The Frog Clock: A Combination of Pomodoro Clock & Task Tracker

[1]Zhe Cao, [1]Simiao Wang, [1]Wenyu Yang

[1]Information Systems, Northeastern University

{cao.zhe1, wang.simi, yang.wenyu}@northeastern.edu

Abstract: **We developed a new version of a timer that combines the functions of the Pomodoro Technique, customization of settings, statistics of activities, checklists, and a reward system. This product is designed to help users focus on one task at a time and establish good habits, which makes it have a wide range of target users from students, professionals, freelancers, to hobbyists. We used object-oriented design in the forms of the Model-View-Controller (MVC) pattern, interface, abstract class, and inheritance. Our product passed user testing with positive feedback and suggestions from them.**

Keywords: ***Java, JavaFx, MVC pattern, Pomodoro Timer***

I. Problem Description

The use of mobile phones and social media platforms, such as TikTok and Twitter, has become a part of our daily lives. However, there is growing concern that their widespread use can easily steal people's attention. This distraction is supported by research from Barton et al. [1] and Mehmood and Taswir [2], which found a negative correlation between social media usage and academic performance. For instance, Junco and Cotton [3] found that using Facebook and texting during homework can lower the students' overall GPA. These findings emphasize the challenge many people face in maintaining focus and multitasking efficiently.

To address this problem, our idea is to design a JavaFX-based timer and task tracker. It is for desktop use since most work is typically done on desktops. The application aims to help users stay focused and manage tasks efficiently.

II. Analysis (Related Work)

There are several strategies to help maintain focus, and among them, the Pomodoro technique, invented by Cirillo [4], has become particularly popular. The term "Pomodoro" is from the Italian word for "tomato" and was inspired by a tomato-shaped kitchen timer from the 1980s. The key idea of this technique is to break work into manageable intervals: people choose a task, focus for 25 minutes, then take a 5-minute break and repeat this cycle for subsequent tasks.

According to Biwer et al. 's work [5], structured work and break periods can enhance both mood and efficiency. As Figure 1 illustrates, compared to those who take breaks based on personal feelings, students who employ the Pomodoro method are more focused, motivated, and feel less fatigued, distracted, and bored. In Kisno's [6] study, the Pomodoro Technique significantly improved the students' reading ability when they study from home.

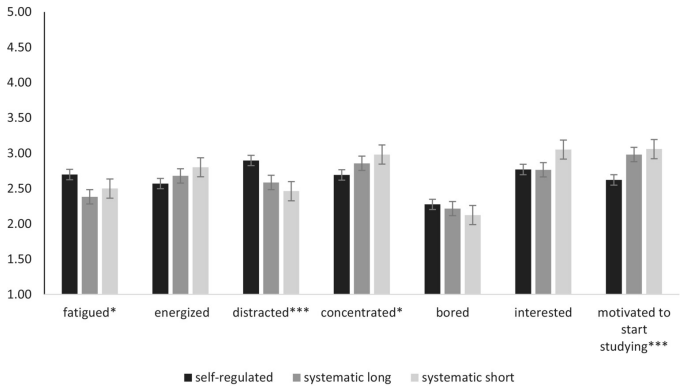


Figure 1. Average task experiences just before the breaks between three groups (Source: [5])

Another approach to enhancing focus and productivity is gamification, which means using game-style mechanics in a non-game area to enhance user participation. The approach involves a rewards system and visual progress trackers. In the reward system, users earn virtual currency when completing tasks, and then can use tokens to purchase the prize, gaining a sense of achievement. The visual progress trackers usually provide chart views or calendar views to present the statistics. Such mechanics are not only for fun but work as positive reinforcement that motivates users.

Faiella and Ricciardi [7] found that gamification is particularly effective in areas that consist of repetition work, therefore, it aligns with the Pomodoro technique. The study by Browne et al.[8] also found that the combination of gamification and the Pomodoro Technique has a significant positive effect on user motivation and productivity.

We also conducted a market analysis by surveying similar apps on the App Store and GitHub and found that they still have room for improvement. Many free tools on GitHub lack a visual task tracker or rewards system. While some tools on the App Store are good, they are not free. Additionally, some applications add social features, which can be another source of distraction ironically.

III. System Design

Drawing from the above analysis and insights, we've designed our application with the following features:

a) Task Management: Task CRUD;
b) Timer Creation: Start a Pomodoro timer for a task;
c) Customize Timer: Modify session & break intervals;
d) Statistics View: Track daily progress;
e) Shop System: Earn tokens to purchase BGM;
f) Media Preferences: Adjust session end reminders and switch BGM.

Importantly, we decided to avoid using social functions, as they could potentially steal attention and offset the main purpose of our application.

A. *Design of the System*

Our system design follows the MVC(Model-View-Controller) design pattern, emphasizing the separation of concerns. Figure 2 provides a flowchart illustrating our design.

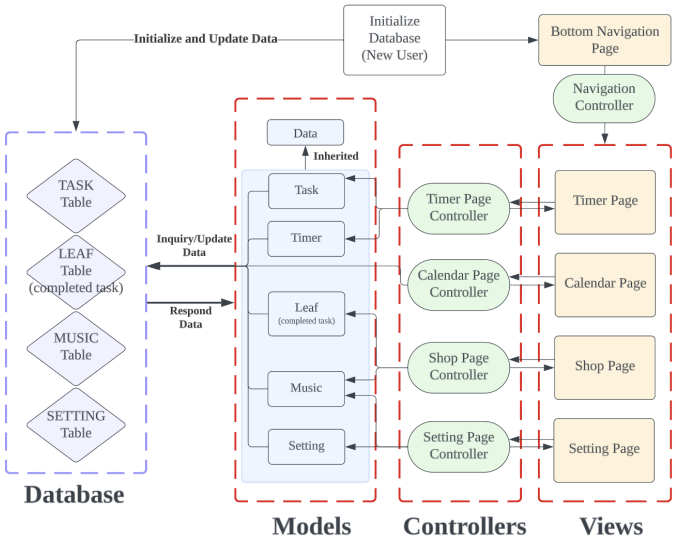


Figure 2. Design Pattern following the MVC pattern
(Created by: Author)

Referring to Figure 2, the Model, View, and Controller are three separate components, but interacts with each other:

➢ Models: Manage data, a bridge to the database;
➢ Views: Define layout and design;
➢ Controllers: Connect views with models.

In the application, each page corresponds to a specific controller. The controller identifies user actions, interacts with the models, and after the models fetch the required data from the database, the controller updates the views accordingly.

B. *The View and Controller Components (UI Design)*

When a new user initializes the application, the bottom navigation bar is loaded. The design gives users a mobile device-like experience. Users can click the buttons to switch between various pages: Timer Page, Calendar Page, Shop Page and Setting Page, as the Figure 3 shows.

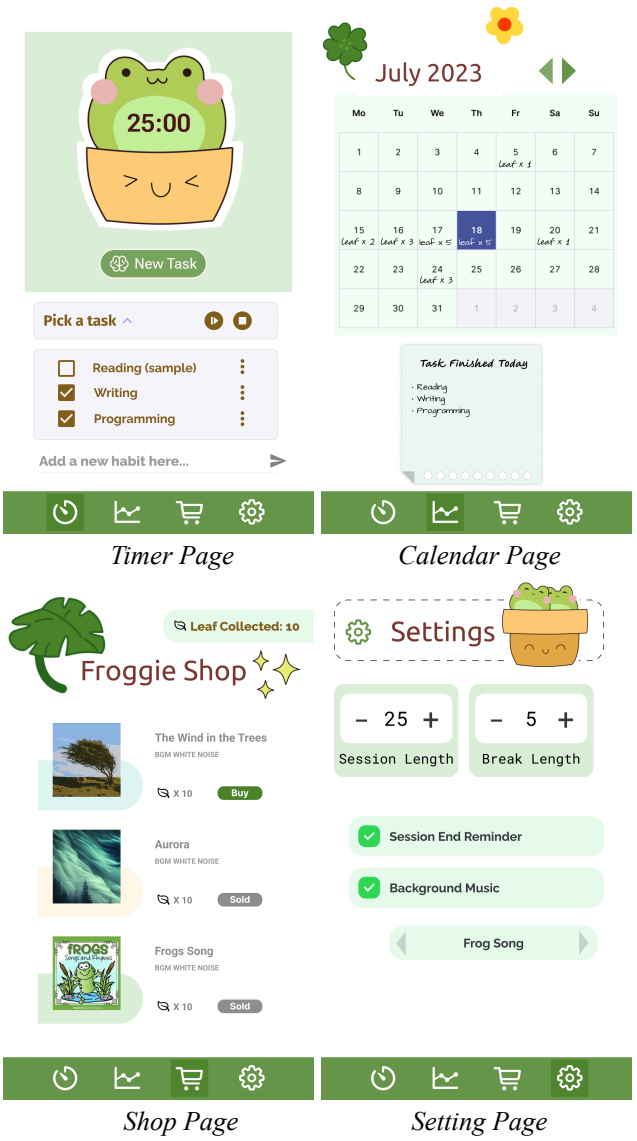


Figure 3. Mobile Device-like UI Design
(Designed by: Author; Graphic resources from Figma[9])

➢ Timer Page: This page is divided into two parts, task management and starting a timer with a selected task. It utilizes clear icons like checkboxes and an enter arrow to guide users for clarity.
➢ Calendar Page: This page visualizes the number of sessions completed daily. Users can navigate to any date and view the corresponding records.
➢ Shop Page: Every time a timer is completed, users earn virtual currency named "Leaf". On the shop page, users can spend these tokens to purchase BGM.
➢ Setting Page: This page provides users with various configuration options.

We also incorporated pop-up windows in our design, as shown in Figure 4. These windows serve as notifications and also allow users to edit tasks without navigating away from their current page.

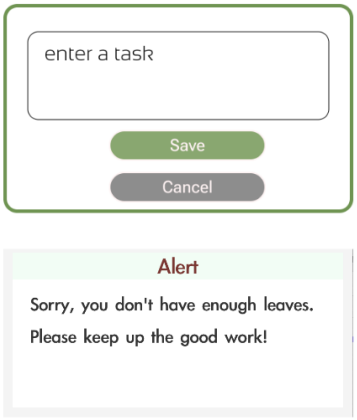


Figure 4. Pop-up Windows
(Created by: Author)

Another point is that the dynamic views are generated by and responsive with the data states. As illustrated in Figure 5, when a task is selected, its color changes. If the timer starts, the "start" button switches to a "pause" button, which aligns with the intuition and user's habits.

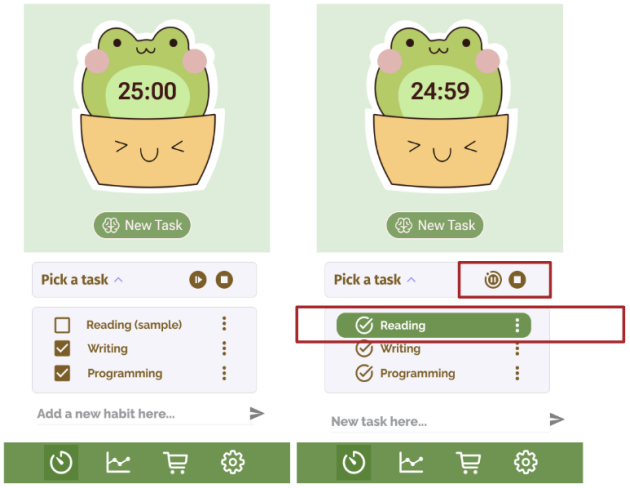


Figure 5. State-based Dynamic Views
(Designed by: Author; Graphic resources from Figma[9])

We utilize Figma for our design sketches. This tool allows us to communicate and cooperate effectively.

C. *The Model Components and the Database*

Within our models, all classes inherit from the superclass "Data" and utilize its methods to connect with the database, as Figure 6 illustrates.
The application uses Java JDBC APIs to connect to the SQLite database. There are four tables in the database: TASK, LEAF, MUSIC, and SETTING. Specifically, the TASK table is responsible for managing current to-do tasks, whereas the LEAF table archives completed tasks. This structure ensures that users can modify their current to-do lists without affecting their task history. Music preferences and configuration options are recorded in the respective tables.

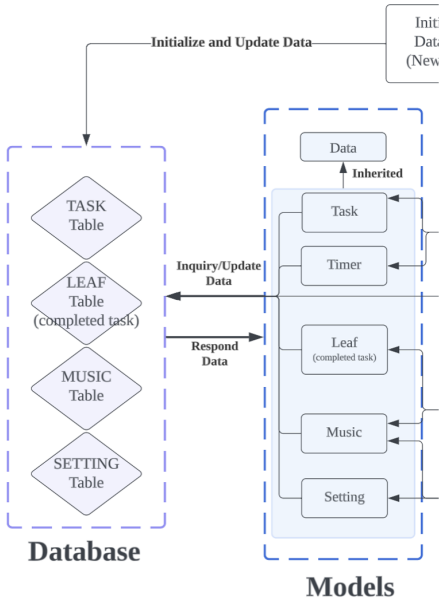


Figure 6. Database & Inheritance Structure within Models
(Created by: Author; Detail from Figure 2)

IV. Implementation

A. *Database Connection*

In order to keep the data stored locally, we use JDBC to connect SQLite. The DatabaseAccessor class is in charge of the connection with the database. When opening the project, one DatabaseAccessor object will be created by the NavigationBarController. The DatabaseAccessor will first connect with the database, examine whether it has all the tables we need. If any table is missing, we'll create the table and initialize basic data like music information and setting for later use.

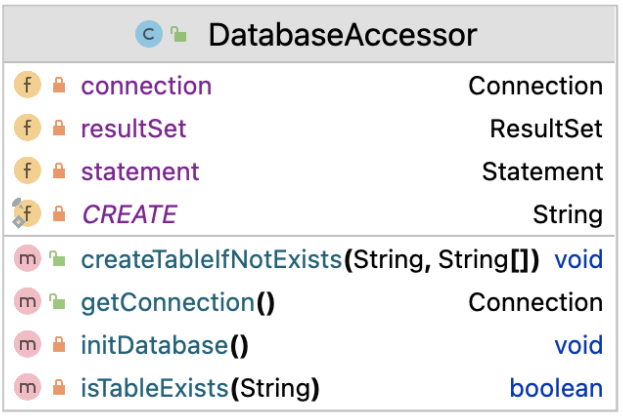


Figure 7. UML of DatabaseAccessor Class
(Created by: Author)

All models that interact with the database like Setting, Music, Task, Leaf, need methods like add/update/delete, so we implement a Data abstract class to handle these methods. Subclasses only need to implement its abstract methods to generate SQL queries for these add/update/delete methods to utilize.

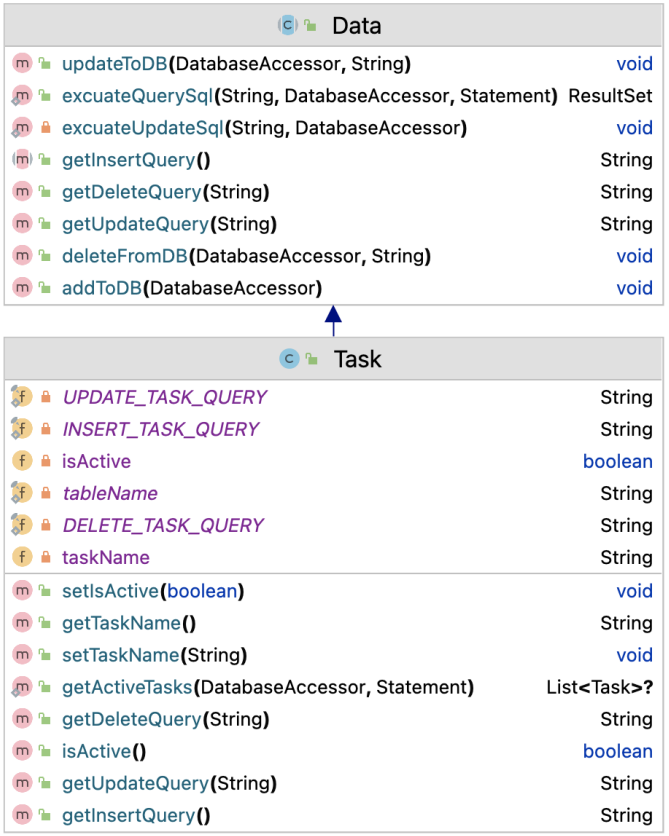


Figure 8. UML of Data and Task Class
(Created by: Author)

B. *Navigation*

The navigation bar has four buttons to navigate users to jump from a page to another. It will initialize a DatabaseAccessor object and pass it to every active page to maintain solo connection with the database to avoid database writing conflicts. What's more, the navigation bar will be temporarily disabled during focusing time to avoid distraction.

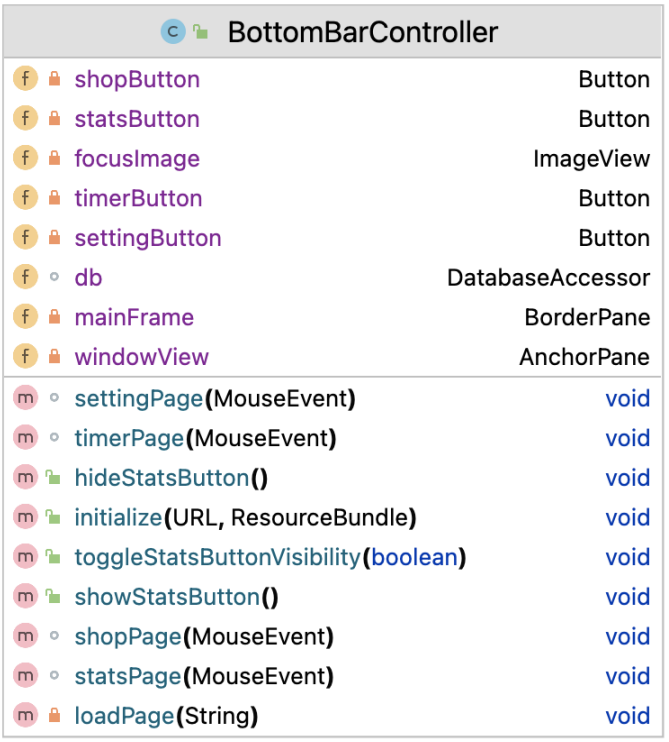


Figure 9. UML of BottomBarController Class
(Created by: Author)

C. *Timer Page*

The Timer page can be separated as two parts: timer section and task panel.

The timer section utilizes the FrogTimer object to create timers based on session length and break length stored in settings. It can also determine whether to play music during focusing time and when the session ends. And it will record every completed session into a leaf object and put it into the database.

Regarding the task panel, we have a VBox inside a Scrollpane to arrange all active tasks. The controller reads all tasks stored in the database into a List<Task>. Then the controller utilizes createTaskCard method to generate a task card for each task and the getLabelStyle method to determine the appearance of each task card. Task statuses can be updated by toggling the checkbox, and tasks can be updated or deleted by clicking the Edit button.

At the bottom of the screen, there is a text field to obtain a new task name from the UI. The controller adds this new task to the database and then calls the createTaskCard method again to generate a new task card and place it within the task panel.

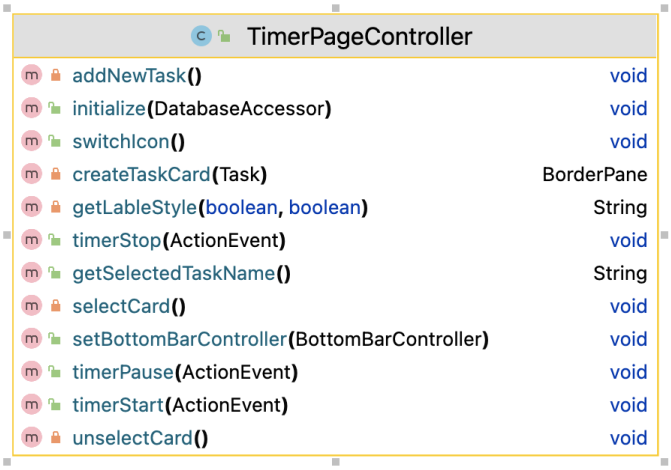


Figure 10. UML of TimerPageController Class
(Created by: Author)

D. *Calendar Page*

When the calendar page is called, CalendarPageController will load all task records completed by current month from the database, and load them into a HashMap, with completion date as the key, and List<String> as the value to accommodate all tasks completed on that day, and will use it to display everyday's earned leaves in the calendar.

When a particular date is selected, the createSticker method will be called. It will get the task list finished on this day from the former hashmap, and transform that into a map to calculate each task's completed times. The result will be displayed in the sticker on the bottom of this page.

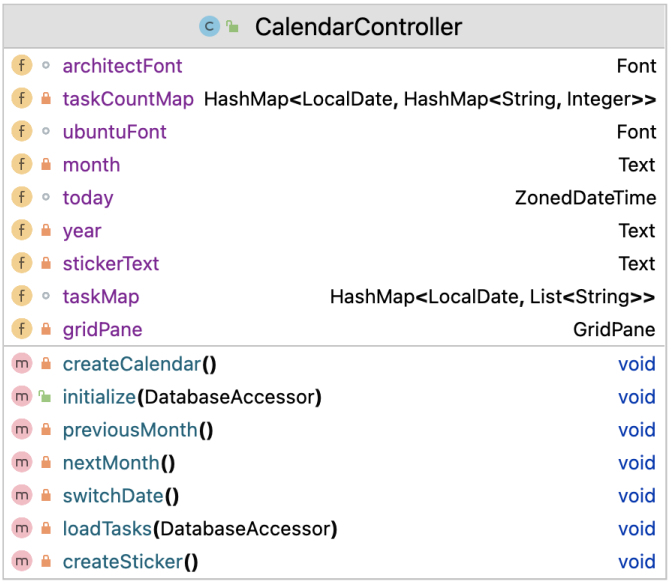


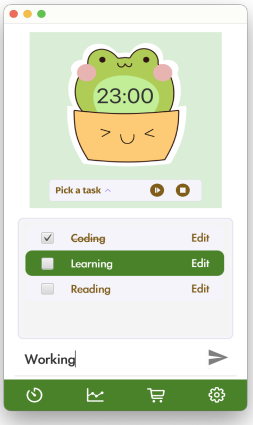
Figure 11. UML of CalendarController Class
(Created by: Author)

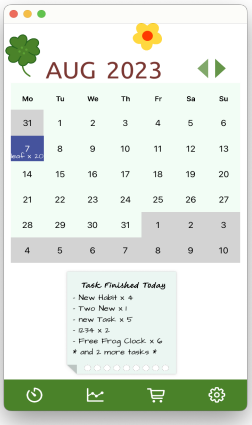
E. *Other Pages*

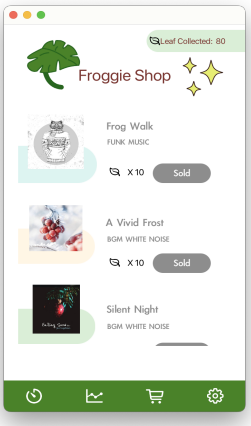
The implementations of the shop page and setting page are pretty similar to the former pages. They will be initialized with a DatabaseAccessor object to read the music and settings in the database, controllers will use these data to set the page. Buttons/checkboxes can modify Music object or Setting object, then update those changes into the database and modify the pages at the same time.

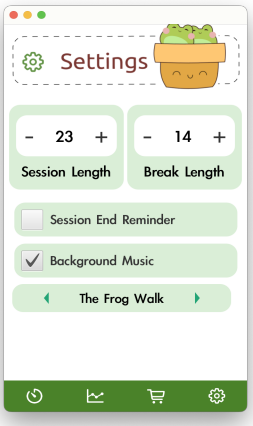
V. Evaluation

After implementation, we have completed all the functions as designed. Here are the screenshots of a sample run, and we will demonstrate the application later.









We also shared our project with our family and friends and received some excellent feedback. They appreciate the delightful UI design and clear statistics that allow them to track their daily progress. The rewarding system is motivating, and the flexible settings provide a customizable experience.

They also provided valuable suggestions to enhance user experience. For instance, new tasks should always appear at the top for easier access, and when editing a task, it's better to pre-populate the existing task name in the pop-up window. We

have incorporated these suggestions and adjusted our implementation accordingly.

VII. Discussion (Reflection)

During the project, we learned an important lesson about finding reusable parts. For instance, we noticed that objects meant to be stored and retrieved from the database all require similar methods like add, update, and delete. So, we created a common interface to make these tasks easier and save time. Another example is that we didn't realize we could reuse the same pop-up window page. Instead, each of us made our own, ending up with three different styles of pop-up windows. What's more, a great inheritance structure will make it easier for further maintenance.

Another takeaway is Good teamwork contributes to success. Breaking down a coding project into smaller and manageable tasks helps us to focus on one problem at a time, and avoid getting overwhelmed by the complexity of the whole project. And cross testing within the team really helps because sometimes the owner can't recognize the bugs of our own. And Adequate communications are necessary, especially when two or more of us are modifying one file at the same time. We learned a lot about how to resolve conflicts and merge branches during this project.

VIII. Conclusions and Future Work

In conclusion, our product is designed to help users to focus on one thing at a time and establish good habits. They can enjoy the visual and audio experience when using this app, as well as the satisfaction of earning rewards for their hard work. Our product is suitable for anyone who wants to improve their productivity with much flexibility of working time length and break intervals.

In the future, we have several things to improve: First, we will fix problems that we currently have, like dialogue windows' appearance consistency and lack of session length unit. Secondly, we aim to offer users more sophisticated visual reports and statistics. Gaining deeper insights into how we spend our time each day will assist us in developing better habits. Thirdly, we would like to provide users with more flexibility to include local music in the shopping system. Lastly, we intend to port the application to mobile platforms and the web, ensuring a consistent user experience and data synchronization across all devices.

IX. Job Assignment

1. Zhe Cao:

   UI Design

   Timer Management

   Calendar Page View and Controller

2. Simiao Wang:

   Sound Management

   Shop Page View and Controller

   Database Schema Design

3. Wenyu Yang:

   Navigation Management

   Task Management

   Setting Page View and Controller

References

[1] Barton B A, Adams K S, Browne B L, et al. The effects of social media usage on attention, motivation, and academic performance[J]. Active Learning in Higher Education, 2021, 22(1): 11-22.

[2] Mehmood S, Taswir T. The effects of social networking sites on the academic performance of students in college of applied sciences, Nizwa, Oman[J]. International Journal of Arts and Commerce, 2013, 2(1): 111-125.

[3] Junco R, Cotten S R. No A 4 U: The relationship between multitasking and academic performance[J]. Computers & Education, 2012, 59(2): 505-514.

[4] Cirillo F. The Pomodoro technique: The acclaimed time-management system that has transformed how we work[M]. Currency, 2018.

[5] Biwer F, Wiradhany W, oude Egbrink M G A, et al. Understanding effort regulation: Comparing 'Pomodoro' breaks and self‐regulated breaks[J]. British Journal of Educational Psychology, 2023.

[6] Kisno K. Pomodoro Technique For Improving Students' Reading Ability During COVID-19 Pandemic[J]. Jurnal Education and Development, 2020, 8(3): 561735.

[7] Faiella F, Ricciardi M. Gamification and learning: a review of issues and research[J]. Journal of e-learning and knowledge society, 2015, 11(3).

[8] Browne R, Raeside L, Gray G. Gamification in education: productivity and motivation through gamified time management software[C]//European Conference on Games Based Learning. Academic Conferences International Limited, 2018: 867-871.

[9] "Froggie Sticker Set." Figma Community. Available at: https://www.figma.com/community/file/1224385126549371634