Web Applications I – Exam # 4 (deadline 2024-01-24 at 23:59)

“Rescuing Surplus Food”

FINAL VERSION – No modifications were made with respect to the preliminary version

Design and implement a web application for rescuing surplus food. The application will enable users to purchase surplus food from local stores and restaurants at discounted prices, reducing food waste. For that purpose, the application must meet the following requirements:

In the application, the surplus food items are packaged into “bags,” and there are two types of bags: “**surprise bags**,” which are bags filled with assorted surplus food items from a store or restaurant without knowing the exact contents beforehand, and “**regular bags**” in which the food items and their quantity can be known (for instance: 2 apples, 1 banana).

On the website's main page, accessible without authentication, it is possible to view a list of establishments (stores and restaurants) participating in the surplus food program. This list of stores and restaurants must be displayed alphabetically, showing their name, address, phone number, and type of cuisine or food category.

After logging in, users can reserve one or more bags by selecting the desired quantity and size. There are three possible sizes: small, medium, and large. Therefore, on this page, the user can see all the available bags at each establishment. Each bag is displayed by showing its type (surprise, regular), its content (only if regular), its price, its size, and a time range for picking up. To simplify the implementation, all bags showcased in the application are set for pickup on the current day, and only future time ranges are enabled. For instance, users cannot select a bag with a pick-up time from 10 AM to 11 AM if it is 6 PM.

After a user sets apart a bag, it appears in their shopping cart. Once they've completed their selections, they proceed to confirm their choices. If the bag is a regular bag, users can view a detailed list of food items and their quantities. Within this bag type, users can remove a maximum of two food items. Notably, even if users choose to remove items from the bag, the overall price of the bag remains unaltered. For surprise bags, users can only add or remove the entire bag from the shopping cart without the option to see or modify its contents.

The system manages two states for each bag:

- Available: Indicates bags ready for reservation.
- Reserved: Designates a bag that other users have chosen and confirmed in their shopping carts (reserved).

Each authenticated user can add to the shopping cart just one bag per establishment within the same day to ensure fairness and allow more users to benefit from the program. Consequently, if the user takes a bag out of the shopping cart, they regain the ability to add a different bag from that same establishment. Additionally, the payment is made at the time of order pickup, the system does not need to handle the pickup process. However, after the user confirms its choices in the shopping cart, it must update the inventory accordingly. Similarly, an authenticated user should be able to visualize how many bags are available and reserved.

Upon confirming the shopping cart, the selected bags are reserved for the user if they are still available (considering that other registered users may request the same items simultaneously).

However, if any selected items are no longer available, the whole reservation will be canceled. In this case, the app will highlight the unavailable bags on the interface for 5 seconds to indicate the reason for cancellation. If, upon confirmation, all selected bags are still available, they will be immediately marked as reserved.

Moreover, authenticated users have the option to delete their own reservations, releasing all the items reserved by them. Deleting a reservation makes these items available again for all users, and the app interface will update to reflect these changes.

The organization of these specifications in different screens (and possibly on different routes) is left to the student.

Project requirements

- The application architecture and source code must be developed by adopting the best practices in software development, in particular, those relevant to single-page applications (SPA) using React and HTTP APIs.
- The project must be implemented as a React application that interacts with an HTTP API implemented in Node+Express. The database must be stored in a SQLite file.
- The communication between client and server must follow the "two servers" pattern, by properly configuring CORS, and React must run in "development" mode with Strict Mode activated.
- The evaluation of the project will be carried out by navigating the application. Neither the behavior of the "refresh" button, nor the manual entering of a URL (except /) will be tested, and their behavior is undefined. Also, the application should never "reload" itself as a consequence of normal user operations.
- The root directory of the project must contain a README.md file and have two subdirectories (client and server). The project must be started by running the two commands: "`cd server; nodemon index.js`" and "`cd client; npm run dev`". A template for the project directories is already available in the exam repository. You may assume that nodemon is globally installed.
- The whole project must be submitted on GitHub, on the same repository created by GitHub Classroom.
- The project **must not include** the `node_modules` directories. They will be re-created by running the "`npm install`" command, right after "`git clone`".
- The project may use popular and commonly adopted libraries (for example day.js, react-bootstrap, etc.), if applicable and useful. Such libraries must be correctly declared in the package.json file, so that the npm install command might install them.
- User authentication (login and logout) and API access must be implemented with passport.js and session cookies. The credentials should be stored in encrypted and salted form. The user registration procedure is not requested.

Database requirements

- The student must implement the project database and be pre-loaded with at least four establishments, 8 bags (half of them surprise and the other half regular), 12 food items, and three registered users, of whom two have already reserved some bags on the current day.

Contents of the README.md file

The README.md file must contain the following information (a template is available in the project repository). Generally, each information should take no more than 1-2 lines.

1. Server-side:
    a. A list of the HTTP APIs offered by the server, with a short description of the parameters and of the exchanged objects
    b. A list of the database tables, with their purpose
2. Client-side:
    a. A list of 'routes' for the React application, with a short description of the purpose of each route
    b. A list of the main React components
3. Overall:
    a. A screenshot of the **application where it is possible to reserve bags**. The screenshot must be embedded in the README by linking two images committed in the repository.
    b. Usernames and passwords of the users.

Submission procedure

To correctly submit the project, you must:

- **Be enrolled** in the exam call.
- Use the provided **link** to **join the classroom** on GitHub Classroom (i.e., correctly **associate** your GitHub username with your student ID) and to **accept the assignment**.
- **Push the project** into the **main branch** of the repository created for you by GitHub Classroom. The last commit (the one you wish to be evaluated) must be **tagged** with the tag **`final`** (note: `final` is all-lowercase, and it is a git 'tag', nor a 'commit message').

Note: to tag a commit, you may use (from the terminal) the following commands:

```
# ensure the latest version is committed
git commit -m "...comment..."
git push

# add the 'final' tag and push it
git tag final
git push origin --tags
```

Alternatively, you may insert the tag from GitHub's web interface (follow the link 'Create a new release').

To test your submission, these are the exact commands that the teachers will use to download and run the project. You may wish to test them on a clean directory:

```
git clone ...yourCloneURL...
cd ...yourProjectDir...
git pull origin main  # just in case the default branch is not main
git checkout -b evaluation final # check out the version tagged with
'final' and create a new branch 'evaluation'
(cd client ; npm install; npm run dev)
(cd server ; npm install; nodemon index.js)
```

Ensure that all the needed packages are downloaded by the npm install commands. Be careful: if some packages are installed globally, on your computer, they might not be listed as dependencies. Always check it in a clean installation.

The project will be tested under Linux: be aware that Linux is case-sensitive for file names, while Windows and macOS are not. Double-check the case of import and require() statements.