1. Project Overview

1.1 Project Objectives

- The goal for this project is to build a functional web application using a custom technology stack consisting of frontend and backend technologies. The project will be built following the Agile methodology accompanied by Github to manage all of the processes involved.

1.2 Scope

- The web application will bring to life a car rental software platform. The application will include features such as creating an account, browsing cars, renting a car, etc. The interface will handle three types of users: Customers, Customer Service Representatives, System administrators.

1.3 Target Audience

- The web application will have primarily adults as its target audience. Of course, users willing to rent a car will need a valid driver's license and identification.

2. Project Approach

2.1 Development Methodology

- The project will be developed following an adapted Agile methodology. This is best suited for our project considering the 10 week timeline of the project. Following Agile development, will allow the team to work on detailed tasks every week. This will reduce the time required to surpass the obstacles encountered along the way.

2.2 Project Timeline

- The project will span 10 weeks. There are 4 major milestones to be met namely Sprint 1, Sprint 2, Sprint 3, and Sprint 4. After Sprint 1, each remaining sprint will last 3-4 weeks.

2.3 Collaboration and Communication

- The team will take advantage of GitHub and Discord in order to optimize collaboration and communication. Github will be home to all the source code and its infrastructure, as well as relevant documents. The Discord server will mainly serve as a communication medium for weekly meetings, and specific channels set up to discuss ideas, issues, suggestions, and more.

3. Technology Stack

3.1 Backend Frameworks

3.1.1 Flask

- **Description**: Flask is a backend framework written in Python, known for its simplicity and ease of use.
- **Rationale**: Flask is a suitable framework for this project since it can be easily integrated into our project. The simplicity of Flask will be practical for a team relatively new to web development.
- **Strengths**: easy to integrate, easily customizable, integrated with Python
- **Weaknesses**: limited for advanced features, scalability challenges for large projects.
- **Use cases:** web applications

3.1.2 Node.js

- **Description:** Node.js enables server-side JavaScript development, leveraging Chrome’s V8 engine.
- **Rationale:** Node.js ensures consistency in codebase and developer skillset with JavaScript, making it a good choice for front-end. It’s asynchronous I/O model ensures efficient handling of concurrent requests. Also, Node.js has a robust community, ensuring ongoing support and updates.
- **Strengths:** High Performance, Scalability, Asynchronous I/O Model, Versatility, Large Ecosystem
- **Weaknesses:** CPU-Bound Tasks, Callback Hell, Maturity
- **Use Cases:** Real-time Applications, APIs and Microservices, Streaming Services, Single-page Applications, Internet of Things Applications

3.1.3 Express.js

- **Description:** Express.js is a backend web application framework for Node.js. Used for building server-side apps with features to handle HTTP requests.
- **Rationale:** Express.js has a large community and documentation. It is easily scalable for large projects. Seamless integration with Node.js
- **Strengths:** large support, easily scalable, ease of integration with Node.js
- **Weaknesses:** steep learning curve
- **Use cases:** building APIs, web applications

3.2 Frontend Frameworks

3.2.1 React

- **Description:** React is a JavaScript library for building user interfaces. It allows developers to create reusable UI components that manage their own state, making it easier to build complex UIs.
- **Rationale:** React's component-based architecture simplifies UI development and promotes code reusability. It's virtual DOM ensures efficient rendering, enabling fast updates and a smooth user experience.
- **Strengths:** Component-Based Architecture, Virtual DOM, Declarative Syntax, Large Ecosystem, Community Support
- **Weaknesses:** Complexity, Learning curve, Toolchain Configuration, SEO Challenges, Rapid Changes
- **Use Cases:** Single-Page Applications, Interactive Web Applications, Mobile Applications, E-commerce Platforms, Content Management Systems

3.2.2 Angular

- **Description:** Angular is a frontend framework based on TypeScript. It is designed for client-side applications with a large catalog of features.
- **Rationale:** Angular is suited for creation of web applications with interactive user interfaces, ensures a responsive experience for the user, and properly handles behavior on different browsers.
- **Strengths:** large feature set, responsive, compatible across browsers
- **Weaknesses:** Relatively steep learning curve
- **Use cases:** Single page applications, enterprise applications

3.2.3 Vue.js

- **Description:** Vue.js is a progressive JavaScript framework used for building user interfaces and single-page applications. Vue.js allows developers to create interactive and reactive UI Components.
- **Rationale:** With its component-based architecture, Vue.js simplifies UI development by enabling the creation of reusable components.. Vue.js also ensures consistent performance across different browsers and platforms.
- **Strengths:** Simplicity, Reactivity, Component-Based Architecture, Flexibility, Performance, Growing Ecosystem.
- **Weaknesses:** Limited Corporate Backing, Smaller Community, Tooling Ecosystem, Learning Curve, Enterprise Adoption

4. Integration and Interoperability

4.1 Backend-Frontend Integration

To integrate React and Flask simple buttons and actions must be designed for the user interface. React will ensure an interactive UI with appealing buttons and design. Flask will be responsible for reading the users' actions and interacting with the car data stored in the database to execute CRUD operations. Flask routes will handle requests and connect to the mySQL database. Postman will help with testing the functionality of the web application and error handling.

4.2 Third-Party Services

- Postman:
  - **Description:** Postman is a popular API development tool that simplifies the process of creating, testing, and documenting APIs. It offers a user-friendly interface for sending HTTP requests, allowing developers to interact with APIs effortlessly.
  - **Strengths:** User-Friendly Interface, Efficiency, Testing Capabilities, Collaboration Tools, Documentation Generation, Monitoring and Insights

- MySQL:
    - **Description:** MySQL is an open-source relational database management system that enables users to store, organize, and manage structured data.
    - **Strengths:** Performance, Scalability, Reliability, Ease of Use, Flexibility, Cost-Effective.

5. Security Considerations

The web application will include a registration and account login feature for users. This feature must securely store account information and implement hashing for storage of passwords. In addition, the user interface must include thorough input validation to make sure the program behaves as intended when encountering unexpected user inputs. Regular bug updates will also help in maximizing the security of the web app.

6. Conclusion

In sum, the car rental web application will be developed following Agile methodology using frontend and backend frameworks. HTML, CSS, and React.js will be used for the frontend interface of the software. The backend technologies used will be Flask, as well as mySQL for the database. The choices are based on ease of use and ease of integration, as well as the opportunity to learn a new framework using a language known to all members.