FACULTY OF COMPUTER SCIENCE

Assignment 2

In

The Class of

CSCI 5709: ADVANCED TOPICS IN WEB DEVELOPMENT

by

Aditya Maheshbhai Purohit [B00952865]

Submitted to

Prof. Gabriella Mosquera

Department of Computer Science

Dalhousie University.

Date: 08th March 2024

1. Application Details – “ParkFlex”:

ParkFlex web application aims to provide a comprehensive online marketplace for parking spots, addressing the common urban challenge of finding convenient and affordable parking. It seeks to bridge the gap between parking spot owners who have underutilized space and drivers facing difficulties in locating parking, especially in densely populated areas. By leveraging technology, the platform aspires to optimize parking space usage, reduce urban congestion, and offer a convenient solution for both spot owners and seekers. [1]

The purpose of ParkFlex is to provide users with a unique and effective experience in booking parking spots around their cities. Also allows users with excess space to utilize it to earn extra money. ParkFlex provides users with functionalities to search for available parking spots, and connect with owners to know them better. [1]

2. Target User Insight:

The website ParkFlex targets mainly 2 categories of users as described below:

Parking Spot Owners: These users are individuals or businesses that have parking spaces available for rent. They could be homeowners with extra driveway space, commercial establishments with unused parking lots, or real estate developers with parking facilities. These owners use the platform to list their parking spots, set pricing and availability, and manage bookings. By utilizing the platform, owners can maximize the use of their parking spaces and generate additional income. They are expected to have basic computer skills like using a browser on a mobile or laptop, navigating a website and familiarity with online listing platforms. [2]

Parking Spot Seekers: This group comprises drivers in need of parking solutions. They could be commuters looking for daily parking, residents in urban areas seeking long-term parking, or visitors/tourists searching for convenient parking options. These users use the platform to search for available parking spots based on location, price, and availability. They can also book parking spots in advance, saving them time and hassle. Seekers are expected to have basic computer skills like using a browser on a mobile or laptop, navigating a website and being comfortable with using online booking platforms. [2]

The user scenarios and use cases for my feature “rating and reviews” are described below:

Task 1: View all ratings and reviews of a post.

User Scenario: Carla Jones, a parking spot seeker, wants to book a parking spot near her office in the downtown area for 6 months. She is interested in a parking spot post but wants to check previous experience with that spot to check for any problems like snow cleaning, parking spot size issues or owner behaviour. She thus checks the ratings and reviews of that post to get her doubts clear by reading past experiences. [2]

Persona: Parking Spot Seeker

Feature: Rating and Reviews (View all ratings and reviews of a post.)

Need: Choose the best parking spot by reading previous user experience.

Context: Get doubts clarified from other past experiences.

Use Case:

1. The user clicks on a specific parking spot post for which they want to check ratings and reviews. [User action]
2. The system redirects the user to the details page of the selected parking spot. [System action]
3. The user scrolls down to the "Ratings and Reviews" section and clicks on the "View all Reviews" button. [User action]
4. The system displays a summary of ratings and reviews for the selected parking spot. [System action]
5. The user chooses to sort the reviews by recent first. [User action]
6. The system sorts the reviews based on the latest submission date. [System action]
7. The user applies a filter to view reviews by rating (e.g., 5, 4, 3, 2, 1, or All). [User action]
8. The system filters the reviews as per the selected rating criteria. [System action]
9. The user reads through the reviews. [User action] [2]

Task 2: Add a rating and review for a particular post.

User Scenario: Carla Jones, a parking spot seeker, had a bad experience for 1-month with a parking spot where she had to clean snow herself instead of the owner. However, before the

booking process, the owner mentioned in the post about snow cleaning from the owner's end. Carla then used the Add review option on that post to share her experience with others, so that they can be careful before booking this spot. [2]

Persona: Parking Spot Seeker

Feature: Rating and Reviews (Add a rating and review.)

Need: To share her experience with a parking spot.

Context: Post issues with a parking spot to make other users aware.

Use Case:

1. The user goes to the details page of a specific parking spot where they have recently parked. [User action]
2. The system displays information about the selected parking spot. [System action]
3. The user scrolls down to the "Ratings and Reviews" section and clicks on the "View all Reviews" button. [User action]
4. The system displays a summary of ratings and reviews for the selected parking spot along with a button "Add your review". [System action]
5. The system presents a form for the user to input their rating (on a scale of 1 to 5 stars) and write a detailed review about their parking experience. [System action]
6. The user rates the parking spot by selecting a number of stars, entering their review and clicking the "Submit" button. [User action]

   6.1 The system shows an error if rating or review is not filled. [System action]

   6.2 The user fills the empty field and presses "Submit" again. [User action]
7. The system validates the input, ensuring all required fields are filled, and then adds the new rating and review to the existing list. [System action]
8. The user can now view their rating and review on the overall ratings and reviews page for that parking spot. [User action] [2]

3. User-Centered Design Approach:

I have chosen the feature "Ratings and Reviews" for this project and it has two tasks:

Task 1: View all ratings and reviews:

Figure 1 shows the web page design of the reviews page which displays ratings and reviews of different users who booked this spot. The parking spot seeker may read through these reviews before proceeding to booking, to get more clarity regarding a particular parking spot. They can also sort and filter the reviews if needed. Moreover, the average rating of a parking spot and the owner, both are visible.

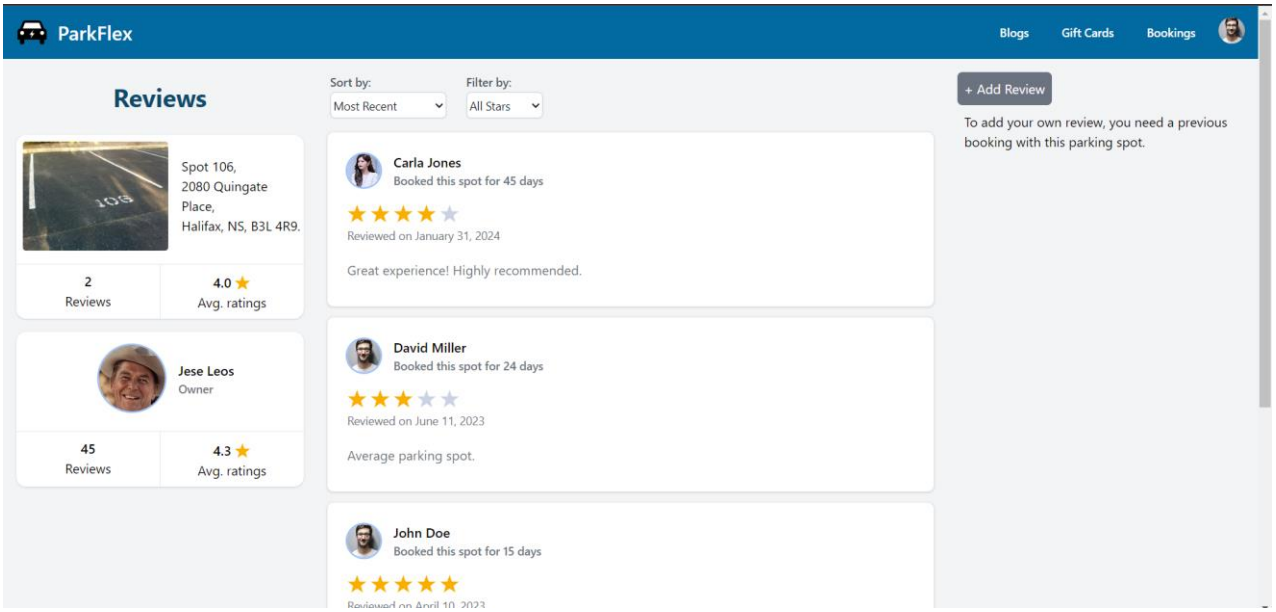

Figure 1: Webpage design of reviews page [3].

Task 2: Add a rating and review:

A low-fidelity prototype is shown in Figure 2 for adding a rating and review. On the review page, a parking spot seeker can add their rating and review once they have completed a booking for that spot. After clicking the add review button, the user will see a pop-up window which will ask them

for a rating out of 5 stars and a written review. They can submit their review once the form is filled and then it will be visible on the ratings and reviews page.

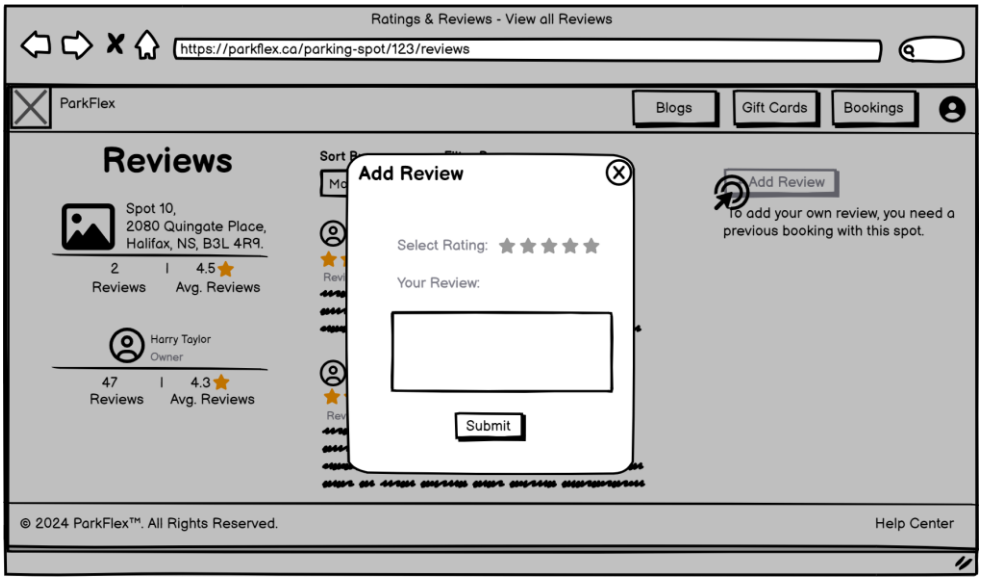

Figure 2: Wireframe of add rating and review task [2].

4. Application Architecture/Workflow:

Figure 3 shows the overall architecture of the ParkFlex web application, which shows the Model-View-Controller design pattern that we will be using. Firstly, the browser will load the react application, which will then start sending requests to specific endpoints defined in the Node.js and Express.js backend. This request is then routed to the appropriate controller, which interacts with the MongoDB via model to retrieve or update data in the database. The controller processes the data according to business logic and sends a response back to the React client, which will be rendered in the browser.

Frontend: React remains our frontend framework for rendering views along with Tailwind CSS for creating responsive components.

Backend: Node.js and Express.js will be used to create APIs and manipulate the database using the model.

Database: MongoDB at the database layer will be used to manage the data. This remains our choice due to its scalability and seamless JavaScript integration [4].

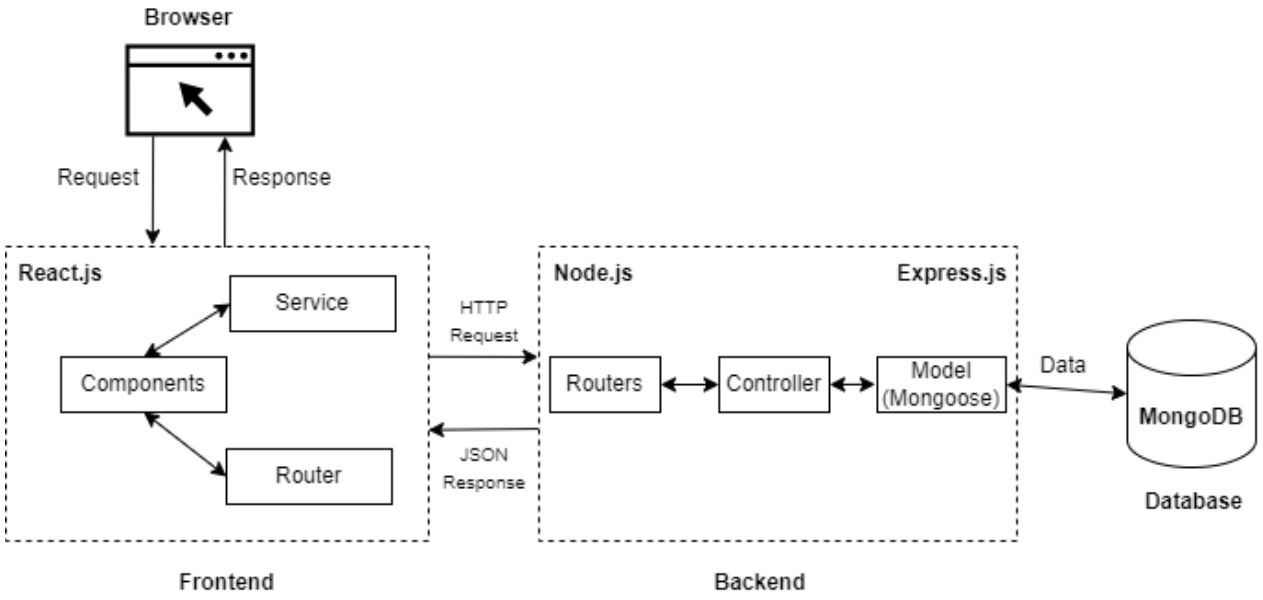

Figure 3: Application Architecture of ParkFlex [5] [6].

5. Interaction Design:

Below are the two tasks of the ratings and review feature that I am working on:

Task 1: View all ratings and reviews:

Figure 4 illustrates the task flow diagram of viewing all the ratings and reviews. The user needs to go to the particular parking spot details page and then click on view all reviews button and then access the reviews and ratings of a parking spot. Moreover, users can also sort and filter the reviews.

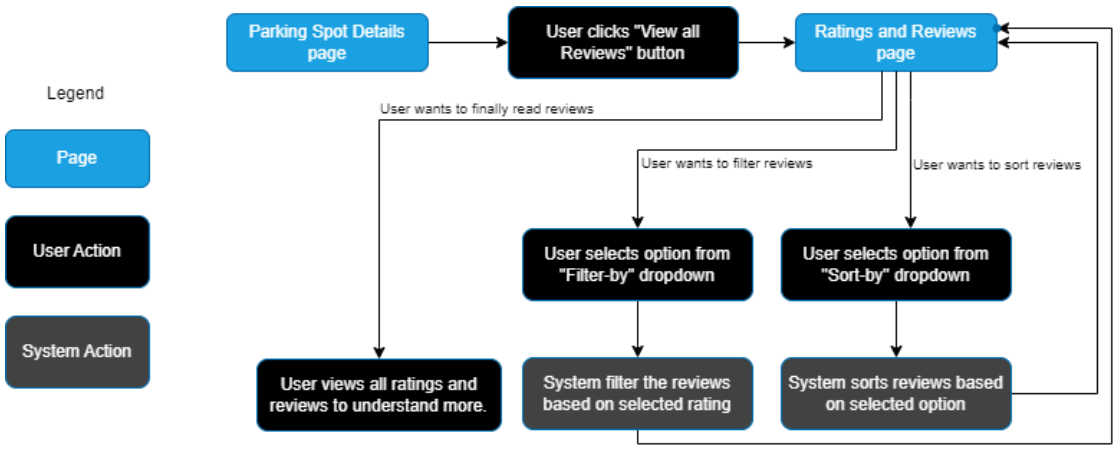

Figure 4: Task flow diagram for view all ratings and reviews task [2].

Figure 5 shows the click streams diagram of viewing all the ratings and reviews starting from the homepage. The user needs to browse all spots, select a particular spot and then they can view all the reviews of that parking spot post.

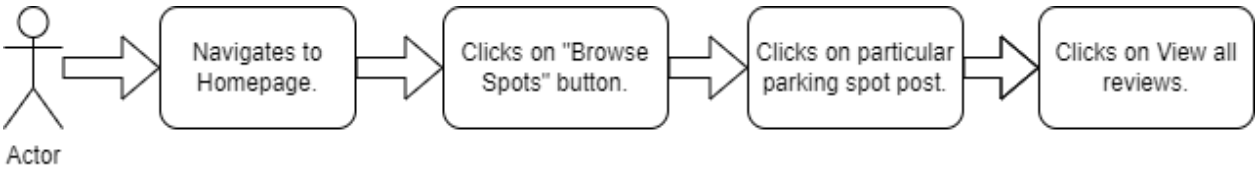

Figure 5: Click streams diagram for view all ratings and reviews task [5].

Task 2: Add a rating and review:

Figure 6 illustrates the task flow diagram of adding a rating and review. The user needs to go to the reviews page, similar to task 1 and if they had any previous booking for that parking spot then they can add their rating and review.

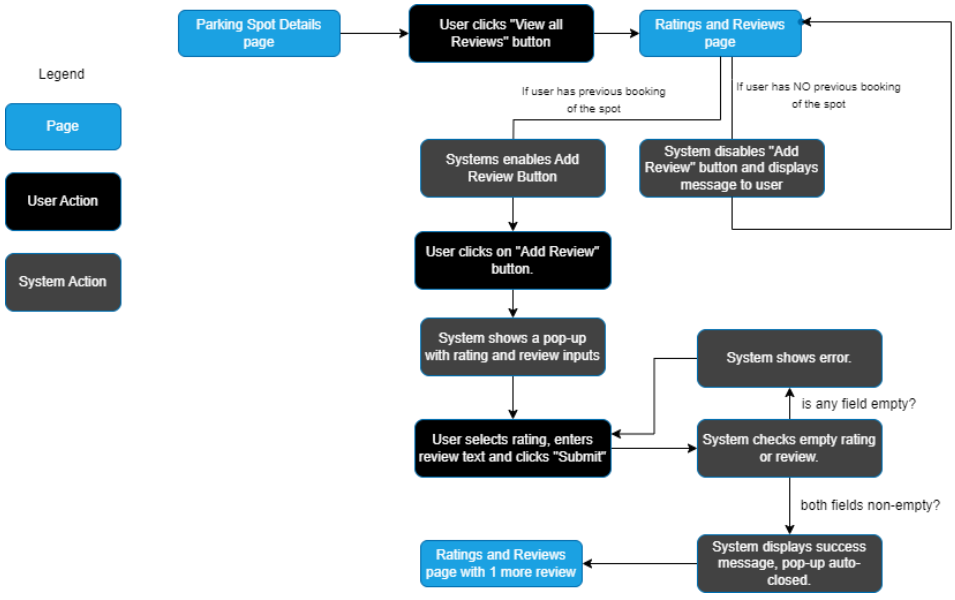

Figure 6: Task flow diagram for add a rating and review task [2].

To add a new rating and review, a click streams diagram is also shown in Figure 7. The user needs to make an extra click, as compared to task 1, on the add review button.

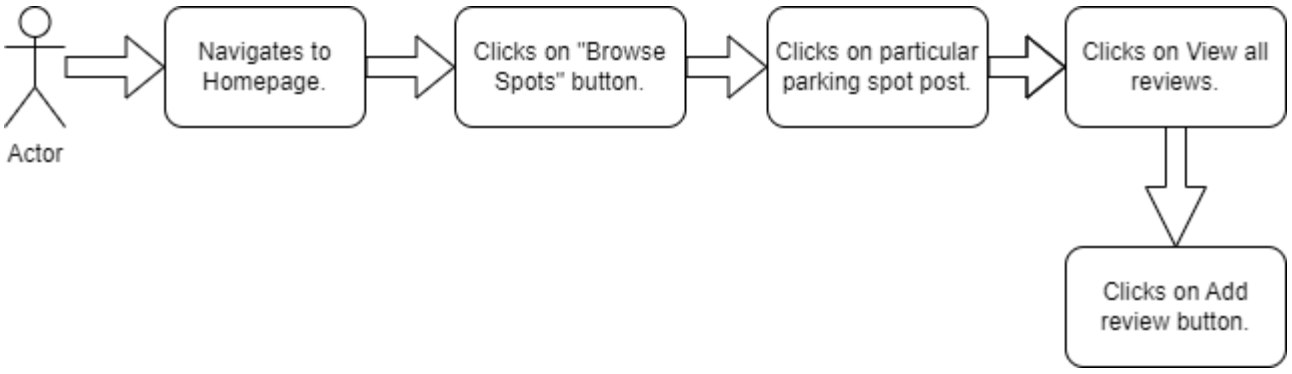

Figure 7: Click streams diagram for add a rating and review task [5].

6. Process and Service Workflow:

The process and service workflow of the ParkFlex is shown in Figure 8. The flow starts from the user-end when they want to view the website. Then, the react components are rendered and viewed in the browser. Once a user makes an interaction, react requests to the Node.js backend which then applies the business logic and updates MongoDB accordingly. Lastly, as per the response from the backend, the react page gets updated and that becomes visible to the user.

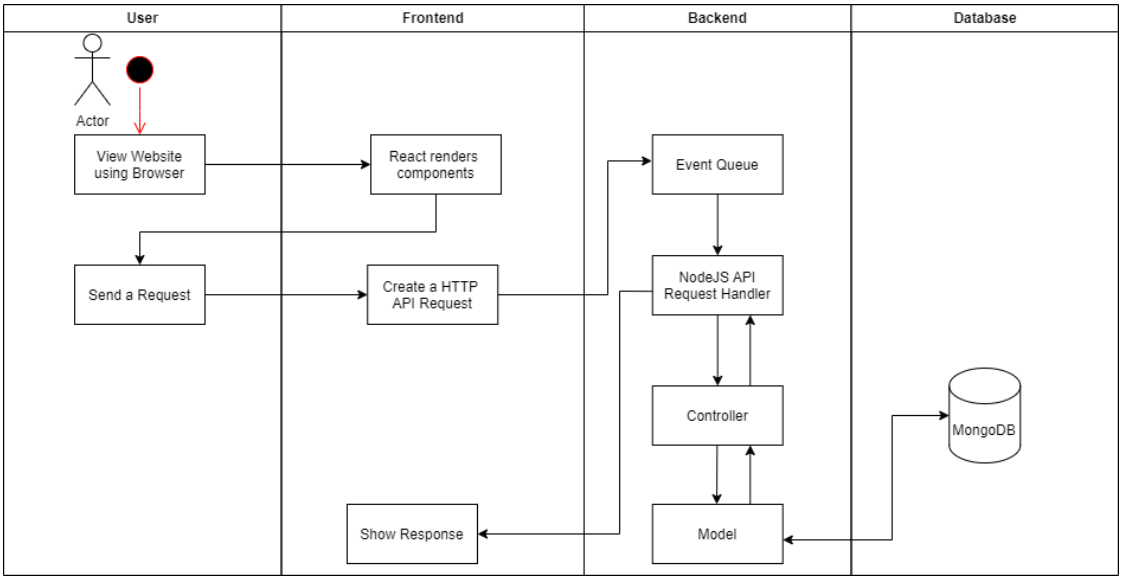

Figure 8: Process and Service workflow of ParkFlex [5].

7. Folder Structure:

Frontend (React.js) and Backend (Node.js) folder structures are different and are described here as follows:

Frontend-related files are arranged in the folder structure as shown in Figure 9. At the top level, there is a src folder and files like package.json, package-lock.json, tailwind.config.js and tsconfig.js. The src folder contains the assets folder which has images, components and feature-specific page folders. The package.json and package-lock.json contain the dependencies for the project. The tailwind.config.js and tsconfig.js files contain the Tailwind CSS and typescript configuration respectively.

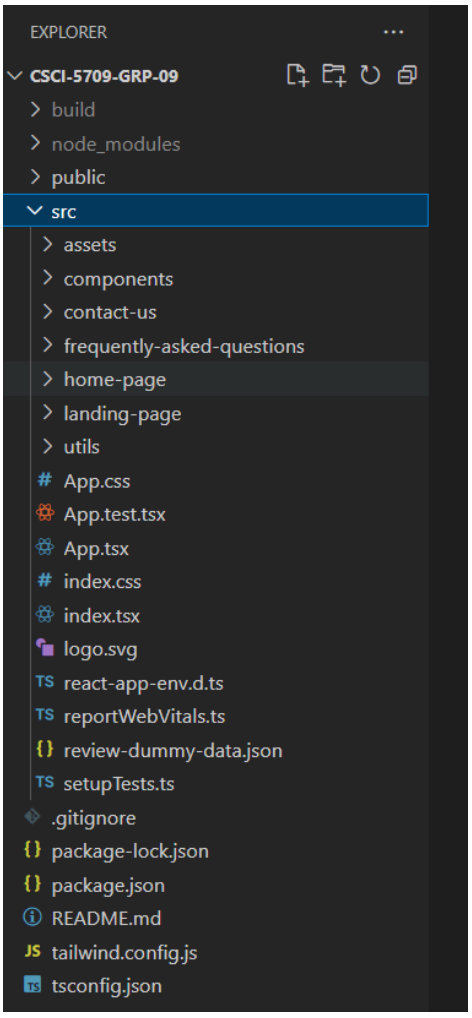

Figure 9: React folder structure [7].

Backend-related files will be arranged in the folder structure as shown in Figure 10. At the top level, there is a src folder which contains the source code in sub-folders routes, controllers, models and views. The node_modules folder contains the installed dependencies. These dependencies are specified by package.json and package-lock.json. The routes folder contains all the routes, the container folder contains business logic and the model folder contains schemas of the database.

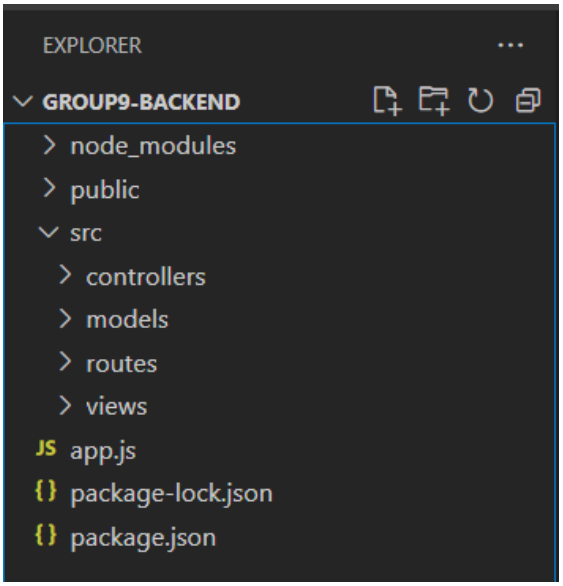

Figure 10: Node.js Project Structure [7] [8].

8. References:

[1] A. Purohit, J. Rana, K. Patel, N. Patel, M. Patel, S. Patel, Group 9 “CSCI5709 Group Proposal” Winter Term, Dalhousie University, [online document], 2024. [Accessed: 08-Mar-2024].

[2] A. Purohit, J. Rana, K. Patel, N. Patel, M. Patel, S. Patel, Group 9 “CSCI5709 Assignment-1” Winter Term, Dalhousie University, [online document], 2024. [Accessed: 08-Mar -2024].

[3] “React app,” *Netlify.app*. [Online]. Available: https://aditya-purohit-csci-5709-web-a1.netlify.app/. [Accessed: 08-Mar-2024].

[4] A. Purohit, J. Rana, K. Patel, N. Patel, M. Patel, S. Patel, Group 9 “CSCI5709 Tutorial-1” Winter Term, Dalhousie University, [online document], 2024. [Accessed: 08-Mar-2024].

[5] “Flowchart maker & online diagram software,” *Diagrams.net*. [Online]. Available: https://app.diagrams.net/. [Accessed: 09-Mar-2024].

[6] S. Yeshwanthini, “Illustration about MERN stack,” *Techiepedia*, 21-Mar-2021. [Online]. Available: https://medium.com/techiepedia/what-exactly-a-mern-stack-is-60c304bffbe4. [Accessed: 09-Mar-2024].

[7] “Visual Studio Code - code editing. Redefined,” *Visualstudio.com*. [Online]. Available: https://code.visualstudio.com/. [Accessed: 09-Mar-2024].

[8] P. prathamgfg, “Folder structure for a Node JS project,” *GeeksforGeeks*, 28-Nov-2023. [Online]. Available: https://www.geeksforgeeks.org/folder-structure-for-a-node-js-project/. [Accessed: 09-Mar-2024].