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
  - profile: profileSchema
  -
- Example Request:

```
{
    "email": "user_test@carns.dev",
    "password": "Password123!",
    "userType": "buyer"
```

```
        "profile": {
            "name": "Foo Bar",
            "phone_number": 1234567890
        }
    }
```

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

getProfile - **GET** /api/user/profile/:id

- Get the profile details of a user given their ID.
- Parameters/Example Request:
    - None
- Responses:
    - 200 - Successfully got the user's profile, returns profile subschema JSON.
    - 404 - Error, not a valid ID or there is no such user.

editProfile - **PATCH** /api/user/profile/:id

- Edit a user's profile details given their ID.
- Parameters/Example Request:
    - None
- Responses:
    - 200 - Successfully edit the user's profile, returns the edited profile subschema JSON.
    - 404 - Error, not a valid ID or there is no such user.

deleteUser - **DELETE** /api/user/:id

- Deletes a user from the database given their ID.
- Parameters/Example Request:
    - None
- Responses:
    - 200 - Successfully deleted the user, returns a JSON of the deleted user.
    - 404 - Error, not a valid ID or there is no such user.

authenticated - **GET** /api/user/authenticated

- Authenticates the user's session with a cookie.
- Parameters/Example Request:
    - None
- Responses:

    - 200 - Successfully created a session with the user.
    - 401 - Error, user not authenticated properly.

logout - **GET** /api/user/logout

- Clears the user's session and deletes the cookie.
- Parameters/Example Request:
    - None
- Responses:
    - 200 - Successfully deleted the user session.

listingRoutes.js

postBuyListing - **POST** /api/listing/post-buy

- Create a listing for sale in the database.
- Parameters:
    - listingName: String
    - isBuy: Boolean
    - buyListingDetails: listingDetails
- Example Request:

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

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

postRentListing - **POST** /api/listing/post-rent

- Create a listing for rent in the database.
- Parameters:
    - vendorID: String

    - listingName: String
    - isBuy: Boolean
    - rentListingDetails: listingDetails
- Example Request:

```
{
        "listingName": "Test Listing 3",
        "isBuy": false
        "rentListingDetails": {
                "listingDescription": "Test description.",
                "vehicleType": "Car",
                "rentPrice": 30000,
                "availabilityStart": "10/11/2022",
                "availabilityEnd": "10/28/2022",
        }
}
```

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

viewBuyListings - **GET** /api/listing/view-buy

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

viewActiveBuyListings - **GET** /api/listing/viewActiveBuyListings/:id

- Get all the active buy listings for sale in the database from a specific vendor (in descending creation order).
- Parameters/Example Request:
    - id: String
- Responses:
    - 200 - All the listing JSON objects are returned in descending creation order.
    - 400 - Error when returning all of the listings in the database.

viewPastBuyListings - **GET** /api/listing/viewPastBuyListings/:id

- Get all the inactive buy listings for sale in the database from a specific vendor (in descending creation order).
- Parameters/Example Request:
    - id: String
- Responses:
    - 200 - All the listing JSON objects are returned in descending creation order.
    - 400 - Error when returning all of the listings in the database.

viewRentListings - **GET** /api/listing/view-rent

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

viewActiveRentListings - **GET** /api/listing/viewActiveRentListings

- Get all the active rent listings for sale in the database from a specific vendor (in descending creation order).
- Parameters/Example Request:
    - None
- Responses:
    - 200 - All the listing JSON objects are returned in descending creation order.
    - 400 - Error when returning all of the listings in the database.
    -

viewExpiredRentListings - **GET** /api/listing/view-rent-expired

- Get all the inactive rent listings for sale in the database from a specific vendor (in descending creation order).
- Parameters/Example Request:
    - None
- Responses:
    - 200 - All the listing JSON objects are returned in descending creation order.
    - 400 - Error when returning all of the listings in the database.

getDetailBuy - **GET** /api/listing/view-detail-buy/:id

- Get a specific buy listing from the database.
- Parameters:
    - id: String
- Example Request:
    - None
- Responses:
    - 200 - Get the buy listing details (returns the subschema JSON)
    - 404 - Error, not a valid listing ID, no such listing or missing the buy listing details.

getDetailRent - **GET** /api/listing/view-detail-rent/:id

- Get a specific rent listing from the database.
- Parameters:
    - id: String
- Example Request:
    - None
- Responses:
    - 200 - Get the rent listing details (returns the subschema JSON)
    - 404 - Error, not a valid listing ID, no such listing or missing the rent listing details.

updateBuyListing - **PUT** /api/listing/update-buy/:id

- Edit a buy listing from the database given the listing ID.
- Parameters:
    - id: String
    - newListingDescription: String
    - newSalePrice: Number
    - newLocation: String
- Example Request:

```
{
        "newSalePrice": 213,
        “newLocation”: “Toronto”
}
```

- Responses:
    - 200 - Successful listing update, returns the updated listing JSON.
    - 404 - Error when editing, couldn’t find the listing or invalid ID.

updateRentListing - **PUT** /api/listing/update-rent/:id

- Edit a rent listing from the database given the listing ID.
- Parameters:
    - id: String
    - newListingDescription: String
    - newRentPrice: Number
    - newLocation: String
- Example Request:

```
{
        "newRentPrice": 7987,
        "newListingDescription": "This is a car."
}
```

- Responses:
    - 200 - Successful listing update, returns the updated listing JSON.
    - 404 - Error when editing, couldn't find the listing or invalid ID.

deleteListing - **DELETE** /api/listing/:id

- Delete a listing from the database given its ID.
- Parameters/Example Request:
    - None
- Responses:
    - 200 - Successful deletion, returns the deleted listing JSON.
    - 404 - Error when deleting, couldn't find the listing or invalid ID.

addRentListingDates - **PUT** /api/listing/add-dates/:id

- Add dates that are rented out to a listing. Updates the unavailable dates for the entire listing and the specific listing booking object for a user.
- Parameters:
    - id: String
    - dates: [Date]
- Example Request:

```
{
        "dates": ["2022-10-21", "2022-10-22"]
}
```

- Responses:

    - 200 - Successful addition of dates to allUnavailableDates and customer's respective booking object.
    - 400 - Failed to add dates: user is not logged in, id is not for a rent listing or date is already taken.
    - 404 - Error, no such user/no such listing.

removeRentListingDates - **PUT** /api/listing/add-dates/:id

- Removes dates that are rented out to a listing. Updates the unavailable dates for the entire listing and the specific listing booking object for a user.
- Parameters:
    - id: String
    - dates: [Date]
- Example Request:

```
{
        "dates": ["2022-10-21"]
}
```

- Responses:
    - 200 - Successful removal of dates from allUnavailableDates and customer's respective booking object.
    - 400 - Failed to remove dates: user is not logged in or id is not for a rent listing
    - 404 - Error, no such user/no such listing.

transactionRoutes.js

logTransaction- **POST** /api/transaction/log

- Create a transaction log/receipt in the database.
- Parameters:
    - customerID: String
    - listingID: Boolean
    - transactionAmount: int
    - dates : [Date] (Optional)
- Example Request:

```
{
        "customerID": "634c62b908bfca7e9f3e6720",
        "listingID": "6351f5e690d442ecbbbc1687",
        "transactionAmount": 300,
        "dates": ["2022-11-20", “2022-11-21”]
}
```

- Responses:
    - 200 - Successful creation of the transaction.
    - 400 - Error creating the transaction, improper field input/failed to add to the database.

logTransaction- **GET** /api/transaction/log

- Gets a list of all of the user's past purchases for a customer.
- Parameters:
  N/A
- Responses:
    - 200 - Fetching of the transactions.
    - 400 - Error fetching the transactions.

/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"
- profile: profileSchema, required
    - name: String
        - Name of user.
    - phone_number: String
        - Phone number of user.

listingModel.js

Schema for the vendor listing.

- vendorID: String, required
- listingName: String, required
- isBuy: Boolean
    - If true, the listing is for Buyers (single sale) and if false, the listing is for renters.
- buyListingDetails: buyListingDetails
    - listingDescription: String
        - Optional description of the listing.

        - vehicleType: String
            - Type of vehicle, eg. “Car”, “Bike” etc.
        - salePrice: Number
            - Sale price
        - location: String
- rentListingDetails: rentListingDetails
    - listingDescription: String
        - Optional description of the listing.
    - vehicleType: String
        - Type of vehicle, eg. “Car”, “Bike” etc.
    - rentPrice: Number
        - rent price per day
    - location: String
    - availabilityStart: Date
        - Start date for the period the vehicle is available for rent
    - availabilityEnd: Date
        - End date for the period the vehicle is available for rent
    - allUnavailableDates: [Date]
        - All dates for which the rent listing is unavailable.
    - booking:
        - customerID: String
        - dates: [Date]
            - All the dates the specific customer has rented out

transactionModel.js
Schema for the transaction (purchase history) model.

- customerID: String, required
    - The customer’s ID from the database associated with the transaction.
- vendorID: String, required
    - The vendor’s ID from the database associated with the transaction.
- transactionAmount: Number, required
    - The cost of the vehicle if it’s a buy listing, or the cost * the amount of days for a rent listing.
- dates: [Date], default: undefined (Optional)
    - Dates rented out for the rent listing transaction.

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:

- useAuth: this method authenticates the user and contains the current user's data. It also contains api calls from authenticationRoutes.js

Navbar.js
This file contains the navigation bar component which is displayed in every screen and allows users to easily switch between screens. It has a different view when the user is logged in or logged out.

Methods:

- useAuth: this method authenticates the user and contains the current user's data. It also contains api calls from authenticationRoutes.js

Profile.js
This file contains the user profile component. The component displays the user's profile information in a neatly formatted way, and contains a button which links to the edit profile page.

Methods:

- useAuth: this method authenticates the user and contains the current user's data. It also contains api calls from authenticationRoutes.js

ProfileSideBar.js

This file contains the user profile sidebar component. The component displays the navigation options in the user profile, it displays different options depending on whether the logged in user is a vendor or a buyer.

Methods:

- useAuth: this method authenticates the user and contains the current user's data. It also contains api calls from authenticationRoutes.js

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. login will be called to authenticate the user credentials.

Methods:

- login: this method will authenticate the fields provided by the user to sign them in

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, signup will be called to create the user object in the database.

Methods:

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

EditProfileForm.js

This file contains the edit profile form component to allow users to make changes to some of their profile information. Once users fill out the new fields and click save changes, editprofile is called and their information is updated in the database.

Methods:

- useAuth: this method authenticates the user and contains the current user's data. It also contains api calls from authenticationRoutes.js
- edit: this method updates the name, email, and phone number fields of the current user

ListingForm.js

This file contains the form for creating a buy vehicle listing. Vendors can fill out fields for vehicles and after all fields have been entered, a listing object will be created in the database.

Methods:

- None

RentListingForm.js

This file contains the form for creating a rent vehicle listing. Vendors can fill out fields for vehicles and after all fields have been entered, a listing object will be created in the database.

Methods:

- none

PastSellList.js

This file represents the component that displays the vendor's past sell list

Methods:

- None

PastSellSingle.js

This file represents the listing card component that is displayed in the vendor's past sell list

Methods:

- none

ListingForm.js

This file contains the form for creating a vehicle listing. Vendors can fill out fields for vehicles and after all fields have been entered, a listing object will be created in the database.

Methods:

- None

RentListingForm.js
This file contains the form for creating a rent vehicle listing. Vendors can fill out fields for vehicles and after all fields have been entered, a listing object will be created in the database.

Methods:
- none

ActiveSellList.js
This file represents list of active sell listings created by the vendor

Methods:
- fetchBuyListings: this method gets all buy listings from the database

ActiveRentList.js
This file represents list of active rent listings created by the vendor

Methods:
- fetchBuyListings: this method gets all rent listings from the database

ActiveHistorySingle.js
This file represents a listing card component in the vendor's active listing screen

Methods:
- useNotification: this method displays notification messages when listing is successfully updated or if there is an error
- handleDelete: This method deletes the listing from the database
- newpost: this method updates the listing fields

BuyDetail.js
This file contains the details of a specific vehicle listing. It displays a description, dates, vendor id, etc.

Methods:
- useAuth: this method authenticates the user and contains the current user's data. It also contains api calls from authenticationRoutes.js

RentDetail.js

This file contains the details of a specific rent vehicle listing. It displays a description, dates, vendor id, etc.

Methods:

- useAuth: this method authenticates the user and contains the current user's data. It also contains api calls from authenticationRoutes.js

ContactInfo.js

This file contains the contact details of a vendor in a card format

Methods:

- useAuth: this method authenticates the user and contains the current user's data. It also contains api calls from authenticationRoutes.js

BuyListing.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:

- fetchBuyListings - retrieves all buy 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:

- fetchRentListings - retrieves all rent listing objects from the database in descending order of creation

BuyCheckout.js

This file represents the checkout page for buy listings.

Methods:

- fetchBuyDetail- retrieves the information of the listing
- transaction/log - Creates and stores the transaction object

RentCheckout.js
This file represents the checkout page for rent listings.

Methods:

- fetchRentDetail- retrieves the information of the listing
- transaction/log - Creates and stores the transaction object

BuyHistory.js
This file represents purchase history of a buyer

Methods:

- fetchListings- retrieves all the past purchases of the user

PastHistorySingle.js
This file represents the card object for the vendor's past sell list

Methods: none

PastSellList.js
This file represents the vendor's past selling history, it contains individual card objects for each vehicle sold

Methods:

- fetchBuyListings - This method calls the listing/view-past-buy/_id API to display the vendor's past listings

VendorBookingTrans.js
This file represents the vendor's past rent history, it contains individual card objects for each rent listing

Methods:

- fetchBuyListings - This method calls the listing/view-active-rent/_id API to display the vendor's past listings

/client/src/components/pages

CreateListingPage.js
This file represents the create buy listing page. It contains the listing form component

Methods: none

CreateRentListingPage.js
This file represents the create rent listing page. It contains the rent listing form component

Methods: none

EditProfilePage.js
This file represents the edit profile page. It contains the edit profile form component

Methods: none

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

Methods: none

SignUpPage.js
This represents the page for user Sign In, it contains the sign up component

Methods: none

SignInPage.js
This represents the page for user Sign In, it contains the sign in component

Methods: none

BuyListingPage.js
This represents the page for Buy listings.

Methods: none

RentListingPage.js
This represents the page for Rent listings.

Methods: none

BuyDetails.js
This represents the listing details page of a particular buy listing

Methods: none

VendorHistoryPage.js
This represents the vendor's listing history page

Methods: none

VendorListingPage.js
This represents the vendor's active listing page for sell listings.

Methods: none

VendorListingPageRent.js
This represents the vendor's active listing page for rent listings.

Methods: none

BuyCheckoutPage.js
This represents the checkout page for buy listings.

Methods: none

RentCheckoutPage.js
This represents the checkout page for rent listings.

Methods: none

BuyerHistoryPage.js
This represents the buyer's past purchases history page

Methods: none

RentDetailsPage.js
This represents the details page for specific vehicles that are up for rent

Methods: none

VendorBookingHistoryPage.js
This represents the vendor's past renting history

Methods: none