**Capstone Project Presentation**

Introduction:

My Spring Boot API project focuses on managing tournaments in various sports. The project revolves around four main entities: AppUser, Player, Team, and Event. These entities are designed to provide a comprehensive system for organizing and managing tournaments efficiently.

**Entities**:

1) **AppUser**:

* AppUser serves as the cornerstone of our project, acting as a central database for users involved in tournament management.
* Three concrete types (Spectator, Coach, and Admin) extend AppUser, each with specific roles and permissions.
* Authentication is based on user roles, allowing access to specific functionalities based on role authorization.

2) **Player**:

* Represents individual players across various sports.
* Users can perform READ operations to retrieve information about players.
* Admins have additional privileges for creating, updating, and deleting operations on player data.

3) **Team**:

* Represents teams participating in tournaments.
* Like players, users can perform READ operations to retrieve team information.
* Admins have exclusive rights for team management operations.

4) **Event**:

* Represents tournaments or events in different sports.
* Users can access event information through READ operations.
* Only Admins can perform event management operations such as creation, update, and deletion.

Endpoints:

I have implemented a series of endpoints to interact with our API:

1) /players:

* GET: Retrieve a list of all players.
* GET/{playerId}: Get details of a specific player.
* POST/create: Create a new player. (Requires admin privileges)
* DELETE/{playerId}/delete: Delete a player. (Requires admin privileges)
* PUT/{playerId}/update: Update player information. (Requires admin privileges)

2) /teams:

* GET: Retrieve a list of all teams.
* GET/{teamId}: Get details of a specific team.
* POST/create: Create a new team. (Requires admin privileges)
* DELETE/{teamId}/delete: Delete a team. (Requires admin privileges)
* PUT/{teamId}/update: Update team information. (Requires admin privileges)

3) /events:

* GET: Retrieve a list of all events.
* GET/{eventId}: Get details of a specific event.
* POST/create: Create a new event. (Requires admin privileges)
* DELETE/{eventId}/delete: Delete an event. (Requires admin privileges)
* PUT/{eventId}/update: Update event information. (Requires admin privileges)
* POST/{eventId}/start: Start an event. (Requires admin privileges)
* POST/{eventId}/stop: Stop an event. (Requires admin privileges)
* POST/{eventId}/resume: Resume a paused event. (Requires admin privileges)
* POST/{eventId}/finish: Finish an event. (Requires admin privileges)
* POST/{eventId}/cancel: Cancel an event. (Requires admin privileges)
* PUT/{eventId}/score/update: Update score for an event. (Requires admin privileges)
* GET/{eventId}/score: View score for an event. (Open to spectators, coaches, and admins)

4) /appusers:

* All CRUD operations require admin privileges.

Security:

* Endpoint access is regulated based on user roles using Spring Security annotations.
* Users are authenticated and authorized to perform specific operations based on their roles.
* Passwords of users are securely hashed using BCrypt password encoder for enhanced security.

Database:

* I utilized PostgreSQL database for my production environment, seamlessly integrated with JpaRepository and Hibernate for entity management.
* For testing purposes, I leveraged H2 database along with Mockito extensions for efficient testing and development.

Conclusion:

My Spring Boot API project offers a robust solution for managing tournaments across various sports. With a well-defined entity structure, secure endpoints, and seamless database integration, my project provides a solid foundation for tournament management applications.