Project Approach & Technology

by

Mahmoud Mustafa (40262016)

Vincent Nguyen (40246406)

Azal Al-Mashta (40179492)

Thomas Mahut (40249811)

Adam Tahle (40237870)

Alex Page (40193184)

A technical report

submitted in partial fulfillment

of the requirements of

SOEN 341

Concordia University

February 2024

Project Overview

Project Objectives

The primary objective of this project is to develop an online car rental application by employing the agile scrum methodology. By the end of the four sprints, we expect to have a middle-fidelity prototype that we showcase to demonstrate what our marketable product would look like. As a team, we are looking to refine our web programming skills, and to specifically gain experience using Express, MongoDB, and front-end libraries such as HTMX and Bootstrap. We are further aiming to familiarize ourselves with agile techniques such as scrum meetings, user stories, task breakdowns, and incremental development.

Scope

The scope of this car rental platform encompasses core features for the online counterpart of an existing car rental company. Although vehicle availability and rental locations will be fabricated, they could be swapped out for actual data and the application would be immediately usable for a real-world car rental company. Core features include account creation, vehicle filtration, and vehicle reservation.

Target Audience

The application’s intended users are tourists needing a rental for the duration of their stay, people from abroad who are staying for an extended period of time, and newly arrived immigrants in need of a rental to ease the transition into their new lives. The

application will thus serve the needs of foreigners needing a rental for a period of time ranging from a few hours to a few weeks.

Project Approach

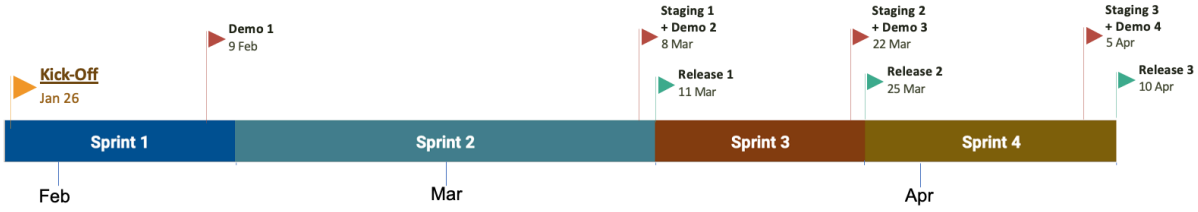
Development Methodology

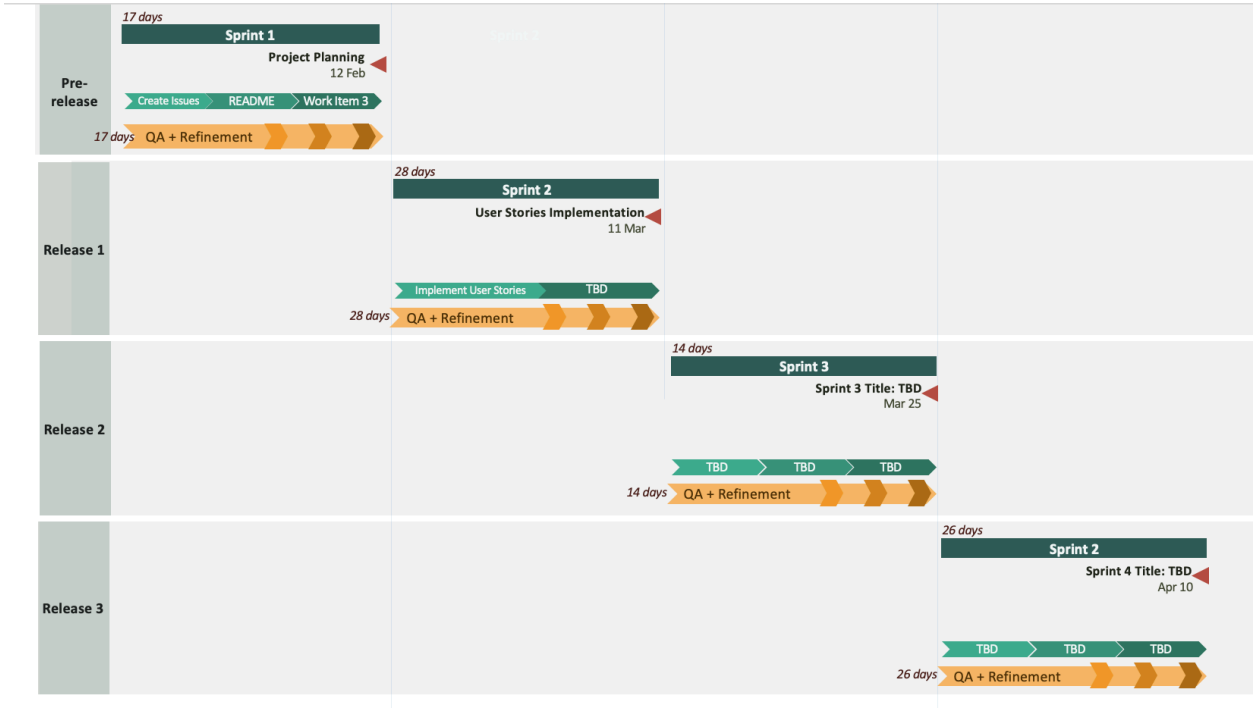
Agile, specifically scrum, is the chosen development methodology for this project, as it enables teams to rapidly develop and deliver software. This choice is made possible by the team's access to an acting client (the teaching assistant) that can provide feedback at each incremental delivery. This feedback loop ensures adaptability to changing requirements, enhancing our ability to deliver a minimum viable product within the given, restricted time frame.

Project Timeline

On the following page is a high-level timeline of the project's major milestones and deadlines. The work items for future sprints have yet to be determined due to the nature of the project.

High-Level Project Timeline





Collaboration & Communication

Informal Communication

The team's informal communication will occur via a WhatsApp group chat where members can communicate and coordinate for meetings (time, location, availability, etc.). All team members are expected to check this group chat at least once per day. In the event of disputes, members are expected to move conversations to the team's private forum on moodle, to which the professor has access and can offer guidance on resolving the conflict.

Formal Communication

As for formal, implementation-oriented communication, GitHub provides a workspace for team members to collaborate, review, and document features and bugs. Sprints and major iterations will be managed using GitHub's milestones feature, and user stories will be represented by GitHub issues which can be broken down into smaller tasks. GitHub allows us to easily label and assign issues, as well as to create branches from them and assign code reviewers for pull requests.

Technology Stack

Backend Frameworks

As per the professor's suggestion to build the application using the model-view-controller (MVC) design pattern, three backend frameworks in particular stood out as viable options: Ruby on Rails, Express, and Laravel.

Ruby on Rails

Description

Ruby on Rails (RoR) is a web application framework that emphasizes convention over configuration. It comes with everything you might need for a database-backed web app, such as an ORM, routing, and even view rendering. It is thus technically a full stack framework, but can be used as a uniquely backend framework when paired with an alternative front end solution.

Rationale

The motivation to consider RoR is that it is heavily opinionated towards the MVC architecture, which can greatly reduce development time in the context of this project. The framework is suited with code generators that automate the creation of boilerplate code for models, controllers, views, and database migrations.

Development is further facilitated thanks to RoR's active record ORM that facilitates database interactions and management, and its rich ecosystem of gems (libraries) to easily integrate common functionality.

RoR is additionally well documented, has a large, active community, and provides testing tools that promote test driven development (TDD). These factors allow developers to decrease the time spent troubleshooting chasing bugs and to focus instead on building core features.

However, our team has virtually no experience with the Ruby programming language, which is considered to have a steep learning curve. Further, the framework's emphasis on convention may lack a desired flexibility to implement specific features.

Qualitative Assessment

In the context of this project, and relative to the other frameworks considered here, RoR's strengths are its community support, ease of integration, extensibility, and learning curve (not counting the learning curve of the Ruby language). Its

disadvantages are then its complex scaling solutions, poor performance, potential difficulty to maintain, and poor out-of-the-box security.

Due to its rapid development, RoR is a popular choice for building MVPs. Large applications like Shopify and GitHub were, and still are, built on RoR. Not only did RoR allow these companies to build fast, but also handle increasingly complex business logic as the years passed.

Express

Description

Express is a node.js web framework, designed to be minimal, flexible, and fast. The framework handles the creation of a node.js web server and simplifies the creation of RESTful APIs, allowing developers to focus on building core functionality.

Rationale

Express.js is widely used and has a large community, such that solutions to common problems are readily available. It is relatively simple to learn, and its use of JavaScript means that developers with no web dev experience only need to learn a single language to work on both the frontend and backend.

Express's minimalistic design allows teams to extend its functionality as needed, which can be particularly beneficial team's whose requirements are prone to rapidly change. Its middleware architecture facilitates the integration of new features, making it ideal for developing modern, efficient, and modular backend systems.

Unlike RoR, it is un-opinionated, meaning the team can decide to follow an architectural pattern of their choice, such as MVC. However, this same opinionated nature means the framework lacks some built-in features that others might provide. This results in a reliance on third-party libraries for certain functionalities. While this means the team can decide which 3rd party technologies to use for security, testing, database management, etc., it simultaneously increases the complexity of the project.

Qualitative Assessment

In the context of this project and relative to the other frameworks considered here, Express's advantages are its community support, scalability, ease of integration, performance, maintenance, learning curve, and extensibility.

Its main disadvantage is its security, as the framework's lack of built-in security features has given it a reputation for having security vulnerabilities. The solution is often to rely on 3rd party libraries for security features, which can increase the complexity of the project and of dependency management.

A few use cases include real-time applications (due to its middleware support and non-blocking I/O operations), API development (due to its flexible and modular design), and single page applications (due to its support for routing and middleware, and overall flexibility).

Laravel

Description

Laravel is an open-source PHP framework designed for web application development. Laravel follows the MVC (model-view-controller) pattern and offers an expressive syntax with sturdy features and a dev-friendly environment. Key focuses of Laravel are its eloquent ORM for database interaction, simple authentication systems, schema version control through database migrations and integrated testing tools. Since being founded in 2011, Laravel became a famous choice thanks to its modular architecture and comprehensive, easy to use toolkit for simple and complex web applications.

Rationale

Laravel offers an intuitive and easy to understand syntax and code structure, which makes it easy to learn/use. Compared with other PHP frameworks, this simplicity accelerates development times, and reduces chances of having difficulties along the way.

Laravel also has a very large community that offers a lot of support through forums, blogs and tutorials. This ensures the availability of necessary resources to learn and overcome any struggles during the development process.

This framework will also come with all necessary equipment such as authentication, database migrations and task scheduling. These built-in tools make it all inclusive in one app and minimize the reliance on third-party programs or libraries.

Qualitative Assessment

Compared to other PHP frameworks, Laravel thrives in code simplicity, community support, built-in features and Eloquent ORM. Which are all very desirable when choosing a framework.

On the other hand though, Laravel may struggle with a steep learning curve, complexity for small projects, version compatibility issues, and shared hosting limitations that require specific server configurations.

Key use cases of Laravel include: enterprise level apps, as it can handle numerous users, ensuring high performance and maintaining secure code; industry specific apps, as developers can leverage its features to build solutions for specific business needs; banking and fintech applications, thanks to its out-of-the-box security and reliability features; e-commerce Platforms, due to its extensibility that can handle integration of payment gateways, inventory management, and order processing.

Frontend Frameworks

Roughly in order of increasing abstraction, the three front-end solutions considered for this project are HTML/CSS/JS & libraries, Vue.js, and React.

HTML, CSS, JS & Libraries

Description

HTML is the standard markup language for creating web pages, CSS is the stylesheet language used to style & format HTML, and JS is the scripting language used to make

web pages dynamic. Libraries such as Bootstrap, JQuery, or HTMX, build on top of these 3 technologies to provide developers with pre-defined and reusable functionality.

Rationale

Since our team is generally inexperienced at building full-fledged web applications, building with basic HTML, CSS, and JavaScript is straightforward and allows members to gain a solid understanding of these core web technologies.

We can offset the increased development time (due to building the client-side from the ground up) by using libraries such as Bootstrap, JQeury, and/or HTMX. These libraries also allow us to easily extend functionality.

Using these core technologies also allow us to have total control of the client-side of the application, which can be beneficial for learning and debugging.

Qualitative Assessment

In the context of this project, and relative to the other front-end frameworks considered here, the following are advantages of using HTML/CSS/JS & libraries: ease of integration, developer experience, responsiveness, performance, cross browser compatibility, community support and high potential capabilities.

The following are then considered as weaknesses of this front-end solution: poor modularity potential and lack of component libraries.

The choice of HTML/CSS/JS/Libraries over a full fledged front-end framework is often made for static sites, prototypes, lightweight applications, and performance-critical applications.

React

Description

React is a javascript library that is commonly used for building user interfaces. Although React is a library, the terms “library” and “frontend framework” are usually interchangeable in React’s context. Every React web app is composed of reusable components which will then make up parts of the user interface. That means we can have a separate component for each part (nav bar, footer, main etc.) React components are reusable code blocks that form the UI of a React application. Also in a team it would permit us to easily combine components written by different people. On top of that we can add interactivity to components and update the screen in response to different user actions. On top of that, React uses a virtual DOM, which improves the smoothness and user experience overall. All of this makes React a great choice for someone who is looking to build an entire app or a dynamic web interface.

Rationale

React’s core concept revolves around working with components, as we said earlier these reusable building blocks will allow devs to create nice modular UI elements. Whether it’s a small button or a whole page, React will make it simple to manage and maintain our user interface.

React uses a virtual DOM to optimize rendering performance. React will calculate the minimal updates needed when data changes so that it efficiently updates the actual DOM. This would result in a faster page rendering and smoother user experience.

React also has a vast ecosystem of third-party libraries and tools with some like React Router, Redux or tailwind CSS that respectively help with routing, state management and styling. This rich ecosystem allows us devs to choose and customize tools tailoring them to our needs.

React has a thriving community with a lot of tutorials, blogs, stack overflow discussions etc. Whether we're beginners or experts, we will find answers to any question we have.

React's maintenance by Meta and its use by many major companies such as Instagram, AirBnB, etc. is evidence for the library's reliability.

The learning curve is gentle, as we are familiar with JavaScript and HTML we can start building React apps quickly, it's simple and reduces the barrier of entry for new devs.

Qualitative Assessment

The strengths of React are that it's easy to learn, it employs virtual DOM, it has reusable components, it follows one-way data flow and it's friendly with search engine optimization (SEO).

The weaknesses associated with React are lack of proper documentation, development speed, JSX complexity (could be a stumbling block for beginners) and problems with SEO (proper implementation requires expertise).

React's key use cases include single-page applications (SPAs), interactive user interfaces, progressive web apps (PWAs), real time apps like chat applications & game applications, and content management systems (CMSs).

Vue.js

Description

Vue.js is a frontend framework for building interactive web interfaces. It focuses on the view layer only, which makes it easy for us to integrate it with other libraries or existing projects. Vue.js provides the benefits of reactive data binding and composable view components with a simple API. It's capable of powering complicated single-page applications when used with the proper tools and supporting libraries.

Rationale

Vue.js has a lenient learning curve. Basic knowledge of HTML,CSS and JavaScript is all that is needed to start working with Vue.js. Learning Vue.js is made easier thanks to its active community, which provides a lot of resources and support.

It is tremendously adaptable, it is flexible and we do not have to understand all the concepts in-depth. It can easily integrate with existing MPAs and SPAs rendered by the server or client.

Vue is also known for offering multiple ways to be able to apply transitions to HTML elements when they are added, updated, or removed from the DOM.

Qualitative Assessment

Vue.js will flourish in simplicity with automating tasks, performance with operations through virtual DOM, a rich ecosystem and active community, a smooth learning curve and easy reusability with reusable single file components.

On the other hand though, Vue.js will face trouble in aspects such as language barrier as it was created by a chinese-american so a big part of the community is non-english speaking, which would raise some issues for english-speaking devs such as ourselves.

Although its community is active, it may be smaller in comparison to other frameworks. Scalability could be a challenge as well with large projects with vue.js' less mature ecosystem. And finally vue.js is reliant on external libraries from the javascript ecosystem for advanced features which would be a negative for the framework.

Some key use cases include: simple, interactive nav bars; built-in editors, as it can handle models with predefined values; dynamic order forms that show the total cost of certain selected services; instant search features.Popular web applications like Netflix, chess.com, GitLab and Grammarly rely on vue.js to create powerful user interfaces.

Chosen Frameworks

In between all the options we've described and researched, we've decided that the most suitable for our team would be Express for backend as it has a flush learning curve which is necessary in our tight time-frame, it also offers great community support for any bumps we might face along the road. Its main disadvantage, a lack of built-in security features, is a consequence of one of its advantages - flexibility. So while we must rely on third party packages to implement basic security measures, this allows us to tailor our security approach precisely to our project's needs.

Given our team's limited experience, we've chosen a straightforward combination of basic HTML, CSS, and JavaScript, complemented by select libraries instead of a full-fledged frontend framework. This streamlined approach prioritizes simplicity, minimizes complexity, and allows precise customization. By opting for a lightweight stack, we strike a balance between efficiency and functionality, ensuring a smoother development process tailored to our project's specific needs.

Integration and Interoperability

Backend-Frontend Integration

Our project will utilize Express on the backend and leverage HTML, CSS, and JavaScript on the frontend (including libraries that consist of these technologies). We will integrate the backend and frontend by using a templating engine to generate HTML

files on the backend, and we will then serve these to the client side with publicly visible CSS and JavaScript files.

The templating engine of choice is the embedded JavaScript (EJS) templating engine, so that templates can be defined using a familiar HTML markup and JavaScript syntax. Component-based reusability and custom functionality will be achieved on the front-end using JavaScript web components, where needed. The HTMX library will be used to add interactivity and partial page reloads, and the Bootstrap library will be employed to facilitate styling and to maintain a consistent look.

Third-Party Services

This project will leverage third party services to improve the user experience, such as the Google Maps API and the Gas Prices API. The Google Maps API (more specifically, either the Routes or Directions API) will be used to provide users with the nearest locations to pickup and dropoff rentals. The Gas Prices API will serve as an additional feature for convenience, so that users know what gas prices to expect when they hit the road.

Security Considerations

Backend Security Measures

On the backend, we will utilize Passport, a Node.js authentication middleware that can handle various user authentication strategies. We will use a *local* strategy, meaning we

will store user credentials in our own MongoDB instance, an approach made easier using the local-passport-mongoose package.

Database connection credentials, among other sensitive environment variables, will be kept in a single *.env* file. This file will be omitted from git commits, and its variables will be automatically loaded into a running process by the *dotenv* node package. This measure separates the sensitive information and credentials from the codebase.

Frontend Security Measures

On the frontend, we will implement cross-site scripting protection using packages like *helmet*, *helmet-csp*, and *cors*. Similarly, cross-site request forgery (CSRF) protection will be achieved using packages like *csurf* and by configuring cookie settings such that XSRF-token cookies are only sent in requests originating from the same site. These measures will help prevent attackers from performing unwanted actions from the client side.

Conclusion

After careful deliberation and analysis, we've decided on the agile development methodology, the Express backend framework, and the core frontend technologies HTML, CSS, and JavaScript for this project.

We chose agile for its flexible approach that prioritizes collaboration, adaptability and customer satisfaction. By delivering softwares in small incremental releases, it enables continuous improvement and responsiveness to changing requirements.

Express has been selected as the project's backend framework for being minimalist, simple and adaptable. The decision to rely on vanilla markup, styling, and scripting languages in addition to basic libraries was made for similar reasons. Both the backend and frontend solutions were selected for both their compatibility with project requirements, and for their beginner-friendly qualities.