# React.js Exam – Train System

Use the web server from the resources, install all its dependencies and run it via node. It will respond on port 5000.

## Create a React application

Create a React application and prepare the initial project structure. You will be creating a **system for booking train tickets**. **Anonymous** users can view the **catalog** (all trains) and their **details**, but in order to **add** tickets to the **cart**, **checkout** and view their **profile** (owned tickets) they have to **login**/**register** into the system. You may use and extend the provided **HTML mock**, or write your own HTML, if you think it will save time – application appearance will **not** be evaluated.

## Add Authentication

Make sure to **validate** all form values on the front-end and provide appropriate **error messages** to the user. The **name** and **e-mail** fields must not be empty and the **password** must be at least 4 characters long.

### User Registration (Sign Up)

To register a user, you need to send a **POST** request to the server on "/**auth**/**signup**" with "**name**", "**email**" and "**password**" data (sent as JSON):

|  |  |  |
| --- | --- | --- |
|  | POST http://localhost:5000/auth/signup | |
| Headers | Content-Type: application/json |
| Request body | {  "name": "new\_userd",  "email": "pass1d23@abv.bg",  "password": "New User"  } |
| Success | {  "success": true,  "message": "You have successfully signed up! Now you should be able to log in."  } |
| Error response | {  "success": false,  "message": "E-mail already exists!"  } |

### User Login

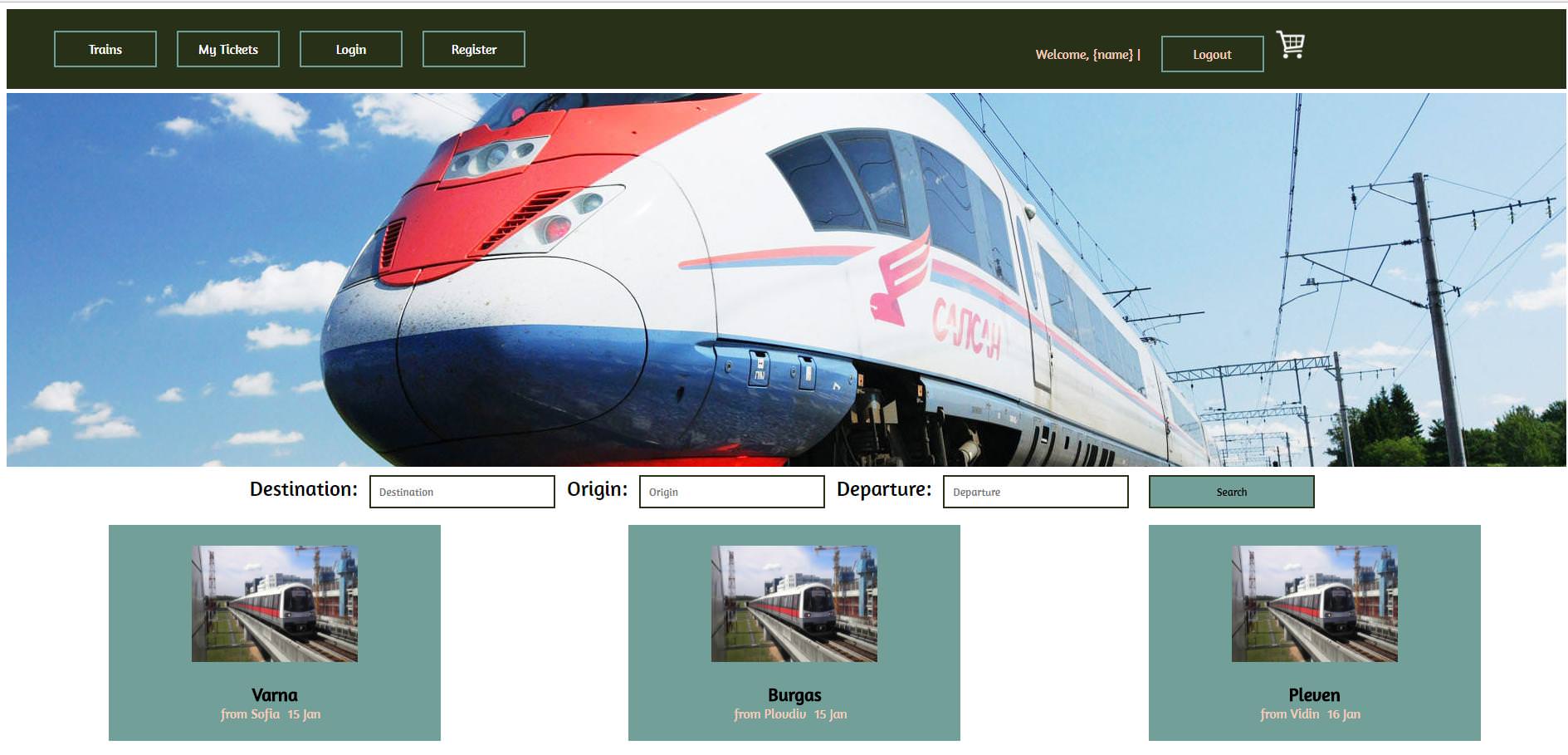
To login a user, you need to send a **POST** request to the server on "/**auth**/**login**" with "**email**" and "**password**" data (sent as JSON). You need to **save the user token** in your application state. Make sure you validate everything on the client application.

|  |  |  |
| --- | --- | --- |
|  | POST http://localhost:5000/auth/login | |
| Headers | Content-Type: application/json |
| Request body | {  "email": "pass1d23@abv.bg",  "password": "New User"  } |
| Success | {  "success": true,  "message": "You have successfully logged in!",  "token": "eyJhbGciOiJ9...",  "user": {  "name": "new\_userd"  }  } |
| Error response | {  "success": false,  "message": "Incorrect email or password"  } |

Successful login returns a “**token**” which is later used to authenticate the CRUD operations.

## Catalog View (25 pts)

Add a view where all available **train trips** for today are listed. Each train has **origin** station, **destination** station, departure **time,** trip duration and **tickets**. Most trips have both first and second class seats, but some **only** have second class! You may use the same placeholder image for all thumbnails (as provided in the resources) or don’t show any thumbnail.



|  |  |
| --- | --- |
| **GET http://localhost:5000/trips** | |
| Success | **[**  **{**  **"origin": "Sofia",**  **"destination": "Plovdiv",**  **"time": "14:30",**  **"arrives": "17:00",**  **"duration": "2:30",**  **"tickets": {**  **"firstClass": 11.3,**  **"secondClass": 9**  **},**  **"\_id": "9fa23c73-ba0e-d271-f30e-44d66aa54016"**  **},**  **{**  **"origin": "Sofia",**  **"destination": "Plovdiv",**  **"time": "21:00",**  **"arrives": "23:15",**  **"duration": "2:15",**  **"tickets": {**  **"secondClass": 9**  **},**  **"\_id": "ba955f48-fa36-16e3-c818-c0192b72e540"**  **},**  ***// The rest of the trips***  **]** |

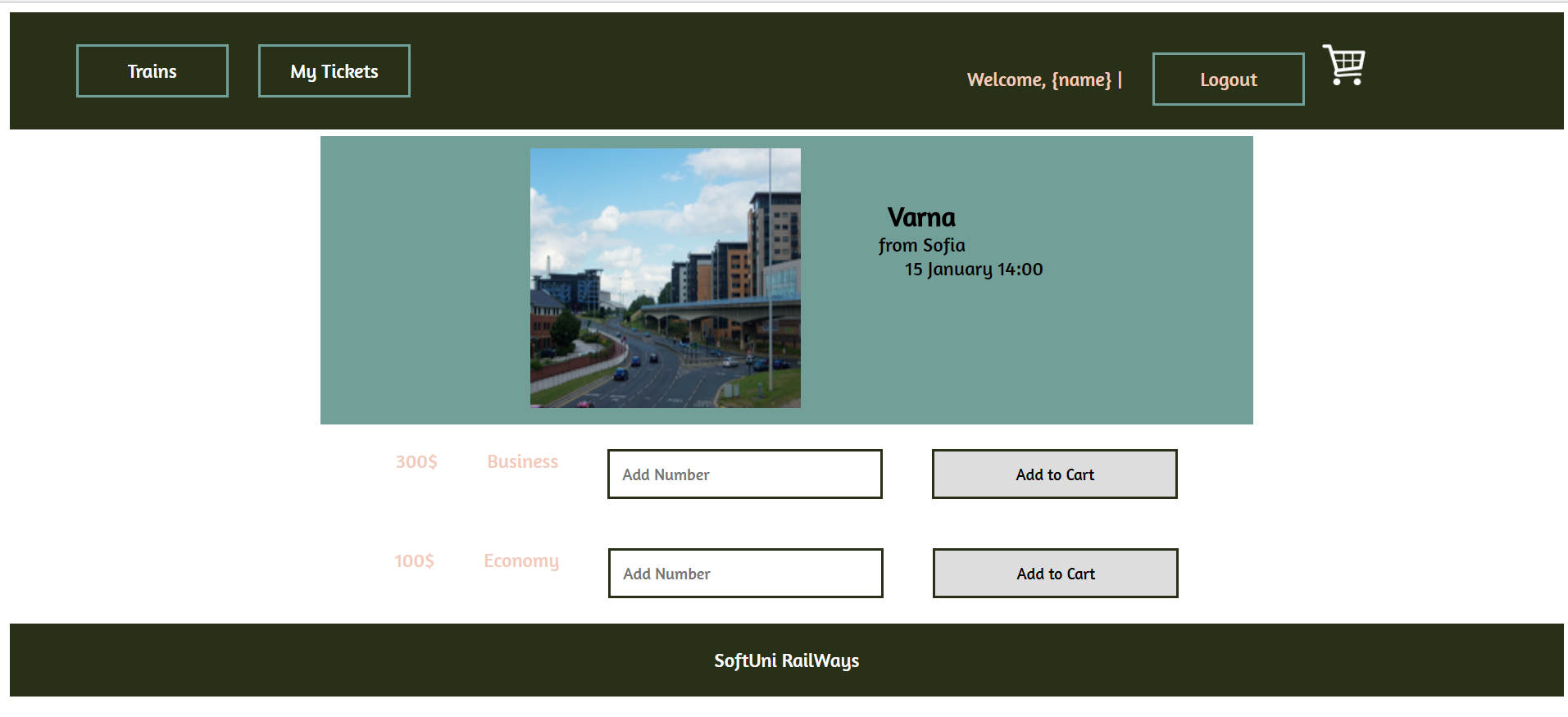
## Search Results (15 pts)

When the user uses the search option, display the results in a view, identical to the catalog. Send a GET request with query parameters **origin**, **destination** and **date** with format dd-mm-yy.

|  |  |
| --- | --- |
| **GET http://localhost:5000/search?origin=Sofia&destination=Plovdiv&date=07-01-18** | |
| Success | **[**  **{**  **"origin": "Sofia",**  **"destination": "Plovdiv",**  **"time": "14:30",**  **"arrives": "17:00",**  **"duration": "2:30",**  **"tickets": {**  **"firstClass": 11.3,**  **"secondClass": 9**  **},**  **"\_id": "9fa23c73-ba0e-d271-f30e-44d66aa54016"**  **},**  ***// The rest of the trips***  **]** |
| Error 401 | **{**  **"error": "Missing query parameter: destination"**  **}** |

## Trip Details View (20 pts)

Clicking on each **individual** train trip leads to a **details** page where **information** about the **trip** is shown (destination, origin, date, time, image), a list of **seats** for the train (each seat has **price** and **type**) and also a **form** for adding tickets (each ticket has a **number** of **seats** for the given train). This route is only for authenticated users so you need to send a header with "**Authorization**" name and value "**bearer {token}**" in order to pass the authentication checks. Make sure your React application **redirects to the login page**, if the user tries to open the trip details page and they’re not logged in.



|  |  |
| --- | --- |
| **GET http://localhost:5000/trips/1424c9fa-e759-cbcf-bca8-4edb8b0f9366** | |
| Request headers | **Authorization: bearer <authToken>** |
| Success | **{**  **"origin": "Sofia",**  **"destination": "Burgas",**  **"time": "06:40",**  **"arrives": "13:05",**  **"duration": "6:25",**  **"tickets": {**  **"firstClass": 26,**  **"secondClass": 20.9**  **},**  **"\_id": "1424c9fa-e759-cbcf-bca8-4edb8b0f9366"**  **}** |

## Add Ticket to Cart (15 pts)

Clicking **[Add to Cart]** sends a **POST** request with the **id** of the **train trip**, the **number** of tickets that the user has **typed,** and the selected **class**. This route is only for authenticated users so you need to send a header with "**Authorization**" name and value "**bearer {token}**" in order to pass the authentication checks. Make sure your React application **redirects to the login page**, if the user tries to open the hotel details page and he's not logged in.



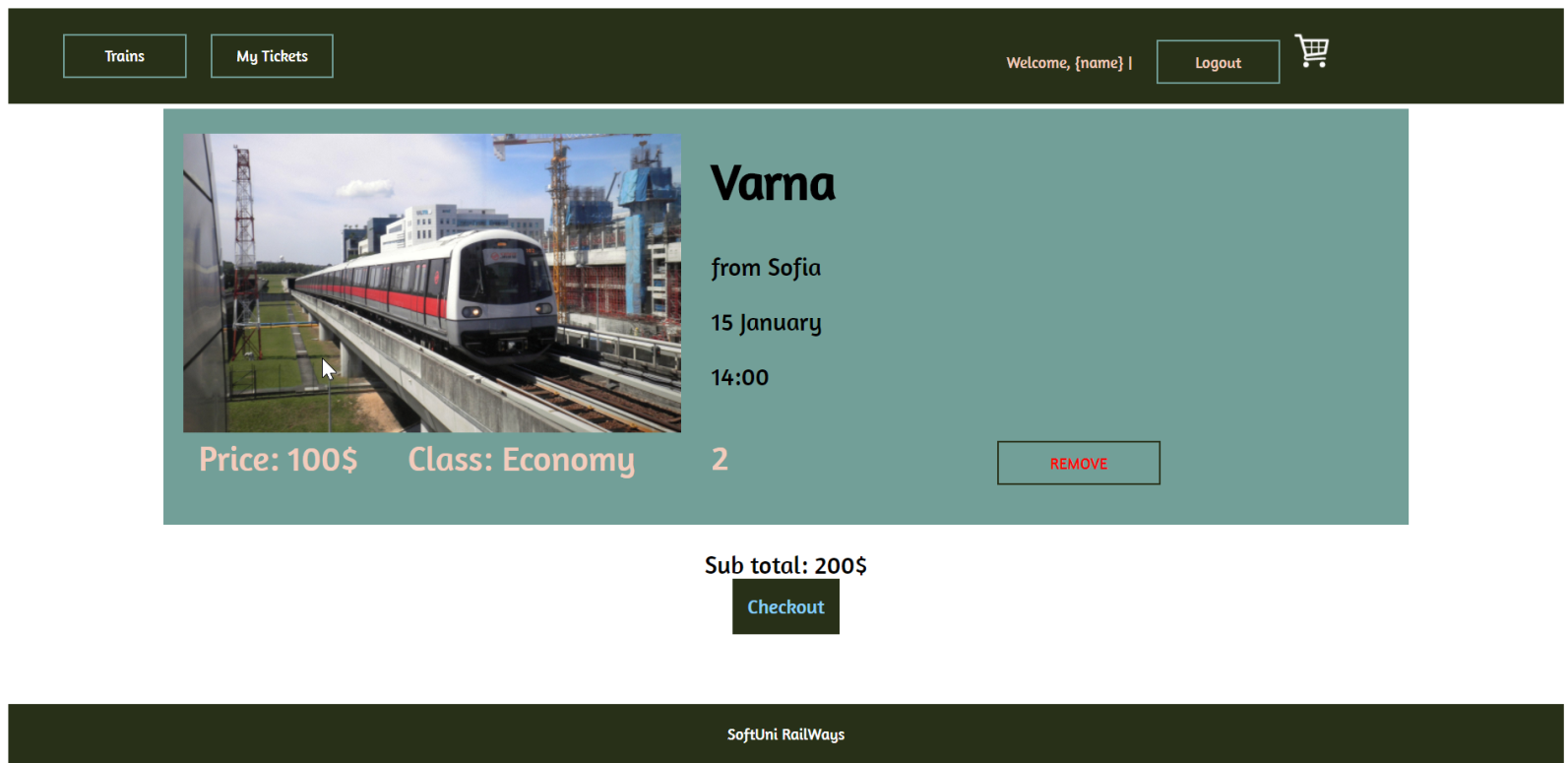
|  |  |
| --- | --- |
| POST http://localhost:5000/cart | |
| Headers | Content-Type: application/json  Authorization: bearer <authToken> |
| Request body | {  "tripId": 1424c9fa-e759-cbcf-bca8-4edb8b0f9366,  "date": "07-01-18",  "class": "secondClass",  "count": 2  } |
| Success | {  "success": true,  "message": "Tickets added to cart",  "ticket": {  "tripId": "1424c9fa-e759-cbcf-bca8-4edb8b0f9366",  "date": "07-01-18",  "class": "secondClass",  "count": 2,  "\_id": "889664b5-2319-cb9a-9f79-db2fcf2720da"  }  } |

## Manage Ticket Cart (15 pts)

This route is only for authenticated users so you need to send a header with "**Authorization**" name and value "**bearer {token}**" in order to pass the authentication checks. Make sure your React application **redirects to the login page**, if the user tries to open the cart details page and they’re not logged in.

### List all Tickets added to Cart (5 pts)

Create a **Cart** view where all information about each **train trip** is listed (destination, origin, date, time, seat price, seat type and number of tickets) and also a **subtotal** of the **purchase** (sum of number of **tickets \* price** for all tickets in the cart).



|  |  |
| --- | --- |
| **GET http://localhost:5000/cart** | |
| Request headers | **Authorization: bearer <authToken>** |
| Success | **[**  **{**  **"tripId": "1424c9fa-e759-cbcf-bca8-4edb8b0f9366",**  **"origin": "Sofia",**  **"destination": "Burgas",**  **"date": "07-01-18",**  **"time": "06:40",**  **"arrives": "13:05",**  **"duration": "6:25",**  **"class": "secondClass",**  **"price": 20.9,**  **"count": 2,**  **"\_id": "03ed8bc0-898c-539f-a9aa-33ff8f2bfaf3"**  **},**  ***// The rest of the tickets***  **}** |

### Remove Tickets from the Cart (10 pts)

Clicking **[Remove]** should delete the train trip from the Cart.

|  |  |
| --- | --- |
| DELETE http://localhost:5000/cart/03ed8bc0-898c-539f-a9aa-33ff8f2bfaf3 | |
| Headers | Authorization: bearer <authToken> |
| Success | {  "success": true,  "message": "Tickets removed from cart"  } |

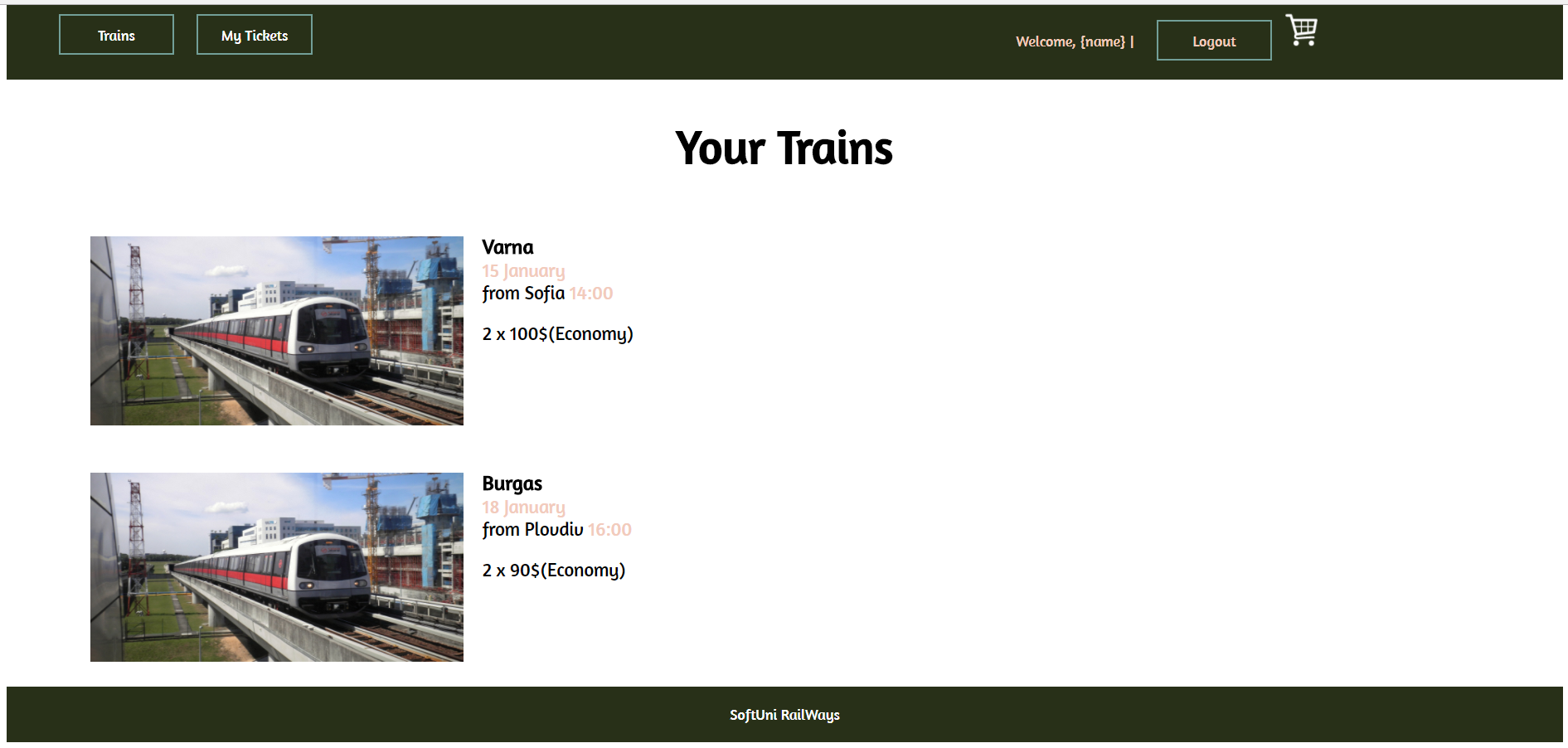
### Checkout

Clicking **[Checkout]** sends **POST** request with no body and the server will take care of all **tickets** as bought.

|  |  |
| --- | --- |
| POST http://localhost:5000/cart/checkout | |
| Headers | Authorization: bearer <authToken> |
| Success | {  "success": true,  "message": "Ticket purchase confirmed"  } |
| Error response | {  "success": false,  "message": "No tickets in cart."  } |

## Profile View (My Tickets) (10 pts)

Clicking **[My Tickets]** in the **navigation** should render a view where all **user bought tickets** are displayed (destination, origin, date, time, duration, number of tickets, price and seat class for each ticket). This route is only for authenticated users so you need to send a header with "**Authorization**" name and value "**bearer {token}**" in order to pass the authentication checks. Make sure your React application **redirects to the login page**, if the user tries to open the profile page and they’re not logged in.



|  |  |
| --- | --- |
| **GET http://localhost:5000/cart/history** | |
| Request headers | **Authorization: bearer <authToken>** |
| Success | **[**  **{**  **"user": "pesho@abv.bg",**  **"tripId": "1424c9fa-e759-cbcf-bca8-4edb8b0f9366",**  **"origin": "Sofia",**  **"destination": "Burgas",**  **"date": "07-01-18",**  **"time": "06:40",**  **"arrives": "13:05",**  **"duration": "6:25",**  **"class": "firstClass",**  **"price": 26,**  **"count": 3,**  **"\_id": "57031467-8958-4bfe-13c2-03a49deb05cd"**  **}**  ***// The rest of the tickets***  **]** |

## (Bonus) Use a State Management Library (10 points)

You may optionally store your data and state, using a state management library, such as Flux, Redux, MobX or similar. For this task to count as completed, you must store and retrieve the **balance** and **expense** information through the library. Partial points will be awarded, but chose wisely.