IDG2100 Full-stack Web Development: exam description

The CampusPong web app

This document contains the description of the final exam for IDG2100 Spring 2023 which will be done in groups of 3. The project must be delivered in this GitHub repository as well as in inspera.

Notice this is an extension of oblig#1, oblig#2 and oblig#3. Therefore, if you decide to reuse code from those assignments, make sure it is properly cited.

This document will be revised and posted in inspera next week.

04/05/2023

Table of contents

Goal

This exam aims to prove and demonstrate that you have achieved the learning outcomes, knowledge, skills and general competence described in the course description.

It is especially important that you show that:

- you understand and can build a full-stack modern web application using the `MERN` stack;
- you can design a REST API following best practices;
- you can successfully document your projects using modern tools such as Swagger and Storybook;
- you can provide basic security to both the back-end and the front-end;
- you can use modern developer tools such as task runners, bundlers, etc.

The CampusPong web app

Great news! The Dean is finally fully on board with the development of a brand-new web application based on the success of the table tennis league application ("Obligs" 1, 2 and 3). The new web application, named CampusPong[1], will take the table tennis league experience to the next level by incorporating exciting new functionalities and user roles.

The landing page of the application will give *visitors*, non-authenticated users, a comprehensive overview of the platform's features, while highlighting the top players in the league. With CampusPong, visitors will be able to see not only the total number of players registered on the platform but also the total number of matches played by all players so far. They will also be able to see the names and scores of the top 5 players in the league, adding an extra element of excitement and competition to the platform.

Players: authenticated users with their own personal profile page where they can easily manage their account information, such as editing their name and changing their password. In the player history page, they will be able to view their match history such as who they played against, who won, and how many points they got. In addition, players can view a ranking page which shows a list of all registered players sorted by total score, where they can easily add other players to a list of favourites for quick access to their profiles and match history. Players will also have access to a

[1] Feel free to change the name of the app if you like.

"My Favorite Players" page that displays the list of players they have bookmarked, making it easier to keep track of their favourites.

Lastly, as authenticated users, each player will have the ability to create and start a new match, which will consist of two players, a set time, and the same configurations as those used in the first "oblig". Players can use the web component from "oblig 1" to play the game and store the results of the new match, including the scores of each game played.

Admins: have access to all user data, except for passwords, and will be responsible for managing users. They will have the ability to create, update, and delete player profiles, as well as generate new passwords for users.

The following table shows a summary of the access rights each type of user may have.

User type	Access rights
External/visitor	• Can view total number of matches played so far • Can view total number of registered users. • Can view top 5 players and their scores
Player	• Can read and modify own user data (including changing password) • Can read info about players (no passwords) • Can see info about matches • Can see info/statistics about a specific match • Can bookmark other players • Can see favourite players. • Can create new matches but cannot change the scores after playing the match
Admin	• Same as other users • Can change the scores of matches. The idea is if players made a mistake entering the results, they will have to ask the admin to fix the error.

	• Can add players. • Can manage/remove players

You have to update the backend and your database from “oblig 2” to be able to develop the above functionalities. You do not need to create problem documentation beforehand, but you need to create swagger documentation to explain your back-end implementation.

Tasks

Your goal is to develop this application by first integrating the functionalities of assignments 1, 2 and 3 and then creating additional user interfaces to support the new functionalities. You also must make adaptations to the back-end to support authentication and authorisation roles for the different user types. If you want to recycle and adapt the code, you have written in the previous “obligs” or if you want to start from scratch is up to you. However, be aware that there are significant adaptations to both the front-end and back-end you need to make on your previous code to pass this assignment (e.g., new user interfaces, additions to the database and back-end code). The pages presented in the following tables are meant to be used as a starting point and you can decide to implement new views or slightly modify them as long as you meet the functionalities and explain the design decisions. Create your own designs and put some effort into making the webpages visually appealing and usable. In the next section, we provide a summary of the functional requirements for the application.

Requirements

The following table depicts a list of requirements for each page/functionality. The last column shows a star-system that defines a tentative grading guide which will be used to evaluate the projects.

- Max grade: D – you must implement all the tasks with 1 star (*)
- Max grade: C – you must implement all the tasks with 1 star (*) and all the tasks with 2 stars (**)
- Max grade: A – you must implement all the tasks with 1 star (*), all the tasks with 2 stars (**) and some of the tasks with 3 starts (***)

Landing page	
External/ Player/ Admin	• (*) Can see info about platform. • (*) Can see total number of players. • (*) Can see total number of matches. • (*) Can see top 5 players and their scores. • (*) Can register new users ○ (***) Send validation email to validate new user's email address is correct

User (player/admin) profile page	
External	• (*) No access
Player	• (*) Can read and modify own user data (including changing password)
Admin	• (*) No access

New match page	
External	• (*) No access
Player	• (*) Can create a new match between him/her and another player. • Can store the results of a match: ○ (*) Option a - the match and its results are created in the same form at the same time (see Figure 1) ○ (**) Option b – the match is created without the results and the user is redirected to a unique URL that shows the web component to count points ("oblig" 1). When the match finishes, the user can save the results (see Figure 2) • (*) After results have been submitted, points are automatically given to players (see "oblig 2", three points for a win and 1 point for a loss)

Admin	• (*) Can create a new match between two players and store its results at the same time.

Match page (e.g.: /match/123)	
External	• (*) No access
Player	• See the information about the match based on approach ◦ (*) if option a – this page will show the information of the match and the results (see Figure 1). ◦ (**) if option b – this page will render different information depending on the status of the match. For finished matches, it will show the statistics of the match and its results. If the match has not been played yet, the page will render the web component built in the first "oblig" and the users will have the ability to play the game. As soon as they are done, the player will be able to "save" the results of the match (see Figure 2). • (*) Cannot edit results after the match has ended
Admin	• (*) Can view all matches. • (**) Can add and edit scores for games/matches played (consider what happens to the rank points). • (**) Can remove matches (consider what happens to the rank points).

Player matches	
external	• (*) No access
Player	• (*) Can see the history of the matches he/she has played.
Admin	• (*) Can see the history of all matches

Leader board	
External	• No access
Player	• (*) Can see all the players ranked by score. • (*) Can bookmark other players. • (*) Can see favourite players.
Admin	• (*) Can see all the players ranked by score.

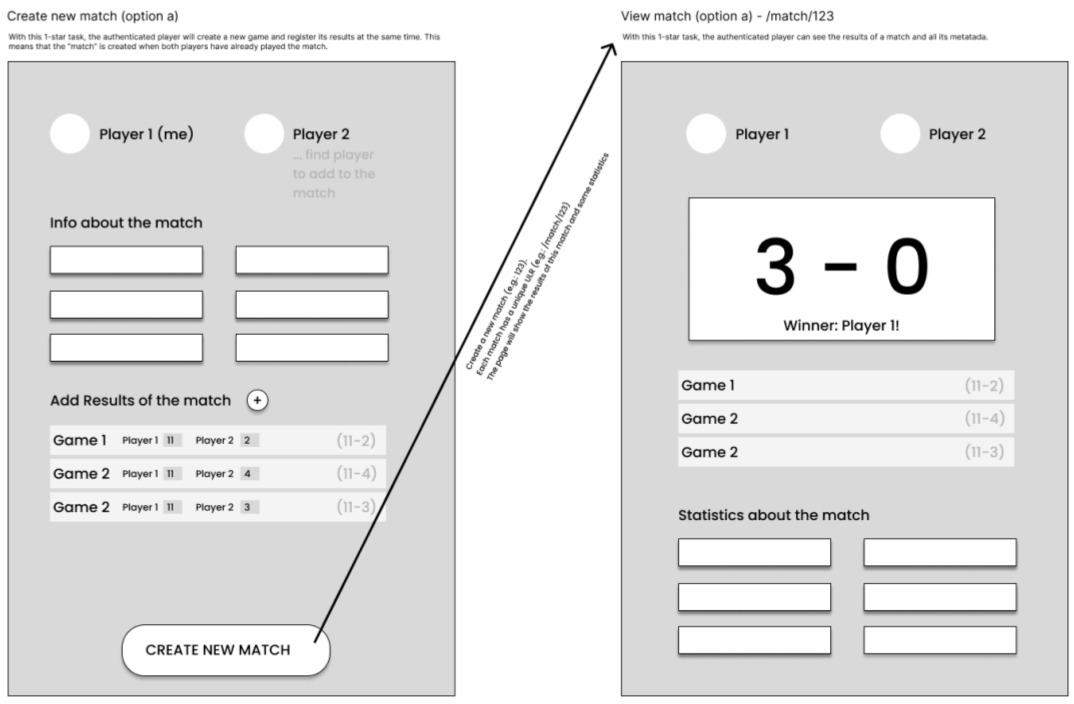
Admin pages	
External	• (*) No access
Player	• (*) No access
Admin	You are in charge of deciding how the admin will administrate the users and how many pages you will use for that. Make sure they have the following functionalites: • (*) Can see list of all users, • (*) Can edit users • (**) Can delete users • (*) Can change passwords of users • (***) Can send an email to the user with the new password

General	
Documentation	• (**) Document the API using Swagger • (***) Document the components using Storybook (at least stateless component) • (*) Export the db with some "dummy" data that can be used to test the system (this task is required even when you deploy your database using online services) • (**) Write a good readme file explaining how to set up a new project and provide scripts (package.json) to deploy the app locally the first time and automatise as many steps as possible.

Security	• (*) Store JWT using localStorage (not secure) • (**) Store JWT in a secure way (HTTP only cookies) • (***) Store JWT in a secure way (Refresh token and HTTP only cookies)
Design and usability	• (***) Make sure your application follows best standards and practices of usability and user experience design (show clear error messages to the user, ensure good navigation design, etc.) • (**) Create an appropriate theme for the application that is visually appealing.

Create/show new match (flow)

The section shows the flow when creating and visualising a new match (option a and b) in the requirements tables.


Create new match (option a)

With this 1-star task, the authenticated player will create a new game and register its results at the same time. This means that the "match" is created when both players have already played the match.

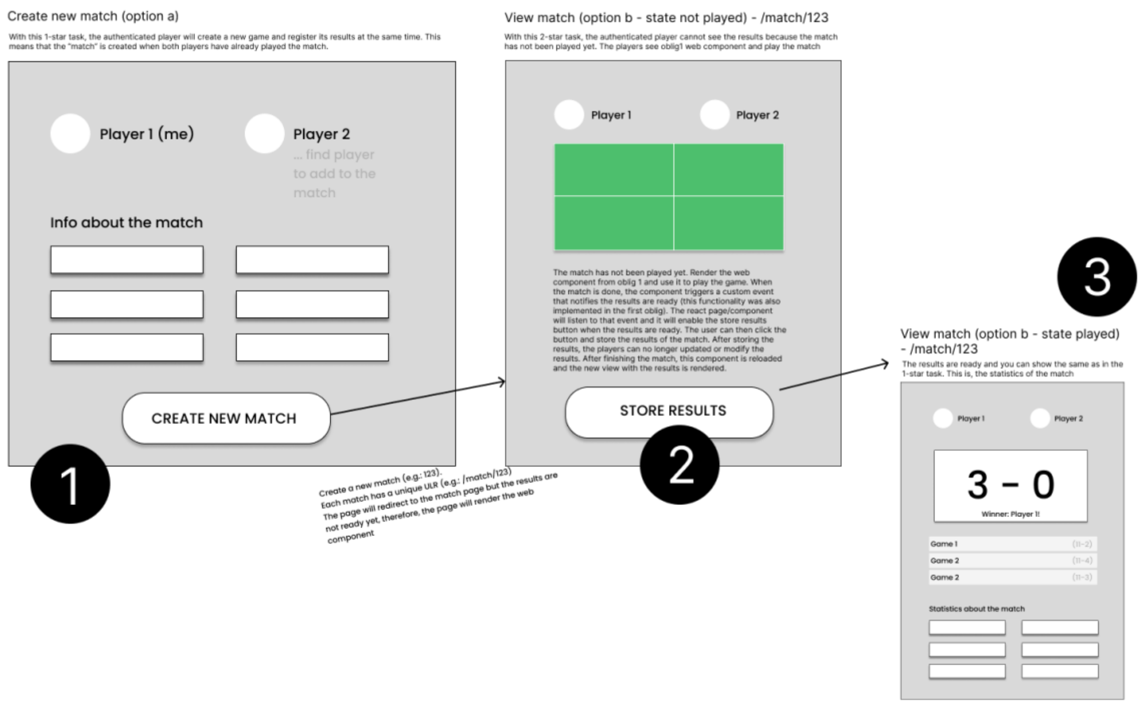
Create a new match (e.g.: 123).
Each match has a unique ULR (e.g.: /match/123)
The page will show the results of this match and some statistics

View match (option a) - /match/123

With this 1-star task, the authenticated player can see the results of a match and all its metatada.

Player 1 (me)
Player 2
... find player to add to the match
Info about the match
Add Results of the match
Game 1 Player 1 11 Player 2 2 (11-2)
Game 2 Player 1 11 Player 2 4 (11-4)
Game 2 Player 1 11 Player 2 3 (11-3)
CREATE NEW MATCH
Player 1
Player 2
3 - 0
Winner: Player 1!
Game 1 (11-2)
Game 2 (11-4)
Game 2 (11-3)
Statistics about the match

Figure 1 Option (a) New match and results at the same time


Create new match (option a)

With this 1-star task, the authenticated player will create a new game and register its results at the same time. This means that the "match" is created when both players have already played the match.

View match (option b - state not played) - /match/123

With this 2-star task, the authenticated player cannot see the results because the match has not been played yet. The players see oblig1 web component and play the match

The match has not been played yet. Render the web component from oblig 1 and use it to play the game. When the match is done, the component triggers a custom event that notifies the results are ready (this functionality was also implemented in the first oblig). The react page/component will listen to that event and it will enable the store results button when the results are ready. The user can then click the button and store the results of the match. After storing the results, the players can no longer updated or modify the results. After finishing the match, this component is reloaded and the new view with the results is rendered.

Create a new match (e.g.: 123).
Each match has a unique ULR (e.g.: /match/123)
The page will redirect to the match page but the results are not ready yet, therefore, the page will render the web component

View match (option b - state played) - /match/123

The results are ready and you can show the same as in the 1-star task. This is, the statistics of the match

Figure 2 Option (b) New match without results + web component + results

Report

This section provides some information and suggestions about how to structure the report. The report should include motivation, process, description, and reflections. The intended audience is someone who knows nothing about the project. Therefore, the report should start with a description of the motivation for the work, including why the application was developed, what need it was designed for, and what problem it solves. The problem description should summarise your understanding of the problem given by the project description. The report should then describe the process, including the steps followed to design and develop the application, the technologies and tools used, and how the collaboration was done. After the motivation and process, the report should describe the final product using text, visualisations, charts, diagrams, screenshots, and code snippets. This section should give an overview of the work to someone who has not seen the actual application. Finally, the report should end with reflections, including a discussion of what went well and what could be improved in the project. It is also possible to reflect on the collaboration techniques used and how they could be improved in future work. This part of the report is not a list of problems encountered but a critical overview of learning moments. It is recommended to finish by recognising what parts of the projects are not ideal and how they could be improved in future work.

The report should follow the following outline:

1. Introduction (about 1 page): Describe the problem that the application is trying to solve.
2. Process (about 2 pages): Describe the process followed, including the steps taken, the technologies and tools used, and how the collaboration was done.
3. Product description (about 5-6 pages): Describe the final product using text, diagrams, screenshots, and code snippets.
4. Reflections (about 1 or 2 pages): Be critical and discuss what went well and what could be improved in future work. Reflect on collaboration techniques used and how they could be improved. Finally, recognise what parts of the projects are not ideal and how they could be improved in future work.
    a. In the first paragraph, provide a summary of the grade you aimed to achieve, and briefly describe the 1-star, 2-star, and 3-star tasks you attempted to implement. Additionally, reflect on the group's performance and assess to what extent you were successful in completing the assigned tasks.

Delivery

This assignment must be delivered in the **GitHub classroom** and **Inspera**.

- **GitHub classroom**: You only need to push your changes and commits to your Git repository to deliver the assignment in GitHub Classroom. Ensure your code's final version is merged into the `main` branch.
- **Inspera**: Before delivering the assignment in Inspera, ensure your project has all the files it needs. Delete any unnecessary file or info (e.g.: `node_modules/` folder). Zip the project and upload the file to Inspera.
- You must also include a short report (5-10 pages without including a cover page, table of contents, references, or attachments).