FRONTEND

Step 1: Understand the Requirements

Gather specific requirements for the university's internal transportation system, such as real-time vehicle tracking, route information, scheduling, booking management, and user authentication.

step 2: Design the User Interface

Create wireframes and design mockups that reflect the university's branding and create a user-friendly interface for students, staff, and other users.

Step 3: Choose the Technology Stack

Utilize HTML for content structure and CSS for visual styling.

Implement JavaScript for interactive features and functionality.

Employ React.js for building a modular and reusable frontend application.

Use React Router for seamless navigation.

Incorporate UI component libraries like Material-UI or Bootstrap for consistent and visually appealing styles.

Step 4: Integrate with APIs

Connect to backend server APIs to fetch data and perform actions related to the internal transportation system.

Integrate APIs for retrieving vehicle data, managing routes and schedules, handling bookings, and providing real-time updates.

Step 5: Manage Packages

Use a package manager like npm or Yarn to handle dependencies and install required libraries and tools.

Step 6: Manage State

Adopt a state management solution like Redux or React Context API to manage and share application-level state across components.

Ensure data consistency and smooth communication between different parts of the application.

Step 7: Test the Application

Implement unit tests and integration tests using frameworks like Jest or React Testing Library.

Verify the reliability and quality of the frontend codebase by identifying and resolving issues and bugs during development.

Step 8: Deploy the Website

Choose a suitable hosting platform or server infrastructure that aligns with the university's requirements.

Consider platforms like Netlify, Vercel, or hosting on university servers.

Step 9: Implement Continuous Integration and Deployment (CI/CD)

Set up CI/CD pipelines using tools such as GitHub Actions or Jenkins.

Automate the build, testing, and deployment processes to ensure smooth integration of changes.

Step 10: Version Control

Utilize a version control system like Git to track changes, enable collaboration, and manage different versions of the frontend codebase.

Step 11: Document the Project

Create comprehensive documentation that includes code comments, architectural decisions, setup instructions, and user guides.

Facilitate future maintenance, updates, and collaboration among developers and stakeholders.