chart is displayed as well. Figure 4.3.34 shows the layout of the account status chart.

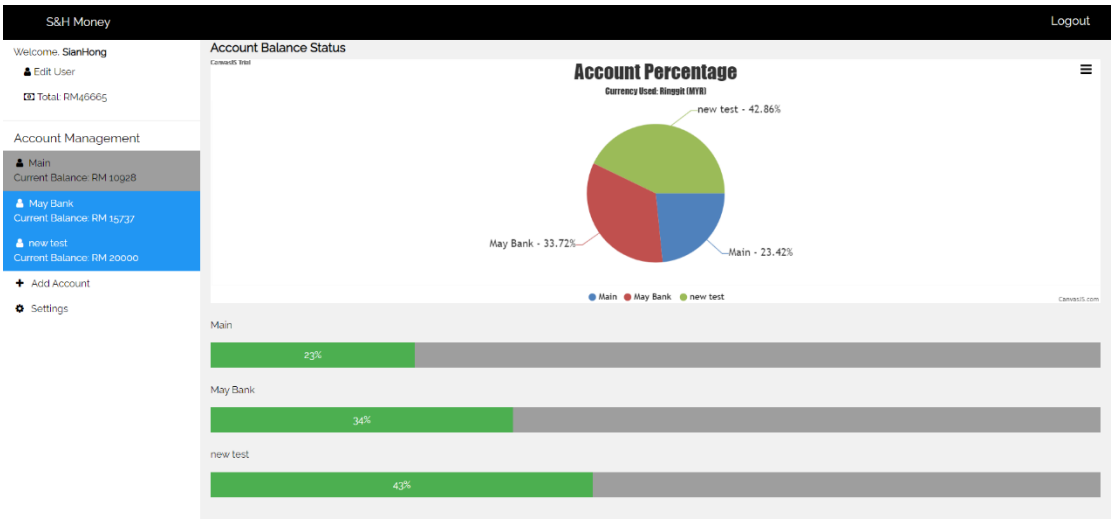


Figure 4.3.34 Account Balance Status Chart

In the figure above, an overall pie chart is shown to let user know which account is being mainly used in his or her personal finance status. In the above pie chart, the user can hover the pie chart to see the accurate percentage for each account. The reason is because this effect can increase the user interface experience. Furthermore, a budget plan chart is displayed below this account status pie chart. The budget plan chart is shown in figure 4.3.35.

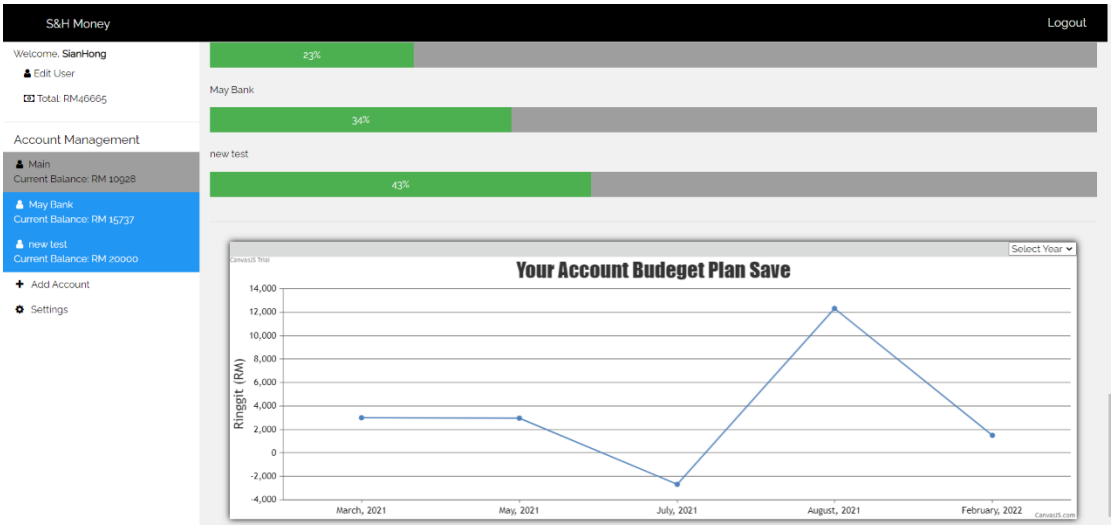


Figure 4.3.35 Budget Plan Saving Line Chart

In the figure above, the line chart will be based on the budget plan that user has added in the personal budget module to summarize the budget save value. This line chart is able to give user has a better view of the result for a particular budget plan.

CHAPTER 5

SYSTEM IMPLEMENTATION

In this chapter, the system implementation of this application is explained in detail by including the project setup, project configuration and module completion for the six modules. The six modules including User Management, Account Management, Income and Expenses Management, Personal Budgeting, Generate Report and Chart. Last but not least, this chapter closes up with a concluding remark regarding the entire chapter.

5.1 Project Setup

Project setup is the first step to develop a project. First, the project environment will be set up. In the below section, the configurations of Visual Studio Code and MySQL are discussed in detail.

PHP language was chosen to develop this application. Since Visual Studio Code supports many programming languages, PHP libraries should be imported to support this application development. The essential external and internal libraries and their respective functions are shown below in Table 4.1.

Table 5.1 External and Internal Libraries Used in Visual Studio Code

Library Name	Implementation Name	Description
UI Component	https://www.w3schools.com/w3css/4/w3.css	Used to provide various CSS designs for UI, such as header and side bar menu.
Font Component	https://fonts.googleapis.com/css?Family=Raleway	Used to provide the font design for this application.
PHP Library	bmewburn.vscode-intelephense-client	Used to provide the PHP library to develop this application.
PHP Debug	felixfbecker.php-debug	Used to help the developer debug the program.

Report Component	//cdn.datatables.net/1.10.25/js/ jquery.dataTables.min.js	Used to provide the library to create an external report for user.
Front-End Function	//code.jquery.com/jquery-3.5.1.js	Used to provide Jqeury library to build a functional UI.
phpMyAdmin	localhost	A SQL database that stores all the data of this application.
gstatic	google.visualization.LineChart google.visualization.arraytoDataTable google.visualization.DataTable google.visualization.PieChart	Used to provide the library of drawing graph and chart
composer	vendor	Used to provide a lot of libraries to connect to Firebase for authentication part.

After configuring the project environment in Visual Studio Code, the application will be able to connect to the phpMyAdmin database and build the back-end functionalities. First of all, a new database will be created in phpMyAdmin. After that, the related table will be made in this database and link all the relationships between the tables. Next, to add the external link to the Visual Studio Code and using the PHP library to connect the database. The last step was creating a project in the firebase in terms of the authentication.

5.2 Setting and Configuration

In this section, the project installation and setting will be discussed in order to launch this application. By default, the user need to download the application folder to use this application. However, in order to successfully launch this application, there have some steps that user need to proceed. Figure 5.2.1 shows this application's folder after user has been downloaded it.

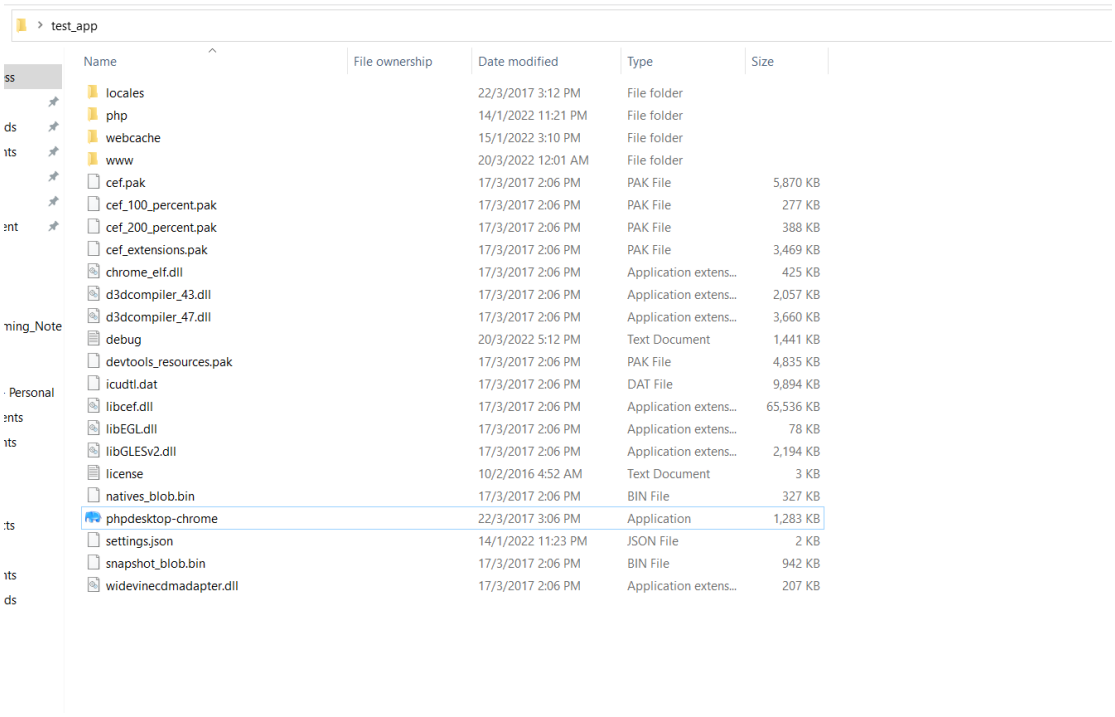


Figure 5.2.1 Application Folder

Firstly, the user has to install XAMPP to host a local database to store the application data. In this XAMPP software, it did provide Apache server and phpMyAdmin database. If the user did not open the server and database, this application would not provide the function correctly. Figure 5.2.2 shows the output for opening the server and database.

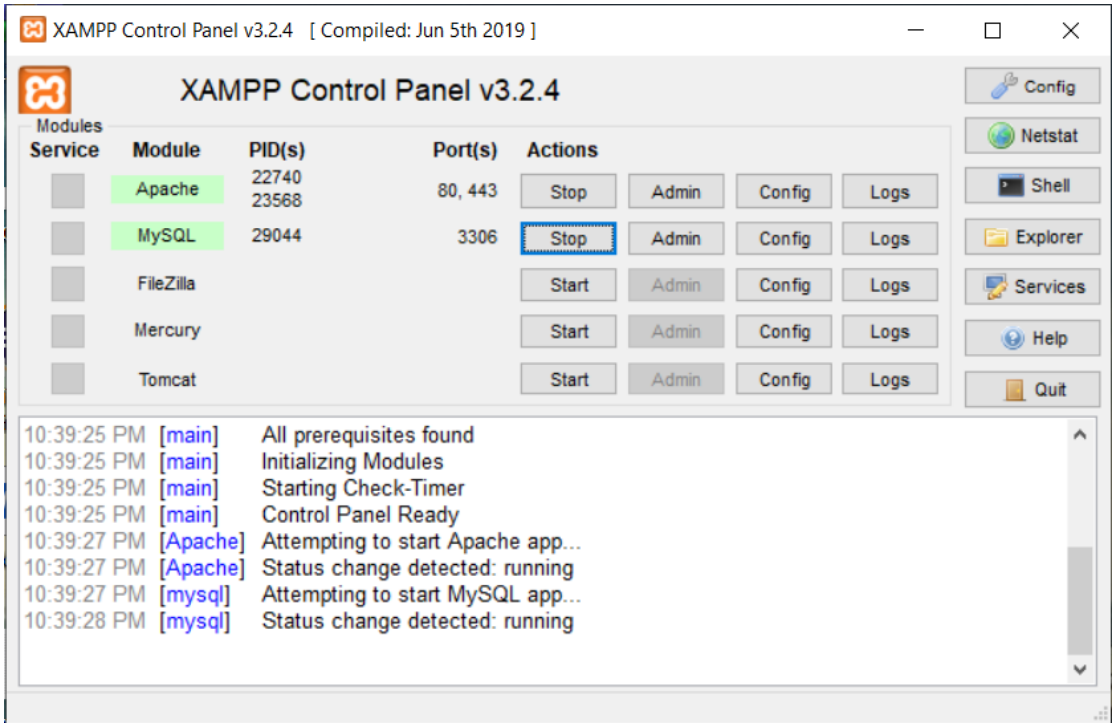


Figure 5.2.2 Starting Server for Local

After that, since this application is a desktop application, therefore, the application folder required some missing dependencies for building the app. Hence, the user has to go to find the XAMPP file location and look for “php” folder and copy it to replace this application’s “php” folder so that the application can be launched without any problems. Figure 5.2.3 shows the “php” folder for XAMPP folder.

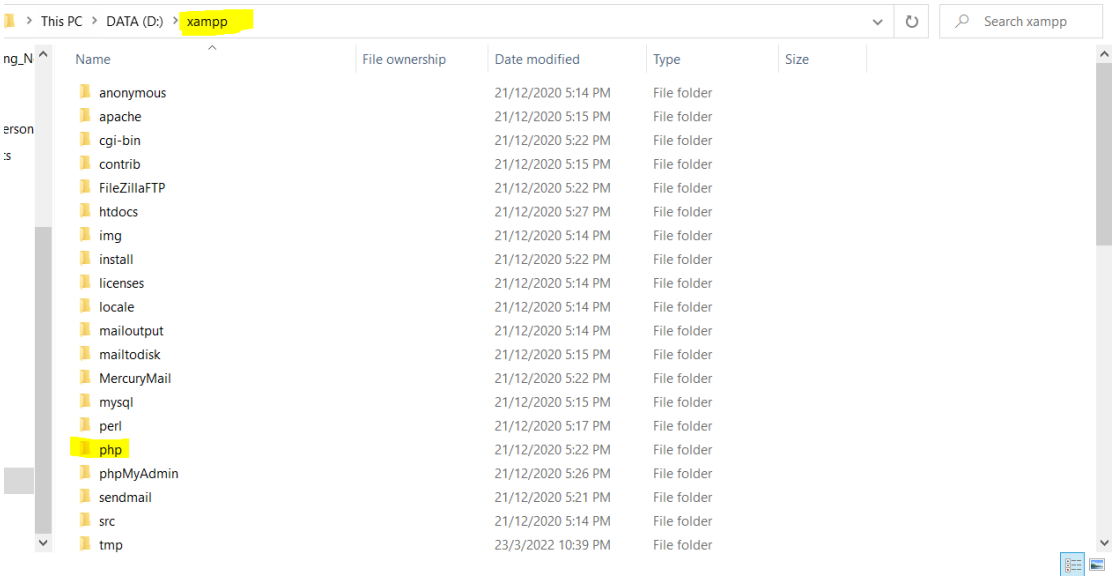


Figure 5.2.3 “php” folder in XAMPP folder

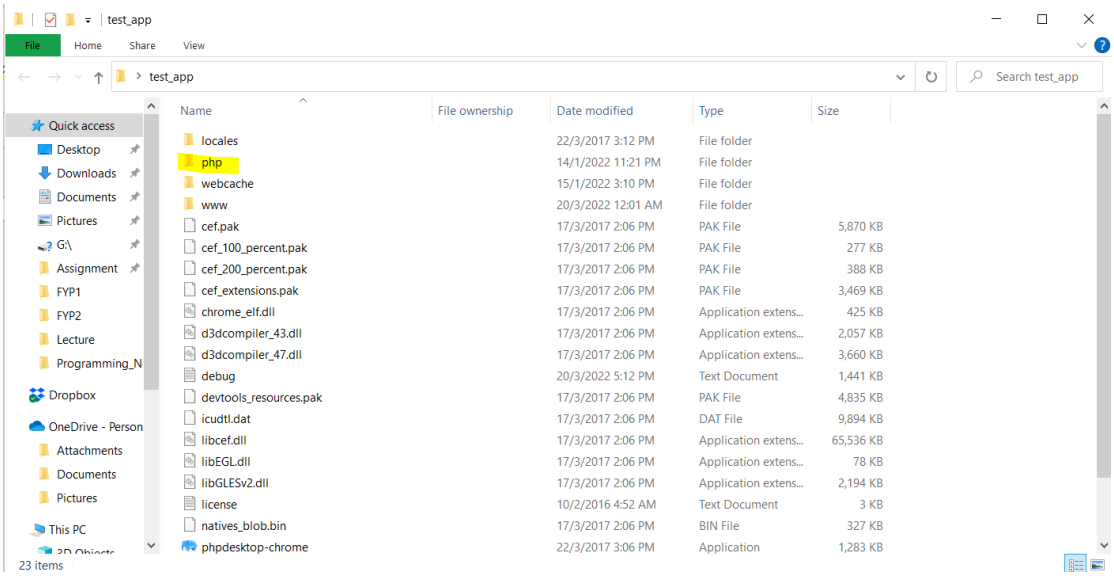


Figure 5.2.4 “php” folder in this application

After the user has replaced the “php” folder, the user can just click on the .exe file to launch this application and start managing their personal finance.

5.4 System Operation

In this section, the developed application functions and process will be discussed in detail with some screenshot pictures of this application. In the following section, the system operation will be separated by module to discuss.

5.4.1 User Management

In Project II, the UI and general functionalities for User management were completed. First, the application will display a login page to the user to perform the authentication. The UI of the login page is shown below in Figure 5.4.1.

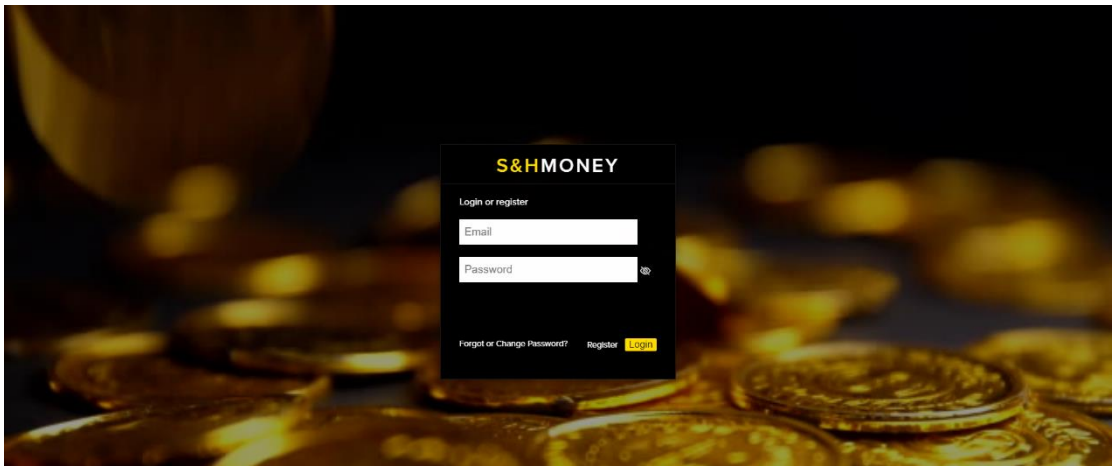


Figure 5.4.1 Login Page

If the user had an user account in the figure above, the user could directly use the user account and login. The back-end PHP code handles the login system with the database. The user has to key in all the input fields to proceed to the login system. Validation will be implemented on the client-side for the empty input. If there is any empty input, the system will focus on the related input field to the user and will not proceed to the authentication. After that, basic validation for the email input will also be implemented to avoid the user key in an invalid email format. If the email input received invalid format, the system would focus on the input to inform the user to key in again. As for the password field, an “eye” icon allows the user to see the password that the user has filled in. When the user clicks on the “eye” icon, the system will change the password visibility and after that, if the user clicks one more time, the password will be shown in a secure mode.

Once all the inputs getting the correct format, the user can click on the *Login* button to do the authentication from the system. After that, the application will check whether the information exists in the system with the email and password. If the password and email existed in the system, the application would switch the login page to the homepage of this application. Furthermore, the validation of this login system can be divided into two situations. The first situation is when the user key in the correct email but the wrong password. In this situation, the system will return "Wrong Password, Please Try Again" message to the user, which means that the user has entered the wrong password only.

On the other hand, when the user entered an email that has not been registered in this application, the system will return an "Invalid Email. Please Try Other" message to the user, which means that the user has entered the wrong email. Hence, this is the authentication step of this login system. Figures 5.4.2 and 5.4.3 below show the error message of the login system.

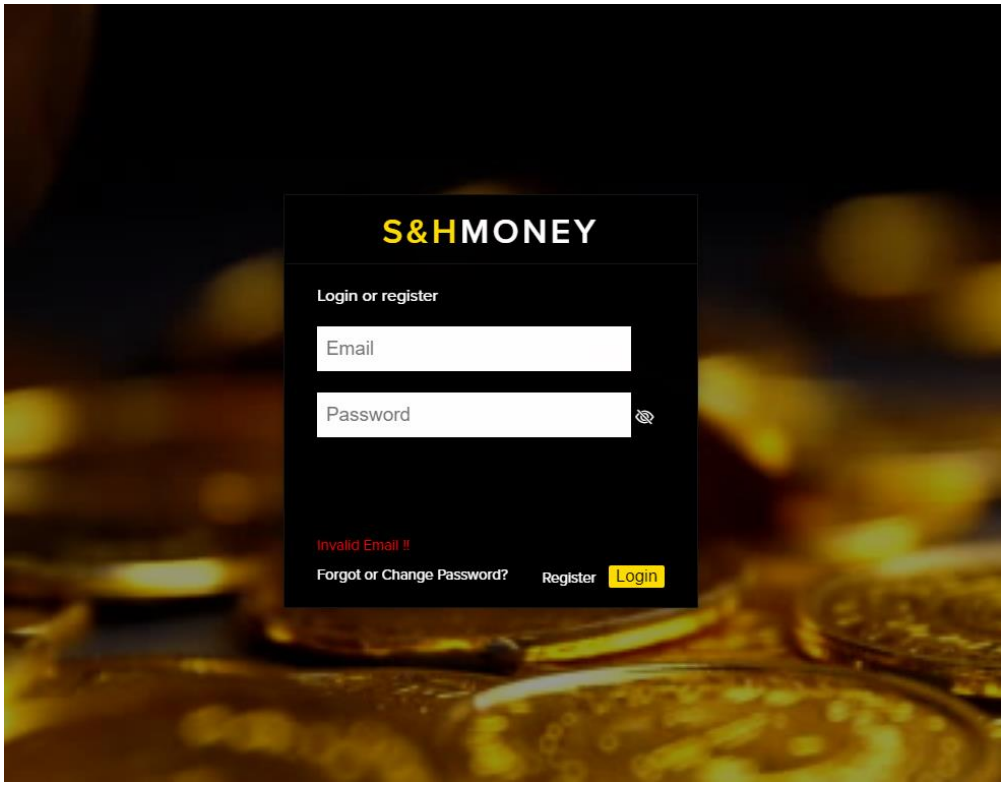


Figure 5.4.2 Wrong Email Validation on Login Page

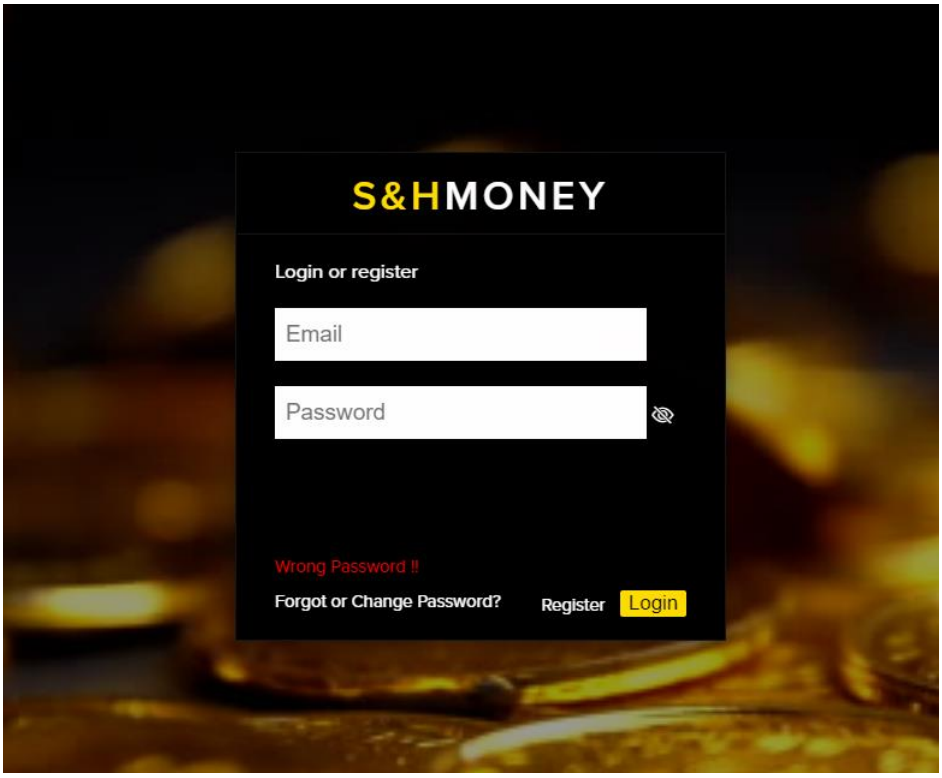


Figure 5.4.3 Wrong Password Validation on Login Page

However, if the user does not have any user account on this application, the user can click on the *register* button to register an account. Figure 5.4.4 shows the Register page.

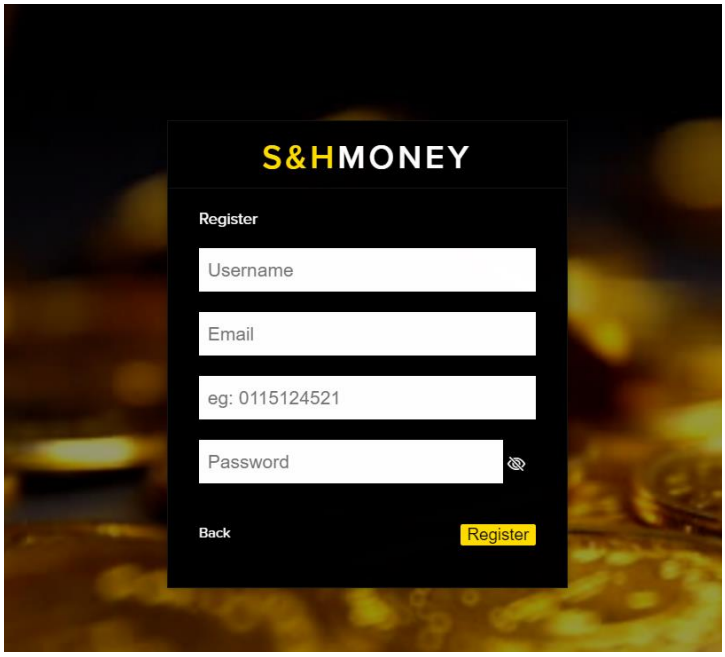


Figure 5.4.4 Register Page

On the Register Page, users can create a new user account by providing their username, email, phone number and password for the user account. First, basic validation such as an error on empty input will be implemented. Next, if the user entered an email that is registered in this application, the system will stop the registration process and prompt an error message to the user. Not only that, the system will also implement the validation on the phone number. The validation will be the same as the email. If these two input fields cause duplicated records in the database, the system will not allow the user to add the user account. In addition, the length of the phone number must be greater than or equal to 10 otherwise the system will return an error message to the user. Similarly, the password field will be the same as the login page. Figures 5.4.5 and 5.4.6 below show the error message of this registration process.

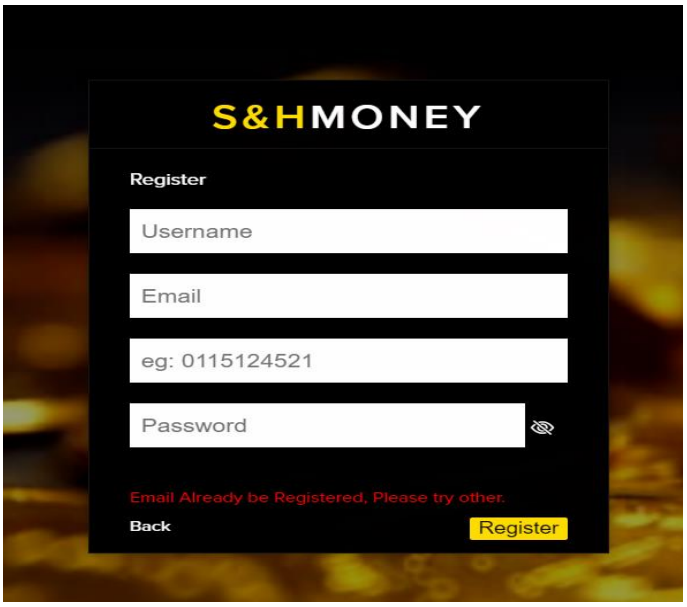


Figure 5.4.5 Duplication Email in the System

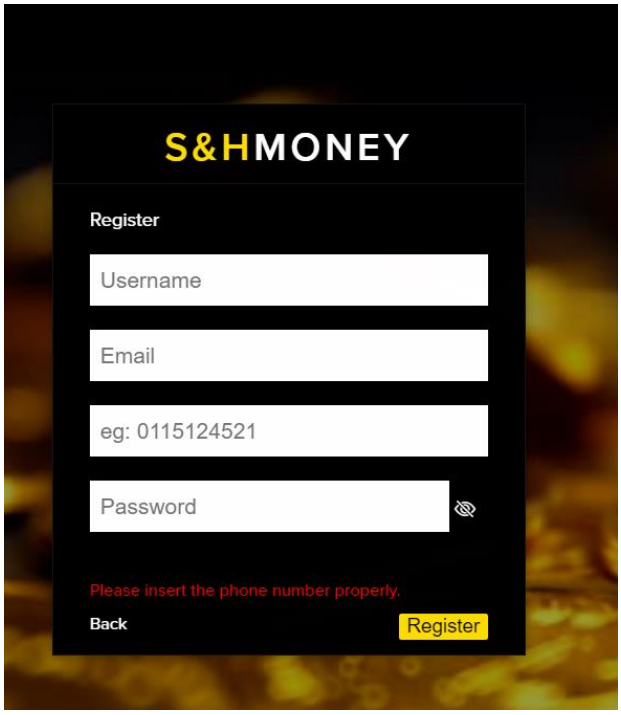


Figure 5.4.6 Invalid Phone Number

Once the user has entered every field in the registration form, the user can click on the *Register* button to send a request to perform system validation such as duplication records. If there are not any duplicated records, the system will create this user account for this application. However, if the user wants to change the password, the user can click on the *Change Password* button on the login page to proceed to this action. The UI of the reset password page is shown below in figure 5.4.7.

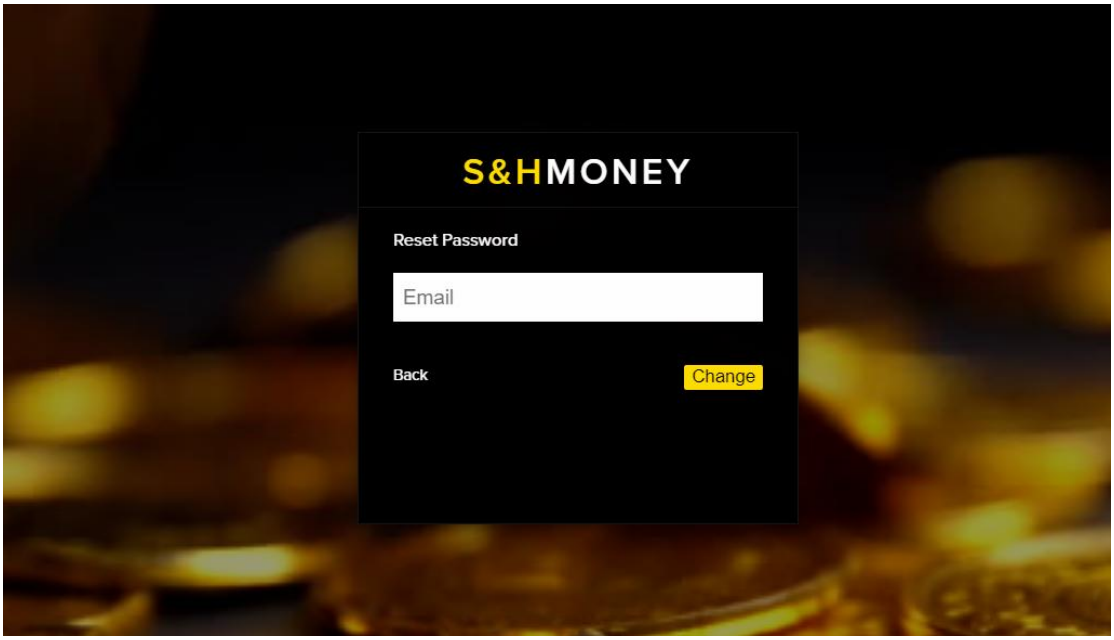


Figure 5.4.7 Reset Password Page

On this page, the user can change their existing password by providing their email. Basic validation of the empty input field will be implemented in this form. Furthermore, if the user entered a non-existing email in this application, the process of changing the password will be terminated by the system and return an error message. The password field will be the same as the email input field as well. Figure 5.4.8 shows the error message when the user filled in a non-existing email.

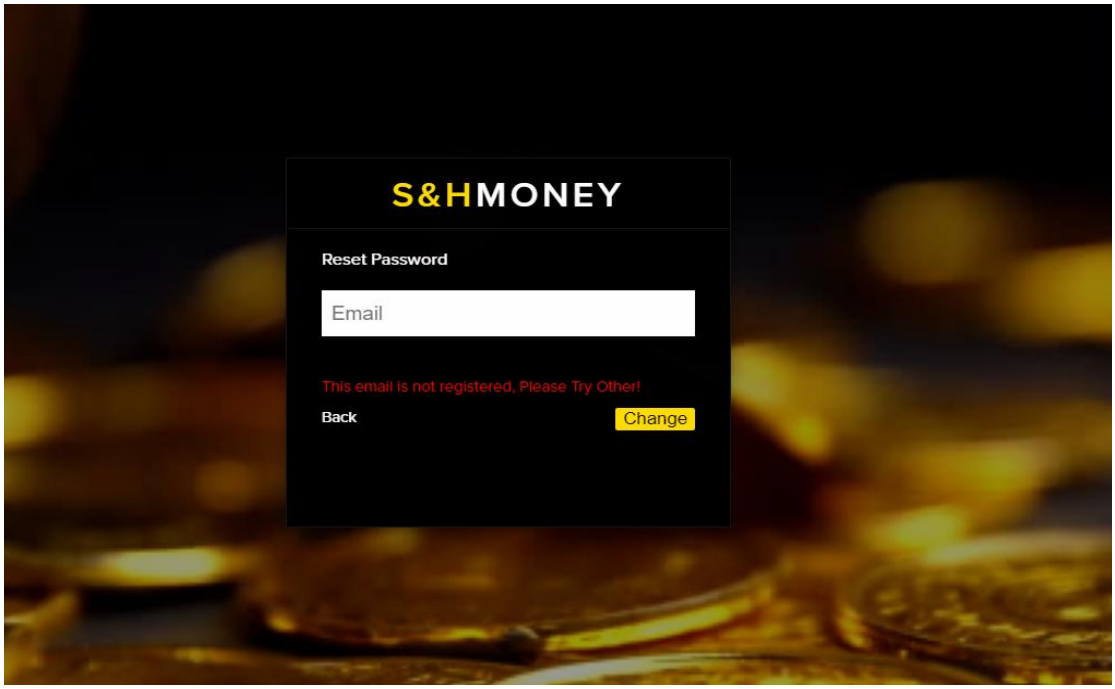


Figure 5.4.8 Non-existed Email

Once the email is correct, the system will proceed to the email verification. By sending a reset password link to the user email. The user just need to go to their email an click the link to reset their password to new password. The email verification content is shown in figure 5.4.9.

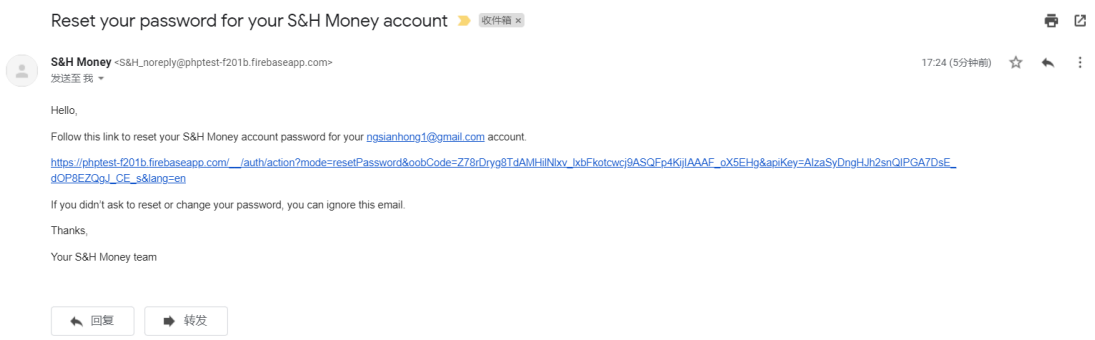


Figure 5.4.9 Email Reset Password

After that, when the user clicks on the reset password link provided by this application, the user will be prompted to a new page in web browser to reset their password. Figure 5.4.10 shows the output when the user successfully reset the password by using email verification.

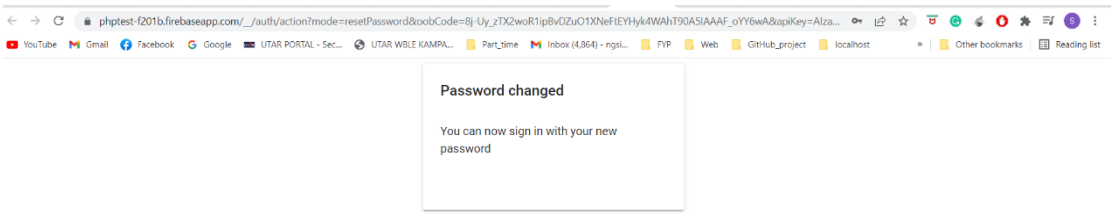


Figure 5.4.10 Success of Resetting the password

Once everything matches the related fields, the system will update the password to the backend system (database and Firebase).

After that, the user can proceed to the login and use the application. The edit user information function will be on the home page. The UI of this function is shown in figure 5.4.11.

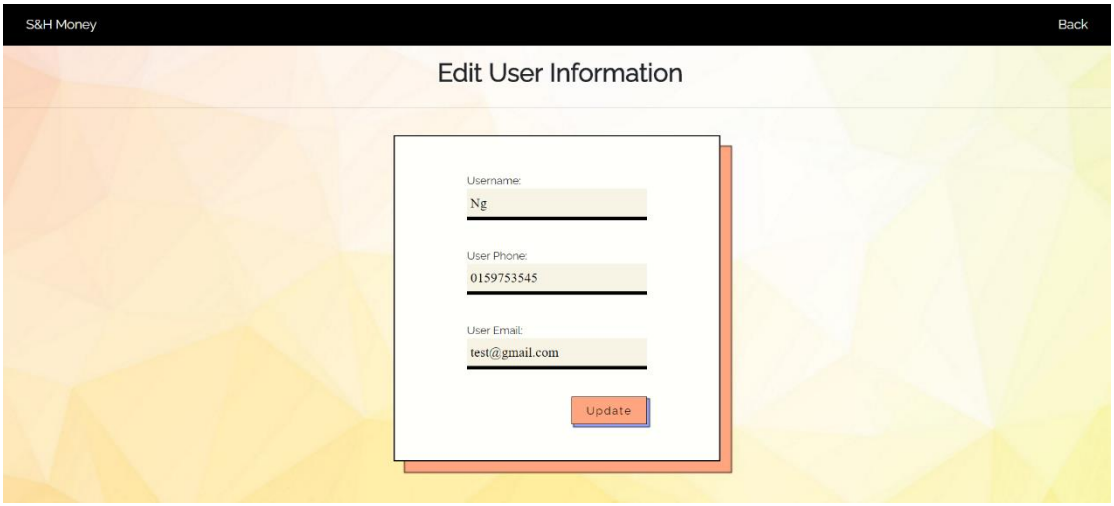


Figure 5.4.11 Update User Account Information

In the figure above, the system will show the user account information once the user clicks on the related button. The validation implemented in the registration will also be used in this update form such as email validation, phone number validation, and duplication checking. The user can modify the information and click on the *Update* button to send the request to the system. Once all the validation has passed, the system will display a success message to the user. The UI of the display success message UI is shown in figure 5.4.12.

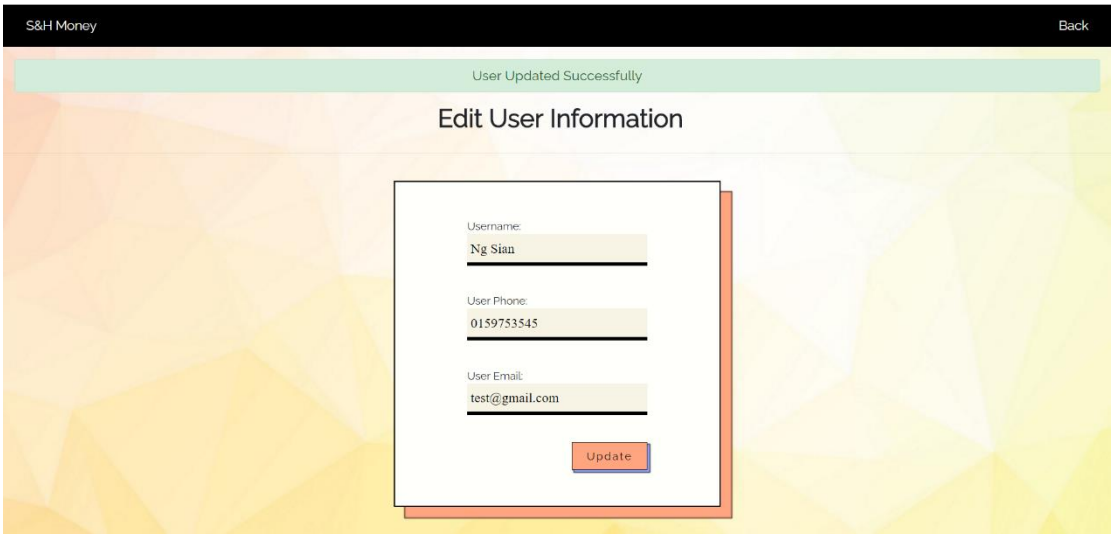


Figure 5.4.12 Success Message of Update User Information

5.4.2 Account Management

The UI and general functionalities for Account Management were completed. In the following section, the preliminary work of this module is discussed.

Once the user has logged in successfully into the application, the application will display the homepage to prompt the user to create an account first and then only manage their personal finance. The homepage UI design is shown in below figure 5.4.13.

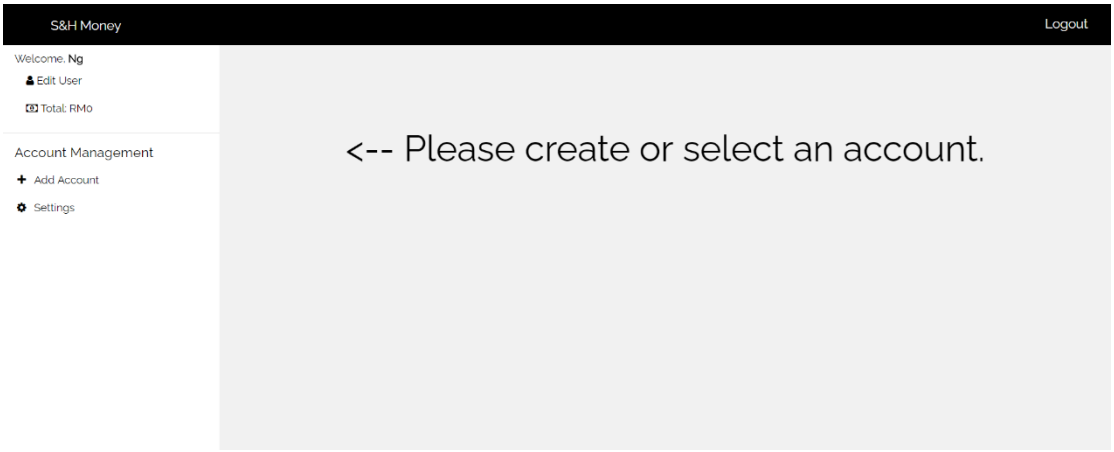


Figure 5.4.13 Home page

After that, the user can proceed to add an account. However, a new user definitely will have zero accounts. Hence, the user has to click *Add Account* button in the side menu to add an account or multiple accounts. The UI of add account page is shown in figure 5.4.14.

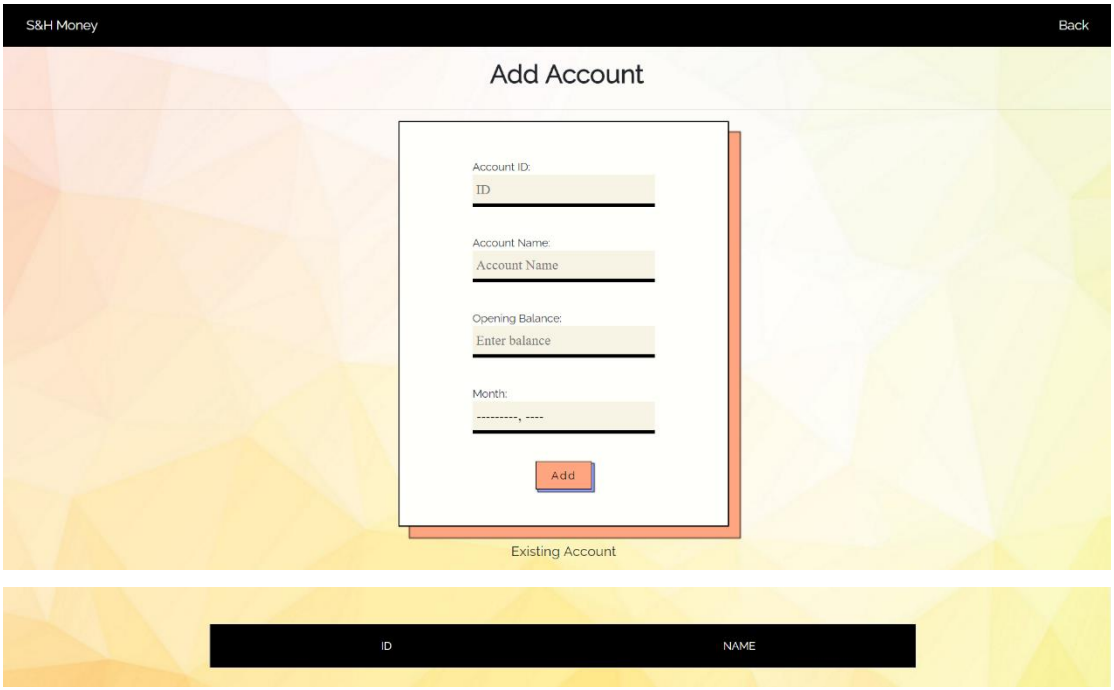


Figure 5.4.14 Add Account Page

In the figure above, the user can add an account by providing account ID, account name, opening balance, and the month of the balance. Empty inputs are strictly not allowed in all the forms in this application. Since that is impossible that a user can have multiple duplicate accounts with the same account ID. Therefore, if the system detects two same account IDs in one user, the system will prompt an error message and stop the add account process. Another validation mentioned in chapter 3 has been implemented, such as the input number in the opening balance. Once all the input has passed the validation, the system will display a success message to the user and show it in the existing account table. The purpose of this table is to let the user refer to which account exists in this application so that the user can avoid adding duplicate records. The UI of the success message is shown in figure 5.4.15.

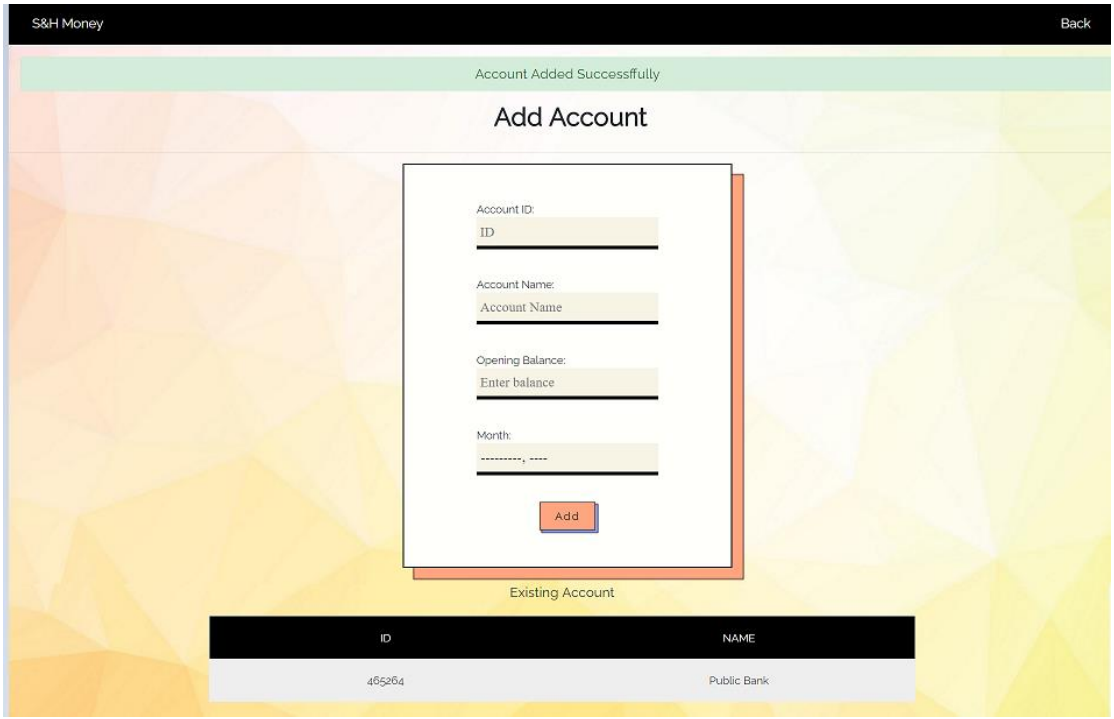


Figure 5.4.15 Success Message of Adding an Account

After that, the user can go back to the homepage by clicking on the *Back* button in the top right corner. The UI design of the home page after adding an account is shown in figure 5.4.16.

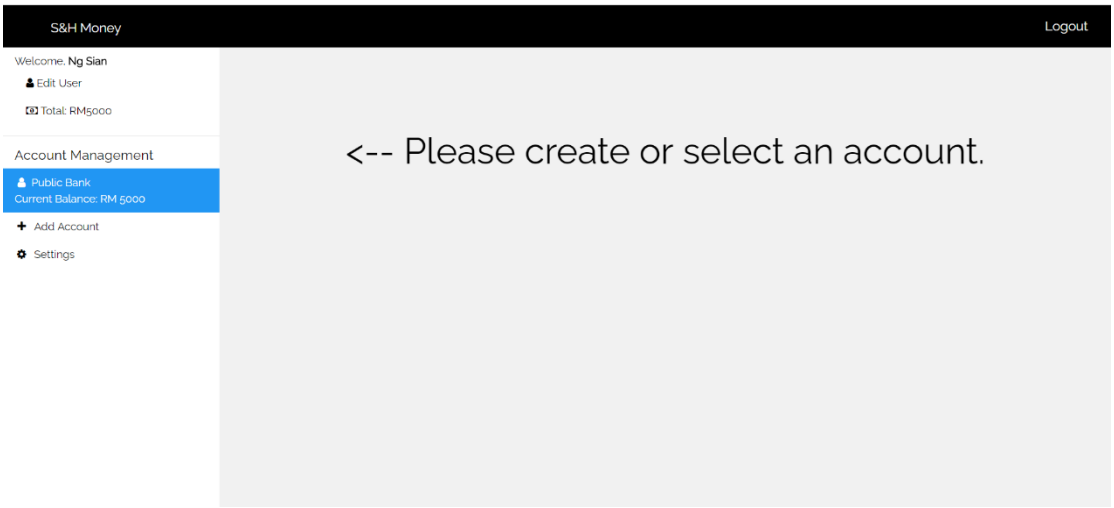


Figure 5.4.16 Homepage with account

In the figure above, the side menu has shown the account to let the user to select. The user can add the account as many as possible and different accounts will store different income, expenses, and personal budget things. In addition, a “total” is used to

display how much money the user has in the side menu. The calculation of this total is adding all the account balances together. Moreover, if the user wants to manage all the accounts information, the user can click on the *Setting* button to delete and update the existing account. The UI design for the setting page is illustrated in figure 5.4.17.

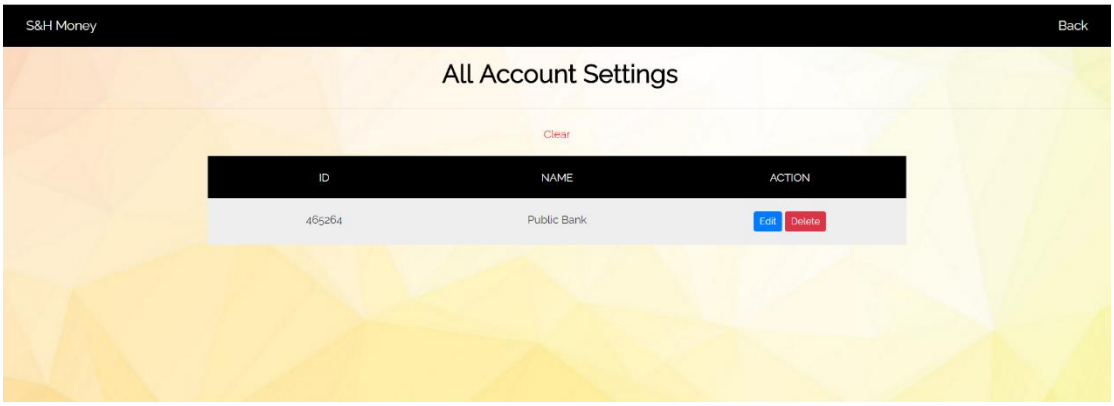


Figure 5.4.17 Account Settings Page

In the figure above, a table will show to let the user manage all accounts information. When the user clicks on the *Edit* button, the system switches to the update page to update the record. The UI of the update account page is shown in figure 5.4.18.

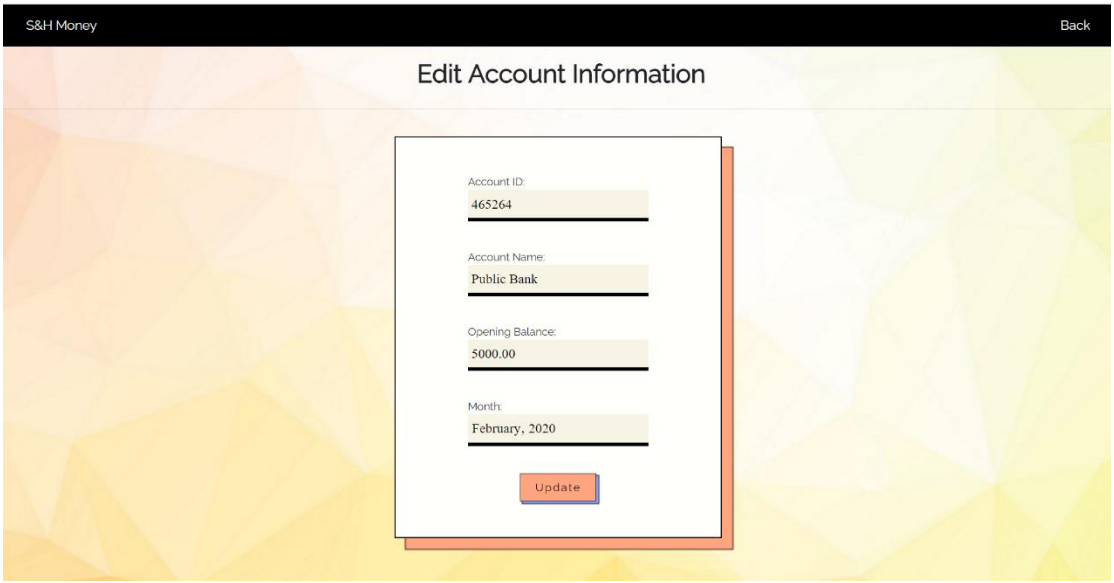


Figure 5.4.18 Update Account Page

Upon clicking the *edit* button on the account setting page, the system will show the record in a form. The user can update the account information by modifying the form and click on the *Update* button. For this update form, all the validation will be the

same as the add account form, such as checking duplicated account ID. If the record that the user wanted to update is not any problem, the system will display a success message to inform the user. The UI of the success message in this update account is illustrated in figure 5.4.19.

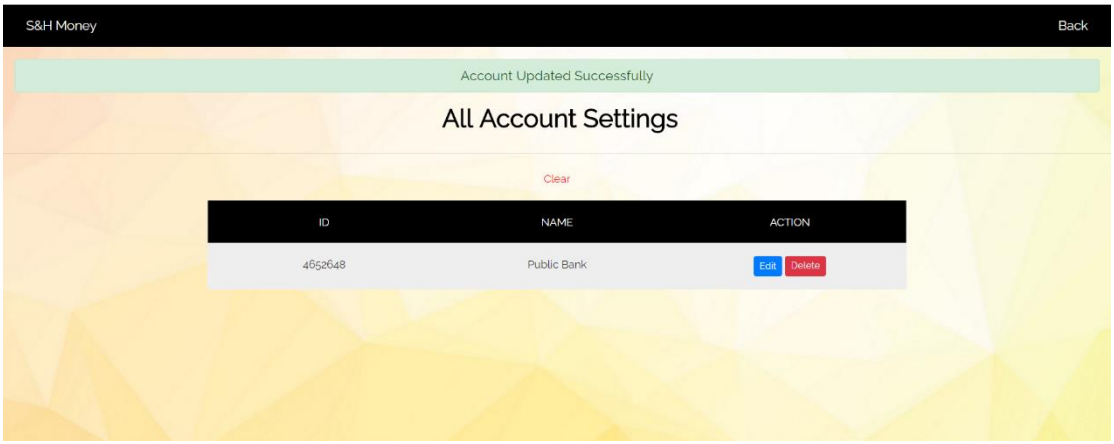


Figure 5.4.19 Success Message of Updating an Account

Next, the user can perform the delete function to delete a particular account by clicking on the *delete* button. After clicking on the *delete* button, the system prompts a message to inform the user the record has been deleted. The example of a notification is shown in figure 5.4.20.

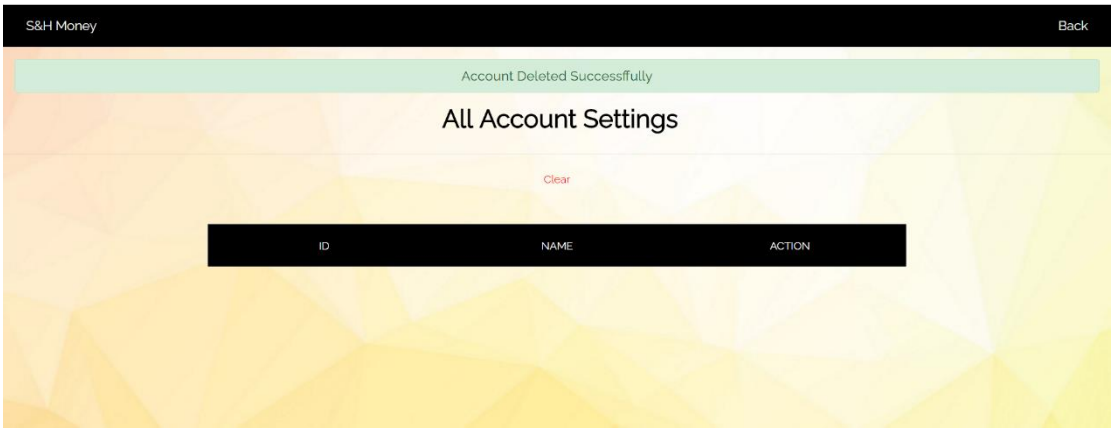


Figure 5.4.20 Delete Account Message

Furthermore, back to figure 5.4.16, the user can select an account to start personal finance. The UI design is shown in figure 5.4.21.

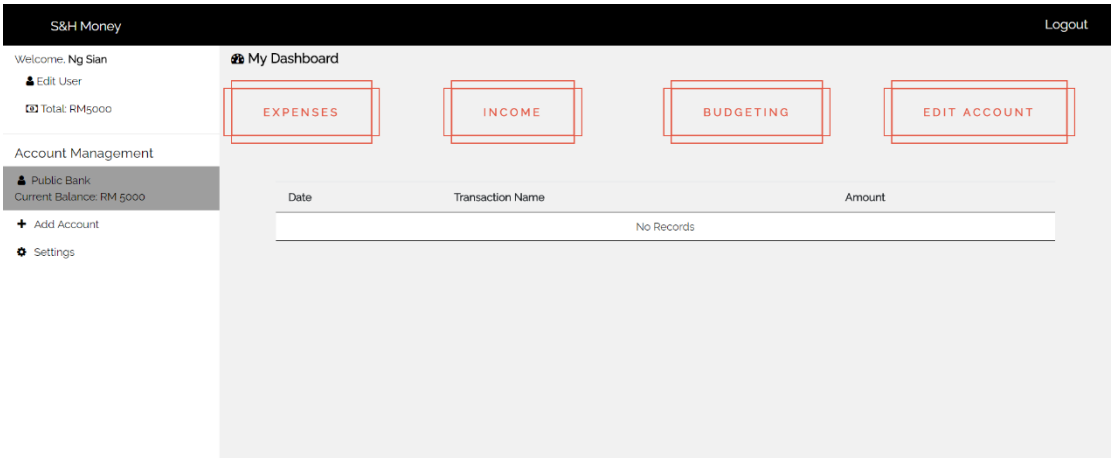


Figure 5.4.21 Homepage with Dashboard

In the figure above, there are four navigation buttons in the dashboard. Each of the buttons will navigate the user to the different pages once the user clicks on it. The *Edit Account* button is the same as the edit function inside the account setting page.

5.4.3 Income and Expenses Management

The UI and general functionalities for Income and Expense Management were completed. The UI and functionalities for income and expenses are the same. Once the user clicks on the Expenses button in the dashboard, the system will switch the user to the expenses home page to manage the expenses. The UI design for the expenses menu is shown in figure 5.4.22.

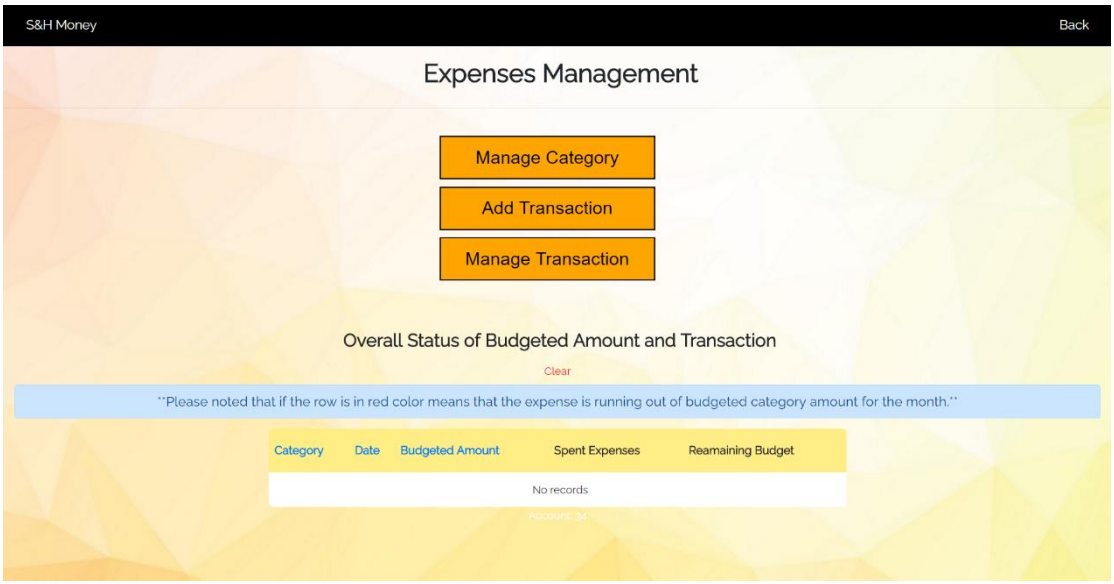


Figure 5.4.22 Expenses Menu Interface

In the figure above, the user can navigate the page based on the option given by the menu. After that, below the menu, a table allows the user to easily track the budget plan with the existing expenses. Upon clicking the *Manage Category* button, the system will switch the page to the manage category page. The user interface of the manage category page is shown below in figure 5.4.23.

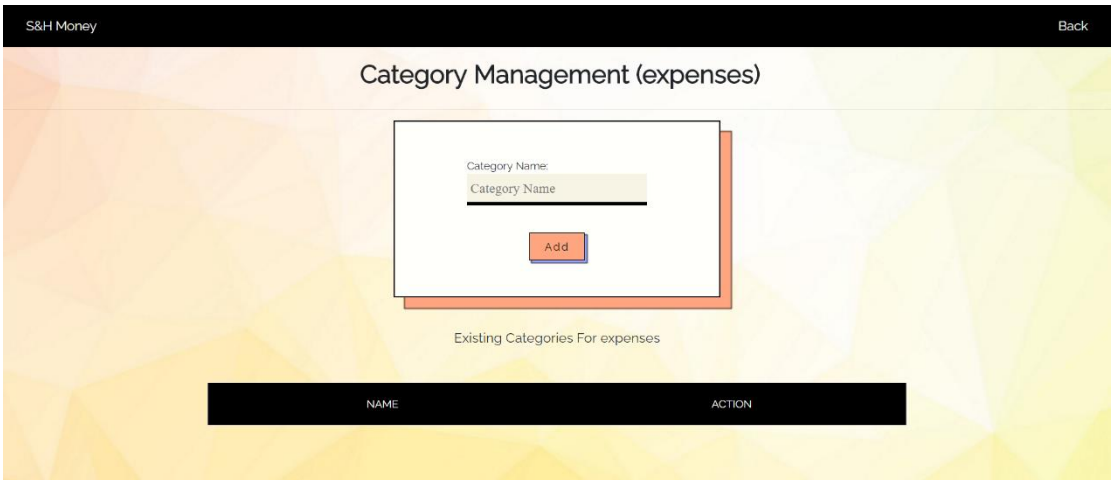


Figure 5.4.23 Category Management Page for Expenses

The figure above shows that the category page is applicable for both income and expenses. First, the user has to add a new category first then only can add the expenses record. Then, the user can add the new category by providing the category name. The only validation in this form is a duplication record. If the user adds the category that is existing in the table below, the system will not allow the user to add it. Once the new category record does not cause any error, the system will display a success message to inform the user. The notification UI is shown in figure 5.4.24.

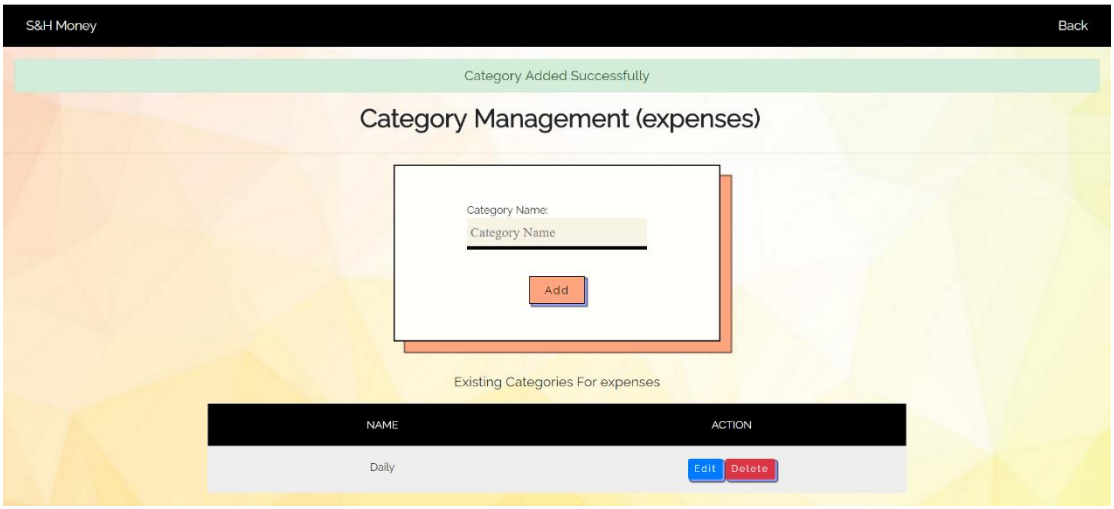


Figure 5.4.24 Success Message of Adding New Category

After that, the user is allowed to update and delete category functions in the table. Once the user clicks on *edit* button in the table, the system will bring the category name to the form to let user update. The UI of the update category is shown in figure 5.4.25.

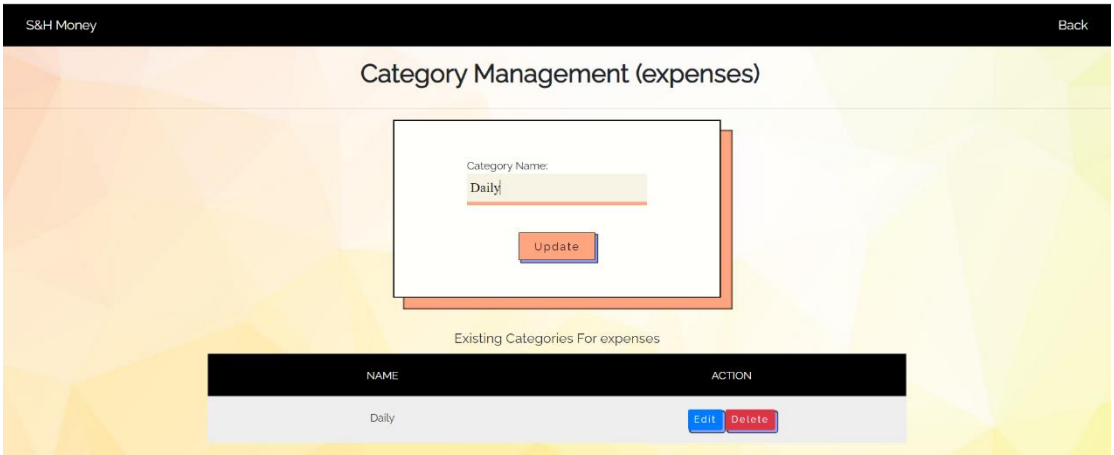


Figure 5.4.25 Update Category Record

In the figure above, the validation will be the same as the add new category. The user can update the category by modifying the form. If the record that the user wanted to update does not cause any error, the system will update the category. Once successfully updated, the system will prompt a success message to the user. The notification of the success message is shown in figure 5.4.26.

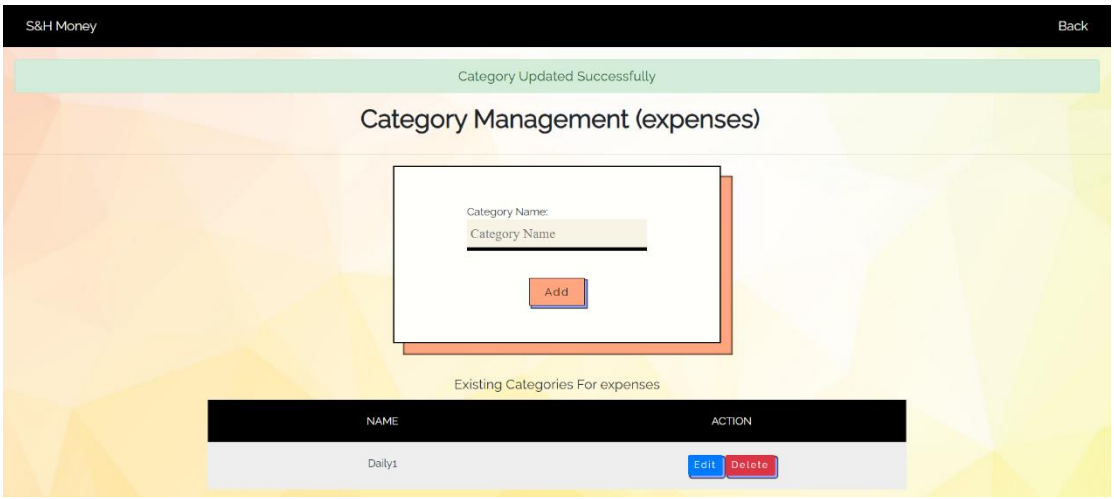


Figure 5.4.26 Success Message of Updating a Category

Next, the user can perform the delete function to delete a particular category by clicking on the *delete* button. After clicking on the *delete* button, the system will also prompt a message to inform the user the record has been deleted. The example of a notification is shown in figure 5.4.27.

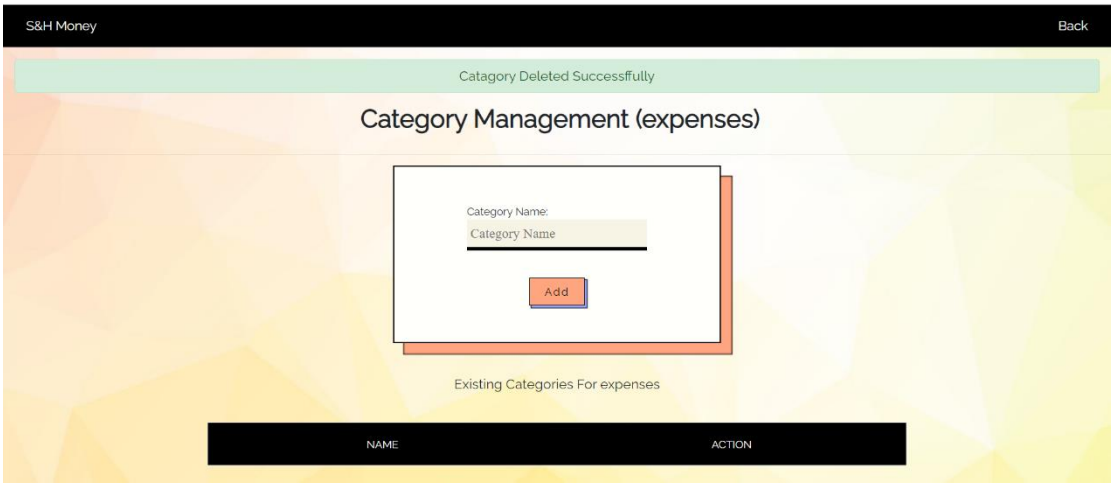


Figure 5.4.27 Success Message of Deleting a Category

After the user has done the category management, the user can go to add the new expenses. The user interface of add new expenses is shown below in figure 5.4.28.

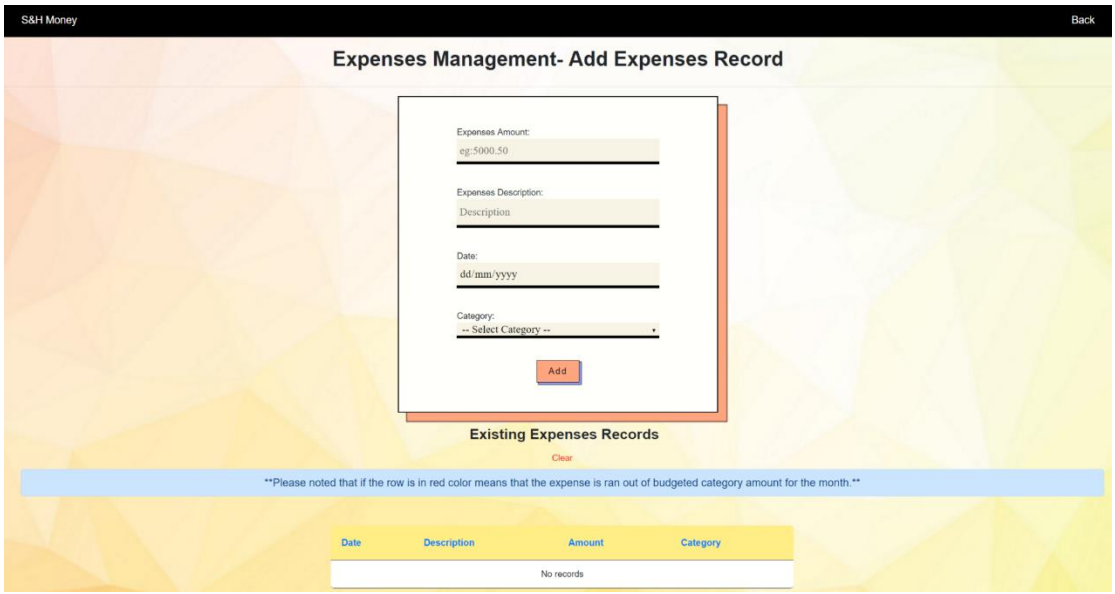


Figure 5.4.28 Add Expenses Page

In the figure above, the user can add a new expense by filling in the required information into the form. The user must fill all the input fields in the textboxes or except for the category fields. This field will use a drop-down list to show all the existing categories. The example of the category drop-down list is shown in figure 5.4.29.

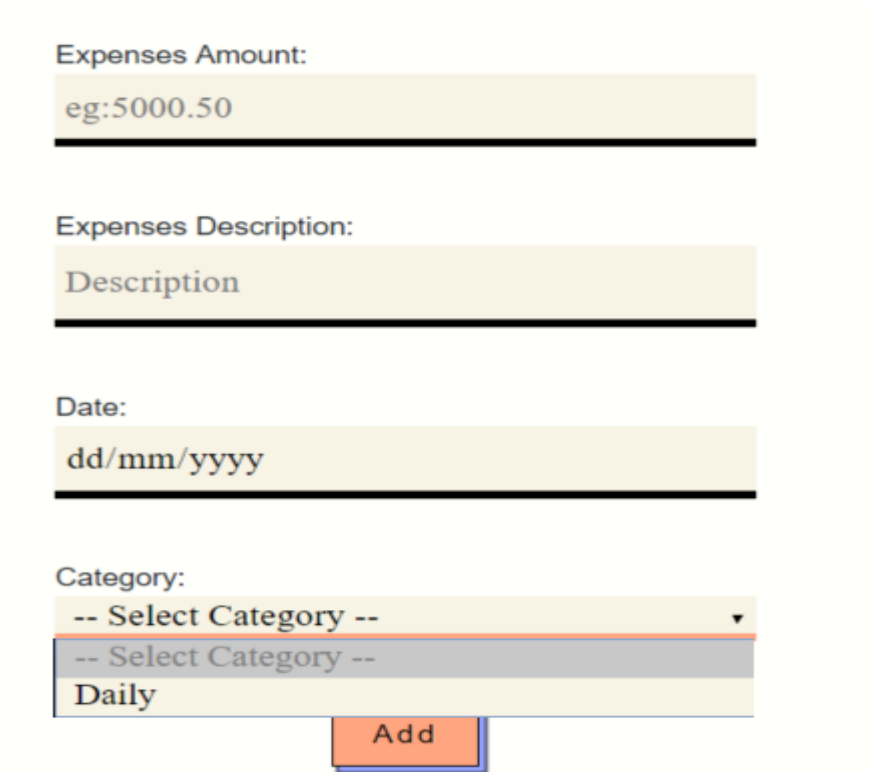


Figure 5.4.29 Category Drop-Down List

In the figure above, the user can choose the category accordingly. Once all the input fields have been entered, the user can check the existence of the new record by referring to the table. Once the user clicks on *Add* button, a validation will be implemented to check whether the new expense exists in the table. The checking condition is based on the expense date and description to determine whether it causes any duplications. If the new record does not cause errors, the system will show a success message to inform the user. The example of the success message of adding new expense is shown in figure 5.4.30.

Figure 5.4.30 Success Message of Adding New Expense

Next, the user can view the expenses details by clicking on the *Manage Transaction* button on the menu. The interface design for the list of transactions is shown in figure 5.4.31.

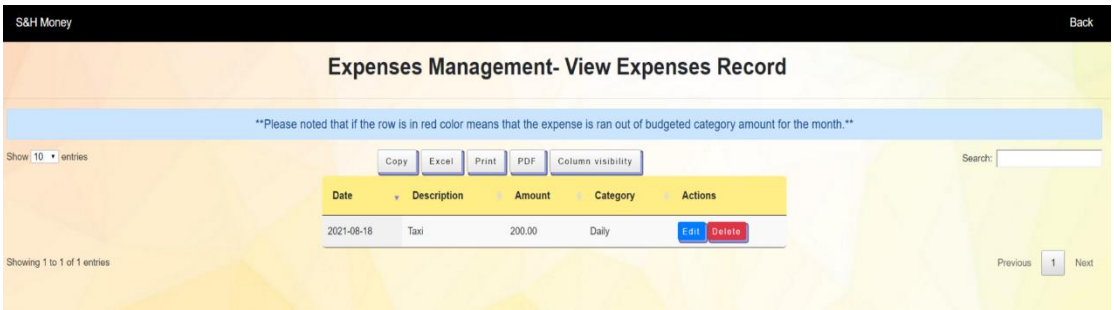


Figure 5.4.31 List of Expenses

In the figure above, the functionalities that users can perform in the list are pagination, dynamic searching, sorting and selecting records per page. In addition, if the row in the list has a red background, it means that the expenses have already run out of this category's budget. Furthermore, edit and delete functionalities can be performed in this expenses list. Once the user clicks on the *Edit* button, the system will switch to the edit expenses page. The UI design of the edit expenses page is shown in figure 5.4.32.

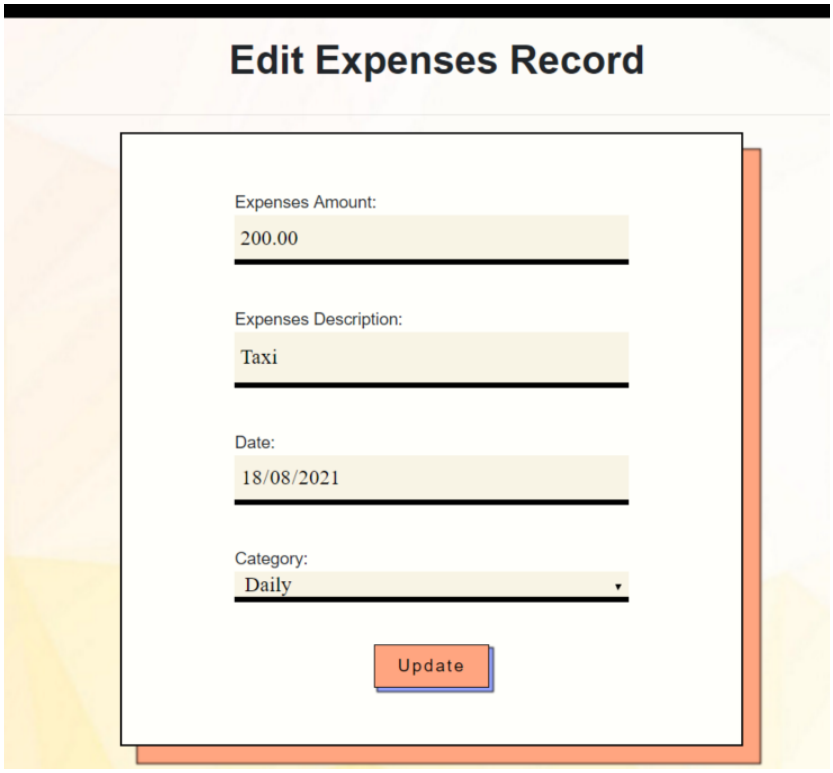


Figure 5.4.32 Edit Expense Page

In the figure above, the system will display the expense information to let users edit the expense. All the validation in this form will be the same as the add new expense form. Once the user has finished making the modifications, the user can click on the *Update* button to update it to the database. After that, the system will also show a success message to the user. The success message is shown in figure 5.4.33.

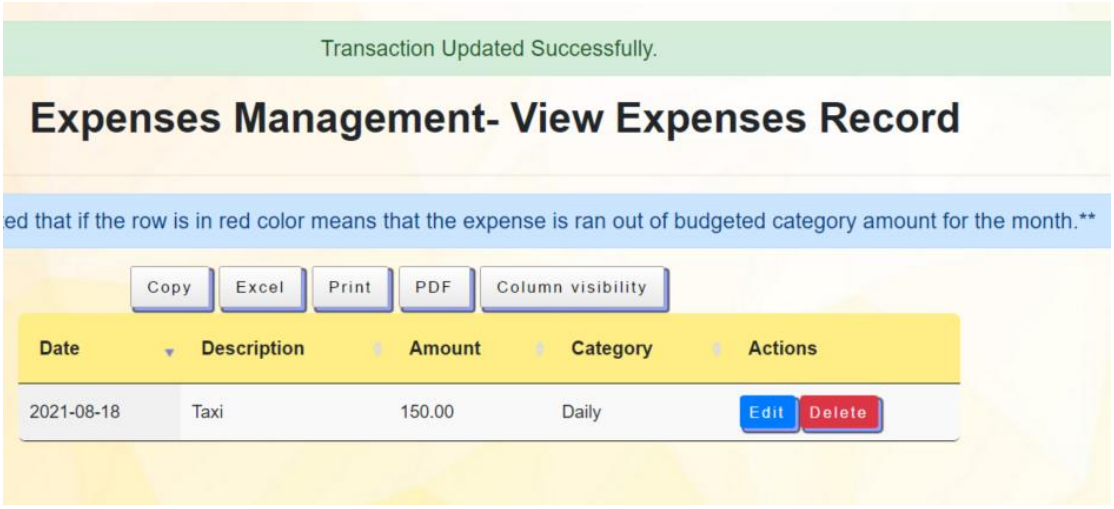


Figure 5.4.33 Success Message of Updating an expense

Furthermore, the user can perform the delete function to delete a particular expense by clicking on the *delete* button. After clicking on the *delete* button, the system will also prompt a message to inform the user the record has been deleted. The example of a notification is shown in figure 5.4.34.

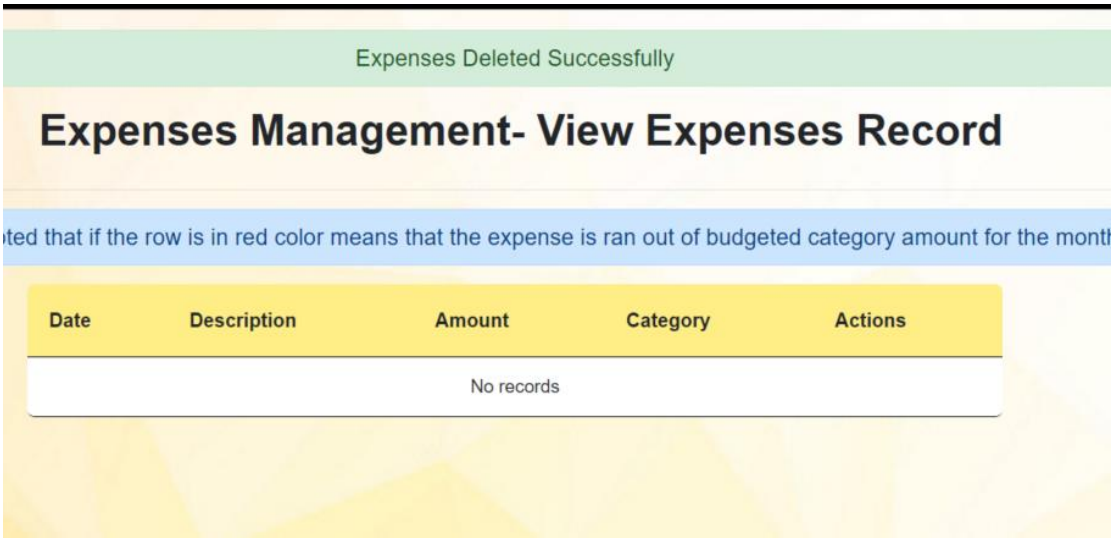


Figure 5.4.34 Success Message of Deleting an Expense

Lastly, as mentioned earlier, the UI and functionalities of income management will be totally the same as this expenses management, such as manage categories for income, add new income, delete income, edit income records, pagination, dynamic searching and so on.

5.4.4 Personal Budgeting

After managing the income and expense, the user can click on the *Budgeting* button in the dashboard, the system will switch the user to the budgeting home page to manage the budget plan. The UI design for the budgeting menu is shown in figure 5.4.35.

Figure 5.4.35 Budget Menu Interface

Upon clicking the *Account Budget Planning* button, the application will display a form to prompt the user to enter the account budget plan information. The interface design of the account budget plan activity is shown in figure 5.4.36.

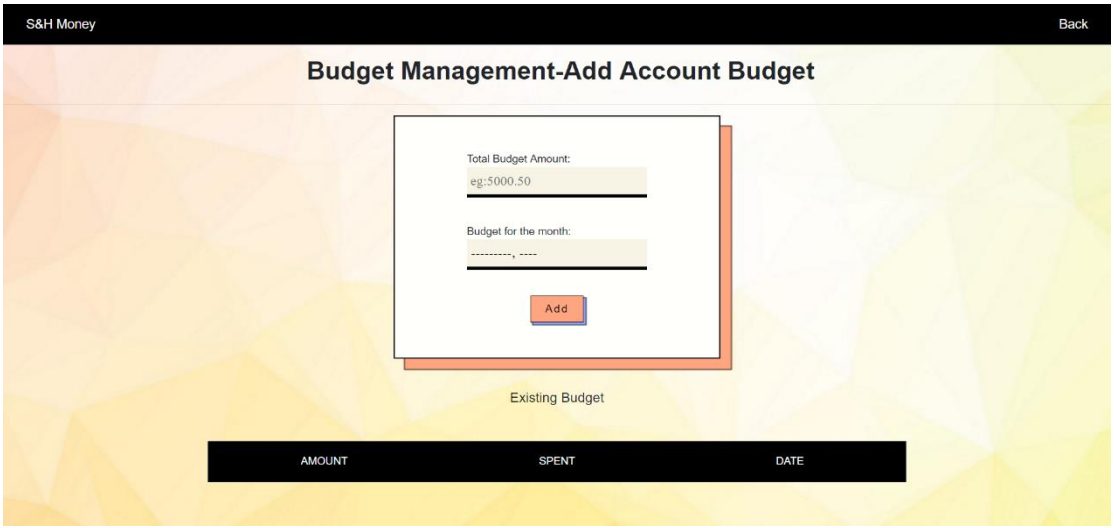


Figure 5.4.36 Add Account Budget Page

In the figure above, the user can add a new account budget plan by providing the total budget amount and the budget month. The basic validations such as empty input, duplicated record, and invalid format will be implemented in this form. Once the details that the user entered have been validated without any problem, the system will

display a success message to the user and store it in the table below. The success message of adding an account budget plan is shown in figure 5.4.37.

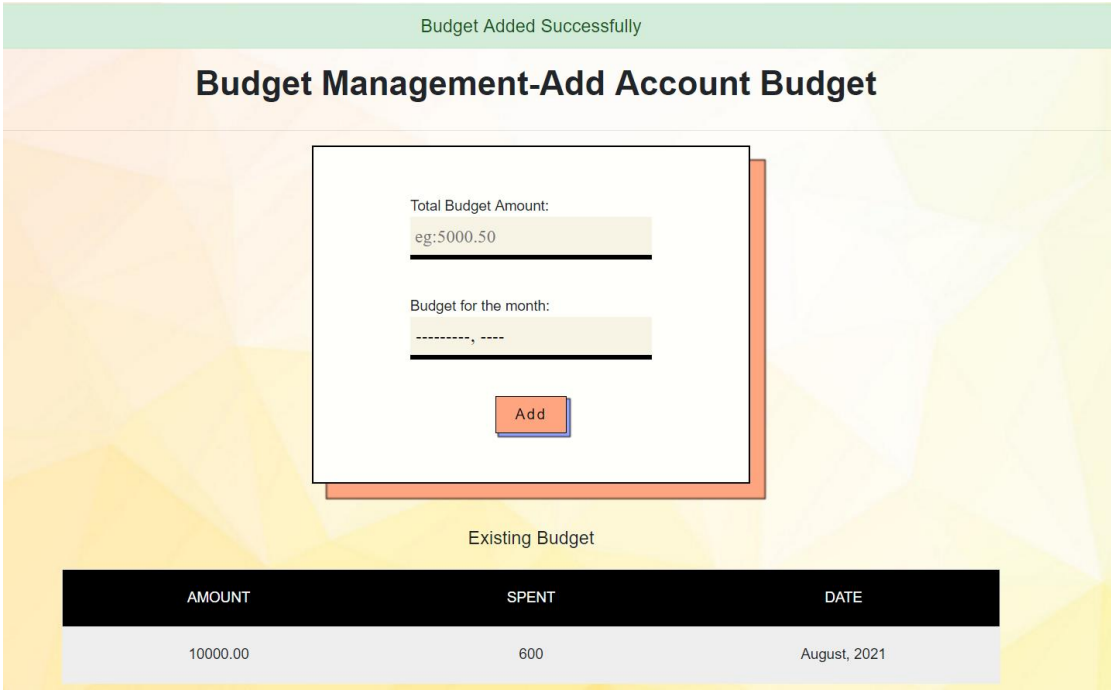


Figure 5.4.37 Success Message of Adding an Account Budget

In the figure above, once the new account budget plan has been added to this application, the system will immediately detect the existing expense for the month and show the status of this account budget plan in the table. However, if the account budget amount is lesser than the spent amount, the related row will be shown in red colour to warn the user the status. The UI of the account budget plan showed a warning message is shown in figure 5.4.38.

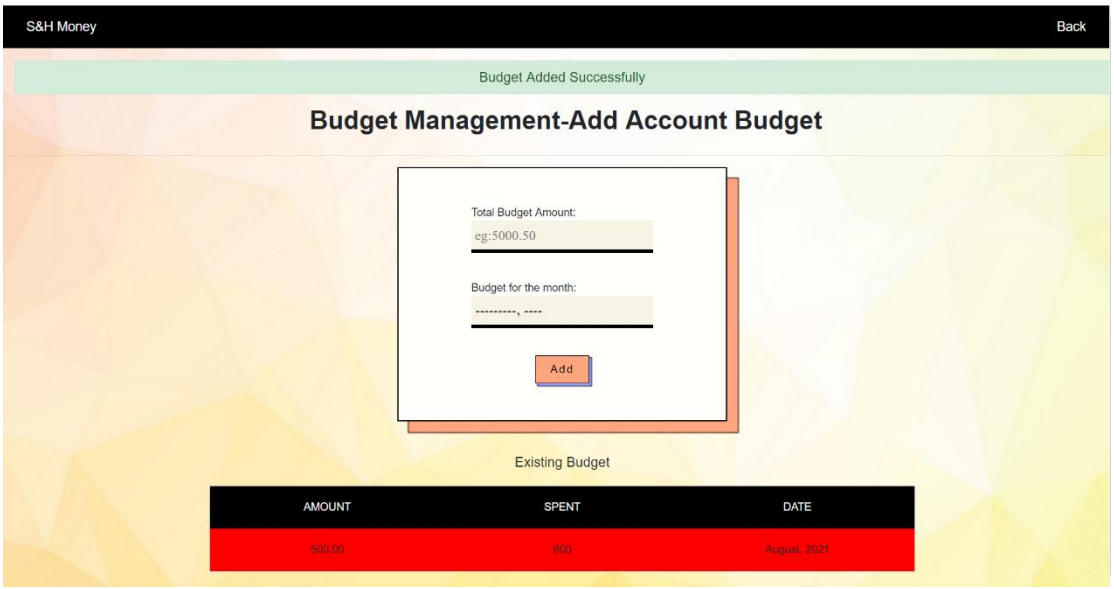


Figure 5.4.38 Warning Message for Account Budget Plan

Once the user has created an account budget plan, the user can add category-budget plans by clicking on the *Category Budget Plan* in the budget menu. The user will then be prompted to add, delete or update a category budget plan. The user interface design is as illustrated in figure 5.4.39

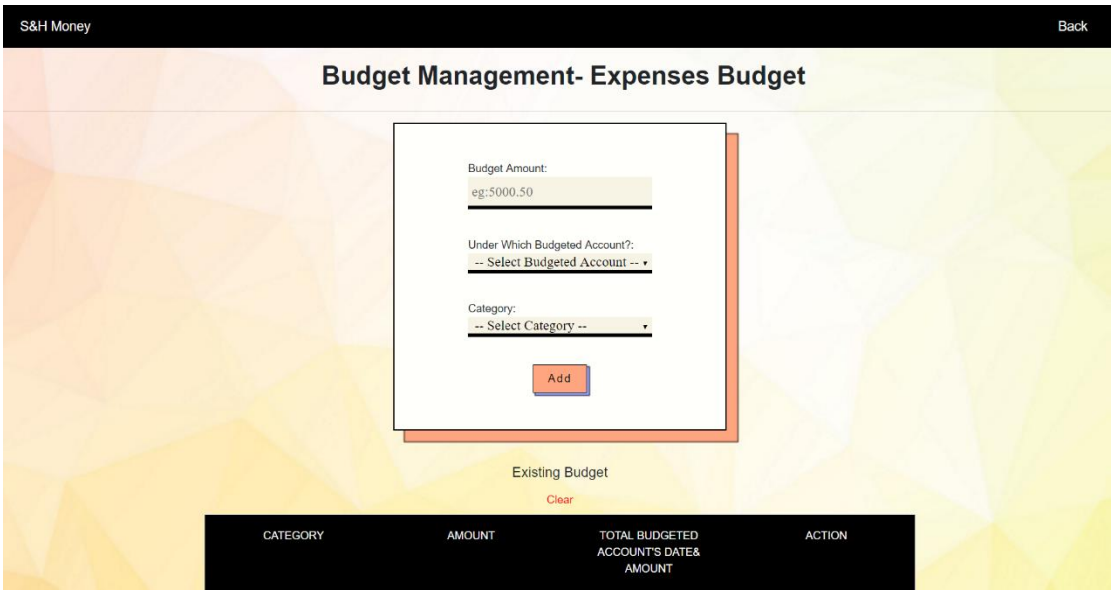


Figure 5.4.39 Category Budget Plan Management

In the figure above, the user can add a category budget plan by providing the budget amount, from which account budget plan and which category. Basic validations will be implemented for this form, such as the invalid format of amount and duplication.

However, if there are no account budget plans or categories, the application will prompt an error message to notify users to add the information. In addition, if the category budget plan exceeded the account budget plan, the application will prompt an error message to the user. Once all the input fields are correct, the system will display a success message to inform the user that the record has been added. The success message of adding a category budget plan is shown in figure 5.4.40.

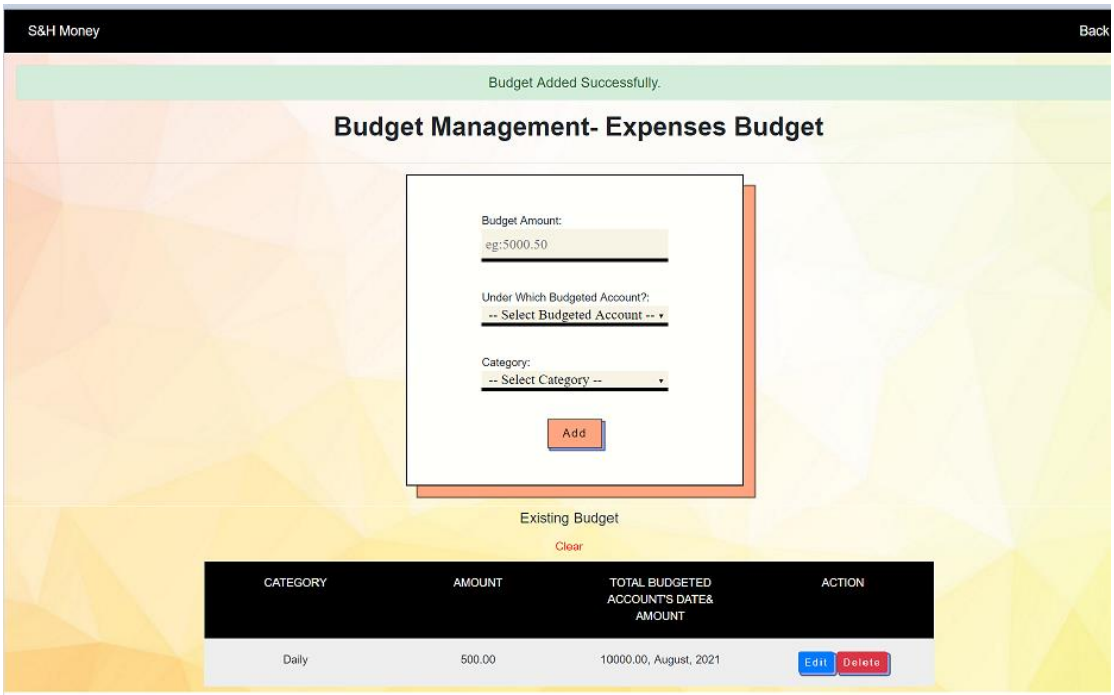


Figure 5.4.40 Success Message of Adding a Category Budget Plan

In the figure above, the success message will show on the top of the page. After that, the table below will automatically show the record immediately. The user can click on the *Edit* button to update a particular category plan. Once the user clicks on the *Edit* button, the user will be redirected to the update form to modify the category budget plan. The UI design is designed as illustrated in figure 5.4.41.

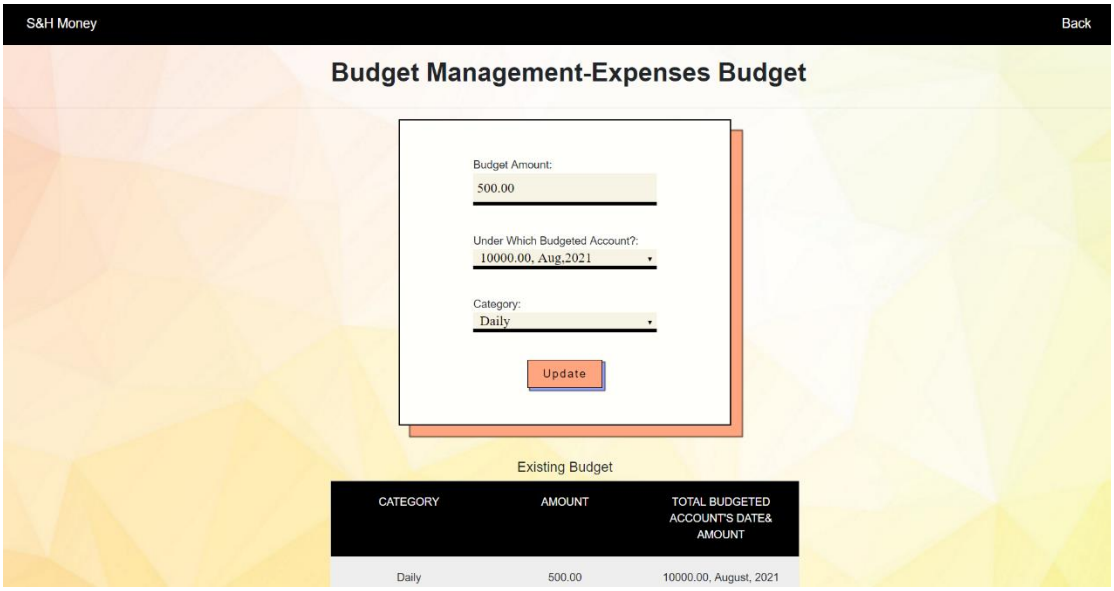


Figure 5.4.41 Update Category Budget Plan Page

In the figure above, the system will display the record in the form for user to modify. The validation of this update function will be the same as the add function. Once the user updates with a valid record, the system will display a success message to let the user know. The success message is shown in figure 5.4.42.

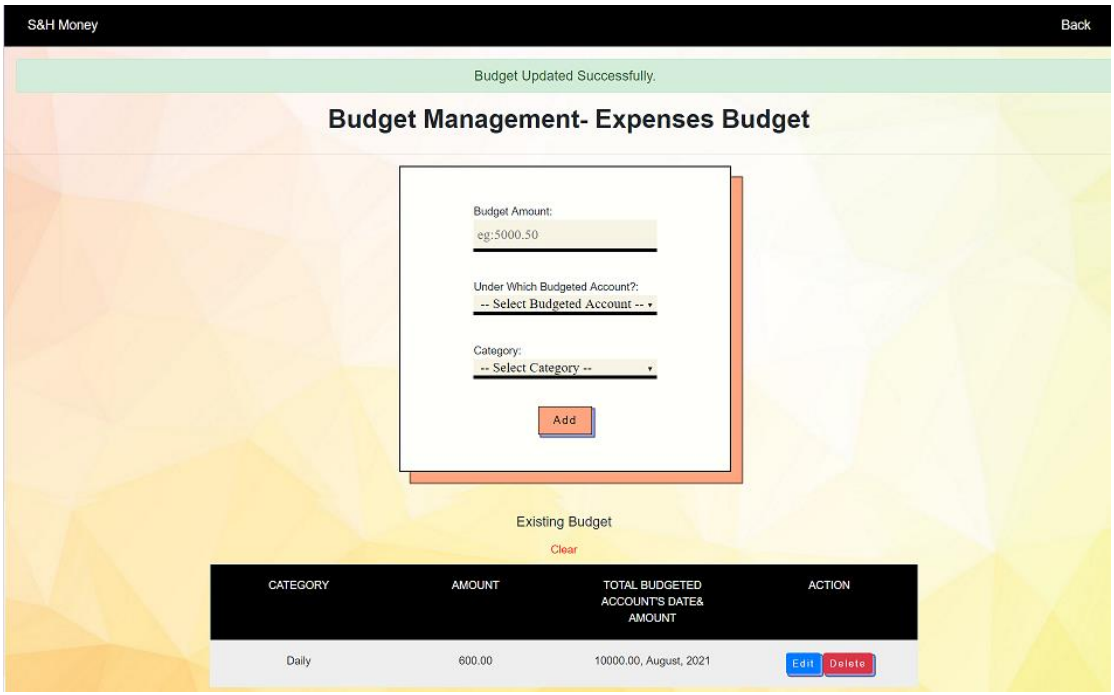


Figure 5.4.42 Success Message of Updating Category Budget Plan

Next, the user can perform the delete function to delete a particular category-budget plan by clicking on *delete* button in the table. After clicking on the *delete* button, the system will also prompt a message to inform the user the record has been deleted. The example of a notification is shown in figure 5.4.43.

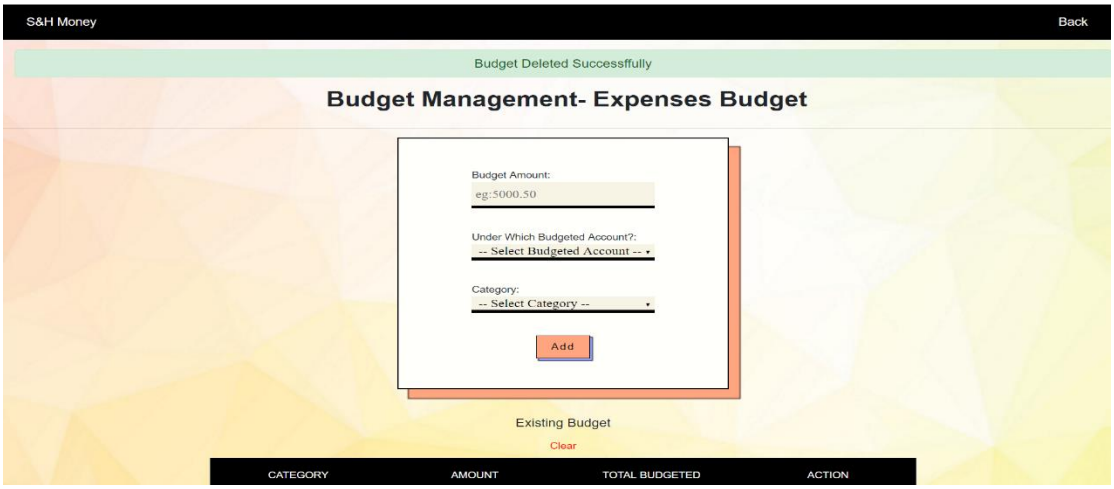


Figure 5.4.43 Success Message of Deleting a Category Budget Plan

These are all the category budget plan management and UI. Next, the user can modify the account budget plan by clicking on the *Edit Account Budget Planning* button in the budget menu, the user will be redirected to the view account budget plan interface. The user interface for the budget management is shown below in figure 5.4.44.

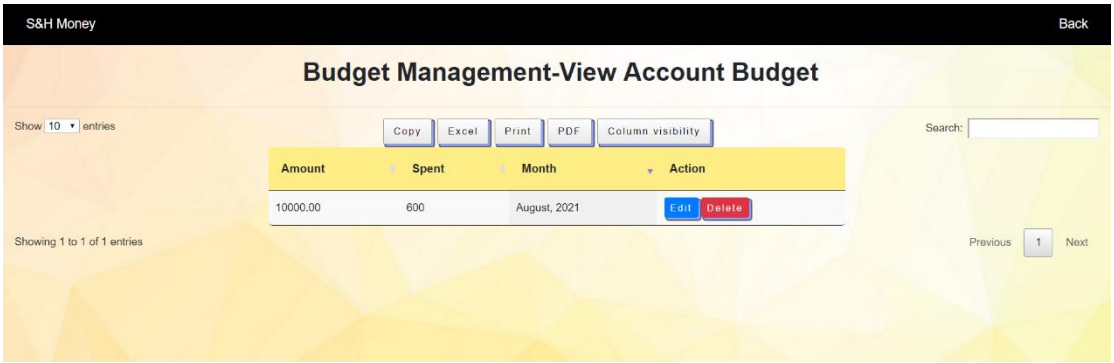


Figure 5.4.44 The List of Account Budget Plan

This table shows the budget plan amount, spent amount, and the Month of the budget plan. Users can refer to the table to see the current status of each of the account budget plans. Next, the user can perform sorting, dynamic searching, select record per page and pagination in this table. In addition, the user can perform the edit and delete

function by clicking on the related button in a particular row. Once the user clicks on the *Edit* button, the user will be redirected to the update form to modify the account budget plan. The UI design of the update form is designed as in figure 5.4.45.

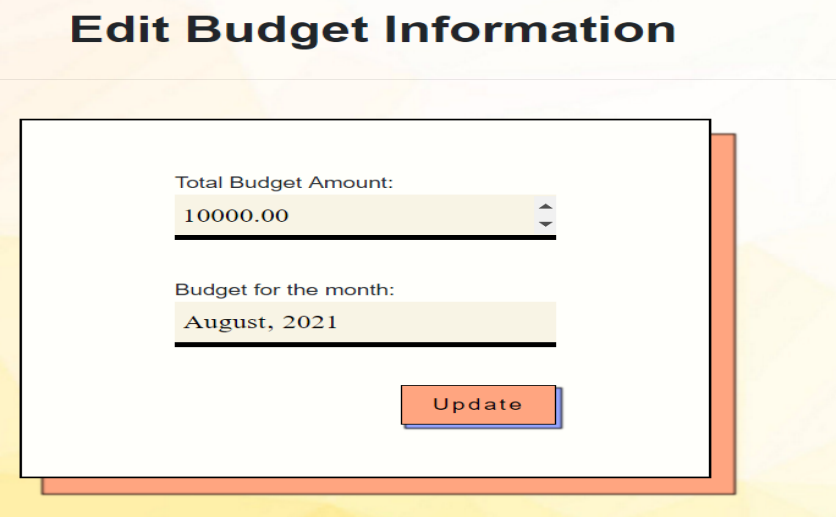


Figure 5.4.45 Edit Account Budget Plan

In the figure above, the system will display the record in the form for user to modify. The validation of this edit function will be the same as the add function, such as empty input, duplication and invalid format input. Once the user updates with valid information, the system will display a success message to let the user know. The success message is shown in figure 5.4.46.

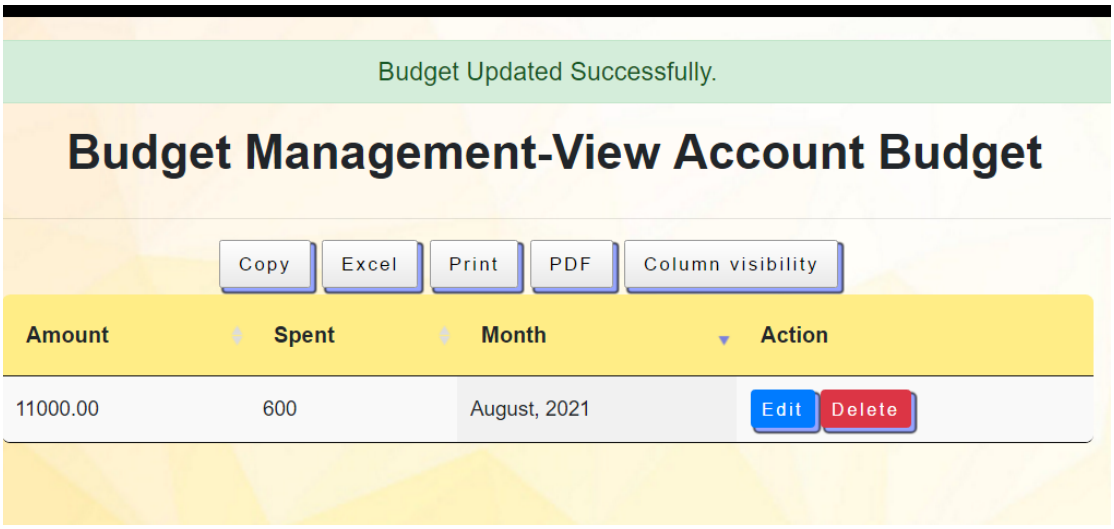


Figure 5.4.46 Success Message of Editing an Account Budget Plan

In the figure above, the success message will be shown at the top of the page. After that, in the table, the record is changed according to the update form. Next, the user can perform the delete function to delete a particular account budget plan by clicking on the *delete* button in the table. After clicking on the *delete* button, the system will display a success message to inform the user that the record has been deleted. The example of a notification is shown in figure 5.4.47.

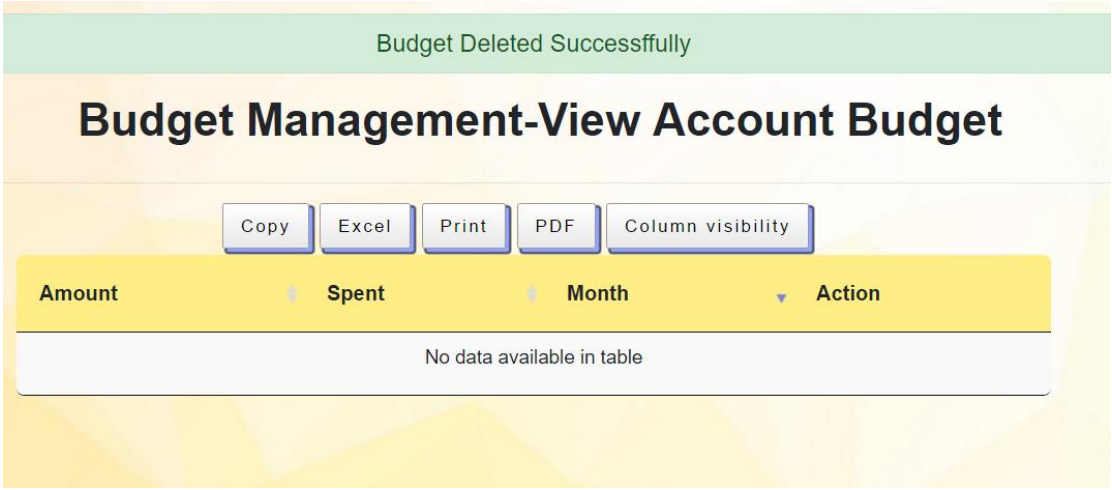


Figure 5.4.47 Success Message of Deleting an Account Budget Plan

These are the UI and functionalities of the personal budgeting module. After that, back to the expenses menu interface, the overall status will show the budget plan with the spent amount (expenses) to keep track of the budget plan. The overall status table is shown in figure 5.4.48.

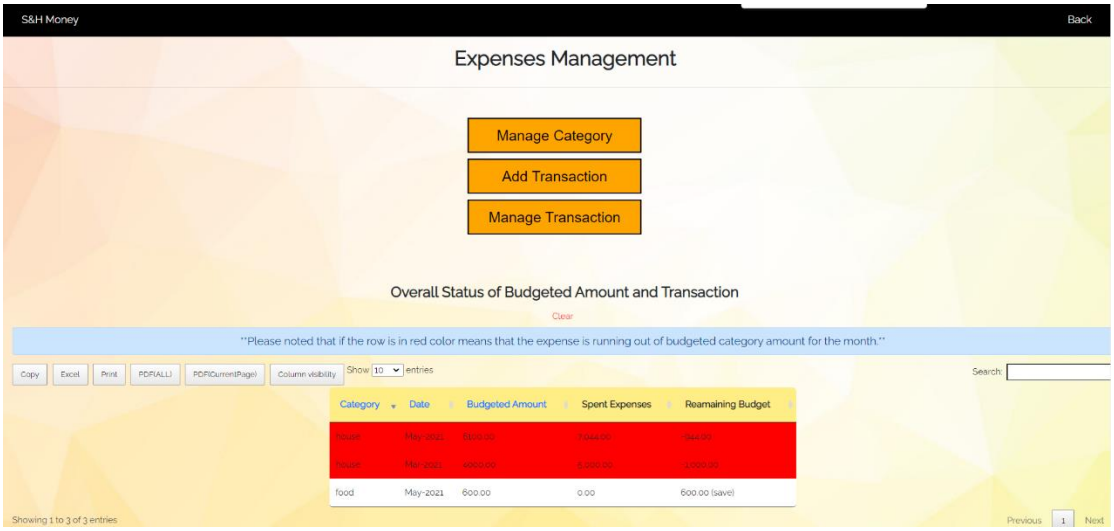


Figure 5.4.48 Overall Status Table of Budgeted Amount and Transaction

In the figure above, once there existed any categories budget plan, the overall status will straight away display the related data such as budgeted amount, spent expenses, and remaining budget to user see. However, if the row is in the red background, the category budget plan is run out of budget. Therefore, it is used to warn user need to save money or spend less money.

5.4.5 Generate Report

First, this module covers income and expenses management and the personal budget module. The reason is that this module is used to generate external reports such as pdf and excel in these modules. Therefore, the user needs to click on the related button to trigger the functionalities of this module. The UI of this module is shown in figure 5.4.49.

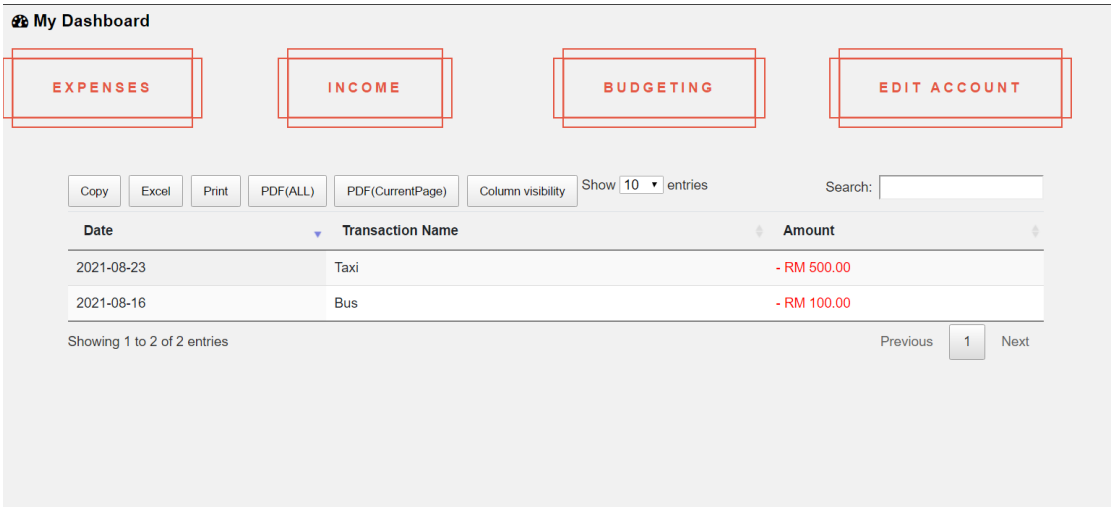


Figure 5.4.49 Generate Report in Home Page

In the figure above, when the user added income or expenses, the table will show the transaction to let users see the overall transaction that was recently involved. If the transaction is an expense in the amount column, the font colour will be in red with a "-" before the amount. On the other hand, if the transaction is income, the font colour will be in green colour with a "+" before the amount. Instead of referring to the recent income and expenses in the dashboard, the user can just see the recent transaction in this home page table. However, if the user wants to keep track of previous transactions, the user needs to go to the respective dashboard to see more details or search in this home page table to see the record. After that, there are quite many buttons such as *Excel, Print, PDF(ALL), PDF(CurrentPage), Column Visibility* in the table above. The *PDF* and *Excel* buttons are used to generate the report to the file type of PDF or Excel. The PDF report can be divided into two different types: generate pdf report containing all the records and generate pdf report containing the record based on the current page. Once the user clicks on one of the buttons, the system will first prompt a file explorer to let the user choose the file location. The UI of selecting file location is shown in figure 5.4.50.

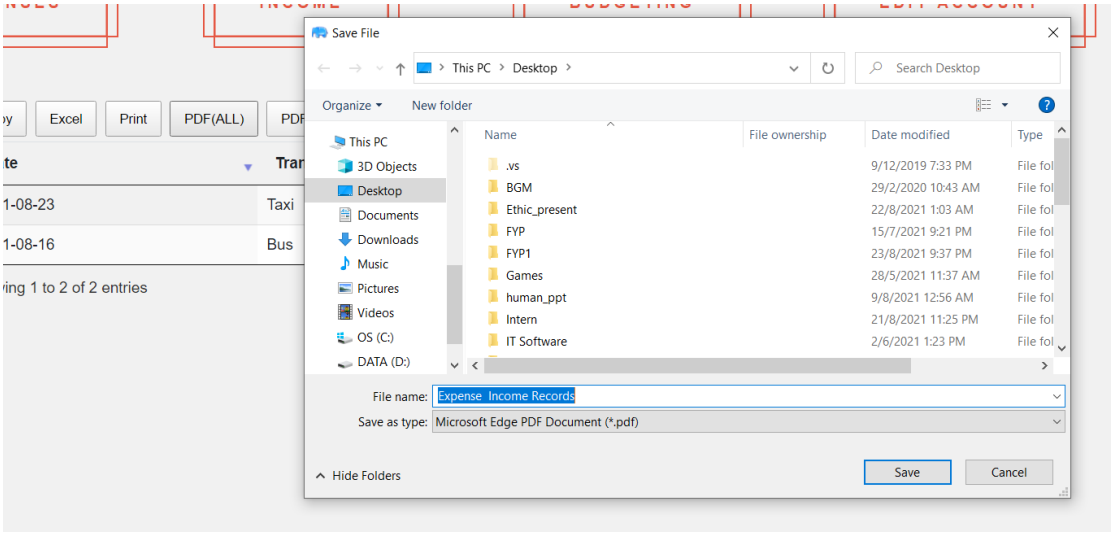
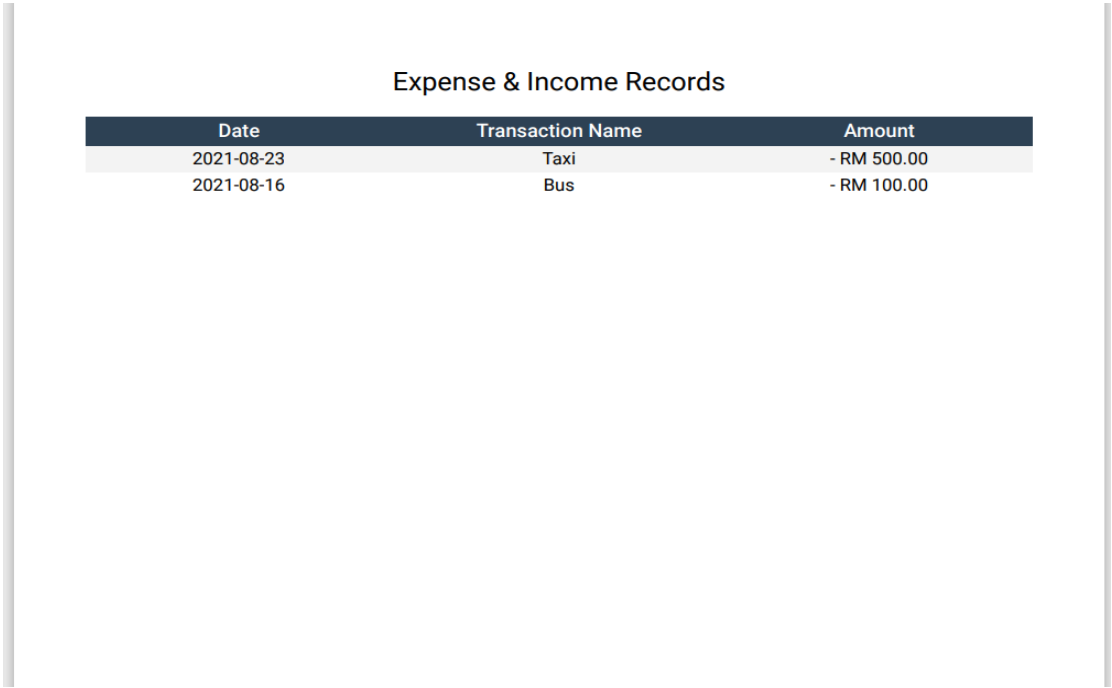


Figure 5.4.50 Selecting File Location

In the figure above, the user can rename the file name and choose the location to export the report to their device. Once the report has been generated, the user can go to the file location to open the pdf file to see the record. The format and UI of the pdf file are shown in figure 5.4.51.



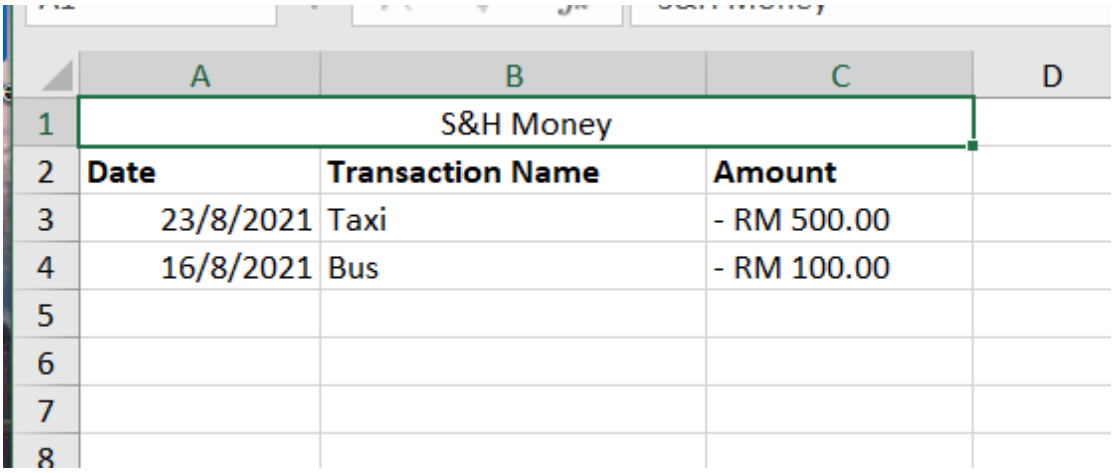
Expense & Income Records

Date	Transaction Name	Amount
2021-08-23	Taxi	- RM 500.00
2021-08-16	Bus	- RM 100.00

Figure 5.4.51 PDF Report Content

After that, if the user clicks on the *Excel* button in the table, the same steps will be taken in figure 5.4.37. Once the excel file has been generated, the user can go to the

file location to open the excel file. The UI and format of the excel file are shown in figure 5.4.52.



	A	B	C	D
1	S&H Money			
2	**Date**	**Transaction Name**	**Amount**	
3	23/8/2021	Taxi	- RM 500.00	
4	16/8/2021	Bus	- RM 100.00	
5				
6				
7				
8				

Figure 5.4.52 Excel Report Content

In the figure above, all the records will store in an excel file. This file is allowed the user to do some manual modification based on the user preference or save it to keep as a copy. Back to the table, as mentioned earlier, if the user clicks on the *Print* button, the system will prompt the printing step. The UI of the printing step is shown in figure 5.4.53.

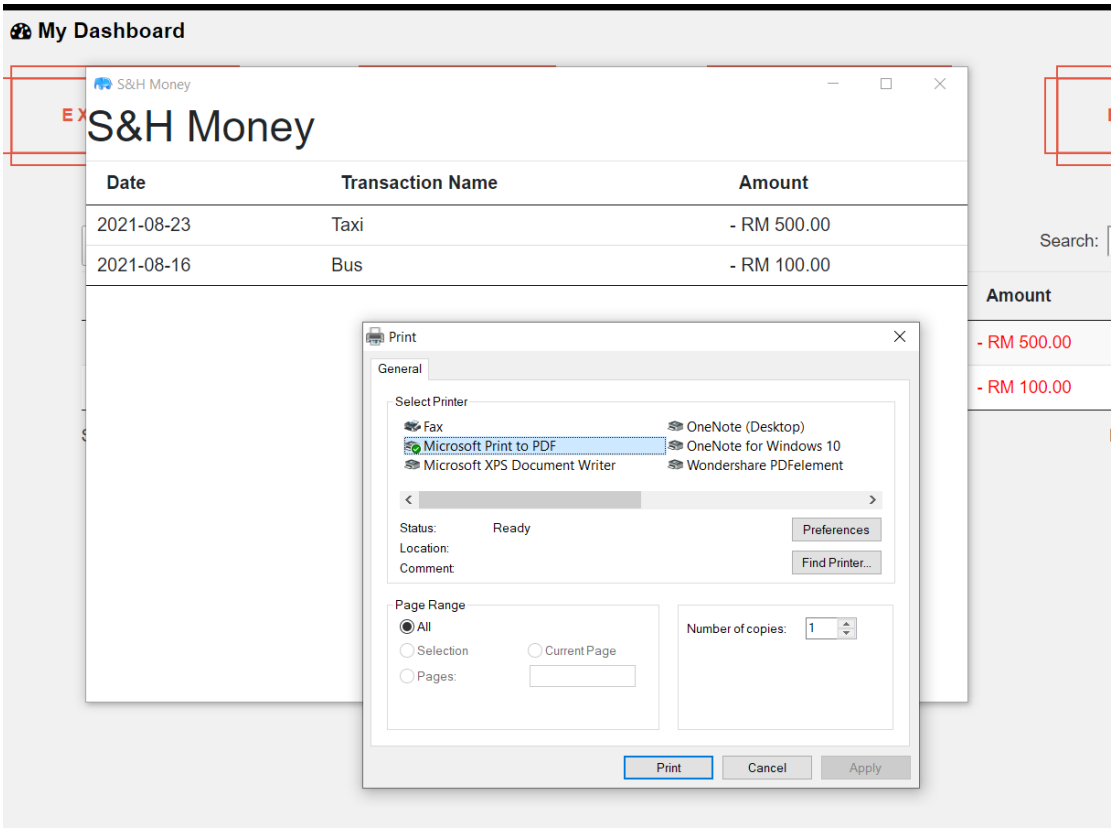


Figure 5.4.53 Transaction Printing Steps

In the figure above, if the user has his or her own printer, the user can connect it and print the record to a hardcopy. All the steps will be the same as the normal printing steps. After that, for the *Column Visibility* button, when the user clicks on the button, the system will show the available columns in the table to let the user select which columns are hidden or showing. The UI of selecting columns visibility is shown in figure 5.4.54.

Figure 5.4.54 Selecting Column Visibility

Based on figure 5.4.54, the user is allowed to click on the columns that are shown by the system. Once the user clicks on one of the columns, the table will immediately change the table columns. The table after selecting some columns visibility is shown in figure 5.4.55.

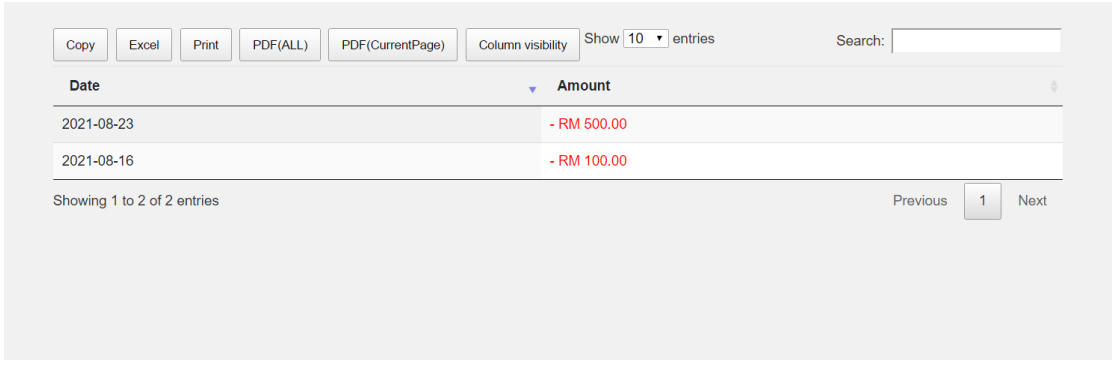


Figure 5.4.55 Table After Selecting Columns Visibility

In the figure above, the table has been filtered after the user performs the column visibility. If the user wants to show back the related columns, the user can just proceed to the same step as how to hide the column visibility. In addition, this generate report module not only covers this table but is also implemented in the income and expenses module and personal budgeting module. The UI of this generate report module in the income and expense module is shown in Figures 5.4.56 and 5.4.57.

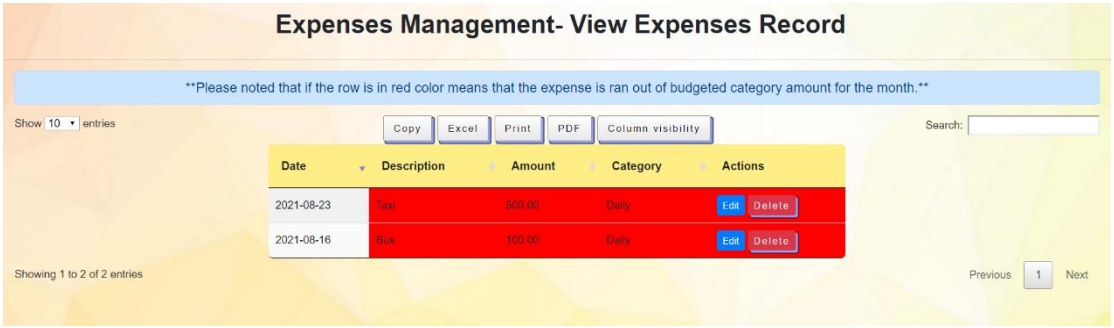


Figure 5.4.56 Generate Report Functions in Expenses List

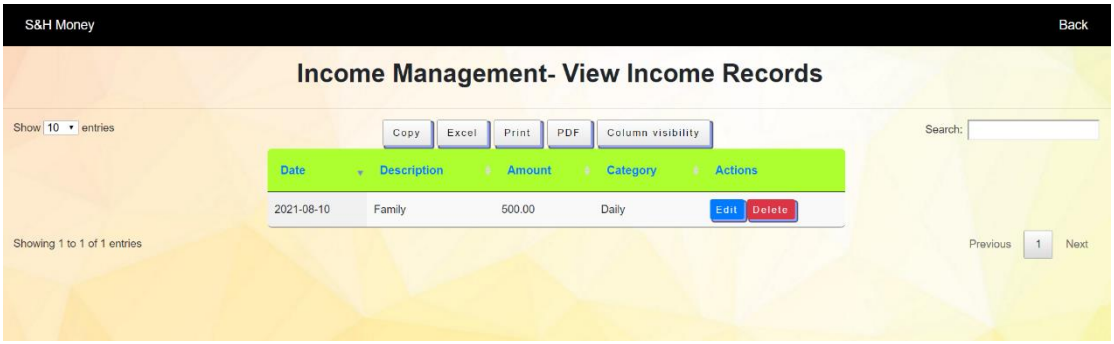


Figure 5.4.57 Generate Report Functions in Income List

In the figure above, all the buttons above the table will work exactly the same as the function explained in the previous figure. Next, the UI of this generate report module in the personal budgeting module is shown in figure 5.4.58.

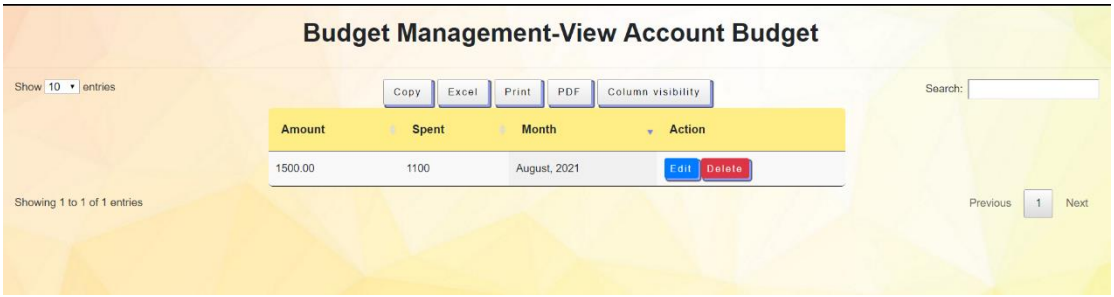


Figure 5.4.58 Generate Report Functions in Personal Budgeting

In figure 5.4.58, the user can perform the generate report function by clicking on those buttons above the table. All the functionalities will be the same as the table on the home page.

5.4.6 Generate Graph and Chart

In this module, the graph and chart generation is based on the record that user has entered to the expenses and income module, budget plan module and account module. First, this module is able to generate a yearly summarised pie chart to let user visualise their personal finance status. Figure 5.4.59 shows the output of yearly expenses and income pie chart.

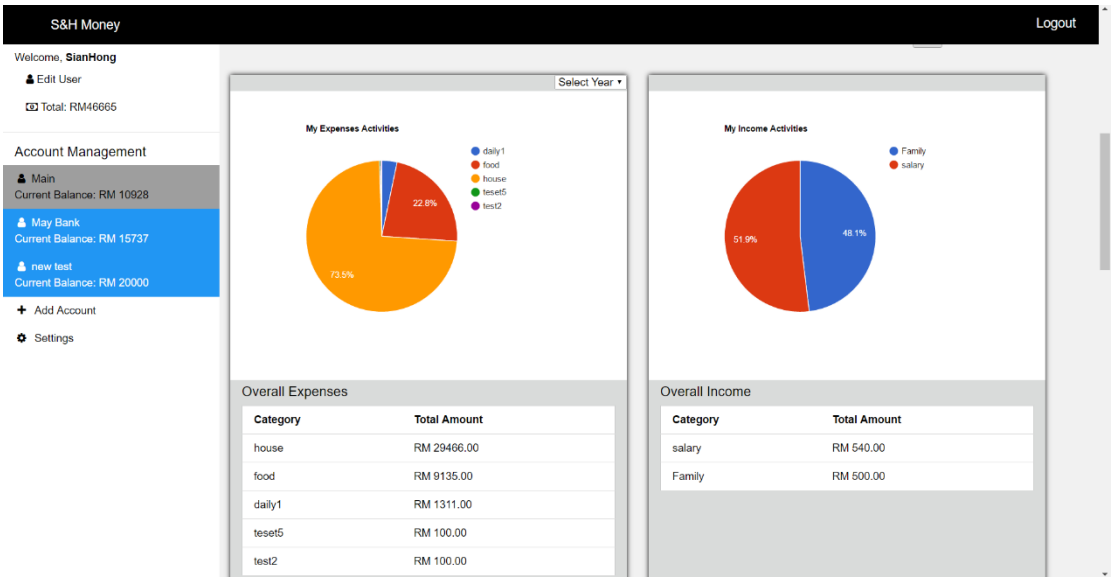


Figure 5.4.59 Pie Chart of Income and Expenses

In the figure above, if the user is added some data to the expenses and income module, this graph and chart module is able to generate the pie chart as shown in figure 5.4.59. After that, there is a table below the pie chart. Basically, the table is used to show how much money for the category in the particular year. By default, the pie chart is shown all the expenses and income without grouped by any years. If the user wants to see particular year overall income and expenses pie chart, the user has to use the filter

function provided by choosing the year on the pie chart dropdown list. The select option for this pie chart is shown in figure 5.4.60.

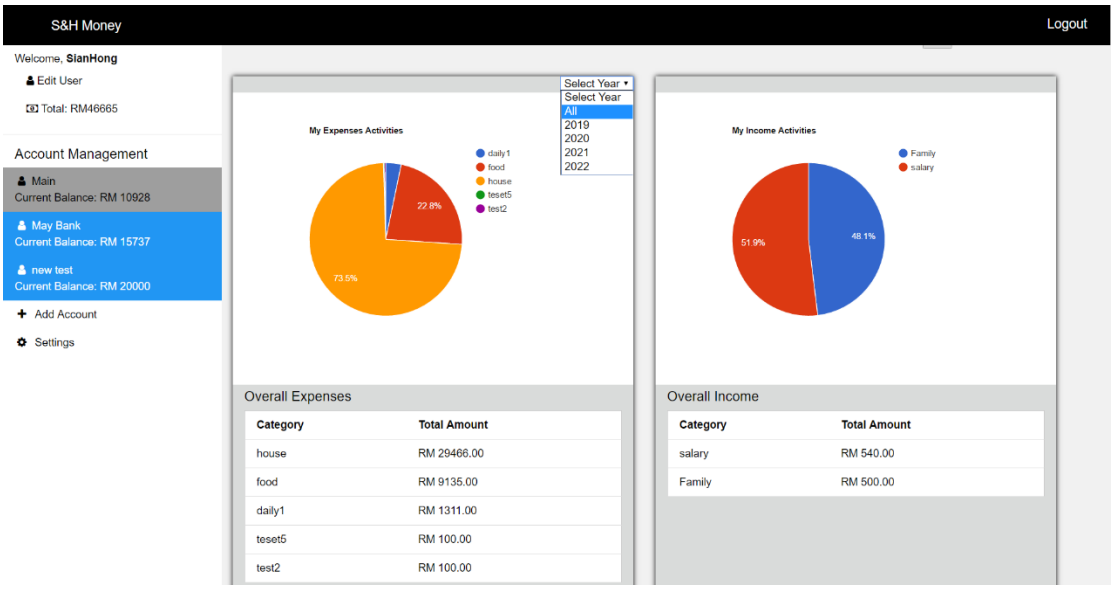


Figure 5.4.60 Filter Function for Pie Chart

In the figure above, the user can choose a year in the drop-down list to perform filter function. After the user has chosen a year, the application will refresh the page to display the related information. Figure 5.4.61 shows the output of using filter function on this pie chart.

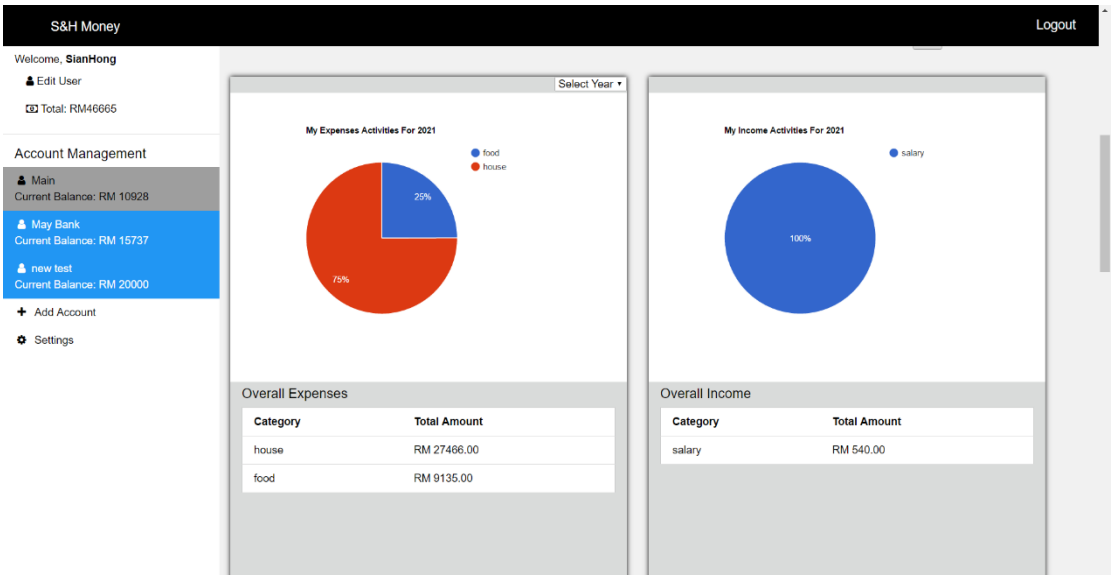


Figure 5.4.61 Filter Function Result

Figure 5.4.61 shows the result of this filter function. The title of the pie chart is shown “My Expenses Activities for 2021” if the user has chosen 2021 in the filter list. The income pie chart is the same as expense as well. Not only that, the overall table below the pie chart will also change the value based on the filter value. Furthermore, in order to give user has a better view of their income and expenses records, this application is able to generate monthly line chart record. This monthly line chart record keep changing when the user has performed the filter function that discussed earlier. If the user choose year 2021, the line chart will be based on the year 2021 to generate the pattern by month categories(Jan, Feb and etc). Figure 5.4.62 shows the output of line chart.

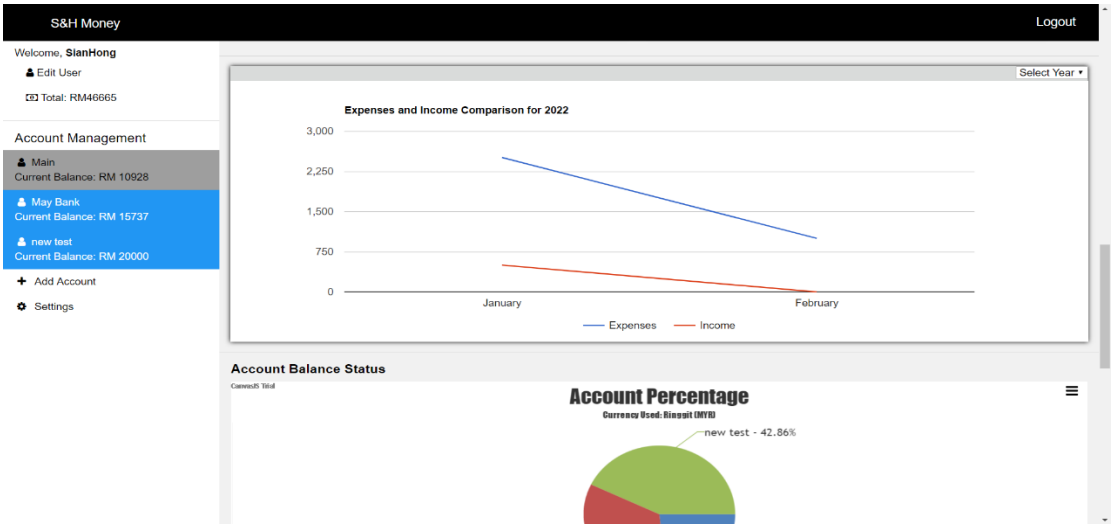


Figure 5.4.62 Income and Expenses Line Chart

In the figure above, if there is not any selected filter value, the line chart by default is shown the current year income and expenses record. However, only the month had income and expenses records will be displayed in the line chart. In other words, if there is not any income and expenses records in January and March but had records in February and April, the line chart only displayed February and April line chart. Moving on, another pie chart is used in this application to show the percentage of each account usage. Figure 5.4.63 shows the output of the account pie chart.

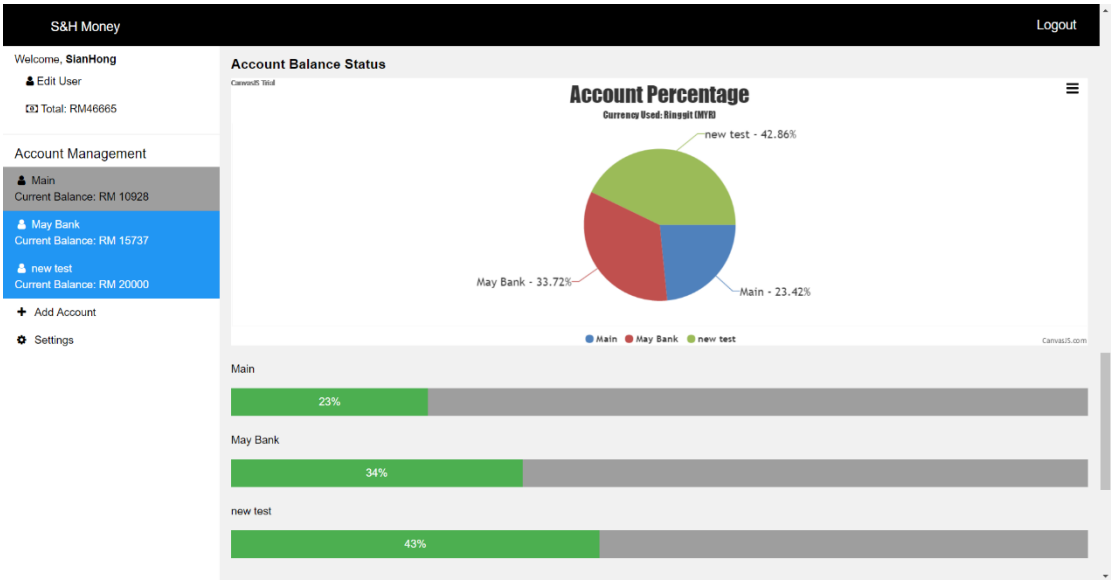


Figure 5.4.63 Account Percentage Pie Chart

In the figure above, the pie chart is used all the account's balance to generate the percentage and after that visualise the percentage in pie chart and a progress bar chart. Furthermore, the user can use mouse to touch on the pie chart in order to see how many balance the respective account have. Figure 5.4.64 shows the output when the user touch on the account pie chart.

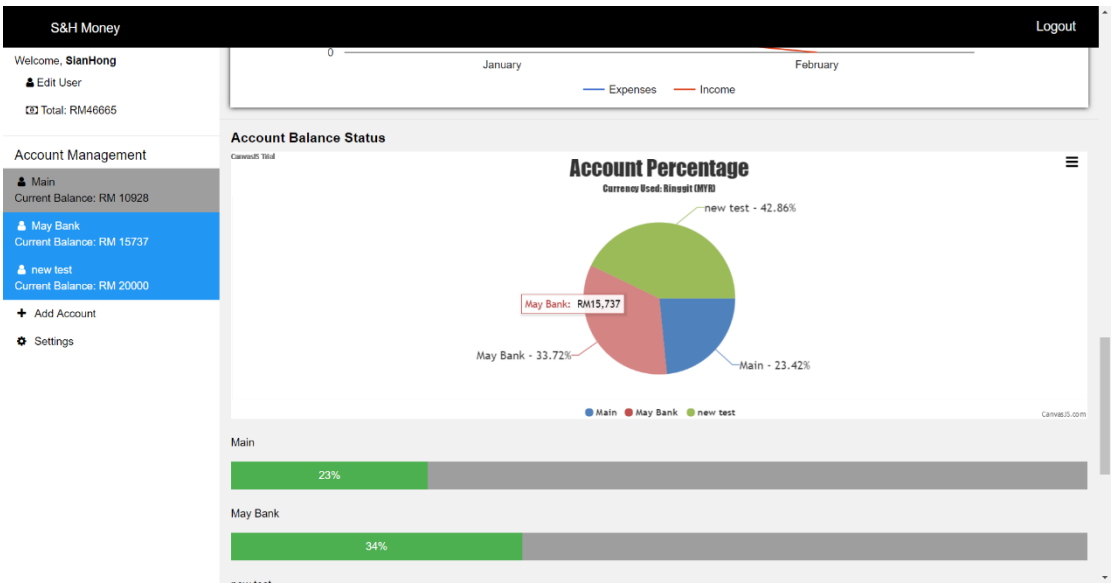


Figure 5.4.64 Account Balance in Pie Chart

After that, an option icon is displayed on the top right corner of the pie chart. This icon is used to export the chart pattern to the external file, such as PNG. JPG and Printing. The option values is shown in figure 5.4.65.

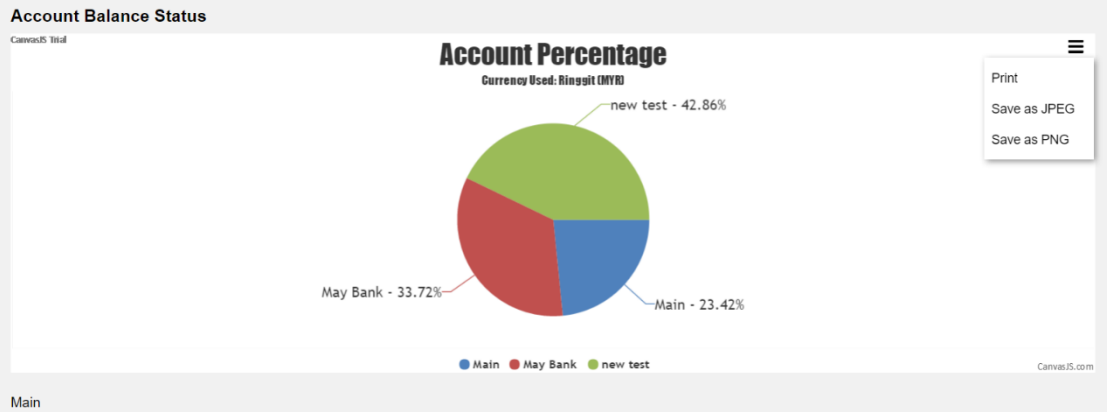


Figure 5.4.65 Export Pie Chart

Once the user clicks on any of the option values, the flow of exporting is totally same as the generate report module, the only different is the export result. Figure 5.4.66 shows the export pie chart result in PNG file format.

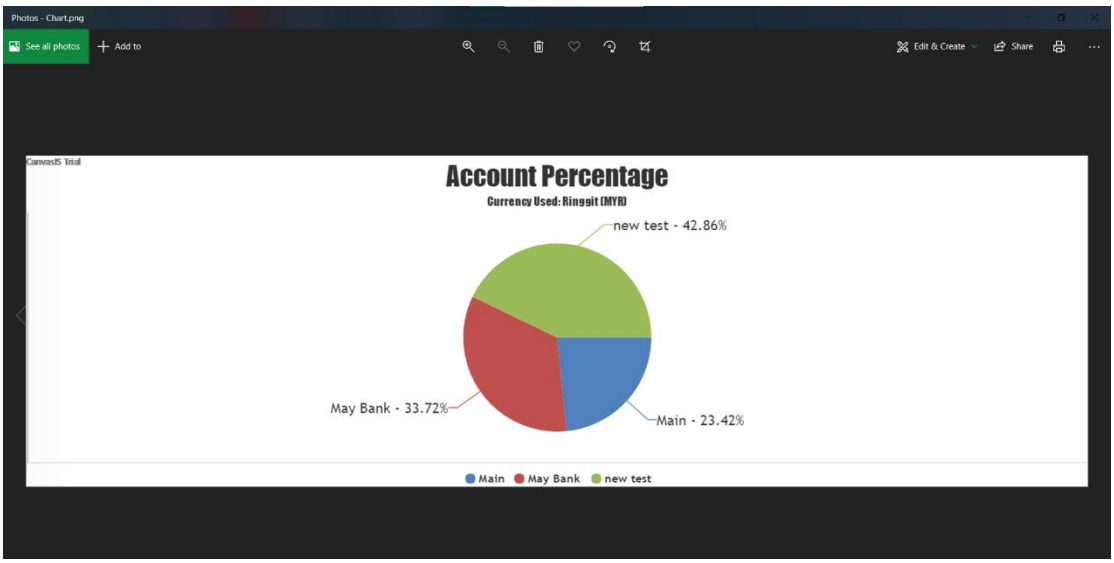


Figure 5.4.66 Export Pie Chart Result

Moreover, a line chart of budget plan is shown below the account pie chart. This budget plan line chart is used to display the how much money have saved for each account budget plan. The basic concept is totally same as the line chart of income and expenses. This account budget plan line chart is only shown the month which had the account budget plan. If the month did not have any account budget plan, the line chart would not display for that month. Figure 5.4.67 shows the output of account budget plan.

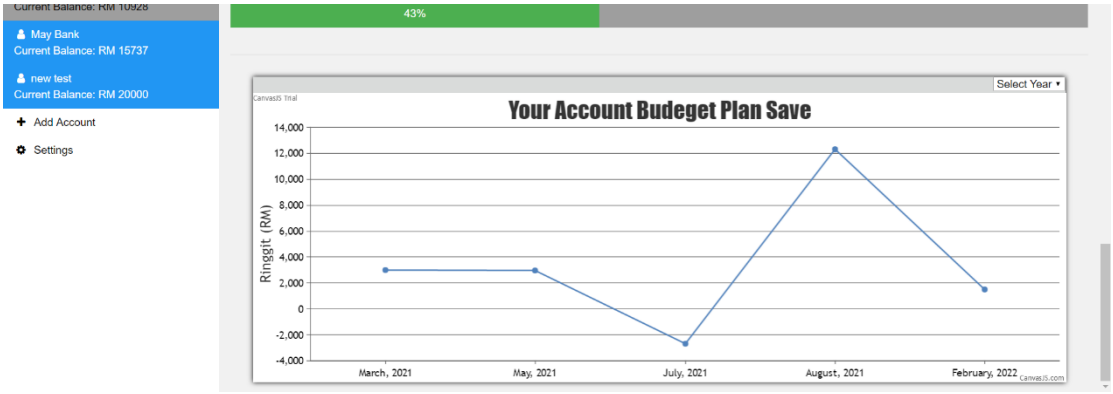


Figure 5.4.67 Line Chart of Account Budget Plan

In the figure above, the filter function is same as the income and expenses line chart. By default, it is shown current year account budget plan if there is not any years chosen in the filter function. After that, if the user wants to see how much money is saved for particular month, the user can touch on the pointer to view the amount. The example output is shown in figure 5.4.68.

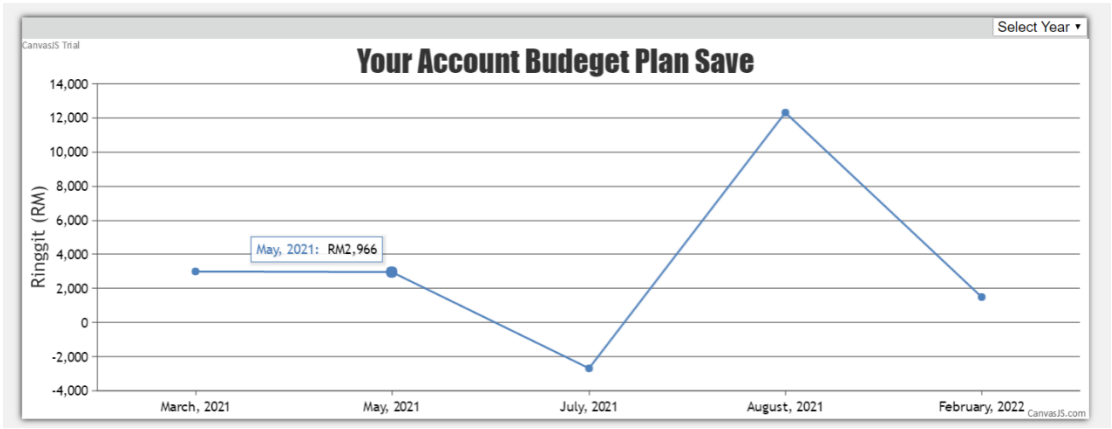


Figure 5.4.68 Amount Saved for Account Budget Plan

In a nutshell, this generate graph and chart module is able to help user visualise their personal finance status and based on the current status to make some important financial decision. The chart included in this application had income and expenses line chart and pie chart, account status pie chart and account budget plan line chart.

5.5 Concluding Remark

In a nutshell, this chapter is mainly discussed how this developed application has been implemented in terms of the project set up, application's setting and configuration as well as the system operation. Since this is a desktop-based application, the screenshots of installation steps of this application has been shown to help to solve any unnecessary confusion. After that, all the functions of this application have been separated by module and included with a lot of screenshots picture for reference. Not only that, a table has been displayed to list down all the required internal or external libraries for this application design, function operation and graph and chart visualization.

In the next chapter, system evaluation and testing for this application will be discussed.

CHAPTER 6

SYSTEM EVALUATION AND DISCUSSION

In this chapter, the system evaluation and discussion of this application is discussed. The chapter will start with the system testing and performance metrics and included the test case for this application. After that, testing result will be shown based on the test cases created earlier. Lastly, the project challenges and objective evaluation will be discussed after the testing result.

6.1 System Performance Metrics

This personal finance application aims to have at most 3 seconds to connect with the backend database. Furthermore, this application aims to process the user's operations, such as the CRUD functions, within a maximum of 3 seconds per operation. Every time the user makes an input to the system, the system will ensure that the maximum process time is under 3 seconds when storing inputs into the MySQL database. Moreover, the graph and chart generation feature will change within a maximum of 5 seconds upon user input. The system performance requirements are paramount in this application because they ensure that university students can quickly keep track of their personal finance.

6.2 Testing Setup and Result

6.2.1 User Module

In this section, the basic testing of the User Management Module is performed. The test cases are described to three activities in this module, namely Register, Login and Change Password activity.

Login Activity:

Table 6.1 Test Case: Press the *Login* button with empty input

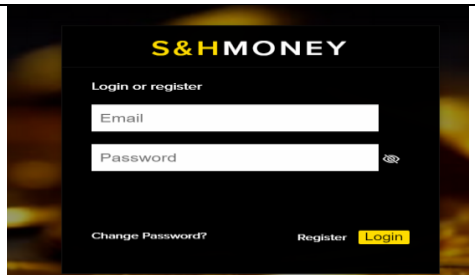
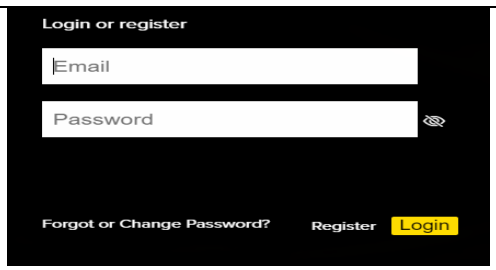
Test Case Name	Press the *Login* button with empty input
Test Case Description	User presses the *Login* button without filling in the email and password.
Input	
Results	
Status (Pass/Fail)	Pass

Table 6.2 Test Case: Valid email but invalid password

Test Case Name	Valid email but invalid password
Test Case Description	User presses the *Login* button with a valid email but an invalid password.

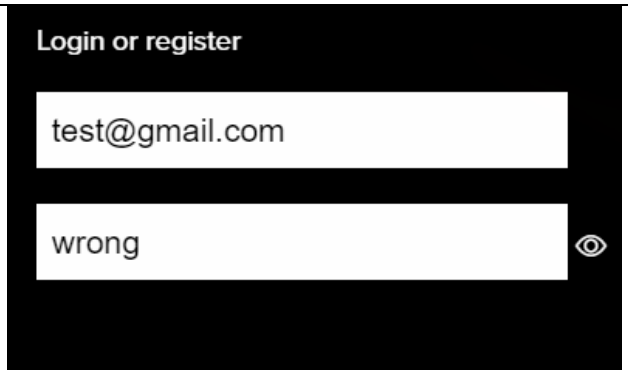
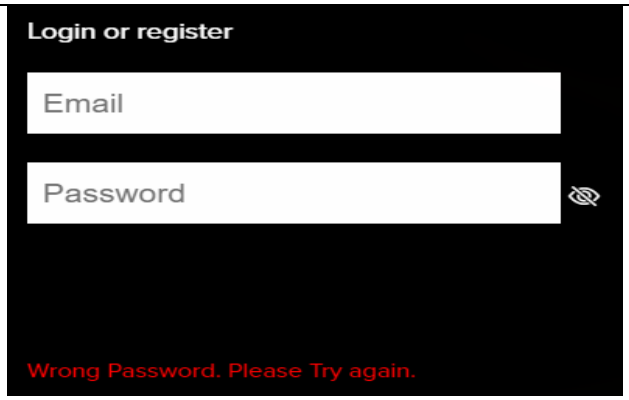
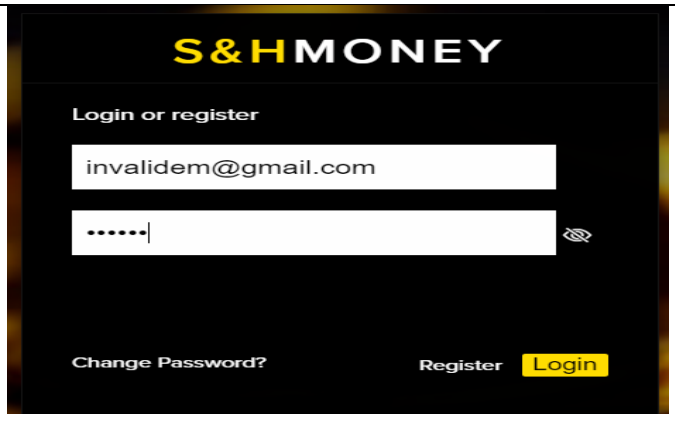
Input	
Results	
Status (Pass/Fail)	Pass

Table 6.3 Test Case: Invalid email and password

Test Case Name	Invalid email and password
Test Case Description	User presses the *Login* button with an invalid email and password.
Input	

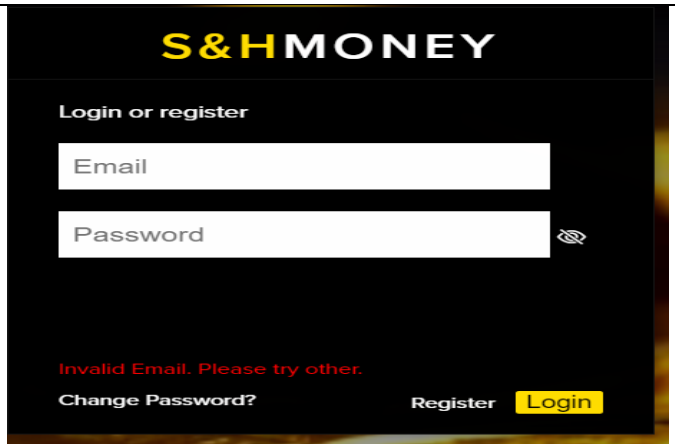
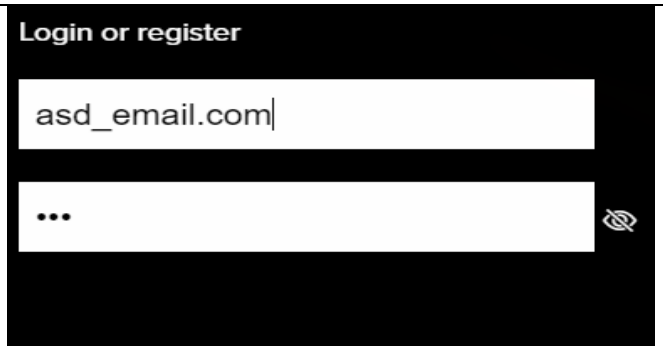
Results	
Status (Pass/Fail)	Pass

Table 6.4 Test Case: Invalid email format

Test Case Name	Invalid email format
Test Case Description	User presses the *Login* button with an invalid email format.
Input	
Results	
Status (Pass/Fail)	Pass

Register Activity:

Table 6.5 Test Case: Empty Input Fields

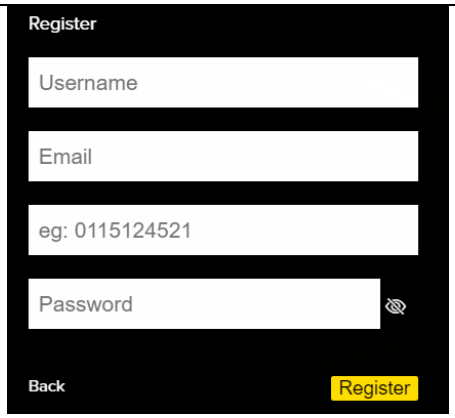
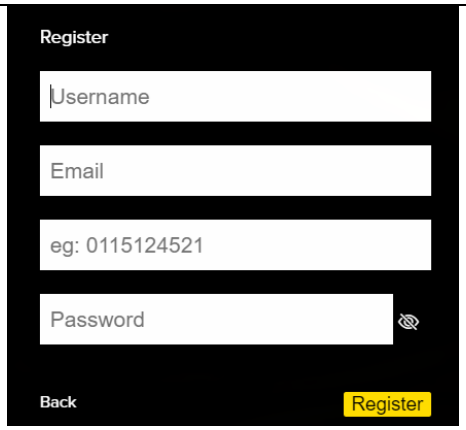
Test Case Name	Empty Input Fields
Test Case Description	User presses the *Register* button without filling in the email and password.
Input	
Results	
Status (Pass/Fail)	Pass

Table 6.6 Test Case: Invalid Phone Number

Test Case Name	Empty Input Fields
Test Case Description	User presses the *Register* button with invalid phone number.

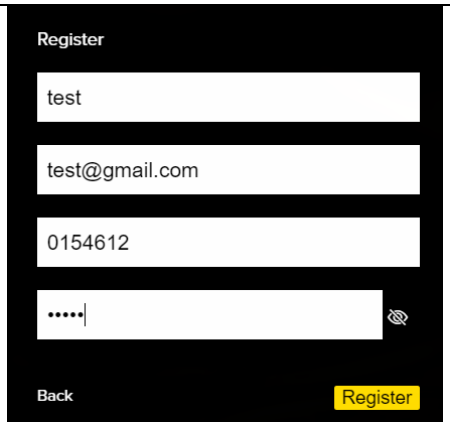
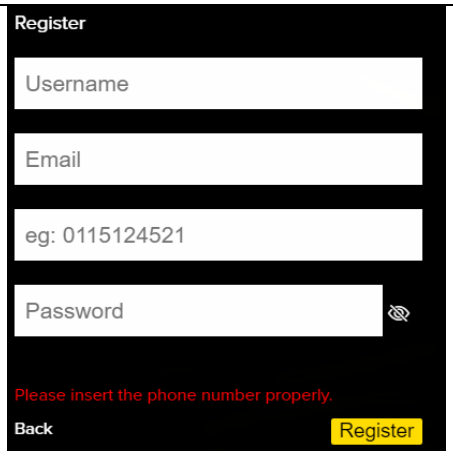
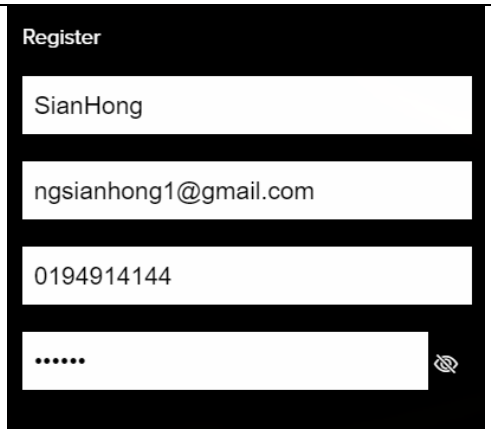
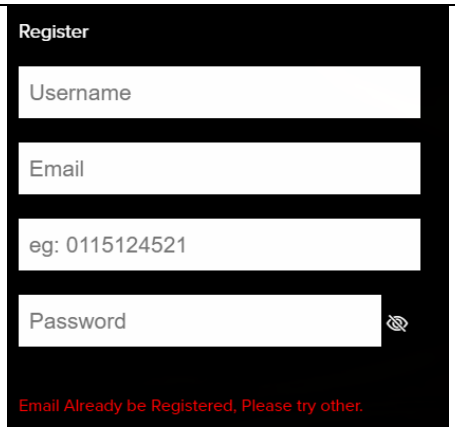
Input	
Results	
Status (Pass/Fail)	Pass

Table 6.7 Test Case: Email and password that exists in the system

Test Case Name	Email and password that exists in the system
Test Case Description	User presses the *Register* button with an email and password that exists in the system.
Input	

Results	
Status (Pass/Fail)	Pass

Forgot or Change Password Activity:

Table 6.8 Test Case: Empty Input Fields

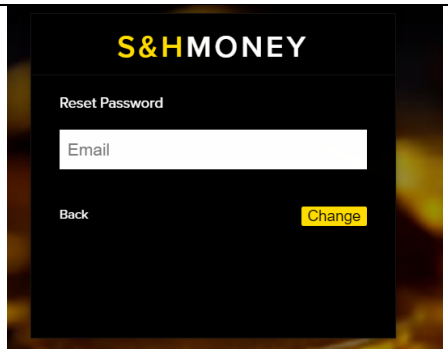
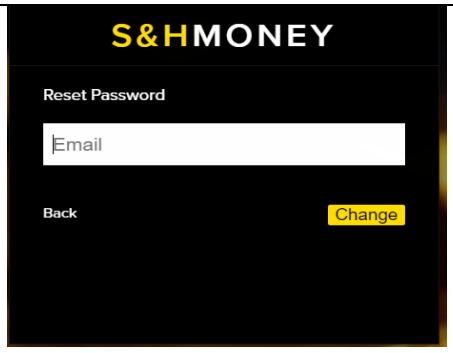
Test Case Name	Empty Input Fields
Test Case Description	User presses the *Change* button without filling in the email.
Input	
Results	
Status (Pass/Fail)	Pass

Table 6.9 Test Case: Invalid email

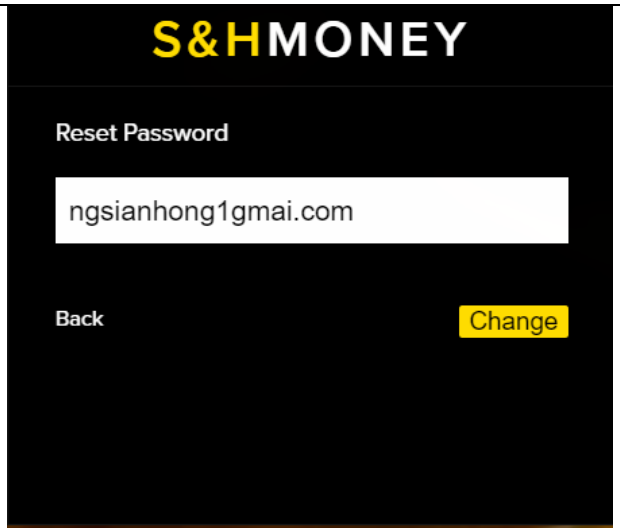
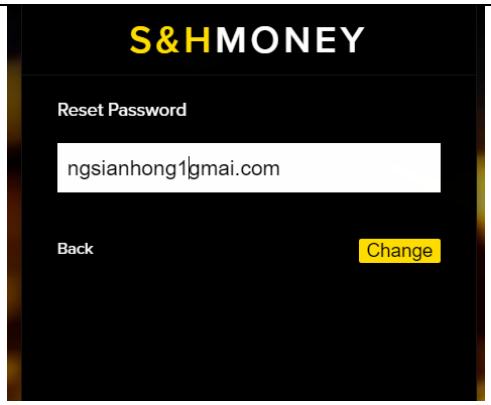
Test Case Name	Invalid email
Test Case Description	User presses the *Change* button with an invalid email.
Input	
Results	
Status (Pass/Fail)	Pass

Table 6.10 Test Case: Not registered email

Test Case Name	Invalid email
Test Case Description	User presses the *Change* button with a not registered email.

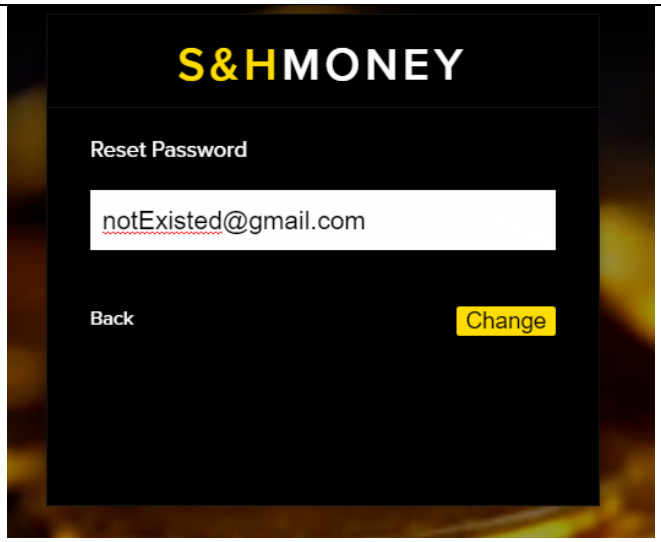
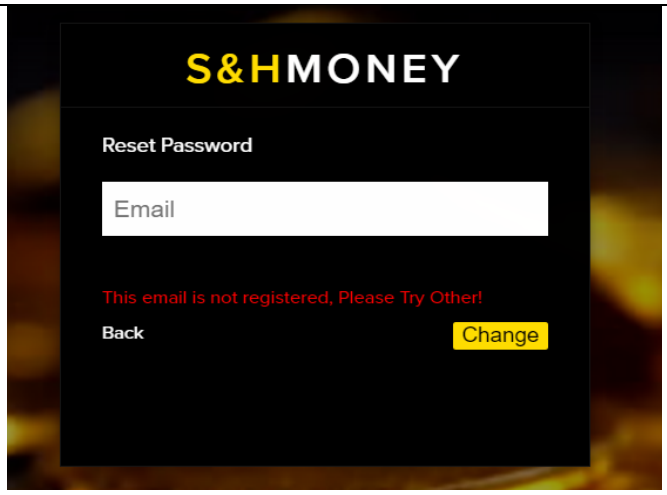
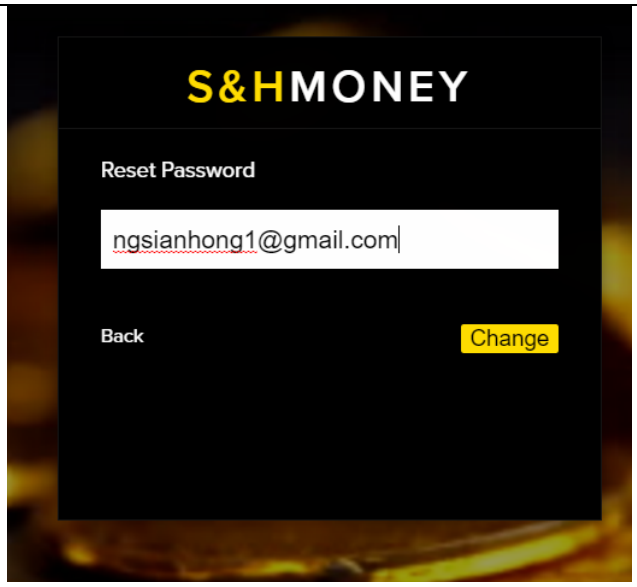
Input	
Results	
Status (Pass/Fail)	Pass

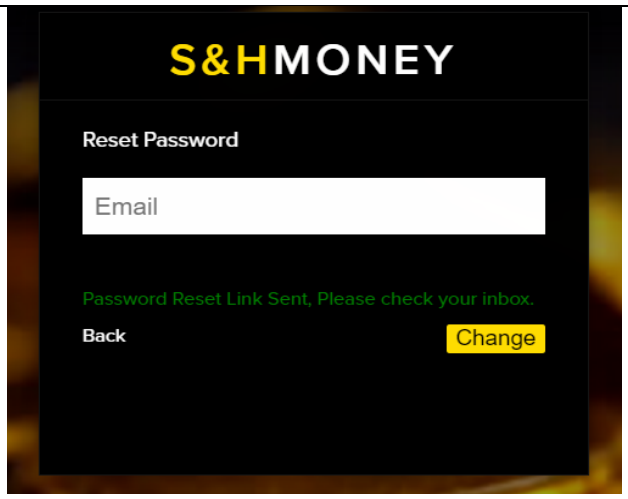
Table 6.11 Test Case: Valid email

Test Case Name	Valid email
Test Case Description	User presses the *Change* button with a valid email.

Input



Results



Reset your password for your S&H Money account 收件箱 ×

S&H Money <S&H_noreply@phptest-f2... 00:34 (0分钟前)
发送至 我

Hello,

Follow this link to reset your S&H Money account password for your ngsianhong1@gmail.com account.

https://phptest-f201b.firebaseapp.com/__/auth/action?mode=resetPassword&oobCode=qKOlzPbY2RUAAo1gM48YSFOlWyFdEbL34pbLDGxTJxsAAAF_t6DUaw&apiKey=AIzaSyDngHJh2snQIPGA7DsE_dOP8EZQgJ_CE_s&lang=en

If you didn't ask to reset or change your password, you can ignore this email.

Thanks,

Your S&H Money team

回复 转发

Status (Pass/Fail)	Pass

6.2.2 Account Module

In this subsection, basic testing of the Account Management module is performed. The test cases are described to two activities in this module, namely Add Account, Update Account activity and View Account activity.

Add Account Activity:

Table 6.12 Test Case: Add with Empty Input Field

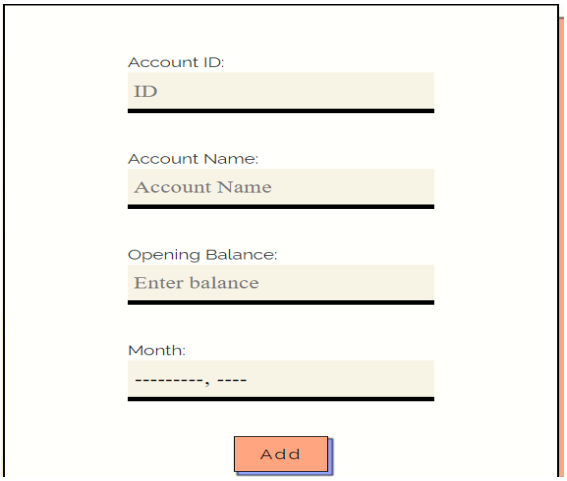
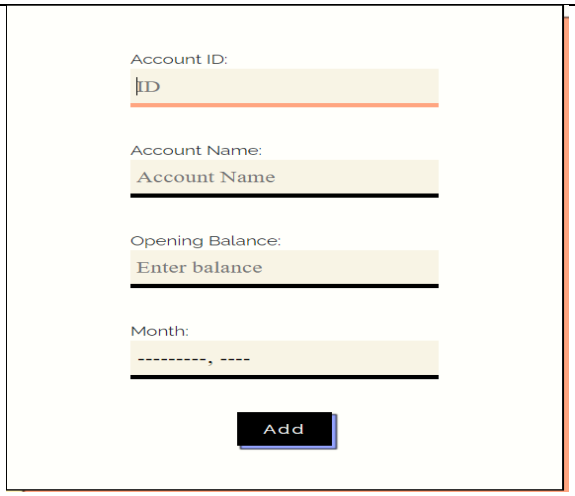
Test Case Name	Add with Empty Input Field
Test Case Description	User presses the *Add* button without filling in the information.
Input	
Results	
Status (Pass/Fail)	Pass

Table 6.13 Test Case: Invalid format of starting balance

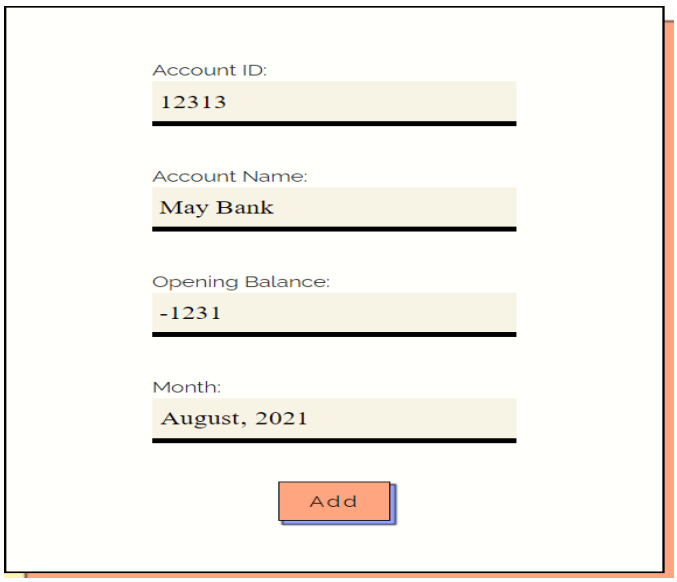
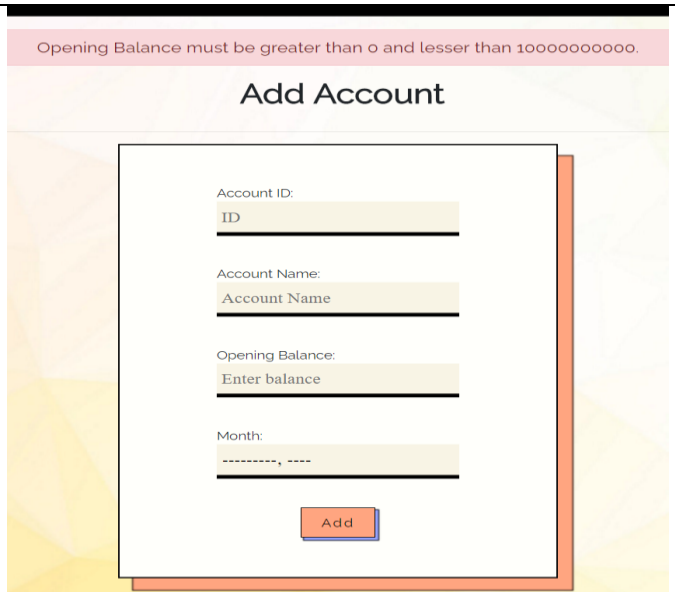
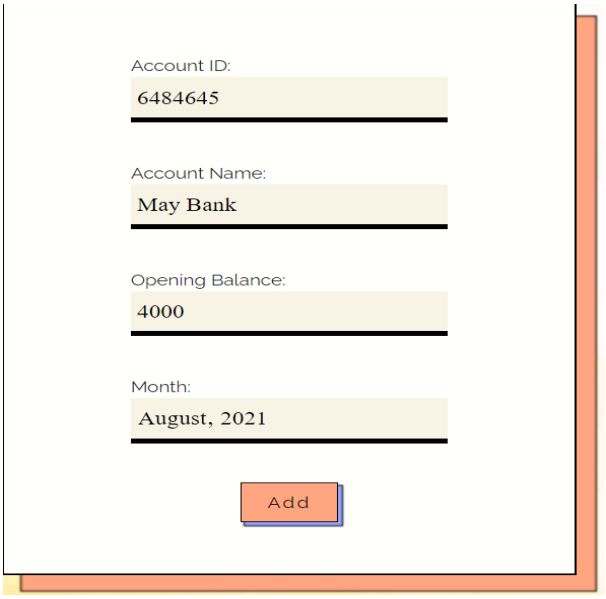
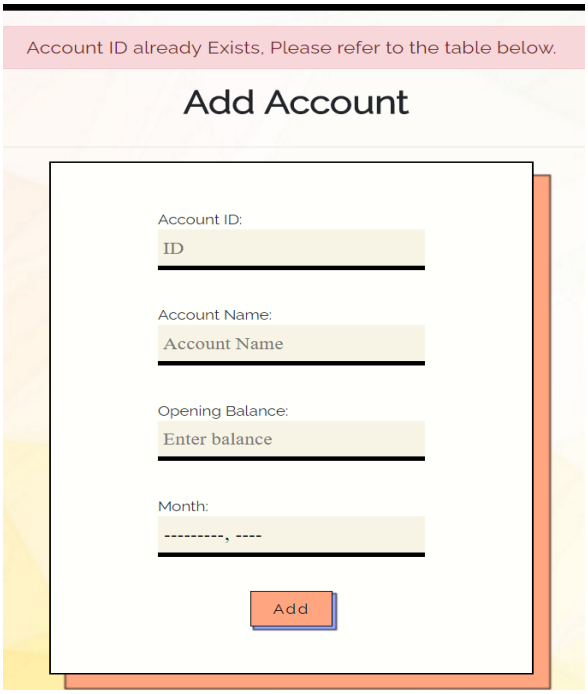
Test Case Name	Invalid format of starting balance
Test Case Description	User presses the *Add* button without filling in the information.
Input	
Results	
Status (Pass/Fail)	Pass

Table 6.14 Test Case: Duplication account ID

Test Case Name	Duplication account ID
Test Case Description	User presses the *Add* button with an account ID that is existing in the system.
Input	
Results	
Status (Pass/Fail)	Pass

Update Account Activity:

Table 6.15 Test Case: Invalid format of starting balance

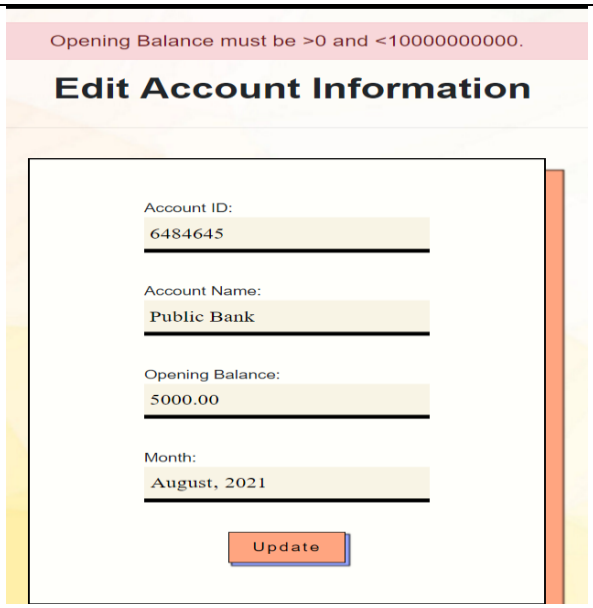
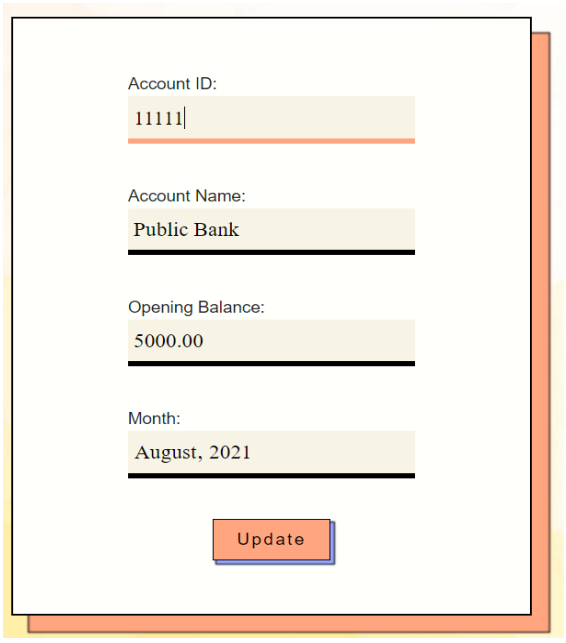
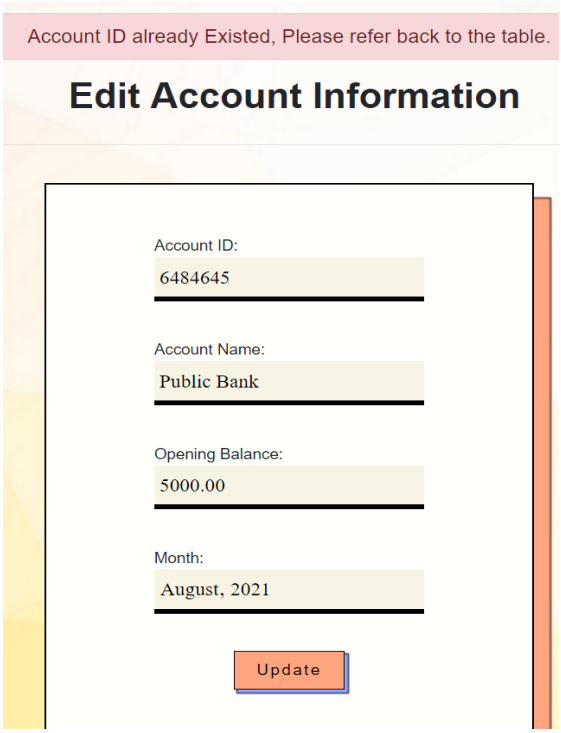
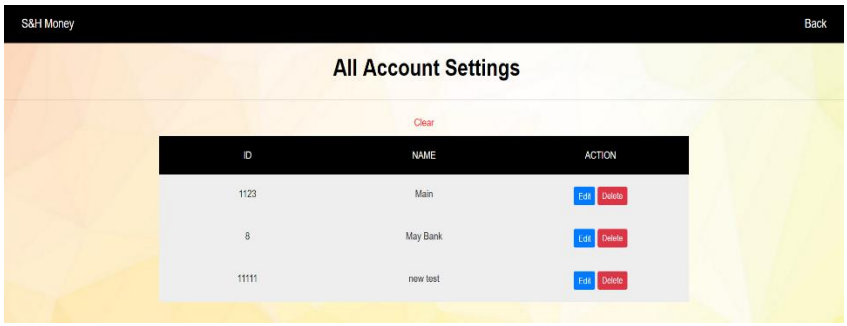
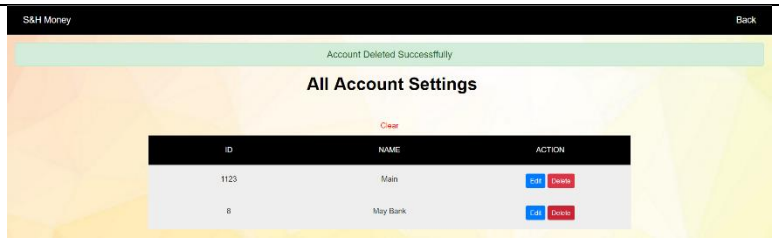
Test Case Name	Invalid format of starting balance
Test Case Description	User presses the *Update* button with an invalid format of starting balance.
Input	
Results	
Status (Pass/Fail)	Pass

Table 6.16 Test Case: Duplication account ID

Test Case Name	Duplication account ID
Test Case Description	User presses the *Update* button with an account name that is existing in the system.
Input	
Results	
Status (Pass/Fail)	Pass

View Account Activity:

Table 6.17 Test Case: Delete account

Test Case Name	Delete an account
Test Case Description	User presses a particular *Delete* button in the record list.
Input	
Results	
Status (Pass/Fail)	Pass

6.2.3 Income and Expenses Module

In this subsection, basic testing of the Income and Expenses Management module is performed. The test cases are described as four activities in this module: add income and expenses, update income and expenses, add category, and update category activity.

Add Income and Expenses Activity:

Table 6.18 Test Case: Empty Input Fields

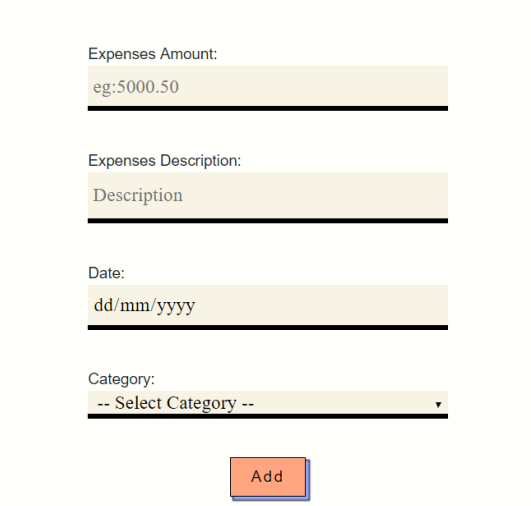
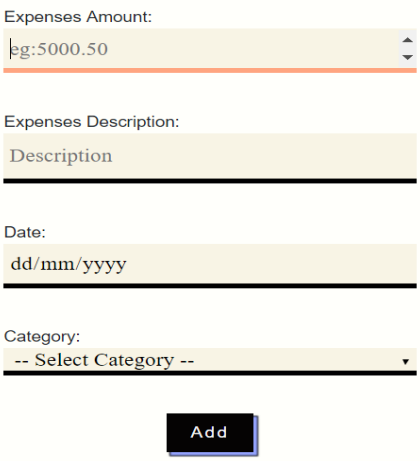
Test Case Name	Empty Input Fields
Test Case Description	User presses the *Add* button without filling in the information.
Input	
Results	
Status (Pass/Fail)	Pass

Table 6.19 Test Case: Invalid format of expense amount

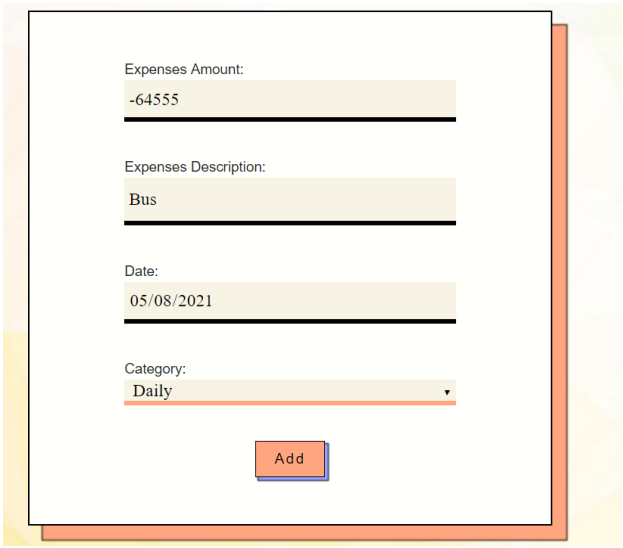
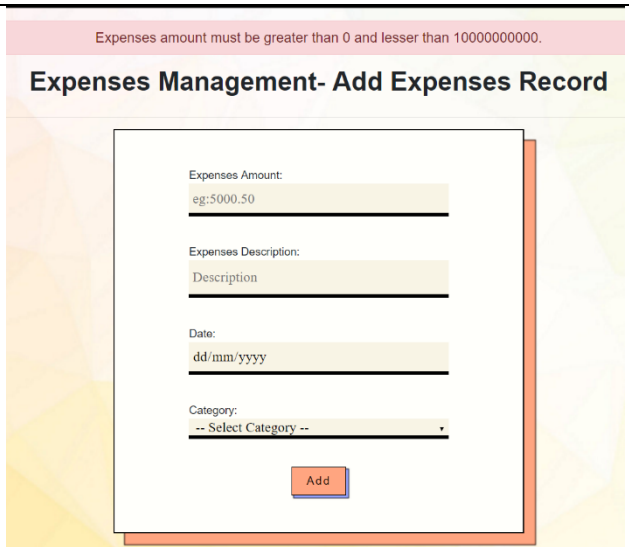
Test Case Name	Invalid format of expense amount
Test Case Description	User presses the *Add* button with an invalid format of amount.
Input	
Results	
Status (Pass/Fail)	Pass

Table 6.20 Test Case: Expense without any categories.

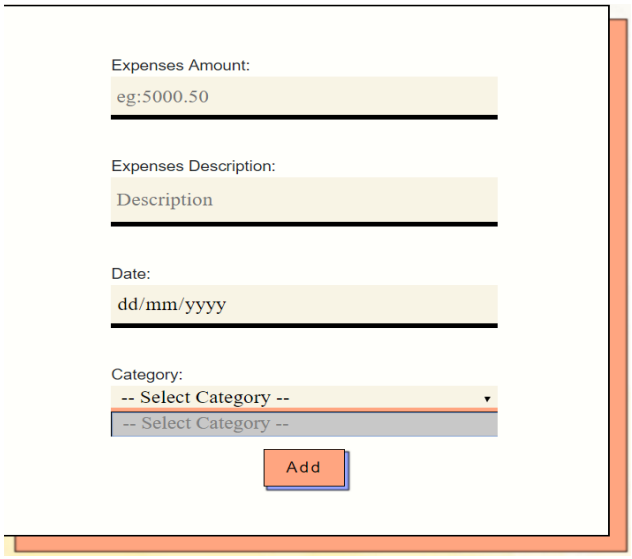
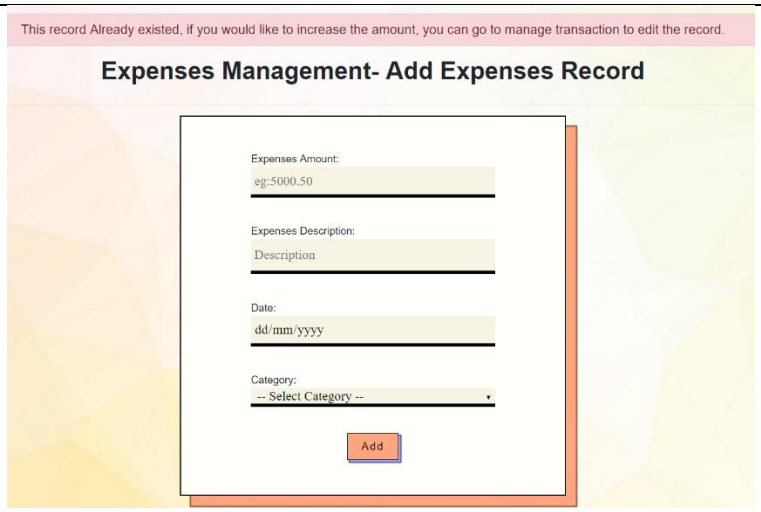
Test Case Name	Expense without any categories.
Test Case Description	User adds an income or expense without any categories.
Input	
Results	
Status (Pass/Fail)	Pass

Table 6.21 Test Case: Income or Expenses that is existing in the system

Test Case Name	Income or Expenses that is existing in the system
Test Case Description	User presses the *Add* button with an income or expense that is existing in the system.

Input	
Results	
Status (Pass/Fail)	Pass

Update Income and Expenses Activity:

Table 6.22 Test Case: Empty input fields for update income and expense

Test Case Name	Empty input fields for update income and expense
Test Case Description	User presses the *Update* button without filling in all the required information.

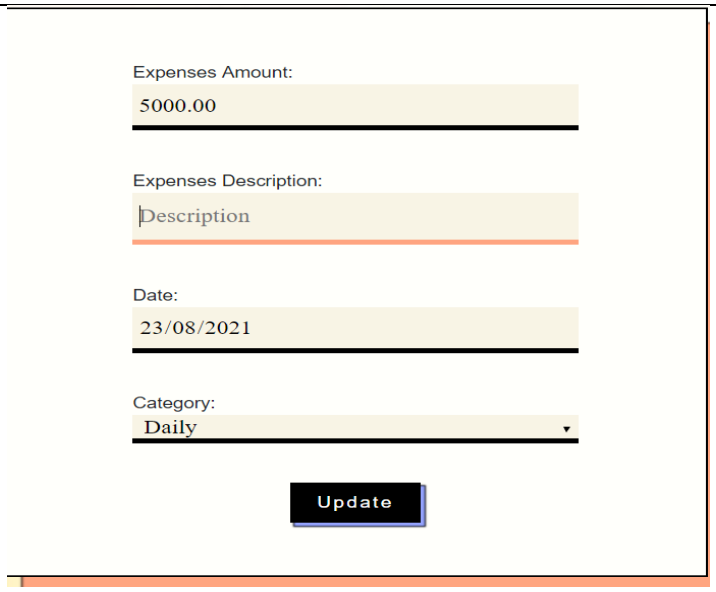
Input	
Results	
Status (Pass/Fail)	Pass

Table 6.23 Test Case: Invalid format of income and expense amount

Test Case Name	Invalid format of income and expense amount
Test Case Description	User presses the *Update* button with an invalid format of the amount.

Input	
Results	
Status (Pass/Fail)	Pass

Table 6.24 Test Case: Update duplication income and expense

Test Case Name	Update duplication income and expense
Test Case Description	User presses the *Update* button with an income or transaction that is existing in the system.

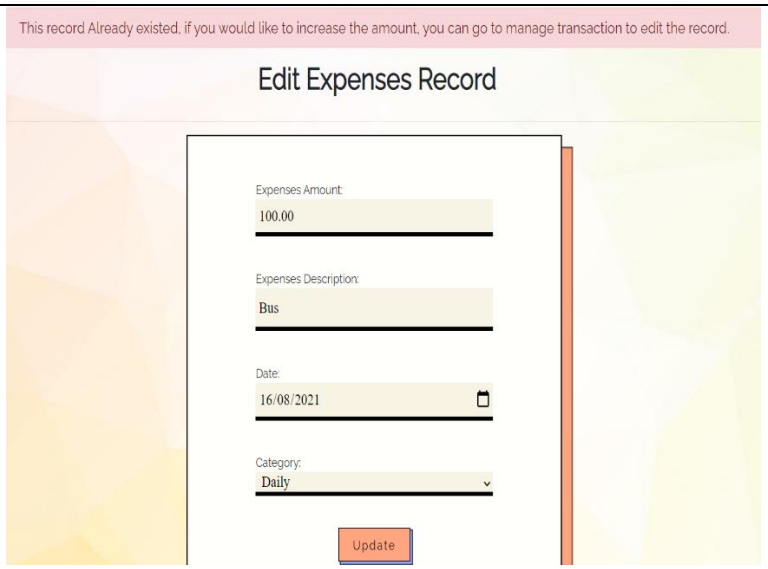
Input	
Results	
Status (Pass/Fail)	Pass

Table 6.25 Test Case: Delete income and expense

Test Case Name	Delete income and expense
Test Case Description	User presses the *Delete* button in the record list.

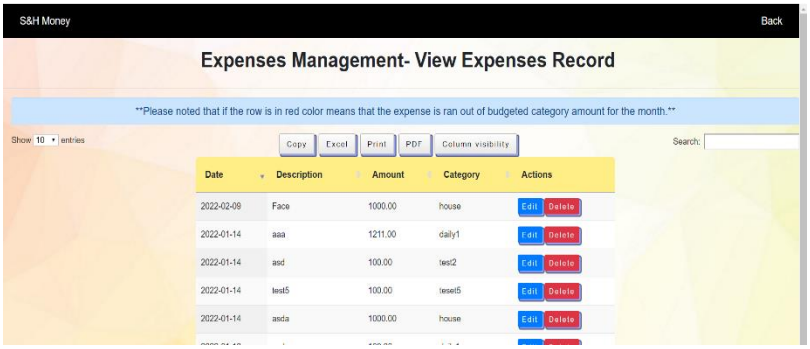
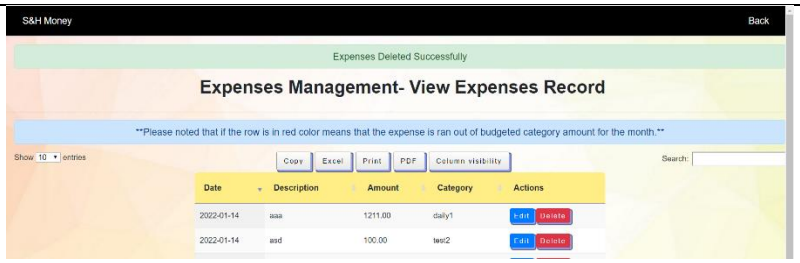
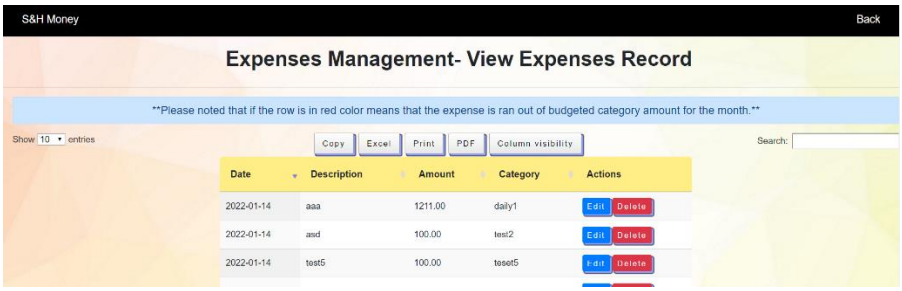
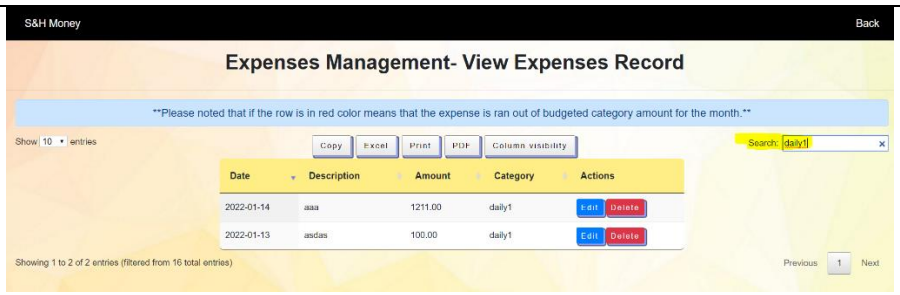
Input	
Results	
Status (Pass/Fail)	Pass

Table 6.26 Test Case: Search income and expense

Test Case Name	Search income and expense
Test Case Description	User key in keyword to the search box to search record
Input	
Results	
Status (Pass/Fail)	Pass

Add Category Activity:

Table 6.27 Test Case: Empty input fields for adding new category

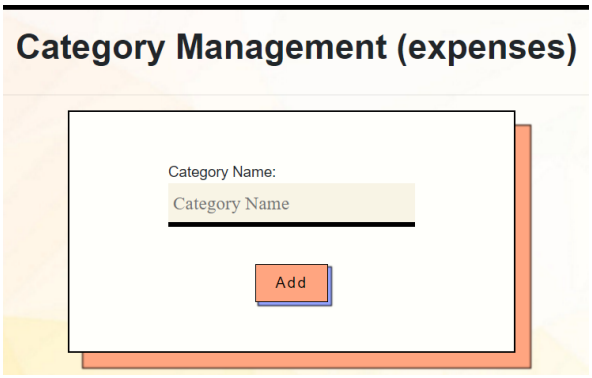
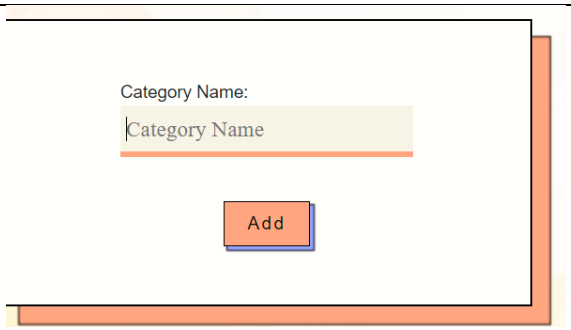
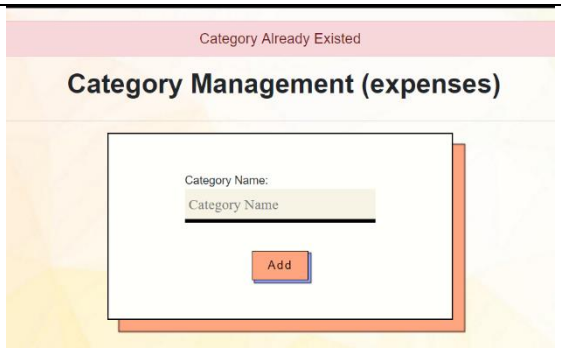
Test Case Name	Empty input fields for adding new category
Test Case Description	User presses the *Add* button without filling in the information.
Input	
Results	
Status (Pass/Fail)	Pass

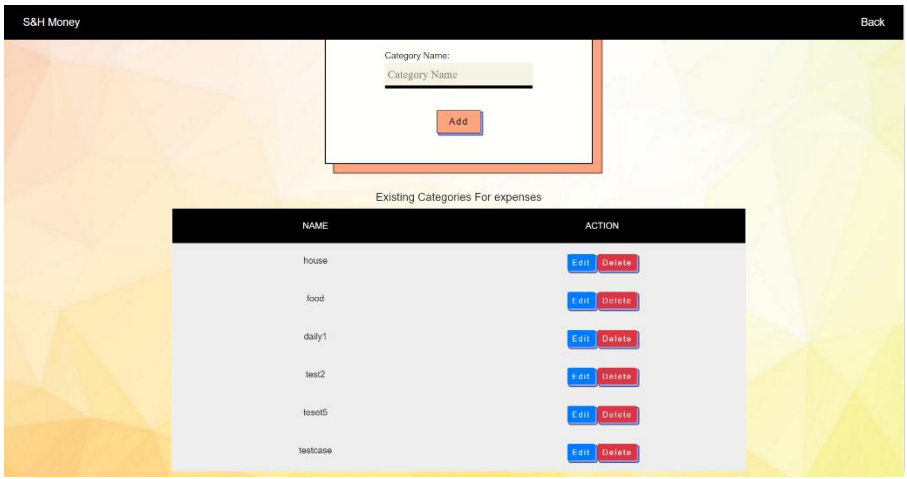
Table 6.28 Test Case: Duplication of category name

Test Case Name	Duplication of category name
Test Case Description	User presses the *Add* button with a category name that is existing in the system.

Input	
Results	
Status (Pass/Fail)	Pass

View Category Activity:

Table 6.29 Test Case: Delete a category which was being used

Test Case Name	Delete a category which was being used
Test Case Description	User presses a particular *Delete* button to delete a category is being used in the applcaition.
Input	

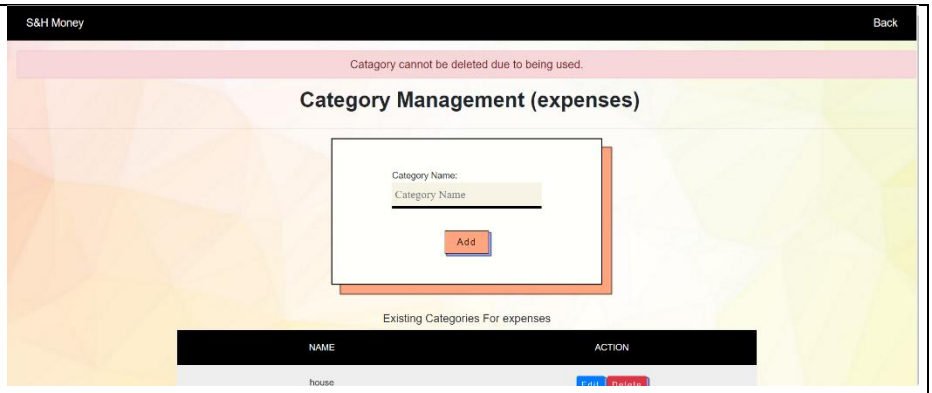
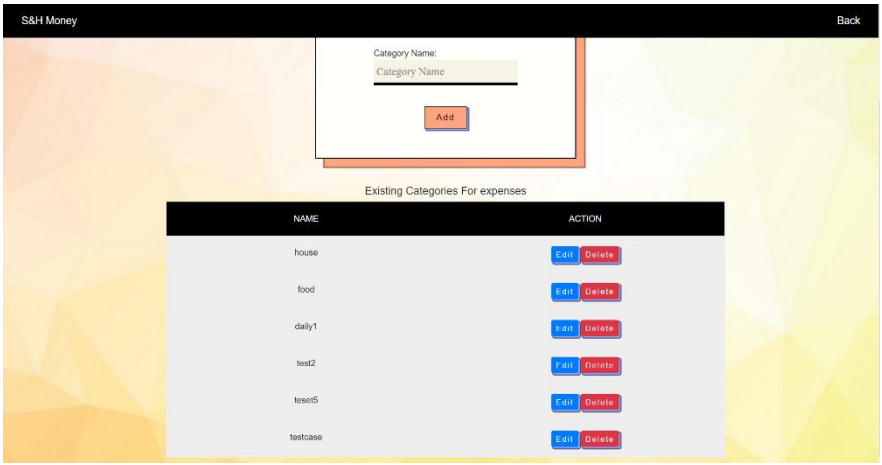
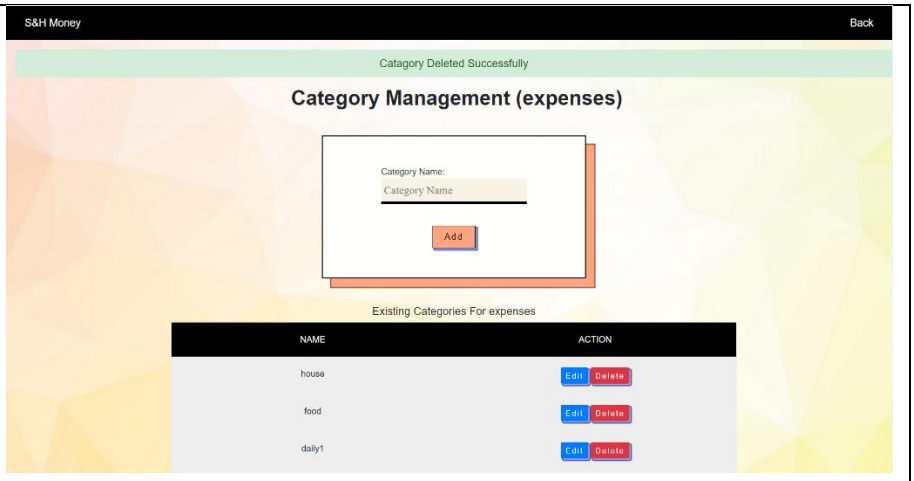
Results	
Status (Pass/Fail)	Pass

Table 6.30 Test Case: Delete a category

Test Case Name	Delete a category
Test Case Description	User presses a particular *Delete* button in the record list.
Input	

Results	
Status (Pass/Fail)	Pass

Update Category Activity:

Table 6.31 Test Case: Empty input fields for updating new category

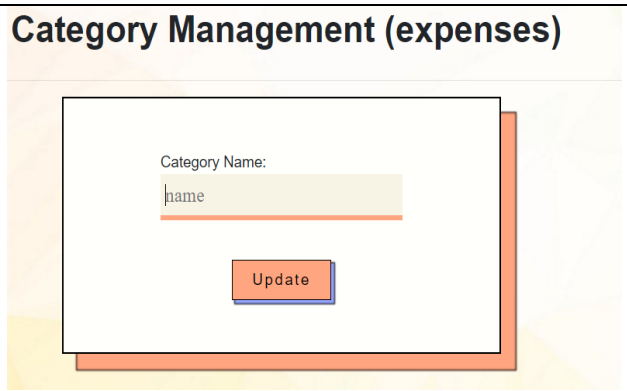
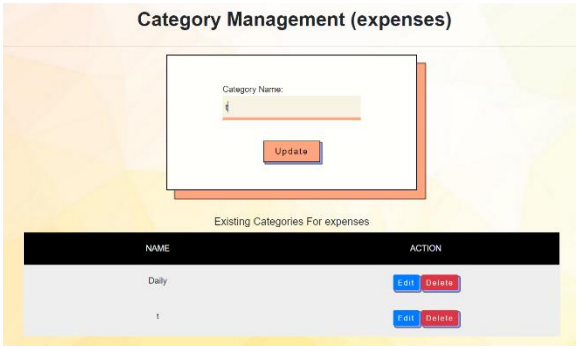
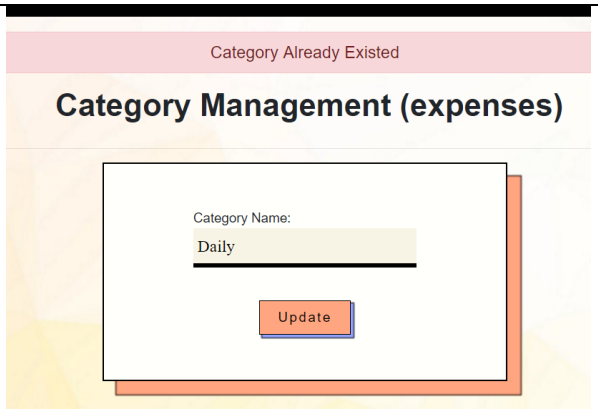
Test Case Name	Empty input fields for updating new category
Test Case Description	User presses the *Update* button without filling in all the required information.
Input	
Results	
Status (Pass/Fail)	Pass

Table 6.32 Test Case: Duplication of updating category name

Test Case Name	Duplication of updating category name
Test Case Description	User presses the *Update* button with a category name that is existing in the system.
Input	
Results	
Status (Pass/Fail)	Pass

6.2.4 Personal Budgeting

In this subsection, basic testing of the Personal Budgeting module is performed. The test cases are described to four activities in this module, namely, add account budget plan, update account budget plan, add category-budget plan, update category-budget plan.

Add Account Budget Plan:

Table 6.33 Test Case: Empty input fields for adding account budget

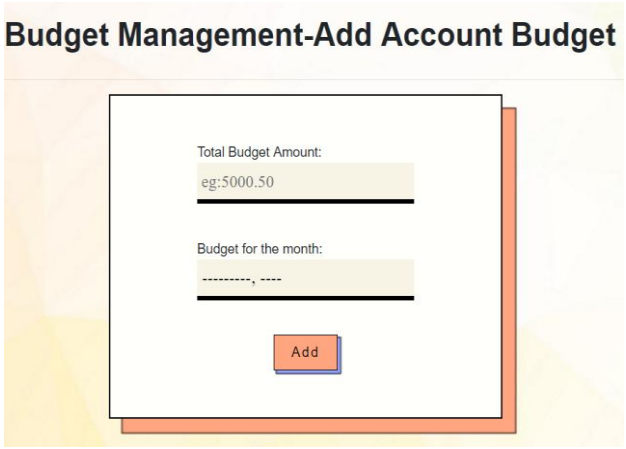
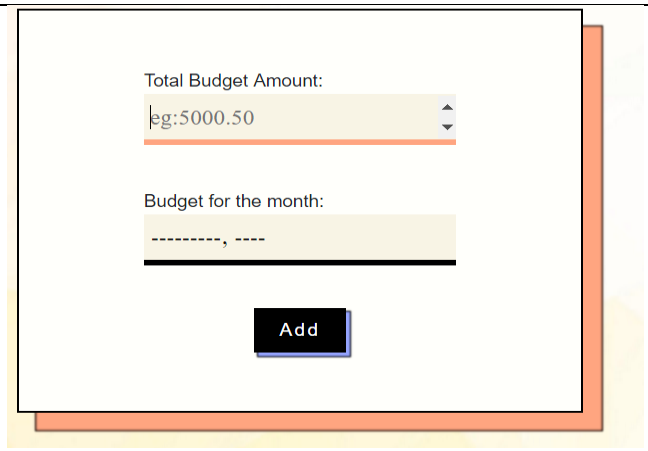
Test Case Name	Empty input fields for adding account budget
Test Case Description	User presses the *Add* button without filling in the information.
Input	
Results	
Status (Pass/Fail)	Pass

Table 6.34 Test Case: Invalid amount for adding account budget

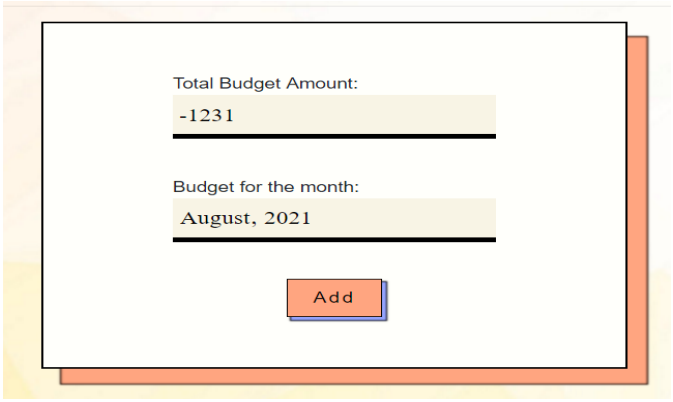
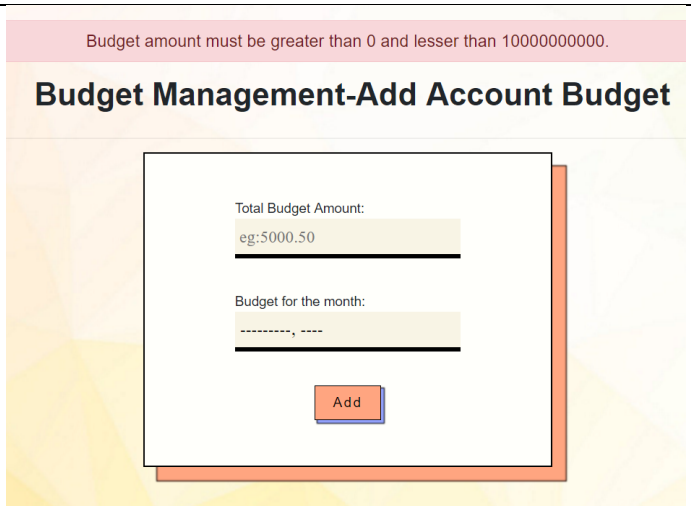
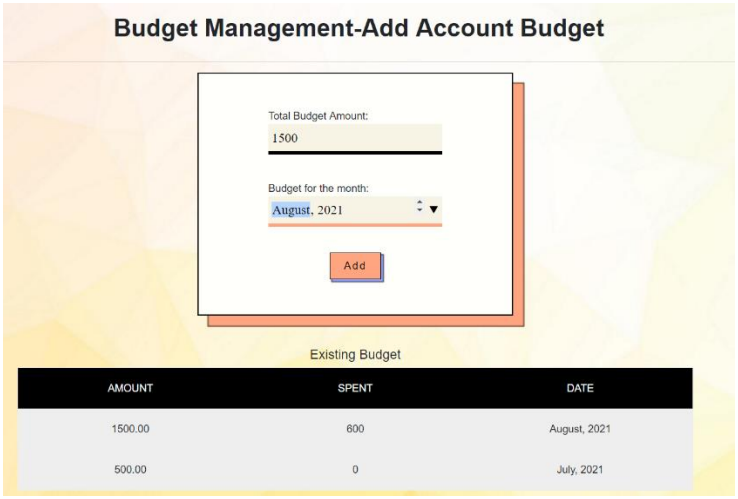
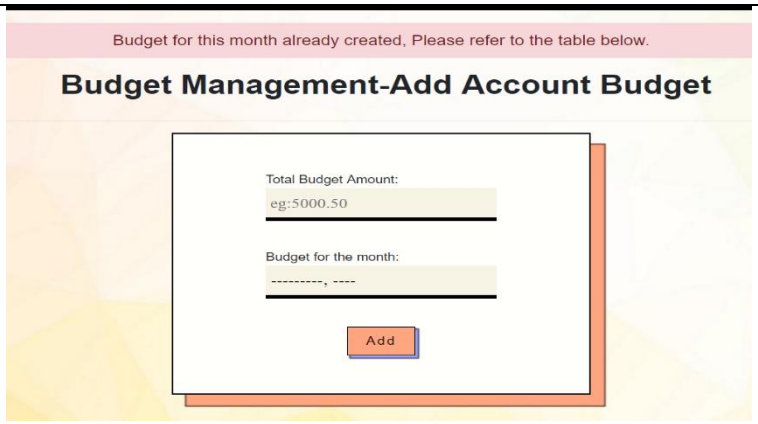
Test Case Name	Invalid amount for adding account budget
Test Case Description	User presses the *Add* button with an invalid format of amount.
Input	
Results	
Status (Pass/Fail)	Pass

Table 6.35 Test Case: Duplication for adding account budget

Test Case Name	Duplication for adding account budget
Test Case Description	User presses the *Add* button with an account budget that is existing in the system.

Input	
Results	
Status (Pass/Fail)	Pass

View Account Budget Plan Activity:

Table 6.36 Test Case: Delete an account budget plan

Test Case Name	Delete an account budget plan.
Test Case Description	User presses a particular *Delete* button in the record list.

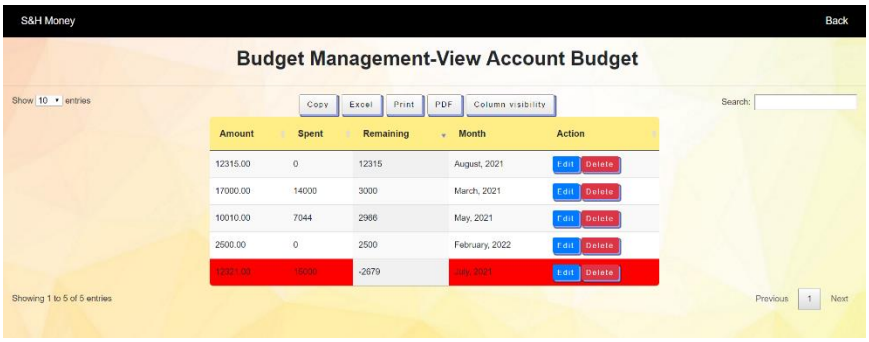
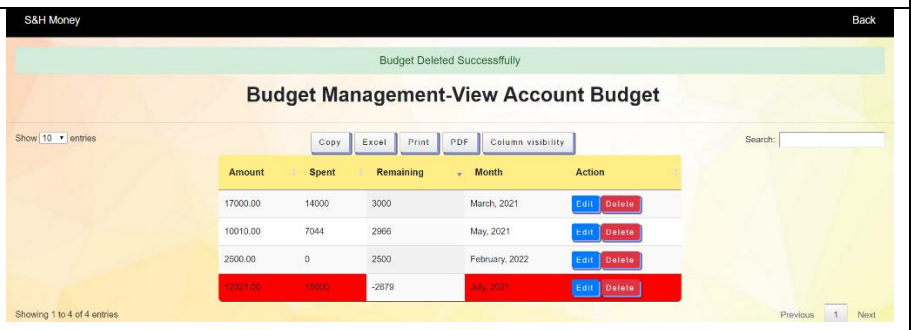
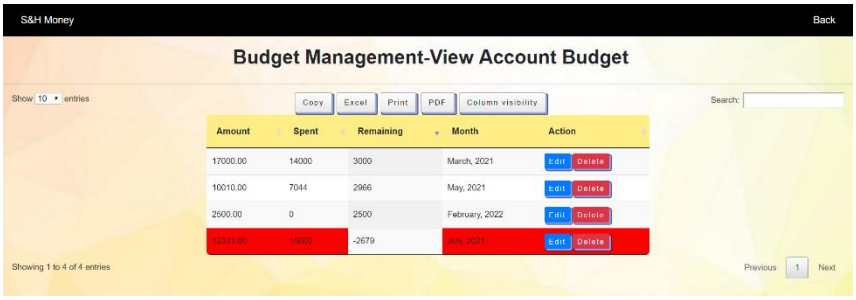
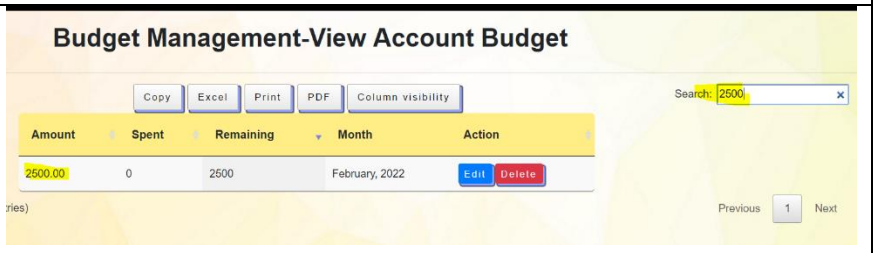
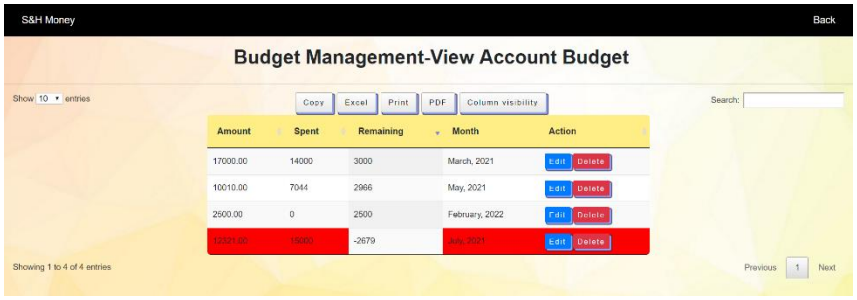
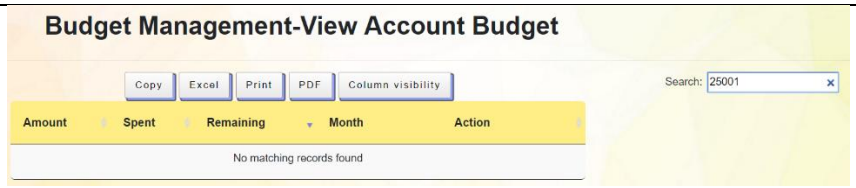
Input	
Results	
Status (Pass/Fail)	Pass

Table 6.37 Test Case: Search account budget plan

Test Case Name	Search account budget plan.
Test Case Description	User key in keyword to the search input fields.
Input	
Results	

Status (Pass/Fail)	Pass

Table 6.38 Test Case: Search not existing account budget plan

Test Case Name	Search not existing account budget plan.
Test Case Description	User key in keyword to the search input fields.
Input	
Results	
Status (Pass/Fail)	Pass

Update Account Budget Plan:

Table 6.39 Test Case: Empty input fields for updating account budget plan

Test Case Name	Empty input fields for updating account budget plan
Test Case Description	User presses the *Update* button without filling in the information.

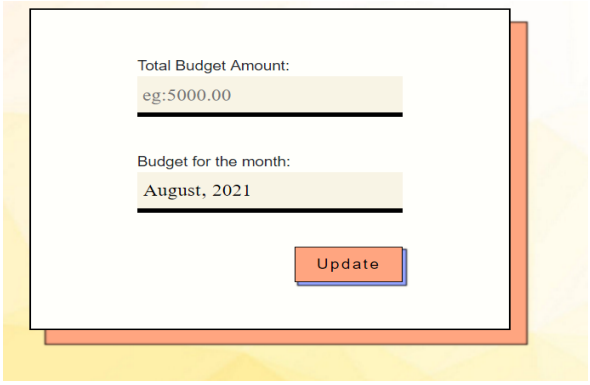
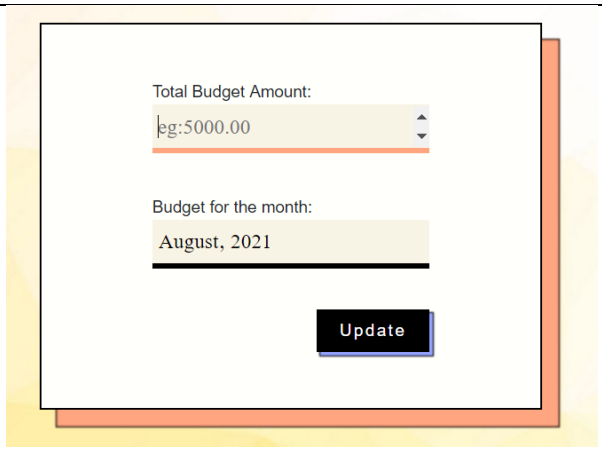
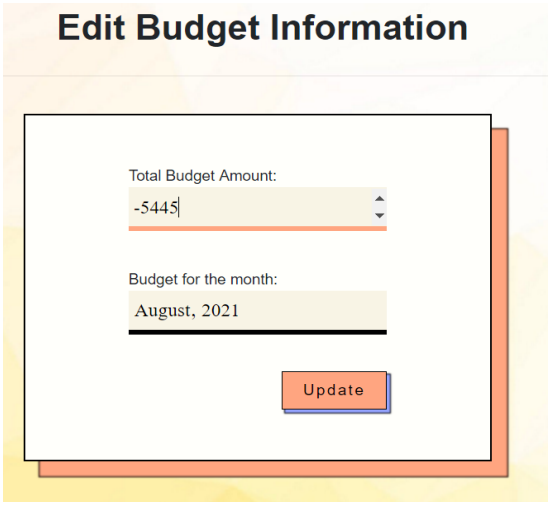
Input	
Results	
Status (Pass/Fail)	Pass

Table 6.40 Test Case: Invalid amount for updating account budget plan

Test Case Name	Invalid amount for updating account budget plan
Test Case Description	User presses the *Update* button with an invalid format of amount and date.
Input	

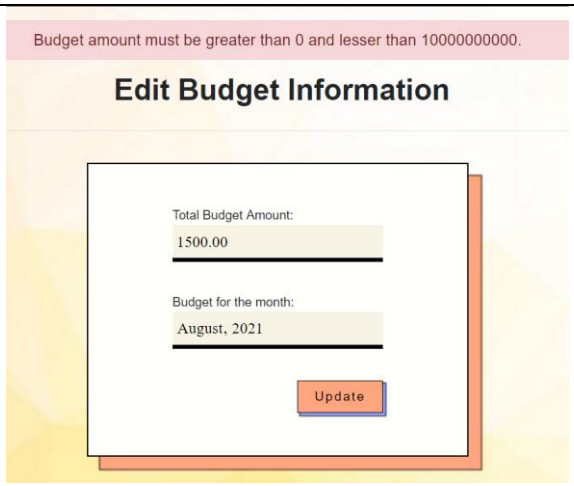
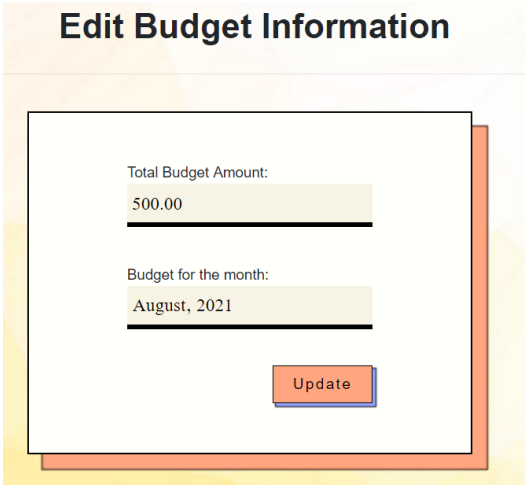
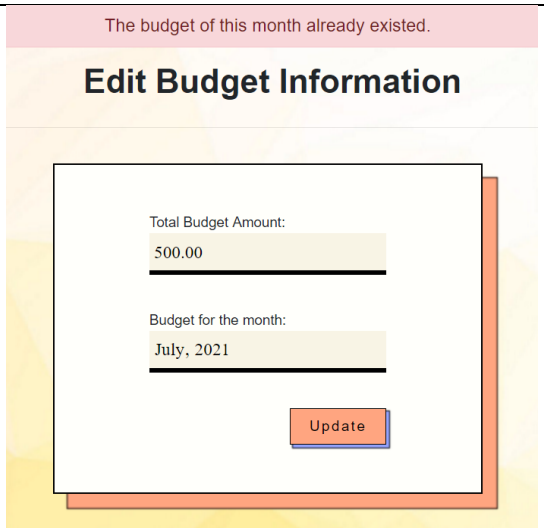
Results	
Status (Pass/Fail)	Pass

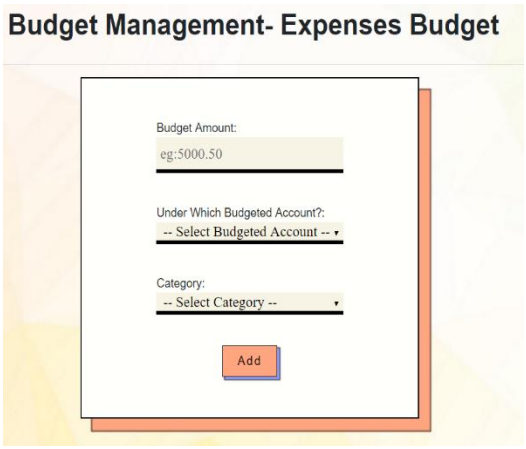
Table 6.41 Test Case: Duplication for updating account budget plan

Test Case Name	Duplication for updating account budget plan
Test Case Description	User presses the *Update* button with a category budget that is existing in the system.
Input	

Results	
Status (Pass/Fail)	Pass

Add Category Budget Plan:

Table 6.42 Test Case: Empty input fields for adding category budget

Test Case Name	Empty input fields for adding category budget
Test Case Description	User presses the *Add* button without filling in the information.
Input	

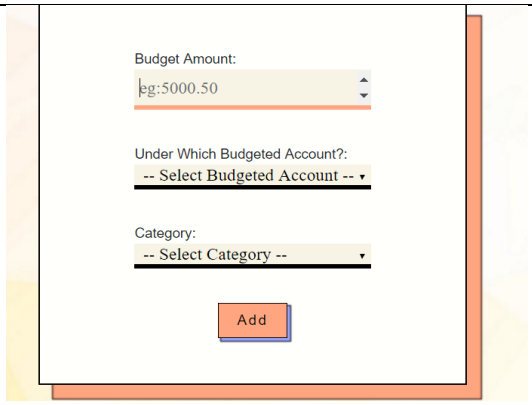
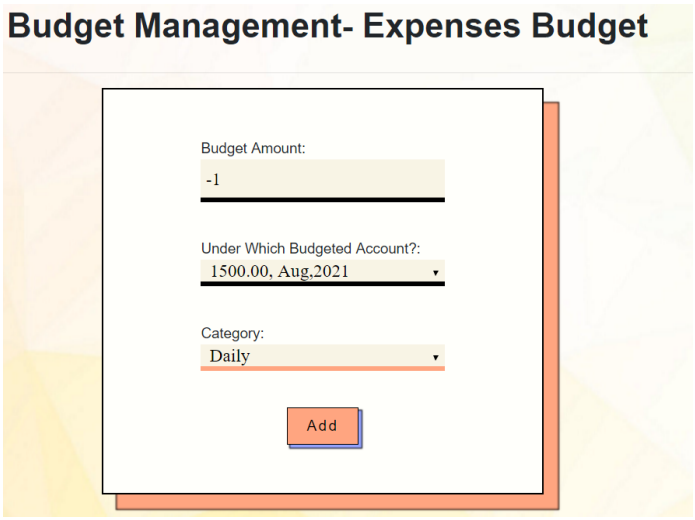
Results	
Status (Pass/Fail)	Pass

Table 6.43 Test Case: Invalid amount for adding category budget

Test Case Name	Invalid amount for adding category budget
Test Case Description	User presses the *Add* button with an invalid format of amount.
Input	

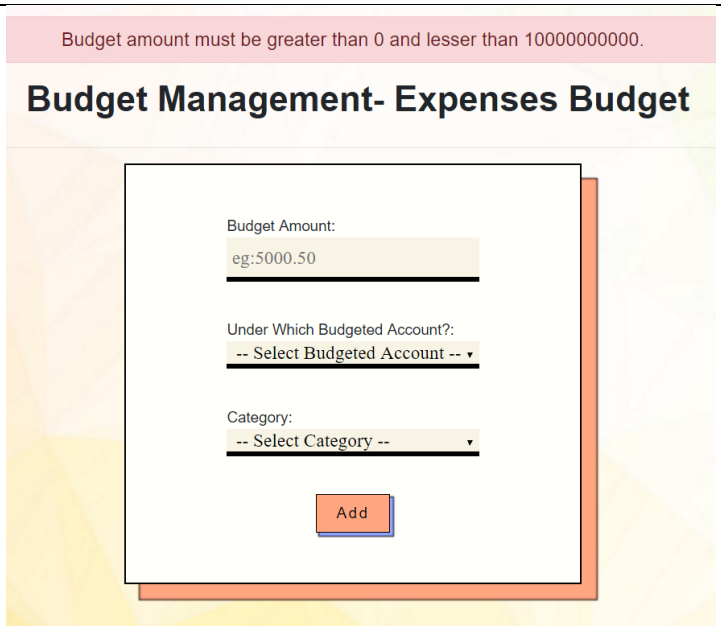
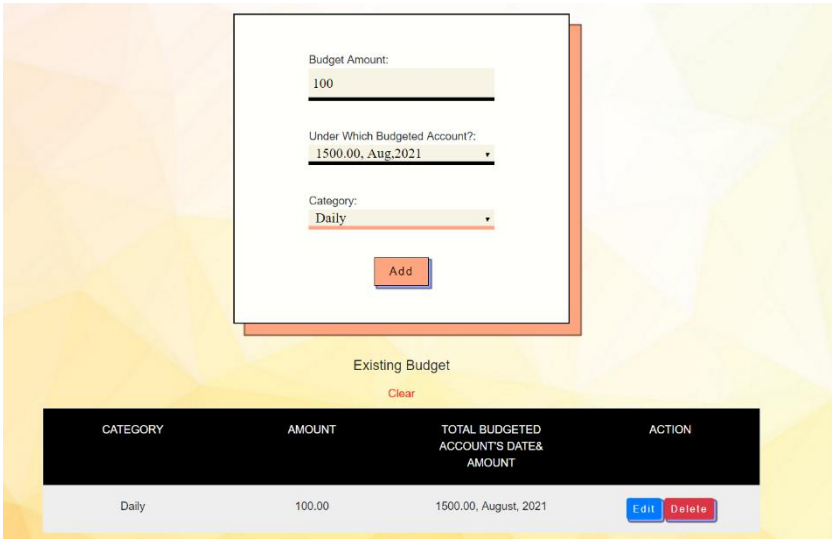
Results	
Status (Pass/Fail)	Pass

Table 6.44 Test Case: Duplication for adding category budget

Test Case Name	Duplication for adding category budget
Test Case Description	User presses the *Add* button with a category budget that is existing in the system.
Input	

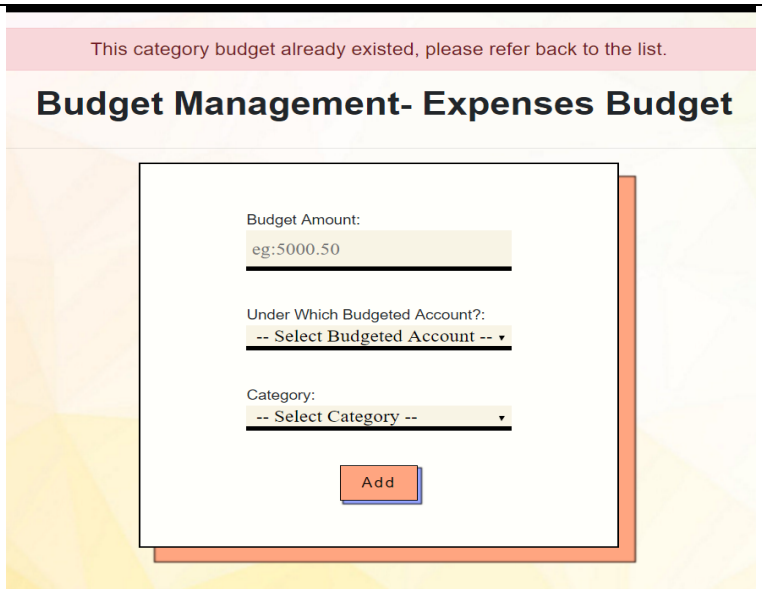
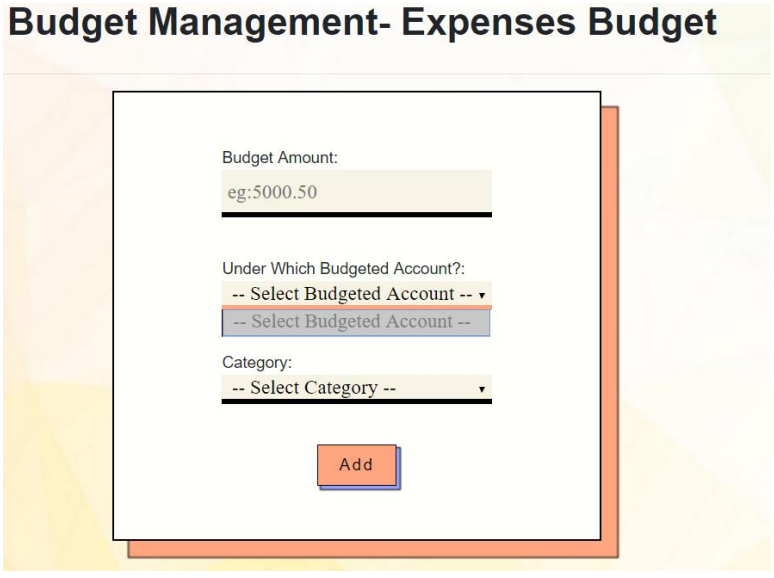
Results	
Status (Pass/Fail)	Pass

Table 6.45 Test Case: Without any account budget

Test Case Name	Without any account budget
Test Case Description	User view the add page without any account budget.
Input	

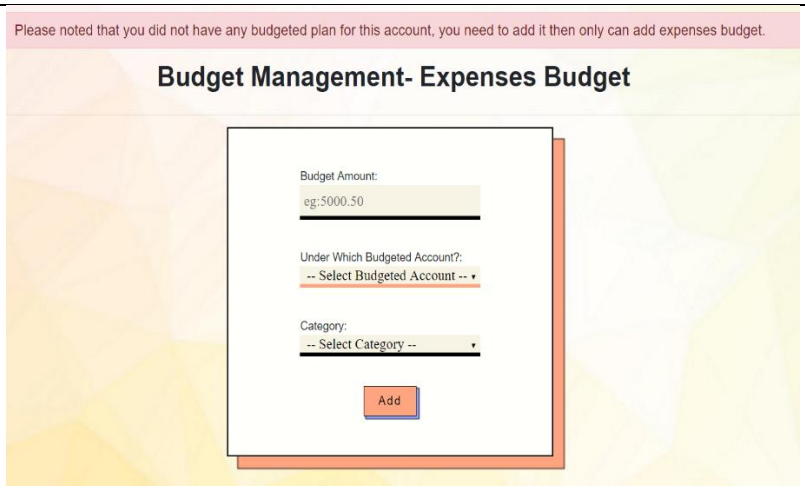
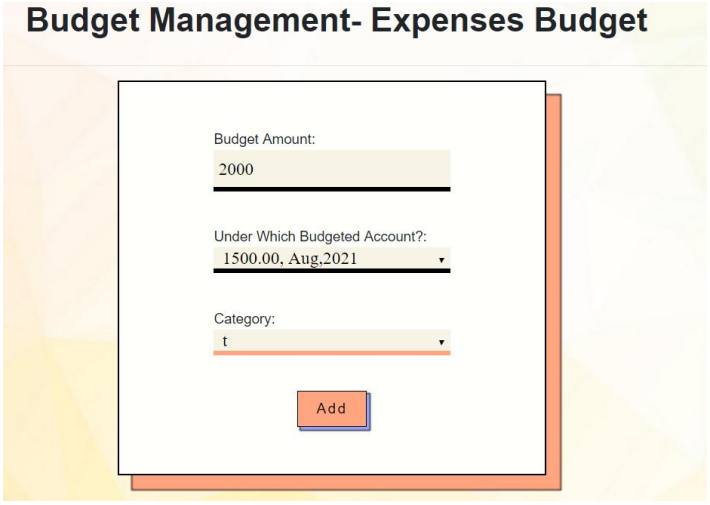
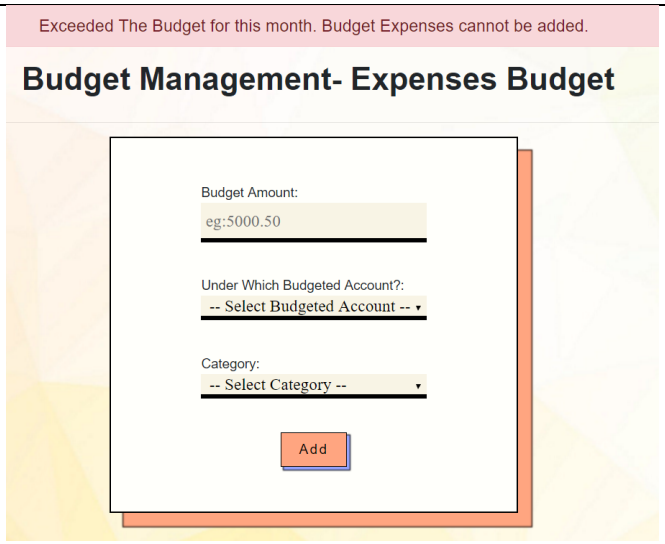
Results	
Status (Pass/Fail)	Pass

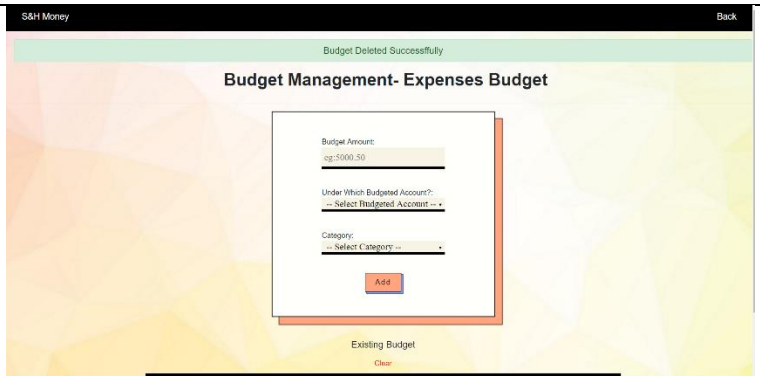
Table 6.46 Test Case: Amount exceed account budget amount

Test Case Name	Category budget amount exceed account budget amount
Test Case Description	User presses the *Add* button with the amount exceed account budget amount.
Input	

Results	
Status (Pass/Fail)	Pass

View Category Budget Plan:

Table 6.47 Test Case: Delete a category budget plan

Test Case Name	Delete a category budget plan.
Test Case Description	User presses a particular *Delete* button in the record list.
Input	
Results	
Status (Pass/Fail)	Pass

Update Category Budget Plan:

Table 6.48 Test Case: Empty input fields for updating

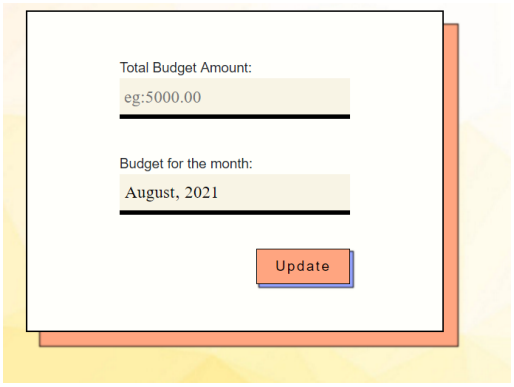
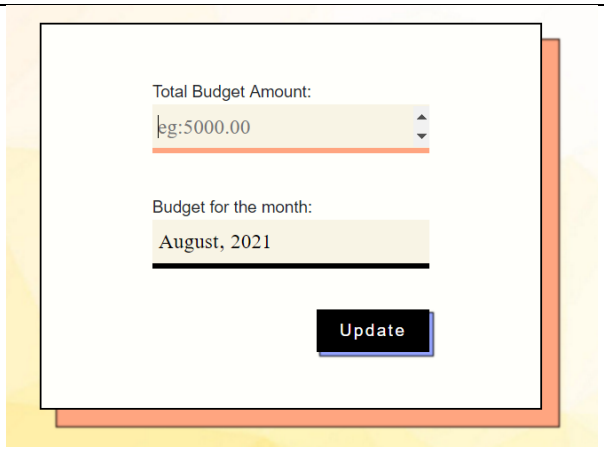
Test Case Name	Empty input fields for updating category budget plan
Test Case Description	User presses the *Update* button without filling in the information.
Input	
Results	
Status (Pass/Fail)	Pass

Table 6.49 Test Case: Invalid amount for updating account budget plan

Test Case Name	Invalid amount for updating account budget plan
Test Case Description	User presses the *Update* button with an invalid format of amount and date.

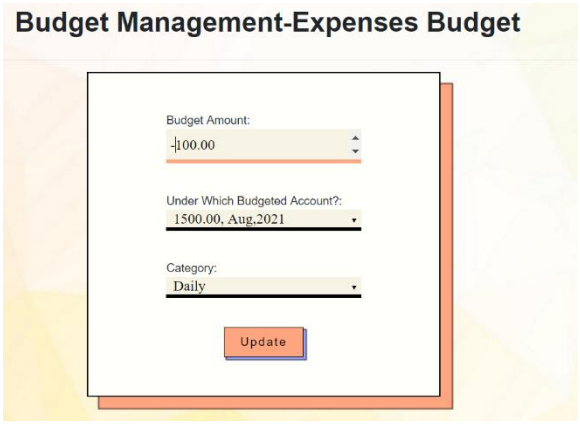
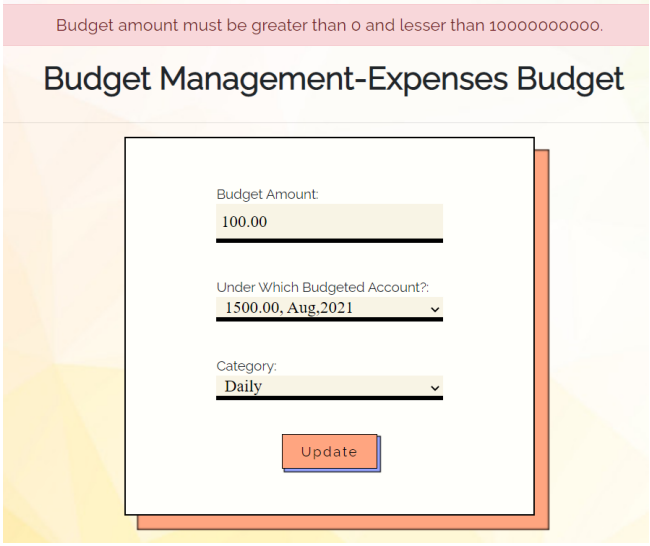
Input	
Results	
Status (Pass/Fail)	Pass

Table 6.50 Test Case: Duplication for updating category budget plan

Test Case Name	Duplication for updating category budget plan
Test Case Description	User presses the *Update* button with a category budget that is existing in the system.

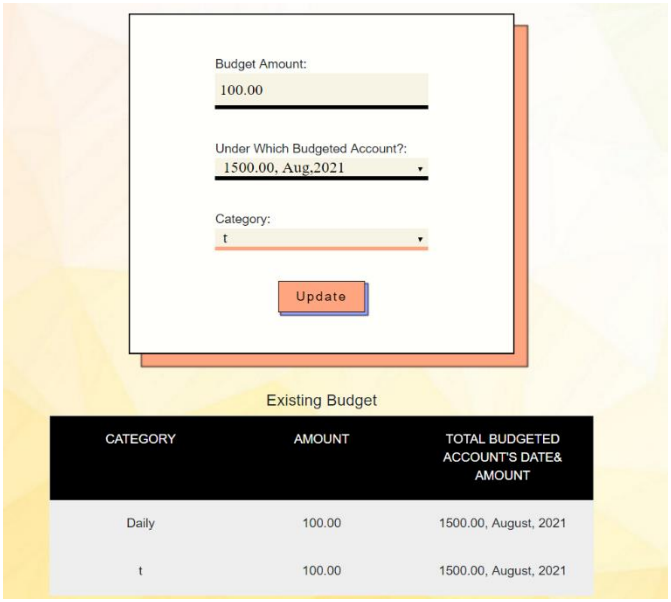
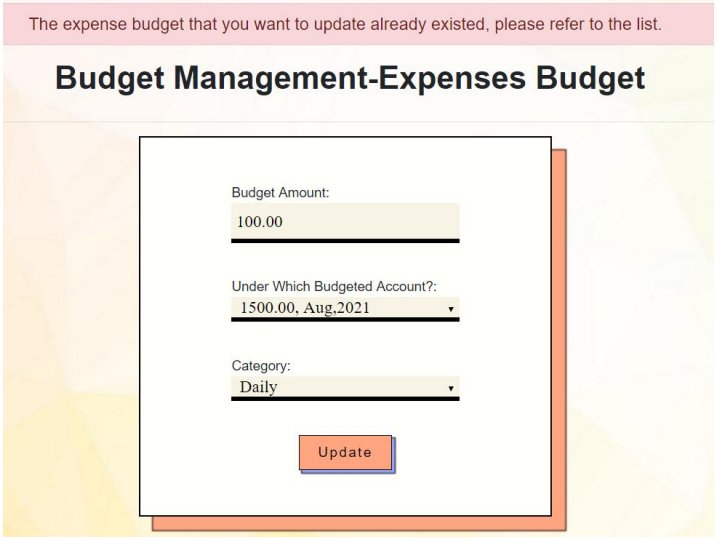
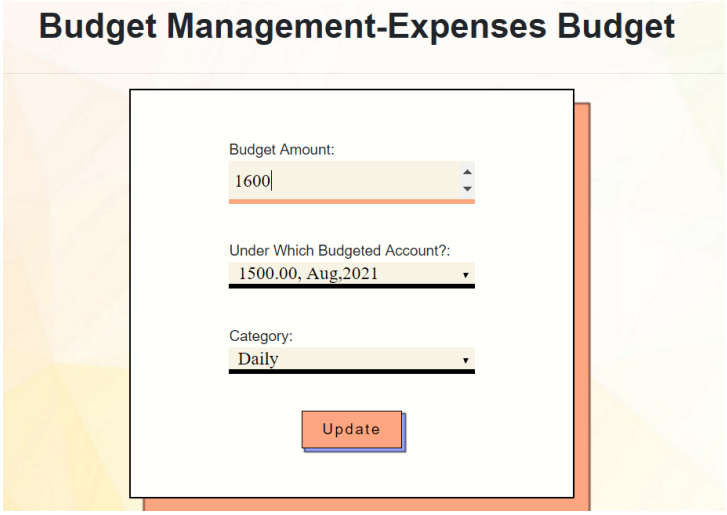
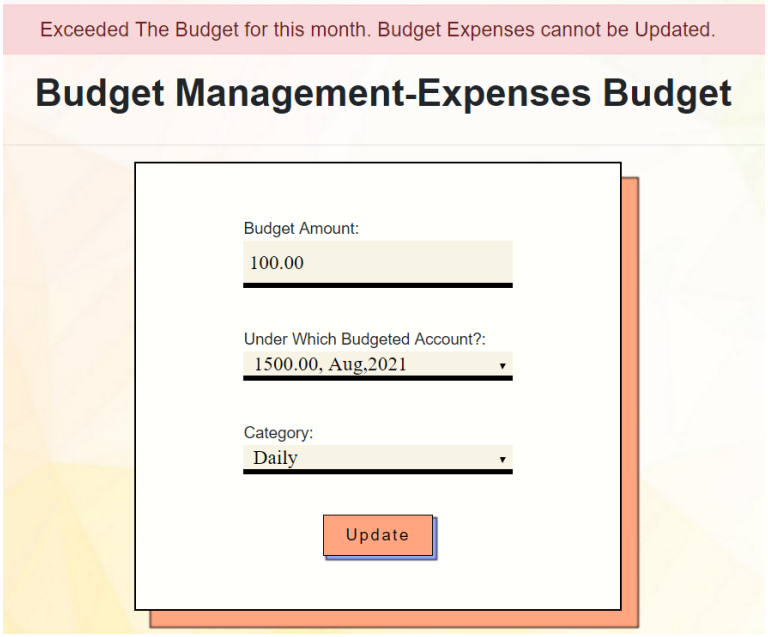
Input	
Results	
Status (Pass/Fail)	Pass

Table 6.51 Test Case: Amount exceed account budget amount

Test Case Name	Amounts exceed account budget amount
Test Case Description	User presses the *Update* button with a category budget amount that is exceed account budget amount.

Input	
Results	
Status (Pass/Fail)	Pass

6.2.5 Report Management Module

In this subsection, basic testing of the Generate Report module is performed. The test cases are described in the following table.

Table 6.52 Test Case: Generate Report without any records in PDF format

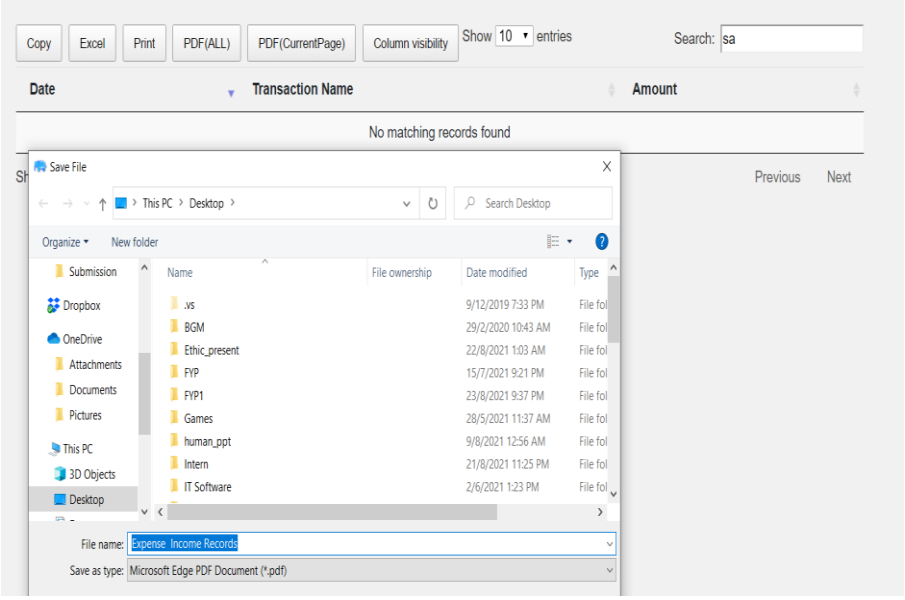
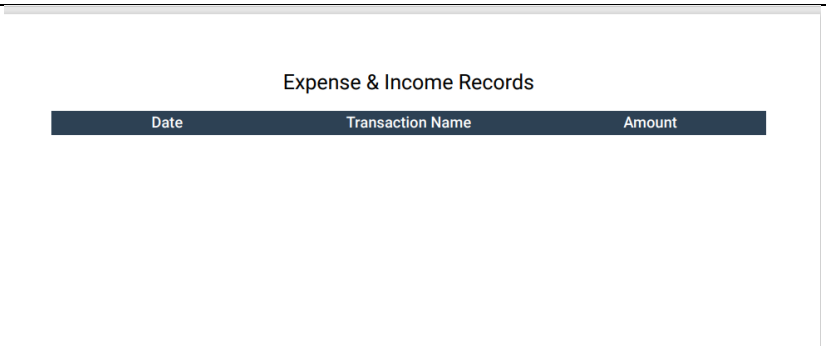
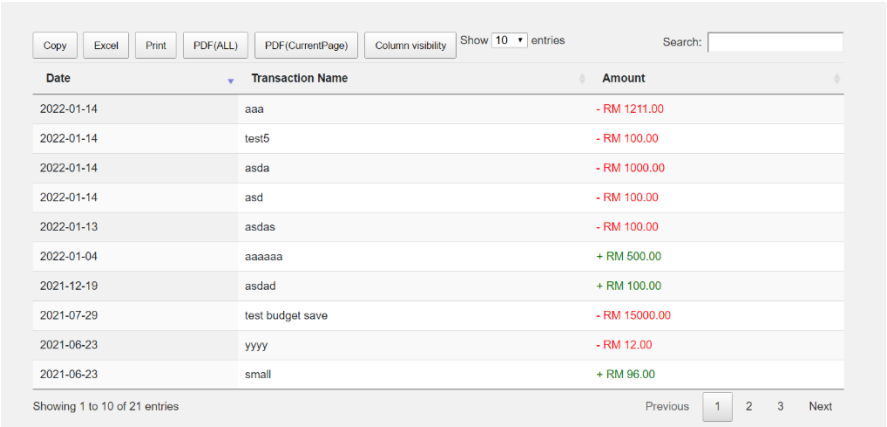
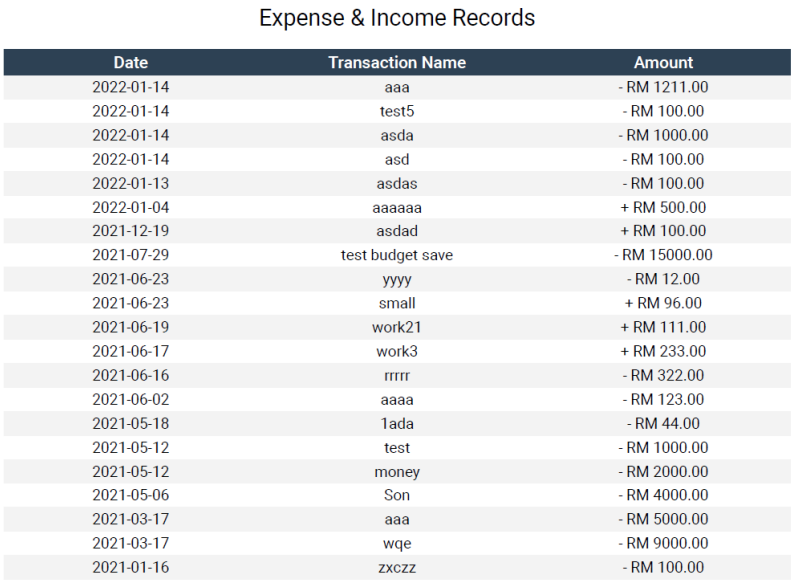
Test Case Name	Generate Report without any records
Test Case Description	User presses the *PDF(ALL)* button without any records.
Input	
Results	
Status (Pass/Fail)	Pass

Table 6.53 Test Case: Generate Report with records in PDF format

Test Case Name	Generate Report with any records
Test Case Description	User presses the *PDF(ALL)* button with some records.
Input	
Results	
Status (Pass/Fail)	Pass

Expense & Income Records

Date	Transaction Name	Amount
2022-01-14	aaa	- RM 1211.00
2022-01-14	test5	- RM 100.00
2022-01-14	asda	- RM 1000.00
2022-01-14	asd	- RM 100.00
2022-01-13	asdas	- RM 100.00
2022-01-04	aaaaaa	+ RM 500.00
2021-12-19	asdad	+ RM 100.00
2021-07-29	test budget save	- RM 15000.00
2021-06-23	yyyy	- RM 12.00
2021-06-23	small	+ RM 96.00
2021-06-19	work21	+ RM 111.00
2021-06-17	work3	+ RM 233.00
2021-06-16	rrrrr	- RM 322.00
2021-06-02	aaaa	- RM 123.00
2021-05-18	1ada	- RM 44.00
2021-05-12	test	- RM 1000.00
2021-05-12	money	- RM 2000.00
2021-05-06	Son	- RM 4000.00
2021-03-17	aaa	- RM 5000.00
2021-03-17	wqe	- RM 9000.00
2021-01-16	zxczz	- RM 100.00

Table 6.54 Test Case: Generate Report with records in current page

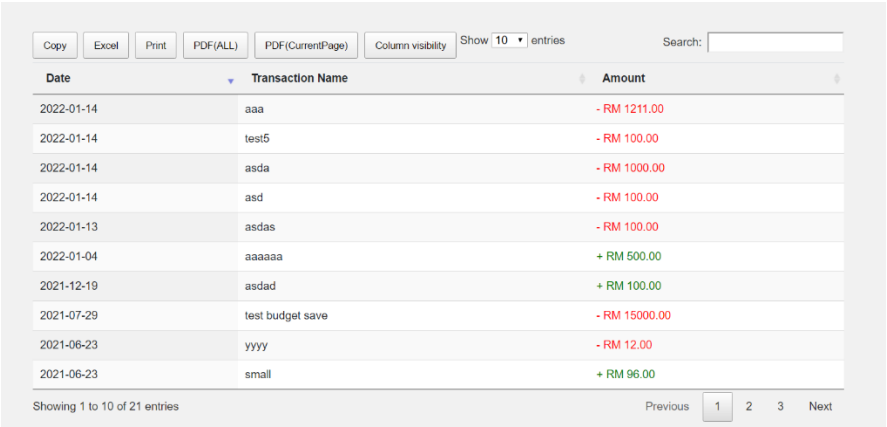
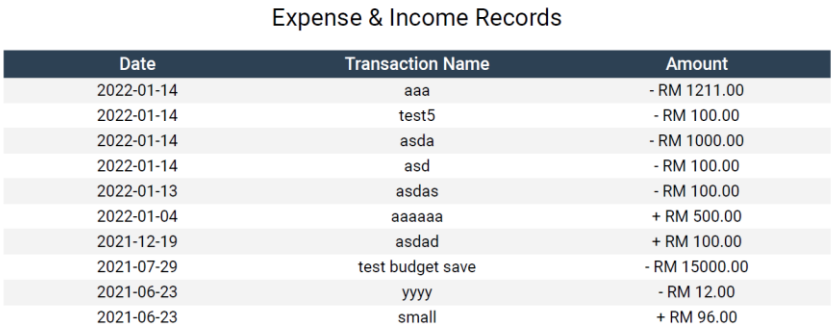
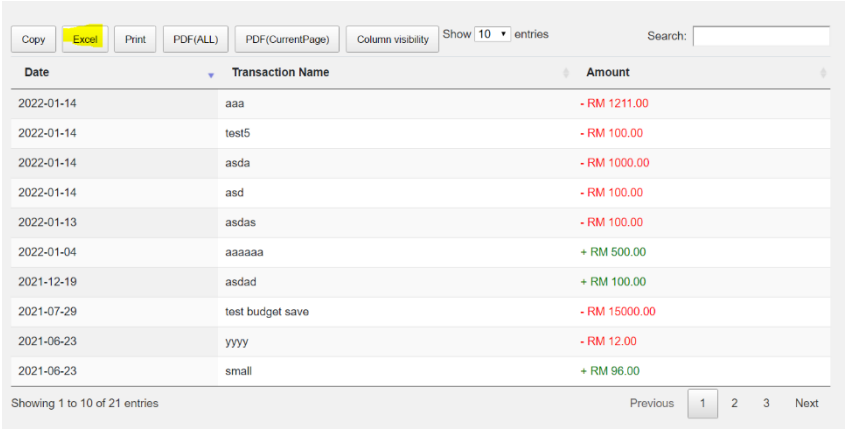
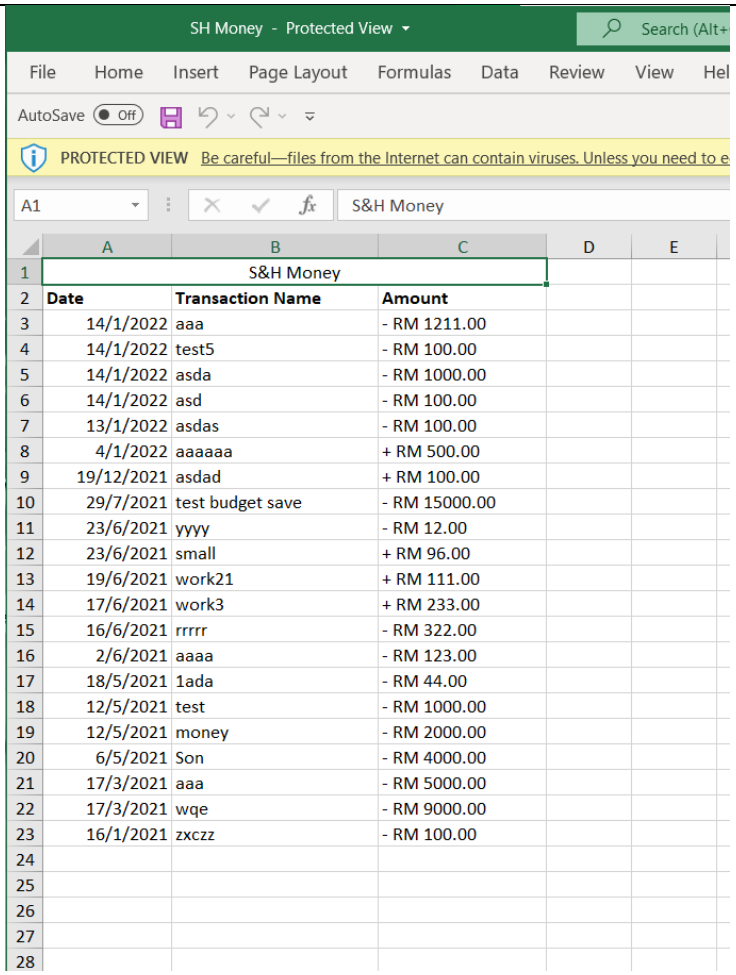
Test Case Name	Generate Report for current page in PDF
Test Case Description	User presses the *PDF(Current)* button to generate pdf.
Input	
Results	
Status (Pass/Fail)	Pass

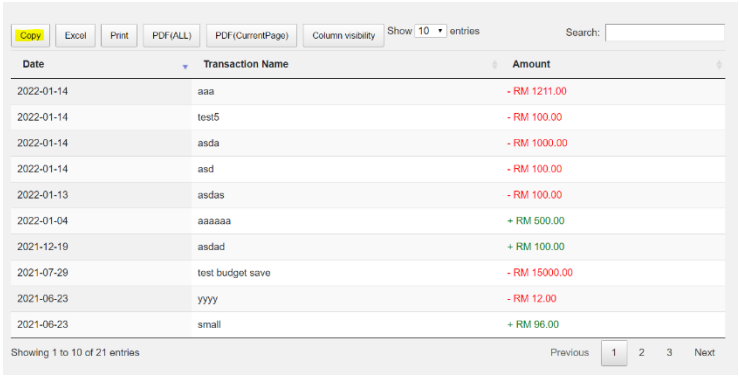
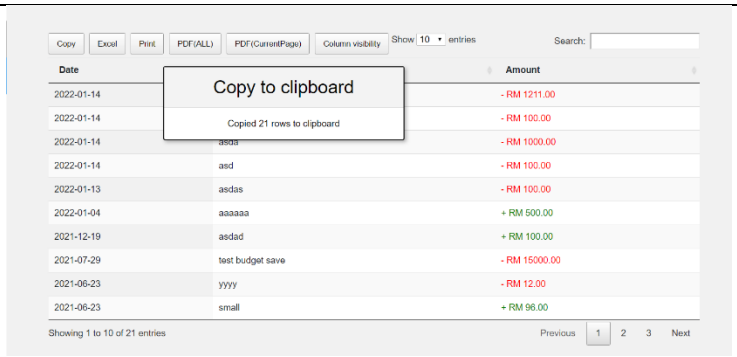
Table 6.55 Test Case: Generate Report with records in Excel format

Test Case Name	Generate Report with records in Excel format
Test Case Description	User presses the *Excel* button to export the data.

Input	
Results	
Status (Pass/Fail)	Pass

	A	B	C	D	E
1		S&H Money			
2	**Date**	**Transaction Name**	**Amount**		
3	14/1/2022	aaa	- RM 1211.00		
4	14/1/2022	test5	- RM 100.00		
5	14/1/2022	asda	- RM 1000.00		
6	14/1/2022	asd	- RM 100.00		
7	13/1/2022	asdas	- RM 100.00		
8	4/1/2022	aaaaaa	+ RM 500.00		
9	19/12/2021	asdad	+ RM 100.00		
10	29/7/2021	test budget save	- RM 15000.00		
11	23/6/2021	yyyy	- RM 12.00		
12	23/6/2021	small	+ RM 96.00		
13	19/6/2021	work21	+ RM 111.00		
14	17/6/2021	work3	+ RM 233.00		
15	16/6/2021	rrrrr	- RM 322.00		
16	2/6/2021	aaaa	- RM 123.00		
17	18/5/2021	1ada	- RM 44.00		
18	12/5/2021	test	- RM 1000.00		
19	12/5/2021	money	- RM 2000.00		
20	6/5/2021	Son	- RM 4000.00		
21	17/3/2021	aaa	- RM 5000.00		
22	17/3/2021	wqe	- RM 9000.00		
23	16/1/2021	zxczz	- RM 100.00		
24					
25					
26					
27					
28					

Table 6.56 Test Case: Copy record to clipboard

Test Case Name	Copy record to clipboard
Test Case Description	User presses the *Copy* button to use the data.
Input	
Results	 
Status (Pass/Fail)	Pass

6.2.6 Graph and Chart Generation Module

In this section, basic testing of the Graph and Chart Generation module is performed. The test cases are described in the following table.

Table 6.57 Test Case: View the generate chart page without data

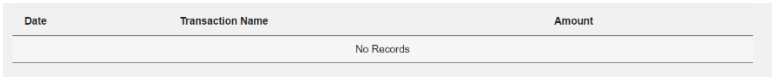
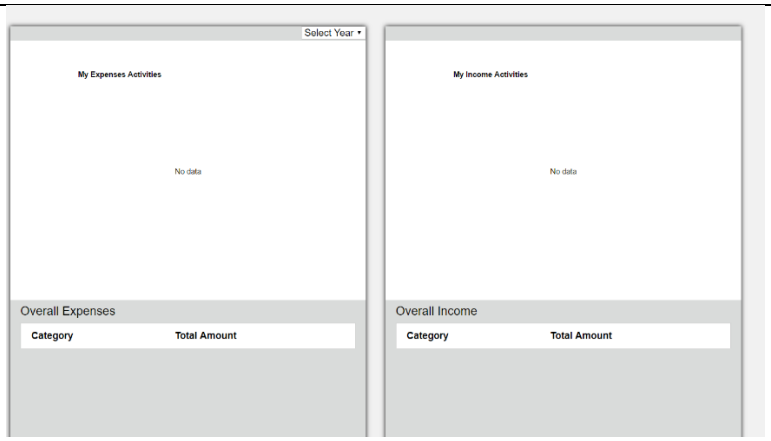
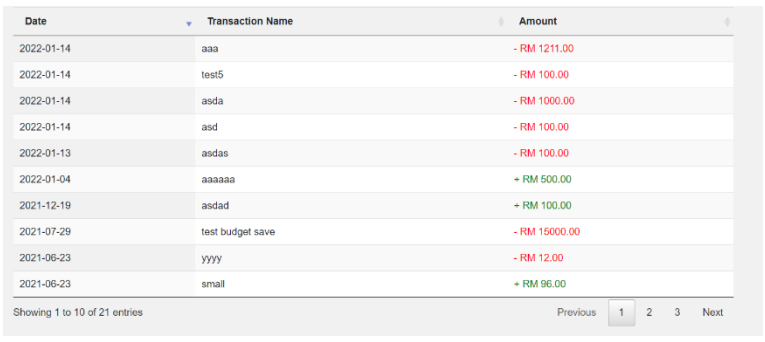
Test Case Name	View the generate chart page without data
Test Case Description	User views the pages that needed graph and chart generation.
Input	
Results	
Status (Pass/Fail)	Pass

Table 6.58 Test Case: View the generate chart page with data

Test Case Name	View the generate chart page with data
Test Case Description	User views the pages that needed graph and chart generation.
Input	

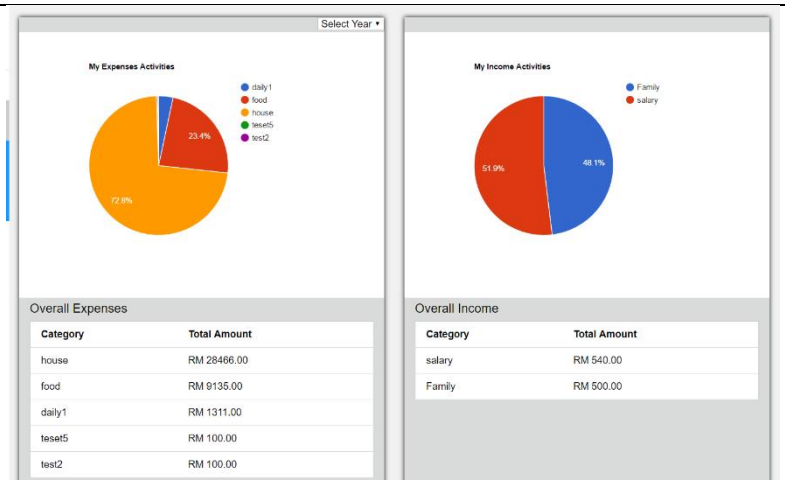
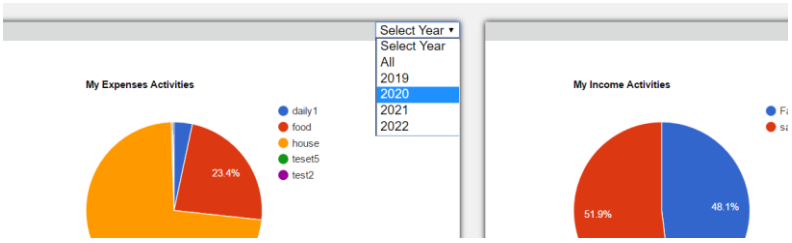
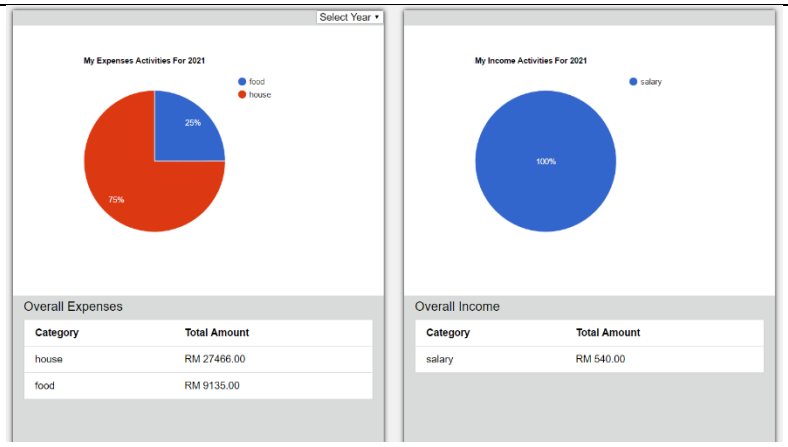
Results	
Status (Pass/Fail)	Pass

Table 6.59 Test Case: Generate chart for specific year

Test Case Name	Generate chart for specific year
Test Case Description	User selects a year to filter the graph and chart.
Input	
Results	
Status (Pass/Fail)	Pass

6.3 Project Challenges

One of the difficulties in the implementation of this system lies in the process of recording the personal finance data. Some of the university students do not have the concept on how personal finance management works. Generally, most university students think that personal finance management involves only adding transactions and processing calculations based on the entered values. However, personal finance management covers many aspects, such as account management, transaction management, budget planning and so on. In order to cater to all university students, including those that might not have experience in managing personal finance, a challenge lies in how to guide the university students to familiarise themselves with the this develope applications. For example, in this application, the basic flow of managing one's personal finance is first to create an account, then to add transactions or budget planning records. This flow has to be hinted to the user in the event that they do not know their way around this application.

Furthermore, another difficulty in the implementation of this application system is related to budget planning. In this application, budget planning is divided into two parts – account budget planning and category budget planning. The main challenge here is that university students may be confused with the functionalities presented to them and may not be able to utilize this application to its full potential. For example, the general flow of performing budget planning is to first add an account budget plan, then a category budget, and so on. The former and the latter are related. However, due to the lack of understanding of this concept, university students may straight away add the category budget planning without having an account budget plan beforehand, which is prohibited by this developed application system. Therefore, the lack of knowledge of how this application system works poses as the biggest challenge in this implementation.

6.4 Objective Evaluation

Below are the three main objectives of this project for reference:

1. To identify the problems of manual paper-and-pen transaction management associated with personal finance.
2. To design a prototype that is useful to improve the personal finance management of university students.
3. To develop a software prototype for university students with personal budgeting and transaction tracking functionalities.

Based on chapter 5 system implementation and chapter 6 system evaluation, the objectives of this project have been achieved. For the first objective, the problem has been identified through the chapter 1 problem statement section. In that section, the content has explained the true that manual paper-and-pen management for personal finance is able to let the user's personal finance record in low security and vulnerable to physical wear and tear. In order to achieve this objective, this developed application has come out to help university student manage their personal finance by using the laptop or PC. Therefore, the first objective has been achieved.

Moving on, for the second objective, this application can make sure that the university student is able to easily manage their personal finance. The reason is because this application provided clear and user-friendly design to automatically guide the university student how to use this application. For example, if the university student is the first time to experience this application, this developed application is able to guide the user to register an user, after that perform login, create account and start managing their personal finance by adding the income and expenses record. A success and an error message will be displayed if the user has done some wrong or correct action. Therefore, the second objective has been achieved.

Last but not the least, the objective states that this application has to develop personal budgeting and transaction tracking functionalities. During the system implementation and system evaluation, this application is able to provide these functionalities to let university students add income and expenses as well as personal budgeting plan. In addition, instead of having whole account personal budgeting plan, this application also is able to provide category budget plan, where university student can be based on the account budget plan amount to assign one or many categories

budget plan. This functionality is able to help university student to manage small amount and portion of personal budgeting plan. Therefore, the third objective in this project has been achieved as well.

6.5 Concluding Remark

In a nutshell, the functionalities of the six modules in this application – User Management, Account Management, Income & Expenses Management, Personal Budgeting, Generate Report and Generate Chart and Graph- have been tested with many different scenarios. The scenarios included valid and invalid data input format and application flows. All of these scenarios is tested without any problems and can ensure that this application is able to handle it. Furthermore, with the evidence of the system evaluation, all of the objectives are successfully achieved and explained in detail. In the next chapter, a summary of this project will be discussed,

CHAPTER 7

CONCLUSION

In this chapter, this project's report and application is summarized in terms of the project's objective, motivation, and developed methods. This chapter will be included the summary of this project and the future recommendation of this developed application.

7.1 Conclusion

In a nutshell, this application is called S&H Money. It aims to solve the manual pen-and-paper methods to manage personal finance for university students by using a desktop-based application.

Money is significant for every university student. The problem is that manual pen-and-paper methods are low security and vulnerable to physical wear and tear, and university students always forget where their money is going. Hence, with the motivation of helping university students to achieve their long-term and short-term personal financial goals, this developed application aims to develop a user-friendly prototype for university students with personal budgeting and transaction tracking functionalities.

After reviewing the previous and existing personal finance applications, the main functionalities discovered were CRUD for income and expense with category, budget planning, multiple accounts to manage personal finance, and the security login page. Therefore, all of these functionalities are included in this application. One of the functionalities that lacked in these existing applications is when the user arrived at the negative balance, these applications do not notify the user. Instead, the action of these applications did is to ignore the balance. Thus, in this project, the application has resolved this problem. With this personal finance management application, university students can better understand how to manage their personal finance.

This application is developed using HTML, CSS, and Bootstrap to design the front-end and PHP is used to handle the back-end functionalities to communicate with the database, MySQL. In addition, Firebase is used to support this application to perform authentication such as login and forgot password. This developed application consists of six modules: User Management module, Account Management module, Income and Expenses Management Module, Personal Budgeting module, Generate Report module and Generate Graph and Chart module. With the completed work, all the functionalities and UI designs for the main module, such as User Management, Account Management, Income and Expenses Management and Personal Budgeting, were finished and implemented. The basic flow of this application is, first, the university student has to register an user, and after perform login to access this application. Secondly, the user has to create an account in order to start managing their personal finance (Income, Expenses, Budgeting, Report and Chart) because the personal finance record will be based on the account to store and display.

In short, this application aims to help university students to achieve their personal financial goals in the future.

In the following section, the recommendations for this project are discussed.

7.2 Recommendation

In terms of recommendations of this project, there are some improvements can be implemented to this application in order to increase the experience of university student.

First, in terms of the user interface, the animation of the page can be added to improve the user experience. The reason is because currently this application only reload the page for filtering data, adding data, updating data, login and so on. It is possible to make the university student feel boring when managing the personal finance application. Thus, in the future, any kinds of design animation can be added to make the user interface more friendly.

Furthermore, the second recommendation will be added a page to provide some ideas to the university student for adding category in the income and expenses module. The reason is because sometimes the user might not be able to put some meaningful naming for category. For example, the user wants to create a category for house income but do not know how to name the category. In this case, a category recommendation page can be added to give user to refer and add the related category. With that, it will save time for the university student to manage their personal finance.

Moving on, since this application provided category budget plan, therefore, a improvement can be made for this functionalities, where providing a hyperlink to navigate university student to create category if there is not any categories existed in the user account. The reason why need to include this function is because to handle a case where user did not create their category on the income and expenses module and directly go to create the personal budgeting plan. By adding this function, the user can easily know how to successfully create a category budget plan.

Ultimately, the aim of this application is to help university student to manage their personal finance effectively and safely. With this application, the university student can easily plan for their future financial plan when they have been graduated from university.

REFERENCES

[1] J. Smith, “The Complete Guide to Money Management for College Students,” *Self*, 14-Nov-2018. [Online]. Available: https://www.self.inc/blog/money-management-for-college-students. [Accessed: 18-Mar-2022].

[2] A. A. Samsukha, “Personal finance mobile app development: Cost & key features,” *emizentech*, 11-Jan-2022. [Online]. Available: https://www.emizentech.com/blog/how-to-develop-a-personal-finance-management-app.html#Types_Of_Financial_Apps_Used_In_The_United_States_In_2018. [Accessed: 18-Mar-2022].

[3] “Personal budget software,” *Goodbudget*. [Online]. Available: https://Goodbudget.com/home. [Accessed: 18-Mar-2022].

[4] Mobills. [Online]. Available: https://web.mobillsapp.com/transactions. [Accessed: 18-Mar-2022].

[5] “Everydollar: Focus your money on what matters,” *EveryDollar.com*, 2019. [Online]. Available: https://www.everydollar.com/app/budget. [Accessed: 18-Mar-2022].

[6] “Moneyline Personal Finance Software,” *nchsoftware*. [Online]. Available: https://www.nchsoftware.com/personalfinance/index.html?kw=personal+finance+free+software&gclid=Cj0KCQiAkuP9BRCkARIsAKGLE8WHTqIXY18asgSwrSz1Ozg8QhMfDc_IFB8CcOQik142YMcAnbcUUp8aAvaMEALw_wcB. [Accessed: 18-Mar-2022].

[7] A. Ansong, “(PDF) Level of knowledge in personal finance by university freshmen business students.,” *ResearchGate*, Sep-2011. [Online]. Available: https://www.researchgate.net/profile/Abraham-Ansong/publication/267218367_Level_of_knowledge_in_personal_finance_by_university_freshmen_business_students/links/56e1593d08ae9b93f79e1b10/Level-of-knowledge-in-personal-finance-by-university-freshmen-business-students.pdf. [Accessed: 18-Mar-2022].

[8] L. Falahati, L. Paim, M. Ismail, and S. A. Haron, “Assessment of university students' financial management skills and educational needs,” *ResearchGate*, Aug-2011. [Online]. Available: https://www.researchgate.net/publication/266879186_Assessment_of_university_students'_financial_management_skills_and_educational_needs. [Accessed: 18-Mar-2022].

[9] L. Frazier, “5 reasons personal finance should be taught in school,” *Forbes*, 30-Jun-2021. [Online]. Available: https://www.forbes.com/sites/lizfrazierpeck/2019/08/29/5-reasons-personal-finance-should-be-taught-in-school/?sh=6b3fda351784. [Accessed: 18-Mar-2022].

[10] T. Roth, “6 benefits of Personal Finance Apps,” *ProFinance Blog*, 03-Oct-2019. [Online]. Available: https://profinanceblog.com/6-benefits-of-personal-finance-apps.html. [Accessed: 18-Mar-2022].

[11] T. Segal, “Money management for college students,” *Investopedia*, 29-Nov-2021. [Online]. Available: https://www.investopedia.com/financial-edge/0712/money-management-for-college-students.aspx. [Accessed: 18-Mar-2022].

APPENDIX A

A.1 Weekly Report

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 3 Year 3 (Y3S3)	**Study week no.:** Week 1
Student Name & ID: Ng Sian Hong & 19ACB04788	
Supervisor: Ts Dr Chan Lee Kwun	
Project Title: Personal Finance Management Application for University Students	

1. WORK DONE

- This week I have spent the whole to go through my FYP1 report in order to let me know where my progress currently is.

2. WORK TO BE DONE

- I have to start to plan and write the proposal for my FYP 2.
- I have to make sure the completed modules can be performed without any problems.

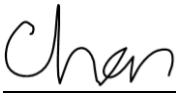
3. PROBLEMS ENCOUNTERED

- So far, I have not encountered any problems yet.

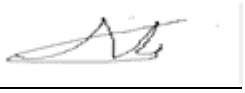
4. SELF EVALUATION OF THE PROGRESS

- Need to clearly know the FYP2 report schedule.
- My application has been developed with more than 70% of code completed.

APPENDIX A



Supervisor's signature



Student's signature

APPENDIX A

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 3 Year 3 (Y3S3)	**Study week no.:** Week 2
Student Name & ID: Ng Sian Hong & 19ACB04788	
Supervisor: Ts Dr Chan Lee Kwun	
Project Title: Personal Finance Management Application for University Students	

1. WORK DONE

- I have clearly known where my progress is since FYP1.
- I have reviewed the project timeline in my FYP1 report.

2. WORK TO BE DONE

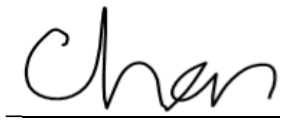
- I have to start to plan and write the proposal for my FYP 2.
- I have to make sure the completed modules can be performed without any problems.

3. PROBLEMS ENCOUNTERED

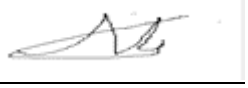
- So far, I have not encountered any problems yet.

4. SELF EVALUATION OF THE PROGRESS

- Need to clearly know the FYP2 report schedule.
- Need to start to extend my FYP1 report.



Supervisor's signature



Student's signature

APPENDIX A

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 3 Year 3 (Y3S3)	**Study week no.:** Week 3
Student Name & ID: Ng Sian Hong & 19ACB04788	
Supervisor: Ts Dr Chan Lee Kwun	
Project Title: Personal Finance Management Application for University Students	

1. WORK DONE

- I have gone through all the chapter in my FYP1 report.
- I have started my application to test my application whether can work properly.

2. WORK TO BE DONE

- I have to start to do the last module, which is Graph and chart display module.
- I need to do some research about how to implement the last module.
- I have to enhance my login system since currently my login unable to let user to perform "forgot password" function.

3. PROBLEMS ENCOUNTERED

- I have no idea whether how to do the last module.
- Currently my system does not have any third party can assist my application perform "forgot password" function.

4. SELF EVALUATION OF THE PROGRESS

- Need to start to extend my FYP1 report.
- Need to start to develop my application.
- My application has been developed with more than 70% of code completed.

Supervisor's signature

Student's signature

APPENDIX A

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 3 Year 3 (Y3S3)	**Study week no.:** Week 4
Student Name & ID: Ng Sian Hong & 19ACB04788	
Supervisor: Ts Dr Chan Lee Kwun	
Project Title: Personal Finance Management Application for University Students	

1. WORK DONE

- I have started my application to test my application whether can work properly.
- I have tried to research on my last module and implement it to my application.
- I have decided to use firebase to help me to do the "forgot password" function.

2. WORK TO BE DONE

- I need to do some research about how to implement the last module.
- I have to enhance my login system since currently my login unable to let user to perform "forgot password" function.
- I have to start extending the FYP2 report.

3. PROBLEMS ENCOUNTERED

- So far no problem.

4. SELF EVALUATION OF THE PROGRESS

- Need to start to extend my FYP1 report.
- Need to start to develop my application.
- My application has been developed with more than 70% of code completed.

Supervisor's signature

Student's signature

APPENDIX A

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 3 Year 3 (Y3S3)	**Study week no.:** Week 5
Student Name & ID: Ng Sian Hong & 19ACB04788	
Supervisor: Ts Dr Chan Lee Kwun	
Project Title: Personal Finance Management Application for University Students	

1. WORK DONE

- I have done to research how to implement the last module and got some tutorials.
- I have found how to use firebase to help my application to assist forgot password.

2. WORK TO BE DONE

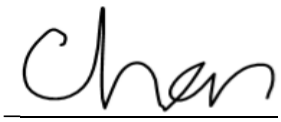
- I need to do the last module based on what the resource I have found, for example, using google api.
- I have to create a project in firebase and then combine to my application to help user to reset password.
- I have to start extending the FYP2 report.

3. PROBLEMS ENCOUNTERED

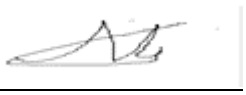
- So far no problem.

4. SELF EVALUATION OF THE PROGRESS

- Need to start to extend my FYP2 report.
- My application has been developed with more than 70% of code completed.



Supervisor's signature



Student's signature

APPENDIX A

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 3 Year 3 (Y3S3)	**Study week no.:** Week 6
Student Name & ID: Ng Sian Hong & 19ACB04788	
Supervisor: Ts Dr Chan Lee Kwun	
Project Title: Personal Finance Management Application for University Students	

1. WORK DONE

- I have done the last module, which is Graph and chart generation module.
- I have done the "forgot password" function to help user to reset password.

2. WORK TO BE DONE

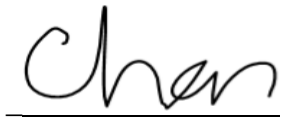
- I need to write the chapter 3 and 4 again since I have done the last module.
- I need to start testing my application.
- I need to start writing my application's verification plans.

3. PROBLEMS ENCOUNTERED

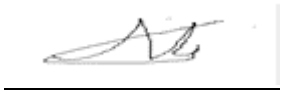
- So far no problem.

4. SELF EVALUATION OF THE PROGRESS

- Need to start to extend my FYP2 report.
- My application has been developed with more than 90% of code completed.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 3 Year 3 (Y3S3)	**Study week no.:** Week 7
Student Name & ID: Ng Sian Hong & 19ACB04788	
Supervisor: Ts Dr Chan Lee Kwun	
Project Title: Personal Finance Management Application for University Students	

1. WORK DONE

- My application has been fully developed.

2. WORK TO BE DONE

- I am still writing the chapter 3 and 4.
- I need to start testing my application.
- I need to start writing my application's verification plans.
- I need to start writing Chapter 5, which system implementation.

3. PROBLEMS ENCOUNTERED

- So far no problem.

4. SELF EVALUATION OF THE PROGRESS

- My application has been developed with more than 100% of code completed.

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 3 Year 3 (Y3S3)	**Study week no.:** Week 8
Student Name & ID: Ng Sian Hong & 19ACB04788	
Supervisor: Ts Dr Chan Lee Kwun	
Project Title: Personal Finance Management Application for University Students	

1. WORK DONE

- My application has been fully developed.
- I have done the chapter 3 part for my report.

2. WORK TO BE DONE

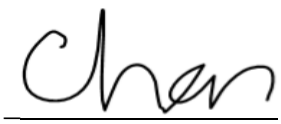
- I am still writing the chapter 4.
- Currently testing on my application.
- I need to start writing my application's verification plans when I perform testing on my application.
- I need to start writing Chapter 5, which system implementation.

3. PROBLEMS ENCOUNTERED

- So far no problem.

4. SELF EVALUATION OF THE PROGRESS

- My application has been developed with more than 100% of code completed.



Supervisor's signature

Student's signature

APPENDIX A

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 3 Year 3 (Y3S3)	**Study week no.:** Week 9
Student Name & ID: Ng Sian Hong & 19ACB04788	
Supervisor: Ts Dr Chan Lee Kwun	
Project Title: Personal Finance Management Application for University Students	

1. WORK DONE

- My application has been fully developed.
- I have fully tested on my application.
- I have done the chapter 4,5,6 and 7 parts for my report.
- I have done to write my application's verification plans and the tested result.

2. WORK TO BE DONE

- I have to do final checking for my draft report.
- I have to use Turnitin to scan my draft report.
- Preparing to submit the draft report to supervisor.

3. PROBLEMS ENCOUNTERED

- So far no problem.

4. SELF EVALUATION OF THE PROGRESS

- My application has been developed with more than 100% of code completed.

Supervisor's signature

Student's signature

APPENDIX A

FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

Trimester, Year: Trimester 3 Year 3 (Y3S3)	**Study week no.:** Week 10
Student Name & ID: Ng Sian Hong & 19ACB04788	
Supervisor: Ts Dr Chan Lee Kwun	
Project Title: Personal Finance Management Application for University Students	

1. WORK DONE

- My application has been fully developed.
- I have fully tested on my application.
- Scanned my draft report to Turnitin.
- I have submitted the draft report to supervisor.

2. WORK TO BE DONE

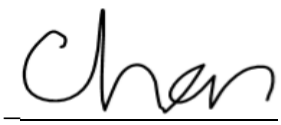
- Do correction of the draft report based on the supervisor's suggestion.
- Do testing on the developed application again.

3. PROBLEMS ENCOUNTERED

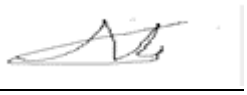
- So far no problem.

4. SELF EVALUATION OF THE PROGRESS

- My application has been developed with more than 100% of code completed.
- Report has been fully completed.



Supervisor's signature



Student's signature

APPENDIX A

FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

Trimester, Year: Trimester 3 Year 3 (Y3S3)	**Study week no.:** Week 11
Student Name & ID: Ng Sian Hong & 19ACB04788	
Supervisor: Ts Dr Chan Lee Kwun	
Project Title: Personal Finance Management Application for University Students	

1. WORK DONE

- My application has been fully developed.
- All the chapters have been completed.

2. WORK TO BE DONE

- Still testing on the developed application based on last week progress.
- Have to start preparing the ppt slides for presentation.

3. PROBLEMS ENCOUNTERED

- So far no problem.

4. SELF EVALUATION OF THE PROGRESS

- My application has been developed with more than 100% of code completed.
- Report has been fully completed.

Supervisor's signature

Student's signature

APPENDIX A

FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

Trimester, Year: Trimester 3 Year 3 (Y3S3)	**Study week no.:** Week 12
Student Name & ID: Ng Sian Hong & 19ACB04788	
Supervisor: Ts Dr Chan Lee Kwun	
Project Title: Personal Finance Management Application for University Students	

1. WORK DONE

- My application has been fully developed.
- All the chapters have been completed.

2. WORK TO BE DONE

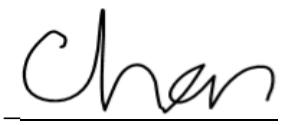
- Still testing on the developed application based on last week progress.
- Have to start preparing the ppt slides for presentation.
- This week has to finish the final testing of my application.
- Turnitin scan for my final version of report.

3. PROBLEMS ENCOUNTERED

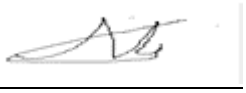
- So far no problem.

4. SELF EVALUATION OF THE PROGRESS

- My application has been developed with more than 100% of code completed.
- Report has been fully completed.



Supervisor's signature



Student's signature

APPENDIX A

FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

Trimester, Year: Trimester 3 Year 3 (Y3S3)	**Study week no.:** Week 13
Student Name & ID: Ng Sian Hong & 19ACB04788	
Supervisor: Ts Dr Chan Lee Kwun	
Project Title: Personal Finance Management Application for University Students	

1. WORK DONE

- My application has been fully developed.
- All the chapters have been completed.
- My application has been fully tested.
- Finished the Turnitin scan.

2. WORK TO BE DONE

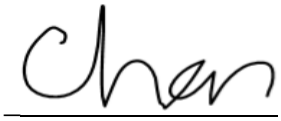
- Have to prepare the ppt slides for presentation.
- Have to submit the report to lab officer.

3. PROBLEMS ENCOUNTERED

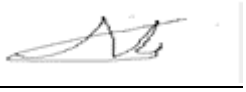
- So far no problem.

4. SELF EVALUATION OF THE PROGRESS

- My application has been developed with more than 100% of code completed.
- Report has been fully completed.

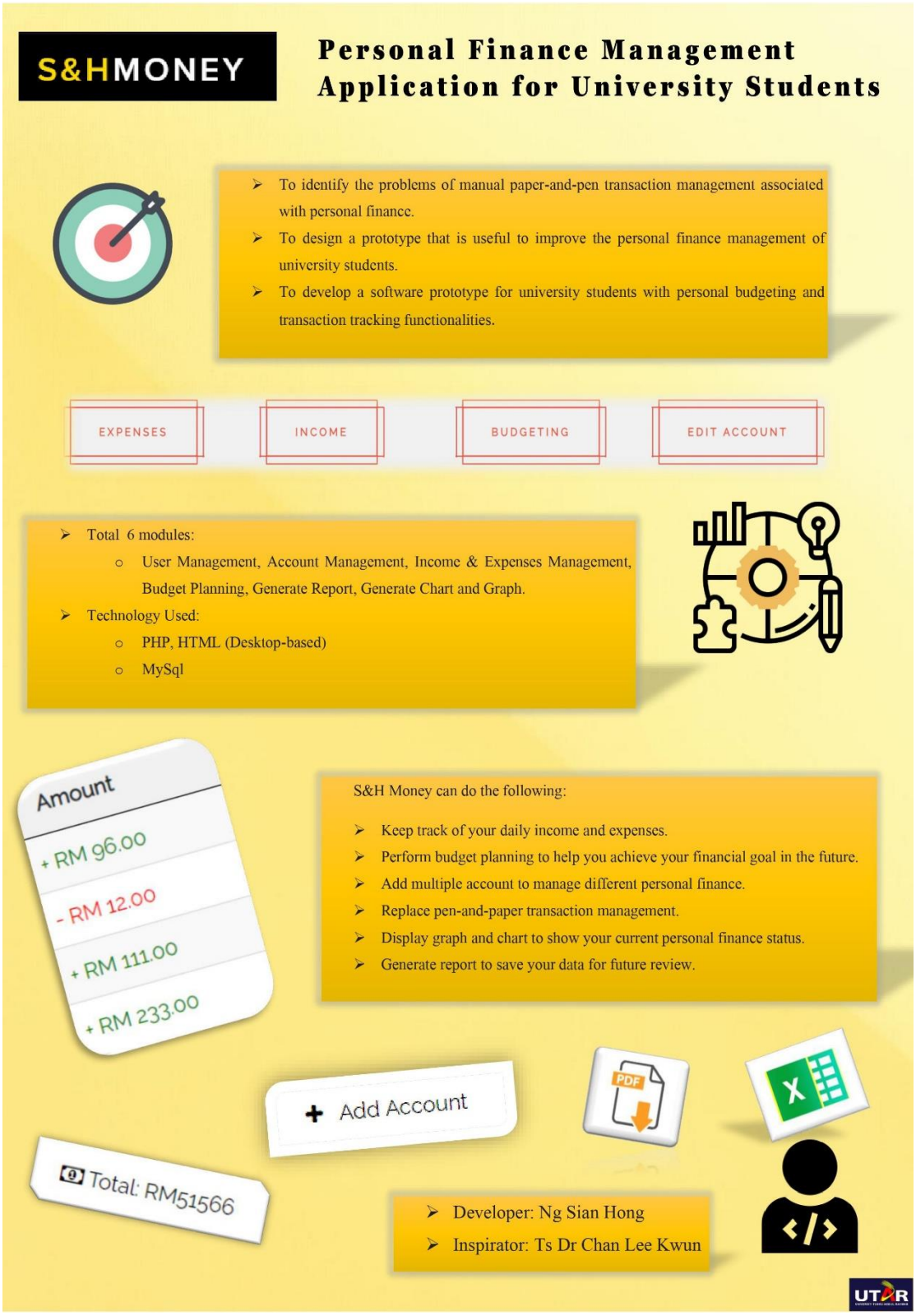


Supervisor's signature

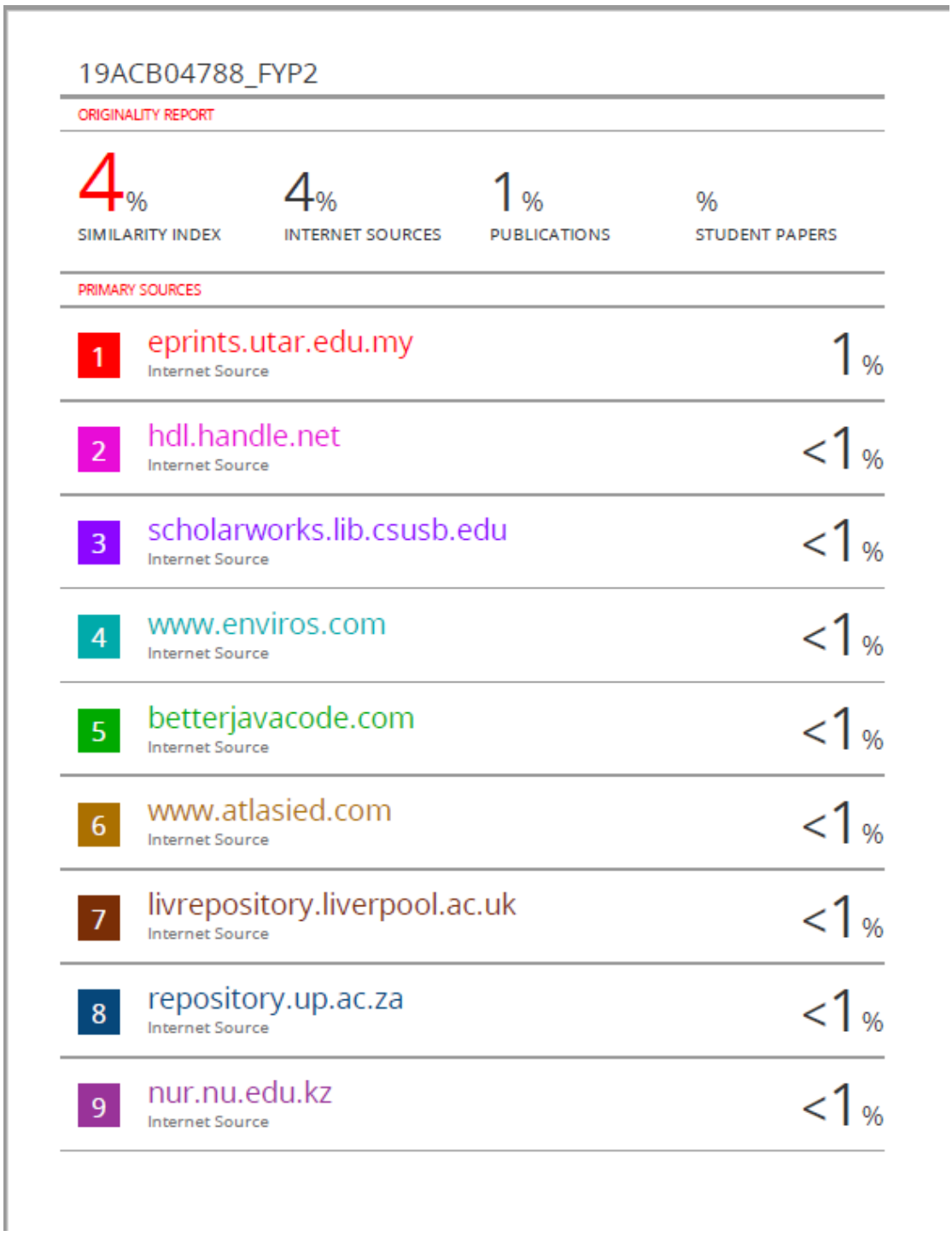


Student's signature

A.2 Poster


S&HMONEY
Personal Finance Management Application for University Students
To identify the problems of manual paper-and-pen transaction management associated with personal finance.
To design a prototype that is useful to improve the personal finance management of university students.
To develop a software prototype for university students with personal budgeting and transaction tracking functionalities.
EXPENSES
INCOME
BUDGETING
EDIT ACCOUNT
Total 6 modules:
User Management, Account Management, Income & Expenses Management, Budget Planning, Generate Report, Generate Chart and Graph.
Technology Used:
PHP, HTML (Desktop-based)
MySql
Amount
+ RM 96.00
- RM 12.00
+ RM 111.00
+ RM 233.00
S&H Money can do the following:
Keep track of your daily income and expenses.
Perform budget planning to help you achieve your financial goal in the future.
Add multiple account to manage different personal finance.
Replace pen-and-paper transaction management.
Display graph and chart to show your current personal finance status.
Generate report to save your data for future review.
+ Add Account
PDF
Total: RM51566
Developer: Ng Sian Hong
Inspirator: Ts Dr Chan Lee Kwun
UTAR

PLAGIARISM CHECK RESULT



19ACB04788_FYP2

ORIGINALITY REPORT

4%	4%	1%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	eprints.utar.edu.my Internet Source	1%
2	hdl.handle.net Internet Source	<1%
3	scholarworks.lib.csusb.edu Internet Source	<1%
4	www.enviros.com Internet Source	<1%
5	betterjavacode.com Internet Source	<1%
6	www.atlasied.com Internet Source	<1%
7	livrepository.liverpool.ac.uk Internet Source	<1%
8	repository.up.ac.za Internet Source	<1%
9	nur.nu.edu.kz Internet Source	<1%

10	3ghm3eu625r2c657owx4k65z-wpengine.netdna-ssl.com Internet Source	<1%
11	www.netsys.com.tw Internet Source	<1%
12	Timothy L. Bailey. " Discovering Novel Sequence Motifs with ", Current Protocols in Bioinformatics, 2002 Publication	<1%
13	Beginning C# 5 0 Databases, 2012. Publication	<1%
14	www.scribd.com Internet Source	<1%
15	Alberto Pettorossi. "Chapter 5 Parsers for Deterministic Context-Free Languages: LR(k) Parsers", Springer Science and Business Media LLC, 2021 Publication	<1%
16	datatables.net Internet Source	<1%
17	researchspace.ukzn.ac.za Internet Source	<1%
18	Isaac Gottlieb. "Excel Charts", Wiley, 2009 Publication	<1%
19	de.scribd.com Internet Source	

		<1%
20	shura.shu.ac.uk Internet Source	<1%
21	Philippe Guéguen, Philippe Roux, Andrea Colombi. "Urban Seismology: Experimental Approach to Soil–Structure Interaction Towards the Concept of Meta - city", Wiley, 2022 Publication	<1%
22	eprints.utm.my Internet Source	<1%
23	David S. Goodsell. "Representing Structural Information with RasMol", Current Protocols in Bioinformatics, 2005 Publication	<1%
24	digital.lib.usf.edu Internet Source	<1%
25	oer.unn.edu.ng Internet Source	<1%
26	www.teco.com.sg Internet Source	<1%
27	Gilbert, Nigel, Troitzsch, Klaus. "Simulation for the Social Scientist", Simulation for the Social Scientist, 2005 Publication	<1%

28	Petrus Paulus Jonker. "Chapter 5 Pipelined low level image processing", Springer Science and Business Media LLC, 1992 Publication	<1 %
29	eprints.lincoln.ac.uk Internet Source	<1 %
30	serval.unil.ch Internet Source	<1 %
31	www.china-planet.com Internet Source	<1 %
32	aerojet.engr.ucdavis.edu Internet Source	<1 %
33	archive.org Internet Source	<1 %
34	coding-god93.tistory.com Internet Source	<1 %
35	ijariie.com Internet Source	<1 %
36	inba.info Internet Source	<1 %
37	repository.lib.tottori-u.ac.jp Internet Source	<1 %
38	www.mrf.com Internet Source	<1 %

39	www.zimstat.co.zw Internet Source	<1%
40	Allen, Phillip E.. "CMOS Analog Circuit Design", Oxford University Press Publication	<1%
41	Arshad Khan. "Jumpstart Tableau", Springer Science and Business Media LLC, 2016 Publication	<1%
42	I. V. Sukhinskiy, E. A. Nepovinnykh, G. I. Radchenko. "Developing a Parking Monitoring System based on the analysis of images from an outdoor surveillance camera", 2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2016 Publication	<1%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		

PLAGIARISM CHECK RESULT

Universiti Tunku Abdul Rahman			
Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1of 1

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	Ng Sian Hong
ID Number(s)	19ACB04788
Programme / Course	Information System Engineering
Title of Final Year Project	Personal Finance Management Application for University Students

Similarity	**Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)**
Overall similarity index: 4 % **Similarity by source** Internet Sources: 4 % Publications: 1 % Student Papers: 0 %	
Number of individual sources listed of more than 3% similarity: 0	

Parameters of originality required and limits approved by UTAR are as Follows:

(i) Overall similarity index is 20% and below, and

(ii) Matching of individual sources listed must be less than 3% each, and

(iii) Matching texts in continuous block must not exceed 8 words

Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

Chan

Signature of Supervisor

Name: Ts Dr Chan Lee Kwun

Date: 22 April 2022

Signature of Co-Supervisor

Name: _____

Date: _____

UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY (KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	19ACB04788
Student Name	Ng Sian Hong
Supervisor Name	Ts Dr Chan Lee Kwun

TICK (√)	DOCUMENT ITEMS Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
√	Front Plastic Cover (for hardcopy)
√	Title Page
√	Signed Report Status Declaration Form
√	Signed FYP Thesis Submission Form
√	Signed form of the Declaration of Originality
√	Acknowledgement
√	Abstract
√	Table of Contents
√	List of Figures (if applicable)
√	List of Tables (if applicable)
√	List of Symbols (if applicable)
√	List of Abbreviations (if applicable)
√	Chapters / Content
√	Bibliography (or References)
√	All references in bibliography are cited in the thesis, especially in the chapter of literature review
√	Appendices (if applicable)
√	Weekly Log
√	Poster
√	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)
√	I agree 5 marks will be deducted due to incorrect format, declare wrongly the ticked of these items, and/or any dispute happening for these items in this report.

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report.

(Signature of Student)

Date: 22nd April 2022