STUDDO: A WEB APPLICATION TO HELP STUDENTS ENHANCE PRODUCTIVITY AND MINIMIZE DIGITAL DISTRACTION

A Capstone Project Presented to the Faculty of the

School of Computer Studies

University of San Jose – Recoletos

Basak, Pardo, Cebu City

In Partial Fulfillment

Of the Requirements for the Degree of

Bachelor of Science in Information Technology

By

TEAM ARCHON:

Jungco, Angelika Joy

Lapiz, Jaime Jr. A.

Rodriguez, Nyllejohnxl L.

May 2023

ABSTRACT

With the growth of technology and online learning, students are increasingly vulnerable to distractions such as social media, streaming platforms, online gaming, etc. These distractions can hinder their ability to focus on class material and ultimately affect their academic performance. To overcome these distractions, students can utilize various productivity applications to manage their time and stay on task during studying or working on school works. Thus, the researchers have developed a productivity application, Studdo, to help students enhance productivity and improve their study habits. The application incorporates the concept of Pomodoro Technique, virtual study room, and study ambience to simulate a better learning environment. Studdo is a web application which mainly used Laravel framework and MySQL database. Upon implementing the project, several test cases were conducted and most of the expected results were met. The researchers concluded that this application is preferably be used by educational institutions especially the teachers and students. It has also been found that there are external factors affecting the efficiency of the application such as browser limitations, internet connection speed, and device used. Therefore, it is recommended to use a device with good internet connection for better experience. The researchers also recommend to explore more frameworks for better or improved way of implementation of the features, and to implement the application as a desktop software for further configuration that perfectly addresses the project objectives.

Keywords: Study Session/Room, Pomodoro Technique, Timer, Study Goals, Productivity

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

RATIONALE OF THE STUDY

Not long ago at the peak of the pandemic, education was conducted virtually where there is a demand for students to use gadgets – smartphones, tablets, laptops, or desktops – for their classes. Even now with face-to-face classes being brought back in some institutions, gadgets are still being used most of the time. This is due to almost all the school deliverables required to be submitted online as the technology is massively developing and as years go by, people opt to be as paperless as possible.

Data has shown that twice as many students were highly distracted when doing schoolwork remotely compared with when they were using their computers in the classroom (Baker, 2020). Meanwhile, in the Philippines, Dalisay and Montemayor (2021) said that Filipino students continue to struggle to maintain their attention to the material presented. It has been suggested that the impact of a shorter attention span and lower level of concentration is felt more strongly by online learners. Distracted students fail in accomplishing their tasks within a short period of time as they struggle to concentrate or tend to lose focus.

Considering this situation, study applications will serve as an efficient and effective tool for enhancing student's productivity and minimizing digital distraction. The way some students approach and perceive studying might change with the help of this application and consequently help them develop a good and well-maintained study habit if used consistently and properly.

Several study applications exist on the internet and offer similar services or features. One example would be StudyStream where students can study together with a group of people in a Zoom meeting. They wouldn't interact with each other verbally, but just with their presence. Another one is Forest where users can study by themselves and make use of focus and break timers.

Studdo, the proposed productivity application, incorporates supervised studying, and the concept of focus and break timers such as the Pomodoro Technique to increase concentration. Supervised studying is the main factor that the developers are trying to assess whether it is effective in determining the subconscious behavior of the students while studying – whether they are being distracted by other things or feeling sleepy perhaps – and provide the students with immediate intervention and feedback of their study habits for them to improve in the next study sessions to come.

REVIEW OF RELATED LITERATURES

Several studies have shown the reality of online learning modality and how it affects the learners. It was discovered that offered learning resources, such as 1-hour lengthy video lectures or timed tasks with long reading materials connected, had a negative impact on some students' attention spans (Cherrett et al, 2009; Dror et al., 2011). In 2016, McCoy emphasized that the students' use of digital devices has been causing distractions, especially when used for non-class related purposes.

Being productive helps in resisting the urge to pick up a gadget for entertainment use, hence minimizing digital distraction. Monks and Cutler (2018) stated that having a

productivity app can help reduce students' social media addiction by monitoring their phone usage, thus improving their studying environment and time management. Additionally, to succeed in self-monitoring, students must learn to keep track of their tasks and the way they think so they can adjust their behaviors and thoughts in order to meet goals or complete tasks (Porter, 2002; Smith, 2002).

One of the most common and effective ways of boosting productivity is applying the Pomodoro Technique which is known as a time management strategy. "The Pomodoro Technique was created with the aim of using time as a valuable ally to accomplish what we want to do the way we want to do it, and to empower us to continually improve our work or study processes" (Cirillo, 2006, p. 1). The concept behind this technique is to break down a task into sessions with specific time allocation, hence making completion of tasks more bearable and less taxing as well as increasing one's level of concentration.

Listening to music also has a correlation with concentration and performance. Music may also help people release anxiety or we can call it an anxiety-reducing measure (Lesiuk, 2005). Singer (2008) and Barker (2008) reported that music increased the chance students remembered what they had learned, by assisting the recall of information. To be more specific, classical music is by far the most effective of all musical genres. Students who listened to a one-hour lecture with classical music playing in the background scored much higher on a quiz. It was hypothesized that the music stimulated students' emotions, making them more receptive to knowledge (Dosseville et.al, 2012).

A study by Lina Jin (2022) focuses on "Study With Me" videos that are posted in online video-sharing platforms. Included in this study, the top 2 reasons why Jin's

respondents watch these “Study With Me” videos are (1) Get Companionship and (2) Get Motivated. According to Jin, virtual study rooms that are initiated by these videos provides students an opportunity to establish customized virtual self-study space, build connections with others, and participate in an online study-support community that is distinct from traditional offline study space.

The objective of this literature review is to help us understand the various factors that affect the attention span and motivation of students, particularly in online learning. The concepts – Pomodoro Technique, Music, and Virtual Study Room – are incorporated as the core features of the developed productivity application.

RELATED WORKS

Studystream.live

With the advancement of mobile devices and social media, students feel more connected with other students that they are likely to study in groups in order to discuss with each other. However, it would pose a dilemma for students, leaving some of the students doing it alone whenever a group study is ineffective. Effective study groups can help students learn the course material better in a deeper, more concrete way. It can generate positive energy and encourage the group study members to motivate and inspire one another.

The sense of companionship provided by study with me videos is even more palpable during the COVID-19 pandemic. At the height of the pandemic last year, more than 1.3 billion students had to stay at home due to school closures. Even as students

start streaming back to campuses, in-person classes have not gone back to full capacity and social distancing measures remain in place. Which is why studystream.live became popular. This web application offers 24/7 “focus rooms” on zoom where students are meant to turn on their camera, and study with up to 1,000 other students. The team behind studystream.live also host onboarding streams each weekday wherein they breakdown and introduce how the focus rooms work and introduce people to etiquette for the students to not have a culture shock as to why no one is talking.

Forest

On the other hand, some existing applications in the market implements the idea of Pomodoro Technique mentioned above in the related studies, such as the Forest app. This productivity application lets users utilize the timers to focus on their tasks, and being productive is equivalent to the growth of the tree. The concept of Forest is about planting and growing a tree while the user is being productive or focusing. On the contrary, leaving the application without finishing the timer also means leaving the planted tree to wither. With the goal of having a fully grown tree, the user will also be motivated to be productive and not abandon the tree that has been planted in the application.

PROJECT OBJECTIVES

The main objective of this research is to develop a web application made for students to increase productivity and to address the issue of digital distraction concerning education. The researchers’ goal for the application are the following:

1. To implement the Pomodoro Technique or similar concepts of work-break timers for the students to increase their concentration.
2. To incorporate music and enhance students' emotions, making them more responsive to information and making learning easier.
3. To support learners in increasing their attention span and motivation, particularly in the online learning environment.
4. To let users be in a study room setting along with other users and simulate a group study or library session for a better learning environment.
5. To measure productivity by implementing study statistics such as the total time they have studied and the study goals they have completed per session.

PROJECT SCOPE AND LIMITATIONS

This study develops a web application that uses Laravel as its main framework and MySQL for data management. As it is a web-based application, users are required to access the application using web browsers that are preferably up to date. Some of the suggested browsers are the most common and popular ones such as Google Chrome, Microsoft Edge, Mozilla Firefox, Opera GX, etc. Users are also recommended to have a web camera to fully utilize the study room video call features. The application's target users are students and teachers, especially those that currently have online setting or utilize online resources for their classes.

Although the purpose of this application is to enhance productivity, external factors are out of the project's scope such as the surroundings of the user. Since it is only implemented as a website, blocking the user from accessing other external websites

cannot be implemented due to browser limitations. The user can still visit other external sites which could possibly be considered as a distraction if it is not relevant to the user's tasks. This factor might not fully assure that the user is staying productive the whole time without distractions. The application will only serve as a tool and an avenue for students to improve their study habits and manage their tasks with the help of the work-break timer concept. However, users enabling their camera throughout the study session will help in manually monitoring their study behaviors by the room hosts.

Study rooms may only allow up to 15 participants to make the group session less crowded. Music options are provided based on the genre being set for the study room only. Users cannot choose other music genres once the study session has started. Additionally, there is no option for the user to add their own music. This is also applied to the background images set for the study rooms.

SIGNIFICANCE OF THE STUDY

Productivity applications aid users to focus their attention on certain tasks they are currently working on. Having in mind the application's goal of increasing productivity, this project would be beneficial to people who have a very short attention span. This application serves as a platform for users to be productive in a sense that the features designed are considering the factors influencing productivity such as the focus and break timers, and study ambience.

Study room includes background music and images, focus and break timers, and the presence of other people under the same study group. Focus and break timers are

inspired by the Pomodoro Technique which gives the users time to take breaks in between the study session which is proven to be effective in studying efficiently. With the help of the breaks, the study session doesn't make the student mentally exhausted.

CONCEPTUAL FRAMEWORK

This study primarily relies on the users' preferences. The users' preferences expected is composed of the music genre, and time management method.

Music sets the environment of the study. It puts the listener in a better mood which in turn improves their studies. This is attributed based on the research, Mozart Effect (1993) found that listening to complex classical music like Mozart improved test scores, and the Blur Effect (1990) where students who listened to BritPop band Blur seemed to do better on tests. Which was found out that it was bigger than the Mozart effect, because the students enjoyed pop music like Blur than classical music.

The researchers use the time management method inspired by the "Pomodoro" technique. Pomodoro technique encourages the students to work with the limited time that they have set. It was developed in the late 1980s by Francesco Cirillo. Its goal is to reduce the internal and external interruptions of focus and flow of study.

The application incorporates the users' preferences like the music genre they prefer, and the time management method they used. These preferences are then used to group the users according to their preferences.

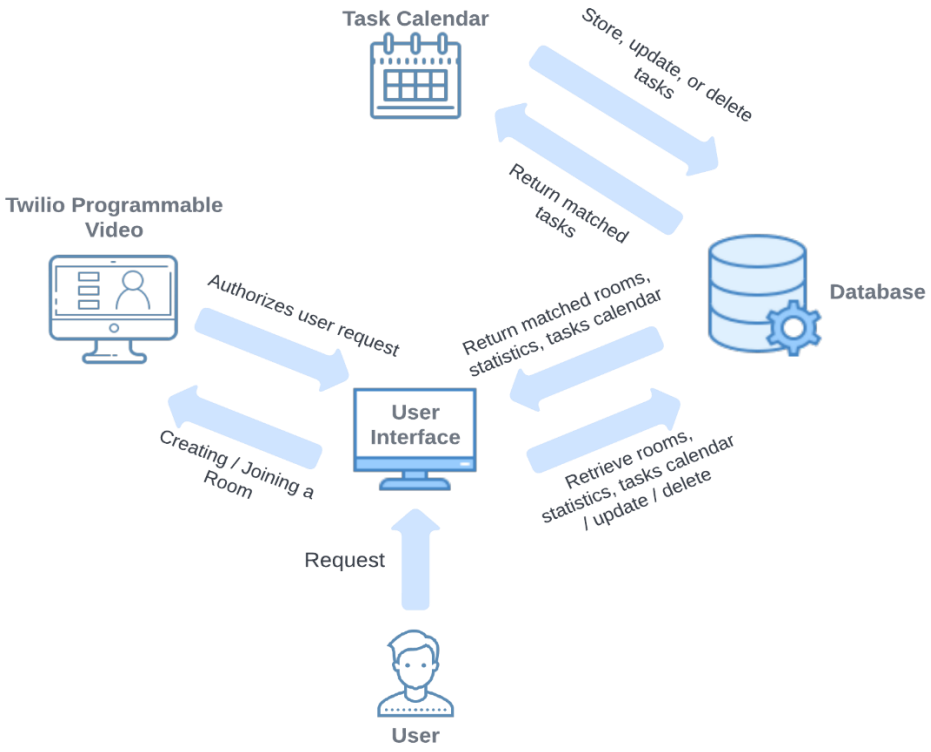

Figure 1: Conceptual Framework

Figure 1 shows the conceptual framework of the application. Through Twilio Programmable Video API, the user can create, join, record, update and complete a room. Furthermore, users who have joined a room can make use of the application's productivity tools – study goals, room timer, background image and music. Users are also given the opportunity to manage and monitor their to-dos or tasks, independent from the study room, through the application's calendar.

CHAPTER II

SOFTWARE REQUIREMENTS AND DESIGN SPECIFICATIONS

This chapter contains a detailed presentation of how the application is developed or structured and is expected to function. It contains the sections for the Use Case Diagram, Use Case Narrative, Activity Diagram, Class Diagram, and User Interface Design.

USE CASE DIAGRAM

The use case diagram presents the various activities that can be done by the user while using the application and the way the system handles and responds to the user's actions.

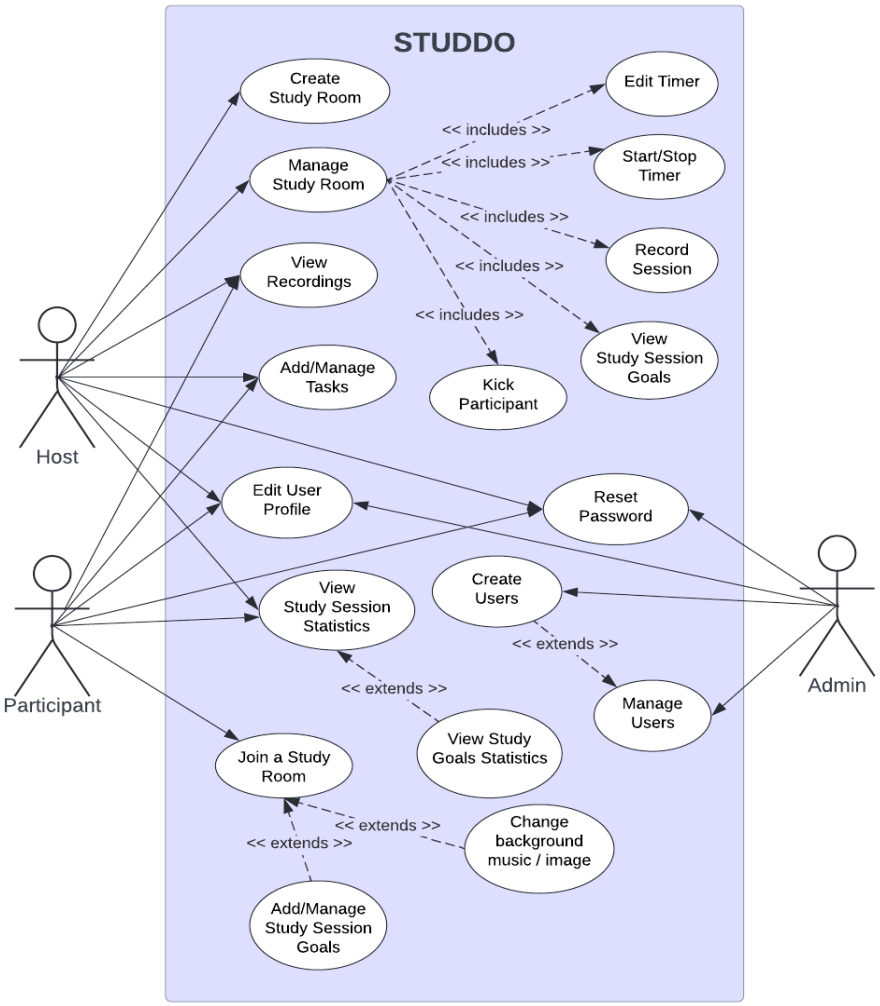

Figure 2: Use Case Model

Figure 2 shows Studdo's use case diagram which depicts the sequence of actions that a certain user (student) can perform. The user can configure room settings to their preference; join a study room; view study session statistics; and add general tasks or To-Dos to Studdo's calendar.

USE CASE NARRATIVE

Use case narratives are used to present a detailed flow of the different use cases present in the application. The breakdown of these events would allow the users to fully grasp what is happening within the system as they continue to interact with the application.

Use Case Name	Create Study Room
Actor(s)	Host
Purpose	Enable the host to personalize study sessions according to preferences or provide them with preset configurations.
Overview	Host can define the preferred study environment with given focus and break timers for each study session, music genre, and subject.
Precondition(s)	User role must be Teacher.
Post Condition(s)	User can join the study room.
Flow of Events	1. User can choose between manual and preset setting up of the study room. 1.1. With preset set-up all fields will be automatically filled.

	1.2. With manual set-up, user provides or selects name, type, timer, subject, and music genre for the study room. 2. The room will be created and the room configuration will be saved and stored in the database. 3. User will be redirected and connected to the room upon successful room creation. 4. Only the host can start, stop, and edit the timer.
Alternate Flow of Events	1. Room will not be created when the room name provided already exists or it is left empty. User will be prompted with a validation message upon submission of form.

Table 1: Create Study Room Use Case Narrative

Use Case Name	Manage Study Room
Actor(s)	User
Purpose	Enables the user to view all study room and join a specific study room.
Overview	The user can view all available study room and join their preferred study room.
Precondition(s)	User must be logged in.
Post Condition(s)	Rooms must be created
Flow of Events	1. User navigates to “Study Rooms” Tab 2. Click which study room to join 3. User selects media devices 4. User will be redirected and connected to the room
Alternate Flow of Events	1. If no study room found, no data or information will be displayed.

Table 2: Manage Study Room Use Case Narrative

Use Case Name	Edit Timer

Actor(s)	Host
Purpose	Allow users to edit the timer in the study room.
Overview	The user can edit the timer in the study room and it will be updated on the next cycle of timer.
Precondition(s)	Host must be Teacher, and must be in a study room
Post Condition(s)	None
Flow of Events	1. Host enters study room 2. Host clicks “Edit Timer” in room toolbar 3. System display window pop-up for editing timer 4. Host edit timer for Work Time/Short Break/Long Break 5. System display window pop-up alert that edited time will change on the next cycle
Alternate Flow of Events	1. If room timer is “Open Time”, “Edit Time” button is disabled and hidden in room toolbar.

Table 3: Edit Timer Use Case Narrative

Use Case Name	Start/Stop Timer
Actor(s)	Host
Purpose	To start/stop timer inside the study room
Overview	The host can start or stop the timer inside the study room and it will reflect on all participants inside the study room
Precondition(s)	Host must be a teacher.
Post Condition(s)	None
Flow of Events	1. Host Enter Study Room 2. Host navigates to timer section 3. Host clicks on Start/Stop Timer Button 3.1. If timer is paused/stopped. Study room start Timer 3.2. If timer is running. Study room stop timer.

Alternate Flow of Events	None

Table 4: Start Stop Timer Use Case Narrative

Use Case Name	Record Session
Actor(s)	User
Purpose	Allow user to record study room session
Overview	The user can record the study room session and is saved to the database.
Precondition(s)	User must be a teacher, and is in a study room.
Post Condition(s)	None
Flow of Events	1. User enters study room 2. User goes to room toolbar and clicks on “Record Session” button 3. Display window pop-up on confirming record session 4. Starts recording the session
Alternate Flow of Events	None

Table 5: Record Session Use Case Narrative Use Case Narrative

Use Case Name	View Study Session Goals
Actor(s)	Host
Purpose	Allow the host to view all study session goals inside the study room
Overview	The Host can view all study session goals and their progress set by the participants of the study room
Precondition(s)	Host must be a teacher and must have a study session goal set by the participants
Post Condition(s)	None.

Flow of Events	1. Host Enter study room 2. Host navigates to room toolbar 3. Host Clicks on "View All Study Session Goals" Button 4. Host is redirected to a new tab and displays all Study Session Goals in the Study room
Alternate Flow of Events	1. If no study session goal detected, no data or information will be displayed.

Table 6: View Study Session Goals Use Case Narrative

Use Case Name	Kick Participant
Actor(s)	Host
Purpose	Allow the host to kick specific participants
Overview	The Host can kick participants from the study room.
Precondition(s)	Host must be a teacher, and is inside a study room with participants
Post Condition(s)	None
Flow of Events	1. Host enter study room 2. Participants enter study room 3. Host Clicks kick icon on the participant's video 4. System Display Window pop-up confirming Host kicking participant 5. Participant Leaves Study Room
Alternate Flow of Events	1. If no participants inside the study room, no button will be displayed

Table 7: Kick Participant Use Case Narrative

Use Case Name	View Recordings
Actor(s)	Users

Purpose	Allow users to view the recordings of study sessions as well as downloading them.
Overview	The user can view the recordings of study sessions that have been recorded and download them.
Precondition(s)	Must have a study session recording in the database.
Post Condition(s)	None
Flow of Events	1. User navigates to Profile tab 2. User navigates to view recordings 3. User clicks on recording link of a specific study session 4. System redirects User to Download Recording
Alternate Flow of Events	None

Table 8: View Recordings Use Case Narrative

Use Case Name	Add/Manage Tasks
Actor(s)	User
Purpose	Add or create to-do tasks and keep track of it through the calendar for task management and monitoring.
Overview	The user can add to-do tasks and it will be displayed both in the to-do list and calendar, if due date is provided. Users can also edit and delete a task and changes will reflect accordingly.
Precondition(s)	User must create a task first.
Post Condition(s)	Tasks with due date will be displayed on the calendar.
Flow of Events	1. Create new to-do by providing the necessary information: Task Name and Due Date (optional). 2. Editing information of created task and changes will reflect accordingly.

	3. Deleting a task will remove it from the to-do list as well as calendar.
Alternate Flow of Events	None

Table 9: Add/Manage Tasks Use Case Narrative

Use Case Name	Reset Password
Actor(s)	User
Purpose	Allow Users to reset their password
Overview	The user can reset their passwords and will be updated in the database
Precondition(s)	User must have data in the database
Post Condition(s)	None
Flow of Events	1. User navigates Profile tab 2. User Clicks on "Reset Password" Button 3. Display Window Pop-up form for Resetting Password 4. User Inputs Updated Password 5. User Click "Reset" Button and is updated to the Database
Alternate Flow of Events	None

Table 10: Reset Password Use Case Narrative

Use Case Name	Create Users
Actor(s)	Admin
Purpose	Allows admin to create teachers/students
Overview	The admin can create teachers/students and is stored to the database

Precondition(s)	Must be an Admin
Post Condition(s)	None.
Flow of Events	1. Admin navigates to dashboard 2. Admin clicks on “Create user” and system displays window pop-up form for adding student/teacher 3. Admin inputs Teacher/Student details 4. Admin clicks “Create” button and is saved to the database and is reflected in the admin dashboard.
Alternate Flow of Events	None

Table 11: Create Users Use Case Narrative

Use Case Name	Manage Users
Actor(s)	Admin
Purpose	Admin can manage user data.
Overview	The admin can add, edit or delete users from the admin dashboard and will be updated in the database.
Precondition(s)	User must be an Admin
Post Condition(s)	None.
Flow of Events	1. Admin navigates to Dashboard 2. Admin toggles between student and teacher and display accordingly. 3. Admin clicks button 1. If Admin clicks edit user button, System Display Window Pop-up for Editing a Specific User Info asking for admin input, Admin Clicks "Update" and is Updated from the Database. 2. If Admin clicks reset password button, System Display Window Pop-up for Resetting a Specific User Password asking for admin

	input, Admin Clicks "Update" and is Updated from the Database. 3. If Admin clicks delete user button, System Display Window Pop-up for Confirming Deleting a Specific User Admin Clicks "Yes" and User is deleted from the Database.
Alternate Flow of Events	None

Table 12: Manage Users Use Case Narrative

Use Case Name	Edit User Profile
Actor(s)	Users
Purpose	Edit user profile preferred by the users
Overview	The user can edit their profile and will be updated in the database.
Precondition(s)	Must have user data in the database
Post Condition(s)	None
Flow of Events	1. User navigates to Profile tab 2. User clicks on “Edit Details” button 3. System displays window pop-up where the user can edit their information 4. User input updated information 5. User clicks “Update” and is updated to the database.
Alternate Flow of Events	None

Table 13: Edit User Profile Use Case Narrative

Use Case Name	Join Study Room
Actor(s)	Participant
Purpose	Allow User to enter a study room.

Overview	A list of available study rooms together with its details will be displayed and users can select which room to join and proceed with their study session.
Precondition(s)	Rooms must be created and camera device should be selected.
Post Condition(s)	Participant cannot control the timer.
Flow of Events	1. User selects from a list of available rooms to join. 2. User will be connected to the selected room and asked to allow media permissions. 3. Add study goals to achieve during the session. 4. Option to change study room background music and image.
Alternate Flow of Events	1. When a room is already completed and at the same time the user clicked to join the said room, the user will not be able to connect to the room. Instead, user is redirected back to the Study Room tab.

Table 14: Join a Study Room Use Case Narrative

Use Case Name	Add/Manage Study Session Goals
Actor(s)	Users
Purpose	Allow users to add, edit and delete study session goals.
Overview	The user can add, edit and delete study session goals inside the study session room.
Precondition(s)	User must be in a study room
Post Condition(s)	None
Flow of Events	1. User enter study room 2. User clicks on either “Add Goal” in room toolbar, “Edit Goal” and “Delete Goal” in study goal section

	1. For “Add Goal” Button, system displays window pop-up on adding a goal asking for user input 2. For “Edit Goal” Button, system displays window pop-up on edit a goal asking for user input 3. For “Delete Goal” Button, system deletes goal the user clicked 2. Added, Edited, or Deleted goal will be reflect and refresh all study goal tables.
Alternate Flow of Events	1. If user haven’t added a goal, no data or information will be displayed.

Table 15: Add/Manage Study Sessions Goals Use Case Narrative

Use Case Name	Change Background Image/Music
Actor(s)	User
Purpose	Changing the Background Image/Music in the study room session.
Overview	The user can change their preferred background image/timer in the study room session.
Precondition(s)	Must be in a study room session.
Post Condition(s)	None
Flow of Events	1. User navigates to room toolbar 2. User clicks on Change Background Image or Change Background Music button 3. User clicks on their desired background image or music. 4. Chosen image will be reflected as well as the chosen music will played in the study session room.

Alternate Flow of Events	None.

Table 16: Change Background Image/Music Use Case Narrative

Use Case Name	View Study Sessions Statistics
Actor(s)	User
Purpose	Keeping track of User's study sessions.
Overview	The user will see a daily, weekly or monthly chart of their study session activity as well as average and total study time. Study goals for each session is also presented.
Precondition(s)	User must have at least one study session.
Post Condition(s)	None
Flow of Events	1. User navigates to Profile tab. 2. Weekly and monthly study time and study goals will be displayed as well as average and total study time. 3. Average, total, and completion rate of study goals is displayed. 4. Recording links (if applicable) are also displayed below.
Alternate Flow of Events	1. If user haven't started a study session yet, no data or information will be displayed.

Table 17: View Study Session Statistics Use Case Narrative

Use Case Name	Leave Study Room
Actor(s)	User
Purpose	To allow user to leave the study room.
Overview	The user can leave the study room at any given time regardless of whether the room is in Work Time or Break Time.

Precondition(s)	User must be in a study room
Post Condition(s)	None
Flow of Events	1. User Enter Study Room 2. User Clicks on "Leave Study Room" Button in room toolbar. 3. System display window pop-up confirming User leaving room. 4. User clicks "Leave" Button redirects the user to study room page
Alternate Flow of Events	None

Table 18: Leave Study Room Use Case Narrative

Use Case Name	End Study Session
Actor(s)	Host
Purpose	To end the study session and the to leave all participants in the study room
Overview	The Host can end the study session and the participants inside the study room will be kicked out of the room and close the study room
Precondition(s)	User must be a teacher and in a study room
Post Condition(s)	None
Flow of Events	1. Host Enter Study Room 2. Participant Enters Study Room 3. Host Clicks on "End Study Session" Button in Room Toolbar 4. System Display Window pop-up confirming Host Ending Study Session All Participants are removed in the Study Room
Alternate Flow of Events	None

Table 19: End Study Session Use Case Narrative

ACTIVITY DIAGRAM

Activity Diagrams are used to show how the system behaves – the different processes that take place whenever the user interacts with the application. The flow of processes for each activity is portrayed by different shapes showing the various paths that exist while the processes are taking place.

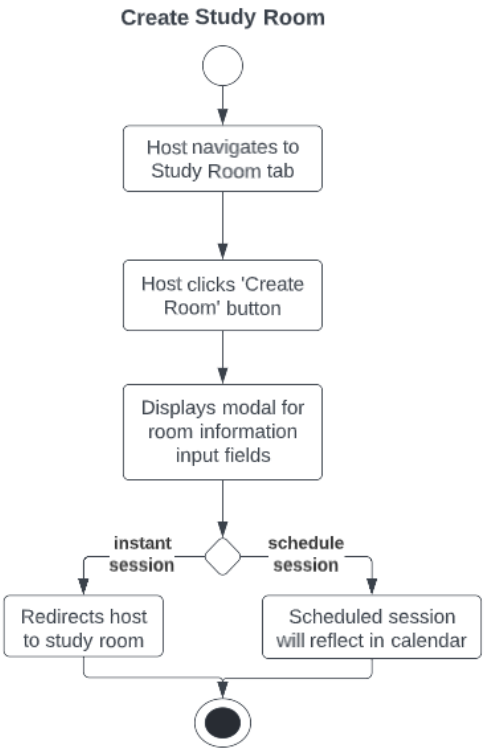

Figure 4: Create Study Room Activity Diagram

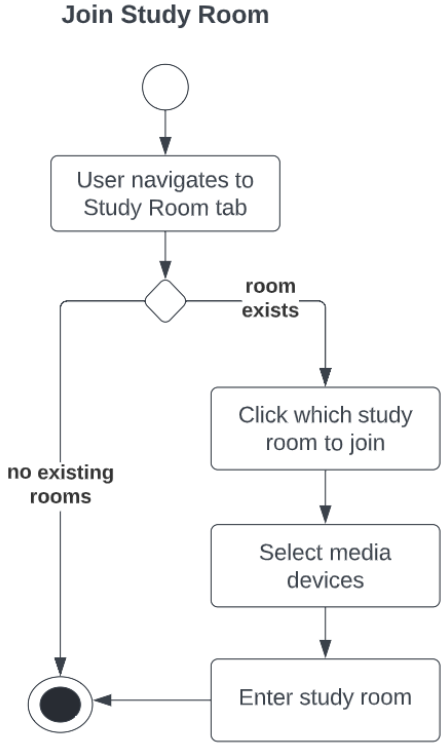

Figure 3: Join Study Room Activity Diagram

Figure 3 shows how the user can interact with the system's room creation function. The user can configure the study room by filling out the information fields from the modal displayed. If the room is set as scheduled session, the session will be displayed on the home page and add to the calendar. If the session is instant, the user will then be redirected to the study room that was created.

Figure 4 shows the user can join a study room that is ongoing or a scheduled room. Upon entering the room, the user will be asked for their media permission. Inside the room, the room host or leader can start the study session by starting the timer based on the set focus and break timers. When the session is completed, their time inside the room will be recorded.

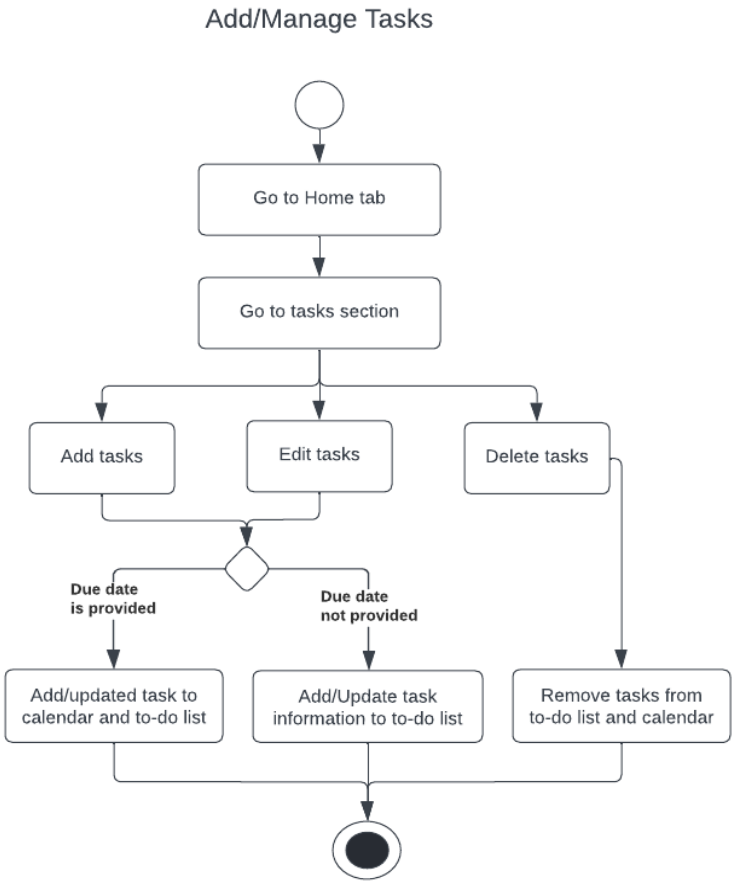

Figure 5: Add/Manage Tasks Activity Diagram

Figure 5 shows how the user can manage their tasks under the to-do list and monitor it in the calendar. User can create, edit and delete tasks. When due date is not set during adding of task, it will not be displayed in the calendar, otherwise task is reflected in the calendar. Same thing for editing a task, if due date is omitted or added, changes

will reflect accordingly. When a task is deleted, it will be removed from the to-do list and/or calendar.

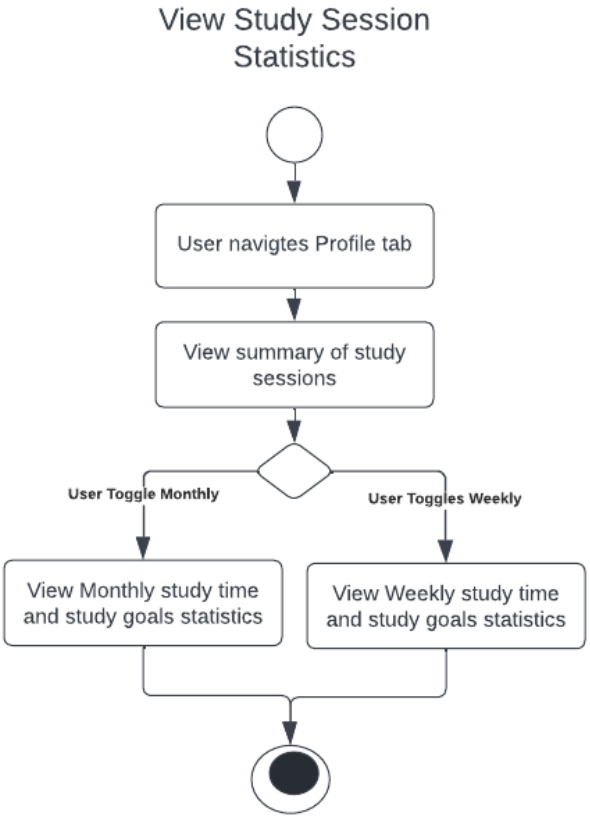

Figure 7: View Study Session Statistics Activity Diagram

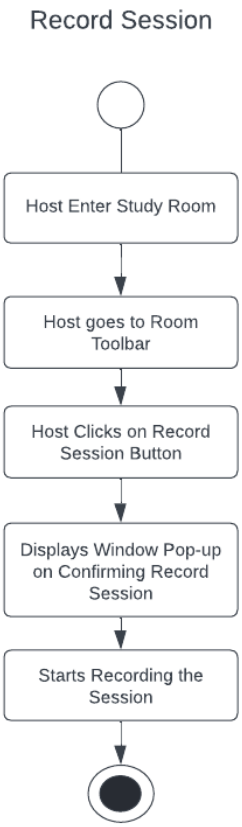

Figure 6: Record Session Activity Diagram

Figure 6 shows how to navigate the Users Study Session Statistics. In which the user can view their monthly/daily study time and study goals statistics. Aside from the chart, an average and total study time will also be displayed. Users will also see a list of their rating or feedback to their previous study sessions.

Figure 7 show how the host can record the study session. In which the host can record the study session and shows how the user navigates to record the study session.

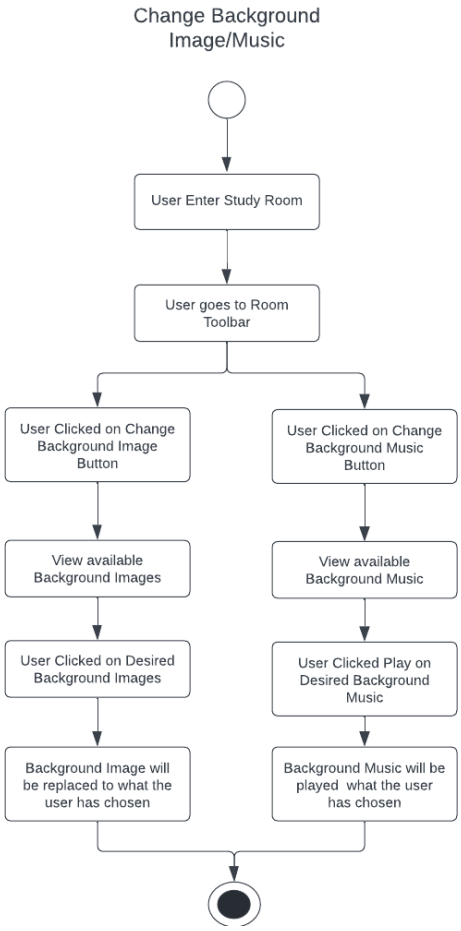

Figure 9: Change Background Image/Music Activity Diagram

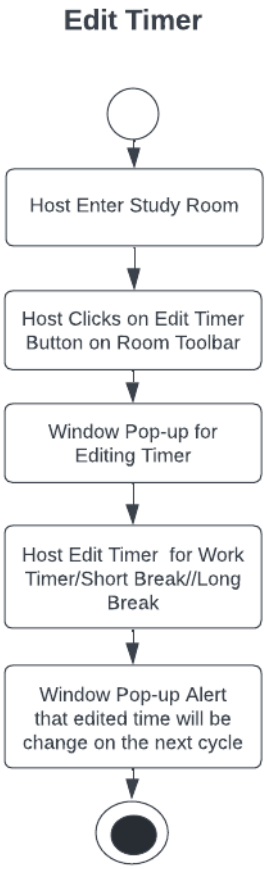

Figure 8: Edit Timer Activity Diagram

Figure 8 shows how the user can change the background music and image. It shows how the user navigate on how to change the background image or music, as well as what are the corresponded action after selecting the desired music or image.

Figure 9 show how the host of the study room can edit the timer in the Study Session. In which the host can edit the timer and it will be reflected on the next cycle of Work Time in the Timer.

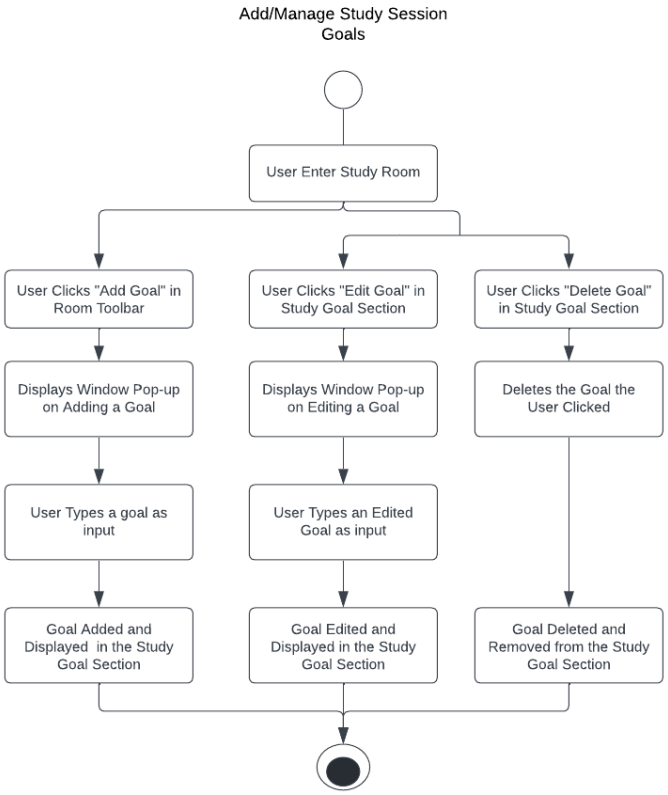

Figure 10: Add/Manage Study Session Goals Activity Diagram

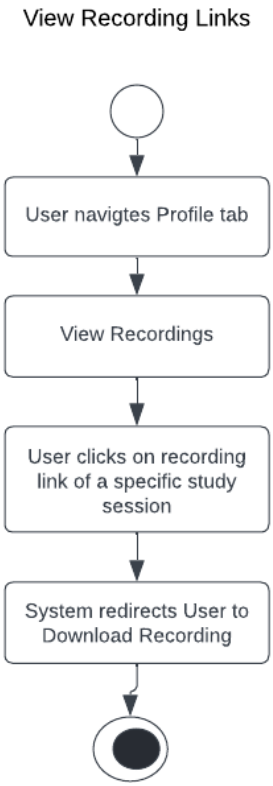

Figure 11: View Recordings Activity Diagram

Figure 10 shows how the user can add, edit and delete study session goals in the room. It shows how the system handles what the user wants to add, edit or delete a study session goal.

Figure 11 shows how the user navigate and download Study Session Recordings. In which the user can locate all their Study Session Recordings as well as the option to download them.

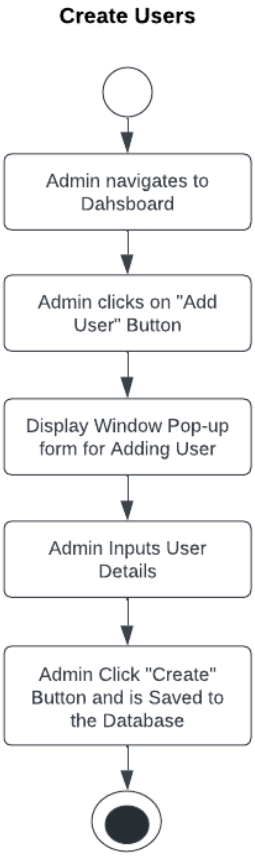

Figure 13: Create Users Activity Diagram

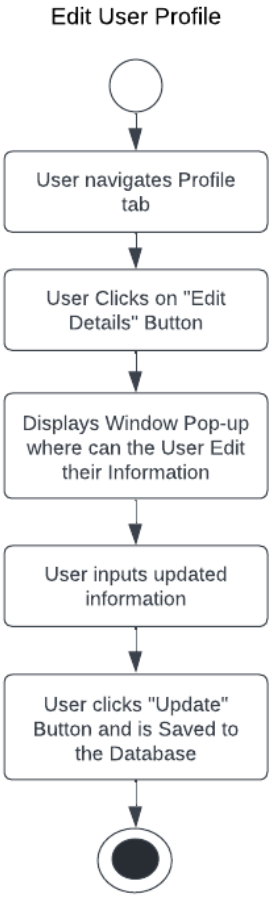

Figure 12: Edit User Profile Activity Diagram

Figure 12 shows how the admin can create Users. This shows the flow of how the admin can create a User from the Admin Dashboard and what would happen after creating a User.

Figure 13 shows how the user Edits their User Profile. It shows how the system handles when the user wants to edit their User Profile and how to navigate in order for the user to edit their profile.

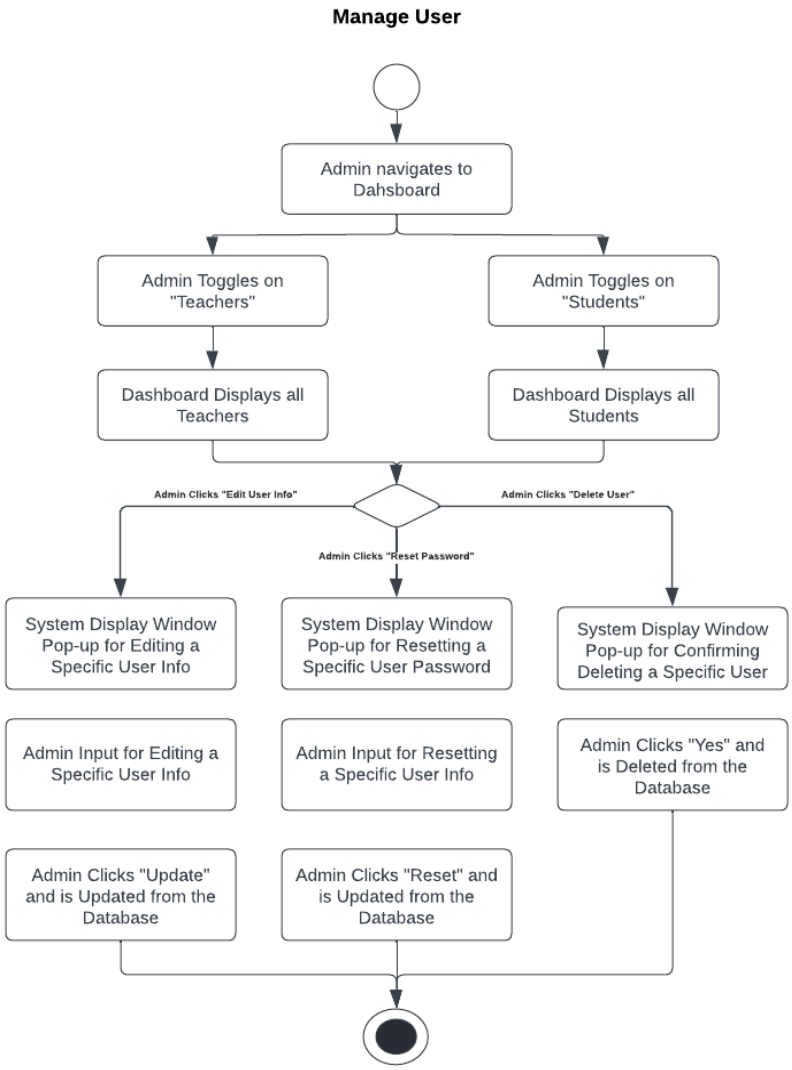

Figure 14: Manage Users Activity Diagram

Figure 14 shows how to the admin can navigate all Users as Teachers or Students. In which the admin can edit, delete users as well as resetting the users' password.

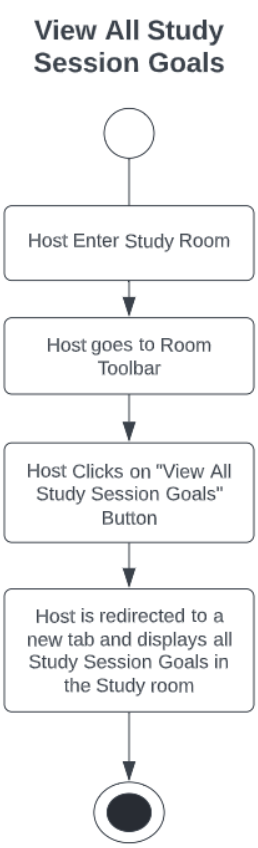

Figure 16: View All Study Session Goals Activity Diagram

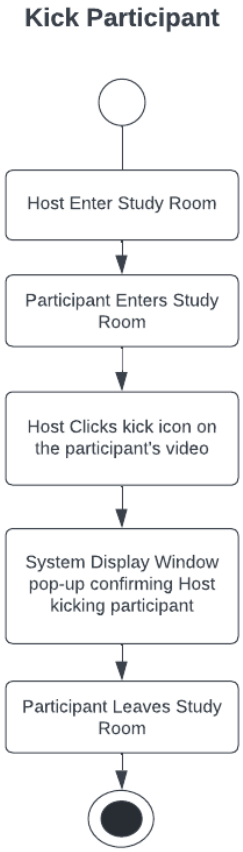

Figure 15: Kick Participant Activity Diagram

Figure 15 shows how to the How can view all study session goals. In which the host can view all study session goals set by the participants and their progress of completed study session goals.

Figure 16 shows how to the host can kick participants in the study room. In which the host can kick a specific participant from the study room.

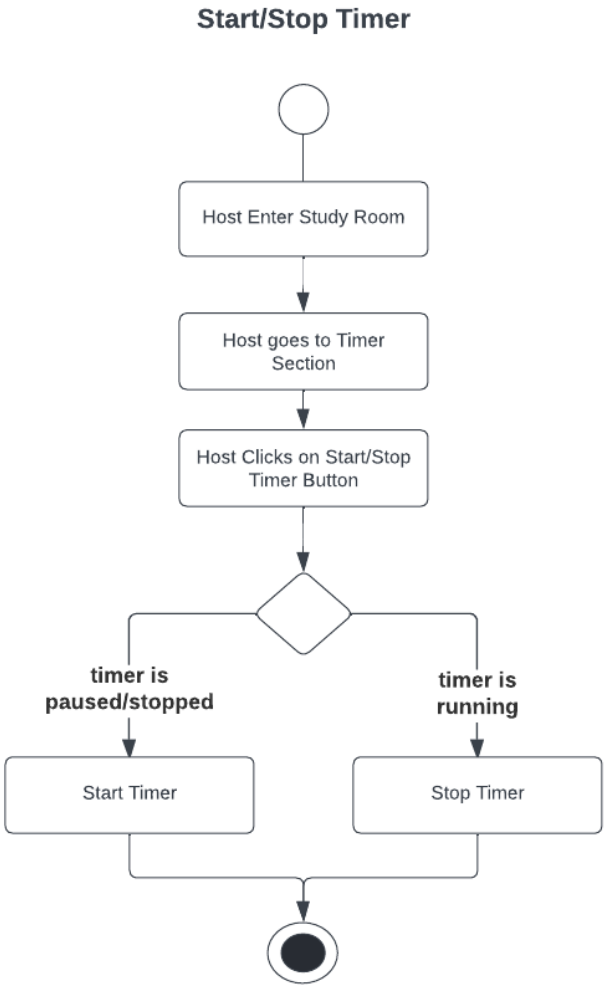

Figure 17: Start/Stop Timer Activity Diagram

Figure 17 shows how to the host can start/stop timer in the study room. In which the host start/stop the timer in the study room meaning work time is in session or session is paused.

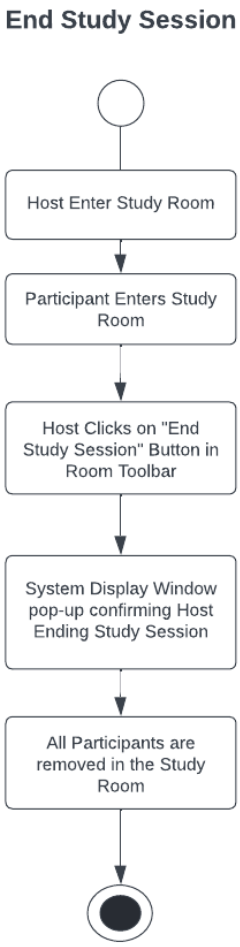

Figure 19: End Study Session Activity Diagram

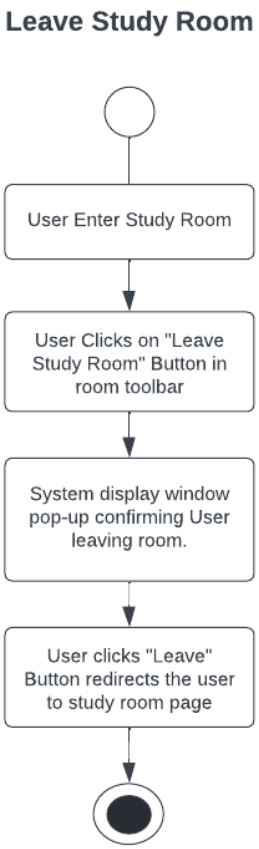

Figure 18: Leave Study Room Activity Diagram

Figure 18 shows how the host can end the study session inside the study room. In which, the host can end the study session, leaves all participants in the study room and closes the study room to prevent any participants joining the study room again.

Figure 19 shows how the user can leave the study room. In which the user can leave the study room at the preference of the user.

CLASS DIAGRAM

A class diagram represents the system's structure as relationships of classes – indicating how the different classes interact with each other. Each class contains their own attributes and operations. The type and cardinality of relationships between classes are also shown.

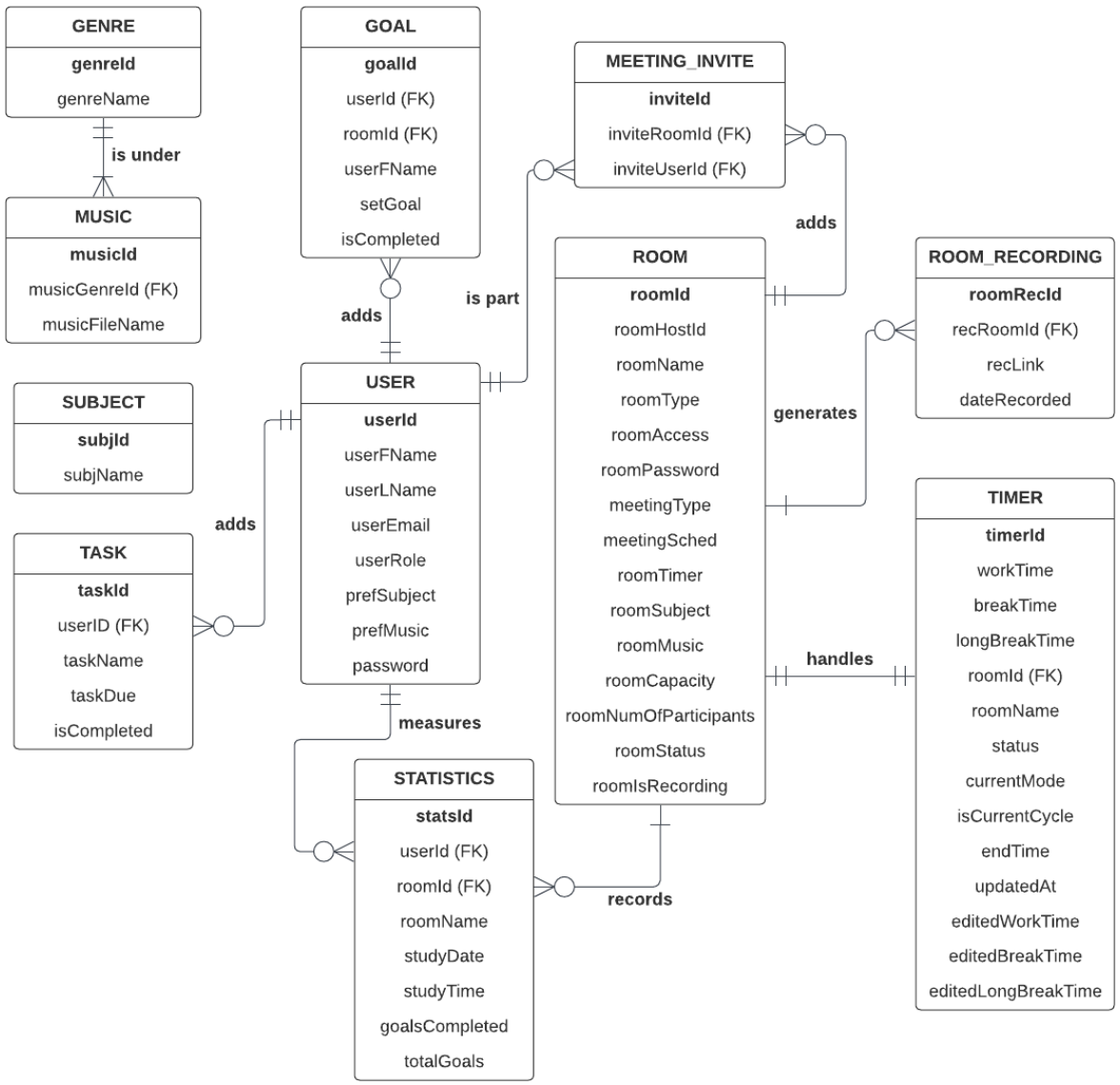

Figure 20: Entity Relationship Diagram

USER INTERFACE DESIGN

Figure 21: Login Screen

Figure 19 is the login screen where the users can login in to their accounts using their registered email and password. Accounts are either given by the Administrator or users register for a new account.

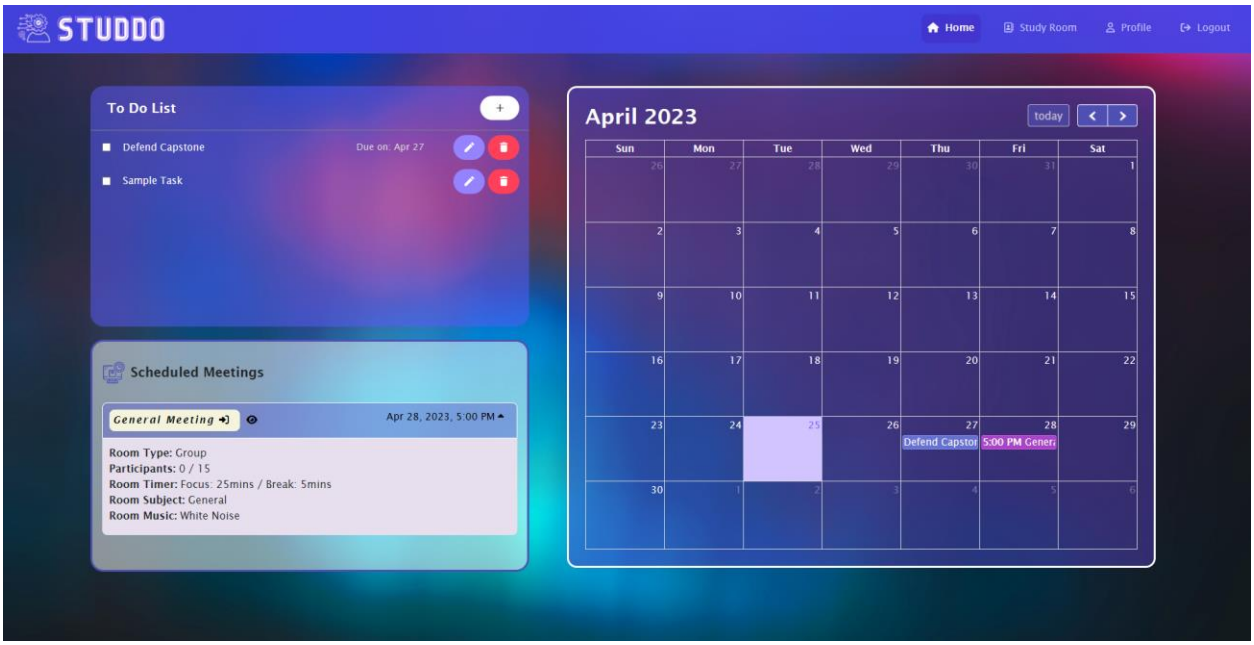

Figure 22: Home Screen

Figure 20 is the Home screen or the Dashboard of the User consisting the To Do List, Scheduled Sessions, and Calendar. The interfaces above shows both when empty and with data on the lists. Data on the calendar will only reflect when there are existing items in To Do List (with due date) and Scheduled Sessions.

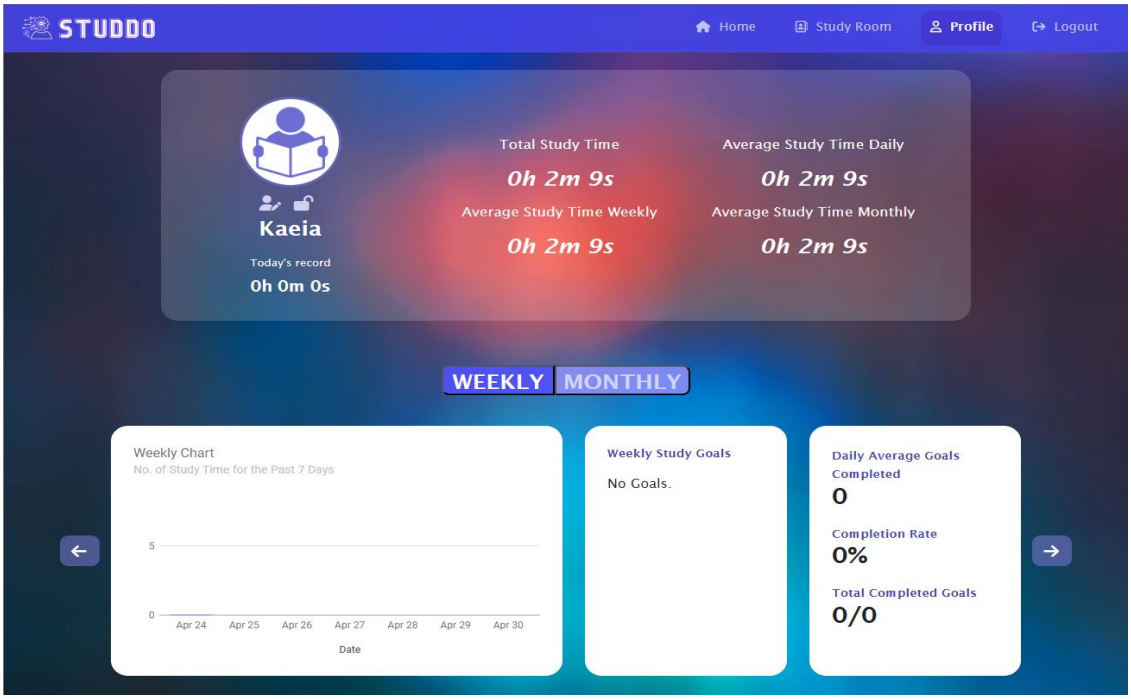

Figure 23: Profile screen (student)

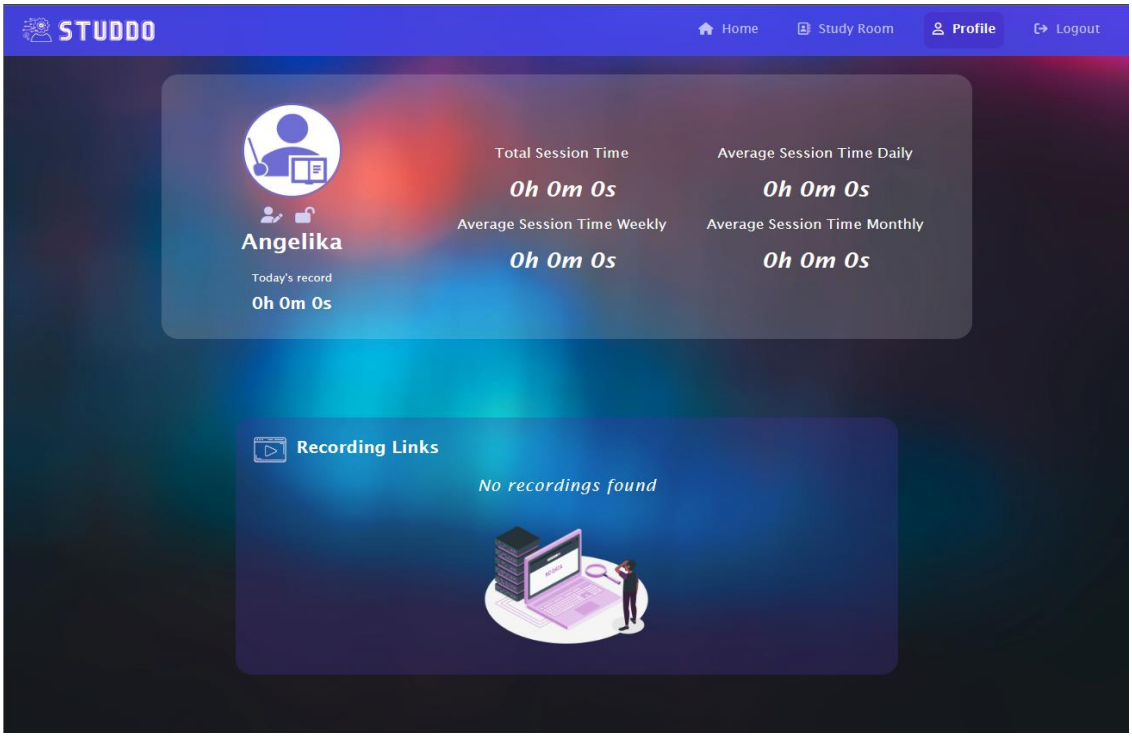

Figure 24: Profile screen (teacher)

Figure 21 and 22 are the Profile screen mainly shows the study statistics of the user and also the recordings. The top area shows the overall or summary statistics of the user's study sessions. This includes the Total Study Time and Average Study Time Daily, Weekly, and Monthly. There is also a weekly and monthly toggle that would display accordingly. Weekly toggle displays the Weekly Chart, Weekly Study Session Goals, Daily Average Goals Completed, Completion Rate, and Total Completed Goals for the week only. User can navigate to the previous/next weeks using the arrow buttons. For the monthly display, just the total study time will be shown per month, and the same set of data for the Study Goals, except that it is for the whole month. Below the statistics, the recording links are displayed for the recorded study sessions.

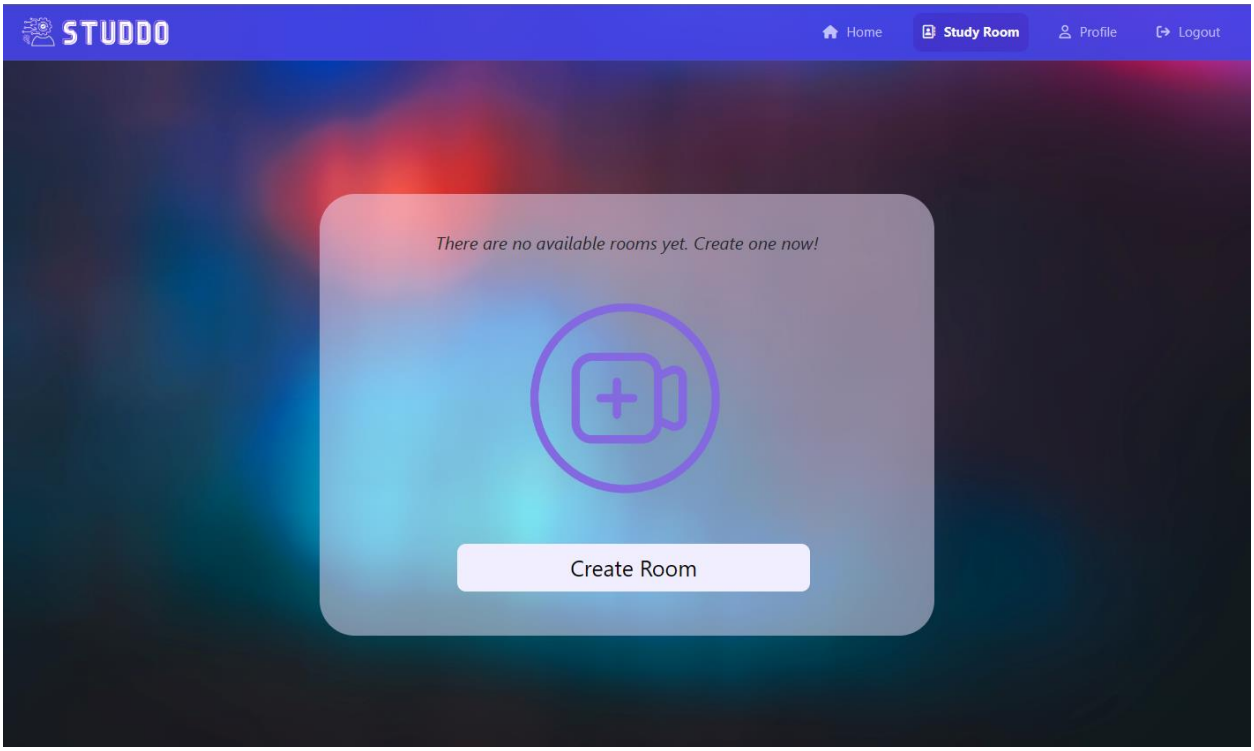

Figure 25: Study Room tab

Figure 23 is the study room mainly consists of all study rooms that are ongoing. It shows all the information of a study room as well as the number of participants inside the study room. If there are no rooms detected, on the host side, the create button will be displayed in order to create a room.

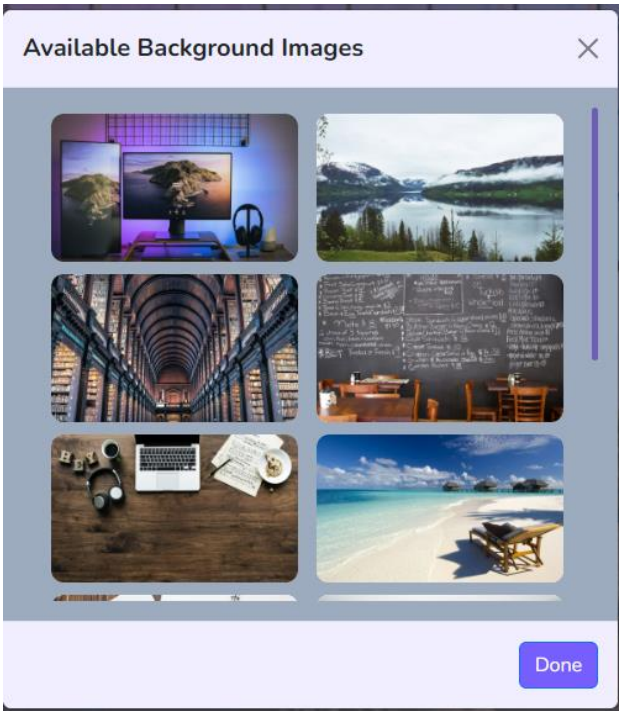

Figure 26: Change Background Image

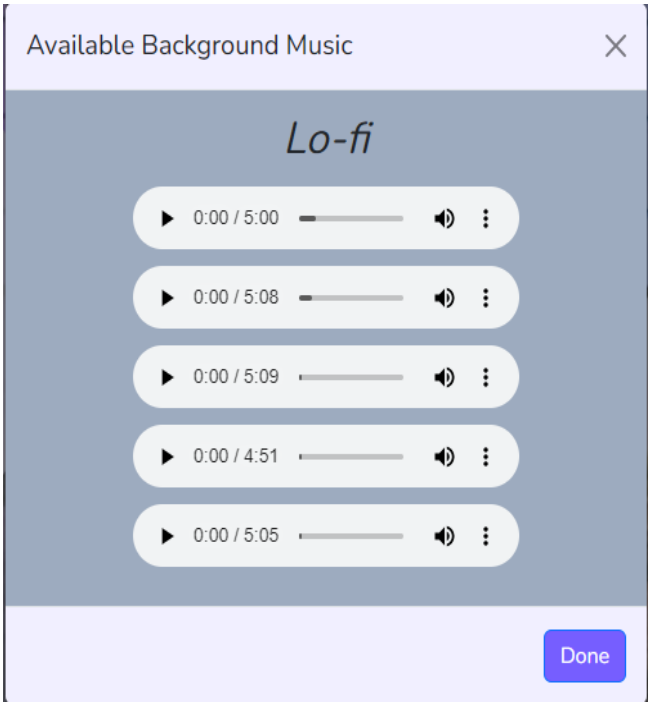

Figure 27: Change Background Music

Figure 24 and 25 are window pop-ups are a study room feature that displays a range of available images and music that are given by the system. This is only found in the room toolbar inside the study room and is available to both host and participant.

Figure 28: Start/Stop Timer

Figure 26 is the Timer that is found on the right side of the study room. It displays the current Focus Timer as well as remaining time of that focus timer. for the host, a button will be displayed to start or stop the timer.

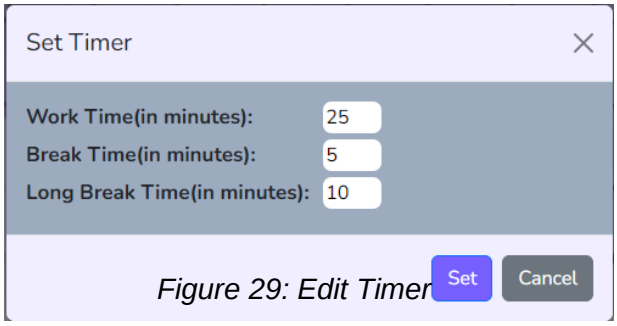

Figure 29: Edit Timer

Figure 27 is a window pop-up from the room toolbar that lets the host edits the amount of time for each focus timer in the study room.

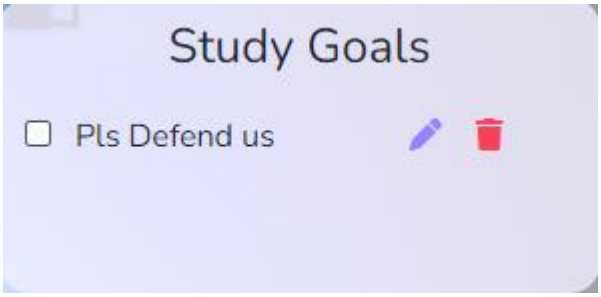

Figure 30: Study Goals

Figure 28 is the study goal section found at the rightmost in the study room. it displays all study goals that is set by the participant.

Figure 31: Study Room

Figure 29 is the interface inside the study room. In the center are the host with the participants with their camera open. On the leftmost part is the room toolbar while on the right where the timer and study goal section can be found.

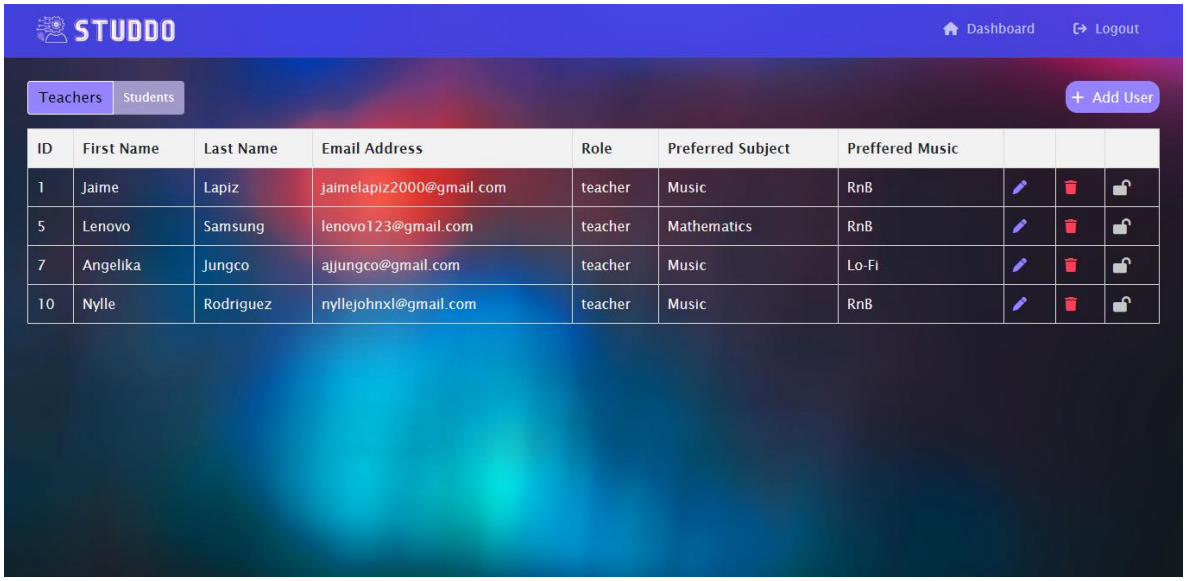

Figure 32: Admin Dashboard

Figure 30 is the admin dashboard that can be only accessed by the admin of the application. It displays all the users that is registered in the database. The admin can add, edit, delete, and reset passwords of the user.

CHAPTER III

SOFTWARE DEVELOPMENT AND TESTING

This chapter describes how the project is implemented including the tools and software used to develop and test the application.

DEVELOPMENT AND TESTING PROCESS

The project is a pure web-based application mainly using Laravel web services in building the application while the database management is handled using SQL or Structured Query Language.

In building the application, the following development environment, tools, and software are used:

- Visual Studio Code 1.72.2 – an IDE that is optimized for building and debugging modern web and cloud applications and supports various programming languages; the main code editor to develop the web application.
- MySQL Workbench 8.0.23 CE – a visual database design tool that integrates SQL development, administration, database design, creation and maintenance into a single integrated development environment for the MySQL database system.
- Laravel Framework 9.21.5 – an open-source PHP web framework based on Symfony; it is intended for developing web applications following the Model-View-Controller (MVC) architectural pattern.

- Twilio Video API – a programmable real-time communications platform that allows you to add video chat functionality to your web, iOS, and Android applications; used to implement WebRTC.
- FullCalendar 5.11.3 – a JavaScript library to display calendar and events
- Google Chart – is an interactive Web service that creates graphical charts from user-supplied information. The user supplies data and a formatting specification expressed in JavaScript embedded in a Web page; in response the service sends an image of the chart.

Development Process

This section presents the process of implementation of the application's features.

1. Add/Manage Tasks

This feature includes a To-Do List that displays the tasks created by the users. When creating a task, task name is required and providing a due date is optional and then saved to the MySQL database. The list displays the values that were returned from the database query. The user can also edit and delete the task.

1.1 Add Task

```
public function create(Request $request) {

    if ($validator->fails()) {
      return redirect()
        ->back()
        ->withInput()
        ->withErrors($validator);
    }

    $task = new Task;
    $task->create([
      'user_id' => Auth::user()->id,
```

```
      'name' => $request->name,
      'task_due' => $request->task_due,
      'isCompleted' => 'false'
    ]);

    $taskInfo = json_decode($task);
    return $taskInfo;
  }
```

Figure 33: Add Task Code

1.2 Edit Task

```
public function taskEdit($id) {
    $task = Task::findOrFail($id);
    $jsonResult = json_encode($task);

    return $jsonResult;
}

public function taskUpdate(Request $request, $id) {
    $task = Task::findOrFail($id);
    $task->update([
      'name' => $request->newTaskName,
      'task_due' => $request->newTaskDue,
    ]);

    return $task;
}
```

Figure 34: Edit Task Code

1.3 Delete Task

```
public function delete($id) {
    $task = Task::findOrFail($id);
    $task->delete();
}
```

Figure 35: Delete Task Code

1.4 Complete/Check Task

```
public function complete($id) {
```

```
    $task = Task::findOrFail($id);

    if ($task->isCompleted == 'false') {
      $task->update([
        'isCompleted' => 'true'
      ]);
    }
   else if ($task->isCompleted == 'true') {
      $task->update([
      'isCompleted' => 'false'
      ]);
   }

    return $task->isCompleted;
  }
```

Figure 36: Complete/Check Task Code

2. View Study Session Statistics

Figure 35 is the function for getting the user statistics in the database. The function automatically loads after the user navigates to profile tab. The function gets the current users ID then gets the Statistics table to get the data of the user.

```
public function getUserStats()
    {
        $currentUser = Auth::user()->id;
        $user = User::where('id', $currentUser)->first();
        $roomRecordings = new RoomRecording;
        $genres = Genre::getAllGenres();
        $subjects = Subject::getAllSubjects();
        $studystats = Statistics::where('user_id', $currentUser)-
          >selectRaw('study_date, sum(study_time) as study_time')-
          >groupBy('study_date')->orderBy('study_date')->get();
        $totalstats = Statistics::where('user_id', $currentUser)-
          >selectRaw('round(avg(study_time),2) as avg_hours,
          sum(study_time)')->first();
        $weekly = Statistics::where('user_id', $currentUser)
           ->selectRaw('week(study_date), sum(study_time)')
          ->groupByRaw('week(study_date)')->get();
        $monthly = Statistics::where('user_id', $currentUser)
```

```
        ->selectRaw('month(study_date), year(study_date),
          sum(study_time)')->groupByRaw('month(study_date),
          year(study_date)')->orderByRaw('month(study_date) asc')->get();
        $recordingStats = Statistics::select('room_id')->where('user_id',
            $currentUser)->distinct()->get();
        $goals = Statistics::where('user_id', $currentUser)->get();

        return view('statistics', with(compact('user', 'genres',
          'subjects', 'studystats', 'totalstats', 'weekly', 'monthly',
          'goals')));
}
```

Figure 37: View Study Session Statistics Code

2.1 Study Time Summary

Study hours are presented in the charts by days per week using Google Chart. Data are taken from the study session timers. Weekly chart presents the number of study hours per day for the past 7 days by default. Users can navigate the previous/next week's using arrow buttons.

Weekly Statistics

```
    google.charts.load('current', {
      'packages': ['bar']
    });
    google.charts.setOnLoadCallback(drawWeeklyChart);

    lastweekbutton.onclick = function() {
      tempDate.setDate(tempDate.getDate() - 7);
      google.charts.setOnLoadCallback(drawWeeklyChart);
    }
    nextweekbutton.onclick = function() {
      tempDate.setDate(tempDate.getDate() + 7);
      google.charts.setOnLoadCallback(drawWeeklyChart);
    }

    var stats = <?php echo $studystats ?>;
    var goals = <?php echo $goals ?>;

    function drawWeeklyChart() {
      var data = new google.visualization.DataTable();
      data.addColumn('string', 'Date');
```

```
    data.addColumn('number', 'No. of Hours');

    var currentDate = new Date(tempDate);
    for (i = 6; i >= 0; i--) {
      weeklyDate[i] = convertDate(currentDate);
      currentDate.setDate(currentDate.getDate() - 1)
    }

    var weekTime = [0, 0, 0, 0, 0, 0, 0];
    var formattedDate = [];
    var totalWeekTime = 0;

    for (i = 0; i < 7; i++) {
      for (j = 0; j < stats.length; j++) {
        studyDate = convertDate(new Date(stats[j]['study_date']));
        if (weeklyDate[i] == studyDate) {
          weekTime[i] = stats[j]['study_time'];

        }
      }

      formattedDate[i] = new Date(weeklyDate[i]).toLocaleDateString('en-
         us', {
        month: "short",
        day: "numeric",
      });

      var hours = weekTime[i] / 3600000;
      var weekStats = hours;

      data.addRow([formattedDate[i], weekStats]);
      totalWeekTime += weekStats;
    }

  var options = {
      chartArea: {
        backgroundColor: 'transparent',
      },
      backgroundColor: 'transparent',
      legend: {
        position: 'none'
      },
      vAxis: {
        viewWindowMode: 'explicit',
        viewWindow: {
```

```
            min: 0,
            max: 8
          }
        },
        chart: {
          title: 'Weekly Chart',
          subtitle: 'No. of Study Time for the Past 7 Days'
        },
        axes: {
          x: {
            0: {
              side: 'bottom',
              label: 'Date'
            }
          }
        },
        bar: {
          groupWidth: "80%"
        }
      };

      var chart = new google.charts.Bar(document.getElementById('stats-
       chart'));

      chart.draw(data, google.charts.Bar.convertOptions(options));
    };
```

Figure 38: Weekly Statistics Code

Monthly Statistics

```
var monthly = <?php echo $monthly ?>;
var stats = <?php echo $studystats ?>;
var goals = <?php echo $goals ?>;

const currentMonth = new Date().getMonth();
const currentYr = new Date().getFullYear();
var tempMonth = currentMonth;
var tempYr = currentYr;
var newMonth;
newMonth = getMonthName(currentMonth);
var monthStats;

var lastmonthbutton = document.getElementById('lastmt');
var nextmonthbutton = document.getElementById('nextmt');
```

```
    displayMonthStats(tempMonth, tempYr, monthly, goals)

    lastmonthbutton.onclick = function() {
      tempMonth -= 1;

      if (tempMonth < 0) {
        tempMonth = 11;
        tempYr -= 1;
      }

      displayMonthStats(tempMonth, tempYr, monthly, goals)

    }

    nextmonthbutton.onclick = function() {
      tempMonth += 1;

      if (tempMonth > 11) {
        tempMonth = 0;
        tempYr += 1;
      }

      displayMonthStats(tempMonth, tempYr, monthly, goals);
    }
  }
```

Figure 39: Monthly Statistics

2.2 View Study Goals Statistics

Figure 38 is a function for displaying the statistics of the user after getting the data from figure 35. The function also converts the data into readable and user-friendly data of the total week time, total week goals, count the number of goals in the week, total completed goals, and goals percentage completion.

```
var weekGoals = [];
var totalWeekTime = 0;
var totalWeekGoals = 0;
var weekGoalsCount = 0;
var totalCompletedGoals = 0;
var goalsPercent = '';
```

```
for (i = 0; i < 7; i++) {
   for (j = 0; j < stats.length; j++) {
     studyDate = convertDate(new Date(stats[j]['study_date']));
     if (weeklyDate[i] == studyDate) {
      weekTime[i] = stats[j]['study_time'];

     }
   }

var goalDiv = document.getElementById('goals-list');
goalDiv.innerHTML = '';
var goalsTitle = document.getElementById('goalsTitle');
goalsTitle.innerHTML = 'Weekly Study Goals';

for (k = 0; k < goals.length; k++) {
   var roomGoalsCompleted = goals[k]['goals_completed'] == null ? 0 :
        goals[k]['goals_completed'];
   var totalRoomGoals = goals[k]['total_goals'] == null ? 0 :
        goals[k]['total_goals'];
   var studyGoalsDate = convertDate(new Date(goals[k]['study_date']));
   if (weeklyDate[i] == studyGoalsDate && totalRoomGoals != 0) {
     var goalItem = document.createElement('span');
     goalItem.setAttribute('id', 'goal-item');
     goalItem.innerHTML = `
       <p id="roomGoals">${goals[k]['room_name']}</p><br>
       <p id="dateGoals">${goals[k]['study_date']}</p> <br>
       <img class="check" src="{{asset('images/check.png')}}" alt="">
       ${roomGoalsCompleted}/${totalRoomGoals} <br>
       <br>`;

     goalDiv.appendChild(goalItem)
     weekGoalsCount++;
     totalCompletedGoals += roomGoalsCompleted;
     totalWeekGoals += totalRoomGoals;
   }
}

avgDayGoals = totalCompletedGoals / weekGoalsCount;
goalsPercent = (totalCompletedGoals / totalWeekGoals) * 100;

document.getElementById('avgCompleted').innerHTML = weekGoalsCount == 0 ?
       `0` : `${avgDayGoals.toFixed(2)}`;
document.getElementById('rate').innerHTML = weekGoalsCount == 0 ? `0%` :
       `${goalsPercent.toFixed(2)}%`;
```

```
document.getElementById('total').innerHTML = weekGoalsCount == 0 ? `0/0` :
      `${totalCompletedGoals}/${totalWeekGoals}`;
```

Figure 40: View Study Goals Statistics Code

3. Create Study Room

```
$twilio = new Client($this->sid, $this->token);
   $exists = $twilio->video->v1->rooms->read(['uniqueName' =>
       $roomDetails['roomName']]);

   if(empty($exists)) {
      $newRoom = new Room;
      $user = new User;
      $setTimer = new CustomTimer;
      $meetingInvite = new MeetingInvites;

      $newRoom->create([
         'roomName' => $roomDetails['roomName'],
         'roomType' => ucfirst($roomDetails['roomType']),
         'roomAccess' => $roomDetails['roomAccess'],
         'roomPassword' => $roomDetails['roomPassword'],
         'meetingType' => $roomDetails['meetingType'],
         'meetingSched' => $roomDetails['roomSchedule'],
         'roomTimer' => $roomDetails['roomTimer'],
         'roomMusic' => $roomDetails['roomMusic'],
         'roomSubject' => $roomDetails['roomSubject'],
         'roomCapacity' => $roomDetails['roomType'] == 'group' ? 15 : 1,
         'roomNumOfParticipants' => 0,
         'roomStatus' => $roomDetails['meetingType'] != 'Instant' ?
            'incoming' :  'creating',
         'roomHostId' => Auth::user()->id,
         'roomIsRecording' => 0,
      ]);
```

Figure 41: Create Study Room Code

The creation of the room is done through a form. The host or creator of the room configure the following room details:

- Room Name
- Room Access (Private or Public)
    - Room Password if private
    - Invite Participants if private
- Meeting Type (Instant or Scheduled)
    - Date if scheduled
- Room Timer (Pomodoro, Animedoro, Open)
- Room Subject
- Room Music

Private rooms have password that is sent through an email together with the other room details to the invited participants as the host is allowed to invite participants in a private room.

Scheduled sessions can be set to occur once, daily or weekly.

- Once – session is removed from the scheduled meeting in the calendar once it is done.
- Daily – session is rescheduled to the following day once it is done.
- Weekly – session is rescheduled to the following week once it is done.

Scheduled sessions that are not started by the host gets expired an hour after its supposed starting time.

After successful creation, room will be created with the use of Twilio Programmable Video API. Twilio handles the live video and audio track publications of all users

connected in the room while the other room details and data are being handled separately in our backend.

4. Manage Study Room

1.1 Start/Stop timer

```
public function statusTimer(Request $request, $roomId){
    $roomName = $request->roomName;
    $transformedName = $roomName;
    $isStarted = false;

    if (strpos($roomName, ' ') != false) {
        $transformedName = str_replace(' ', '', $roomName);
    }

    $timer = CustomTimer::where('roomId', $roomId)->first();

    if ($timer->status == 'ongoing') {
        $timer->update([
            'updatedAt' => $request->updatedAt,
        ]);
    }

    if($timer->status == 'stop'){
        $isStarted = true;
        $timer->status = 'ongoing';
    }
    else{
        $isStarted = false;
        $timer->status = 'stop';
    }

    broadcast(new StartTimer($isStarted, $transformedName))->toOthers();

    $timer->save();
}
```

Figure 42: Start/Stop Timer

The timer is divided into work time, short break, and long break. It starts with the work time and once it is finished it will immediately proceed with the break time. The timer alternates between short break and long break every cycle. When the timer is edited, it only takes effect in the next cycle, not during the cycle when it is edited.

Participants of the room will be listening to a broadcast which will be triggered by the host when the start button is pressed – creating a single reference of time which allows the synchronization of timer across all participants in the room.

- End time – work time or break time, in minutes, is added to the date and time when the start button was pressed, and the result will be stored in the endTime column of the database. The end time gets updated every time each work and break time finishes.

1.2 Edit Timer

Figure 41 is the function of updating the state of the timer in the study room. The function updates whenever the timer switches from work time to short break or long break.

```
public function updateTimer(Request $request, $roomId)
{
    $time = CustomTimer::where('roomId', $roomId)->first();
    $time->update([
        'editedWorkTime' => $request->workTime,
        'editedBreakTime' => $request->breakTime,
        'editedLongBreakTime' => $request->longBreakTime,
    ]);
    $time->save();
}
```

Figure 43: Edit Timer Code

1.3 View Study Session Goals

Within the session room, study goals can be added, edited, deleted and marked as complete by the participants. On the other side, the host is able to view all the study goals of each participant including the goal status.

```
    public function getAllGoals($roomName, $roomId) {
        $goalModel = new Goal();
        $identity = Auth::user()->userFName;
        $test = array();
        $twilio = new Client($this->sid, $this->token);

        $participants = $twilio->video->v1->rooms($roomName)
          ->participants->read(['status' => 'connected']);
        foreach ($participants as $participant) {
            $user = $goalModel::getGoalsByIdentity($participant
          ->identity);

            if (sizeof($user) != 0) {
                $g = $goalModel::getGoalsByRoomId($user[0]->userId,
                 $roomId);
                if(!$g->isEmpty()) {
                    array_push($test, $g);
                }
            }
        }
        return view('room.allstudygoals')->with(compact('roomName',
          'test', 'participants'));
    }
```

Figure 44: View Study Session Goals Code

1.4 Kick Participant

Figure 43 is the function on what will happen when the host kicks a participant. The function finds the participant with the same ID and kicks the participant from the study room.

```
public function kickParticipant(Request $request) {
    $roomName = $request->input('roomName');
```

```
    $transformedName = $roomName;

    if (strpos($roomName, ' ') != false) {
        $transformedName = str_replace(' ', '', $roomName);
    }
    $participant = $request->input('participant');

    $room = Room::where('roomName', $roomName)->where('roomStatus',
'ongoing')->first();
    $timer = CustomTimer::where('roomId', $room->roomId)->first();

    broadcast(new ParticipantKickedOut($participant, $transformedName,
$timer))->toOthers();

    return "SUCCESS";
}
```

Figure 45: Kick Participant Code

1.5 Record Study Session

Twilio's Programmable Video API is also being used in recording of sessions. The recorded video contains a gallery view of all the participant's video tracks together with their audio tracks. After the host stops or ends the recording, a redirect URL will be stored in the database which is what the user will use in accessing the recording after the processing is finished – the longer the duration, the longer the processing time will be.

```
    public function recordRoom(Request $request) {
        $twilio = new Client($this->sid, $this->token);
        $room = Room::where('roomName', $request->roomName)
          ->where('roomStatus', '!=', 'completed')->first();

        if ($request->record == 'start') {
            $recording_rules = $twilio->video->v1->rooms($request
          ->roomName)->recordingRules->update(["rules" => [["type" =>
           "include", "all" => true]]]);
            $room->update(['roomIsRecording' => 1]);
            return $recording_rules->roomSid;
        }
```

```
        else {
            $recording_rules = $twilio->video->v1->rooms($request
            ->roomName)->recordingRules->update(["rules" => [["type" =>
             "exclude", "all" => true]]]);

            $composition = $twilio->video->v1->compositions
                ->create($recording_rules->roomSid,
                        [
                            "audioSources" => ["*"],
                            "videoLayout" => [
                                "grid" => [
                                    "video_sources" => [
                                        "*"
                                    ]
                                ]
                            ],
                            "format" => "mp4"
                        ]
                );

            $uri = "https://video.twilio.com/v1/Compositions/$composition
             ->sid/Media?Ttl=3600";
            $room->update(['roomIsRecording' => 0]);
            return $uri;
        }
    }

    public function saveRecording(Request $request) {
        $currentDate = new DateTime('now', new
          DateTimeZone('Asia/Hong_Kong'));

        $newRecording = new RoomRecording;
        $room = Room::where('roomName', $request->roomName)
          ->where('roomStatus', 'ongoing')->first();

        $newRecording->create([
            'recRoomId' => $room->roomId,
            'recLink' => $request->recordLink,
            'dateRecorded' => $currentDate
        ]);
    }
```

Figure 46: Record Study Session Code

1.6 End Study Session

```
public function endMeeting(Request $request) {
    $twilio = new Client($this->sid, $this->token);
    $transformedName = $request->roomName;

    if (strpos($request->roomName, ' ') != false) {
        $transformedName = str_replace(' ', '', $request->roomName);
    }

    $room = Room::where('roomName', $request->roomName)
      ->where('roomStatus', 'ongoing')->first();
    $timer = CustomTimer::where('roomId', $room->roomId)->first();
    $exists = $twilio->video->v1->rooms->read(['uniqueName' => $room
      ->roomName]);

    if ($exists) {
        $twilio->video->v1->rooms($request->roomName)
      ->update("completed");
    }

    if ($room->meetingType == 'Instant' || $room->meetingType ==
      'Scheduled') {
        $room->roomStatus = 'completed';
    } else {
        if ($room->meetingType == 'Daily') {
            $room->meetingSched = Carbon::parse($room->meetingSched)
      ->addDays(1);
        }
        else if ($room->meetingType == 'Weekly') {
            $room->meetingSched = Carbon::parse($room->meetingSched)
      ->addDays(7);
        }
        $room->roomStatus = 'incoming';
    }
    broadcast(new EndSession($transformedName, $timer))->toOthers();

    $room->roomNumOfParticipants = 0;
    $room->save();
}
```

Figure 47: End Study Session Code

Figure 47 show the function of what would happen if the host ends the study session in the study room. The function kicks all participants in the study room and updates and closes the room to prevent the participants from entering the closed study room.

2. Join Study Room

```
$participants = $twilio->video->v1->rooms($roomName)->participants-
>read(['status' => 'connected']);

foreach ($participants as $participant) {
    if ($participant->identity == $identity) {
        $participantExists = true;
        break;
    }
}

if (!empty($room)) {
    $hostDetails = User::find($room->roomHostId);

    if ($room->meetingType != 'Instant') {
        $meetDate = new DateTime($room->meetingSched, new DateTimeZone
         ('Asia/Hong_Kong'));
        $interval = $currentDate->diff($meetDate);
    }
    if($currentUser == $room->roomHostId) {
        $room->roomStatus = 'ongoing';
        $room->save();
    }

    $genre = Genre::where('genreName', $room->roomMusic)->first();
    $musicList = Music::with('getRoomMusic')->where('musicGenreId',
      $genre->genreId)->get();
    $roomTimer = $room->roomTimer;

    if (($room->roomNumOfParticipants < $room->roomCapacity) && $room
      ->roomStatus == 'ongoing') {

        if (!$participantExists) {
            $room->roomNumOfParticipants = count($participants) + 1;
            $room->save();
```

```
        }

        return view('room.room', [
            'accessToken' => $token->toJWT(),
            'roomName' => $roomName,
            'roomId' => $room->roomId,
            'roomMusic' => $room->roomMusic,
            'roomTimer' => $roomTimer,
            'musicList' => $musicList,
            'identity' => $identity,
            'currentUser' => $currentUser,
            'roomHostId' => $room->roomHostId,
            'hostDetails' => $hostDetails,
            'cameraId' => $cameraId,
            'micId' => $micId,
            'roomIsRecording' => $room->roomIsRecording,
        ])->with(compact('goalCollection', 'roomTimer', 'timer'));
    }
    else if ($room->roomStatus == 'incoming') {
        if ($room->roomHostId != $currentUser) {
            return redirect()->back()->with('joinMessage', 'Host hasn\'t
started the meeting yet.');
        }
    }
    else if ($room->roomStatus == 'completed') {
        return redirect()->back()->with('joinMessage', 'Room doesn\'t
exist anymore.');
    }
    else {
        return redirect()->back()->with('joinMessage', 'Room is already
full');
    }
}
else {
    return redirect()->back()->with('joinMessage', 'Room not found or has
expired.');
}
```

Figure 48: Join Study Room Code

Figure 45 is the function in creating a study room. The function checks if the room exists in the database, then checks if the room has users inside the study room. If both conditions has been met, the user will then be redirected to the study

room along with the study room details as well as the participants inside the study room.

3. Change background music/image

```
function changeBimg(imageNum) {
    let imgUrl = "url(/storage/bimg/background/image" + imageNum +
      ".png)";
    document.body.style.backgroundImage = imgUrl;
}
```

Figure 49: Change Background Image Code

```
<div class="img-container">
    <h2 id="musicLabel">{{ $roomMusic }}</h2>
    @foreach ($musicList as $music)
      @if(Storage::disk('public')->exists("bmusic/$music->musicFileName"))
        @if($roomMusic == 'Instrumental' || $roomMusic == 'Classical' ||
$roomMusic == 'White Noise')
          @php
            preg_match('~\-(.*?)\.~', $music->musicFileName, $label);
          @endphp
          <p class="musicLabel">
          <img src="{{ asset('storage/images/'.$label[1].'.png') }}"
class="musicLabelImg">
            {{ucfirst($label[1])}}
          </p>
        @endif
        <audio class="roomMusic" controls loop>
          <source src="{{ asset('storage/bmusic/'.$music->musicFileName)
}}" type="audio/mpeg">
        </audio>
        <br>
      @endif
    @endforeach
</div>
```

Figure 50: Change Background Music Code

Figure 46 and 47 is the function where the user can change the background music/image in the study room. When clicked from the room toolbar the function displays a window pop-up that holds all the background image/music that is

available. Once the user has their desired background image/music it will then be reflected to the study room.

4. Add/Manage Study Session Goals

4.1 Add Study Goal

```
public function storeGoal(Request $request){
    $newGoal = new Goal();
    $identity = Auth::user()->userFName;
    $newGoal->create([
        'setgoal' => $request->setgoal,
        'isCompleted' => 'false',
        'userFname' => $identity,
        'roomId' => $request->roomId,
        'userId' => $request->userId
    ]);
}
```

Figure 51: Add Study Goal Code

Figure 48 is the function in adding a study goal of the participant to the database. The function calls the goal table and stores the input of the participant to the table.

4.2 Edit Study Goals

```
public function editGoal($id){
    $goal = Goal::findOrFail($id);

    return view('editgoal')->with(compact('goal'));
}

public function updateGoal(Request $request, $id){
    $goal = Goal::find($id);
    $goal->setgoal = $request->changeGoal;
    $goal->save();
}
```

Figure 52: Edit Study Goal Code

Figure 49 is the function in editing a study goal of the participant and updates it to database. The function calls the goal table, find the id of the goal that is needed to edit and updates the row of the table with the input of the participant.

4.3 Delete Study Goal

```
public function destroyGoal($id){
    $student = Goal::find($id);
    $student->delete();
    return redirect()->back();
}
```

Figure 53: Delete Study Goal Code

Figure 50 is the function deleting the study goal of the participant. The function find the ID of the study goal that the participant wants to delete study goal and updates the table from the database.

4.4 Complete Study Goal

```
public function complete($id) {
    $goal = Goal::findOrFail($id);

    if ($goal->isCompleted == 'false') {
        $goal->update([
            'isCompleted' => 'true'
        ]);
    }
    else if ($goal->isCompleted == 'true') $goal->update([
        'isCompleted' => 'false'
    ]);

    return $goal->isCompleted;
}
```

Figure 54: Complete Study Goal Code

Figure 51 is the function in updating the status of the study goal of the participant in the study room. The function checks if the status of the study goal is completed or not.

5. Edit User Profile

```
public function editUser($userId) {
    $user = User::findOrFail($userId);
    $jsonResult = json_encode($user);

    return $jsonResult;
}

public function updateUser(Request $request, $userId) {
    $userDetails = $request->userData;
    $user = User::findOrFail($userId);

    $request->validate(
        [
            'firstName' => [Rule::requiredIf(function() use ($request) {
                return is_null($request->get('userData')['firstName']);
            })],
            'lastName' => [Rule::requiredIf(function() use ($request) {
                return is_null($request->get('userData')['lastName']);
            })],
            'emailAdd' => [Rule::requiredIf(function() use ($request) {
                    return is_null($request->get('userData')['emailAdd']);
            })],
        ],
        [
            'firstName.required' => 'Please provide a first name.',
            'lastName.required' => 'Please provide a last name.',
            'emailAdd.required' => 'Please provide an email.',
        ],
    );

    try {
        $result = $user->update([
            'userFName' => $userDetails['firstName'],
            'userLName' => $userDetails['lastName'],
```

```
            'email' => $userDetails['emailAdd'],
            'prefSubject' => $userDetails['prefSubject'],
            'prefMusic' => $userDetails['prefMusic'],
        ]);
        return $result;
    } catch (QueryException $e) {
        $errorCode = $e->errorInfo[1];
        if ($errorCode == 1062) {
            $errors = new MessageBag(['emailAdd' => ['This email is
already taken. Please choose another one.']]);
            return response()->json(['errors' => $errors], 422);
        }
    }
}
```

Figure 55: User Edit Profile Code

Figure 52 is the function of editing the user profile in the database. This function validates first the user input and checks if the user is in the database before updating the user profile.

6. Reset Password

```
public function resetPassword(Request $request, $userId) {
    $oldPassword = $request->oldPassword;
    $newPassword = $request->newPassword;

    $user = User::findOrFail($userId);
    $oldPasswordFromDB = User::getPasswordById($userId);

    $request->validate(
        [
            'oldPassword' => [Rule::requiredIf(function() use
($oldPassword) {
                return is_null($oldPassword);
            })],
            'newPassword' => [Rule::requiredIf(function() use
($newPassword) {
                return is_null($newPassword);
            })],
        ],
        [
            'oldPassword.required' => 'Please enter your old password.',
```

```
            'newPassword.required' => 'Please provide a new password.'
        ]
    );
    if(!Hash::check($oldPassword, $oldPasswordFromDB)) {
        $error = new MessageBag(['oldPassword' => ['The old password you
provided is incorrect.']]);
        return response()->json(['errors' => $error], 422);
    }

    $result = $user->update([
        "password" => Hash::make($newPassword)
    ]);

    return $result;
}
```

Figure 56: Reset Password Code

Figure 53 is the function of resetting the user's password and updating it to the database. The function validates if the old password is the same in the database before getting the new password and updating it to the database.

7. Manage User

7.1 Create User

```
public function createUser(Request $request) {
    $userDetails = $request->userData;
    $accountPassword =
$userDetails['lastName'].$userDetails['firstName'];
    $accountPassword = lcfirst(str_replace(' ', '',
ucwords($accountPassword)));
    $accountPassword .= date('Y');
    $userDetails['password'] = $accountPassword;

    $request->validate(
        [
            'firstName' => [Rule::requiredIf(function() use ($request) {
                return is_null($request->get('userData')['firstName']);
```

```
            })],
            'lastName' => [Rule::requiredIf(function() use ($request) {
                return is_null($request->get('userData')['lastName']);
            })],
            'emailAdd' => [
                Rule::requiredIf(function() use ($request) {
                    return is_null($request-
>get('userData')['emailAdd']);
                }),
            ],
        ],
        [
            'firstName.required' => 'Please provide a first name.',
            'lastName.required' => 'Please provide a last name.',
            'emailAdd.required' => 'Please provide an email.',
        ]
    );

    $newUser = new User;

    try {
        $newUser->create([
            'role' => $userDetails['role'],
            'userFName' => $userDetails['firstName'],
            'userLName' => $userDetails['lastName'],
            'email' => $userDetails['emailAdd'],
            'password' => Hash::make($accountPassword),
        ]);

        $this->sendEmail($userDetails);
    } catch (QueryException $e) {
        $errorCode = $e->errorInfo[1];
        if ($errorCode == 1062) {
            $errors = new MessageBag(['emailAdd' => ['This email is
already taken. Please choose another one.']]);
            return response()->json(['errors' => $errors], 422);
        }
    }
}
```

Figure 57: Create User Code

7.2 Edit User

```
public function updateUser(Request $request, $userId) {
    $userDetails = $request->userData;
    $user = User::findOrFail($userId);

    $request->validate(
        [
            'firstName' => [Rule::requiredIf(function() use ($request) {
                return is_null($request->get('userData')['firstName']);
            })],
            'lastName' => [Rule::requiredIf(function() use ($request) {
                return is_null($request->get('userData')['lastName']);
            })],
            'emailAdd' => [
                Rule::requiredIf(function() use ($request) {
                    return is_null($request-
>get('userData')['emailAdd']);
                }),
            ]
        ],
        [
            'firstName.required' => 'Please provide a first name.',
            'lastName.required' => 'Please provide a last name.',
            'emailAdd.required' => 'Please provide an email.',
        ]
    );

    try {
        $result = $user->update([
            'userFName' => $userDetails['firstName'],
            'userLName' => $userDetails['lastName'],
            'email' => $userDetails['emailAdd'],
            'role' => $userDetails['role'],
        ]);
        return $result;
    } catch (QueryException $e) {
        $errorCode = $e->errorInfo[1];
        if ($errorCode == 1062) {
            $errors = new MessageBag(['emailAdd' => ['This email is
already taken. Please choose another one.']]);
            return response()->json(['errors' => $errors], 422);
        }
    }
}
```

Figure 58: Admin Edit User Code

7.3 Delete User

```
public function deleteUser($userId) {
    $user = User::findOrFail($userId);
    $user->delete();
}
```

Figure 59: Delete User Code

Figure 54, 55, and 56 are for the admins to create, edit or delete user in the database from the admin dashboard. For creating a user, the function validates the admin input before storing it to the database. For editing a user, the function finds the specific user in the database before validating the admin input for editing, after that it is then updated to the database. For deleting a user, the function asks for confirmation for deleting the specific user before finding the user in the database and delete it.

8. Leave Study Room

```
public function leaveRoom(Request $request) {
    $twilio = new Client($this->sid, $this->token);
    $participants = $twilio->video->v1->rooms($request->roomName)-
>participants->read(['status' => 'connected']);

    $room = Room::where('roomName', $request->roomName)-
>where('roomStatus', 'ongoing')->first();
    $room->roomNumOfParticipants = count($participants) - 1;

    if ($room->roomNumOfParticipants == 0) {
        $exists = $twilio->video->v1->rooms->read(['uniqueName' => $room-
>roomName, 'status' => 'in-progress']);

        if ($exists) {
            $twilio->video->v1->rooms($room->roomName)-
>update("completed");
        }

        if ($room->meetingType == 'Instant' || $room->meetingType ==
'Scheduled') {
            $room->roomStatus = 'completed';
        } else {
```

```
            if ($room->meetingType == 'Daily') {
                $room->meetingSched = Carbon::parse($room->meetingSched)-
>addDays(1);
            }
            else if ($room->meetingType == 'Weekly') {
                $room->meetingSched = Carbon::parse($room->meetingSched)-
>addDays(7);
            }
            $this->updateRecurSessTimer($room->roomTimer, $room->roomId);
            $room->roomStatus = 'incoming';
        }
    }

    $room->save();
}
```

Figure 60: Leave Study Room Code

Figure 60 shows the function of what would happen when a participant leaves the study room. The function updates the number of participants inside the study room. If there no more participants inside the room, the function updates the database that the room is completed and will not allow any users to enter the room again.

Testing Process

This section covers the results of the integration testing done on the system during the development. The tables represent the test cases performed on the different use cases that are defined in the application along with its description, test data, expected result, actual result. The status shows whether the test case passed or failed. This testing process is performed to ensure that the application or system is working as expected and its requirements are met.

Test Case 01: Create Study Room

Use Case	Create Study Room

Description	Enter valid room name
Test Data	Room Name: Science Room
Expected Result	Room is created with the name Science Room
Actual Result	Room is created with the name Science Room
Status	Passed

Table 20: Create Study Room Test Case 1

Test Case 02: Create Study Room

Use Case	Create Study Room
Description	Enter empty room name
Test Data	Room Name:
Expected Result	Validation message is displayed
Actual Result	Validation message is displayed
Status	Passed

Table 21: Create Study Room Test Case 2

Test Case 03: Create Study Room

Use Case	Create Study Room
Description	Enter room name that is currently in-progress
Test Data	Room Name: Science Room
Expected Result	Validation message is displayed
Actual Result	Validation message is displayed
Status	Passed

Table 22: Create Study Room Test Case 3

Test Case 04: Create Study Room

Use Case	Create Study Room
Description	Display room password field
Test Data	User selects private room access option
Expected Result	Room password field is displayed
Actual Result	Room password field is displayed
Status	Passed

Table 23: Create Study Room Test Case 4

Test Case 05: Create Study Room

Use Case	Create Study Room
Description	Enter valid room password
Test Data	Password: P@ssword123
Expected Result	Room is created with password: P@ssword123
Actual Result	Room is created with password: P@ssword123
Status	Passed

Table 24: Create Study Room Test Case 5

Test Case 06: Create Study Room

Use Case	Create Study Room
Description	Enter empty room password
Test Data	Password:
Expected Result	Validation message is displayed
Actual Result	Validation message is displayed

Status	Passed

Table 25: Create Study Room Test Case 6

Test Case 07: Create Study Room

Use Case	Create Study Room
Description	Display schedule field
Test Data	User selects scheduled session option
Expected Result	Schedule field is displayed
Actual Result	Schedule field is displayed
Status	Passed

Table 26: Create Study Room Test Case 7

Test Case 08: Create Study Room

Use Case	Create Study Room
Description	Select valid date from date picker
Test Data	Schedule: 01/05/2023 04:00 am
Expected Result	Scheduled room is created with schedule 01/05/2023 04:00 am
Actual Result	Scheduled room is created with schedule 01/05/2023 04:00 am
Status	Passed

Table 27: Create Study Room Test Case 8

Test Case 09: Create Study Room

Use Case	Create Study Room
Description	No selected date
Test Data	Schedule: dd/mm/yyyy --:-- --

Expected Result	Validation message is displayed
Actual Result	Validation message id displayed
Status	Passed

Table 28: Create Study Room Test Case 9

Test Case 10: Create Study Room

Use Case	Create Study Room
Description	Display invite participants field
Test Data	User selects private room access or scheduled option
Expected Result	Invite participants field is displayed
Actual Result	Invite participants field is displayed
Status	Passed

Table 29: Create Study Room Test Case 10

Test Case 11: Create Study Room

Use Case	Create Study Room
Description	Add valid email address for invite participants
Test Data	Invited User: kshow123@gmail.com
Expected Result	User will be invited to the scheduled session
Actual Result	User will be invited to the scheduled session
Status	Passed

Table 30: Create Study Room Test Case 11

Test Case 12: Create Study Room

Use Case	Create Study Room

Description	Add invalid email address for invite participants
Test Data	Invited User: jlapiz123
Expected Result	Validation message will be displayed
Actual Result	Room is created but no user will be invited to the scheduled session
Status	Failed

Table 31: Create Study Room Test Case 12

Test Case 13: Create Study Room

Use Case	Create Study Room
Description	Add repeating email address for invite participants
Test Data	Invited User: kshow123@gmail.com kshow123@gmail.com
Expected Result	Validation message will be displayed
Actual Result	Room is created but invitedUsers array is filtered out by removing duplicate emails
Status	Failed

Table 32: Create Study Room Test Case 13

Test Case 14: Create Study Room

Use Case	Create Study Room
Description	Create instant study session
Test Data	User selects instant session type option
Expected Result	User will be redirected to the newly created room
Actual Result	User will be redirected to the newly created room
Status	Passed

Table 33: Create Study Room Test Case 14

Test Case 15: Create Study Room

Use Case	Create Study Room
Description	Create scheduled study session
Test Data	User selects scheduled session type option and sets date and time
Expected Result	Session will be created Appear in scheduled sessions section
Actual Result	Session will be created Appear in scheduled sessions section
Status	Passed

Table 34: Create Study Room Test Case 15

Test Case 16: Manage Study Room

Use Case	Manage Study Room
Description	View study room page
Test Data	User navigates to study room page
Expected Result	Available rooms will be displayed if there is one, otherwise empty display
Actual Result	Available rooms will be displayed if there is one, otherwise empty display
Status	Passed

Table 35: Manage Study Room Test Case 1

Test Case 17: Edit Timer

Use Case	Edit Timer
Description	Update work time, break time or long break time with valid data
Test Data	Work Time: 15 Break Time: 5 Long Break Time: 8
Expected Result	Timer data will be updated in the database

	Edited timer takes effect in the next cycle
Actual Result	Timer data will be updated in the database Edited timer takes effect in the next cycle
Status	Passed

Table 36: Edit Timer Test Case 1

Test Case 18: Edit Timer

Use Case	Edit Timer
Description	Update work time, break time or long break time with 0 or less
Test Data	Work Time: 0 Break Time: -1 Long Break Time: 0
Expected Result	Validation message will be displayed
Actual Result	Validation message will be displayed
Status	Passed

Table 37: Edit Timer Test Case 2

Test Case 19: Start/Stop Timer

Use Case	Start/Stop Timer
Description	Start Timer
Test Data	User clicks start timer button
Expected Result	Timer will start and end time will be stored in the database
Actual Result	Timer will start and end time will be stored in the database
Status	Passed

Table 38: Start/Stop Timer Test Case 1

Test Case 20: Start/Stop Timer

Use Case	Start/Stop Timer

Description	Switch mode to break time
Test Data	User waits for work time to finish
Expected Result	Timer duration will be set to break time value endTime and studyDuration are updated in the database
Actual Result	Timer duration will be set to break time value endTime and studyDuration are updated in the database
Status	Passed

Table 39: Start/Stop Timer Test Case 2

Test Case 21: Start/Stop Timer

Use Case	Start/Stop Timer
Description	Switch mode to work time
Test Data	User waits for break time to finish
Expected Result	Timer duration will be set to work time value endTime is updated in the database
Actual Result	Timer duration will be set to work time value endTime is updated in the database
Status	Passed

Table 40: Start/Stop Timer Test Case 3

Test Case 22: Start/Stop Timer

Use Case	Start/Stop Timer
Description	Stop Timer
Test Data	User clicks stop timer button
Expected Result	Timer duration will be paused endTime, updatedAt and studyDuration are updated in the database
Actual Result	Timer duration will be paused endTime, updatedAt and studyDuration are updated in the database
Status	Passed

Table 41: Start/Stop Timer Test Case 4

Test Case 23: Record Session

Use Case	Record Session
Description	Start recording
Test Data	User clicks recording button in the toolbar and clicks yes to start recording
Expected Result	Recording starts
Actual Result	Recording starts
Status	Passed

Table 42: Record Session Test Case 1

Test Case 24: Record Session

Use Case	Record Session
Description	Stop recording
Test Data	User clicks recording button in the toolbar and clicks yes to stop recording
Expected Result	Recording stops Recording data containing the link is stored in the database
Actual Result	Recording stops Recording data containing the link is stored in the database
Status	Passed

Table 43: Record Session Test Case 2

Test Case 25: View Study Session Goals

Use Case	View Study Session Goals
Description	View study session goals of room participants
Test Data	User clicks view all study goals button in toolbar
Expected Result	New window opens and user is redirected

	Study session goals are displayed if there is any, otherwise empty
Actual Result	New window opens and user is redirected Study session goals are displayed if there is any, otherwise empty
Status	Passed

Table 44: View Study Session Goals Test Case 1

Test Case 26: Kick Participant

Use Case	Kick Participant
Description	User kicks participant
Test Data	User clicks kick button under the selected participant and clicks yes to kick
Expected Result	Kicked participant will be redirected to the study room page Study duration is updated in the database
Actual Result	Kicked participant will be redirected to the study room page Study duration is updated in the database
Status	Passed

Table 45: Kick Participant Test Case 1

Test Case 27: View Recordings

Use Case	View Recordings
Description	View study session recording
Test Data	User clicks a link from the recording list section
Expected Result	If recording is still being processed, pop-up message will appear, otherwise recording is downloaded in local machine
Actual Result	If recording is still being processed, pop-up message will appear, otherwise recording is downloaded in local machine
Status	Passed

Table 46: View Recordings Test Case 1

Test Case 28: Add/Manage Tasks

Use Case	Add/Manage Tasks
Description	Add task without due date
Test Data	Task Name: Finish Math Project Due Date:
Expected Result	Task is added in the task list section
Actual Result	Task is added in the task list section
Status	Passed

Table 47: Add/Manage Tasks Test Case 1

Test Case 29: Add/Manage Tasks

Use Case	Add/Manage Tasks
Description	Add task with due date
Test Data	Task Name: Capstone Defense Due Date: May 02, 2023
Expected Result	Task is added in the task list section as well as calendar
Actual Result	Task is added in the task list section as well as calendar
Status	Passed

Table 48: Add/Manage Tasks Test Case 2

Test Case 30: Add/Manage Tasks

Use Case	Add/Manage Tasks
Description	Add task with empty data
Test Data	Task Name: Due Date:
Expected Result	Validation message is displayed
Actual Result	Validation message is displayed
Status	Passed

Table 49: Add/Manage Tasks Test Case 3

Test Case 31: Add/Manage Tasks

Use Case	Add/Manage Tasks
Description	Edit a task (adding due date)
Test Data	Task Name: Task Edited Due Date: May 03, 2023
Expected Result	Task is updated and added in the calendar If already has a due date, move to new date in the calendar
Actual Result	Task is updated and added in the calendar If already has a due date, move to new date in the calendar
Status	Passed

Table 50: Add/Manage Tasks Test Case 4

Test Case 32: Add/Manage Tasks

Use Case	Add/Manage Tasks
Description	Edit a task (removing due date)
Test Data	Task Name: Task Edited Due Date:
Expected Result	Task is updated and removed from the calendar
Actual Result	Task is updated and removed from the calendar
Status	Passed

Table 51: Add/Manage Tasks Test Case 5

Test Case 33: Add/Manage Tasks

Use Case	Add/Manage Tasks
Description	Delete a task
Test Data	User clicks delete button and yes to confirm deletion
Expected Result	Task is removed from the task list section If has a due date, remove from the calendar

Actual Result	Task is removed from the task list section If has a due date, remove from the calendar
Status	Passed

Table 52: Add/Manage Tasks Test Case 6

Test Case 34: Reset Password

Use Case	Reset Password
Description	Reset password (user-side) with valid data
Test Data	Old Password: P@ssword123 New Password: newP@ssword123
Expected Result	Password is updated
Actual Result	Password is updated
Status	Passed

Table 53: Reset Password Test Case 1

Test Case 35: Reset Password

Use Case	Reset Password
Description	Reset Password with invalid data
Test Data	Old Password (user-side): New Password:
Expected Result	Validation message is displayed
Actual Result	Validation message is displayed
Status	Passed

Table 54: Reset Password Test Case 2

Test Case 36: Reset Password

Use Case	Reset Password
Description	Reset Password (user-side) with incorrect old password

Test Data	Old Password: wrongPassword New Password: newPassword
Expected Result	Validation message is displayed
Actual Result	Validation message is displayed
Status	Passed

Table 55: Reset Password Test Case 3

Test Case 37: Reset Password

Use Case	Reset Password
Description	Reset Password (admin-side) with valid data
Test Data	New Password: newP@ssword123
Expected Result	Password is updated
Actual Result	Password is updated
Status	Passed

Table 56: Reset Password Test Case 4

Test Case 38: Create Users

Use Case	Create Users
Description	Create user with valid data
Test Data	First Name: Jaime Last Name: Lapiz Email Address: jaimelapiz@gmail.com
Expected Result	User is created and an email is sent containing account details including the auto-generated password
Actual Result	User is created and an email is sent containing account details including the auto-generated password
Status	Passed

Table 57: Create Users Test Case 1

Test Case 39: Create Users

Use Case	Create Users
Description	Create user with invalid data
Test Data	First Name: Last Name: Email Address:
Expected Result	Validation message is displayed
Actual Result	Validation message is displayed
Status	Passed

Table 58: Create Users Test Case 2

Test Case 40: Create Users

Use Case	Create Users
Description	Create user with existing email address
Test Data	First Name: Jaime Last Name: Lapiz Email Address: existing@gmail.com
Expected Result	Validation message is displayed
Actual Result	Validation message is displayed
Status	Passed

Table 59: Create Users Test Case 3

Test Case 41: Manage Users

Use Case	Manage Users
Description	Edit user details with valid data
Test Data	First Name: Jaime Edited Last Name: Lapiz Edited Email Address: edited@gmail.com
Expected Result	Account details are updated

Actual Result	Account details are updated
Status	Passed

Table 60: Manage Users Test Case 1

Test Case 42: Manage Users

Use Case	Manage Users
Description	Edit user details with invalid data
Test Data	First Name: Last Name: Email Address:
Expected Result	Validation message is displayed
Actual Result	Validation message is displayed
Status	Passed

Table 61: Manage Users Test Case 2

Test Case 43: Manage Users

Use Case	Manage Users
Description	Edit user details with existing email address
Test Data	First Name: Jaime Last Name: Lapiz Email Address: existing@gmail.com
Expected Result	Validation message is displayed
Actual Result	Validation message is displayed
Status	Passed

Table 62: Manage Users Test Case 3

Test Case 44: Manage Users

Use Case	Manage Users
Description	Delete user
Test Data	User clicks the delete button and yes to confirm deletion
Expected Result	User data is deleted
Actual Result	User data is deleted
Status	Passed

Table 63: Manage Users Test Case 4

Test Case 45: Edit User Profile

Use Case	Edit User Profile
Description	Edit user profile with valid data
Test Data	First Name: Jaime Edited Last Name: Lapiz Edited Email Address: edited@gmail.com
Expected Result	User profile is updated
Actual Result	User profile is updated
Status	Passed

Table 64: Edit User Profile Test Case 1

Test Case 46: Edit User Profile

Use Case	Edit User Profile
Description	Edit user profile with invalid data
Test Data	First Name: Last Name: Email Address
Expected Result	Validation message is displayed
Actual Result	Validation message is displayed

Status	Passed

Table 65: Edit User Profile Test Case 2

Test Case 47: Edit User Profile

Use Case	Edit User Profile
Description	Edit user profile with existing email address
Test Data	First Name: Jaime Last Name: Lapiz Email Address: existing@gmail.com
Expected Result	Validation message is displayed
Actual Result	Validation message is displayed
Status	Passed

Table 66: Edit User Profile Test Case 3

Test Case 48: Join Study Room

Use Case	Join Study Room
Description	Select valid media devices
Test Data	User selects camera and mic devices
Expected Result	User is redirected to the study room If private, room password is required before redirecting to the study room
Actual Result	User is redirected to the study room If private, room password is required before redirecting to the study room
Status	Passed

Table 67: Join Study Room Test Case 1

Test Case 49: Join Study Room

Use Case	Join Study Room
Description	Select invalid media devices

Test Data	User selects camera and mic devices
Expected Result	Validation message is displayed
Actual Result	Validation message is displayed
Status	Passed

Table 68: Join Study Room Test Case 2

Test Case 50: Join Study Room

Use Case	Join Study Room
Description	Join public study room
Test Data	User clicks join study room button
Expected Result	User is redirected to the study room If study room capacity is full, redirected to study room page
Actual Result	User is redirected to the study room If study room capacity is full, redirected to study room page
Status	Passed

Table 69: Join Study Room Test Case 3

Test Case 51: Join Study Room

Use Case	Join Study Room
Description	Join private study room with valid password
Test Data	Password: password
Expected Result	If correct, user is redirected to the study room, otherwise asked to input again If room capacity is full, redirected to study room page
Actual Result	If correct, user is redirected to the study room, otherwise asked to input again If room capacity is full, redirected to study room page
Status	Passed

Table 70: Join Study Room Test Case 4

Test Case 52: Join Study Room

Use Case	Join Study Room
Description	Join private study room with invalid password
Test Data	Password:
Expected Result	Validation message is displayed
Actual Result	Validation message is displayed
Status	Passed

Table 71: Join Study Room Test Case 5

Test Case 53: Join Study Room

Use Case	Join Study Room
Description	Join study room that just got completed but study room page is not refreshed yet
Test Data	User clicks the join button
Expected Result	User is redirected to study room page with message “Room not found or has expired”
Actual Result	User is redirected to study room page with message “Room not found or has expired”
Status	Passed

Table 72: Join Study Room Test Case 6

Test Case 54: Join Study Room

Use Case	Join Study Room
Description	Join scheduled study session that isn’t started by the host yet
Test Data	User clicks the join scheduled study session button
Expected Result	Validation message is displayed

Actual Result	Validation message is displayed
Status	Passed

Table 73: Join Study Room Test Case 7

Test Case 55: Join Study Room

Use Case	Join Study Room
Description	Join scheduled study session where date and time matches or exceeds the current date and time but no more than 1 hour
Test Data	User clicks the join scheduled study session button
Expected Result	If host, study session is started and user is redirected to the study room If participant, redirected to the study room
Actual Result	If host, study session is started and user is redirected to the study room If participant, redirected to the study room
Status	Passed

Table 74: Join Study Room Test Case 8

Test Case 56: Join Study Room

Use Case	Join Study Room
Description	Join scheduled study session exceeding 1 hour its schedule and is not started yet
Test Data	User clicks the join scheduled study session button
Expected Result	User is redirected back with message “Room not found or has expired”
Actual Result	User is redirected back with message “Room not found or has expired”
Status	Passed

Table 75: Join Study Room Test Case 9

Test Case 57: Join Study Room

Use Case	Join Study Room

Description	Join study room where user is part of it already
Test Data	User clicks the join study room button
Expected Result	User is redirected back with message “You’re already inside the room”
Actual Result	User is redirected back with message “You’re already inside the room”
Status	Passed

Table 76: Join Study Room Test Case 10

Test Case 58: Add/Manage Study Session Goals

Use Case	Add/Manage Study Session Goals
Description	Add study session goal with valid data
Test Data	Goal: Introduction to Aerodynamics
Expected Result	Study session goal is displayed in the goal list section
Actual Result	Study session goal is displayed in the goal list section
Status	Passed

Table 77: Add/Manage Study Session Goals Test Case 1

Test Case 59: Add/Manage Study Session Goals

Use Case	Add/Manage Study Session Goals
Description	Add study session goal with invalid data
Test Data	Goal:
Expected Result	Validation message is displayed
Actual Result	Validation message is displayed
Status	Passed

Table 78: Add/Manage Study Session Goals Test Case 2

Test Case 60: Add/Manage Study Session Goals

Use Case	Add/Manage Study Session Goals
Description	Edit study session goal with valid data
Test Data	Goal: Edited Goal
Expected Result	Goal is updated
Actual Result	Goal is updated
Status	Passed

Table 79: Add/Manage Study Session Goals Test Case 3

Test Case 61: Add/Manage Study Session Goals

Use Case	Add/Manage Study Session Goals
Description	Edit study session goal with invalid data
Test Data	Goal:
Expected Result	Validation message is displayed
Actual Result	Validation message is displayed
Status	Passed

Table 80: Add/Manage Study Session Goals Test Case 4

Test Case 62: Add/Manage Study Session Goals

Use Case	Add/Manage Study Session Goals
Description	Complete study goal
Test Data	User clicks the checkbox beside the goal
Expected Result	Goal complete status is updated Goal name is striked-through Goal checkbox is checked

Actual Result	Goal complete status is updated Goal name is striked-through Goal checkbox is checked
Status	Passed

Table 81: Add/Manage Study Session Goals Test Case 5

Test Case 63: Add/Manage Study Session Goals

Use Case	Add/Manage Study Session Goals
Description	Unmark study goal as complete
Test Data	User clicks the checkbox beside the goal
Expected Result	Goal complete status is updated Goal name's strike-through is removed Goal checkbox' check is removed
Actual Result	Goal complete status is updated Goal name's strike-through is removed Goal checkbox' check is removed
Status	Passed

Table 82: Add/Manage Study Session Goals Test Case 6

Test Case 64: Add/Manage Study Session Goals

Use Case	Add/Manage Study Session Goals
Description	Delete study session goal
Test Data	User clicks delete button beside the goal
Expected Result	Goal is deleted and removed from goal list section
Actual Result	Goal is deleted and removed from goal list section
Status	Passed

Table 83: Add/Manage Study Session Goals Test Case 7

Test Case 65: Change Background Image/Music

Use Case	Change Background Image
Description	Change study room's background image
Test Data	User clicks background image button in toolbar and selects from any of the provided images
Expected Result	Selected image is rendered as the background image of the study room
Actual Result	Selected image is rendered as the background image of the study room
Status	Passed

Table 84: Change Background Image/Music Test Case 1

Test Case 66: Change Background Image/Music

Use Case	Change Background Music
Description	Change study room's background music
Test Data	User clicks on the background music button in toolbar and plays from any of the music provided
Expected Result	Selected background music is played as the background music of the study room
Actual Result	Selected background music is played as the background music of the study room
Status	Passed

Table 85: Change Background Image/Music Test Case 2

Test Case 67: View Study Session Statistics

Use Case	View Study Session Statistics
Description	View study session statistics
Test Data	User navigates to the profile page
Expected Result	Study session statistics is displayed If student, chart with weekly and monthly is displayed
Actual Result	Study session statistics is displayed If student, chart with weekly and monthly is displayed
Status	Passed

Table 86: View Study Session Statistics Test Case 1

Test Case 68: View Study Session Statistics

Use Case	View Study Session Statistics
Description	View weekly study session statistics (student-side)
Test Data	User clicks weekly button
Expected Result	Study session statistics filtered and calculated by week Can navigate by week
Actual Result	Study session statistics filtered and calculated by week Can navigate by week
Status	Passed

Table 87: View Study Session Statistics Test Case 2

Test Case 69: View Study Session Statistics

Use Case	View Study Session Statistics
Description	View monthly study session statistics (student-side)
Test Data	User clicks monthly button
Expected Result	Study session statistics filtered and calculated by month Can navigate by month
Actual Result	Study session statistics filtered and calculated by month Can navigate by month
Status	Passed

Table 88: View Study Session Statistics Test Case 3

Test Case 70: Leave Study Room

Use Case	Leave Study Room
Description	Leave study room during work time
Test Data	User clicks the leave button in the toolbar
Expected Result	User is redirected to study room page Study duration is updated in the database

Actual Result	User is redirected to the study room page Study duration is updated in the database
Status	Passed

Table 89: Leave Study Room Test Case 1

Test Case 71: Leave Study Room

Use Case	Leave Study Room
Description	Leave study room during break time
Test Data	User clicks the leave button in the toolbar
Expected Result	User is redirected to the study room page
Actual Result	User is redirected to the study room page
Status	Passed

Table 90: Leave Study Room Test Case 2

Test Case 72: End Study Session

Use Case	End Study Session
Description	Host ends the study session during work time
Test Data	User clicks the end study session button in toolbar
Expected Result	Users are redirected to the study room page Study duration is updated in the database
Actual Result	Users are redirected to the study room page Study duration is updated in the database
Status	Passed

Table 91: End Study Session Room Test Case 1

Test Case 73: End Study Session

Use Case	End Study Session
Description	Host ends the study session during break time
Test Data	User clicks the end study session button in toolbar
Expected Result	Users are redirected to the study room page
Actual Result	Users are redirected to the study room page
Status	Passed

Table 92: End Study Session Test Case 2

CHAPTER IV

SUMMARY, CONCLUSION, AND RECOMMENDATIONS

SUMMARY OF FINDINGS

The researchers have implemented the application following the project objectives. The application has been implemented and worked as intended. From Login, To Do List,

Calendar, Study Room, and Statistics, all the major features were simulated and confirmed to be working.

According to the test cases, almost all the expected results were met considering the different test data. Although there were a few test cases that were not working as expected, but it is not considered a major issue that can break the application since the feature is still working. Only that some validation message is not displayed on the front-end, but the validation was handled properly on the back-end.

During the debugging process, it was found out that the response rate of the timer inside the study room depends on the internet connection and the device where the application is accessed. A few seconds delay was noticed whenever the timer has been started or stopped. This delay may affect the execution of the functions for the timer, and may greatly affect the recording of the statistics. Therefore, to have a much faster response, the application must be accessed using a device with good internet connection.

Also, at the start of the project conceptualization, the researchers initially planned to incorporate website blocking feature in the application. However, after several research and multiple failed attempts of implementing, it was found out that website blocking feature is not applicable to the Laravel-based application due to browser restrictions and privacy policy. There were some solutions such as implementing middleware inside the application to interfere with user requests and identify if the user accessed a link that is blocked. However, it is limited within the application only. Therefore, users can still access other websites outside the application (i.e., entered URL directly on the address bar, accessed on a new tab, browser, etc.). Another possible solution is to utilize a browser extension to detect websites that the user accessed, but it is also out of the researchers'

scope. With the same reason of browser restrictions and privacy policy, the Laravel-based application would not be able to detect whether the extension was installed by the user.

CONCLUSION

In this study, the researchers have identified the issue affecting online learners' focus and also identified the factors that can contribute to their productivity. Having this in mind, Studdo, a web application has been implemented to address the issue and offer a platform for the learners solely for enhancing productivity incorporated with the concepts of Pomodoro Technique, virtual study room, study ambience, and task management. The application also includes room host that monitors the participants inside the study room.

After all the development process, it has been concluded that an application implemented as purely Laravel-based has some limitations. Without utilizing other frameworks, some features can be difficult to implement. Aside from that, it was also concluded that the application must be accessed using a device with good internet connection to achieve a better response rate.

The researchers have concluded that Studdo is more preferable to be used by educational institutions especially those that are still doing online classes. Teachers will serve as the room host that will monitor their students as the participants inside the study room.

RECOMMENDATIONS

Due to the limitations of the currently developed website application, the researchers would recommend to explore more widely used frameworks (e.g., React.js, Vue.js, etc.) to be used for implementing the features instead of Laravel only. Also, it is a good option to have the application be implemented as desktop software to further configure the system that best addresses the project objectives defined and possibly the initial plan of implementing website blocking feature.

The implemented application also lacks input from actual teachers and students aside from the researchers and those involved in the development. Thus, it is recommended to gather more inputs and have other teachers and students test the application to really determine if the application did increase their productivity and improve their study habits.

REFERENCES

Baker, J. (2020, April 11). Twice as many students distracted at home as at school, New Data Shows. *The Sydney Morning Herald*. https://www.smh.com.au/national/twice-as-many-students-distracted-at-home-as-at-school-new-data-shows-20200410-p54iu7.html

Barker, C. (2008). “Teacher, producer create CD for math success”, New York Amsterdam News 99, 16-32.

Cirillo, F. (2006). The pomodoro technique (the pomodoro).

Dalisay, J., & Montemayor, T. (2021). The influence of online learning towards the attention span and motivation of college students

Dosseville, F., Laborde, S., Scelles, N. (2012). Music during lecture: will students learn better? *Learning and Individual Differences*, 22(2), 258-262. https://doi.org/10.1016/j.lindif.2011.10.004

Lesiuk, T. (2005). The effect of music listening on work performance. *Psychology of Music,* 33(2), 173-191. https://doi.org/10.1177/0305735605050650.

McCoy, B. (2016). Digital distractions in the classroom phase II: student classroom use of digital devices for non-class related purposes.

Monks, M., Cutler, C. (2018). Could productivity apps combat students’ social media addiction in order to maximize efficiency? [version 1].

Porter, L. (2002). Cognitive skills. In L. Porter (Ed.), Educating young children with special needs (pp. 191-209). Crows Nest, Australia: Allen & Unwin.

Singer, M. J. (2008). “Accessing the Musical Intelligence in Early Childhood Education”, Australian Journal of Early Childhood 33, no. 2: 49-56.

Smith, S. W. (2002). Applying cognitive-behavioral techniques to social skills instruction. ERIC/OSEP digest. Arlington, VA: ERIC Clearinghouse on Disabilities and Gifted Education.

Jin, L. (2022). Virtual Study room on social media “Study with me” videos on Bilibili among Chinese university students.