Full Stack Development with MERN

Project Documentation format

1. Introduction

• **Project Title:** Educational Organization Using ServiceNow

• **Team Members: Team ID :** LTVIP2026TMIDS66400

Team Size : 4

Team Leader : Gajjala Bala Dinesh Reddy

Team member : Mangali Dinesh

Team member : Kuruva Dilip

Team member : Muthuru Venkata Lakshmi

2. Project Overview

• **Purpose:** The Educational Organization System is designed to streamline administrative and academic processes within educational institutions. The system provides a centralized platform to manage student records, course information, service requests, and communication between students, teachers, and administrators.

The main goal is to automate manual processes, improve efficiency, and provide easy access to academic information.

Features:

- User registration and login
- Student profile management
- Course management
- Service request submission and tracking
- Result management
- Admin dashboard and report generation
- Notification system

3. Architecture

• **Frontend:** The frontend is built using React.js and provides a responsive user interface. It handles user interaction, form validation, and communicates with backend APIs using HTTP requests.

Technologies Used:

- React.js
- HTML, CSS, JavaScript
- Axios (API communication)

• **Backend:**

The backend handles business logic, authentication, API routing, and data processing.

Technologies Used:

- Node.js
- Express.js
- REST API architecture

• **Database:**

MongoDB is used to store user data, course details, and service requests. Data is stored in collections such as users, courses, and requests.

Operations:

- Data storage
- Data retrieval
- Update and delete operations

4. Setup Instructions

• **Prerequisites:**

- Node.js
- MongoDB
- Git
- Code editor (VS Code recommended)

1. **Installation:** Clone the repository
2. `git clone <repository-url>`
3. Install frontend dependencies
4. `cd client`
5. `npm install`
6. Install backend dependencies
7. `cd server`
8. `npm install`

9. Configure environment variables (.env file).
10. Start MongoDB service.

.

5. Folder Structure

• **Client:** • components → UI components

- pages → Application pages
- services → API calls
- App.js → Main component

• **Server:**

- routes → API routes
- models → Database schema
- controllers → Business logic
- config → Database configuration
- server.js → Entry point

6. Running the Application

• Provide commands to start the frontend and backend servers locally. ○ **Frontend:** `npm start` in the client directory.

Frontend

```
cd client
npm start
```

○ **Backend:** `npm start` in the server directory.

cd server

npm start

7. API Documentation

• Document all endpoints exposed by the backend.
• Include request methods, parameters, and example responses.

Endpoint	Method	Description
/api/register	POST	Register new user
/api/login	POST	User authentication
/api/students	GET	Retrieve student details
/api/courses	GET	Retrieve course details
/api/request	POST	Submit service request
/api/results	GET	View student results

Example Response:

```
{
  "status": "success",
  "message": "User registered successfully"
}
```

8. Authentication

• Explain how authentication and authorization are handled in the project. • Include details about tokens, sessions, or any other methods used.

Authentication is implemented using JWT (JSON Web Token).

- Secure login with email and password
- Token-based session management
- Role-based access control for admin and users

9. User Interface

• Provide screenshots or GIFs showcasing different UI features.

The system provides:

- Registration page
- Login page
- Student dashboard
- Admin dashboard
- Service request interface
- Result viewing page

(Screenshots can be attached here.)

10. Testing

• Describe the testing strategy and tools used.

Testing includes:

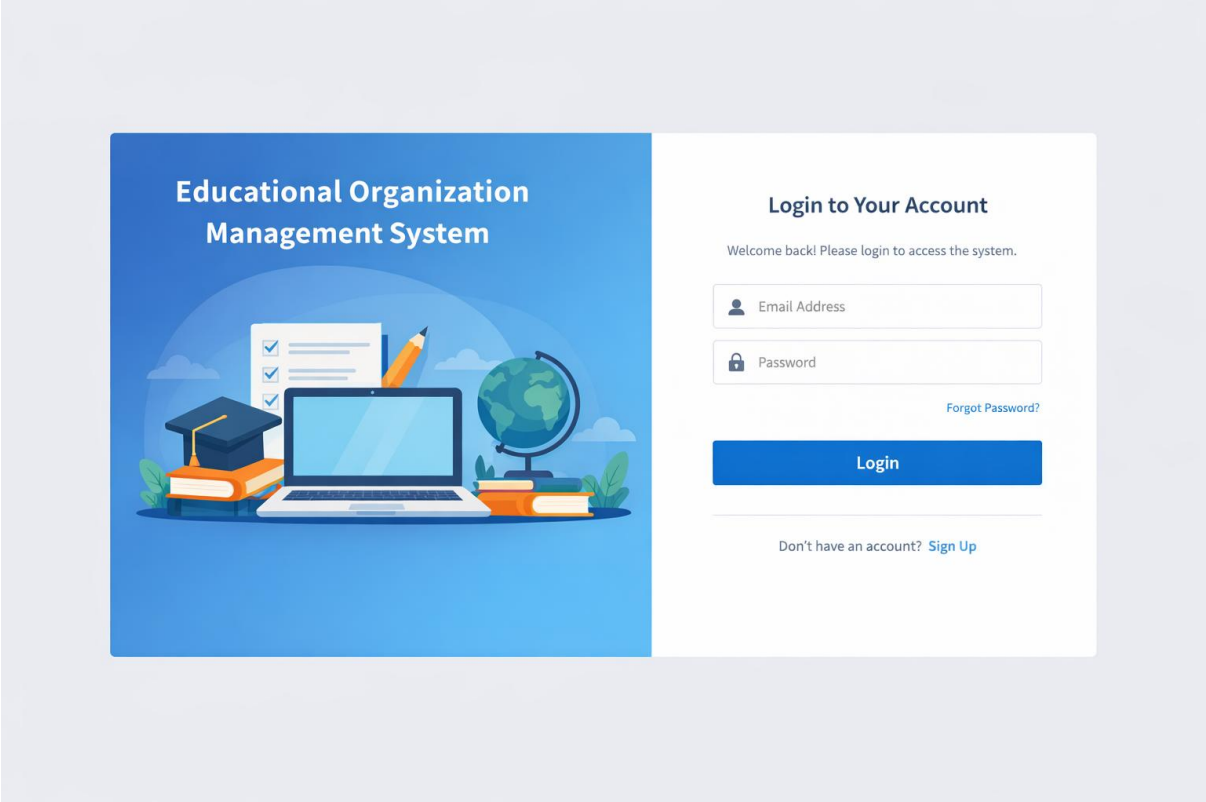
- Unit testing for backend APIs
- Manual testing for UI components
- Integration testing for database operations

Tools used:

- Postman (API testing)
- Browser testing

11. Screenshots or Demo

• Provide screenshots or a link to a demo to showcase the application.



12. Known Issues

• Document any known bugs or issues that users or developers should be aware of.

- Performance may reduce with large datasets.
- Limited third-party integration.
- UI improvements required for better user experience.

13. Future Enhancements

• Outline potential future features or improvements that could be made to the project.

- Mobile application support
- AI-based analytics for student performance
- Real-time notifications
- Advanced reporting dashboard
- Integration with external educational services