



Frog Clock

-- A Combination of Pomodoro Clock &
Task Tracker

CSYE 6200 Summer 2023
Professor: Jones Yu
Team: Leap Frog
Simiao Wang, Wenyu Yang, Zhe Cao



Outline

- ❖ I. Problem Description
- ❖ II. Analysis (Related Work)
- ❖ III. System Design
- ❖ IV. Implementation
- ❖ V. Evaluation
- ❖ VII. Discussion (Reflection)
- ❖ VIII. Conclusions and Future Work
- ❖ IX. Job Assignment
- ❖ X. Live demo

Introduction

❖ Problem Description:

- **Maintaining Focus & Efficient Multi-tasking**
- Common issues among the young generation
- Mobile devices & social media "steal" our concentration
 - Using Facebook during homework lowers GPA ¹

❖ Our Idea:

- **Timer & Task Tracker** using JavaFx
- Desktop Application
- Enhance Focus & Task Efficiency



¹ Junco, R., & Cotten, S. R. (2012). No A 4 U: The relationship between multitasking and academic performance. Computers & Education, 59(2), 505-514.

Analysis



- ❖ Pomodoro Technique²
 - Pomodoro: Italian word for *Tomato*
 - Worked as a kitchen timer
 - Typically 25-minute work intervals with short breaks

- ❖ Gamification (Reward System)
 - Visualize statistics
 - Earn virtual currency & rewards
 - Enhances motivation to finish tasks³



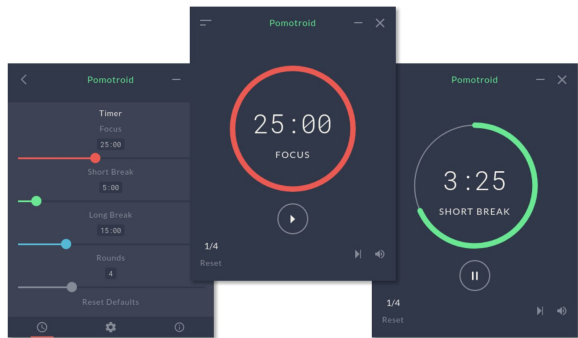
² https://en.wikipedia.org/wiki/Pomodoro_Technique

³ Browne, R., Raeside, L., & Gray, G. (2018, October). Gamification in education: productivity and motivation through gamified time management software. In European Conference on Games Based Learning (pp. 867-871). Academic Conferences International Limited.

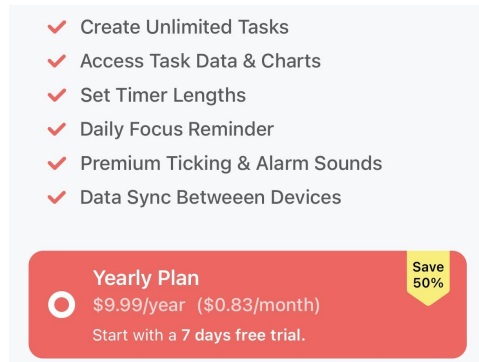
Analysis



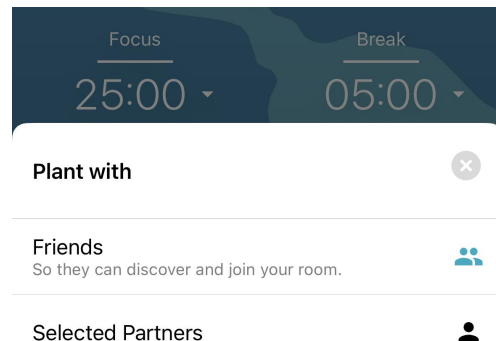
- ❖ Existing Applications
 - Limited integrated: no task tracker or rewards system
 - Many are not free
 - Social features: leading to fragmented attention



Limited Function

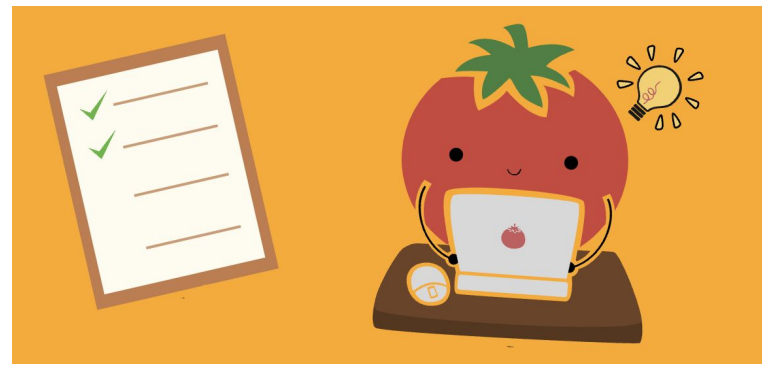


Membership Cost



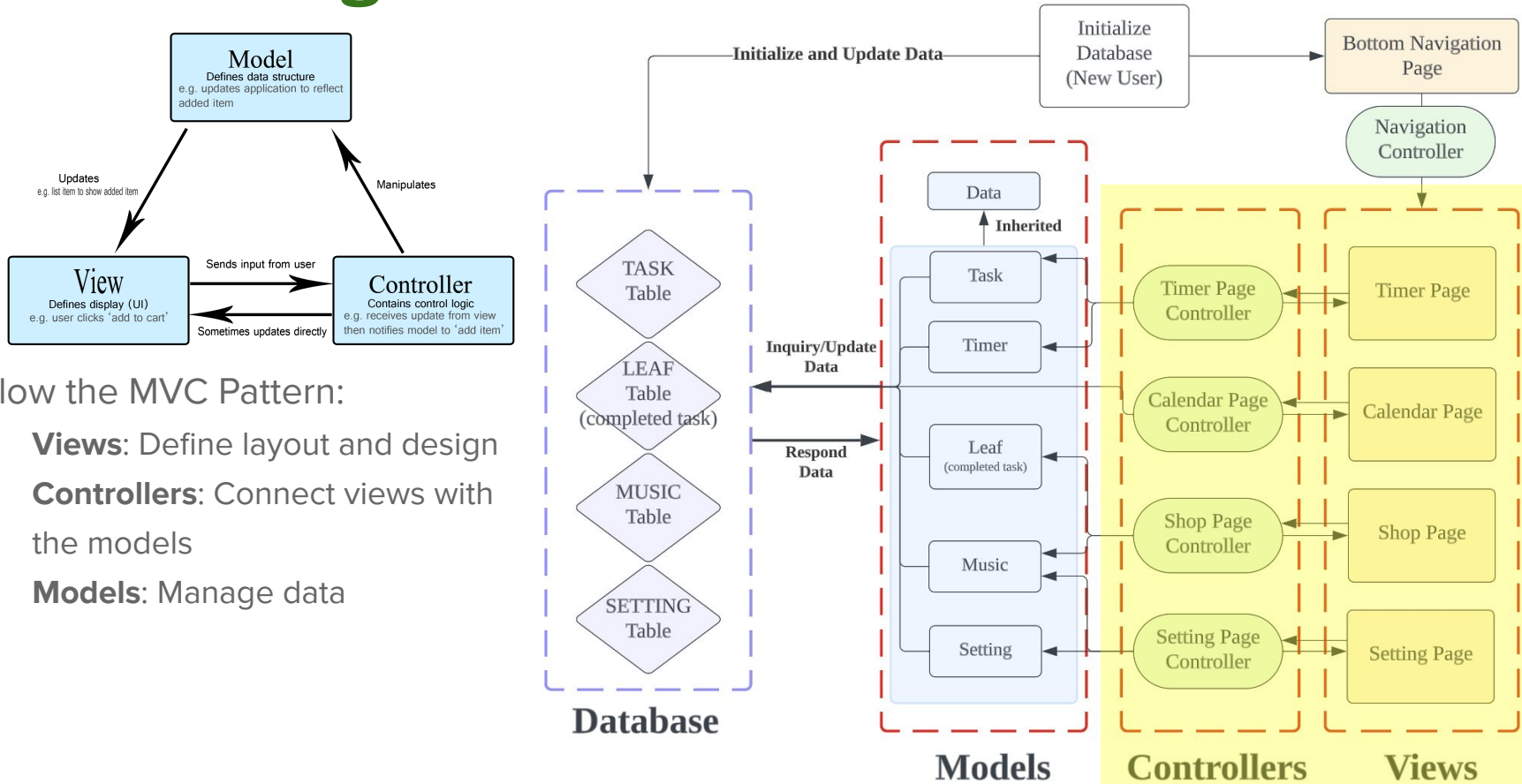
Distracting Social Features

System Design



- ❖ Features of Our Application:
 - **Task Management:** Task CRUD
 - **Timer Creation:** Start a Pomodoro timer for a task
 - **Customize Timer:** Modify session and break intervals
 - **Statistics View:** Track daily progress
 - **Shop System:** Earn "tokens" to purchase BGM
 - **Media Preferences:** Adjust session end reminders and switch BGM

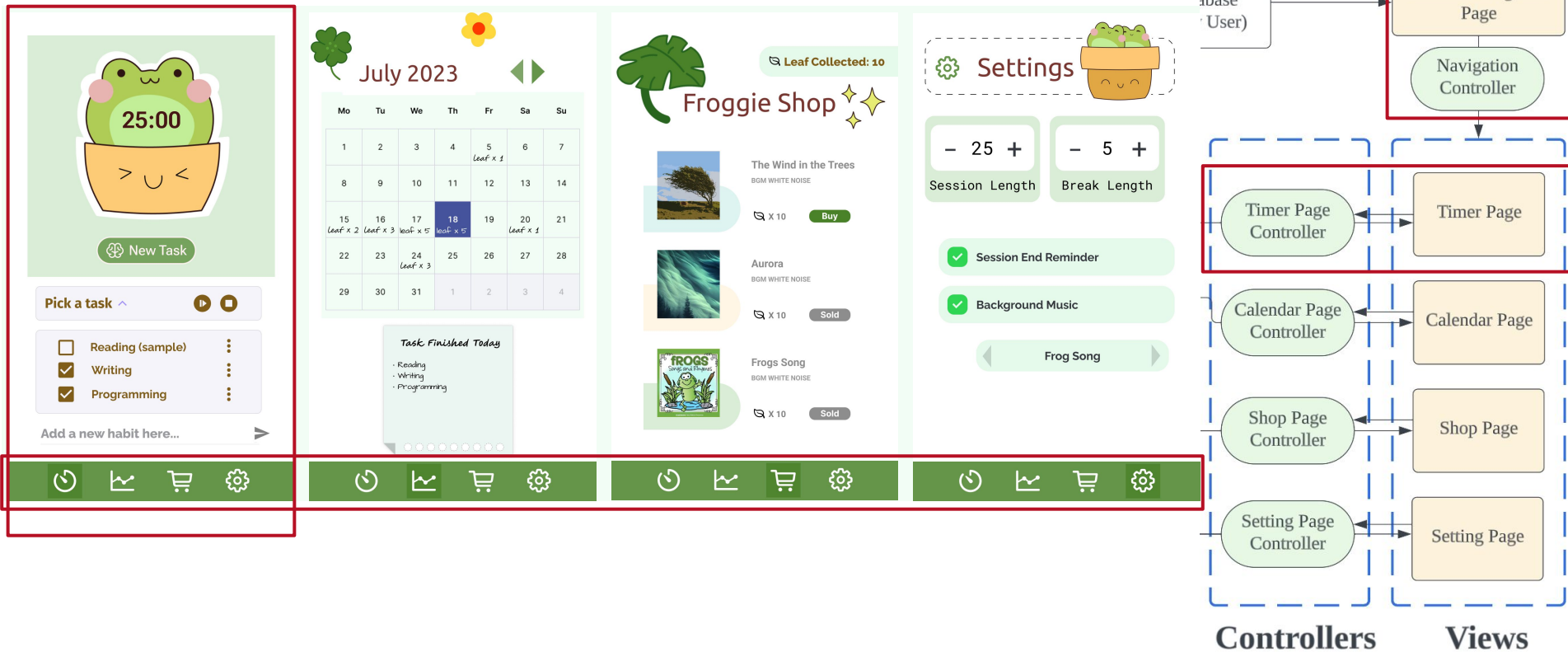
System Design - MVC



Follow the MVC Pattern:

- ❖ **Views:** Define layout and design
- ❖ **Controllers:** Connect views with the models
- ❖ **Models:** Manage data

System Design - View & Controller



System Design - View & Controller

❖ Pop-up windows

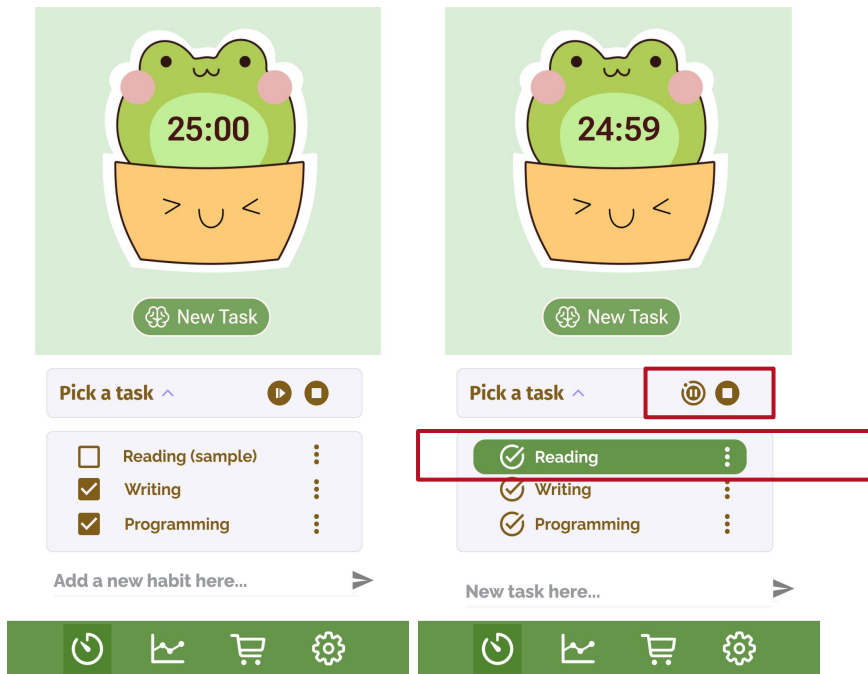
Save

Cancel

Alert

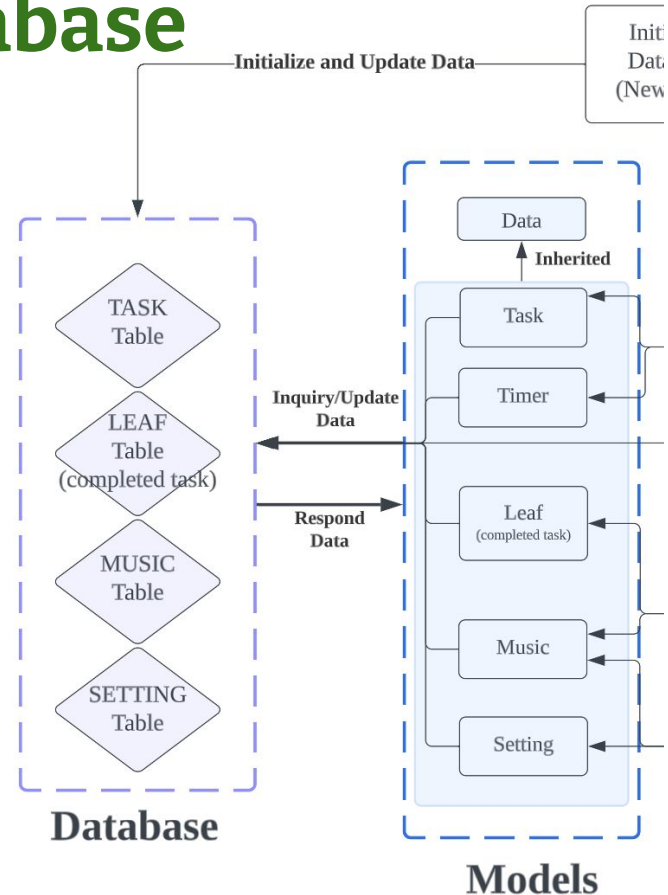
Sorry, you don't have enough leaves.
Please keep up the good work!

❖ State-based dynamic views



System Design - Models & Database

- ❖ SQLite database + JDBC API
- ❖ 4 Tables:
 - **Task:** To-do list tracker
 - **Leaf:** Completed task records
 - **Music:** Media data
 - **Setting:** Configuration data



Implementation - Timer Page

Static variables to generate SQL queries

Use List<Task> to accept selected task store in database

Implement abstract methods to get SQL queries

Getters and Setter to access private variables

Control the timer and music based on settings

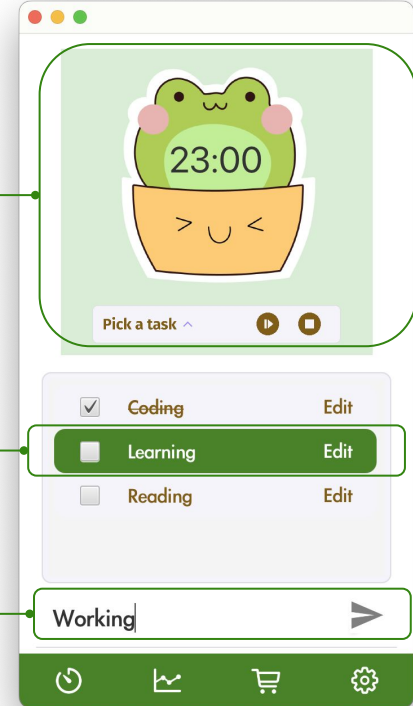
```
class Data {
    + addToDB(DatabaseAccessor) void
    + deleteFromDB(DatabaseAccessor, String) void
    + excuateQuerySql(String, DatabaseAccessor, Statement) ResultSet
    + excuateUpdateSql(String, DatabaseAccessor) void
    + getDeleteQuery(String) String
    + getInsertQuery() String
    + getUpdateQuery(String) String
    + updateToDB(DatabaseAccessor, String) void
}
```

```
class Task {
    + DELETE_TASK_QUERY String
    + INSERT_TASK_QUERY String
    + UPDATE_TASK_QUERY String
    + isActive boolean
    + tableName String
    + taskName String
    + getActiveTasks(DatabaseAccessor, Statement) List<Task>
    + getDeleteQuery(String) String
    + getInsertQuery() String
    + getTaskName() String
    + getUpdateQuery(String) String
    + isActive() boolean
    + setisActive(boolean) void
    + setTaskName(String) void
}
```

```
class FrogTimer {
    + breakTimeRemaining long
    + currentTime AnimationTimer
    + db DatabaseAccessor
    + isBreak boolean
    + isRunning boolean
    + lastBreak long
    + lastSession long
    + lastUpdate long
    + timeRemaining long
    + getInsertQuery() String
    + handleBreak(long, Label, TimerPageController) void
    + handleSession(long, Label, String, MediaPlayer) void
    + pauseCountdown(Label, MediaPlayer) void
    + startCountdown(Label, TimerPageController, MediaPlayer) void
    + stopCountdown(Label) void
    + updateTimeLabel(Label, long) void
}
```

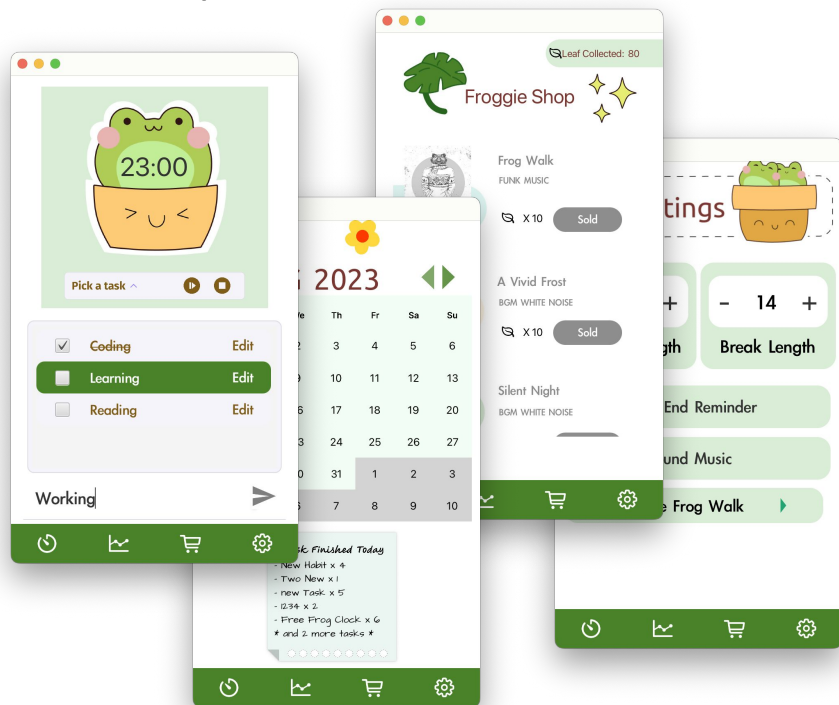
```
class TimerPageController {
    + BROWN String
    + FONT_AND_SIZE String
    + LINE_THROUGH String
    + WHITE String
    + addNewTaskButton Button
    + bottomBarController BottomBarController
    + db DatabaseAccessor
    + font Font
    + mediaPlayer MediaPlayer
    + newTaskTextField TextField
    + pauseButton Button
    + selectedTaskCard BorderPane
    + selectedTaskName String
    + startButton Button
    + taskPannel VBox
    + timeLabel Label
    + timer FrogTimer
    + addNewTask() void
    + createTaskCard(Task) BorderPane
    + getLabelStyle(boolean, boolean) String
    + getSelectedTaskName() String
    + initialize(DatabaseAccessor) void
    + selectCard() void
    + setBottomBarController(BottomBarController) void
    + switchIcon() void
    + timerPause(ActionEvent) void
    + timerStart(ActionEvent) void
    + timerStop(ActionEvent) void
    + unselectCard() void
}
```

```
class DatabaseAccessor {
    + CREATE String
    + connection Connection
    + resultSet ResultSet
    + statement Statement
    + createTableIfNotExists(String, String[]) void
    + getConnection() Connection
    + initDatabase() void
    + isTableExists(String) boolean
}
```



Evaluation

❖ Sample Runs



❖ User Feedback

- Delightful desktop application that can help users to focus.
- Showing clear statics to trace everyday's progress.
- Rewarding system to encourage staying focus.
- Flexible Setting to provide custom experience.

❖ Improve Suggestions

- New Task should go to the top. ✓
- Edit should pre-populate the existing item name. ✓
- Completed item shouldn't start timer. ✓

Discussion and Reflection

Programmers are
LAZY~



1

Abstract common patterns and reuse them

- Abstract classes and interfaces help to finish common tasks.
- Lack of notice might result in multiple effort to finish same task.
- Great inheritance structure make it easier for testing, debugging and maintenance.

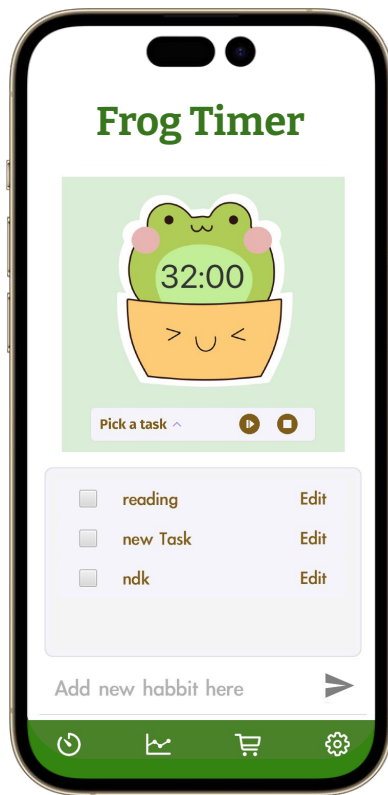
2

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.
- Cross testing can provide us with fresh opinions when finding bugs.
- Adequate communications are necessary, especially when we are modifying one file at the same time.

Future Work

- ❖ Provide users with more sophisticated visual reports and statistics to help users build habits.
- ❖ Allow users to choose local music as background music.
- ❖ Transplant the program to the web and mobile platforms to provide users with consistent experience and data synchronization services.



Job Assignments

- ❖ Zhe Cao:
 - UI Design
 - Timer Management
 - Calendar Page View and Controller
- ❖ Simiao Wang
 - Sound Management
 - Shop Page View and Controller
 - Database Schema Design
- ❖ Wenyu Yang
 - Navigation Management
 - Task Management
 - Setting Page View and Controller