Exam project work

Tecnologie e applicazioni web, a.y. 2023/2024

This year's project involves developing a web application, including a server with REST-style APIs and a SPA front-end, that allows students to buy and sell used university textbooks by creating and bidding on auctions.

The application must handle two types of users: **students** and **moderators**. Students can access the app to sell or buy books. Moderators can forcefully take action on the active student's auctions, like editing or deleting the content.

The application workflow should resemble an "eBay-style" (www.ebay.it) auction site. Any user can log in to see the list of books currently for sale, even if not registered. The search can be narrowed down according to various parameters, including the book's title, the course in which it is used (and in which university), the geographical area of the seller, and the current auction price.
After registering and authenticating to the system, a student can bid at one or more auctions in progress. At the end of the auction, the student who has offered the highest price (as long as it exceeds the reserve price) is the winner and can proceed with the purchase by contacting the student seller.
A student can create new listings, specifying the book's details, the duration of the auction, the starting price, and the reserve price[1]. Once the auction has been started, it can only be removed by a moderator or when it expires.

Buyers can communicate, even in real-time, with sellers by leaving messages. Messages are associated with each book for sale. A message thread can be private, i.e. visible only to the sender and recipient, or public, i.e. visible to all students who observe an auction in progress.

[1] Note: The reserve price (i.e. the minimum amount that a seller will accept as the winning bid) is only visible to the advertiser. Other students watching the ad may only know about it once the auction is complete.

Architecture

The system must comprise:

- A REST-style **backend** webservice, implemented in TypeScript and running on Node.js. The service must also use:
    - MongoDB DBMS for data persistence
    - Express.js for routing

- A web **front-end** implemented as a Single Page Application (SPA) using the Angular framework.

Note: Each component (Backend, MongoDB, Front-end) must run in a separate Docker container.

Requirements

Your application must implement (at least) the following features:

1. User management
    1. Registration of new students;
    2. Registration of new moderators "by invitation". Moderators cannot register themselves but must be registered by another moderator who sets a temporary name and password. Then, at the first login, the new moderator must set her/his credentials (password, name, surname, etc.);
    3. Deletion of existing students (moderators only)
    Note: Login is required for all the system functions available to moderators.
2. Listing management
    1. Placing a new listing to start the auction. (students only);
    2. Display of active auctions (all users);
    3. Ability to filter active listings on different parameters, such as book's title, degree course, user, price, etc. (all users);
    4. Ability to bid on a listing. To be valid, the bid must exceed the current auction price. (students only);
    5. Sending messages to a seller. A message can be private or public. The seller can reply to the message, and the visibility of the reply will be the same as the first message sent by the potential buyer. Messages are associated with a specific auction. (students only);
    6. Editing the auction's information (moderator only);
    7. Deleting an auction before it expires (moderator only);
3. Sales
    1. At the end of the auction, the student who has bid the highest price (as long as it exceeds the reserve price) is the winner. The system must notify the winning student and the advertiser of the auction's outcome.

4. Statistics
    1. The system must provide each student with the listings they have made, the auctions they have participated in, and the ones in which they have been the winner.
    2. The system must provide each moderator with the number of (i) active auctions successfully concluded and (ii) auctions closed without reaching the reserve price.

You are free to implement additional features beyond those listed above. Implementing additional features does not preclude or guarantee the achievement of honors but contributes to defining a positive judgment.

When the backend starts, it must preload a small number of students and moderators into the database to simplify application testing during the exam.

Submission

Projects must be submitted via Moodle on the following page:

https://moodle.unive.it/mod/assign/view.php?id=806246

If working in a group, each member must submit the same file named `<group_name>.zip`.

The package must contain:

- **The report of each student**, in PDF format, named `<student_firstname>_<student_surname>_<matriculation_number>.pdf`
- A `README.txt` file with instructions on how to run the entire application. For example, you can briefly summarize the shell commands to compile the sources, start backend and frontend, etc.
- All the application source code, clearly divided in backend and frontend.

NOTES:

- Do not submit any external library. Please delete all the `node_modules` directories before submission.
- Students can freely form groups of up to 3 people. Moodle will not help or enforce group submission. Individual students can simply use their surname as their group name.

Report

A report describing the system architecture, the software components used, and how they contribute to achieving the required functionalities must be submitted with the application. The report is **strictly individual** and will serve as a basis for the oral discussion of the project.

The report must contain:

- A description of the system **architecture**, listing all the different components and their relationships (i.e. How they work together to implement the application requirements).
- A description of the **data model**, including the collections and the structure of the documents inside each collection.
- A precise description of the **REST APIs**. Such a description must contain the list of endpoints, together with their parameters and what data is exchanged (give examples in JSON format whenever possible).
- A description of how user **authentication** is managed, along with the associated workflow.
- A description of the Angular frontend, with a concise list of the implemented **components, services, and routes**.
- Some examples, possibly with **screenshots**, showing the typical application workflow for each different user role.

Q&A

For any questions about the project work, or the course in general, don't hesitate to contact the teacher at:
filippo.bergamasco@unive.it

I'll be happy to arrange a meeting if needed!

Additional information about the exam is available at:
https://moodle.unive.it/mod/page/view.php?id=776703

Prof. Filippo Bergamasco,
Tecnologie e Applicazioni Web, a.y. 2023/2024