1. Introduction & Objectives

The aim of this project was to create a fully functional car rental web application, covering front end design, back end API development, and database management. The overarching objectives included an intuitive user interface for browsing and reserving rental vehicles, an API for managing data (cars, users, protection plans), and a secure database to store user credentials, car information, and reservations. The concept revolved around streamlining the car rental process so that prospective clients could select locations and dates, compare vehicles with different transmission/fuel types, choose protection plans, and finalize reservations in one cohesive environment.

The methodology combined HTML, CSS, and JavaScript, external libraries like jQuery and Daterangepicker for interactive features on the front end, Java, SpringBoot on the back end, and MariaDB as the database. By separating concerns between the front end (user interface) and the back end (REST API), the design allowed for flexibility and scalability. This multi-tier approach ensured that the user interface could consume the same back-end services that any future mobile or desktop clients might also utilize.

2. Concept to Implementation

- Main Page: The project begins with a central landing page (“Book and Drive, Feel Alive!”) where users can see an overview of the service and a prominent button to log in or sign up.
- Login/Sign-In Page: Once users click “Login/Sign In,” they are taken to a page that prompts them for credentials. In the background, a request is made to the Java-based API to verify the username and password against records stored in the MariaDB database. If the credentials match, the session is established and the user gains access to the subsequent pages. The username and password of the signed-in user are saved in the database
- Car Options Page: Successful login leads to the main reservation flow. Here, users can select pickup and return dates, choose a pickup location, and filter cars based on brand or transmission type. Each car listing shows an image, brand, transmission, fuel type, and daily rate. Users can add protection plans for insurance coverage or other add-ons. After selecting a car and any extras, the system calculates a final price based on the dates, car type, and chosen protection options.

- Reservation Completion & “My Reservations”: Upon completing a reservation, the confirmed data (vehicle, pickup date, etc.) is stored in the database. The system retrieves these records whenever the user clicks “My Reservations,” allowing them to see all past or upcoming rentals associated with their account.

3. Results & Goal Assessment

The final web application meets the initial goal of providing a functional car rental workflow: from browsing cars to reserving one, complete with date/time and location selection. The interface is consistent, responsive web page design, the back end effectively handles data management, and the database preserves all relevant records. The solution is sufficiently modular and extendable—further enhancements, such as adding payment integration or push notifications, could be made without substantial rework to the architecture.

Although the system achieves the primary goals, several refinements could be considered. For example, more robust error handling could be integrated for booking conflicts, or a dynamic pricing engine could be introduced for peak times.

4. Reflection & Lessons Learned

Ensuring consistent communication between the frontend and the Spring Boot REST APIs was a key challenge. Learning effective API design and the intricacies of data exchange between layers required careful endpoint structuring, request/response formats, and error handling strategies. Managing the database schema in MariaDB to handle user authentication and reservation details also demanded rigorous planning to maintain data integrity. Properly formatted JSON requests and responses and accurate endpoint definitions were crucial. Overall, this project emphasized the importance of clear service-layer definitions, strong testing practices, and robust data handling to create a smooth user experience.

Conclusion

This project demonstrates a full-cycle approach to building a car rental platform: from concept to detailed implementation, culminating in a workable system that aligns closely with the stated objectives. Overall, the project has confirmed the viability of a multi-tier architecture and highlighted the importance of meticulous planning, testing, and iterative improvement in delivering a robust web application.