Task Manager Web App (Trello-lite)

1. Introduction
1.1 Purpose

The purpose of this project is to design and implement a Task Manager Web Application that allows users to create, organize, and manage their tasks and projects efficiently. The system will provide task categorization, status tracking, deadlines, and priority management.

This SRS defines the functional and non-functional requirements of the system to ensure clarity for both development and deployment.

1.2 Scope

The Task Manager Web Application will be a multi-user web system with the following capabilities:

User Authentication: Registration, login, logout, and profile management.

Task Management: CRUD (Create, Read, Update, Delete) operations for tasks.

Project/Category Grouping: Users can assign tasks to categories or projects.

Task Status Tracking: Tasks can be marked as To-Do, In Progress, or Done.

Deadline & Priority: Tasks can have due dates and priority levels.

Dashboard Overview: A summary of tasks by status and progress.

(Optional Advanced Feature): Drag-and-drop task board (Kanban style).

The application will be developed using:

Backend: Django + Django REST Framework (API)

Frontend: React.js (UI)

Database: PostgreSQL (or SQLite for dev)

1.3 Definitions, Acronyms, and Abbreviations

CRUD: Create, Read, Update, Delete

API: Application Programming Interface

UI: User Interface

JWT: JSON Web Token (authentication method)

2. Overall Description

2.1 User Classes and Characteristics

Regular Users:

Register, log in, and manage personal tasks.

Create and organize tasks into categories.

View progress and task completion rates.

Admin Users:

Manage user accounts (activate/deactivate).

Monitor system performance.

Ensure data consistency and security.

2.2 Product Perspective

Frontend (React): A responsive, interactive interface for users to interact with the system.

Backend (Django): Provides REST APIs to handle authentication, task operations, and reporting.

Database (PostgreSQL): Stores user accounts, tasks, categories, and status history.

2.3 Operating Environment

Frontend: Runs on any modern web browser (Chrome, Firefox, Edge).

Backend: Python/Django hosted on a cloud platform (e.g., Heroku, AWS, or DigitalOcean).

Database: PostgreSQL (cloud-hosted or local).

3. Functional Requirements

3.1 Authentication & User Management

FR1: The system shall allow users to register with email and password.

FR2: The system shall allow users to log in/log out.

FR3: The system shall securely store hashed passwords.

FR4: The system shall allow users to edit their profile.

3.2 Task Management

FR5: Users shall be able to create new tasks with title, description, deadline, and priority.

FR6: Users shall be able to update and delete tasks.

FR7: Users shall be able to categorize tasks under projects.

FR8: Users shall be able to change task status (To-Do, In Progress, Done).

3.3 Dashboard & Reporting

FR9: The system shall display task statistics (e.g., completed tasks vs pending).

FR10: The system shall provide filter and search functionality for tasks.

FR11: (Optional) Users shall be able to drag and drop tasks across categories.

4. Non-Functional Requirements
4.1 Performance

NFR1: The system shall support at least 100 concurrent users.

NFR2: The system shall respond to API calls within <500 ms on average.

4.2 Security

NFR3: All passwords must be hashed and salted.

NFR4: The system shall use HTTPS for secure communication.

NFR5: JWT or session-based authentication will be implemented.

4.3 Usability

NFR6: The UI shall be responsive for desktop and mobile devices.

NFR7: Users shall be able to learn the system within 10 minutes of use.

4.4 Reliability & Availability

NFR8: The system shall have 99.9% uptime (for production).

NFR9: Database shall have daily backup support.

5. System Models
5.1 Use Case Diagram

Actors: User, Admin

User: Register, Login, Manage Tasks, View Dashboard.

Admin: Manage Users, Monitor System.

5.2 ER Diagram (simplified)

User (id, name, email, password)

Project (id, title, user_id)

Task (id, title, description, deadline, priority, status, project_id, user_id)

Relationships:

One User → Many Projects

One Project → Many Tasks

6. Future Enhancements

Collaboration features (team-based projects).

Real-time notifications (WebSockets).

Calendar integration (Google Calendar API).

Mobile app version.