Design Description *version 2* *(November 28th, 2022)*

Project Code Defenders - Robo Tournament

Team Codebenders

POLITECNICO MILANO 1863

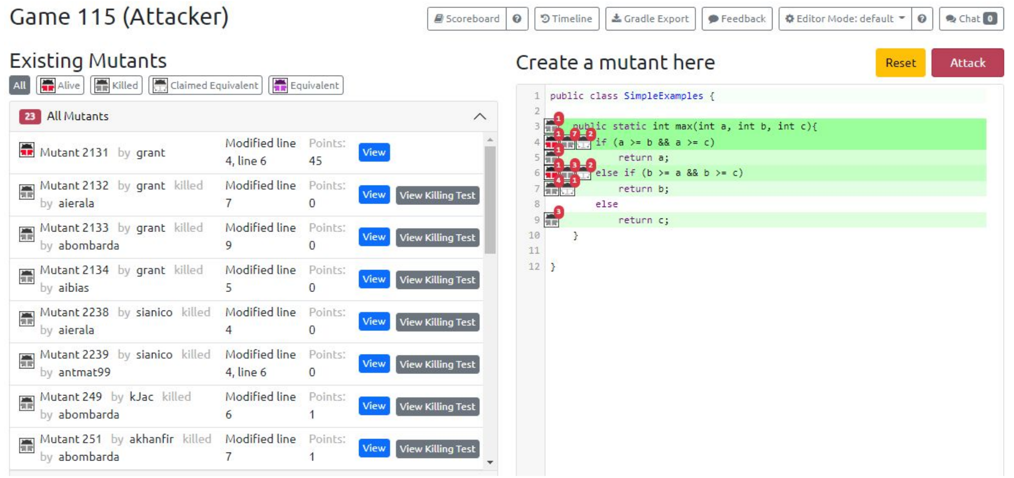
Mälardalen University

Table of content

Project vision

Code Defenders

- Software quality and testing are at the heart of software engineering, but they may not always get enough attention from software engineering education.
- CodeDefenders (web game) proposes the use of **gamification** to teach **mutation testing** and to strengthen code writing and testing skills.
- The game supports **team play and competition** by having Attackers - Defenders teams whose goal is to inject errors into code or write unit tests to catch them.
- The **"CodeDefenders: RoboTournament"** project aims at enriching the game by adding support for students tournaments and games against bots.



Game 115 (Attacker)

Existing Mutants

23 All Mutants

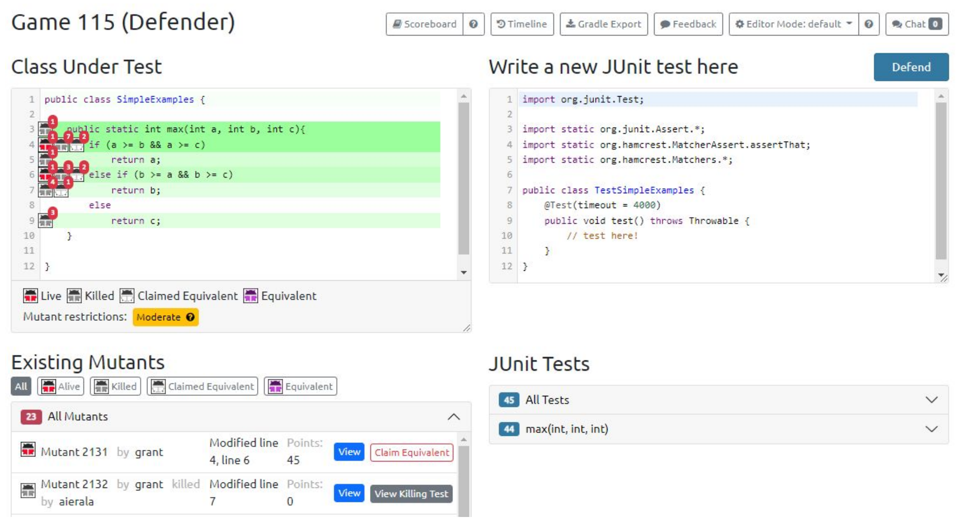
Mutant 2131 by grant	Modified line 4, line 6	Points: 45	View	
Mutant 2132 by grant killed by aierala	Modified line 7	Points: 0	View	View Killing Test
Mutant 2133 by grant killed by abombarda	Modified line 9	Points: 0	View	View Killing Test
Mutant 2134 by grant killed by aibias	Modified line 5	Points: 0	View	View Killing Test
Mutant 2238 by sianico killed by aierala	Modified line 4	Points: 0	View	View Killing Test
Mutant 2239 by sianico killed by antmat99	Modified line 4, line 6	Points: 0	View	View Killing Test
Mutant 249 by kJac killed by abombarda	Modified line 6	Points: 1	View	View Killing Test
Mutant 251 by akhanfir killed by abombarda	Modified line 7	Points: 1	View	View Killing Test

Create a mutant here

```java
public class SimpleExamples {

    public static int max(int a, int b, int c){
        if (a >= b && a >= c)
            return a;
        else if (b >= a && b >= c)
            return b;
        else
            return c;
    }

}
```



Game 115 (Defender)

Class Under Test

```java
public class SimpleExamples {

    public static int max(int a, int b, int c){
        if (a >= b && a >= c)
            return a;
        else if (b >= a && b >= c)
            return b;
        else
            return c;
    }

}
```

Live Killed Claimed Equivalent Equivalent

Mutant restrictions: Moderate

Write a new JUnit test here

```java
import org.junit.Test;

import static org.junit.Assert.*;
import static org.hamcrest.MatcherAssert.assertThat;
import static org.hamcrest.Matchers.*;

public class TestSimpleExamples {
    @Test(timeout = 4000)
    public void test() throws Throwable {
        // test here!
    }
}
```

Existing Mutants

23 All Mutants

Mutant 2131 by grant	Modified line 4, line 6	Points: 45	View	Claim Equivalent
Mutant 2132 by grant killed by aierala	Modified line 7	Points: 0	View	View Killing Test

JUnit Tests

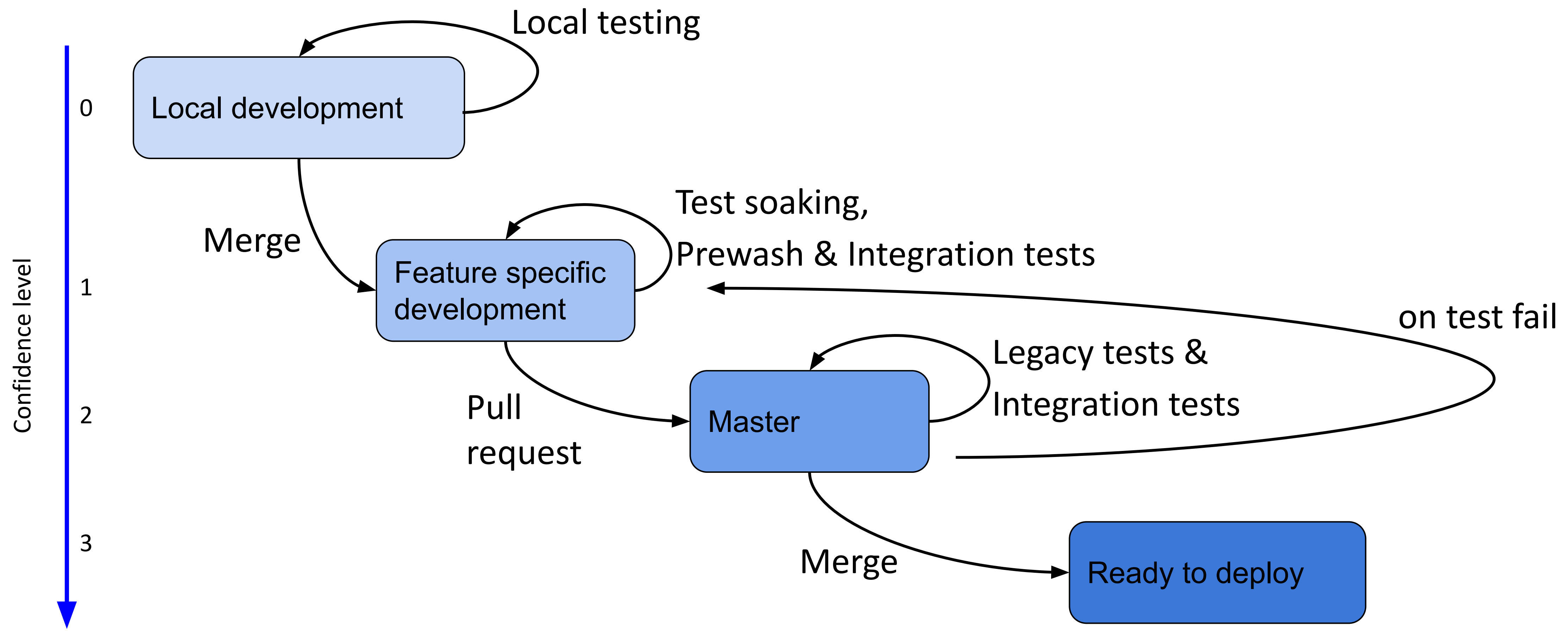
45 All Tests

44 max(int, int, int)

POLITECNICO MILANO 1863

Mälardalen University

Development and Testing workflow



Project requirements

- Design and implement a set of **OpenAPIs for CodeDefenders** which can be used from the tournament application to manage games and players.

- Implement the **tournament application**. This application must use CodeDefenders as a remote service (through the APIs) and must include at least two tournaments modalities.

- Implement a **streaming** component which allows users to follow in progress games live. This component can optionally include an "overall tournament view" showing schedule, standings and other information for each tournament.

- Design and implement a set of **APIs** which allows users to train **bots** over past games data and to let those bots play CodeDefenders.

POLITECNICO
MILANO 1863

Mälardalen University

What the project is not going to address

- The tournament application will be an **external application**, developed separately. It won’t be a plugin of CodeDefenders nor an application running on the same host.
- The tournament application will implement only the **tournament** and **streaming logic**. It won’t reimplement or modify in any way the game logic, which is already coded in CodeDefenders and will be accessible through our APIs.
- We won’t implement an **AI** playing CodeDefenders. This project requirement is **optional** and we are not planning to realize it because of the current lack of AI knowledge within our team.

- Streaming component will not respect hard real-time constraints

Desired functionalities (User Stories organized in Epics)

Tournament Management

CDF-32 Login/Register
CDF-41 Display tournaments info
CDF-33 Create Tournament
CDF-34 Join tournament
CDF-42 Matchmaking

Play games in a tournament

CDF-36 Starting games with notification
CDF-38 Return to tournament app from game page

Watch a streamed tournament game

CDF-39 View game stream
CDF-40 Notification of game stream update

Team Management

CDF-35 Team creation
CDF-54 Team management
CDF-37 Join team

Play with bots in a tournament

CDF-43 Bots can play
CDF-44 Bots can be trained

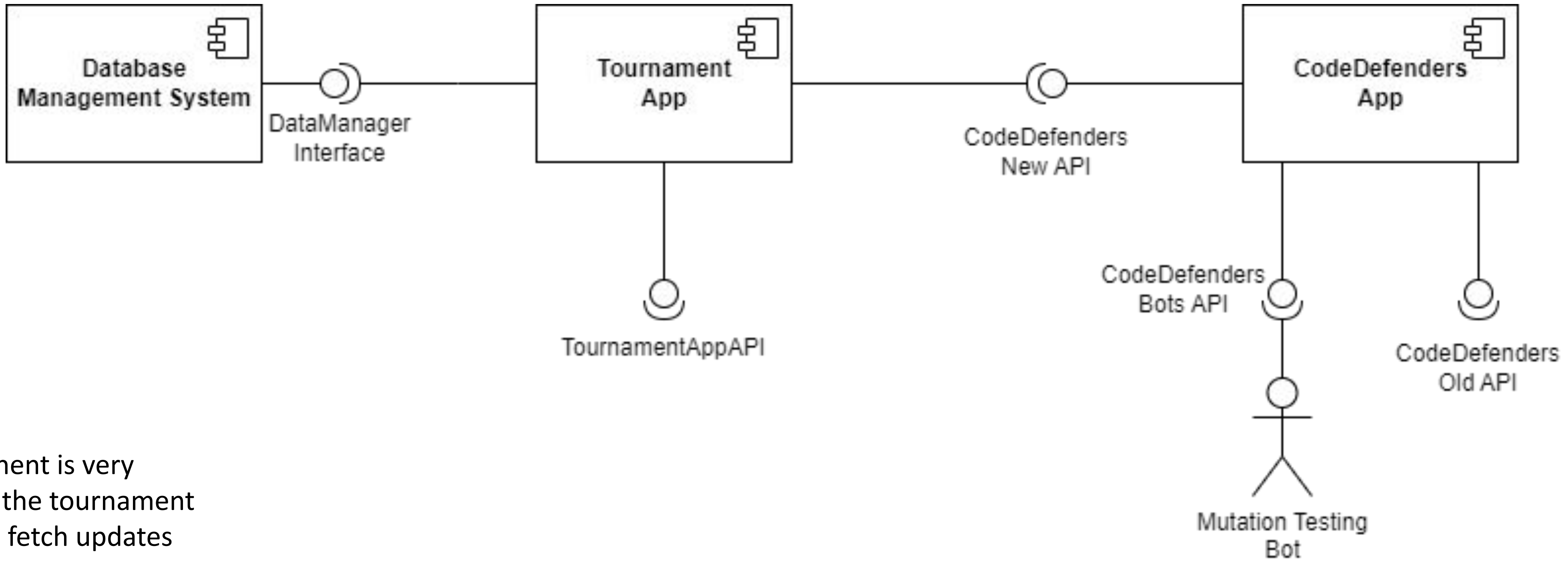
Avoid CodeDefenders overload

CDF-31 Low latency
CDF-69 Efficient flow of updates

POLITECNICO MILANO 1863

Mälardalen University

High-level component diagram



The streaming component is very likely to be internal to the tournament app, exploiting APIs to fetch updates from CD.

POLITECNICO MILANO 1863

Mälardalen University

Software architecture

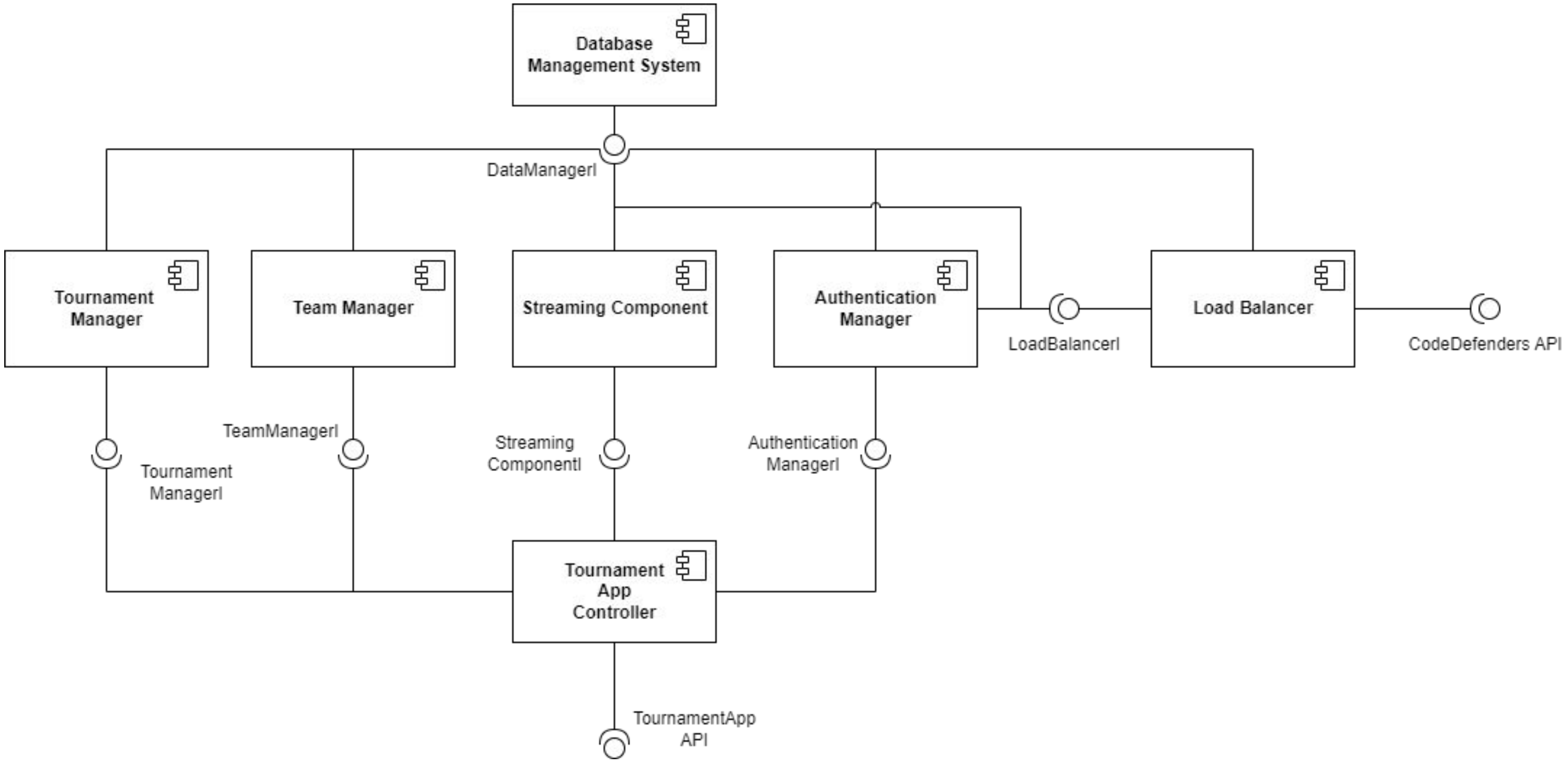
Each software component addresses the following user stories

- Tournament app:
  - CDF-41, CDF-33, CDF-34, CDF-42, CDF-35, CDF-37, CDF-54, CDF-36, CDF-38, CDF-39, CDF-40
- Code defenders APIs:
  - For bots: CDF-43, CDF-44
  - Other: CDF-32, CDF-39, CDF-36

POLITECNICO MILANO 1863

Mälardalen University

Zoom on the tournament app



Software architecture

Each software component addresses the following user stories

- Team manager:
  - CDF-35, CDF-37, CDF-54
- Tournament manager:
  - CDF-41, CDF-33, CDF-34, CDF-42
- Streaming component:
  - CDF-39, CDF-40
- Authentication manager:
  - CDF-32
- Tournament app, other: CDF-36, CDF-38

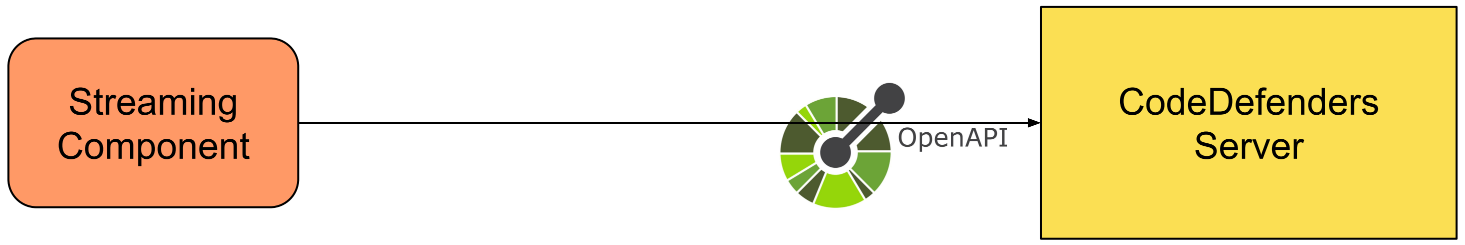
POLITECNICO MILANO 1863

Mälardalen University

Important design decisions

- Streaming component
- Load Balancer
- APIs for bots
- Database design

Streaming component (option 1)



The streaming component periodically updates (frequently to give the impression of real time):

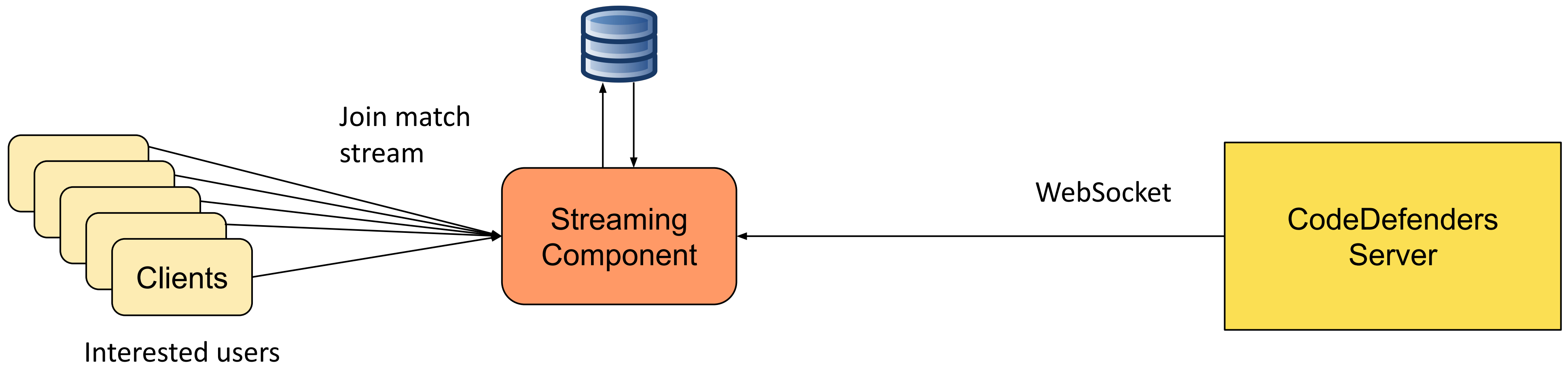
- Polls the CodeDefenders server to retrieve the events happened
- Only for the new events happened notifies the interested users

This solution would be really inefficient (streaming component continuously polling the server by calling the API) and also difficult to implement (we should keep track only of the new events and notifying only them).

POLITECNICO
MILANO 1863

Mälardalen University

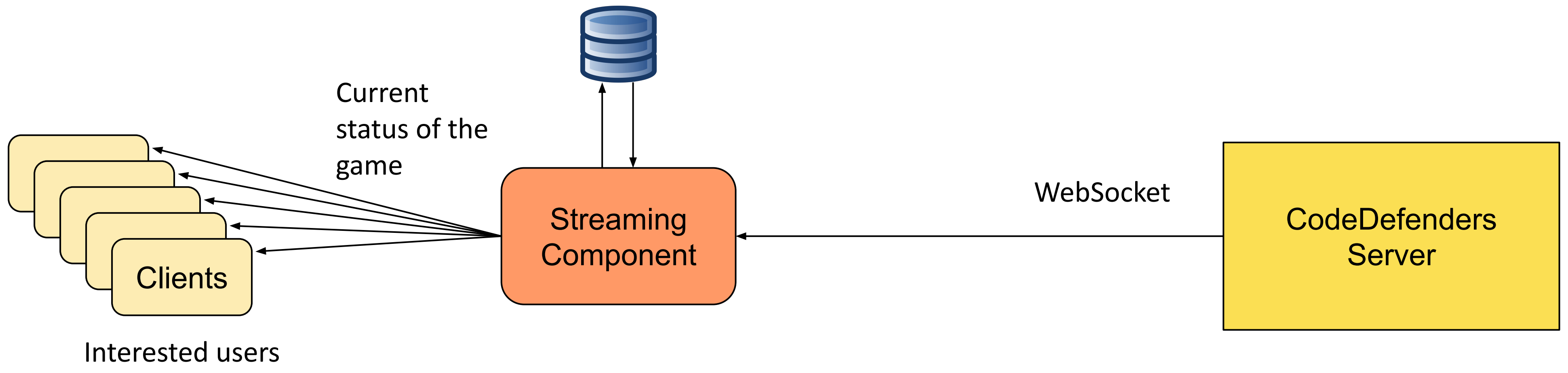
Streaming component (option 2)



CodeDefenders server establishes a WebSocket with the Streaming Component and sends events to it.

Clients interested in an ongoing game join the stream by sending a request to the Streaming Component.
It responds by querying its local db and retrieving the current status of the game (e.g. points for each team).
Then a WebSocket is established between each client and the Streaming Component.

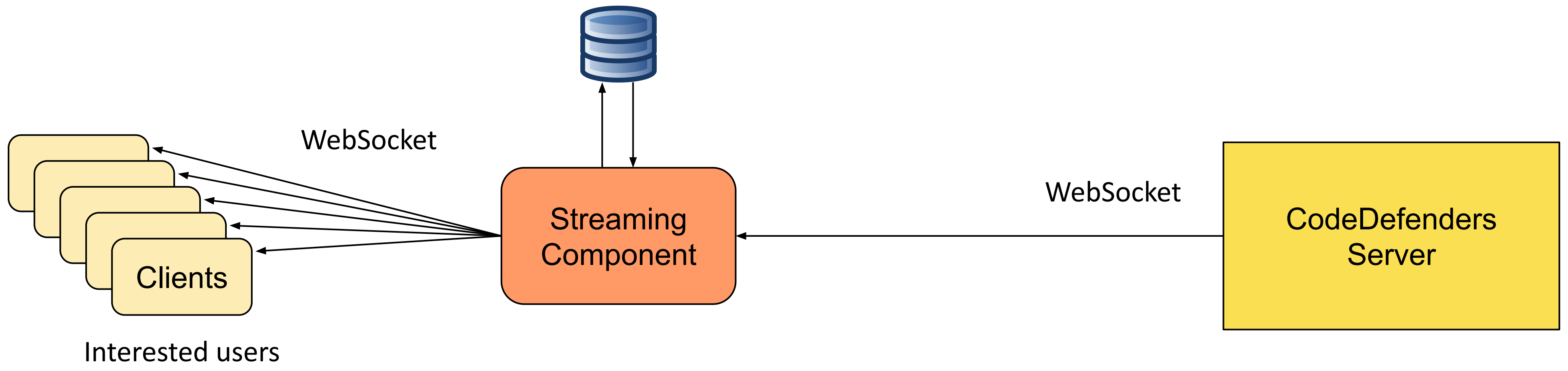
Streaming component (option 2)



CodeDefenders server establishes a WebSocket with the Streaming Component and sends events to it.

Clients interested in an ongoing game join the stream by sending a request to the Streaming Component.
It responds by querying its local db and retrieving the current status of the game (e.g. points for each team).
Then a WebSocket is established between each client and the Streaming Component.

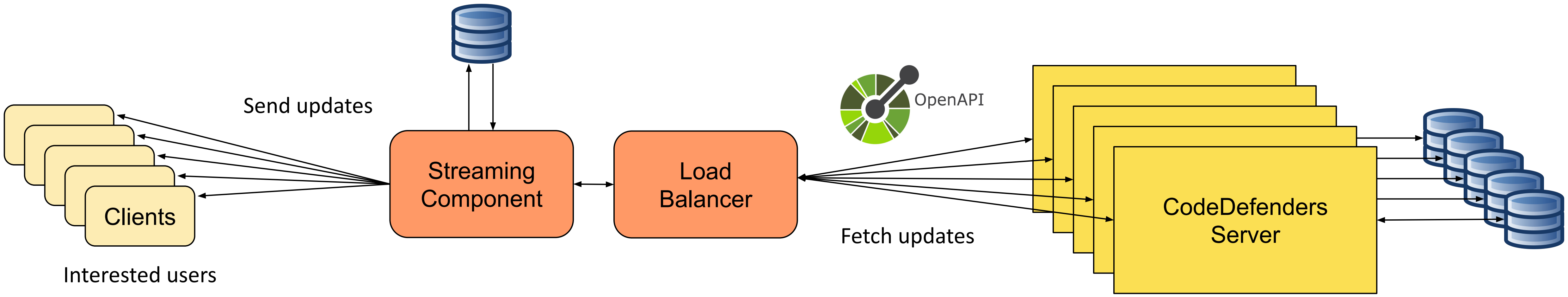
Streaming component (option 2)



Every time an event takes place during the game, CodeDefenders server sends an update to the Streaming Component through the WebSocket.
The Streaming Component receives the events and updates its local database (only for important events such as point scored).
The Streaming Component forwards all the events to the interested clients by means of WebSockets.
The client receives the event and displays visual effects if they are active.

This solution would be efficient (events are notified only once) and would also reuse a paradigm already used in CodeDefenders to handle other types of events (i.e. chat messages).

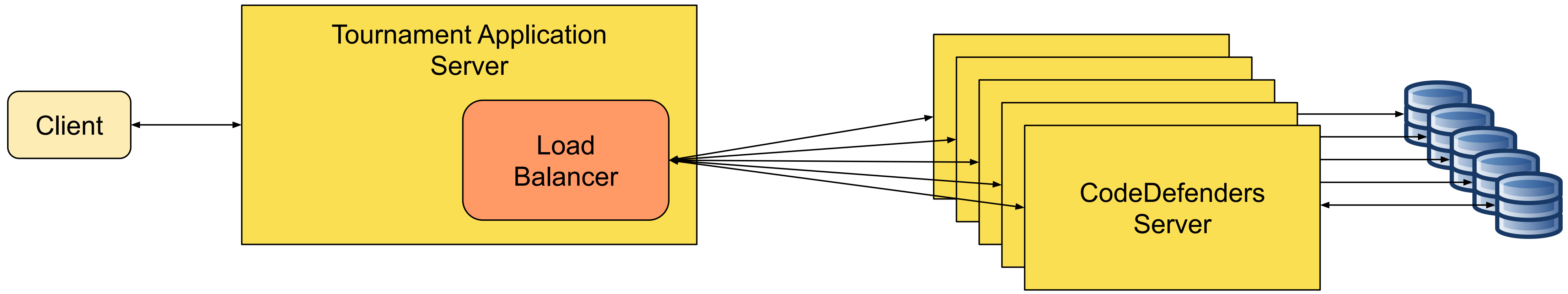
Streaming component



Since the customer explicitly required that also our streaming component must communicate with CodeDefenders by means of OpenAPIS, we had to opt for option 1.
The streaming component periodically (real time updates are not required):

- Polls the CodeDefenders server instance hosting a specific match to retrieve the events happened
- Only for the new events happened notifies the interested users

Load Balancer



We will implement our own load balancer.

When a new game needs to be created the load balancer selects the CodeDefenders server with the minimum number of active games and issues a create game request to it.

The tournament app stores a mapping between active games and CodeDefenders instances in order to be able to redirect players and spectators requests to the correct server.The load balancer component addresses CDF-31.

- The load balancer is completely transparent for clients and applications
  - Clients will send requests and receive responses as if the were talking directly to a CodeDefenders server. They are unaware of the architecture behind the load balancer.

- The load balancer exploits the concept of stickiness
  - The first time a client interacts with the load balancer it gets a cookie that lasts until the end of the session.
  - Every time the same client sends a new request, the load balancer recognizes the cookie and sends the request always to the same CD server.
  - This mechanism is necessary so that the web session for one client is managed only by one CD server.

Exposed CodeDefenders APIs

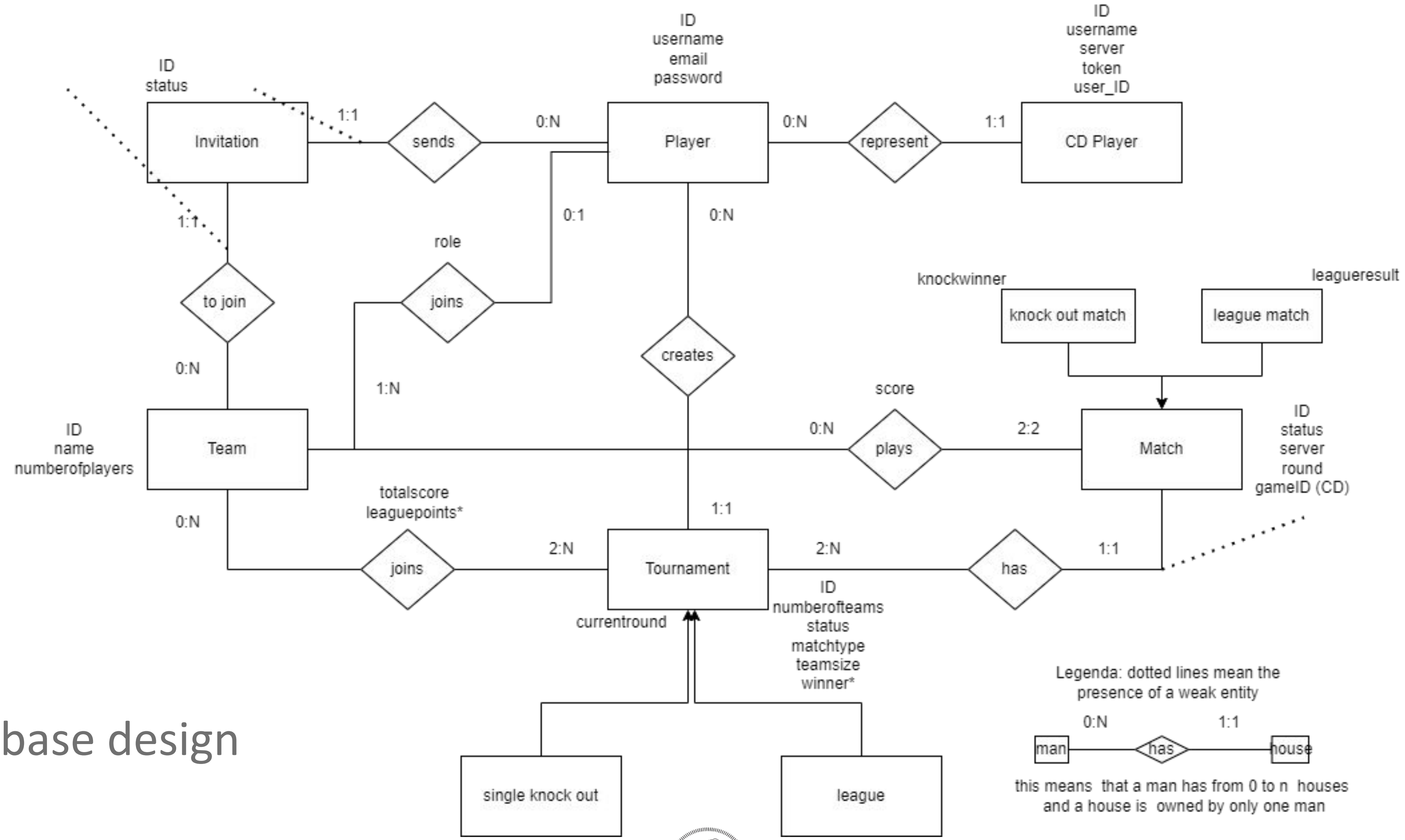
/api/classes	GET	Get the list of uploaded classes
/api/class	GET	Get a class by its ID
/api/history	GET	Get the history of all played games, or filter by class ID and/or user ID
/api/player	GET	Returns a player's username and userId from its playerId
/api/user	GET	Returns a user's informations from its ID
/api/game	GET	Get the status of the game with the specified ID
/admin/api/game	POST	Create a new game with the specified class, settings and teams
/admin/api/game/start	POST	Start a game
/admin/api/game/end	POST	End a game
/api/game/settings	GET	Get the settings of the game with the specified ID
/api/game/test	GET	Get a test by its ID
	POST	Upload a test providing its code and target game

POLITECNICO MILANO 1863

Mälardalen University

Exposed CodeDefenders APIs

/api/game/mutant	GET	Get a mutant by its ID
	POST	Upload a mutant providing its code and target game
/api/game/mutant/equivalences	GET	Get unresolved equivalence claims for the game with the specified ID
/api/game/mutant/equivalence/claim	POST	Claim the mutant with the specified ID as equivalent, meaning that it doesn't affect the behavior of the code
/api/game/mutant/equivalence/resolve	POST	Resolve the pending equivalence claim for a mutant, either by accepting it as equal or uploading the code for a killing test
/admin/api/auth/register	POST	Register a new user with a username, password, email triplet
/admin/api/auth/token	GET	Get an authentication token for the Bot API, tied to a specific user

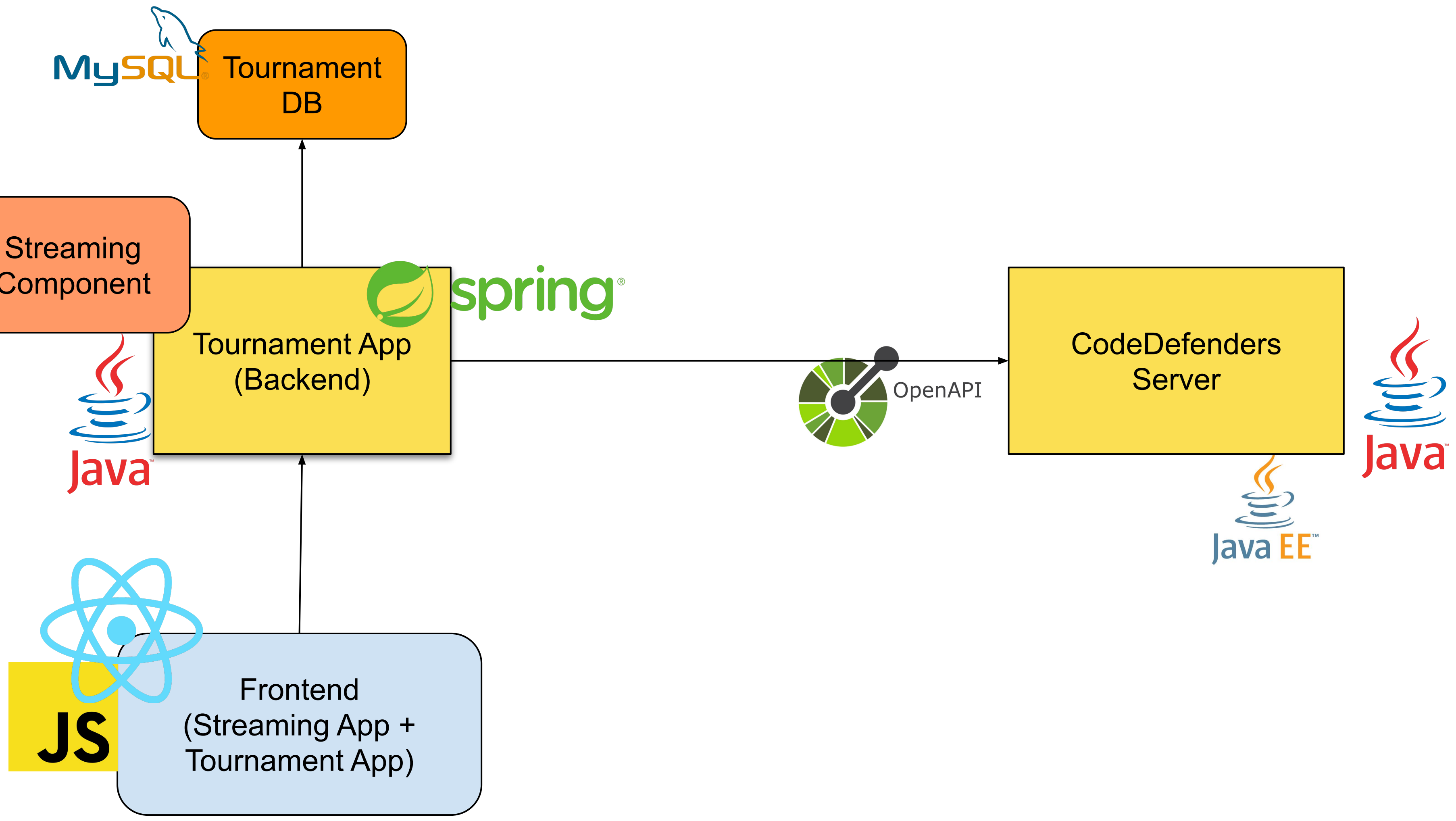


Database design

Database Design

- Players compete in matches through their teams
- Teams can be of variable size
  - this way tournaments with teams of max size of one player are possible, in other words, 1v1 matches are supported
  - tournament logic can but doesn't need to set conditions for team size
- Each match is assigned to exactly one tournament, and has exactly two teams as competitors (attackers vs defenders)
- Database design does not differentiate human players and bots

Technologies we are going to use



Motivations for technologies

Java + Spring:

- Java knowledge quite already spread among the team
- Integrates very well with concepts taken from JavaEE
- More support than JavaEE on forums and so on (also tutorials online)
- Allows to structure a Java Web application in an easy way

MySQL:

- The structure of the data we need will suit perfectly in a relational database.
- Team already familiar with this technology.

React:

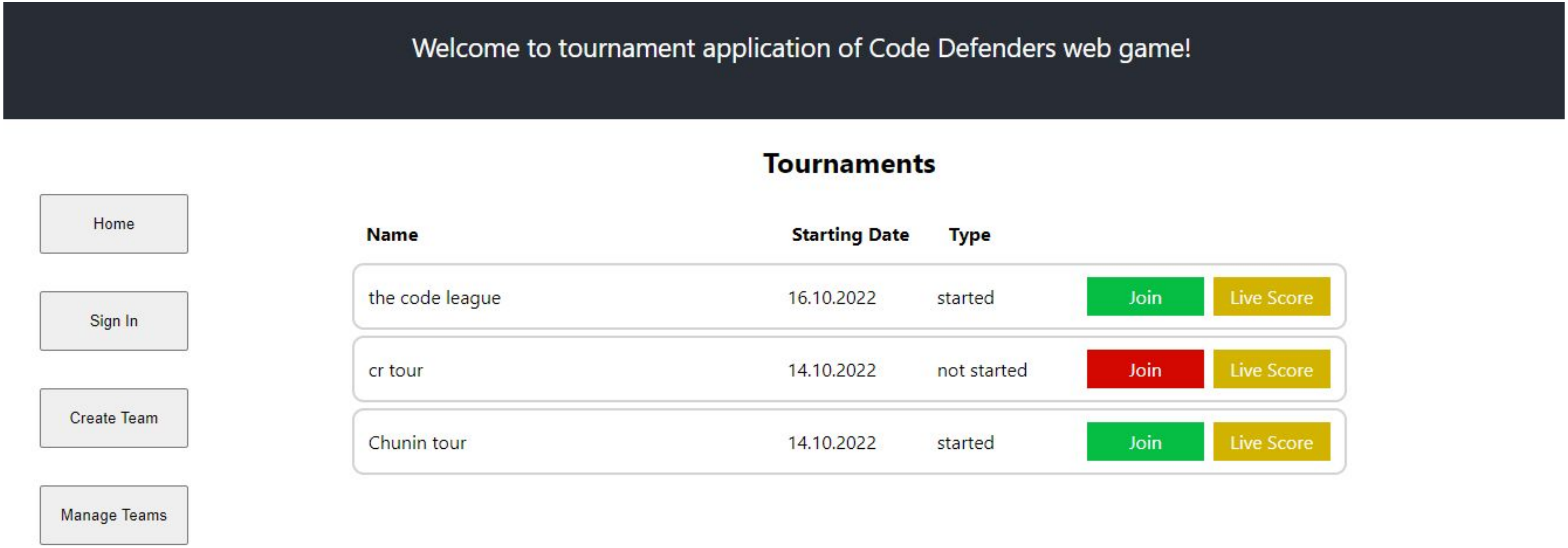
- Free, open-source, and explanatory JavaScript library with simplistic learning curve
- Used for building simple or complex user interfaces, stable front-end framework
- Supports multi-purpose, clean architecture and platform-specific modules

Graphical User Interface

Tournament web app should have user interface for

- Team management
    - CDF-35, CDF-37, CDF-54
- Tournaments overview
    - CDF-34, CDF-36, CDF-41
- Tournaments creation
    - CDF-33, CDF-42
- Watching live score of matches (streaming component)
    - CDF-39, CDF-40

With support for desktop and mobile devices



Welcome to tournament application of Code Defenders web game!

Home

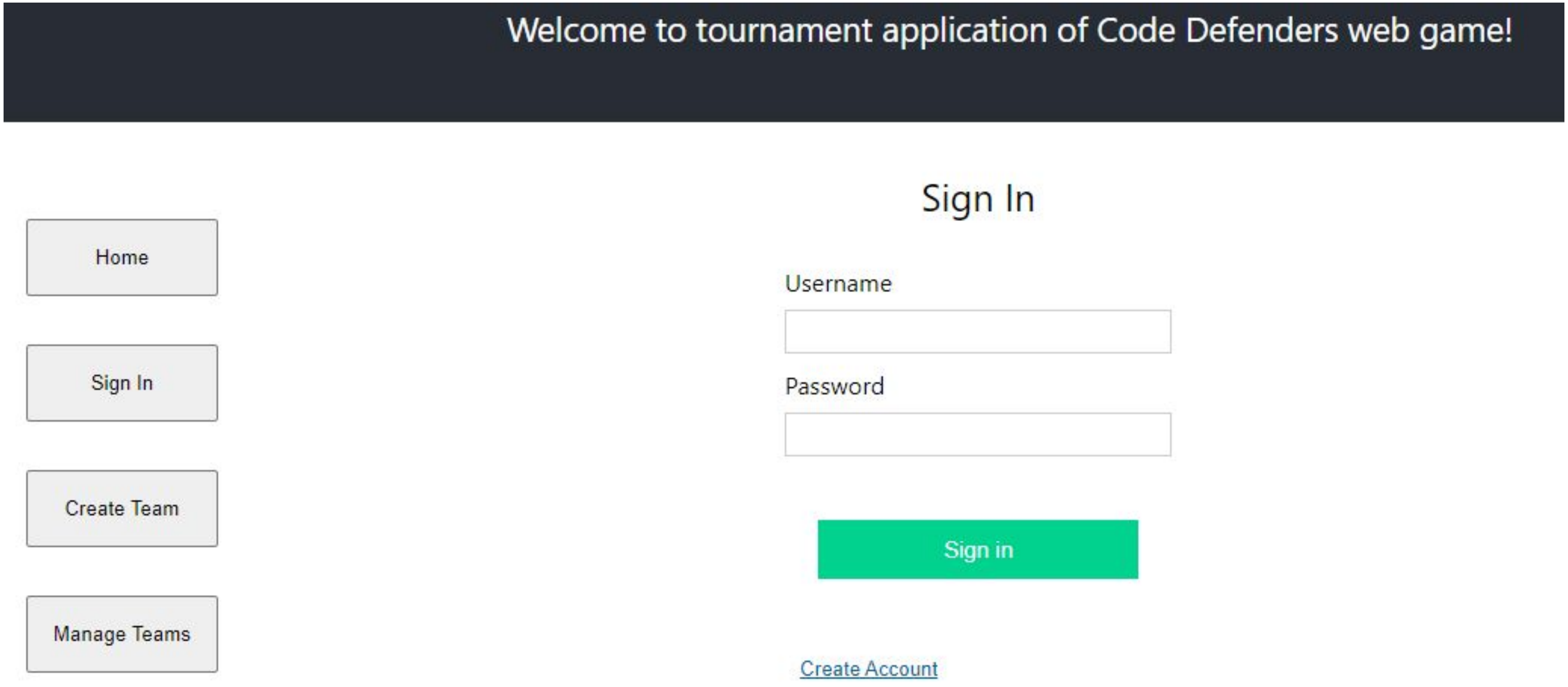
Sign In

Create Team

Manage Teams

Tournaments

Name	Starting Date	Type		
the code league	16.10.2022	started	Join	Live Score
cr tour	14.10.2022	not started	Join	Live Score
Chunin tour	14.10.2022	started	Join	Live Score

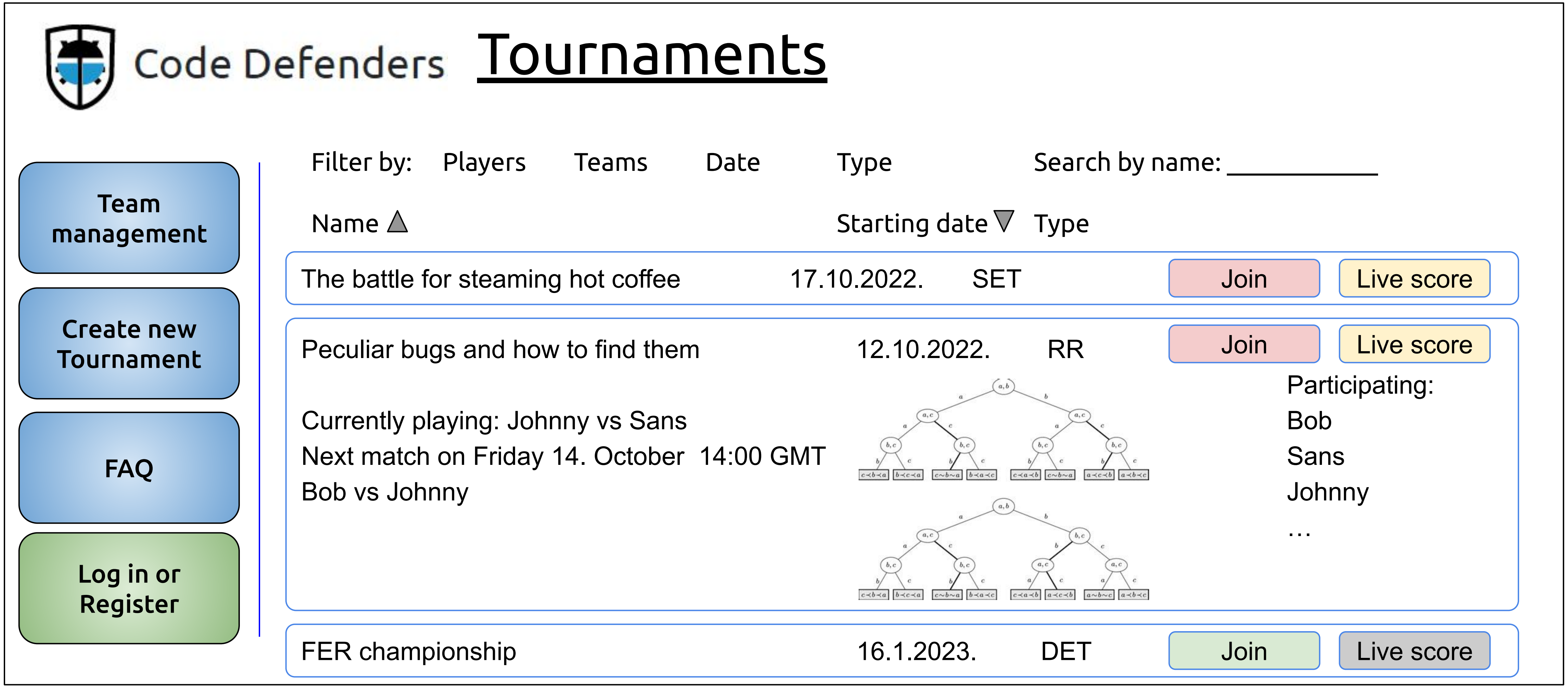

Welcome to tournament application of Code Defenders web game!

Sign In

Username

Password

Home page



Code Defenders **Tournaments**

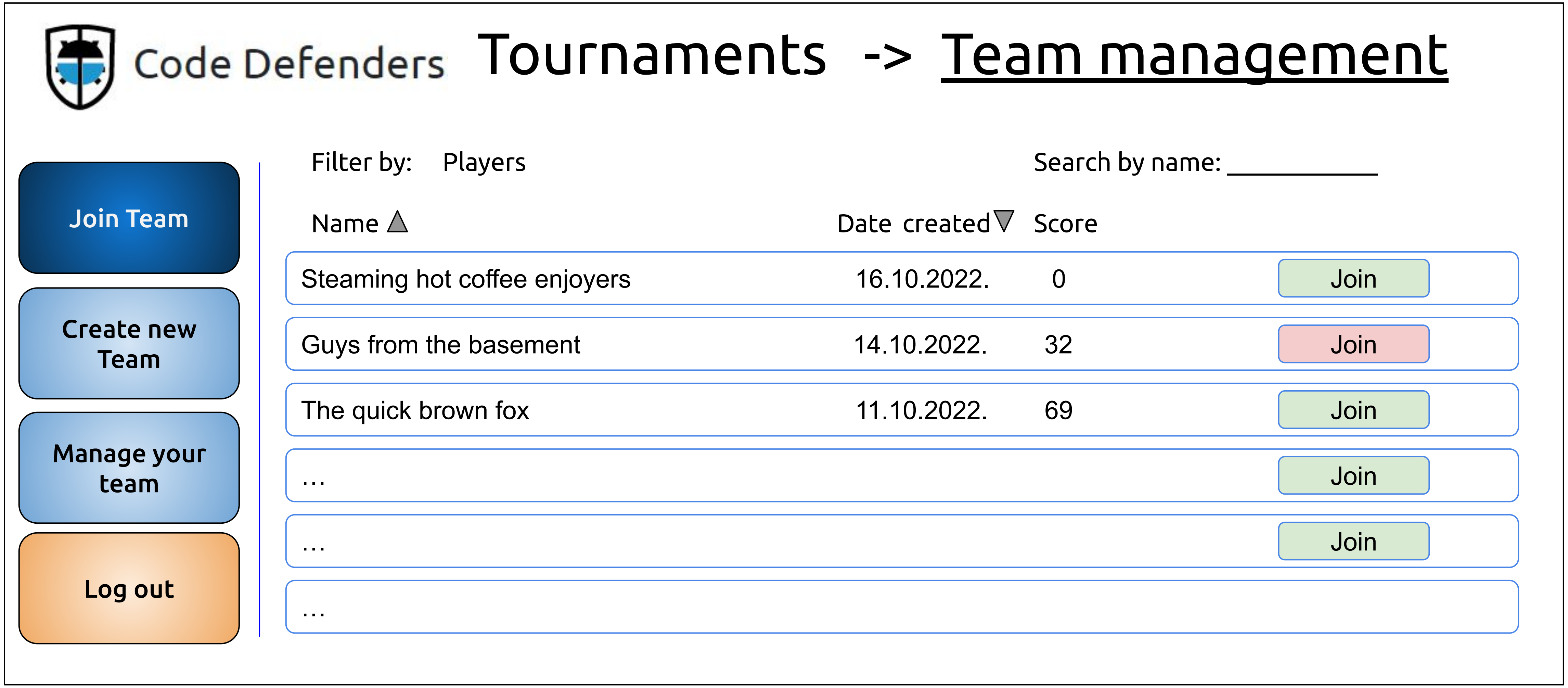
Team management
Create new Tournament
FAQ
Log in or Register

Filter by: Players Teams Date Type Search by name: ___________

Name △	Starting date ▽	Type		
The battle for steaming hot coffee	17.10.2022.	SET	Join	Live score
Peculiar bugs and how to find them	12.10.2022.	RR	Join	Live score
FER championship	16.1.2023.	DET	Join	Live score

Currently playing: Johnny vs Sans
Next match on Friday 14. October 14:00 GMT
Bob vs Johnny

Participating:
Bob
Sans
Johnny
…



Code Defenders **Tournaments -> Team management**

- Join Team
- Create new Team
- Manage your team
- Log out

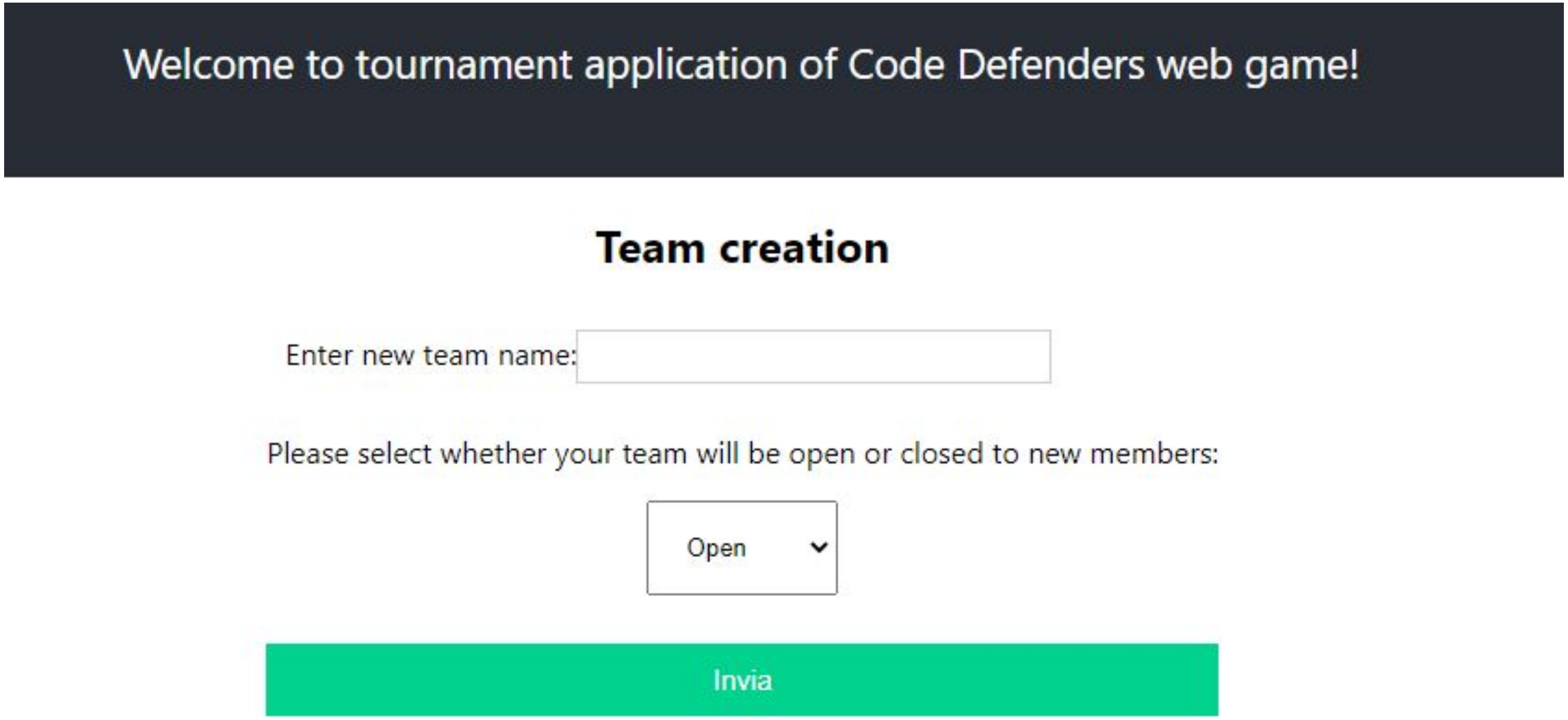
Filter by: Players

Search by name: ___________

Name ▲	Date created ▼	Score	
Steaming hot coffee enjoyers	16.10.2022.	0	Join
Guys from the basement	14.10.2022.	32	Join
The quick brown fox	11.10.2022.	69	Join
…			Join
…			Join
…			

POLITECNICO MILANO 1863

Mälardalen University



Welcome to tournament application of Code Defenders web game!

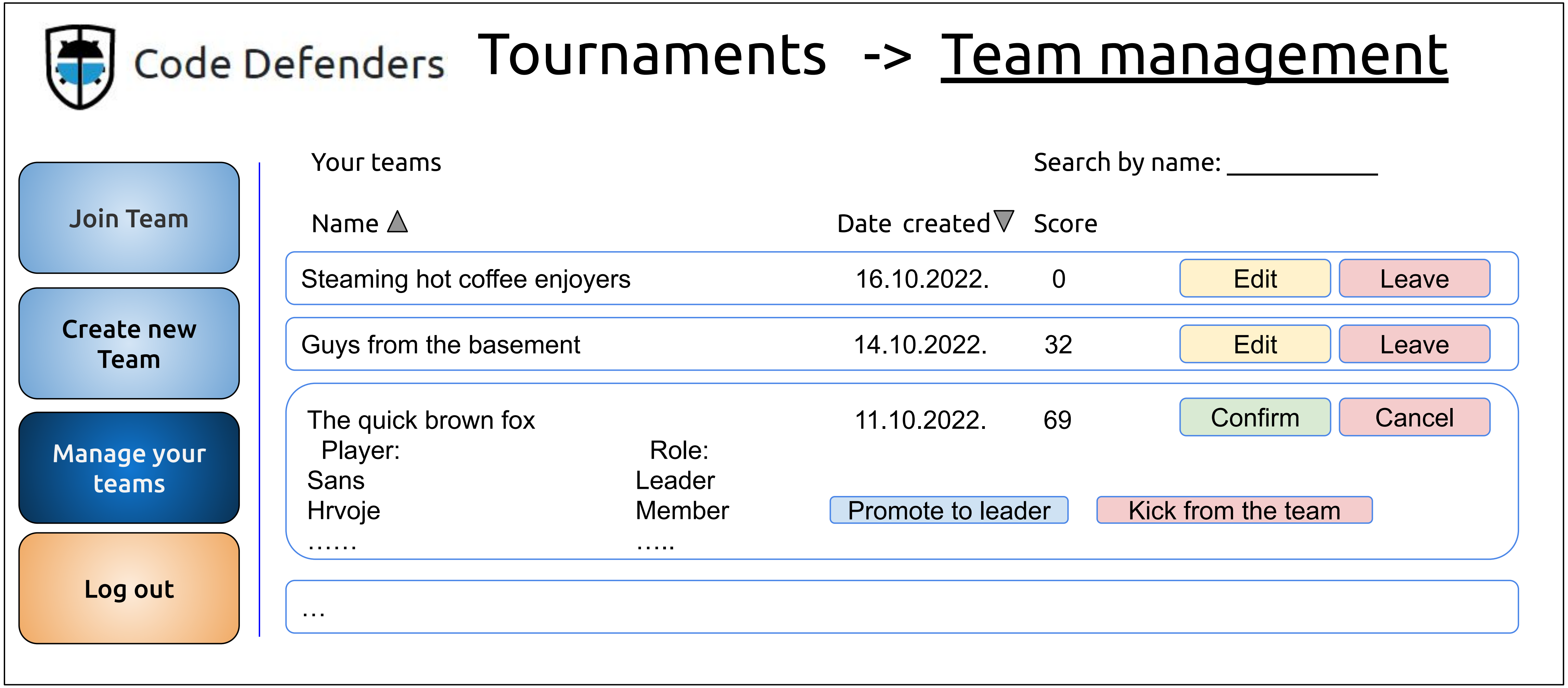
Team creation

Enter new team name:

Please select whether your team will be open or closed to new members:

Open

Invia



Code Defenders Tournaments -> Team management

Join Team

Create new Team

Manage your teams

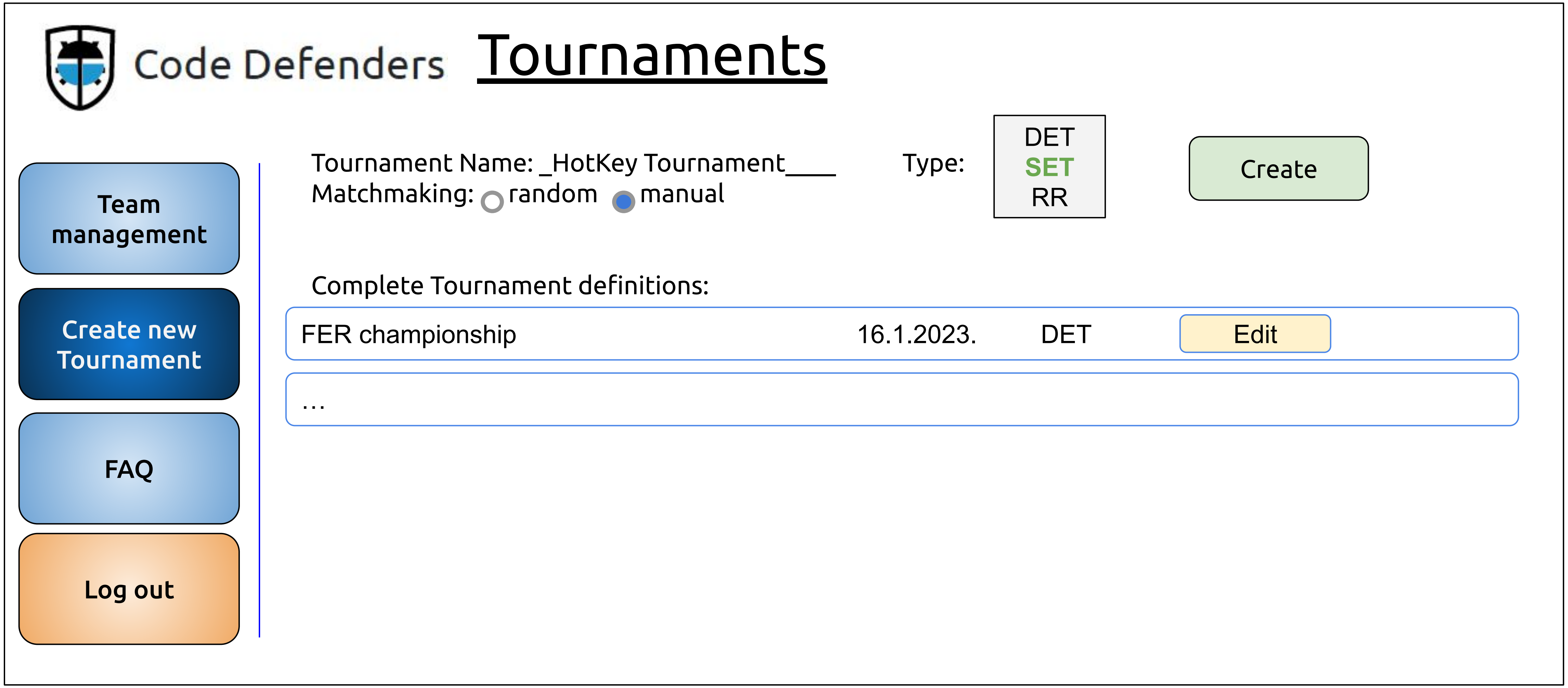
Log out

Your teams

Search by name: ___________

Name ▲	Date created ▼	Score		
Steaming hot coffee enjoyers	16.10.2022.	0	Edit	Leave
Guys from the basement	14.10.2022.	32	Edit	Leave
The quick brown fox	11.10.2022.	69	Confirm	Cancel
…				

Player:	Role:		
Sans	Leader		
Hrvoje	Member	Promote to leader	Kick from the team
……	…..		



Code Defenders **Tournaments**

Team management

Create new Tournament

FAQ

Log out

Tournament Name: _HotKey Tournament____

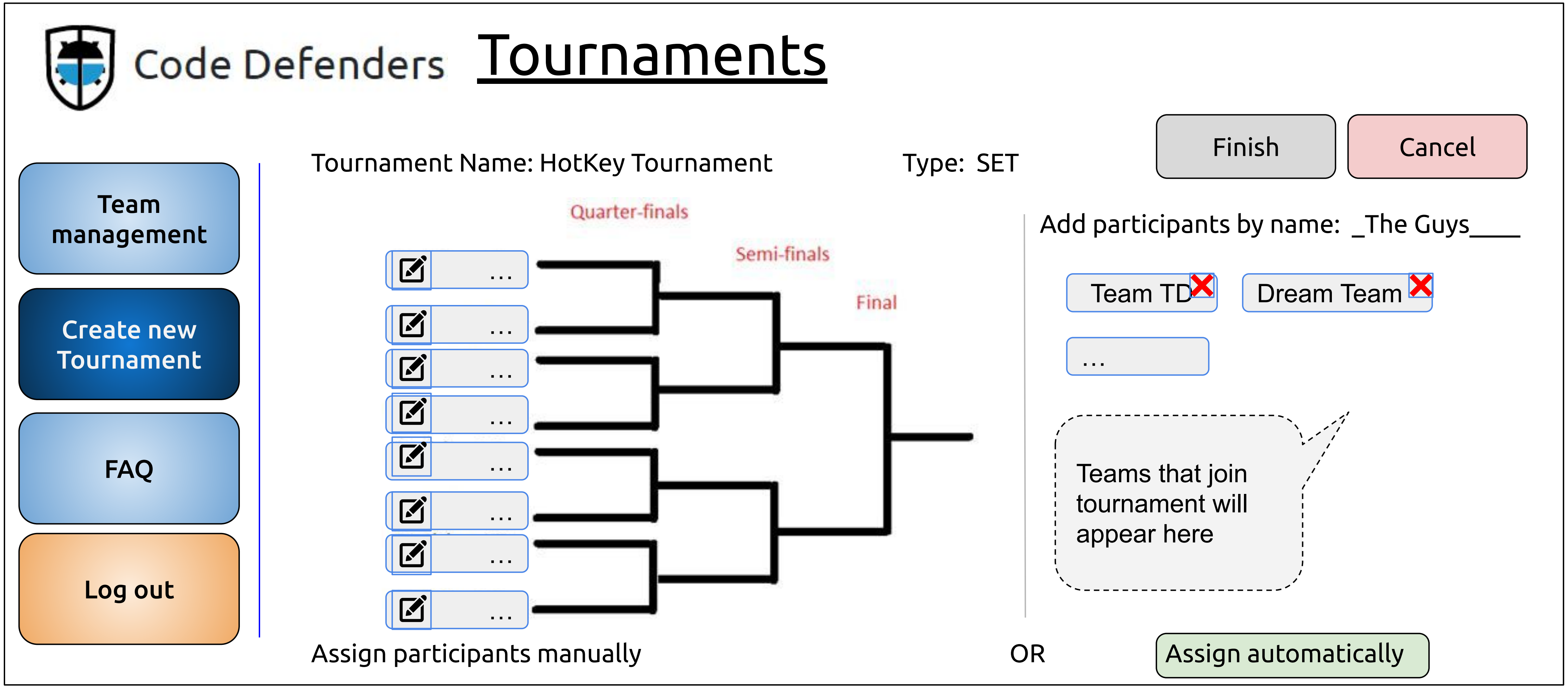
Matchmaking: random [✓ manual]

Type: DET [✓ SET] RR

Create

Complete Tournament definitions:

FER championship	16.1.2023.	DET	Edit
…			



Code Defenders **Tournaments**

Tournament Name: HotKey Tournament

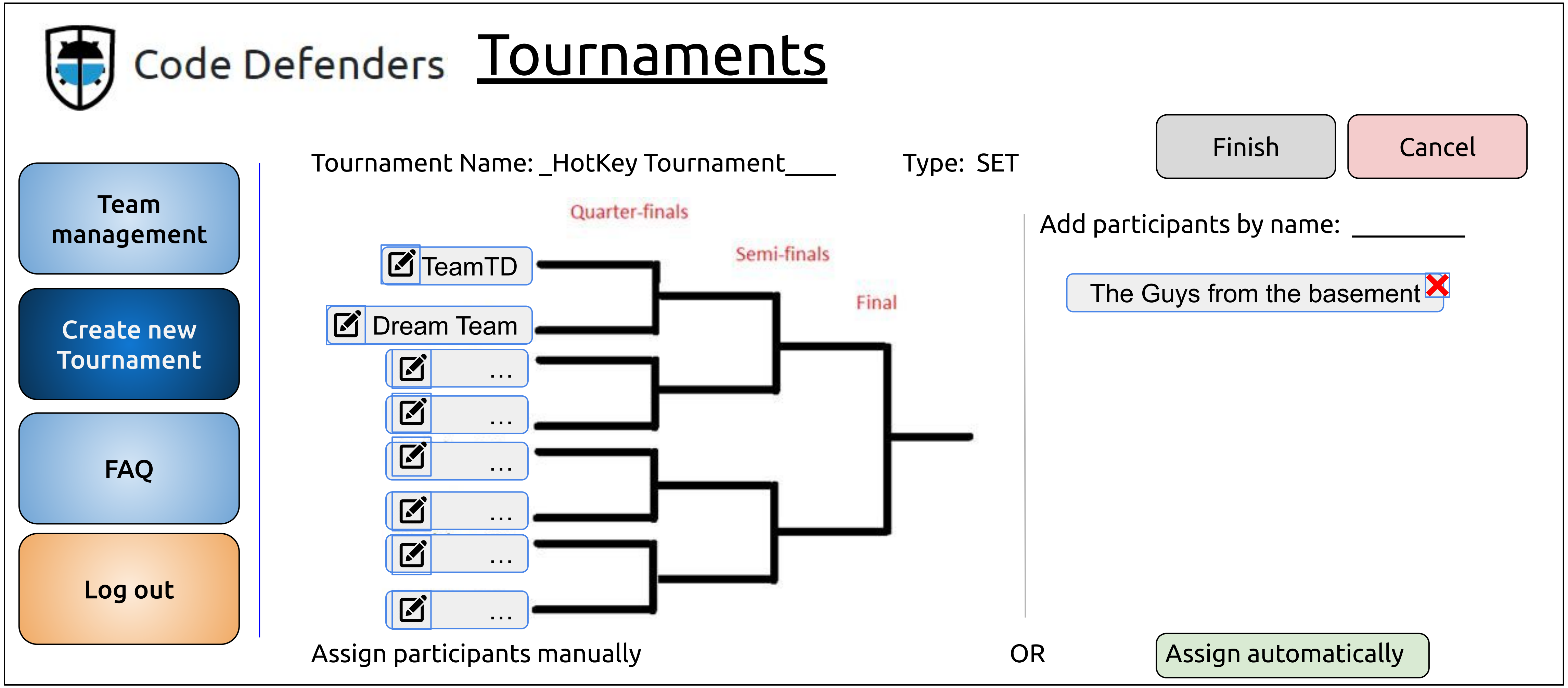
Type: SET

Add participants by name: _The Guys____

Teams that join tournament will appear here

Assign participants manually

OR



Tournament Name: _HotKey Tournament____

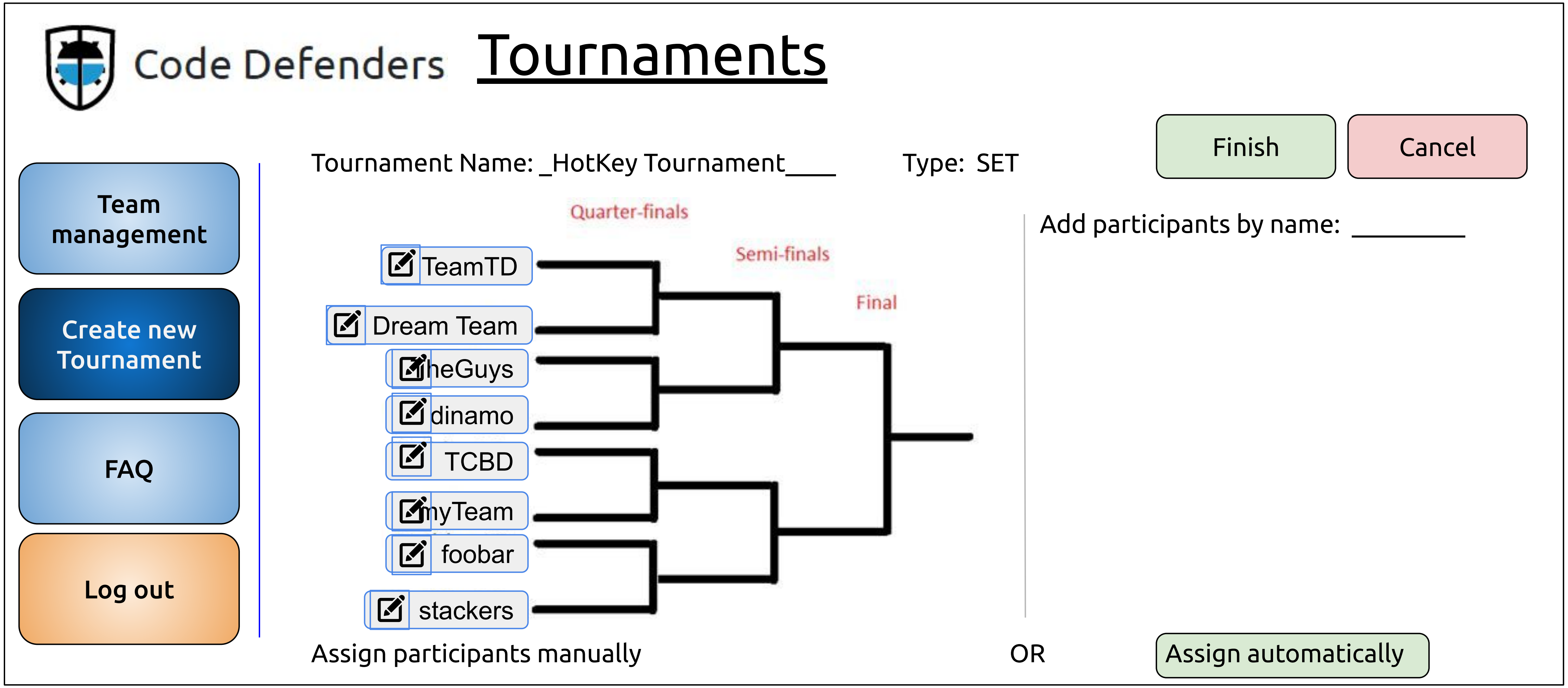
Type: SET

Add participants by name: _________

The Guys from the basement

Assign participants manually

OR



Code Defenders Tournaments

Team management

Create new Tournament

FAQ

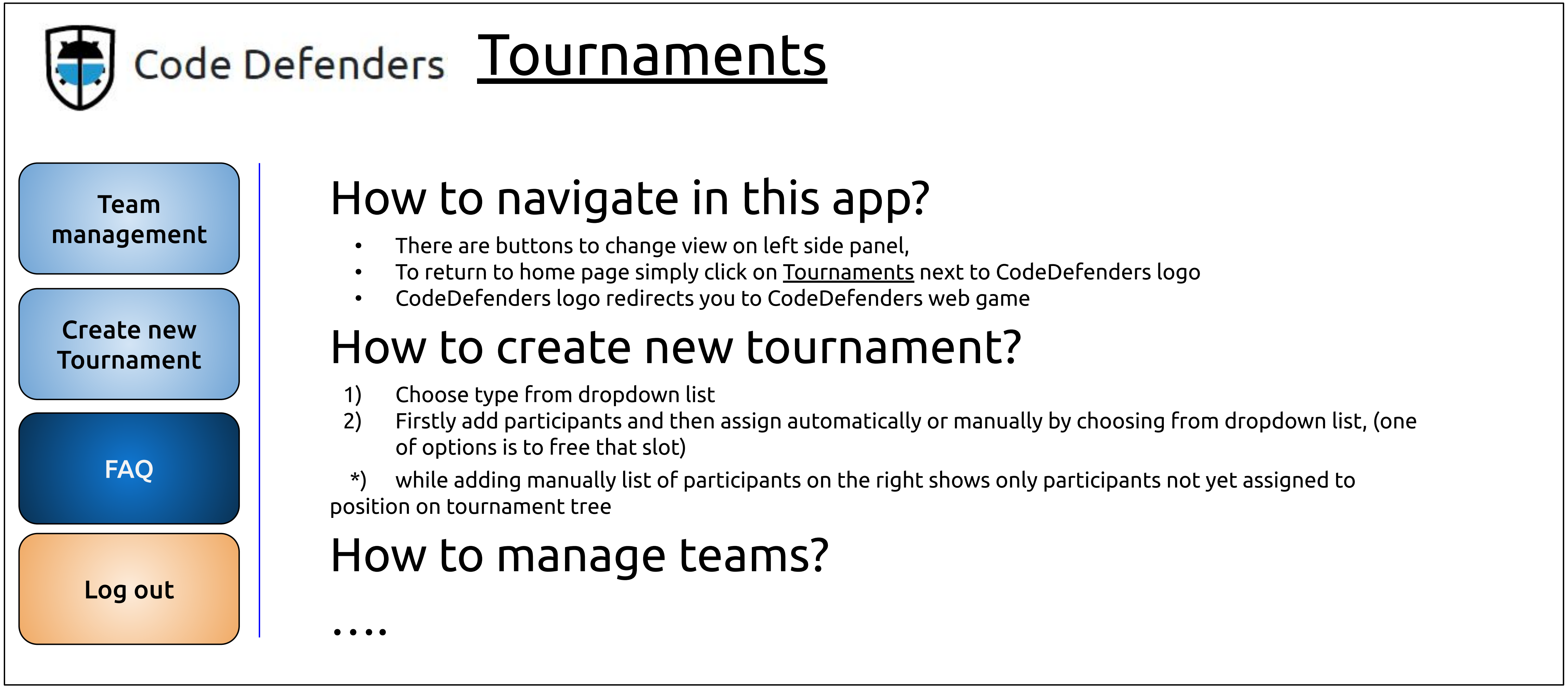
Log out

Tournament Name: _HotKey Tournament____ Type: SET

Finish Cancel

Add participants by name: _________

Assign participants manually OR Assign automatically



Tournaments

How to navigate in this app?

- There are buttons to change view on left side panel,
- To return to home page simply click on Tournaments next to CodeDefenders logo
- CodeDefenders logo redirects you to CodeDefenders web game

How to create new tournament?

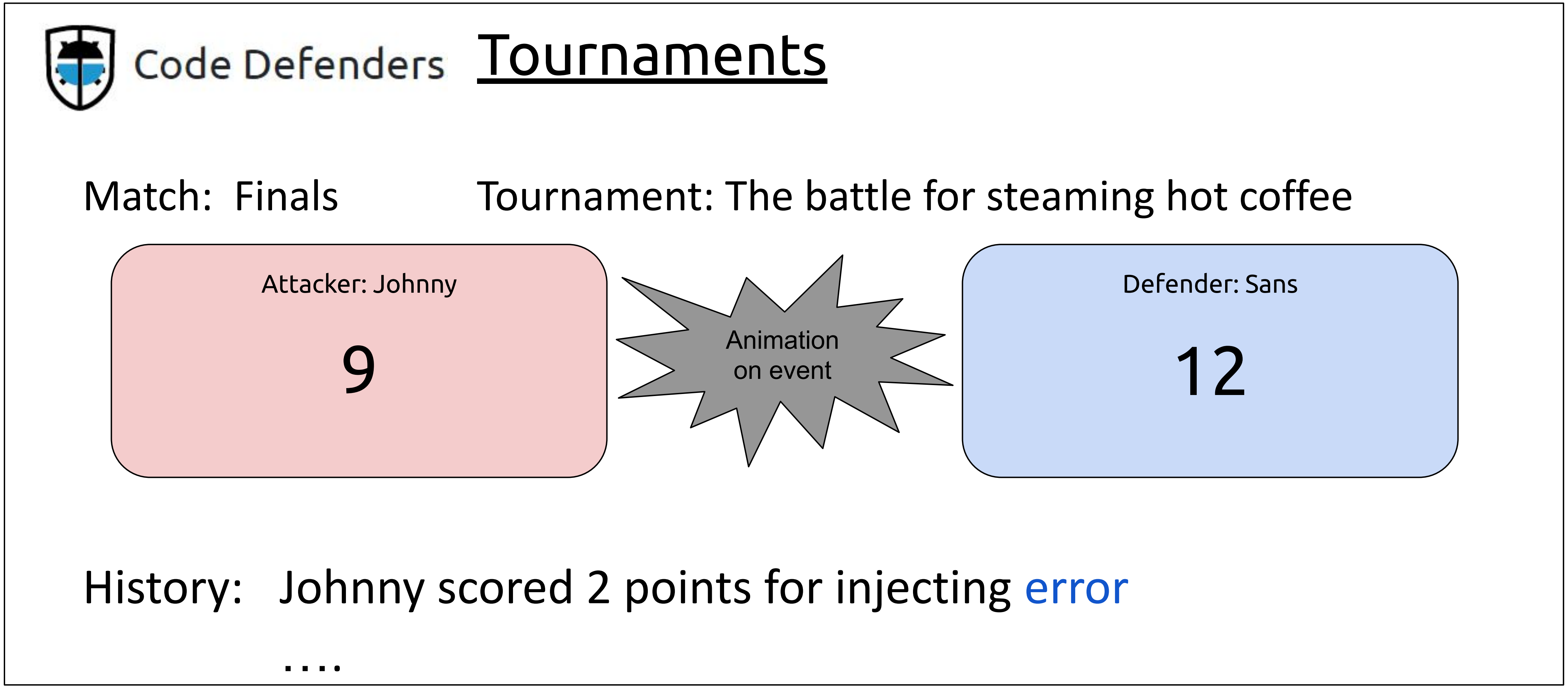
1) Choose type from dropdown list
2) Firstly add participants and then assign automatically or manually by choosing from dropdown list, (one of options is to free that slot)

*) while adding manually list of participants on the right shows only participants not yet assigned to position on tournament tree

How to manage teams?

….

Live score



Code Defenders **Tournaments**

Match: Finals

Tournament: The battle for steaming hot coffee

Attacker: Johnny

9

Animation on event

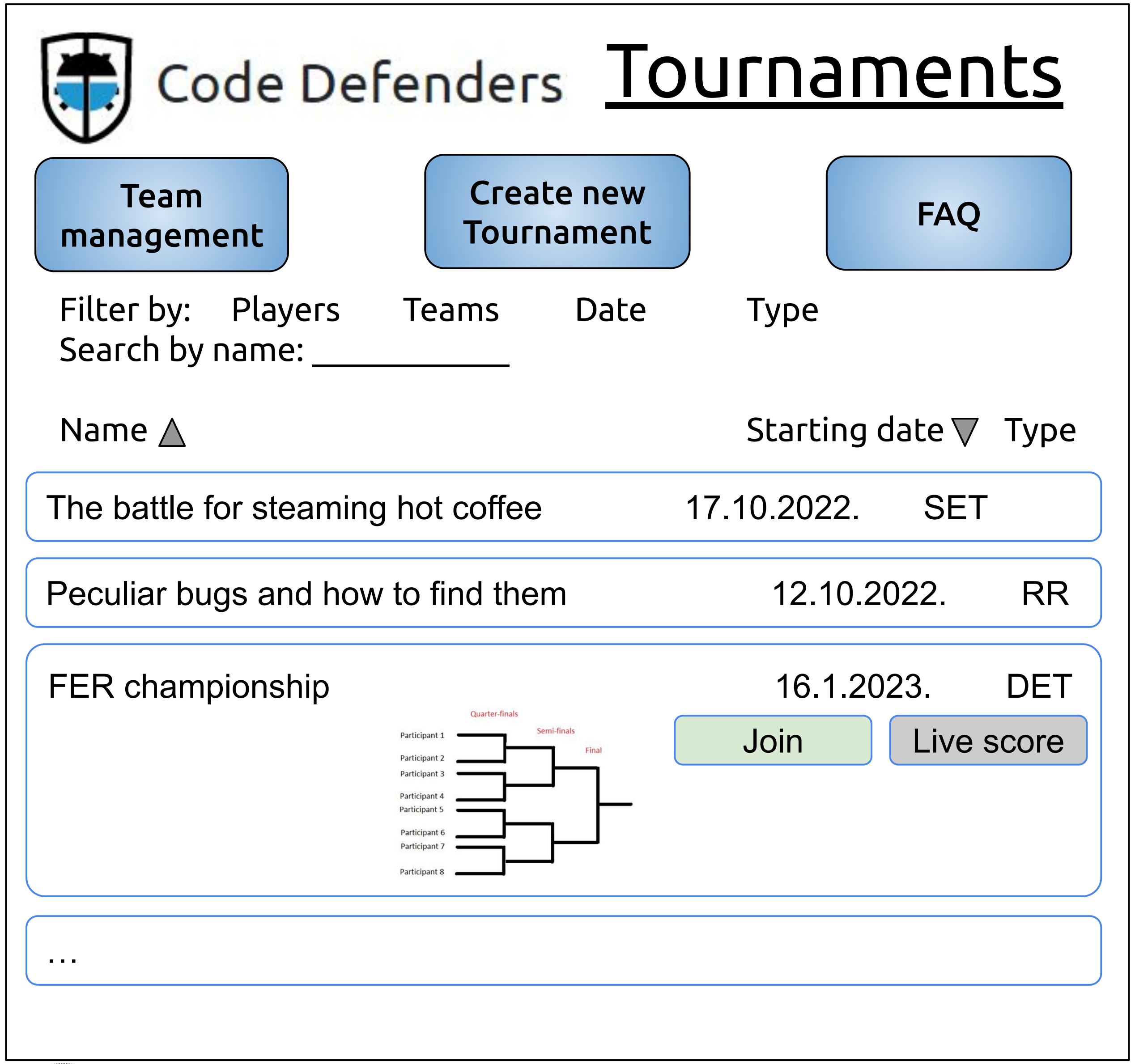
Defender: Sans

12

History: Johnny scored 2 points for injecting error

....

Home page
Mobile phone support



Code Defenders **Tournaments**

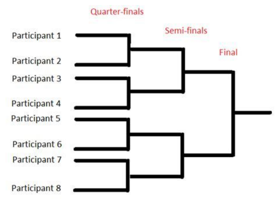
Team management | Create new Tournament | FAQ

Filter by: Players Teams Date Type

Search by name: ___________

Name ▲	Starting date ▼	Type
The battle for steaming hot coffee	17.10.2022.	SET
Peculiar bugs and how to find them	12.10.2022.	RR
FER championship	16.1.2023.	DET
…		

Join | Live score



Detailed backend software design

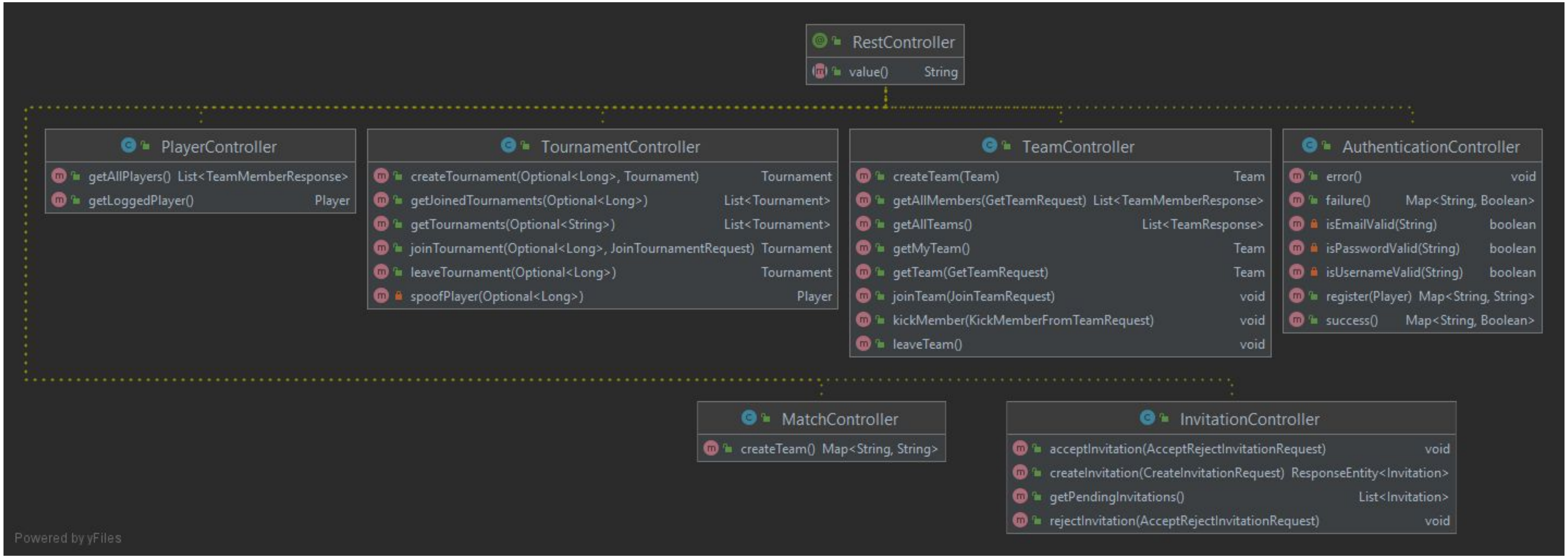
In the following slides we provide class diagrams describing backend components.

Backend has been implementing following Spring Boot conventions.

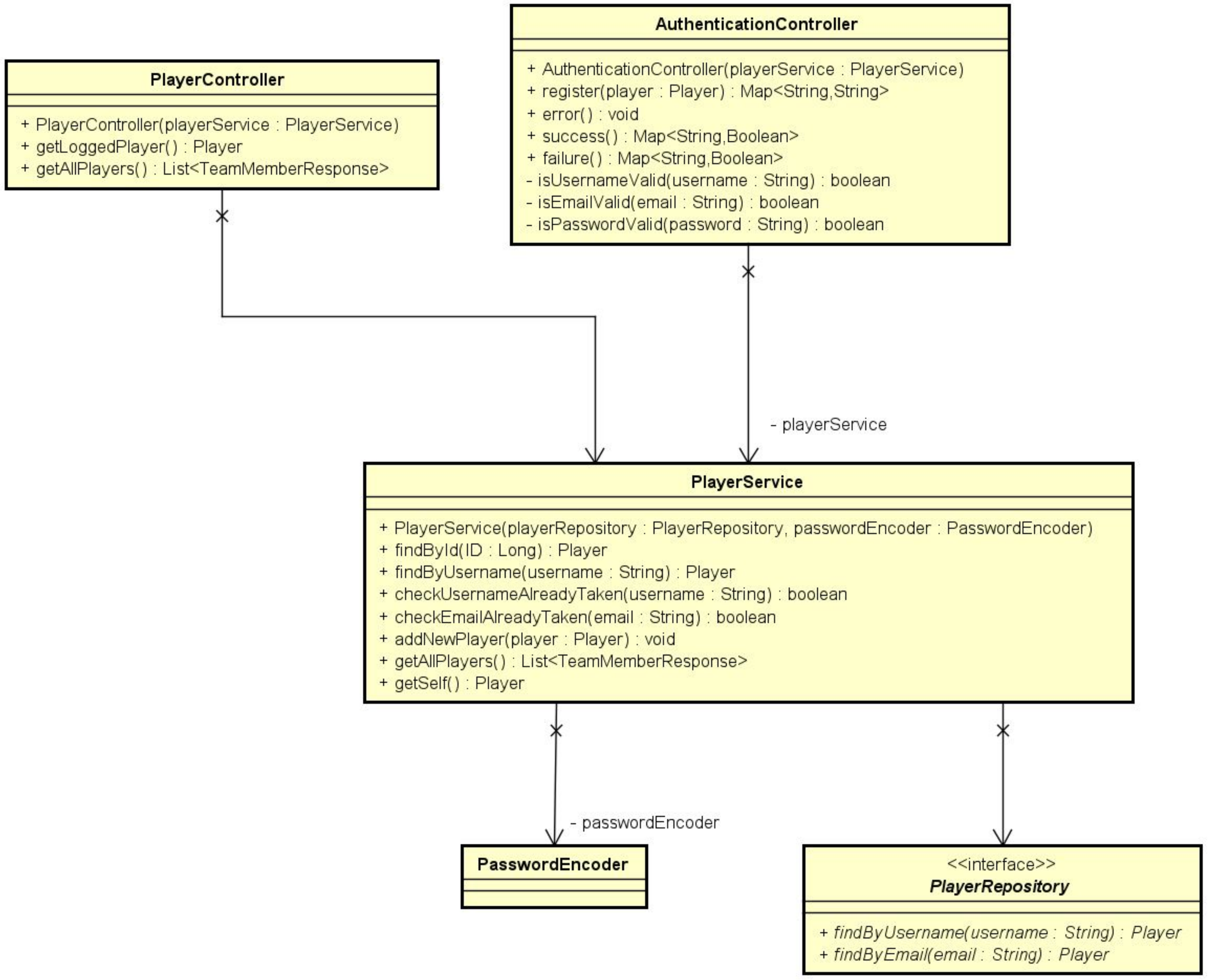
POLITECNICO
MILANO 1863

Mälardalen
University

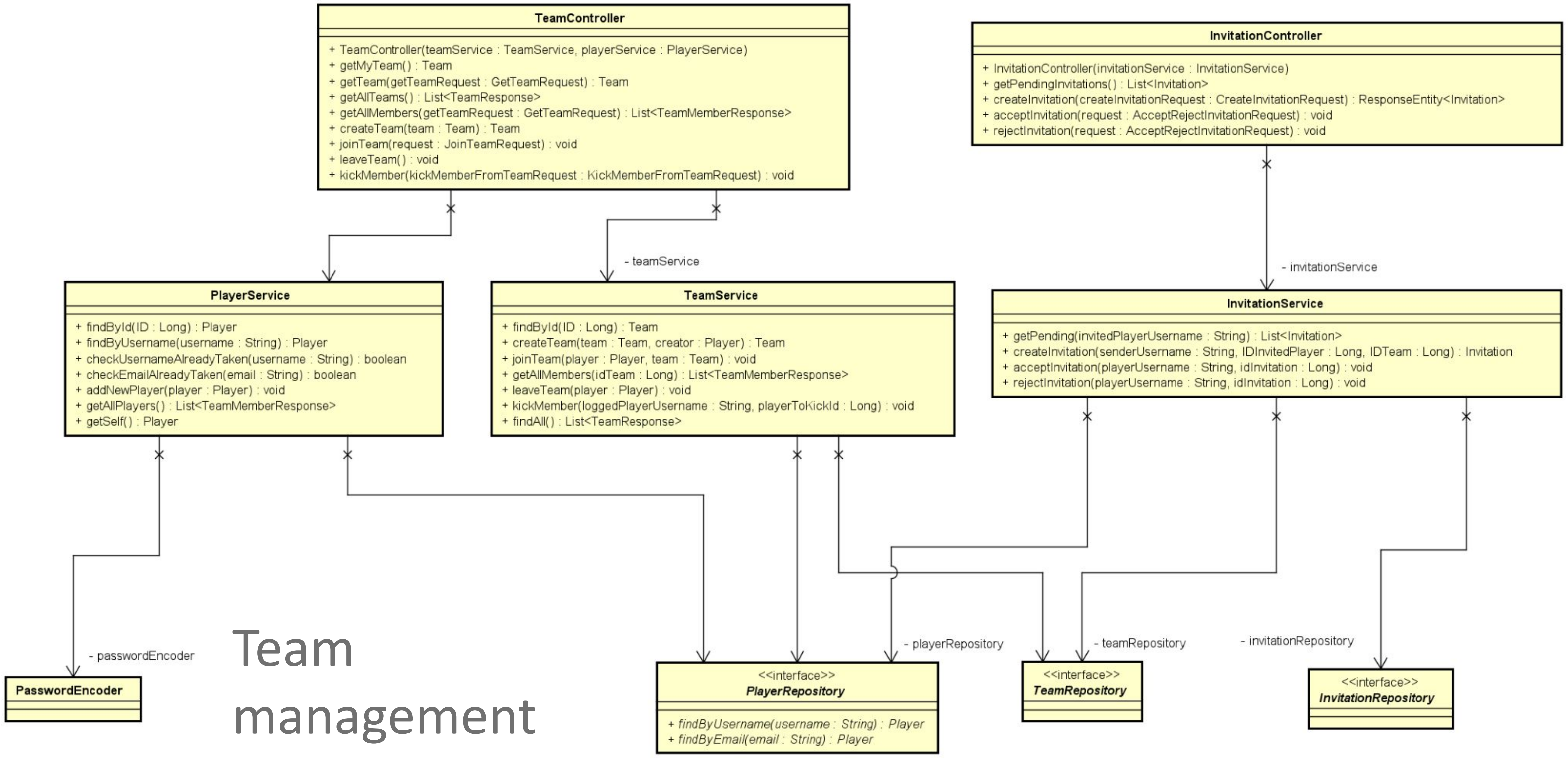
Class Diagrams (Controllers package)


RestController
value() String
PlayerController
getAllPlayers() List<TeamMemberResponse>
getLoggedPlayer() Player
TournamentController
createTournament(Optional<Long>, Tournament) Tournament
getJoinedTournaments(Optional<Long>) List<Tournament>
getTournaments(Optional<String>) List<Tournament>
joinTournament(Optional<Long>, JoinTournamentRequest) Tournament
leaveTournament(Optional<Long>) Tournament
spoofPlayer(Optional<Long>) Player
TeamController
createTeam(Team) Team
getAllMembers(GetTeamRequest) List<TeamMemberResponse>
getAllTeams() List<TeamResponse>
getMyTeam() Team
getTeam(GetTeamRequest) Team
joinTeam(JoinTeamRequest) void
kickMember(KickMemberFromTeamRequest) void
leaveTeam() void
AuthenticationController
error() void
failure() Map<String, Boolean>
isEmailValid(String) boolean
isPasswordValid(String) boolean
isUsernameValid(String) boolean
register(Player) Map<String, String>
success() Map<String, Boolean>
MatchController
createTeam() Map<String, String>
InvitationController
acceptInvitation(AcceptRejectInvitationRequest) void
createInvitation(CreateInvitationRequest) ResponseEntity<Invitation>
getPendingInvitations() List<Invitation>
rejectInvitation(AcceptRejectInvitationRequest) void
Powered by yFiles

Player and authentication class diagram


PlayerController
+ PlayerController(playerService : PlayerService)
+ getLoggedPlayer() : Player
+ getAllPlayers() : List<TeamMemberResponse>
AuthenticationController
+ AuthenticationController(playerService : PlayerService)
+ register(player : Player) : Map<String,String>
+ error() : void
+ success() : Map<String,Boolean>
+ failure() : Map<String,Boolean>
- isUsernameValid(username : String) : boolean
- isEmailValid(email : String) : boolean
- isPasswordValid(password : String) : boolean
- playerService
PlayerService
+ PlayerService(playerRepository : PlayerRepository, passwordEncoder : PasswordEncoder)
+ findById(ID : Long) : Player
+ findByUsername(username : String) : Player
+ checkUsernameAlreadyTaken(username : String) : boolean
+ checkEmailAlreadyTaken(email : String) : boolean
+ addNewPlayer(player : Player) : void
+ getAllPlayers() : List<TeamMemberResponse>
+ getSelf() : Player
- passwordEncoder
PasswordEncoder
<<interface>>
PlayerRepository
+ findByUsername(username : String) : Player
+ findByEmail(email : String) : Player

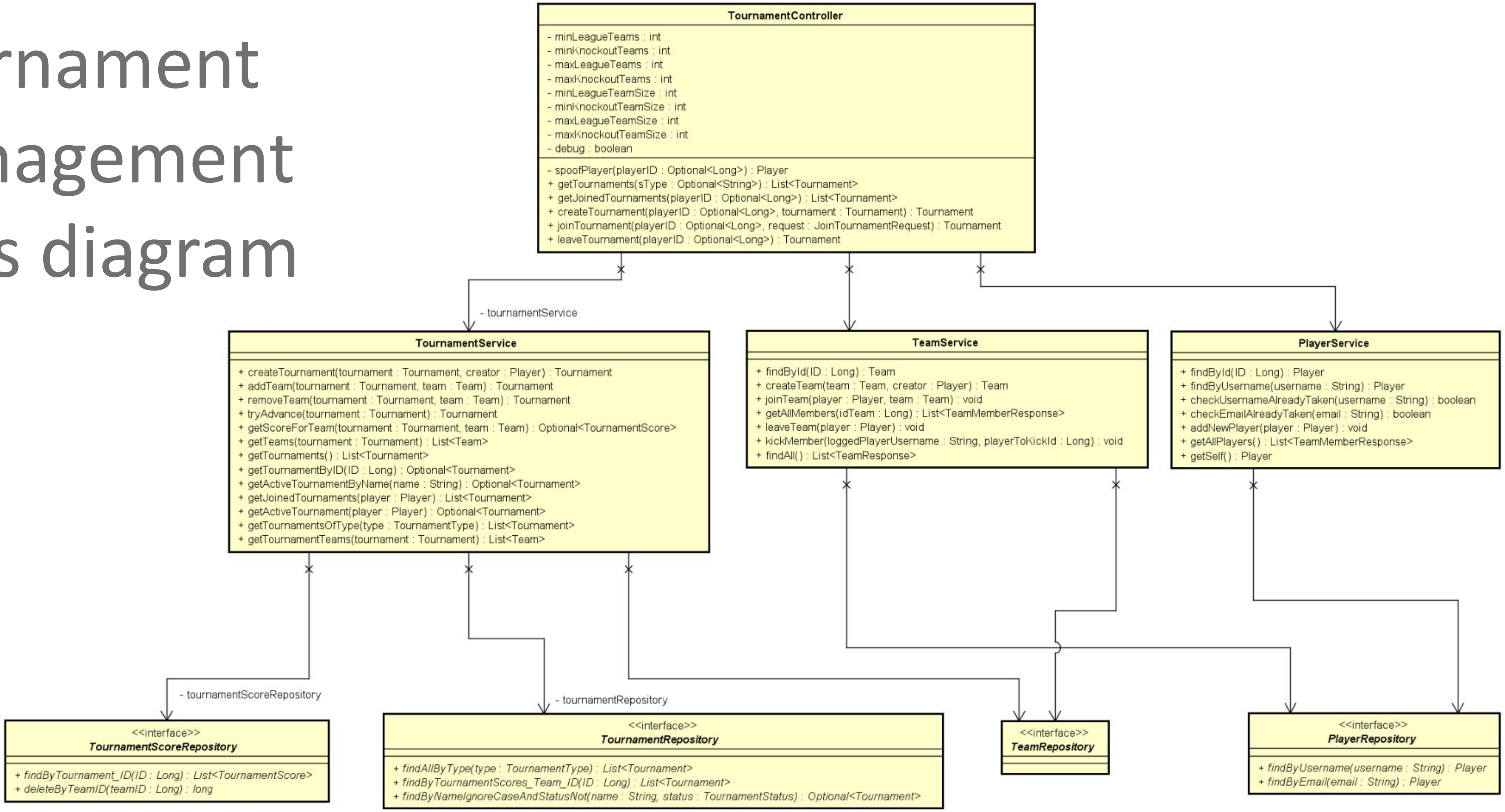
Some details have been omitted for readability


TeamController
+ TeamController(teamService : TeamService, playerService : PlayerService)
+ getMyTeam() : Team
+ getTeam(getTeamRequest : GetTeamRequest) : Team
+ getAllTeams() : List<TeamResponse>
+ getAllMembers(getTeamRequest : GetTeamRequest) : List<TeamMemberResponse>
+ createTeam(team : Team) : Team
+ joinTeam(request : JoinTeamRequest) : void
+ leaveTeam() : void
+ kickMember(kickMemberFromTeamRequest : KickMemberFromTeamRequest) : void
InvitationController
+ InvitationController(invitationService : InvitationService)
+ getPendingInvitations() : List<Invitation>
+ createInvitation(createInvitationRequest : CreateInvitationRequest) : ResponseEntity<Invitation>
+ acceptInvitation(request : AcceptRejectInvitationRequest) : void
+ rejectInvitation(request : AcceptRejectInvitationRequest) : void
- teamService
- invitationService
PlayerService
+ findById(ID : Long) : Player
+ findByUsername(username : String) : Player
+ checkUsernameAlreadyTaken(username : String) : boolean
+ checkEmailAlreadyTaken(email : String) : boolean
+ addNewPlayer(player : Player) : void
+ getAllPlayers() : List<TeamMemberResponse>
+ getSelf() : Player
TeamService
+ findById(ID : Long) : Team
+ createTeam(team : Team, creator : Player) : Team
+ joinTeam(player : Player, team : Team) : void
+ getAllMembers(idTeam : Long) : List<TeamMemberResponse>
+ leaveTeam(player : Player) : void
+ kickMember(loggedPlayerUsername : String, playerToKickId : Long) : void
+ findAll() : List<TeamResponse>
InvitationService
+ getPending(invitedPlayerUsername : String) : List<Invitation>
+ createInvitation(senderUsername : String, IDInvitedPlayer : Long, IDTeam : Long) : Invitation
+ acceptInvitation(playerUsername : String, idInvitation : Long) : void
+ rejectInvitation(playerUsername : String, idInvitation : Long) : void
- passwordEncoder
- playerRepository
- teamRepository
- invitationRepository
PasswordEncoder
<<interface>>
PlayerRepository
+ findByUsername(username : String) : Player
+ findByEmail(email : String) : Player
<<interface>>
TeamRepository
<<interface>>
InvitationRepository

Team management class diagram

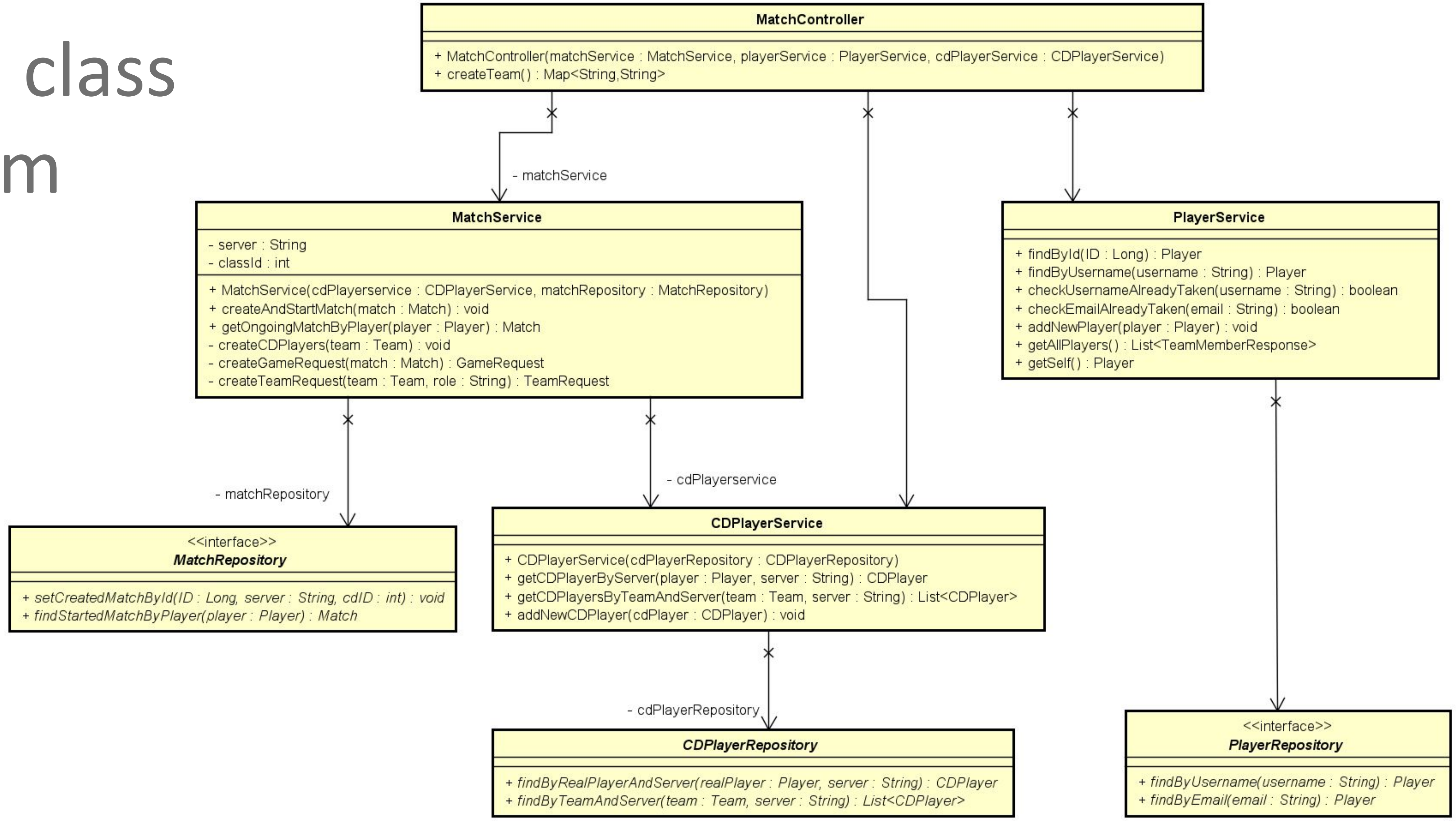
Some details have been omitted for readability

Tournament management class diagram


TournamentController
- minLeagueTeams : int
- minKnockoutTeams : int
- maxLeagueTeams : int
- maxKnockoutTeams : int
- minLeagueTeamSize : int
- minKnockoutTeamSize : int
- maxLeagueTeamSize : int
- maxKnockoutTeamSize : int
- debug : boolean
- spoofPlayer(playerID : Optional<Long>) : Player
+ getTournaments(sType : Optional<String>) : List<Tournament>
+ getJoinedTournaments(playerID : Optional<Long>) : List<Tournament>
+ createTournament(playerID : Optional<Long>, tournament : Tournament) : Tournament
+ joinTournament(playerID : Optional<Long>, request : JoinTournamentRequest) : Tournament
+ leaveTournament(playerID : Optional<Long>) : Tournament
- tournamentService
TournamentService
+ createTournament(tournament : Tournament, creator : Player) : Tournament
+ addTeam(tournament : Tournament, team : Team) : Tournament
+ removeTeam(tournament : Tournament, team : Team) : Tournament
+ tryAdvance(tournament : Tournament) : Tournament
+ getScoreForTeam(tournament : Tournament, team : Team) : Optional<TournamentScore>
+ getTeams(tournament : Tournament) : List<Team>
+ getTournaments() : List<Tournament>
+ getTournamentByID(ID : Long) : Optional<Tournament>
+ getActiveTournamentByName(name : String) : Optional<Tournament>
+ getJoinedTournaments(player : Player) : List<Tournament>
+ getActiveTournament(player : Player) : Optional<Tournament>
+ getTournamentsOfType(type : TournamentType) : List<Tournament>
+ getTournamentTeams(tournament : Tournament) : List<Team>
TeamService
+ findById(ID : Long) : Team
+ createTeam(team : Team, creator : Player) : Team
+ joinTeam(player : Player, team : Team) : void
+ getAllMembers(idTeam : Long) : List<TeamMemberResponse>
+ leaveTeam(player : Player) : void
+ kickMember(loggedPlayerUsername : String, playerToKickId : Long) : void
+ findAll() : List<TeamResponse>
PlayerService
+ findById(ID : Long) : Player
+ findByUsername(username : String) : Player
+ checkUsernameAlreadyTaken(username : String) : boolean
+ checkEmailAlreadyTaken(email : String) : boolean
+ addNewPlayer(player : Player) : void
+ getAllPlayers() : List<TeamMemberResponse>
+ getSelf() : Player
- tournamentScoreRepository
- tournamentRepository
<<interface>>
TournamentScoreRepository
+ findByTournament_ID(ID : Long) : List<TournamentScore>
+ deleteByTeamID(teamID : Long) : long
<<interface>>
TournamentRepository
+ findAllByType(type : TournamentType) : List<Tournament>
+ findByTournamentScores_Team_ID(ID : Long) : List<Tournament>
+ findByNameIgnoreCaseAndStatusNot(name : String, status : TournamentStatus) : Optional<Tournament>
<<interface>>
TeamRepository
<<interface>>
PlayerRepository
+ findByUsername(username : String) : Player
+ findByEmail(email : String) : Player

Some details have been omitted for readability

Match class diagram


MatchController
+ MatchController(matchService : MatchService, playerService : PlayerService, cdPlayerService : CDPlayerService)
+ createTeam() : Map<String,String>
- matchService
MatchService
- server : String
- classId : int
+ MatchService(cdPlayerservice : CDPlayerService, matchRepository : MatchRepository)
+ createAndStartMatch(match : Match) : void
+ getOngoingMatchByPlayer(player : Player) : Match
- createCDPlayers(team : Team) : void
- createGameRequest(match : Match) : GameRequest
- createTeamRequest(team : Team, role : String) : TeamRequest
PlayerService
+ findById(ID : Long) : Player
+ findByUsername(username : String) : Player
+ checkUsernameAlreadyTaken(username : String) : boolean
+ checkEmailAlreadyTaken(email : String) : boolean
+ addNewPlayer(player : Player) : void
+ getAllPlayers() : List<TeamMemberResponse>
+ getSelf() : Player
- matchRepository
- cdPlayerservice
<<interface>>
MatchRepository
+ setCreatedMatchById(ID : Long, server : String, cdID : int) : void
+ findStartedMatchByPlayer(player : Player) : Match
CDPlayerService
+ CDPlayerService(cdPlayerRepository : CDPlayerRepository)
+ getCDPlayerByServer(player : Player, server : String) : CDPlayer
+ getCDPlayersByTeamAndServer(team : Team, server : String) : List<CDPlayer>
+ addNewCDPlayer(cdPlayer : CDPlayer) : void
- cdPlayerRepository
CDPlayerRepository
+ findByRealPlayerAndServer(realPlayer : Player, server : String) : CDPlayer
+ findByTeamAndServer(team : Team, server : String) : List<CDPlayer>
<<interface>>
PlayerRepository
+ findByUsername(username : String) : Player
+ findByEmail(email : String) : Player

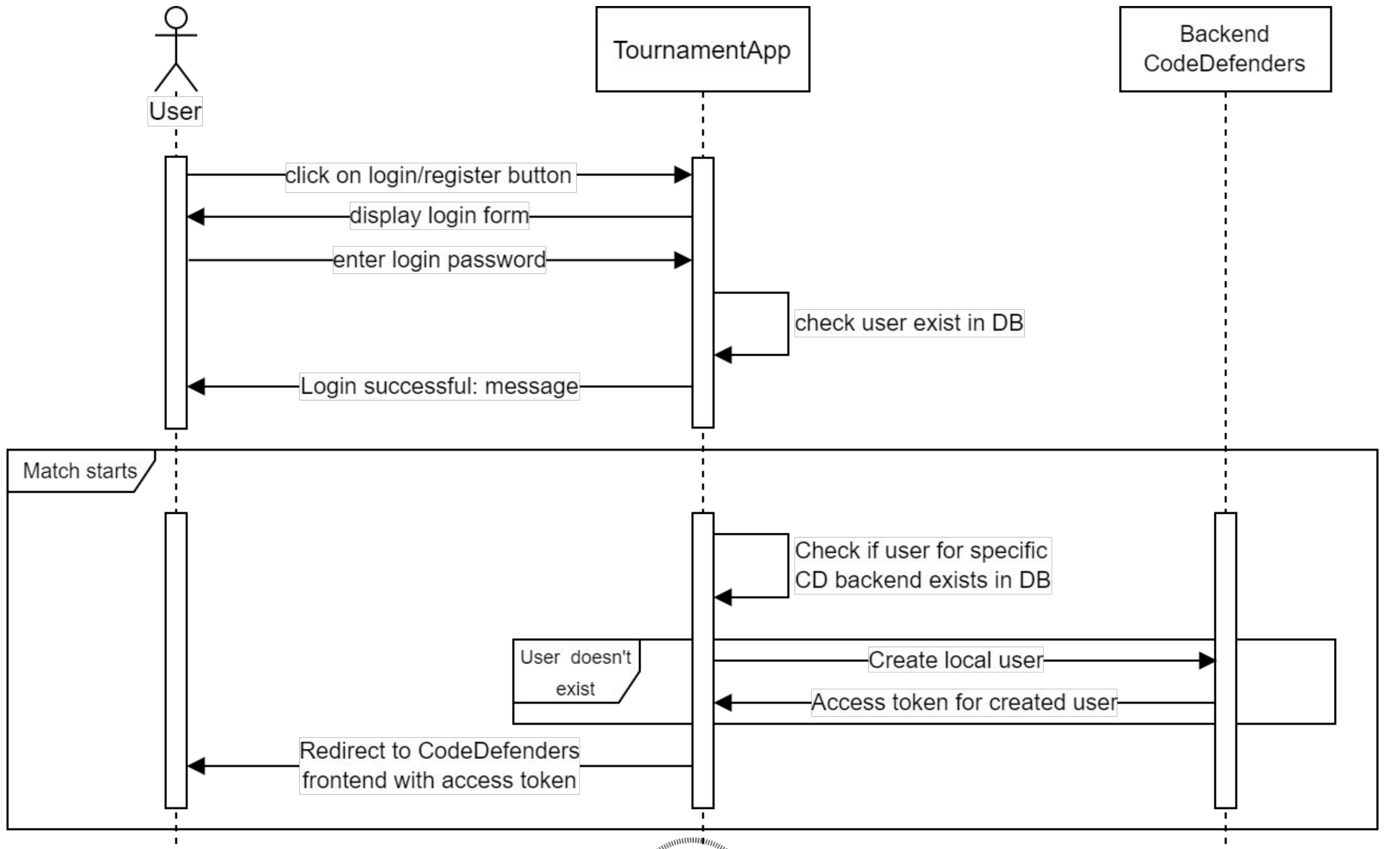
Some details have been omitted for readability

Detailed frontend software design

A detailed description of each component design for frontend will be added later on and updated at the beginning of each sprint to match its current implementation.

Authentication sequence

The users register and login on our TournamentApp, with no interaction with the CodeDefenders backend.
When a match starts, we check if our user has a corresponding user on the CodeDefenders instance hosting the game. If it doesn't, we create it and get the access token; otherwise it already exists and we already have the token.
We redirect the user to the target game page on CodeDefenders specifying the authentication token, so that the CodeDefenders login page is skipped


User
TournamentApp
Backend CodeDefenders
click on login/register button
display login form
enter login password
check user exist in DB
Login successful: message
Match starts
Check if user for specific CD backend exists in DB
User doesn't exist
Create local user
Access token for created user
Redirect to CodeDefenders frontend with access token

Codebenders

contact info:

fanny.delnondedieu@fer.hr
dominik.brdar@fer.hr

hrvoje.rom@fer.hr

simone.mezzaro@mail.polimi.it

fabio.patella@mail.polimi.it

andrea2.restelli@mail.polimi.it