



DALHOUSIE
UNIVERSITY

FACULTY OF COMPUTER SCIENCE

Assignment 2

In

The Class of

CSCI 5709: ADVANCED TOPICS IN WEB DEVELOPMENT

by

Jay Navinbhai Rana [B00932024]

Submitted to

Prof. Gabriella Mosquera Department of
Computer Science
Dalhousie University.

Contents

1. Application Details:.....	3
2. Application Feature	3
3. Target User Insight:	3
4. User-Centered Design Approach:	4
4. Application Architecture	7
5. Interaction Design	8
1. Task Flow Diagram and Click Stream diagram	8
2. Use Cases	14
6. Process Workflow	16
7. Folder Structure.....	18
References:	21

1. Application Details:

ParkFlex is a dynamic and innovative platform designed to streamline the process of parking by connecting parking spot owners with drivers seeking spaces. Its goal is to optimize parking spot utilization, ease the hassle of finding parking in busy areas, and provide an additional income stream for space owners. The application encompasses features such as real-time availability checks, a secure payment gateway, and a user-friendly interface that caters to both spot owners and seekers [5].

2. Application Feature

Feature: Authentication

This feature is fundamental to the security and functionality of the application as it governs how users access and interact with the service. The tasks included within this feature encompass user registration, user login, password recovery, user logout, and profile management.

- **User Registration:** Allows new users to create an account by providing essential details and setting up a secure password [5].
- **User Login:** Enables existing users to access their accounts by entering their credentials [5].
- **Password Recovery:** Offers a mechanism for users to reset their passwords if forgotten, ensuring they can regain access to their accounts [5].
- **User Logout:** Provides users the ability to securely exit their accounts, ensuring no unauthorized access on shared or public devices [5].
- **Profile Management:** Permits users to view and edit their profile information, maintaining the latest and most accurate user data [5].

By developing these tasks, ParkFlex ensures that the system is accessible and secure for all user interactions related to finding, listing, and managing parking spaces [5].

3. Target User Insight:

User Personas and Scenarios:

Carla Jones (Parking Spot Seeker): A time-constrained professional who prioritizes convenience and proximity when looking for parking. She seeks an application that allows her to find and book parking spots without the need to physically check availability. Carla represents users who value efficiency and security in their parking choices, requiring a reliable platform that offers real-time updates and seamless payment transactions [6].

Harry Taylor (Parking Spot Owner): An entrepreneurial individual who wants to generate passive income from his unused parking plots. He desires a platform where he can easily list and manage his parking spots, engage with potential renters, and be notified of any updates or booking activities. Harry represents parking spot owners who aim to maximize their earnings with minimal hassle and time investment [6].

Application Usage Assumptions:

For Seekers like Carla: The assumption is that they will use ParkFlex due to its efficient search and booking system, ability to check spot availability in real-time, and for the security in transactions the platform provides [6].

For Owners like Harry: They are assumed to be drawn to the platform for its ease of listing management, the potential to reach a wide customer base, and its reliable payment and notification systems [6].

4. User-Centered Design Approach:

1. Login Page

Below image showcases the low fidelity prototype of the Login Page for ParkFlex, designed to facilitate a seamless and secure entry for users seeking parking solutions. The page layout is intuitive, offering clear fields for email and password entry, and a prominent login button for quick access. A convenient 'Forgot Password??' link is available for those needing to reset their credentials, emphasizing user accessibility and account recovery. For new users eager to join ParkFlex, a 'Sign Up' link invites them to create an account, ensuring the platform is welcoming and navigable for all users [6].

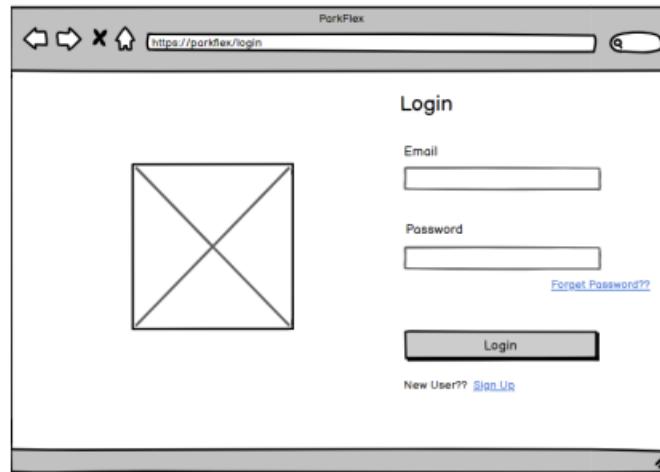


Fig 1. Wireframe of Login Page [2]

2. Registration Page:

The low fidelity wireframe illustrates the registration page for ParkFlex, thoughtfully laid out to guide new users through the process of creating their account. This page presents a series of clearly marked fields where users can input their first name, last name, email, and password, including a confirmation for the password to ensure accuracy [6]. The design is intentionally straightforward, allowing for quick and effortless navigation while emphasizing the importance of information security. A notable feature is the prominent "JOIN PARKFLEX" button, which invites users to complete their registration and join the ParkFlex community. For those who already have an account, a convenient "Login" link is provided, reinforcing the ease of switching between the registration and login processes [6].

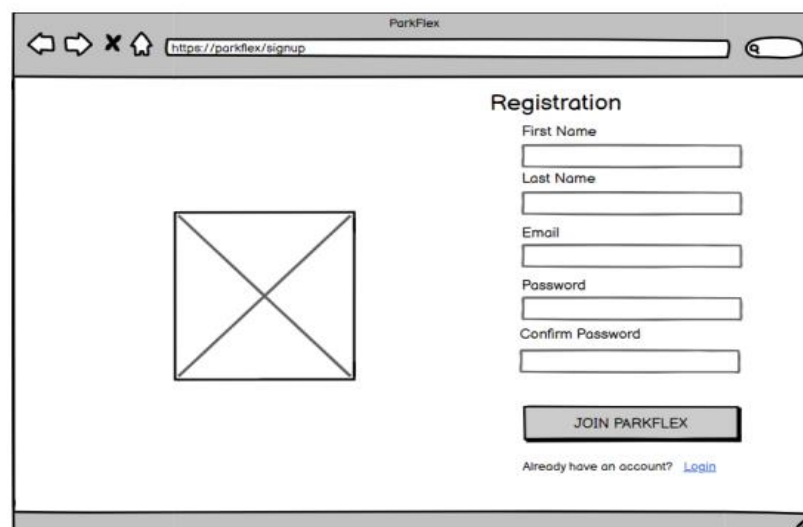
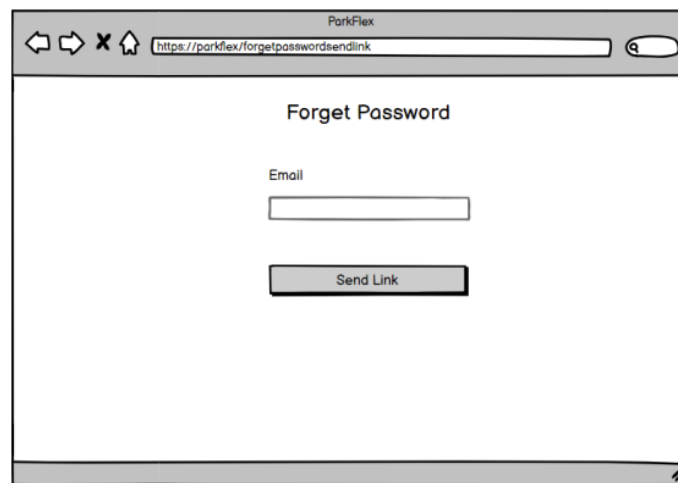


Fig 2. Wireframe of Registration Page [2]

3. Forget Password Page:

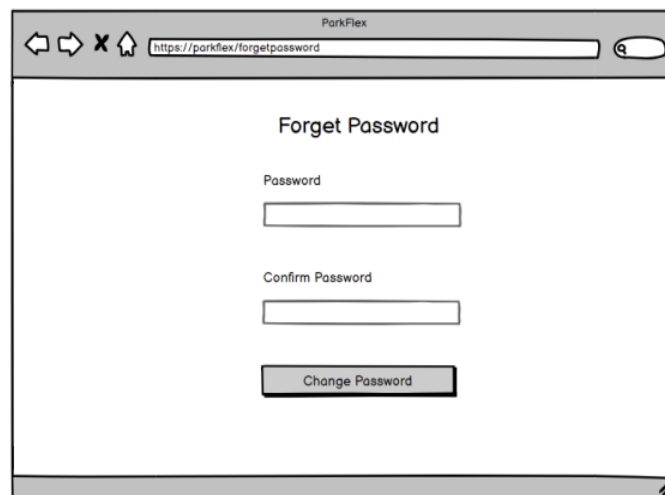
The first image displays a crucial recovery feature, allowing users who have forgotten their password to easily request a reset link. The minimalist design, focusing on a single Email input field and a 'Send Link' button, underscores the application's commitment to user convenience and accessibility [6].

The second image continues the password recovery journey, providing a secure, two-step form where users can set a new password with confidence. The clean layout, featuring Password and Confirm Password fields, alongside a 'Change Password' button, demonstrates ParkFlex's dedication to offering a seamless and secure user experience, resonating with the immediate needs and expectations of the user base [6].



A wireframe of a web browser window titled "ParkFlex". The address bar shows the URL "https://parkflex/forgetpasswordsendlink". The main content area is titled "Forget Password". Below the title is a single input field labeled "Email". Below the input field is a button labeled "Send Link".

Fig 3. Wireframe of Forget Password Page 1 [2]



A wireframe of a web browser window titled "ParkFlex". The address bar shows the URL "https://parkflex/forgetpassword". The main content area is titled "Forget Password". Below the title are two input fields: the first is labeled "Password" and the second is labeled "Confirm Password". Below the input fields is a button labeled "Change Password".

Fig 4. Wireframe of Forget Password Page 2 [2]

4. Application Architecture

This architecture diagram describes the flow of data and requests for the ParkFlex application:

- The Browser represents the user interface where ParkFlex users interact with the application.
- When an action is taken (such as logging in or registering), the Browser sends an HTTP request to the ReactJS front end.
- ReactJS receives the request and uses the Router to direct it to the correct Component. These components handle the request, possibly involving state management or further logic, and generate a response.
- If the action requires backend processing (like checking credentials), ReactJS sends an HTTP request to the NodeJS backend.
- Within NodeJS, Routers determine the route logic for the request, passing it to the correct Controller.
- The Controller processes the request, using the Model to interact with the MongoDB database as needed, such as retrieving user information or saving new user data.
- After the Model performs the necessary operations on the data in MongoDB, it sends the result back to the Controller, which then passes it back through the Router to ReactJS.

Finally, ReactJS updates the Component, and the Browser displays the results to the user.

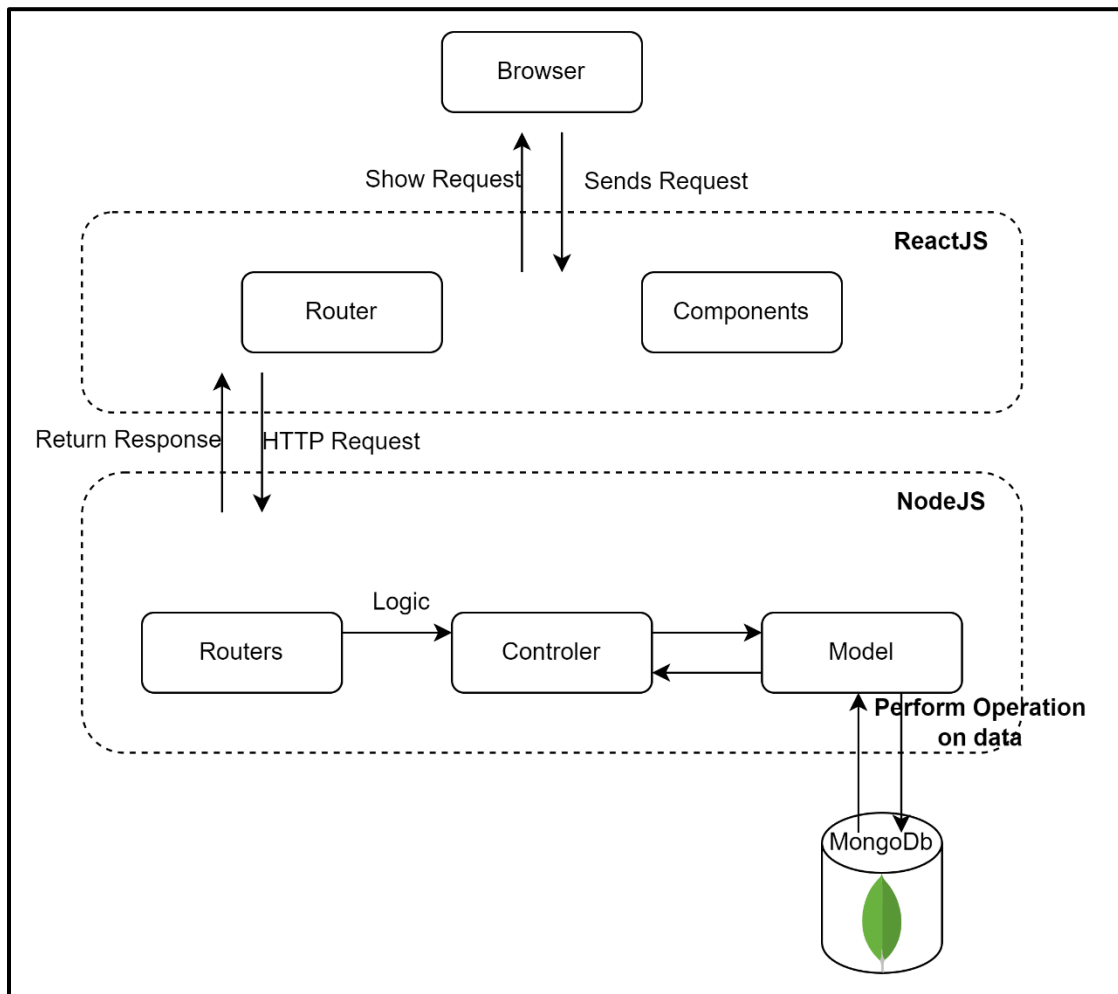


Fig 5. Application Architecture [4]

5. Interaction Design

1. Task Flow Diagram and Click Stream diagram

Task 1: Login

The figure illustrates the task flow diagram for the user login process in the ParkFlex application. Starting with the user initiating the login process, the system displays the login page where the user is prompted to enter their email and password. If any of the fields are left empty, the system prompts the user to enter the missing details. Upon entering the required information, the system checks the credentials; if they are valid, the system grants access and redirects the user to the home page. If the

credentials are incorrect, the flow circles back, indicating a system error that prompts the user to re-enter their details. This diagram effectively communicates the sequence of user and system actions leading to successful or unsuccessful login attempts [6].

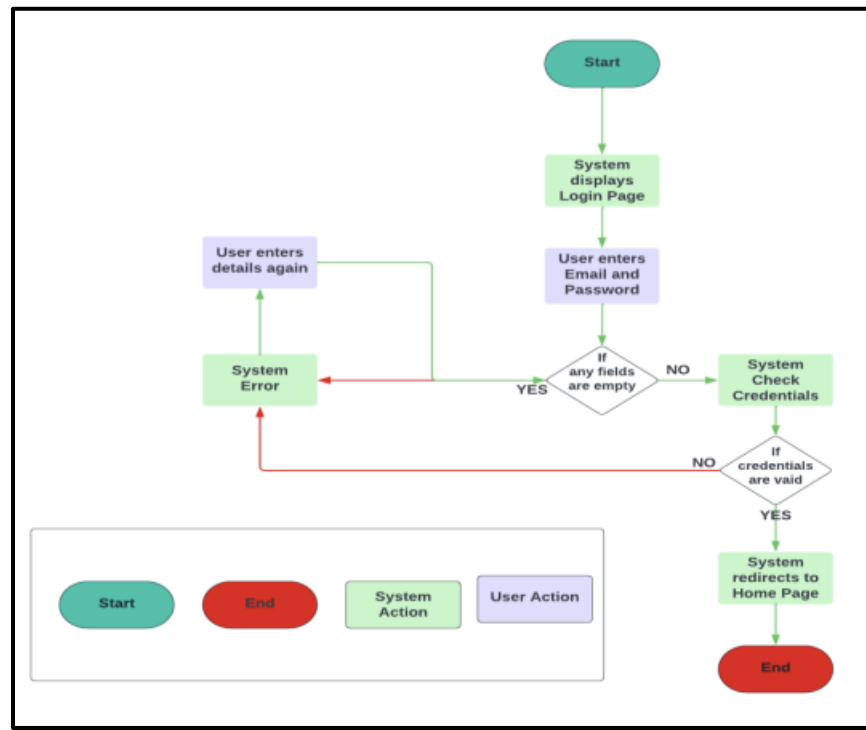


Fig 6. Task Flow diagram of Login [3] [1]

The figure provides a clickstream diagram detailing the steps a user follows to log in to the ParkFlex application. The process initiates with the user arriving at the home page. From there, the user clicks on the login option, which leads them to the login page. Once on the login page, the user is required to fill in the necessary details, typically an email and password. After entering their information, the user must click the login button to proceed. This diagram succinctly maps out the user's navigation path during the login process, from start to finish, emphasizing the simplicity and user-friendly nature of accessing their ParkFlex account.

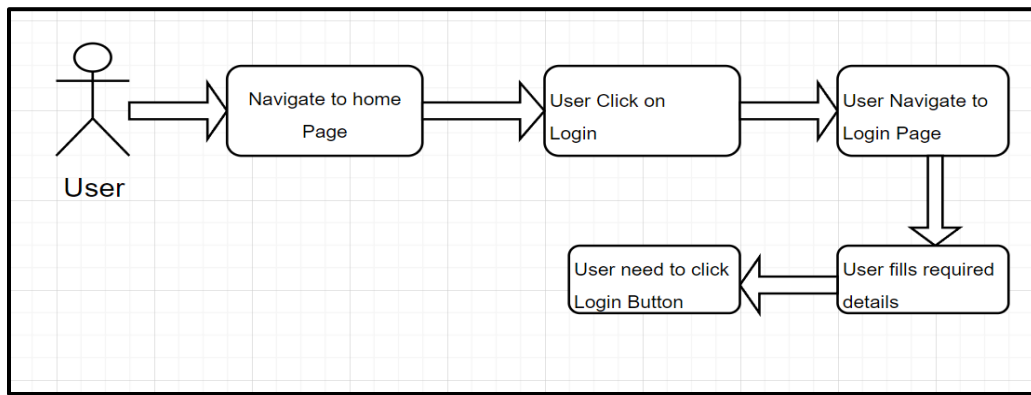


Fig 7. Click Stream diagram of Login [4]

Task 2: Registration

The figure provides a task flow diagram for the registration process on the ParkFlex platform. It begins with the system displaying the registration page, where a new user is prompted to enter their details, including email and password. If any required fields are left empty, the flow loops back, requesting the user to complete the missing information [6].

The process includes validation steps where the system checks if the email is already registered and whether the password requirements are fulfilled, including a matching confirmation password. If the email is already in use, or if the password does not meet the criteria or does not match, the user is directed to correct the information [6].

Upon successful validation, where the email is unique and the password is suitable, the system confirms the registration with a message, "Registered Successfully," and then redirects the user to the login page, signifying the completion of the registration process and the start of the user's journey with ParkFlex. This end-to-end flow ensures a user-friendly registration process, encouraging new users to join with confidence [6].

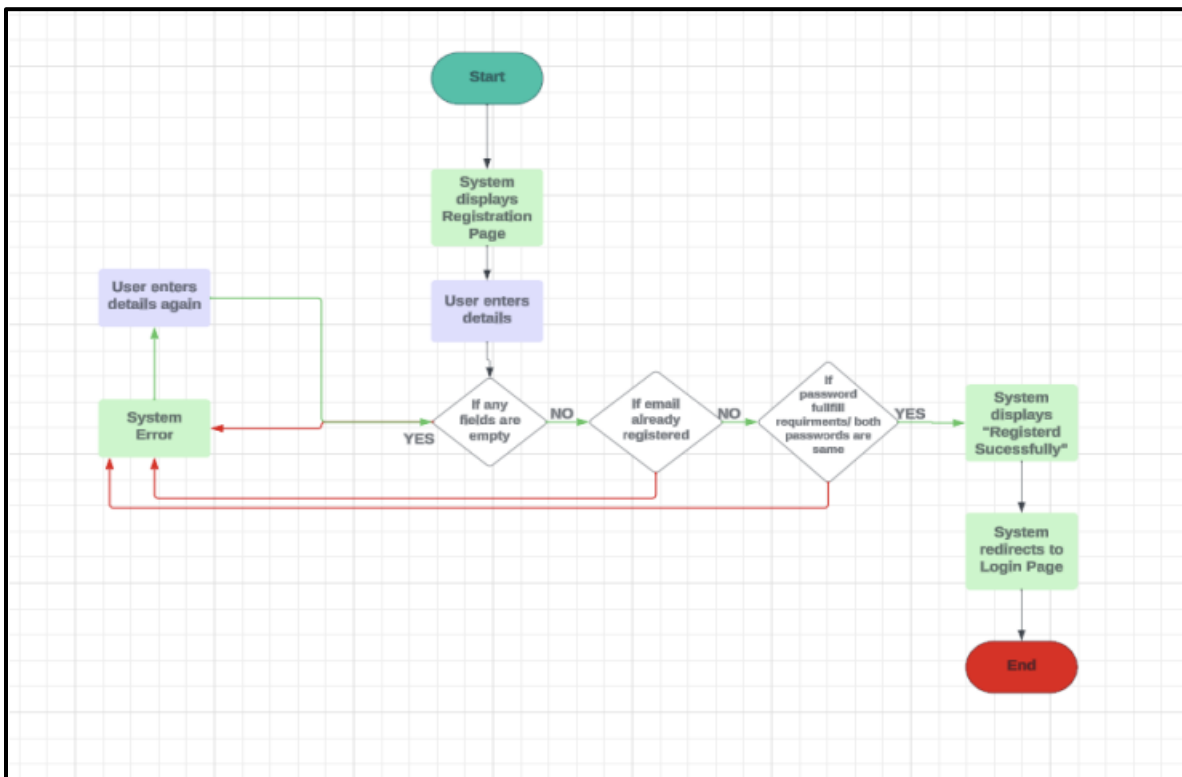


Fig 8. Task Flow diagram of Registration [3][1][6]

The clickstream diagram presents the process a user follows to register for an account on the ParkFlex application. It begins with the user navigating to the login page. There, the user finds and clicks on the "Sign in" button, which leads them to the registration page. This decision point acknowledges that users may arrive at the login page first but need an option to register as new users. Upon reaching the registration page, the user is prompted to fill in the required details for creating a new account. After the user fills out this information, they must click the "Register" button to complete the process. This diagram effectively captures the user's journey from the initial point of contact to successful registration, highlighting key interaction points on the user interface.

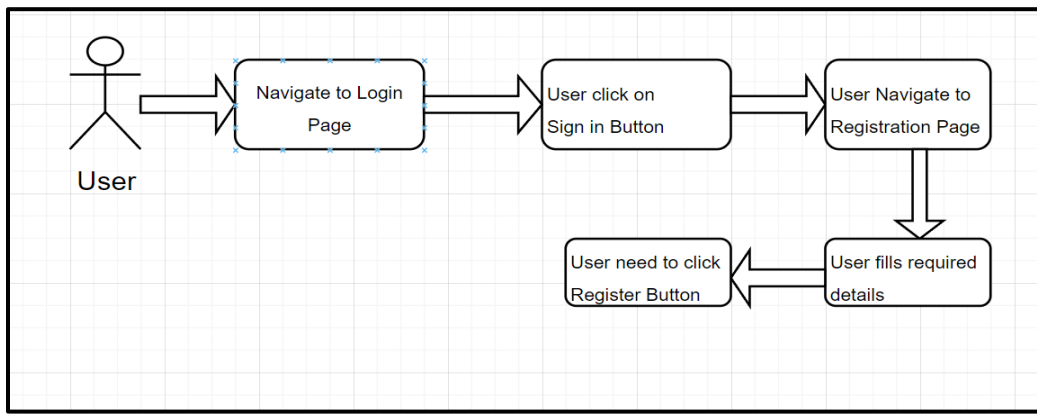


Fig 9. Click Stream diagram of Registration [4]

Task 3: Forget Password

The below task flow diagram describes the password recovery process for users who have forgotten their password on the ParkFlex application. The process starts from the login page, where the user clicks on the 'Forgot Password' option. The user is then prompted to enter their email address. If the field is left empty or the entered email is in an invalid format, a system error prompts the user to enter the email again. If the email is correctly formatted, the system checks if the email is registered. If not, the user is informed that the email isn't associated with an account. If the email is registered, the system sends a password reset link to the provided email. The user is then expected to check their email, click on the password reset link, which redirects them to the password reset page. On this page, the user enters a new password, confirms it, and submits the form. If the process is successful, the system updates the password and redirects the user to the login page, indicating the successful completion of the password reset process [5].

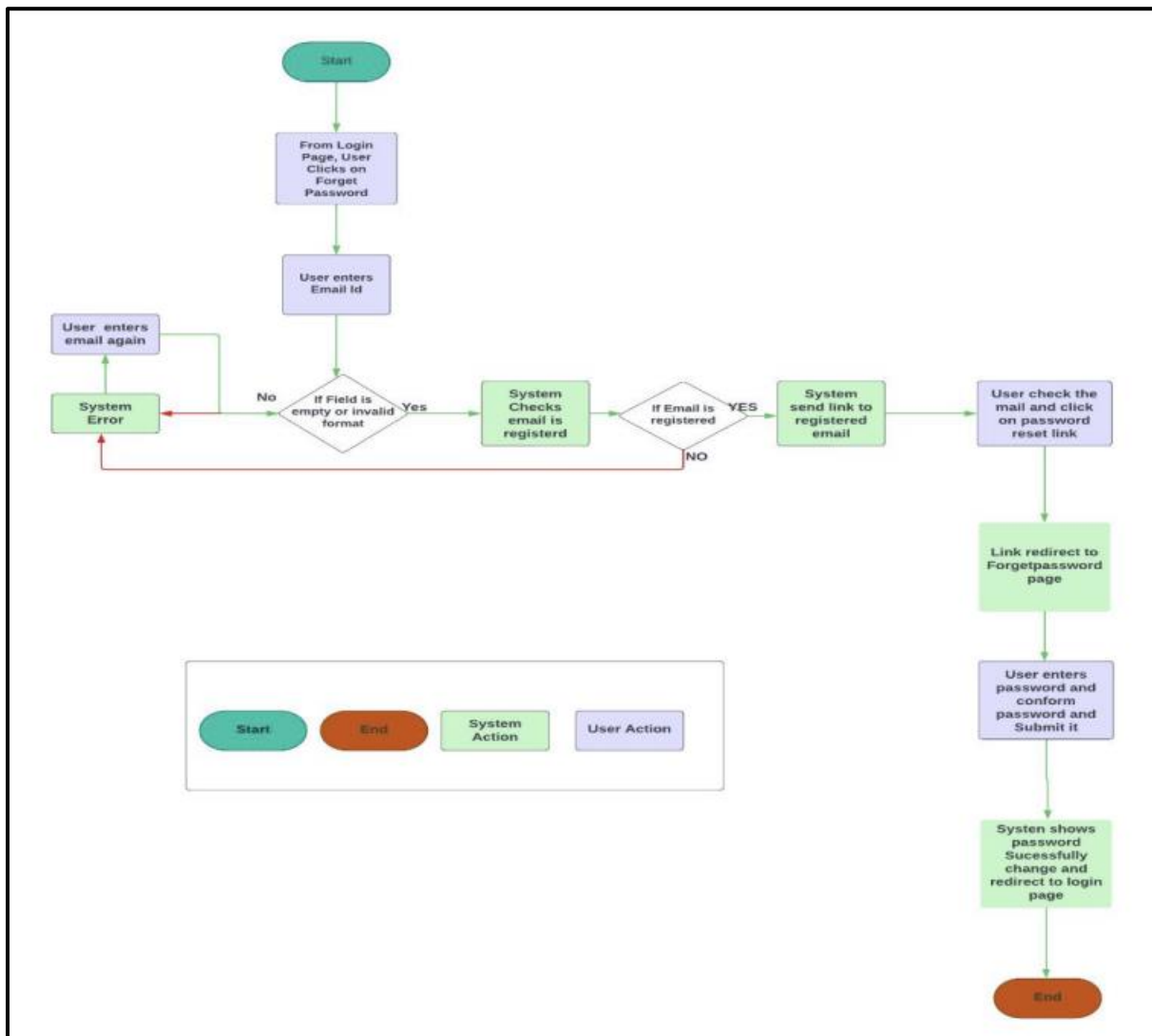


Fig 10. Task Flow diagram of ForgetPassword [1] [6]

The clickstream diagram outlines the user journey for resetting a forgotten password on the ParkFlex platform. The user starts at the Login page and clicks on the 'Forget Password' link. This action takes the user to the 'Forget Password' page, where they are prompted to enter their email address associated with their account. After submitting their email, they receive an email with a 'forget password link', which when clicked, navigates them to the 'Reset Password' page. Here, the user can enter and submit a new password, thus completing the password reset process. This clickstream is a visual representation of the steps a user takes within the ParkFlex interface to successfully reset their password.

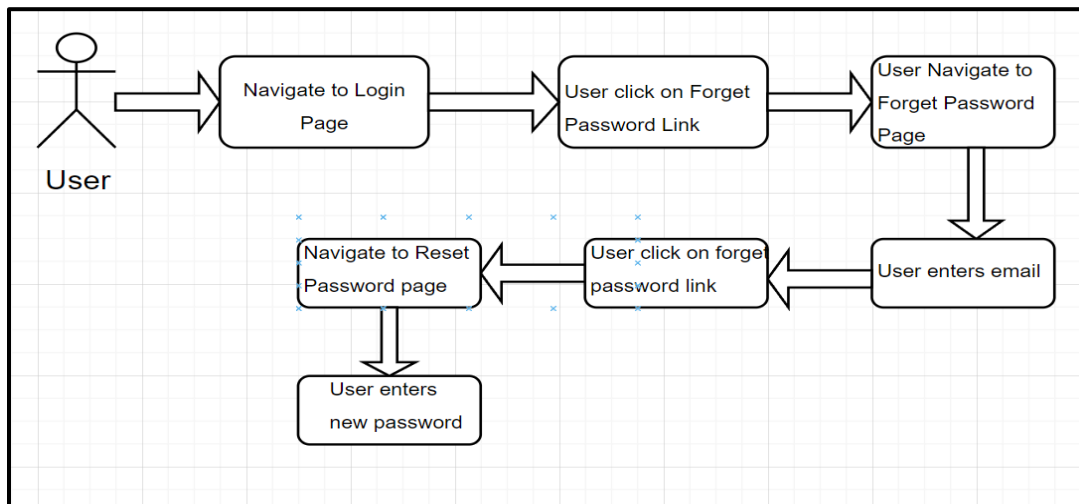


Fig 11. Click Stream diagram of ForgetPassword [4]

2. Use Cases

Task 1: Login

1. User visits the 'Park Flex' homepage [user action] [6].
2. User clicks on 'Login' button [user action] [6].
3. System displays the login page, requesting user's email and password [system action] [6].
4. User enters their registered email and password [user action] [6].
5. User clicks on 'Login' button [user action] [6].
 - 5.1 System checks the credentials [system action] [6].
 - 5.1.1 System displays a message if the email has not been registered [system action] [6]
 - 5.1.2 User chooses to sign up or enters a different email address to try again [user action] [6].
 - 5.1.3 User clicks on 'Login' button again [user action] [6].
 - 5.2 System checks if the password matches the registered account [system action] [6].
 - 5.2.1 System displays a message if the password is incorrect [system action] [6].
 - 5.2.2 User enters the correct password [user action] [6].
 - 5.2.3 User clicks on 'Login' button again [user action] [6].

5.2.4 User clicks on 'Forgot Password' link if they cannot remember the password [user action] [6].

6. System validates the credentials and grants access to the user account [system action] [6].

7. User is redirected to their home screen. [system action] [6].

Task 2: Registration

1. User visits the 'Find Your Spot' homepage [user action] [6].

2. User clicks on 'Sign Up' button [user action] [6].

3. System displays the registration page, requesting user's details such as First Name, Last Name, Email and Password. [system action] [6].

4. User enters their preferred details. [user action] [6].

5. User clicks on 'Create Account' button [user action] [6].

5.1 System checks if the email has already been registered [system action] [6].

5.1.1 System displays a message letting the user know their email has already been used [system action] [6].

5.1.2 User enters a new email address to register [user action] [6].

5.1.3 User clicks on 'Create Account' button again [user action] [6].

5.2 System checks if the password meets the minimum-security requirements [system action] [6].

5.2.1 System displays a message letting the user know their password does not meet the requirements [system action] [6].

5.2.2 User enters a new password that meets the requirements [user action] [6].

5.2.3 User clicks on 'Create Account' button again [user action] [6].

6. System validates the registration information and creates the account [system action] [6].

7. System validates the registration information and creates the account [system action] [6].

8. User is redirected to the 'Login' page [system action] [6].

Task 3: Forget Password

1. User visits the 'ParkFlex' homepage [user action] [6].
2. User clicks on 'Login' button [user action] [6].
3. System displays the login page, requesting user's email and password [system action] [6].
4. User clicks on 'Forgot Password' link [user action] [6].
5. System displays the password recovery page, requesting user's email address [system action] [6].
6. User enters their registered email address [user action] [6].
7. User clicks on 'Submit' or 'Send Reset Link' button [user action] [6].
 - 7.1 System checks if the email is registered [system action] [6].
 - 7.1.1 System displays a message if the email is not recognized [system action] [6].
 - 7.1.2 User re-enters the email address or chooses to register [user action] [6].
8. System sends a password reset link to the user's registered email [system action] [6].
9. User checks their email and clicks on the password reset link [user action] [6].
10. System displays the password reset page, requesting a new password [system action] [6].
11. User enters a new password that meets the security requirements [user action] [6].
12. User clicks on 'Reset Password' button [user action] [6].
 - 12.1 System validates the new password and updates the user's account with the new password [system action] [6].
13. System displays a message confirming the password has been reset [system action] [6]
14. User is redirected to the 'Login' page to access their account with the new password [system action] [6].

6. Process Workflow

The process workflow diagram for the ParkFlex application illustrates how the different layers of the application architecture interact with each other from the user's actions to the backend processing and data storage.

- The user starts by accessing the ParkFlex website, interacting with the user interface that's built with Bootstrap and Tailwind for styling.
- On the frontend, when the user needs to submit information, they fill out forms that are validated using React Hook Forms to ensure data integrity before the information is sent.
- After form validation, the data or request is submitted through the frontend, which then sends an API call via Axios, a promise-based HTTP client for the browser and Node.js.
- React components are dynamically displayed and Zustand is used for simple, yet powerful state management across these components. React Query is utilized for managing the data fetching state, maintaining the application's performance, and reducing unnecessary re-renders.
- When the request reaches the backend, Node.js with Express.js framework receives the API request. If there is a need for queuing due to high traffic or other reasons, the request is temporarily stored in a queue.
- The request is then routed to the appropriate controller, which contains the logic to handle the request. The controller communicates with the model to interact with the MongoDB database, performing operations like CRUD (Create, Read, Update, Delete).
- Once the database operation is complete, MongoDB sends the data back through the model to the controller, which then sends a response back to the frontend.
- Finally, the frontend processes the response and the results are shown to the user. If there are any notifications to be displayed, React Toasts are used to show these messages to the user in an unobtrusive manner.

This diagram emphasizes the clear separation of concerns within the ParkFlex application, ensuring modularity and maintainability, while providing a responsive and user-friendly experience.

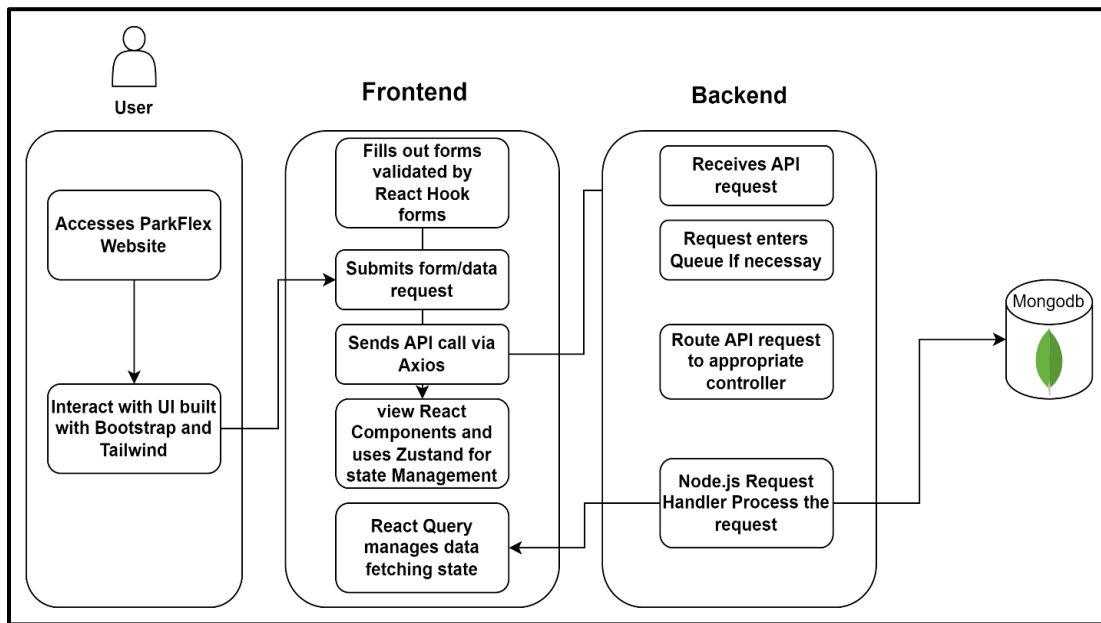


Fig 12. Process Workflow diagram

Process WorkFlow of Authentication Feature

The Authentication feature of the ParkFlex application operates through a coordinated process flow where users initiate actions on the frontend, interacting with forms for registration, login, or password resets. React components handle the input validation, and upon submission, Axios facilitates API requests to the Node.js backend. The backend, in turn, manages these authentication-related requests checking credentials, registering new users, or processing password changes against the MongoDB database. Responses are then relayed back to the frontend, where state management with Zustand and notifications via React Toasts provide users with real-time feedback on the outcome of their actions.

7. Folder Structure

For the frontend, the application leverages Bootstrap and Tailwind CSS for styling, alongside React Hook Forms for form validation. It also uses Zustand for state management, Axios for HTTP requests, React Query for data fetching management, and React Toasts for notifications. On the backend, Node.js with the Express.js framework is used. The database employed is MongoDB [4].

Frontend:

The displayed folder structure is indicative of a React-based frontend in the ParkFlex application.

Inside the client directory, the src folder is where the bulk of the application's code resides, including

assets for static files like images, components for React UI components, and store for state management. Key files like App.css and App.tsx define the main component's styling and logic, while index.css and index.tsx set up global styles and the React entry point. Configuration files for TypeScript and testing utilities are also present, facilitating type checking and test configurations [5].

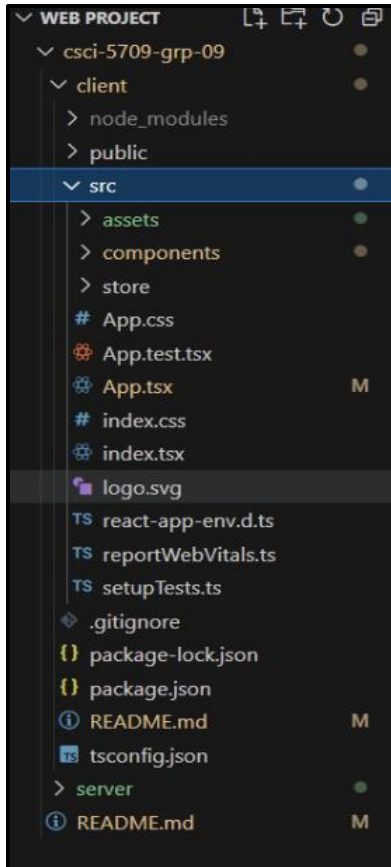


Fig 13. Frontend Folder Structure [7]

Backend:

This backend folder structure for the ParkFlex application organizes the Node.js/Express server into a set of clear, functional directories: config for settings and configurations, controllers for request handling logic, middlewares for request preprocessing functions, models for data schema definitions interacting with MongoDB, routes for defining API endpoints, and utils for shared utility functions. The server.js file is the entry point of the server that ties all these pieces together. The README.md file likely contains documentation or instructions related to the backend.

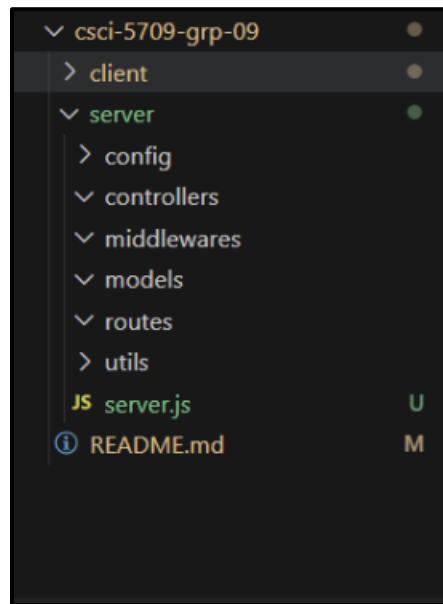


Fig 14. Backend Folder Structure [7]

References:

- [1] Jay N. Rana, "CSCI 5709 ASSIGNMENT 1." Dalhousie University, [online document], 2024. [Accessed 7 March-2024]
- [2]"Balsamiq Cloud", Balsamiq.cloud, 2024. [Online]. Available: <https://balsamiq.cloud/s3vq5xn/> . [Accessed 7 March-2024].
- [3] "Lucid," LucidChart. [Online]. Available: <https://www.lucidchart.com/pages/> [Accessed: Feb 07, 2024]
- [4] draw.io - free flowchart maker and diagrams online, "Flowchart Maker & Online Diagram Software," Draw.io, 2024. [Online]. Available: <https://www.draw.io/> . [Accessed 6 March-2024]
- [5] Jay N.Rana, Neel P. Patel, Aditya M. Purohit, Ketul Patel, Mann Patel, Shubham V. Patel, "Project Proposal." Dalhousie University, [online document], 2024. [Accessed: 10 March-2024]
- [6] Jay N. Rana, Aditya M. Purohit, Ketul Patel, Mann Patel, Shubham V. Patel, Neel P. Patel, "CSCI 5709 ASSIGNMENT 1(Group)." Dalhousie University, [online document], 2024. [Accessed: 10 March-2024]
- [7] Jay N. Rana, "ParkFlex", Visual Studio Code,[Tool], [Accessed: 9 March-2024]