

CS 30700 Design Document

02.19.2021

Team 17 :

Steven Bass

Luke Irons

Logan Sweeney

William White

Austin Wilson

Index

1. Purpose	3
1.1. Non-Functional Requirements	4
1.2. Functional Requirements	5
2. Design Outline	7
2.1. High Level Overview of the System	8
2.2. Detailed Overview of the Client/Server/Database	9
2.3. Broad Overview of the Sequence of Events	10
3. Design Issues	11
3.1. Functional Issues	11
3.2. Non Functional Issues	13
4. Design Details	14
4.1. Class Design	14
4.2. Description of Classes and Models	15
4.3. Details of the Different States of the UI	16
4.4. Sequence of Events When User Starts Application	17
4.5. Sequence of Events When User Requests a Route	18
4.6. Activities for User Login	19
4.7. UI Mockup	20

1. Purpose

Rout is designed for users to be able to quickly make accurate, looping routes by providing a single location to start and end at. Users will be able to open their web browser of choice, input the starting/ending point without making an account, and receive a route that adheres to their specifications. In addition to the creation of the route, Users will also receive information about roughly how long it will take them to complete their route, how long their route is, and how many calories they will burn upon completion of the route. By allowing users to access this information without forming an account, Rout outpaces its competitors in capturing a user market that is either not willing to provide personal information (email, names) or is looking for a quick way to plot routes on the fly.

Although account creation is not necessary by design, Rout does allow for the creation of an account. Users who wish to save route information, easily share routes to other users, and keep track of secondary information such as total calories burned will be able to do so through account creation. Although competitors offer similar features, Rout will be designed with user accessibility held at paramount importance. Users will be able to access their saved information easily without fear of any potential security leak or data loss.

1.1 Non-Functional Requirements

1. Performance

- a. The client application should run smoothly with no visible latency.
- b. Route requests should be processed within one second.
- c. The web server should support at least 1,000 concurrent users.

2. Backend

- a. The user login database should support at least 10,000 unique users.
- b. The route-generation algorithm should minimize calls to the Google Maps API.

3. Security

- a. User passwords should be stored in a salted SHA-256 hash.
- b. Repeated login attempts should be capped to prevent brute force attacks on accounts.
- c. Password resets should be tied to secure outside services (user email/phone).

4. Usability

- a. The application should be easy to understand and use with minimal explanation.
- b. The application should run on all modern web browsers.

As a developer,

- I would like the pathfinding algorithm to run in polynomial time.
- I would like administrator accounts which give access to backend data.
- I would like to be able to push changes to the UI without taking down the site for management.
- I would like to implement a pathfinding algorithm which uses the Google Maps Directions API.
- I would like to use the Google Maps API to create possible routes.
- I would like a Javascript framework for making calls to the Google Maps API.
- I would like to be able to edit user information within the database.
- I would like a MySQL database for storing account information.
- I would like to be able to view usage metrics of Rout.

1.2 Functional Requirements

User Account:

As a user,

- I would like to find a new route without needing an account.
- I would like to register a new Rout account.
- I would like to login to my Rout account.
- I would like to logout of my Rout account.
- I would like to be able to reset my password through email.
- I would like to be able to reset my password through SMS.
- I would like to be able to secure my account with 2 factor email authentication.
- I would like to be able to secure my account with 2 factor SMS authentication.
- I would like to be able to customize my nickname on my profile.
- I would like to be able to customize my profile with a profile picture.

User Interface:

As a user,

- I would like the UI of the website to be easily understood and used.
- I would like a brief tutorial on the UI the first time I use the application.
- I would like to adjust the color of the UI to dark mode.
- I would like the website to have a visually appealing color palette.
- I would like to be able to adjust default measurements between Miles/Kilometers.
- I would like to have a UI element to customize what type of route I am looking for.

Route Planning:

As a user,

- I would like for outside temperature to be displayed in the route planning screen.
- I would like for outside weather conditions (rain, snow, etc) to be displayed in the route planning screen.
- I would like for date/time information to be displayed in the route planning screen.
- I would like to drop a pin to use as a start point.
- I would like to use an address as a start point.

- I would like to have autocomplete when typing an address.
- I would like to be able to see important turning points on my route through Google 360° Street View.
- I would like to find multiple routes from the same starting point.
- I would like to find a route using distance.
- I would like to find a route using time and a pace.
- I would like to find a route by hilliness.
- I would like to find a route only on sidewalks.
- I would like a “difficulty score” based on the intensity of my route.
- I would like to plot routes with specific waypoints to pass through.
- I would like to track my progress along a route in real time via GPS.

Route Completion:

As a user,

- I would like to save a route for future use.
- I would like to view and use saved routes.
- I would like to be able to export saved routes as a file for sharing purposes.
- I would like to be able to share routes with other users through a URL.
- I would like to be able to share my routes through popular social media services (Facebook, Twitter, Instagram).
- I would like to be able to rate a route following its completion.
- I would like to know the pace of the previous route.

User Statistics:

As a user,

- I would like to see how many calories I’ve burned after taking a route.
- I would like to be able to visualize the amount of calories I’ve burned.
- I would like to view my total calories burned over all routes I’ve taken.
- I would like to compare my pace with distances similar to those ran by famous or olympic runners.
- I would like the information and statistics of previous routes to be stored.

2. Design Outline

The structure of Rout falls under the Client-Server model. Our central server (hosted through AWS) will handle multiple client connections simultaneously. We will also have a module for handling client-Maps API requests whenever new route information is required. The server will be tasked with handling any non route calculating requests from the client whereas the maps module handles the core functionality of the web app, the creation of new routes.

1. Client

- a. The client will allow the user to interact directly with Rout's functionality.
- b. The client will make requests to the Maps module whenever a new route is being created.
- c. The client will pass relevant route information from the Maps module to the server.
- d. The client will have a comprehensive UI that allows users to create routes, view past routes, and customize their experience.

2. Server

- a. The server will handle all interactions between client requests and the database.
- b. The server will receive past route info, calories tracked, and any client-requested information from the database.
- c. The server will send client information (username, password, etc) to the database for storage.
- d. The server will send new route information from the client to the database.

3. Maps API Module

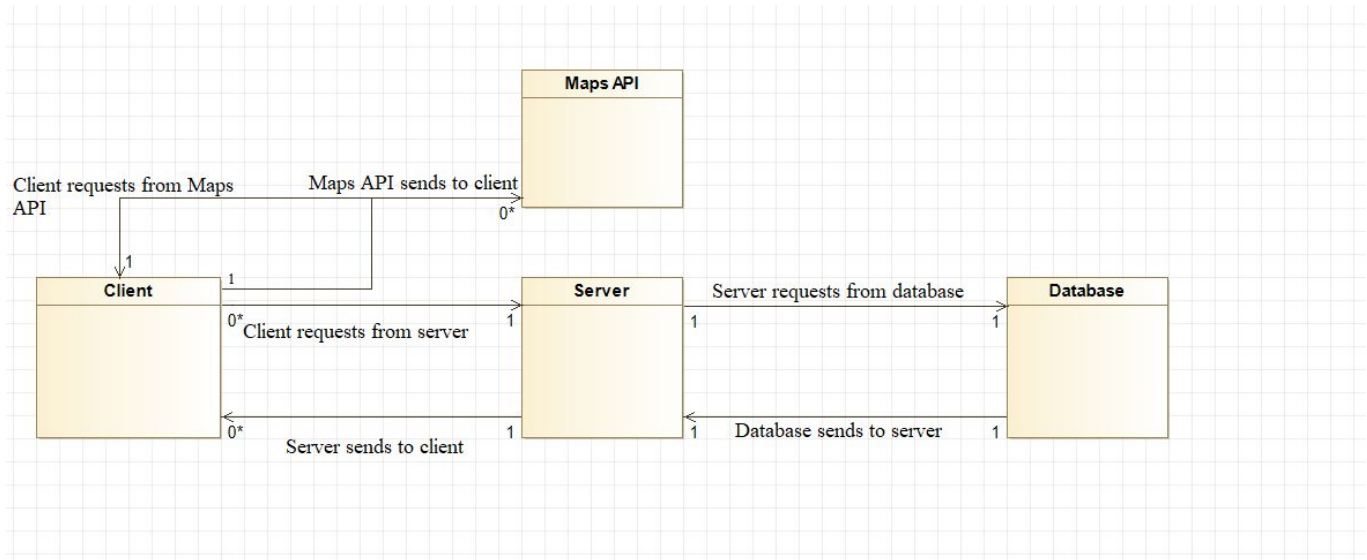
- a. The Maps API module will handle requests from the client for new Routes.
- b. The Maps API will not directly interact with the server or database.

4. Database

- a. The database will store saved routes, calories, user information, and default information (default profile pics etc).

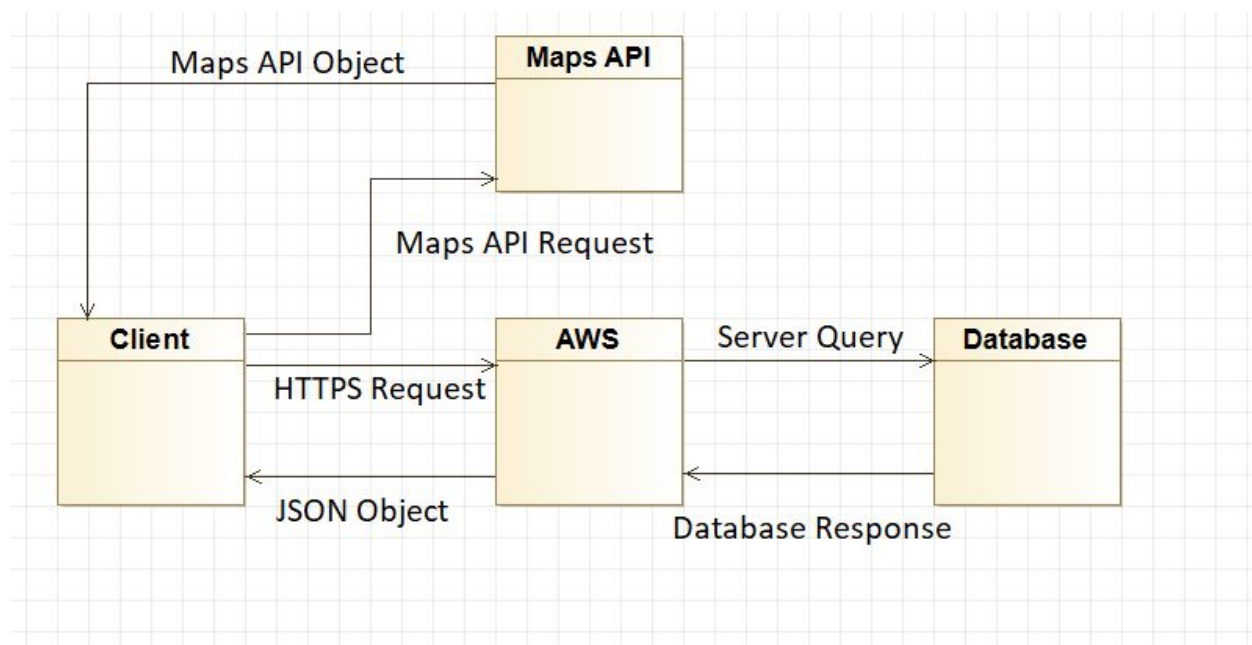
2.1 High Level Overview of the System

Multiple Rout clients can send requests for routes from the Maps API and send the route information to the server to be parsed and returned to the UI. If the client requests data from the server (past routes, profile pictures, etc), the server will request the data from the database and pass the data to the client.



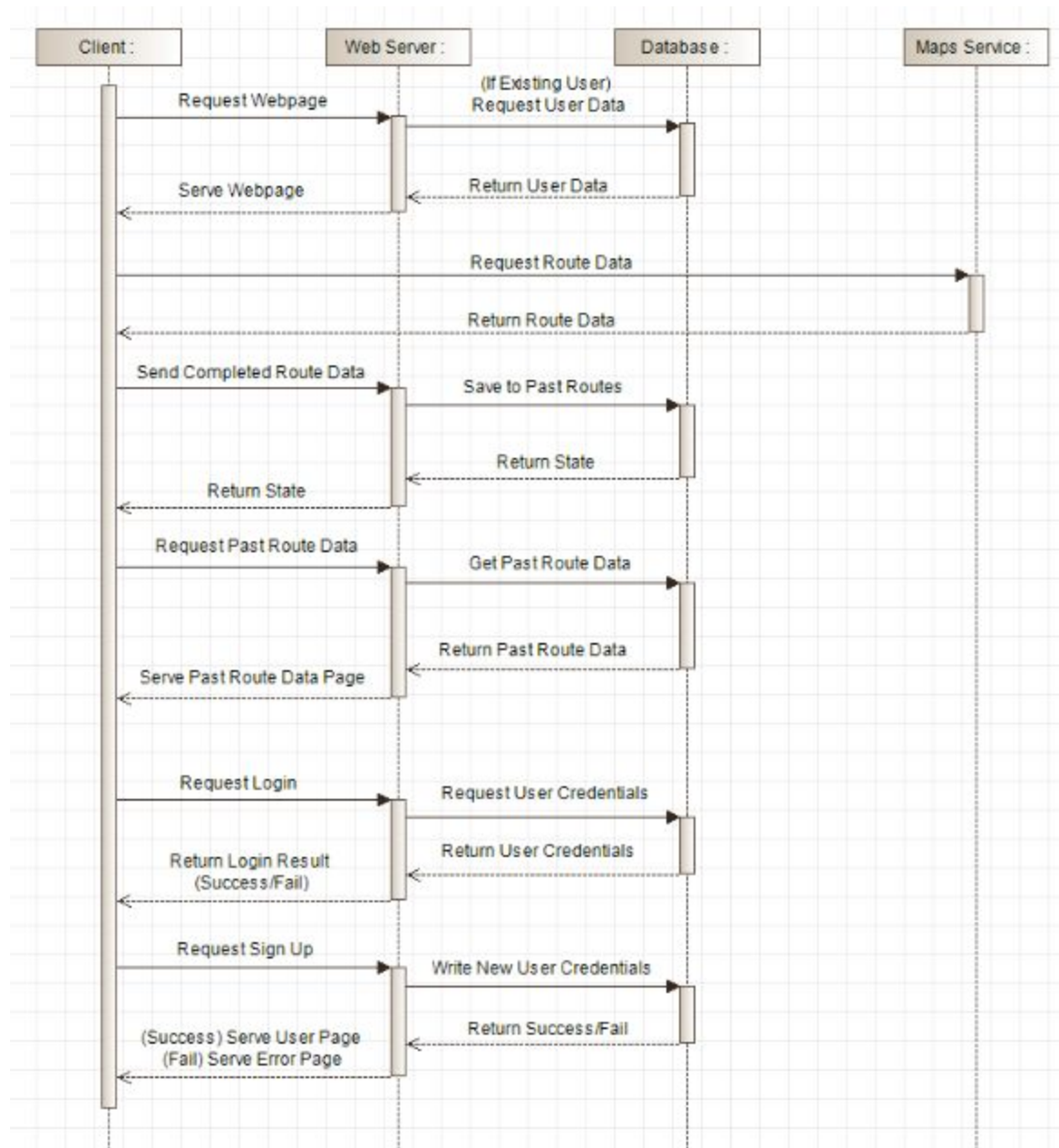
2.2 Detailed Overview of the Client/Server/Database

The Maps API and Client will communicate through Maps API requests. The client will make requests for route objects and then pass them to the AWS server for parsing. The server, hosted through AWS, will pass the parsed routes to the database if the route is to be saved and returned to the client regardless of if the route is saved or not. A JSON object storing the route will be sent to the client to be shown on the UI for the client to see and interact with.



2.3 Broad Overview of the Sequence of Events

Communication primarily takes place between the client and the webserver, with the server requesting information from the database as needed. The exception to this rule is when requesting a new route, the client directly contacts the maps service to get the necessary route data, bypassing the central server for all new route requests.



3. Design Issues

3.1 Functional Issues

1. Should users need to login to use Rout?

Option 1: Users will not be required to login to access the features of Rout

Option 2: Users will be required to login to access the features of Rout

Option 3: Users will not be required to login, but they will still be able to access the majority of Rout features.

Decision: We chose option 3 as it provides the most flexibility to Rout users. Although features such as saving route information and tracking total calories burned would be nice to provide to users without accounts, since we cannot associate the data in any meaningful way without user information, we opted out of allowing accountless users to utilize those features. However, we still want to provide users who are either too busy or concerned with privacy to use the basic features of Rout without an account.

2. Should users be given access to two-factor authentication to protect their account?

Option 1: Users will not be allowed two-factor authentication.

Option 2: Users will be allowed email only two-factor authentication.

Option 3: Users will be able to choose between SMS, email, and no two factor authentication.

Decision: We chose option 3 once again for its flexibility in user preferences. Although SMS protection may be difficult to implement, we found it important to allow users customization when it comes to protecting their account. Option 1 was in consideration for a bit, but ultimately it was decided that allowing users to secure their account would be a beneficial move for the security of Rout.

3. Will and how would we provide user statistics?

Option 1: There will be no user statistics for previous routes.

Option 2: There will be user statistics only using specific inputs given for previous routes.

Option 3: There will be user statistics gained from population averages for previous routes.

Decision: We choose options 2 and 3 so that we can provide users with statistics to observe growth. By allowing us to use population averages we can also be more flexible and give users estimates when they do not provide us with all necessary information, but also inputs the given information whenever possible for the most accurate statistics.

4. How will sharing work between users?

Option 1: Users will be able to share routes between each other by using built in Facebook, Twitter, and Instagram sharing links and by requesting a “shareable” URL with the route information attached.

Option 2: Users will be able to share routes solely with a shareable URL.

Option 3: Users will not be able to share routes.

Decision: We decided to allow users more choices for sharing through popular social media apps and a shareable URL. Option 1 covers all of these bases and allows users ultimate flexibility in their approach to sharing routes. The idea is to compile an image every time a user shares a route that includes some basic route info and a URL other users can visit to run the same route.

3.2 Non-Functional Issues

1. What kind of database should we use for the backend of Rout?

Option 1: MongoDB

Option 2: mySQL

Option 3: Microsoft SQL

Decision: We opted to use mySQL for our database as some of our team members are familiar with mySQL and vouched for it personally.

2. Which languages should we use for implementing our frontend services?

Option 1: Python.

Option 2: Java

Option 3: JSReact

Decision: We chose to use JSReact for the frontend development of Route. React is an open-source library for Javascript built for web development, essentially it is perfect for our purposes.

3. Where should we source our mapping information?

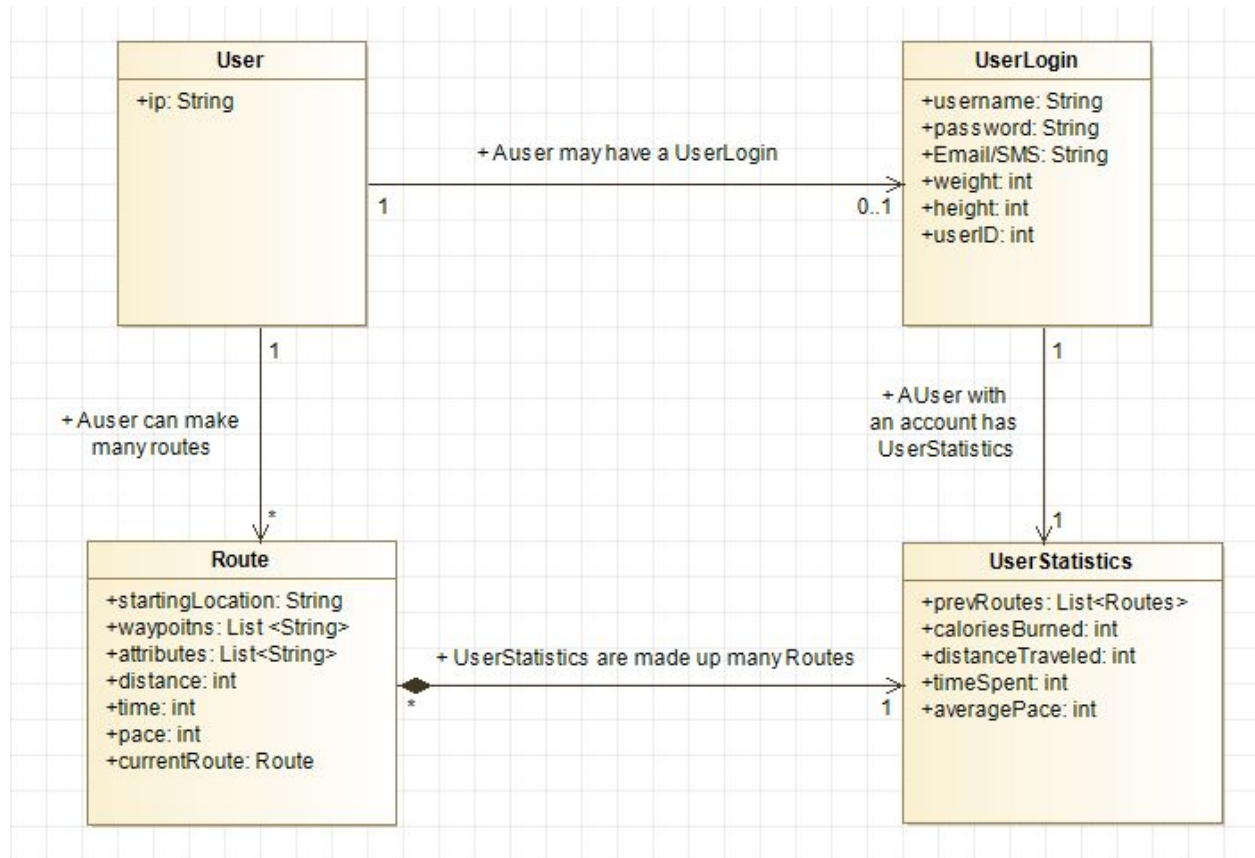
Option 1: Stadia

Option 2: Google Maps API

Decision: We chose to work with the Google Maps API as it is free to use (up to 20000 requests) for developers and most of us on the team were familiar with Google Maps.⁵⁵

4. Design Details

4.1 Class Design



4.2 Description of Classes and Models

These classes were designed as the main classes that our application would implement. Each class has a list of variables that is either needed or useful for our application to function.

User:

- Represents a user that does not need to be logged in.
- There are no necessities for users without a log in, but an IP address will help in finding a user's location easier.

UserLogin:

- Represents a user that is logged in.
- Not necessary for the creation of a route.
- Contains information for logging in like username, password, and email.
- Each user with an account will be given a unique ID for database purposes.
- Users will also be allowed to change their weight and height after the creation of an account to get more accurate results about calories burned.

Route:

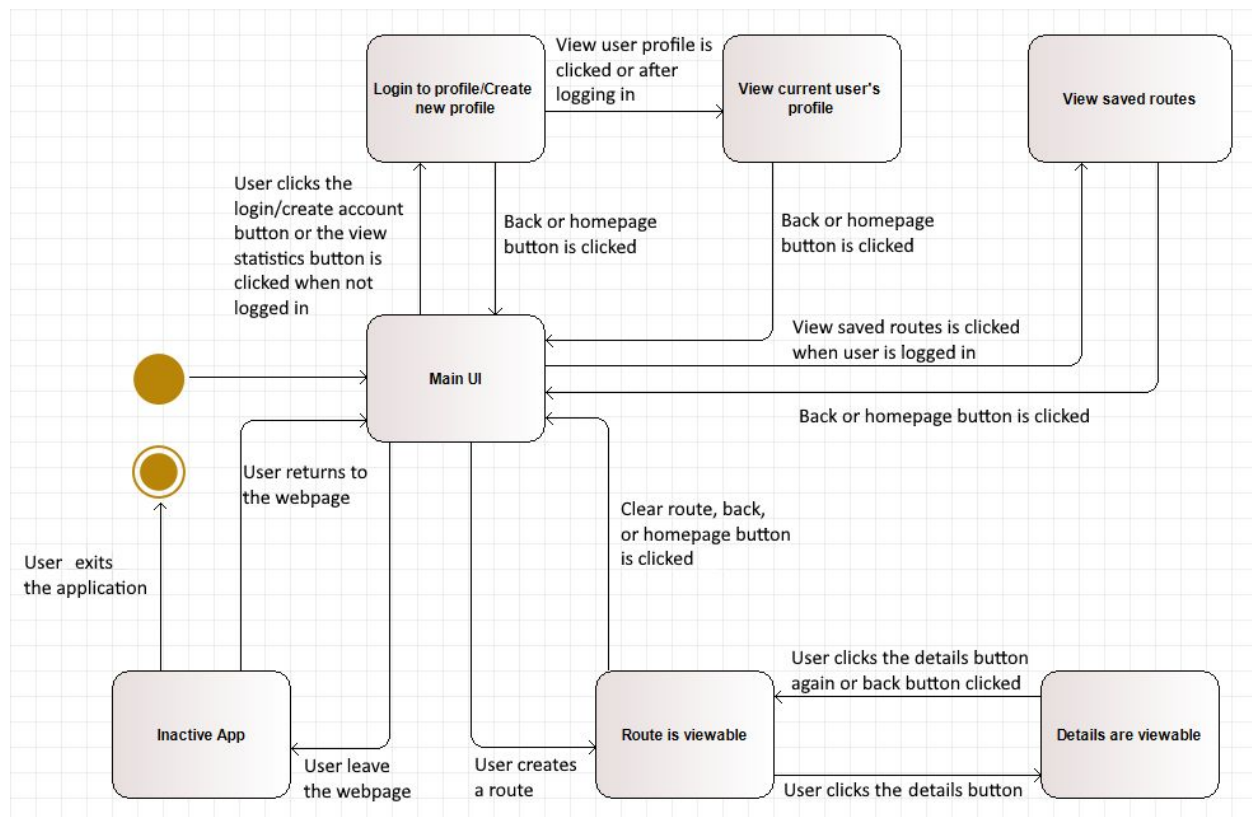
- Represents a route being created by a user.
- To create a route a starting location must be specified.
- Waypoints can be set to direct the route that we will compute.
- Attributes will hold the possible specifications that users will define including routes with more nature, less hills, no highways, etc.
- To create a route users will also at minimum have to give either distance or a time and pace so that we can calculate a route.
- Routes will be stored in Route objects containing the above information for use with sharing routes and previous routes.

UserStatistics:

- Represents a user's stored statistics for only those with an account.
- Users will have a list of previous routes that contain routes they have run/walked.
- Statistics that would prove useful for the tracking of progression are stored like the amount of calories burned, total distance traveled, total time spent traveling, and average paces of routes.

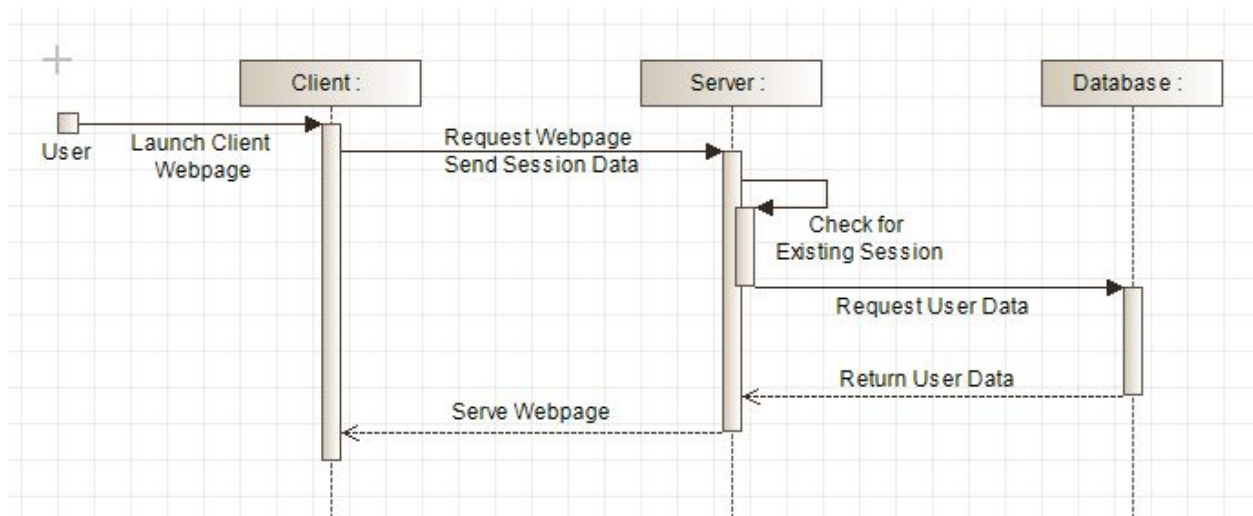
4.3 Details of the Different States of the UI

This diagram describes the various states that our applications user UI can be in. It consists of the different web pages and widgets that we will display in our web application. After entering the website the user will be in the Main UI which consists mainly of the Google Maps API. The user has many buttons that can be clicked to enter different pages and show specific widgets and after doing so can return to the Main UI by clicking the back button or clicking the specific button that showed the widget again. When the user is not currently on the web page the application will be inactive, which resumes activity when returning. The application is exited when the webpage is closed in the user's browser. The webpage does not specifically have to go through the inactive state to exit the application, but it is shown this way for the cleanliness of the diagram.



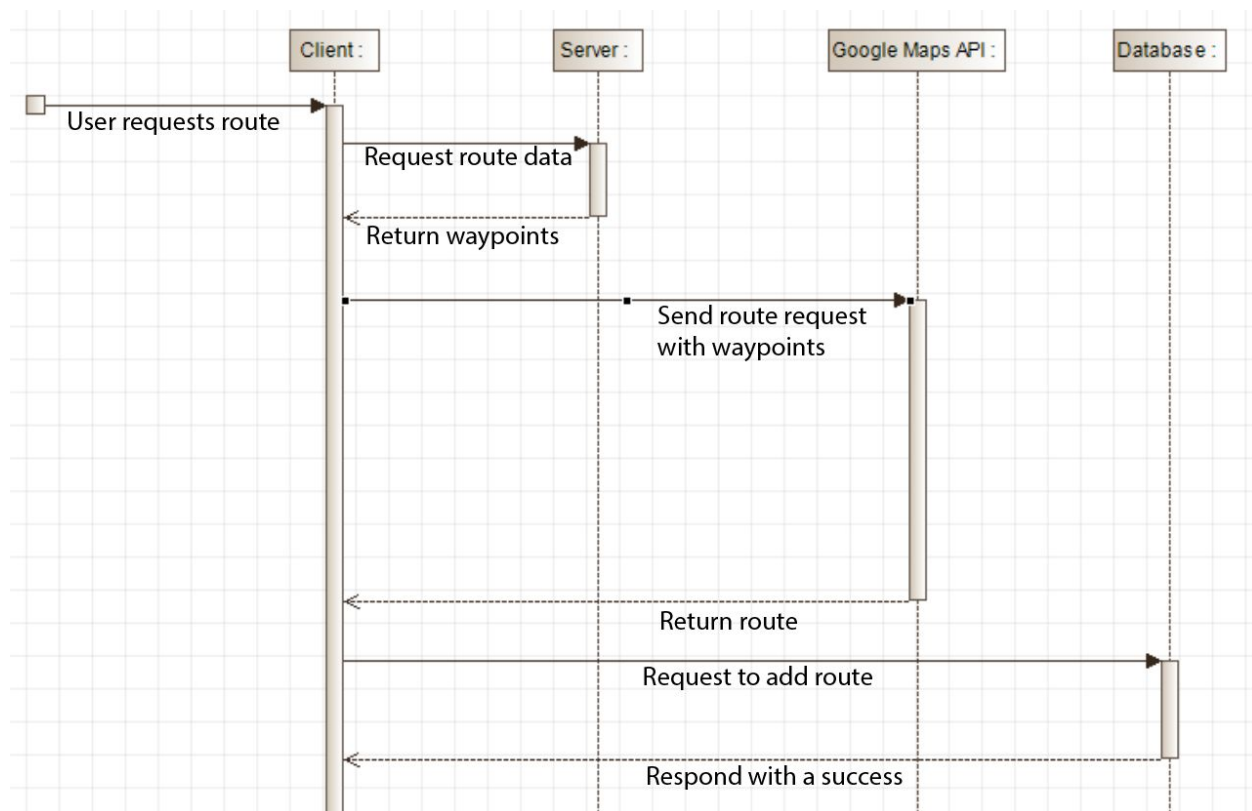
4.4 Sequence of Events When User Starts Application

When the user connects to the webpage, their browser will send session data to the web server which can be checked against existing user sessions. If the session key matches, the user is automatically logged-in; otherwise, they are served the guest landing page. If the session key exists but does not match, users receive a notification that they have been logged out.



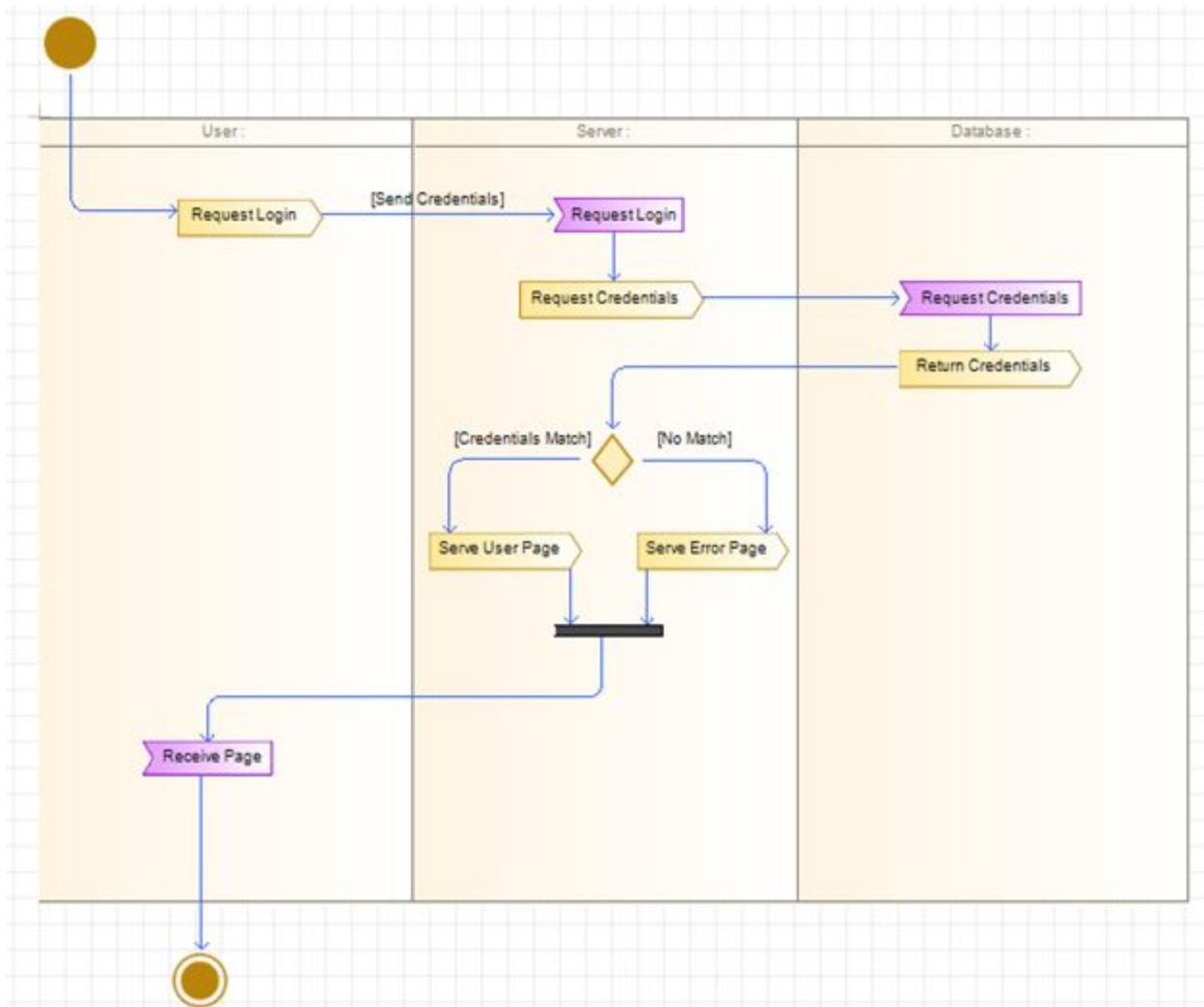
4.5 Sequence of Events When User Requests a Route

After a user has specified the data needed to make a request and makes the request for a route this sequence diagram will begin. The route data given as input is given to the server, parsed, then returned as waypoints for the Google Maps API to use. These waypoints are then given to the API to calculate the route for the user. The returned route will then be requested to send to the database for storage of past routes.



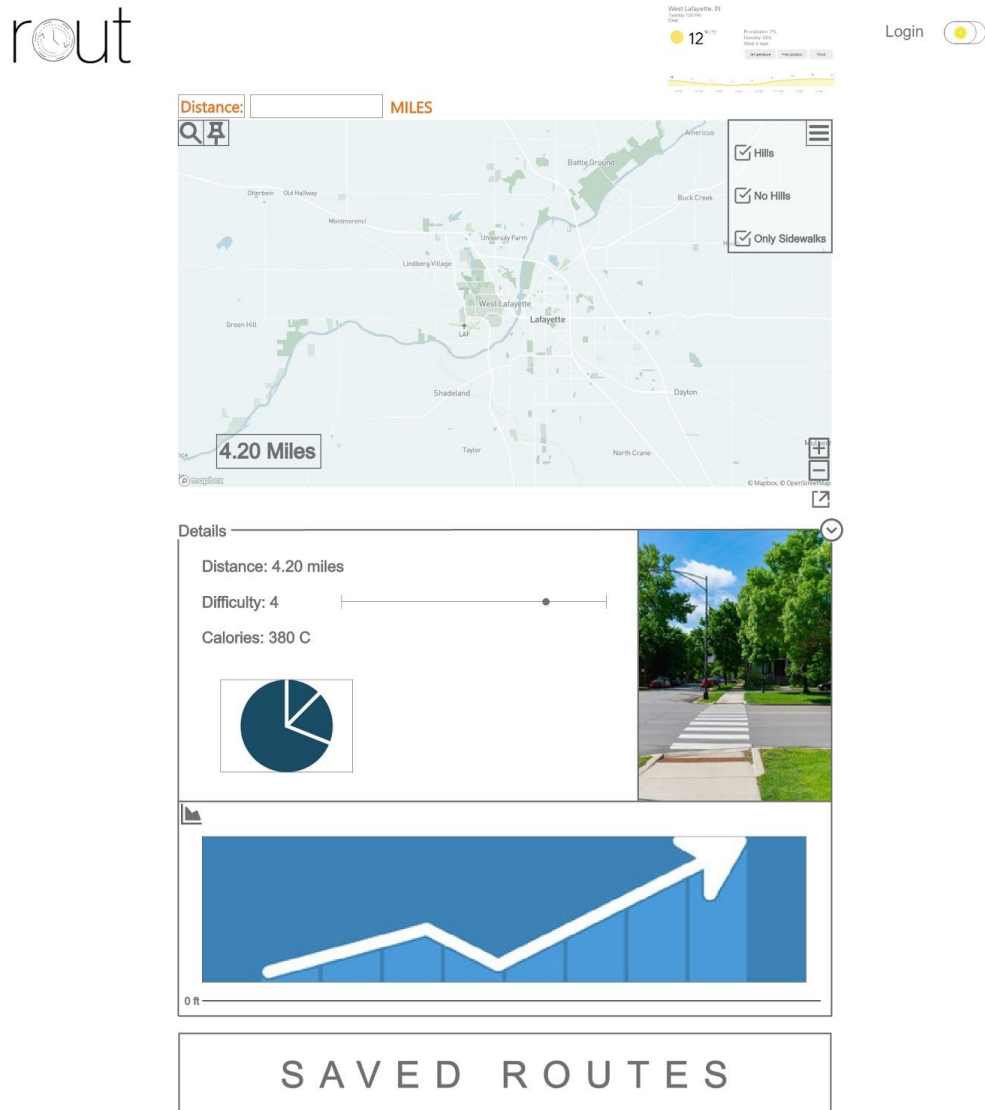
4.6 Activities for User Login

To log in, users will send their username and password to the server, where the server will compare the password against the password of the corresponding user row in the database, serving the user their landing page if the passwords match, or an error page if the passwords do not match.



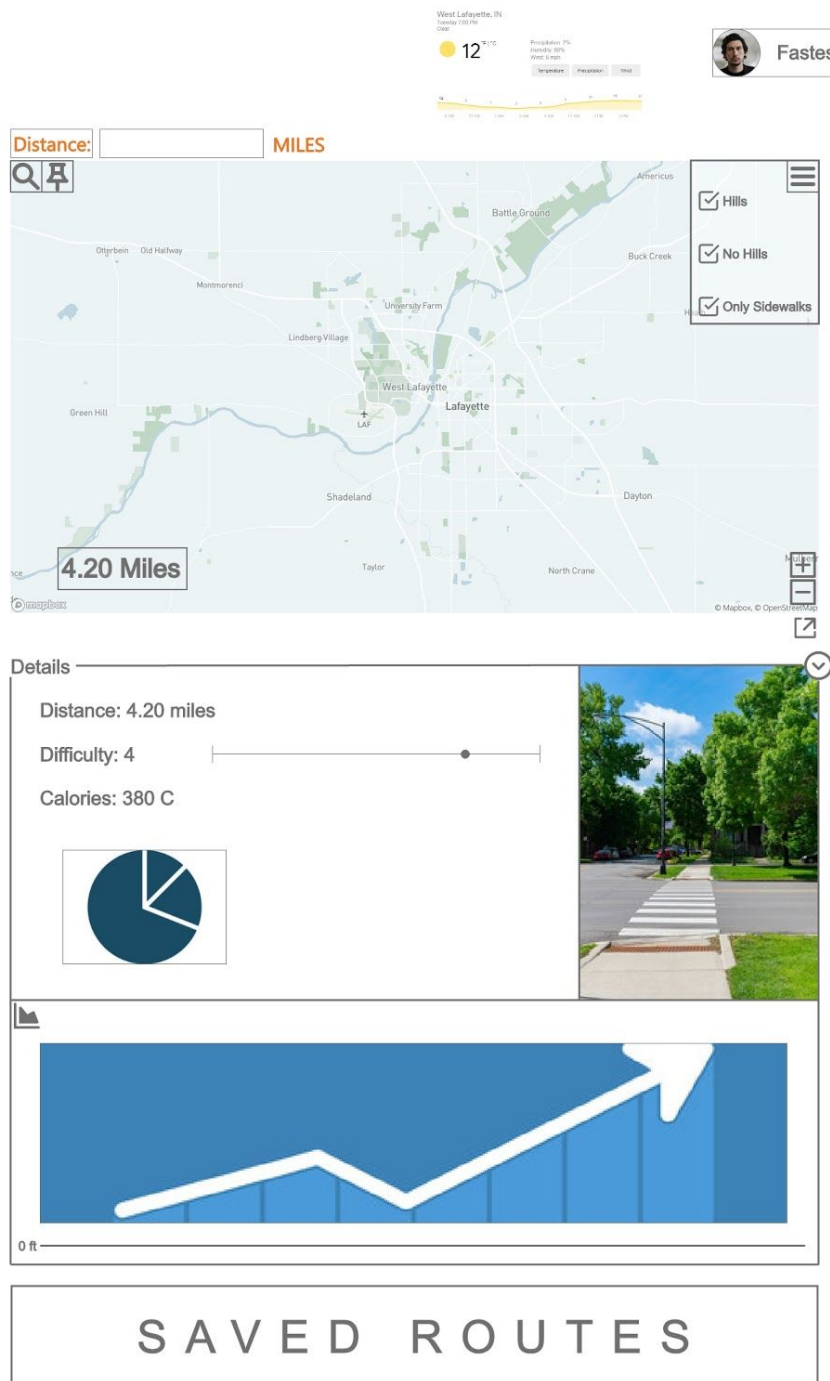
4.7 Mockup UI

This is the home page of the application, showing the map, route details, login, saved routes, as well as a dark mode toggle. It has an option to input a given distance. The colored word “Distance” will work as a toggle between inputting just distance or just inputting time and a pace. The colored word “Miles” will work similarly so that the user can toggle between miles and kilometers.



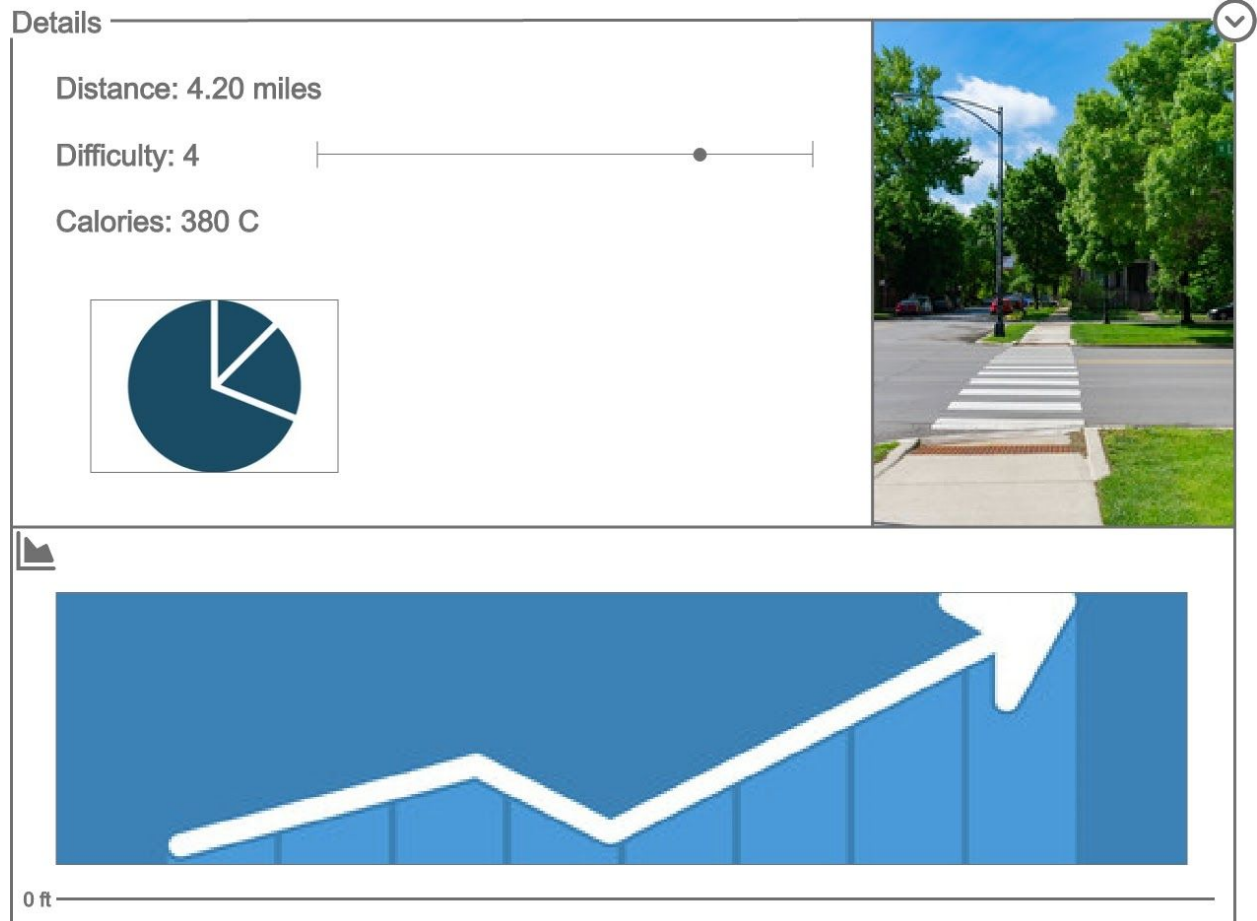
This is the home page with a user that is logged in. The User Profile in the top right will also function as a button to access the user profile page.

rou**t**



rou**t**

The details drop down will include details about a chosen route including distance, difficulty, and calories. The picture towards the right will be the first intersection and the graph at the bottom will show information about altitude changes throughout the route.



This is the login page, used for logging into a user account. New users can register using the “New? Register Here!” button.



Welcome!

Email:

Password:

[Forgot Password?](#)

SIGN IN

New? Register Here!

This is the page for registering a new user account simply using email.



Register to Save Routes and More!
It's Free!

Username:

Email:

Password:

Register

The Profile page is where the user will be able to edit their profile picture, username, email, and password. It will also allow the user to add their height and weight to more accurately predict calories burned. Finally, this is where the user will be able to enable two-factor authentication.



Profile

[Edit](#)

<input type="text" value="Username:"/>	<input type="text"/>
<input type="text" value="Email:"/>	<input type="text"/>
<input type="text" value="Password:"/>	<input type="text"/>

Additional Information

<input type="text" value="Height:"/>	<input type="text"/>
<input type="text" value="Weight:"/>	<input type="text"/>









☒ Two Factor Authentication

<input type="text" value="Email:"/>	<input type="text"/>
<input type="text" value="Phone:"/>	<input type="text"/>

The Saved Routes page is only accessible to logged in users and will display previously logged routes with any information saved from them such as distance, difficulty, and metrics like pace for users that log that.



SAVED ROUTES

<p>Wednesday February 17th, 2020</p> <p>Distance: 4.20 miles</p> <p>Difficulty: 4</p> <p>Calories: 380 C</p> <p>Average Pace: 8:01/mile</p> 	
<p>Tuesday February 16th, 2020</p> <p>Distance: 4.20 miles</p> <p>Difficulty: 4</p> <p>Calories: 380 C</p> <p>Average Pace: 8:01/mile</p> 	
<p>Monday February 15th, 2020</p> <p>Distance: 4.20 miles</p> <p>Difficulty: 4</p> <p>Calories: 380 C</p> <p>Average Pace: 8:01/mile</p> 	
<p>Sunday February 14th, 2020</p> <p>Distance: 4.20 miles</p> <p>Difficulty: 4</p> <p>Calories: 380 C</p> <p>Average Pace: 8:01/mile</p> 	
<p>Saturday February 13th, 2020</p> <p>Distance: 4.20 miles</p> <p>Difficulty: 4</p> <p>Calories: 380 C</p> <p>Average Pace: 8:01/mile</p> 