Web Applications I – Exam # 2 (deadline 2023-07-09 at 23:59)

“Airplane Seats”

FINAL VERSION – Modifications are reported in “red”

Design and implement a web application for booking seats on a plane. The application must meet the following requirements.

For simplicity, an airplane is composed of a set of seats arranged in a grid format with F rows and P seats per row. The number of rows and seats per row depends on the type of plane being booked. There are three types of planes: *local* (F = 15, P = 4), *regional* (F = 20, P = 5), and *international* (F = 25, P = 6).

On the main page of the website, accessible without authentication, it is possible to select a plane from the available types. The page shows the seat availability for the selected plane. The graphical implementation of the two-dimensional seat visualization is left to the student. Each seat should have a code indicating its row (1, 2, 3, ...) and position within the row (A, B, C, ...). For example, the first seat in the 10th row will have the code “10A,” while the sixth seat in the 3rd row will have “3F.” In the non-authenticated view, each seat should display its status (either occupied or available) in addition to its code. The status can be represented by colors, icons, or any other visual elements chosen by the student. The seat visualization page must also ~~continuously~~ display the number of occupied seats, available seats, and the total number of seats on the plane.

After authentication (login), a user should be able to select a plane from the available types and make a new reservation for *one or more seats* using one of the following two alternative methods:

1. Specify the number of seats to be booked, and the system will automatically assign seats (if enough seats are available). The logic for seat selection is left to the student (random, by row, etc.).
2. Directly interact with the seats using a two-dimensional visualization similar to the non-authenticated seat visualization page (more details in the next paragraph).

In the latter case, the reservation process must take into account the seats’ status (displaying them accordingly). In this case, seats can have three states:

- Occupied seats cannot be requested.
- Available seats can be requested by clicking on them. Once clicked, the seat should *immediately* be displayed as Requested on the screen.
- Requested seats, which can be released if clicked (becoming Available again).

The seat visualization should always display the number of occupied, available, requested, and total seats. These numbers should be updated every time the user performs an action on the seats.

Note that while the user is booking seats, other registered users may request and/or occupy the same seats. This is normal and should neither be prevented nor notified to the user.

Once the seat selection operations are completed using either of the two methods, the user can *confirm* or *cancel* their reservation. If, at the time of confirmation, all selected seats are still available, they should

Commented [1]: Just a general question: If we chose to do exam #2 instead of exam #1, is there any way we should notify that?

Is it enough to accept the Exam #2 assignment on Github or do we have to 'unselect' Exam #1 in a formal way?

Commented [2]: No, just don't submit exam #1 (no tag "final" or no code except the template folders in the repo, for instance)

Commented [3]: As far as the Requested seats are concerned, should we keep track of this information within the database? Or is it something we can manage on the client?

Commented [4]: It should be fine managing requested seats on the client

Commented [5]: Is the seat availability supposed to be shown on the main page where you select the type of plane as well? Or is it enough to only show the different plane types?

Commented [6]: it's not required

Commented [7]: list of available flight and the graphical visualization of occupied and available seats must be all in the same page, without using routing?

Commented [8]: there is no specific requirement about which routes you will use.
The singular form of "the page" implies that the same page contains the 'selector' for the plane and the ... [1]

Commented [9]: Can we have multiple planes per t ... [2]

Commented [10]: One per type, 3 planes in total

Commented [11]: So can we use the type as identifier?

Commented [12]: it's your implementation choice. B ... [3]

Commented [13]: What is meant by "continuously"? ... [4]

Commented [14]: no, you should not query the serv ... [5]

Commented [15]: Is there a limit for the number of ... [6]

Commented [16]: no, there is no limit specified (afte ... [7]

Commented [17]: it is not clear to me if both metho ... [8]

Commented [18]: Both must be implemented. The ... [9]

Commented [19]: It is an out out or could be mixed ... [10]

Commented [20]: Alternative: one OR the other

Commented [21]: How can be changed the state f ... [11]

Commented [22]: By clicking on the seat. In this ... [12]

Commented [23]: Does this mean that the actual ... [13]

Commented [24]: Actual number, in addition to the ... [14]

Commented [25]: I'm sorry is the number of ... [15]

Commented [26]: yes, correct

Commented [27]: Should the available number de ... [16]

Commented [28]: Yes, every time a seat is reques ... [17]

all become occupied simultaneously. Otherwise, the reservation will be canceled, and the cancellation reason should be indicated by highlighting the seats already occupied by others on the two-dimensional seat visualization (for a duration of 5 seconds).

A logged-in user may additionally delete his/her own reservation (if it exists), thereby releasing all seats that were occupied by him/her. Deleting the reservation will make the seats available again and update the visualization accordingly. As a general constraint, an authenticated user can only make one seat reservation (including one or more seats, as before) per plane.

The organization of these specifications into different screens (and potentially different routes) is left to the student and is subject to evaluation.

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 project database must be implemented by the student, and must be pre-loaded with *at least* 2 users who have a confirmed reservation of at least 2 seats each, on two different planes; and with at least 2 other users without any reservation.

Commented [29]: Hello again, I have a doubt wrt this part:
I was thinking to use an approach based on multiple promises with Promise.all(), however I'm not sure if in this case a single failure will stop all other promises from being fulfilled and therefore do not update the database. Shall we use transactions instead?

Commented [30]: This is a design/implementation choice, which is left to you. The fundamental thing is to satisfy the requirement.

Commented [31]: is it ok if after the user tries to book the seat and one or more of them is occupied the reservation is canceled AND the two dimensional visualisation is re-rendered showing the seats that are now occupied instead of showing them for 5 seconds?

Commented [32]: The duration of 5 seconds is not optional

Commented [33]: ok if i understood correctly, the page firstly loads the grid with the informations retrieved by the server, and the user1 will visualise a grid with the fetched available and occupied seats, if in the mean time another user2 has booked a seat that is showed available for the user1 the user1 must not be able to book it and this will be showed by leaving the seat occupied for five seconds instead of showing the seat occupied, so after the 5 seconds pass the seat becomes available again ?

Commented [34]: More or less. Let's say that user 1 is reserving 3 seats (1A, 2A, 3A), for example. While user 1 is doing this, user 2 successfully completes the reservation of seat 3A (which is then occupied). When user 1 confirms his/her reservation, the app will notice the unavailability of seat 3A (since it was already ... [18]

Commented [35]: When should this be controlled? Before selecting more seats (so the new reservation is blocked) or after, when the user tries to reserve th ... [19]

Commented [36]: The app should prevent a user to have more than one seat reservation per plane.

Commented [37]: As regards the cancellation of reservations, do we have to indicate in some way to the user which seats he has already reserved? Or is i ... [20]

Commented [38]: For the cancellation of a not-confirmed reservation, no, as a reserved seat is a temporary condition until it is confirmed (or do you mean ... [21]

Commented [39]: No, what I mean is the following: if the user wants to cancel some reservations, he will see the grid with all the seats (available or occupied); but ... [22]

Commented [40]: Just to clarify: are you speaking of cancelling an ongoing reservation or deleting the existing reservation? Because in the text they are ... [23]

Commented [41]: if you are speaking of "delete", it's not requested. At the end of the deletion of the reservation, the grid must be updated accordingly (i.e., withou ... [24]

Commented [42]: Yes, I was referring to "delete a reservation", thank you

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 o 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 **screen for booking the seats**. The screenshot must be embedded in the README by linking the image committed in the repository.
   b. Usernames and passwords of the users.

Commented [43]: what program are you using to read the MD file, I'm having trouble seeing the image even though following through the link open the right one, is it still fine?

Commented [44]: the image should be visible both in VS Code and on GitHub. If it's not visible, there's something wrong with your embedding.

Commented [45]: (check for absolute pathnames and/or filename case mismatches, or the embedding syntax, ...)

Commented [46]: The screenshots are visible on GitHub, but not in VS Code, hopefully this won't be a problem

Commented [47]: it should be fixed

Submission procedure

To correctly submit the project, you must:

- **Be enrolled** in the exam call.
- **Accept the invitation** on GitHub Classroom, and correctly **associate** your GitHub username with your student ID.
- **Push the project** in 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.

Page 1: [1] Commented [8] Fulvio Corno 6/21/23 11:42:00 AM

there is no specific requirement about which routes you will use.
The singular form of "the page" implies that the same page contains the 'selector' for the plane and the availability of that selected plane. They may 'live' at the same route or at different routes.

Page 1: [2] Commented [9] Alessia Leclercq 6/22/23 9:24:00 AM

Can we have multiple planes per type or just a single one?

Page 1: [3] Commented [12] Fulvio Corno 6/23/23 10:47:00 AM

it's your implementation choice. But, considering that there is exactly one plane per type, it seems the simplest solution.

Page 1: [4] Commented [13] David Bass 6/19/23 1:46:00 PM

What is meant by "continuously"? Should we query the database after a few seconds all the time to update the occupied and requested seats? Because later you talk about not informing the users if someone else requested a seat.

Page 1: [5] Commented [14] Fulvio Corno 6/19/23 2:01:00 PM

no, you should not query the server, the requirement only asks to report the totals for the currently displayed seats (with the information available in the client).
I deleted the word 'continuously', it should be clearer this way.

Page 1: [6] Commented [15] Anonymous 6/22/23 3:55:00 PM

Is there a limit for the number of seats that one user can reserve or the limit is the max number of available seats?

Page 1: [7] Commented [16] Fulvio Corno 6/23/23 8:30:00 AM

no, there is no limit specified (after all, he is paying for them...)

Page 1: [8] Commented [17] Anonymous 6/26/23 3:14:00 PM

it is not clear to me if both methods should be implemented or if we have to choose one of the two

Page 1: [9] Commented [18] Luigi De Russis 6/26/23 3:38:00 PM

Both must be implemented. The logged-in user can choose one of them

Page 1: [10] Commented [19] Anonymous 6/27/23 10:58:00 AM

It is an out out or could be mixed? While giving the number of seats to be booked (alternative 1), should the system immediatly display them on the plane, or just after the booking is completed?

Page 1: [11] Commented [21] Pietro Montresori 6/22/23 3:17:00 PM

How can be changed the state from 'requested' to 'available'?

Page 1: [12] Commented [22] Luigi De Russis 6/22/23 3:26:00 PM

By clicking on the seat. In this moment, you are interacting with the seats using a 2d visualization. First click on an available seat -> requested. Second click on the same seat (now requested) -> available. Another click on the same seat (now back to be available) -> requested again.

Page 1: [13] Commented [23] Alessia Leclercq 6/19/23 1:51:00 PM

Does this mean that the actual number is displayed, or it is implicit in the visualization map?

Page 1: [14] Commented [24] Luigi De Russis 6/19/23 1:52:00 PM

Actual number, in addition to the visualization map.

Page 1: [15] Commented [25] Alessia Leclercq 6/25/23 4:59:00 PM

I'm sorry is the number of available seats updated according to the number of reserved seats also? I mean, when I reserve a seat I should also see the available number decrement, right?

Page 1: [16] Commented [27] Kalller 6/27/23 8:24:00 PM

Should the available number decrease every time a seat is selected?

Page 1: [17] Commented [28] Luigi De Russis 6/27/23 8:37:00 PM

Yes, every time a seat is requested, the number of available seats should decrease. Occupied + available + requested should be equal to total seats

Page 2: [18] Commented [34] Luigi De Russis 6/26/23 5:36:00 PM

More or less. Let's say that user 1 is reserving 3 seats (1A, 2A, 3A), for example. While user 1 is doing this, user 2 successfully completes the reservation of seat 3A (which is then occupied). When user 1 confirms his/her reservation, the app will notice the unavailability of seat 3A (since it was already occupied by user 2) and highlight seat 3A on the 2d seat visualization for 5 seconds as the motivation for the cancellation.

Page 2: [19] Commented [35] Anonymous 6/27/23 9:58:00 AM

When should this be controlled? Before selecting more seats (so the new reservation is blocked) or after, when the user tries to reserve the selected seats (before inserting the new reservation it will be controlled if the user already has a reservation on that plane)

Page 2: [20] Commented [37] francesco Santoro 6/26/23 5:48:00 PM

As regards the cancellation of reservations, do we have to indicate in some way to the user which seats he has already reserved? Or is it not mandatory?

Page 2: [21] Commented [38] Luigi De Russis 6/26/23 6:01:00 PM

For the cancellation of a not-confirmed reservation, no, as a reserved seat is a temporary condition until it is confirmed (or do you mean "occupied"?)

Page 2: [22] Commented [39] francesco Santoro 6/26/23 7:04:00 PM

No, what I mean is the following: if the user wants to cancel some reservations, he will see the grid with all the seats (available or occupied); but is there a way for the user to see which reservations he has already performed in the past? Or is it not requested and will only see the regular grid with all the reservation (even those made by other users)?

Page 2: [23] Commented [40] Luigi De Russis 6/26/23 7:13:00 PM

Just to clarify: are you speaking of cancelling an ongoing reservation or deleting the existing reservation? Because in the text they are different things and reading your description it seems you are referring to "delete a reservation".

Page 2: [24] Commented [41] Luigi De Russis 6/27/23 5:06:00 AM

if you are speaking of "delete", it's not requested. At the end of the deletion of the reservation, the grid must be updated accordingly (i.e., without the seats that were included in the just deleted reservation)