Auctions Website
Web Development Technologies

Georgios Nikolaou
`sdi1800134`

Nefeli Tavoulari
`sdi1800190`

Spring Semester 2022

HELLENIC REPUBLIC
National and Kapodistrian
University of Athens

Contents

1 Introduction

In this assignment we implemented an auctions website, an Ebay clone, called BidIt. In this website, one can create an auction, browse auctions and bid. The main technologies used are:

- React Framework for the frontend
- Express - Node.js for the backend (REST API)
- PostgreSQL as the database

We chose the specific technologies as they are among the most popular ones in the field, they are well documented and offer, of course, many possibilities. We created a RESTful API using HTTP requests and JSON for data interchange between frontend and backend, in order to ensure component independence and flexibility which we tested with Postman. In the following sections we will discuss in more detail all the challenges we faced and the services we implemented.

2 Installation

- sudo service postgresql start : to start the db
- cd api; npm install; npm start : to install packages and start backend
- cd ui; npm install; npm start : to install packages and start frontend

3 Database

The postgreSQL database we used consists of the following tables:

- account : used to login/register/get users/get user etc
- auction : used to create/update/delete/see auction(s) etc
- auction_category : intermediate table because of many to many relationship between auction and category
- bid : used to create bid/get bids
- category : used to create/delete/get category/categories
- message : used to send message/get inbox/sent messages
- newsletter : used to add an email

The tables' fields can be seen in detail in api/database.sql.

4 Authentication

The user can login/register in any page of the website opening the modal popup that is found in the navigation bar, in order to be able to bid or sell. When the user signs up, the password they selected is encrypted and saved in the database with the other data and a JSON Web Token is generated that is saved in the session and which is deleted when the user logs out. We tried to validate as much as we could the input of the user in the all the forms and on error we send an appropriate message to the user.

5 Newsletter

We created a simple Newsletter form, where a user can add their email to the database in order to receive emails on new auctions etc.

6 Auctions

The user can create an auction by filling the corresponding form, specifying the auction details. They can also edit an auction before it has been started, they can delete it, they can view it, they can start it etc. All started and non terminated auctions are displayed and can be accessed by all users, even non authenticated ones.

7 Administration

On the installation of the app an admin user is created, who can see all users' data, approve or delete a user registration and download in json or xml format all the auctions of the website. Only the specific user has access to the corresponding endpoints.

8 Messaging - Notifications

Using conditional rendering and performing queries to the database we get the inbox/sent/new message page. When a user receives a message it is displayed right away in the inbox, since setInterval is used to refresh the page occasionally. Notification for new messages is shown in all the pages until the user goes to their inbox. For this, we store in the account table the number of messages they have and we compare it to the new number of received messages, in order to show notifications.

9 OpenStreetMap

For this task we make a GET call to the OpenStreetMap API using the seller's address and city data, in order to get the coordinates of the auction. The we use these coordinates in combination with the React Leaflet library to see this location in the map.

10 Filters

In the page where users browse through the auctions, we have added some filters, so that the user can see what they really want. So, we filter the auctions, using some db queries, according to the category, price and location. For these we used checkboxes, dropdown menu and a range slider in the user interface, where all the possible options are shown from the backend. Every time the user changes the state of one of these, a new 'select' query is created. When the user chooses many filters, the auctions shown should satisfy them all at the same time(intersection).

11 Image Upload - File Download

For the image upload we used React Dropzone in the frontend, along with FormData and multer in the backend, in order to upload the file. The url of the file is stored in the database, so that it can be easily retrieved when needed. For the auctions export in XML format we used the exportFromJSON function with the appropriate arguments. The admin just presses the corresponding button and downloads a file with all the auctions of the website.

12 Search Engine

We created a simple search engine that helps display the auctions that match a specific description, using Postgres Full-Text Search. First of all, we tokenized and performed stemming on the description of each auction and added the result to a new column. We then created a new column with the Generalized Inverted Index and so we were able to perform queries using the user input.

13 Recommender System

14 Challenges

The most challenging parts of this project were:

- finding the right library/way to perform some more advanced tasks, such as creating the map, uploading images, exporting json/xml

- performing more advanced queries to filter the auctions shown
- creating the search engine
- restricting access in pages, hide elements from non authenticated users
- getting to know react hooks, handlers, asynchronous functions and conditional rendering

15 Conclusion

During this assignment, we had the opportunity to do research on plenty of concepts and technologies. We familiarized with web development, coming across realistic scenarios and finding solutions using different libraries and frameworks. All in all, me managed to create a user-friendly website with all the basic and some more advanced functionalities of such an application.