System Design Document
Project Name: Moody

Project Team: Najwa, Abdihakim, Hamza
Product Owner: Abdihakim
Scrum Demo TA: Claudia

Sprint 3

Table of Contents

1. Introduction

Overview

Moody is a Java-based GUI application designed to help users manage tasks effectively. It provides users with the ability to create, categorize, track completion, and manage deadlines for tasks. The application includes user authentication, task management with due dates, a priority system, and a timer functionality for tracking task deadlines.

2. Goals

- **User Login**: Implement user login functionality with validation and authentication.
- **Task Management**: Enable users to create and categorize tasks into priority levels.
- **Task Completion**: Allow users to track completed tasks and update task status.
- **Due Date and Timer**: Allow users to set due dates for tasks and track deadlines with a timer.
- **Settings Management**: Provide user settings for password reset and logout functionalities.
- **UI Design**: Ensure a user-friendly interface using Java Swing components.
- **Task Progression Bar:** Implement a dynamic progress bar to visually display the percentage of completed tasks.
- **Reminder Feature:** Add a reminder system to notify users about pending tasks.

3. System Architecture

The **Moody** application is a standalone desktop GUI application built using Java Swing for the user interface. It consists of the following modules, with the addition of **Timer** and **Due Date** components to manage task deadlines effectively:

1. **Login Module**: Handles user authentication. The user enters their credentials, and the system validates them before redirecting the user to the task management page.
2. **Task Management Module**: Allows users to create, categorize, and manage tasks. This module is responsible for organizing tasks by priority, tracking task completion, and managing **Due Dates** and **Timers**:
    - **Due Dates**: Each task can have a due date set by the user. This feature is implemented using Java's LocalDate or Date to store the task's deadline.
    - **Timer**: Each task that has a due date will be assigned a **Timer** object, which starts counting down to the task's deadline as soon as it is set. The timer triggers alerts when the due date is approaching or has passed.
3. **Task Progression Bar:** A new feature added to visually track the progress of task completion. The backend will calculate the percentage of completed tasks, and the frontend will display this as a dynamic progress bar. This bar will update in real-time as tasks are added, completed, or removed.
4. **Reminder Feature:** A reminder system to notify users about pending tasks. The backend will queue reminders to avoid overlapping reminders. Users will receive notifications at scheduled times, ensuring they stay on track with task deadlines.
5. **User Settings Module**: Manages user settings such as password reset and logout functionalities. This module is responsible for providing the user with the ability to update credentials and log out of the application.
6. **Timer and Due Date Management**:

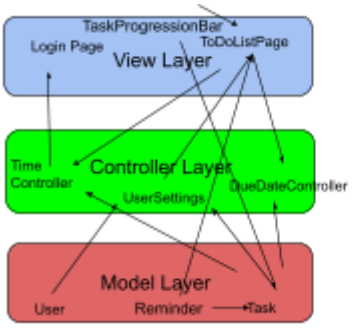
- **Timer**: The **Timer** class is a critical component for tracking the countdown for each task. When a user sets a due date for a task, the **Timer** object is initialized to begin counting down from the set time.
- The timer continuously updates the **ToDoListPage** to show the remaining time for each task's due date. A notification is triggered when the task's deadline is near (e.g., within 24 hours).
- If the due date has passed, the task's status is updated automatically to indicate the deadline has been missed.

4. Class-Responsibility-Collaborator (CRC) Cards

Class	Responsibilities	Collaborators
LoginPage	Display login form; verify user credentials against `User` data; redirect to `ToDoListPage` upon successful login.	`User`, `ToDoListPage`
User	Manage and store user credentials; validate and update passwords.	`LoginPage`, `ToDoListPage`
ToDoListPage	Display main UI for task management; organize tasks into priority tabs; manage completed tasks and user settings.	`User`, `LoginPage`
Task	Represent individual tasks with details like name, priority, and completion status; manage task behaviors like updating completion.	`ToDoListPage`
UserSettings	Handle user settings management, such as password reset and logout; interact with `User` for credential updates.	`User`, `ToDoListPage`
Timer	Track remaining time for each task based on the due date; notify the **ToDoListPage** when a task's deadline is near or has passed.	`Task, ToDoListPage`

DueDate	Represent and manage task deadlines, allowing users to set, update, and validate due dates. Provide integration with `Timer` for tracking and notifications.	`Task, Timer`
TaskProgressionBar	Display the progress of each task based on completion percentage. Update dynamically as tasks are marked as complete or modified. Provide visual feedback to the user regarding the status of their tasks.	`ToDoListPage, Task`
Reminder	Manage reminders for upcoming or overdue tasks. Notify users at scheduled times when tasks are approaching their due dates or have passed.	`ToDoListPage, Task`

5. Software Architecture Diagram



Interactions:

- **LoginPage** communicates with **User** to verify login credentials.
- Upon successful login, **LoginPage** redirects to **ToDoListPage**.
- **ToDoListPage** interacts with **Task** to manage tasks (create, update, move tasks).
- **Task** communicates with **TimeController** to set and manage countdowns for task deadlines.

- **Task** communicates with **DueDateController** to set and track task due dates.
- **TimeController** interacts with **Timer** to manage countdowns and track time for task deadlines.
- **ToDoListPage** communicates with **UserSettings** for password reset and logout functionalities.
- **UserSettings** interacts with **User** to update preferences and settings like notifications or theme.
- **ToDoListPage** communicates with **NotificationService** to alert the **User** about task deadlines or changes.
- **ToDoListPage** communicates with **TaskProgressionBar** to update the task progress based on completion status.
- **Task** communicates with **TaskProgressionBar** to send task completion status and update progress.
- **Task** communicates with **Reminder** to provide due date information for setting reminders.
- **ToDoListPage** communicates with **Reminder** to display notifications about due tasks.

6. Detailed Design

6.1 Login Flow

- **User Input**: The user enters their username and password on the login screen.
- **Authentication**: The system validates the credentials entered by the user against the **User** object.
- **Access Control**: If the credentials are valid, the login page closes, and the **ToDoListPage** is displayed.

6.2 Task Management Flow

- **Task Creation**:
  - The user inputs task details (name, priority, and due date) on the **ToDoListPage**.
  - Upon clicking the "Add Task" button, the **addTask()** method is triggered, creating a new task, which is displayed in the appropriate priority tab.
- **Due Date**:
  - Each task can have a **due date** entered by the user.
  - The task is stored with a **due date** value (e.g., `Date` or `LocalDate` in Java).
- **Timer**:
  - The **Task** class includes a **Timer** object that starts counting down as soon as the task's due date is set.
  - A timer is initialized for each task when the due date is entered. The **Timer** class counts down to the task's due date and triggers an alert when the deadline is approaching or has passed.

    - The **ToDoListPage** displays a countdown next to each task, indicating the remaining time until the due date.
- **Task Progression Bar:**
    - A TaskProgressionBar is displayed on the ToDoListPage, showing overall task progress based on the number of completed tasks versus total tasks. The progress bar updates dynamically whenever a task is marked as completed or a new task is added.
- **Reminder:**
    - The **Reminder** system manages notifications for tasks with due dates. Using a queued notification approach, reminders are triggered without overlap when a task's deadline is near or overdue.
- **Task Completion**:
    - When the user selects a checkbox next to a task, it is marked as complete. The task is moved to the "Completed Tasks" tab using **moveToCompletedTasks()**, and the checkbox is disabled to prevent further modification.
- **Task Deletion**: A future enhancement will be added to allow users to delete tasks.

6.3 User Settings Flow

- **Password Reset**:
    - In the **User Settings** panel, the user enters their current password and a new password.
    - Upon clicking "Reset Password," the system verifies the current password and updates it if valid, using **setPassword()**.
    - A success or failure message is displayed based on the outcome.
- **Logout**:
    - Clicking "Logout" closes the task management page and reopens the login page for authentication.
    -

7. User Interface Design

Login Page (LoginPage)

Components:

- Username and password fields.
- Login button to validate credentials.
- Forgot password button (future enhancement).

Main Task Management Page (ToDoListPage)

Components:

1. **Tabbed Pane for Categories**
    - **Stressed:** Displays high-priority tasks.
    - **Productive:** Displays moderate-priority tasks.
    - **Completed Tasks:** Lists tasks marked as complete.
    - **User Settings:** Provides user account management options.
2. **Task Input Panel**

- **Text Field:** For entering task names.
- **Priority Dropdown:** To select task priority (High, Medium, Low).
- **Due Date Field:** A simple date picker to set the due date for tasks.
- **Timer Setup:**
    i. Input fields to set a task timer duration (e.g., 25 minutes).
    ii. Option to specify study break durations (e.g., 5 minutes).
- **Add Task Button:** Adds the task to the appropriate priority category.

3. **Task Panels (Within Categories)**

- **Task Name:** Displays the task name.
- **Due Date Display:** Shows the date the task is due (e.g., "Due: 2024-11-20").
- **Timer Control:**
    i. Start, pause, and reset buttons for the task timer.
    ii. Countdown timer showing remaining task time (e.g., "Time left: 15:00").
- **Add Task Button:** Adds the task to the appropriate priority category.
- **Break Timer Notification:**
    i. **When the task timer completes, prompts the user to take a study break.**
    ii. **Displays a countdown for the break duration (e.g., "Break time: 5:00").**
- **Completion Checkbox: To mark tasks as complete. Completed tasks are moved to the "Completed Tasks" tab.**
- **Visual Alerts and Notifications:**
    i. Timer completion triggers an alert (visual and/or audio notification).
    ii. Overdue tasks display a visual alert in red text or with a warning icon.

4. **TaskProgressionBar**

- Displays a visual progress bar indicating the percentage of tasks completed versus total tasks.
- Dynamically updates as tasks are completed or added.

5. **Visual Alerts and Notifications:**

- **Timer Completion Alert:** Triggers visual/audio notifications when a task's timer completes.
- **Overdue Task Alert:** Displays overdue tasks in red text or with a warning icon.
- **Reminder Alert:** Notifies users of upcoming deadlines without overlap using queued notifications.

User Settings Panel (UserSettings)

Components:

- **Password Reset Fields:** For entering the current password and a new password.
- **Reset Password Button:** To confirm and update credentials.
- **Logout Button:** Closes the task management page and reopens the login page.

8. Future Enhancements

- **Recurring Tasks**: Enable users to set tasks that repeat on a schedule (e.g., daily, weekly).
- **Reminder Notifications**: Implement reminders and notifications for high-priority or upcoming tasks, triggered by the task's timer.
- **Persistent Data Storage**: Introduce data persistence through file storage or database integration to save tasks and user credentials across sessions.