

**North South University**

**Department of Electrical & Computer Engineering**

**Project Report**

*Video Demonstration Link:* [*CSE327\_Grp4\_ShortDemo.mp4*](https://drive.google.com/file/d/1cD2GGtzaNDevf2FynJ4pWl67e67KPL1l/view?usp=sharing)

*Prepared by:*

| *Zuhayer Islam* | *223 2061 642* |
| --- | --- |
| *Tanvir Ahmed* | *223 1047 642* |
| *Shefa Tabassum* | *223 2993 042* |
| *Faheema Shaheed Tamanna* | *223 2144 642* |

*Course: Cse327 Section: 06 Group: 04*

*Submitted to Dr. Mohammad Rezwanul Huq(MRH1)*

*Date: 5 September,2025*

**Software Requirements Specification** **For** **Intelligent Task Planner**

**Table of Contents**

[**1. Introduction**](#_2ul4hpeouest)

[1.1 Purpose](#_6rsppyfoqos4)

[1.2 Intended Audience](#_231r1q74xe8l)

[1.3 Intended Use](#_75b40f6asxts)

[1.4 Product Scope](#_jowcrsaxywfm)

[1.5 Definitions and Acronyms](#_c8v5ytmi7hl6)

[**2. Overall Description**](#_i9pf6ynk3nps)

[2.1 User Needs](#_hkm3ut6bop43)

[2.2 Assumptions and Dependencies](#_lrbeypmjb21f)

[**3. System Features and Requirements**](#_4dl8krf3dc98)

[3.1 Functional Requirements](#_hjvm3tjfxcry)

[1. FR1: User Registration](#_5jktwarklx36)

[2. FR2: User Login](#_56c9eb7uwyy3)

[3. FR3: Add Task](#_lrv5dym501h9)

[4. FR4: Edit Task](#_x1oobhtjpqb3)

[5. FR5: Delete Task](#_1l3ihjbfwqwc)

[6. FR6: View Tasks](#_nu0crfyl4n5u)

[7. FR7: View Analytics](#_tpvzhleacoks)

[8. FR8: Manage Profile](#_brb0s929wk7j)

[9. FR9: View Schedule](#_gqks0b9gilbt)

[10. FR10: Add Pomodoro Timer](#_qkczjhl8fwgy)

[11. FR11: User Logout](#_7fux21aq2ogz)

[3.2 Non Functional Requirements](#_ntijwhgav50x)

[1. Usability](#_16i2e24u87ge)

[3.3 Attributes of the Requirements](#_lqf2h8tmbtsm)

[3.4 External Interface Requirements](#_xe1eqw9tn812)

[1. User Interface](#_8dkmj4hh3cnn)

[2. Hardware Interfaces](#_tyr2o5ecg4yd)

[3. Software Interfaces](#_wgc3vgm63dra)

[4. Communication Interfaces](#_d6g1vao6wn0j)

[**4. Other Requirements**](#_ho9xvjjhzl58)

[4.1 Database Requirements](#_qi5oal92oa4s)

[4.2 Legal and Regulatory Requirements](#_fxmv1txdyr5u)

[4.3 Internationalization and Localization](#_4kl63a4cawyn)

[4.4 Risk Management (FMEA Matrix)](#_9snzts4anyla)

[**5. Appendices**](#_vvlth7iny6aq)

[**5.1 Use Case Diagram**](#_i3jdy0wvpff1)

[5.2 To Be Determined (TBD) List](#_nfbff91y677e)

[5.3 Contributions in the SRS](#_i63yq4n8gkj9)

[1. Class Diagram](#)

[2. ER Diagram (Database Design)](#)

[3. Sequence Diagram](#)

[4. Activity Diagrams](#)

[4.1 Activity Diagram for Registration/Login](#)

[4.2 Add/Edit Task](#)

[5. State Machine Diagram](#_494ba9ib8b9s)

[6. User Access Control](#_rd0jikoukk9j)

[7. Project Scheduling (using CPM)](#_2woo1h1bmv6r)

[8. Functionality-wise System Description (with screenshots)](#_mgh3o28oiifg)

[Add New Task Form – Functional Description](#_8yqe9hcripnw)

[Input Fields:](#_mzcwidnsjngv)

[Submit Button:](#_jtg29yte6e0r)

[Purpose:](#_x9b54eouuqmf)

[Edit Task Form – Functional Description](#_tst39tq54l0)

[Pre-Filled Fields:](#_l3i5bbbzs8wr)

[Submit Button:](#_qbuqpt1h3bml)

[Backend Processing:](#_uspyxdf21uyx)

[Purpose:](#_dif605rf9604)

[**Task List Page – How Tasks Are Displayed and Updated**](#_ke7zedowetz3)

[1. Task Display Section](#_2l39y6h9jscu)

[2. Buttons for Interaction](#_qrjx0spedbx7)

[3. Filtering and Views](#_fdsokviwepty)

[4. Automatic Page Update After Edit](#_8rkw9x46cas1)

[**Analytics Page – Functional Description**](#_esu36kykm9ew)

[Key Components](#_a24fsnawm6s6)

[Backend (Django Views)](#_yzr8mwgrycy7)

[Frontend Integration](#_nj37uwz6514p)

[9.Modeling and Implementation Challenges](#_1mdftucxl4oj)

[10. Conclusion](#_qwgm36otl2vh)

[11.Overall Contribution](#_zhw0eke23ej3)

# 1. Introduction

## 1.1 Purpose

This document is a detailed software requirements specification (SRS) for the ‘Intelligent Task Planner’ web application named ‘Productivo’. Productivo is a Django-based web application designed to help university students manage their academic workload and daily tasks. It aims to provide smart scheduling recommendations based on user inputs such as task deadlines, priorities, and available time.

## 1.2 Intended Audience

This product is primarily designed for students, particularly university students, to help them manage their schedules effectively.

## 1.3 Intended Use

The system allows registered users to manage their tasks and schedules effectively. It will support priority-based task sorting, smart time slot recommendations according to the user’s schedule, reminder notifications, and a Pomodoro timer and visual analytics to help track time management efficiency. The system will also allow students to make manual changes to their schedules when necessary, while adapting intelligently over time.

## 1.4 Product Scope

This system will:

* Allow students to add, edit, delete, and view academic tasks.
* Provide AI-assisted or rule-based scheduling suggestions.
* Support task prioritization, time blocking, reminders, and Pomodoro timing.
* Display analytics on time usage and productivity trends.
* Allow manual adjustment of task schedules.
* Sync with Google Calendar.

## 1.5 Definitions and Acronyms

* Task: A scheduled activity for a given day. The user inputs details such as day, date, priority, task type, and duration.
* Pomodoro Timer: A 25-minute timer that users can use to take breaks between tasks to improve work efficiency.
* UI: User Interface
* API: Application Programming Interface
* FR: Functional Requirement
* NFR: Non Functional Requirement
* EARS: Easy approach to Requirements Syntax.
* Gherkin: A simple language used to write test scenarios easy for technical or non -technical stakeholders to understand.

# 2. Overall Description

## 2.1 User Needs

University students require a platform to efficiently manage their day-to-day

academic activities. Productivo aims to provide this solution by addressing several

common challenges:

* Students require a platform to plan their schedules effectively and monitor their daily progress.
* Students need the application to provide timely reminders so that they don’t miss deadlines.
* Students need timer options to provide structured breaks between study sessions to maintain focus and productivity.
* Students need to be able to view and categorize tasks based on type, such as assignments, exams, or personal tasks, for better organization and clarity.
* Students need suggestions to improve their schedules.

## 2.2 Assumptions and Dependencies

Assumptions:

* Users will access the system through modern web browsers (e.g., Chrome, Firefox).
* Users will have a stable internet connection.
* Users are expected to register and log in before accessing core features.
* Students will input accurate information related to their tasks (e.g., date, time, type, priority).
* The system will be primarily used by students.

Dependencies:

* The system is developed using Django, which requires a compatible Python environment.
* The application depends on a relational database (e.g., PostgreSQL, MySQL) to store user data and task information.
* Google Calendar API integration is required to enable calendar sync functionality.
* Reminder notifications may depend on a third-party library or email/SMS service (e.g., SMTP, Twilio, or Push services).
* The Pomodoro timer and real-time components may rely on JavaScript (with WebSockets or AJAX).

# 3. System Features and Requirements

## 3.1 Functional Requirements

Functional requirements describe the specific behavior and functions the system

must perform. It covers user requirements, provides explanations for each requirement, and outlines possible scenarios of system behavior in response to user interactions.

### FR1: User Registration

**EARS:** The system shall allow a new user to register by providing their first name, last name, date of birth, a unique username, email address and password.

**Gherkin:**

**Scenario:** Successful Registration

Given the user has entered a unique username and valid credentials

When the user clicks on the register button.

Then the user will be redirected to the Login page

And an email shall be sent notifying the user about their successful registration.

**Scenario:** Unsuccessful Registration

Given the user has entered a non-unique username or invalid credentials

When the user clicks on the register button

Then the user will be notified that there is an error in the data they provided

### FR2: User Login

**EARS:** The system shall allow a user to login by providing their username and password.

**Gherkin:**

**Scenario:** Successful Login

Given the user has entered the correct username and password

When the user clicks on the login button

Then the user will be redirected to the Dashboard Page

**Scenario:** Unsuccessful Login

Given the user has entered incorrect credentials

When the user clicks on the login button

Then the user will be notified that there is an error in the data they have provided and

the user will stay in the Login Page.

### FR3: Add Task

EARS: The system shall allow a user to add a Task by providing Task details such as title, description, priority, category, deadline, and reminder.

**Gherkin:**

**Scenario:** Successful Addition of Task

Given the user has entered the task details (title, description, priority, category,

deadline, and reminder)

When the user clicks on the “Add Task” button

Then the user will be redirected to the Dashboard Page and the Task will appear on

Dashboard Page, Task Page, Schedule Page and Analytics Page.

**Scenario:** Task collides with pre-existing Tasks

Given the user has entered a date and timing that overlaps with a previously added

Task.

When the user clicks on the "Add Task" button

Then the system notifies the user about the overlapping task

And the user chooses to either confirm or reject adding the overlapping task

When the user confirms

Then the task is added

And the user is redirected to the Dashboard page

And the task appears on the Dashboard page, Task page, Schedule page, and Analytics

page

When the user rejects

Then the task is not added

And the user remains on the Add Task page to modify the task details

### FR4: Edit Task

**EARS**: The system shall allow a user to edit a Task by providing Task details such as title, description, priority, category, deadline, and reminder.

**Gherkin:**

**Scenario:** Successful edit of Task

Given the user has modified the task details (title, description, priority, category,

deadline, and reminder.)

for editing

When the user clicks on the “Edit Task” button

Then the user will be redirected to the Dashboard Page and the updated Task will

appear on Dashboard Page, Task Page, Schedule Page and Analytics Page.

### FR5: Delete Task

**EARS**: The system shall allow a user to delete a Task.

**Gherkin:**

**Scenario:** Successful Deletion of Task

Given the user chooses a Task for deletion

When the user clicks on the “Delete Task” button

Then the user will be redirected to the Dashboard Page and the Task will be removed

from the Dashboard Page, Task Page, Schedule Page and Analytics Page.

### FR6: View Tasks

**EARS:** The system shall allow a user to view the following:

* View All tasks
* View Today’s tasks
* View Upcoming tasks
* View Completed tasks

**Gherkin:**

**Scenario:** Successful view of Task

Given the user chooses a Task for viewing

When the user clicks on the “Tasks” button

Then the user will be redirected to the Task Page.

When the user selects the "All" view

Then the user sees all their scheduled tasks

When the user selects the "Today" view

Then the user sees tasks scheduled for today

When the user selects the "Upcoming" view

Then the user sees tasks scheduled for upcoming dates

When the user selects the "Completed" view

Then the user sees all completed tasks

### FR7: View Analytics

**EARS:** The system shall allow the user to view their weekly/monthly/quarterly

progress.

**Gherkin:**

**Scenario:** Successful view of the progress

Given the user chooses to view their task analytics

When the user clicks on the “Analytics” button

Then the user is redirected to the Analytics Page.

When the user selects a particular view (weekly/monthly/quarterly)

Then the user sees their progress through charts/graphs, Task completion rate,

productivity tips etc.

### FR8: Manage Profile

**EARS:** The user shall be able to edit their profile information and other preferences

for using the application.

**Gherkin:**

**Scenario:** Successful change in user settings

Given the user chooses to view their Settings

When the user clicks on the “Settings” button

Then the user is redirected to the Settings Page which offers different settings

(Profile/ Preference/ Notifications/ Account).

When the user clicks on some settings and modifies then clicks “save changes” button

Then the users settings are edited.

### FR9: View Schedule

**EARS:** The user shall be able to view their daily/ weekly/monthly schedule

**Gherkin:**

**Scenario:** Successful view of schedule

Given the user chooses to view their schedule

When the user clicks on the “Schedule” button

Then the user is redirected to the Schedule Page which offers different schedule

views (Daily/ Weekly/ Monthly).

### FR10: Add Pomodoro Timer

**EARS:** The user shall be able to add a Pomodoro Timer to allow breaks between an activity.

**Gherkin:**

**Scenario:** Successful addition of the timer

Given the user chooses to add ‘Pomodoro Timer’ from their Dashboard

When the user clicks on the “Add PD Timer” button

Then the user can view a 25-minute countdown timer on the screen

And after 25 minutes, a 5-minute break timer is automatically started

And an alarm rings to indicate the start of the break

And after 5 minutes, another alarm rings to prompt the user to resume their activity

And after 4 such cycles, a long break of 15–30 minutes is offered

### FR11: User Logout

**EARS:** The user shall be able to logout of the system and will be redirected to the homepage.

## 3.2 Non Functional Requirements

This section outlines the quality attributes the system must meet, including usability,

performance, scalability, security, maintainability, and portability.

### Usability

The system shall provide a clean user interface. New students should be able to use the application without any prior training and complete registration and login within 3 minutes.

1. **Performance**

The system shall respond to 95% of request in within 2 seconds.

1. **Scalability**

The system shall support a minimum of 1000 concurrent users, maintaining performance levels defined in the previous section (Performance).

1. **Security**

* The system shall use hashed passwords to protect the user passwords.
* The system shall not allow unauthorized users to access the Admin (API). Only authenticated users with appropriate roles shall have access to the admin site.

1. **Maintainability**

* The codebase shall follow Django development best practices
* At least 80% of the codebase shall be documented using inline comments.

1. **Portability**

The system shall be fully functional on the latest two versions of major web browsers, including Chrome, Firefox, Safari, and Microsoft Edge. It shall be responsive and usable on desktop or laptop screen resolutions ( 1024 px width).

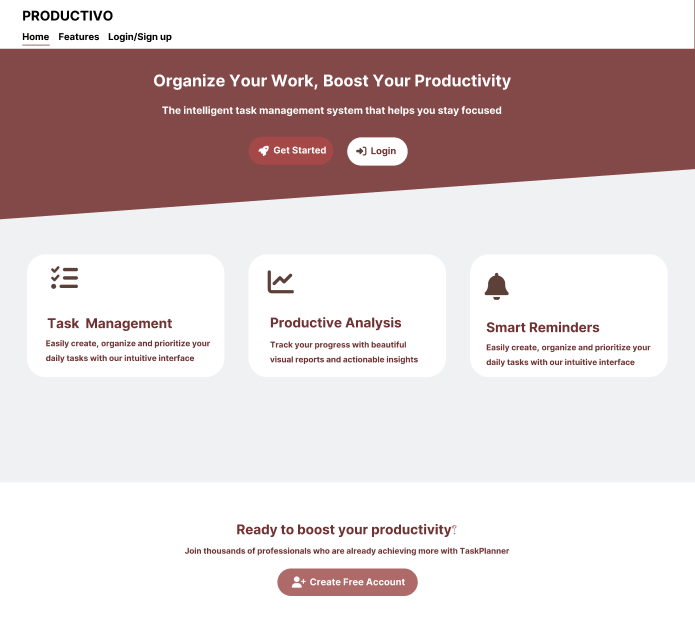
## 3.3 Attributes of the Requirements

| **Requirement Number** | **Priority** | **Status** |
| --- | --- | --- |
| FR1-User Registration | High | Approved |
| FR2-User Login | High | Approved |
| FR3-Add Task | High | Approved |
| FR4-Edit Task | Medium | Approved |
| FR5-Delete Task | Medium | Approved |
| FR6-View Tasks | High | Approved |
| FR7-View Analytics | Low | Approved |
| FR8-Manage Profile | Low | Approved |
| FR9-View Schedule | Medium | Approved |
| FR10-Add Pomodoro Timer | Low | Approved |
| FR11-User Logout | High | Approved |
| NFR1-Usability | High | Approved |
| NFR2-Performance | High | Approved |
| NFR3-Scalability | Medium | Approved |
| NFR4-Security | High | Approved |
| NFR5-Maintainability | Medium | Approved |
| NFR6-Portability | Medium | Approved |

**Note:** All requirements are sourced from discussions with teammates based on the requirements of the CSE327 project, as of August 4th 2025.

## 3.4 External Interface Requirements

### 1. User Interface

The system provides a web-based user interface accessible through a browser on devices connected to the local network. The interface includes pages for user login, registration, dashboard, and profile management. It follows responsive design principles for usability on standard desktop/laptop screens.









### 

### 

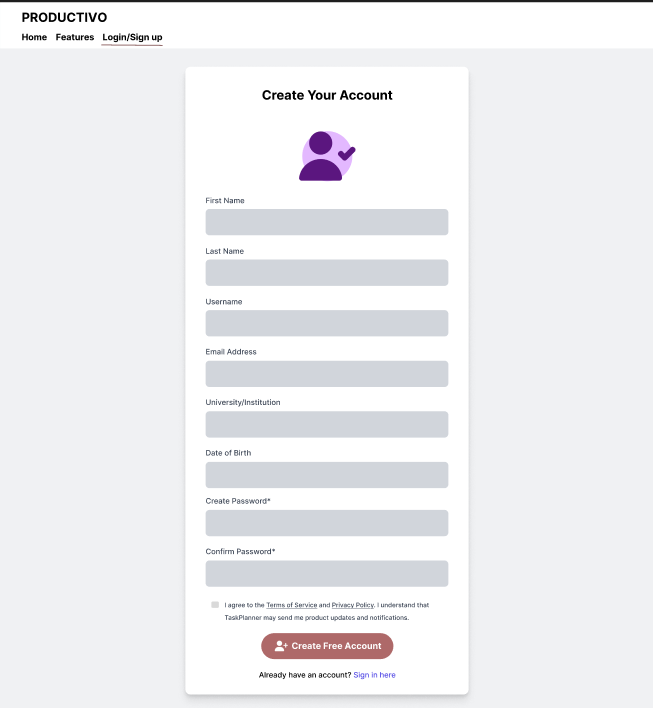
### 

### 

### 

### 

### 



### 

### 

### 

### 

### 

### 

### 2. Hardware Interfaces

Not applicable since system does not interact with any external hardware components.

### 3. Software Interfaces

The system does not integrate with any external software systems or third-party APIs. It uses SQLite as the internal database backend, which is managed locally within the Django environment.

### 4. Communication Interfaces

The system is hosted locally using Django’s built-in development server and is accessible over the internal network. All communication between client browsers and the server occurs over HTTP.

# 4. Other Requirements

## 4.1 Database Requirements

The system uses SQLite as the backend database during development. It stores user credentials, course information, and other application data in a relational format. The Django Object Relational Mapping is used for database operations.

## 4.2 Legal and Regulatory Requirements

The project is currently developed for academic purpose and therefore does not include any specific legal or regulatory requirements. However, the project adheres to the ethical guidelines related to the production of responsible software.

## 4.3 Internationalization and Localization

Internationalization and Localization are not within the scope of this project. The application is developed to be used in English only.

## 4.4 Risk Management (FMEA Matrix)

Risk Management is not within the scope of this project.

# 5. Appendices

## 5.1 Use Case Diagram

## 

## 

## 5.2 To Be Determined (TBD) List

1. Integration with Google Calendar to Sync Tasks.
2. Mechanism for providing task management suggestions (AI driven or rule based logic)

## 5.3 Contributions in the SRS

* *Tanvir Ahmed (223 1047 642)* – Created the Use Case Diagram
* *Faheema Shaheed Tamanna (223 2144 642)* – Designed the User Interface Mockups
* *Zuhayer Islam (223 2061 642)* –

Wrote several sections of the document, formatted the content, and integrated all components into a cohesive structure. Also ensured that all requirements were clearly understandable and conformed to the standards of a Software Requirements Specification (SRS).

* *Shefa Tabassum (223 2993 042)* –

Refined the Use Case Diagram after team discussion and analysis, contributed to the

functional requirements section and edited the document for clarity, consistency, and

overall quality.

Each team member contributed in their own way, bringing unique strengths to the development and completion of this Software Requirements Specification (SRS) document.

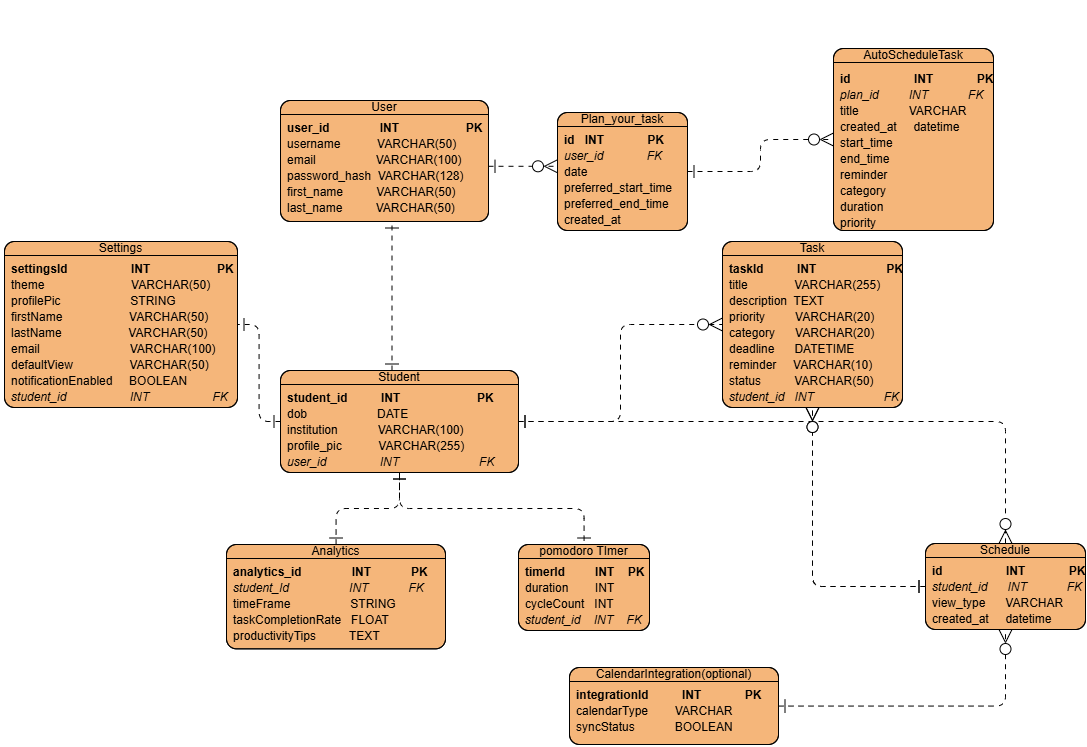
**System Modelling Diagrams**

## Class Diagram



Completed By: *Shefa Tabassum (223 2993 042)*

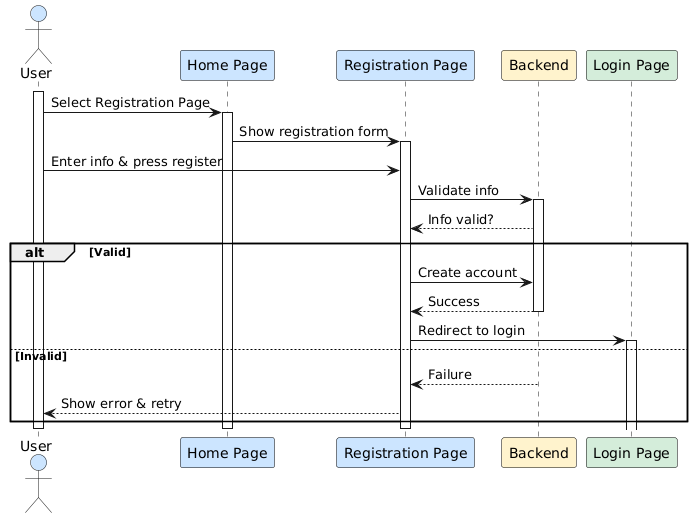
## ER Diagram (Database Design)



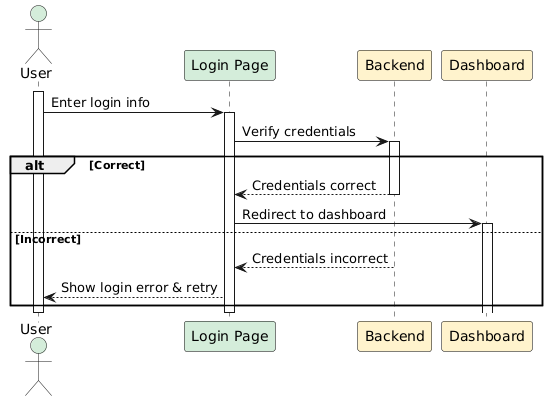
Completed By: *Shefa Tabassum (223 2993 042)*

## Sequence Diagram

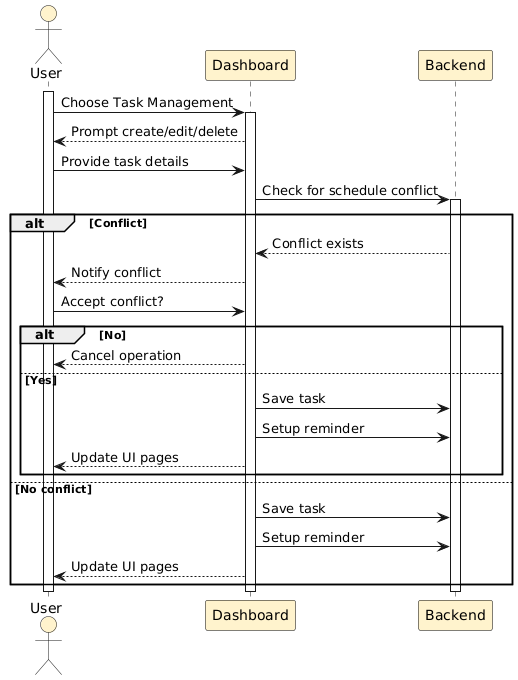
**3.1. Registration flow**



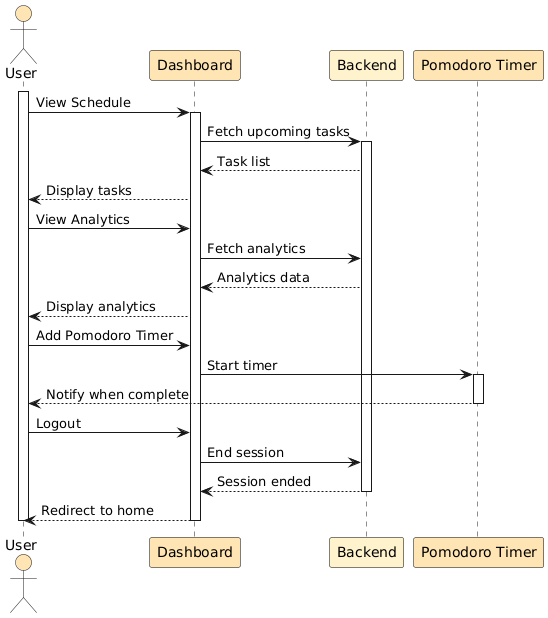
**3.2. Login flow**



**3.3. Task management**



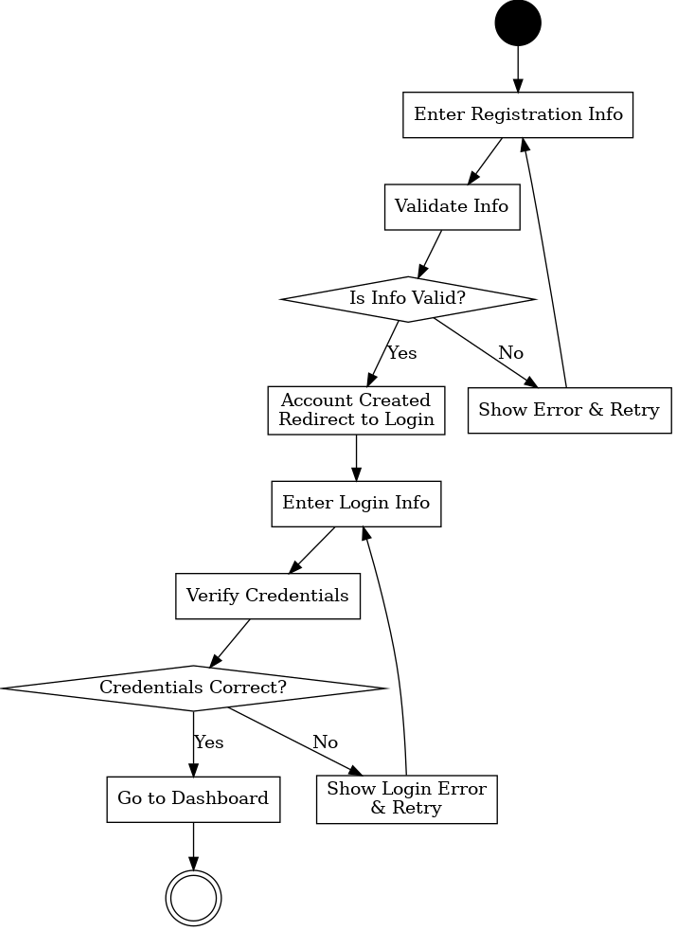
**3.4. Other Dashboard Actions**



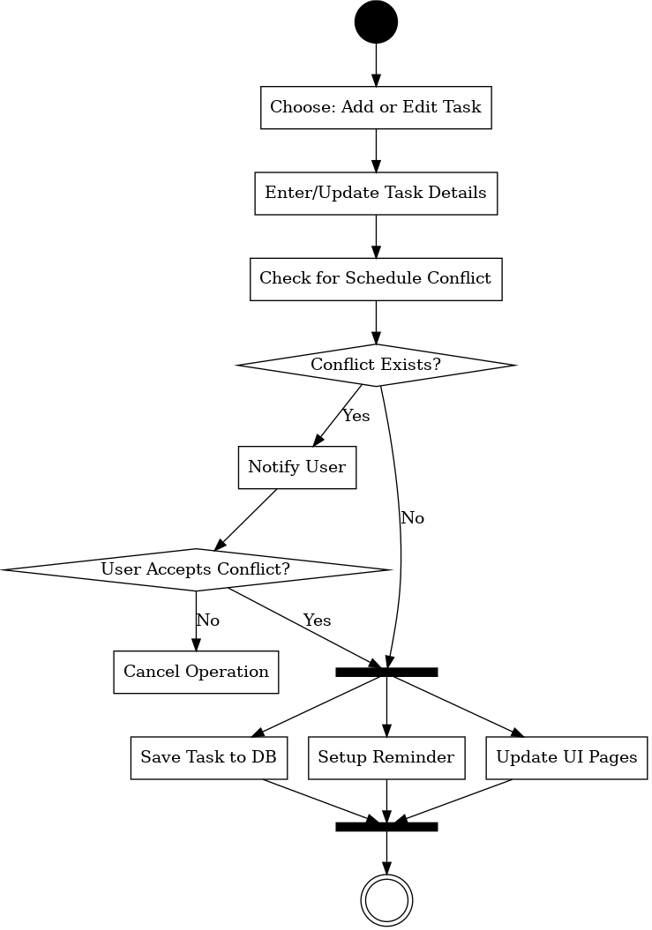
Completed By: *Faheema Shaheed Tamanna (223 2144 642)*

## Activity Diagrams

### 4.1 Activity Diagram for Registration/Login

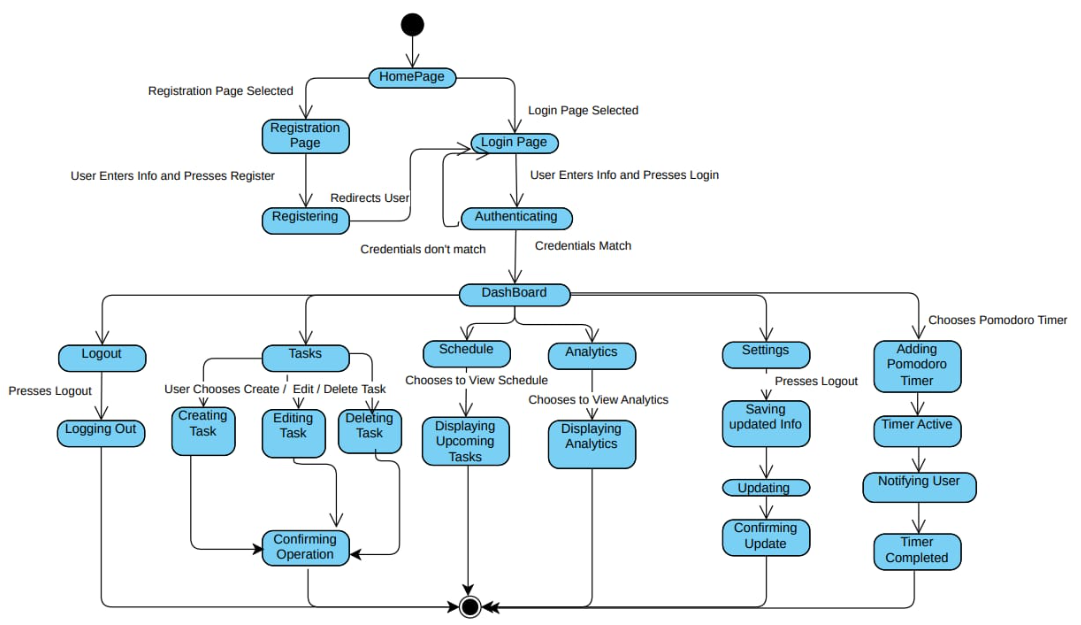


### Add/Edit Task



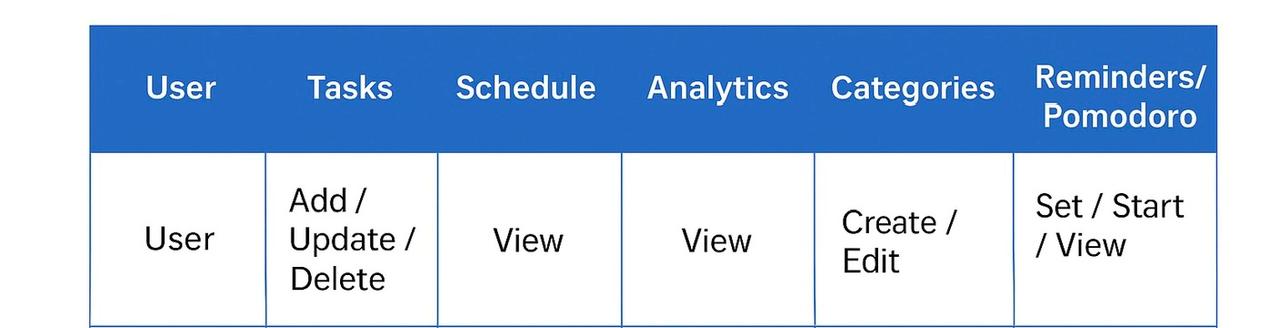
Completed By: *Tanvir Ahmed (223 1047 642)*

## State Machine Diagram



Completed By: *Zuhayer Islam (223 2061 642)*

## 6. User Access Control



## 7. Project Scheduling (using CPM)

| **Activity** | **Immediate Predecessor** | **Duration (Days)** |
| --- | --- | --- |
| Requirement Analysis | – | 3 |
| System Design | Requirement Analysis | 3 |
| Database Design | System Design | 2 |
| Frontend Development | System Design | 4 |
| Backend Development | Database Design | 5 |
| Task Management Module (Add/Edit/Delete) | Backend Development, Frontend Development | 4 |
| Schedule Module | Task Management Module | 3 |
| Analytics Module | Schedule Module | 3 |
| Profile & Settings | Backend Development | 2 |
| Pomodoro Timer & Reminders | Task Management Module | 2 |
| Integration & Testing | All Modules | 4 |
|  |  |  |

## 

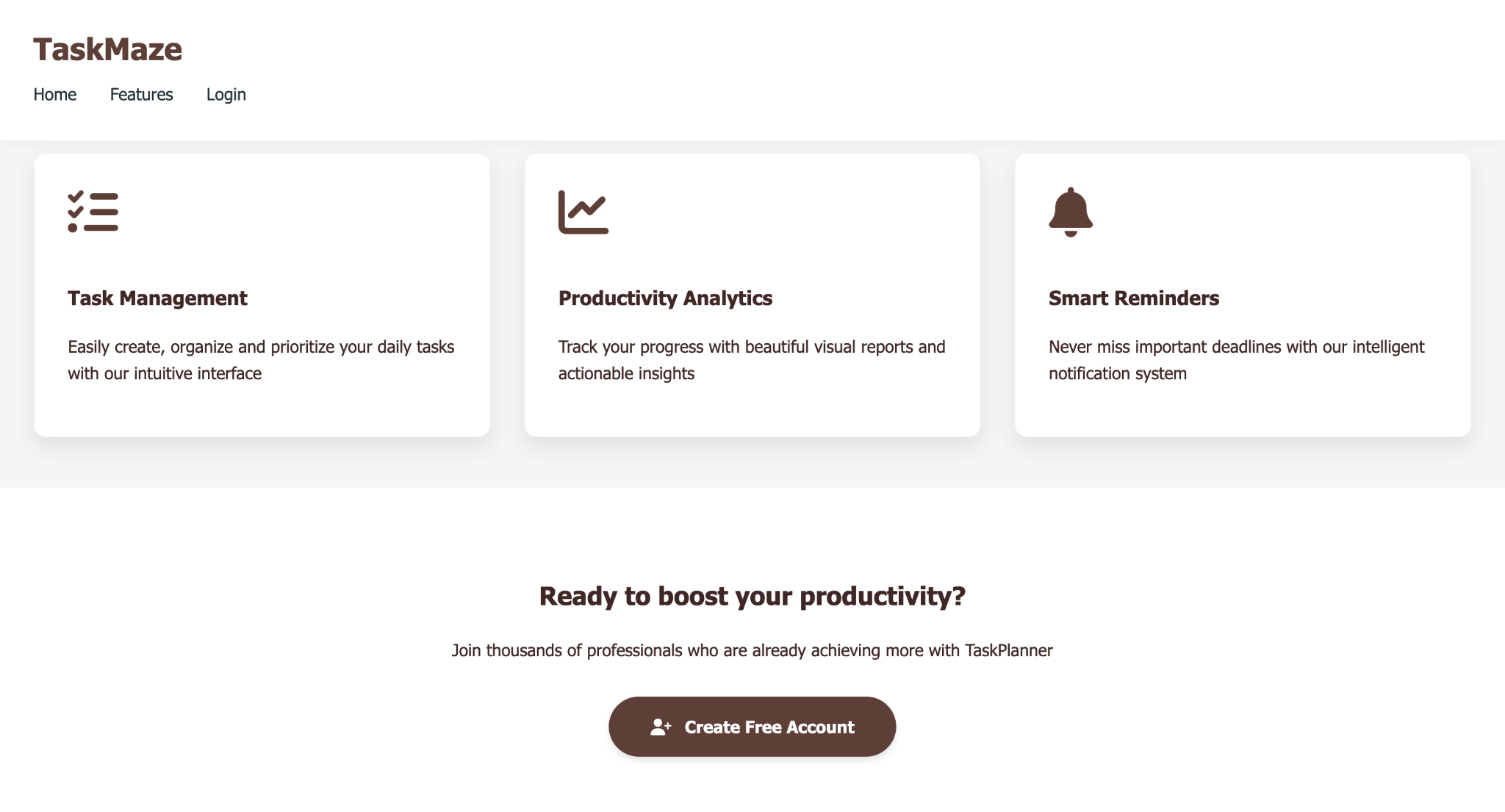
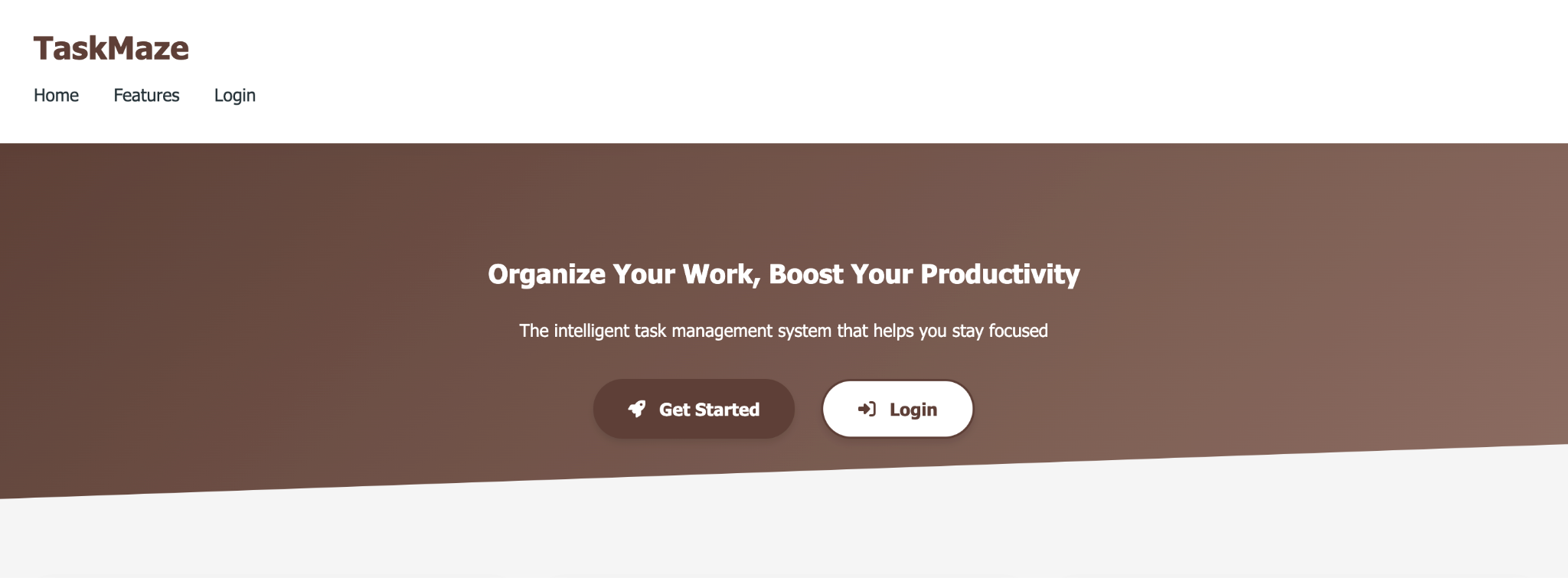
## 

The total project duration is **27** days. The critical path can be measure by observing the functionalities with 0 slack. So, our critical path is:

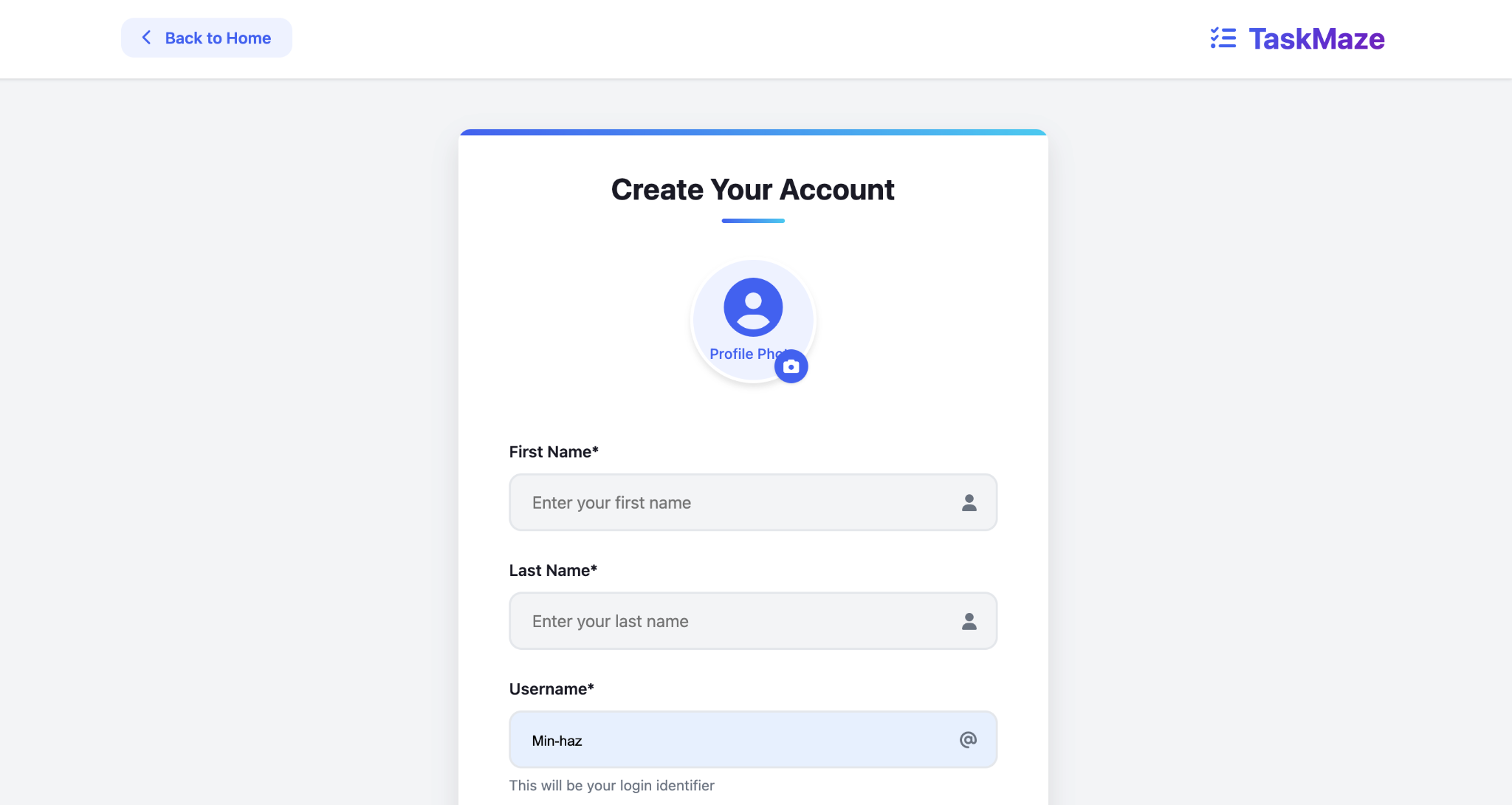
Requirement Analysis → System Design → Database Design → Backend Development → Task Management Module → Schedule Module → Analytics Module → Integration & Testing

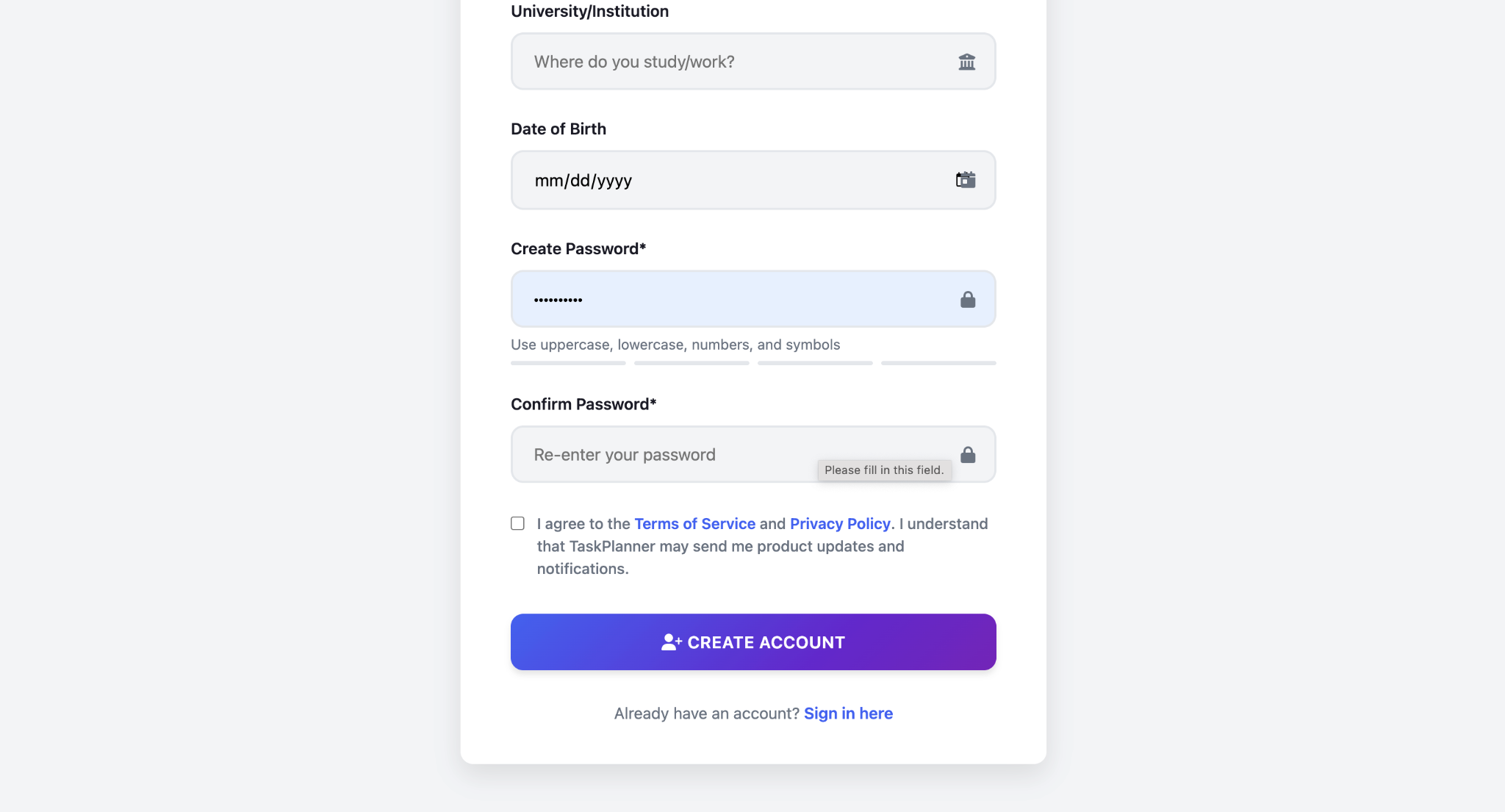
## 

## 8. Functionality-wise System Description (with screenshots)

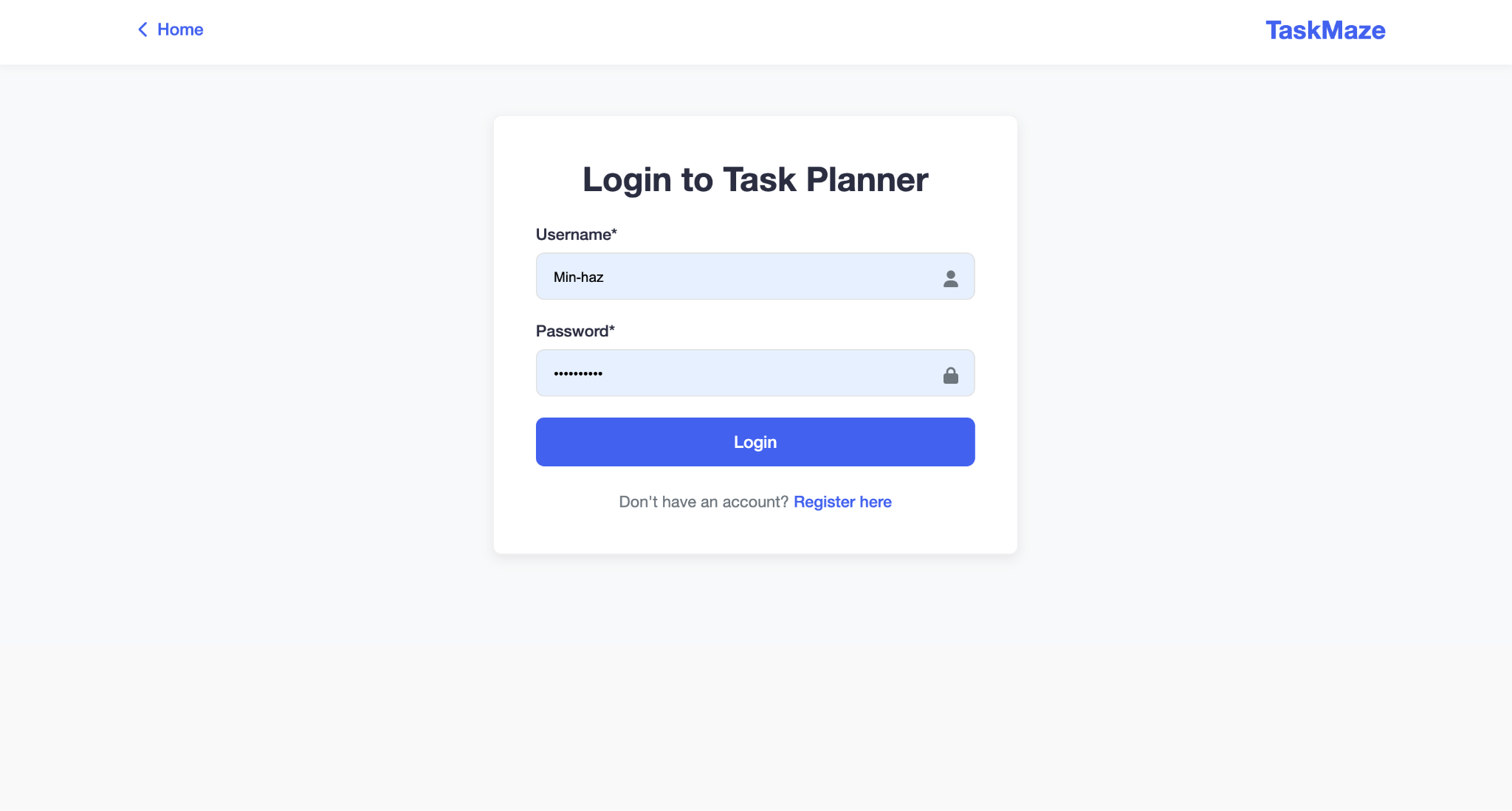


The screenshot above shows the landing page of the website. Users can look into the functionalities provided by the software and create an account to get started.

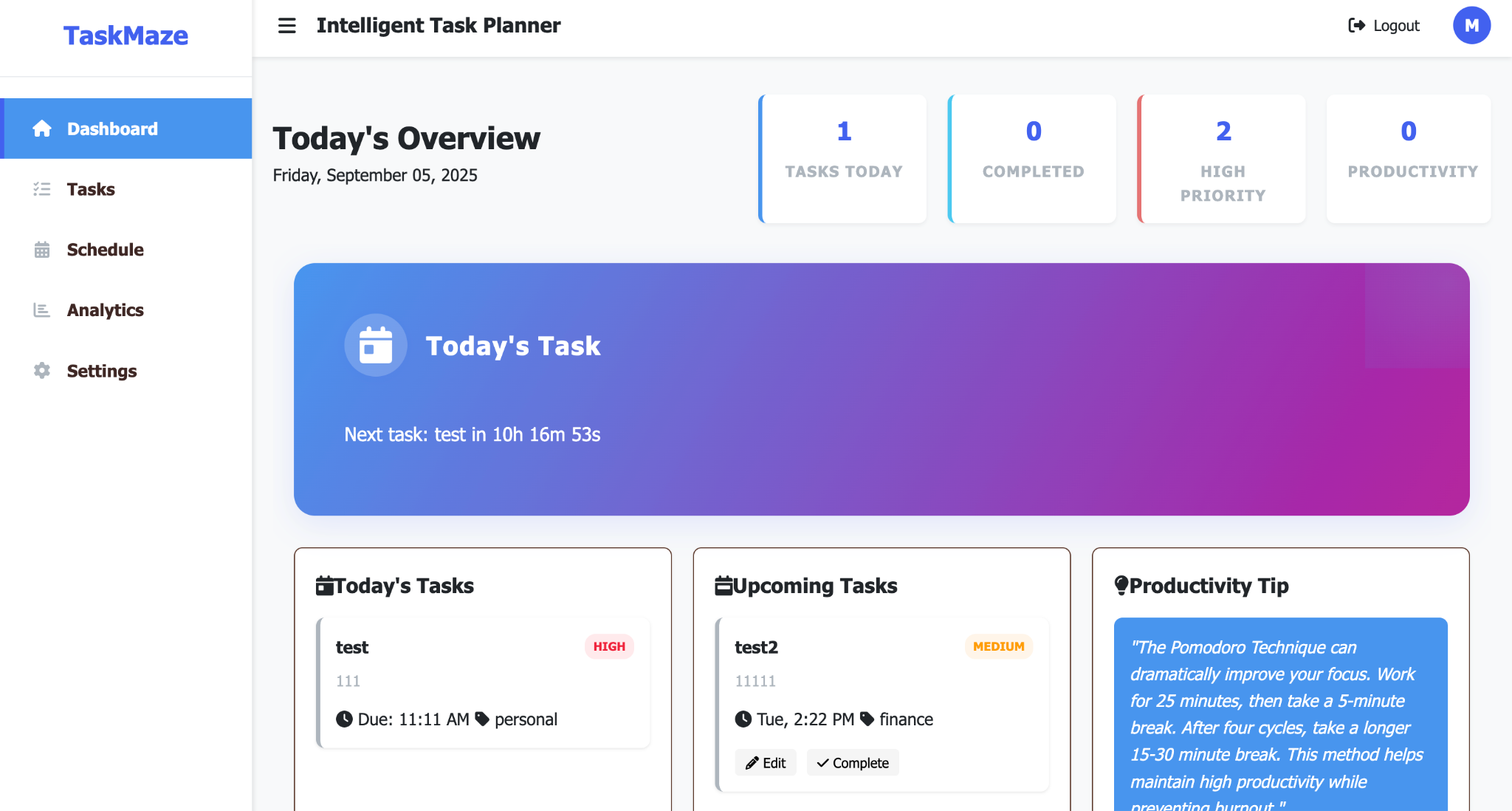




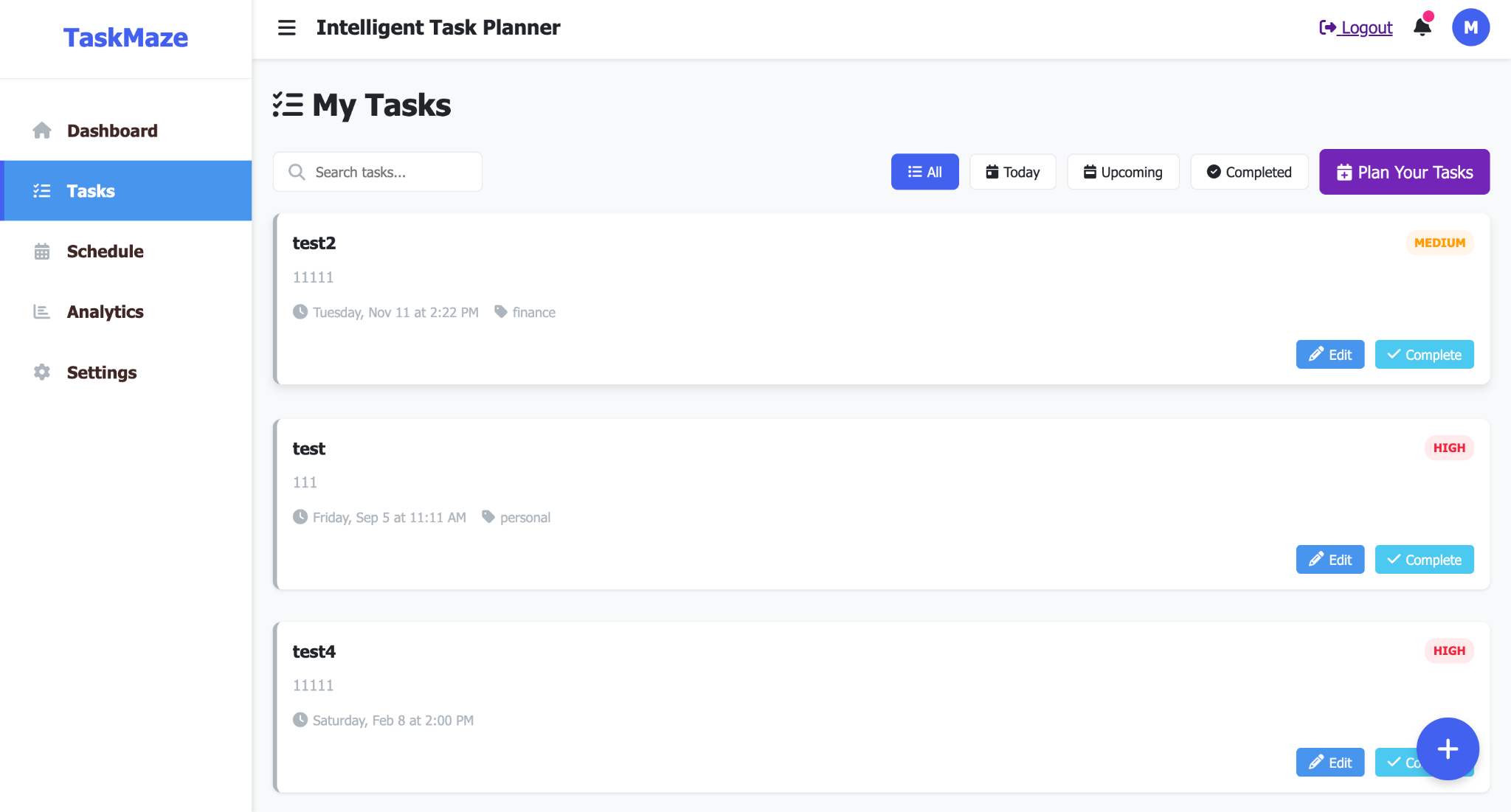
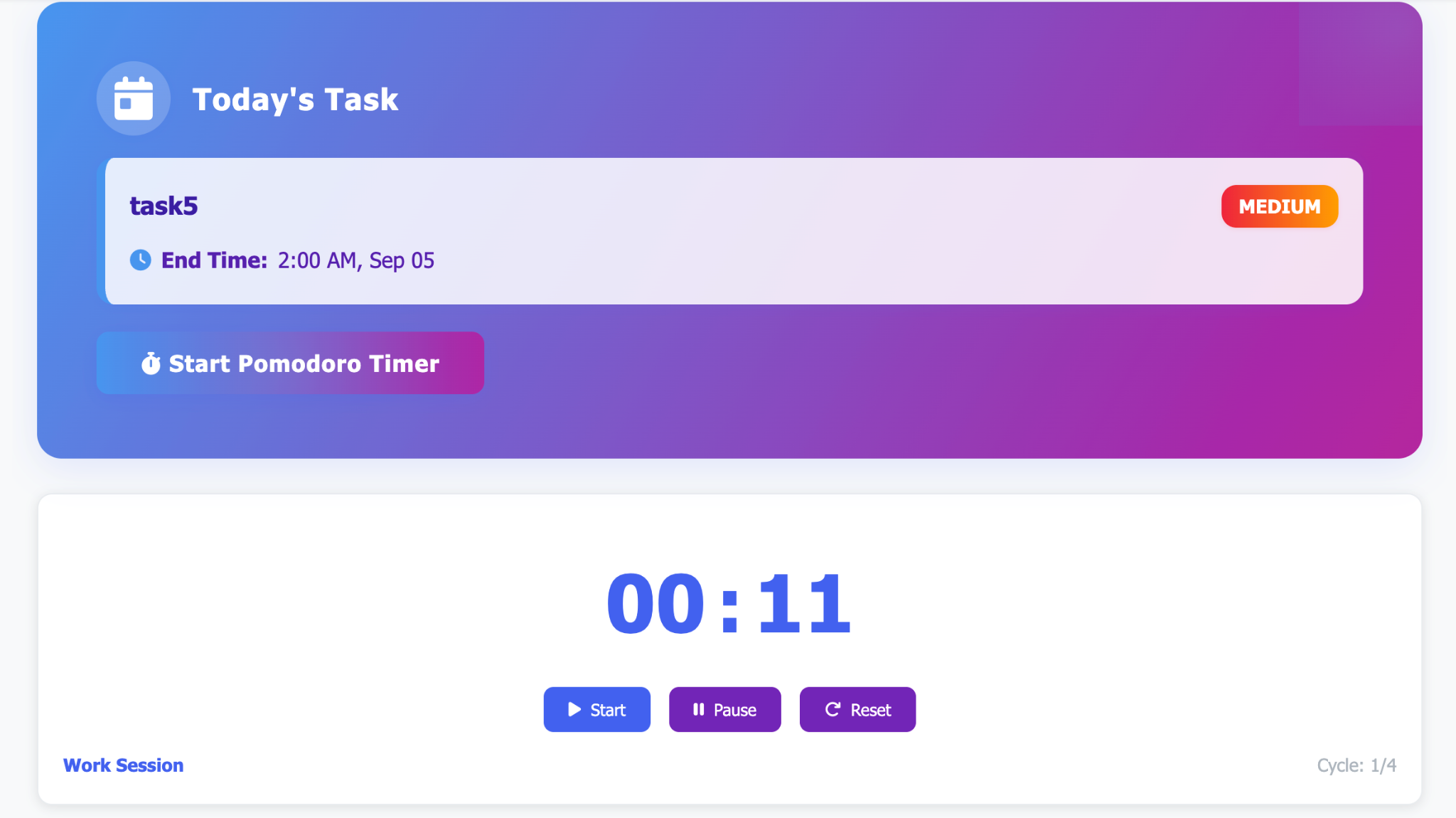
In the account creation page, the user enters their details and a unique username along with a strong password meeting the requirements of the application. The password strength is checked by the system. Once the user fills up all the fields correctly and clicks on create account, the user is mailed (using Django’s send mail function), confirming that their account has been created.



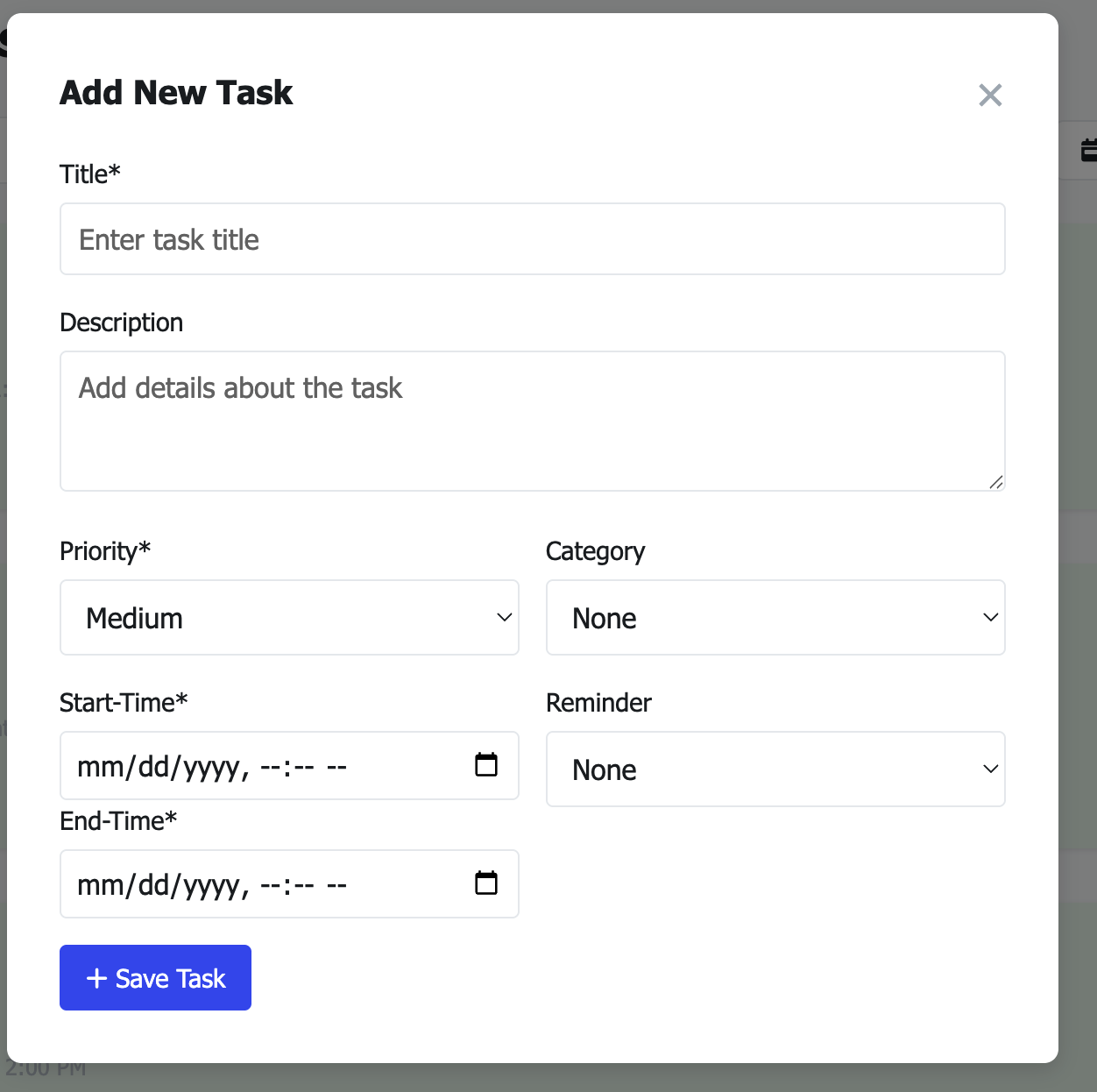
The user can log in providing their credentials which are validated by the system. If the credentials are correct the user is allowed to login. This validation was achieved using Django’s authenticate() function.



Once the student logs in successfully, the student is redirected to the Dashboard where the student can view their tasks for the day, upcoming tasks, along with the count of the productivity and high priority tasks. The user can also view the countdown to the next task.



Once tasks are added to the task page and a task is currently running, the Pomodoro timer can be added. The timer allows users to manage times more efficiently. The user can see a timer run for 25 minutes in which they carry out the task. After the 25 minutes time ends a short break of 5 minutes is provided. After 4 working cycles, a longer break of 15 minutes is provided. In this way the user can stop over working and manage the task efficiently. The cycle continues until the Task time is completed.



### Add New Task Form – Functional Description

This form is part of the task management system and is used to **create a new task**. It appears as a modal popup when the user clicks the "Add Task" button. The form allows the user to input essential information about the task, including its timing, priority, and optional details like category and reminders. Once submitted, the data is validated and sent to the backend via AJAX for storage in the database.

#### Input Fields:

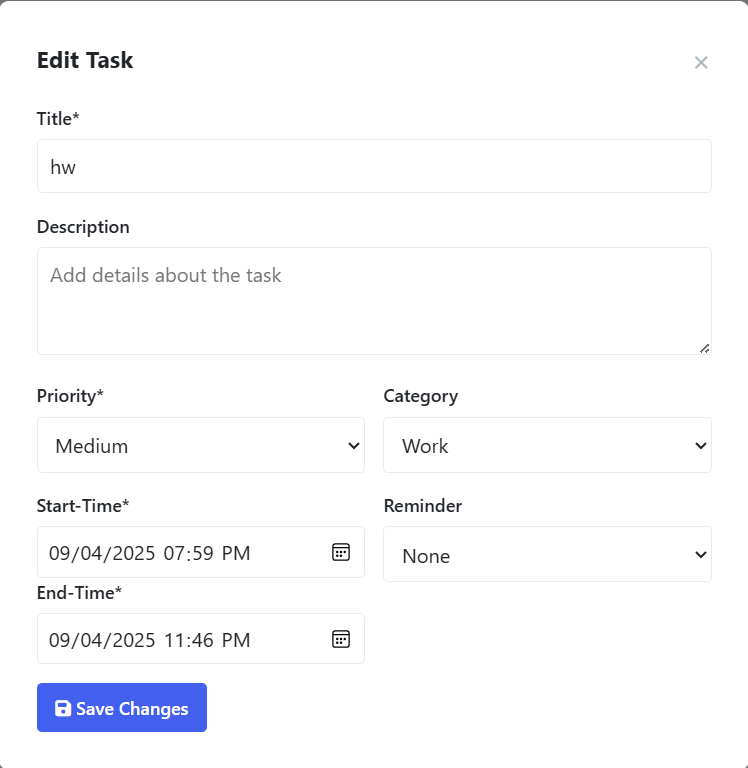
1. **Title (Required)** A short title for the task. This is a mandatory field and serves as the identifier for the task in the task list.
2. **Description (Optional)** A text area for users to provide more details or notes about the task.
3. **Priority (Required)** A dropdown menu allowing the user to select the task's urgency level. The options are typically:
   * Low
   * Medium (default)
   * High
4. **Category (Optional)** A dropdown to classify the task under predefined categories like Work, Personal, Finance, etc.
5. **Start Time (Required)** A date-time input field where the user selects when the task is supposed to start. This must be filled for the task to be saved.
6. **End Time (Required)** Another date-time input where the user specifies when the task should end. It must be later than the start time.
7. **Reminder (Optional)** A dropdown that allows users to set an automatic reminder before the task's end time. Options include 15 minutes, 30 minutes, 1 hour, 1 day, or none.

#### Submit Button:

* **+ Save Task** This button triggers the form submission. Once clicked:  
  + Client-side validation checks for required fields and logical consistency (e.g., end time must be after start time).
  + If valid, the form data is sent via AJAX to the backend.
  + The backend also checks for time conflicts (task clashes) before saving the task.
  + If the task is successfully added, the modal closes and the updated task list is shown.

#### Purpose:

The main purpose of this form is to allow users to efficiently and accurately **add new tasks to their schedule**, while preventing invalid or overlapping entries through proper validation both on the client and server sides.



### Edit Task Form – Functional Description

This modal form provides users with an interface to **update an existing task**. It is triggered when the user selects an edit option for a specific task from the task list. The form is pre-populated with the task's current details and allows the user to modify any field as needed. Upon submission, the updated data is validated and sent to the backend via an AJAX request.

#### Pre-Filled Fields:

1. **Title (Required)** Displays the current title of the task. The user can modify this to update the task's name.
2. **Description (Optional)** Displays any existing description. Can be edited to update additional task notes or left blank.
3. **Priority (Required)** A dropdown showing the current priority level. Options include Low, Medium, and High. The user can change this value if needed.
4. **Category (Optional)** A dropdown to change the task’s classification. For example, from “Work” to “Personal” or “Study.”
5. **Start Time (Required)** Displays the task's current start time. The user can change it to adjust the schedule.
6. **End Time (Required)** Displays the task's current end time. The user can change this value as long as it is later than the start time.
7. **Reminder (Optional)** Allows changing the reminder setting to notify the user before the task ends.

#### Submit Button:

* **Save Changes** This button submits the modified task data. On click:  
  + The system performs **client-side validation**, including:  
    - Required field checks
    - Ensuring end time is later than start time
    - Ensuring the reminder time (if set) is logically valid

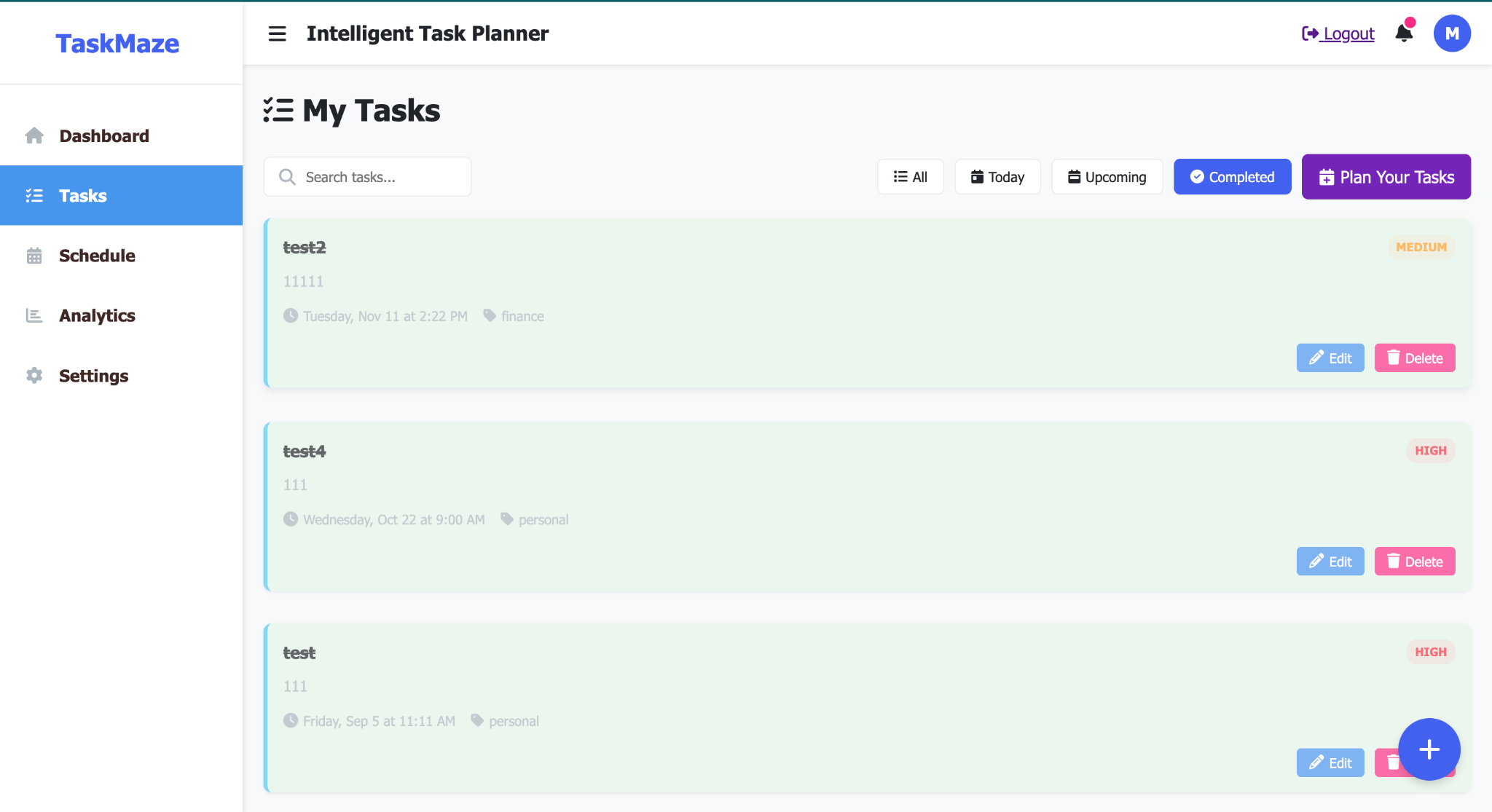
An AJAX POST request is sent to the backend endpoint:  
  
 /edit-task/<task\_id>/

#### Backend Processing:

* The Django view (edit\_task) receives the request and:  
  + Identifies the task using the given task\_id and the logged-in user.
  + Checks whether the new time range overlaps with any other task, excluding the current one.
  + If a clash is detected, it returns a JSON error.
  + Otherwise, it updates the task in the database and returns a success response.

#### Purpose:

The Edit Task form is designed to enable users to modify existing task details without needing to delete and recreate them. It ensures flexibility while maintaining data consistency and conflict-free scheduling, thanks to validation and clash detection logic both on the frontend and backend.



### Task List Page – How Tasks Are Displayed and Updated

This screenshot shows the **main task management interface** where all the user’s tasks are listed and managed. It serves as a dynamic dashboard for viewing, editing, and deleting tasks. Here's how tasks are updated and reflected on this page:

#### 1. Task Display Section

* Each task is shown as a card containing:  
  + **Title** (e.g., "test2", "test4")
  + **Description** (e.g., "1111", "111")
  + **Date & Time** of the task (start time)
  + **Category** tag (e.g., finance, personal)
  + **Priority** badge (e.g., MEDIUM, HIGH)

#### 2. Buttons for Interaction

Each task card includes:

* **Edit Button**
  + Opens the **Edit Task modal**.
  + The form fields are pre-filled with the current task's values.
  + Once the user saves changes:  
    - An AJAX POST request is sent to /edit-task/<task\_id>/.
    - On successful update, the modal closes and the task list reloads automatically using location.reload().
    - The updated task details (e.g., time, title, category) are re-rendered.
* **Delete Button**
  + Sends an AJAX request to delete the task.
  + After deletion, the task card disappears without a full page reload.

#### 3. Filtering and Views

Above the task list, there are control buttons that allow the user to:

* Filter tasks by:  
  + **All**
  + **Today**
  + **Upcoming**
  + **Completed**
* Search for tasks by title using the search bar.
* Launch the **"Plan Your Tasks"** feature.

These controls likely use JavaScript event listeners and trigger AJAX fetches or in-browser filtering to update the task display dynamically without a full page refresh.

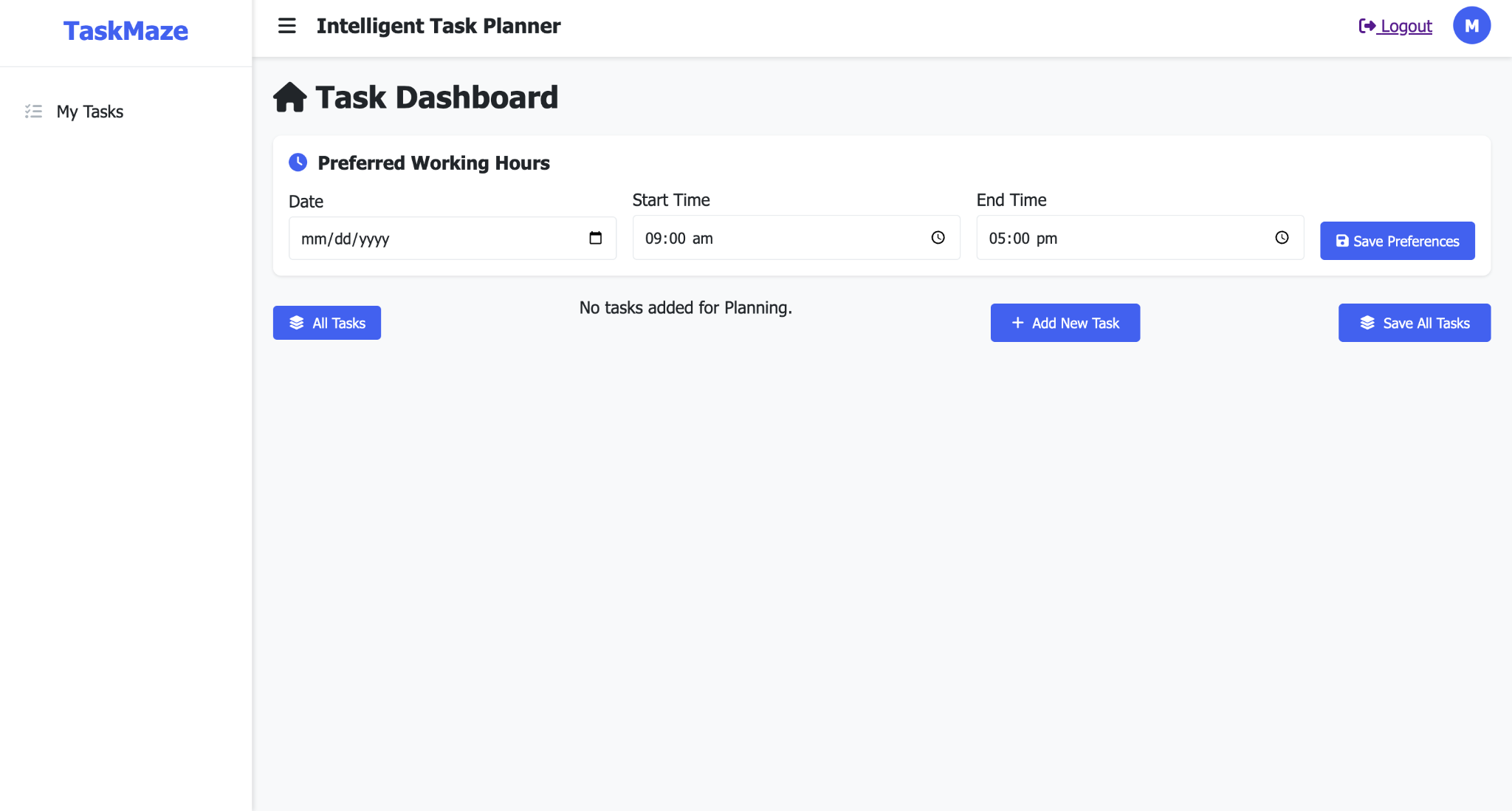
#### 4. Automatic Page Update After Edit

When a task is edited via the modal form:

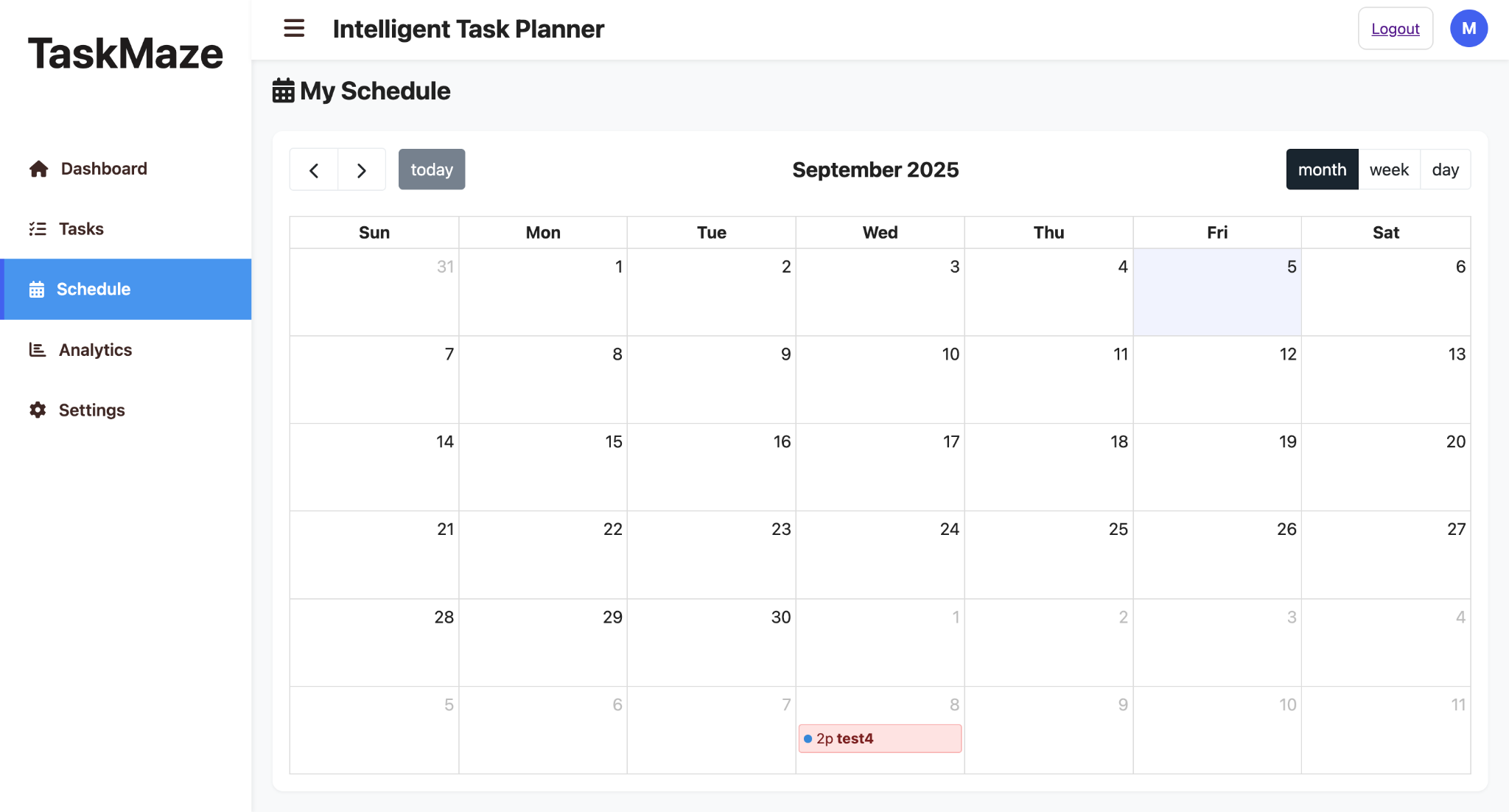
* The frontend JavaScript sends a request and waits for the server response.

If the response contains { success: true }, the page is automatically refreshed with:  
  
 location.reload();

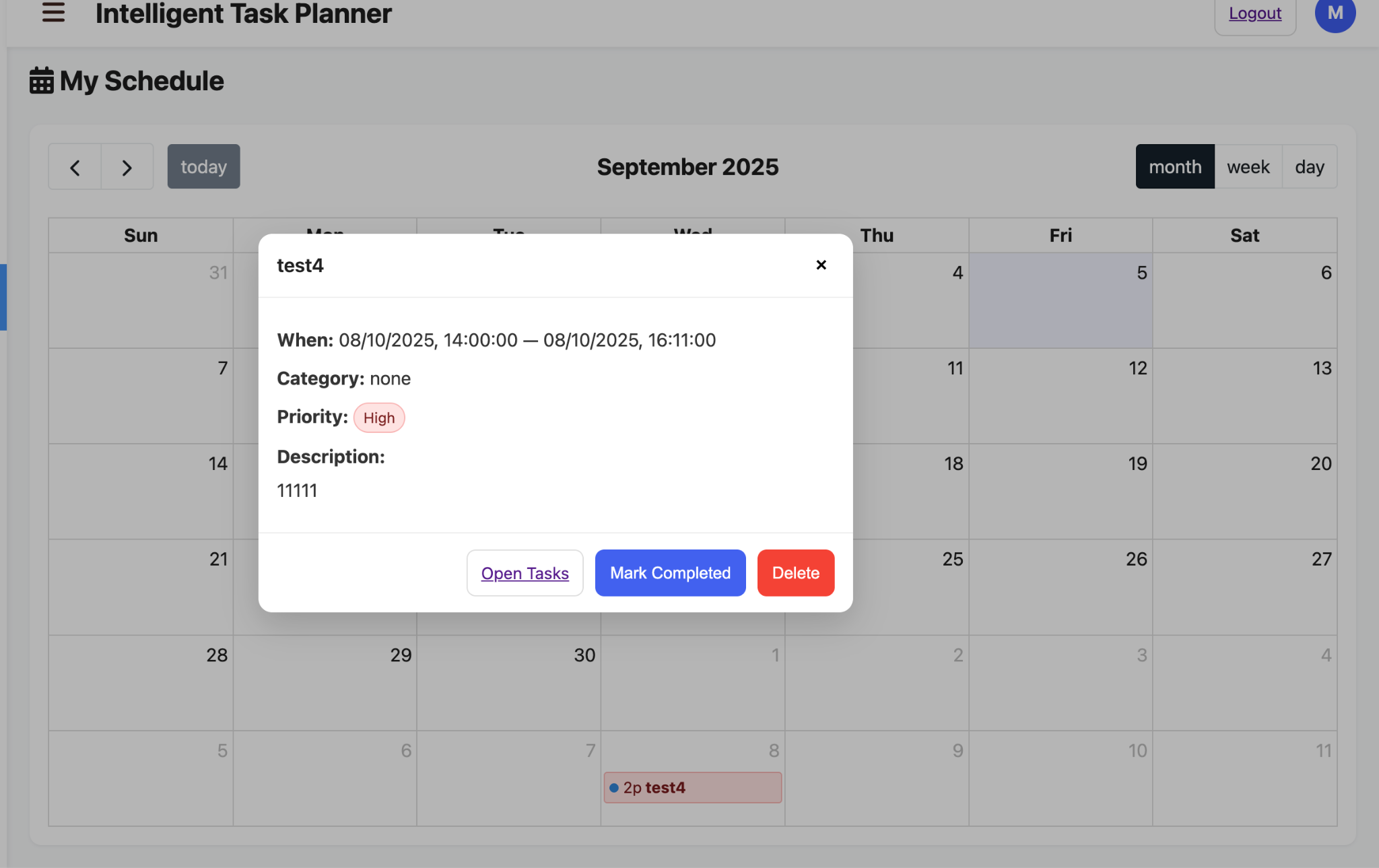
This reload causes the updated task list to be fetched from the backend and re-rendered, so the user immediately sees the new values.



The Intelligent Task Planner provides the smart task planning option for a day. The user chooses the date and preferred start and end time for the day they want to plan their tasks for. Once the preferences are saved, the user can add tasks either flexible (with only duration) or fixed tasks (which have a fixed start and end time). After the tasks have been added, the user can click on AutoSchedule. The software automatically schedules each task scheduling the flexible tasks by priority (high priority first) next by duration (low duration task scheduled earlier) and lastly by historic task completion time, (i.e. if a user completed more tasks at 9 pm then the software schedules the high priority task at that time so that the user has greater chance of completing the task). The fixed tasks’ time are kept as it is.



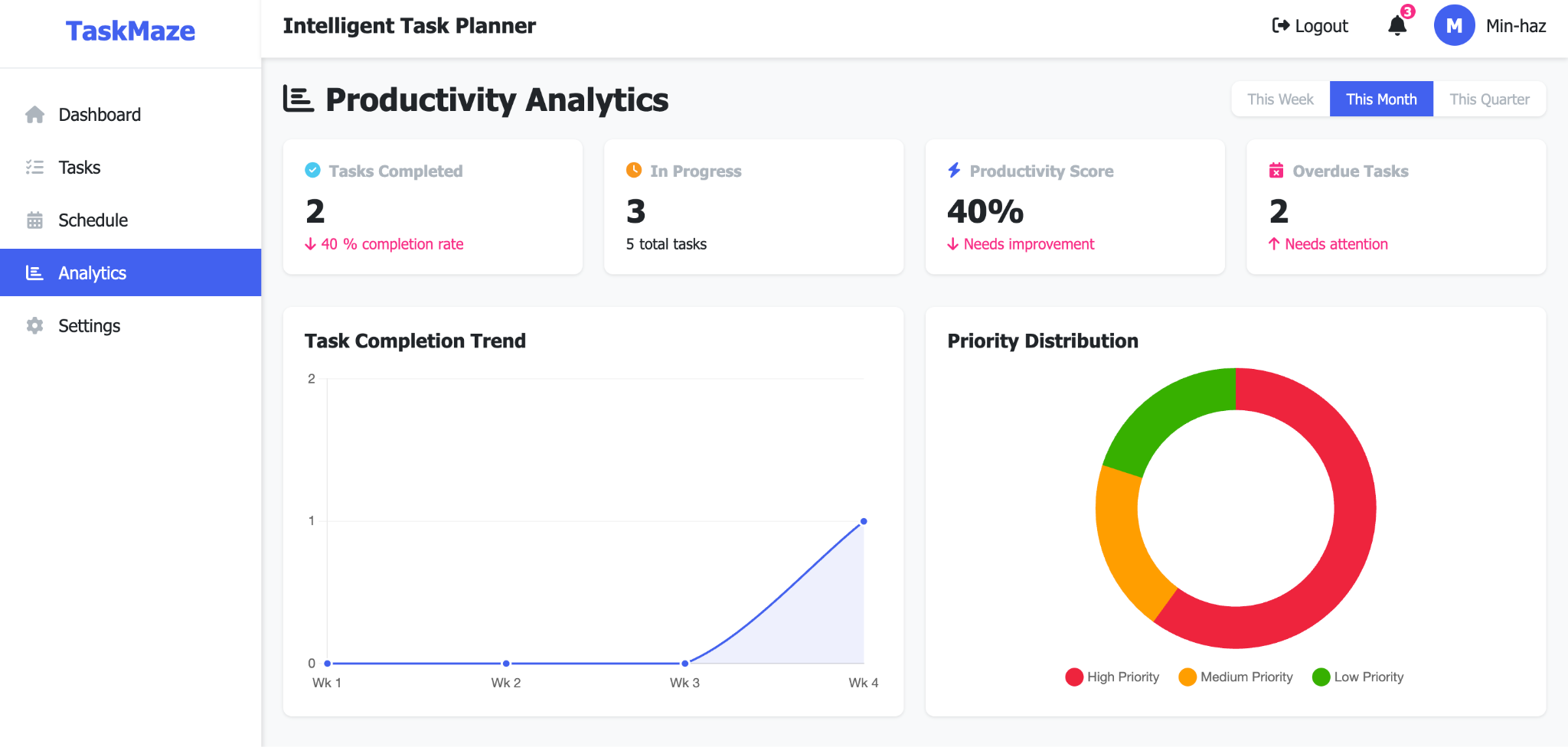
The **Schedule** page allows the user to view their tasks on the calender. It shows each task with colours indicating their priority level.



The task can also be viewed, marked complete or deleted by clicking on it on the calender.

### Analytics Page – Functional Description

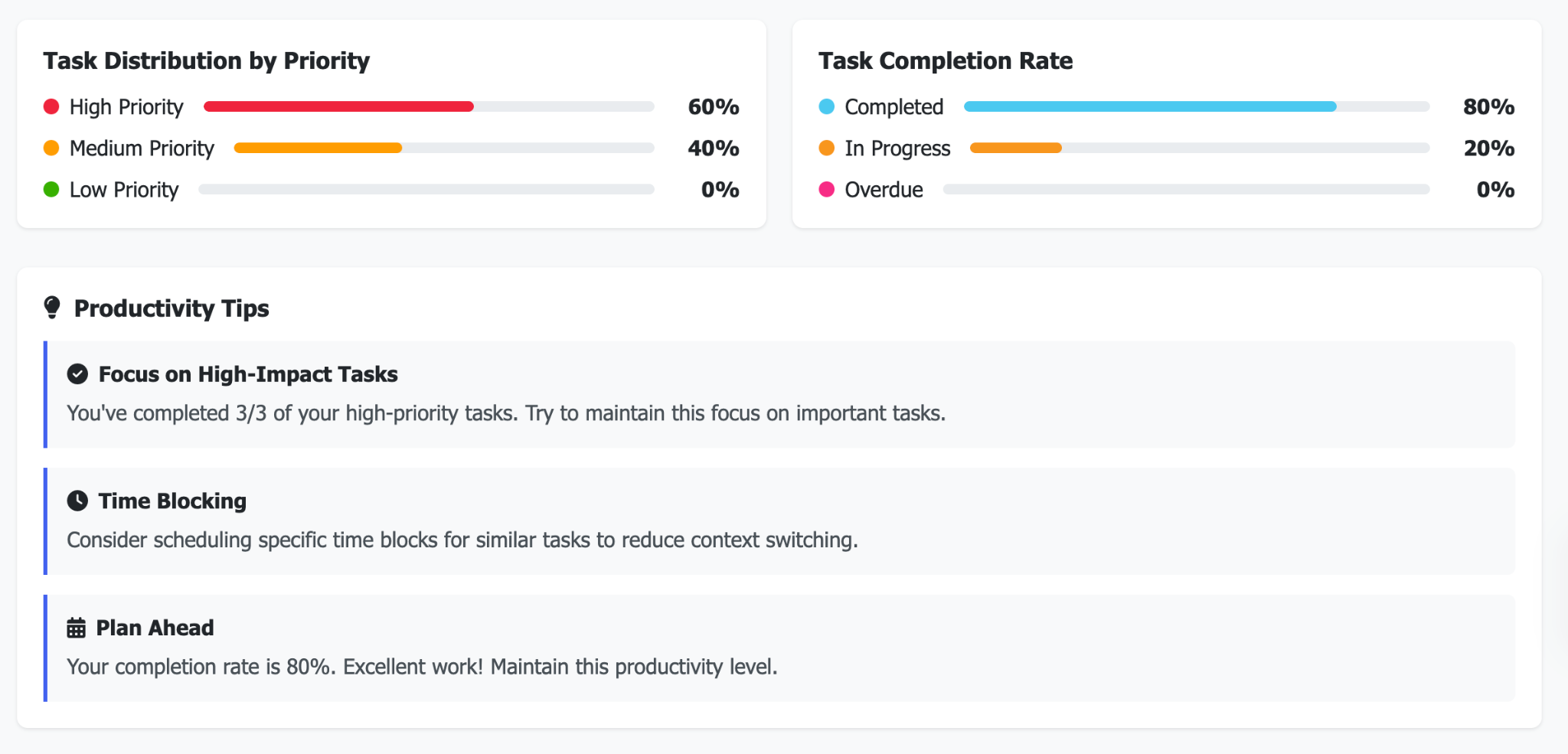
The **Analytics page** provides users with clear productivity insights through visual data and performance metrics. It transforms raw task data into actionable intelligence, helping users spot patterns, track progress, and improve task management.



### Key Components

1. **Productivity Overview**
   * Tasks Completed
   * In Progress
   * Productivity Score
   * Overdue Tasks
2. **Interactive Visualizations**
   * Completion Trend Chart
   * Priority Distribution Chart
3. **Task Distribution Analysis**
4. **Personalized Productivity Tips**
   * Contextual recommendations based on current performance
   * Actionable suggestions for better task focus and efficiency

### 



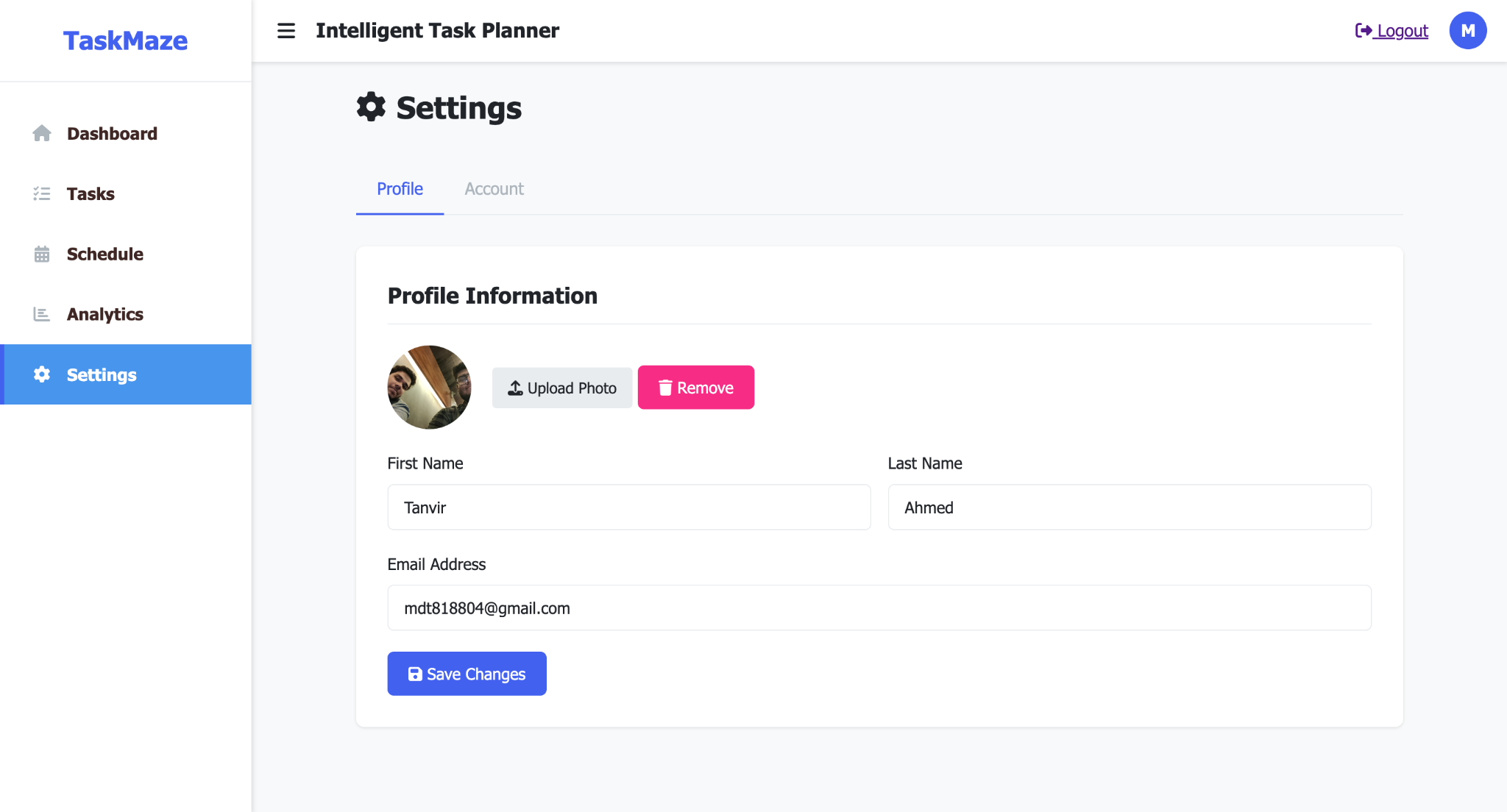
### Backend (Django Views)

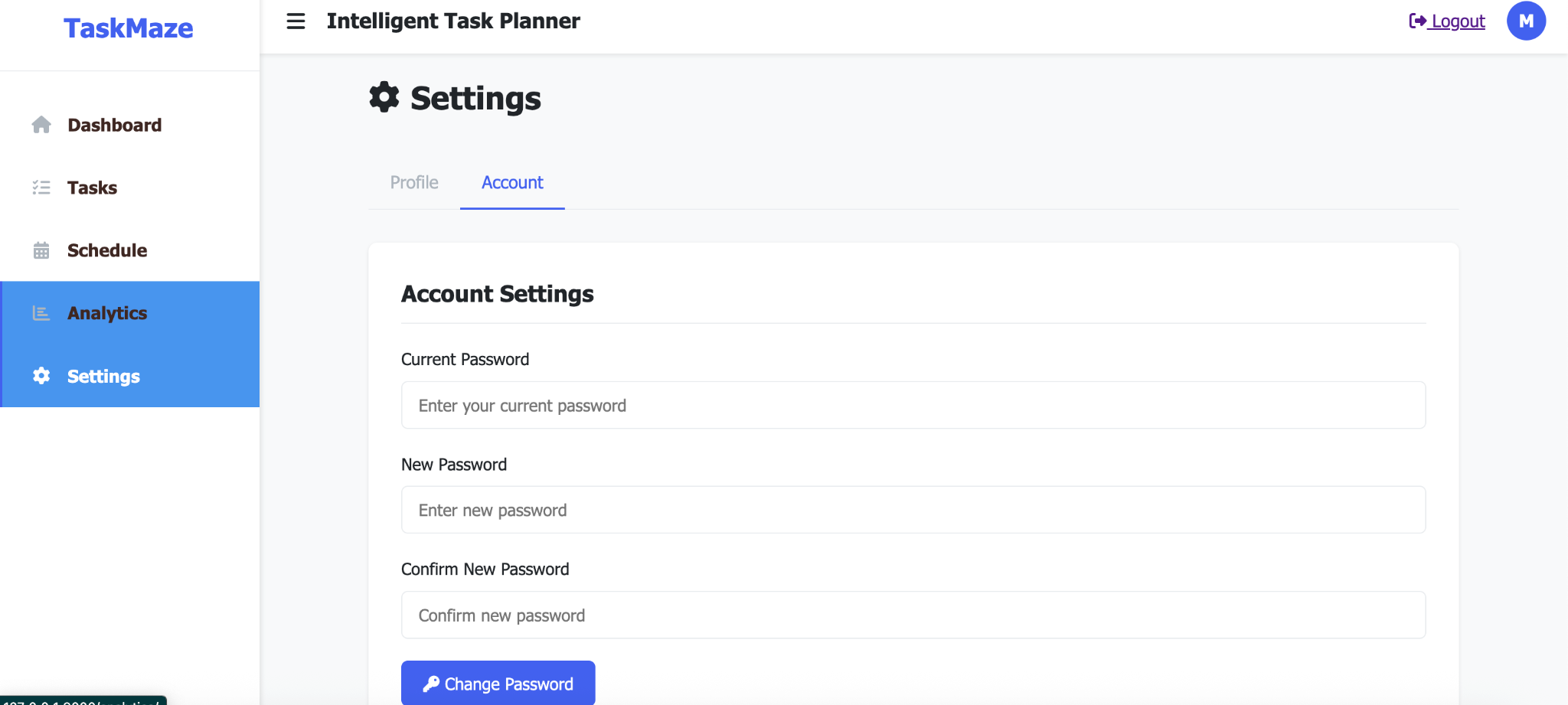
* **Metrics Calculation**: Total, completed, in-progress, overdue tasks
* **Percentage Calculations**: Completion rate, priority distribution, status distribution
* **Time Filtering**: Weekly, monthly, and quarterly aggregation
* Chart Data Generation: Prepared via generate\_chart\_data() for trend and distribution graphs

### Frontend Integration

* Data passed to templates as Python objects and JSON for Chart.js
* Responsive design with mobile-friendly charts and collapsible sidebar

It elevates the planner from a simple task organizer to a comprehensive productivity platform.





In the Settings, the user can edit their profile details or change their password.

## 9.Modeling and Implementation Challenges

During the development of the Intelligent Task Planner, the team encountered several modeling and implementation challenges:

1. **Requirement Ambiguity**Translating user needs into clear functional requirements was challenging since students had diverse expectations about scheduling, reminders, and analytics. Defining the scope without overcomplicating the system required iterative discussions and refinements.
2. **Task Overlap Handling**Modeling task conflicts was complex. The system had to detect overlapping tasks while still allowing users to override or adjust their schedules. Designing the logic for conflict detection and resolution required careful planning in both the backend and frontend.
3. **Integration with Future Features**Although the initial implementation relies on Django and SQLite, some features such as **Google Calendar sync** and **AI-driven task recommendations** were left as “To Be Determined (TBD).” Modeling these features in diagrams and requirements while keeping them optional created dependencies that could affect future scalability.
4. **Consistency Across Diagrams**Ensuring consistency between the **Use Case, Class, ER, Sequence, Activity, and State Machine diagrams** was challenging. Small changes in functional requirements often required updating multiple diagrams, which demanded strong version control and coordination among team members.
5. **Frontend–Backend Synchronization**AJAX-based task addition, editing, and deletion created challenges in maintaining synchronization between the task list and database. Handling edge cases like delayed server responses, invalid input, and concurrent edits required robust validation at both ends.
6. **Analytics Visualization**Designing intuitive and meaningful analytics dashboards required mapping abstract productivity metrics into concrete, visually interpretable charts. Balancing performance with usability while keeping charts responsive was a significant challenge.
7. **Scalability and Performance Concerns**Although SQLite was chosen for development simplicity, future scalability demands (supporting 1000+ concurrent users) may require migration to a more robust database (e.g., PostgreSQL). Anticipating these changes in the design phase while working with lightweight tools was a notable constraint.

## 

## 

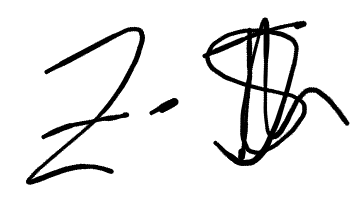
## 10. Conclusion

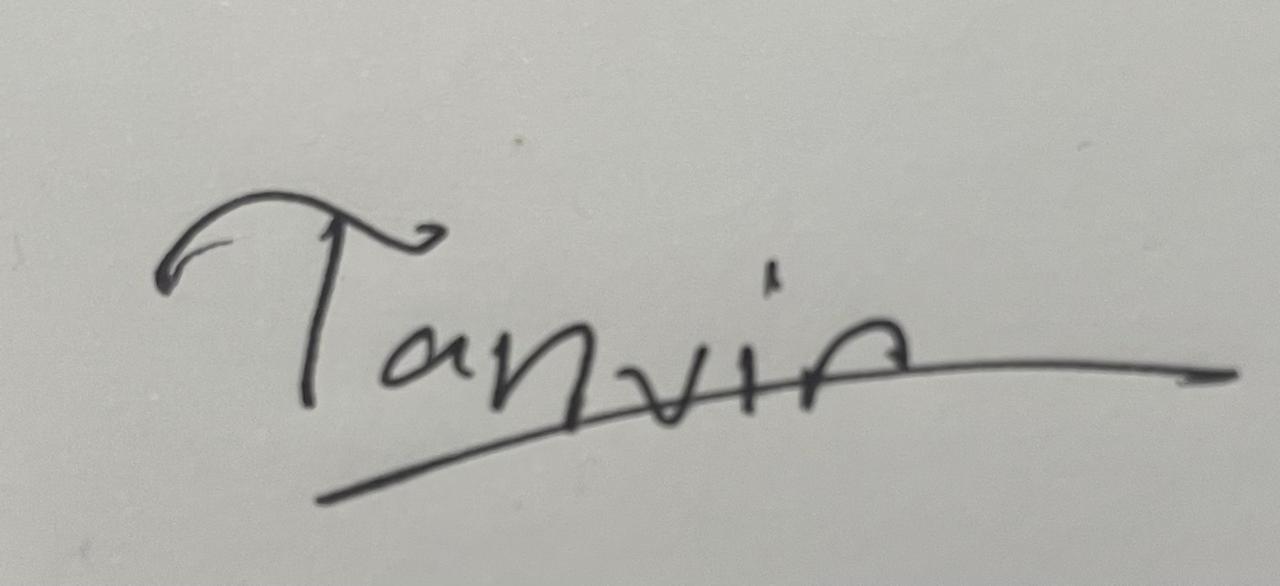
In summary, our system, **Intelligent Task Planner (TaskMaze)**, provides a simple and user-friendly platform for students to manage their academic and personal tasks effectively. Users can easily add, edit, delete, and view tasks, check their schedules, track progress through analytics, and even use productivity features like the Pomodoro timer — all in one place. We have tried our best within the given time frame to implement and demonstrate the core functionalities. Some advanced features, such as Google Calendar integration and AI-driven task suggestions, are yet to be completed, but we have ensured that the main requirements of the project are met and presented.

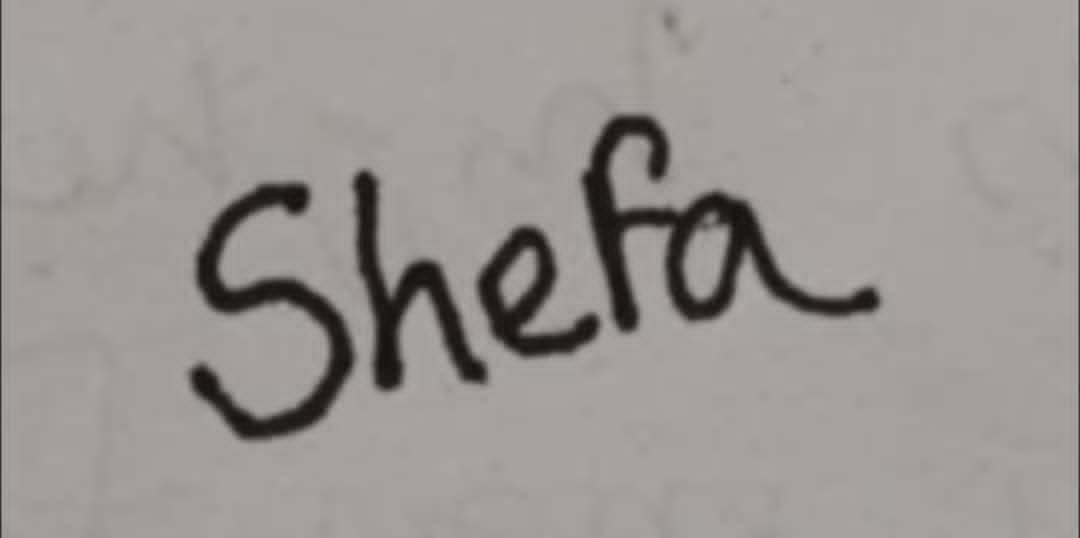
## 11.Overall Contribution

| **Requirement Number** | **Completed By** |
| --- | --- |
| FR1-User Registration | Zuhayer Islam |
| FR2-User Login | Zuhayer Islam |
| FR3-Add Task | Shefa Tabassum |
| FR4-Edit Task | Tanvir Ahmed and Shefa Tabassum |
| FR5-Delete Task | Shefa Tabassum |
| FR6-View Tasks | Tanvir Ahmed |
| FR7-View Analytics | Tanvir Ahmed |
| FR8-Manage Profile | Zuhayer Islam |
| FR9-View Schedule | Faheema Shaheed Tamanna |
| FR10- Add/Edit/Delete Task from Calendar | Faheema Shaheed Tamanna |
| FR11-Add Pomodoro Timer | Zuhayer Islam |
| FR12-User Logout | Zuhayer Islam |
| FR13- Smart Task Scheduling | Zuhayer Islam |
| Overall Front End | Tanvir Ahmed (Majority) |
| Non Functional Requirements | Everyone |

The following contributions table is approved by:

[Zuhayer Islam 2232061642](mailto:islamzuhayer2003@gmail.com) - 

[Tanvir Ahmed 2231047642](mailto:tanvir.ahmed32@northsouth.edu)-

[Shefa Tabassum 2232993642](mailto:shefa.tabassum@northsouth.edu) - 

[Faheema Shaheed Tamanna 2232144642](mailto:faheema.tamanna@northsouth.edu) - 