Mid Semester Project Progress Report II

On

BrightHuman Web Application

submitted in partial fulfilment for award of

BACHELOR OF TECHNOLOGY

Degree

In

COMPUTER SCIENCE & ENGINEERING

2024–25

Under the Guidance of:

Dr. Pramod Kumar Sagar
Associate Professor
CSE Department

Submitted By:

Harsh Singh (2100330100100)
Vansh Kabaria (2100330100245)
Umesh Dixit (2100330100242)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

RAJ KUMAR GOEL INSTITUTE OF TECHNOLOGY

5th K.M. STONE DELHI-MEERUT ROAD, GHAZIABAD

Affiliated to Dr. A.P.J. Abdul Kalam Technical University, Lucknow

Department of Computer Science & Engineering

Project Progress Report

1. Course : Bachelor of Technology
2. Semester : VIIth
3. Branch : Computer Science & Engineering
4. Project Title : Bright Human Web Application
5. Details of Students:

S. No.	Roll No.	Name	Role as	Signature
1.	2100330100242	Umesh Dixit	Team Leader	
2.	2100330100100	Harsh Singh	Coder, Tester	
3.	2100330100245	Vansh Kabaria	Report	

6. SUPERVISOR:

Dr. Pramod Kumar Sagar

Remarks from Project Supervisor:

……
……
……

SYNOPSIS

Bright Human – Task Management and Performance Tracking System

Effective task management and performance tracking are critical for organizations to streamline operations and boost productivity. Bright Human addresses these needs by offering a centralized system where administrators can efficiently allocate tasks, monitor their progress, and evaluate employee performance through comprehensive dashboards. By leveraging modern technologies, Bright Human aims to improve collaboration, enhance accountability, and support data-driven decision-making.

Objective

The main objective of Bright Human is to create a task management and performance tracking application using a modern technology stack. The system will allow seamless task assignments, real-time progress tracking, and detailed performance evaluation with graphical insights.

Scope of the Project

Bright Human is designed for teams in small to medium-sized organizations. The application will offer:

- Task assignment and status updates.
- Task prioritization (High/Low).
- Real-time notifications for deadlines and updates.
- Proof submission for completed tasks.
- Performance tracking through interactive dashboards.

By simplifying task allocation and tracking workflows, the system ensures higher productivity and accountability.

System Features

The following features will be included in Bright Human:

Admin Module:

- Create, assign, and prioritize tasks for employees.

- Monitor task progress and generate performance reports.

Employee Module:

- View assigned tasks and deadlines.
- Update task status and upload proof of completion.
- Receive notifications for task assignments and updates.

Performance Dashboard:

- Weekly, Monthly, and Yearly performance analytics using graphs.
- A comparison of employee productivity and task efficiency.

Notifications:

- Alerts for task assignments, changes, and upcoming deadlines.

Methodology

Requirement Analysis:

- Identify user needs and functional specifications.
- Gather requirements for performance metrics and task workflows.

System Design:

- Data Flow Diagrams (DFD) to define system processes.
- Entity-Relationship Diagram (ERD) for database structure.

Development:

- Front-End: React, HTML, CSS, and JavaScript to build a responsive interface.
- Back-End: Node.js for managing the application logic and APIs.
- Database: MongoDB for storing task data and performance metrics.
- Version Control: Git for collaborative development.

Testing:

- Conduct unit, integration, and system testing to validate functionality.

Deployment:

- Deploy on cloud platforms such as AWS or Azure for scalability and availability.

Expected Outcomes

- A user-friendly application for efficient task management.
- Improved task prioritization and status tracking.
- Accurate performance evaluation through real-time dashboards.
- Enhanced team collaboration and accountability.

Tools and Technologies

- Front-End: React.js, HTML, CSS, JavaScript
- Back-End: Node.js
- Database: MongoDB
- Version Control: Git
- Hosting: Docker, AWS, or Azure

Conclusion

Bright Human will transform how organizations manage tasks and evaluate performance. By utilizing a modern technology stack, the application ensures a seamless and intuitive experience for both administrators and employees, fostering a productive and accountable work environment.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES:

CHAPTER 1

INTRODUCTION

1.1 Background and Motivation

Task management is a crucial aspect of personal and professional life, involving the planning, organization and tracking of tasks to achieve specific goals. Traditionally, task management relied on physical tools like paper planners, whiteboards, or basic digital solutions. However, with the advent of remote work, globalization, and the increasing complexity of team dynamics, these methods have become inadequate.

1.2 Problem Statement

Organizations and individuals face challenges in managing tasks effectively due to fragmented communication, lack of real-time collaboration, and difficulty in tracking progress. This often leads to missed deadlines, inefficient resource utilization, and reduced productivity. The existing task management solutions are often device-dependent, lack scalability, or fail to integrate seamlessly with collaborative tools, making them unsuitable for teams with diverse needs.

1.3 Objective

The primary objective of the BrightHuman Application is to create a productivity and goal-setting app that helps individuals—such as students, professionals, and team leaders—achieve their personal and professional goals by providing tailored schedules, tracking task completion, and offering performance analytics. The app will focus on enhancing users' productivity by analyzing their routines, goals, and interests to suggest optimized schedules that are personalized to their specific needs.

1.4 Scope

The scope of the BrightHuman Application encompasses a wide range of functionalities and features that cater to different user segments, including students, employees, team leaders, and anyone seeking a personalized approach to task management and goal achievement.

1.5 Background and related work

Table 1.1 Comparison of various methodology suggested by authors

SR. NO.	PAPER NAME	AUTHOR(S)	YEAR	METHODOLOGY
1	Personalized Task Scheduling and Monitoring Using Machine Learning	Liu, X., Zhang, Y., & Wang, L.	2023	This paper explores the use of machine learning for personal task scheduling and monitoring, which is relevant to the personalized scheduling features of our application.
2	Behavioral Analysis and Prediction of User Productivity in Task Management Applications	Singh, A., & Gupta, R.	2022	This research examines how user behavior can be analyzed and predicted to enhance productivity in task management applications, relevant to goal tracking and performance analysis.
3	Enhancing Goal Achievement through Adaptive Scheduling Algorithms	Kim, J., & Park, H.	2023	Focuses on adaptive scheduling algorithms that adjust based on user progress and changing goals, aligning with your application's scheduling and goal-setting features.
4	A Comprehensive Review of Productivity Tools and Their Impact on Goal Achievement	Brown, T., & Johnson, M.	2024	This review paper discusses various productivity tools and their effectiveness in helping users achieve their goals, providing context for our application's goal management features.
5	Secure and Scalable Personal Data Management with MongoDB and Node.js	Patel, N., & Lee, A.	2023	Explores the use of MongoDB and Node.js for secure and scalable personal data management, relevant to the technology stack used in our project.

Liu, X., Zhang, Y., & Wang, L. propose on Personalized Task Scheduling and Monitoring Using Machine Learning. This paper explores the use of machine learning for personal task scheduling and monitoring, which is relevant to the personalized scheduling features of our application.

Singh, A., & Gupta, R. propose on Behavioral Analysis and Prediction of User Productivity in Task Management Applications. This research examines how user behavior can be analyzed and predicted to enhance productivity in task management applications, relevant to goal tracking and performance analysis.

Kim, J., & Park, H. propose on Enhancing Goal Achievement through Adaptive Scheduling Algorithms. Focuses on adaptive scheduling algorithms that adjust based on user progress and changing goals, aligning with your application’s scheduling and goal-setting features.

Brown, T., & Johnson, M. propose on A Comprehensive Review of Productivity Tools and Their Impact on Goal Achievement. This review paper discusses various productivity tools and their effectiveness in helping users achieve their goals, providing context for our application's goal management features.

Patel, N., & Lee, A. propose on Secure and Scalable Personal Data Management with MongoDB and Node.js". Explores the use of MongoDB and Node.js for secure and scalable personal data management, relevant to the technology stack used in our project.

CHAPTER 2

HARDWARE AND SOFTWARE REQUIREMENTS:

2.1. Hardware Technologies

Development Hardware

- **Development Machines**:
    - **PCs or Laptops**: Equipped with modern processors (e.g., Intel Core i5/i7 or AMD Ryzen), at least 8GB of RAM, and sufficient storage (SSD recommended for speed) to handle development tasks and software tools.
    - **Operating System**: Windows, macOS, or Linux, depending on developer preference and compatibility with development tools.
- **Server Hardware** (for Deployment):
    - **Web Servers**: Hosted on cloud-based virtual machines or physical servers with appropriate resources (e.g., AWS EC2 instances, DigitalOcean Droplets) to handle application traffic.
    - **Database Servers**: Servers with adequate processing power and memory to manage the MongoDB database efficiently.

2.2 Software Technologies

Development Tools and Frameworks

Front-End Technologies:

- **React**: JavaScript library for building the user interface and handling dynamic content.
- **HTML/CSS**: Core technologies for structuring and styling web pages.
- **JavaScript**: Programming language used in conjunction with React for creating interactive web features.

- **Back-End Technologies**:
    - **Node.js**: JavaScript runtime for server-side development, enabling asynchronous processing and real-time capabilities.
    - **Express.js**: Web application framework for Node.js to handle routing, middleware, and server-side logic.
- **Database**:
    - **MongoDB**: NoSQL database for storing user profiles, tasks, schedules, and performance metrics. MongoDB Atlas can be used for cloud-based database management.

- **Development and Deployment Tools**:
    - **Integrated Development Environment (IDE)**:
        - **Visual Studio Code**: IDE used for coding in JavaScript, HTML, CSS, and Node.js.
    - **Version Control**:
        - **Git**: Version control system for managing code changes and collaboration.
        - **GitHub/GitLab**: Platforms for hosting repositories and facilitating team collaboration.
- **Communication and Collaboration Tools**:
    - **Zoom** or **Google Meet**: For virtual meetings and discussions.

CHAPTER 3

SDLC METHODOLOGIES

The **BrightHuman Application** aims to provide users, such as students, employees, or team leaders, with a tool for managing tasks and achieving their goals. The development process follows a structured approach, utilizing modern technologies for both front-end and back-end development. Below is a detailed overview of the technologies, languages, and tools used in the project.

3.1 Technologies and Tools

Front-End Development

- **Framework**:
    - **React**: A powerful JavaScript library used for building dynamic and responsive user interfaces. React's component-based architecture enables the creation of reusable UI elements, enhancing the application's interactivity and efficiency.
- **Languages**:
    - **JavaScript**: The main language used to build interactive features on the front end with React.
    - **HTML/CSS**: Used for structuring and styling web pages to ensure a clean and user-friendly design.

Back-End Development

- **Framework**:
    - **Node.js**: A JavaScript runtime that allows for efficient server-side development. Node.js is well-suited for building scalable and high-performance applications due to its event-driven architecture and non-blocking I/O model.
- **Libraries**:
    - **Express.js**: A lightweight web application framework for Node.js. Express handles routing, middleware, and request handling, enabling smooth interaction between the front-end and back-end systems.

- **Languages**:
    - **JavaScript**: Used for server-side logic with Node.js, allowing for consistent development across the front end and back end.

Database

- **NoSQL Database**:
    - **MongoDB**: A NoSQL database chosen for its flexibility in managing unstructured data. MongoDB is ideal for storing user profiles, tasks, schedules, and performance analytics, offering a schema-less structure that allows for dynamic and scalable data storage.
    - MongoDB's JSON-like document structure is well-suited for handling the varied data types that the application processes, such as user goals, task details, and proof submissions (like images or videos).

3.2 Development Tools

- **Version Control**:
    - **Git**: Used for version control, enabling multiple developers to collaborate and track changes in the codebase. Repositories are hosted on platforms like **GitHub** or **GitLab** for efficient team collaboration.
- **Integrated Development Environment (IDE)**:
    - **Visual Studio Code**: A versatile IDE used for coding the front end (React) and back end (Node.js). It offers features like debugging, code completion, and extensions that streamline development.

3.3 Implementation Approach

- **Development Cycle**:
    - The project follows an **Agile methodology** to ensure flexibility and adaptability. The development process is broken down into smaller, manageable sprints, allowing for continuous improvement and iteration based on feedback.

- **Core features**: Development starts with basic features like user profile creation, task management, and scheduling. Advanced features such as proof submission, performance tracking, and analytics will be added in later stages.

- **Technological Workflow**:
  - **React** handles the front-end UI, offering users a seamless experience for managing tasks and goals.
  - **Node.js** with **Express.js** manages server-side operations, handling user requests and providing communication between the front end and the database.
  - **MongoDB** stores and retrieves data, such as user profiles, tasks, schedules, and performance metrics. Its flexibility and scalability ensure efficient handling of large volumes of user-generated data.

- **API Design**:
  - RESTful APIs are designed using **Express.js** to handle requests between the client (React front-end) and the server (Node.js back-end). The APIs enable actions like creating user profiles, scheduling tasks, marking tasks as complete, and generating performance reports.

3.4 Testing

- **Unit Testing**:
  - Individual components and modules are tested to ensure each piece of functionality works as expected.

- **Integration Testing**:
  - Integration testing ensures smooth interaction between the front-end (React), back-end (Node.js), and database (MongoDB).

- **User Acceptance Testing (UAT)**:
  - Engage users to test the application's functionality, performance, and ease of use. Based on the feedback, adjustments are made to improve the user experience and overall performance.

CHAPTER 4

RISK ASSESSMENT

4.1 Technical Risks

a. System Downtime

- **Description:** Failure of the backend server or database, leading to system unavailability.
- **Impact:** High (Tasks cannot be assigned, updated, or monitored).
- **Likelihood:** Medium.
- **Mitigation:**
    - Use a cloud-based infrastructure (e.g., AWS, Azure) with auto-scaling and failover support.
    - Set up regular health checks and alerts for system monitoring.
- **Contingency Plan:** Switch to a backup server and restore from the latest database backup.

b. Data Loss or Corruption

- **Description:** Accidental deletion or corruption of task, user, or analytics data.
- **Impact:** High (Critical system data might be lost).
- **Likelihood:** Low.
- **Mitigation:**
    - Implement database backups at regular intervals.
    - Use database transaction logs for recovery.
    - Validate all inputs to prevent accidental data corruption.
- **Contingency Plan:** Restore data from the most recent backup.

c. Integration Failures

- **Description:** Issues in communication between modules (e.g., Task Manager and Analytics Module).
- **Impact:** Medium.
- **Likelihood:** Medium.

- **Mitigation:**
    - Use well-defined APIs with robust error handling mechanisms.
    - Perform end-to-end testing during development.
- **Contingency Plan:** Roll back to the previous stable version of the integration.

4.2 Security Risks

a. Unauthorized Access

- **Description:** Malicious actors gaining access to sensitive data or admin privileges.
- **Impact:** High.
- **Likelihood:** Medium.
- **Mitigation:**
    - Implement role-based access control (RBAC).
    - Use encryption (e.g., SSL/TLS) for data transmission.
    - Enforce strong password policies and multi-factor authentication.
- **Contingency Plan:** Conduct an immediate security audit and revoke unauthorized access.

b. Data Breaches

- **Description:** Leakage of user or task data due to vulnerabilities.
- **Impact:** High.
- **Likelihood:** Low.
- **Mitigation:**
    - Regularly update dependencies and libraries to patch vulnerabilities.
    - Use vulnerability scanning tools.
    - Secure the database with encryption and restricted access.
- **Contingency Plan:** Notify affected parties and deploy a fix immediately.

4.3 Project Management Risks

a. Missed Deadlines

- **Description:** Delays in completing project milestones due to resource or time constraints.
- **Impact:** Medium.
- **Likelihood:** Medium.
- **Mitigation:**
    - Use Agile methodology with clear sprint goals.
    - Monitor progress using Gantt charts and burn-down charts.
- **Contingency Plan:** Reprioritize features to ensure core functionalities are delivered on time.

b. Resource Unavailability

- **Description:** Key team members becoming unavailable during critical phases.
- **Impact:** Medium.
- **Likelihood:** Medium.
- **Mitigation:**
    - Maintain detailed documentation to facilitate onboarding of new team members.
    - Cross-train team members on multiple modules.
- **Contingency Plan:** Allocate additional resources or hire temporary experts.

4.4 Operational Risks

a. Poor User Adoption

- **Description:** Users (e.g., employees, admins) finding the system too complex or unhelpful.
- **Impact:** Medium.
- **Likelihood:** Medium.
- **Mitigation:**
    - Conduct usability testing with potential users.
    - Provide comprehensive user training and help documentation.
- **Contingency Plan:** Collect user feedback and release updates to improve usability.

b. Performance Bottlenecks

- **Description:** Slow system performance under high user load.

- **Impact:** Medium.
- **Likelihood:** Medium.
- **Mitigation:**
    - Optimize code and database queries.
    - Conduct load testing to identify bottlenecks.
    - Use caching and load balancers.
- **Contingency Plan:** Upgrade system resources (e.g., CPU, memory) as needed.

4.5 Legal and Compliance Risks

a. Data Privacy Violations

- **Description:** Non-compliance with data protection laws (e.g., GDPR, CCPA).
- **Impact:** High.
- **Likelihood:** Low.
- **Mitigation:**
    - Ensure the project complies with relevant data privacy laws.
    - Implement features like data anonymization and user consent forms.
- **Contingency Plan:** Work with legal experts to rectify non-compliance and notify affected users.

4.6 External Risks

a. Third-Party API Failure

- **Description:** Downtime or bugs in APIs used for additional functionality.
- **Impact:** Medium.
- **Likelihood:** Medium.
- **Mitigation:**
    - Choose reliable third-party services.
    - Implement fallback mechanisms for critical functionalities.
- **Contingency Plan:** Switch to an alternate provider or temporarily disable non-critical API features.

CHAPTER 5

DATAFLOW DIAGRAM(DFD) AND ENTITY RELATIONSHIP DIAGRAM (ER DIAGRAM):

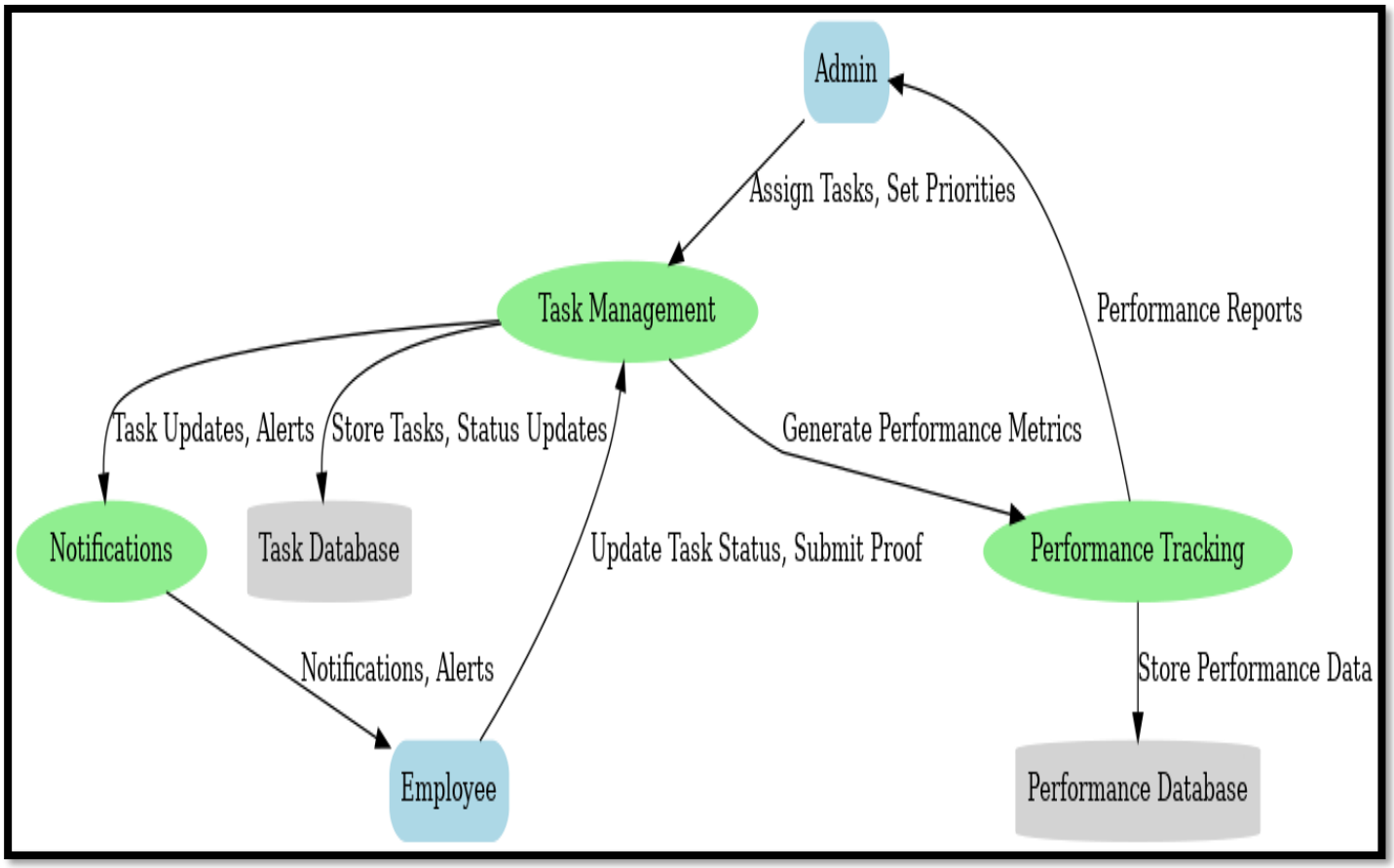


Fig. 5.1 Data Flow Diagram

Level 0 - Context Diagram

This level shows the interaction between the system and external entities:

- Admin: Assigns tasks, sets priorities, and reviews performance reports.
- Employee: Updates task status, submits proofs, and receives notifications.

Level 1 - Detailed Diagram

Entities:

1. **Admin:**

- Creates tasks and assigns them to employees.
- Defines task priorities (High/Low).
- Receives performance reports to monitor employee efficiency.

2. **Employee:**
   - Updates task progress (Complete, In Progress, Rejected).
   - Submits proof of task completion.
   - Receives notifications about assigned tasks and deadlines.

Processes:

1. **Task Management:**
   - Manages tasks assigned to employees.
   - Tracks task progress and status updates.
   - Stores task-related data in the Task Database.
2. **Performance Tracking:**
   - Uses task data to calculate performance metrics.
   - Generates graphical reports (weekly, monthly, yearly) for the admin.
   - Stores performance metrics in the Performance Database.
3. **Notifications:**
   - Sends alerts and updates to employees about tasks, deadlines, and changes.

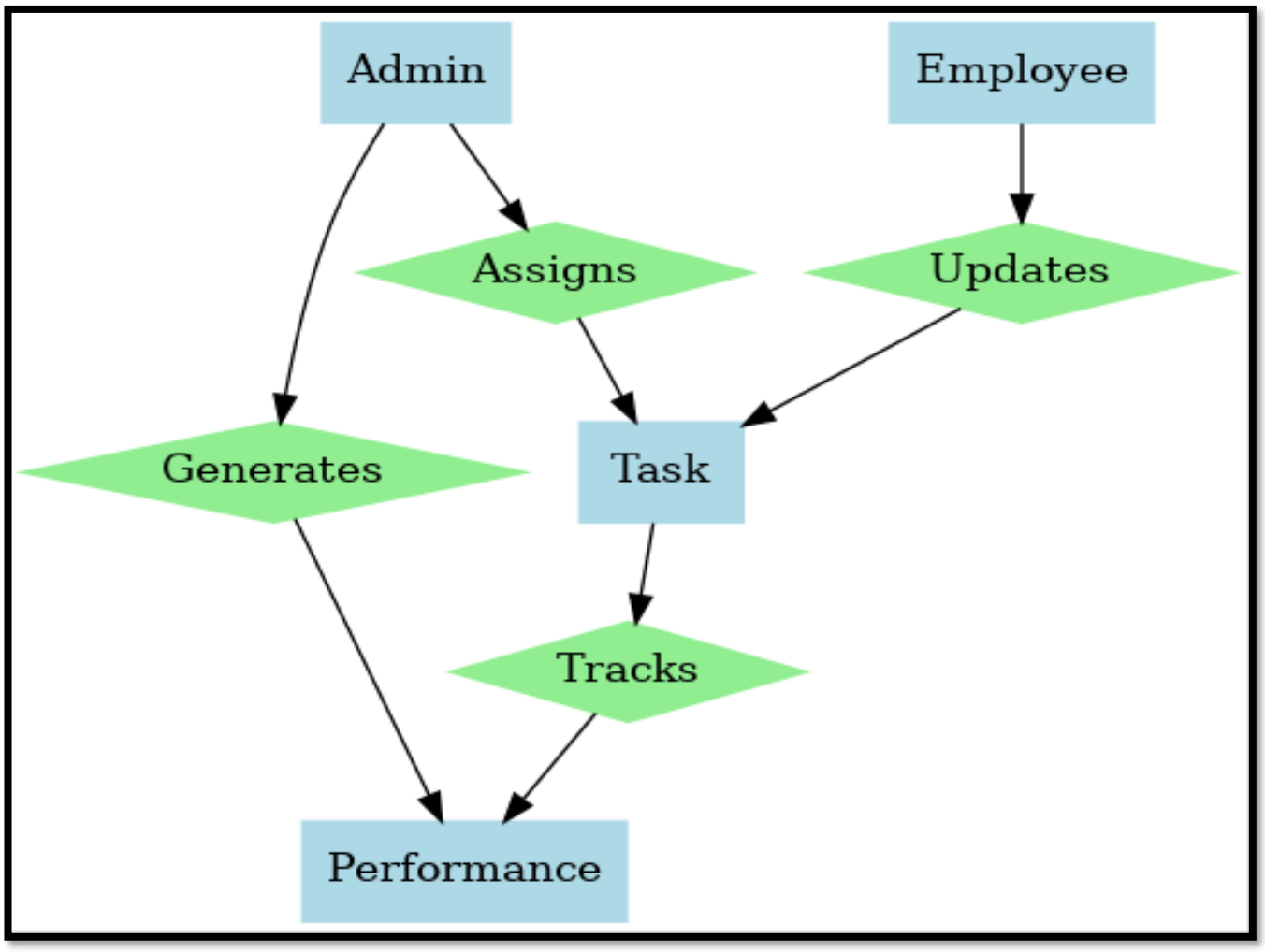
Data Stores:

1. **Task Database:**
   - Contains all task-related information such as assigned tasks, priorities, status updates, and proof submissions.
2. **Performance Database:**
   - Stores metrics derived from task completion data to evaluate individual and team performance.

Data Flow

1. **Admin to Task Management:**
    - Admin assigns tasks and sets priorities.
    - Task details are stored in the Task Database.
2. **Employee to Task Management:**
    - Employees update task status and submit proofs.
    - These updates are stored in the Task Database.
3. **Task Management to Notifications:**
    - Sends notifications to employees about new tasks, updates, or deadlines.
4. **Task Management to Performance Tracking:**
    - Shares task completion data to calculate performance metrics.
5. **Performance Tracking to Admin:**
    - Generates performance reports and shares them with the admin for review.
6. **Performance Tracking to Performance Database:**
    - Stores calculated performance metrics for future analysis.

Fig. 5.2 ER Diagram



CHAPTER 6

SOFTWARE REQUIREMENTS AND SPECIFICATIONS

- Purpose: This SRS defines requirements for BrightHuman, a web application that helps users achieve their goals by suggesting schedules, tracking tasks, and analyzing performance.
- Scope: The application supports students, professionals, and team leads. Users set profiles, goals, and routines; the system suggests tasks and tracks completion. Users can view analytics to measure progress.
- References:
- Node.js, Express, MongoDB docs
- W3C standards

6.1 Overall Description

- Product Perspective: A standalone web app with Node.js/Express backend, MongoDB database, and HTML/CSS/JS frontend. Accessible via any modern browser.
- User Classes:
- Students: Academic goals (e.g., JEE prep)
- Professionals: Career-related goals
- Team Leads: (Future enhancement) track team goals
- Admin: Manage system and users
- Operating Environment: Web-based, responsive, HTTPS-enabled.
- Assumptions & Constraints:
- Stable internet required.
- Compliant with web standards and security best practices.

6.2 System Features

- **3.1 User Profile & Goals**

- Description: Users register and create profiles with name, bio, interests, and goals. Requirements:
- Register with unique credentials
- Edit profile details anytime
- Store user data securely in MongoDB
- **3.2 Goal/Routine Setup**
- Description: Users define daily routines and goals that guide task suggestions. Requirements:
- Input multiple goals and routines
- Update goals/routines anytime
- **3.3 Task Generation**
- Description: System suggests tasks based on user goals and routines. Requirements:
- Automatically generate daily/weekly tasks
- Allow manual task addition/removal
- **3.4 Task Completion & Proof**
- Description: Users mark tasks done and may upload proof (image/video). Requirements:
- Mark tasks complete/pending
- Optional proof submission
- Store proofs securely (local or cloud storage)
- **3.5 Performance Analytics**
- Description: Track and display user progress over time. Requirements:
- Show completed vs. pending tasks
- Provide daily/weekly/monthly analytics
- Export or print performance reports

- **3.6 Administrative Functions**
- Description: Admins manage users and content. Requirements:
- View/edit user details
- Remove inappropriate content
- Generate usage reports

6.3 External Interface Requirements

- User Interface: Responsive web UI with dashboard, profile editor, and analytics pages.
- Hardware/Software:
- Client: Modern browser
- Server: Node.js/Express on cloud, MongoDB database
- Communication: JSON over HTTPS REST APIs
- 5. Non-Functional Requirements
- Performance: Handle concurrent users with acceptable load times (<2s).
- Security:
- Use HTTPS
- Hash passwords (e.g., bcrypt)
- Prevent XSS/SQL injection

6.4 Non-Functional Requirements

- Performance: Handle concurrent users with acceptable load times (<2s).
- Security:
- Use HTTPS
- Hash passwords (e.g., bcrypt)
- Prevent XSS/SQL injection

Reliability & Availability:

- 99% uptime target

- Regular backups
- Maintainability:
- Modular code, documented
- Easy feature extension
- Portability & Usability:
- Accessible on desktop/mobile
- Comply with basic accessibility guidelines

6.5 Other Requirements

- Legal & Compliance:
- Comply with data privacy laws (e.g., GDPR)
- Provide Terms & Privacy Policy
- 7. Appendices
- Sample User Flow:
- Student registers, sets goal (e.g., JEE), defines study routine
- System suggests daily tasks (study sessions, revision)
- Student marks tasks complete, uploads proof if required
- Reviews monthly progress chart
- Future Enhancements:
- Calendar integrations
- Gamification (badges, points)
- Team collaboration features

CHAPTER 7

PROJECT MODULE DESIGN

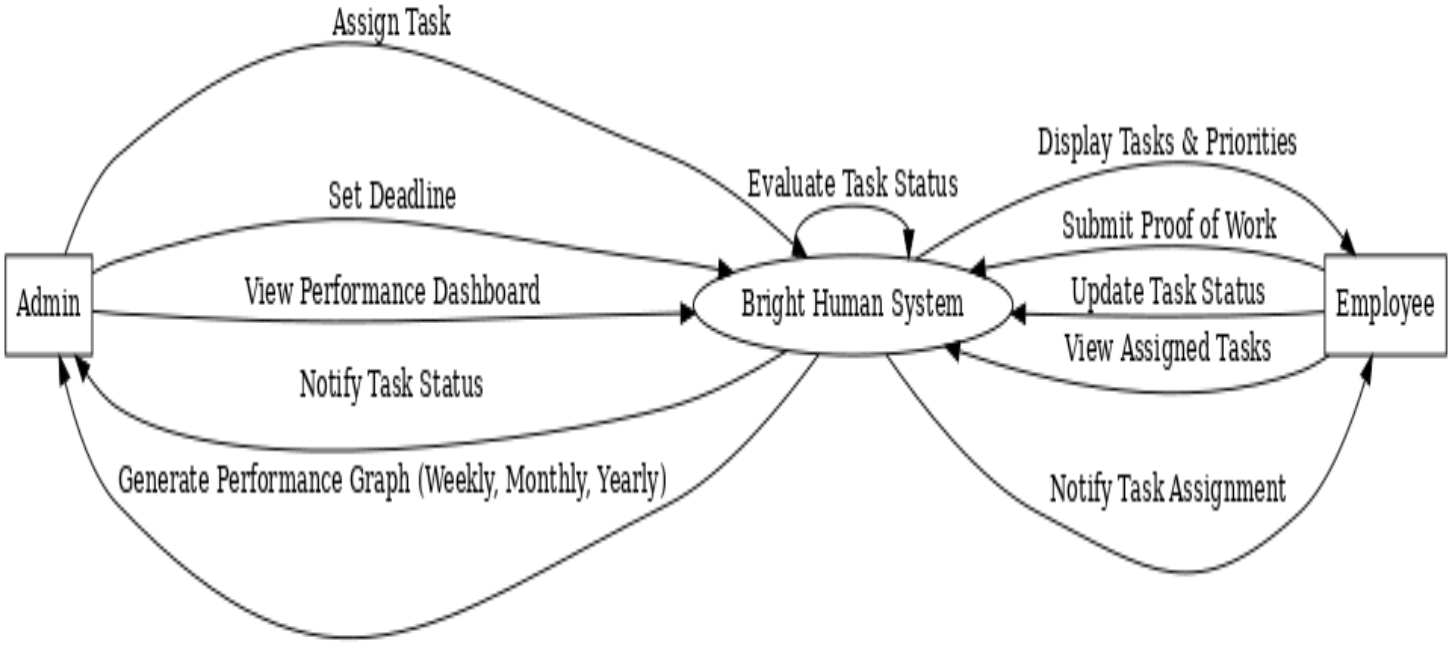


Fig. 8.1 Sequence Diagram

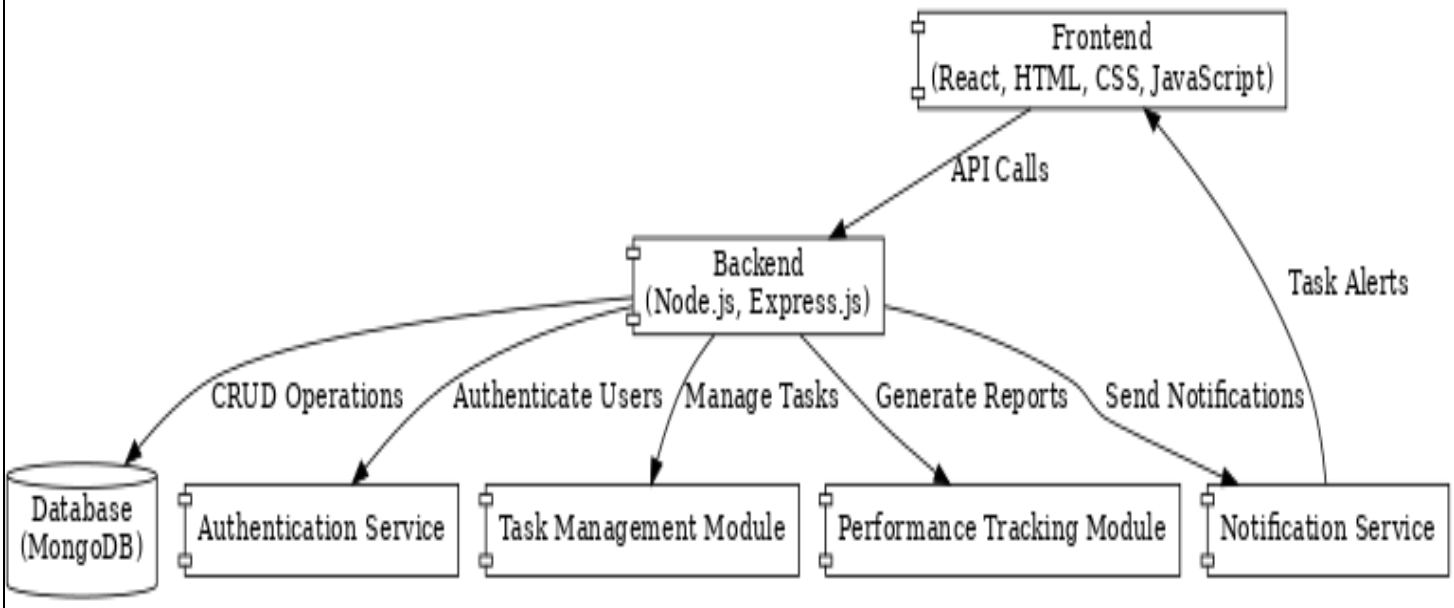


Fig. 8.2 Component Diagram

CHAPTER 8

PROJECT SNAPSHOTS

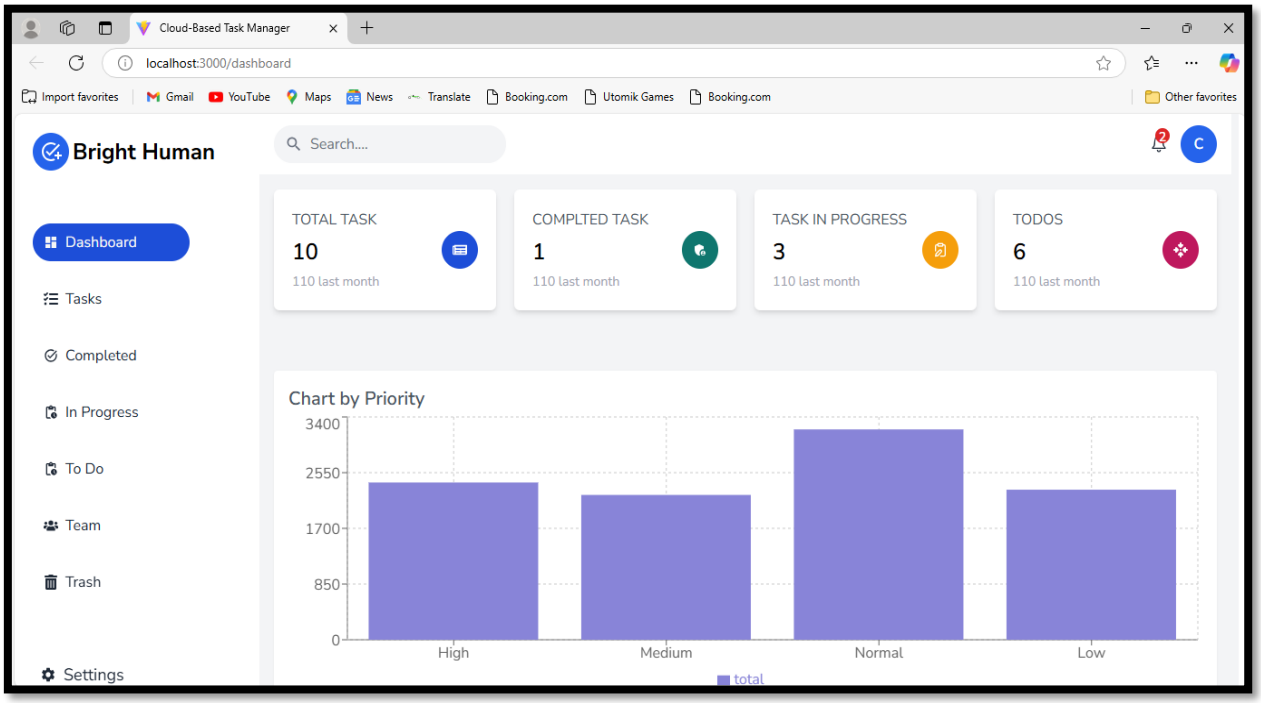


Fig. 8.1 Main Dashboard

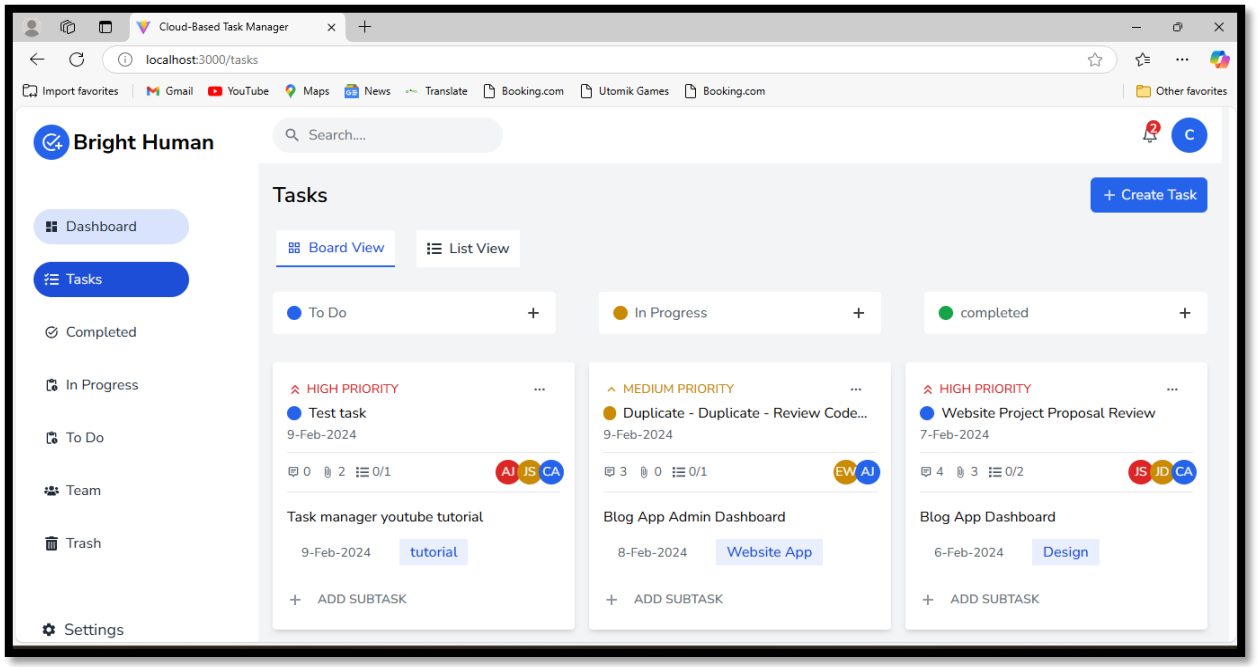


Fig. 8.2 Tasks

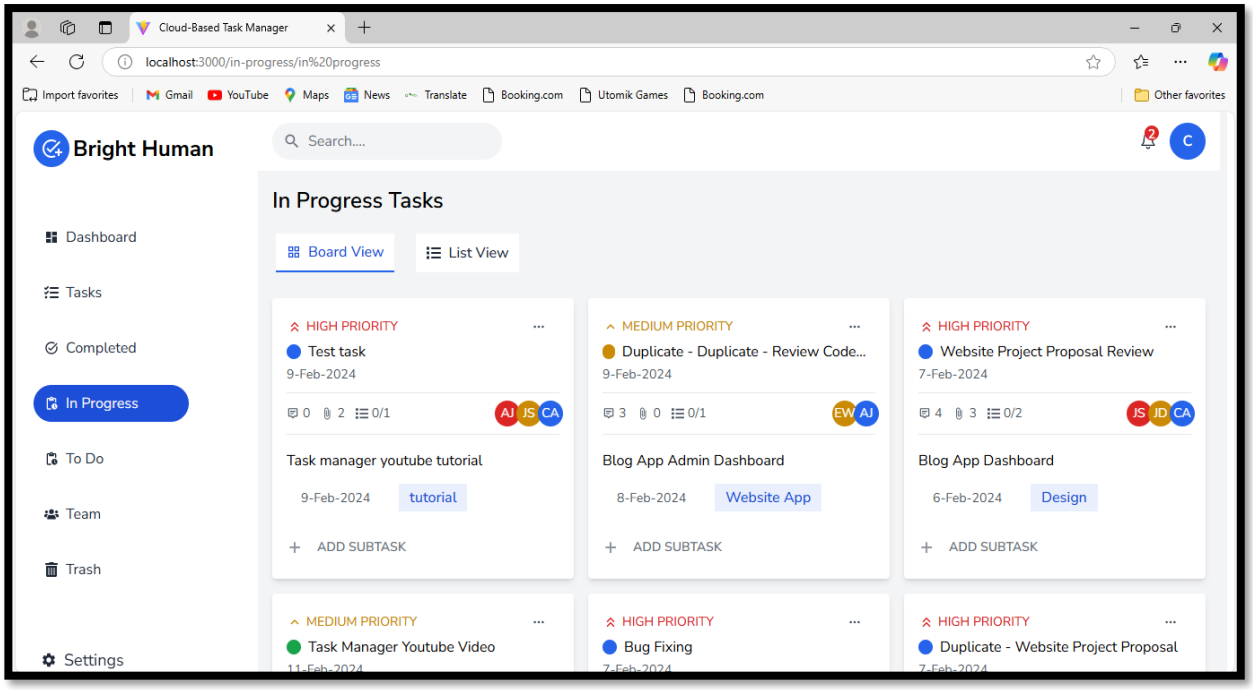


Fig. 8.3 In Progress Tasks

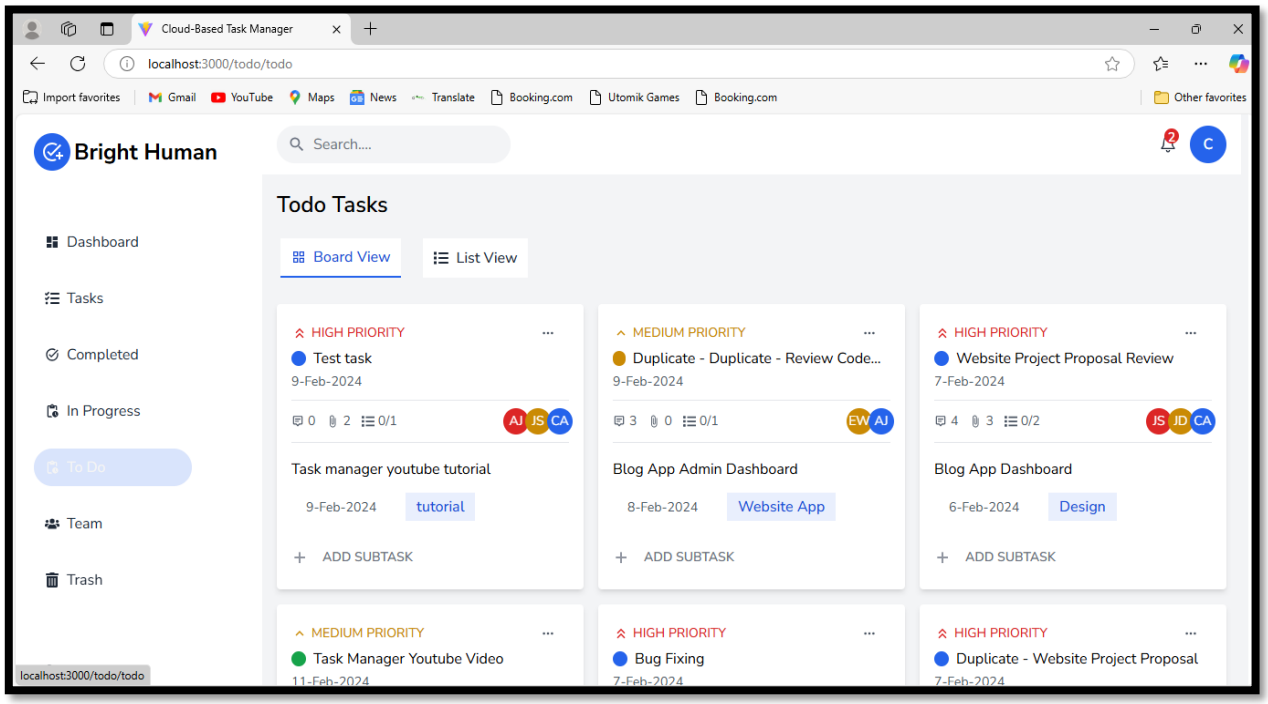


Fig. 8.4 Todo Tasks

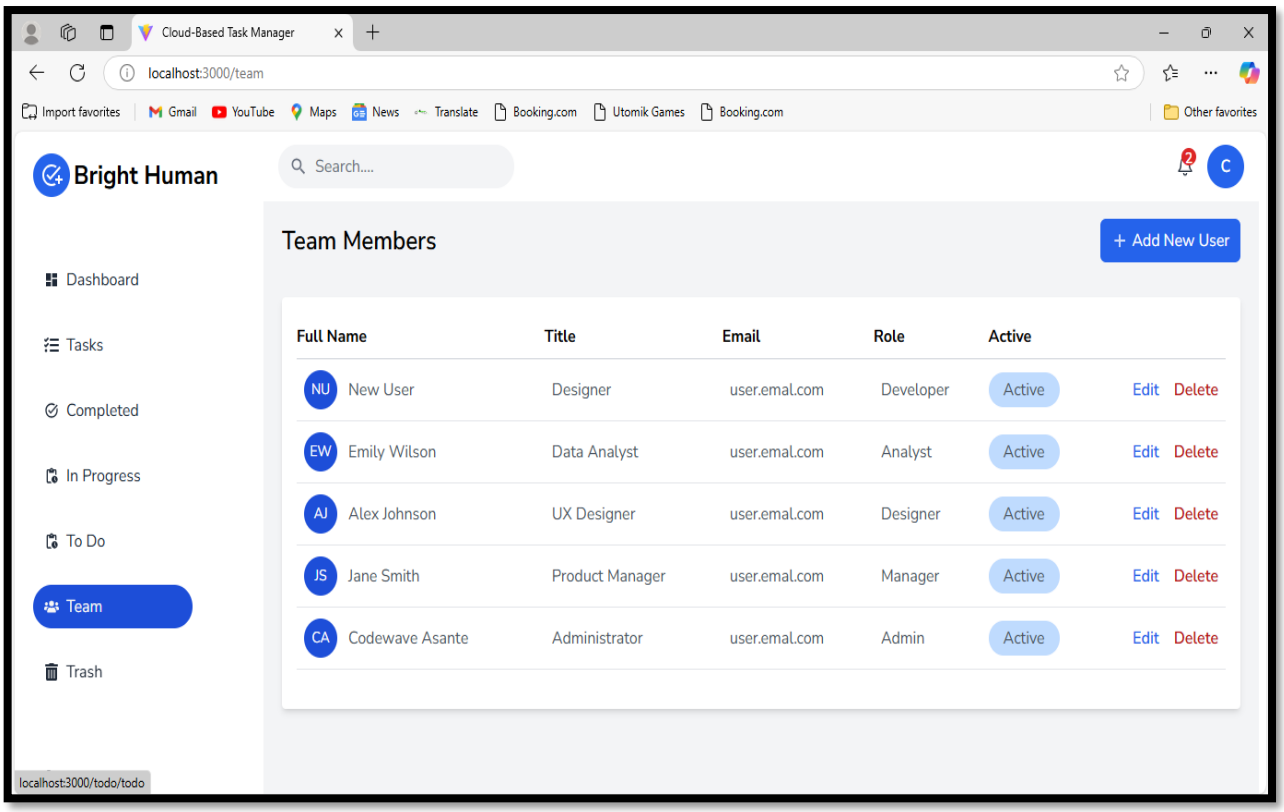


Fig. 8.5 Team Members

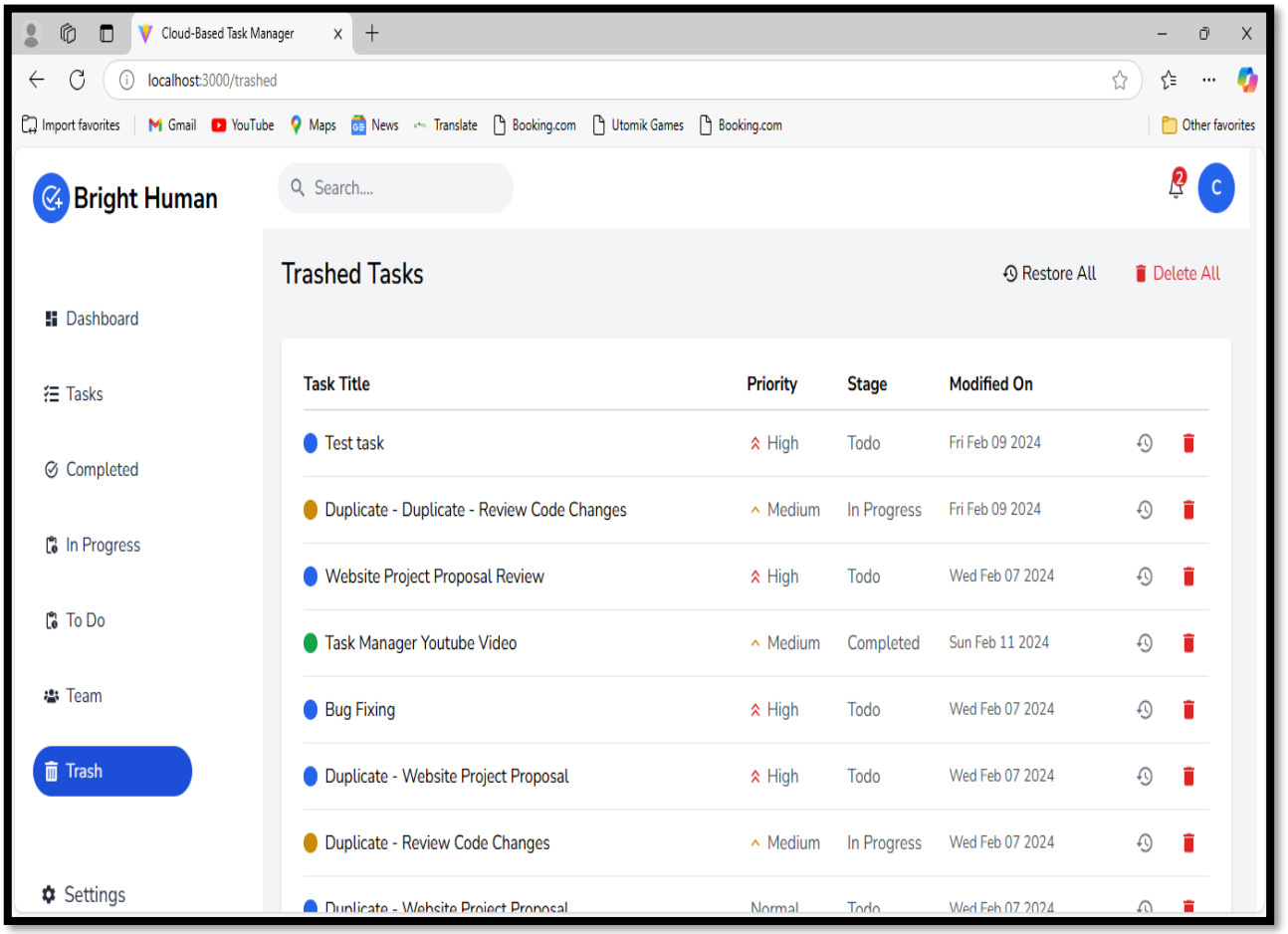


Fig. 8.6 Trashed Tasks

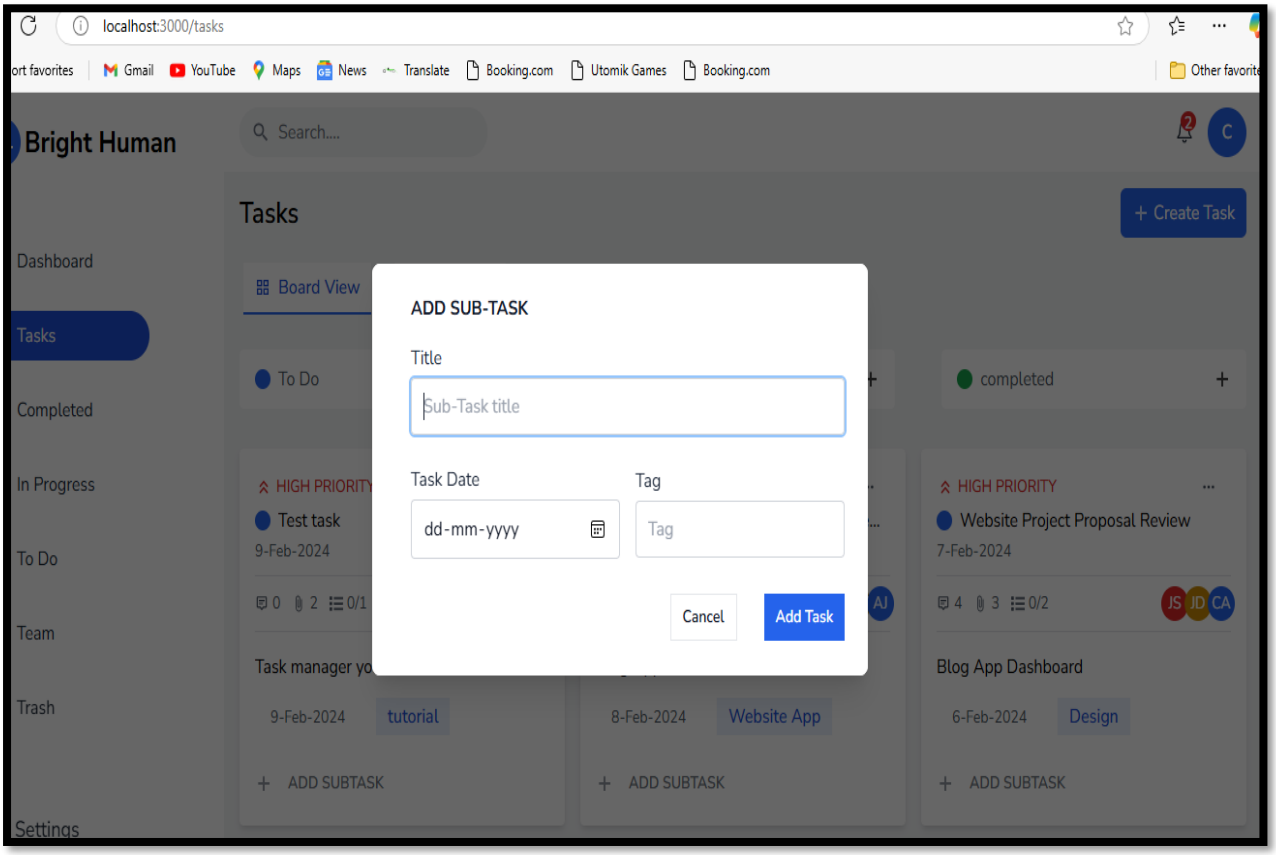


Fig. 8.7 Add Sub Task

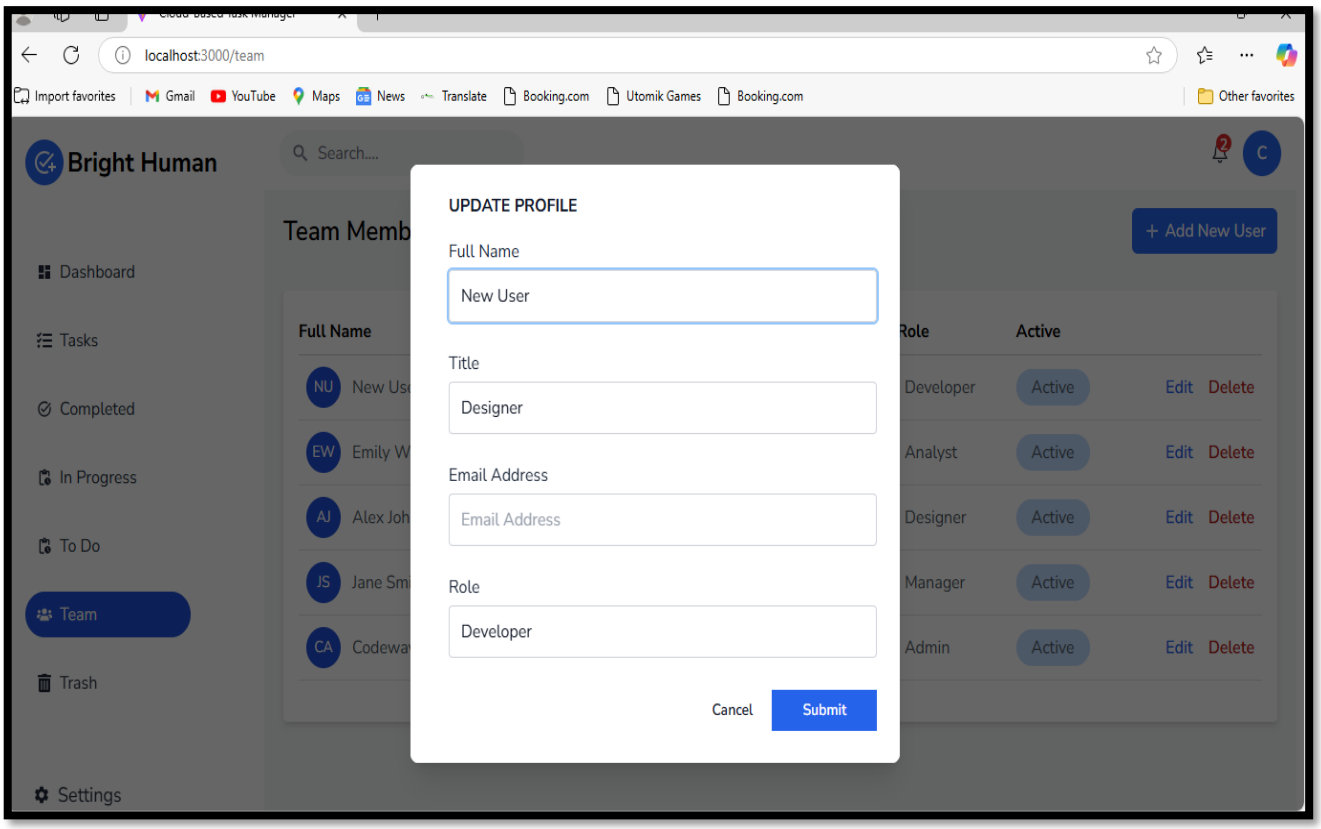


Fig. 8.8 Update Profile

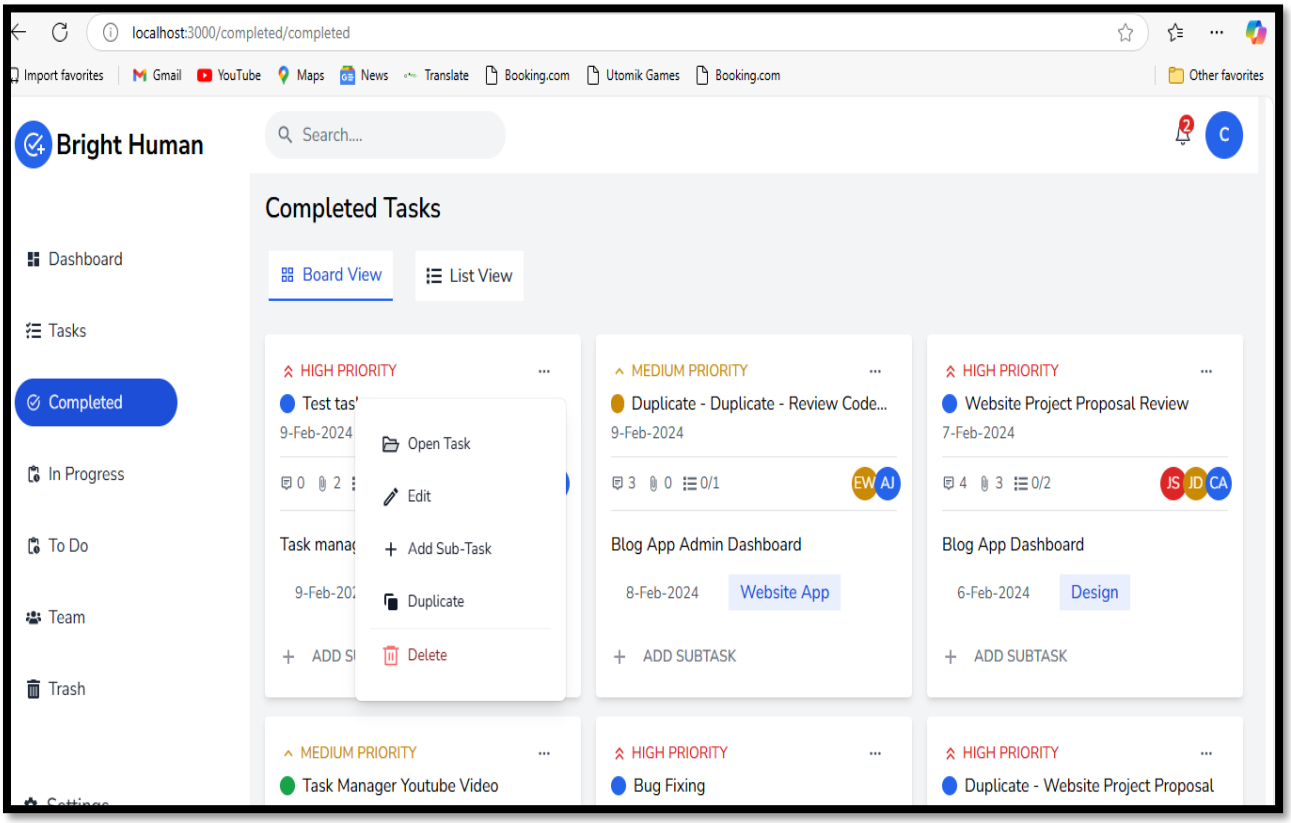


Fig. 8.9 Completed Tasks

REFERENCES

- Liu, X., Zhang, Y., & Wang, L.“ Personalized Task Scheduling and Monitoring Using Machine Learning”, Journal of Computer-Supported Cooperative Work, Volume 6, Issue 6, June 2023

- Singh, A., & Gupta, R “Behavioral Analysis and Prediction of User Productivity in Task Management Applications”, International journal of innovative research in technology, Volume 49, Issue 11, 2022

- Kim, J., & Park, H. “Enhancing Goal Achievement through Adaptive Scheduling Algorithms”, International Journal of Productivity and Performance Management, Volume 8, Issue 2, 2023

- Brown, T., & Johnson, M. “A Comprehensive Review of Productivity Tools and Their Impact on Goal Achievement”, Journal of Web Engineering, Volume 5, Issue 1, 2024

- Patel, N., & Lee, A. “Secure and Scalable Personal Data Management with MongoDB and Node.js"” , International Journal of Creative Research Thoughts, Published in Theseus, 2023