# FSD Project Report

# 

# 

# Project Title:    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# 

# Submitted by

# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Submitted to

# Ms. Pooja Prajapati

# 

# Vivekanand Education Society’s Institute of Technology Chembur,

# Mumbai – 400 074

# 

# Declaration

# I,\_\_\_\_\_\_\_\_\_\_\_\_, Class:  \_\_\_\_\_\_\_\_, student of Semester \_\_\_\_\_\_,

# Third year Engineering from Department of Information Technology, V. E. S. Institute of Technology, hereby declare that the project titled \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ,submitted to the University of Mumbai for the academic year \_\_\_\_\_\_\_\_\_\_\_\_, is a record of an original work done by me under the guidance of Ms. Pooja Prajapati, Assistant Professor, Department of Information Technology, V.E.S. Institute of Technology, Mumbai -400074.

# This project report is submitted as a partial fulfillment of the requirement of the subject Full Stack Development in Semester \_\_\_\_\_\_\_\_\_\_\_\_.

# 

# 

# 

# Date:                                                                                                                     Name:

# 

# 

# Place:                                                                                                               Signature:

# CERTIFICATE

# This is to certify that the report titled

# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# that is being submitted by : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# from Class \_\_\_\_\_\_\_\_\_\_\_\_ , Department of Information Technology, V.E.S Institute of Technology

# In partial fulfillment for the subject Full Stack Development to the University of Mumbai is a record of bonafide work carried out by him/her under my guidance and supervision during the Academic Year \_\_\_\_\_\_\_\_\_\_\_\_.

# 

# 

# 

# 

# Ms. Pooja Prajapati

# Assistant Professor

# Department of Information Technology

# V.E.S. Institute of Technology Mumbai - 400 074.

# Mini Project Report

## Project Title: Real-Time Task Manager (MERN Stack)

### 1. Objective

The goal of this project is to design and implement a **real-time collaborative task management system** that allows multiple users to:

* Register and log in securely using JWT-based authentication.
* Create, update, and view tasks in real-time.
* Collaborate effectively with live updates using **WebSockets**.
* Manage user roles and ensure secure access using REST APIs.

### 2. Technologies Used

|  |  |  |
| --- | --- | --- |
| **Layer** | **Technology** | **Description** |
| **Frontend** | React.js + Tailwind CSS | For building an interactive and responsive UI. |
| **State Management** | React Context API | To manage global authentication and task state. |
| **Backend** | Node.js + Express.js | RESTful API server for handling requests and responses. |
| **Database** | MongoDB + Mongoose | NoSQL database to store users and task data. |
| **Authentication** | JWT (JSON Web Token) | For secure login and access control. |
| **Real-time Communication** | Socket.IO | For real-time task updates between users. |
| **Testing Tool** | Postman | For API testing and validation. |
| **Deployment** | Render / Vercel / Docker | For cloud hosting and CI/CD deployment. |

### 3. System Architecture

#### Architecture Overview:

Frontend (React) → REST API (Express + Node.js) → MongoDB Database

↘────── WebSockets (Socket.IO) ─────↙

**Flow:**

1. The user registers or logs in through the React frontend.
2. The credentials are sent to the Express backend, which validates and returns a JWT token.
3. The token is stored in React Context for authentication.
4. Users can create or modify tasks; changes are saved in MongoDB.
5. Socket.IO updates all connected clients instantly.

### 4. Functional Modules

#### 1 User Authentication

* Register new users with hashed passwords using bcrypt.
* JWT-based login and token verification middleware.
* Context API manages user sessions on the client side.

#### 

#### 

#### 2 Task Management

* Users can add, update, and delete tasks.
* Tasks are categorized as **To-Do**, **In-Progress**, and **Done**.
* Each update triggers a real-time refresh using Socket.IO.

#### 3 Dashboard

* Displays all tasks visually.
* Users can drag tasks across categories.
* Status updates are instantly reflected for all users online.

#### 4 API Endpoints

|  |  |  |
| --- | --- | --- |
| **Method** | **Endpoint** | **Description** |
| POST | /api/auth/register | Create new user |
| POST | /api/auth/login | Authenticate user and return JWT |
| GET | /api/tasks | Fetch all tasks |
| POST | /api/tasks | Add new task |
| PUT | /api/tasks/:id | Update task status |
| DELETE | /api/tasks/:id | Delete task |

### 5. Testing

The API was tested using **Postman**:

* Verified successful login and JWT generation.
* Checked that task CRUD operations required valid JWT.
* Tested multiple concurrent users with Socket.IO — updates propagated instantly.

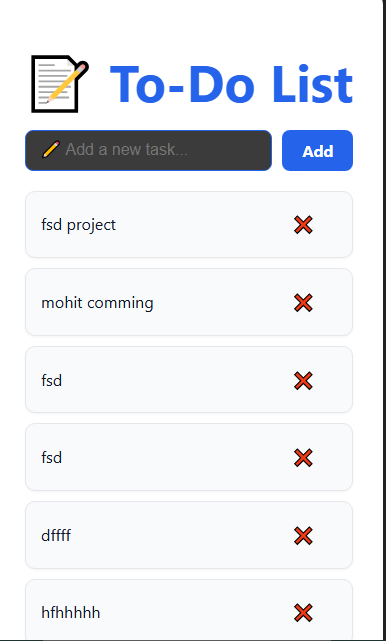
### 6. Results and Discussion

* Achieved **secure login and authentication** using JWT.
* Implemented **real-time updates** through Socket.IO.
* User interface is **responsive and interactive** using TailwindCSS.
* **Context API** provides smooth global state handling.
* MongoDB ensured **scalable storage** and fast data retrieval.

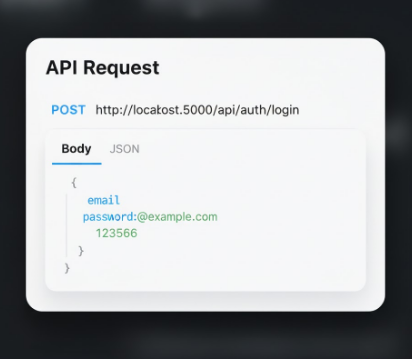
### 7. Screenshots (Suggested for Final Report)

Include:

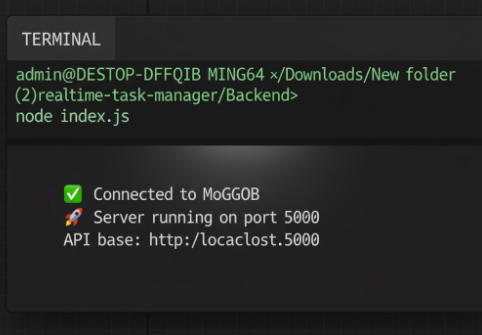
1:TODO\_LIST:



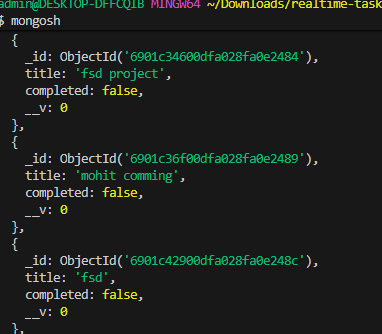
2.Postman API Test



3.VS Code terminal output (Server running)



4.output of mongodb:



### 8. Future Enhancements

* Add role-based access (Admin / User).
* Include notifications for task updates.
* Integrate file uploads or comments for each task.
* Deploy with Docker containers and CI/CD pipelines.

### 

### 

### 

### 9. Conclusion

This project demonstrates a modern full-stack web application integrating:

* **React Hooks + Context API** for efficient frontend logic.
* **Express + MongoDB** for scalable RESTful APIs.
* **JWT** for secure authentication.
* **Socket.IO** for real-time communication.

It provides a practical implementation of key technologies used in today’s **industry-standard MERN applications**.

### 10. References

* React Documentation
* Express.js Docs
* [MongoDB Docs](https://www.mongodb.com/docs/)
* Socket.IO Docs
* JWT Guide
* Postman Docs