Task Manager App

A PROJECT REPORT

Submitted by

Mohit Sharma - 23BAI70733

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

Chandigarh University

November, 2025

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1 Introduction to Project

In today’s fast-paced world, managing daily tasks efficiently is very important. People often struggle to remember their work, deadlines, and personal responsibilities. A Task Manager application helps users to organize their tasks in a simple and systematic way. It allows users to add tasks, update them when needed, and delete tasks that are completed.

This project is a **Task Manager Web Application** developed using **HTML, CSS, JavaScript** for the front-end and **Node.js with Express.js** for the back-end. The data is stored in **MongoDB Atlas**, which is a cloud-based database. The application provides a user-friendly interface and helps users maintain their daily work plan easily.

1.2 Identification of Problem

The main aim of this project is to create a simple, responsive, and functional web application where users can manage their tasks efficiently. The system focuses on making task management easier rather than providing complex features. It is especially useful for students, office workers, or anyone who needs to keep track of their daily activities.
This project demonstrates the use of the **MERN-like technology stack** (without React) and shows how front-end and back-end work together with a database to create a dynamic web application. The project also includes concepts of CRUD operations (Create, Read, Update, Delete), which are fundamental in web development.

CHAPTER 2: BACKGROUND STUDY

A Task Manager application is a commonly used tool for organizing daily activities and improving productivity. Before developing this project, it is important to understand the background concepts and technologies involved. This chapter explains the basic idea of task management and the technologies used to build the application.

2.1 Task Management

Task management refers to the process of planning, tracking, and completing tasks. Earlier, people used notebooks or reminders to manage their work. However, with increasing workload and digital transformation, computer-based task management systems are more effective. They allow users to store tasks easily, update status, and remove completed tasks instantly.

2.2 Need for a Web-Based Task Manager

With the growth of web applications, users prefer systems that can be accessed anytime and from anywhere. A web-based task manager:

- Does not require installation
- Works on mobile and computer browsers
- Stores data online securely
- Makes task handling faster and more convenient

This makes a web-based system better than manual or offline methods.

2.3 Full Stack Web Development

Full stack development involves both **front-end** (client side) and **back-end** (server side) development.

- The **front-end** deals with user interface and how the user interacts with the system.
- The **back-end** handles data processing, storage, and business logic.

This project uses:

- **HTML** for structure
- **CSS** for styling
- **JavaScript** for interaction
- **Node.js & Express.js** for server and API handling
- **MongoDB Atlas** for cloud database storage

CHAPTER 3: DESIGN FLOW/PROCESS

The design flow of the Task Manager Web Application describes how the system works from the user's point of view, and how each component interacts in the background. The development process followed a structured approach to ensure the system is simple, efficient, and user-friendly.

3.1 System Workflow

The overall workflow of the system is simple:

1. **User opens the application** in the web browser.
2. The **Home Page** displays the task list and an input field to add a new task.
3. When the user **adds a task**, it is sent to the server and stored in the database.
4. The application then **retrieves and displays all stored tasks** from the database.
5. The user can **edit/update a task** if any change is required.
6. The user can **delete a task** when it is completed.
7. The database is updated accordingly each time.

3.2 Architecture of the System

```
User (Browser)
     |
     |  (HTTP Requests)
     v
Front-End (HTML, CSS, JavaScript)
     |
     |  (API Calls / Fetch Requests)
     v
Back-End (Node.js + Express.js Server)
     |
     | (Database Queries)
     v
Database (MongoDB Atlas)
```

3.3 Front-End Design

- The front-end provides the **user interface**.
- It contains an input box to enter a task and buttons to **Add**, **Update**, and **Delete** tasks.
- **CSS** is used to style the web page and make it visually clean and responsive.
- **JavaScript** handles dynamic changes such as updating the list without refreshing the page.

3.4 Back-End Design

- The back-end is built using **Node.js and Express.js**.
- It handles all incoming requests and communicates with the database.
- It contains routes such as:
  - POST /addTask → to create a task
  - GET /tasks → to display tasks
  - PUT /updateTask → to modify a task
  - DELETE /deleteTask → to remove a task

3.5 Database Design

- **MongoDB Atlas** stores data in the form of documents inside a collection.
- Each task is stored as a document with fields like:
- {
-   id: unique_id,
-   taskName: "Task Description"
- }
- The database allows easy storage and retrieval of tasks.

3.6 Design Considerations

- The system should be **simple and easy to use**.
- The interface must be **responsive and clean**.
- Data must be **stored securely in the cloud**.
- The system should allow **fast updates** without page refresh (handled by JavaScript and Express API).

CHAPTER 4: RESULTS ANALYSIS AND VALIDATION

After successful implementation, the Task Manager web application performs the following operations effectively:

1. Add Task:
   The user can enter a task and add it to the list. The new task is immediately stored in the database and shown in the interface.
2. View Tasks:
   All saved tasks are displayed on the screen every time the application is opened. Tasks are fetched from MongoDB Atlas.
3. Update Task:
   The user can modify an existing task. The updated task replaces the old one in the database and is immediately reflected on the screen.
4. Delete Task:
   The user can remove any task that is completed. The task is deleted from the database and removed from the task list.

These results show that the system performs CRUD operations (Create, Read, Update, Delete) correctly.

CHAPTER 5: CONCLUSION AND FUTURE WORK

5.1 Conclusion

The Task Manager Web Application successfully provides a simple and effective platform for users to manage their daily tasks. The system allows users to add, view, update, and delete tasks easily. The use of HTML, CSS, and JavaScript makes the interface user-friendly, while Node.js and Express.js handle server-side processing. The task data is stored securely in MongoDB Atlas, ensuring that the tasks remain available even after closing or refreshing the application.

Overall, the project demonstrates the essential concepts of full stack development, including client-server interaction, API handling, and database integration. The system meets all the defined objectives and works smoothly for daily task management.

5.2 Future Work

Although the current system works efficiently, several improvements can be made to enhance its functionality:

1. User Authentication:
   Adding login and signup features so each user can have their own personal task list.

2. Task Categories/Priority Levels:
   Users can categorize tasks (e.g., Personal, Study, Work) or mark them as High/Medium/Low priority.

3. Reminder and Notification System:
   Adding alerts to notify users when a task deadline is near.

4. Responsive Mobile App Version:
   The system can be extended into a mobile application for easier access.

5. Task Completion Progress Tracking:
   Showing progress charts or completion percentages to motivate users.

By implementing these features, the Task Manager App can become more advanced, customizable, and useful for a larger number of users.