TaskFlow Project Documentation

Project Overview Document

Task Flow is a task management web application designed to help users efficiently organize and track their work. The application provides a user-friendly interface that allows users to create, update, and delete tasks, as well as categorize them by status (pending, In Progress, Completed).

Functional Requirements Document (FRD)

TaskFlow is a task management system for small teams. It enables users to create accounts, manage their tasks, assign tasks to teammates, and track task statuses
Main Functionalities:
1. User Authentication
- Users can register and log in using their email and password.
- Authenticated sessions use JWT for security.

2. Task Management (CRUD)
- Users can create, view, update, and delete tasks .
- Tasks can be assigned to specific users and categorized by status (Pending, In Progress, Completed)
- A task can include a title, description, deadline, assignee , priority, rating ,and status.

3. Dashboard
- Logged-in users see a dashboard displaying some of tasks assigned to them.
- Tasks can be filtered by status and priority.
- A summary section shows total tasks, completed tasks, and pending tasks.

4. Peer Evaluation
- After completing a task, a user can rate Task with (1-5 stars).

Technical Requirements Document (TRD)

1. Architecture and Stack:
   - Frontend : React.js with Recoil for state management, React Router (NavLink) for navigation, Axios for API communication, Bootstrap for styling and
   responsive layout, and built-in alert messages for user feedback.

   - Backend: Node.js with Express.js.
   - Database: MongoDB (via Mongoose ODM).
   - Authentication: JWT + bcrypt for password hashing.

2. API Overview:
- POST api/auth/register
- POST api/auth/login
-GET api/auth/users

-PUT api/auth/profile

- GET api/tasks (with JWT)
- POST api/tasks
- PUT api/tasks/:id
- DELETE api/tasks/:id
- POST api/tasks/:id/rate

3. Tools and Libraries:
- Backend: Express, Mongoose, dotenv, bcrypt, jsonwebtoken
- Frontend: React, Axios, Bootstrap, NavLink
- Dev Tools: Postman, Swagger, GitHub

Database Schema

Collections:

1. Users
- _id: (ObjectId)
- name: (String)
- email: (String)
- password: (String, hashed)
- profileImage: (String, optional)
- rating: (number)
- numOfRatings: (number)

2. Tasks
- _id: (ObjectId)
- title: (String)
- description: (String)
- status: (String)
- deadline: (Date)
- assignee: (UserId, ref: User)
- createdBy: (UserId, ref: User)
- ratingGiven: (Number, optional)
-Priority: (String)
-Task Done: (Number)
-Check List: (String)
- createdBy: (ObjectId)
- ratingGiven: (Number)

Endpoints and Models are modularized to ensure scalability.