Appendix A. Project Approach and Technology Stack Selection

1. Project Overview

1.1 Project Objectives

This project aims to create an intuitive and user-friendly web-application to facilitate the booking of vehicle rentals for periods of time of a few hours up to a few weeks. The web-application should be an interface between customer users who wish to rent vehicles, and the companies who wish to offer car rental services.

1.2 Scope

The scope of the project is to deliver a functional, user-friendly web-application used to facilitate vehicle rental bookings. This project includes key features such as the ability to reserve a car for a specific period of time, as well as the ability for users to view their current reservations, with options to possibly modify or cancel their booking. Additionally, all available cars will be listed according to the time and dates the user entered for their reservation. Users will be able to create or login to their account, update their information, i.e. email and password, or delete their account. System administrators will be granted access to features such as posting vehicles that are available to rent, as well as remove or modify already existing listings. The admins will also be able to modify or delete a customer user's reservations.

1.3 Target Audience

The target audience for this web-application are individuals aged between 21 and 65. The target audience begins at 21 because by this point, most individuals have a driver's license and are mature enough to operate a rental vehicle. The cutoff age is 65 because after this point the individuals will raise the concerns of the insurance providers, which will cost more money to insure them. We will also be targeting individuals looking to rent vehicles, whether it be for a holiday getaway or simply for any other reason they might need one, as well as individuals/companies that are interested in posting a vehicle they have available for rent onto the web-application.

2. Project Approach

2.1 Development Methodology

Agile was chosen as our development methodology for its incremental approach, which allows for continuous delivery of code and working software in iterations that are manageable. This approach also promotes collaboration, which ensures that the codebase is constantly fresh in our minds. Unlike traditional development methods, e.g. waterfall, Agile allows us to change and redefine our project through increments and ensures that adjustments can be easily made based on feedback from users, or new constraints. Agile methodologies will reduce the need for large reconstruction and ensure flexibility when adapting to changing needs or requirements.

2.2 Project Timeline

Sprint 1 - February 12th

- Create repository, ReadMe file and Wiki pages
- 6 user stories
- Appendix Project Approach and Technology Stack Selection

Sprint 2 - March 11th

- Front-end components
  - Page to display available cars for rent
  - Header with dropdown calendar for user to pick dates for their reservation
- Back-end components
  - Functionality to create an account
  - Functionality to successfully login with the correct email and password

Sprint 3 - March 25th

- Ability to create a reservation, as well as modify/cancel it.
- Admins of the site are able to alter the reservation information of a user.

Sprint 4 - April 10th

- Ability for admins to alter the information of cars

2.3 Collaboration and Communication

Our 2 main communication channels are Instagram and Discord in case certain members cannot attend an in-person meeting. Instagram is used to coordinate when meetings will occur, as well as any quick questions or concerns anyone has. However, solely communicating on instagram has not proved as efficient as working together in-person as a team. Therefore, we aim to meet up three times a week. The first time being sometime during the week from Monday to Thursday, and the other 2 meetings are before the lab session with the T.A, and another meeting takes place after the lab to discuss who will be assigned which task based on the T.A's feedback. As for the collaboration of the code, we will use GitHub in order to merge our code all together.

3. Technology Stack

1) React
2) Node Js
3) Express Js
4) Mongodb

3.1 Backend Frameworks

- 3.1.1 Express Js

  Description: Express JS is a framework that works with Node Js that enables people to use its features for web and mobile development. It can be seen as a layer on top of Node Js that helps people out with routes and servers.

  Rationale: Express Js provides a minimalist and flexible approach to web application development, making it pretty user friendly because it does not impose many restrictions when programming. As well as the fact that it integrates well with the Node Js ecosystem which we will be using.

  Justification for choosing Framework A: Express Js has abundant community support with a bunch of documentation and ongoing development in the language making it a great choice. Express.js is highly scalable, making it suitable for building applications of varying sizes and complexity. It allows developers to architect their applications in a modular and scalable manner, enabling easy horizontal scaling by distributing the workload across multiple servers or processes. Express.js is designed to be lightweight and opinionated, making it effortless to integrate with other libraries, frameworks, and tools.

  Qualitative Assessment:

Strengths: Node.js employs an event-driven, non-blocking I/O model, which allows it to handle a large number of concurrent connections efficiently. Node.js allows developers to use JavaScript for both server-side and client-side programming, enabling full-stack development with a unified language and codebase. Node.js has a large and active community of developers, contributors, and enthusiasts who contribute to its ongoing development, provide support, and share knowledge through forums, meetups, conferences, and online communities.

Weaknesses: Asynchronous programming in Node.js heavily relies on callbacks, which can lead to callback hell – a complex and difficult-to-maintain code structure characterized by nested callbacks and callback chains. Node.js operates on a single-threaded event loop, which means it can only utilize a single CPU core at a time.

Use Cases: Building real-time applications that require bidirectional communication between clients and servers (e.g. chat applications, collaborative tools, online gaming platforms, and live streaming services), and building lightweight and high-performance API servers that serve as backends for web and mobile applications.

- 3.1.2 Django

Description: Django is a high-level web framework written in Python that encourages rapid development and clean, pragmatic design. It follows the Model-View-Template (MVT) architectural pattern, which is similar to the Model-View-Controller (MVC) pattern.

Rationale: Choosing Django as a web development framework can be justified based on several factors: Performance, Security and Maintenance. Django is optimized for performance and scalability, allowing developers to build high-traffic web applications efficiently. Django's clean and modular design, along with its extensive documentation and community support, simplifies the process of maintaining and extending web applications over time.

Qualitative Assessment:

Strengths: Django's built-in features and conventions enable rapid development of web applications, allowing developers to focus on implementing business logic rather than reinventing the wheel. It is also very scalable due to its modular architecture.

Weaknesses: Django's monolithic architecture may be less suitable for projects that require microservices-based architectures or have strict performance requirements for specific components. There is also quite a bit of a learning curve when dealing with it.

Use Cases: Content Management Systems, ECommerce platforms, social network sites.

- 3.1.3 Spring

Description: Spring is a powerful and widely adopted framework for building enterprise-level Java applications. It provides comprehensive infrastructure support and a rich set of features for developing robust, scalable, and maintainable applications.

Rationale: Spring is a very lightweight container and has very efficient dependency injection mechanisms which in turn will allow us to have great scalability. It also provides good security due to its built in features. With a dependency injection being so easy, it allows for good maintainability as well.

Justification for Spring: Spring's lightweight container and efficient dependency injection mechanism contribute to excellent performance and scalability. Spring provides robust security features and best practices for securing Java applications against common security threats and vulnerabilities.

Qualitative Assessment:

Strengths: Spring has a large and active community of developers, contributors, and users who provide support, share knowledge, and contribute to the ongoing development of the framework and its ecosystem. Spring's flexible architecture and pluggable design allow developers to customize and extend its functionality to meet the specific requirements of their applications.

Weaknesses: Spring's comprehensive feature set and complex configuration options may have a steep learning curve for beginners or developers new to Java enterprise development. Spring's XML-based configuration and annotation-driven approach may lead to verbose configuration files and boilerplate code, especially in large-scale applications with numerous components and dependencies.

Use Cases: Enterprise applications, Microservice Architecture, RESTful APIs

3.2 Frontend Frameworks

- 3.2.1 React

Description: React is a JavaScript library for building user interfaces, primarily focused on developing single-page applications (SPAs). React utilizes a virtual DOM (Document Object Model) to efficiently update and render UI components, resulting in improved performance and responsiveness.

Rationale: React's virtual DOM and efficient reconciliation algorithm optimize rendering performance by minimizing DOM updates and re-renders. React's component-based architecture and declarative programming model simplify the process of building and maintaining complex user interfaces. React has a large and active community of developers, contributors, and libraries that provide extensive support, documentation, and third-party components for building React applications.

Qualitative Assessment:

Strengths: React's component-based architecture promotes code reuse, modularity, and maintainability by encapsulating UI logic and state within reusable components. React's virtual DOM and efficient reconciliation algorithm optimize rendering performance by minimizing DOM updates and re-renders.

Weaknesses: React's component-based architecture and ecosystem of tools and libraries may have a steep learning curve for beginners or developers new to frontend development. eact's built-in state management can become complex and difficult to manage in large-scale applications.

Use Cases: Single Page Applications, Progressive Web Apps, Component Libraries, Interactive Dashboards.

- 3.2.2 Angular

Description: Angular is a TypeScript-based open-source framework for building web applications, maintained by Google and a community of developers. Angular provides a comprehensive platform for building modern, dynamic, and interactive single-page applications (SPAs) and progressive web apps (PWAs).

Rationale: Angular's architecture and built-in optimizations, such as ahead-of-time (AOT) compilation and lazy loading, contribute to excellent performance and rendering speed. Angular's modular architecture and dependency injection mechanism promo code organization, reusability, and maintainability, making it well-suited for building large-scale applications with complex requirements.

Qualitative Assessment:

Strengths: Angular's modular architecture and component-based structure enable developers to organize code logically, promote code reuse, and facilitate collaboration among team members. Angular is built on top of TypeScript, a superset of JavaScript that adds static typing and advanced language features to the JavaScript ecosystem.

Weaknesses: Angular's comprehensive feature set and opinionated architecture may have a steep learning curve for beginners or developers new to frontend development. Angular's opinionated approach and built-in abstractions may introduce complexity and boilerplate code, especially for small or simple applications.

Use Cases: Enterprise applications, Admin Dashboards, Real-time Applications.

- 3.2.3 Next Js

Description: Next.js is a React-based open-source framework for building server-side rendered (SSR) and statically generated web applications. Next.js simplifies the development workflow by providing a zero-configuration setup and built-in support for features like routing, data fetching, and API routes, enabling developers to focus on building great user experiences.

Rationale: Next.js optimizes performance by implementing server-side rendering (SSR) and automatic code splitting out of the box. Next.js provides a seamless development experience with features like file-based routing, API routes, and built-in CSS and Sass support. Next.js supports static site generation (SSG) in addition to server-side rendering (SSR), enabling developers to pre-render entire pages at build time and serve them as static assets.

Qualitative Assessment:

Strengths: Next.js supports static site generation (SSG), allowing developers to pre-render entire pages at build time and serve them as static assets. ext.js provides a rich set of developer tools, including Next.js CLI, developer-friendly error reporting, and real-time updates with hot module replacement (HMR).

Weaknesses: Next.js may have a learning curve for developers new to server-side rendering (SSR) and static site generation (SSG) concepts, as well as React development principles. ext.js's server-side rendering (SSR) and static site generation (SSG) capabilities may introduce

complexity for simple or static websites that do not require dynamic content or server-side logic.

Use Cases: Content Websites, E-Commerce Platforms, Dashboards and Analytics tools.

4. Integration and Interoperability

4.1 Backend-Frontend Integration

This project uses an MVC architecture pattern to integrate the Express Js Back end API, with the React front end Application. The models and controllers are housed in an API folder which is using the aforementioned Express js Framework. The React-client folder exists adjacent to the api folder, and will act as the views portion of the projects. Routes in the api project are fetched by react functions which can handle data and display them in views.

The API and react servers run concurrently thanks to the configuration of the *npm start* script in the package.json folder within the react-client.

4.2 Third-Party Services

MongoDB will be used for the database system for the web application. It will be connected to a third party MongoDB data cluster that will be connected to the Express Js api, using a Mongo URI key. This data cluster will contain the User and Car models to be implemented during the development process.

5. Security Considerations

5.1 Frontend Security

Input Validation and Sanitization: We plan to implement client-side validation to ensure that user inputs are sanitized and validated before sending them to the backend. This will help prevent injection attacks and ensure better data integrity.

Secure Authentication: Next, we will make use of industry-standard authentication mechanisms such as JSON Web Tokens (JWT) to securely authenticate users. This includes ensuring that sensitive user data, such as passwords, is hashed before transmission and storage.

5.2 Backend Security

Authentication and Authorization: This project will implement robust authentication mechanisms, such as JWT, and enforce authorization rules to ensure that only authenticated users can access protected resources, notably rentals, while also having 2 degrees of access: regular user and admin.

Input Validation: We will be validating and sanitizing all user inputs on the server-side to prevent injection attacks such as SQL injection or NoSQL injection.

6. Conclusion

In conclusion, Agile development methodologies will be used for this project, adopted for its incremental approach, promoting continuous delivery and collaboration. It allows for flexibility in adapting to changing requirements and feedback from users. The project timeline is structured into sprints with clear objectives. The timeline ensures manageable iterations and progress tracking, to facilitate efficient development.

A brief summary of the technology stack includes, starting with the backend framework, Express Js; selected for its minimalist and flexible approach to web development, it integrates well with Node Js, providing robust routing and server capabilities. Its lightweight nature and abundant community support make it suitable for building scalable backend APIs. For frontend frameworks, React is chosen for its component-based architecture and efficient rendering capabilities. React simplifies the development and maintenance of complex user interfaces. Its large community and extensive ecosystem further support the project's goals.

For integration and interoperability, by employing an MVC architecture pattern, the backend Express Js API will be integrated seamlessly with the React frontend application, ensuring smooth communication and interaction between the two layers.

The security considerations include, in the frontend, client-side validation and sanitization, implemented to ensure data integrity and prevent injection attacks, as well as secure authentication mechanisms such as JSON Web Tokens (JWT), utilized to protect sensitive user data. In the backend, robust authentication mechanisms and authorization rules will be enforced to control access to protected resources. Additionally, input validation and sanitization on the server-side will mitigate security vulnerabilities.

By selecting a robust project approach and technology stack, along with stringent security measures, our development team aims to deliver a functional, user-friendly web-application for vehicle rental bookings that meets the needs of both users and administrators, while ensuring data privacy and integrity.