Project Report: Debatverse

1. Project Overview

Project Name: Debatverse

Technology Stack: MERN (MongoDB, Express.js, React.js, Node.js)

Purpose: Create a platform for users to participate in and respond to debates while enabling administrators to manage the platform.

Debatverse is a web application designed to foster discussion and engagement on various topics through structured debates. Users can log in, create debates, respond to them, and explore features like sorting and filtering debates.

2. Key Features

Authentication and Authorization

Login/Signup: Users and admins can log in securely using JSON Web Tokens (JWT).

Role-based Access:

Admin: Full access to manage users, debates, and the platform.

User: Limited access to create debates and respond to them.

Debate Management

Create Debate: Users can post a debate question with 2-6 options.

Respond to Debate: Users can select an option and submit their response.

Sorting and Filtering: Users can sort debates by criteria such as popularity or date and filter debates based on categories or keywords.

Admin Features

View, edit, or delete debates.

Manage user accounts.

Analyze user engagement metrics.

3. Technology Breakdown

Frontend (React.js)

Components:

- Login and Registration Pages.

- Dashboard for users and admins.

- Debate creation and response interface.

- Sorting and filtering tools.

State Management: Context API for managing authentication and user data.

Styling: CSS Modules for a modern, responsive design.

Backend (Node.js + Express.js)

Authentication: JWT for secure login and session handling.

API Endpoints:

- User operations: /login, /register, /profile.

- Debate operations: /create-debate, /get-debates, /respond-to-debate.

- Admin operations: /admin/manage-users, /admin/manage-debates.

Middleware: Custom middleware for role-based access control.

Database (MongoDB)

Collections:

- Users: Stores user details, roles, and authentication tokens.

- Debates: Stores debate questions, options, responses, and metadata (e.g., creation date, tags).

- Responses: Stores user responses linked to debates.

4. Development Process

Phase 1: Planning

Defined the project scope and features.

Created wireframes for the UI.

Phase 2: Backend Development

Set up the Node.js and Express.js environment.

Designed the database schema for users, debates, and responses.

Implemented JWT-based authentication and API endpoints.

Phase 3: Frontend Development

Developed the React components for the user interface.

Integrated the frontend with backend APIs.

Implemented sorting and filtering functionality.

Phase 4: Testing and Deployment

Conducted unit testing for backend APIs.

Tested the applications UI for responsiveness and usability.

Deployed the application on a cloud platform (e.g., Heroku or AWS).

5. Challenges and Solutions

1. JWT Integration

Challenge: Managing token expiry and secure storage.

Solution: Implemented token refresh logic and stored tokens in HTTP-only cookies.

2. Debate Filtering and Sorting

Challenge: Optimizing database queries for large datasets.

Solution: Used MongoDBs aggregation framework to efficiently sort and filter data.

3. User Role Management

Challenge: Ensuring secure access to admin-only features.

Solution: Developed middleware to verify user roles before accessing specific routes.

6. Future Scope

Advanced Analytics: Add user engagement analytics for admins.

Real-time Features: Enable live debate responses using WebSockets.

Mobile App: Develop a mobile version of the platform.

7. Conclusion

Debatverse is a comprehensive platform for debate creation and participation. By leveraging the MERN stack, we ensured a scalable and efficient application that can cater to a wide user base. The project provided valuable experience in full-stack web development, particularly in implementing role-based access, handling complex data relationships, and ensuring seamless user interaction.

Submitted by:

Sameer Sethi

Email: sameersethithree@gmail.com

GitHub: Gladiatorxf1

LinkedIn: Sameer Sethi