wander

Design Document

Team 18

Venkata "Sai" Edupulapati

Akshaya Kumar

Shreya Sanapala

Theerapatra "Visa" Thongdee

Vidya Vuppala

Index

Purpose **1**

Functional Requirements 2

Non-Functional Requirements 4

Design Outline **6**

High Level Overview 6

Sequence of Events Overview 8

Design Issues **9**

Functional Issues 9

Non Functional Issues 12

Design Details **15**

Class Design 15

Descriptions of Classes and Interaction between Classes 15

Sequence Diagram 18

Navigation Flow Map 21

UI Mockups 24

Purpose

Traveling the world can be a very fulfilling experience; however, planning the trip itself can be extremely challenging. With all the elements that need to be considered and in order to plan and execute a trip that is relaxing, entertaining, and lively throughout the whole journey, to-be travelers may be put off by the amount of work required to do so. Enter Wander.

The purpose of the project is to develop a website that provides an all-encompassing solution that simplifies the process of travel planning and ensures quality entertainment along the way. Through the use of personal music preferences and external travel data, Wander can create a customized music playlist to enhance the journey from the user's starting point and an itinerary of popular attractions to visit based on a user's input. Several existing platforms deal with itinerary creation and playlist personalization, but not necessarily both simultaneously. However, while there are existing applications that provide generic, lackluster, and oftentimes ill-suited travel itineraries for the user, our website eliminates tedious planning and prioritizes the journey, not just the destination.

Functional Requirements

1. User Account/Profile

   As a user,

   a. I would like to be able to log in with my account.
   b. I would like to be able to delete my account.
   c. I would like to be able to create a profile with a username
   d. I would like to be able to modify my username.
   e. I would like to be able to add a profile picture to my account.
   f. I would like to be able to modify my profile picture.
   g. I would like to be able to delete my profile picture.
   h. I would like to be able to view a profile page with my profile information.
   i. I would like to be able to change my password if needed.
   j. I would like to be able to reset my password if needed.
   k. I would like to be able to delete my account if needed.

l. I would like to be able to edit my account information (ex. demographic, music preference, etc.) as needed.
m. I would like to be able to view my listening data (total minutes, genres, artists).

2. Music Preferences

   As a user,

   a. I would like to be able to link my Spotify account to my Wander account.
   b. I would like to be able to change which Spotify account is linked to Wander.
   c. I would like to be able to remove my Spotify account.
   d. I would like to be able to take a quiz to determine my music preferences.
   e. I would like to be able to retake the quiz for different results.
   f. I would like to be able to see my past music preferences (i.e. past answers to quizzes, past results).
   g. I would like to be able to reset my quiz history.
   h. I would like to be able to toggle on preferences for a playlist (ex. seasonal, genre, mood, location).
   i. I would like to be able to see my current favorite music preferences.

3. Playlist Creation

   As a user,

   a. I would like to be able to get a music playlist for my journey.
   b. I would like to be able to see my playlist embedded on the website.
   c. I would like to be able to edit my playlist.
   d. I would like to be able to export my playlist.
   e. I would like to be able to see my past curated playlists.

4. Itinerary Creation

   As a user,

   a. I would like to be able to enter a starting location.
   b. I would like to be able to edit a starting location.
   c. I would like to be able to enter my destination.
   d. I would like to be able to edit my destination.
   e. I would like to be able to set my travel preference (ex. car, plane) for each trip.

   f. I would like to be able to edit my travel preference (ex. car, plane).
   g. I would like to be able to get a travel itinerary for my destination.
   h. I would like to be able to edit my itinerary.
   i. I would like to be able to see the overall map of my travels (i.e. show the map from starting point to destination).
   j. I would like to be able to see a smaller map of my itinerary midpoints (i.e. show a map from location A to location B to location C).
   k. I would like to be able to have a real-time map to see my current location.
   l. I would like to be able to export my itinerary.
   m. I would like to be able to see my past travel history.

5. Website Customization/Appearance

   As a user,

   a. I would like to be able to access Wander on all devices as a website.
   b. I would like to be able to personalize the appearance of the website (light vs. dark theme, etc.)
   c. I would like to be able to hear sound effects from the website.
   d. I would like to be able to see animations within the website.

6. Sharing

   As a user,

   a. I would like to be able to share my non-collaborative playlist within the website with one other user of this website (if time allows).
   b. I would like to be able to share my non-collaborative itinerary within the website with one other user of this website (if time allows).
   c. I would like to be able to share my non-collaborative playlist within the website with multiple other users of Wander (if time allows).
   d. I would like to be able to share my non-collaborative itinerary within the website with multiple other users of Wander (if time allows).
   e. I would like to be able to collaborate with a friend to create a playlist (if time allows).

f. I would like to be able to collaborate with a friend to create an itinerary (if time allows).

Non-Functional Requirements

1. Performance

   As a developer,

   a. I would like the website to run smoothly without crashing.
   b. I would like the website to be responsive to all user requests.
   c. I would like the application to be launched in 5 seconds.
   d. I would like the server to be able to handle concurrent users.
   e. I want both the client and server to elegantly handle errors and the failure of other components that they interact with.

2. Server

   As a developer,

   a. I would like the server to be able to store user playlist data in a Firebase database.
   b. I would like the server to be able to store itinerary data in a Firebase database.
   c. I may want the server to be able to save user state and preferences.

3. Appearance

   As a developer,

   a. I would like to have an aesthetically pleasing website interface.

4. Security

   As a developer,

   a. I would like to allow the users to choose whether to share data location or not.
   b. I would like to protect the user's password and other personally identifiable information stored in the database.
   c. I would like the user data collected (e.g. photos, names, email addresses) to be anonymized or properly secured.

5. Usability

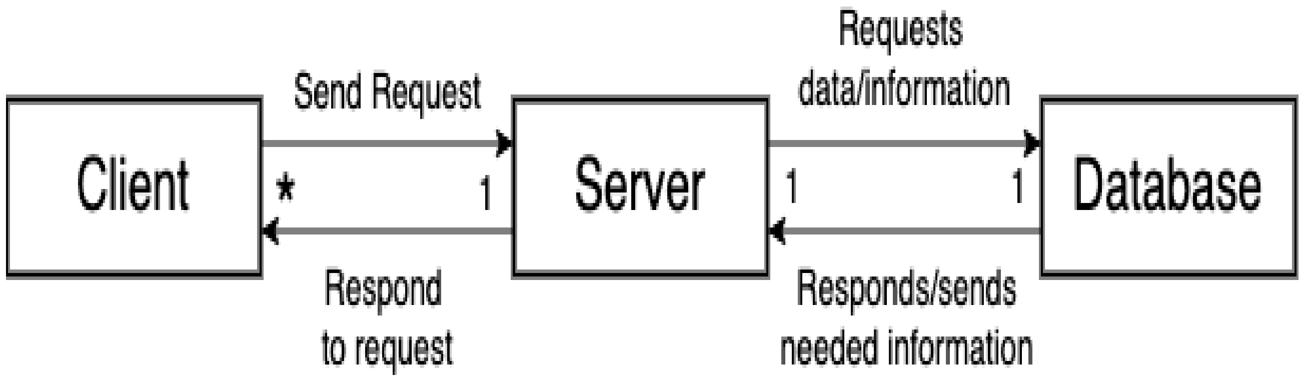
   As a developer,

a. I would like the website to be accessible on all platforms and devices.
b. I would like to have an easy-to-use website interface.

Design Outline

High Level Overview

Our project will be a website that will allow users to listen to a personalized playlist based on their location during the trip as well as their music preferences determined via their Spotify account or in-app quizzes. This application will use the client-server model. A client-server model is a distributed application structure that partitions tasks or workloads between the providers of a resource (i.e. servers) and service requests (i.e. clients). Here, the clients can send requests of data and information through the website's interface. With these requests, such as itinerary additions and playlist updates, being sent to the server to get processed and handled, the database will respond through sending the needed information over to the server then to the client where the user will see the corresponding changes made to the website interface.



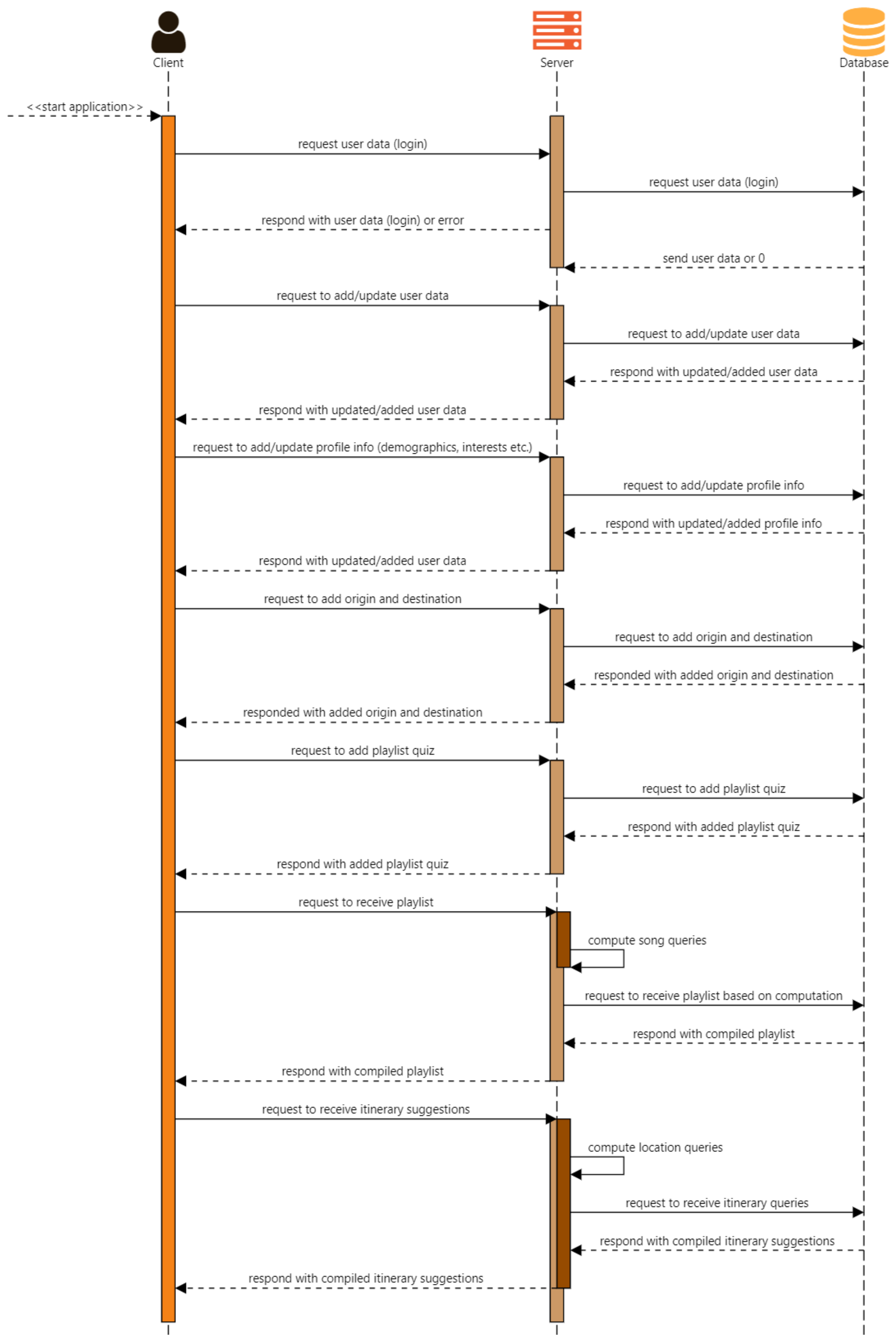
1. Client
    a. Client provides user with an interface from our system
    b. Client sends requests to the server when data is needed
    c. Client receives and responds to requests from the server
    d. Client updates the user interface with the corresponding changes

2. Server
    a. Server receives requests from client
    b. Server handles and validates requests sent from client
    c. Server requests needed data, files, or information from database
    d. Server responds to client by compiling and generating information from the database
3. Database
    a. Database stores all the data in the website (e.g. account details, playlists, itinerary)
    b. Database responds to requests sent by the server by sending the needed data and information

Sequence of Events Overview

Our project will follow a simple sequence of client → server → database interaction. The client will request data or request to add necessary information. The server then sends the request to the database or computes appropriate queries to request information from the database. The database either sends a response that information is saved or the appropriate information is sent. The server sends the appropriate response which is either the information asked or the appropriate response.

Sequence Diagram Overview


Client
Server
Database
<<start application>>
request user data (login)
request user data (login)
respond with user data (login) or error
send user data or 0
request to add/update user data
request to add/update user data
respond with updated/added user data
respond with updated/added user data
request to add/update profile info (demographics, interests etc.)
request to add/update profile info
respond with updated/added profile info
respond with updated/added user data
request to add origin and destination
request to add origin and destination
responded with added origin and destination
responded with added origin and destination
request to add playlist quiz
request to add playlist quiz
respond with added playlist quiz
respond with added playlist quiz
request to receive playlist
compute song queries
request to receive playlist based on computation
respond with compiled playlist
respond with compiled playlist
request to receive itinerary suggestions
compute location queries
request to receive itinerary queries
respond with compiled itinerary suggestions
respond with compiled itinerary suggestions

Design Issues

Functional Issues

1. What information do we need for registering an account?
    - Option #1: Username and Password
    - Option #2: Username, Password, and Security Question
    - Option #3: Username, Password, and Email
    - Option #4: Username, Password, Email, and Spotify login

Choice: Option #3

Justification: We decided to go with Option #3 over the other options. For Option #1, we felt that although it would be the most simple to implement, it would restrict us from being able to reset their passwords or send them notification updates. We also chose it over Option #2 because we felt that emails would have a much more standardized format as compared to the specific answers to each security question. For these reasons, we felt it would be much easier to just store email addresses instead. Finally for Option #4, we felt that while having the Spotify account details would have been convenient, it was ultimately nonessential. Instead of having to go through the process of setting up another authentication sequence, we felt it would just be much more simple to have an embedded Spotify playlist which they could view by logging in directly through Spotify.

2. What factors do we take into account when generating a new personalized playlist?
    - Option #1: Curate finite number of songs based on user's past listening history
    - Option #2: Create playlist based on user's past listening history and current preferences (determined through a short quiz)
    - Option #3: Generate new playlist of unique songs that are similar but new to the user and what they have already listened to before.
    - Option #4: Create playlist based on a combination of user preferences, estimated travel time, and what is currently trending at the user's final destination region.

Choice: Option #4

Justification: We decided not to choose Option #1 because we felt it was too simplistic for the scope of the project and also not very flexible for the user. In addition, we also felt that it was not much different from some of the features already existing within Spotify. We also decided that while Option #2 was definitely better than the previous one, it also lacked a genuine connection to the core concept of our application, which is based around traveling (even despite factoring in estimated travel time). We also eliminated Option #3 because while we felt that this option would ultimately produce one of the more unique experiences for the users, it would currently be too computationally intensive and intellectually challenging for us to not only first identify songs which the user has not already encountered, but also implement a mechanism for quantifying the similarity between two songs. A combination of all of these factors led us to decide on Option #4. We felt that not only did it appeal to the identity of our application, but it could also provide an option to users that isn't already readily available to them.

3. When do we handle destination tracking in our application?
    - Option #1: Provide a map interface that constantly updates estimated travel times and road conditions incrementally.
    - Option #2: Have a ticker on the map that dynamically updates the user's approximate location based on estimated travel time.
    - Option #3: Request permission to track the user's location while using the app and provide live updates about travel times.

Choice: Option #2

Justification: Once again, we felt that although Option #1 would be the easiest to implement, it also did not provide much added value to the user to simply know how long it would take them from their starting location when they had begun their trip already. Unfortunately, we also felt that Option #3 would not be the best option either due to the potential security and privacy implications that could arise from needing to track a user's live location over an extended period of time. For those reasons, we felt it was best to go with Option #2. We felt that it was the perfect balance because we could not only show them an estimate of their location, but we could also provide them with information such as time elapsed since beginning the trip and how much estimated time was remaining.

4. How should the user’s prior and current listening preferences be determined?
    - Option #1: Analyze their listening history through access to their Spotify account.
    - Option #2: Present them with a short quiz at setup of the account, after which they can update their choices as needed.
    - Option #3: Present them with a quiz prior to every trip to gain the most up-to-date information about their listening preferences.

Choice: Option #3

Justification: We decided to go with Option #3 because the user’s music preferences may shift due to the characteristics of a particular trip. For example the user may wish to listen to a more upbeat playlist on a trip with friends and family, but on a trip with their significant other, they may wish to listen to romantic music. By coupling each quiz with each trip, the user does not need to reset their preferences each time they wish to have a different mood as they would have to if we went with Option #2. Option #3 also serves to maintain the user’s listening preferences when they restore a past trip. We felt that while it was viable to go through their Spotify listening data as outlined in Option #1, it would most likely be more accurate and personalized for the user if they simply gave us direct input into exactly what they were looking for.

Non Functional Issues

1. What web service should we use?
    - Option #1: GitHub Pages
    - Option #2: Firebase Hosting
    - Option #3: Amazon Web Services
    - Option #4: Microsoft Azure

Choice: Firebase Hosting

Justification: We decided to use Firebase Hosting because it seemed like the most straightforward way to connect it with our database of choice. In addition, given that our website includes dynamic components, we realized that GitHub Pages was not the best option. Finally, an added complication to our consideration was the fact that our database framework would also largely be responsible for user authentication. To that end, we recognized that while third party software could be used to connect Firebase with the likes of Azure and AWS, Google's own hosting service (naturally) seemed to be the most simple way.

2. What language/framework are we using to implement the front end of the application?
    - Option #1: React
    - Option #2: Flutter
    - Option #3: Angular
    - Option #4: Swift

Choice: React

Justification: For similar reasons to our database of choice, we also decided to use React to develop the front end of our application because we felt it would be the most simple. Angular is a language that has been around for a long time, however we felt that the documentation and tutorials available on React were more straightforward to follow. Our final reason for choosing React over the remaining options above however was that we wanted to develop an elementary understanding of JavaScript, and we felt that with React itself being a JavaScript library, it would provide us with the best opportunity to do so.

3. What type of database is the most appropriate for the data we are collecting?
    - Option# 1: MongoDB
    - Option #2: Google Firebase
    - Option #3: PostgreSQL
    - Option #4: Amazon RDS

Choice: Google Firebase

Justification: In the end, we decided to go with Google Firebase because we felt like not only did it provide us with enough storage for the likes of our user data, but it also came with the added feature of being able to set up secure authentication with the use of Google's 'Firebase Security Rules.' In addition, we chose it over PostgreSQL because we felt that while it did provide more features for data management and extraction, it would ultimately be too complex and unnecessary for us to learn and implement. Finally, while Amazon RDS also provided similar services to Google Firebase, we were turned away by the fact that the service would very quickly present monetary costs.

4. What method should we use to implement map functionality?
    - Option #1: Google Maps API
    - Option #2: Waze Transport SDK
    - Option #3: Positionstack API

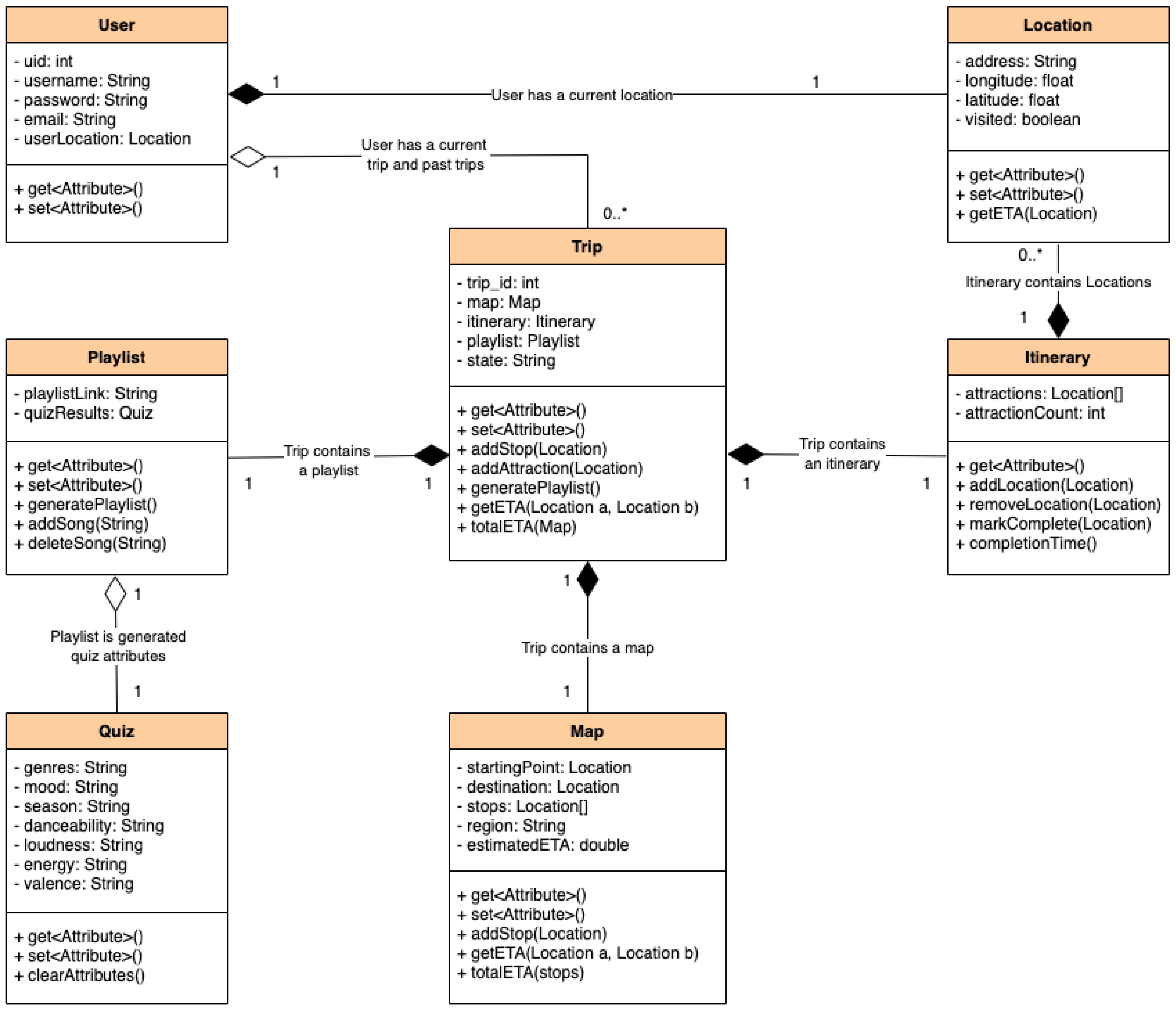
Choice: Google Maps API

Justification: To begin, we mainly decided to stick with APIs and SDKs as listed above over other alternatives such as HTML Geolocation API because we recognized the need to embed a snapshot of the user's actual location at a given time. To that end, out of all of these options, Google Maps API seemed like the best option both in terms of tutorials and features. As for tutorials, Google provides great documentation on how to integrate it with web-based applications. In terms of features, it would eliminate the need for us to keep track of and constantly update the user's estimated time of arrival, as the map component would already be capable of doing so on its own.

Design Details

Class Design

A class design represents an abstraction of one or several classes in the system's implementation. Our class design is based on an understanding of the types of objects in our domain and how they will be represented in our application. Our classes relate to the main features of our application (i.e. creating the itinerary, building the playlist, making the final trip, etc.), while the fields and methods in each dictate how we will go about implementing our main features.



Descriptions of Classes and Interaction between Classes

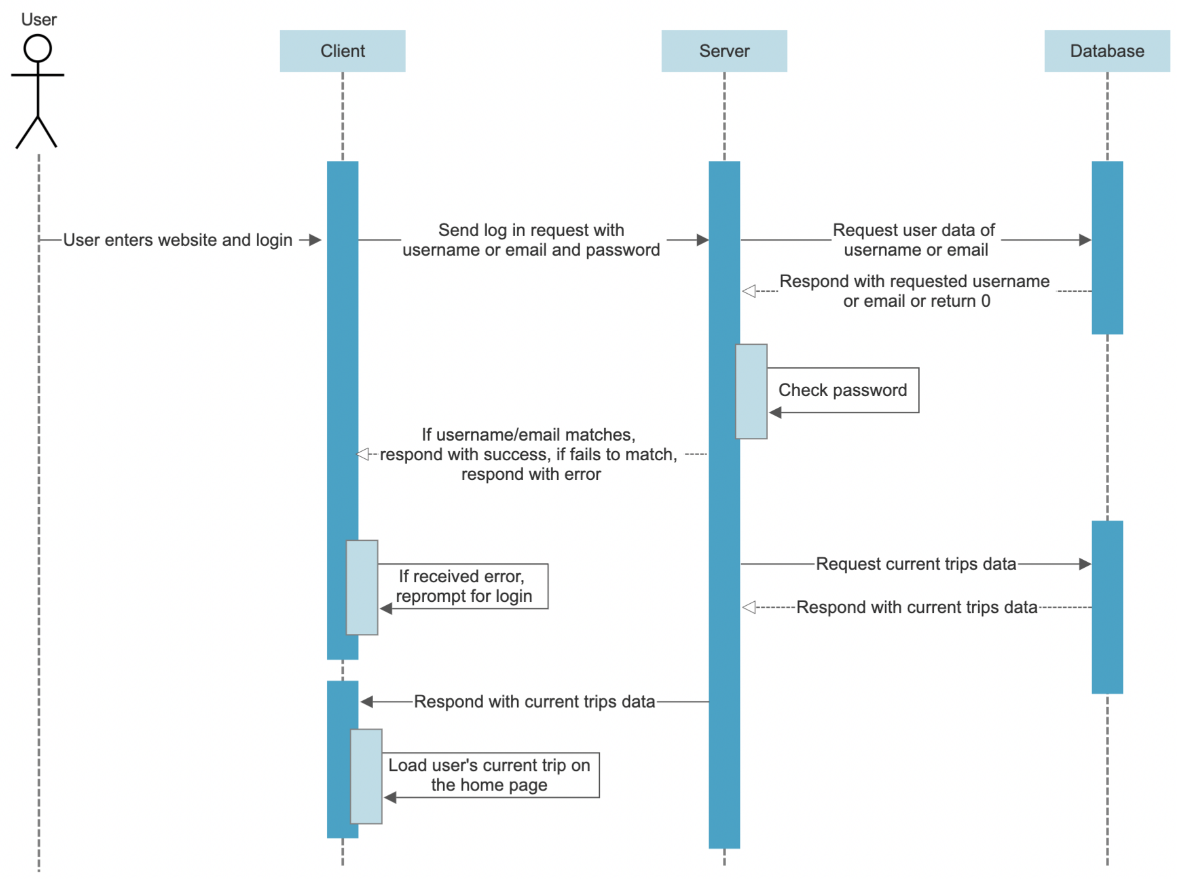
- User
    - A user object is created when a person registers for a Wander account.
    - When a user registers an account, a unique user ID is assigned.
    - When registering an account, a user enters a username which is used as another unique identifier or a login key.
    - A user inputs an email that is used as a login key or a unique identifier.
    - A password is used when a user signs into their account for authentication purposes.
- Trip
    - When a user requests a new trip, a Trip object is created.
    - Each Trip has a trip ID as a unique identifier.
    - Each Trip object has an associated Map object containing the information about the starting point, destination, and stops of the trip.
    - Each Trip object contains an Itinerary object that holds the information about the different itinerary events and attractions.
    - Each Trip object contains a Playlist object that holds the information about the generated playlist.
    - A Trip object can have the state of "current" or "past" to indicate whether a trip is in the history or is currently active.
    - There are either only one or zero trips that are classified with the "current" status at any given time.
- Playlist
    - A Playlist object is created within a Trip object to maintain attributes of the custom playlist that is generated based on the user's preferences.
    - Attributes like the quiz results, mood, and season can be recorded within a Playlist object.
    - Playlist attributes along with details from the Map object like the region of a Trip are used in the playlist generation algorithm.
    - A Playlist object contains a link to the Spotify playlist.

- Quiz
    - A Quiz object is created when the user takes the playlist generation quiz
    - Each Quiz object contains attributes like genres, mood, season, danceability, loudness, energy, and valence.
    - A Quiz object’s attributes are a factor in the playlist generation algorithm.
- Map
    - A Map object is created within a Trip object to maintain a detailed view of the trip details.
    - Each Map object contains essential attributes like the starting point and destination of a trip through Location objects
    - Each Map may have multiple stops between the starting point and destination represented by Location objects.
    - A region attribute may be populated to enhance the playlist generation algorithm.
- Itinerary
    - An Itinerary object is created within a Trip object to maintain attributes of the custom itinerary of the user.
    - Contains multiple Itinerary attractions represented by Location objects.
- Location
    - Location objects are used within the Map and Itinerary classes to keep track of important stops on the user’s trip and itinerary.
    - Location objects contain the address and coordinates of attractions, locations, and stops.
    - Each Location object records whether it has been already visited.
    - A location object is used to track the estimated location of the user.

Sequence Diagram

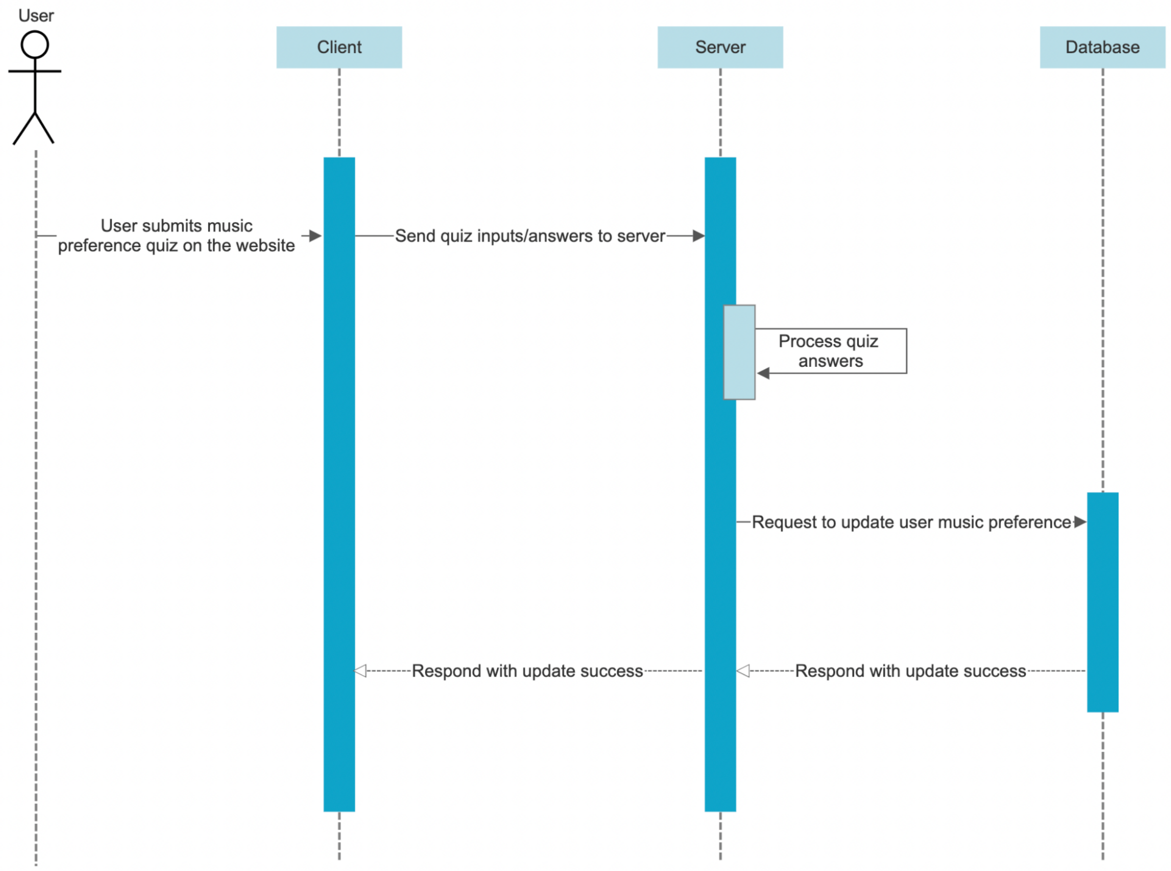
1. Sequence of Events When Users Log In

When a user enters the website and tries to log in, the first event that occurs is that the client sends a login request with the username/email and password input that the user has entered to the server where the server will then request the user data of username or email from the database. Then, the database would either respond to the request with the requested username and email or return a 0 if it is not found. The server then checks the password then returns the username/email if it was a success then request the current trips data in order to load the user's current trip on the homepage for the user on the client. However, if there was an error, reprompt the log in for the user.


User
Client
Server
Database
User enters website and login
Send log in request with username or email and password
Request user data of username or email
Respond with requested username or email or return 0
Check password
If username/email matches, respond with success, if fails to match, respond with error
If received error, reprompt for login
Request current trips data
Respond with current trips data
Respond with current trips data
Load user's current trip on the home page

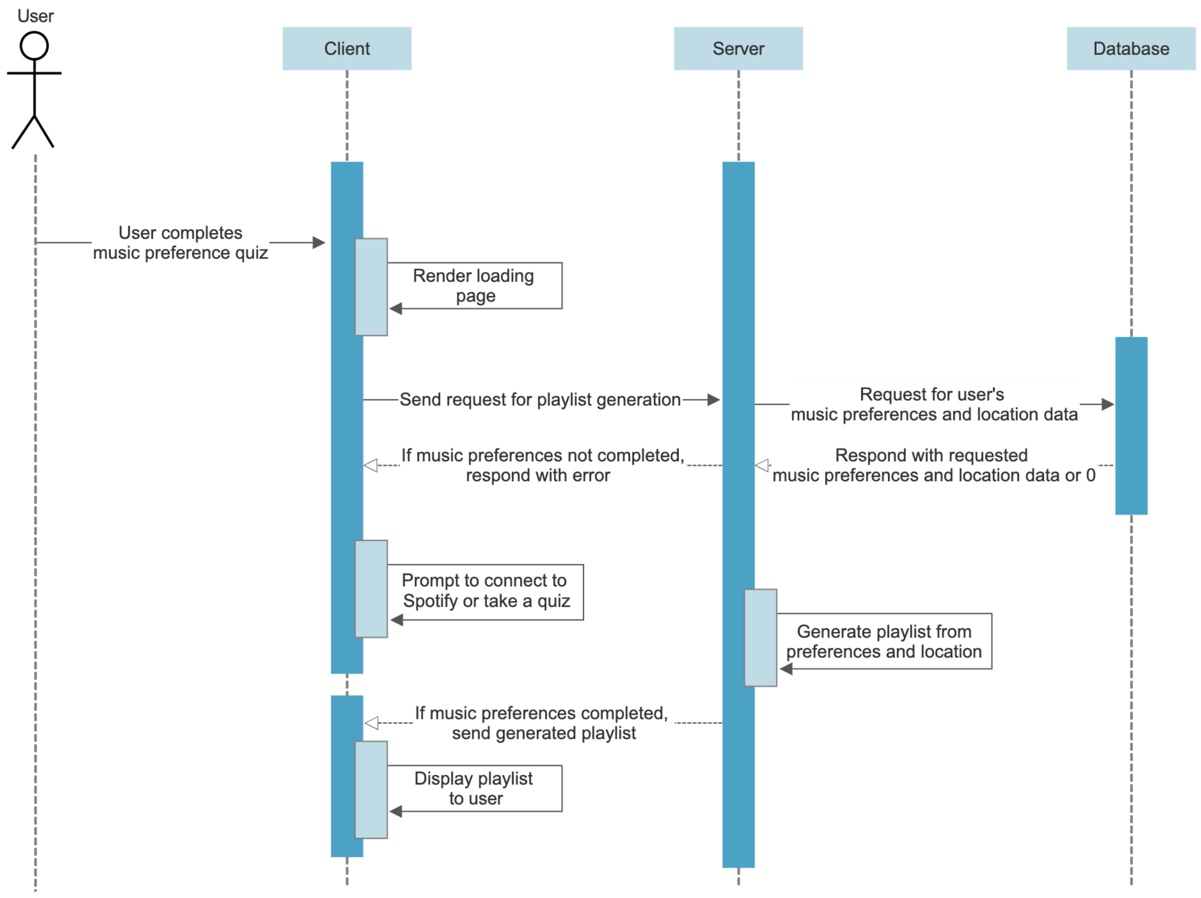
2. Sequence of Events When Users Takes a Quiz to Set Music Preferences

When each user initiates a new trip, they are prompted with a quiz that is used to generate a customized playlist quiz geared towards the user's unique preferences. After the user submits their quiz, the server processes the input and requests an update to the database to change the characteristics of the playlist generation.


User
Client
Server
Database
User submits music preference quiz on the website
Send quiz inputs/answers to server
Process quiz answers
Request to update user music preference
Respond with update success
Respond with update success

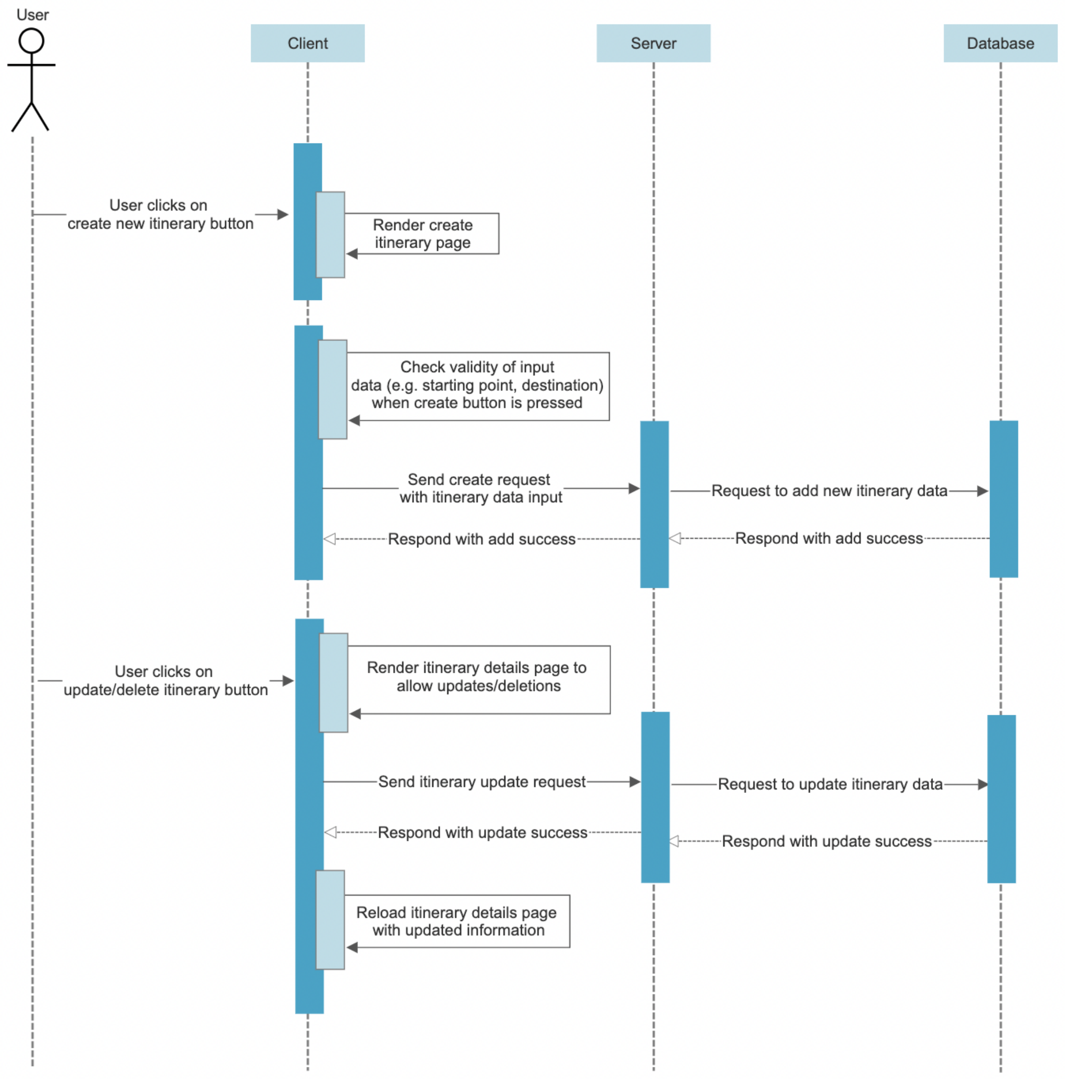
3. Sequence of Events When Users Want to Generate a New Playlist

This sequence begins initially when the user finishes the playlist generation quiz. The application will then display a loading page while a request is sent to the server for playlist generation. During this process, the server will obtain user preference and location data from the database. The server will then run our appropriate scripts to generate the playlist and finally send it back to the client, which will display it to the user.


User
Client
Server
Database
User completes music preference quiz
Render loading page
Send request for playlist generation
Request for user's music preferences and location data
If music preferences not completed, respond with error
Respond with requested music preferences and location data or 0
Prompt to connect to Spotify or take a quiz
Generate playlist from preferences and location
If music preferences completed, send generated playlist
Display playlist to user

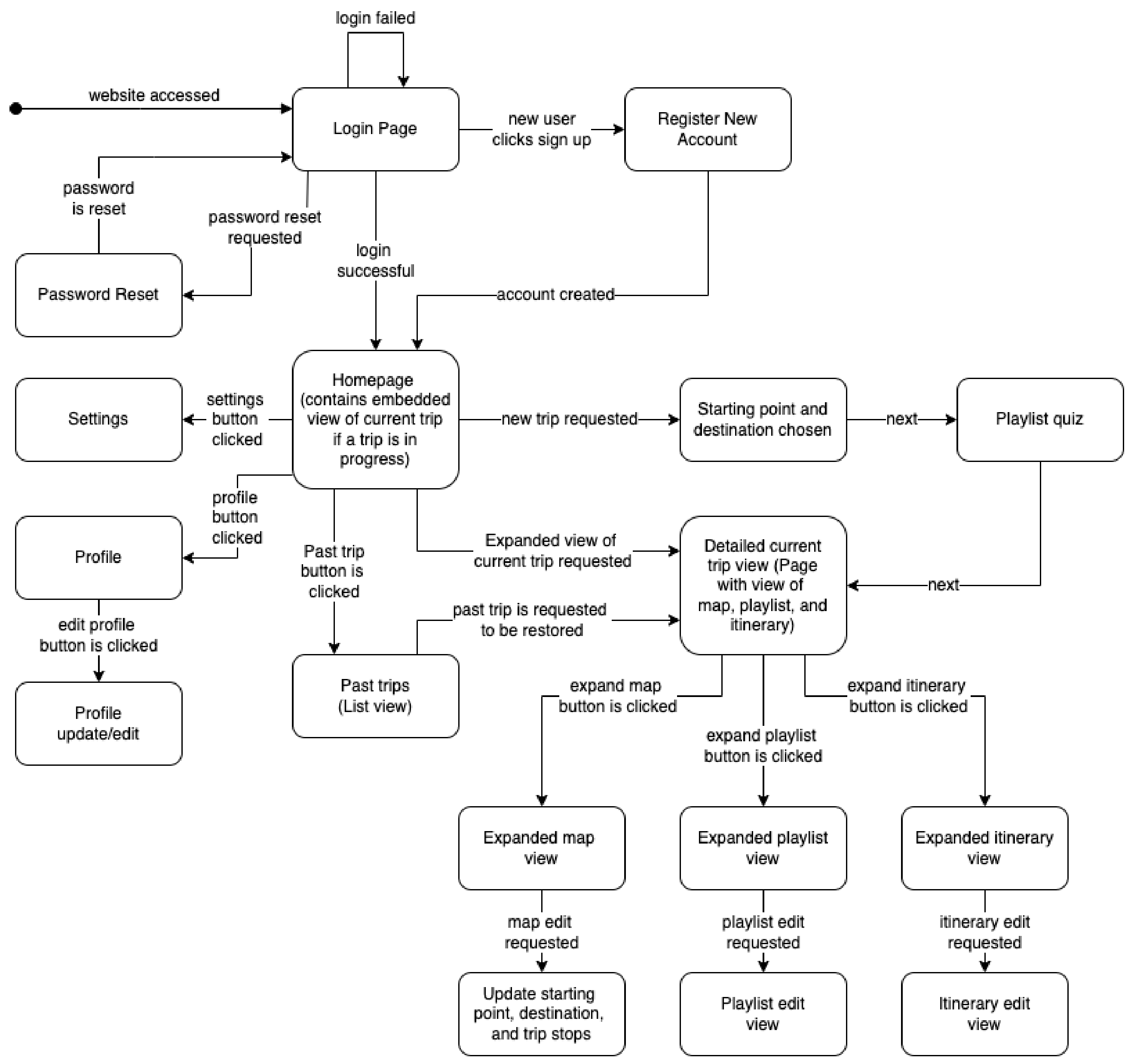
4. Sequence of Events When Users Create or Update an Itinerary

Users have the ability to create a new itinerary and update an existing itinerary. When a user chooses to create a new itinerary, a new page pops up which allows them to input their desired source and destination location. The server and database work together to add the new data and respond with the status of the addition (i.e. if it was a success or not). When a user chooses to update their itinerary, a different page is created which allows them to edit their trip details. Similar to the new itinerary creation, the server and database attempt to update the data based on the validity of the input and return whether the update was successful.


User
Client
Server
Database
User clicks on create new itinerary button
Render create itinerary page
Check validity of input data (e.g. starting point, destination) when create button is pressed
Send create request with itinerary data input
Request to add new itinerary data
Respond with add success
Respond with add success
User clicks on update/delete itinerary button
Render itinerary details page to allow updates/deletions
Send itinerary update request
Request to update itinerary data
Respond with update success
Respond with update success
Reload itinerary details page with updated information

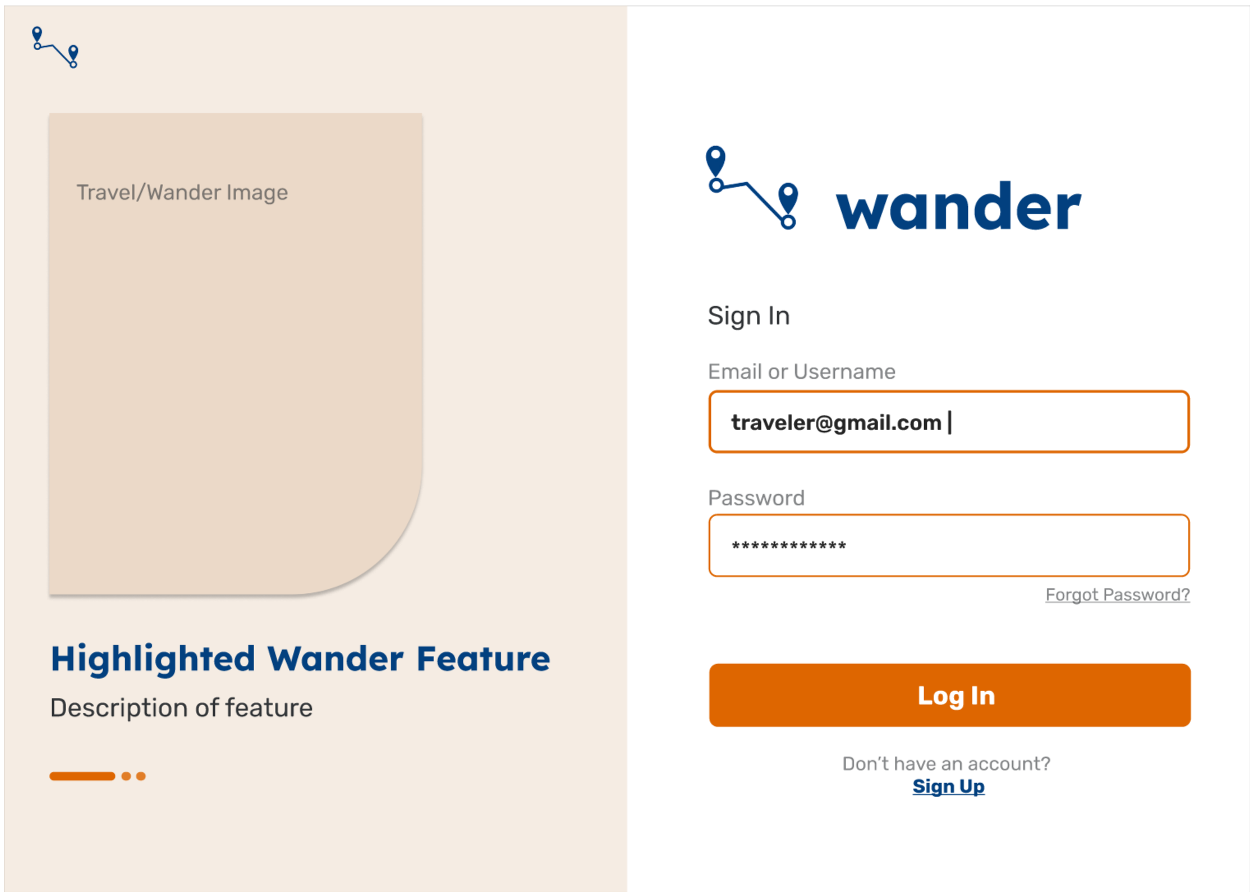
Navigation Flow Map

The diagram below illustrates the navigation capabilities of Wander. When users first access the webpage, they will be prompted with a login screen. If the user requires a password reset or is registering a new account, they can access those pages through the initial login screen. The website's main functionality is based on modularized "trips" containing the travel information, playlist, and itinerary, with each trip object's state being classified as either current or past. The current trip can be accessed from the homepage or on the detailed current trip page, where each component can be viewed in an expanded form and can be updated. When the user requests a new trip, they are prompted with a page asking for the trip details and a playlist quiz to gauge the user's listening preferences. The playlist and itinerary are then generated and displayed with an interactive map in the detailed current trip view.


login failed
website accessed
Login Page
new user clicks sign up
Register New Account
password is reset
password reset requested
login successful
Password Reset
account created
Homepage (contains embedded view of current trip if a trip is in progress)
settings button clicked
Settings
new trip requested
Starting point and destination chosen
next
Playlist quiz
profile button clicked
Profile
Past trip button is clicked
Expanded view of current trip requested
Detailed current trip view (Page with view of map, playlist, and itinerary)
next
past trip is requested to be restored
edit profile button is clicked
Profile update/edit
Past trips (List view)
expand map button is clicked
expand playlist button is clicked
expand itinerary button is clicked
Expanded map view
Expanded playlist view
Expanded itinerary view
map edit requested
playlist edit requested
itinerary edit requested
Update starting point, destination, and trip stops
Playlist edit view
Itinerary edit view

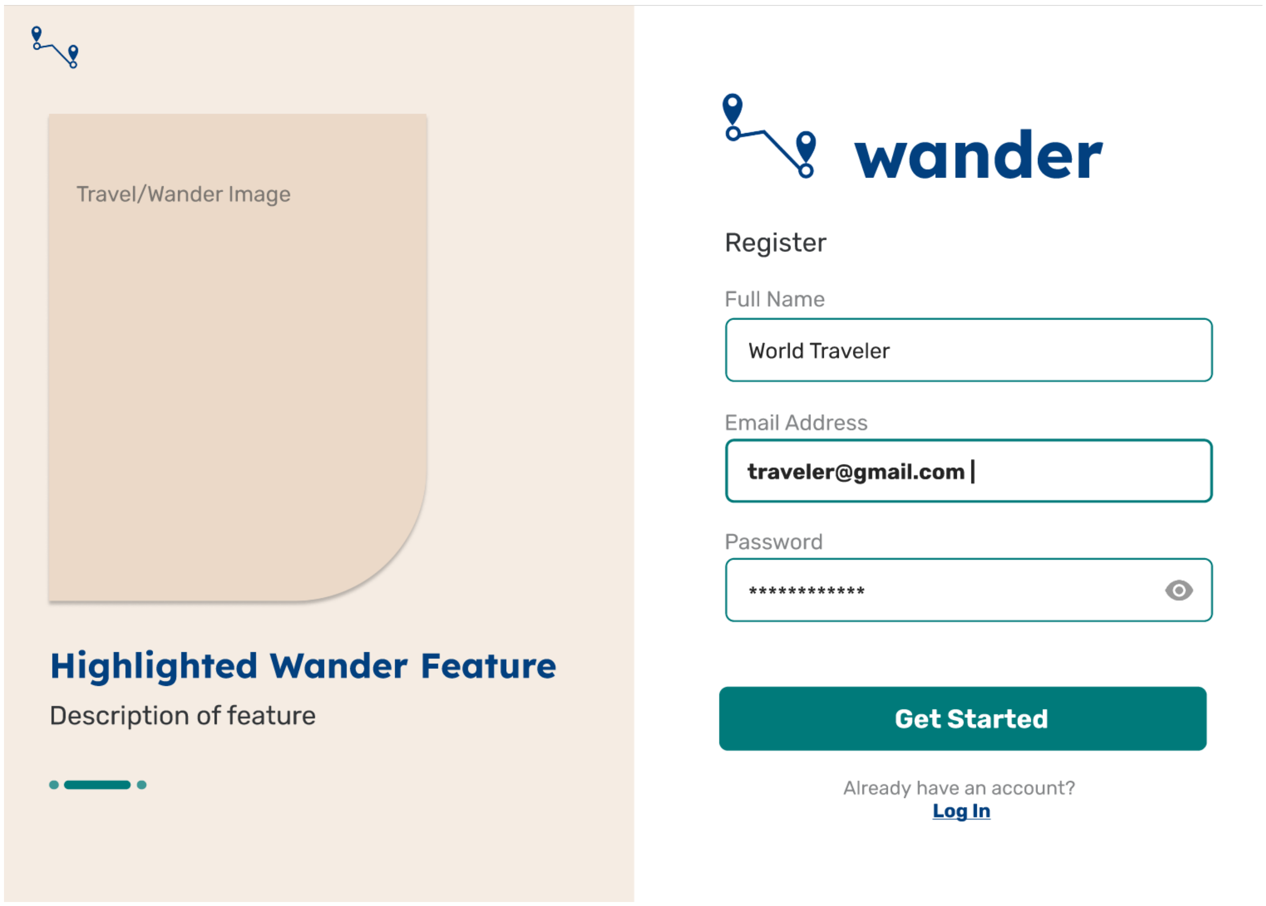
UI Mockups

Login Page



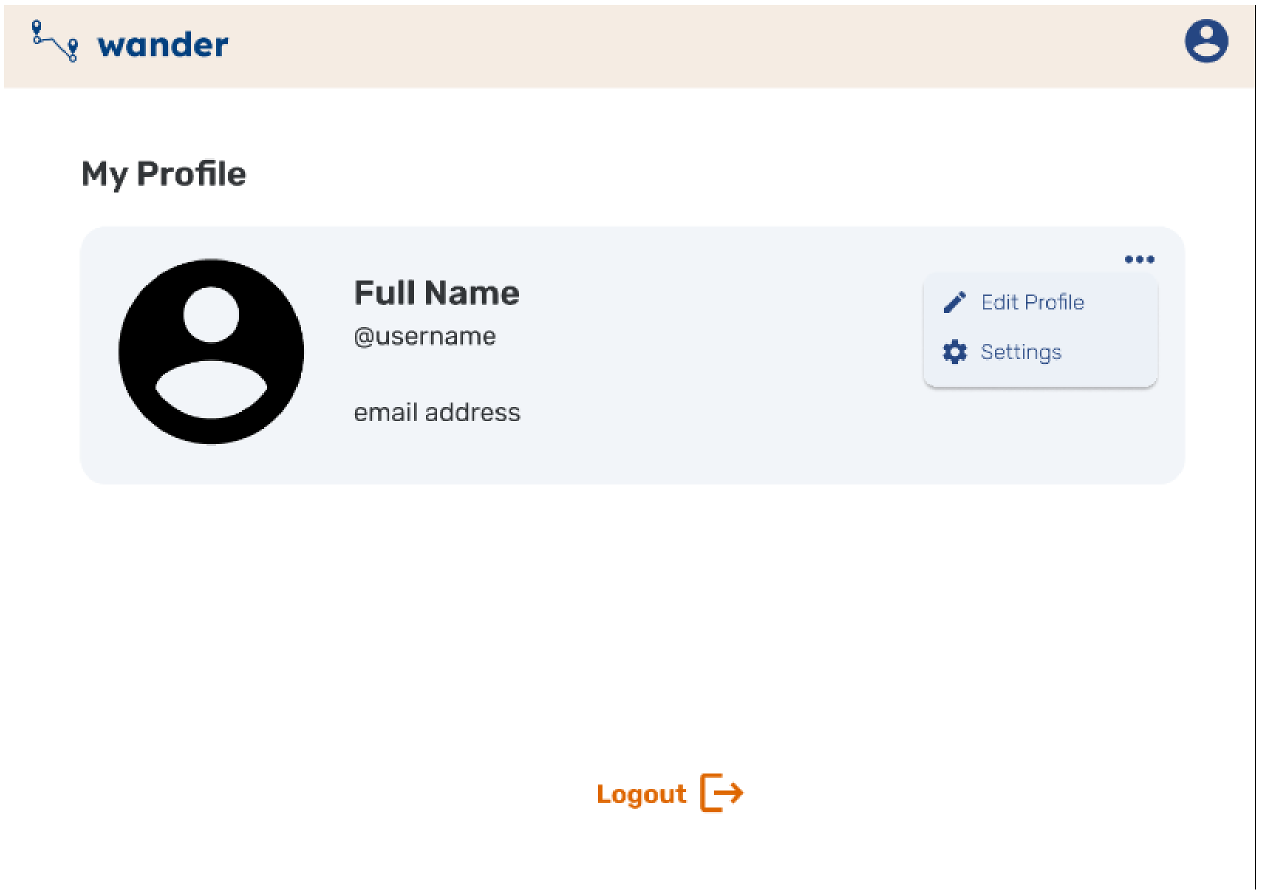
The login page predominantly features the Wander logo and a highlighted feature of the website. Users can log in with their email or username and password on the right hand side of the screen. There is a button for the user to reset their password if necessary. There is also an option for new users to sign up for an account.

Registration



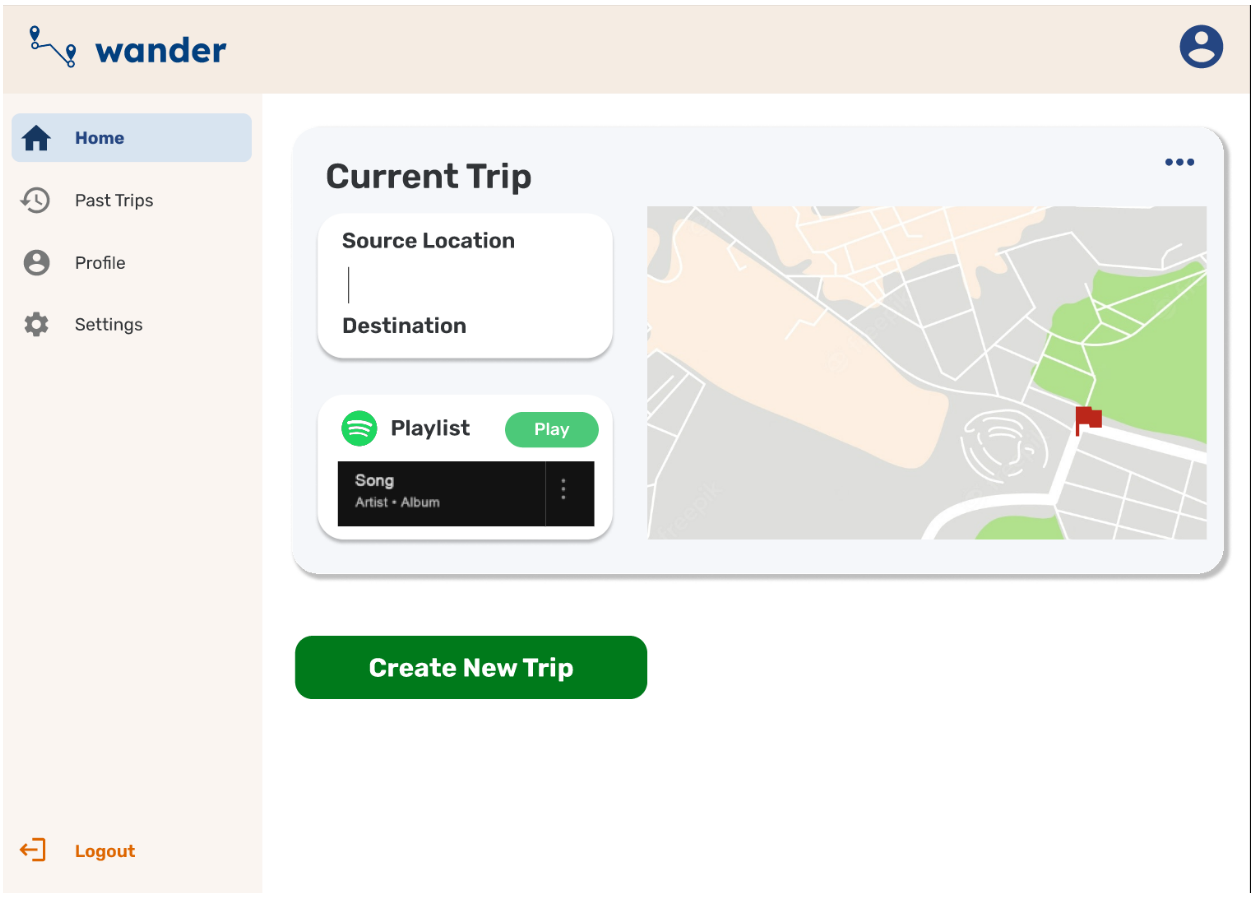
New users are directed to the registration page. To create an account, the user must input their full name, an email address, and a password. There is an option to return to the Login page if necessary. When users click the ‘Get Started’ button, the Wander database verifies the user’s input and stores the new account information.

Profile Page



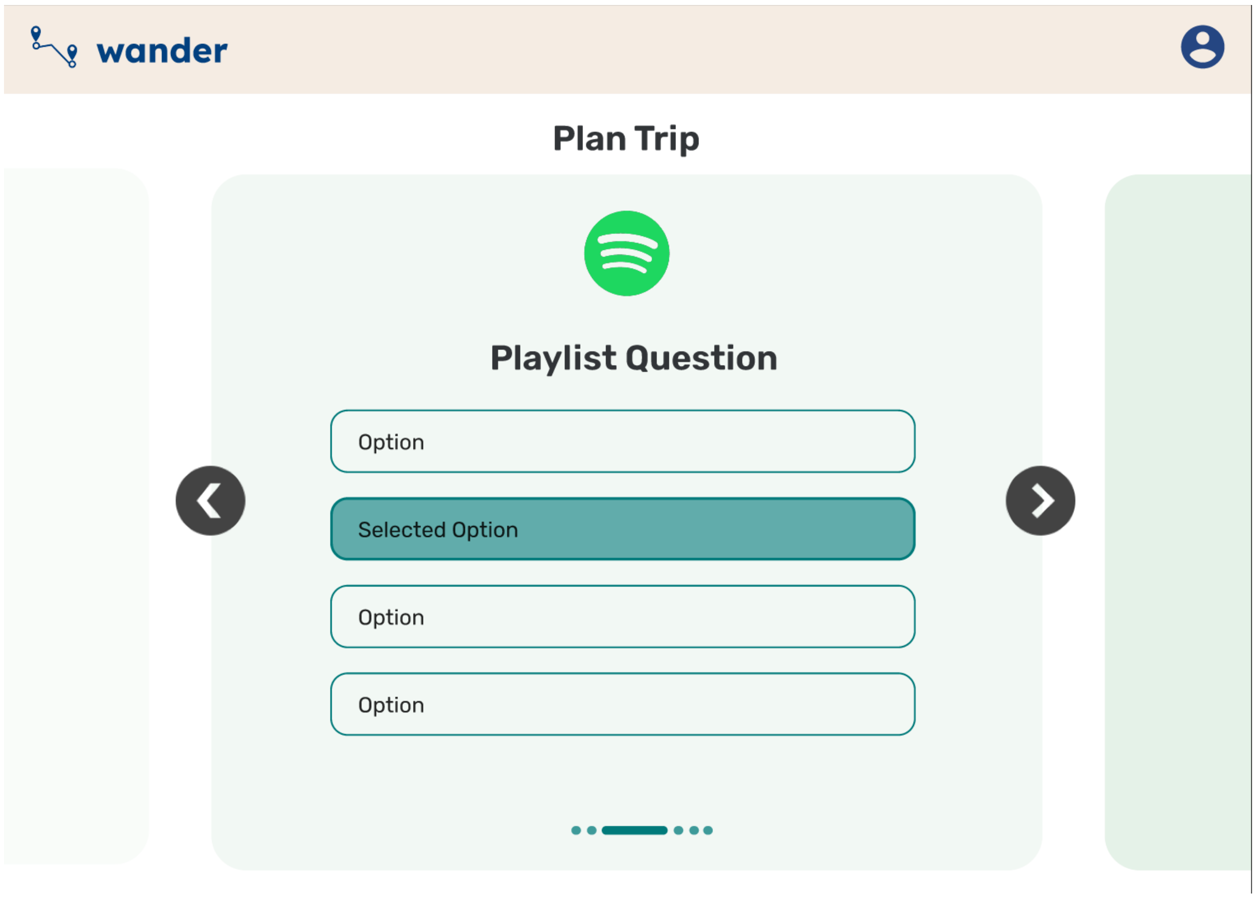
After a user creates an account, their profile page displays their name, username, profile picture, and email address. The profile page also has the option for users to edit their profile or go to the settings page to edit more complex details. The ‘Logout’ button allows users to sign out.

Home Page



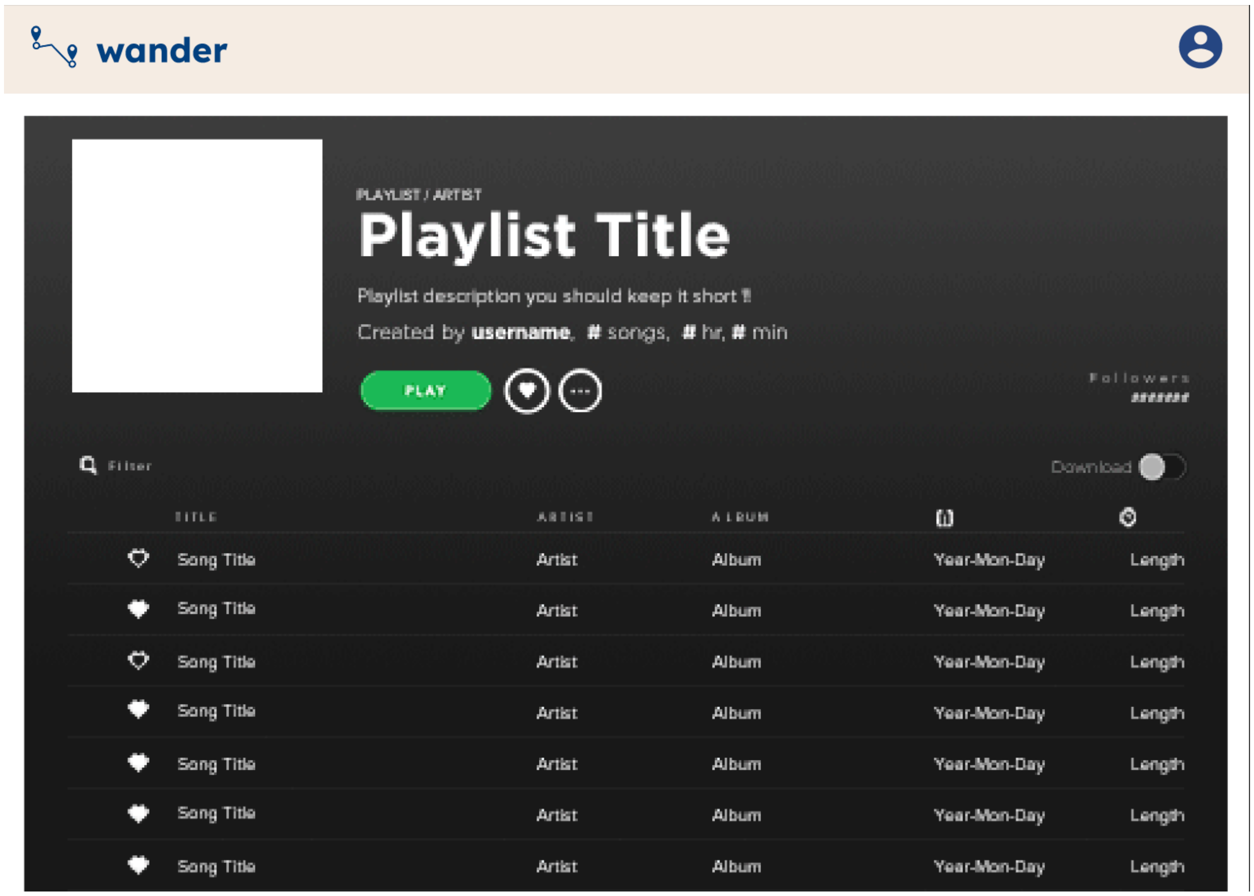
The home page contains links to all of the other features in Wander. If a user has a trip in progress, the home page displays an embedded view of the current trip. The 'Create New Trip' button sends users to the quiz page where they can begin planning their next trip. The side menu contains links to a user's list of travel history, their profile, and settings. The user can also log out. When each of the side menu links are clicked, a new page is opened so the user can access more functionality.

Playlist Creation Quiz



When planning a trip, users take a quiz. One of the quiz questions is about the user's music preferences. The quiz page above is a card from a carousel menu of questions. Users can go back and forth between quiz questions and select their answers. A progress bar at the bottom keeps users informed of how many questions are remaining.

Custom Playlist


wander

PLAYLIST / ARTIST

Playlist Title

Playlist description you should keep it short !!

Created by **username**, # songs, # hr, # min

PLAY

Followers
#######

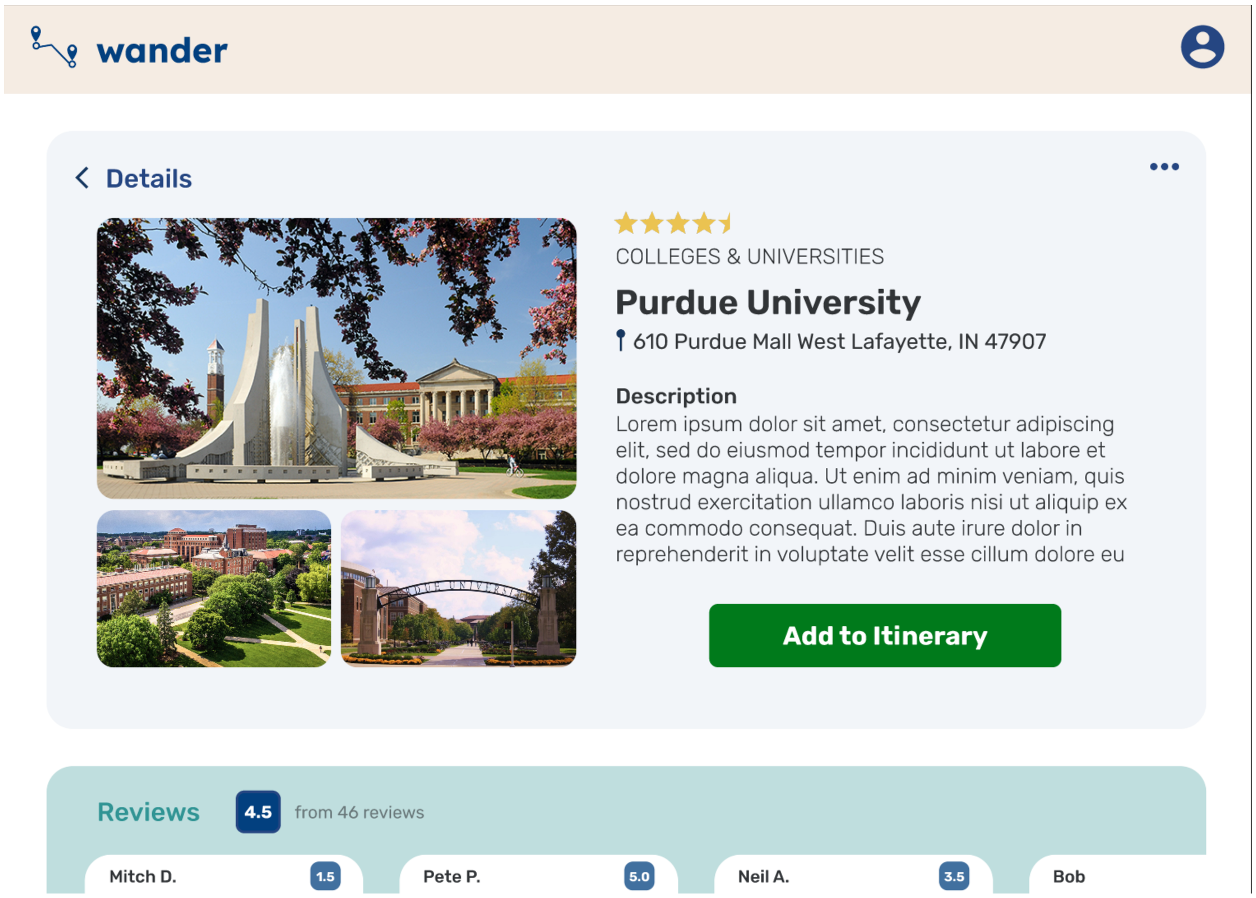
Filter

Download

	TITLE	ARTIST	ALBUM		
♡	Song Title	Artist	Album	Year-Mon-Day	Length
♥	Song Title	Artist	Album	Year-Mon-Day	Length
♡	Song Title	Artist	Album	Year-Mon-Day	Length
♥	Song Title	Artist	Album	Year-Mon-Day	Length
♥	Song Title	Artist	Album	Year-Mon-Day	Length
♥	Song Title	Artist	Album	Year-Mon-Day	Length
♥	Song Title	Artist	Album	Year-Mon-Day	Length

This page gives users a full view of their personalized playlist. All Spotify functionality is included: editing the playlist, changing the playlist title, changing the playlist description, playing the playlist, etc.

Itinerary Creation



wander

Details

★★★★½

COLLEGES & UNIVERSITIES

Purdue University

610 Purdue Mall West Lafayette, IN 47907

Description

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu

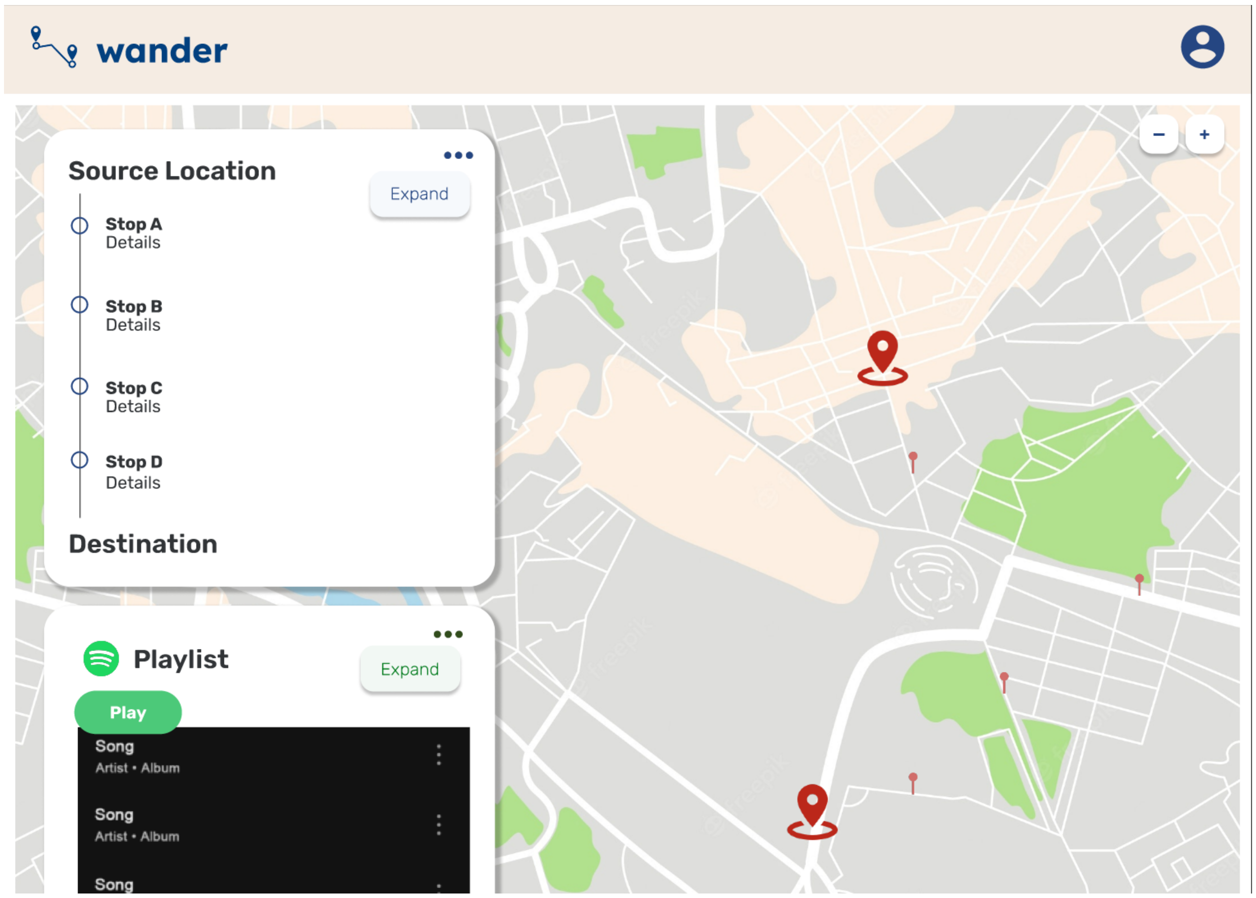
Add to Itinerary

Reviews 4.5 from 46 reviews

Mitch D.	1.5
Pete P.	5.0
Neil A.	3.5
Bob	

Users are allowed to edit their generated itineraries. This is an example page of a potential location that a user might add to their itinerary. The user can read a description and reviews of the location. The “Add to Itinerary” button will add the location to the already created travel plan.

Current Trip View



After a user creates their itinerary and takes a quiz, they are directed to a page highlighting their trip details. The page primarily focuses on a map with the various locations from their itinerary emphasized on it. The left side of the page has the user's custom itinerary and playlist. A button exists in every section (i.e. itinerary, playlist, and map) that the user can click on which will redirect them to an expanded version of what they clicked on.