Assignment1

Tennis Management App

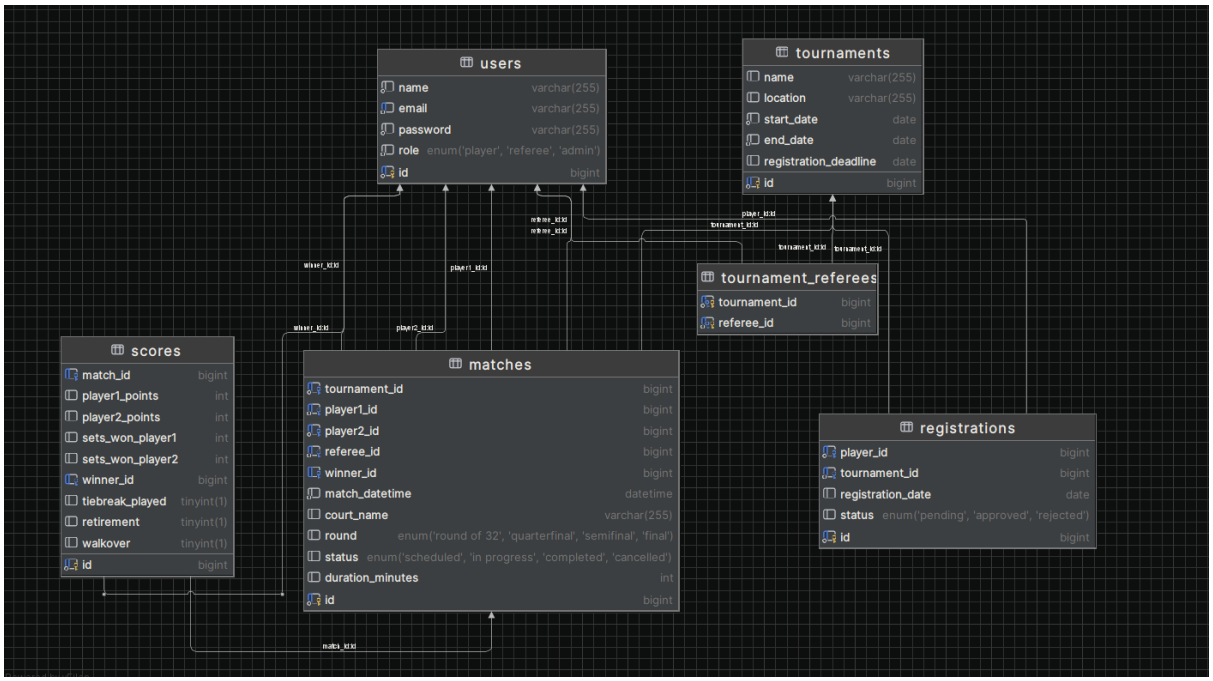
Girda Nora-30434

Project Overview

The Tennis Tournament Management System is a full-stack web application designed to streamline the organization of tennis tournaments. It supports three roles: Administrator, Referee, and Player. Each role has access to role-specific dashboards and capabilities. The system allows registration, tournament scheduling, referee assignment, match score updates, and viewing of match results and schedules.

Database diagram

The database diagram illustrates the relational schema used to support the Tennis Tournament Management System. It contains core entities such as User, Tournament, Match, Score, and Registration. Each entity is represented as a table with clearly defined primary keys and foreign keys to maintain referential integrity.



Use Cases and Use Case Diagram

Use Case: Register/Login

- **Actor**: All users
- **Description**: Allows users to register or log in using email and password.
- **Precondition**: None

- **Postcondition**: User is authenticated and redirected to the correct dashboard based on role.

Use Case: Register for Tournament

- **Actor**: Player
- **Description**: Player can register for an open tournament.
- **Precondition**: Tournament must be open and before the registration deadline.
- **Postcondition**: Registration entry is saved.

Use Case: View Schedule

- **Actor**: Player
- **Description**: Player views all matches scheduled in a tournament.
- **Precondition**: Matches must be generated.
- **Postcondition**: Schedule is displayed.

Use Case: View Scores

- **Actor**: Player
- **Description**: View score and match result once completed.
- **Precondition**: Match must be completed.
- **Postcondition**: Score and outcome are shown.

Use Case: View Assigned Matches

- **Actor**: Referee
- **Description**: View matches they have been assigned to.
- **Precondition**: Referees must be assigned by admin.
- **Postcondition**: Match list is shown.

Use Case: Submit Scores

- **Actor**: Referee
- **Description**: Input and update scores for matches in progress.
- **Precondition**: Match must be in progress and assigned.
- **Postcondition**: Scores are updated.

Use Case: Manage Users

- **Actor**: Admin
- **Description**: Create, edit, or delete users.
- **Precondition**: Admin must be authenticated.
- **Postcondition**: User info is updated in DB.

Use Case: Create Tournaments

- **Actor**: Admin
- **Description**: Create new tournaments with start/end dates and location.
- **Precondition**: Valid form inputs.
- **Postcondition**: Tournament saved.

Use Case: Assign Referees

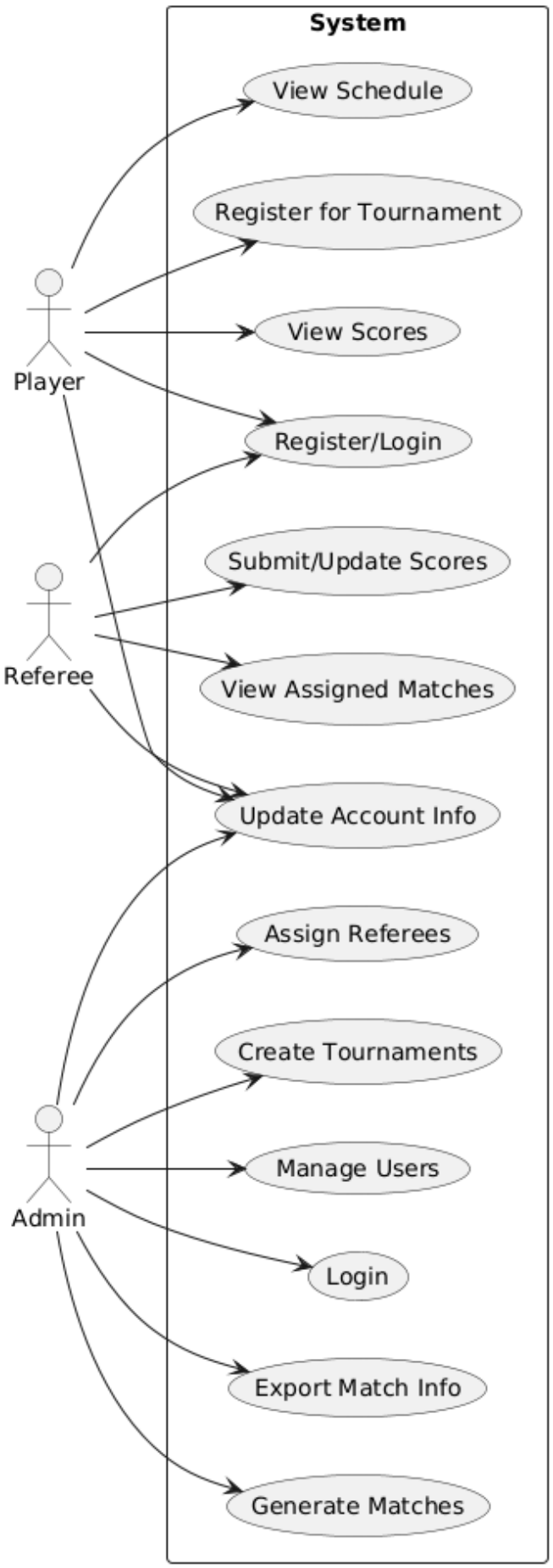
- **Actor**: Admin
- **Description**: Assign available referees to a tournament.
- **Precondition**: Referees and tournament exist.
- **Postcondition**: Assignment saved.

Use Case: Generate Matches

- **Actor**: Admin
- **Description**: Automatically create matches based on players registered.
- **Precondition**: Enough approved players.
- **Postcondition**: Matches are created.

Use Case: Export Match Info

- **Actor**: Admin
- **Description**: Export match data to CSV or TXT.
- **Precondition**: Matches exist.
- **Postcondition**: File downloaded.


System
View Schedule
Register for Tournament
View Scores
Register/Login
Submit/Update Scores
View Assigned Matches
Update Account Info
Assign Referees
Create Tournaments
Manage Users
Login
Export Match Info
Generate Matches
Player
Referee
Admin

Layers & Calss Diagram

The class diagram represents the **layered architecture** of the Tennis Tournament Management System and the relationships between its core components. It reflects how the application is organized into multiple logical layers to promote separation of concerns, maintainability, and scalability.

Controller Layer

- Contains REST controllers for each user role (AuthController, AdminController, PlayerController, RefereeController, UserController, MatchController).
- Responsible for handling HTTP requests, invoking service methods, and returning appropriate responses.

Service Layer

- Implements the business logic for each role and operation.
- Contains interfaces and their implementations (e.g., PlayerTournamentService, RefereeService, AdminService, UserService, MatchService).
- Applies design patterns (e.g., Strategy, Factory) to manage role-specific behaviors.

Model Layer (Entities)

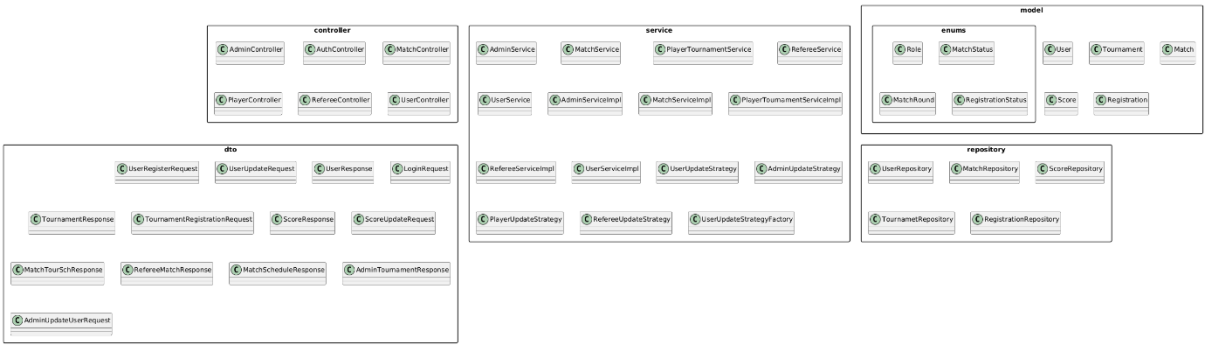
- Defines the domain entities such as User, Tournament, Match, Score, and Registration.
- Annotated with JPA annotations to map them to database tables.
- Includes relationships (e.g., OneToMany, ManyToOne, ManyToMany) between entities.

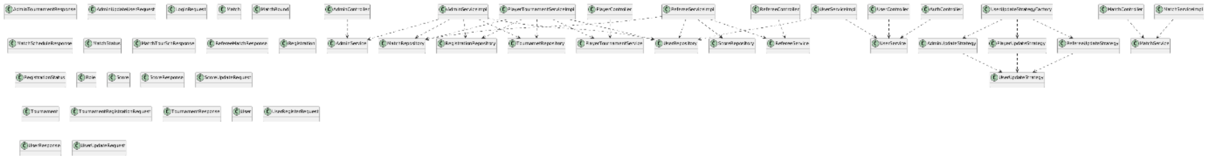
DTOs (Data Transfer Objects)

- Used to expose only necessary data to the frontend and hide internal models.
- Examples: UserResponse, LoginRequest, TournamentResponse, ScoreUpdateRequest, MatchScheduleResponse.

Repository Layer

- Contains interfaces extending JpaRepository for each entity.
- Enables CRUD operations and custom queries without manual SQL.




AdminTournamentResponse
AdminUpdateUserRequest
LoginRequest
Match
MatchRound
AdminController
AdminServiceImpl
PlayerTournamentServiceImpl
PlayerController
RefereeServiceImpl
RefereeController
UserServiceImpl
UserController
AuthController
UserUpdateStrategyFactory
MatchController
MatchServiceImpl
MatchScheduleResponse
MatchStatus
MatchTourSchResponse
RefereeMatchResponse
Registration
AdminService
MatchRepository
RegistrationRepository
TournamentRepository
PlayerTournamentService
UserRepository
ScoreRepository
RefereeService
UserService
AdminUpdateStrategy
PlayerUpdateStrategy
RefereeUpdateStrategy
MatchService
UserUpdateStrategy
RegistrationStatus
Role
Score
ScoreResponse
ScoreUpdateRequest
Tournament
TournamentRegistrationRequest
TournamentResponse
User
UserRegisterRequest
UserResponse
UserUpdateRequest