

MONITRA: A prototype to monitor study habits and posturing in children

By

Dennis Chong Kwang Khai

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

FEBRUARY 2026

ACKNOWLEDGMENTS

I would like to express my sincere thanks to my initial supervisor, Dr Ng Hui Fuang, who had given me the opportunity to work on this project. It is also thanks to his ideas and advice that this project can take on several meaningful explorations on the project's method, providing clarity and additional knowledge.

I would also like to express my biggest gratitude and thanks to my new supervisor, Cik Ashikin binti Ali, for her genuine discussions, patience, and care. Monitra was able to gain clarity, allowing me to bring her suggestions to life through our discussion. It is thanks to her, that Monitra can succeed as a proof-of-concept and take shape of what it is today.

I would also like to thank my family and relatives for support in both mental and financial. Without the helping hands that was given to me, I wouldn't be where I am today, to accomplish what I had accomplished.

Last but not least, I would like to thank my significant other and my friends, specifically Pang Jing Thean and Jimmy Ding Jia Kang for assisting me in Monitra's system testing, patiently participated while the application broke down.

COPYRIGHT STATEMENT

© 2026 Dennis Chong Kwang Khai. All rights reserved.

This Final Year Project proposal is submitted in partial fulfillment of the requirements for the degree of Bachelor of Computer Science (Honours) at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project proposal represents the work of the author, except where due acknowledgment has been made in the text. No part of this Final Year Project proposal may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

ABSTRACT

Lack of parental supervision on children is common in current times, leading to potential issues academically, and study completion. Moreover, prolonged hours of sitting during study contributes to posture issues, which are common among people but tend to be ignored. Posture issue awareness, combined with children's studying well-being, can be supported or solved by their guardians, however due to the demanding nature of their work, majority of parents are unable to keep track of their children's well-being. The result of this project, Monitra, is developed to solve this situation. Monitra is a prototype that monitors children's study habits and posture during study sessions, leveraging Visual Language Model (VLM) technology for its ability to capture contextual information of human actions. It contains 2 applications, a desktop processing client for children responsible for frame grabbing and processing from webcam or an external camera that supports RTSP URL, and a mobile visualisation client for parents that has dashboard information and push notification. Through the evaluation results, it is found that Visual Language Model (VLM) are able to perform monitoring tasks and perform exceptionally well in their tasks when utilising an external camera.

Area of Study: Desktop App development, Mobile App development.

Keywords: AI, VLM, LLM, Posture Detection, Mobile Application, Desktop Application

TABLE OF CONTENTS

TITLE PAGE	i
COPYRIGHT STATEMENT	ii
ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
TABLE OF CONTENTS	v-viii
LIST OF FIGURES	ix-xii
LIST OF TABLES	xiii
LIST OF ABBREVIATIONS	xiv
CHAPTER 1 PROJECT BACKGROUND	1
1.1 Problem Statement and Motivation	1
1.1.1 Motivation	2
1.2 Project Objectives	2
1.3 Project Scope	2 – 3
1.4 Contributions	3 – 4
CHAPTER 2 LITERATURE REVIEW	5
2.1 Existing work on children monitoring	5
2.1.1 Children On-screen Monitoring Application	5 – 6
2.1.2 Qustodio	6 – 7
2.1.3 CuboAI	7 - 8
2.2 Existing work on posture monitoring	8
2.2.1 UprightGo	8 – 9
2.2.2 PostureMinder	9 – 10
2.2.3 ZenPosture	10 – 11
2.3 Work done on using VLM for monitoring purposes	11
2.3.1 Idefics Model for Driver Monitoring System	11
2.4 Limitation of Previous Studies	12
2.5 Proposed Solution	13 – 14

CHAPTER 3 MONITRA: A PROTOTYPE TO MONITOR	15
STUDY HABITS AND POSTURING IN CHILDREN	
3.1 System Architecture Diagram	15
3.1.1 Monitra Overview	15 – 16
3.1.2 Use Case Diagram	17
3.1.3 Activity Diagram	18 – 19
3.2 Hardware Setup	20
3.3 Software Setup	21
3.3.1 Monitra Mobile IDE and Languages/Framework	21 – 22
3.3.2 Monitra Desktop IDE and Languages/Framework	22 – 24
3.3.3 Monitra’s Cloud Service	24
3.3.4 Monitra’s VLM	24
3.4 Application Architecture	24 – 25
CHAPTER 4 MONITRA SYSTEM DESIGN	26
4.1 System Block Diagram	26 – 27
4.2 System Flowchart and Implementation	27
4.2.1 System Flowchart and Implementation for mobile application	28
4.2.1.1 Registration of new user (Parent)	28 – 29
4.2.1.2 Login of existing user (Parent)	30 – 31
4.2.1.3 Homepage of application	31 – 33
4.2.1.4 Settings page	34 – 35
4.2.1.5 Changing user credentials and child account details	36 – 37
4.2.1.6 Dashboard page	38 – 39
4.2.1.7 Edit, delete, or view study session page	40 – 41
4.2.1.8 Alert history page	42 – 43

4.2.1.9 Live feed page	43 – 45
4.2.2 System Flowchart and Implementation for desktop application	46
4.2.2.1 Login for parent and child	46 – 49
4.2.2.2 Homepage of desktop application	49 – 52
4.2.2.3 Desktop application Session Manager	53 – 54
4.2.2.4 Desktop application Frame Manager	55 – 56
4.2.2.5 Desktop application VLM Analyser	57 – 58
4.2.2.6 Desktop application Live Feed Manager	59 – 60
4.3 Implementation Issues and Challenges	61
CHAPTER 5 SYSTEM EVALUATION AND DISCUSSION	62
5.1 System Testing	62
5.1.1 Test subject 1	62 – 67
5.1.2 Test subject 2	68 – 72
5.1.3 Test subject 3	72 – 76
5.1.4 Results discussion	77 – 78
5.2 Feature Testing	78
5.2.1 Background Managers	78
5.2.1.1 Session Manager	78 – 79
5.2.1.2 Frame Manager	79
5.2.2 Event Drive Communication	80
5.2.2.1 Real-time posture notification	80
5.2.2.1 Push Notification for subject detection	80 – 81
5.2.3 Live Feed feature	81 – 82
5.3 Use Case Testing	82 – 84

5.4 Project Challenges	84
5.5 Objective Evaluations	85
5.6 Concluding Remark	85
CHAPTER 6 CONCLUSION AND RECOMMENDATION	
6.1 Conclusion	86
6.2 Recommendation	86
REFERENCES	87 – 88
APPENDIX A	A89–A91
POSTER	A92

LIST OF FIGURES

Figure Number	Title	Page
Figure 2.1	Example of report via Telegram by COSM	5
Figure 2.2	Reminder of PostureMinder extension	9
Figure 2.3	Configuration of PostureMinder extension	10
Figure 2.4	BLIP result of image captioning	12
Figure 3.1	General view of Monitra's architecture	16
Figure 3.2	Use case diagram of Monitra	17
Figure 3.3	Activity diagram of mobile application	18
Figure 3.4	Activity diagram of desktop application	19
Figure 3.5	Application Architecture	25
Figure 4.1	System block diagram of Monitra	27
Figure 4.2	Flowchart of Registration	28
Figure 4.3	Wireframe for Registration	29
Figure 4.4	Screenshot of Registration page	29
Figure 4.5	Flowchart of Login	30
Figure 4.6	Wireframe of Login	31
Figure 4.7	Screenshot of Login page	31
Figure 4.8	Flowchart of homepage	32
Figure 4.9	Wireframe of homepage	33
Figure 4.10	Wireframe of child creation page	33
Figure 4.11	Wireframe of account option page	33
Figure 4.12	Screenshot of homepage	33

Figure 4.13	Screenshot of child creation page	33
Figure 4.14	Screenshot of account option page	33
Figure 4.15	Flowchart of setting page	34
Figure 4.16	Wireframe of setting page	35
Figure 4.17	Screenshot of setting page	35
Figure 4.18	Flowchart of Settings option	36
Figure 4.19	Screenshot of changing username page	37
Figure 4.20	Screenshot of changing password page	37
Figure 4.21	Screenshot of changing email page	37
Figure 4.22	Screenshot of selecting children account page	37
Figure 4.23	Screenshot of changing children name page	37
Figure 4.24	Flowchart of Dashboard page	38
Figure 4.25	Dashboard page of daily	38
Figure 4.26	Dashboard page of daily	38
Figure 4.27	Dashboard page of weekly	39
Figure 4.28	Dashboard page of weekly	39
Figure 4.29	Dashboard page of monthly	39
Figure 4.30	Dashboard page of monthly	39
Figure 4.31	Flowchart of edit session	40
Figure 4.32	Wireframe of study sessions toggle	41
Figure 4.33	Wireframe of creating session	41
Figure 4.34	Screenshot of study sessions toggle	41
Figure 4.35	Screenshot of creating session	41

Figure 4.36	Screenshot of deleting session	41
Figure 4.37	Flowchart of alert history	42
Figure 4.38	Screenshot of alert history page	43
Figure 4.39	Screenshot of alert detail page	43
Figure 4.40	Flowchart of Live Feed page	44
Figure 4.41	Screenshot of Live Feed page	45
Figure 4.42	Flowchart of parent login	47
Figure 4.43	Wireframe of parent login	47
Figure 4.44	Screenshot of parent login	48
Figure 4.45	Wireframe of child login	48
Figure 4.46	Screenshot of child login	49
Figure 4.47	Flowchart of homepage and camera page	50
Figure 4.48	Wireframe for desktop homepage	51
Figure 4.49	Screenshot of desktop homepage	51
Figure 4.50	Wireframe of camera page	52
Figure 4.51	Wireframe of RTSP UI	52
Figure 4.52	Screenshot of desktop camera page and RTSP UI	52
Figure 4.53	Flowchart of Session Manager	54
Figure 4.54	Flowchart of Frame Manager	56
Figure 4.55	Flowchart of VLM Analyser	58
Figure 4.56	Flowchart of Live Feed manager	60
Figure 5.1	Dashboard of test subject 1	67
Figure 5.2	Dashboard of test subject 1	67

Figure 5.3	Dashboard UI of test subject 2	72
Figure 5.4	Dashboard UI of test subject 2	72
Figure 5.5	Dashboard UI for test subject 3	77
Figure 5.6	Dashboard UI for test subject 3	77

LIST OF TABLES

Table Number	Title	Page
Table 2.1	Limitations and Solutions	15
Table 3.1	Specifications of laptop	19
Table 3.2	Specifications of mobile phone	19
Table 3.3	Specifications of external camera	23
Table 3.4	IDE for mobile application	23
Table 3.5	Mobile application libraries	
Table 3.6	IDE for desktop application	
Table 3.7	Desktop application libraries	
Table 5.1	Webcam results for test subject 1	
Table 5.2	External camera results for test subject 1	
Table 5.3	Webcam results for test subject 2	
Table 5.4	External camera results for test subject 2	
Table 5.5	Webcam result for test subject 3	
Table 5.6	External camera result for subject 3	
Table 5.7	Aggregated results of 3 test subjects	
Table 5.8	Session Manager	
Table 5.9	Frame manager	
Table 5.10	Real-time posture notification feature	
Table 5.11	Push notification feature	
Table 5.12	Live Feed feature	
Table 5.13	Use case testing table for mobile application	
Table 5.14	Use case testing table for mobile application	

LIST OF ABBREVIATIONS

<i>AI</i>	Artificial Intelligence
<i>COSM</i>	Children On-Screen Monitoring
<i>VLM</i>	Visual-Language Model
<i>LLM</i>	Large-Language Model
<i>CV</i>	Computer Vision
<i>UI</i>	User Interface
<i>RTSP</i>	Real-time Streaming Protocol
<i>MVVM</i>	Model-View-ViewModel
<i>GUI</i>	Graphical User Interface

CHAPTER 1

INTRODUCTION

1.1 Problem Statement

Lack of parental supervision due to work demand, leaving parents unaware of their children's study habits, unable to ensure their academic well-being.

In today's household, parents often need to work long hours, leaving children with minimal supervision, however children are expected to make use of their time, performing self-study session. The lack of supervision, combined with existing technologies, can reduce their study efficiency. While parents prefer to monitor their children's study habits, work schedules or privacy concerns often prevent them from doing so, leaving **parents unaware of their study well-being**, preventing them to support their children academically when needed.

Lack of intervention and awareness of posture health among children.

Additionally, extended study session may **negatively impact one's posture health**. Research has found that prolonged hour of study can lead to negative effects on student's posture, health and comfort, directly affecting their concentration in class, and ways of intervention includes regular breaks or ergonomic environment. Hence, children may be at greater risk, especially since they may lack ergonomic support and awareness.

Expensive resources to leverage existing state-of-the-art technology.

Vision Language Model (VLM) is a technology that understands both image input and text input. It is a combination of Large Language Model (LLM), with a visual encoder that allows the LLMs to 'see', allowing the model to understand contextual information of an image. However, this great capability comes at a cost of extensive computational resources, where it is difficult to utilize the model locally on consumer-grade computers without a certain amount of Graphical Processing Unit (GPU) memory.

1.1.2 Motivation

This project is motivated by the need for an assistant that can enable parents to stay informed about their children's study habits and posture health, allowing parents to make informed decisions toward their children when needed. Furthermore, there is no existing system currently that specifically informs parents about children's study habits and their posture issues in real-time. Although research and products have been explored in their own domain, there is a lack of widespread practical application and official documentation that justifies the use of VLM as core functionality. This is often caused by the resource-intensive nature, and potential hallucinations.

However, the ability of VLM to understand contextual information of an image input combined with text understanding presents an opportunity to explore the development of a more sophisticated system. Therefore, this project aims to leverage the capabilities of VLM while minimizing the cost and risk needed. By leveraging VLM, an AI-based assistant can provide better information on children's studying activity and posture well-being, offering a real-time and non-intrusive way for parents to monitor children's study habits and posture well-being.

1.2 Project Objectives

1. To develop an AI-based assistant on monitoring children's study habits and posture health.
2. To promote posture health at young age via real-time notification.
3. To validate the effectiveness of Monitra with at least 3 subjects.

1.3 Project Scope

Prior to feature decision of Monitra, we first confirm the scope of monitoring for both studying activity and posture domain. To accurately evaluate the Vision-Language Model's (VLM) zero-shot classification capabilities, "study activity" is defined primarily by the subject's behavioural actions, and, when visible, the presence of study-related physical objects (e.g., stationery). This activity is evaluated over a 10-second temporal window, sampled at 2.5-second intervals. Furthermore, within the posture monitoring domain, "good" and "bad" postures are explicitly defined as an upright seated position and a slouched (rounded) back, respectively.

Monitra consists of 2 applications, similar in theory to a publisher-subscriber model. The desktop application acts as the publisher responsible for processing video frames on children, while the mobile application acts as the subscriber, listening to cloud storage and processing data into visuals for parent monitoring. Monitra's desktop application has two methods of camera inputting, which is through webcam or an external camera that supports RTSP URL. In the domain of posture monitoring, the desktop application has real-time notification when bad posture thresholds are triggered while in the domain of study habits monitoring, an alert will be triggered, resulting in a push notification on Monitra's mobile application if their child is not present in the study environment. Monitra mobile application serves as the parental client, supporting the creation of multiple child profiles. Each profile features an individualized analytics dashboard, creation of study sessions, and alert history. The dashboard aggregates data across daily, weekly, and monthly timeframes using four primary visualizations: two pie charts quantifying study engagement and posture quality, a line chart tracking focus trends over time, and a stacked bar chart visualising posture variations throughout the session. Additionally, live viewing of study session from mobile application will also be introduced acting as a centralized application for parents to monitor their children only during active study session.

1.4 Contribution

The primary contribution of Monitra is to demonstrate a proof-of-concept framework that delivers contextual insights into a child's study habits through the utilization of Vision-Language Models (VLM). Unlike traditional computer vision methods limited to basic object detection, VLM are able to generate nuanced behavioral context with acceptable accuracy. This highlights how future intelligent decision-support systems can empower busy or remote parents to stay updated on their child well-being. Crucially, Monitra is designed to be an intelligent decision-support system; while it autonomously processes and classifies children's studying activity, it does not replace human judgement. Instead, it acts as a trustable assistant allowing parents to make informed decisions regarding their children's education and development.

Additionally, the feature of Monitra to give real-time notification on bad posture will be able to help remind children of their posture well-being when studying, allowing children to actively correct their sitting posture such as head, neck or their spine. Studies has shown that posture

CHAPTER 1

health is critical among teens who are experiencing a growth spurt, and having a bad posture in that time period would lead to future posture health problems [1][2]. Hence by utilizing a real-time notification and sending information to parents via mobile application, children will either correct their posture during study session or have their parents to advise them.

CHAPTER 2 LITERATURE REVIEW

Currently, there is none existing technology or product that specifically monitor both children studying activities and posture at the same time. However, plenty of methods have been explored and developed in their own domain, such as monitoring children's activities in different ways, and posture monitoring that ranges from camera sensors to wearable devices.

2. 1 Existing work on children monitoring

In the context of existing method of monitoring children, it includes tracking children behaviour digitally, parental control, different software that uses AI and tracks children studies behaviour, to manually monitoring devices such as using Wi-Fi camera.

2.1.1 Children On-screen Monitoring Application (COSM)

This project specifically tracks children's activity during online class sessions, developed to tackle online class issues during Covid-19 pandemic. The system works by checking user's devices for E-learning applications such as Microsoft Teams, Zoom, Google Meet, and other related applications, and if any of the applications exist, the system will then be put to work [3]. Artificial Intelligence was not used to achieve the objectives and instead, it has functionalities that includes on-screen monitoring, browser activity tracking, entertainment activity tracking and other methods such as keylogger and mouse tracker. In scenarios where student's activity were not aligned with studying, the system sends alerts via Telegram app to inform parents [3].

The strength of this system is the combination of functionalities it contains to identify children's activity. According to the author, the mouse tracking feature is a module that sends a quick and brief report to the parent when the mouse is idle for a set amount of time [3], indicating that the child may not be active during online class session. Furthermore, the module will also produce few beeping sounds to act as a reminder for children to be active in class should they loses focus or distracted. This feature helps children to stay focused during online classes, as it is common for them to be present but not concentrating. Other notable functions also includes the software's keylogger module, where it records all keys that are entered on the keyboard in a text file which can be review by parents [3]. This provides clarity to parents on their children's study well-being.

Despite having various functionalities that tracks children's studying activities, the software are unable to keep track on their activity if children were to conduct self-study sessions without being in an online classroom. Online classes are conducted frequently during quarantine time to tackle Covid-19 pandemic, and with the pandemic tackled, this software of monitoring children is not as effective anymore. Furthermore, as mentioned, it does not effectively monitor children study activity, for children may still be studying even without joining an online class. Additionally, the project method of informing parents is via Telegram that sends alerts, reports, and even text files but this isn't user friendly for it lacks a good user interface for accessibility.

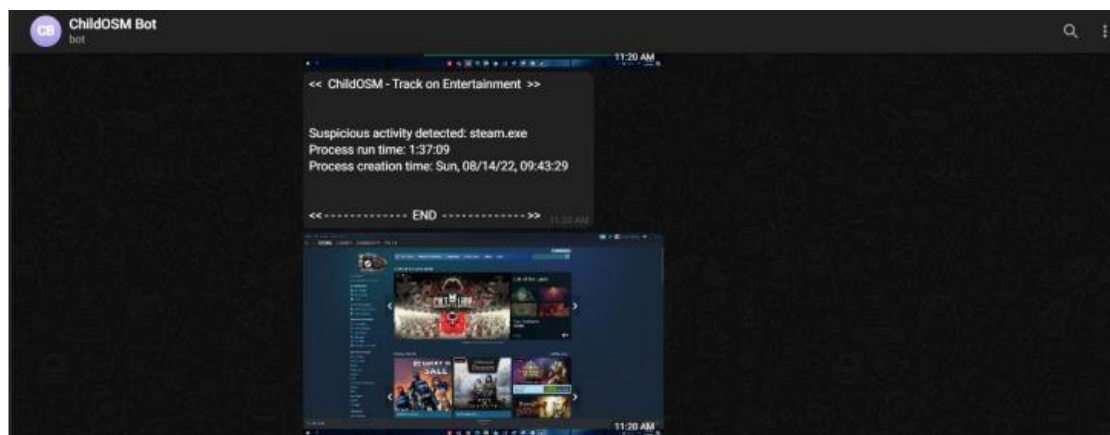


Figure 2.1, Example of report via Telegram by COSM

2.1.2 Qustodio

Qustodio is a parental control app that monitors children activity digitally instead of just focusing on studying activity. The app works across different devices such as Android, Mac, Windows, and even in iPhone and iPad. It has 2 apps where one is for parents and one for targeted device, whereby in the parent account, users can add multiple child account and monitor their digital activity. Qustodio has features that effectively monitor and manage children's account activity [4], and that includes **(i)** filtering content & apps, **(ii)** monitoring activity such as browser history and activity timeline, **(iii)** setting screen time limits or pausing Internet connection, **(iv)** a GPS tracking that allows parents to know their whereabouts, and **(v)**, a feature that tracks calls and SMS of the targeted device. Additionally, AI-powered alerts were implemented to send reports and alerts to parent's app where the AI automatically notify the parents of any suspicious activity or harmful content.

With all these features, Qustodio works very well in informing parents of their children's behaviour digitally and controlling children's activity digitally. This is especially important in this technology era where children spend majority of their time using their devices. However, even with its extraordinary features when it comes to monitoring children activity, it has the same limitation as CSOM, where it is not able to monitor children offline study activity. This is because Qustodio only aim to monitor children's activity digitally and not the physical activity of children. It lacks the ability to monitor children's activity physically aside from tracking their location. Furthermore, Qustodio raises privacy concerns due to its features of monitoring calls and messages of children, as well as accessing browser history. Although it is a strict way to monitor their activity digitally, this would make children to feel insecure as if their actions are constantly monitored which is not recommended for children's wellbeing.

2.1.3 CuboAI

CuboAI, a monitoring system that comes with their own camera hardware, with a purpose to monitor babies' activity and provide alerts when triggered by crying or entering restricted zone specified. Although the work done here is not monitoring children studying activities, it is also worth taking note of their ability to monitor babies, by using their own camera, video frames were captured and sent to their own cloud storage and perform their own AI processes features. The features include detecting baby's covered face, cry detection, danger zone coverage, breathing analysis and temperature detection [5]. This undoubtedly shows that it performs very well in their own niche of monitoring baby using AI to assist parents. Moreover, this product comes with their own camera mobile stand and floor stand to place the camera, as well as CuboAI Smart Baby Monitor, Smart Temp for temperature detection, and Sleep Sensor Pad.

In order for parents to view playbacks of longer time in the past, a payment is required meaning the review time of recorded video are limited and needs a paid cloud subscription. Additionally, despite having great features, privacy issue arises on how the product handle the recorded video and information. Recorded videos are needed to be sent to their cloud storage for processing, but the issue of privacy remains here whereby their data are being sent over to their cloud storage, and the vulnerability of data breaches or related security risks exists. Furthermore, even with having a great security to protect their data digitally [6], the issue of how parents are

unable to control how long their data is stored points out that privacy issue is still a concern here, especially for families that are more sensitive about their privacy.

2.2 Existing work on posture monitoring

Posture monitoring has been explored in various methods that ranges from hardware combination such as wearable devices that provides reminders or ergonomic chair with software, to purely software monitoring using AI.

2.2.1 UprightGo

UprightGo is a wearable device that monitors user's posture by using biofeedback science. It comes with a small product that can be wear like a necklace behind their back or using a provided adhesive to stick on user's back [7]. The main part of the product is that it contains 2 enhanced gyroscope sensors to detect bad posture and provide a real-time reminder by vibrating the device [7]. Hence, if it were to sense bad posture of user's back such as slouching or other related back bad posture, it will produce vibration that helps remind users of their posture. Additionally, the product also has the capability to store user's posture information in their application, showing the information in a dashboard [8]. This enables the user to lookback on their posture health for up to 30 days and to be aware of their posture health whether it is improving or do they need more efforts and awareness. Auto calibration is also done by the product where it will first recognize user's body posture and perform optimal training on itself about good and bad posture before wearing it throughout the day.

However, despite having a real-time vibration for posture detection, few limitations exist where firstly, the auto calibration and setup of the product can be complex. This is because if users are unable to have a good starting position of their posture to let the product know what is good and what is bad [9], a poor setup will cause the product to not function optimally. Hence, the feedback of using vibration will not be triggered if the setup was wrong ultimately affecting the product's functionality. Not to mention, many users who bought the product stated that it is not useful due to this reason. Secondly, by wearing a device behind user's back all day, this raises concerns about their comfortability. Using adhesive or a necklace like product to place the product behind their back, it will still feel uncomfortable to a few degrees. This is due to

having an electronic device that produce vibration, sticking on user's back the whole period of using it. Thirdly, the product only measures bad posture of upper back of the user, by placing the product in between shoulder blades to monitor slouching and so on. This results in correcting only the user's upper back posture, neglecting other posture issues such as leg-crossing or neck tilt.

2.2.2 PostureMinder

PostureMinder, unlike UprightGo, is a simple software based to address posture issue by sending reminder [10]. It is an extension that can be downloaded from Google and provides reminders on interval that would remind users to be aware of their posture. While it has no features that monitor user's posture, it is still worth including as one of the works done for existing solutions to tackle posture health issues. This is due to the easy usage and implementation, where users just need to download the extension, configure reminder intervals and users are even allowed to toggle the sound of reminder. Not to mention, it also has a walk reminder, and its interval can be adjusted as well. Although it does not have monitoring features directly, a reminder of posture correction still achieves the objective of posture monitoring, which is to let users have a better spinal health and increase their awareness.

The limitations of this extension definitely refer to the non-existent monitoring posture, although it works well in what they tend to offer, which is just to provide reminders, it is also way beneficial to directly monitor user's posture instead of providing reminders only especially with existing technology out there. Besides that, the extension of course only works if Google Chrome is running in the background, so it will not be able to provide a 24/7 reminder if Google application is closed, limiting its usage.

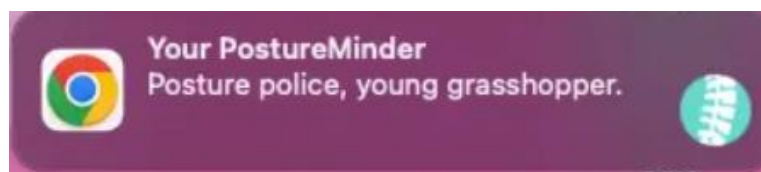


Figure 2.2, Reminder of PostureMinder extension

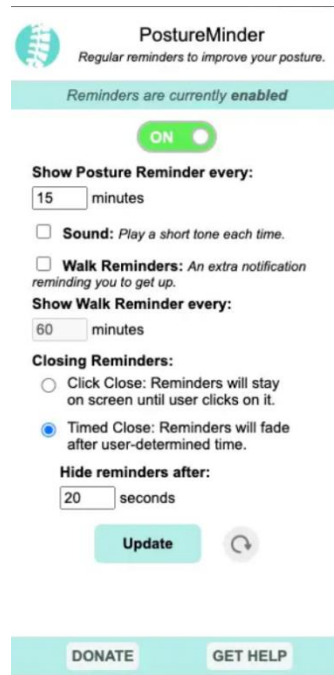


Figure 2.3, Configuration of PostureMinder extension

2.2.3 ZenPosture

ZenPosture is a software-based posture monitoring, that accesses user's webcam and perform posture analysis using AI. This application access user's webcam to capture user's upper body, mainly to tackle posture issue when working [11]. Analyzation of upper body posture using AI is done locally on user's devices and not being sent over cloud to be processed. Like any other posture monitoring software, ZenPosture's way of informing users on their bad posture is similar to PostureMinder, where it uses a lightweight alerting system to provide real time reminders to user [11]. It has been stated that their AI uses TensorFlow to train data and plot points on user's shoulder and nose, to determine bad posture [12]. Moreover, the app also provides personalized insights of user's progress along with statistics and improvement recommendations.

Despite having a unique idea, limitations still exist in their product, whereby the accuracy or effectiveness of the AI during monitoring will be affected by webcam quality. This is due to external environments that may affect the webcam such as potential dirt on the webcam, or lightings of environment, all affecting the accuracy of their AI model. The application currently does not have any ways such as reminder or informing ways to inform users of potential causes that will affect the quality of monitoring, hence it may result in inaccurate response. Additionally, even though privacy issue may not be of concern here due to the frames of video captured are processed locally, this introduce a new problem, where user's webcam and AI

processing are running as long as the application is toggled on, forcing a heavy load of work on user's devices. This in return, may affect user's work productivity that ZenPosture aim to achieve.

2.4 Work done on using VLM for monitoring purposes

Although they are no big products of software that uses VLM to monitor types of things currently, various of research has been done and some notable studies will be examined below.

2.4.1 Idefics Model for Driver Monitoring System

This study shows the implementation of VLM to monitor drivers in vehicles. The specific model chosen was Idefics, a model that is suitable for in-vehicle applications [13] and the purpose of this study is to use VLM to understand driver's state in driving scene compared to traditional methods that uses a combination of sensors and algorithms. It was shown that the study uses various method on Idefics to successfully make an accurate response of driver's state, and the methods include Zero-shot prompting, Few-shot prompting, and Prompt-engineering. Although Zero-shot prompting were not accurate, few-shot prompting shows significant results. Meanwhile, prompt engineering case has mix results, where the accuracy are not consistent due to the prediction of multiple variables in one time and the study made an important point that stated for future work, variables could be separated to multiple agents, meaning each agent handles only one variables and passes all the information to a model that can conclude the result, which may increase the accuracy[13].

2.5 Limitation of Previous Studies

Previous studies in the context of monitoring children's study activities have multiple limitations and that includes the inability to monitor children's physical studying activities. This is because both Qustodio and COSM lack the ability to understand what children are doing physically since their features focus on digital activities. Although, CuboAI does monitor physical activity of babies, privacy issues were the problem due to them storing and processing data on their own cloud. Moreover, in projects like CSOM, it lacks a user-friendly interface that shows reports or dashboards, and instead, reports are being sent via Telegram in text format.

In the context of work done for posture monitoring, products that are wearable such as UprightGo [7], are not comfortable to be worn all day long for every day. Having an electronic device sticking on user's back may not be comfortable despite having a good feature to combat bad posture and contradicts one of the objectives which is to promote a comfortable yet positive environment for posture. Meanwhile, in the software tool such as PostureMinder, it does not monitor and get to know user's posture conditions but merely providing reminder, and this leads to not having a report overall for users to know their posture health overall. Application such as ZenPosture has its own limitations where it only manages to correct posture of upper body, limiting the effect of having a good posture. Additionally, ZenPosture uses webcam to monitor user's posture, and this may lead to inaccurate predictions of their AI algorithms due to lighting or related effects. Besides that, the AI processing the image frames were all done locally and this may impact user's device if their device is not able to handle the process, since webcam and processing of video frames happens simultaneously.

In addition to these studies, a VLM research on monitoring also has its own unspoken limitation, which is, in [13], Idefics model were used to monitor driver's state with a correct response, however, this was done on 1 frame each time, and the issue is, one frame of a video is not sufficient to confirm whether the driver is driving or stopping. However, the model would still give response that conclude driver was driving along with other related action gotten from the frames.

2.5 Proposed Solutions

This project aims to create Monitra, a software system that processes video streams from an external camera to monitor children's studying activities physically. To achieve this, **Vision-Language Model (VLM)**, will be utilized to capture the context of children's action.

Additionally, this project will introduce a **mobile application** to address the user interface limitation in [3]. This enables parents to view the activity of children, enabling them to have better user experience by showing dashboard view that aggregates data across daily, weekly and monthly.

Furthermore, to tackle posture monitoring issues such as the limitation faced by UprightGo which is the comfortability of users wearing a device behind their back, this project solution is the camera that is used for monitoring children activity. By using a **separate camera** to send video frames to an application, this solves the need for wearing devices on the body. Additionally, a separate camera would also solve the issue of ZenPosture where they use a webcam such as lighting of environment and quality of video frames that could impact the result of VLM. However, webcam functionality will still be included in Monitra to provide flexibility to children.

Moreover, when processing frames using VLM, it is crucial to understand the context of what the project aims to achieve to not wrongly or inaccurately process frames. As an example, the study of [13] has a limitation of processing one frame and made a conclusion of result and it reflects to this project scenario as well. Temporal context is crucial to understand the user's movement in a period, especially in activity and posture recognition. Hence, the solution of Monitra is to **process multiple frames** over a period for posture detection to truly confirm the results.

Limitations	Proposed Solution in Monitra
Inability to monitor children studying physically	Using a separate camera
User's device performance lowered due to heavy resource usage	Leveraging proprietary API with efficient frame-grabbing algorithm
Lack of user-friendly reporting	Mobile app for parents to view report
Comfortability issue in posture detection	Using a separate camera
Environment affects video quality using webcam	Using a separate camera
Potential of wrong captioning using VLM on just one image	Process several frames over a period of time for concrete results

Table 2.1 Limitations and Solutions

CHAPTER 3 MONITRA: A prototype to monitor study habits and posturing in children

Throughout this chapter, the architecture of Monitra is explained from a high-level perspective to a low-level perspective, starting with the application overview on how Monitra ecosystem works, to the architecture of each application in detail before specifying the tools used in this project.

3.1 System Architecture Diagram

In this subsection, Monitra's architecture as a whole is discussed, along with the application's use case and a UML diagram, detailing the process flow.

3.1.1 Monitra overview

As shown in Figure 3.1, Monitra's system as a whole, includes two applications, a desktop application and a mobile application. In order for the system to work, study sessions must be created from mobile application which will be stored in a cloud database. This allow the desktop application to continuously poll for study session's data from the cloud database and automatically starts the VLM processing pipeline when matched. The VLM processing pipeline is responsible for connecting to an input, either external camera or webcam by default, then perform frame sampling for VLM processing, sending output to the cloud database. Subsequently, the output is also parsed locally to check whether if subject is present in the environment and their sitting posture classification. Push notification message will be sent and relay by cloud function to the mobile application, while real time notification is triggered on desktop application for bad posture detection. Additionally, the VLM processing pipeline also triggers the live feed feature, where it is responsible for streaming camera input to the mobile application with UDP Hole Punching or falling back to TURN server when needed. As for the mobile application, its responsibilities are to listen for data on the cloud database and further processed the data by desktop application for dashboard visualisation.

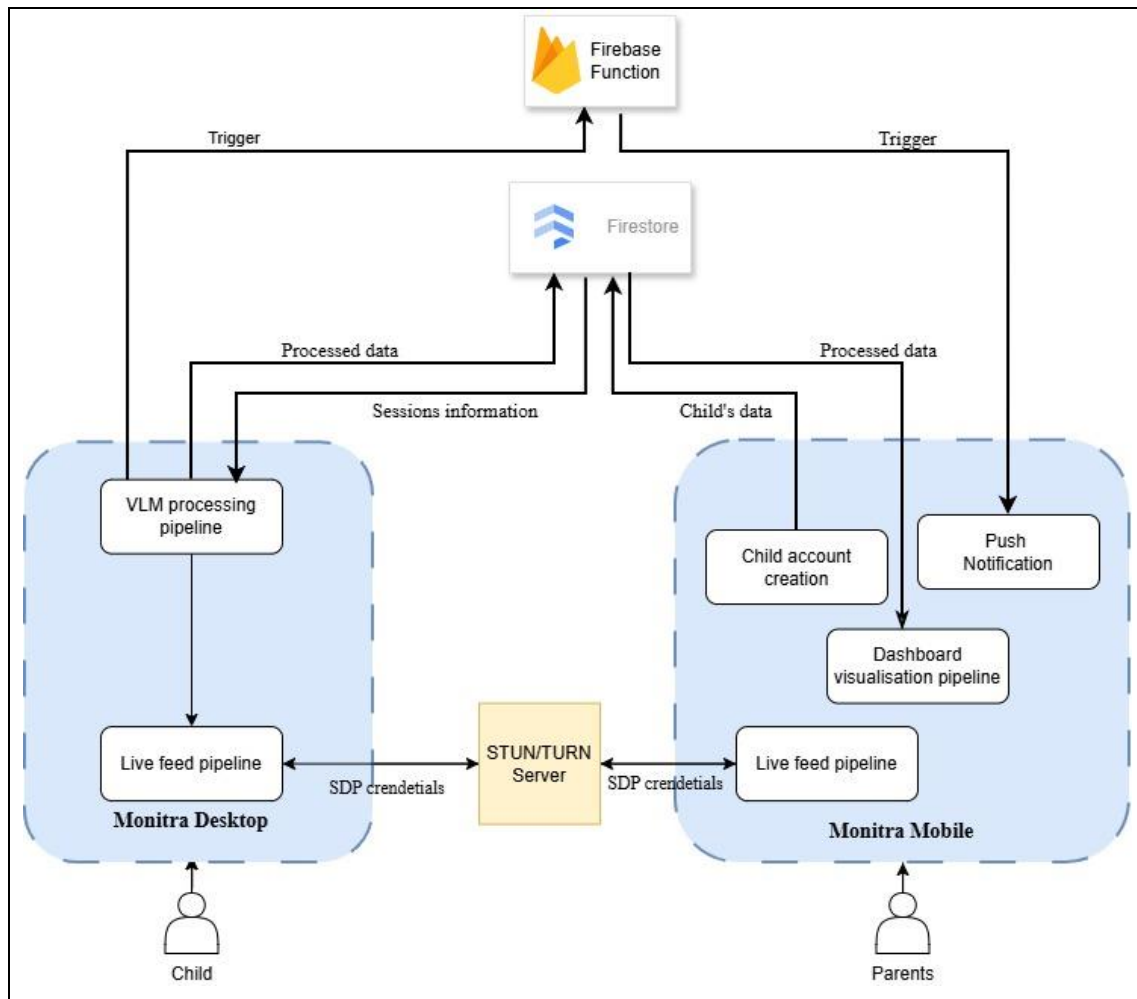


Figure 3.1 General view of Monitra's architecture

3.1.2 Use Case Diagram

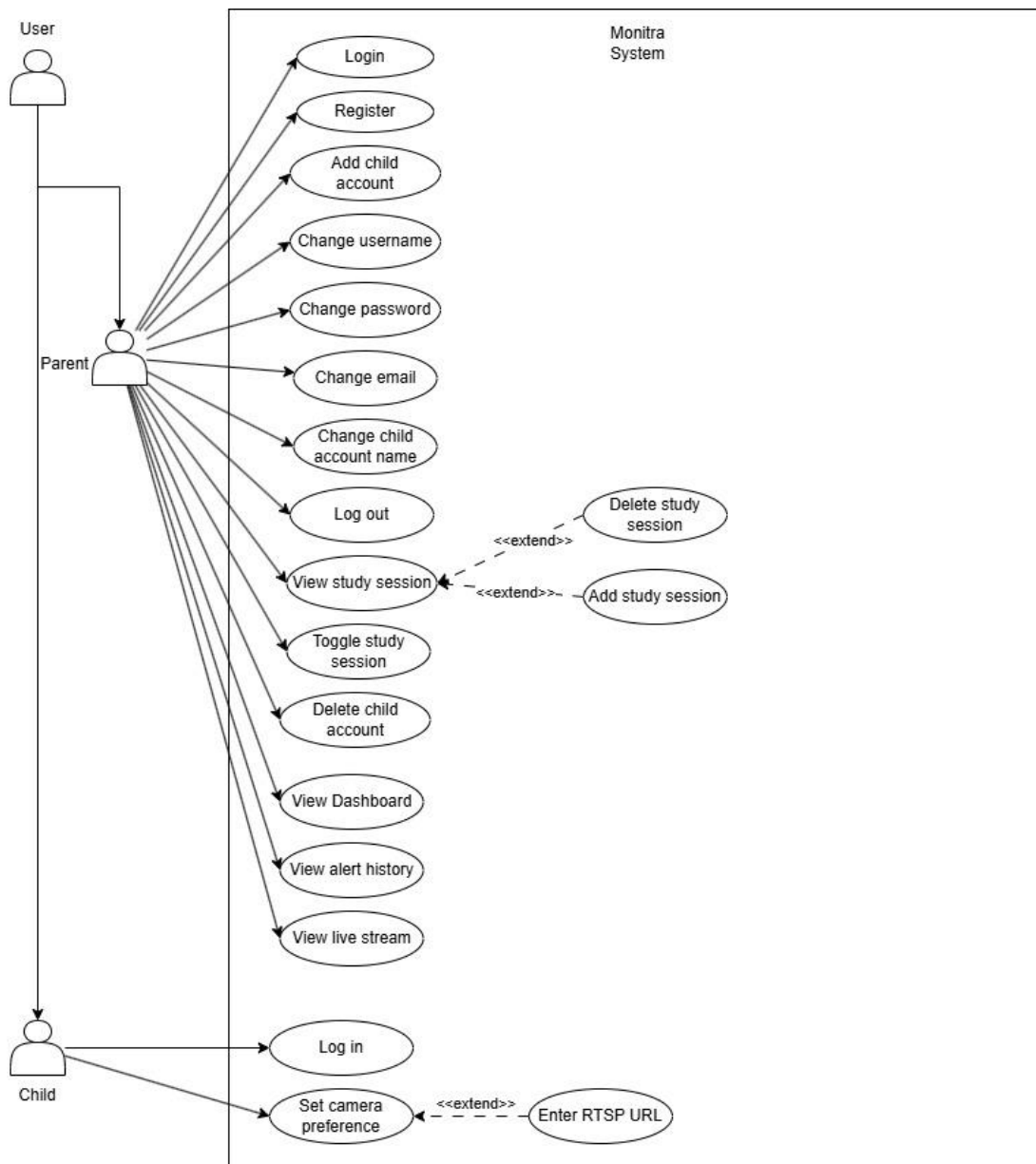


Figure 3.2 Use case diagram of Monitra

3.1.3 Activity Diagram

The nature of Monitra consist of two applications, hence two activity diagrams are modeled out to show the flow of application respectively. Figure 3.3 shows the flow of mobile application; by successfully logging in or registering an account, parents are landed on a homepage. The homepage contains navigation bar that allows navigation to setting's view, while the main page has children cards for each account (if created) and a button to add child account. On the other hand, Figure 3.4 shows the desktop application flow; the user flow consists only of logging in the application and selecting their preferred camera. The VLM Monitoring Pipeline would run indefinitely, triggering live stream pipeline only if any sessions are active. For detailed activity diagram on the mobile application's modules, please refer to Appendix A.

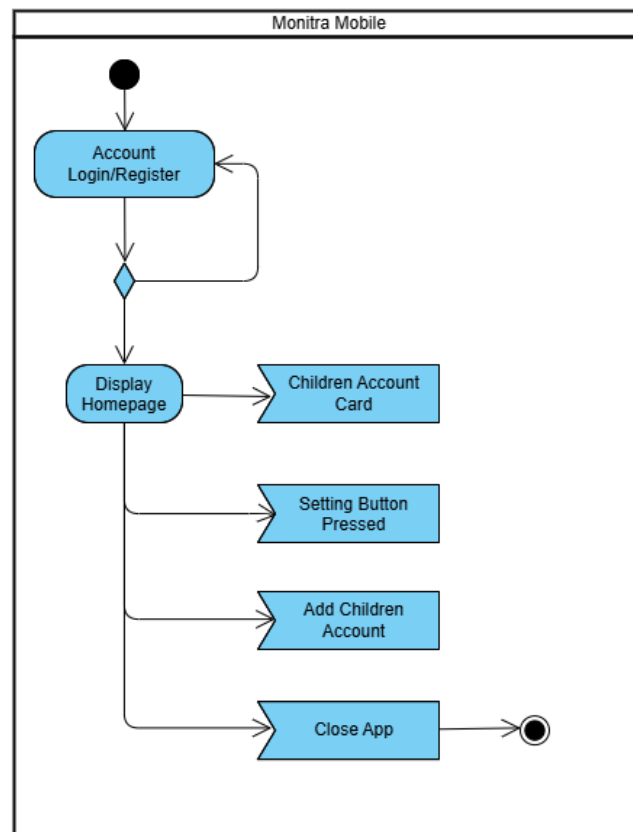


Figure 3.3 Activity diagram of mobile application

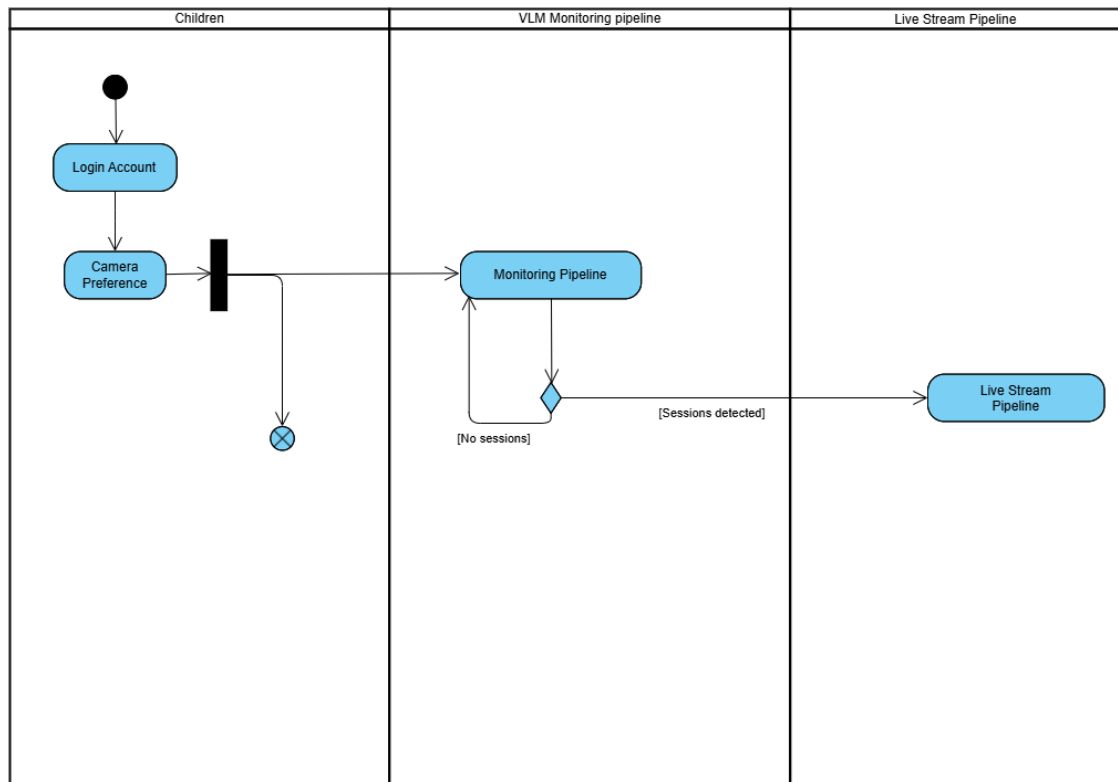


Figure 3.4 Activity diagram of desktop application

3.2 Hardware Setup

This subsection describes the hardware specification utilized for Monitra, which includes a processing unit (laptop), mobile phone and a high-resolution external camera.

Processing unit (laptop)

Description	Specifications
Model	MSI GF63 Thin 10UC
Processor (CPU)	Intel(R) Core(TM) i5-10500H CPU 2.50GHz
Operating System	Windows 11 Pro
Graphic Processing Unit (GPU)	RTX 3050 (4GB)
Memory	8.00 GB (7.82 GB usable)
Storage	477 GB

Table 3.1 Specification of laptop

Mobile Phone Specification

Description	Specifications
Model	RMX3780
Processor (CPU)	MediaTek Dimensity 6100+
Operating System	Android 15
Memory	8.00 GB
Storage	256 GB

Table 3.2 Specifications of mobile phone

External Camera

Description	Specifications
Model	Tapo C220

Table 3.3 Specifications of external camera

3.3 Software Setup

This subsection outlines the software setup process, covering the tools used to develop Monitra, cloud services, and any external services such as API service. Each application in Monitra utilizes different tools which will be described in this subsection.

3.3.1 Monitra Mobile Integrated Development Environment (IDE) and Languages/Framework

Monitra's mobile application is developed in Visual Studio Code (VSCode), on Dart with Flutter framework. Flutter framework is chosen for its currently built-in Material Design 3 (MD3) widgets for ease of development, saving cost while offering sleek user experience. Libraries installed for Flutter to develop Monitra is also shown in Table 3.5.

Visual Studio Code	Version	1.117.0
	Operating System	Windows_NT x64 10.0.26200
Dart	Version	3.9.2
Flutter	Version	3.35.6

Table 3.4 IDE Specification

Libraries	Version	Description
firebase_auth	6.1.3	Firestore authentication for login and registration of accounts
firebase_core	4.3.0	Firestore core library to utilize other Firestore library

firebase_storage		Utilized for storing image proof during push notification.
cloud_firestore	6.1.2	Firestore library to utilize its cloud storage
firebase_messaging	16.1.3	Used for push notification.
provider	6.1.5+1	Library for MVVM architecture in Flutter.
day_picker	2.3.0	Library for ready-to-use day picker API widgets
fl_chart	1.2.0	Core library for dashboard visualisation
flutter_webrtc	1.4.1	Core library for live stream feature in Monitra
flutter_launcher_icons	0.14.4	Library for configuring app's icon/logo
flutter_native_splash	2.4.7	Library for customization of splash screen

Table 3.5 Mobile application libraries

3.3.2 Monitra Desktop Integrated Development Environment (IDE) and Languages/Framework

Monitra's desktop application is developed in Visual Studio 2026, on C# with .NET ecosystem, and Windows Presentation Foundation (WPF) framework for its User Interface. WPF is chosen as the UI framework due to its efficiency and lightweight ability when it comes to resource management compared to Flutter. Monitra's desktop application requires the processing of frames either from webcam or external camera, hence resource management is critical to eliminate potential performance bottlenecks. Table 3.6 details the IDE environment as well as external libraries required for Monitra's development recorded in Table 3.7.

Visual Studio 2026	Version	18.2.1
	Operating System	Windows_NT x64 10.0.26200
.NET Framework	Version	4.8.09221
WPF	Version	4.8

Table 3.6 IDE for desktop application

Libraries	Version	Description
CommunityToolkit.Mvvm	8.4.0	Library to perform MVVM data binding in WPF
FirebaseAuthentication.net	4.1.0	Firebase library in .NET for authentication
Google.Cloud.Storage.V1	4.14.0	SDK for manual REST API communication with Firebase Firestore
WPF-UI	4.2	Library for polished design in WPF framework
Google.GenAI	1.5.0	Library for API integration.
OpenCVSharp4	4.13.0	.NET OpenCV library for camera connection
OpenCVSharp4.runtime.win	4.13.0	.NET OpenCV library for windows
SIPSorcery	10.0.3	Core library for UPD Hole Punching with WebRTC standard

SIPSorcery.Media.Encoders	10.0.4	Library to encode camera input for live streaming
Microsoft.Toolkit.Uwp.Notifications	7.1.3	Library for real-time notification on windows for bad posture detection.

Table 3.7 Desktop application libraries

3.3.3 Monitra's Cloud Service

In both Monitra's applications, cloud storage is leveraged for ease of data retrieval. Hence, Firebase services are chosen for their ease of integration in Flutter mobile application, specifically Firestore, Firebase Storage, and Firebase Authentication. Firestore is a NoSQL cloud database that allows nested data architecture, storing data beyond what is allowed in a relational database such as boolean value, and timestamp objects. Firebase Storage on the other hand is utilized for its capability to store images that are needed for Monitra's proof of snapshot required for push notifications.

3.3.4 Monitra's VLM

The adoption of VLM API is the objective of Monitra's study, which is to leverage its zero-shot capabilities. Among the available API providers, Gemini's multimodal architecture is particularly advantageous because it was trained natively across modalities from the ground up, allowing for superior spatial reasoning and fine-grained visual comprehension in complex VQA tasks compared to stitched-together models [17]. Hence, this project chooses Gemini-2.5-flash due to its compatibility of price-performance, an low-latency feature.

3.4 Application Architecture

Extensive research and study has been taken to determine the suitable architecture for both desktop and mobile applications. According to [14][15], developing application in .NET WPF often uses the Model View ViewModel (MVVM) architecture for greater maintenance and scalability, utilizing a concept named Data Binding. Similarly, developing an application in Flutter framework is also recommended to follow the MVVM architecture as mentioned in [16].

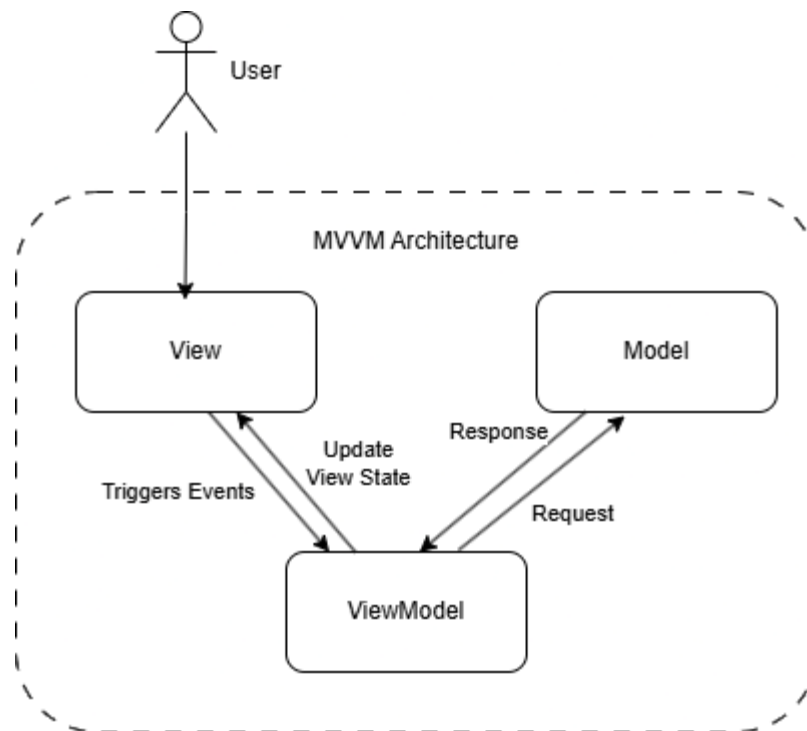


Figure 3.5 Application Architecture

CHAPTER 4 MONITRA SYSTEM DESIGN

This chapter outlines the system design of Monitra. We first present the system block diagram to illustrate the high-level interaction between components. Following this, detailed process flowcharts are provided. To demonstrate the practical implementation of these workflows, each flowchart is immediately accompanied by the wireframes design and screenshots of the corresponding user interface (UI) from the completed application.

4.1 System Block Diagram

System block diagram below visualises the components and interactions of Monitra on a higher level of abstraction, leaving only the building blocks of the system. The primary entry point lies in the **GUI block** and the **Management and Monitoring block**. GUI block represents Monitra's desktop application user interface, allowing children to select a camera input which is subsequently saved by the session manager in the AI Processing block. Second entry point originates from Monitra's mobile application, allowing parents to create and manage child's account session, saving the results into the cloud storage. Additionally, this block is also responsible for displaying processed data through the Dashboard UI and tries to establish a peer-to-peer connection in response to the AI Processing block when needed.

The **AI Processing** block is the core operation of Monitra, where the Session Manager is responsible to continuously checks if study session is to be conducted. Once confirm, the Frame Manager would start the video capture according to the selected input preference, routing video frames to VLM Analyzer for processing, while simultaneously triggering the Live Feed Manager to establish a peer-to-peer connection.

Finally, the **Cloud Synchronization Block** handles centralized data storage and event routing. It actively listens for state changes and event triggers generated by the AI Processing block, subsequently relaying these updates as push notifications to the receiver within the mobile application.

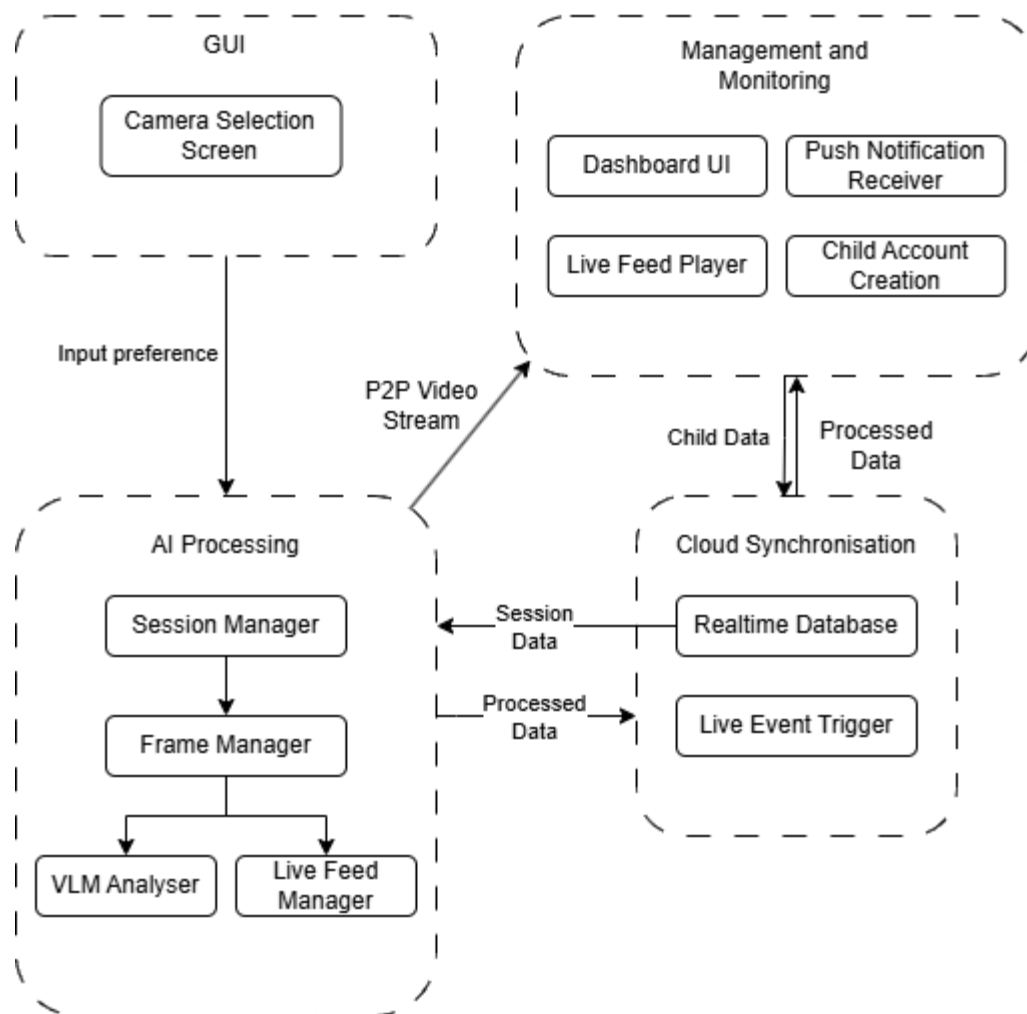


Figure 4.1 System block diagram of Monitra

4.2 System Flowchart and Implementation

This subsection details the procedural logic for both the mobile and desktop applications of the Monitra system. To illustrate the translation from system logic to visual design, the flowcharts are presented alongside their initial wireframes and finalized user interface (UI) screenshots. It should be noted that certain module, specifically the detailed settings pages, alert history, and live feed viewer were developed during later iterative phases. Consequently, these features bypassed the initial wireframing stage and are documented strictly through their process flowcharts and final UI implementations.

4.2.1 System Flowchart and Implementation for mobile application

Monitra's mobile application flowchart covers the features of login, registration, creating child accounts, changing user's settings such as username, password, email address, and editing child account information. Additionally in child accounts, the flowchart is also shown for each feature such as dashboard visualisation, creating study sessions, alert history, and live feed session.

4.2.1.1 Registration of new user (Parent)

By utilising Firebase Authentication, user's credential can be check initially when entering the application. Missing credentials imply that the user is either a new user or previously deleted their account or application. Hence, users with credentials are directly routed to the landing page, while missing credentials or new users are needed to perform login or registration respectively.

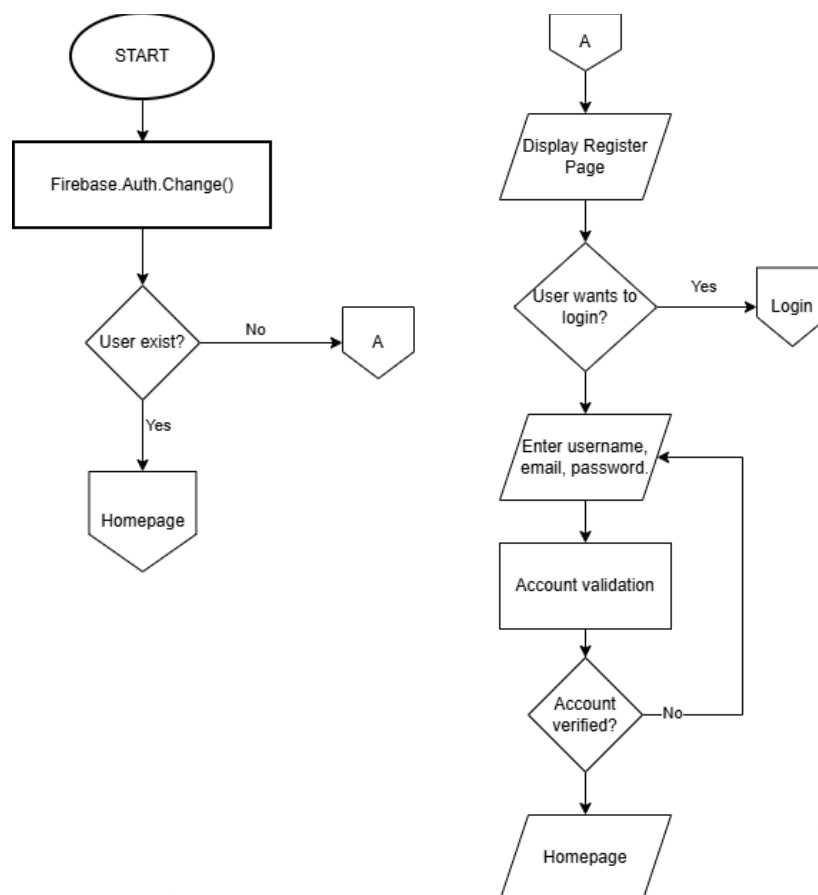
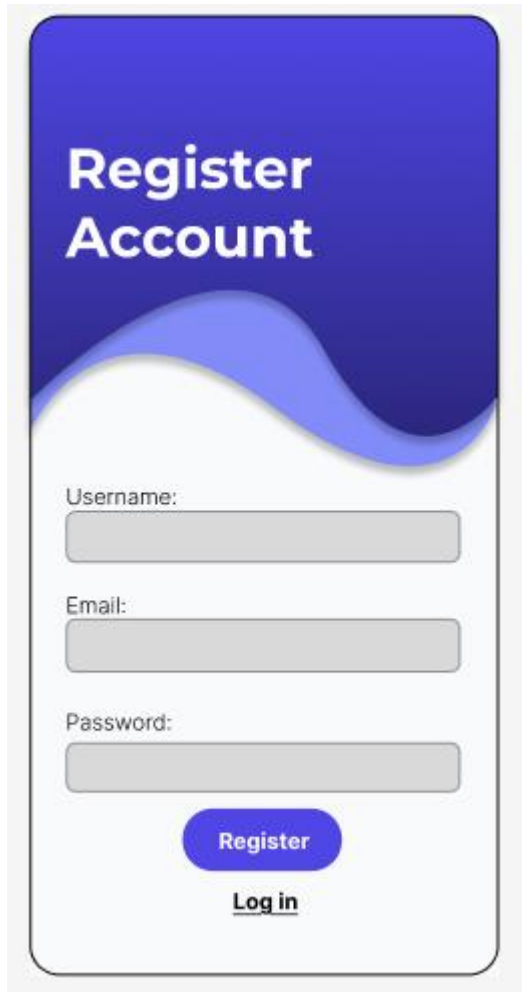
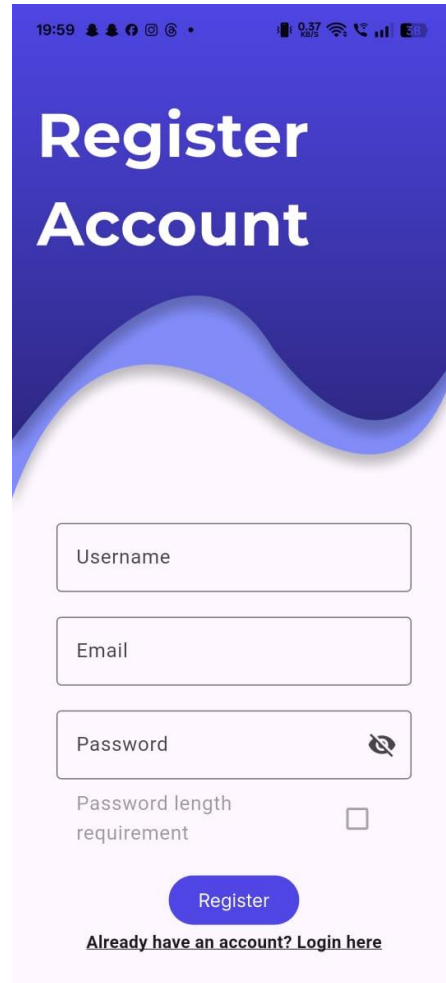


Figure 4.2 Flowchart of Registration



A wireframe for a registration page. The top section has a dark blue header with the text "Register Account" in white. Below the header is a light blue wavy decorative element. The main content area is white and contains three input fields labeled "Username:", "Email:", and "Password:". Below these fields is a blue "Register" button and a "Log in" link.

Figure 4.3 Wireframe for Registration



A screenshot of a mobile application registration page. The top status bar shows the time 19:59 and various icons. The page has a dark blue header with the text "Register Account" in white. Below the header is a light blue wavy decorative element. The main content area is light pink and contains three input fields labeled "Username", "Email", and "Password". The "Password" field has a toggle icon. Below these fields is a "Password length requirement" checkbox and a blue "Register" button. At the bottom, there is a link that says "Already have an account? Login here".

Figure 4.4 Screenshot of Registration
page

4.2.1.2 Login of existing user (Parent)

For users who deleted their application but has an existing account, users can attempt login with email and password, which will be validated by Firebase Authentication, and routes to application homepage upon successful login.

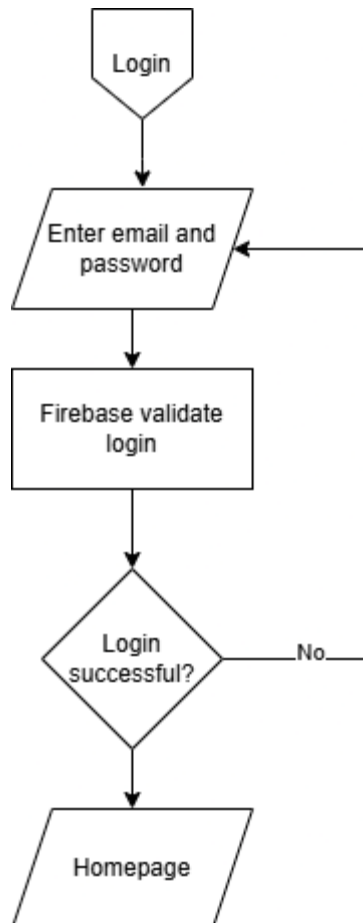


Figure 4.5 Flowchart of Login

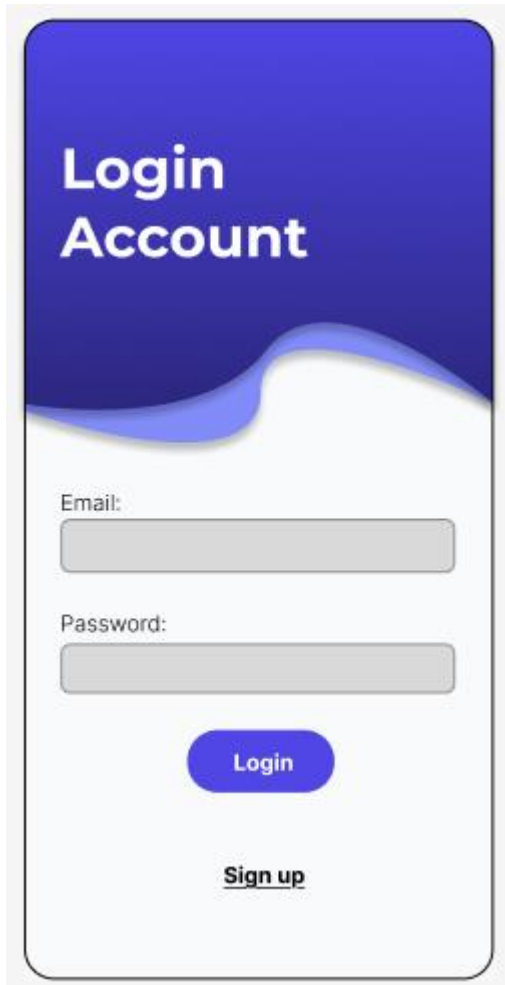


Figure 4.6 Wireframe of Login

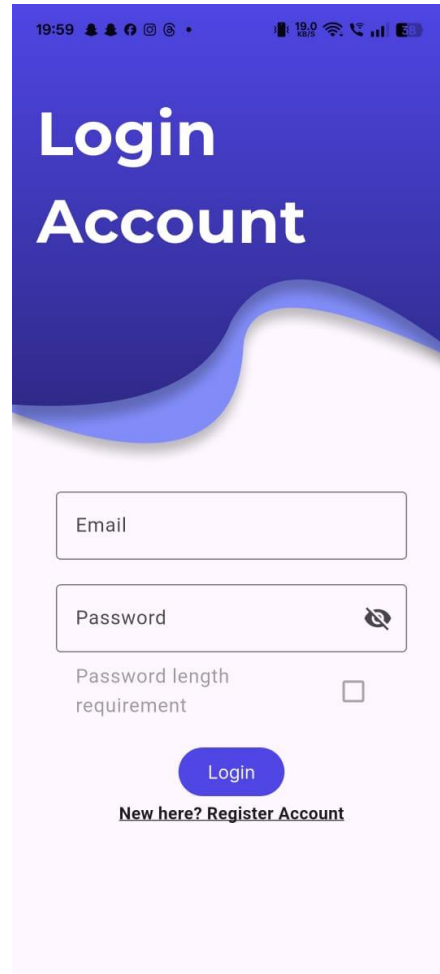


Figure 4.7 Screenshot of Login page

4.2.1.3 Homepage of application

Users directed to homepage of application are able to view their children accounts or add new child accounts as shown in below flowchart. Creating a child account requires users to enter children's name and age, which is subsequently stored in Firestore while simultaneously updating the homepage with the newly created account. Additionally, pressing any children card in the homepage leads to a master page that contains independent section of dashboard, alert history, creating study session, and live feed page.

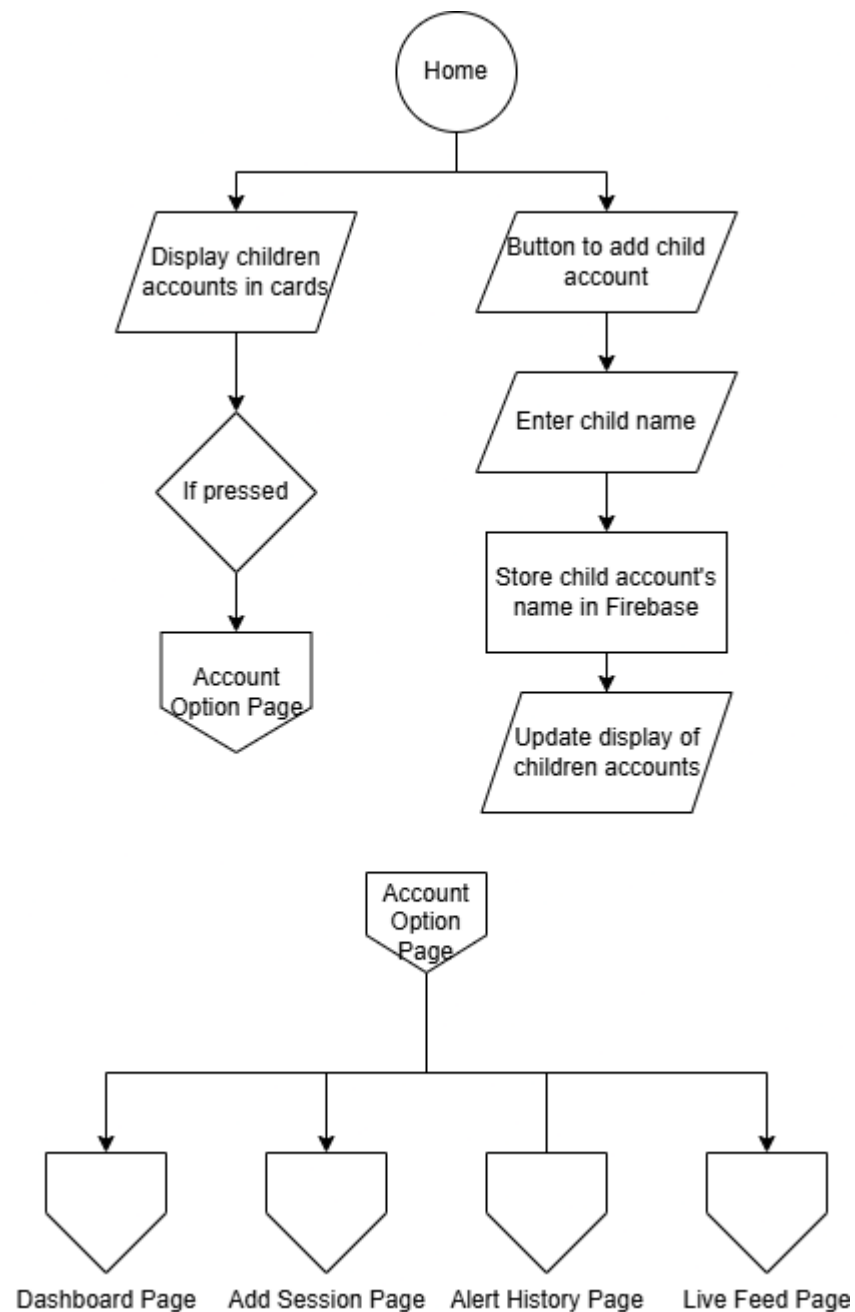


Figure 4.8 Flowchart of homepage

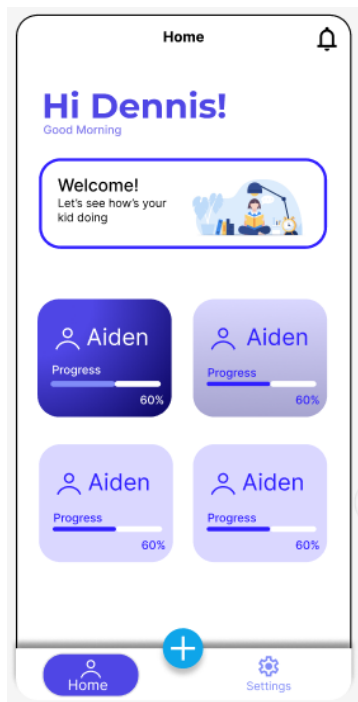


Figure 4.9 Wireframe of homepage

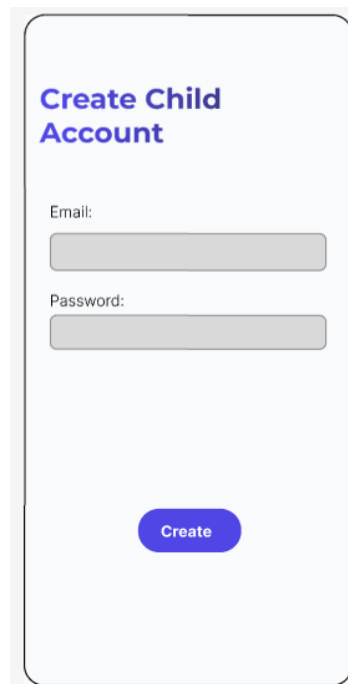


Figure 4.10 Wireframe of child creation page

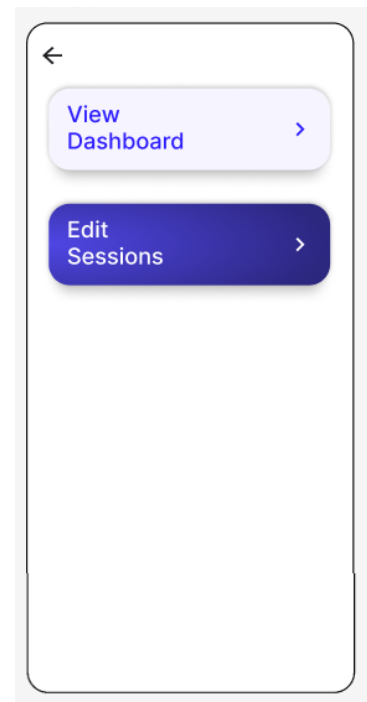


Figure 4.11 Wireframe of account option page

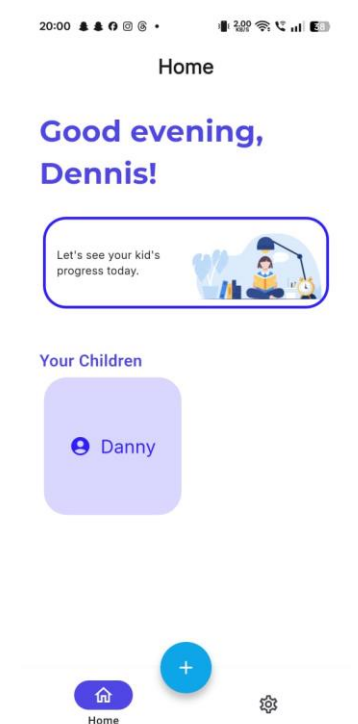


Figure 4.12 Screenshot of homepage

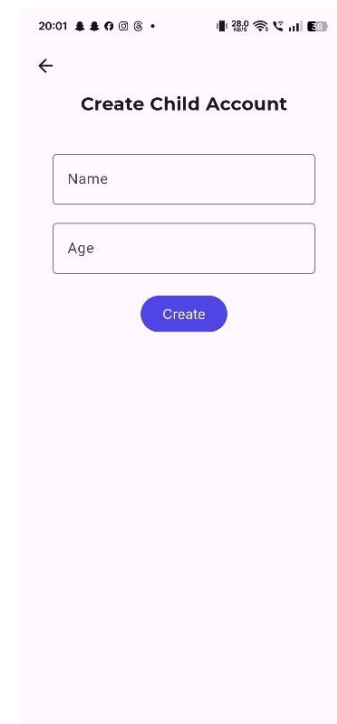


Figure 4.13 Screenshot of child creation page

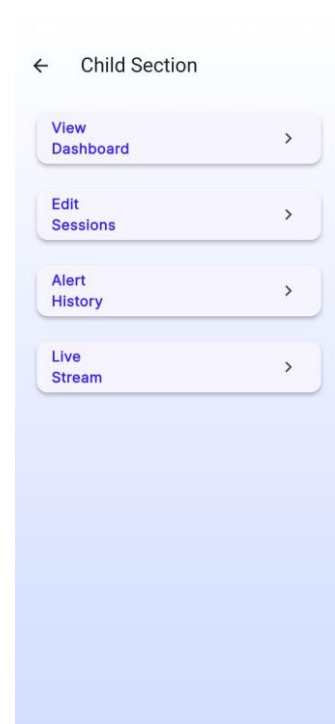


Figure 4.14 Screenshot of account option page

4.2.1.4 Settings page

By navigating to settings page, users are allowed to edit credentials such as username, email, or password, each with its own separate logic and page. Additionally, users can choose to log out the application, redirecting them to the login page after Firebase Authentication successfully logs out the user.

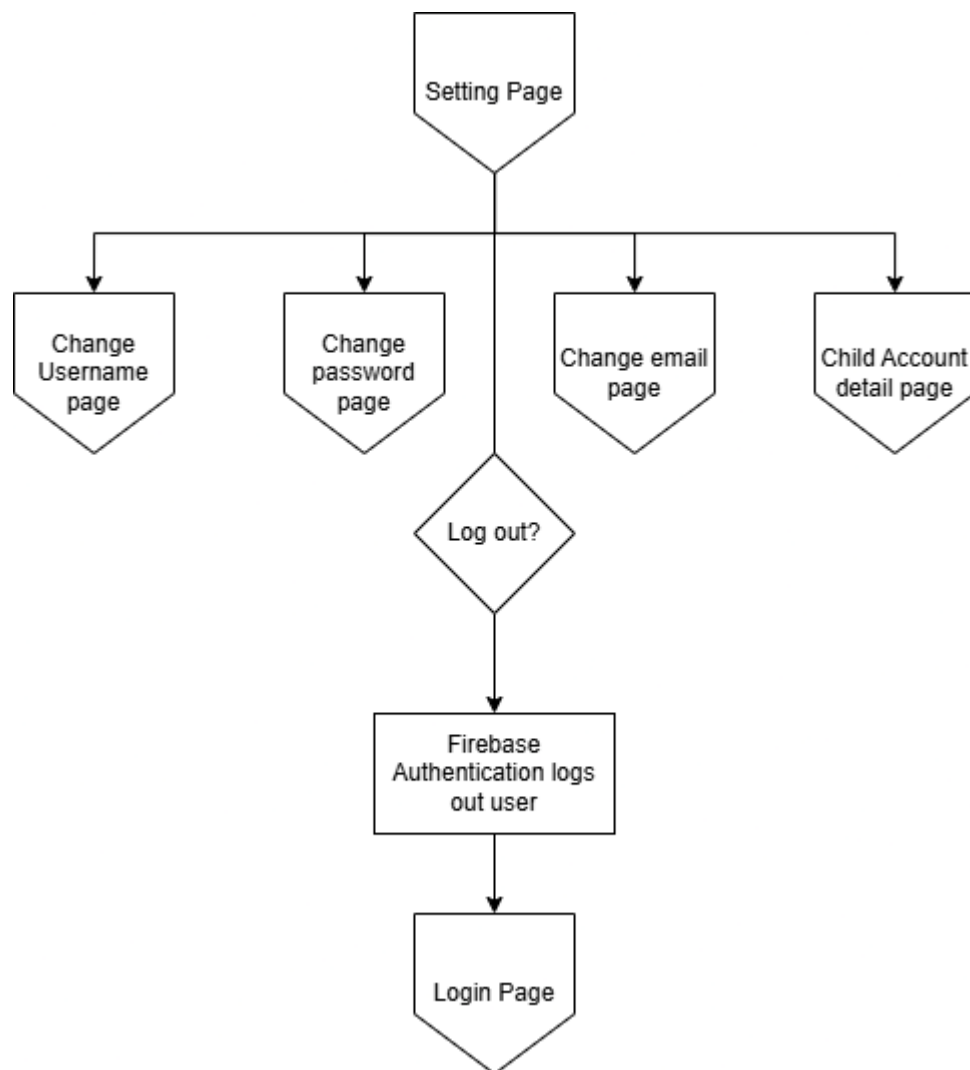


Figure 4.15 Flowchart of setting page

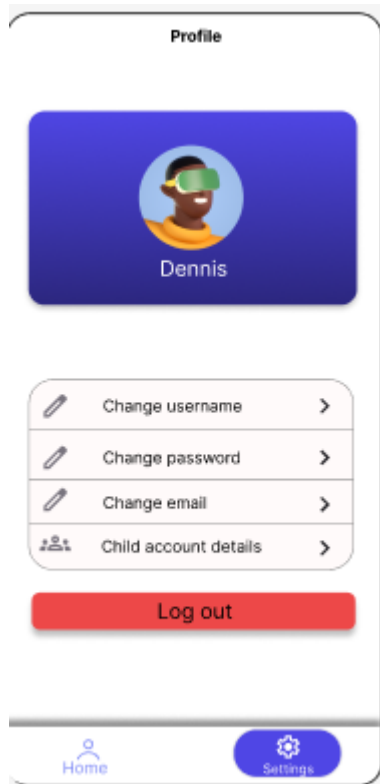


Figure 4.16 Wireframe of setting page

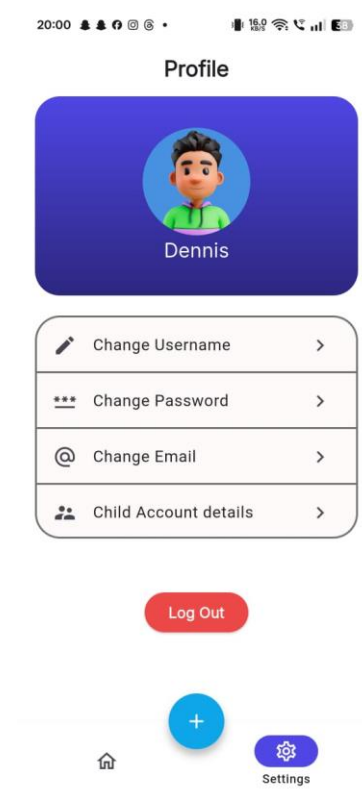


Figure 4.17 Screenshot of setting page

4.2.1.5 Changing user credentials and child account details

Changing username and email has the same flow as shown in below flowchart, where the update happens in Firestore and Firebase Auth respectively. However, changing password requires a requirement to be met before triggering Firebase Auth for update. Furthermore, child account detail option allows for child account name changes, by entering new a name, the data would be updated in Firestore. Each of these features routes the users back to Setting page after any successful operation.

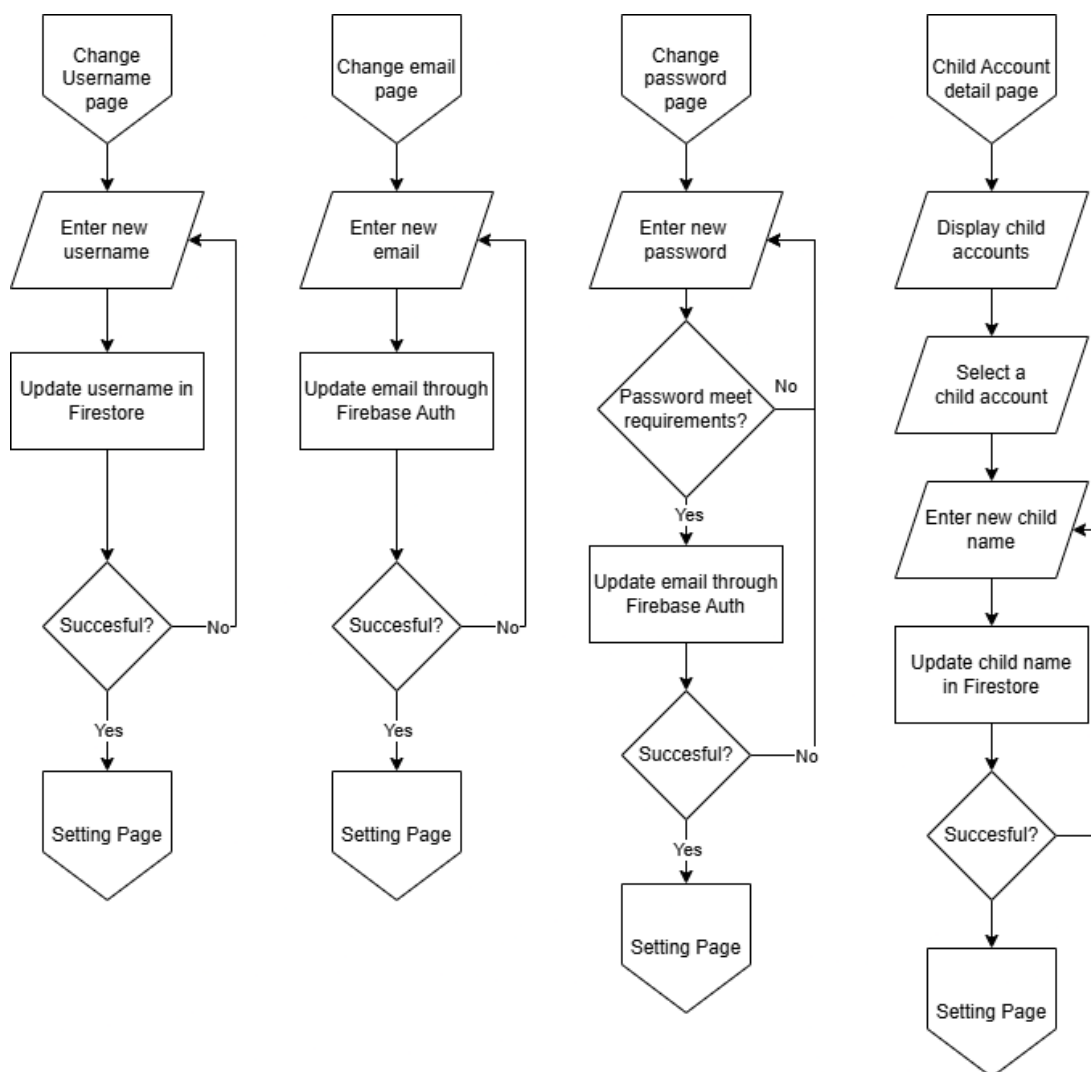


Figure 4.18 Flowchart of Settings option

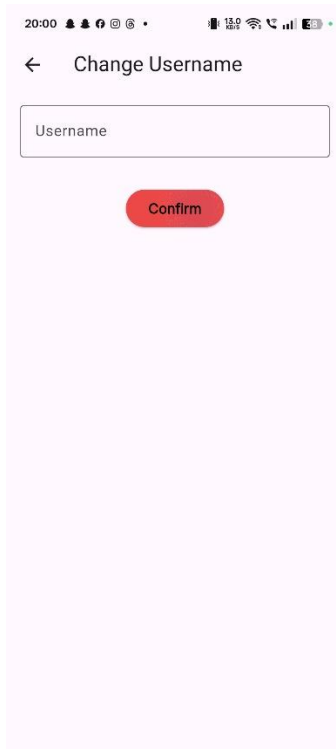


Figure 4.19 Screenshot of changing username page

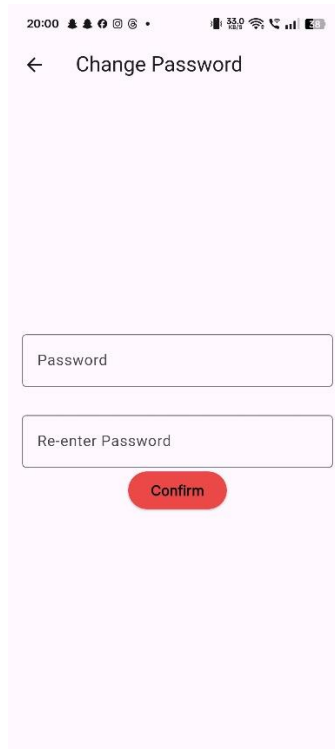


Figure 4.20 Screenshot of changing password page

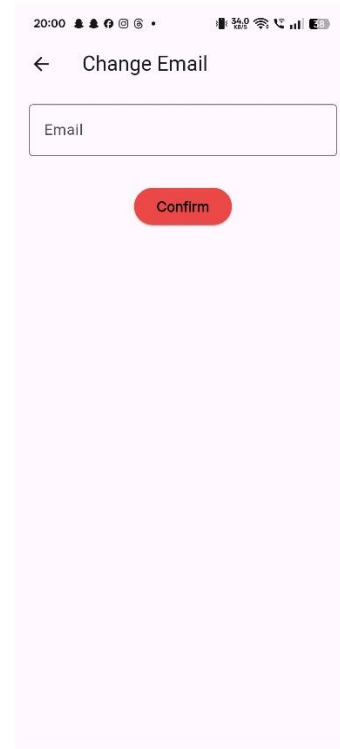


Figure 4.21 Screenshot of changing email page



Figure 4.22 Screenshot of selecting children account page

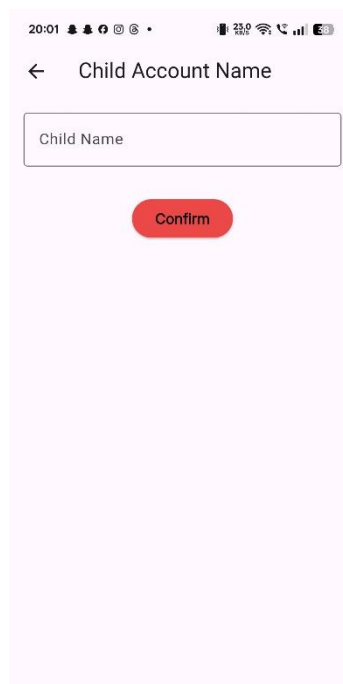


Figure 4.23 Screenshot of changing children name page

4.2.1.6 Dashboard page

Figure 4.24 shows the simple logic of dashboard page, whereby navigating to the page triggers Firebase data fetching, and processing the fetched data for visualisation in terms of daily, weekly, and monthly.

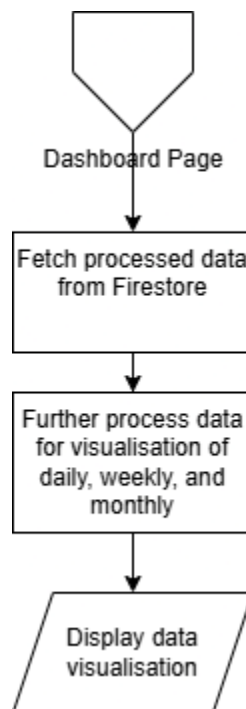


Figure 4.24 Flowchart of Dashboard page



Figure 4.25 Dashboard page of daily

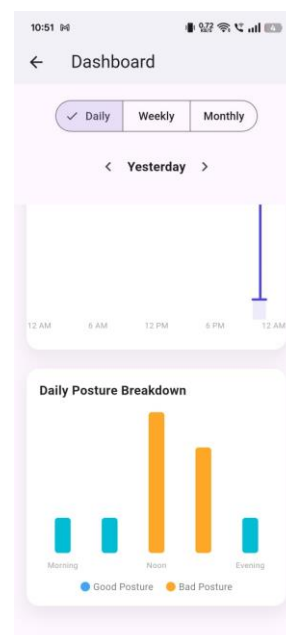


Figure 4.26 Dashboard page of daily



Figure 4.27 Dashboard page of weekly



Figure 4.28 Dashboard page of weekly



Figure 4.29 Dashboard page of monthly



Figure 4.30 Dashboard page of monthly

4.2.1.7 Edit, delete, or view study session page

By further navigating into Edit session from Account options, users are able to view existing sessions fetched from Firestore as shown from the flow in below flowchart. Each session has the option to be toggled on or off, saving its value as a boolean value in Firestore, indicating if a session is activated. Additionally, users can also create new session by picking a time and day or saving the new session to Firestore with default values.

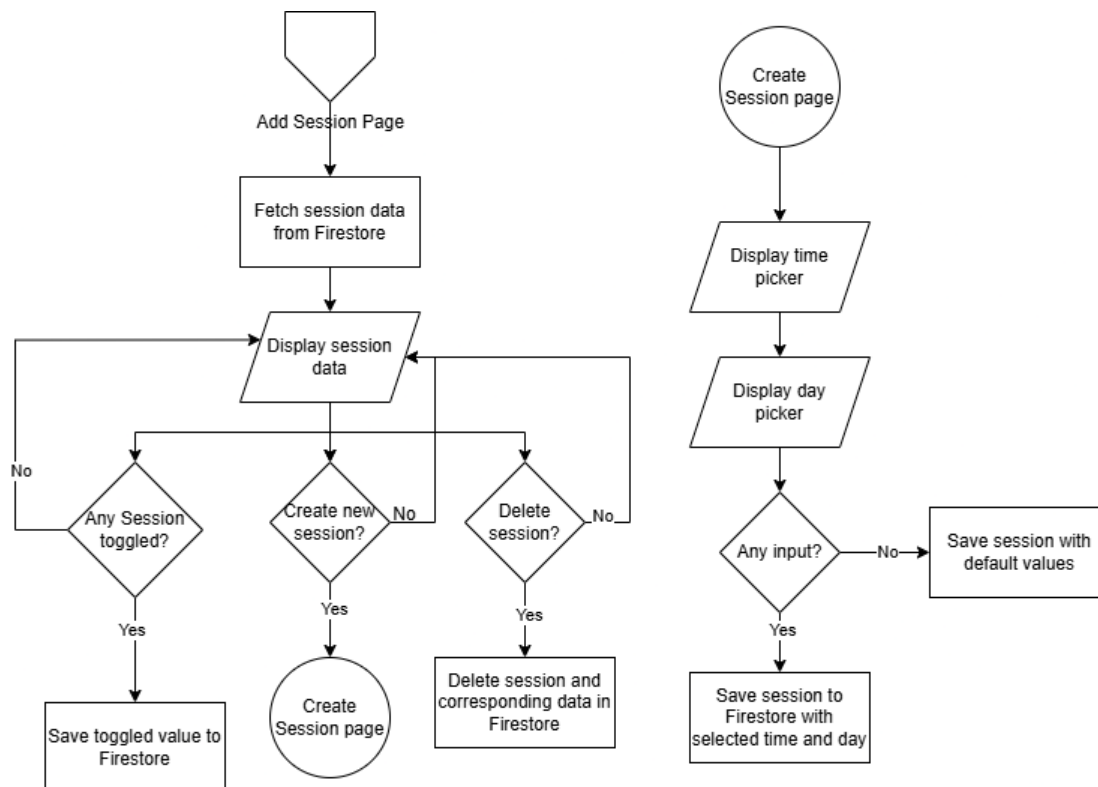


Figure 4.31 Flowchart of edit session

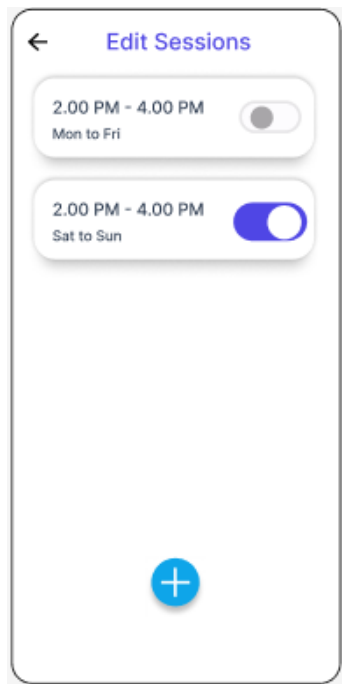


Figure 4.32 Wireframe of study sessions toggle

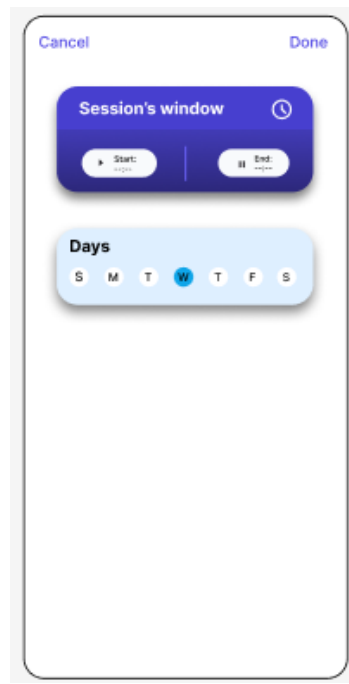


Figure 4.33 Wireframe of creating session

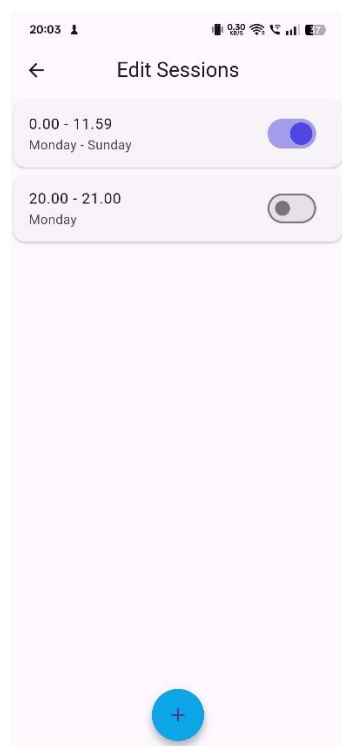


Figure 4.34 Screenshot of study sessions toggle

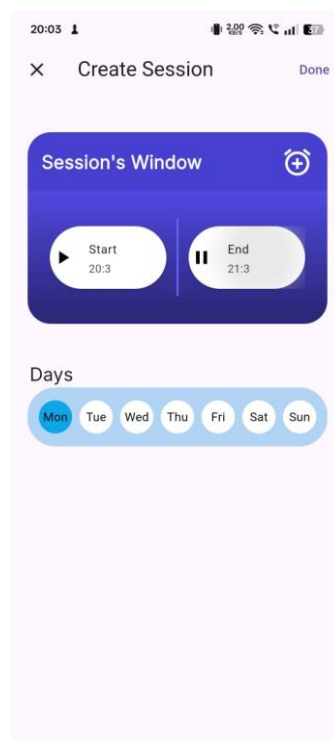


Figure 4.35 Screenshot of creating session

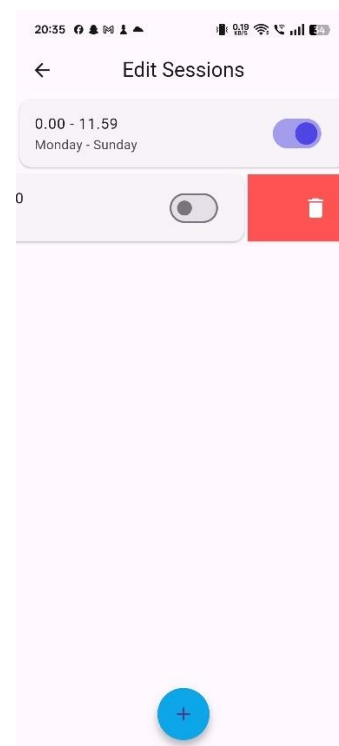


Figure 4.36 Screenshot of deleting session

4.2.1.8 Alert history page

In alert history page, the application first fetched existing history from Firestore and display each alert found. By pressing any of the alert cards, users can navigate further to view the detail of the specific alert as shown in both the flowchart and the screenshot of UI.

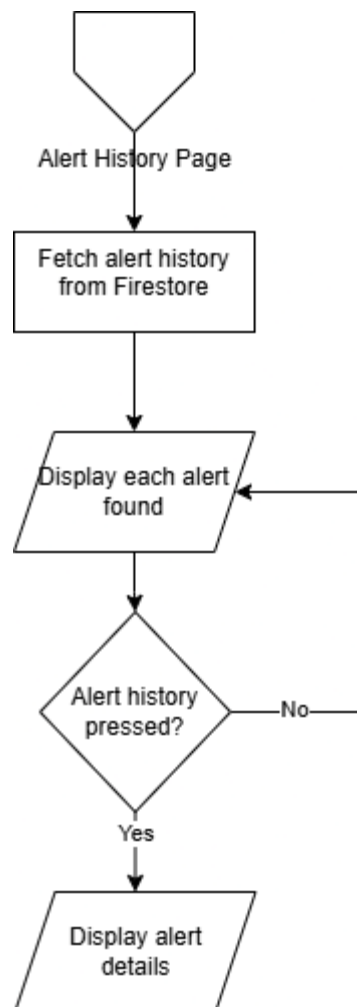


Figure 4.37 Flowchart of alert history

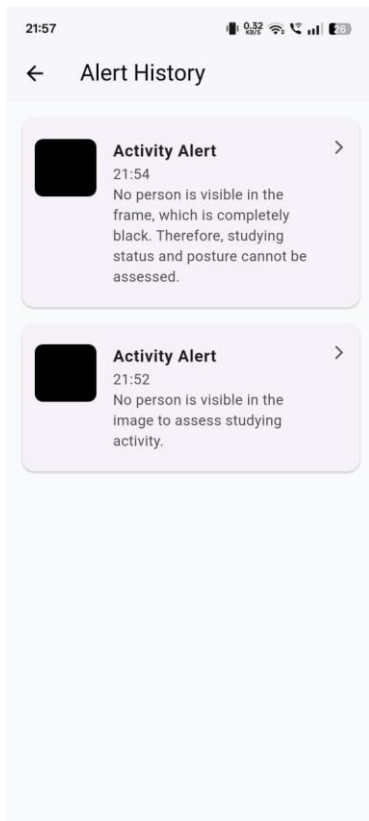


Figure 4.38 Screenshot of alert history page

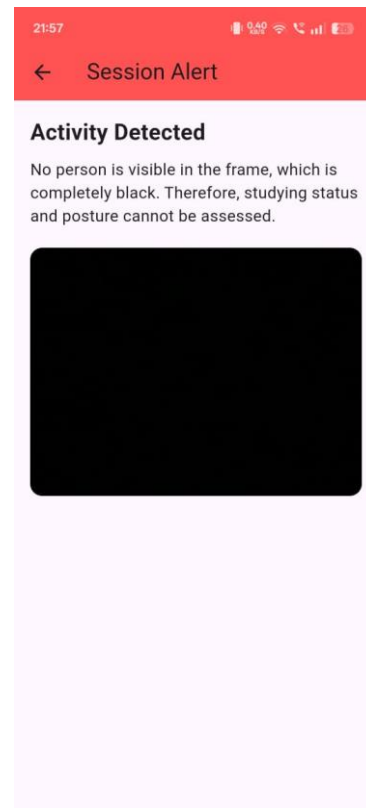


Figure 4.39 Screenshot of alert detail page

4.2.1.9 Live feed page

Figure 4.40 illustrates the conditional logic on the WebRTC connection phase. Due to the unpredictable nature of Network Address Translation (NAT) and strict enterprise firewalls, a direct peer-to-peer connection is not always guaranteed. To account for this, the flowchart models a standard fallback mechanism. The process initiates by querying a STUN server to attempt a direct connection. A decision node then evaluates the success of the NAT traversal. If successful, the flow proceeds directly to establishing the peer-to-peer link. Conversely, if traversal fails, the system executes a conditional fallback path, allocating a relay through a TURN server (using Metered) and routing the media stream through this relay to ensure uninterrupted delivery.

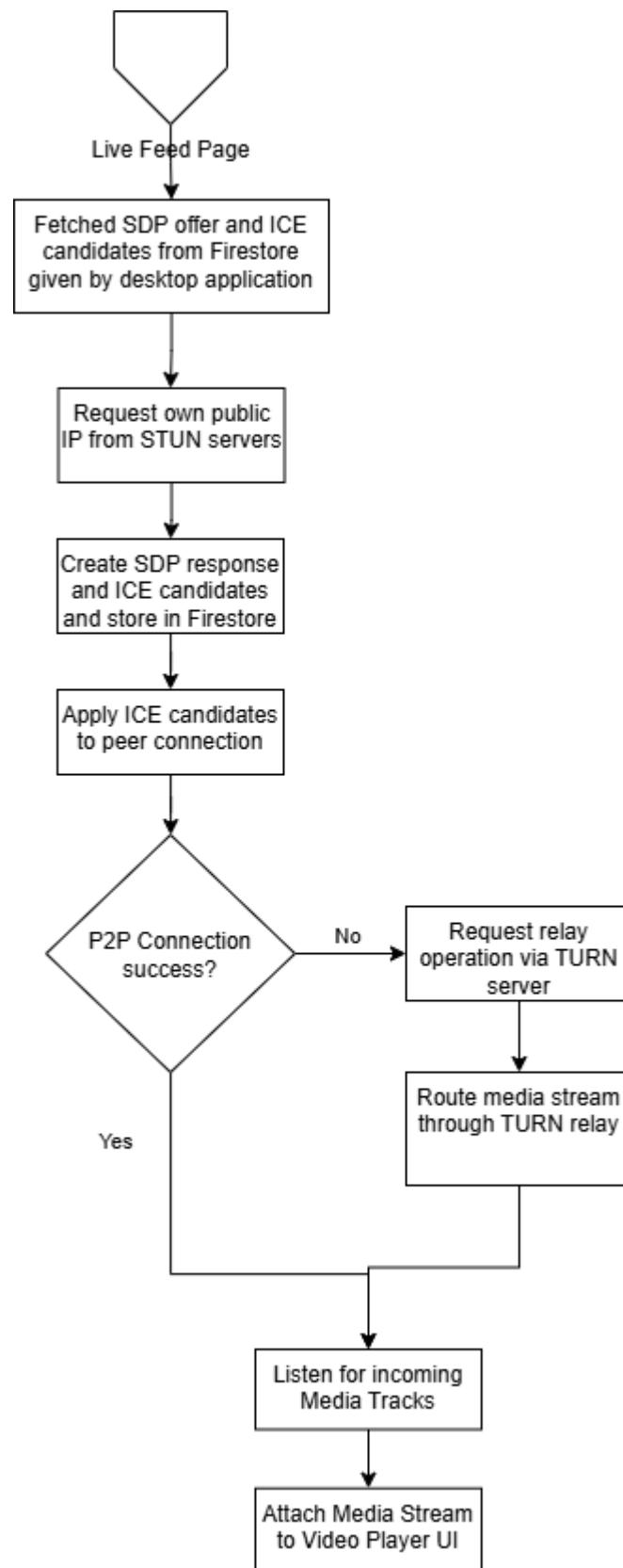


Figure 4.40 Flowchart of Live Feed page

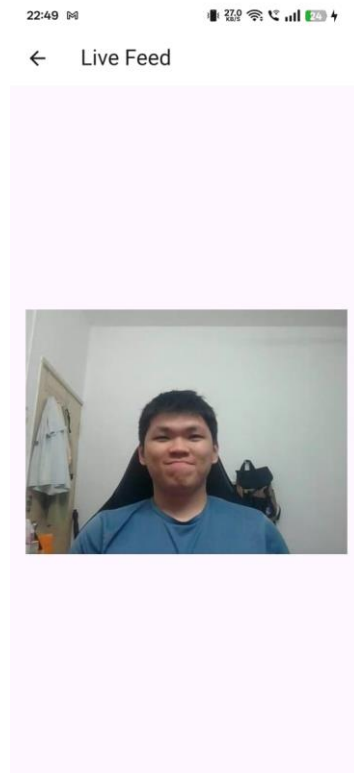


Figure 4.41 Screenshot of Live Feed page

4.2.2 System Flowchart and Implementation for desktop application

Monitra's desktop application flowchart covers the features of login in as the parent and child, selecting camera preferences, and several backend processes. Backend processes includes Session Manager, Frame Manager, VLM Processing, Live Feed Manager, push notification for subject detection, and real-time notification for bad posture detection.

4.2.2.1 Login for parent and child

Figure 4.42 illustrates the two-stage authentication logic utilized by the Monitra desktop application. The execution flow initiates with the primary security gate: the Parent Login UI. Upon receiving the parent's credentials, the system processes the request via Firebase Authentication. A decision node evaluates the response; if authentication fails, the logic loops back to prompt the user again. Upon a successful login, the system transitions to the secondary authentication phase, invoking the Child Login Module. Here, the interface prompts for the child's specific account credentials. These inputs undergo backend validation to verify the existence of the child's account. Similar to the primary gate, invalid entries loop back to the input state, whereas a successful validation advances the system execution to the Homepage Module.

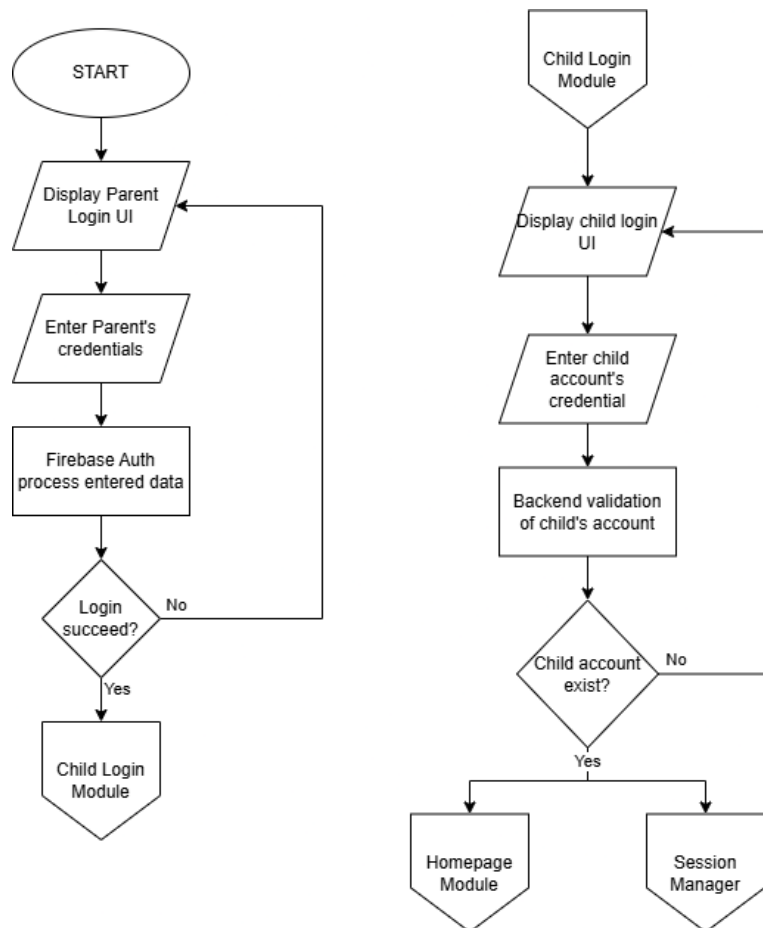


Figure 4.42 Flowchart of parent login

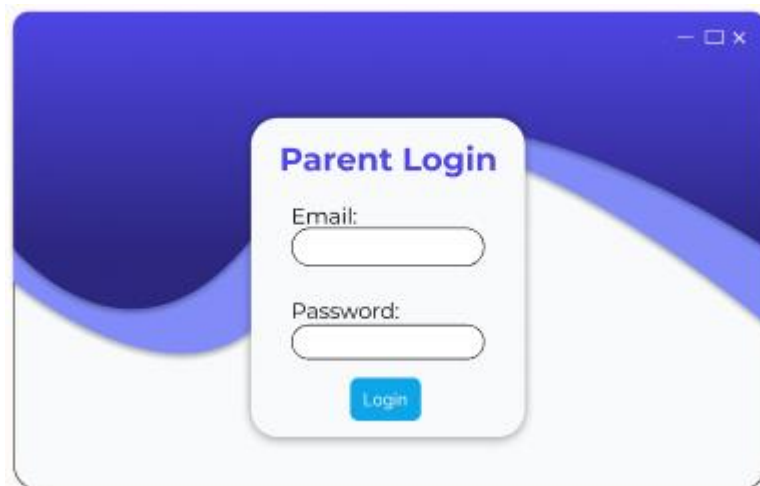


Figure 4.43 Wireframe of parent login

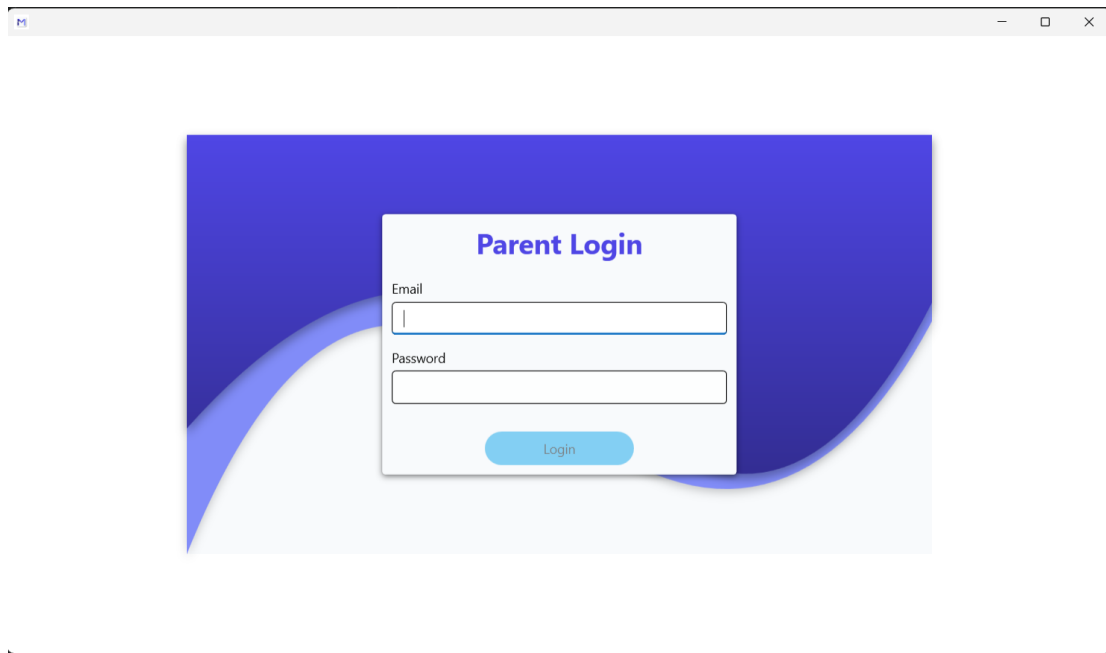


Figure 4.44 Screenshot of parent login

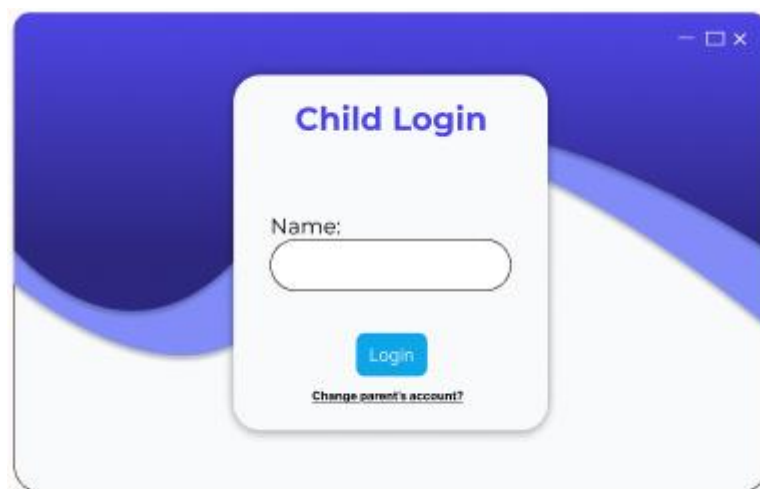


Figure 4.45 Wireframe of child login

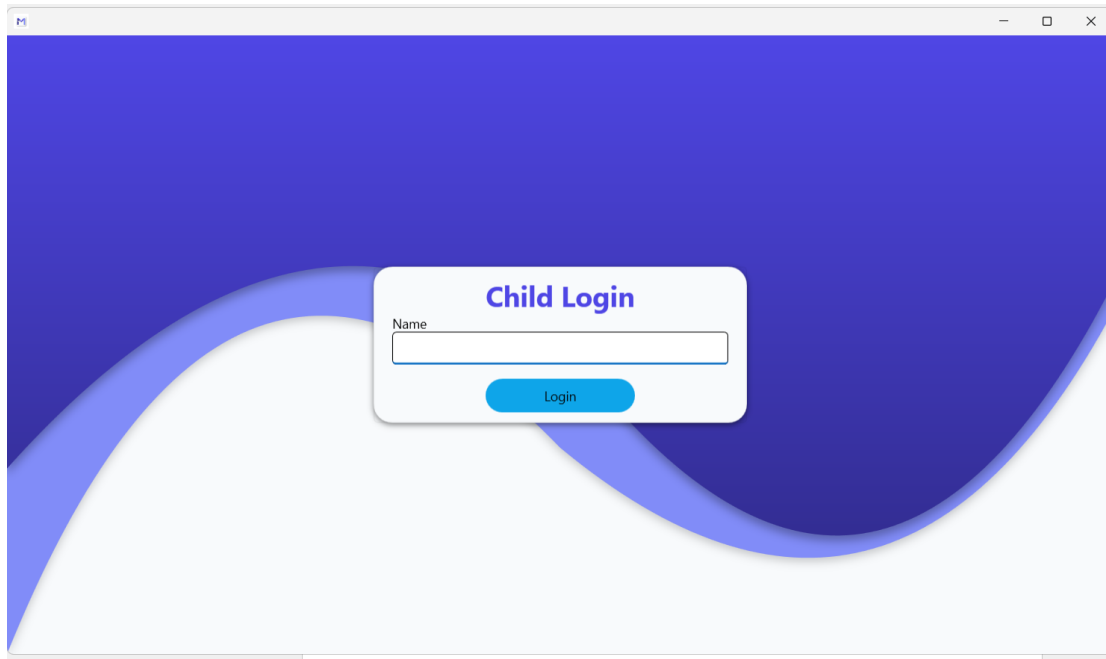


Figure 4.46 Screenshot of child login

4.2.2.2 Homepage of desktop application

Figure 4.47 details the procedural logic for the application's camera connection module. The sequence initiates from the Homepage Module, which transitions the user into the camera setup phase. The system first evaluates whether the desired hardware is present within a locally cached list of previously connected cameras. If a historical record exists, the system automatically retrieves the configuration and proceeds to connect. Conversely, if the camera is unrecognized or new, the flow diverts to a manual data entry state, prompting the user to input the specific RTSP (Real-Time Streaming Protocol) URL and authentication credentials. Both the automated and manual pathways converge at the core execution step, where the system utilizes the OpenCV library to initiate the network connection. A subsequent decision node validates the connection status; a failure triggers an error message and redirects the user back to the start of the connection module for retry, whereas a successful initialization outputs a confirmation message and terminates the setup workflow.

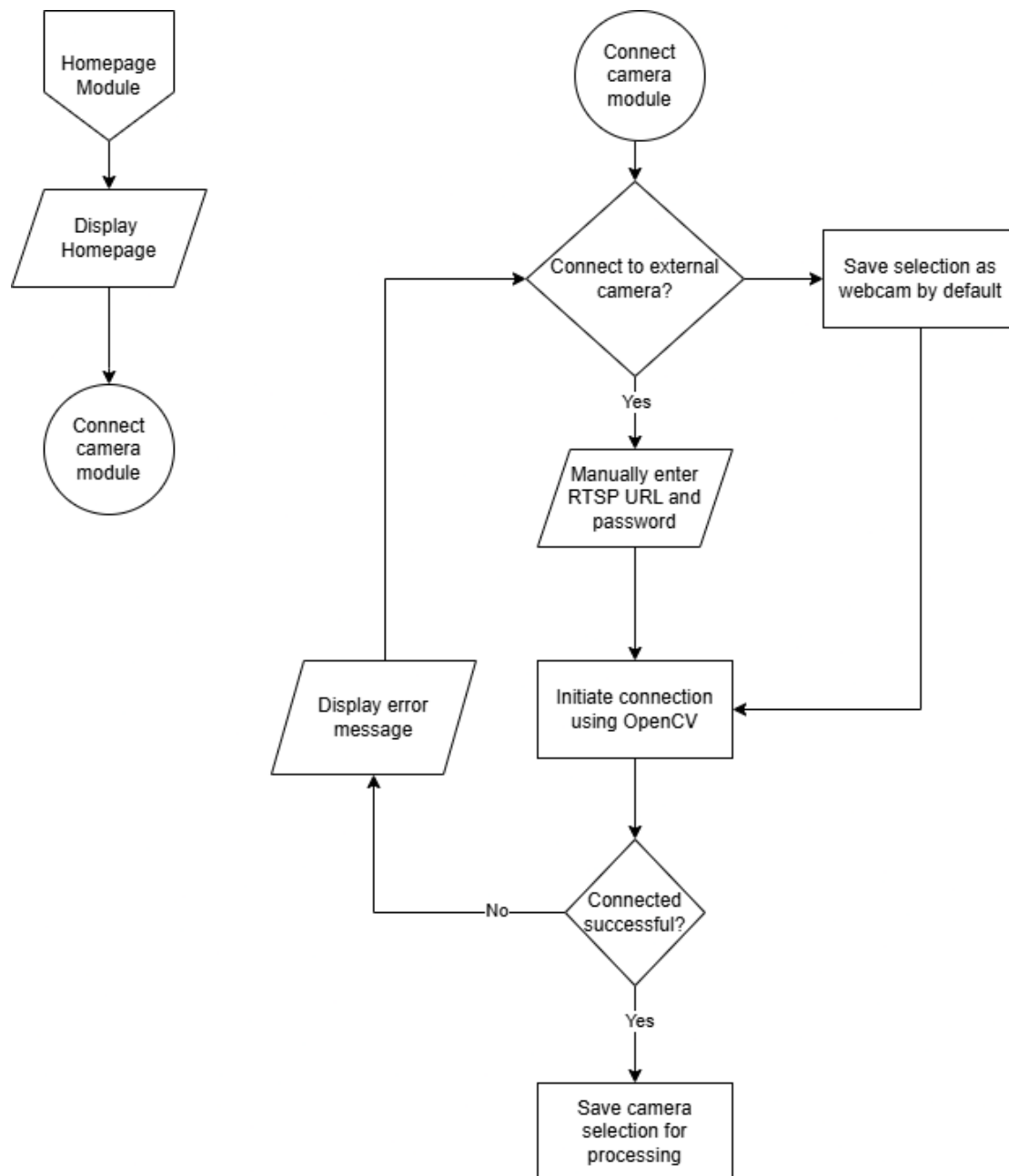


Figure 4.47 Flowchart of homepage and camera page

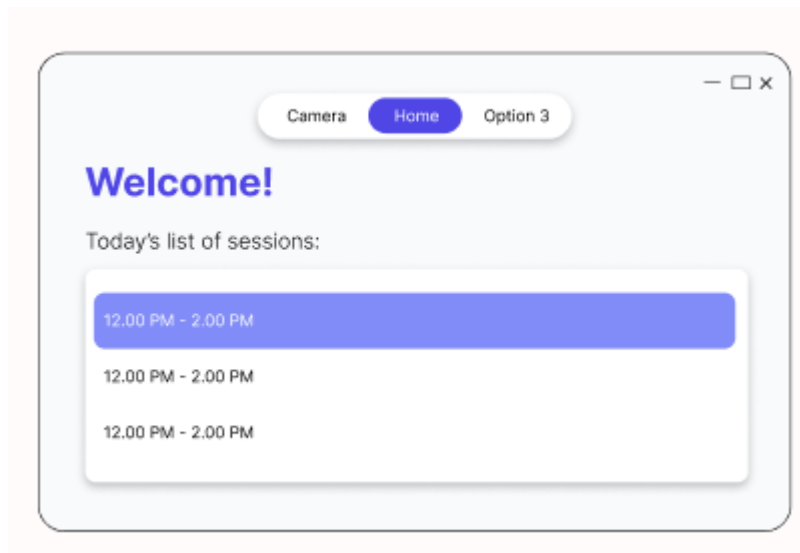


Figure 4.48 Wireframe for desktop homepage

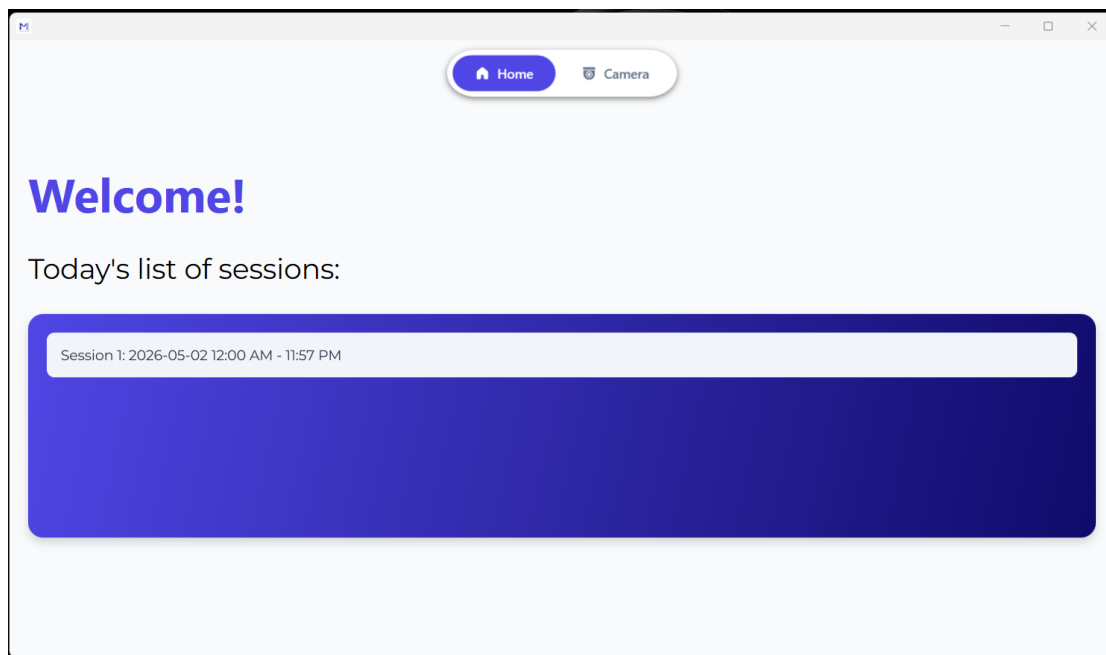


Figure 4.49 Screenshot of desktop homepage

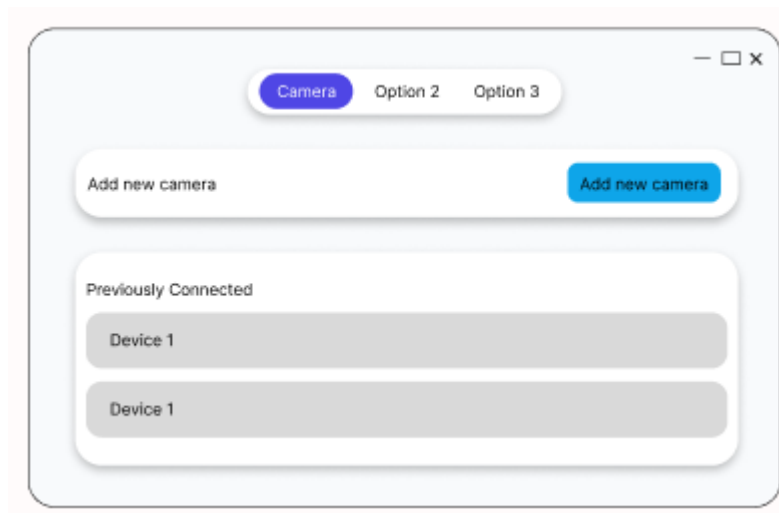


Figure 4.50 Wireframe of camera page

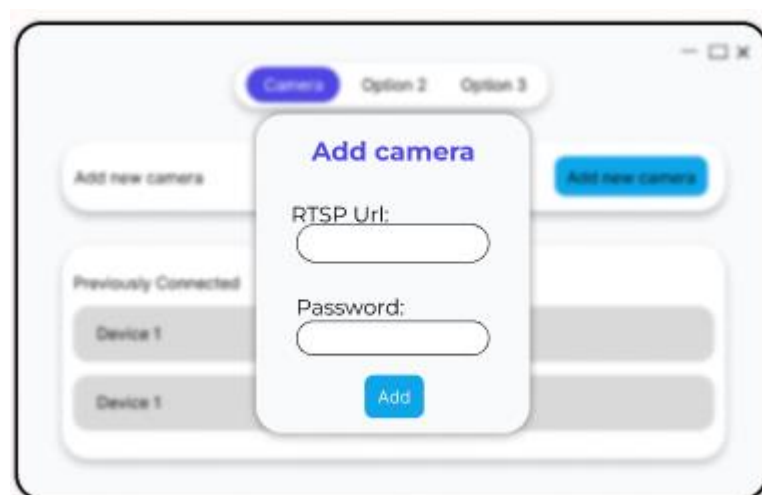


Figure 4.51 Wireframe of RTSP UI

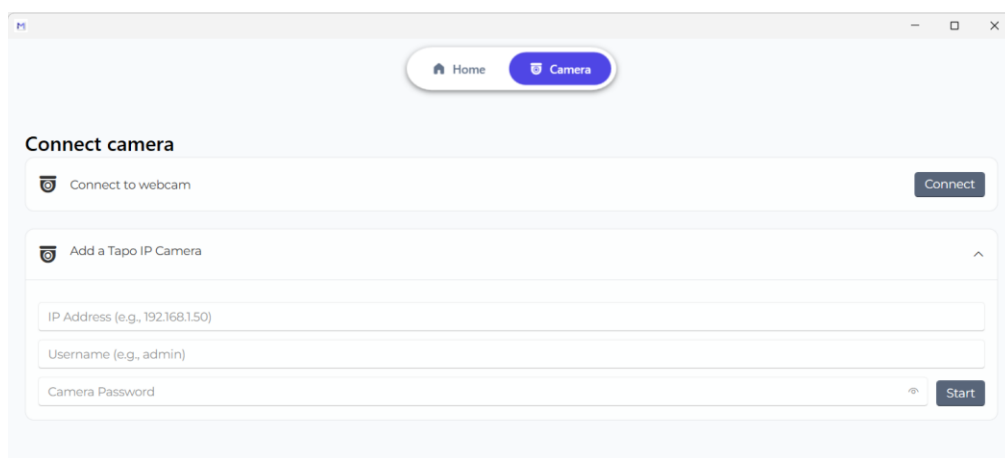


Figure 4.52 Screenshot of desktop camera page and RTSP UI

4.2.2.3 Desktop application Session Manager

Figure 4.53 illustrates the background execution logic of the Monitra Session Manager. Once user log in, the background flowchart dictates a continuous polling loop executing every 30 seconds. The system validates the currently selected child profile and conditionally fetches updated schedule parameters from Firestore. It then evaluates the current system time against the active schedules. Based on this evaluation, the logic sets a boolean flag to determine the required monitoring state, which triggers a final decision node to either start or stop the background camera module.

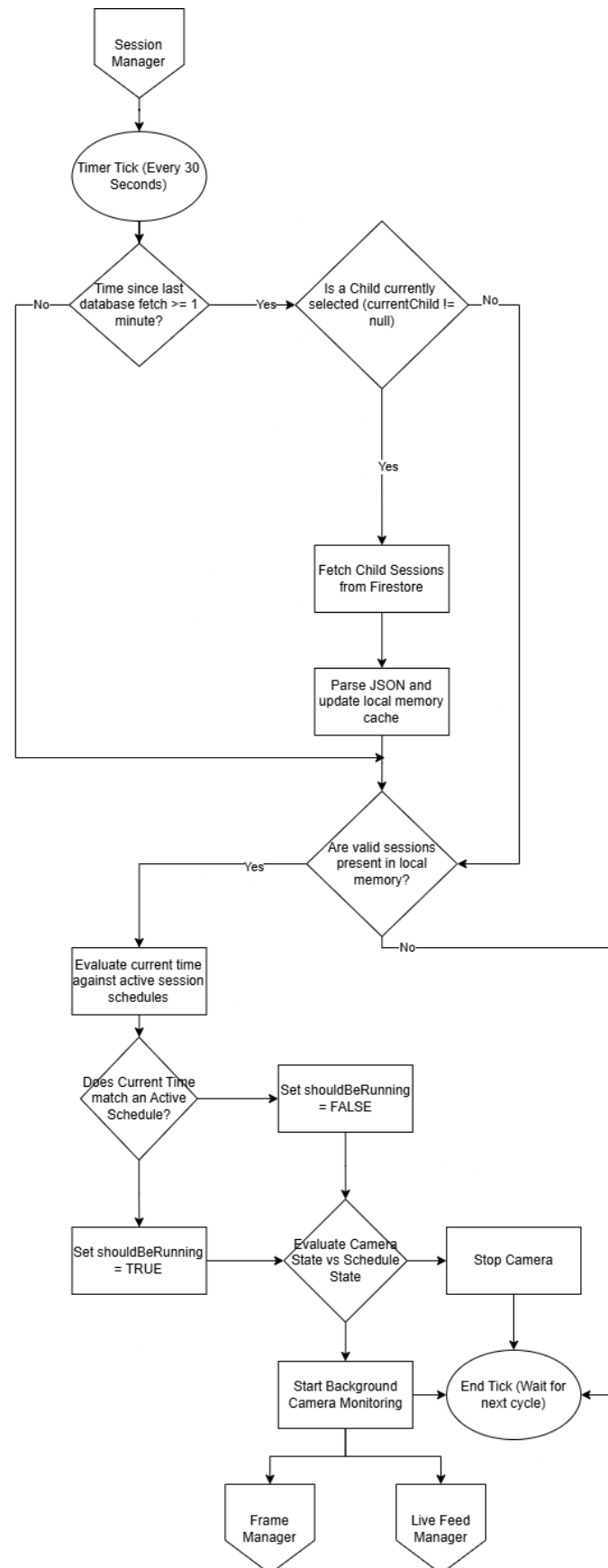


Figure 4.53 Flowchart of Session Manager

4.2.2.4 Desktop application Frame Manager

Figure 4.54 shows the logic on the application's frame sampling mechanism. Designed to optimize computational load, the system alternates between an Active sampling phase and a Cooldown resting phase. Upon receiving a video frame, the logic first evaluates the system's current state. During the 10-second Active window, the system captures frames at mathematically distributed intervals (every 2.5 seconds), encoding and storing them in a temporary batch. Once the Active window elapses, the compiled batch is pushed to a blocking queue for downstream VLM processing, and the system transitions into a 20-second cooldown state. During this cooldown, incoming frames are intentionally ignored to prevent queue saturation. Upon completion of the cooldown timer, the system resets and returns to the Active state, creating a continuous, regulated sampling loop.

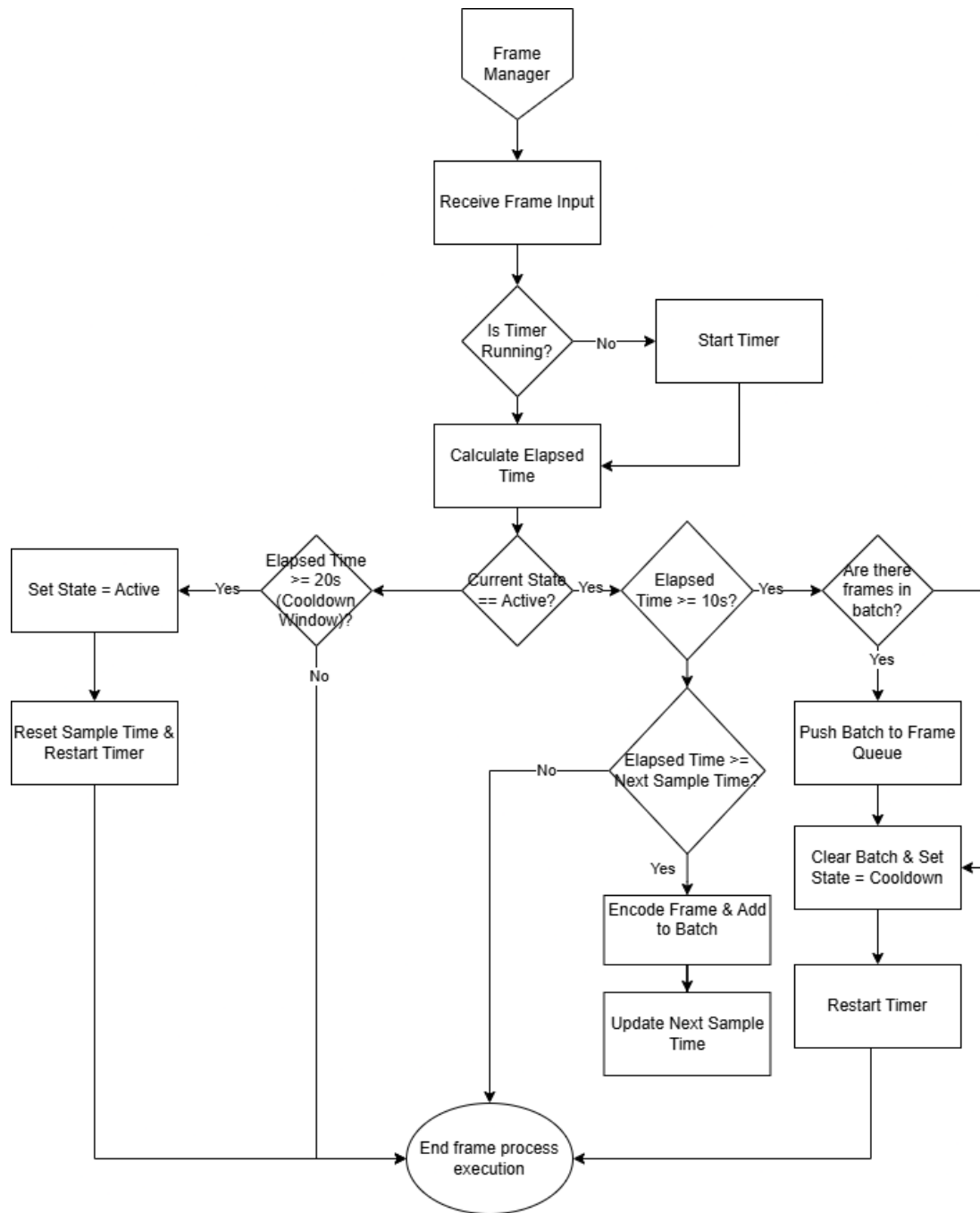


Figure 4.54 Flowchart of Frame Manager

4.2.2.5 Desktop application VLM Analyser

Figure 4.55 illustrates the execution logic of the VLM Analyzer module, which processes batched image frames via the Gemini API. The workflow initiates by wrapping the sequential frames and strict JSON schema requirements into an API payload. Upon receiving a successfully parsed response, the system evaluates the output to determine if subject is present with a threshold of three counts, and bad posture detection with threshold of three counts as well. Two of these thresholds triggers a push notification to mobile app, and a real-time notification respectively. Regardless of the evaluation path taken, all logical branches converge to synchronize the final structured analysis data with the Firestore database, ensuring real-time cloud consistency before processing the next queue item.

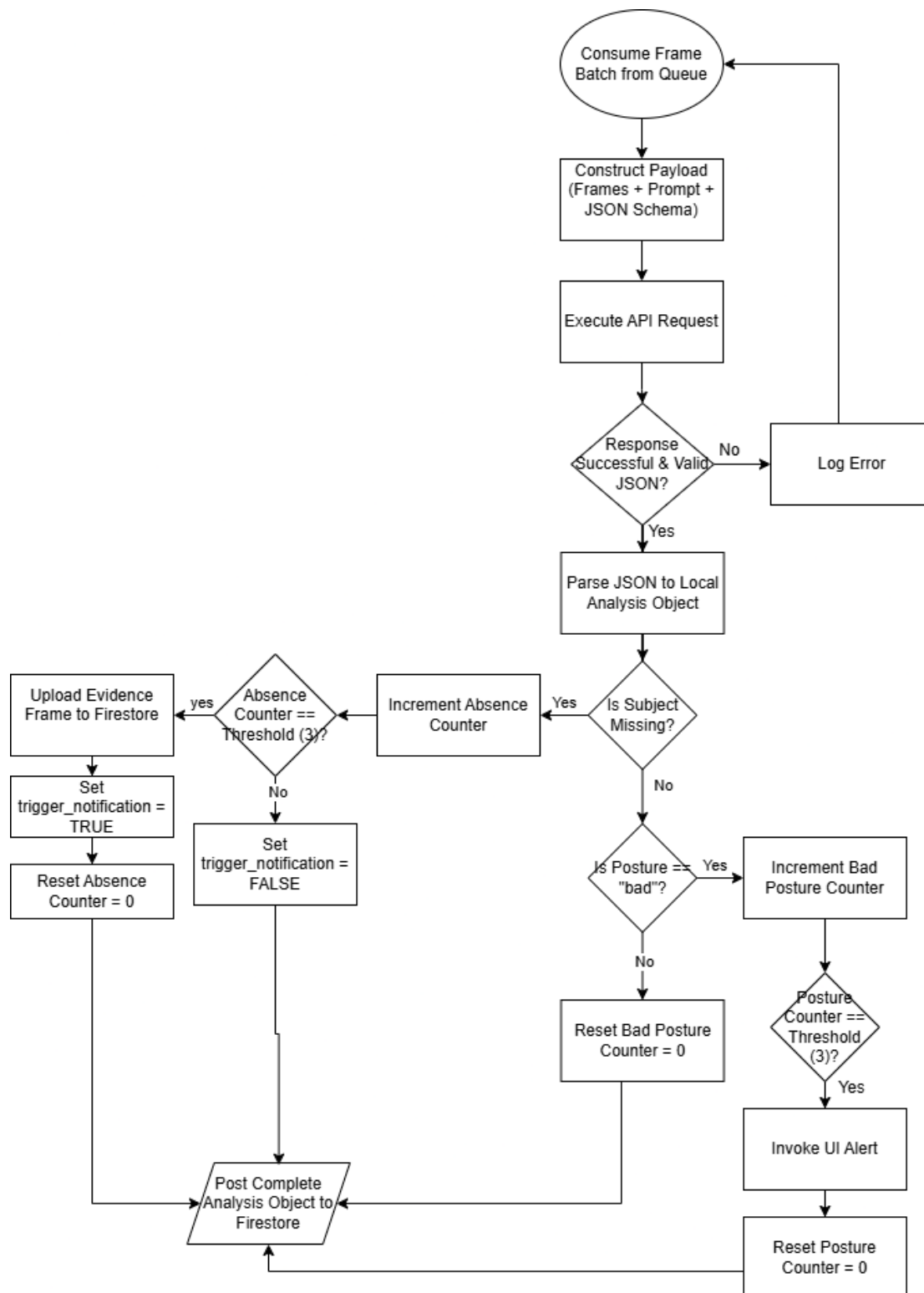


Figure 4.55 Flowchart of VLM Analyser

4.2.2.6 Desktop application Live Feed Manager

Figure 4.56 illustrates the event-driven architecture of the WebRTC Edge Node. The execution is divided into a primary signalling sequence and parallel event threads. Upon initialization, the system actively flushes existing network connections before generating a new WebRTC session equipped with STUN and TURN fallback configurations. It gathers ICE candidates, packs a local SDP offer, and uploads it to Firestore. The system then enters a non-blocking asynchronous polling loop, waiting for the mobile application to provide an SDP answer to secure the peer-to-peer tunnel. Concurrently, a parallel media thread evaluates incoming video frames; if the connection is active, frames are dynamically resized, encoded to VP8, and streamed. To ensure high availability and fault tolerance, an event-driven recovery thread actively monitors the network state. Should the tunnel collapse or fail, this thread automatically triggers a 3-second delay before recursively rebooting the entire Edge Node initialization sequence to re-establish the connection.

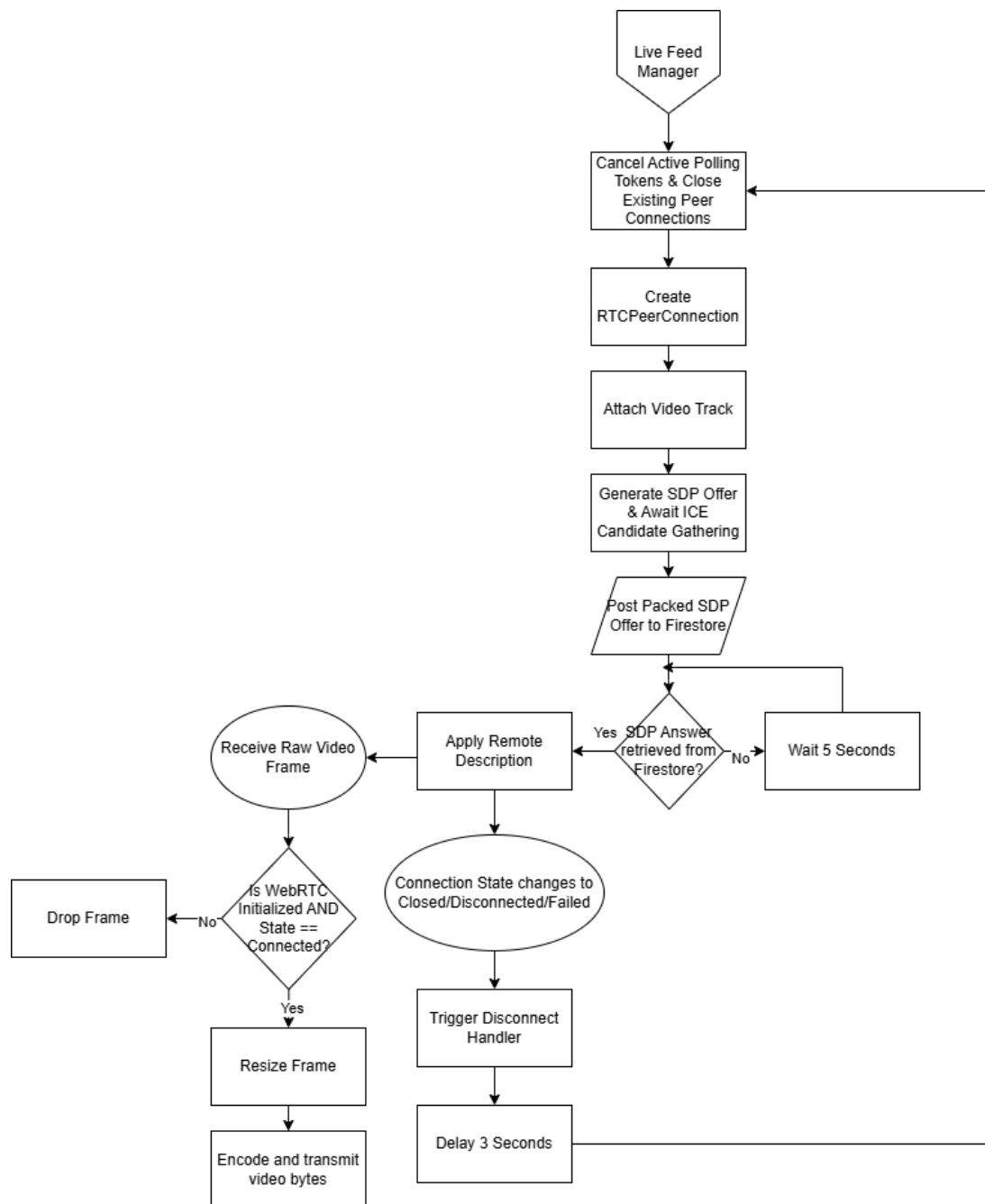


Figure 4.56 Flowchart of Live Feed manager

4.3 Implementation Issues and Challenges

Throughout the implementation of Monitra, several major issues were met and resolved. The first implementation issue was the complexity of data binding in WPF application. There are multiple libraries available each with its own method and trade off in terms of development efficiency, resulting in a non-standard way of implementing the MVVM architecture. Secondly, unlike the Flutter environment, the C# .NET ecosystem lacks a native, officially supported Firebase SDK. To integrate cloud services into the desktop client, a custom REST API interface had to be created from scratch, requiring manual traversal and deserialization of complex JSON payloads for every GET request. Additionally, implementation of Session Manager and Frame Manager in Monitra's desktop application was logically challenging due to the continuity nature of the application, continuously polling data and designing a simple algorithm to automatically on and off monitoring. Finally, the implementation of Live Feed Manager is the most challenging of all, due to the complexity of the lifecycle and event-driven architecture of WebRTC. Initial testing revealed that direct peer-to-peer signalling frequently failed due to restrictive NAT environments (STUN server failures). This was technically resolved by engineering a conditional fallback mechanism that dynamically re-routes media traffic through Metered TURN servers to guarantee stream delivery.

CHAPTER 5 SYSTEM EVALUATION AND DISCUSSION

This chapter describes Monitra's evaluation and the discussion of its performance on 3 test subjects, alongside the use case testing, project challenges and objectives evaluation.

5.1 System Testing and Performance Metrics

Monitra's testing involves the processing of subject in 4 different combinations on 2 inputs (webcam, and external camera), rather than a full studying session. The combinations of test cases refers to the studying activity and posture classification, totalling of 4 test cases which is:

1. Studying, Good Posture
2. Not Studying, Good Posture
3. Studying , Bad Posture
4. Not Studying, Bad Posture

The actual output will then be compared with the expected output to determine the VLM's zero-shot capability. To ensure a comprehensive evaluation, the system was tested using two distinct video input sources: a standard integrated webcam and an external RTSP-enabled camera. The execution logs from the console, and the corresponding dashboard UI updates were recorded and tabulated alongside supporting screenshot evidence. Finally, the individual results across all test subjects were combined to facilitate a comprehensive discussion and analysis. Additionally, to protect the privacy of the participant, all scenario snapshots included in this report have been intentionally blurred. These snapshots serve as direct visual evidence for each evaluated test case, documenting the subject's environment as captured by the designated input source.

5.1.1 Test subject 1

Due to the specific field of view and environmental constraints during the recording of Test Subject 1, the webcam snapshots do not capture supplementary desk objects (such as stationery, books, or mobile devices) that typically contextualize "studying" or "distracted" states. As these testing conditions cannot be exactly replicated, the visual evidence and subsequent VLM analysis for this subject rely only on the user's physical posture, gaze, and screen orientation to determine their activity state, independent of object interaction.

Webcam

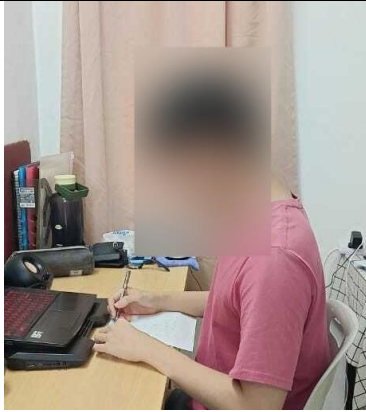
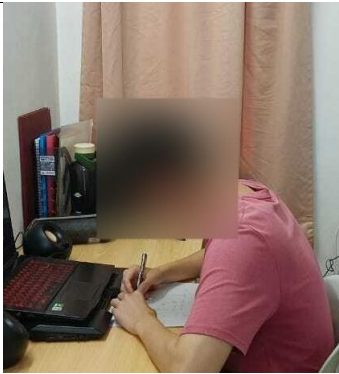
Studying, Good Posture	Scenario snapshot:
	
	Expected output:
	Studying with good posture
	Actual output:
	<pre> Gemini Result: {"is_studying": "not studying", "posture": "bad", "is_present": true, "justification": "The person is looking down, but no study materials are visible, and their expression doesn't suggest active engagement."} Status: not studying Posture: bad Reason: The person is looking down, but no study materials are visible, and their expression doesn't suggest active engagement. Is Present: True </pre>
Studying, Bad Posture	Scenario snapshot:
	
	Expected output:
	Studying with bad posture
	Actual output:
	<pre> === 10s Active Window closed. Batch pushed. Starting 20s cooldown. === Consumer: Processing new batch with Gemini API... Gemini Result: {"is_studying": "studying", "posture": "bad", "is_present": true} Status: studying Posture: bad Reason: Is Present: True </pre>
	Scenario snapshot:

Not Studying, Good Posture		
	Expected output:	
	Not studying with good posture	
	Actual output:	
	<pre>Status: not studying Posture: bad Reason: The person's head is bent down, obscuring any potential study materials or activities. Is Present: True</pre>	
Not Studying, Bad Posture	Scenario snapshot:	
		
	Expected output:	
	Not studying with bad posture	
	Actual output:	
	<pre>Status: not studying Posture: bad Reason: The person is looking down and smiling, not engaged in any study-related activity or materials. Is Present: True</pre>	

Table 5.1 Webcam results for test subject 1

External Camera

	Scenario snapshot:
--	---------------------------

Studying, Good Posture		
	Expected output:	
	Studying with Good Posture	
	Actual output:	
	<pre>Status: studying Posture: good Reason: Is Present: True</pre>	
Studying, Bad Posture	Scenario snapshot:	
		
	Expected output:	
	Studying with Bad Posture	
	Actual output:	
	<pre>Status: studying Posture: bad Reason: The person is actively writing on paper at a desk, indicating engagement in a study-related task. Is Present: True</pre>	
	Scenario snapshot:	

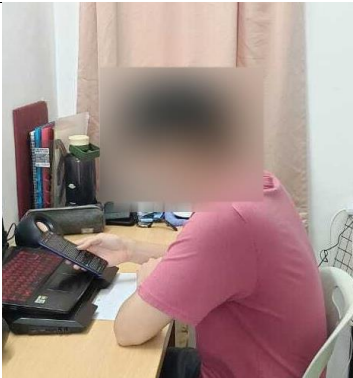
Not Studying, Good Posture		
	Expected output:	
	Not Studying with Good Posture	
	Actual output:	
	<pre>Status: not studying Posture: good Reason: The person is sitting at a desk but not interacting with any study materials; their hands are resting. Is Present: True</pre>	
Not Studying, Bad Posture	Scenario snapshot:	
		
	Expected output:	
	Not Studying with Bad Posture	
	Actual output:	
	<pre>Status: not studying Posture: bad Reason: The person is distracted by a phone, not engaging with the laptop or study materials. Is Present: True</pre>	

Table 5.2 External camera results for test subject 1

Dashboard UI

Dashboard results aggregates all the results processed in webcam and IP sessions.



Figure 5.1 Dashboard of test subject 1

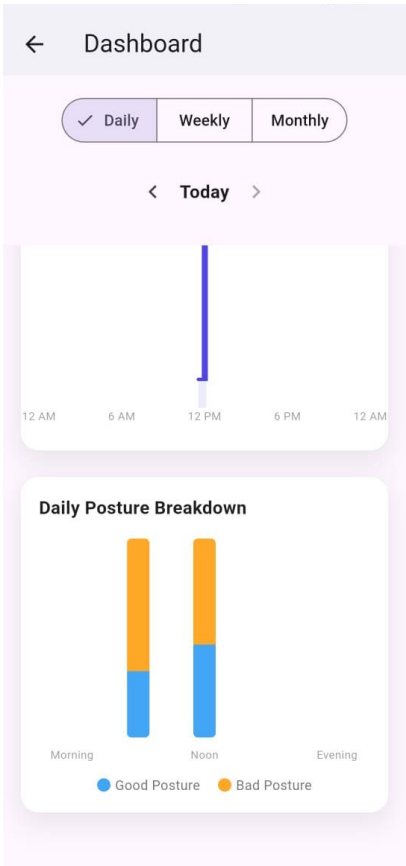


Figure 5.2 Dashboard of test subject 1

5.1.2 Test subject 2

Webcam

Studying, Good Posture	Scenario snapshot:
	
	Expected output:
	Studying with good posture
	Actual output:
	<pre>Gemini Result: {"is_studying": "not studying", "posture": "bad", "is_present": true, "justification": "The person is looking down, but no study materials are visible, and their expression doesn't suggest active engagement."} Status: not studying Posture: bad Reason: The person is looking down, but no study materials are visible, and their expression doesn't suggest active engagement. Is Present: True</pre>
Studying, Bad Posture	Scenario snapshot:
	
	Expected output:
	Studying with bad posture
	Actual output:
	<pre>Status: studying Posture: cannot determine Reason: Is Present: True</pre>
	Scenario snapshot:

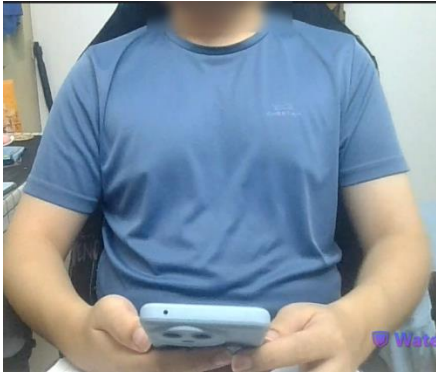
Not Studying, Good Posture	
	Expected output:
	Not studying with good posture
	Actual output: <pre>Status: not studying Posture: bad Reason: The person is holding and looking at a mobile device, indicating engagement with it rather than studying materials. Is Present: True</pre>
Not Studying, Bad Posture	Scenario snapshot:
	
	Expected output:
	Not studying with bad posture
	Actual output: <pre>Status: not studying Posture: bad Reason: The person is looking down and interacting with a smartphone, indicating they are distracted from studying. Is Present: True</pre>

Table 5.3 Webcam results for test subject 2

External Camera

Studying, Good Posture	Scenario snapshot:
	
	Expected output:
	Studying with Good Posture
	Actual output:
	<pre>Status: studying Posture: good Reason: Is Present: True</pre>
Studying, Bad Posture	Scenario snapshot:
	
	Expected output:
	Studying with Bad Posture
	Actual output:
	<pre>Status: studying Posture: bad Reason: Is Present: True</pre>
	Scenario snapshot:

Not Studying, Good Posture	
	Expected output:
	Not Studying with Good Posture
	Actual output:
	<pre> Status: not studying Posture: good Reason: The person is looking at and interacting with a mobile phone, not engaged with the laptop or study materials. Is Present: True </pre>
Not Studying, Bad Posture	Scenario snapshot:
	
	Expected output:
	Not Studying with Bad Posture
	Actual output:
	<pre> Status: not studying Posture: bad Reason: The person is looking at and interacting with a mobile phone, indicating they are not engaged in studying. Is Present: True </pre>

Table 5.4 External camera results for test subject 2

Dashboard UI



Figure 5.3 Dashboard UI of test subject

2

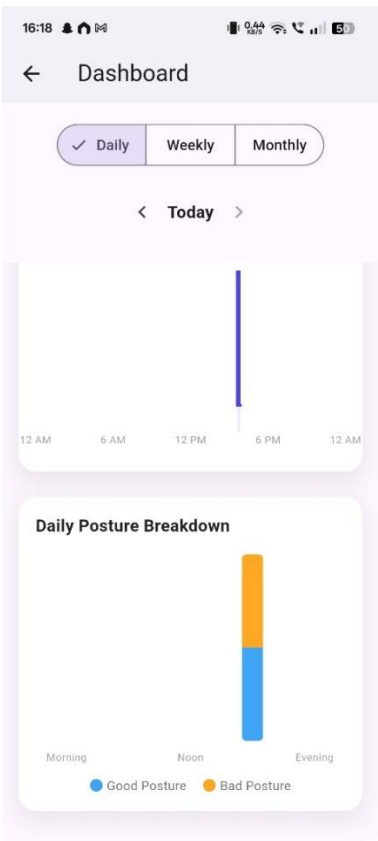
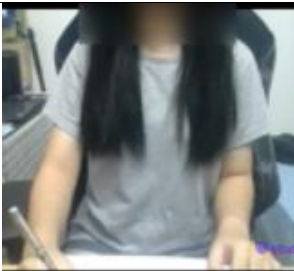


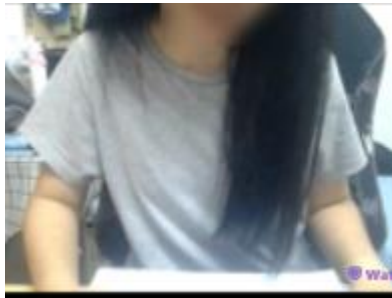
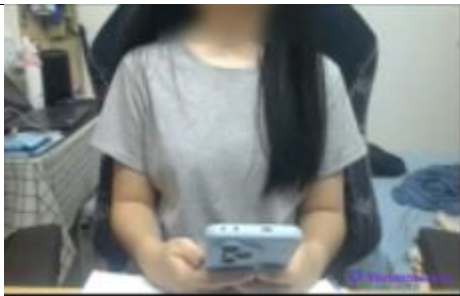
Figure 5.4 Dashboard UI of test subject

2

5.1.3 Test subject 3

Webcam

Studying, Good Posture	Scenario snapshot:	
		
	Expected output:	
	Studying with good posture	
	Actual output:	

	<pre>Status: studying Posture: cannot determine Reason: Is Present: True</pre>
Studying, Bad Posture	Scenario snapshot:
	
	Expected output:
	Studying with bad posture
	Actual output:
	<pre>Status: studying Posture: cannot determine Reason: Is Present: True</pre>
Not Studying, Good Posture	Scenario snapshot:
	
	Expected output:
	Not studying with good posture
	Actual output:
	<pre>Status: not studying Posture: good Reason: The person is actively using a mobile phone, indicating they are not engaged in studying activities. Is Present: True</pre>
	Scenario snapshot:

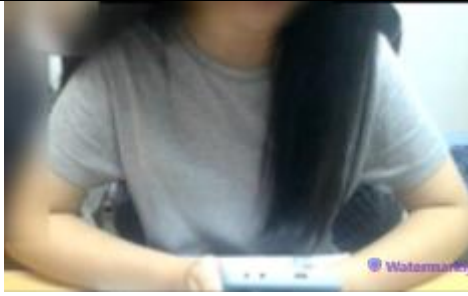
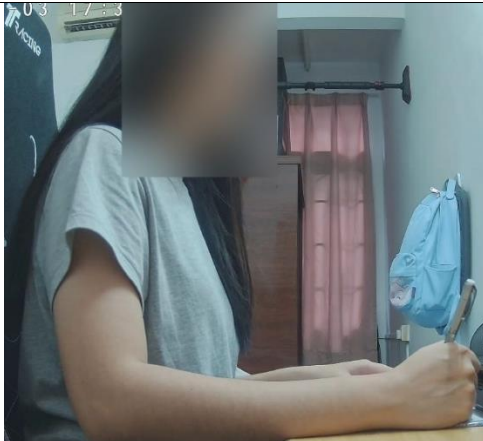
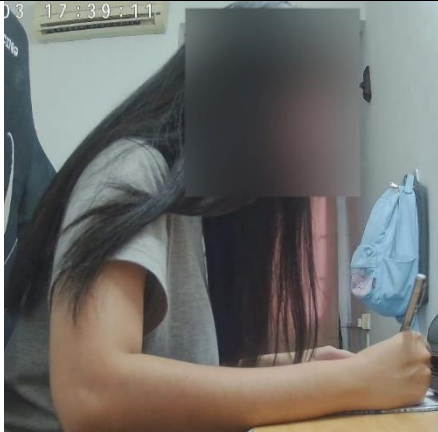
Not Studying, Bad Posture		
	Expected output:	
	Not studying with bad posture	
	Actual output:	<pre>Status: not studying Posture: bad Reason: The person is holding and looking at a phone, indicating engagement with the device rather than study materials. Is Present: True</pre>

Table 5.5 Webcam result for test subject 3

External Camera

Studying, Good Posture	Scenario snapshot:	
		
	Expected output:	
	Studying with Good Posture	
	Actual output:	
		<pre>Status: studying Posture: good Reason: Is Present: True</pre>
	Scenario snapshot:	

Studying, Bad Posture		
	Expected output:	
	Studying with Bad Posture	
	Actual output:	
	<pre>Status: studying Posture: bad Reason: Is Present: True</pre>	
Not Studying, Good Posture	Scenario snapshot:	
		
	Expected output:	
	Not Studying with Good Posture	
	Actual output:	
	<pre>Status: not studying Posture: bad Reason: The person is actively engaged with a mobile phone, not interacting with the study materials on the desk. Is Present: True</pre>	
	Scenario snapshot:	

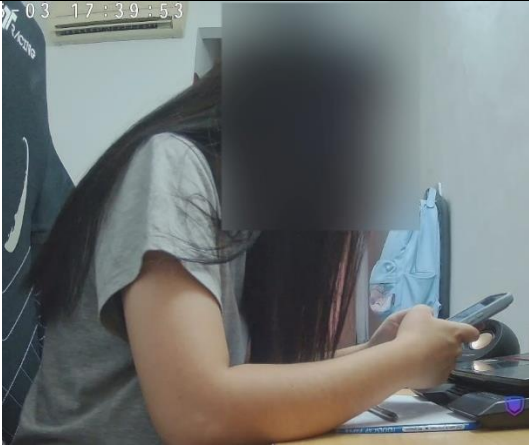
Not Studying, Bad Posture		
	Expected output:	
	Not Studying with Bad Posture	
	Actual output:	
	<pre>Status: not studying Posture: bad Reason: The person is focused on a mobile device, not engaging with study materials or a computer. Is Present: True</pre>	

Table 5.6 External camera result for subject 3

Dashboard UI

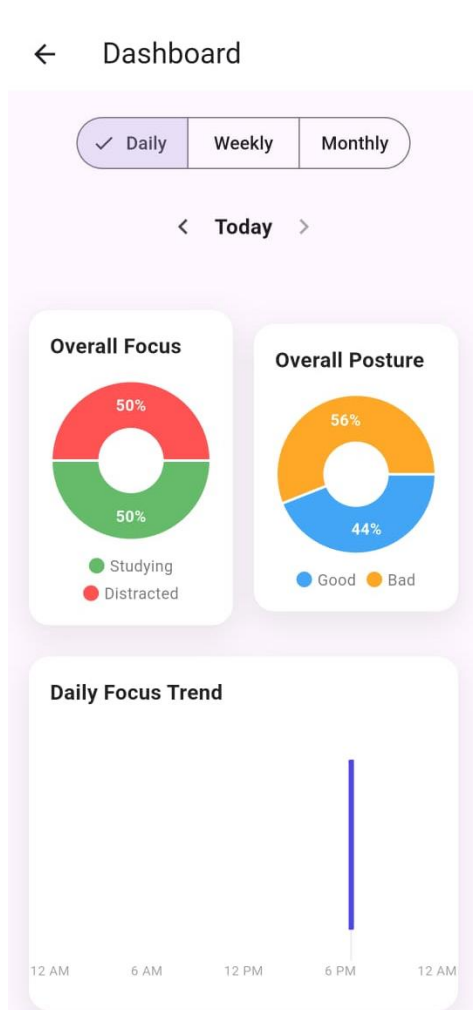


Figure 5.5 Dashboard UI for test subject 3

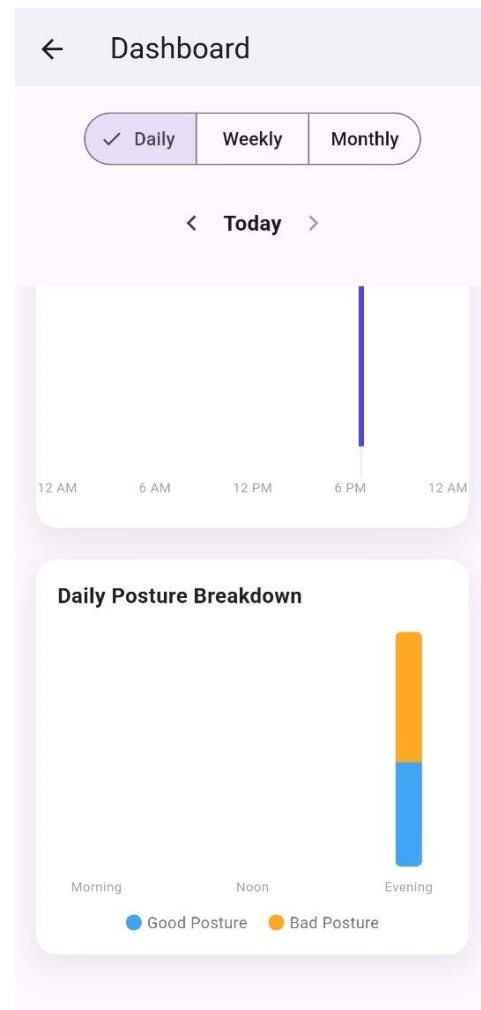


Figure 5.6 Dashboard UI for test subject 3

5.1.4 Results discussion

As tabulated in Table 5.7 the result of VLM monitoring on both types of input shows that external camera enables superior results in both study activity and posture monitoring. Posture monitoring has the highest error rate compared to study activity monitoring, and this mostly due to the limitations of VLM and its prompt engineering. Despite performing posture analysis by simply asking the VLM adhere to the RULA standards, it is difficult for VLM to extract such feature through webcam frames due to its front facing angle. Conversely, by utilising an external camera, this enables the VLM to have a clearer and broader view of subject's posture and activity, resulting in the significant improvement. Additionally, it is also worth noting that although the results of using external camera satisfactory, the VLM is still vulnerable to the frame's environment. As an example, the testing of Not Studying with Good Posture for Test

3 resulted in wrong posture which is likely due to the subject's long hair and shirt's fold, confusing the VLM's ability to perform proper analysis.

	Test 1	Test 2	Test 3
Webcam	2/4	1 / 4	2/4
External camera	4/4	4/4	3/4
Wrong	1 activity and 2 posture in webcam	1 activity and 3 posture in webcam	2 posture wrongs in webcam and 1 posture wrong in external camera

Table 5.7 Aggregated results of 3 test subjects

5.2 Feature Testing

While the previous section evaluated the core algorithmic accuracy of the VLM, this subsection isolates and verifies the system's **background managers, event-driven communications and user-facing features**. Specifically, it validates the frame sampler manager, session manager, real-time desktop posture alerts, cross-platform mobile push notifications, and the WebRTC live feed establishment.

5.2.1 Background Managers

Background Managers of Monitra will be evaluated here with proof of work based on the console output.

5.2.1.1 Session Manager

Objective	Test Method	Expected Outcome
Periodically fetches latest session data from cloud storage to automatically start or stop the monitoring session.	Create a session that matches current time and wait for the manager to automatically manage the monitoring process.	Monitoring process to be started when current time matches the session created via mobile application and stop when time does not match.
Visual Evidence		

```

Successfully parsed svSeC2KL2A4lAcX0HMfq sessions.
Background agent pulled fresh schedule from Firestore.
Schedule matched: Starting background camera.
FrameQueue has been reset for a new session.
Webcam monitoring started in the background.

=== BOOTING EDGE NODE ===
[Captured IP]: 1688061845 1 udp 2113937663 192.168.0.105 57544 typ host generation 0
Creating SDP offer...
Waiting for Cloud IPs...
Sampled frame at 0.0s
Sampled frame at 2.5s
Offer generated and packed with TURN/STUN Relay IPs!
Edge Node is listening... waiting for mobile app connection.

```

Table 5.8 Session Manager

5.2.1.2 Frame Manager

Objective	Test Method	Expected Outcome
Saves the selected preference and start sampling frame when session manager allows. Frame sampled in this manner has an interval of 2.5 seconds across 10 seconds. A period of 20 second cooldown is followed after before sampling process happens again.	Create a matching session and wait for sampling process to begin.	Sampling process will begin immediately after a session is found by session manager.
Visual Evidence		
<pre> === 10s Active Window closed. Batch pushed. Starting 20s cooldown. === Consumer: Processing new batch with Gemini API... Gemini Result: {"is_studying": "not studying", "posture": "bad", "is_present": true, "justification": Status: not studying Posture: bad Reason: The person is looking down and appears drowsy, not actively engaged with any visible study ma Is Present: True No sessions to run === 20s Cooldown finished. Starting new 10s Active Window. === Sampled frame at 0.1s Sampled frame at 2.6s Sampled frame at 5.1s Sampled frame at 7.6s === 10s Active Window closed. Batch pushed. Starting 20s cooldown. === Consumer: Processing new batch with Gemini API... Gemini Result: {"is_studying": "not studying", "posture": "cannot determine", "is_present": true, "ju } Status: not studying Posture: cannot determine Reason: The person is looking directly at the camera, and no study materials are visible in the frame Is Present: True </pre>		

Table 5.9 Frame manager

5.2.2 Event Drive Communication

5.2.2.1 Real-time posture notification

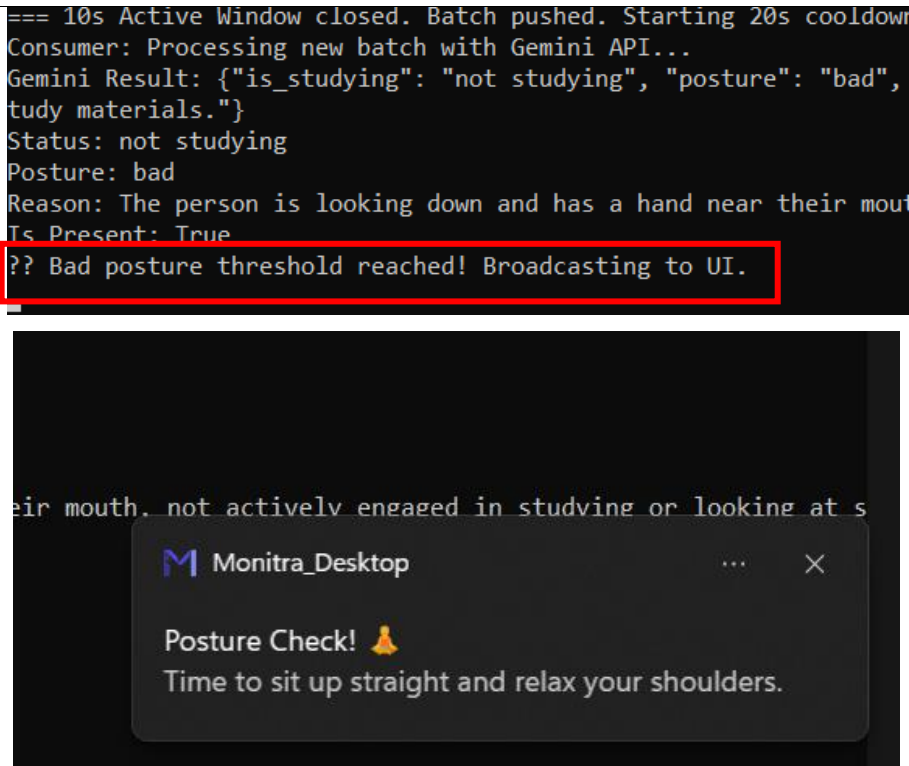
Objective	Test Method	Expected Outcome
To pop out a notification to remind the subject of their posture, on desktop application when bad posture met the threshold (3).	Purposely perform 3 bad postures to trigger notification.	Notification will pop out in the desktop app on the third classification when bad posture is detected 3 times in a row.
Visual Evidence		
 <pre> === 10s Active Window closed. Batch pushed. Starting 20s cooldown Consumer: Processing new batch with Gemini API... Gemini Result: {"is_studying": "not studying", "posture": "bad", study materials."} Status: not studying Posture: bad Reason: The person is looking down and has a hand near their mouth Is Present: True ?? Bad posture threshold reached! Broadcasting to UI. </pre> Monitra_Desktop Posture Check! 🧘 Time to sit up straight and relax your shoulders.		

Table 5.10 Real-time posture notification feature

5.2.2.1 Push Notification for subject detection

Objective	Test Method	Expected Outcome
Push a notification to Monitra mobile application to alert parents if the subject monitored is not detected in frames with a threshold of 3.	Purposely close the camera with an object, or any method, to ensure no subject is visible in the frame.	Push notification is expected to reach mobile application when threshold is met.

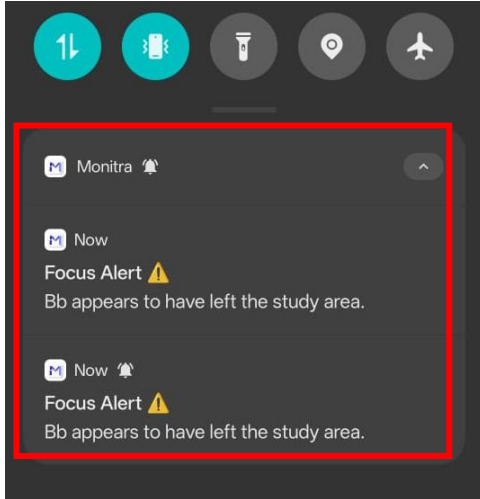
Visual Evidence
<pre> Status: not studying Posture: cannot determine Reason: No person is visible in the frame, making it impossible to assess studying activity or posture Is Present: False ?? Subject missing for threshold duration. Triggering alert! </pre>


Table 5.11 Push notification feature

5.2.3 Live Feed feature

Objective	Test Method	Expected Outcome
To provide parents a centralized platform for a quick view of their child in their study environment. Automatically retries connection upon unsuccessful action.	Perform multiple tries to re-enter the live feed page when a session is ongoing.	A live feed view to be displayed from the camera input in the mobile application.
Visual Evidence		
<pre> === BOOTING EDGE NODE === [WebRTC Status]: closed WebRTC connection closed successfully. [Captured IP]: 1688061845 1 udp 2113937663 192.168.0.105 56426 typ host generation 0 Creating SDP offer... Waiting for Cloud IPs... [Captured IP]: 2483126283 1 udp 1677730047 115.135.197.219 63777 typ srflx raddr 0.0.0.0 rport 0 generation 0 [Captured IP]: 644297015 1 udp 8447 206.189.151.205 23229 typ relay raddr 0.0.0.0 rport 0 generation 0 Offer generated and packed with TURN/STUN Relay IPs! Edge Node is listening... waiting for mobile app connection. </pre>		

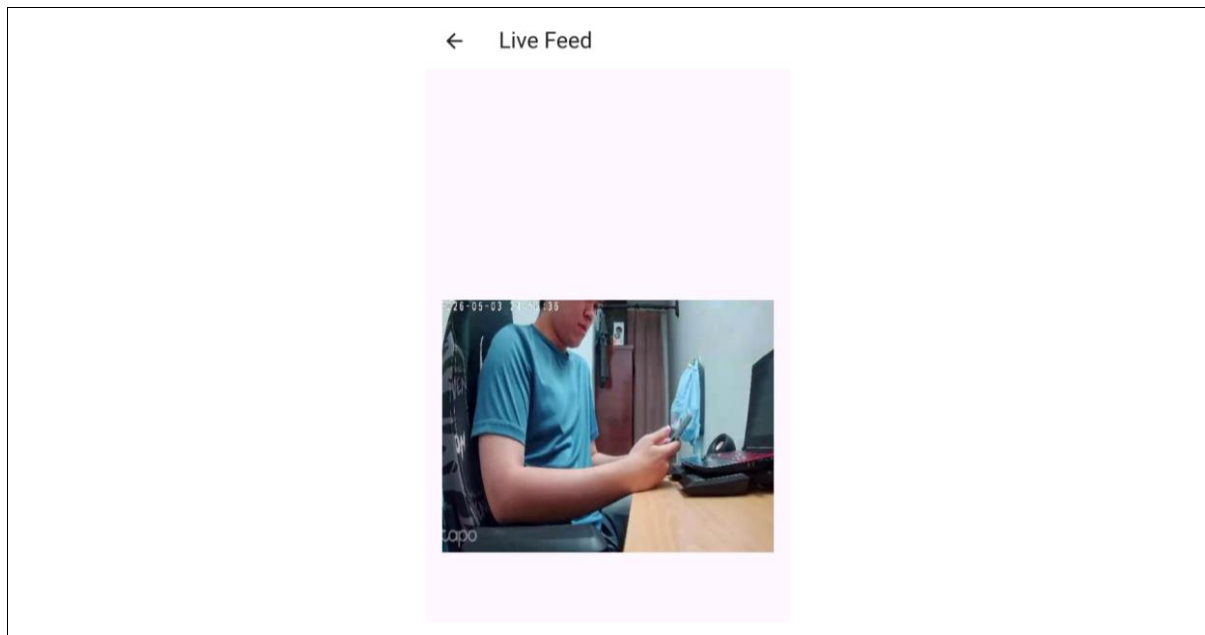


Table 5.12 Live Feed feature

5.3 Use Case Testing

Monitra mobile application:

Use case	Test Condition	Expected Result	Actual Result	Pass/Fail
Login	Enter email and password	Login correctly and redirects user to Homepage	Login successfully and redirects user to homepage.	Pass
Register	Enter email and password with confirmation	Account created by Firebase Auth and data structure is created in Firestore	Account was created correctly in both Auth and Firestore	Pass
Add child account	Create a child account with custom name	Child account with custom name is created under parent's account in Firestore	Child account created successfully under parent's account in Firestore	Pass
Change username	Enter new username	New username is displayed in settings	New username successfully displayed in settings	Pass
Change Password	Enter new password	Password must pass requirement and change upon next sign in	Password requirement of length ≥ 8 works	Pass

			and able to change password	
Change email	Enter new email	Email is changed upon next sign in and in Firestore record	Successfully changing email and Firestore records	Pass
Change child account name	Select child account and enter new name	Changes the child account name in Firestore	Successfully changing the child account name in Firestore	Pass
Log Out	Press log out button in settings page	Pop out a log out confirmation before logging out completely	Correctly shows a log out popup confirmation and successfully logs out the user upon confirmation	Pass
View study session	Navigates to child account's session to have the options to delete or create new study session	Able to delete study session and its nested records in Firestore and able to create new study session, recording the data in Firestore	Successfully remove study session data from Firestore and UI, and creating sessions successfully saves the data in Firestore	Pass
Toggle study session	Pressing the radio toggle button in each session	Able to store the toggled value in Firestore and update the UI	Correctly stores the toggled value in Firestore and updates the UI	Pass
Delete child account	Press and hold child account card	Able to delete the nested data in child account and the account itself	Correctly delete the nested data in child account and the account itself	Pass
View Dashboard	Navigate to dashboard page in account option	Able to view dashboards daily, weekly, and monthly	Successfully display dashboard data across daily, weekly, and monthly	Pass
View alert history	Navigate to alert history page in account option	Able to view all past alerts and its details	Successfully display all alerts history and its	Pass

			details when pressed	
View live stream	Navigate to live feed page in account option	Able to view live feed when sessions are active	Successfully view live feed when a session is active	Pass

Table 5.13 Use case testing table for mobile application

Monitra desktop application:

Use case	Test Condition	Expected Result	Actual Result	Pass/Fail
Log in	Opening desktop app	Log in the user when credentials are correct	Successfully log in the user when credentials are correct	Pass
Set camera preference	Navigating to camera page from homepage	Able to select webcam or RTSP camera	Correctly saves the selected preference	Pass

Table 5.14 Use case testing table for mobile application

5.4 Project Challenges

Beyond the technical challenges addressed during implementation chapter, Monitra project also faces broader challenges that impacted the project timeline and development scope. A major project challenge was the shift of scope during the initial development of Monitra. Initially, the core processing was to utilised open-source VLM, running directly on desktop client, however this introduces severe performance bottlenecks, shifting the entire project scope to a proof-of-concept that leverages proprietary API. Furthermore, another major challenge is the domain knowledge required to implement the Live Feed Manager. Grasping the theoretical concepts of UDP Hole Punching, ICE operations, and STUN/TURN were barely sufficient to bridge the gap from theory to technical implementations.

Additionally, resource availability severely impacted development efficiency, particularly the backend integration. Unlike mobile frameworks such as Flutter, which benefit from native Firebase SDKs, the C# .NET ecosystem requires manual engineering of a REST API. The lack of centralized documentation, standardized tutorials, and community resources for manual Firebase REST integration in C# forced a heavy reliance on independent trial-and-error.

5.5 Objective Evaluations

Overall, Monitra has successfully demonstrated its proof-of-concept which is the ability to leverage VLM as its core processing for studying activity and posture monitoring. This allows it to achieve the first objective by developing a dual-application architecture that delivers actionable data to parents. Additionally, Monitra also has the feature to remind the monitored subject regarding their posture in real time, aligning with the second objective of this study which is to promote posture health at young age via real time notification. Finally, the third objective of this study was achieved by performing testing across three subjects. The evaluation proved the system's capacity to accurately classify study and posture states, while also highlighting critical environmental constraints; notably, that integrating an external RTSP camera yields significantly superior analytical accuracy compared to standard, sub-optimal webcam angles.

5.6 Concluding Remark

In summary, this chapter detailed the system evaluation and its performance testing. Across three subjects, it is found that studying activity classification performs similarly in both camera inputs but having physical objects undoubtedly helps the VLM to give better results. Additionally, posture classification on external camera input gives the highest result due to its broader and clearer view of the subject's body from the side but may still be vulnerable to objects that may obscure the subject's posture such as hair or shirt's folds.

Besides the VLM evaluation, Monitra's features such as the background managers, event-driven features and live feed features were proved to be stable, covering all major cases such as automatic reconnection for live feed viewing, automatic monitoring process, and resource efficient frame sampler.

CHAPTER 6 CONCLUSION AND RECOMMENDATION

6.1 Conclusion

In conclusion, Monitra was developed to solve the challenge of limited parental supervision, offering a non-intrusive solution to monitor study habits and mitigate posture-related health issues among children. The proposed solution, Monitra, successfully bridges this gap through a dual-application ecosystem. By leveraging a Vision-Language Model for zero-shot behavioural classification alongside features such as push notification for subject detection, dashboard visualisation aggregating data across daily, weekly, and monthly, and a WebRTC for live media routing, the system provides parents with actionable, real-time insights.

6.2 Recommendation

While Monitra met its objectives as a prototype, leveraging a proprietary VLM for its superior reasoning capabilities, it is not practical to be deployed in a real scenario. This is because the API limit would easily be reached and requires clients to handle their own API key. Similarly, switching to opensource VLM would introduce performance bottlenecks or costs if it is deployed on cloud infrastructure. In order to leverage VLM's intelligence, Monitra can leverage other machine learning algorithms or lightweight deep learning models for majority of the monitoring sessions. The role of VLM in Monitra would then be shifted to an “emergency model”, triggering the usage of proprietary VLM or self-hosted VLM based on conditions such as confidence threshold. The reason behind this is because VLM's superior reasoning isn't necessary most of the time when simple Artificial Intelligence models could classify the action or posture with high confidence. If models were not able to give high confidence results due to external factors such as obscured parts in image or any uncertainties, VLM can then be triggered to give its result, greatly reducing the cost of its usage.

Additionally, in the domain of posture monitoring, lightweight classification models can be utilized to provide a robust performance, disregarding camera input types. By offloading the posture classification task from the proposed Monitra's concept, token usage can be reduced while providing better accuracy.

REFERENCES

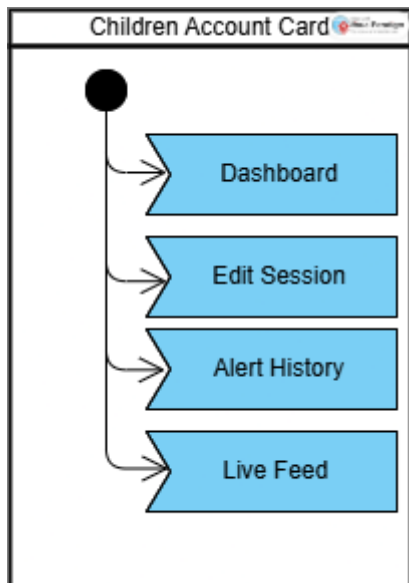
- [1] "View of A Study On The Effects Of Students' Posture, Comfort And Health In Consequences Of Prolonged Sitting Among Senior Cambridge Students," *Journalppw.com*, 2024. <https://journalppw.com/index.php/jpsp/article/view/16827/10660> (accessed Nov. 01, 2024).
- [2] K. Harold, "Poor Posture in Teens: It's Not Just a Bad Habit - Cincinnati Children's Blog," *Cincinnati Children's Blog*, Jan. 07, 2025. <https://blog.cincinnatichildrens.org/healthy-living/child-development-and-behavior/poor-posture-in-teens-not-just-a-bad-habit/>
- [3] C. X. Lam, "Children On-Screen activity monitoring application - UTAR Institutional Repository," *Utar.edu.my*, Sep. 2022, doi: http://eprints.utar.edu.my/4696/1/fyp_CS_2022_LCX.pdf.
- [4] M. Romero, "Features," *Qustodio*. <https://www.qustodio.com/en/features/>
- [5] "CuboAi Smart Health Bundle," *Cubo AI*, 2022. <https://us.getcubo.com/products/cuboai-smart-health-bundle> (accessed May 01, 2025).
- [6] "Security," *Cubo AI*, 2022. <https://us.getcubo.com/pages/security> (accessed May 01, 2025).
- [7] "UPRIGHT Posture Training Device - Everyday Posture Coaching," *UPRIGHT Posture Training Device*. <https://www.uprightpose.com/>
- [8] "UPRIGHT GO 2™," *Upright Technologies Ltd*, 2023. https://store.uprightpose.com/products/upright-go2?srsId=AfmBOor9upjPNhbItYtmBYxGKfEdQWlH8F9mFfUYlHJbgrjGpn8O_Mk (accessed May 02, 2025).
- [9] "Reddit - The heart of the internet," *Reddit.com*, 2020. https://www.reddit.com/r/PostureTipsGuide/comments/epfhz5/does_anyone_have_any_experience_with_the_upright/ (accessed May 02, 2025).
- [10] "PostureMinder Extension," *PostureMinder*, 2025. <https://postureminder.app/> (accessed May 02, 2025).
- [11] Z. P. Team, "Zen Posture | AI-Powered Posture Correction," *Zen Posture*, 2025. <https://www.zenposture.app/> (accessed May 02, 2025).
- [12] ZenPosture, "ZenPosture," *Devpost*, Feb. 09, 2025. <https://devpost.com/software/zenposture> (accessed May 02, 2025).
- [13] P. N. Cañas, M. Nieto, O. Otaegui, and I. Rodríguez, "Exploration of VLMs

REFERENCES

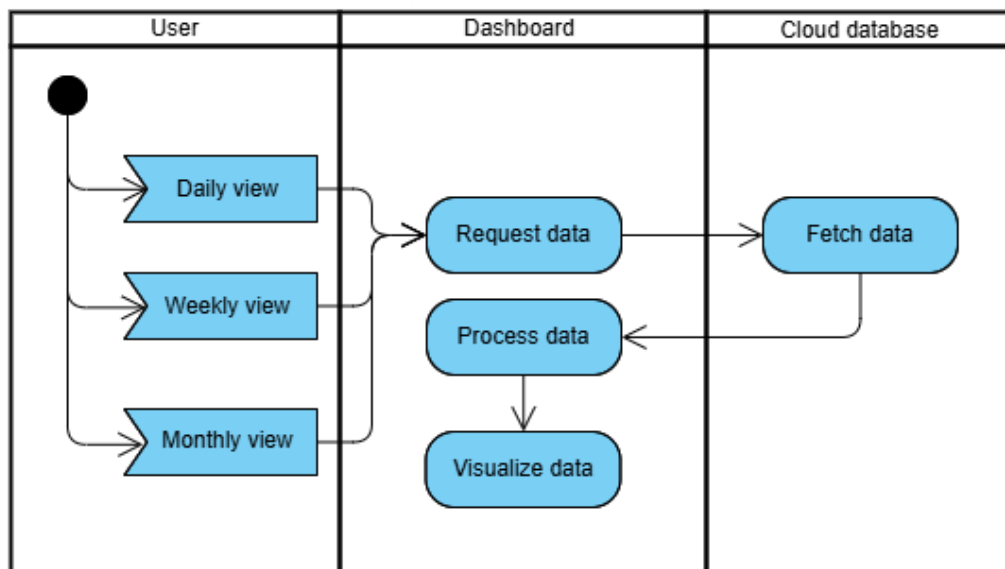
- for driver Monitoring Systems applications,” *arXiv.org*, Mar. 15, 2025.
<https://arxiv.org/abs/2503.12281v1>
- [14] MESCIUS inc, “WPF Development Best Practices for 2024,” *Medium*, Feb. 05, 2024.
<https://medium.com/mesciusinc/wpf-development-best-practices-for-2024-9e5062c71350> (accessed May 02, 2026).
- [15] M. Team, “10 WPF Best Practices [2024],” *Postsharp.net*, Nov. 19, 2024.
<https://blog.postsharp.net/wpf-best-practices-2024> (accessed May 02, 2026).
- [16] “Guide to app architecture,” *Flutter.dev*, 2026. <https://docs.flutter.dev/app-architecture/guide#mvvm> (accessed May 02, 2026).
- [17] R. Anil *et al.*, “Gemini: A Family of Highly Capable Multimodal Models,” *arXiv.org*, Dec. 18, 2023. <https://arxiv.org/abs/2312.11805>

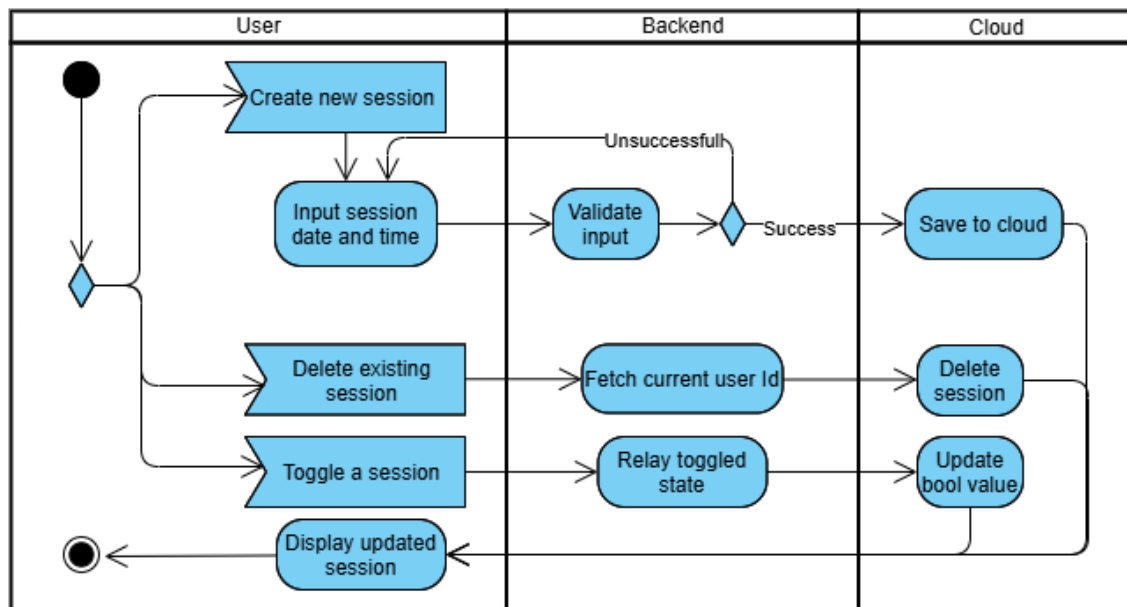
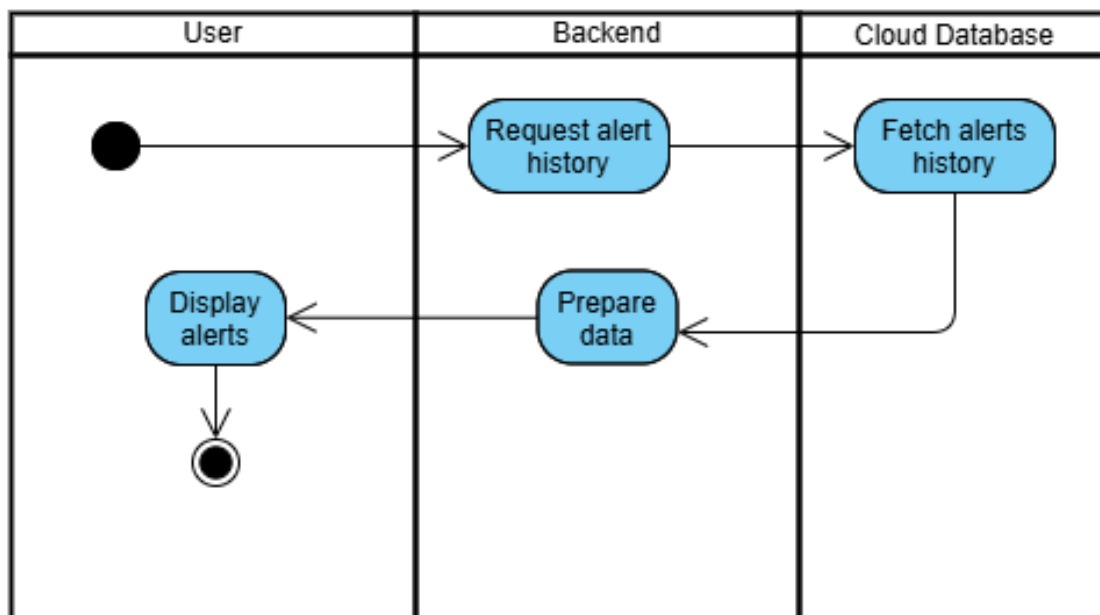
APPENDIX A

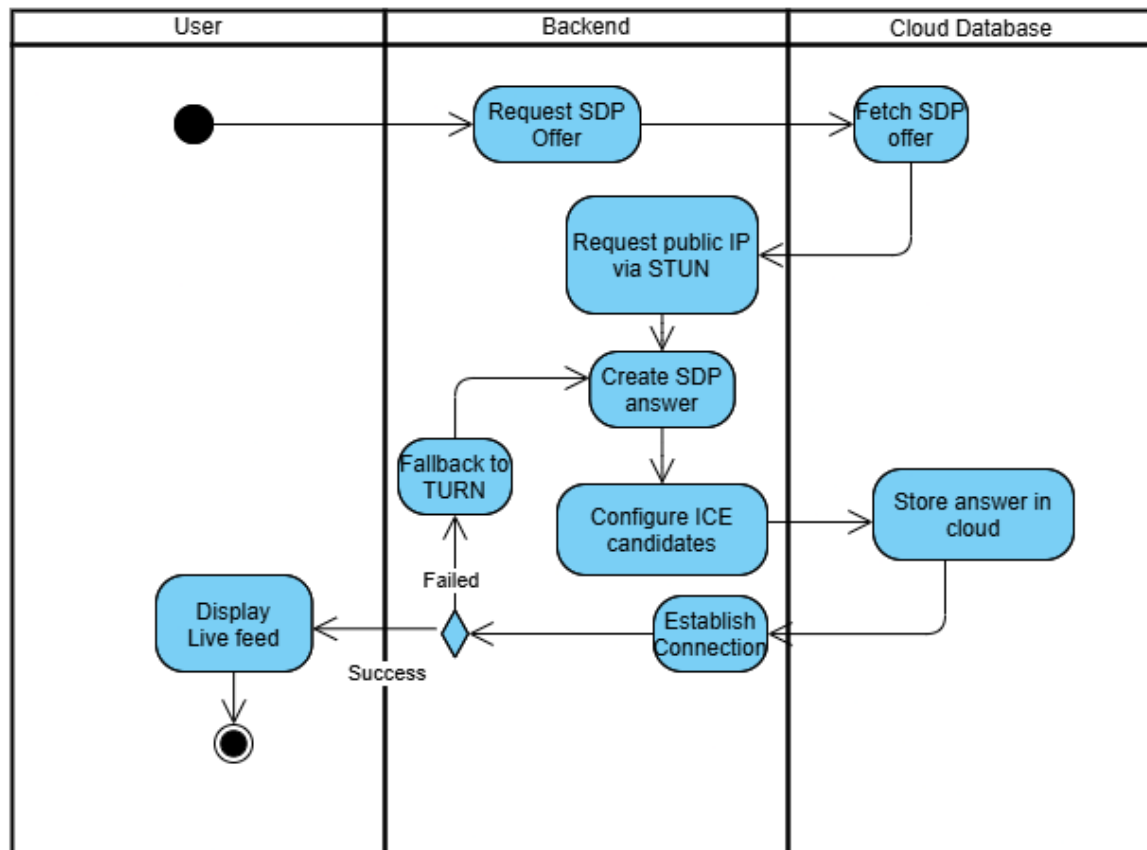
Children Account Card Activity Diagram



Dashboard Activity Diagram



Edit Session Activity Diagram**Alert History Activity Diagram**

Live Feed Activity Diagram

POSTER



Student: Dennis Chong Kwang Khai 21ACB04446
Supervisor: Cik Ashikin binti Ali

**MONITRA:**

A prototype to monitor
study habits and posturing
in children

**INTRODUCTION**

Monitra is a proof-of-concept application that aims to leverage a proprietary Vision Language Model (VLM) API as an assistant that monitors children's study habits and their posture, enabling busy parents to observe and understand their child's well-being in a non-intrusive way.

OBJECTIVES

1. To develop an AI-based assistant on monitoring children's study habits and posture health.
2. To promote posture health at young age via real-time notification.
3. To validate the effectiveness of Monitra with at least 3 subjects.

METHODS/FEATURES

- Consists of 2 application: Desktop application for children, and Mobile application for parent
- Desktop application process frames using VLM from webcam/external camera (RTSP)
- Mobile application for session creation and data dashboard
- Real-time notification are triggered on desktop app with a threshold of 3
- Push notification for mobile app when children not in environment
- Live feed viewing via WebRTC for active session