Web Applications I – Exam # 1 (deadline 2020-06-28 at 23:59)

"Car Rental"

⚠ PRELIMINARY VERSION – THE FINAL VERSION WILL BE PUBLISHED AFTER THE VIDEO CHAT ON FRIDAY 12/06 ⚠

Design and implement a web application for managing a car rental company.

The company owns a set of vehicles. Vehicles are divided into 5 categories (A…E), according to their size. Each car is also characterized by a brand (e.g., Audi, Ford, Fiat, Mercedes, …) and a model (i.e., a string).

Unauthenticated users may browse a list containing the full set of vehicles, and the list may be filtered by category and/or brand (the user may simultaneously select one or more categories and one or more brands).

Once authenticated, users have access to an *interactive configuration page*, that replaces the original browsing page. In the configurator, they can set the parameters for their new rental. Parameters include: starting day, end day, car category, driver's age, number of extra drivers, estimated number of kilometers, extra insurance.

For each combination, the page should interactively show the number of available cars (matching all criteria), and the *rental price* for that solution. The rental price is computed by considering different factors (see Pricing Table below), that are also affected by the number of available vehicles (when fewer vehicles would remain, the price grows higher). Also, after 3 successful rentals, the customer gains a 10% discount over all his/her new rentals.

If the user accepts the proposal in the interactive configurator, he/she must pay immediately, as soon as he/she confirms the reservation (you may develop a "stub" API for fake payments, that receives credit card information[1] and the amount; no validation is required in this service).

Authenticated users, in addition, may check the list of their future reservations (and possibly cancel them), and check the history of their past rentals in a dedicated page.

Pricing Table

Item	Price
Category A car	80 €/day
Category B car	70 €/day
Category C car	60 €/day
Category D car	50 €/day
Category E car	40 €/day
Less than 50 km/day	-5%
Less than 150 km/day	0%
Unlimited km/day	+5%
Driver's age under 25	+5%

[1] Full name, card number, CVV code

Driver's age over 65	+10%
Extra drivers	+15%
Extra insurance	+20%
Less than 10% vehicles in the same categories are left in the garage in the days of the rental	+10%
Frequent customer (more than 3 rentals)	-10%

Project requirements

- The application architecture and the source code should be developed by adopting the best practices in software design, in particular for single page applications using React and REST.
- The project must be realized as a React Application, interacting with a REST API implemented in Node+Express. The database should be stored in an SQLite file.
- The communication between client and server must follow the "React Development Proxy" pattern and React must run in development mode.
- The top-level directory must have a README.md file and contain two sub-directories (client and server). The project must be run with the following commands: "cd server; nodemon server.js" and "cd client; npm start". A skeleton of the project directories will be provided.
- The whole project must be submitted on GitHub, in the repository created by GitHub Classroom.
- The project must not include the node_modules directories. They shall be re-populated by running "npm install", immediately after "git clone".
- The project may use popular and commonly adopted libraries (such as moment.js, react-bootstrap, etc.), if applicable and useful.
- User login and access should be protected with a JWT token, stored in an http-only cookie. No additional protection is required.
- The project data-base must be defined by the student and must be preloaded with at least 5 example users (the registration procedure is not required), and at least 20 cars of at least 5 different brands.

Contents of README.md

The README.md file must contain the following information (a template will be available in the project skeleton – in general each explanation should be no longer than 1-2 lines):

1. A list of the React Application Routes, with a short description of each route's purpose
2. A list of REST APIs offered by the server, with a short description of the parameters and the exchanged entities
3. A list of the database tables, with their purpose
4. A list of the main React components adopted in the application.
5. A screenshot of the interactive configurator (embedding a picture stored in the repository).
6. The usernames and passwords of the 5 test users. Identify one user, which is a frequent customer.