CMPS 350 – Web Development Fundamentals
Practical Final Exams

- The exam duration is 120 minutes. So, read the exam questions carefully and plan your time accordingly.
- Push your code to GitHub regularly (at least every 30 minutes) to avoid unpleasant surprises, as your computer might hang!
- The Exam is an open book. In case of plagiarism, both parties will receive 0 points. Hence do not share or receive any code from anyone.
- Once you complete the Exam, you should:
    - Add your name and a screenshot for each question to the provided testing sheet.
    - Push your code and testing sheet to your GitHub repo under the **final** subfolder.
    - Demo your work before leaving the Exam.

GOOD LUCK WITH YOUR EXAMINATION!

Preparation

1. Sync cmps350 Lab repo to get the Final Exam Files
2. Download the **Final Exam Folder** from GitHub and move it to your repository.
3. Open the **SquadBuilder** project and Run **npm install** to get all the dependencies
4. Create the .env file
5. Make sure you have the **Prisma** extension in your VS Code to get the Prisma code highlighting.
6. Add "**type" : "module**" into your **package.json** file
7. After you migrate your project, please make sure to remove the "**type:module**" or your project will not run when you type **npm run dev**

Squad Builder App

In this Exam, you will demonstrate your proficiency in front-end and back-end web development by creating a web application for squad building. The application will be built using Next.js, React, and Prisma. It will allow users to form teams. Before you start the task, a demo of the application's functionality will be presented, and you can refer to the accompanying figures for the UI design.

1. **Creating SquadBuilder database models [5 pts]:** You should create two models named [**Team** and **Player**] with the following properties. The Ids are **auto-generated**. **[5 pts]**

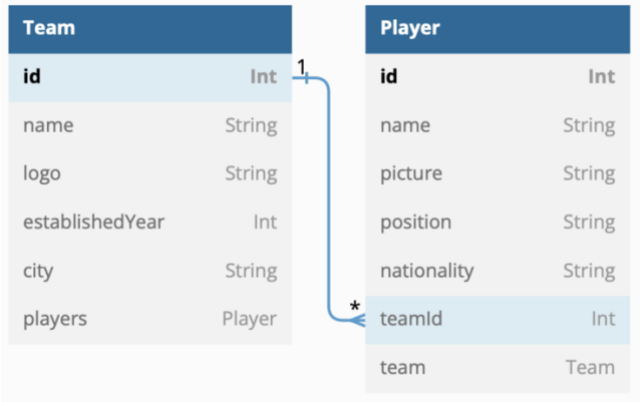

Figure 1Database structure

Important: Once you complete the design of the models, run the following commands to generate the tables corresponding to your Prisma Schema and seed your database with the provided data under the data folder.

npx prisma migrate dev --name init

Make sure you check if your database is properly seeded by executing the following command.
npx prisma studio

2. **Creating the Squad Builder app's repository [30 pts]:** Create a **squadBuilder-repo.js** file under **api/teams** folder and implement the following methods/functions.

Method / Functions	Description
getTeams()	Return all teams inside the database, including their players. Sort them in ascending order by name.
addTeam(team)	Adds new team to the database
deleteTeam(id)	Removes the matching team from the database
getTeamPlayers(teamId)	Returns all the players that belong to the given team. Sort them in descending order.
addPlayer(player)	Adds a new player to the database
updatePlayer(player, id)	Updates specific player

3. **Creating the SquadBuilder API Routes [30 pts]:** Implement handlers for the routes shown in the table below. The route handlers should use the methods or functions from the **SquadBuilder-repo** you implemented above.

HTPP Verb	URL	Functionality
GET	/api/teams	Return all teams
POST	/api/teams	Adds new team
DELETE	/api/teams/:teamId	Deletes team
GET	/api/teams/:teamId/players	Gets all the players that have the given team Id
POST	/api/teams/:teamId/players	Adds New player
PUT	/api/teams/:teamId/players/:pid	Updates Player

Important: Test your API's using Postman before moving to the next Step

4. **Creating the Squad Builder App User Interface [40 Pts , includes the Bonus]**

The **Player App should** have the following features shown below:

4.1. [15 pts] **Create Team List Page** When the root page loads, retrieve all the teams and players of those teams and display them, as shown in figure 2 below. Under each team, you should show all the players that belong to that team.

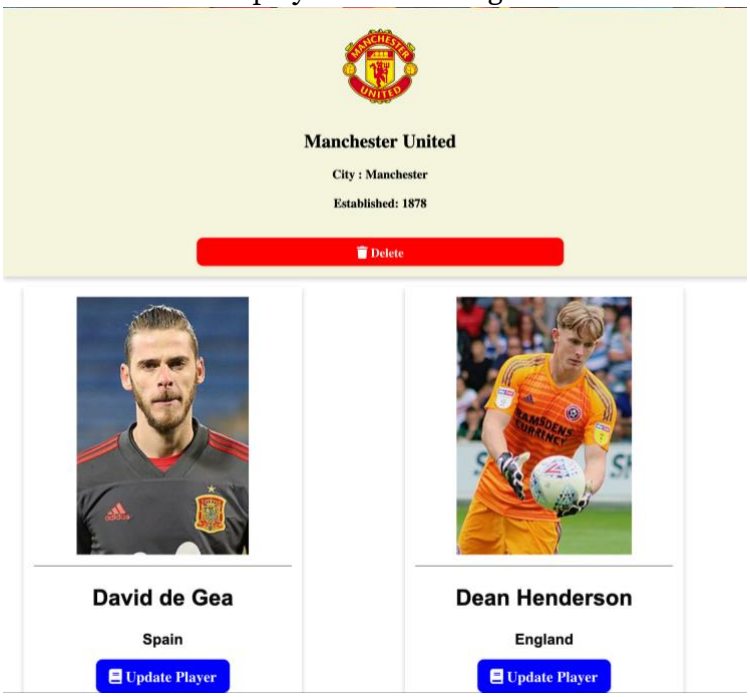

Figure 2 Home Page

4.2. [15 pts] **Delete Team**: within the team card, add a "**Delete Team**" button allowing users to delete a given team. Once the user confirms the deletion, the team and all its associated players will be promptly removed from the database. Subsequently, the **Team List Page** will be refreshed to reflect the updated information.

4.3. [10 pts] **Update Player**: Implement an Update button that allows the user to update a player of a team. The team id should be pre-populated and should be read-only.