Final Presentation

Project Code Defenders - Robo Tournament
Team CodeBenders

Agenda

- Our Team
- Project Vision
- Requirements
- Design Description
  - Components
  - Technologies
  - Frontend
  - Backend
- Functionalities of final product
- Scrum implementation
- Work done on project
  - Cumulative Flow Diagram, Sprint burndown charts, comparison diagrams
  - Product backlog burndown
  - Experiences from the distributed software development project
  - Testing Strategy carried and Validation of final product
- Demo
- QnA

Our Team

Testers

Frontend

Product Owner

Fanny Delnondedieu

Scrum Master

Dominik Brdar

Hrvoje Rom

Fabio Patella

Backend

Simone Mezzaro

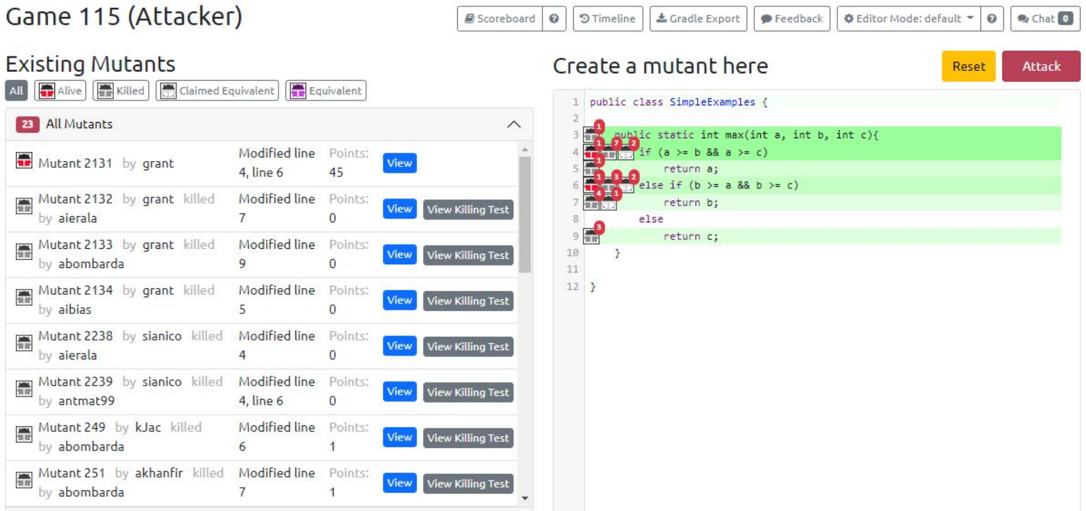
Riccardo Nava

Andrea Restelli

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

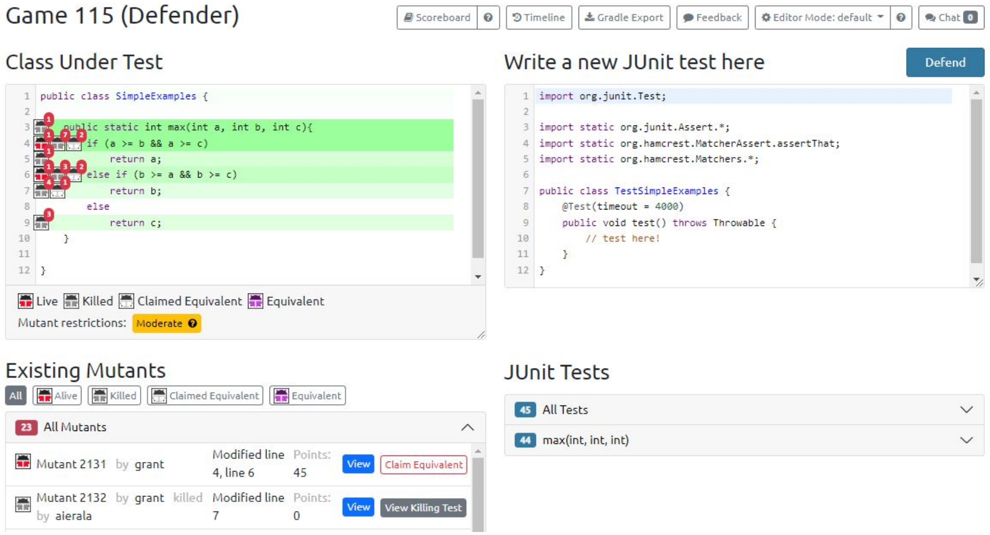
Mutant 2131 by grant	Modified line 4, line 6	Points: 45	View
Mutant 2132 by grant killed by aierala	Modified line 7	Points: 0	View View Killing Test
Mutant 2133 by grant killed by abombarda	Modified line 9	Points: 0	View View Killing Test
Mutant 2134 by grant killed by aibias	Modified line 5	Points: 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

```
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

```
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

```
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)

- For more details refer to Project Plan document

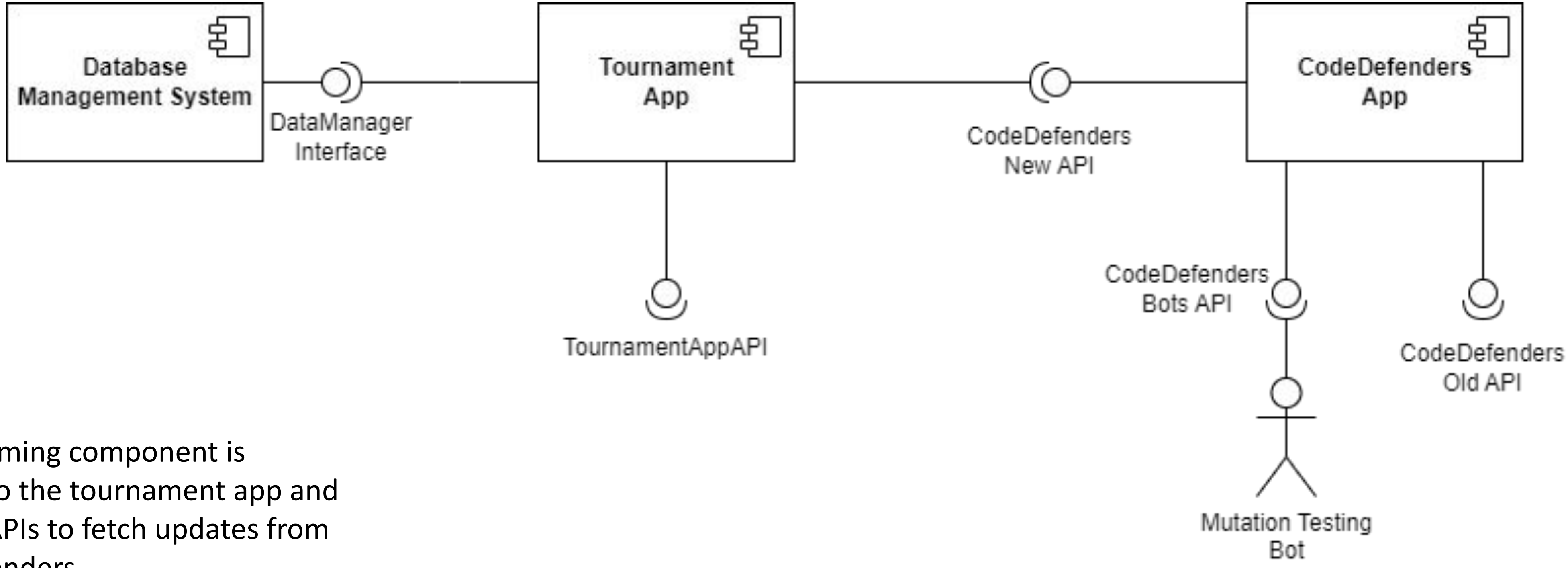
POLITECNICO
MILANO 1863

Mälardalen
University

Project requirements

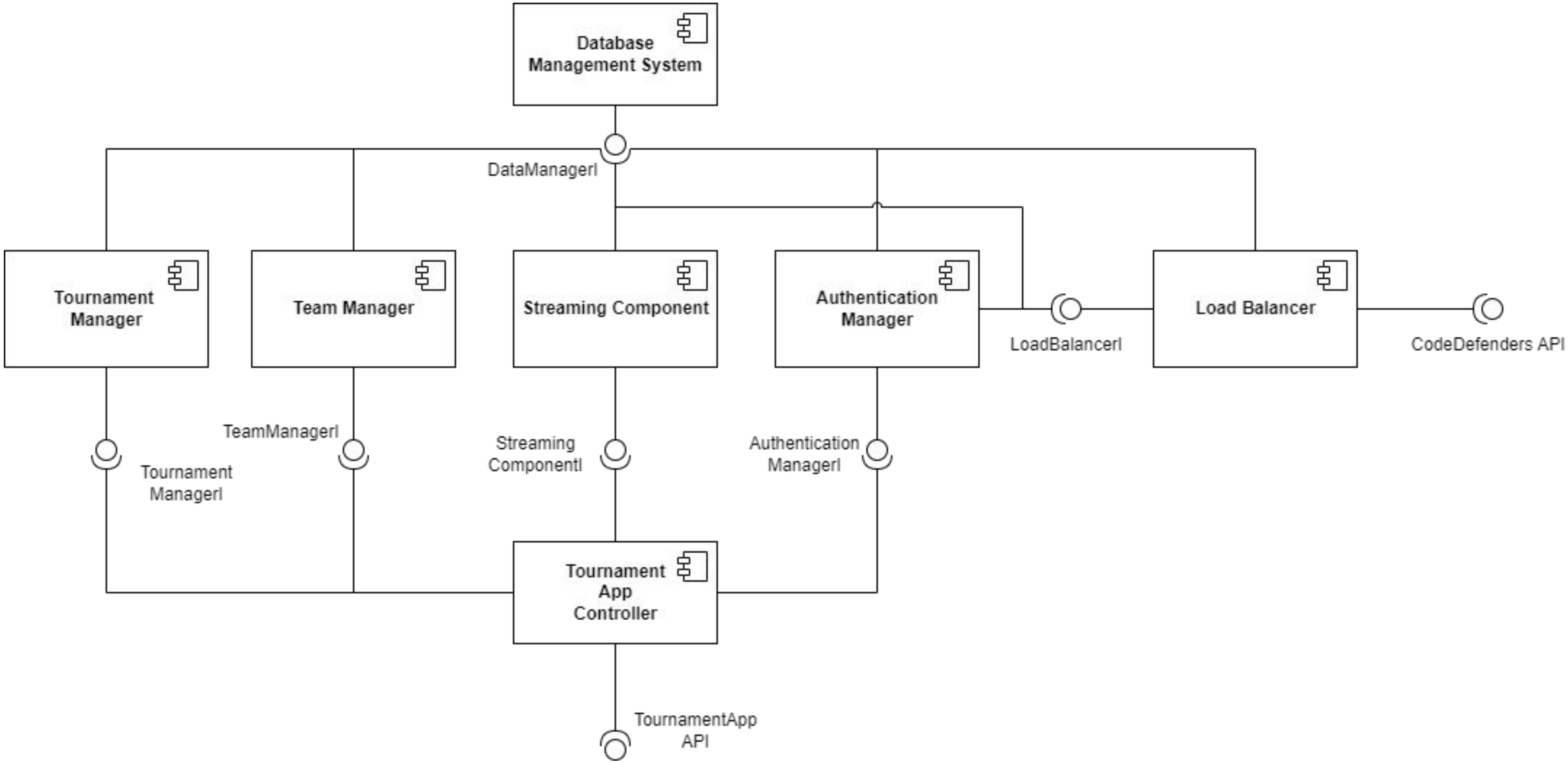
- Implement a **tournament application**. This application must use CodeDefenders as a remote service (through APIs) and must include at least two tournaments modalities.
- Design and implement a set of **OpenAPIs for CodeDefenders** which can be used from the tournament application to manage games and players.
- Implement a **load balancing** mechanism which allows the tournament application to communicate with **multiple CodeDefenders servers** and to always create games on the less loaded server.
- 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.
- Please refer to document *Requirements Definition* for more details

High-level component diagram

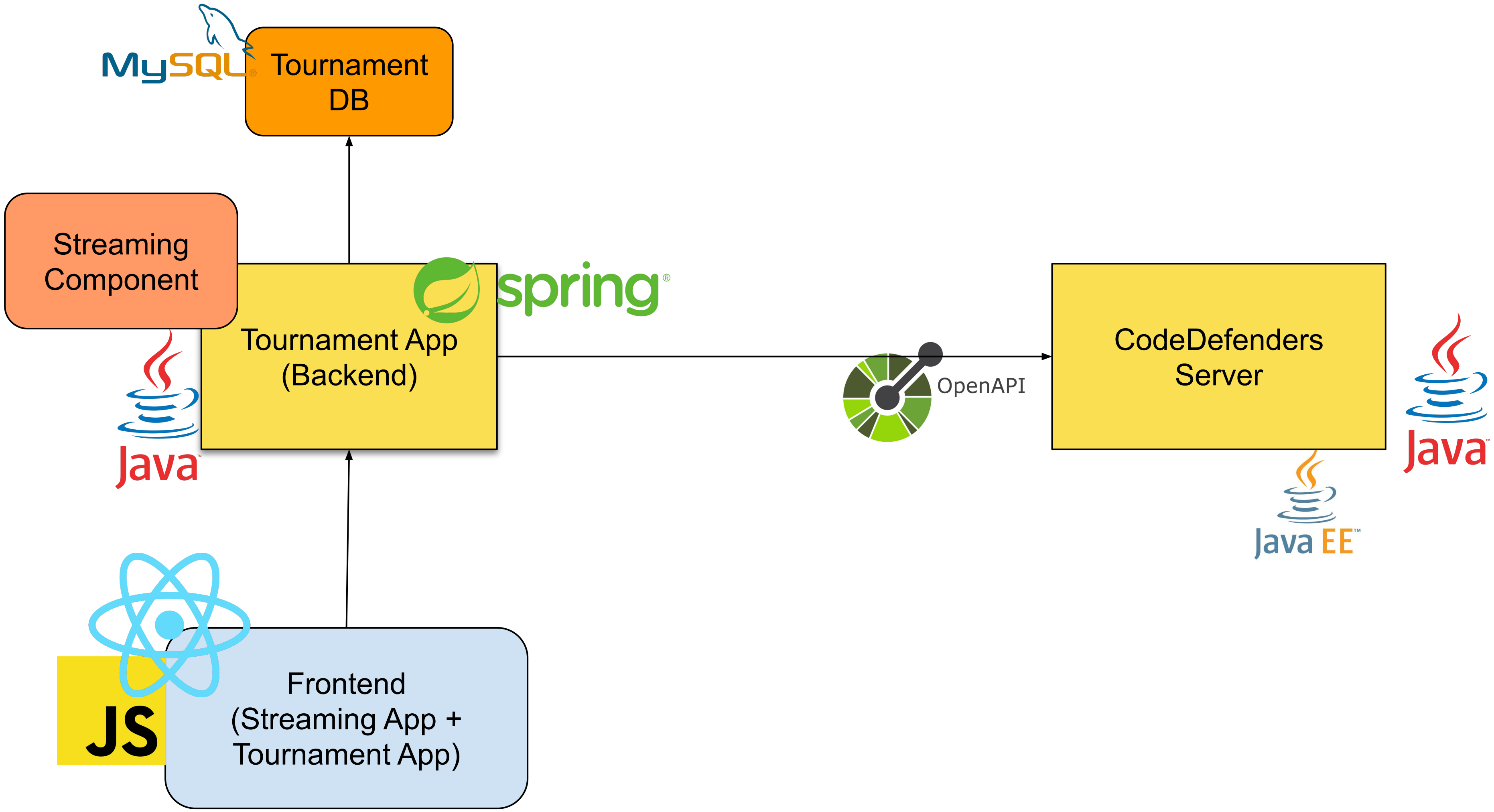


The streaming component is internal to the tournament app and exploits APIs to fetch updates from CodeDefenders

Zoom on the tournament app



Technologies we used



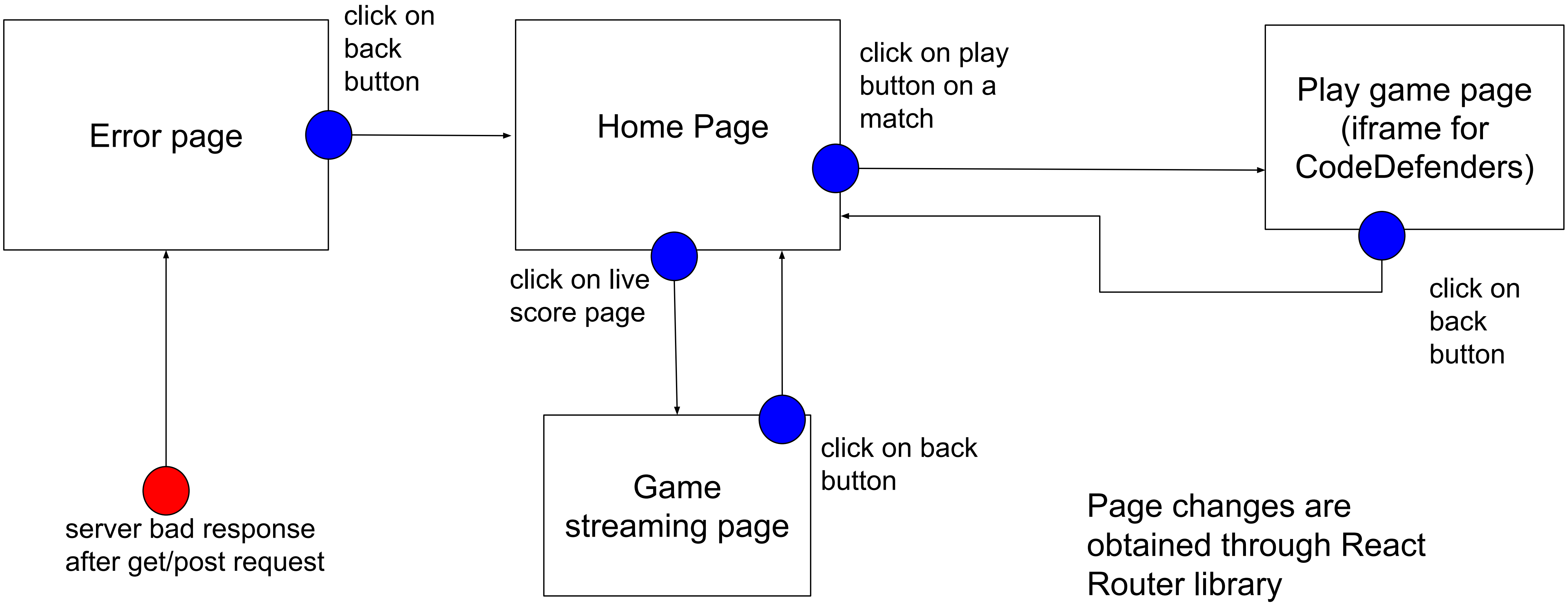
Frontend software design

Frontend is organized in three main pages:

- Home page: contains all the different sections of the Tournament Application (login and registration, tournaments, team creation and management)
- Play game page: embeds CodeDefenders interface to play the game with an overlay containing a timer and a back button
- Streaming page: shows streamed events and scores

An additional error page is used for unexpected errors.

Frontend high level pages division design

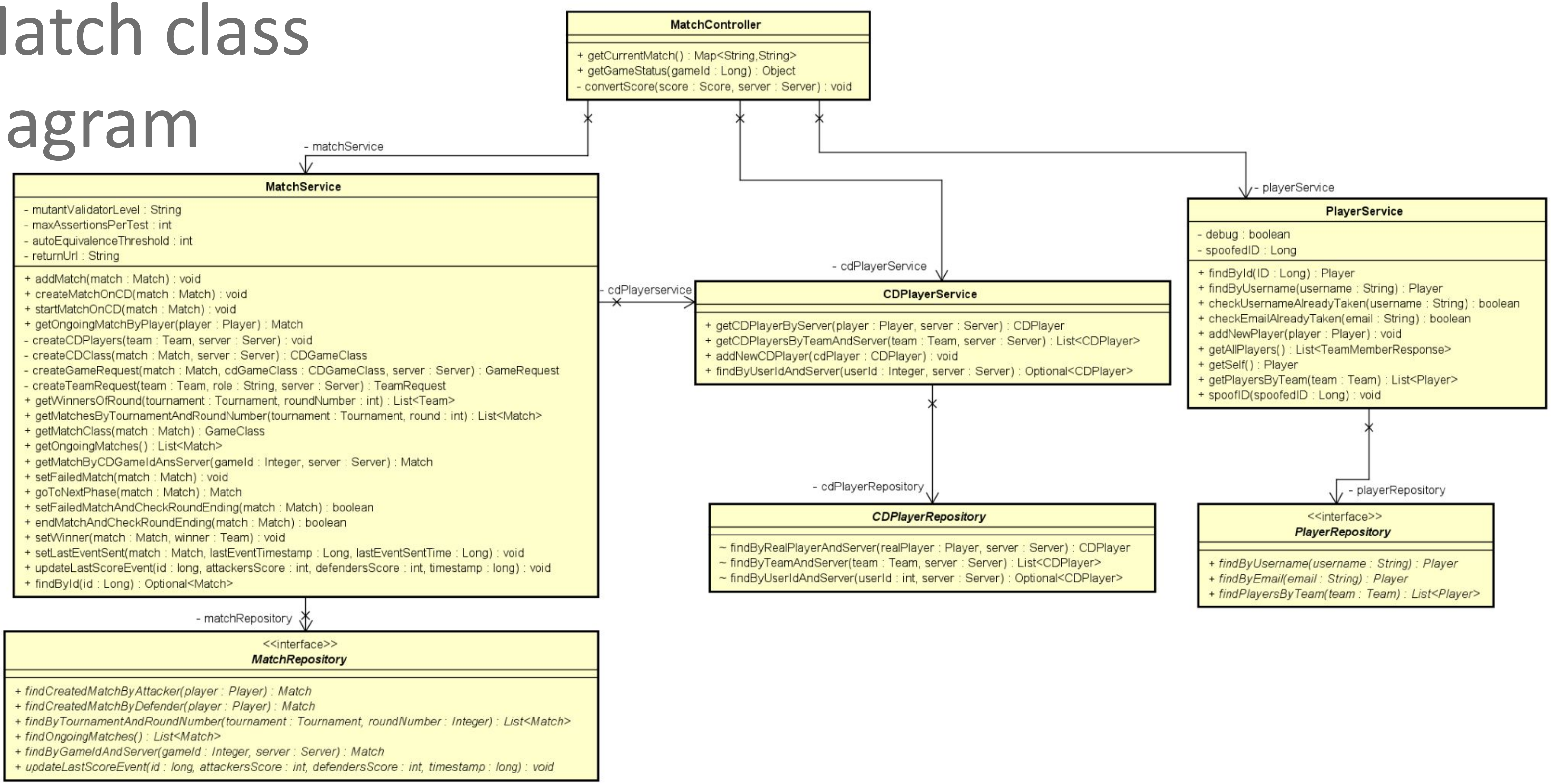


Page changes are obtained through React Router library

Backend software design

- Spring controllers: this layer interfaces with clients and receives their requests
- Spring services: this layer implements the business logic of the application
- JPA repositories and entities: this layer manages the communication with the database leveraging JPA functionalities

Match class diagram



MatchController

```
+ getCurrentMatch() : Map<String,String>
+ getGameStatus(gameId : Long) : Object
- convertScore(score : Score, server : Server) : void
```

MatchService

```
- mutantValidatorLevel : String
- maxAssertionsPerTest : int
- autoEquivalenceThreshold : int
- returnUrl : String

+ addMatch(match : Match) : void
+ createMatchOnCD(match : Match) : void
+ startMatchOnCD(match : Match) : void
+ getOngoingMatchByPlayer(player : Player) : Match
- createCDPlayers(team : Team, server : Server) : void
- createCDClass(match : Match, server : Server) : CDGameClass
- createGameRequest(match : Match, cdGameClass : CDGameClass, server : Server) : GameRequest
- createTeamRequest(team : Team, role : String, server : Server) : TeamRequest
+ getWinnersOfRound(tournament : Tournament, roundNumber : int) : List<Team>
+ getMatchesByTournamentAndRoundNumber(tournament : Tournament, round : int) : List<Match>
+ getMatchClass(match : Match) : GameClass
+ getOngoingMatches() : List<Match>
+ getMatchByCDGameIdAnsServer(gameId : Integer, server : Server) : Match
+ setFailedMatch(match : Match) : void
+ goToNextPhase(match : Match) : Match
+ setFailedMatchAndCheckRoundEnding(match : Match) : boolean
+ endMatchAndCheckRoundEnding(match : Match) : boolean
+ setWinner(match : Match, winner : Team) : void
+ setLastEventSent(match : Match, lastEventTimestamp : Long, lastEventSentTime : Long) : void
+ updateLastScoreEvent(id : long, attackersScore : int, defendersScore : int, timestamp : long) : void
+ findById(id : Long) : Optional<Match>
```

CDPlayerService

```
+ getCDPlayerByServer(player : Player, server : Server) : CDPlayer
+ getCDPlayersByTeamAndServer(team : Team, server : Server) : List<CDPlayer>
+ addNewCDPlayer(cdPlayer : CDPlayer) : void
+ findByUserIdAndServer(userId : Integer, server : Server) : Optional<CDPlayer>
```

PlayerService

```
- debug : boolean
- spoofedID : Long

+ 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
+ getPlayersByTeam(team : Team) : List<Player>
+ spoofID(spoofedID : Long) : void
```

CDPlayerRepository

```
~ findByRealPlayerAndServer(realPlayer : Player, server : Server) : CDPlayer
~ findByTeamAndServer(team : Team, server : Server) : List<CDPlayer>
~ findByUserIdAndServer(userId : int, server : Server) : Optional<CDPlayer>
```

<<interface>>
PlayerRepository

```
+ findByUsername(username : String) : Player
+ findByEmail(email : String) : Player
+ findPlayersByTeam(team : Team) : List<Player>
```

<<interface>>
MatchRepository

```
+ findCreatedMatchByAttacker(player : Player) : Match
+ findCreatedMatchByDefender(player : Player) : Match
+ findByTournamentAndRoundNumber(tournament : Tournament, roundNumber : Integer) : List<Match>
+ findOngoingMatches() : List<Match>
+ findByGameIdAndServer(gameId : Integer, server : Server) : Match
+ updateLastScoreEvent(id : long, attackersScore : int, defendersScore : int, timestamp : long) : void
```

Some details have been omitted for readability

Functionalities of final product

- Authentication: login and registration
- Team:
  - Create team
  - Join team
  - Invite other players
  - Kick team member
  - Manage team
- Game streaming
- Bots API

- Tournament
  - Create tournament
  - Display tournaments
  - Join tournament
  - Start tournament
  - Progress through the tournament
  - Tournament ending
- Game
  - Start game
  - Play game on CodeDefenders

Future improvements for our project

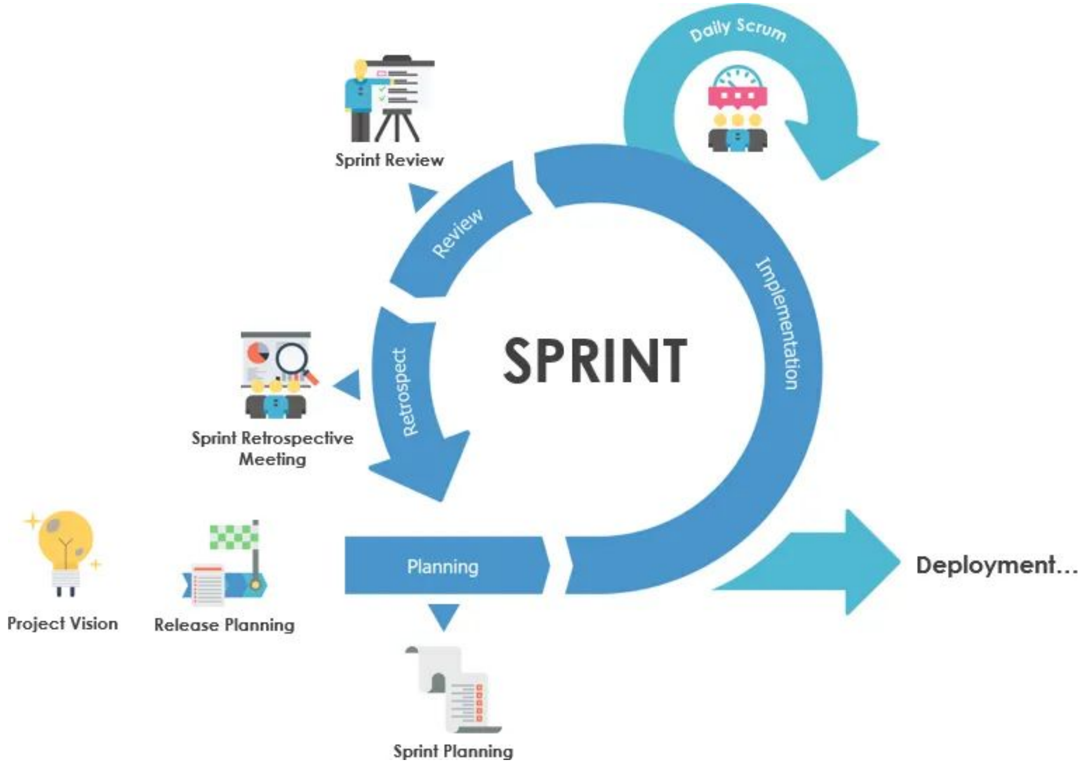
- Increase the possibility to customize game settings
  - Let the tournament creator select additional games settings (as in CodeDefenders) and allow the possibility of creating melee games that are currently not supported by the Tournament Application
- Improve interface and usability
  - Interface can be improved by adding a list of sent invitations in the teams management section and a tournament tree for knockout tournaments
- Matchmaking
  - Add the possibility to pair opposing teams based on their strength to increase fairness
- Fault tolerance to CodeDefenders failures
  - Implement a fault tolerance mechanism allowing the application to reschedule games of failed CodeDefenders instances on other active servers
- Improve support for Bots
  - Add some kind of filtering to allow the bot to request only the type of data it needs. Implement some default AI bots able to play CodeDefenders

SCRUM process

It is all thanks to SCRUM and our team members.

When we started this project, only two members had some experience with scrum process.

However, sprint by sprint we showed our commitment by recognizing issues we came across and finding a way to improve our team self-organization and project management.



Backlog grooming proved to be very helpful

Sprint Retrospective helped us discover what we could improve and what are we happy with

Sprints time plan

Sprint 0 : 10.10.2022. - 17.10.2022.

Sprint 1 : 18.10.2022. - 31.10.2022.

Sprint 2 : 1.11.2022. - 14.11.2022.

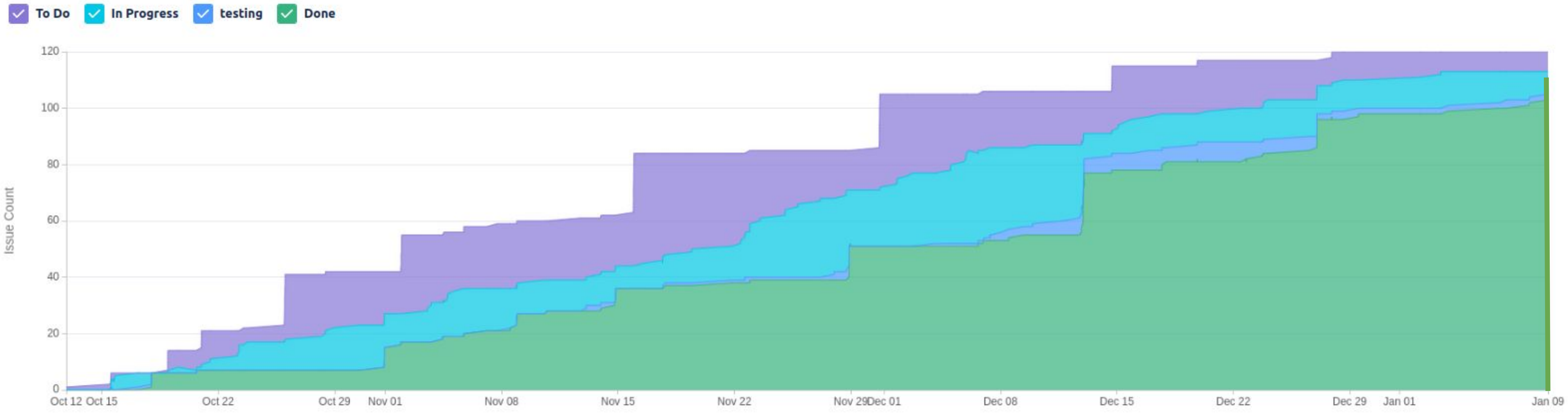
Sprint 3 : 15.11.2022. - 28.11.2022.

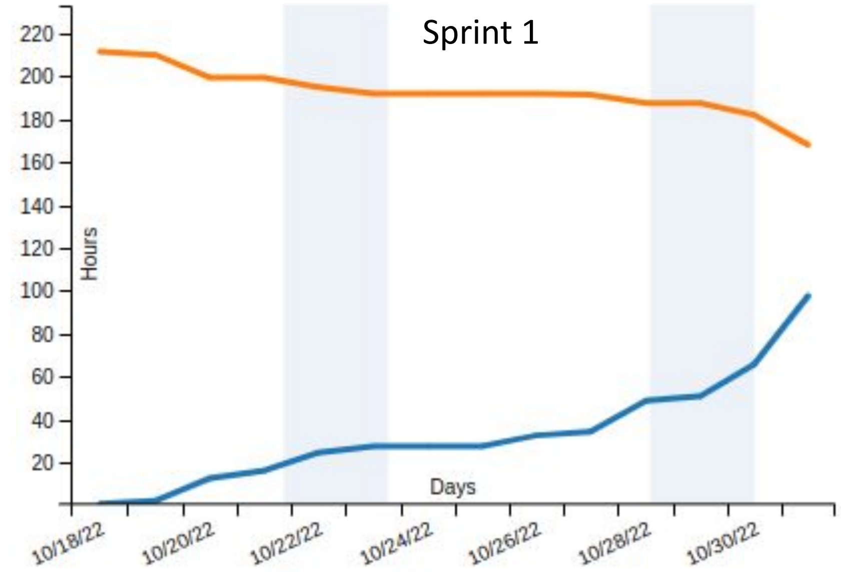
Sprint 4 : 30.11.2022. - 12.12.2022.

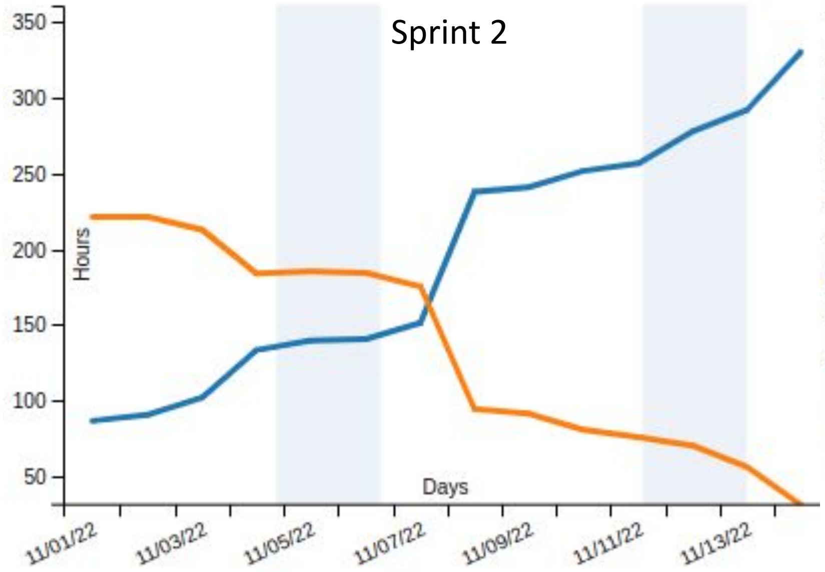
Sprint 5 : 14.12.2022. - 25.11.2000.

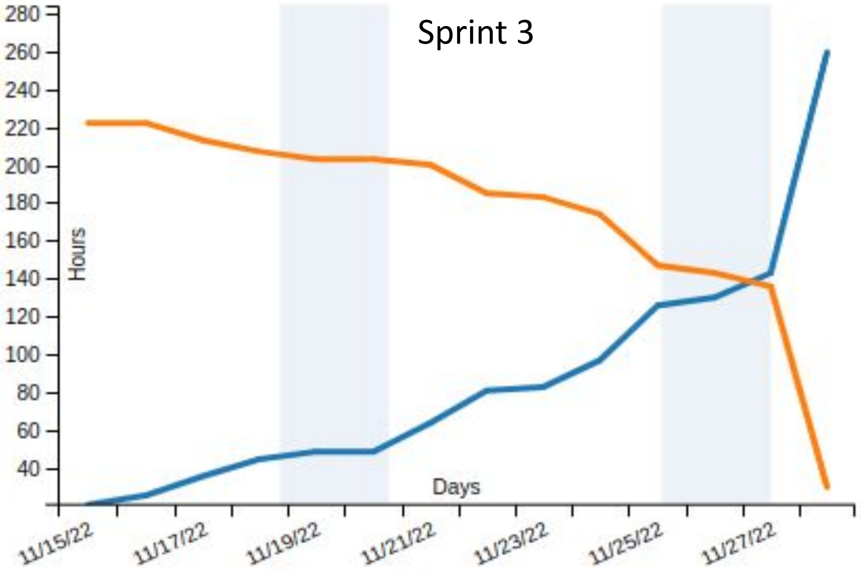
Sprint 6 : 27.12.2022. - 9.1.2023.

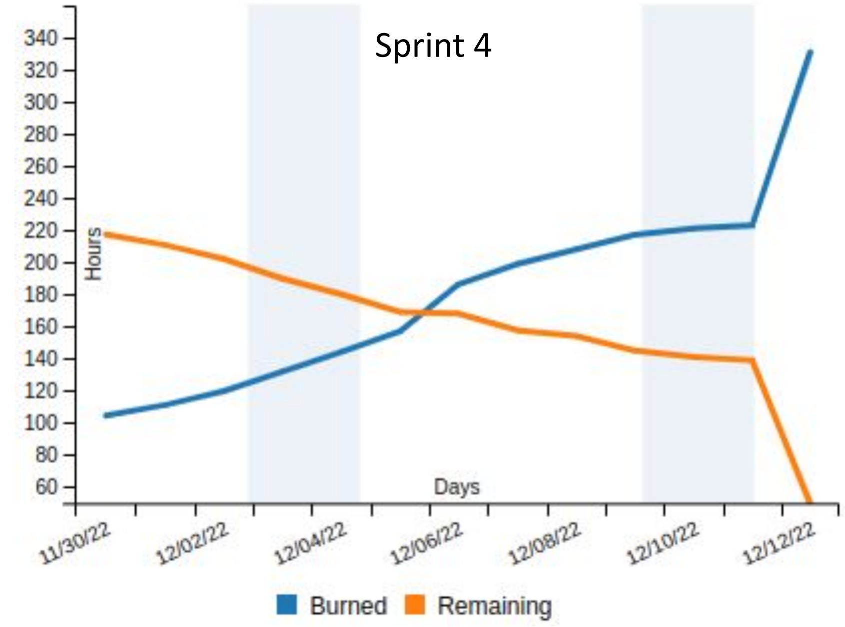
Cumulative Flow Diagram

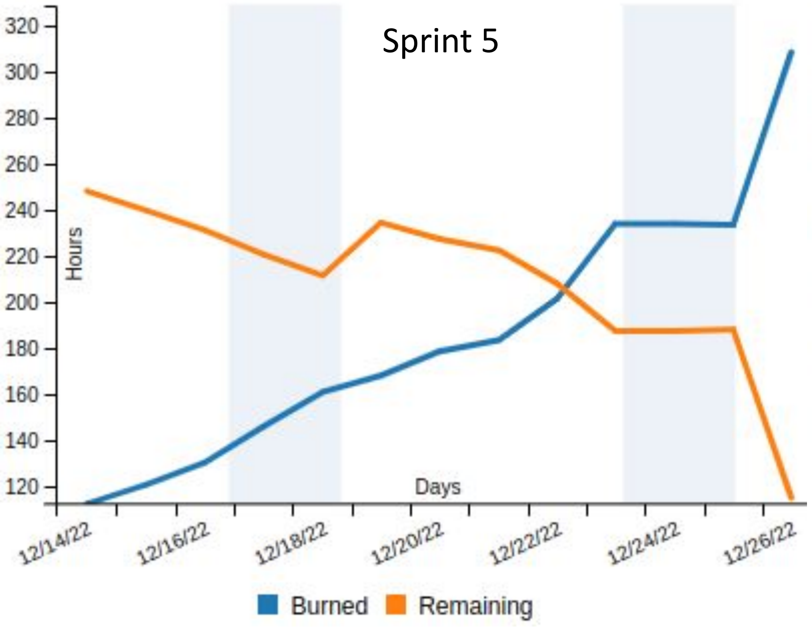


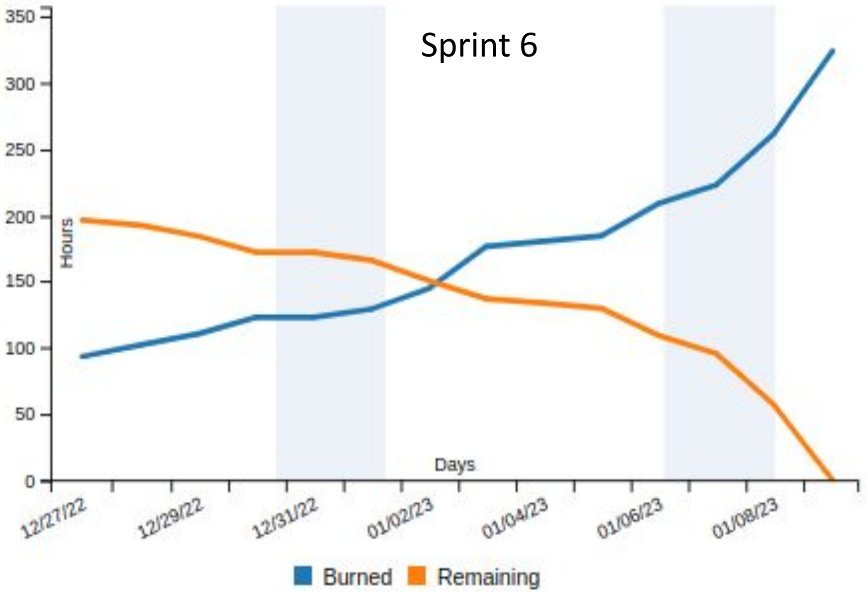

Sprint 1

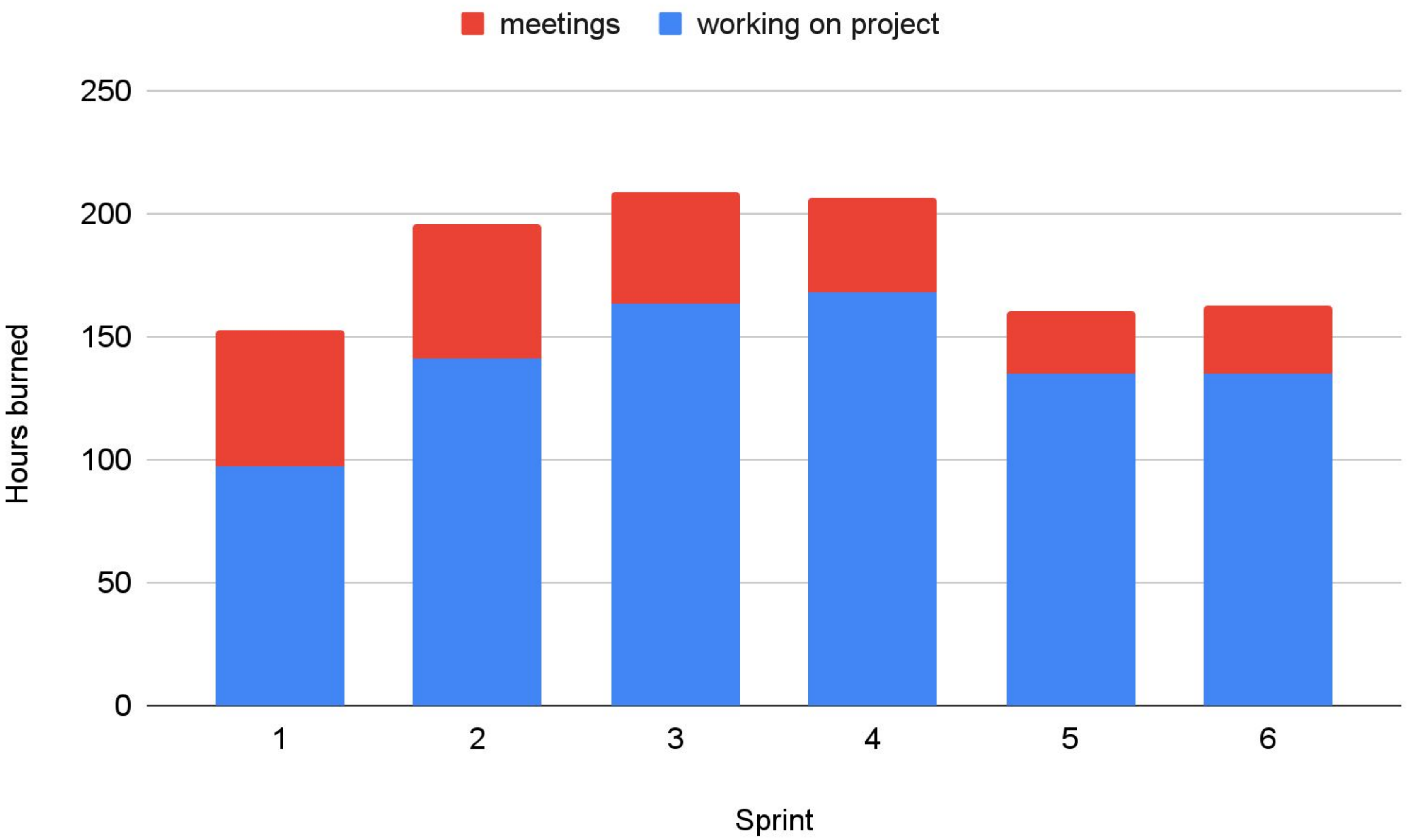

Sprint 2

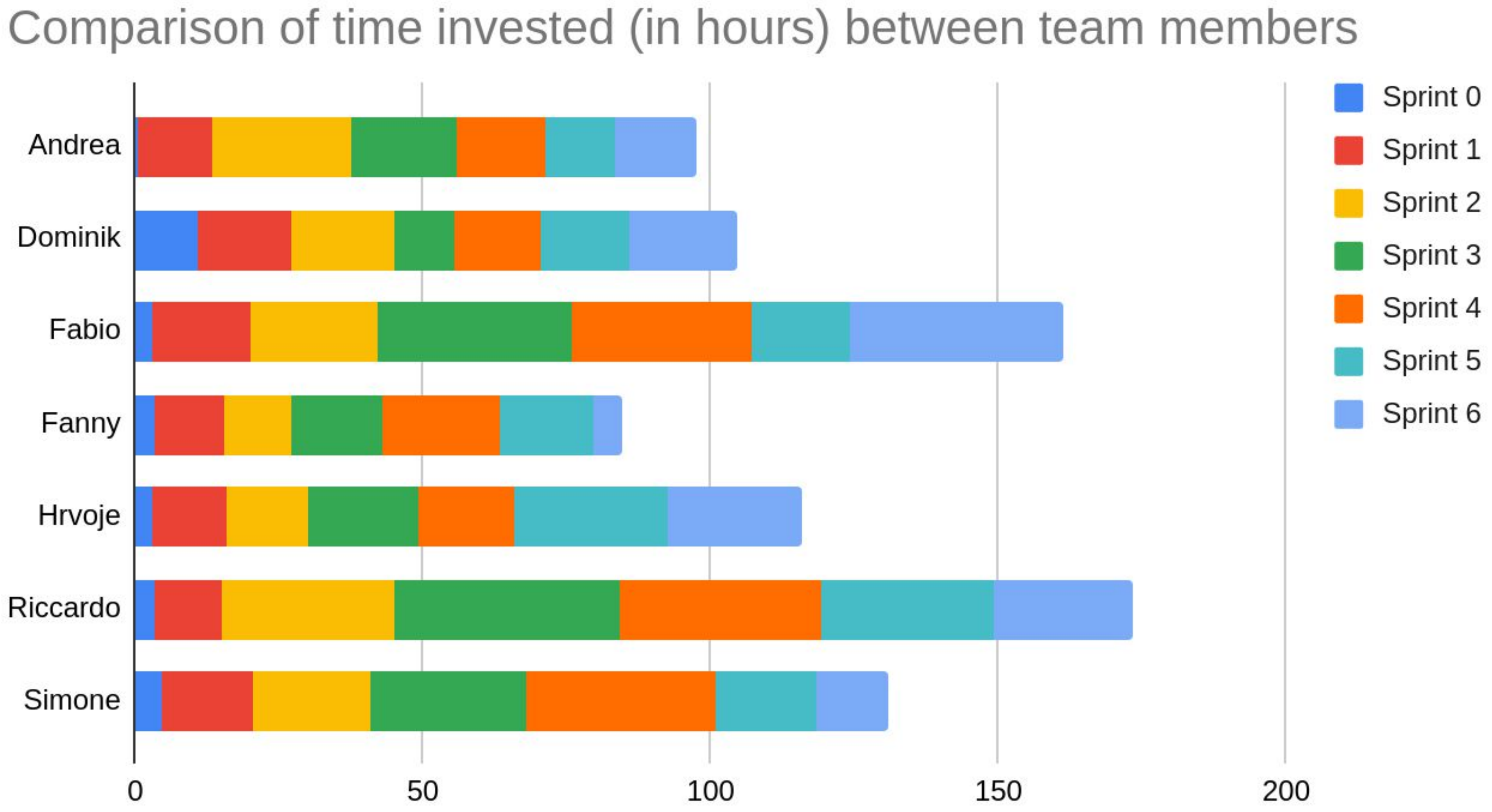

Sprint 3


Sprint 4


Sprint 5


Sprint 6


🟥 meetings
🟦 working on project
Hours burned
250
200
150
100
50
0
1
2
3
4
5
6
Sprint


Comparison of time invested (in hours) between team members
Andrea
Dominik
Fabio
Fanny
Hrvoje
Riccardo
Simone
0
50
100
150
200
Sprint 0
Sprint 1
Sprint 2
Sprint 3
Sprint 4
Sprint 5
Sprint 6

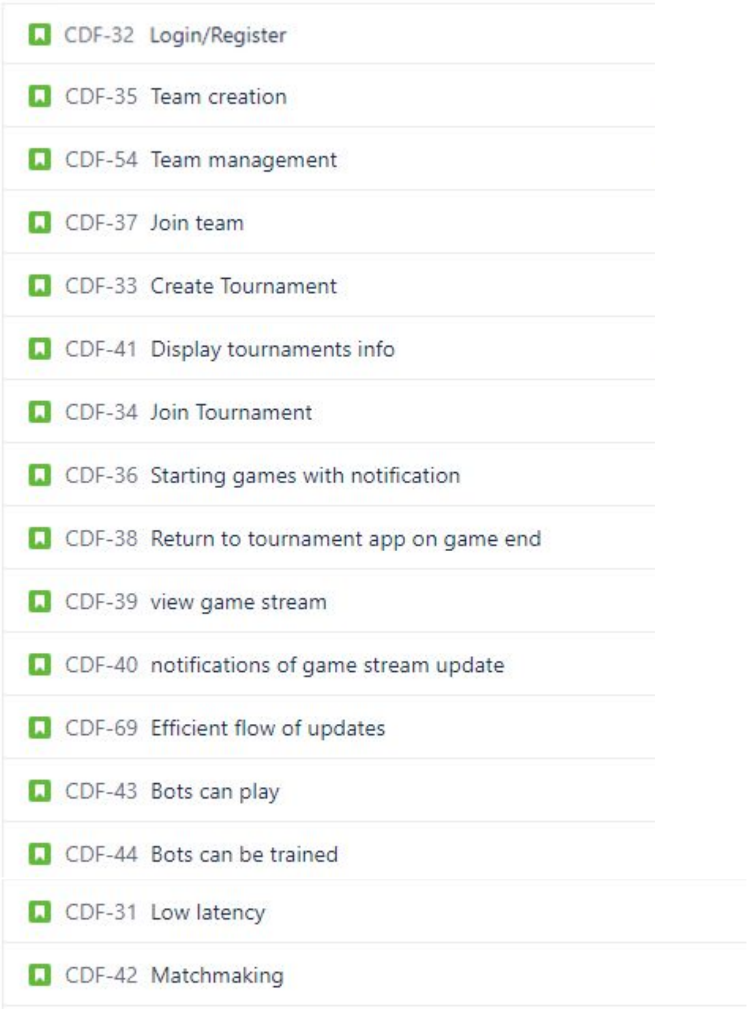
Competence Matrix

Name of the team member	Communication skills	Programming	Writing documentation	Agile process
Fanny	8	6 -> 7	6	8
Dominik	7	7	6 -> 7	7 -> 8
Hrvoje	6 -> 7	4 -> 6	8	1 -> 6
Andrea	8	6	6 -> 7	0 -> 6
Fabio	6	8	6 -> 7	0 -> 5
Riccardo	4 -> 5	8	6 -> 7	0 -> 6
Simone	4 -> 4.5	8 -> 8.5	6	0 -> 6

* skills are ranked 0-10 (0 meaning no skill at all, and 10 meaning excellent)

Product backlog burndown

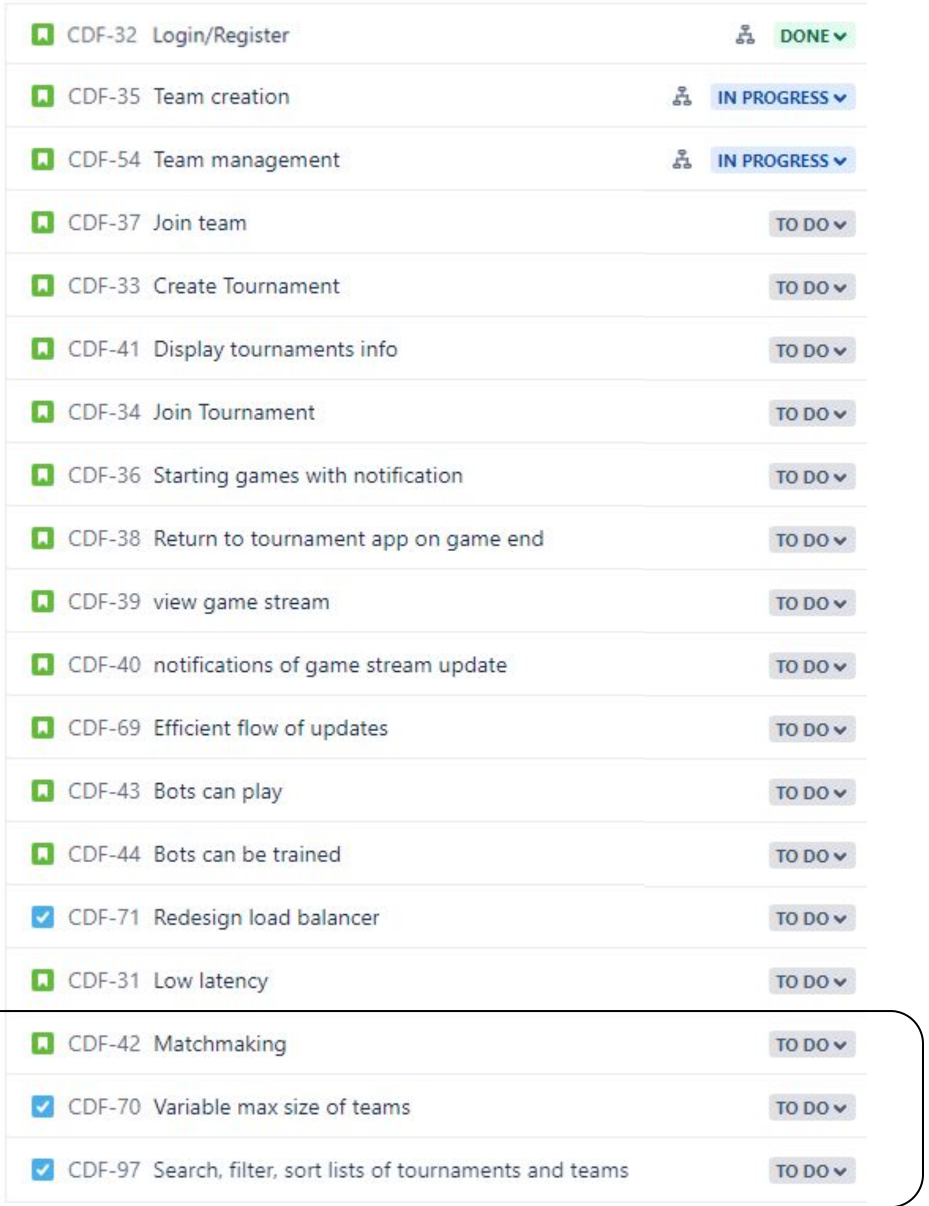
After sprint 1



CDF-32	Login/Register
CDF-35	Team creation
CDF-54	Team management
CDF-37	Join team
CDF-33	Create Tournament
CDF-41	Display tournaments info
CDF-34	Join Tournament
CDF-36	Starting games with notification
CDF-38	Return to tournament app on game end
CDF-39	view game stream
CDF-40	notifications of game stream update
CDF-69	Efficient flow of updates
CDF-43	Bots can play
CDF-44	Bots can be trained
CDF-31	Low latency
CDF-42	Matchmaking

Product backlog burndown

After sprint 2

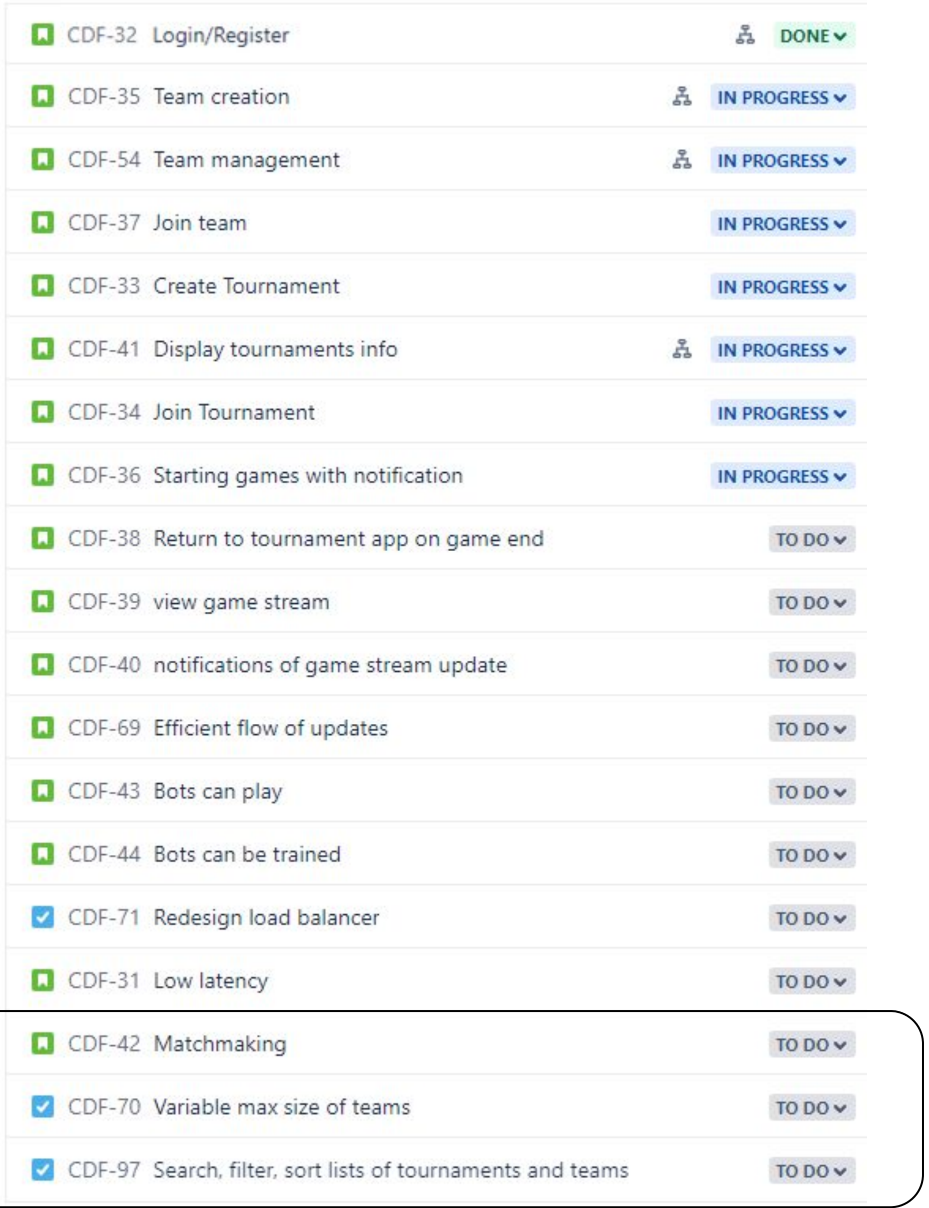


CDF-32	Login/Register	DONE
CDF-35	Team creation	IN PROGRESS
CDF-54	Team management	IN PROGRESS
CDF-37	Join team	TO DO
CDF-33	Create Tournament	TO DO
CDF-41	Display tournaments info	TO DO
CDF-34	Join Tournament	TO DO
CDF-36	Starting games with notification	TO DO
CDF-38	Return to tournament app on game end	TO DO
CDF-39	view game stream	TO DO
CDF-40	notifications of game stream update	TO DO
CDF-69	Efficient flow of updates	TO DO
CDF-43	Bots can play	TO DO
CDF-44	Bots can be trained	TO DO
CDF-71	Redesign load balancer	TO DO
CDF-31	Low latency	TO DO
CDF-42	Matchmaking	TO DO
CDF-70	Variable max size of teams	TO DO
CDF-97	Search, filter, sort lists of tournaments and teams	TO DO

not mandatory

Product backlog burndown

After sprint 3

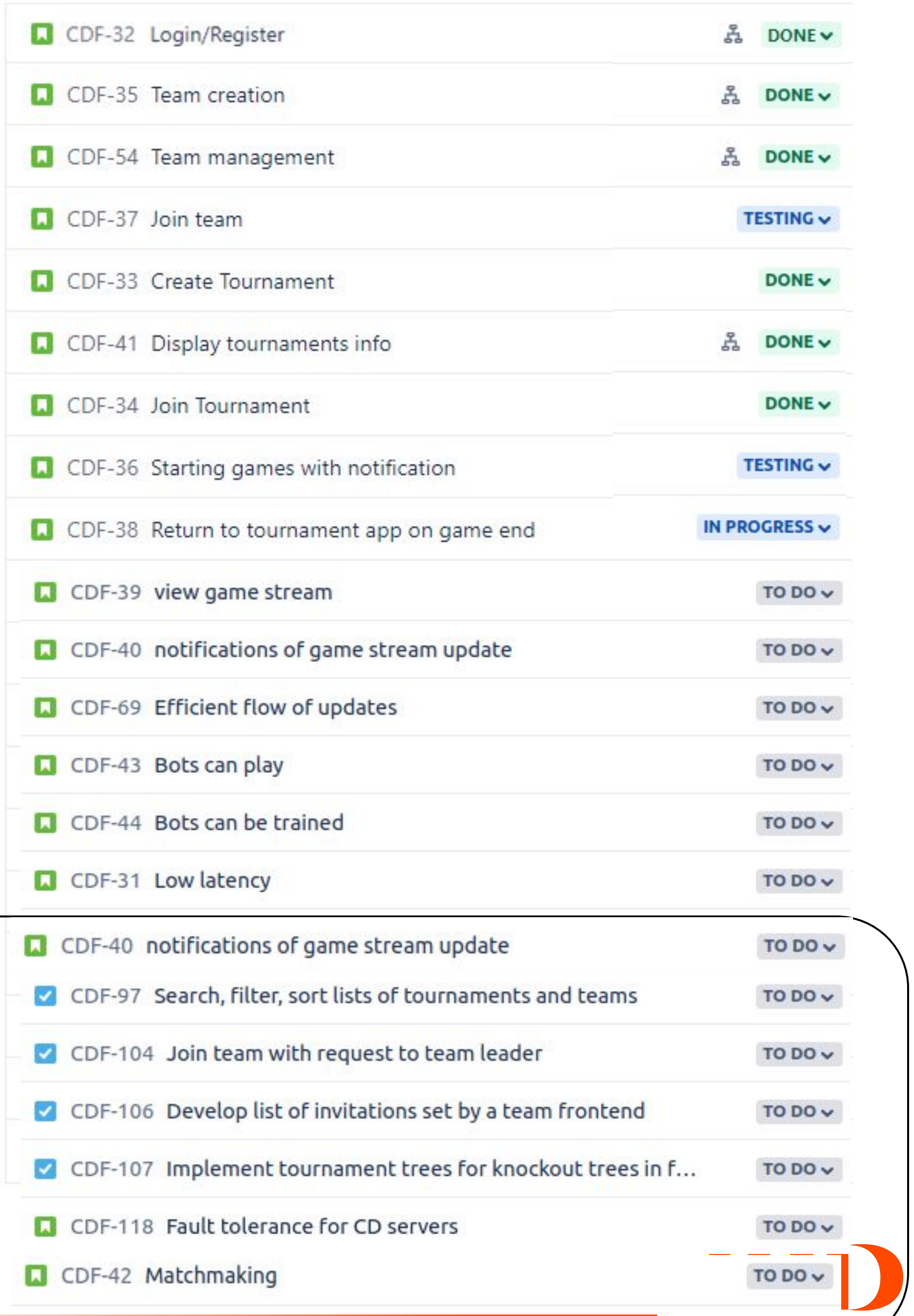


🟩	CDF-32	Login/Register	DONE
🟩	CDF-35	Team creation	IN PROGRESS
🟩	CDF-54	Team management	IN PROGRESS
🟩	CDF-37	Join team	IN PROGRESS
🟩	CDF-33	Create Tournament	IN PROGRESS
🟩	CDF-41	Display tournaments info	IN PROGRESS
🟩	CDF-34	Join Tournament	IN PROGRESS
🟩	CDF-36	Starting games with notification	IN PROGRESS
🟩	CDF-38	Return to tournament app on game end	TO DO
🟩	CDF-39	view game stream	TO DO
🟩	CDF-40	notifications of game stream update	TO DO
🟩	CDF-69	Efficient flow of updates	TO DO
🟩	CDF-43	Bots can play	TO DO
🟩	CDF-44	Bots can be trained	TO DO
☑	CDF-71	Redesign load balancer	TO DO
🟩	CDF-31	Low latency	TO DO
🟩	CDF-42	Matchmaking	TO DO
☑	CDF-70	Variable max size of teams	TO DO
☑	CDF-97	Search, filter, sort lists of tournaments and teams	TO DO

not mandatory

Product backlog burndown

After sprint 4

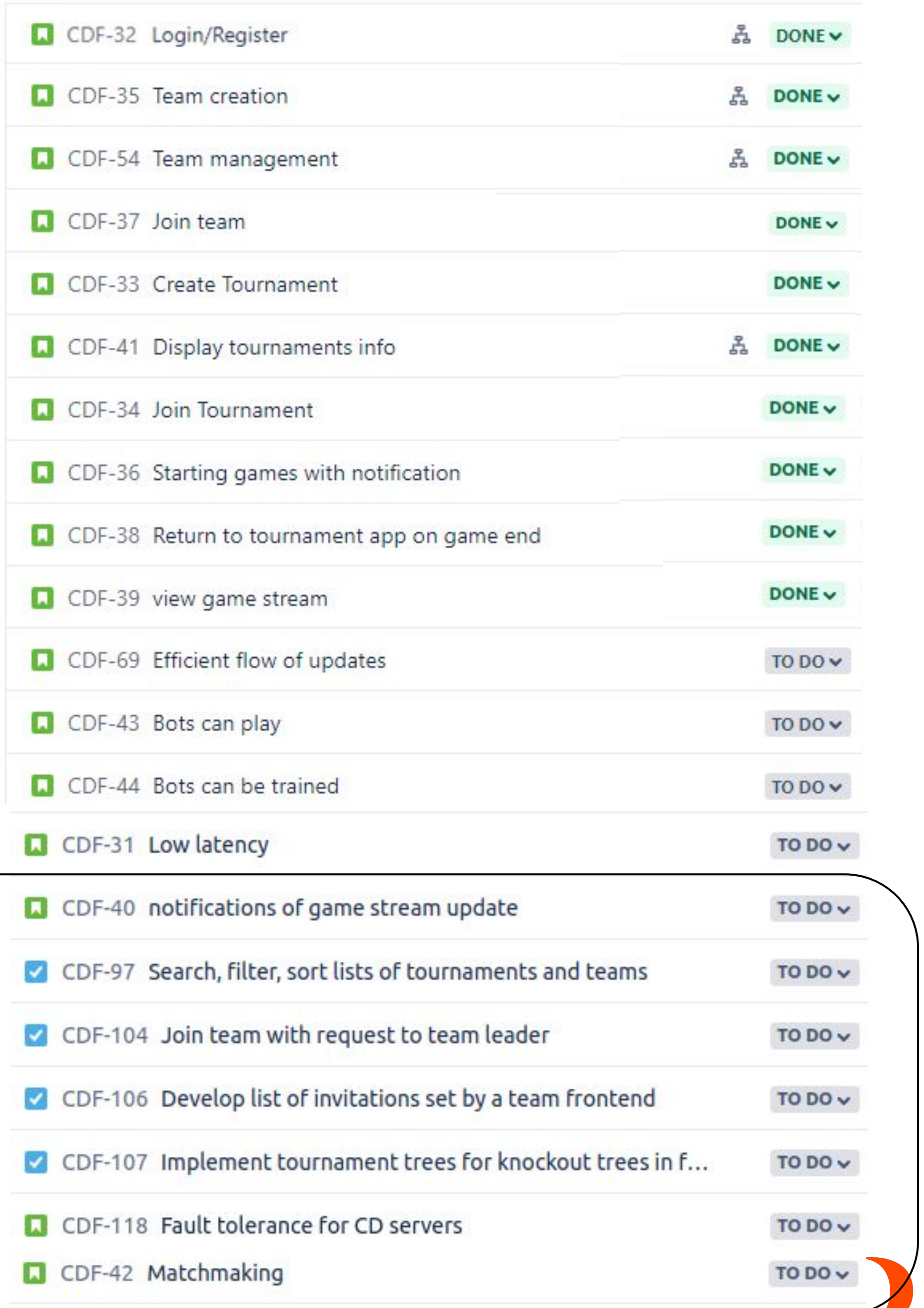


CDF-32	Login/Register	DONE
CDF-35	Team creation	DONE
CDF-54	Team management	DONE
CDF-37	Join team	TESTING
CDF-33	Create Tournament	DONE
CDF-41	Display tournaments info	DONE
CDF-34	Join Tournament	DONE
CDF-36	Starting games with notification	TESTING
CDF-38	Return to tournament app on game end	IN PROGRESS
CDF-39	view game stream	TO DO
CDF-40	notifications of game stream update	TO DO
CDF-69	Efficient flow of updates	TO DO
CDF-43	Bots can play	TO DO
CDF-44	Bots can be trained	TO DO
CDF-31	Low latency	TO DO
CDF-40	notifications of game stream update	TO DO
☑ CDF-97	Search, filter, sort lists of tournaments and teams	TO DO
☑ CDF-104	Join team with request to team leader	TO DO
☑ CDF-106	Develop list of invitations set by a team frontend	TO DO
☑ CDF-107	Implement tournament trees for knockout trees in f...	TO DO
CDF-118	Fault tolerance for CD servers	TO DO
CDF-42	Matchmaking	TO DO

not mandatory

Product backlog burndown

After sprint 5

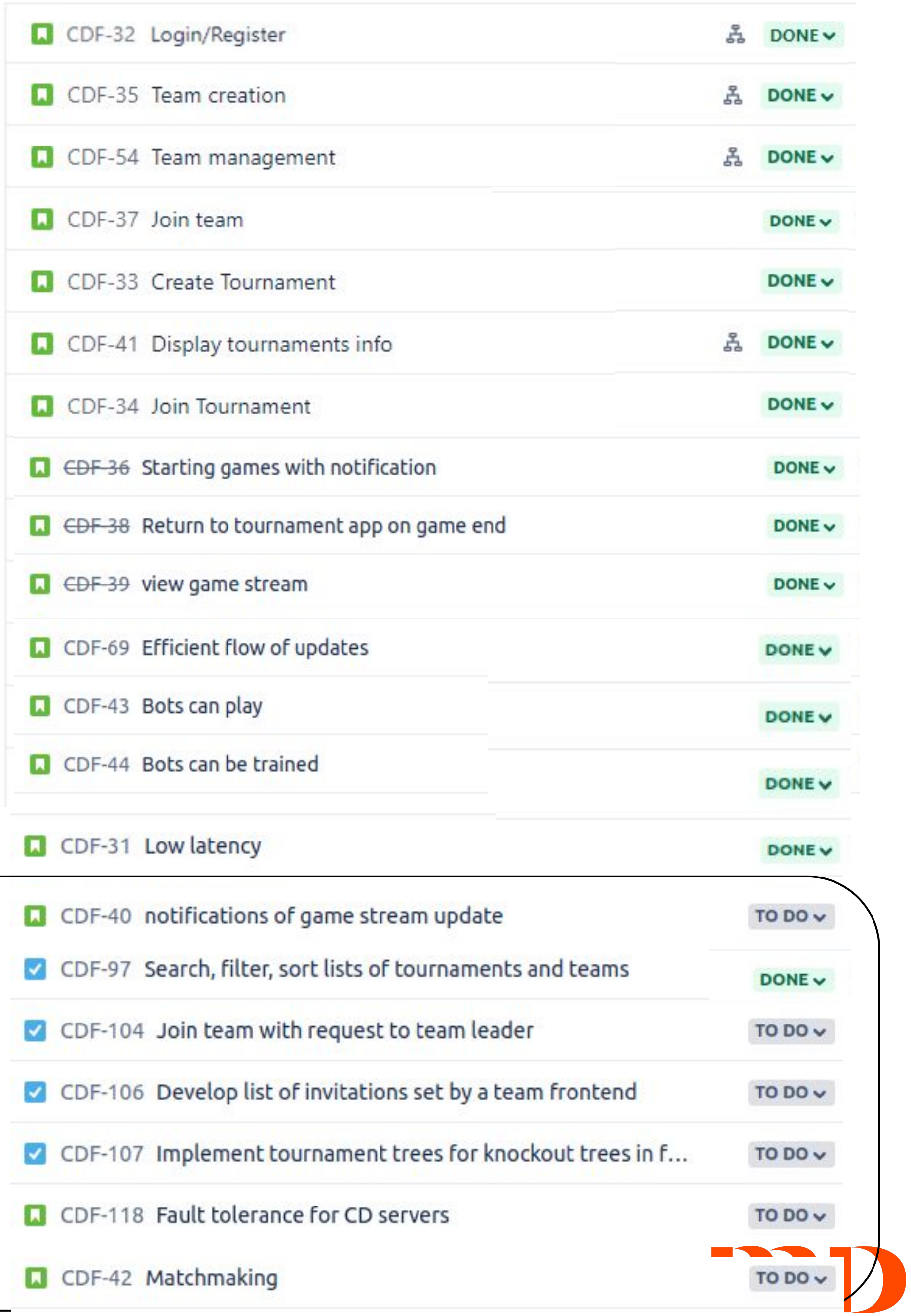


🟩	CDF-32	Login/Register	DONE ⌄
🟩	CDF-35	Team creation	DONE ⌄
🟩	CDF-54	Team management	DONE ⌄
🟩	CDF-37	Join team	DONE ⌄
🟩	CDF-33	Create Tournament	DONE ⌄
🟩	CDF-41	Display tournaments info	DONE ⌄
🟩	CDF-34	Join Tournament	DONE ⌄
🟩	CDF-36	Starting games with notification	DONE ⌄
🟩	CDF-38	Return to tournament app on game end	DONE ⌄
🟩	CDF-39	view game stream	DONE ⌄
🟩	CDF-69	Efficient flow of updates	TO DO ⌄
🟩	CDF-43	Bots can play	TO DO ⌄
🟩	CDF-44	Bots can be trained	TO DO ⌄
🟩	CDF-31	Low latency	TO DO ⌄
🟩	CDF-40	notifications of game stream update	TO DO ⌄
☑	CDF-97	Search, filter, sort lists of tournaments and teams	TO DO ⌄
☑	CDF-104	Join team with request to team leader	TO DO ⌄
☑	CDF-106	Develop list of invitations set by a team frontend	TO DO ⌄
☑	CDF-107	Implement tournament trees for knockout trees in f...	TO DO ⌄
🟩	CDF-118	Fault tolerance for CD servers	TO DO ⌄
🟩	CDF-42	Matchmaking	TO DO ⌄

not mandatory

Product backlog burndown

After sprint 6



CDF-32	Login/Register	DONE
CDF-35	Team creation	DONE
CDF-54	Team management	DONE
CDF-37	Join team	DONE
CDF-33	Create Tournament	DONE
CDF-41	Display tournaments info	DONE
CDF-34	Join Tournament	DONE
~~CDF-36~~	Starting games with notification	DONE
~~CDF-38~~	Return to tournament app on game end	DONE
~~CDF-39~~	view game stream	DONE
CDF-69	Efficient flow of updates	DONE
CDF-43	Bots can play	DONE
CDF-44	Bots can be trained	DONE
CDF-31	Low latency	DONE
CDF-40	notifications of game stream update	TO DO
CDF-97	Search, filter, sort lists of tournaments and teams	DONE
CDF-104	Join team with request to team leader	TO DO
CDF-106	Develop list of invitations set by a team frontend	TO DO
CDF-107	Implement tournament trees for knockout trees in f...	TO DO
CDF-118	Fault tolerance for CD servers	TO DO
CDF-42	Matchmaking	TO DO

not mandatory

Experiences from the distributed project

What worked well

Teamwork atmosphere was amazing regardless of being distributed, yes there were issues but all of our team members were committed to improving our process.
We managed to finish our project as we expected or even better in some parts and that made us happy.
We learned new technologies and had experience with them on the project beyond trivial

Collaboration between team members with different backgrounds allowed to share and improve our skills, including ability to follow SCRUM process.
Despite being in different locations we managed to find a fixed schedule for meetings and to stick with it.

Incredible team
The division of roles was effective, there was a group of people for every aspect and someone to ask for feedback
The dailies, while sometimes hard to not miss, kept everyone up to date on the state of the project

Sharing knowledge and helping each other since we have different technical backgrounds
Organization was clear from the beginning
We took enough time at the beginning to set up organization tools and understand the project proposal
We didn’t had internal conflicts

We learned a lot of new technical things and also gained experience working in an amazing team

We learned to communicate with a real customer and to fulfill his requirements
We cheered together when Croatia made it to the semifinal of the World Cup
Even if we were strangers at the beginning of the project, everyone helped each other during the project
We learned to exploit SCRUM to organize our work

Although our teammates were from different country we saw that we have a lot of things in common as engineering students
Everybody took strong ownership of the things they did, nobody left their task half done and transferred it to somebody else because it was hard
Collaboration was very good and everybody was super helpful with transferring knowledge
Feedback was always constructive and made in a effort to improve our project and process

Experiences from the distributed project

What we struggled to do

Keeping track of all messages
Sometimes 2 weeks were not enough for finishing chunks of work (that couldn't be split into smaller tasks) that we took in sprints which shows that we struggled with task breakdown

We had some difficulties with some elements of the SCRUM process, such as task refinement and estimation and product backlog.
Adhering to our GitHub and testing strategy took some time.

Being the first time I experienced SCRUM, it was difficult for me to remember to log hours, update tasks status and also to familiarize with Jira
I struggled a bit to have a good coverage with unit testing because of the nature of the application (web application with little to no logic)

Working so much time at each sprint was sometimes difficult due to exams or personal reasons.
Evening meetings were sometimes tiring.

Balancing time between this project and all the other lectures and personal business
Having to handle the SCRUM aspects on top of the product was, while effective, more work to do

It was hard to commit to attend daily meetings
There was a lot of rules and deliverables for the DSD course which was annoying to write but I also recognize why we had to do it
Defining our process was sometimes hard to do because we didn't have a lot of previous experience working on a project like this

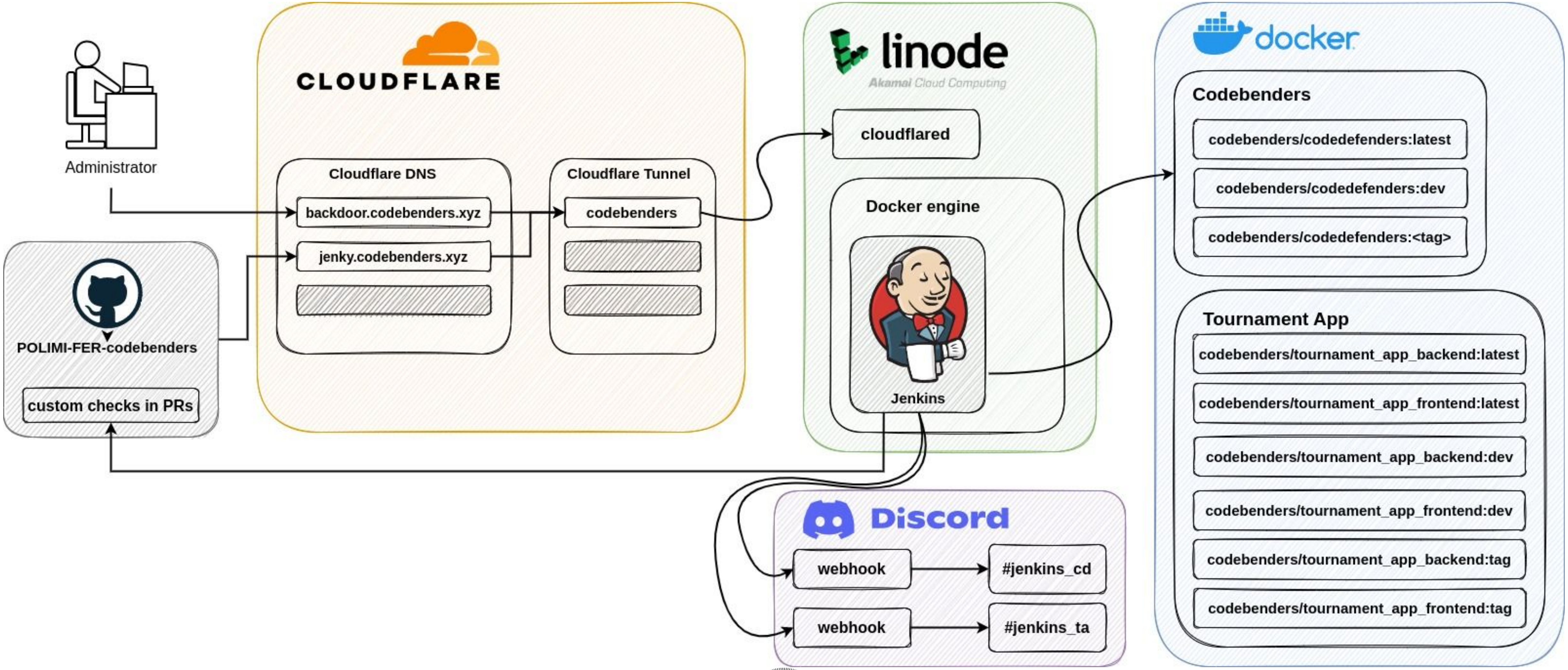
Learning without lectures a lot of technical stuff just searching it online

Experiences from the distributed project

Conclusions

- Even though we were new to SCRUM and working on projects of this scope, we managed to recognize our issues and improve in many aspects.
- We learned how to work in a distributed team. All of us showed efforts in maintaining good communication, collaboration and helping each other.
- We managed to finish the project and our customer made it clear that we did a great job.

Testing Infrastructure Diagram



Administrator

POLIMI-FER-codebenders

custom checks in PRs

CLOUDFLARE

Cloudflare DNS

backdoor.codebenders.xyz

jenky.codebenders.xyz

Cloudflare Tunnel

codebenders

linode

Akamai Cloud Computing

cloudflared

Docker engine

Jenkins

Discord

webhook

#jenkins_cd

webhook

#jenkins_ta

docker

Codebenders

codebenders/codedefenders:latest

codebenders/codedefenders:dev

codebenders/codedefenders:<tag>

Tournament App

codebenders/tournament_app_backend:latest

codebenders/tournament_app_frontend:latest

codebenders/tournament_app_backend:dev

codebenders/tournament_app_frontend:dev

codebenders/tournament_app_backend:tag

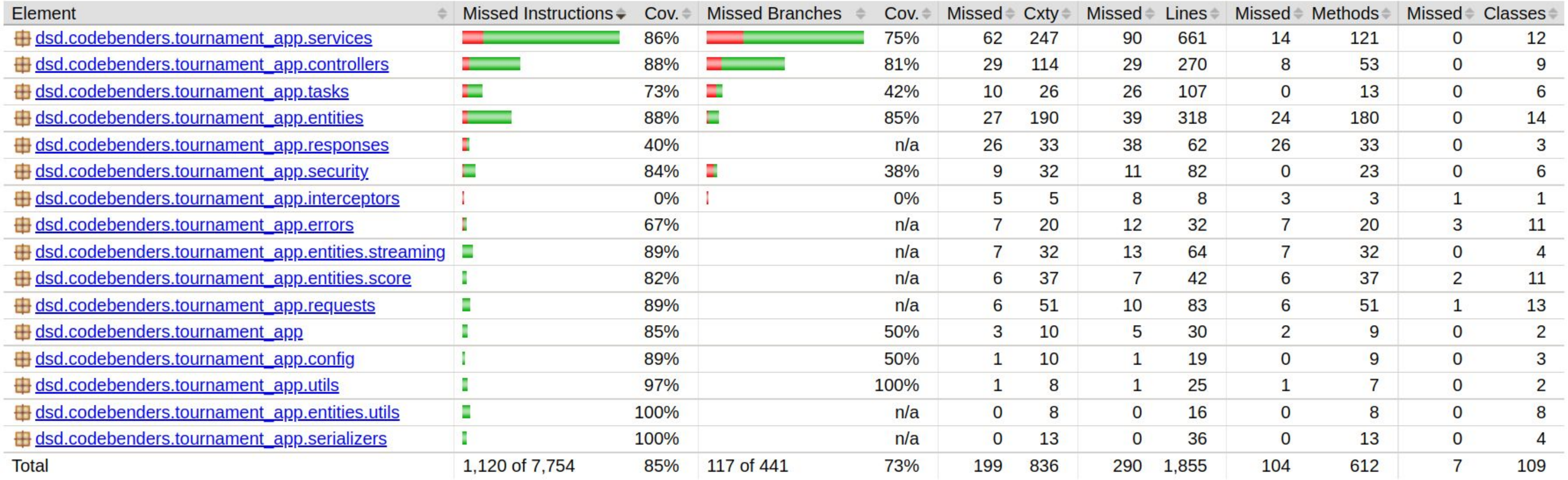
codebenders/tournament_app_frontend:tag

Unit and Integration Testing

- Backend testing:
  - written exploiting the unit testing framework by Spring and using JUnit 5 and Mockito
  - mainly focused on testing the service layer, containing the application logic
- Frontend testing: manual testing
- Integration testing:
  - using API requests to simulate usage of our backend and comparing expected and actual responses
  - Unirest library for making API requests

Testing coverage

Integration and unit tests combined achieved an overall coverage of **85%**.



Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
dsd.codebenders.tournament_app.services		86%		75%	62	247	90	661	14	121	0	12
dsd.codebenders.tournament_app.controllers		88%		81%	29	114	29	270	8	53	0	9
dsd.codebenders.tournament_app.tasks		73%		42%	10	26	26	107	0	13	0	6
dsd.codebenders.tournament_app.entities		88%		85%	27	190	39	318	24	180	0	14
dsd.codebenders.tournament_app.responses		40%		n/a	26	33	38	62	26	33	0	3
dsd.codebenders.tournament_app.security		84%		38%	9	32	11	82	0	23	0	6
dsd.codebenders.tournament_app.interceptors		0%		0%	5	5	8	8	3	3	1	1
dsd.codebenders.tournament_app.errors		67%		n/a	7	20	12	32	7	20	3	11
dsd.codebenders.tournament_app.entities.streaming		89%		n/a	7	32	13	64	7	32	0	4
dsd.codebenders.tournament_app.entities.score		82%		n/a	6	37	7	42	6	37	2	11
dsd.codebenders.tournament_app.requests		89%		n/a	6	51	10	83	6	51	1	13
dsd.codebenders.tournament_app		85%		50%	3	10	5	30	2	9	0	2
dsd.codebenders.tournament_app.config		89%		50%	1	10	1	19	0	9	0	3
dsd.codebenders.tournament_app.utils		97%		100%	1	8	1	25	1	7	0	2
dsd.codebenders.tournament_app.entities.utils		100%		n/a	0	8	0	16	0	8	0	8
dsd.codebenders.tournament_app.serializers		100%		n/a	0	13	0	36	0	13	0	4
Total	1,120 of 7,754	85%	117 of 441	73%	199	836	290	1,855	104	612	7	109

POLITECNICO
MILANO 1863

Mälardalen
University

Validation: acceptance test plan

- We listed all the actions that can be performed in our application
  - Each action maps one or more requirements by the customer and is covered by one specific acceptance test
  - Each test has ID, name and description (what action it is testing), link to User Story containing mapped project requirement, prerequisite (if any), procedure (instructions to carry out the test), and how to evaluate results (pass criteria, if not met - test failed)
- We planned to go through all the tests together with our customer to verify that they are successful and that the behavior of the application is the one required by him

Validation: acceptance test report

- As planned, we had a meeting with the customer on Saturday 07/01/2023 to show him the final version of the product and go through acceptance tests together
- During the meeting we went through our list of acceptance tests and performed them together
  - 100% of the tests were successful
  - The customer expressed his satisfaction for the product we implemented and commented on some possible future improvements
- Some of the optional requirements weren’t implemented and, as such, were not covered by our acceptance tests. Overall, the acceptance tests covered 27/31 of the requirements initially identified

Summary of the acceptance tests

User Story	Test	Status	Comment
CDF-32 Login/Register	Test-1 Registration	PASS	
	Test-2.1 Successful Login	PASS	
	Test-2.2 Rejected Login	PASS	
CDF-41 Display Tournaments Info	Test-3 Display Tournaments Information	PASS	
CDF-35 Team Creation	Test-4 Create a Team	PASS	
CDF-54 Team Management	Test-5 Leave the Team	PASS	
	Test-6 Kick Members Out of the Team	PASS	
	Test-7 Promote Team Member as Leader	PASS	
	Test-8 Invite Players to the Team	PASS	

User Story	Test	Status	Comment
CDF-37 Join Team	Test-9 Join an Open Team	PASS	
	Test-10.1 Accept Received Invitations	PASS	
	Test-10.2 Decline Received Invitations	PASS	
CDF-33 Create Tournament	Test-11 Create Tournament	PASS	
	Test-20 Upload a Class	PASS	
	Test-21 Choose a Class	PASS	
CDF-34 Join Tournament	Test-12 Join Tournament	PASS	*Teams with too many players can not join, as a future improvement we could allow the team leader to select who can participate*
CDF-36 Starting Games	Test-13 Tournament is Started	PASS	
	Test-14 Games are Split in Phases	PASS	
	Test-15 Users can Play Games	PASS	
CDF-38 Leave Game and Game End	Test-16 Return to Tournament App	PASS	

User Story	Test	Status	Comment
CDF-39 View Game Stream	Test-17 Join a Game Streaming	PASS	
CDF-40 Notifications of Game Stream Updates	Test-18 Receive Game Streaming Updates	PASS	
CDF-31 Load Balancing	Test-19 Load Balancing	PASS	
	Test-20 Upload a Class	PASS	
	Test-21 Choose a Class	PASS	
	Test-22.1 Register a CD Server Instance	PASS	*No admin page so we need to manually send API requests. Admin page could be added as a future improvement.*
	Test-22.2 Update a CD Server Instance	PASS	*PUT instead of POST request is more appropriate when updating a resource*
	Test-22.3 Remove a CD Server Instance	PASS	*What happens to the ongoing matches hosted on this instance if it is removed? This operation is allowed only when there are no ongoing tournament matches hosted on the instance to be removed. As a future improvement we can handle also the case when this prerequisite is not satisfied (i.e. fault tolerance).*

User Story	Test	Status	Comment
CDF-43 Bots can Play	Test-23 Bots can Play a Game	PASS	
CDF-44 Bots can be Trained	Test-24 Bots can be Trained	PASS	*Right now the API is working as expected so the test is passed. As a future improvement we could suit the data returned depending on the data needed by the bots for training purposes. For example by adding filters.*

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