UNIVERSITÉ Concordia UNIVERSITY

Project Approach & Technology
Sprint 1

Course: SOEN 341

Due: February 12, 2024

Team Name: SOEN 341

Hrag Bankian - 40245363
Sevag Merdkhanian - 40247912
Élian Gadbois-Roy - 40208293
Johnny Dang - 40245598
David Cohen - 40130406
Harun Slahaldin Omar - 40250981

1. Project Overview..3
1.1 Project Objectives..3
1.2 Scope..3
1.3 Target Audience..4
2. Project Approach..5
2.1 Development Methodology..5
2.2 Project Timeline..5
2.3 Collaboration and Communication..6
3. Technology Stack..7
3.1 Backend Frameworks..7
3.1.1 Next.js..7
3.1.2 Java Spring..7
3.1.3 Django..8
3.2 Frontend Frameworks..8
3.2.1 React..8
3.2.2 Angular..9
3.2.3 Svelte..9
3.3 Final decision..9
4. Integration and Interoperability..10
4.1 Backend-Frontend Integration..10
4.2 Third-Party Services..11
5. Security Considerations..12
6. Conclusion..13

1. Project Overview

1.1 Project Objectives

The primary goal of the project is to develop a middle-fidelity prototype of a car rental web application following Agile methodologies. This project aims to provide an interface for customers to browse, reserve, and manage vehicle rentals, while also enabling customer service representatives (CSR) to efficiently handle check-in and check-out processes. The system administrator will have administrative functionalities to manage vehicles, user accounts, and reservations.

1.2 Scope

The project will focus on developing a functional prototype with the following key features and functionalities:

1. **Customer Features:**
    - Browse vehicles for rent.
    - Start, modify, or cancel reservations.
    - Add extra equipment during reservation.
    - Find nearest branches.
    - Provide ratings and reviews for rented vehicles and overall rental experience.
2. **Customer Service Representative Features:**
    - Check-in process for customers with or without reservations
    - Check-out process including final billing and payment settlement.
3. **System Administrator Features:**
    - CRUD operations on vehicles
    - CRUD operations on user accounts
    - CRUD operations on reservations

1.3 Target Audience

This system is designed to serve Customers, Customer Service Representatives, and System Administrators and their respective needs.

1. **Customer's Needs:**
    - Provide a catalog to browse available vehicles for rent.
    - Efficient reservation process with options to specify pickup and return locations, dates, and vehicle preferences.
    - Ability to add extra equipment during reservation (ex: car gets delivered to your house).
    - Convenient access to modify or cancel reservations.
    - Find nearest rental branches for pickup and return.
    - Provide feedback and ratings for rented vehicles and overall rental experience.

2. **Customer Service Representative's Needs:**
    - Checking in customers, whether they have a reservation or not.
    - Capability to confirm reservations, review rental agreements, and process payments efficiently.
    - Ability to handle the check-out process including vehicle inspection, final billing, and payment settlement.
    - System to confirm the completion of the return process and update reservation status accordingly.

3. **System Administrator's Needs:**
    - Manage vehicle catalog including adding, editing, and removing vehicle listings.
    - Manage user accounts, including registration, login, and updating user information.
    - Ability to view, modify, or cancel reservations as necessary.
    - System to track rental activity, manage inventory, and ensure smooth operation of the rental process.

2. Project Approach

2.1 Development Methodology

The development methodology will be Agile Scrum, as the initial requirements for the project are broad, and some decisions are best made later in the development process. Agile's iterative approach allows for flexibility and adaptation as precise requirements are defined over time. By breaking down the project into smaller, manageable increments, agile enables continuous feedback and collaboration between developers and stakeholders (the TA and Professor). This assures that the website always meets stakeholders' requirements, all while allowing for small improvements overtime. This also minimizes development time involving the re-designing of any features. In addition, Agile allows for early testing and feedback from users, allowing developers to rapidly fix bugs and issues. Overall, Agile Scrum is the ideal method for this project.

2.2 Project Timeline

Here are some major milestones and deadlines for Sprint 2.

1. February 16: CRUD operations on users.
2. February 20: CRUD operations on vehicles
3. February 26: CRUD operations on reservations
4. March 3: Browse vehicles for rent
5. March 7: Start a reservation
6. March 11: Manage reservations

CRUD Operations on users, vehicles and reservations come first as they ensure that the application's backend infrastructure is robust, reliable, and capable of handling data management effectively. Afterwards, the ability to browse vehicles will be added, as users cannot start a reservation for a car when there are no cars available to browse. Then, the functionality for starting a reservation will be implemented. Finally, the ability to view, modify or delete a reservation will be added. The details for what these features involve are outlined in the Sprint Planning.

2.3 Collaboration and Communication

For effective collaboration and communication, the team will leverage tools such as GitHub and Git for version controlling code. GitHub issues will serve as the main platform for defining and assigning user stories to team members. In addition, GitHub Wikis will enable the team to maintain high-quality documentation for the project and its progress, as well as assist in Sprint Plannings. These tools will ensure that the team is on the same page at all times, while minimizing any unnecessary confusion or problems. In addition to GitHub Wikis, Google sheets will be used to plan sprints during Sprint Plannings, mainly to define concrete tasks with estimated story points, due dates, priority and risks. These tasks will be the result of discussion within the team as well as the internal stakeholders. While Sprint Plannings will be held before any given sprint, Sprint Reviews and Retrospectives will be held after sprints to summarize what was done during the sprint, what went well and what could be done better, as well as for updating any documentation. In addition to these meetings, the team will meet for meetings during sprints as needed.

3. Technology Stack

3.1 Backend Frameworks

3.1.1 Next.js

- Description:
  - Next.js is a React framework that uses Node.js. It can be used as a full stack framework in order to build a multitude of applications.
- Rationale:
  - The whole project could be written using the same framework.
  - Really easy to deploy, less confusing for people who don't have web development experience because it uses a lot of abstraction.
- Qualitative Assessment:
  - Strengths: Really easy to use and deploy something simple fast,
  - Weaknesses: Relatively new and not mature, weaker documentation relative to older and more flexible solutions.

3.1.2 Java Spring

- Description:
  - Java Spring is a Java backend framework that is widely used in the industry.
- Rationale:
  - Useful to learn if you want to work in the industry as a backend developer.
  - Forces good practices and organization of code.
  - Is a robust framework with decades of documentation and probably the most secure.
- Qualitative Assessment:
  - Strengths: Very well documented, forces good practices, very secure and robust, uses a language that everyone knows in SOEN.
  - Weaknesses: Complexe to deploy and setup, has more overhead than other options since its java and has a lot of features we won't be using.

3.1.3 Django

- Description:
  - Django is arguably the most popular Python backend framework which allows for fast deployment of web applications.
- Rationale:
  - Simpler than Java Spring but still has a lot of features and a mature community.
  - Easier to learn than Java Spring since it has less features and is more straight to the point.
- Qualitative Assessment:
  - Strengths: Easy learning curve, easy language to understand for newcomers to programming, has less overhead than Java Spring.
  - Weaknesses: Complex to deploy compared to Next.js, more complex to understand for newcomers to web development.

3.2 Frontend Frameworks

3.2.1 React

- Description:
  - React is a javascript frontend framework developed at Meta.
- Rationale:
  - Next.js is a backend framework based on React which would make everything easier to understand.
  - React is the most popular frontend framework right now and its momentum doesn’t seem to slow down. Useful to learn if you want to work in the industry. Lots of documentation.
- Qualitative Assessment:
  - Strengths: Most documented frontend framework by far, easy to deploy with Next.js, a lot of people in the team know about it or worked with it.
  - Weaknesses: Might want to try something new instead of react since almost everyone knows it.

3.2.2 Angular

- Description:
  - Angular is a javascript frontend framework developed at Google.
- Rationale:
  - Angular is widely used for enterprise applications because of its structure.
  - It would be beneficial to learn because it is widely used in the industry for complex applications.
  - Well organized by default.
- Qualitative Assessment:
  - Strengths: Forces us to use good structure (separate all files type by default), well recognised in the industry
  - Weaknesses: More complicated to deploy and develop, especially for new developers. Not integrated with any backends by default.

3.2.3 Svelte

- Description:
  - Svelte is a relatively new open source front end project
- Rationale:
  - Svelte is really simple to use, uses the latest tech that improves performance like SSR and is easy to learn.
  - It would be really easy to implement more complex stuff using Svelte kit or other component library.
- Qualitative Assessment:
  - Strengths: Most reactive framework using new tech, really performant, scoped style.
  - Weaknesses: Relatively new and not established, not a lot of documentation.

3.3 Final decision

After long deliberation, we decided to use Next.js for both our front end and back end. Creating a full stack application with Next.js is the most convenient solution for us because we have team members who are not as experienced in web development. Next.js permits us to code everything in one language (Typescript) so the learning curve will be smaller for newcomers to web programming. It also allows us to deploy our project directly to Vercel’s free plan without any additional configuration. Having team members that have worked with Next.js in the past, we think this is the best compromise. This is because even though it's a new framework, we can refer to all of React's documentation if we struggle with bugs or issues.

4. Integration and Interoperability

4.1 Backend-Frontend Integration

Next.js provides a robust platform for building full-stack web applications, offering features and tools to effectively integrate frontend and backend technologies while ensuring security, performance, and maintainability. Here's some of the strategies that will be used with the help of those features and tools:

- API Design: Next.js provides built-in API routes, allowing us to define server-side endpoints directly within our application. We can create API routes using the 'app/api' directory, making it easy to define backend logic and communicate with the frontend through RESTful or GraphQL APIs.
- Folder Structure: Next.js encourages a clear separation of concerns by providing a structured project layout. The '/app' directory mostly contains all the routes, components, and logic for the application. Within this folder, 2 main directories are used to separate the frontend and backend. First, the '/app/ui' directory contains all the UI components for our application, such as tables, forms, buttons, nav-bar, etc. Secondly, the '/app/lib' directory contains functions used in our application, such as reusable utility functions and data fetching functions as well as the type definition of our data and how they should be accepted and stored.
- Use of Frameworks and Libraries: Next.js is a React framework that provides many features out of the box, such as server-side rendering, static site generation, and API routes. It integrates seamlessly with other popular libraries and frameworks, allowing us to leverage the vast ecosystem of React and Node.js libraries for both frontend and backend development.
- Data Storage/Fetching: By leveraging MongoDB as the backend database and Next.js as the frontend framework, we aim to implement a comprehensive booking system with real-time availability tracking and flexible reservation options to optimize the user experience. MongoDB will ensure efficient storage and retrieval of rental vehicle data, while Next.js will facilitate seamless communication between the frontend and backend through API routes.

4.2 Third-Party Services

In this project, two third-party APIs will play crucial roles:

Firstly, we'll integrate a Car API to populate our application with a comprehensive catalog of rental vehicles. By leveraging this API, we can effortlessly access a wealth of information about available cars, ensuring that our users have access to a diverse selection of vehicles for their rental needs. This integration will streamline the process of displaying vehicle details such as the model, type, price range, etc in a user-friendly format, enhancing the overall browsing experience.

Secondly, we'll implement Google's Map API to enhance the user interface by providing a visual representation of key locations such as rental pickup points, branch locations, and nearby Airports. By integrating this API, we can dynamically generate interactive maps that offer users valuable context and help them navigate their rental experience more efficiently. This visual component will significantly improve user engagement and satisfaction, contributing to the overall success of the application.

5. Security Considerations

There are many important security concerns when it comes to an online service like renting out cars. A ton of information from the rentee is required in order to prevent but also assure liability in case of accidents, theft, or traffic law violations, amongst other possible consequences which the renter would otherwise be responsible for. The best way to survey these concerns is to observe a real-life example. Communauto is one such service, although the features it offers are quite extensive compared to the specifications of this project.

The user data to be collected to set up a reservation specified in the project outline only consist of postal code they are renting at, and pickup and return dates. Apart from a brief mention of user accounts, that is all. User accounts alone, however, entail a whole lot of sensitive information being kept in one place.

Communauto, for example, requires copies of your driving record and claims history. A driving record by itself contains: full name, driver's license number, date of birth, sex, height, license status, health conditions pertaining to safe driving, address, etc. This is a goldmine of information asking to be sold: this would result in consequences ranging from simple privacy breaches to identity theft. Of course, such a service is not required to keep all of this information online, and records are surely discarded upon client registration, but this does not bar the information that is conserved, or the information exchanged at time of registration.

That being said, what can be done to prevent access to these user data?

- CAPTCHAS's could prevent brute-forcing access
- Two factor authentication could prevent access to persons having obtained user credentials
- Frontend-backend encryption
- Frequent updates
- Password strength requirements
- Use a secure hosting service

6. Conclusion

In conclusion, our project approach and technology stack have been meticulously chosen to ensure the successful development of a middle-fidelity prototype for a car rental web application.

We have opted for an Agile Scrum methodology, which offers flexibility and adaptability to evolving project requirements. By breaking down the project into manageable increments, we can continuously engage with stakeholders, incorporate feedback, and ensure that the final product meets their needs effectively. This approach allows for early testing, rapid bug fixes, and iterative improvements, aligning perfectly with our project objectives.

Our technology stack revolves around Next.js, a versatile React framework that serves as both the frontend and backend solution. This choice was made to streamline development, particularly for team members with varying levels of web development experience. Next.js offers simplicity, ease of deployment, and the ability to code in a single language (Typescript), which reduces the learning curve for newcomers and ensures faster development. Additionally, Next.js integrates seamlessly with MongoDB for efficient data storage and retrieval, further enhancing our application's functionality.

In terms of frontend frameworks, we considered various options but ultimately settled on React due to its popularity, extensive documentation, and compatibility with Next.js. This decision ensures familiarity among team members and provides access to a vast ecosystem of resources and libraries to support development efforts.

For backend frameworks, while Java Spring and Django were strong contenders, we concluded that Next.js provides a more cohesive solution for our full-stack application. Its simplicity, ease of deployment, and alignment with React make it the ideal choice for our project's backend needs.

Furthermore, our integration and interoperability strategy prioritize clear API design, structured folder organization, and seamless data communication between frontend and backend components. Leveraging third-party services such as a Car API and Google's Map API enhances the user experience and adds valuable functionality to our application.

Finally, we acknowledge the critical importance of security considerations, especially when handling sensitive user data. While our prototype may not require extensive personal information collection like some real-world car rental services, we are committed to implementing measures such as CAPTCHAs, two-factor authentication, encryption, and frequent updates to safeguard user data and ensure privacy and security.

Overall, our chosen project approach and technology stack are tailored to meet our project objectives efficiently, empower our team members, and deliver a robust and user-friendly car rental web application prototype.