WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

DESIGN AND DEVELOPMENT OF A TO-DO REMINDER APPLICATION USING NODE.JS, EXPRESS, AND MONGODB

Pagadala Isaac[1], Dr. G.N.R. Prasad[2], P. Rambabu[3]*

[1] MCA Student, E-Mail: isaacpagadala@gmail.com
[2] Sr. Asst. Professor, E-Mail: gnrp@cbit.ac.in
[3] Asst. Professor, E-Mail: rambabup_mca@cbit.ac.in

ABSTRACT :

The growing dependence on digital tools for task and time management has created a demand for intuitive and efficient reminder applications. This paper presents the design and development of a To-Do Reminder Application built using Node.js, Express.js, MongoDB, HTML, CSS, and JavaScript. The project aimed to provide users with a lightweight, web-based platform to create, view, update, and delete reminders with real-time responsiveness. The system architecture followed a full-stack approach, integrating a simple frontend interface with a scalable backend and a NoSQL database for data persistence. RESTful APIs were implemented to manage communication between the frontend and backend, while Git and GitHub were used for version control. The resulting application demonstrates CRUD operations, user interaction through dynamic forms, and persistent reminder storage, highlighting key principles of full-stack development and practical implementation of modern web technologies.

Keywords: To-Do Application, Task Management, Full-Stack Development, Node.js, Express.js, MongoDB, RESTful APIs, CRUD Operations.

1.0 INTRODUCTION

Task management has become an essential part of productivity in both personal and professional contexts. Forgetting deadlines and activities without a reminder system can negatively impact efficiency. The objective of this project was to design and develop a To-Do Reminder Application that allows users to set, update, and manage tasks effectively.

The overall aim was to gain hands-on experience in full-stack web development, covering frontend design, backend logic, and database integration. By developing this application, the project bridged the gap between theoretical knowledge of web technologies and their real-world implementation.

2.0 SYSTEM STUDY / REQUIREMENT ANALYSIS

2.1 User Analysis

- **Target Audience:** Students, professionals, and individuals needing a lightweight task reminder system.
- **User Environment:** Accessible via modern web browsers on desktops and laptops.
- **User Requirements:** Ability to add, view, update, and delete reminders with attributes like title, description, priority, and due date.

2.2 Functional Requirements

- Reminder creation with title, description, importance, and date.
- CRUD operations (Create, Read, Update, Delete).
- Real-time updates on reminder lists.
- Notifications or confirmation messages for user actions.

2.3 Non-Functional Requirements

- **Performance:** Fast CRUD operations and database response.
- **Usability**: Clean, simple, and responsive UI.
- **Reliability**: Data persistence with MongoDB.
- **Scalability:** Extendable for additional features like notifications or multi-user support.

2.4 Technology Stack Used

- **Frontend:** HTML5, CSS3, JavaScript.
- **Backend**: Node.js with Express.js framework.
- **Database**: MongoDB (NoSQL).
- **Tools**: Git & GitHub for version control, VS Code as IDE, Postman for API testing.

3.0 SYSTEM DESIGN AND IMPLEMENTATION

The application was designed using a modular approach to ensure maintainability and scalability.

3.1 System Design

- **Use Case Diagram**
  (Insert Fig. 1: Use Case Diagram here – showing user actions such as Add Reminder, View Reminder, Update Reminder, Delete Reminder)
- **Data Flow Diagram (DFD)**
  (Insert Fig. 2: Data Flow Diagram here – depicting data flow between User, Application Server, and MongoDB Database)
- **Entity-Relationship Diagram (ERD)**
  (Insert Fig. 3: ER Diagram here – entities include Reminders with fields Title, Description, Priority, and Due Date)

3.2 Modules Implemented

- **User Interface Module**: HTML/CSS/JS interface for form inputs and displaying reminders.
- **CRUD Module:** Handles reminder creation, retrieval, updating, and deletion.
- **API Module**: RESTful APIs built with Express.js for client-server communication.
- **Database Module**: MongoDB schema for reminders and efficient query management.
- **Version Control Module**: Git and GitHub for project management.
- **Notification Module (Future Scope)**: To alert users about upcoming tasks.

4.0 RESULTS AND DISCUSSION

The application successfully implemented a functioning To-Do Reminder System with the following outcomes:

- **Home Screen**
  (Insert Fig. 4: Home Screen with input form for Title, Description, Importance, and Date)
- **Reminder List Screen**
  (Insert Fig. 5: Reminder List displaying all stored reminders with priority and due date)
- **Update Reminder Screen**
  (Insert Fig. 6: Update Screen showing pre-filled reminder details for editing and saving)
- **Delete Confirmation**
  (Insert Fig. 7: Notification/Confirmation screen upon successful deletion or update)

The backend efficiently handled CRUD operations via REST APIs, and MongoDB ensured persistent storage. Testing with Postman validated API endpoints, and browser testing confirmed UI responsiveness across devices.

5.0 CONCLUSION

This project provided valuable insights into full-stack web development by combining frontend design, backend logic, and database integration. It demonstrated practical implementation of CRUD operations, RESTful APIs, and NoSQL database management.

- **Summary**: A fully functional To-Do Reminder Application was built using modern web technologies.
- **Relevance**: Reinforced academic learning in Web Technologies, DBMS, and Software Engineering.
- **Career Scope**: The skills gained—such as server-side programming, database handling, and client-server communication—can be directly applied in software development roles. Future enhancements could include push notifications, authentication, and mobile deployment.

6.0 REFERENCES

1. Ethan Brown, *Web Development with Node and Express: Leveraging the JavaScript Stack*
2. Brad Traversy, *Modern Full-Stack Development: Using TypeScript, React, Node.js, Webpack, and Docker*
3. Kyle Simpson, *You Don't Know JS* (Book Series)
4. Eric Elliott, *Programming JavaScript Applications*
5. Node.js Documentation – https://nodejs.org/en/docs

6. Express.js Guide – https://expressjs.com/en/starter/installing.html
7. MongoDB Documentation – https://www.mongodb.com/docs
8. Postman API Tool – https://www.postman.com/
9. MDN Web Docs (HTML/CSS/JS) – https://developer.mozilla.org/
10. GitHub – https://github.com
11. Stack Overflow – https://stackoverflow.com
12. Medium Technical Blogs – https://medium.com/tag/web-development