GINA CODY
SCHOOL OF ENGINEERING
AND COMPUTER SCIENCE

SOEN 341 Software Processes: Project Approach and Technology Stack Selection

Team: Error 404
40212432 Sunil Kublalsingh
40245502 George Ezzat
40214571 Alex Bellerive
40244003 Marco Sicoli
40176937 Sumit Patel
40044952 Sebastian Iaricci

Project Approach and Technology Stack Selection **2**
1. Project Overview **2**
1.1 Project Objectives 2
1.2 Scope 2
1.3 Target Audience 2
2. Project Approach **3**
2.1 Development Methodology 3
2.2 Project Timeline 3
2.3 Collaboration and Communication 3
3. Technology Stack **4**
3.1 Backend Frameworks 4
3.1.1 Node.js 4
3.1.3 Ruby on rails 5
3.2 Frontend Frameworks 6
3.2.1 React.js 6
3.2.2 Angular 7
3.2.3 Django Python 7
4. Integration and Interoperability **8**
4.1 Backend-Frontend Integration 8
4.2 Third-Party Services 8
5. Security Considerations **9**
6. Conclusion **9**

Project Approach and Technology Stack Selection

1. Project Overview

1.1 Project Objectives

The goal and objectives of this project is to create a car rental website with an ample number of features for a seamless user experience. This website will allow the user to rent a vehicle for short periods of time, ranging from hours to weeks. The user will be able to view a catalog of a variety of vehicle types. During the users' reservations, they can choose from a variety of packets to add on to their rental experience. The user will also be able to edit their reservation, find the nearest branch, rate their experience, and leave a review for the specific car.

1.2 Scope

The scope of the project is to build a car-rental application. The key features of this application includes a secure login portal database for users. The database should manage the users based on the CRUD (create a user, read user information, update or edit user information, and delete a user) principals. The application will provide clients with a reservation system which asks users for location and drop off points for the vehicle as well as a schedule system which illustrates the duration the user intends to rent a vehicle for. Next, the user will choose a vehicle based on certain attributes such as price, type (car, van, truck), and size (compact, standard, intermediate). Payment information will then be added as well using a credit card entry portal. The reservation will also follow the CRUD principles to maximize user-friendliness and ease of use. Finally, there will be a review system which makes it possible for the users to add comments and ratings on their experience.

1.3 Target Audience

This application is for users who want a seamless experience doing booking and using a short-term rental vehicle to supply their needs.

2. Project Approach

2.1 Development Methodology

Agile is used throughout this project due to its importance on frequent, small updates to build a project, which is represented through the several sprints for this project. Continuous meetings and communication over platforms such as Discord and during in-person meetings are also key traits of the Agile principle which describes that interactions and individuals contain greater importance over tools and processes. The feedback review system that the application will contain correlates with the Agile principle of customer collaboration over contract negotiations. Finally, the principle of working software over documentation is highlighted in the project's emphasis on a very large coding requirement to create the application over writing reports.

2.2 Project Timeline

- **Sprint 1. (Feb. 12):**
    - Finish task breakdown for each team member (assign roles to each member).
    - Complete the project approach and technology stack selection, user stories, README file, and GitHub repository set up.
    - Complete as much of the preliminary program's code as possible.
- **Sprint 2. (Mar. 11):**
    - Finish preliminary coding to set up the foundation of the project's code. This will include setting up the login portal database, car catalog, and design choices for the presentation of the application.
    - Begin the main reservation system design and begin coding its foundation.
- **Sprint 3. (Mar. 25):**
    - Finish main coding to set up the reservation system (includes the booking, payment, and review feedback systems).
    - Begin writing the final report and presentation.
- **Sprint 4. (Apr. 10):**
    - Review the application and verify for bugs to make small edits if necessary.
    - Finish the report and presentation requirements.

2.3 Collaboration and Communication

Communication channels and collaboration tools used during this project are: Discord for message communication and virtual team meetings, Google docs and sheets or Microsoft Word and Excel for creating, sharing and editing reports and documents for the project, Github is used

so each team member can add and edit their assigned parts of the project code to the main program so it can be run and tested.

3. Technology Stack

3.1 Backend Frameworks

3.1.1 Node.js

Description: Open source server environment by using JavaScript on the server.

- **Rationale:**
  - Easy to set ups as a server.
  - Scalable.
  - Javascript
  - Large ecosystem
  - Large community for support
  - Fast
  - Cross platform
- **Qualitative Assessment**:
  - Strengths
    - Large community for support and troubleshooting.
    - Uses JavaScript so it is one language less to learn.
    - Fast execution
    - Large ecosystem for installing packets.
  - Weaknesses
    - Might be overkill for small applications.
    - Hard to maintain at scale.
    - Heavy computations.
- **Use Cases** (Source)
  - Netflix
  - NASA
  - LinkedIn
  - Twitter

Conclusion: Since a team member has experience with Node.js we can have a more seamless experience integrating this backend system. We also know that Node.js works well with MongoDB as well as our choice of front end frameworks so we know the integration is there and doable.

3.1.2 MongoDB

- **Description:** Database management system that stores data into documents.

- **Rationale:**
    - JSON-like documents meaning easy integration with node.js
    - Easily Scalable and maintained.
    - Easy to integrate
    - High Performance
    - Cross platform compatibility
- - **Qualitative Assessment**:
    - Strengths
        - Large community for support and troubleshooting.
        - Scalability
        - No sql, an easy learning curve
        - Horizontal Scaling
    - Weaknesses
        - Not the best for a lot of transactions
        - Heavy memory usage
        - No join operations like in SQL
- **Use Cases** (Source)
    - Uber
    - Lyft
    - Coinbase
    - Forbes

Conclusion: Since a team member has experience with MongoDB we can have a more seamless experience integrating this backend system. We also know that MongoDB works well with our choice of front end frameworks so we know the integration is there.

3.1.3 Ruby on rails

Description: Open source web application framework that uses the Ruby programming language, and aligns with the MVC architecture

- **Rationale:**
    - Opensource
    - Secure
    - Scalable
    - Rapid development
- **Qualitative Assessment**:
    - Strengths

        - RESTful routing
        - Integrated testing
        - Security
        - Mature ecosystem
    - Weaknesses
        - Learning curve, Ruby programming language
        - Performance
        - Monolithic
        - MySQL and PostgreSQL only
- **Use Cases** (Source)
    - Kickstarter
    - Shopify
    - Twitter
    - Github
    - Hulu

3.2 Frontend Frameworks

3.2.1 React.js

Description: A JavaScript library that promotes modularity by allowing the use of components that encapsulate part of the user interface.

- **Rationale:**
    - Modularity
    - Reusability
    - Modern
    - Responsive
    - Flexibility
    - OOP aspects
- **Qualitative Assessment:**
    - Strengths
        - Widely used.
        - A lot of documentation, tutorials.
        - Modular.
        - Reusability
        - Virtual DOM
        - JSX syntax very similar to HTML
    - Weaknesses
        - Learning curve

        - Community not as mature
        - Large Library
- **Use Cases** (Source)
    - Airbnb
    - Cloudflare
    - Dropbox
    - BBC
    - Facebook

Conclusion: This is the framework that we choose. A few members in the team have experience with Reactjs so we can achieve faster development time rather than being completely blind. We will also be learning Javascript/Typescript, HTML, and CSS which is not too much of a learning curve and this would be a great framework to include in our knowledge.

3.2.2 Angular

Description: A structural JavaScript framework that allows the creation of dynamic web applications

- **Rationale:**
    - Component based library
    - Very powerful CLI tool (*ng add @.../...* command)
    - Object oriented aspects
    - Single page application
- **Qualitative Assessment:**
    - Strengths
        - Widely used.
        - Dependency Injection
        - Modular.
        - Reusability
        - HTML Syntax
        - Dynamic
    - Weaknesses
        - Learning curve
        - Strictly TypeScript
        - Large Library
- **Use Cases** (Source)
    - USA Today
    - Gmail

3.2.3 Django Python

Description: High level python framework for specifically making web pages.

- **Rationale:**
  - Rapid Development
  - Fast
  - Secure
  - Scalable
- **Qualitative Assessment:**
  - Strengths
    - Scalable
    - Simple
    - Uses python
  - Weaknesses
    - Learning curve
    - Monolithic
    - Not the best for small projects
- **Use Cases** (Source)
  - Instagram
  - Spotify
  - Youtube

4. Integration and Interoperability

4.1 Backend-Frontend Integration

In our project, our strategy is to use the client server model. The backend, made with Node.js, will be handling all server side operations such as storing, updating and deleting data. The front end is the client side made with React.js that will interface with the server by passing data to it. The front end will prepare all the data prior to passing it towards the server for proper storing.

4.2 Third-Party Services

Material UI will be used as a third-party service to React.js. This will provide a large library of customizable React components. This will make our program easier to follow, more efficient to process, and allow us to have a cohesive modern design.

5. Security Considerations

Security measures for the frontend and backend would be user privacy. For the front end, we will have multiple layers of validation. We will first validate that all user input fields are the correct type and valid information for our database, for example start date, end date, and username would require validation before being passed to the database. Our second major validation system would be sanitizing all input fields to prevent any SQL injections and Cross Site scripting. For the backend, when storing users passwords, we will be implementing hashing and salting on the passwords in the case of a security breach.

6. Conclusion

For this project, it was determined that React.js will be the best choice for the front-end code due to its abundant use, extensive variety of capabilities, and flexibility. Node.js and MongoDB will be used for the back-end code due to their easy integration with each other, large community, varied ecosystem, and fast execution properties. The Agile methodologies will be used throughout this project as it best fits with the requirements outlined to complete the application. Some of the Agile principles used are interactions and individuals over tools, programing over documentation, customer collaboration over contracts, and the separation of the project into small, frequently updated phases.