UE21CS352B - Object Oriented Analysis & Design using Java

Mini Project Report

FOOTBALL LEAGUE MANAGEMENT

Submitted by:

MOHAMMED ZULQARNAIN	**PES1UG21CS340**
NANDAN KS	**PES1UG21CS360**
MALIK REHAN	**PES1UG21CS319**
MUKESH	**PES2UG21CS307**

6th Semester ***F*** *Section*

Prof. Bhargavi Mokashi
Assistant Professor

January - May 2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

1. Project Description: a football league management system featuring team, player, match, and league management capabilities. Admins can easily handle team creation, player transfers, match scheduling, and league standings through a user-friendly interface. Keep track of team details, player profiles, match schedules, and league standings effortlessly.

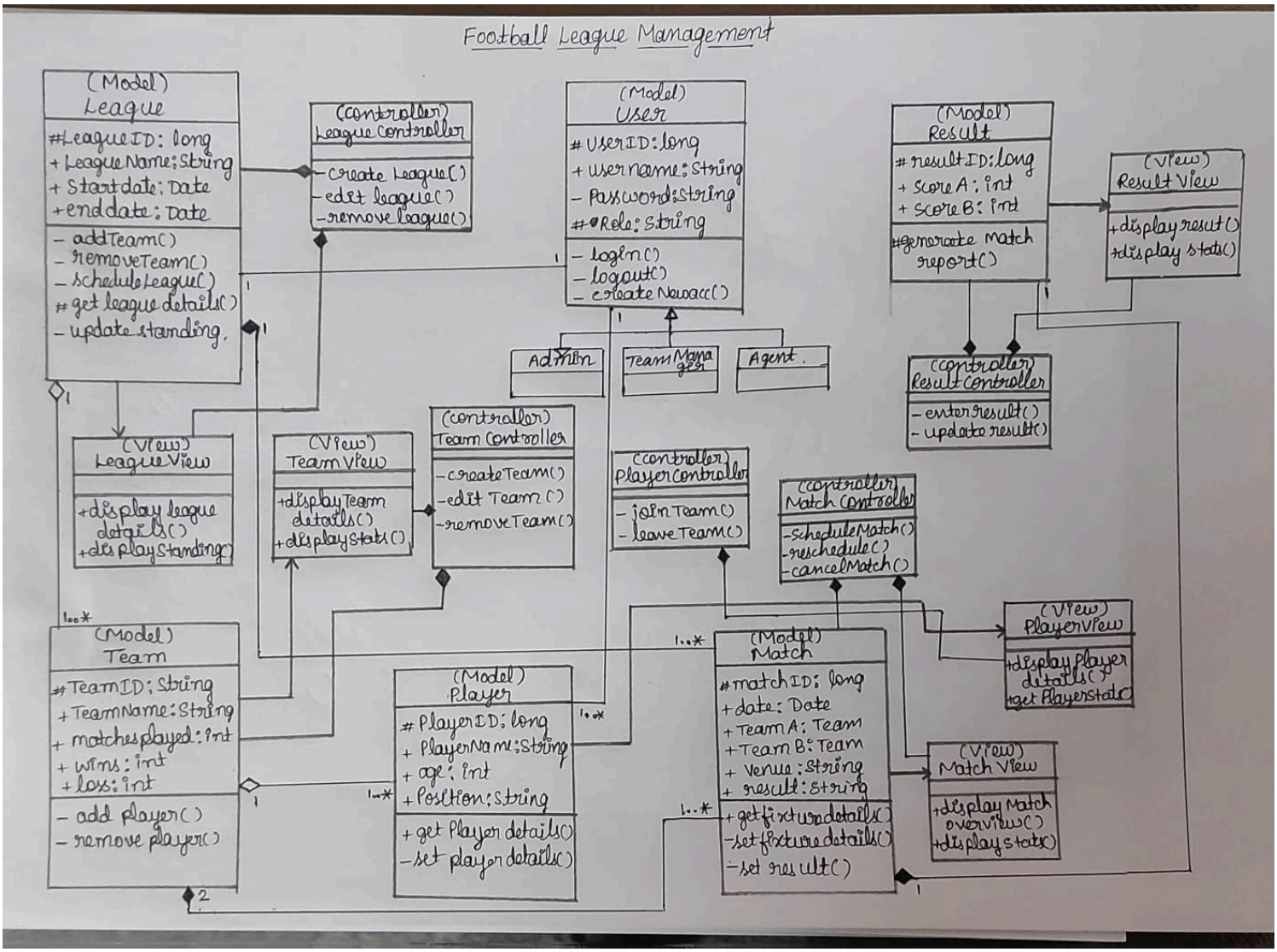
- Team Management: The system will allow the creation, modification, and deletion of football teams. Each team will have a name, coach, and list of players.
- Player Management: Players can be added to teams with details such as name, position, and age. Players can also be transferred between teams.
- Match Management: The system will schedule matches between teams, manage match results, and update team standings based on match outcomes.
- League Management: The system will support the creation of leagues, with multiple teams participating. It will track league standings, including points, goals scored, and goals conceded.
- User Interface: The system will provide a user-friendly interface for administrators to manage teams, players, matches, and leagues. Users will be able to view team details, player profiles, match schedules, and league standings.

2. Analysis and Design Models:

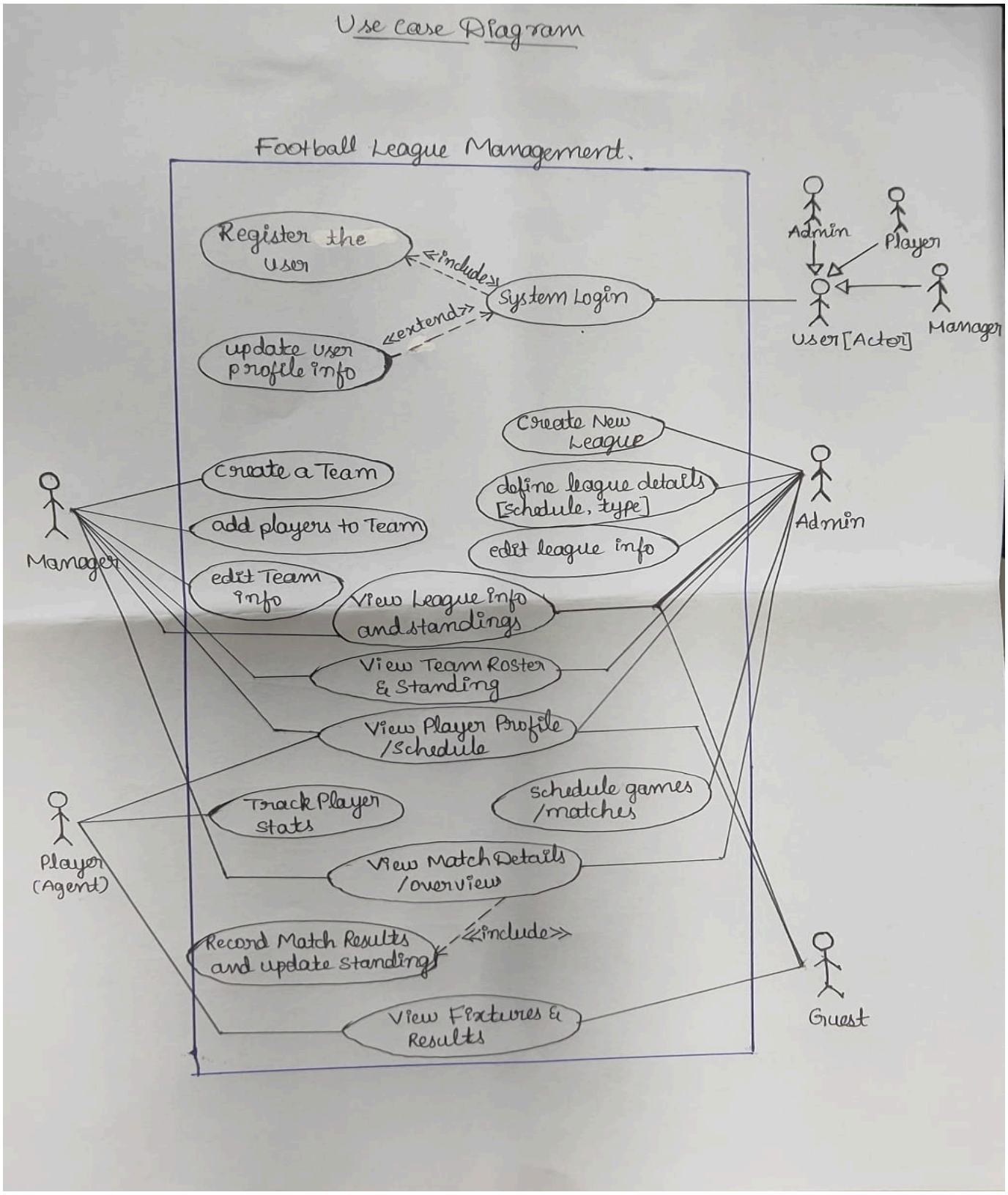
- Our architecture pattern is MVC.

- Decomposition of our project was done based on functionality. · The below diagrams give an understanding as to how we
- designed our application and its use cases.

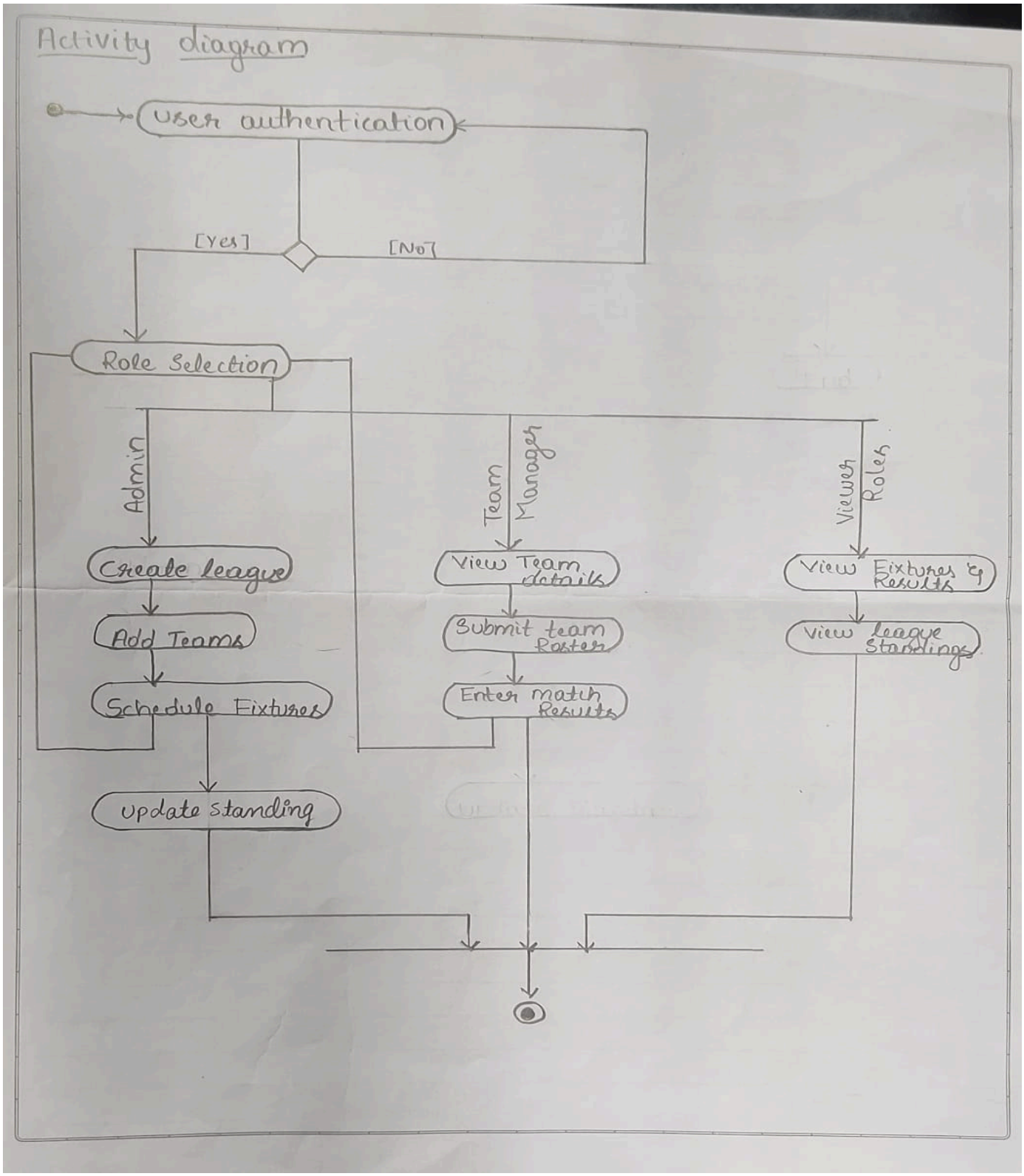
Class Diagram:



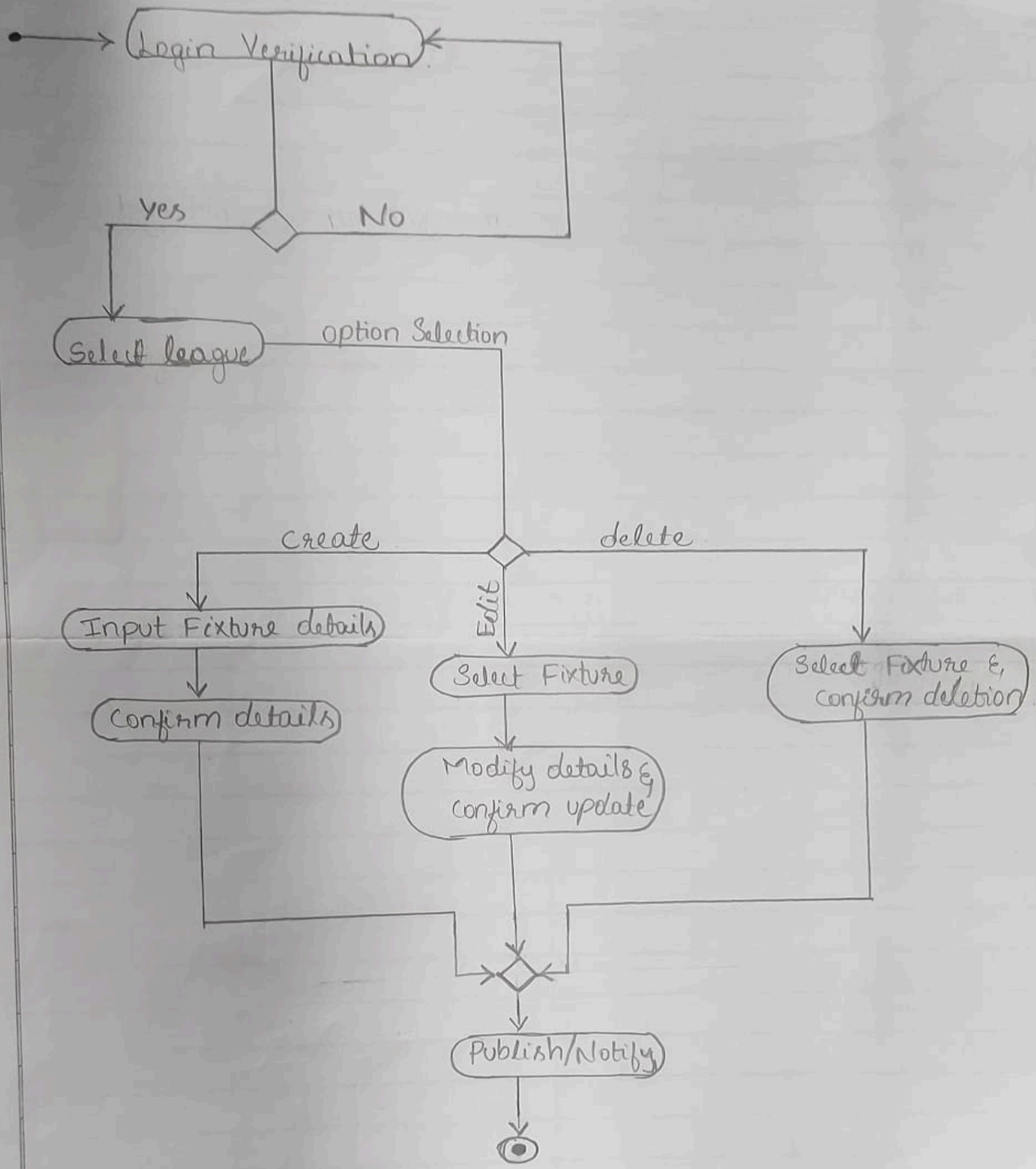
Use case diagram:



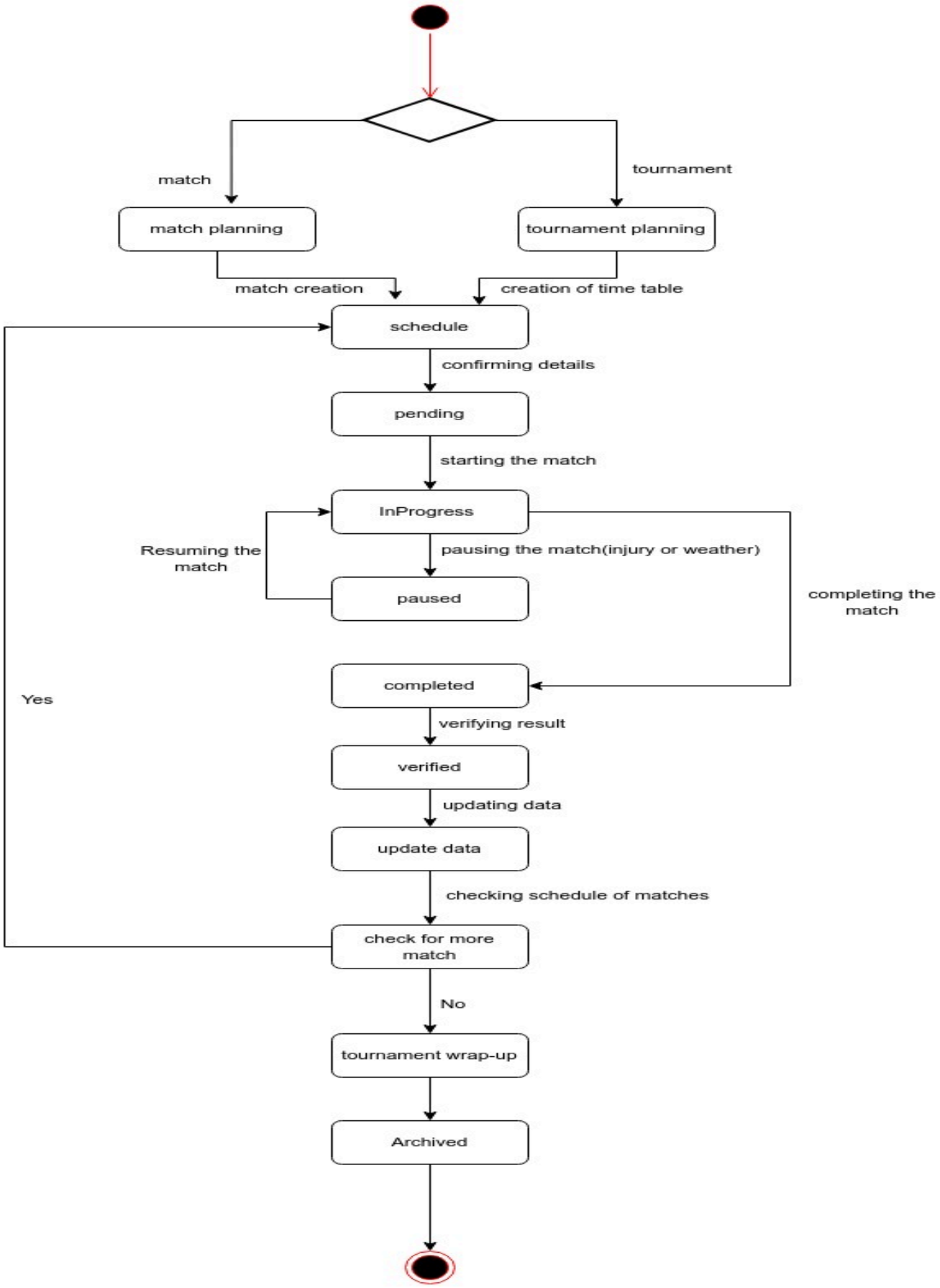
Activity Diagram:



Managing Match Fixtures



State Diagram:



Design Principles we used in out project:

- **MVC Design Pattern :**
  In adherence to the Model-View-Controller (MVC) design pattern, our project exhibits a structured separation of concerns, encompassing the Model, View, and Controller layers. The Model layer is represented by repository classes like LeagueRepository.java, TeamRepository.java, and PlayerRepository.java, which manage data persistence and retrieval for leagues, teams, and players respectively. The Controller layer orchestrates application behavior through files such as LeagueController.java, TeamController.java, PlayerController.java, and HomeController.java, handling user requests and responses. Complementing these layers, the View layer comprises HTML templates including index.html, player.html, league.html, addPlayer.html, team.html, etc., each associated with their corresponding controllers. These view files present dynamic content to users, ensuring a seamless and intuitive user experience while maintaining a clear separation of concerns within our application architecture.

- **Open/Closed Principle (OCP):**
  our project adopts abstraction and inheritance to enable extension without modification of existing code. Through interfaces or abstract classes, we define contracts that encapsulate variation points, isolating core functionalities from potential changes. Leveraging polymorphism, new functionalities can be introduced by extending existing classes or implementing interfaces, promoting code reuse and flexibility. This approach ensures our system remains flexible, maintainable, and scalable, allowing seamless adaptation to evolving requirements without compromising existing functionality.

- **Builder Pattern:**
  In our project, we've implemented the Builder design pattern to streamline the creation of League objects. This pattern separates the construction logic into a separate League class, offering a fluent interface for setting attributes such as leagueName, startDate, and endDate. By utilizing method chaining and encapsulating the construction process, we ensure readability, maintainability, and flexibility in creating League instances. This approach simplifies object creation, promotes consistency, and enhances code maintainability across our

project.

Code :

```
public class League {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int leagueID;

    @Column(name = "leagueName", nullable = false, length = 255)
    private String leagueName;

    @Column(name = "StartDate")
    @Temporal(TemporalType.DATE)
    private Date startDate;

    @Column(name = "EndDate")
    @Temporal(TemporalType.DATE)
    private Date endDate;

    // Getters and Setters
    public int getLeagueID() {
        return leagueID;
    }

    public void setLeagueID(int leagueID) {
        this.leagueID = leagueID;
    }

    public String getLeagueName() {
        return leagueName;
    }

    public void setLeagueName(String leagueName) {
        this.leagueName = leagueName;
    }

    public Date getStartDate() {
        return startDate;
    }

    public void setStartDate(Date startDate) {
        this.startDate = startDate;
    }
    public Date getEndDate() {
        return endDate;
    }
    public void setEndDate(Date endDate) {
        this.endDate = endDate;
    }

}
```

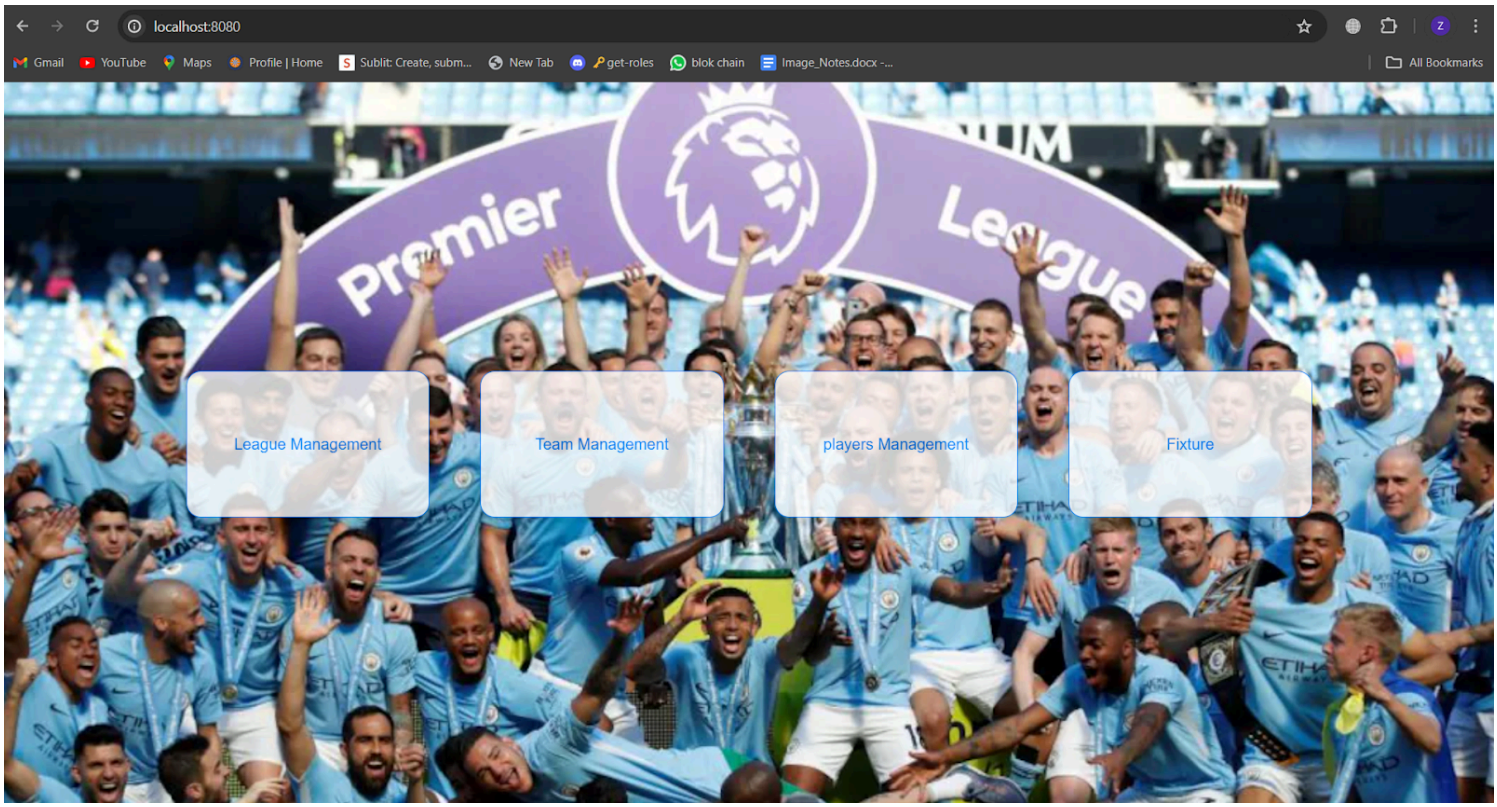
below we are using it

```
@PostMapping("/addleague")
public String createLeague(@RequestParam String leagueName, @RequestParam
@DateTimeFormat(iso = DateTimeFormat.ISO.DATE) Date startDate, @RequestParam
@DateTimeFormat(iso = DateTimeFormat.ISO.DATE) Date endDate, RedirectAttributes
redirectAttributes) {
   League newLeague = new League();

   // ******************implemented builder design pattern***************
   newLeague.setLeagueName(leagueName);
   newLeague.setStartDate(startDate);
   newLeague.setEndDate(endDate);
   leagueRepository.save(newLeague);
   redirectAttributes.addFlashAttribute("message", "League added successfully!");
   return "redirect:/league"; // Redirect back to the league page or a confirmation
page

}
```

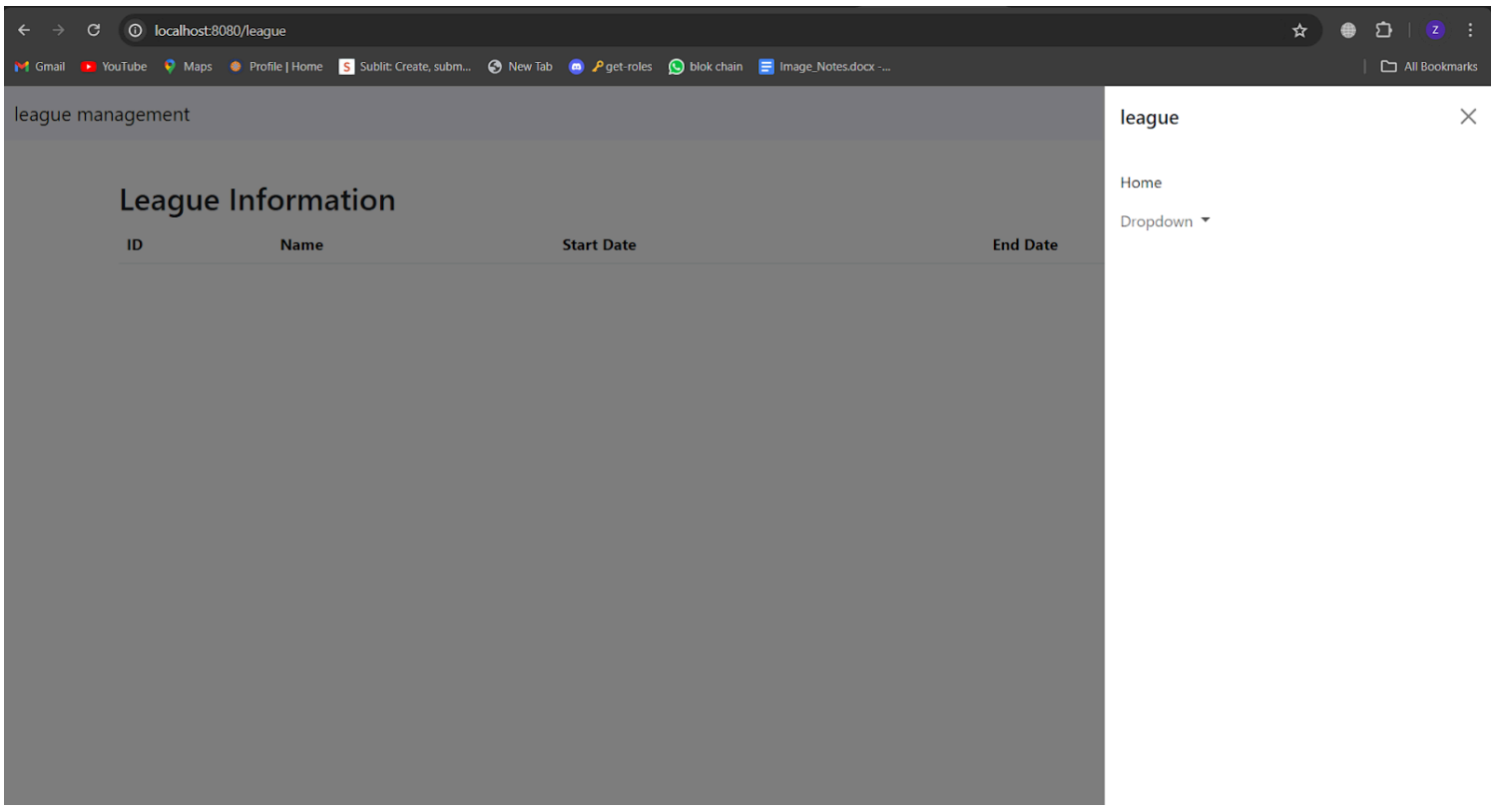
- **Facade design patter** :
  Implementing the Facade design pattern in our project enhances the overall structure and simplifies interactions between subsystems. The Facade pattern provides a unified interface to a set of interfaces in a subsystem, thus hiding its complexities from clients. In our project, we introduce a Facade class that serves as a single entry point to various subsystems or modules. This Facade class encapsulates the interactions with underlying components, shielding the client from the details of subsystem initialization, configuration, and usage. By providing a simplified interface, the Facade pattern promotes loose coupling and high cohesion, facilitating easier maintenance, testing, and future modifications. Additionally, it enhances readability and reduces dependencies, contributing to a more robust and scalable architecture. Overall, the adoption of the Facade design pattern streamlines system interactions and fosters a more efficient and manageable codebase.

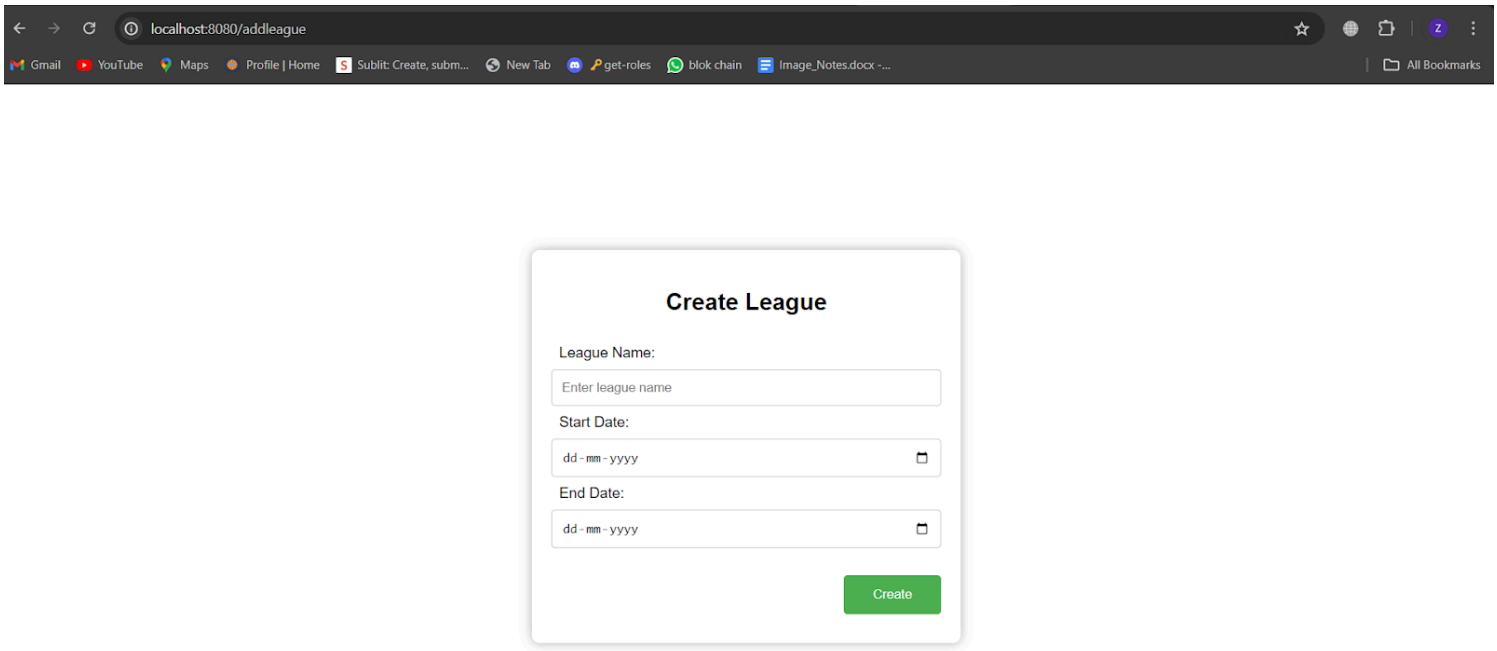
Screenshots of website

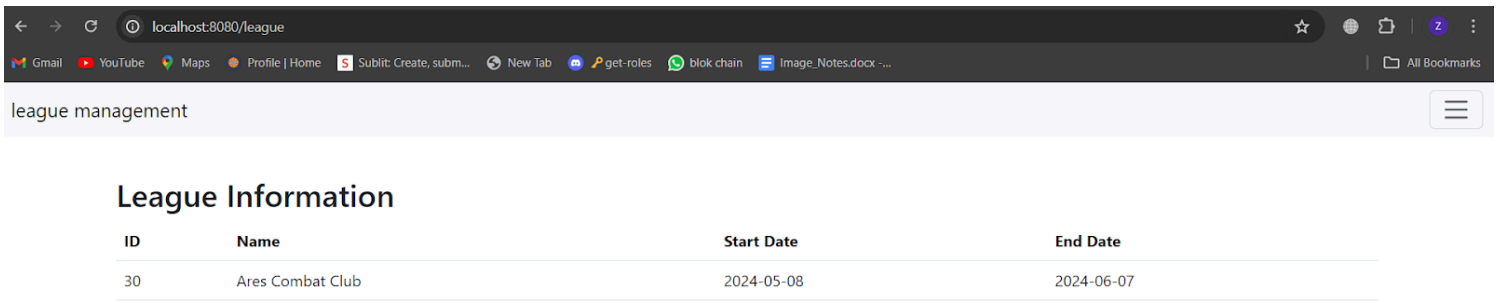
Home page



League management

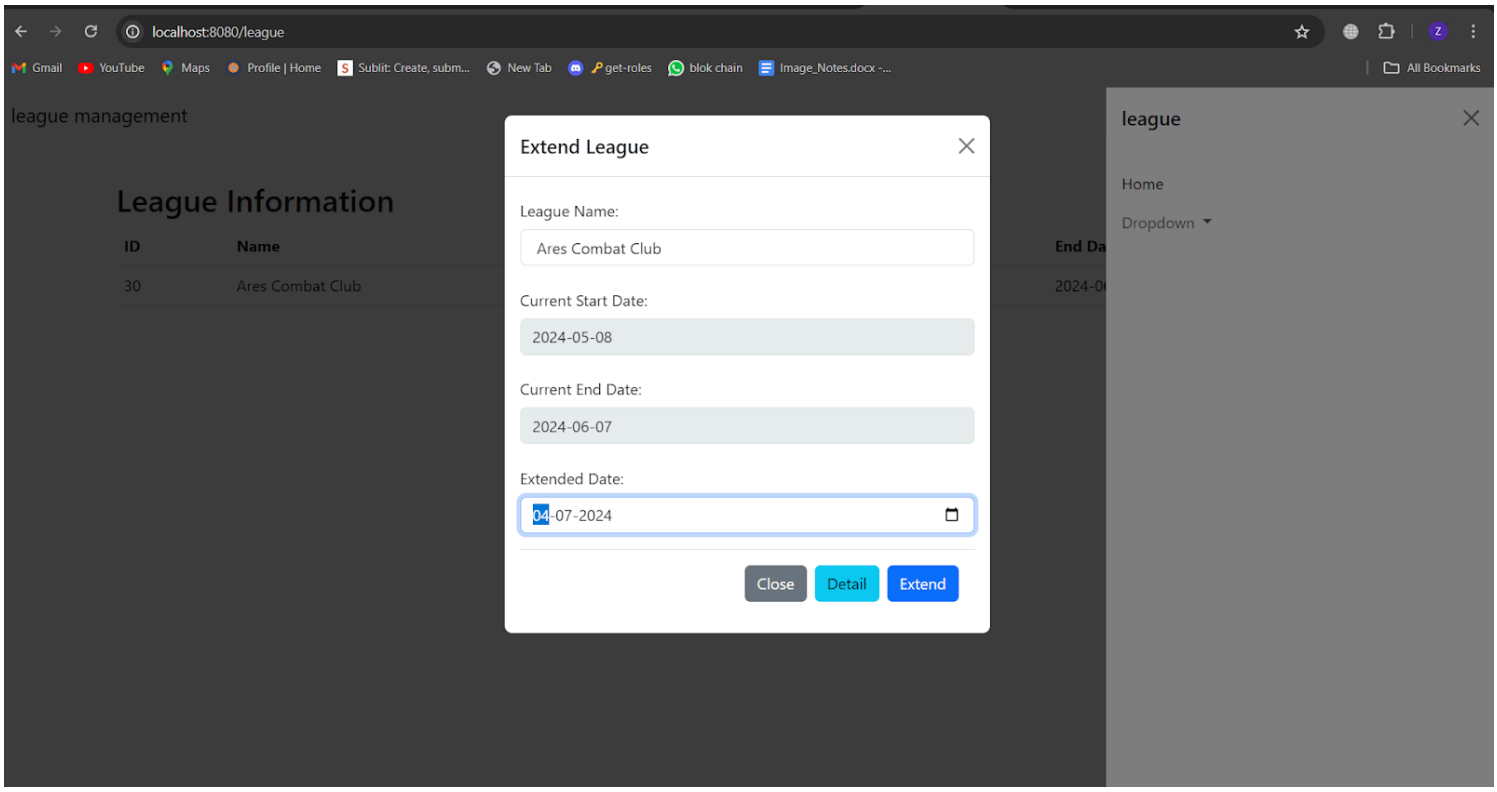


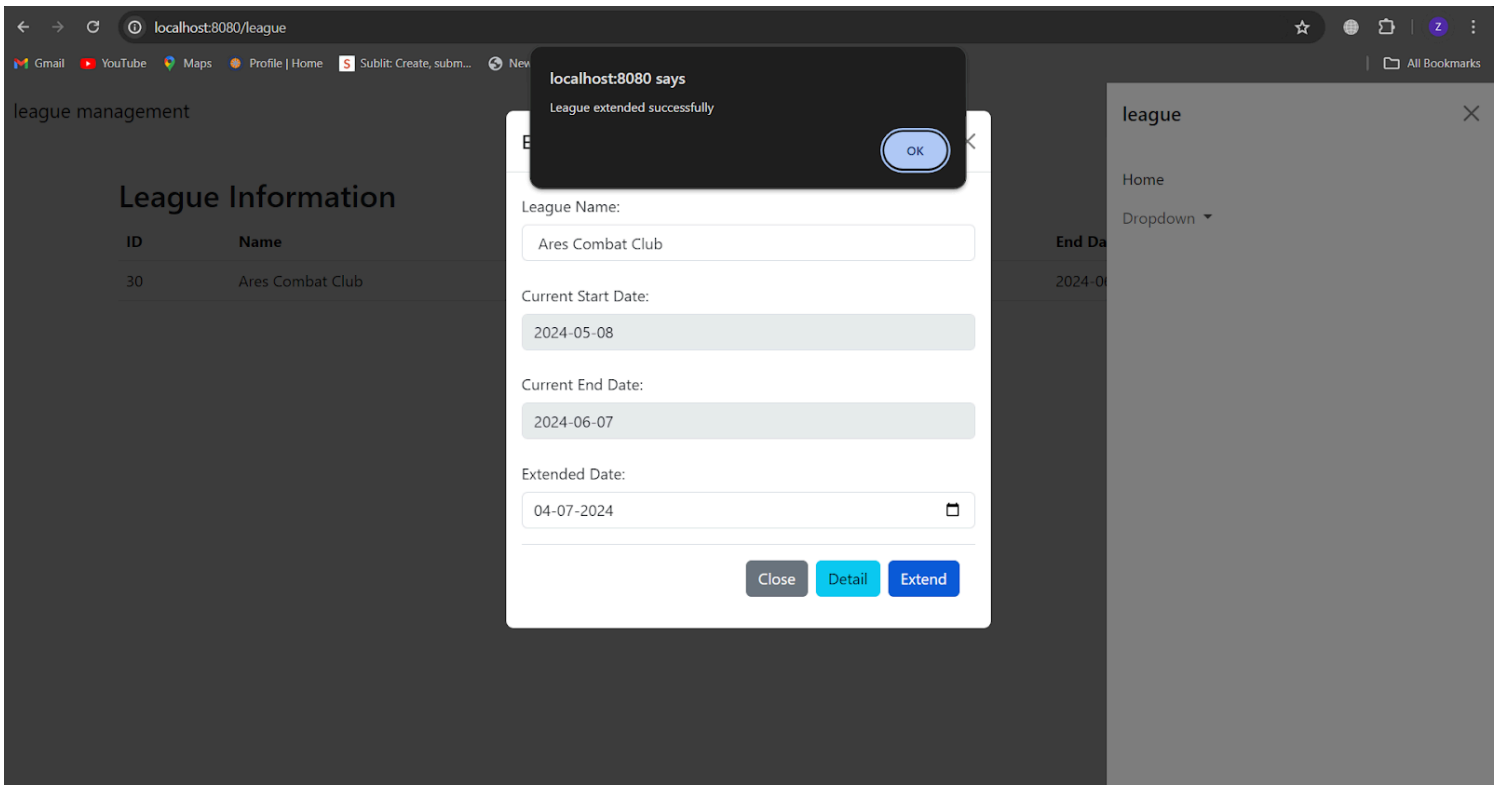


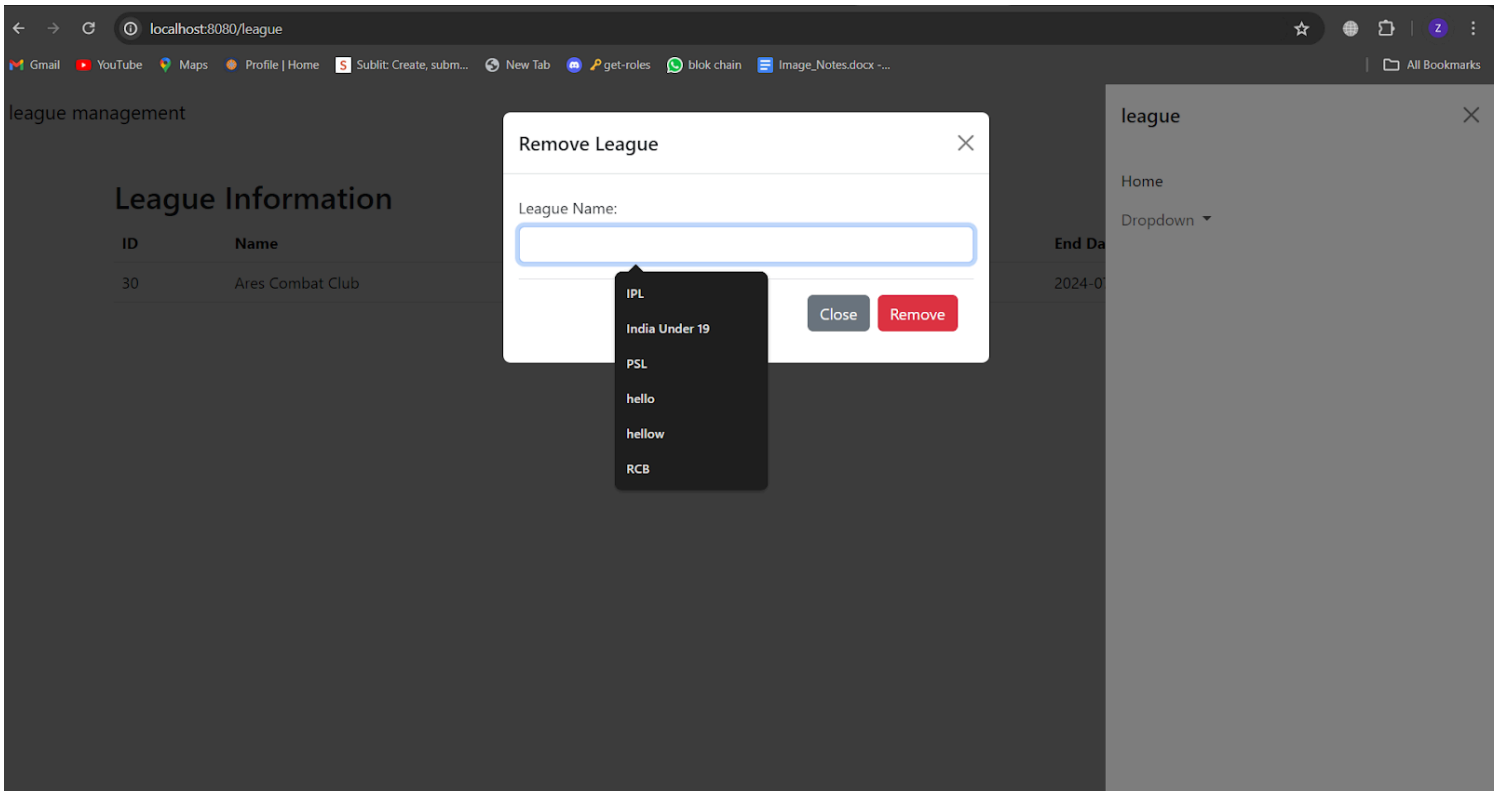


League Information

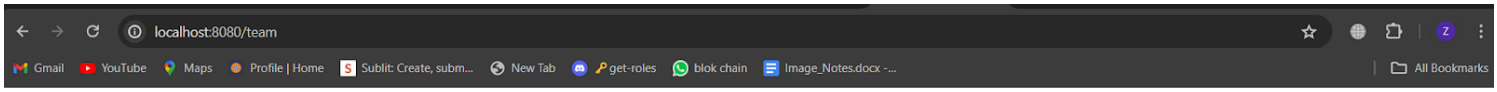
ID	Name	Start Date	End Date
30	Ares Combat Club	2024-05-08	2024-06-07

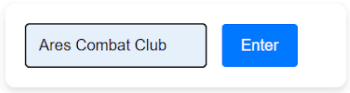

localhost:8080/league
Gmail
YouTube
Maps
Profile | Home
Sublit: Create, subm...
New Tab
get-roles
blok chain
Image_Notes.docx -...
All Bookmarks
league management
League Information
ID
Name
End Da
30
Ares Combat Club
2024-0
Extend League
League Name:
Ares Combat Club
Current Start Date:
2024-05-08
Current End Date:
2024-06-07
Extended Date:
04-07-2024
Close
Detail
Extend
league
Home
Dropdown

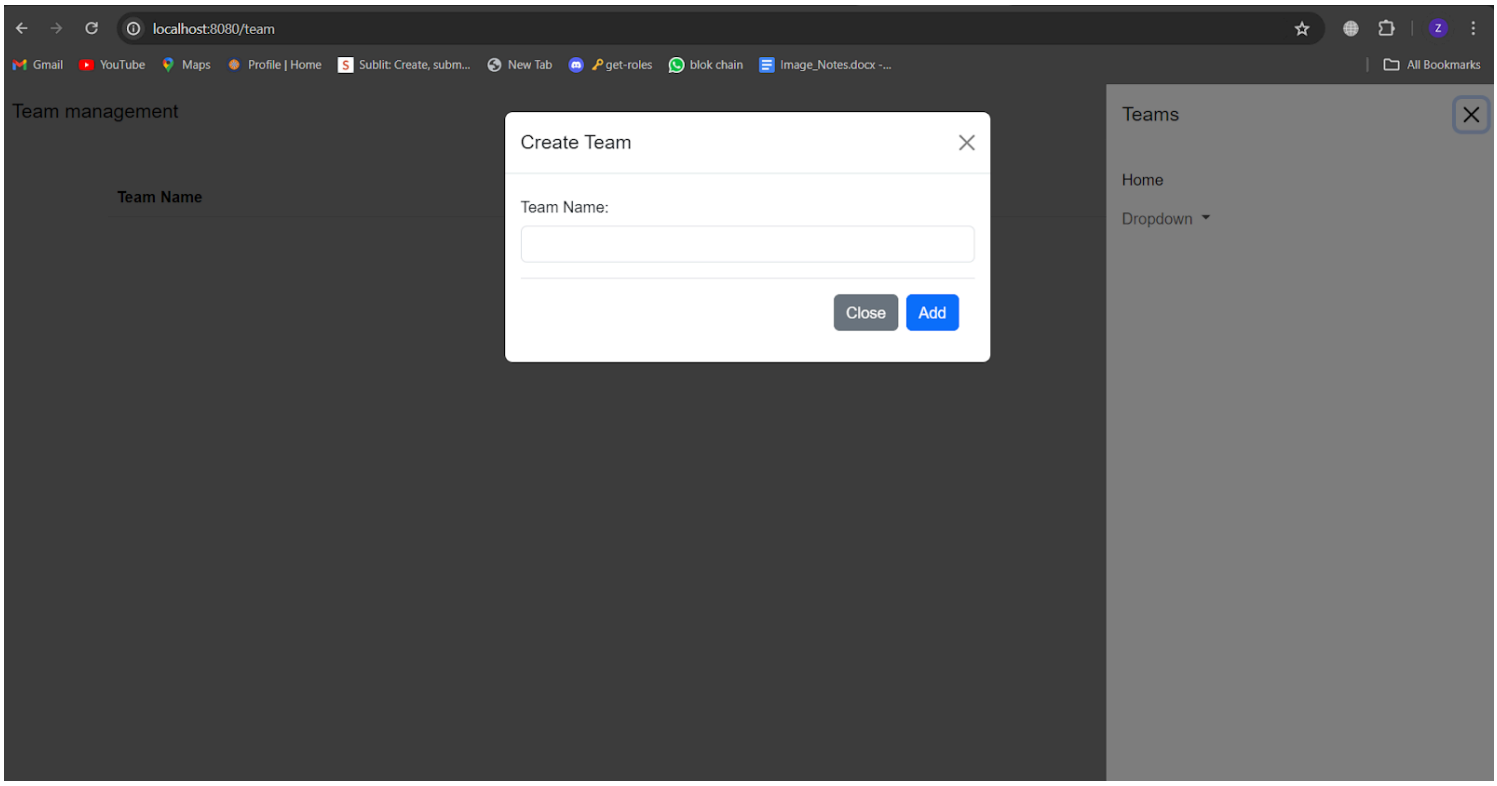

localhost:8080/league
Gmail
YouTube
Maps
Profile | Home
Sublit: Create, subm...
All Bookmarks
localhost:8080 says
League extended successfully
OK
league management
League Information
ID
Name
End Da
30
Ares Combat Club
2024-0
League Name:
Ares Combat Club
Current Start Date:
2024-05-08
Current End Date:
2024-06-07
Extended Date:
04-07-2024
Close
Detail
Extend
league
Home
Dropdown



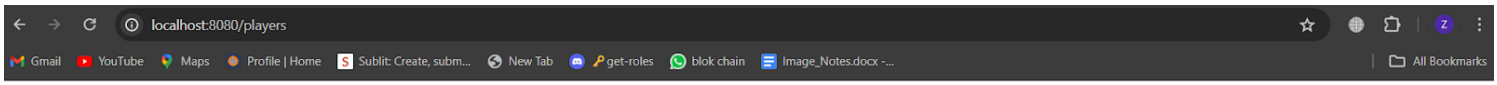
Team Management

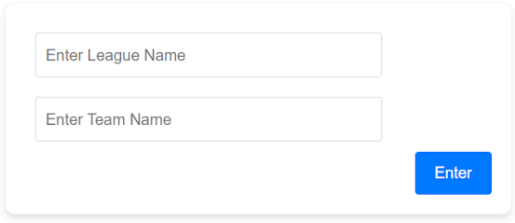


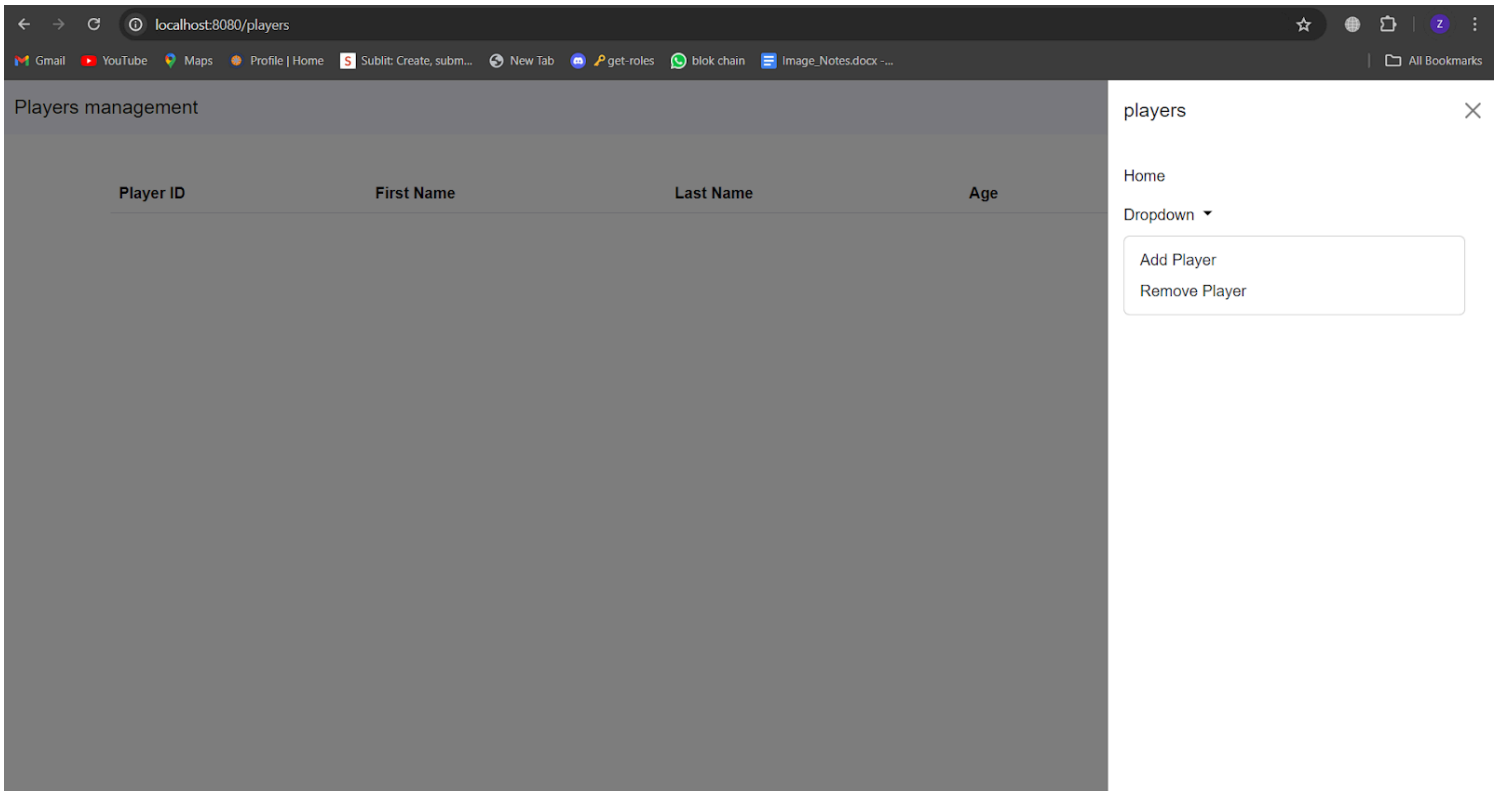




Player Management







Avien MySql Cloud Service