Contents

1. INTRODUCTION

1.1. Purpose

` This document shows the requirements for the Task Tracker Application, a tool designed to help its users manage their tasks efficiently and can be accessed on the web. It highlights both functional and non-functional requirements needed to design the system.

1.2. Problem Statement

There is a common challenge of personal and professional task management, a problem caused by the complexity of modern life. Individuals often struggle to maintain organized to-do lists, leading to missed deadlines, decreased productivity, and increased stress. The Task Tracker application allows a centralized and easy-to-use interface to create, prioritize, and monitor tasks with features such as due date reminders, activity history, and statistical summaries. By making task management easier, this application allows users to regain control of their time, become more productive, and more easily achieve their goals.

1.3. Scope

The scope of the project is to create a web application where the user can create, edit, delete, and mark a task as done. The application will also include user authentication, activity log, and task statistics. The application will be accessed using a web browser.

1.4. Definitions

- **UI**: User Interface
- **ER diagram**: Entity Relationship diagram
- **DFD diagram**: Data Flow diagram
- **MVP**: Minimum Viable Product.
- **System**: This is the application to be developed.
- **Task**: This is a unit of work with title, description, due date and priority.
- **User**: This is a registered person with an email and a password.
- **Actor**: This is a User or other system using the application.
- **Use Case**: This is an action that accomplishes some Task within the system.

- **FR:** Functional requirements, these define what the system should do. They describe the services the system provides, how it should react to particular inputs, and how it should behave in specific situations.
- **N-FR:** Non-functional requirements, these define how the system should be. They describe the system's qualities, such as performance, usability, security, and reliability.

2. OVERALL DESCRIPTION

2.1. System Features

Product. This is a standalone application that doesn’t utilize other systems. User authentication is done by the default django authentication manager.

Product Functions. The major functions include task management i.e. create, update, delete, read or mark as complete and static display of tasks available together with user’s recent activity.

2.2. User classes and Characteristics.

The target users are people who want to manage their tasks personal tasks, these can include both professional and non-professional tasks.

2.3. Operating Environments.

The application will be deployed on a web server and accessible via standard web browsers (Chrome, Firefox, Safari, Edge). The backend is developed using the Django framework and a relational database (PostgreSql).

2.4. Design and Implementation.

The application will be developed in python and the django framework. The database used will be a relational database. The application must be responsive on a range of devices. The application must be secure, and protect user data.

2.5. Assumptions and Dependencies.

It is assumed that users will have access to a stable internet connection. The application's performance will depend on the server's resources.

3. Specific Requirements.

3.1. Functional Requirements (FR).

- FR1: **User Authentication.**
    - ➢ The system shall allow users to register, login and logout.

- The system should allow users to edit their profile, such as changing their profile picture.(Req. Validation)

- FR2: **Task Management.**
  - The system should allow users to create new tasks with a title, description, priority, and due date.
  - The system should allow users to edit existing tasks.
  - The system should allow users to delete tasks.
  - The system should allow users to mark tasks as complete.
  - The System should allow users to see due date, priority, and description for each task in the 'Your Tasks' section on the home screen. (Req. Validation).
  - The system should allow users to search for Tasks by providing any task attribute such as “priority”.
- FR3: **Tracking User Activity.**
  - The system should log all task creation, updates, completion, and deletion events.
  - The system should display a list of recent task activities.
- FR4: **Searching Tasks.**
  - The system should allow users to search for tasks by title, task description, priority, or due date.
- FR5: **Task Statistics.**
  - The system shall display total, completed, pending, next up, and overdue task counts to the user.

3.2. <u>Non-Functional Requirements (NFR).</u>

- NFR1: Usability:
  - The UI shall be intuitive and easy to navigate.
  - The application shall provide clear error messages and feedback basing on user actions.
- NFR2: Performance.
  - The system shall load pages within 3s.
  - It shall handle up to 200 concurrent users without significant degradation.
- NFR3: Security.
  - User passwords shall be stored securely.
  - The system shall protect user data from unauthorized access.
- NFR4. Maintainability.

    - ➢ The code base shall be well written, commented and documented for easy maintainability.
- NFR5. Reliability.
    - ➢ The application shall be available 99% of the time.

3.3. User Interface Requirements.

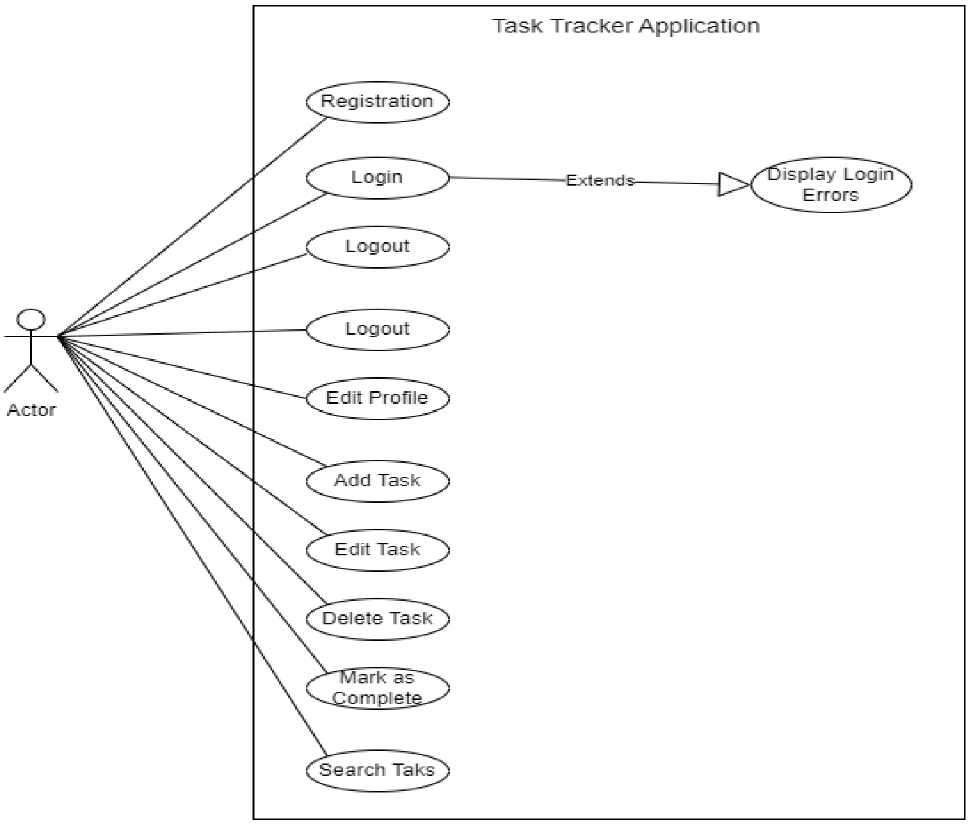
- The UI shall be web-based and responsive.
- The UI shall be consistent with the modern web design principles.

4. Use Case Diagram

This shows User interactions with the System.

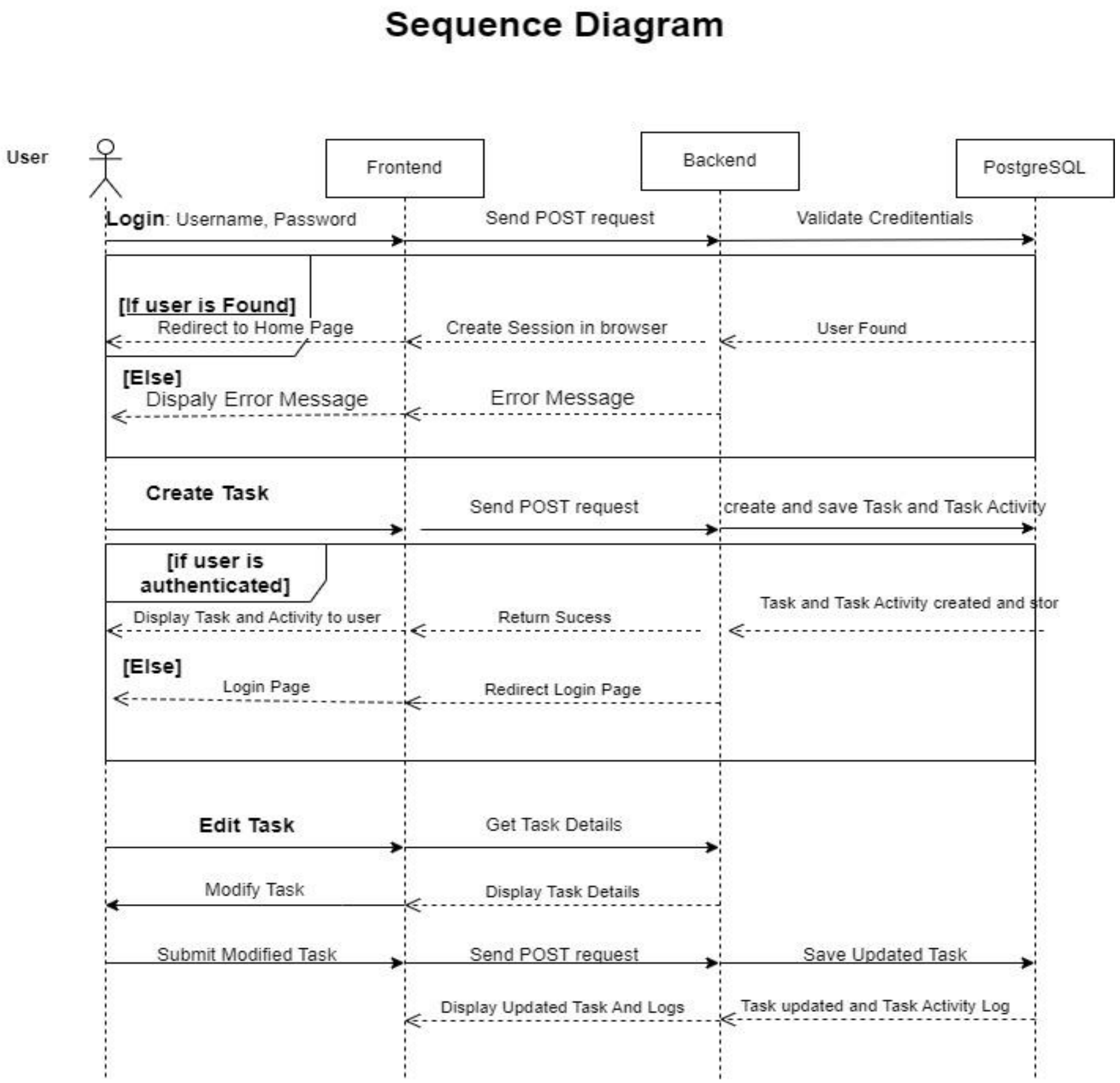
- ✓ The User (Actor) is represented by a Stick Figure.
- ✓ The System is represented by a Rectangle.
- ✓ The Use Case is represented by an ellipse or oval shape.
- ✓ The relationship between two Use Cases is represented by an arrow.

Use case Diagram



5. Sequence Diagram.

This shows the flow of events in a particular Use case. It models the flow of messages or interactions between objects or components in a system over time.

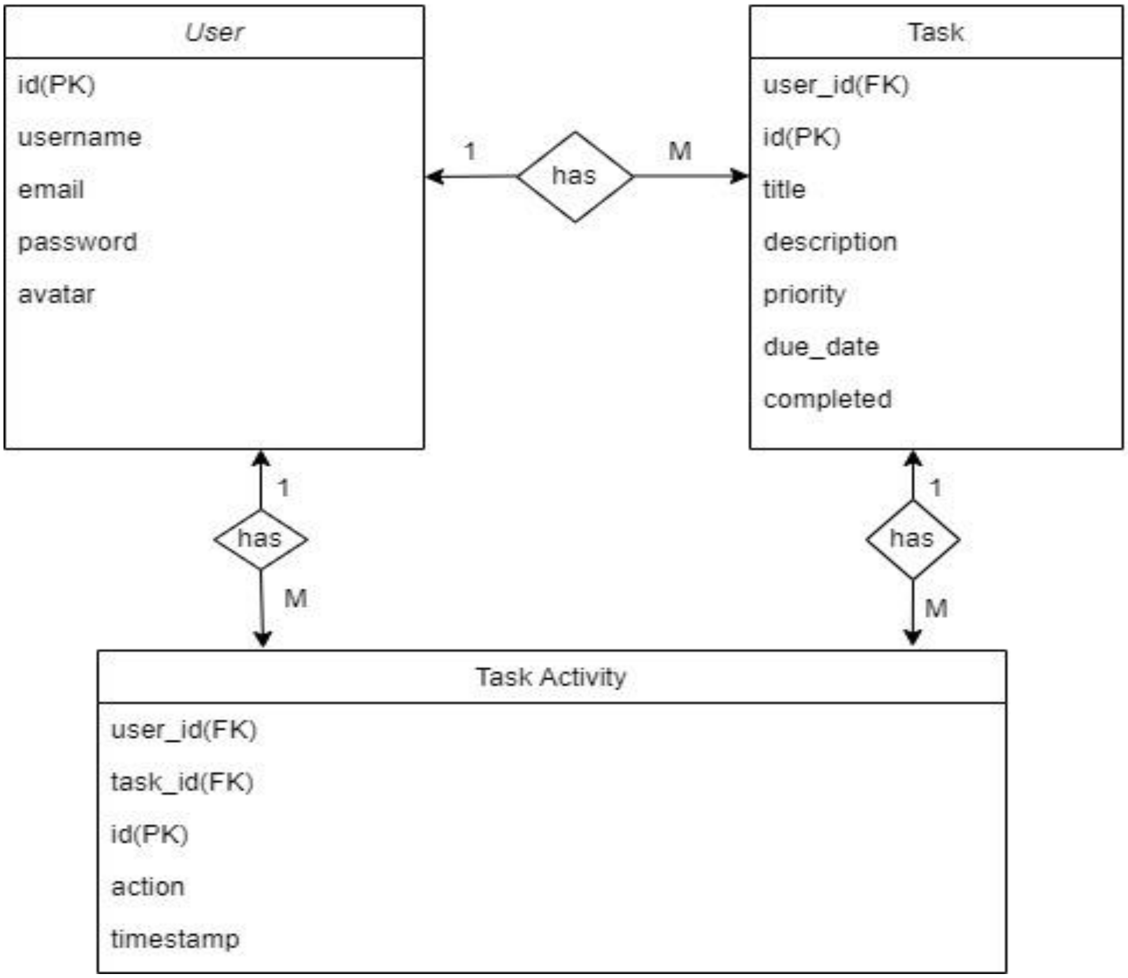


6. Entity Relationship (ER) Diagram.

This models the relationships between entities in a database. The entities include;

1. User.
2. Task.
3. Task Activity.

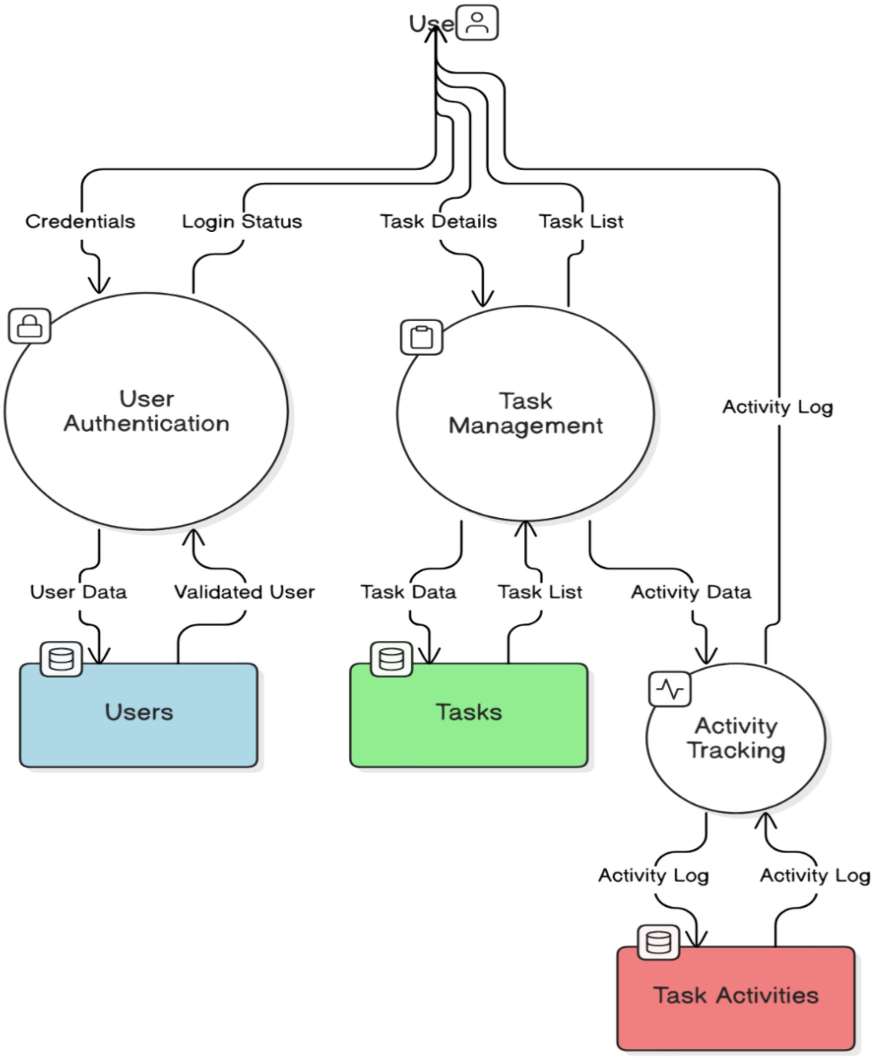
Entity Relationship Diagram



7. Data Flow Diagram (DFD).

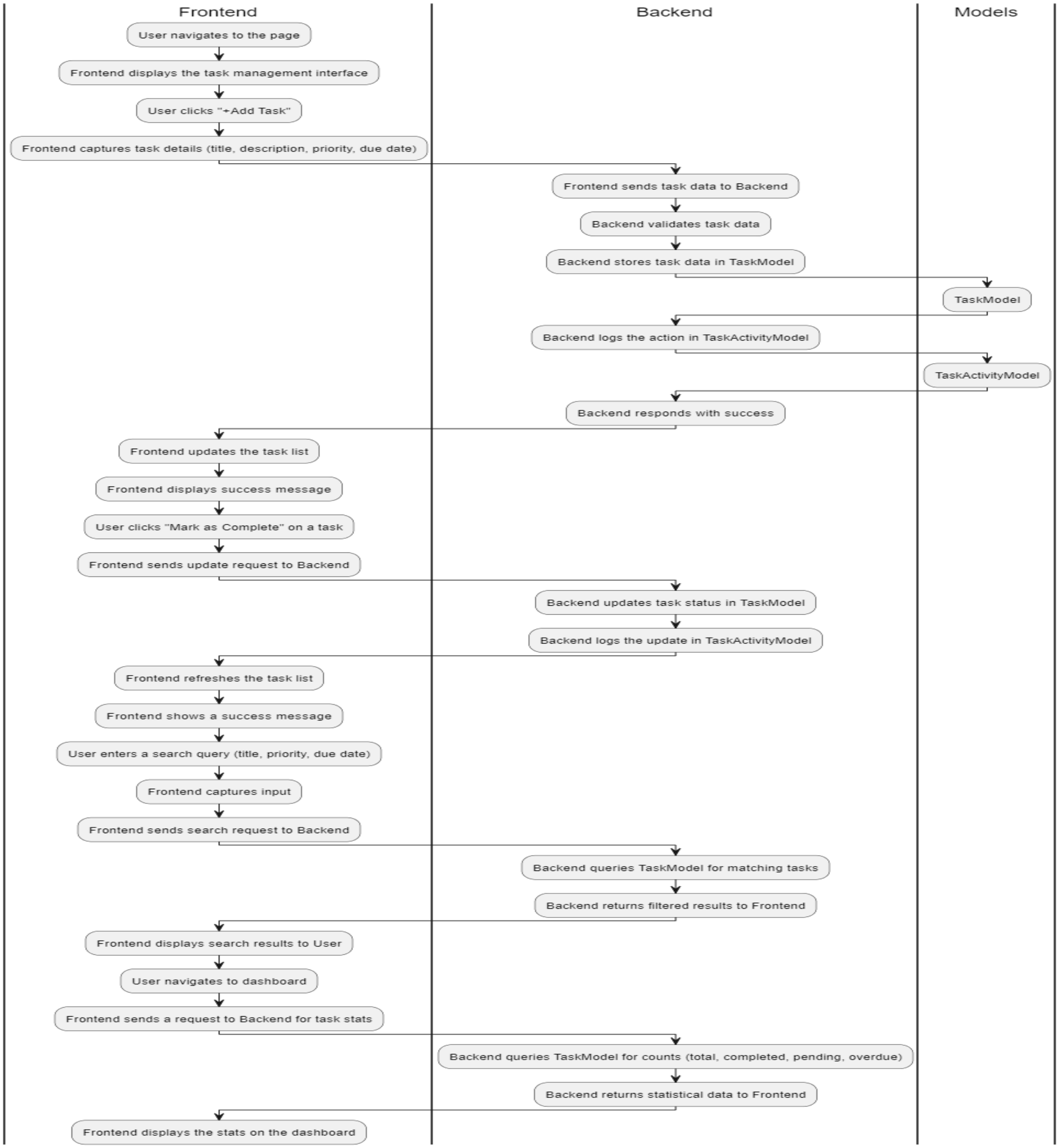
This shows how data moves through the Systems showing inputs, processes, storage and Outputs. It has;

1. External Entity: The User that sends and receives data.
2. Process: Represented by a circle and performs calculations or business rules on the data.
3. Data Store. Represented by a storage location or an open rectangle.



8. Activity Diagram

An activity diagram shows how a process works. It illustrates the different activities involved and the order in which they happen.


Frontend
Backend
Models
User navigates to the page
Frontend displays the task management interface
User clicks "+Add Task"
Frontend captures task details (title, description, priority, due date)
Frontend sends task data to Backend
Backend validates task data
Backend stores task data in TaskModel
TaskModel
Backend logs the action in TaskActivityModel
TaskActivityModel
Backend responds with success
Frontend updates the task list
Frontend displays success message
User clicks "Mark as Complete" on a task
Frontend sends update request to Backend
Backend updates task status in TaskModel
Backend logs the update in TaskActivityModel
Frontend refreshes the task list
Frontend shows a success message
User enters a search query (title, priority, due date)
Frontend captures input
Frontend sends search request to Backend
Backend queries TaskModel for matching tasks
Backend returns filtered results to Frontend
Frontend displays search results to User
User navigates to dashboard
Frontend sends a request to Backend for task stats
Backend queries TaskModel for counts (total, completed, pending, overdue)
Backend returns statistical data to Frontend
Frontend displays the stats on the dashboard