AJITH V

Mern Stack Developer

8838040971

ajithkumararun1111@gmail.com

No 81,New Street,Valady,Lalgudi,
Tiruchirappalli-621218

CAREER OBJECTIVE

Dynamic and motivated MERN stack fresher eager to leverage academic foundation and practical experiences to contribute effectively to a forward-thinking tech team. seeking an entry-level position where I can apply my skills in web development, collaborate with peers, and continuously learn and grow in a supportive environment. passionate about creating seamless user experiences and solving complex problems through innovative solutions in the ever-evolving world of technology.

SUMMARY

• Recent graduate with a passion for web development, particularly in the MERN stack (MongoDB, Express.js, React.js, Node.js).

• Skilled in front-end and back-end technologies, eager to contribute to dynamic projects and
learn from experienced professionals

EDUCATION

• M.Sc. -Computer Science (2021-2023)
• National College Tiruchirappalli
• Percentage – 70%
• B.Sc.-Computer Science (2018-2021)
• Bishop Heber College Tiruchirappalli
• Percentage – 60.2%

• Higher secondary (2018)
• Government Higher Secondary School
Tiruchirappalli
• Percentage – 75%

TECHNICAL SKILLS

• Proficient in HTML, CSS, JavaScript
• Experience with React.js for building interactive user interfaces
• Familiarity with Node.js and Express.js for server-side development
• Basic understanding of MongoDB and database management and MySQL
• Strong problem-solving abilities and a willingness to learn new technologies
• Other Technologies worked: Git, GitHub, CDN, Bootstrap, Material-Ui, Netlify, Render

PROJECTS

Project 1:

• **Project Name**: Docter Management System
• **Role**: Full Stack Developer
• **Description**: A Doctor Management System built using the MERN stack (MongoDB, Express.js, React.js, Node.js) would be a web application designed to streamline various aspects of managing doctors, patients, appointments, and medical records.

Project details:

• **MongoDB**: This NoSQL database would store all the data related to doctors, patients, appointments, and medical records. Collections within MongoDB could include 'doctors', 'patients', 'appointments', and 'medical records'.
• **Express.js**: This would serve as the backend framework for handling HTTP requests, routing, and middleware. Express.js would manage the communication between the client side React.js application and the MongoDB database.

• **React.js**: The front end of the application would be built using React.js, providing a dynamic and responsive user interface. React components would handle the presentation layer, allowing users to interact with the system to schedule appointments, view medical records, and manage patient information.
• **Node.js**: This would serve as the runtime environment for running the backend server. Node.js would handle incoming requests from the frontend, process data, interact with the MongoDB database, and send responses back to the client.

• **Authentication and Authorization**: The system would include user authentication and authorization mechanisms to ensure secure access to sensitive data. This could involve implementing features like user registration, login, and role-based access control.
• **Appointment Scheduling**: Users would be able to schedule appointments with doctors through the system. The application would provide a calendar interface where users can view available time slots, select a convenient time, and book appointments.
• **Medical Records Management**: The system would allow doctors to create and maintain electronic medical records for each patient. This could include storing information such as medical history, diagnoses, prescriptions, and treatment plans.
• **Search and Filter Functionality**: Users would have the ability to search for doctors by specialty, location, availability, etc. Patients could also search for their medical records using filters like date, type of visit, or diagnosis.
• **Notifications**: The system could send notifications to users for appointment reminders, updates on medical records, or any other relevant information.
• **Admin Dashboard**: An admin dashboard would be provided to manage system settings, user accounts, and oversee the overall functioning of the application

Project 2:

• **Project Name**: Bike Service Application
• **Role**: Full Stack Developer
• **Description**: Bike Service Application built using the MERN stack (MongoDB, Express.js, React.js, Node.js) would be a web application designed to streamline various aspects of managing admin, customers, mechanics, appointments, and Notification between mechanic and customer

Project details:

• **MongoDB**: The application would utilize MongoDB as the database to store various data related to users, bikes, service centers, appointments, and service records. Collections within MongoDB could include 'users', 'bikes', 'service centers', 'appointments', and 'service records'.
• **Express.js**: Serving as the backend framework, Express.js would handle HTTP requests, routing, and middleware. It would manage communication between the client-side React.js application and the MongoDB database, as well as provide APIs for various functionalities.
• **React.js**: The front end of the application would be built using React.js to offer users a dynamic and intuitive interface. React components would handle the presentation layer, allowing users to interact with the application to book appointments, view service history, and manage bike information.
• **Node.js**: As the runtime environment for the backend server, Node.js would handle incoming requests from the frontend, process data, interact with the MongoDB database, and send responses back to the client.
• **Authentication and Authorization:** The application would include user authentication and authorization mechanisms to ensure secure access to user accounts and sensitive data. This could involve features such as user registration, login, password reset, and role-based access control.
• **Bike Registration:** Users would be able to register their bikes within the system by providing details such as make, model, year, and registration number. This information will be stored in the database for future reference.

• **Appointment Booking:** Users would be able to schedule service appointments with their preferred service centers through the application. They could select the date and time for the appointment, choose the type of service required, and provide any additional notes or requests.
• **Notifications:** Users could receive notifications for appointment reminders, updates on service status, or any other relevant information via email or in-app notifications.
• **Admin Dashboard:** An admin dashboard would be provided to manage system settings, service centers, user accounts, and oversee the overall functioning of the application.

PERSONAL SKILLS

• Pre-Planning things well in advance
• Ability to work well in a team environment and individual environment.
• Good interpersonal skills with positive attitude