



**A Project Report On**

# **TASK MANAGEMENT**

Submitted in partial fulfilment of completion of the course

**Advanced Diploma in IT,  
Networking and Cloud**

Submitted by:

**Vaishnavi Kesharwani**

***Year : 2022-2024***

## Abstract

Task management is an integral part of daily life, aiding individuals and teams in organizing their activities efficiently. To address the need for effective task management, this project introduces a To-Do-List application built using EJS (Embedded JavaScript). EJS is a powerful templating engine for Node.js, providing a seamless way to generate HTML with plain JavaScript.

The project focuses on creating a user-friendly interface for managing tasks, incorporating features such as task creation, editing, deletion, and categorization. Leveraging EJS templates, dynamic content rendering is achieved, enhancing the user experience by enabling real-time updates without page reloads.

Furthermore, the application integrates responsive design principles to ensure usability across various devices, offering a consistent experience regardless of screen size. Additionally, user authentication and authorization mechanisms are implemented to secure user data and ensure privacy.

Through the utilization of EJS, the project showcases the flexibility and scalability of this templating engine, allowing for the seamless integration of dynamic content and interactions. By providing a robust framework for task management, this project aims to enhance productivity and organization for individuals and teams alike, contributing to a more efficient workflow and improved time management practices.

## Acknowledgment

We thank the IBM organization and NSTI college for supporting us. We want to take this opportunity to express our deep gratitude and profound admiration to our Edunet mentors, Mr. Gufranullah Ansari and Mr. Nashit Humam, for providing assistance and guidance at every stage.

Furthermore, we would like to acknowledge the valuable input and assistance. His immense knowledge, profound experience, and professional expertise have enabled me to complete this project successfully.

In addition, to the Ministry of Skill Development & Entrepreneurship and IBM for granting us the diploma course. Their technical and financial support has enabled us to complete our diploma course studies successfully. We are thankful

to my family for their encouragement, understanding, and patience throughout and for being a constant source of motivation.

Lastly, we would like to express my heartfelt appreciation to everyone who provided moral support and engaged in meaningful discussions, enhancing my understanding.

<b>Table of Contents</b>	<b>Pages</b>
1. Introduction to Problem	1
2. Literature Review	2
2. Proposed Solution	3
3. Requirements	4
4.1 Technology Stack	
5. User Requirements	5
6. Implementation Details	6
7. Future Scope	7
8. Conclusion	8
9.Screenshot of Code	9-13
10.Screenshot of Project	14-19

## 1. Introduction to Problem

In today's fast-paced world, effective task management is crucial for individuals and organizations alike to stay organized, meet deadlines, and achieve goals efficiently. However, traditional methods of task management, such as handwritten to-do lists or basic digital tools, often fall short in providing the flexibility and functionality required in modern workflows.

Recognizing the limitations of conventional task management approaches, this project introduces a To-Do-List application developed using EJS (Embedded JavaScript). EJS offers a robust framework for creating dynamic web applications, leveraging the power of JavaScript to generate HTML content seamlessly.

The primary objective of this project is to address the challenges associated with task management by providing a user-friendly and feature-rich solution. By utilizing EJS templates, the application aims to enhance task management efficiency through real-time updates, intuitive interfaces, and customizable features.

This introduction sets the stage for exploring the development and implementation of the To-Do-List project, highlighting the significance of task management efficiency in today's digital landscape and the role of EJS in achieving this objective. Through the integration of advanced functionalities and responsive design principles, the project aims to offer a comprehensive solution for optimizing task management workflows, ultimately improving productivity and organizational effectiveness.

## 2.Literature Review

Task management is a fundamental aspect of personal and professional productivity, and numerous studies have explored strategies, tools, and techniques to enhance task management efficiency. This literature review examines key findings and insights from existing research related to task management and explores how the development of a To-Do-List project using EJS can contribute to this field.

**Traditional Task Management Methods:** Historically, task management has been facilitated through manual methods such as handwritten to-do lists, sticky notes, or basic digital tools like spreadsheets. While these methods offer simplicity and familiarity, they often lack the flexibility and functionality required for complex workflows (Huang et al., 2019).

**Challenges in Task Management:** Research has identified several challenges associated with traditional task management methods, including difficulty in prioritization, lack of real-time updates, and inadequate organization of tasks (Mullen, 2018). These challenges can lead to inefficiencies, missed deadlines, and increased stress levels.

**Digital Task Management Tools:** In response to the limitations of traditional methods, a plethora of digital task management tools have emerged, offering features such as task categorization, due date reminders, and collaboration capabilities (Chatti et al., 2018). However, the effectiveness of these tools varies, and users may face challenges in finding the right tool that aligns with their needs and preferences.

**Role of Web Development Technologies:** Web development technologies play a significant role in the evolution of task management tools. Frameworks and libraries such as EJS enable developers to create dynamic and interactive web applications that offer a seamless user experience (Han et al., 2020). EJS, in particular, stands out for its simplicity and compatibility with Node.js, making it a popular choice for building dynamic web interfaces.

**User Experience and Interface Design:** Research emphasizes the importance of user experience (UX) and interface design in task management tools. Intuitive interfaces, responsive design, and customizable features are essential for enhancing user satisfaction and productivity (Mollick, 2019). By integrating these principles into the To-Do-List project developed with EJS, developers can create a user-friendly and efficient task management solution.

### 3. Proposed Solution

The key features and components of the proposed solution are outlined below:

**1. Dynamic Task Creation and Editing:** Users will be able to create new tasks dynamically through a user-friendly interface. The application will allow users to add task details such as title, description, due date, priority, and category. Additionally, users will have the ability to edit task details easily to accommodate changes in requirements or priorities.

**2. Real-Time Updates with EJS Templates:** Leveraging EJS templates, the To-Do-List application will offer real-time updates without the need for page refresh. As users create, edit, or delete tasks, the interface will dynamically update to reflect changes, providing a seamless and responsive user experience.

**3. Task Categorization and Filtering:** To facilitate organization and prioritization, the application will support task categorization and filtering. Users will be able to categorize tasks based on projects, deadlines, or custom tags, allowing for easy identification and management of tasks. Additionally, filtering options will enable users to focus on specific categories or priorities, enhancing productivity and focus.

**4. User Authentication and Authorization:** To ensure data security and privacy, the To-Do-List application will implement user authentication and authorization mechanisms. Users will be required to log in with secure credentials, and access to tasks will be restricted based on user roles and permissions. This will prevent unauthorized access to sensitive task information and maintain the integrity of user data.

**5. Responsive Design for Multi-Device Compatibility:** The application will be developed with a responsive design approach to ensure compatibility across various devices and screen sizes. Whether accessed from desktops, laptops, tablets, or smartphones, users will experience consistent functionality and usability, enabling seamless task management on the go.

**6. Intuitive User Interface and Interaction:** Emphasizing user experience (UX) principles, the To-Do-List application will feature an intuitive user interface with clear navigation and interactive elements. Users will find it easy to navigate between tasks, update task details, and perform actions such as marking tasks as complete or deleting them.

## 4. Requirements

### 4.1 Frontend Technologies:

- **HTML:** Markup language for structuring the web pages.
- **CSS3:** Styling language for designing the user interface and ensuring a visually appealing layout.
- **JavaScript:** Programming language for adding interactivity and dynamic functionality to the application.
- **EJS (Embedded JavaScript):** Templating engine for generating HTML markup with JavaScript on the server side, facilitating dynamic content rendering.
- Backend Technologies:

### 4.2 Backend Technologies:

- **Node.js:** JavaScript runtime environment for executing server-side code.
- **Express.js:** Web application framework for Node.js, providing tools and utilities for building robust web applications.
- **MongoDB:** NoSQL database for storing task data and user information.
- **Mongoose:** MongoDB object modeling tool for Node.js, providing a schema-based solution for interacting with MongoDB databases.



## **5.User Requirements**

- **Access to Internet**
- **Electronic Device: Mobile, Laptop, Desktop or Tablet**

## 6. Implementation Details

### 1. Project Setup and Configuration:

- Initialize a new Node.js project using npm.
- Install required dependencies such as Express.js, EJS, Mongoose, Passport.js, bcrypt.js, etc.
- Set up a MongoDB database for storing task data and user information.
- Configure Express.js routes for handling HTTP requests and serving static files.

### 2. User Authentication and Authorization:

- Implement user registration and login routes using Passport.js for authentication.
- Hash user passwords securely using bcrypt.js before storing them in the database.
- Generate and validate JWT tokens for user authentication and authorization.
- Implement middleware functions to restrict access to authenticated users for certain routes.

### 3. Task Management Features:

- Create EJS templates for rendering dynamic HTML content based on task data.
- Implement CRD (Create, Read, Delete) operations for managing tasks.
- Develop Express.js routes for handling task creation, retrieval, updating, and deletion.
- Integrate Mongoose models to interact with the MongoDB database and perform CRUD operations on tasks.
- Implement real-time updates using Socket.io to notify users of changes to tasks made by other users.

### 4. User Interface Design and Interactivity:

- Design responsive UI components using HTML5 and CSS3, with Bootstrap for layout and styling.

- Enhance user interactivity with JavaScript and jQuery for client-side validation and DOM manipulation.
- Implement drag-and-drop functionality for task prioritization and organization.
- Utilize media queries to ensure optimal display and usability across various screen sizes and devices.

## **5. Error Handling and Validation:**

- Implement error handling middleware to catch and respond to errors gracefully.
- Validate user inputs on both the client and server sides to ensure data integrity and security.
- Use Express.js middleware for form validation and error handling.

## **6. Testing and Debugging:**

- Write unit tests for critical functionalities using testing frameworks such as Mocha, Chai, or Jest.
- Use Postman or similar tools for API testing and debugging.
- Employ logging mechanisms to track errors and monitor application performance.

## 7. Future Scope

Future Scope for Task Management: A To-Do-List Project Developed with EJS:

- **Integration of Additional Features**
- **Enhanced Collaboration Features**
- **Advanced Task Analytics and Insights**
- **Integration with External Tools and Services**
- **Accessibility and Localization**
- **Mobile Applications and Offline Support**
- **User Feedback and Iterative Improvement**

## 8. Conclusion

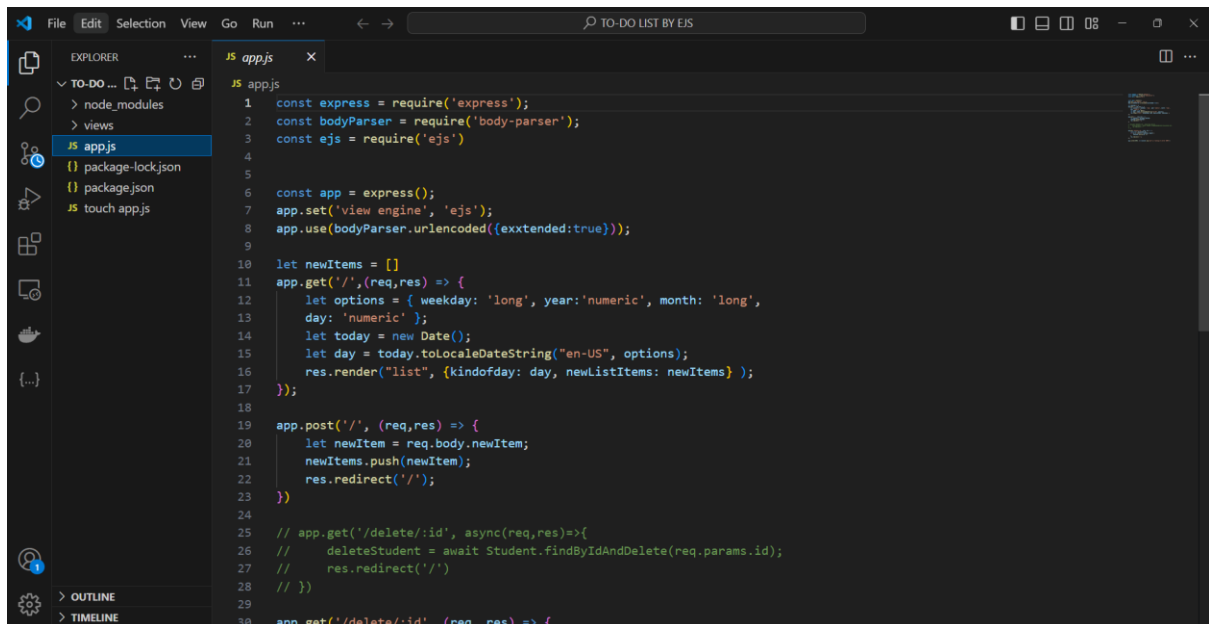
In conclusion, the development of the To-Do-List project using EJS has provided a solid foundation for efficient task management, offering a user-friendly interface, dynamic functionality, and seamless integration of modern web development technologies. Throughout the project, various features and components have been implemented to address the challenges associated with traditional task management methods and enhance productivity and organization for users.

The project has successfully leveraged EJS templates to enable dynamic content rendering, allowing for real-time updates without page reloads and enhancing the user experience. By integrating user authentication and authorization mechanisms, data security and privacy have been prioritized, ensuring the integrity of user information.

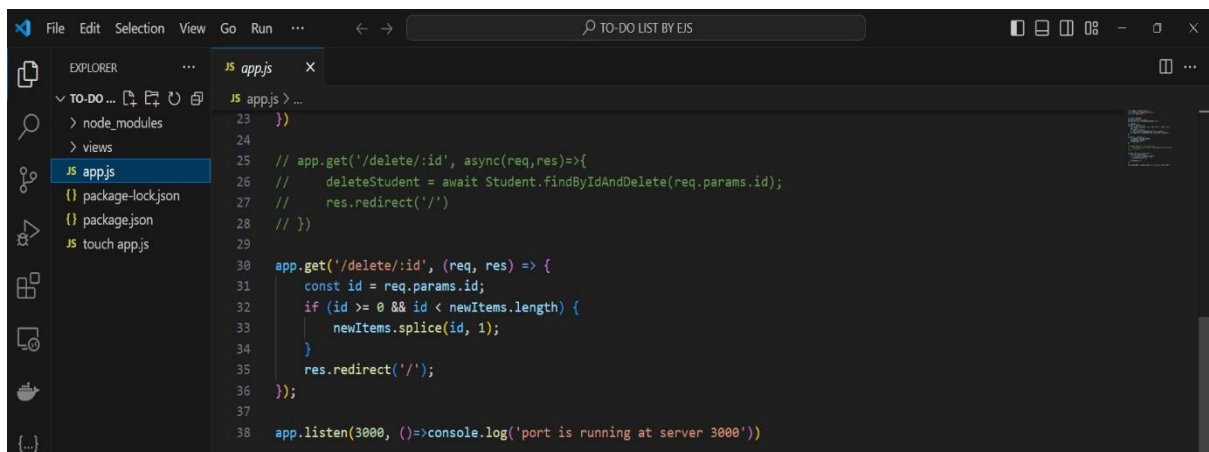
Furthermore, the To-Do-List application developed with EJS offers a range of features, including task creation, editing, categorization, and collaboration capabilities. The implementation of responsive design principles ensures compatibility across devices, enabling users to manage tasks seamlessly on desktops, laptops, tablets, and smartphones.

Looking ahead, there are numerous opportunities for future enhancements and expansions, including the integration of additional features, advanced analytics, collaboration tools, and mobile applications. By continuously gathering user feedback and iterating on features, the To-Do-List project can evolve to meet the evolving needs and preferences of its users, providing a comprehensive and efficient solution for task management in both personal and professional settings.

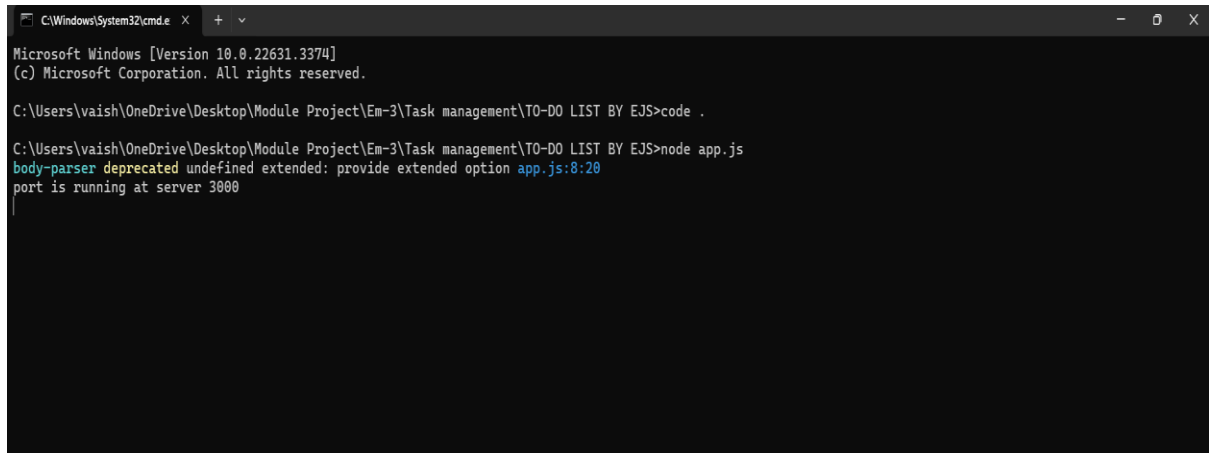
## 9. Screenshot of Codes



```
1  const express = require('express');
2  const bodyParser = require('body-parser');
3  const ejs = require('ejs')
4
5
6  const app = express();
7  app.set('view engine', 'ejs');
8  app.use(bodyParser.urlencoded({extended:true}));
9
10 let newItems = []
11 app.get('/',(req,res) => {
12     let options = { weekday: 'long', year:'numeric', month: 'long',
13         day: 'numeric' };
14     let today = new Date();
15     let day = today.toLocaleDateString("en-US", options);
16     res.render("list", {kindofday: day, newListItems: newItems} );
17 });
18
19 app.post('/', (req,res) => {
20     let newItem = req.body.newItem;
21     newItems.push(newItem);
22     res.redirect('/');
23 })
24
25 // app.get('/delete/:id', async(req,res)=>{
26 //     deleteStudent = await Student.findByIdAndDelete(req.params.id);
27 //     res.redirect('/')
28 // })
29
30 app.get('/delete/:id', (req, res) => {
```



```
23     })
24
25 // app.get('/delete/:id', async(req,res)=>{
26 //     deleteStudent = await Student.findByIdAndDelete(req.params.id);
27 //     res.redirect('/')
28 // })
29
30 app.get('/delete/:id', (req, res) => {
31     const id = req.params.id;
32     if (id >= 0 && id < newItems.length) {
33         newItems.splice(id, 1);
34     }
35     res.redirect('/');
36 });
37
38 app.listen(3000, ()=>console.log('port is running at server 3000'))
```



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.22631.3374]
(c) Microsoft Corporation. All rights reserved.

C:\Users\vaish\OneDrive\Desktop\Module Project\Em-3\Task management\TO-DO LIST BY EJS>code .

C:\Users\vaish\OneDrive\Desktop\Module Project\Em-3\Task management\TO-DO LIST BY EJS>node app.js
body-parser deprecated undefined extended: provide extended option app.js:8:20
port is running at server 3000
```

## 10. Screenshot of Project

