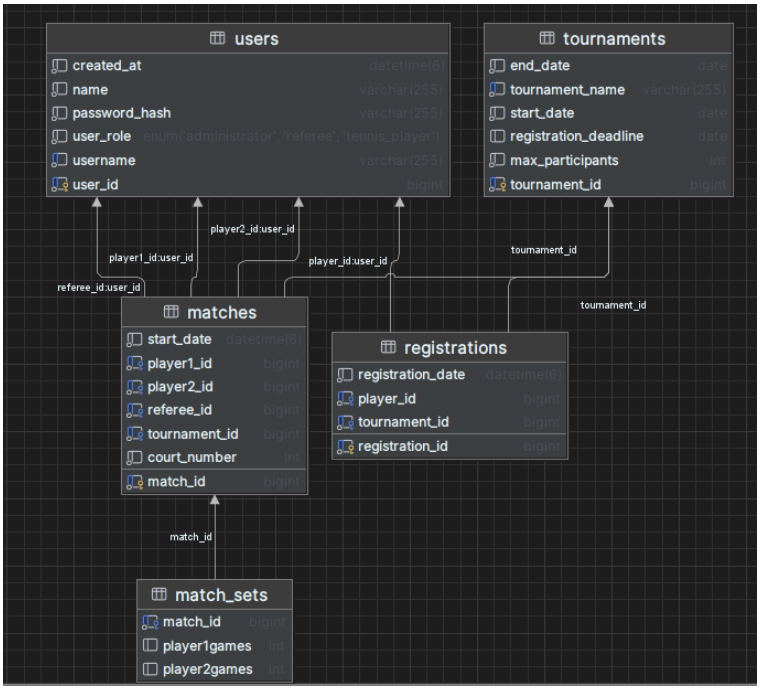
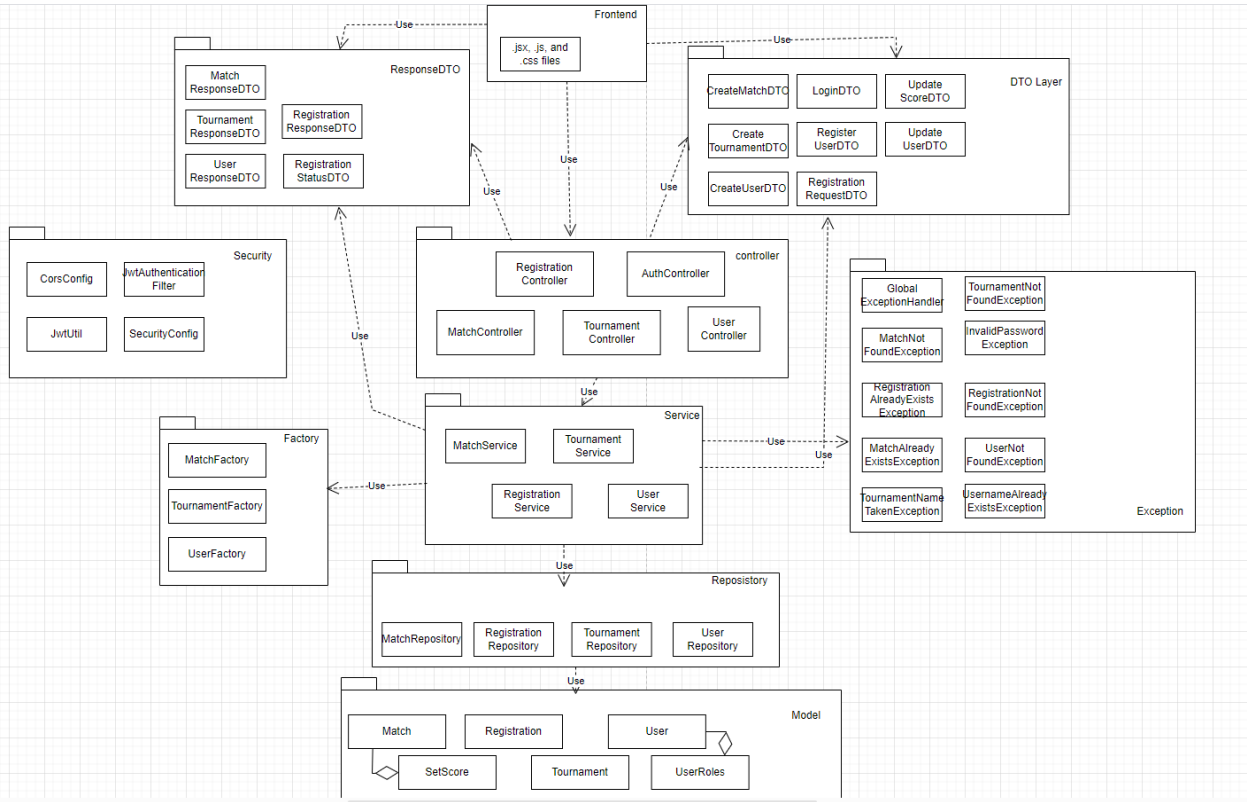
Assignment1
Tennis Tournament App

DATABASE DIAGRAM:



PACKAGE DIAGRAM:



The **Package Diagram** provides an overview of the high-level organization of the Tennis Tournament Management System. It highlights the **modular architecture** of the system, illustrating how major functional components (packages) are grouped and how they interact through **dependencies**. This structure supports separation of concerns, scalability, and maintainability.

Controller

Handles HTTP requests and delegates actions to the Service Layer. It includes:

- AuthController, UserController, MatchController, TournamentController, RegistrationController

These controllers serve as entry points for the application's REST API.

Service

Contains the business logic of the application. Services coordinate data flow between controllers and repositories:

- UserService, MatchService, TournamentService, RegistrationService

Each service focuses on one domain concept and uses the corresponding repository.

Repository

Implements data persistence by interfacing with the database using Spring Data JPA:

- UserRepository, MatchRepository, TournamentRepository, RegistrationRepository

These repositories manage the entities from the **Model** package.

Model

Defines the core domain entities:

- User, Match, Tournament, Registration, SetScore, UserRoles

These classes represent the system's core data structures.

DTO Layer

Holds **Data Transfer Objects** that structure the data exchanged between the frontend and backend:

- Request DTOs: CreateMatchDTO, LoginDTO, UpdateUserDTO, etc.
- Response DTOs: UserResponseDTO, MatchResponseDTO, etc.

Using DTOs ensures encapsulation and prevents exposing internal entity structures.

Factory

Responsible for converting between DTOs and entities (and vice versa):

- UserFactory, MatchFactory, TournamentFactory

This encapsulates transformation logic and simplifies the service layer.

Security

Configures the application's security using Spring Security and JWT:

- JwtAuthenticationFilter, SecurityConfig, JwtUtil, CorsConfig

It enables authentication and authorization across protected endpoints.

Exception

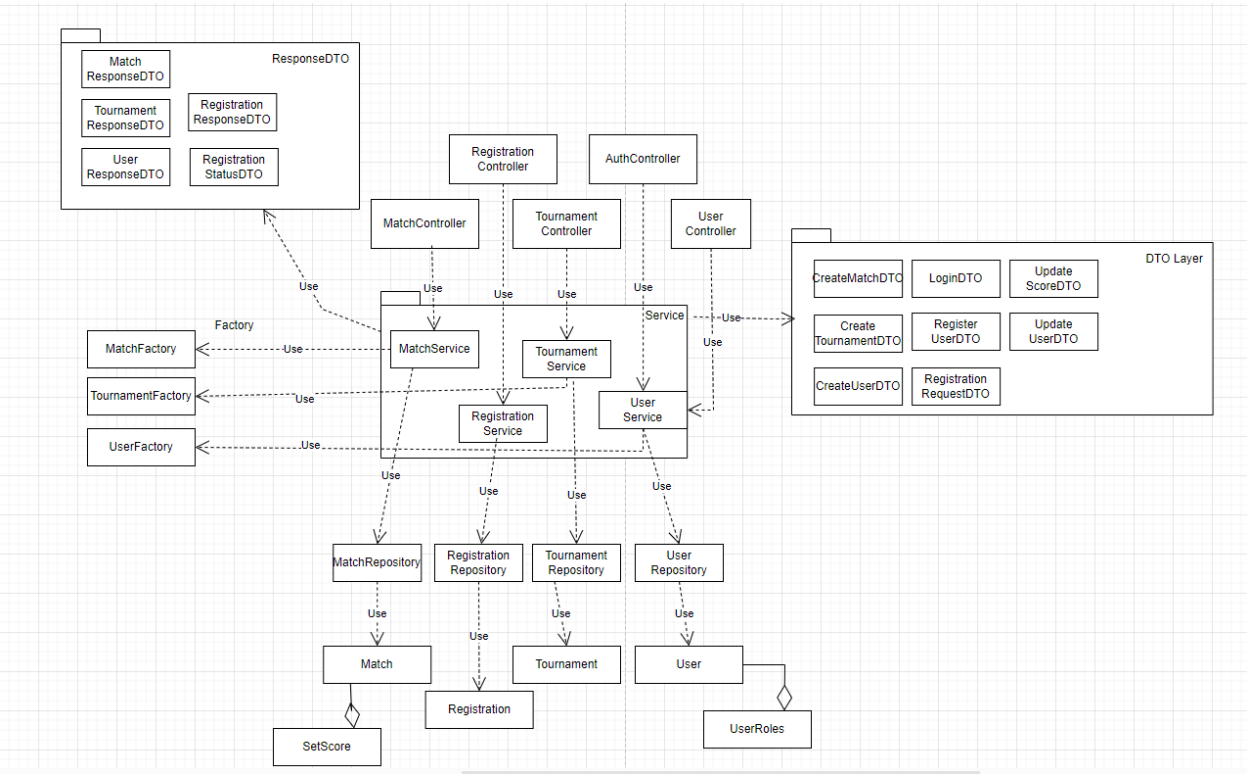
Contains centralized exception handling and custom exception classes:

- GlobalExceptionHandler processes all uncaught exceptions.
- Specific exceptions like UserNotFoundException, TournamentNotFoundException, etc., provide precise error reporting.

Frontend

Represents the client-side layer (React application), which interacts with the backend using DTOs and displays data through .jsx, .js, and .css files.

Class Diagram



The **Class Diagram** provides a detailed representation of the internal structure of the Tennis Tournament Management System, showcasing all the key classes, their responsibilities, and the relationships between them. It visually defines the software's static structure and supports understanding of how data and logic flow throughout the backend application

Controllers

The **controllers** are entry points for handling HTTP requests and routing them to the appropriate service classes.

- AuthController, UserController, MatchController, TournamentController, RegistrationController
- These classes are responsible for authentication, user management, tournament registration, and match processing.

Services

Services contain business logic and orchestrate operations between controllers, factories, repositories, and DTOs:

- UserService, MatchService, TournamentService, RegistrationService

Each service encapsulates a domain-specific task, such as handling user registration or processing tournament logic.

Factories

Factories convert between domain entities and DTOs. They are used by the service layer to prepare API responses and handle incoming request data.

- UserFactory, MatchFactory, TournamentFactory

This helps isolate mapping logic and supports better code reusability and testing.

DTO Layer

Data Transfer Objects (DTOs) represent structured request/response data between the frontend and backend. These classes include:

- **Request DTOs**: CreateMatchDTO, RegisterUserDTO, LoginDTO, etc.
- **Response DTOs**: MatchResponseDTO, UserResponseDTO, etc.

DTOs ensure that internal model structures are not exposed directly to the client, enforcing encapsulation.

Repositories

Repositories interact directly with the database and provide CRUD operations for models.

- UserRepository, MatchRepository, TournamentRepository, RegistrationRepository

These classes use Spring Data JPA to abstract away database queries.

Models (Entities)

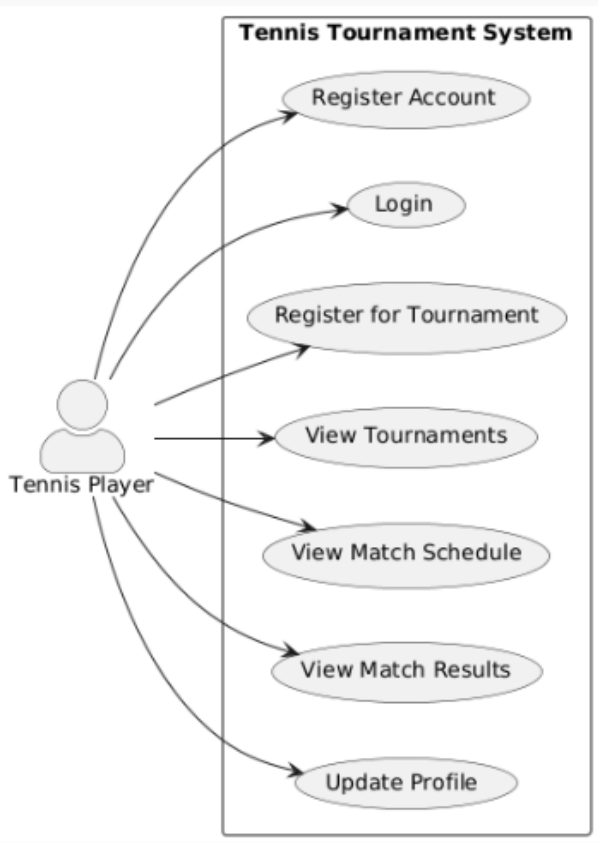
Entities represent persistent domain data and are managed by JPA.

- User, Match, Tournament, Registration, SetScore, UserRoles

Each entity has relationships:

- Match contains SetScore, and links to User (as players and referee) and Tournament
- Registration links a User to a Tournament
- User has a UserRoles enum to manage role-based access

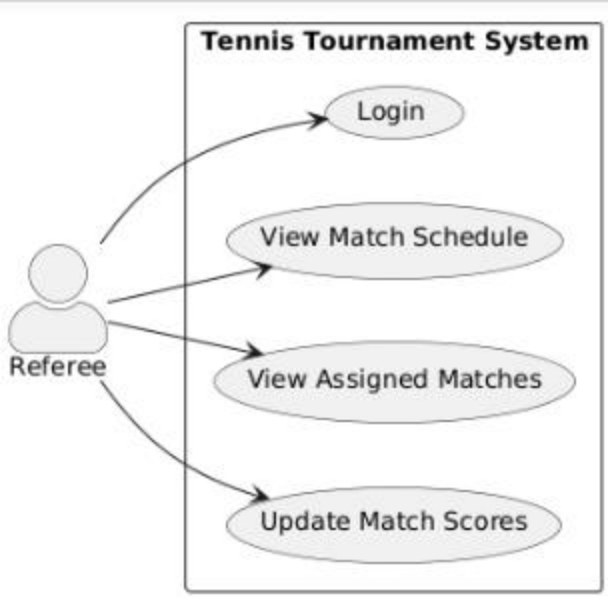
Use Case Diagrams:



The **Tennis Player** is a registered user who can participate in tournaments. Their interactions include:

- **Register Account**: Create a new user profile in the system.
- **Login**: Authenticate using their credentials.
- **Register for Tournament**: Sign up for open tournaments before the registration deadline.
- **View Tournaments**: Browse active and upcoming tournaments.
- **View Match Schedule**: Access personal match schedule and details.
- **View Match Results**: Check completed matches and results.
- **Update Profile**: Modify personal account information such as name or password.

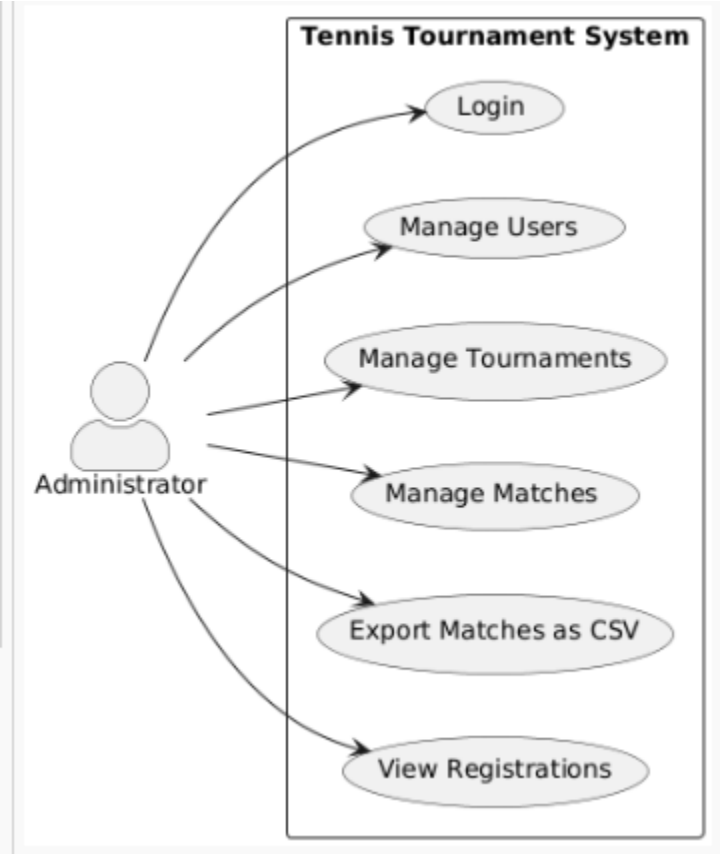
These use cases are designed to streamline the user experience for players, focusing on tournament participation and schedule tracking.



The **Referee** is responsible for overseeing matches and recording scores. Their use cases include:

- **Login**: Secure access to the referee dashboard.
- **View Match Schedule**: View all upcoming matches assigned to the referee.
- **View Assigned Matches**: Filter only the matches they are officiating.
- **Update Match Scores**: Submit or update scores of completed matches based on match progress.

This role ensures the integrity of the scoring process and helps maintain match data accuracy within the system.



The **Administrator** manages the system and has the highest level of privileges. Their available actions include:

- **Login**: Access the administrative dashboard.
- **Manage Users**: Create, update, or delete user accounts.
- **Manage Tournaments**: Organize and update tournament details (e.g., dates, max participants).
- **Manage Matches**: Schedule and delete match information including players, time, and court.
- **Export Matches as CSV**: Download all match data in CSV format for external analysis or record-keeping.
- **View Registrations**: Monitor the players registered for each tournament.

These administrative features ensure full control over the application's data and user management, supporting system scalability and reliability.