Documentation

Backend

/server/routes

[Functions called in /server/controller/authenticationController or /server/controller/listingController respectively.]

authenticationRoutes.js

loginUser - **POST** /api/user/login

- Verify the user’s information matches the email and password of a user in the database.
- Parameters:
  - email: String
  - password: String
- Example Request:
  ```
  {
          "email": "user_test3@carns.dev",
          "password": "Password123!"
  }
  ```
- Responses:
  - 200 - Successful user login (matches user in database)
  - 400 - Error, improper field input/user not found in database.

signupUser - **POST** /api/user/signup

- Create a user in the database given their email, password and user type.
- Parameters:
  - email: String
  - password: String
  - userType: String
- Example Request:
  ```
  {
          "email": "user_test3@carns.dev",
          "password": "Password123!",
          "userType": "buyer"
  }
  ```

- Responses:
    - 200 - Successful creation of the user in the database.
    - 400 - Error, improper field input/failed to create the user in the database.

listingRoutes.js

postListing - **POST** /api/listing/post

- Create a listing in the database.
- Parameters:
    - vendorID: String
    - listingName: String
    - listingDetails: listingDetails
- Example Request:

```
{
        "vendorID": "123",
        "listingName": "Test Listing 3",
        "listingDetails": {
                "listingDescription": "Test description.",
                "isBuy": true,
                "vehicleType": "Car",
                "listingSalePrice": 30000
        }
}
```

- Responses:
    - 200 - Successful creation of the listing.
    - 400 - Error creating the listing, improper field input/failed to create a listing in the database.

viewListings - **GET** /api/listing/view

- Get all the listings in the database (in descending creation order).
- Parameters/Example Request:
    - None
- Reponses:
    - 200 - All the listing JSON objects are returned in descending creation order.
    - 400 - Error when returning all of the listings in the database.

/server/models

authenticationModel.js
Schema for user login, authentication and user profiles.

- email: String, required
    - Email of the user for login.
- password: String, required
    - Password for user login.
- userType: String, required
    - Type of user: “buyer” or “vendor”

listingModel.js
Schema for the vendor listing.

- vendorID: String, required
- listingName: String, required
- listingDetails: listingDetails, required
    - listingDescription: String
        - Optional description of the listing.
    - isBuy: Boolean
        - If true, the listing is for Buyers (single sale) and if false, the listing is for renters.
    - vehicleType: String
        - Type of vehicle, eg. “Car”, “Bike” etc.
    - listingSalePrice: Number
        - Sale price (rent price per day if isBuy is false)

Frontend

/client/src

App.js
This file contains the mainline logic of the frontend. It displays the navigation bar and landing page and contains routes to other pages.

Methods: None

/client/src/components

Hero.js
This file contains the main hero component displayed on the landing page. It displays a graphic and contains button that directs the user to the sign up page

Methods: none

Navbar.js
This file contains the navigation bar component which is displayed in every screen and allows users to easily switch between screens.

Methods: none

/client/src/components/pages

LandingPage.js
This file contains the combined components for the landing page. It currently contains the Hero component.

Methods: none

Buy Listing.js
This file represents the buy listing page where users buy cars. The page will call viewListing with isBuy=true and display all listings in the database in descending order or creation.

Methods:
- viewListing - retrieves all listing objects from the database in descending order of creation

RentListing.js
This file represents the rent listing page where users rent cars. The page will call viewListing with isBuy=false and display all listings in the database in descending order or creation.

Methods:
- viewListing - retrieves all listing objects from the database in descending order of creation

SignUp.js
This file contains the signup component to allow users to create an account. It contains a name field, email field, password/confirm password fields, and a checkbox to set the account type. After fields have been entered, signupUser will be called to create the user object in the database.

Methods

- signupUser - this method will create a user object in the database with the provided fields

SignIn.js
This file contains the sign in component to allow users with accounts to access the features of the site. It contains an email and password field. loginUser will be called to authenticate the user credentials.

Methods

- loginUser - this method will authenticate the fields provided by the user to sign them in.