**SOEN 341 - SS**

**Minhtu Chau - 40232883**

**Hawraa Al-Adilee - 40246450**

**Carmen Derderian - 40244084**

**Mohamad Edelby - 40251628**

**Amira Toslim - 40207999**

**Software Process**

**Department of Computer Science and Software Engineering (CSE)**

**Sprint 1**

**Professor: Rodrigo Morales Alvarado**

**"I certify that this submission is my original work and meets the Faculty's Expectations of Originality"**

**February 12, 2024**

SPRINT 1

# 1. Project Overview:

## 1.1. Project Objectives

The main objective of this project is to develop a car rental Web application that is user-friendly, efficient, and reliable. This web application has an objective to simplify the process of renting vehicles for various types of users, such as the customer looking to rent a car, the customer service representatives managing rental processes, and system administrators overseeing the platform's operations.

These are some key objectives:

- User experience: Creating an interface that will allow the customers to easily browse, reserve, and manage their rentals.
- Simplify operations: Automate the car rental process, to make it easier and efficient for customer service representatives and system administrators (CRUD for reservations).
- Up to date vehicle management: Implement vehicle management capabilities to allow updates to the availability of the rental cars (CRUD for vehicles).
- Strong user management: Implement secure user management system that supports different roles enabling different permissions (CRUD for user accounts).
- Scalability and Reliability: Ensure the amount of growing user and inventory the web application could handle to ensure high performance and reliability.
- Reviews: Integrate possibility of feedback and rating for the rental cars by customers.
- Check-out process: Implement efficient checking-out process with assured inspection of vehicle, review of rental agreement and payment which is reserved as a function for the customer service representative.

## 1.2. Scope

The scope of Sprint 1 is about establishing the project's foundational elements. This includes setting up the GitHub repository and detailing the project's structure within the README file. The README will offer an overview of the project, outlining the core features, the project objectives, and the proposed implementation strategy. Key tasks involve deciding on the framework and programming languages that will be used for development. Organizing tasks and assigning roles to each team member. This organization extends to maintaining a log of activities, tracking the time spent on each task, and formulating a plan for sprint 2.
The core functionalities considered include:

- Browse vehicles: Customers can view available vehicles that can be filtered by type location and availability.
- Reservation: The customers must be able to make, modify and cancel reservations.
- User account management: The users can register, log in, manage their profiles, and view their rental car history.
- Different user permissions: The administrators and CSR must be able to manage vehicle listing and user accounts.

## 1.3. Target Audience

The web application targets three different types of users: customers, CSR and administrators.

- Customers: These users are seeking to rent a vehicle. They need an efficient, easy to use platform to browse through vehicles, reserve and manage their rental history.
- Customer Service Representatives: These users must assist the customers with reservations, check-in, and check-out. They need different permissions to manage reservations, interact with customers, and the availability of vehicles.
- System Administrators: These users maintain the web application. They have the most rights. They manage the vehicle listing, the user accounts, and the settings.

## 2. Project Approach:

### 2.1. Development Methodology

This project will be using the Agile development methodology [1]. This aligns with the dynamic nature of this project and the need for rapid prototyping. Agile is good because of its adaptability, iterative methodology, and focus on customer feedback and collaboration. These factors are essential for creating a user-centered rental vehicle application. With this strategy, the team can respond to changes fast, deliver prototypes through sprints presented to the SCRUM master which will allow us to continuously improve the web application based on the feedback.

### 2.2. Project Timeline

This project is structured into 4 sprints over a 10-week period. Each sprint is about 2 and a half weeks. This is an overview of the timeline:

- Weeks 1-2: Sprint 1 focuses on project setup, defining core functionalities, and initial development tasks such as setting up the development environment and discussing basic application structure.
- Weeks 3-5: Sprint 2 is dedicated to developing key features identified in Sprint 1, with a focus on implementing user stories related to vehicle browsing and reservation functionalities.
- Weeks 6-8: Sprint 3 will build upon the work from the previous sprints, adding more complex features such as user account management and advanced vehicle filtering options.
- Weeks 9-10: Sprint 4 will finalize the prototype with a focus on refining features, conducting user testing, and addressing any remaining issues or feedback.

### 2.3. Collaboration and Communication

The team uses different tools and practices to collaborate and communicate.

- GitHub: It will be used as the central collaboration method. On this platform, the team will manage the code, track user issues, and document. The project repository will be held here, manage pull requests and track progress with a Minutes file which includes detailed log of each team member and wiki pages that will serve as planning for next Sprints.
- Discord: It will be used as the main communication platform. Where we communicate daily about quick updates and informal discussions.
- In-Person meetings: These will be scheduled during the team's mutual school breaks. This will facilitate discussion, brainstorming and collaborative work.
- Meeting Minutes: There will be a minute file on GitHub, that will document each meeting and log of each member's activity.

## 3. Technology Stack:

### 3.1. Backend Frameworks

#### 3.1.1. Node.js

Node.js is a runtime environment that executes JavaScript code outside the browser. It utilizes an event-driven, non-blocking I/O model, making it well-suited for building scalable and real-time applications. [2]

Rationale:

- Justification: Node.js shines in a feature-rich car rental application due to its asynchronous and non-blocking I/O, ensuring real-time responsiveness for features like live vehicle tracking and instant reservation updates. The fast V8 engine enhances the

user experience, enabling quick browsing through a diverse fleet and prompt booking confirmations. With a unified language, Node.js simplifies development for dynamic interfaces, facilitating interactive vehicle filtering. Real-time functionalities, including live notifications for reservation status and pricing updates, benefit from Node.js' efficiency. Its microservices support aids in developing modular components for managing user profiles, reservation histories, and seamless integration with external services, ensuring a streamlined and scalable car rental application. Node.js' efficient API handling further enhances geolocation services, providing accurate tracking for location-based features. [3]

- Community Support: Node.js has a large and active community, fostering continuous development and providing a rich ecosystem of modules and libraries. This support enhances the development process for a car rental application.
- Scalability: Node.js is known for its scalability, particularly in handling concurrent connections. This is crucial for a car rental platform that may experience varying levels of user activity.
- Ease of Integration: Node.js excels in integrating with various services and APIs due to its asynchronous nature. This makes it seamless to connect with payment gateways, geolocation services, and other third-party components in the car rental application.

Qualitative Assessment: [4]

Strengths

- Community Support: Node.js has a large and active community, fostering continuous development and providing a rich ecosystem of modules and libraries. This support enhances the development process for a car rental application.
- Scalability: Node.js is known for its scalability, particularly in handling concurrent connections. This is crucial for a car rental platform that may experience varying levels of user activity.
- Ease of Integration: Node.js excels in integrating with various services and APIs due to its asynchronous nature. This makes it seamless to connect with payment gateways, geolocation services, and other third-party components in the car rental application.

Weaknesses

- Single-threaded Nature: While Node.js is highly scalable, its single-threaded event loop can become a bottleneck for CPU-bound tasks, potentially impacting performance in scenarios with extensive computation requirements.
- Less Suitable for CPU-Intensive Tasks: Node.js may not be the best choice for applications requiring intensive CPU processing due to its single-threaded nature.
- Callback Complexity: The asynchronous nature of Node.js can lead to callback complexity, where deeply nested callbacks may make code harder to read and maintain. However, this challenge can be addressed with modern JavaScript features and design patterns, ensuring a more straightforward and maintainable code structure.

Use Cases

- Dynamic Pricing: Utilize Node.js for real-time adjustment of rental rates based on demand, ensuring flexible and competitive pricing strategies.
- Live Vehicle Tracking:Employ Node.js and WebSockets for real-time tracking of vehicle locations, offering users immediate updates on vehicle positions.
- Instant Notifications:Implement Node.js with WebSockets for instant reservation updates and notifications, ensuring a seamless and responsive user experience.
- Secure Authentication: Leverage Node.js and Express.js for secure APIs, using asynchronous programming to efficiently process user authentication requests.
- Scalable Microservices:Adopt Node.js and Express.js for a scalable microservices architecture, allowing independent development, deployment, and scaling of services.

### 3.1.2 Django

Description: Django is an open-source web framework written in python that provides tools and conventions for building web applications. [5]

Rationale:

- Justification: Selecting Django for a car rental application is justified by its capacity for swift development, robust security features, and efficient database management through its Object-Relational Mapping (ORM). The built-in authentication system and user-friendly admin interface enhance security and streamline fleet management. Django's modularity and versatility support the development of various components, contributing to maintainability. The framework's scalability and optimization features ensure optimal performance, making it a solid choice for building a secure, feature-rich, and efficient car rental platform. [6]
- Performance: Django's ORM allows query optimization for improved database performance. It also provides built-in caching that enhances response times by storing frequently accessed data. Lastly, Django supports horizontal scaling and load balancing for handling increased traffic.
- Security: Django has built-in Security Features**,** in fact, it addresses SQL injection, XSS, CSRF, and clickjacking. It possesses robust authentication, permissions and access control features. Additional security is provided through middleware for HTTP header configuration.
- Maintenance: Django makes maintenance easier through many measures. It has à built-in admin interface that streamlines routine data and database management. As well as modular middleware architecture that eases updates and maintenance. Furthemore, Extensive resources and community support facilitate ongoing maintenance. And Lastly, Clear versioning and release processes ensure compatibility and enable confident updates.

Qualitative Assessment:

Strengths: [7]

- Built-in Admin Interface: Django includes a powerful admin interface, automatically generated based on your application's models, making it easy to manage and visualize data.
- ORM (Object-Relational Mapping): Django's ORM simplifies database interactions, allowing developers to work with databases using Python objects, enhancing readability and maintainability.
- Security: Django incorporates built-in security features such as protection against common web security vulnerabilities, making it a secure choice for web application development.
- Scalability: Django's design principles and the ability to break applications into reusable components contribute to its scalability, making it suitable for both small projects and large-scale applications.

Weaknesses

- Steep Learning Curve: For beginners, Django's extensive feature set might result in a steeper learning curve compared to lighter frameworks. However, this is often offset by the productivity gains once mastered.
- Opinionated Structure: Django enforces a specific project structure and follows certain conventions, which may be limiting for developers who prefer more flexibility and minimal structure.
- Overhead for Small Projects: The full-featured nature of Django might be overkill for small projects, as it includes components that might not be necessary for simpler applications.

Use Cases

- User Registration and Authentication:Django has à built-in authentication system for user registration and login.

- Car Listings and Details: Django admin for managing car listings, and QuerySets for filtering and searching.
- Booking and Reservation System:Models for bookings, views using Django forms for reservation handling, and Django signals for preventing double-bookings.
- User Profiles and History:User profiles using Django's built-in User model, views for managing user data, and Django signals for tracking user activities.
- Payment Gateway Integration:Django views for handling payment transactions, integration with third-party payment gateway APIs.
- Review and Rating System:Models for storing user reviews, views for displaying and submitting reviews, and Django templates for rendering review components.
- Geolocation and Navigation Integration:Integration with geolocation services using Django's views and templates, and possibly Django REST framework for API integration.

### 3.1.3 Spring

Description: Spring is a comprehensive, open-source framework for building enterprise-level Java-based applications. It provides a modular and flexible infrastructure for developing and deploying a wide range of applications, from small-scale projects to large, complex enterprise systems. Spring follows the Inversion of Control (IoC) and Dependency Injection (DI) principles and offers various modules for diverse functionalities. [8]

Rationale:

- Justification: Opting for Spring in a car rental application is justified by its robust enterprise integration, support for microservices, and effective Dependency Injection (DI). In the context of car rentals, where seamless integration with services like payment gateways is crucial, Spring's enterprise features prove invaluable. Its DI principle enhances maintainability, and compatibility with microservices aligns with the need for scalability, offering distinct advantages in building a comprehensive, adaptable, and feature-rich car rental application. [9]
- Community Support: Spring boasts a vibrant and extensive community that contributes to its ongoing development and support. The active community provides a wealth of

resources, forums, and third-party libraries, ensuring a robust ecosystem and reliable assistance for developers.

- Learning Curve: While Spring has a learning curve, especially for beginners, its comprehensive documentation, tutorials, and educational resources make it accessible. The investment in learning Spring is justified by its versatility and extensive features.
- Extensibility: Spring's modular architecture and support for dependency injection contribute to its extensibility. Developers can easily integrate additional modules and libraries to tailor the framework to specific requirements.

Qualitative Assessment:

Strengths: [10]

- Modular Architecture: Spring's modular design promotes code organization and flexibility, allowing developers to build and scale components independently.
- Dependency Injection: The Inversion of Control (IoC) and Dependency Injection (DI) principles enhance maintainability and testability by reducing tight coupling between components.
- AOP (Aspect-Oriented Programming): Spring's support for AOP allows developers to modularize cross-cutting concerns, improving code maintainability.
- Enterprise Integration: Spring offers robust solutions for enterprise integration, making it suitable for building complex systems like a car rental application.

Weaknesses

- XML Configuration Overhead: While modern versions of Spring offer annotation-based configuration, some legacy projects may still involve XML configuration, which can be verbose.
- Convention Over Configuration: Developers who prefer more convention over configuration may find Spring's extensive configurability overwhelming.

Use cases

- User Management and Security: Leverage Spring Security for robust user authentication and authorization, ensuring secure registration processes and role-based access control.
- Efficient Car Listings:Utilize Spring MVC and Spring Data JPA for dynamic and responsive car listings, allowing seamless retrieval and presentation of detailed car information.
- Scalable Booking System: Implement Spring Boot for developing microservices, ensuring scalability and flexibility in handling booking transactions with embedded server capabilities.
- Comprehensive Admin Dashboard: Develop a robust admin dashboard using Spring MVC and Thymeleaf, and leverage Spring Boot for creating microservices to manage fleet data efficiently.
- Interactive Review System:Implement a review system using Spring Data JPA for storing and retrieving user reviews, and utilize Spring MVC for displaying dynamic and interactive review components.
- Geolocation Integration and Notifications: Utilize Spring's support for geolocation integration and asynchronous processing for notifications, enhancing the overall user experience and ensuring timely alerts for users and administrators.

## 3.2. Frontend Frameworks

### 3.2.1. React

Description: React is a library used for the front-end aspect of a web development. [11]

Rationale:

- Justification:VIts component-based architecture, which perfectly fits the modular structure of an application like this, justifies React for the car rental website project. Every component of the platform, including user profiles, booking forms, and car listings, may be created as separate, reusable parts. This modularity facilitates a streamlined

development process, allowing for concurrent work on various aspects of the application and simplifying maintenance and updates. React's virtual DOM-based update method guarantees optimal performance and a seamless user experience, which is essential for dynamic interactions such as vehicle selection and reservation administration. React's robust ecosystem and robust community support also provide a plethora of libraries and tools that improve functionality and expedite development. With these benefits, React is clearly the best option for building a responsive, maintainable, and user-friendly car rental platform. It has become one of the most used library front-end frameworks in modern web development. In fact, there are many popular websites that use React, such as: Facebook, Instagram, Netflix, Reddit, Uber, Discord[11].

- Capabilities: React is really good for routing the website's components. In fact, it is very efficient for building dynamic user interfaces since reusable UI components will make it easier for developers to create complex applications.
- Responsiveness: React uses a virtual DOM(document virtual model) that allows it to render components in an efficient way. In fact, the components are sorted in a tree. Therefore, it is easier to update the necessary parts of an interface and minimize the render time.
- cross-browser compatibility: like mentioned previously, since the library uses components that are reusable, and a virtual DOM, the user interface can consistently work across different web browsers.

Qualitative Assessment: [12]

- Strengths: The Virtual DOM setup of the framework allows this component-based architecture to be more effective when it comes to write or read a code.
- Weaknesses: It is easy to read and write the codes in React, but it has to compensate with the fact that it needs a lot of updates. However, React has rapid release cycles.

Use Cases:

- Vehicle Catalog Display: React can be used to create a dynamic and responsive vehicle catalog, where users can filter and sort through available vehicles based on criteria like

car type, price, or availability. React's state management can handle user inputs and queries efficiently, updating the UI in real-time.

- Reservation System:The reservation process involves multiple steps, from selecting dates and vehicles to adding additional services and confirming the booking. React's component-based approach allows each step to be encapsulated in its own component, making the flow easy to manage and navigate.
- User Authentication and Profile Management: React can facilitate the development of user authentication flows, including sign-up, login, and password recovery, as well as profile management interfaces where users can view and edit their information and booking history.
- Interactive Location Selection: Integrating maps for selecting rental locations or drop-off points can be made interactive and responsive with React, enhancing the user's experience in choosing precise locations.
- Real-time Availability Checks: React can efficiently handle real-time updates to vehicle availability, ensuring that users have up-to-date information on which vehicles are available for their selected dates and locations.
- Ratings and Reviews: A feature allowing customers to rate and review their rental experiences can be built with React, providing an interactive platform for user feedback that dynamically updates as new reviews are added.
- Customer Support Chat: Incorporating a real-time chat feature for customer support can be achieved with React, offering users immediate assistance directly within the platform.

### 3.2.2 Angular

Description: Angular is an open-source front-end web application framework by google. It is written in Typescript and makes it easier to create feature-rich, dynamic single-page web applications (SPAs). [13]

Rationale:

- Justification: Selecting Angular as the frontend framework provides a thorough, organized method for creating dynamic single-page apps; this is particularly useful for intricate or enterprise-level projects. Because of its object-oriented capabilities and robust

typing, TypeScript is used to improve code dependability and developer experience. Angular's component-based architecture significantly enhances its suitability for large-scale applications, such as a car rental platform, by promoting reusability and modularity. In such a project, different functionalities like browsing vehicles, managing reservations, and user accounts can be encapsulated within distinct components. This modularity allows for isolated development and testing of features, which is particularly beneficial in a complex application where different functionalities might share similar UI elements or data structures.The Angular CLI simplifies development activities. The choice of Angular for projects needing reliable, scalable, and high-performance online apps is further supported by its advanced capabilities, such as server-side rendering and support for Progressive online Apps.

- User Interface Features: Create complex user interfaces using tools and features offered by Angular. It has a component-based architecture. Therefore, large-scale applications are easier to manage and maintain since reusable, modular UI elements can be created.
- Reactiveness: Two-way data binding and dependency injection. This helps to create real-time, responsive applications. The overall user experience is improved by the framework's ability to immediately refresh the view when data is changed such as the availability of cars in this case.
- Cross-Browser Compatibility: It's able to function across different browsers while remaining consistent. It builds user interface (UI) declaratively, which simplifies the management of browser-specific nuances.

Qualitative Assessment:

Strengths: [14]

- Modularity: Angular's modular structure encourages code reusability and maintainability, promoting a clean and organized codebase.
- Two-way Data Binding: This feature simplifies the synchronization between the model and the view, reducing the need for boilerplate code and enhancing development speed.
- Dependency Injection: Angular's dependency injection system facilitates the management of components and services, making it easier to test and maintain code.

Weaknesses:

- Learning Curve: Due to its comprehensive feature set and complex concepts, Angular may have a steeper learning curve for developers who are new to the framework.
- Performance Overhead: The framework's extensive features can sometimes lead to a higher initial load time, which may impact the performance of applications, especially on slower networks.

Use Cases:

- Single-Page Application (SPA) Architecture:Angular's proficiency in building SPAs allows for fluid navigation and dynamic content loading without page refreshes, essential for a seamless user journey through vehicle selection, reservation, and account management on the car rental platform.
- Reactive Forms for Complex Booking Processes:The complex forms involved in booking a vehicle, including date selection, options for add-ons, and payment processing, benefit from Angular's reactive forms. This feature provides robust form validation and real-time feedback, enhancing the user's booking experience.
- Modular Development with Lazy Loading:Angular's modularity and support for lazy loading help in organizing the car rental platform into distinct features like vehicle browsing, user profiles, and reservation management. This not only improves maintainability but also optimizes load times by fetching resources as needed.
- Advanced Routing and Navigation:Angular's router facilitates sophisticated routing mechanisms, enabling features like guarded routes for authentication, nested routes for detailed vehicle information pages, and route resolvers for pre-fetching data, ensuring smooth transitions and consistent user experiences.
- Internationalization (i18n) for Global Users:The car rental platform can easily cater to a global audience by leveraging Angular's built-in internationalization capabilities, allowing for the translation of text and the adaptation of date formats and currencies to local preferences

### 3.2.3 SwiftUI

Description:

SwiftUI is known to be the main user interface toolkit used by all Apple platforms. It is a front-end framework that presents visually appealing interfaces that are interactive. [15]

Rationale:

- Justification: Choosing SwiftUI for a car rental application brings several distinct advantages. SwiftUI's modern design paradigm ensures a visually appealing user interface, enhancing the overall user experience. Its deep integration with the Apple ecosystem allows for a unified development experience across iOS, macOS, watchOS, and tvOS, ensuring consistency and optimization for various Apple devices. The synergy with Swift programming language enhances developer productivity, contributing to a robust and maintainable codebase. SwiftUI inherently incorporates accessibility features, promoting inclusivity within the app. Additionally, SwiftUI's future-proof development approach and streamlined prototyping capabilities enable efficient adaptation to upcoming features and iterative design processes. Overall, SwiftUI offers a forward-looking, user-friendly, and efficient solution for developing a sophisticated and visually engaging car rental application.: For a car rental web application, the SwiftUI would be beneficial if the design for Apple devices. In fact, it will also enhance the user-experience for clients that already own Apple devices, since SwiftUI has a consistent design across all platforms.
- Ease of Integration: SwiftUI is chosen for its seamless integration with Swift, Apple's programming language, fostering efficient communication between application logic and the user interface. This native integration accelerates development and ensures a cohesive architecture for the car rental app.
- Component Libraries: SwiftUI provides a rich set of native components and controls, reducing the need for third-party libraries. This built-in toolkit contributes to a consistent and visually appealing UI design for the car rental platform.
- Developer Experience: SwiftUI prioritizes developer experience by offering declarative syntax, real-time previews, and interactive development tools. This enhances the overall

development experience, making it easier for developers to create and iterate on features within the car rental application.

Qualitative Assessment:

Strengths:

- Declarative Syntax: SwiftUI's declarative syntax simplifies UI development, allowing developers to focus on describing the desired UI state rather than managing complex imperative code. This promotes readability and accelerates development.
- Cross-Platform Compatibility: SwiftUI is cross-platform, enabling developers to create user interfaces for iOS, macOS, watchOS, and tvOS using a unified codebase. This versatility ensures a consistent user experience across different Apple devices.
- Interactive Previews: SwiftUI offers real-time previews, enabling developers to visualize UI changes instantly during the development process. This feature enhances the iterative development of dynamic and interactive elements in the car rental app.

Weaknesses:

- Limited Deployment to Older iOS Versions: SwiftUI is available from iOS 13 onwards, limiting deployment to devices running older iOS versions. This consideration is crucial when targeting a diverse user base with varying device capabilities.
- Learning Curve: While SwiftUI simplified UI development, there may be a learning curve for developers transitioning from UIKit. However, SwiftUI's intuitive design and comprehensive documentation help mitigate this challenge over time.

Use Cases :

- Dynamic Vehicle Listings: SwiftUI is instrumental in creating dynamic and visually appealing interfaces for browsing and selecting vehicles, utilizing its declarative syntax for efficient data presentation.
- Real-time Updates and Notifications: Leveraging SwiftUI with Combine framework facilitates real-time updates and live notifications, providing users with instant confirmation and alerts for changes in reservation status or vehicle availability.

- User Authentication Views: SwiftUI can be used to design secure and user-friendly authentication views, enhancing the car rental app's login and registration processes with its native integration capabilities.

## 4. Integration and Interoperability

### 4.1. Backend-Frontend integration

- Outline the strategy for integrating the chosen backend and

frontend technologies.

To develop a car rental web application, we will be using React as the library for the front-end framework and Node.js for the back-end framework. In fact, these two main frame-works are very common and compatible with many other projects similar to ours. To begin, we are going to define, using Node.js command lines, the end points to allow the front-end frame-work to work with the back-end. This will allow us to connect the front-end components to all the data in the back-end. In our case, we are using React Vite, a tool that React uses to increase the speed and performance of our web application to satisfy the user-experience. On top of the back-end and front-end frameworks, we will also add a database to allow all the data, for example a user's account, to be stored safely. We are choosing to use firebase because of the size and performance that we are looking for. Then, we will plan out the hierarchy of the web application to have a concise mapping of the routes and an efficient architecture for our codes. The layout of each required page will also be designed in a way that is coherent throughout the whole application. According to all the requirements, we will then add all the APIs that will offer third-party services. For the car rental web application, we will need an API to mark down the location of a user, to handle the transactions and to handle the login information.

### 4.2. Third-party services

- Maps and Geological Services [19]

Geological services are a must for the application to keep search results within the preferred region of the user. Google Maps API is an efficient integration to the application as it is an interactive map and offers directions.

- Payment processing [20]

  To make the car rental process smooth, a payment gateway such as PayPal is a must. Such APIs facilitate the online payment process securely.

- Email and SMS notifications (optional)

  Once reservations are completed, the user must be contacted through e-mail and/or SMS which can be done through services like SendGrid and/or Twilio respectively. Constant updates and reminders would also require fast communication with the client.

- Identity verification [16]

  To verify the identity of the user while creating an account or during rental transactions, Jumio would serve as a third-party service for efficiency and security.

- Fuel services (optional)

  Information about nearby fuel stations would offer a better client experience. This could be done through GasBuddy API.

- Vehicle tracking (optional)

  The rental company must be updated about the condition of rented vehicles. To monitor the location and condition of their product, Geotab helps to analyze data from vehicles.

## 5. Security Considerations

The website application security aspect must be considered both at the front-end and back-end levels. As developers, we must prioritize security to protect both the owner and the users of the website.

Back-end security

- Authentication and authorization

  To use the rental car rental application efficiently, the user is first asked to sign up or log in. To verify users identities and ensure their data and password are successfully stored, secure authentication mechanisms must be implemented. Multi-Factor Authentication

(MFA) adds a layer of security beyond the username and password login method. This implementation would also help to restrict access to specific resources based on user roles and permissions. Opting for this option optimizes the user's experience as it ensures security without impacting the efficiency of the application. [22]

- Data validation and Sanitization

  User's login information must be validated and sanitized to prevent security vulnerabilities, such as SQL injection, and other injection attacks. There exists prepared statements and Object Relational Mapping (ORM) libraries. These prepared pieces of software interact with the database securely and filter user input. Data validation which checks the user's input is about data type, form (e.g.: e-mail) and length. Data sanitization is the process of cleaning and removing potentially harmful characters before processing the content. This process might involve converting special characters into equivalents to avoid injection attacks. At this step, prepared statements come into action to automatically handle data sanitization. Prepared statements are a better option than ORM libraries. This method of data validation is not recommended as it escapes user inputs before inserting it in a query. Also, it is database specific and may not cover all scenarios. [18]

- Secure payment processing

  A car rental application would involve a large amount of money. Using secure payment gateways to handle financial transactions and to protect payment data such as SSL/TSL encryption. [21] This implementation helps protect against man-in-the-middle attacks.

Front-end security:

- Data handling and storage

  Front-end security deals with local storage, client-side storage, or data stored on a user's device. Security at that level is to avoid storing sensitive information like passwords or personal data in local storage. All data should be sanitized and validated before storing it.

- Input validation

  The user's inputs on the client side must be validated to optimize user experience.

- Geological security

As the application may use geolocation services, it must request user permission before accessing this data by clearly communicating why and how this data is required.

## 6. Conclusion

In wrapping up, we've chosen React for the front end to ensure a user-friendly interface that responds smoothly while opting for Node.js on the back end for optimal operational capabilities. Embracing Agile methodologies, our approach allows us to complete the car rental website within the required time by engaging in multiple interactions with clients, gathering feedback, and addressing uncertainties throughout the development journey. Our tech stack prioritizes security and efficiency, incorporating Google Maps API for location services, PayPal for seamless online payments, and Jumio for streamlined identity verification. While we're considering email and SMS notifications based on project needs, potential integrations with GasBuddy and Geotab aim to boost the client experience by providing info on nearby fuel stations and enabling vehicle tracking for the rental company. This thoughtful mix of third-party services ensures a robust, feature-rich application that meets user needs while reinforcing security. We made these choices to make a web application that developers could make efficiently and give an ideal experience for the user.

## 7. References


[1] Rodrigo Morales Alvarado, Lecture slides, SOEN 341, Concordia University

[2] Node.js tutorial, https://www.w3schools.com/nodejs/ (accessed Feb. 10, 2024).

[3] T. Capan, “Why the hell would I use node.js? A case-by-case tutorial: Toptal®,” Toptal Engineering Blog, https://www.toptal.com/javascript/why-the-hell-would-i-use-node-js#:~:text=Proxy-,Node.,data%20from%20multiple%20source%20points. (accessed Feb. 10, 2024).

[4] “5 main node.js advantages and disadvantages,” EPAM Anywhere, https://anywhere.epam.com/en/blog/node-js-pros-and-cons (accessed Feb. 10, 2024).

[5] “Django,” Django Project, https://www.djangoproject.com/ (accessed Feb. 10, 2024).

[6] MozDevNet, “Django Introduction - Learn Web Development: MDN,” MDN Web Docs, https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction (accessed Feb. 10, 2024).

[7] M. Deery et al., “9 pros and cons of the django framework: A coder’s guide,” CareerFoundry, https://careerfoundry.com/en/blog/web-development/django-framework-guide/ (accessed Feb. 10, 2024).

[8] “Spring framework6.1.3,” Spring Framework, https://spring.io/projects/spring-framework/ (accessed Feb. 10, 2024).

[9] T. Contributor, “What is Spring Framework?: Definition from TechTarget,” App Architecture, https://www.techtarget.com/searchapparchitecture/definition/Spring-Framework#:~:text=Spring%20is%20considered%20to%20be,focus%20on%20writing%20business%20logic. (accessed Feb. 10, 2024).

[10] “Java Spring Pros and Cons - Javatpoint,” www.javatpoint.com, https://www.javatpoint.com/java-spring-pros-and-cons (accessed Feb. 10, 2024).

[11] React, https://react.dev/ (accessed Feb. 10, 2024).

[12] “Pros and cons of reactjs - javatpoint,” www.javatpoint.com, https://www.javatpoint.com/pros-and-cons-of-react (accessed Feb. 10, 2024).

[13] Angular, https://angular.io/ (accessed Feb. 10, 2024).

[14] AltexSoft, “Pros and cons of Angular Development Framework,” AltexSoft, https://www.altexsoft.com/blog/the-good-and-the-bad-of-angular-development/ (accessed Feb. 10, 2024).

[15] A. Inc., “SwiftUI overview - xcode - apple developer,” Overview - Xcode - Apple Developer, https://developer.apple.com/xcode/swiftui/ (accessed Feb. 10, 2024).

[16] Jumio, “Jumio/implementation-guides: Description of server apis and callbacks,” GitHub, https://github.com/Jumio/implementation-guides/ (accessed Feb. 11, 2024).

[17] Team, The Geotab. “Introduction.” *Geotab Developers*, geotab.github.io/sdk/software/introduction/ (accessed 11 Feb. 2024).

[18] OWASP. “SQL Injection Prevention · OWASP Cheat Sheet Series.” *Owasp.org*, Owasp, 2021, cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html./(accessed Feb. 12, 2024).

[19] Google maps platform | google for developers, https://developers.google.com/maps (accessed Feb. 11, 2024).

[20] “1. PayPal API Overview - PayPal APIs: Up and Running, 2nd Edition [Book],” *www.oreilly.com*. https://www.oreilly.com/library/view/paypal-apis-up/9781449321666/ch01.html/(accessed Feb. 12, 2024).

[21] K. Radage, “What is SSL in Payments?,” *Credit Card Processing and Merchant Account*, Oct. 15, 2023. https://www.clearlypayments.com/blog/what-is-ssl-in-payments/#:~:text=The%20utilization%20of%20SSL%2FTLS (accessed Feb. 12, 2024).

[22] mimecast, “What Is Multi-Factor Authentication (MFA)?,” *Mimecast*. https://www.mimecast.com/content/what-is-multi-factor-authentication/ (accessed Feb. 12, 2024).