Easy access to Counter-Strike information

Final Report 02369[1][2]

Jarl Boyd Roest
s224556@dtu.dk

Jakob Skov Agergaard
s224570@.dtu.dk

Philip Muff Førrisdahl
s224566@dtu.dk

Anton Grandelag Helsgaun
s224268@dtu.dk

Esben Elnegaard
s224555@dtu.dk

1 INTRODUCTION

This project is an e-sport application that we've chosen to call "Counter-Strike Companion". Counter-Strike Companion provides quick information about the customers favorite events and teams. It's a platform for gamers and e-sports enthusiasts who loves the game Counter-Strike and want to know all of what is happening in the Counter-Strike community.
But this project has also been a learning process for this group. We've made this report to give an insight in our process and what we've learned.
Firstly, when we had to select a project we had to work on for the next 3+ months, we let our passion for gaming, and our desire to enhance the experience for every e-sports enthusiast, choose the project for us. The traditional platforms often left the user confused and with too many options as to how to get information. Therefore, we saw the need for a new platform, that could provide quick general information about Counter-Strike.
The app is far from ready to launch and the full experience will therefore not be possible to provide. But our goal is to show an alpha version of Counter-Strike Companion, consisting of a few user-stories.

1.1 Challenges

During the process of making the app, we have met multiple challenges. One of the first challenges was to find a working API that had the information we wanted. We did a lot of research before finding one and it is still not completely pain free. As of now we are not all the way through with a solution to this problem.
One of the bigger general problems was the difference in expectations between the course 62550 and this course. A lot of tasks seemed misplaced. This is particularly valid for the business and requirements parts since course 62550 made it so there was no need for these analysis.
We have dealt with this the best we could and adapted along the way to not steer out of scope for either course. We think we have found a golden mean.

1.2 Changes

The project is divided into 3 iterations, and in all of them, there have been a lot of changes. We had to keep an open mind to everything and not be fully locked in to any ideas since we haven't used a lot of the tools before. Therefore the major changes we've made are not necessarily fully our choice, but something forced onto us in order to make Counter-Strike Companion a good app. Often the changes consisted of redoing something we already had done because a better or more correct way was discovered.

There is a full overview of our Change log in the appendix. This is not documenting the changes of how we are doing stuff only the what and why.

2 CUSTOMER NEEDS

The primary users of the app, would be users of other e-sports platforms such as HLTV and Liquipedia. This could be a passionate gamer, e-sports enthusiast, content creator, or community leader who seeks a comprehensive, user-friendly platform to engage with the gaming world.
If we would choose to monetize the app, the business model would be, that the users would pay for their use of the app through adds throughout the application. This would be in cooperation with specific ad-companies, and other e-sports organisations that are interested in working with us and that we are interested in working with.
We have chosen to divide our users into several distinct user groups, to further identify their different needs and wants.

(1) The Enthusiast: This user plays counter strike regularly and follows the news, roster changes, tournaments and other information about the game.
(2) The Gambler: The user who doesnt necessarily care for CS as a game, but is still financially invested in the form of gambling websites and skin trading.
(3) The Novice: Doesn't play the game but still follows their favourite team(s) in larger tournaments like Majors.
(4) Community/Event organisations,
such as Blast.tv, ESL, FaceIt etc. Valve, the owner of CS and sponsor of Major tournaments
(5) E-sport Teams, such as Faze-clan, Liquid, G2, Vitality etc.
(6) Professional players
(7) Valve, as the creator of the game

To get a better overview of the most important stakeholders, and their wants, we have made a stakeholder matrix with interest and impact as parameters. This BPMN diagram of use case 2 can be seen in bigger version in the appendix.

Stakeholder	Role	Interest	Impact
Users	Users will use the platform and it's features, that provides them with their individual value.	**High**. A customizable and intuitive user experience, that is better than the competitors	**Medium**. We will adapt the platform to the users needs and wants, through user-interaction, -observation and -tests.
Developers	Provide the users, with the desired product.	**High**. To profit of our application by providing value to our users and out-compete	**High**. As developers, but also product owners, we have the power to steer the project in the direction we would like.
Event organization	Organize events and make sure enthusiasts come to the event	**Medium**. They need people to participate in events. Platform could be direct marketing	Low. They have no way to affect the development.
Valve	Has no direct desire about the platform, but wants to protect the overall health of the game and game.	**Low**. Only cares about indirect impact on overall playerbase of CS, and not so much the specific third party data provider.	Unless Valve removes CS from the market, which would not be in their interest, they will likely have **very low impact**.
E-sport teams	Plays tournaments and are finansially invested in the game and their fan base.	**medium**. They want their fan base to know when they are playing and how it is going. Also want attendence to live matches.	**Low**. They have no way to affect the development.

Figure 1: Stakeholder diagram

2.1 Use case diagram

Our use-case diagram shows how our different intended users, gain value from interacting with the different functionalities of our app.

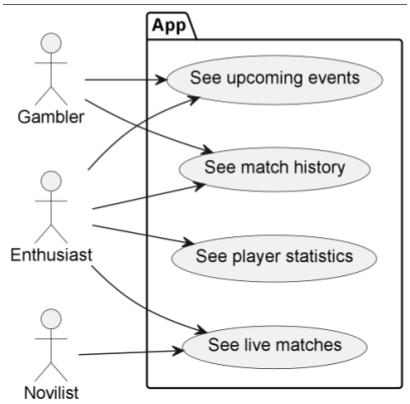

Figure 2: Use-case Diagram

2.2 User requirements

We wrote 4 user stories at the start of our development process using the AQUSA framework to identify the main features that would satisfy our different types of users. These were

US1: As an enthusiast, I want to look at the match result of my favourite team, so that I can see if they won.

US2: As an enthusiast, I want to look at the statistics of my favourite team, so that I can see how they are doing.

US3: As a member of the community, I want to be able to vote on which team I think will win, so that I can experience the thrill of gambling without risking money.

US4: As a Novice I want to be able to see how my favourite player has been doing in a tournament, so that I can see if he has played well.

Additionally we wanted to ensure that all of our user stories followed the INVEST framework. US1 follows the INVEST framework accordingly

- **Independent** The user story is independent in that it doesn't rely on other user stories.
- **Negotiable** The user story is negotiable as it doesn't go into specific detail about how it's done, and what a favourite team is. During the project we have had discussion about the term 'favourite team' and whether it meant that it should be possible to select a favourite team in the app, or if it just meant that the user would go to the match results of the team that they are a fan of.
- **Valuable** In being a sports app, the match result of a team is highly valuable to be able to see.
- **Estimable** For all the user stories it has been difficult to estimate due to our lack of experience, but apart from that, the user stories are entirely estimable
- **Small** The user story is small in the sense that it focuses on a single functionality, however it does need other types of information such as scores, team names and dates.
- **Testable** The user story is testable by seeing if the recent results of a team is being displayed.

AQUSA-core works an acceptance test for user stories. We have run all of our user stories through AQUSA-core to get quick feedback on their form: Here are the outputs of the program:

```
==================================================
                    AQUSA-Core
    Requirements Engineering Lab, Utrecht University
          Fabiano Dalpiaz, Garm Lucassen
==================================================
```

Figure 3: Entire output from AQUSA-core

As there are no further output lines from AQUSA, this means the program doesn't point to any faults in our user stories - an indicator of success.

Our acceptance tests for the user stories would be as follows:

AT1: Given i am on my favourite teams page, when i look at their recent matches then i should see the result of these.

AT2: Given i am on a teams page when i look for their statistics then i should see them

AT3: Given i navigated to a live or upcoming match when i put in my vote for who is going to win then i get visual confirmation that i voted and get the result of the match and my voting showed.

AT4: Given i am on my favourite players page when i look at the statistics then i see the correct and relevant statistics.

The development goals were set based on the User-stories to emphasize a Behavior-driven-development process. Though we didn't come through with the actual execution of this process;
Cucumber is supposed to be behavior-driven but we don't have a lot of stuff in our app that is testable with cucumber. It is meant to implement the acceptance tests, but when everything is hung up on visual compose-elements it made it difficult. We have tried very hard to get the composeTestRule to work together with cucumber but without luck.

There is not really a lot of logic in our app yet. So far it is almost just a front for an API. We are pulling and displaying chosen values. We have made a cucumber test that tests whether the average player age for a team is correctly calculated. This would be a part of US2. The result of the cucumber test is shown in the appendix.

3 PROJECT GOALS

3.1 Project vision

Counter-strike Companion will provide the customer with an easy way to see and navigate through relevant information, making the customer experience seamless.
We have chosen to address the customer problem of not being able to find relevant information easily in the current most popular e-sports apps. These apps are primarily desirable and useful, but they miss the key factor in being usable. We aim to fulfill all three of these key product attributes with our app.
Therefore, Counter-strike Companion will provide extra information for the team that a user is following, giving a stronger personalization for each user.

3.2 MosCoW analysis

In addition to defining user-goals, based off our user-stories, we also made a prioritized list of potential features that should or should not be implemented. This took place at the start of iteration 1 and produced the following MosCoW.

Must have	List of popular/upcoming sports events
	Detailed information about an event
	Live scores for ongoing matches
	See upcoming matches
	See and make predictions about upcoming results
	Simple server/back-end storing predictions
Should have	Player and team profiles
	News page (From HLTV)
	Player profile (Username/password)
	Notifications
	Customizable home/personal page
Could have	Search functionality (Depending on difficulty)
	Basic security (in terms of profile; No plaintext/Oauth/ect.)
	Backup of customized home page based on profile
	Graphs of player and team statistics
Won't have	Betting
	Monetization

Figure 4: MoSCoW

Even though this MoSCoW was made at the start of iteration 1, it has not changed much throughout the process, which indicates that everybody in the group were aligned with the project goals, from the start.

3.3 Design goals and changes

The visual-design goal of the project has shifted considerably during the iterations. Below is showcased how the prototype has changed from the first iteration to the second iteration. This change was made based on feedback from the first iteration by the lectures from Shape.

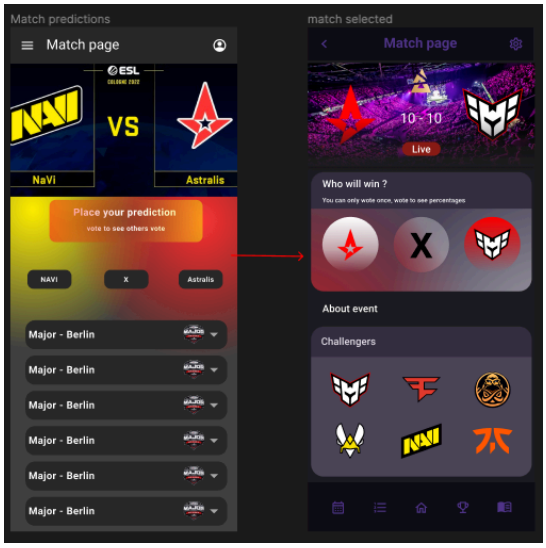

Figure 5: Change in appearance in second iteration

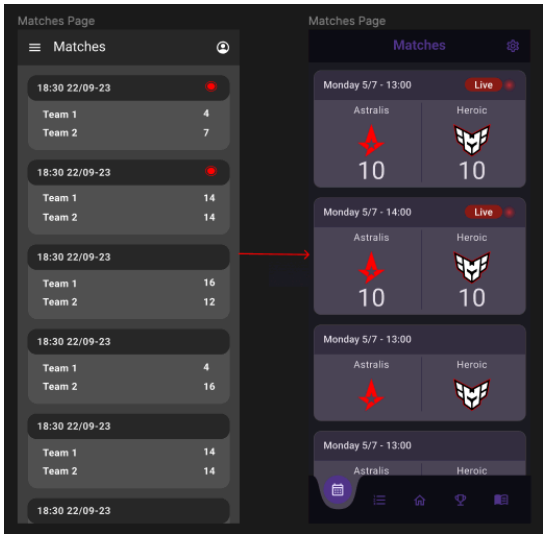

Figure 6: Change in appearance in second iteration

3.4 Use Cases

The following use cases are based on the identified user goals, from the user stories. We end up with the core features: being able to access information for counter-strike matches and players, in the past and present.
Even though the use-cases are similar, they all give value to the different kinds of users we have identified earlier.

UC1	**See match result**
Primary actor	Gambler
Stakeholders and interests	Gambler wants to see if he won his bet Stakeholder: Gambling sites; wants to revenue based on live matches bets
Preconditions	User has app installed and is at homepage
Post conditions	User has seen whether or not he has won his bet
Main success scenario	1. User at home page 2. User directs to desired team 3. User directs to 'Match results' for given team 4. User sees match results for the team
Extensions	1. User is not at the home page 2. User finds team on events page 3. User finds desired match and match result

Table 1: Use case 1

UC2	**See player statistics**
Primary actor	Enthusiast
Stakeholders and interests	Primary actor Stakeholder: Professionel players wants their fans to see their player profile to see their statistics
Preconditions	User has app installed and is at home page
Post conditions	User has seen player statistics
Main success scenario	1. User is at home page 2. User finds 'Match results' page 3. User directs to specific team 4. User finds specific player on team 5. User directs to player page
Extensions	1. User is not at the home page 2. User finds team on events page 3. User directs to player page

Table 2: Use case 2

UC3	**See upcoming events**
Primary actor	Enthusiast
Stakeholders and interests	Enthusiast wants to see upcoming events Stakeholder: e-sports organizations wants users to access their event information
Preconditions	User has app installed and is at homepage
Post conditions	User has seen desired events
Main success scenario	1. User findes given event displayed on home screen, which navigates to the 'events' page 2. User directs to event, to check which teams are going to play
Extensions	1. User is not at home page 2. User findes 'upcoming events' in the bottom navigation bar 3. User findes desired information about the events

Table 3: Use case 3

UC4	**See live matches**
Primary actor	Novice
Stakeholders and interests	Novice wants to see if his favorite team is playing E-sport events; wants their match displayed
Preconditions	User has app installed and is at homepage
Post conditions	User has seen desired live match
Main success scenario	1. User sees 'live matches' displayed on home screen 2. User directs to 'live matches' 3. User scrolls list and finds desired match
Extensions	1. User is not at the home page 2. User presses 'live matches' button in the bottom navigation bar 3. User finds desired match

Table 4: Use case 4

3.5 Measures of Success

To know whether the customer got their desired benefits, we have identified the core functionality of the app through user-stories and use-cases as shown earlier, so given that we implement those features, we satisfy our expected primary users. This we will be monitoring through the acceptance tests mentioned earlier. If they pass we should successfully have implemented the functionality they desire.
We also intend to do further testing, and especially user-testing with different types of test-subjects, that simulate our expected users. This includes

test-subjects that have a varied interest and understanding of the professional Counter-strike scene, from novices to enthusiasts.
If we reach the point where we are able to launch our app in the android app-store, a valuable measurement would be an Net promoter score, to test our users willingness to recommend our app. This will give us an indication of the overall costumer satisfaction.

4 SYSTEM ARCHITECTURE

4.1 System Overview

This section seeks to give a brief overview of the system architecture, without going into a lot of specifics.

Overall Architecture

The overall architecture of the app follows the MVVM architectural pattern, which is the obvious choice when making apps in Jetpack compose, especially because our app does not have enough data processing and business logic to warrant a separate controller/BL layer.

As such, we have chosen to divide our package structure to fit the MVVM pattern, but we have coupled each screen with its own ViewModel, in the UI package instead of packaging all the viewModels together, which results in a higher separation of concerns and looser coupling at the cost of some more lines of code.

An overview of the structure of our project in modules and packages can be seen in the following package diagram:

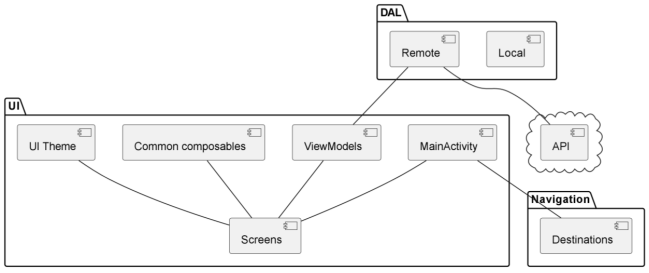

Figure 7: Package diagram of the project

User interaction

We also made a state-chart diagram to display, in general terms, our App's different states.

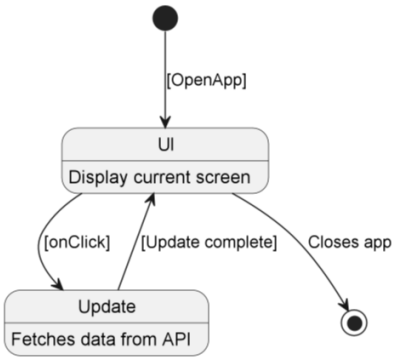

Figure 8: Statechart of app state. Not that the UI still displays while the app fetches more data

In the appendix (for readability), a BPMN diagram shows a more detailed view of how a user may interact with the system.

External interactions

The app mainly functions as a front end which pulls, analyzes and shows data from the API (meaning that the we, in regards to the API, follow the repository pattern). One of the main challenges is the limit rate of API calls we can make per second; One solution to this would be create a server that fetches things from the API, and then the different instances of the app fetches from this server (This is also great for protecting the API key) - however, this is beyond the scope of the project.

The system in the current form interacts with one external API in a client-server architecture, as well as with the Android operating system to a limited degree, though operating system interactions are handled by Jetpack Compose and high level Kotlin functions, and not something we do explicitly, with the exception of asking the system for permissions to access internet.
In future iterations, it is planned that the app will also interact with an external server that we host, which will allow users to make predictions on which team will win an upcoming match, however, this is yet to be implemented.

4.2 Analysis and Design

To solve the previously mentioned issue of slow loading times, because of the rate limit from the API, we plan to implement the caching-aside pattern, where each player image would be stored to a package in the data layer, and the rest of the player data would be stored in a local Room database. This will be implemented within the next few sprints, and is therefore not included in the iteration included in this report, however it would be implemented with a cache-aside pattern, as shown in the following BPMN diagram:

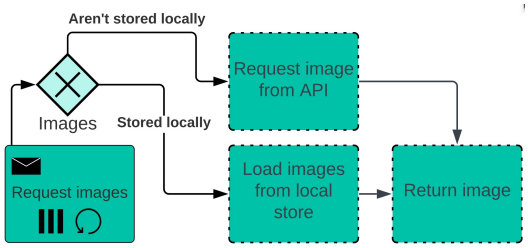

Figure 9: Planned implementation of cache-aside

As seen in the diagram, requests may be made in parallel - however, this means that the app can very easily exceed the max number of requests allowed per second. To prevent this, we have implemented a simple rate limiter/limiting pattern which makes sure that there is sufficient time between each request, 200ms, while they can still run asynchronously, meaning that as soon as 200ms has passed, the next request can be sent, regardless of whether the previous request has received a reply.

Other software considerations

Along with the cache-aside pattern, we also use singletons for both our instance of Gson (for (de)serializing JSON text) and for our OkHTTP client as these greatly save on system resources and memory usage as compared to create a new instance for every request. To make testing easier and increase performance, we have implemented state hoisting throughout our composables, and done our best to separate concerns. Due to us needing to make a lot of request to the server, we only use a single instance of the

client to access the server instead of making new ones each time. This in turn makes the app run faster. We also only pass the minimal information between screens, like a unique player ID or team ID, as opposed to entire objects. This greatly decreases the likelihood of inconsistent data between screens/viewmodels by having only a single source of truth, which holds all the data; If caching-aside is implemented properly this will not really mess with the SSoT.
The state hoisting pattern is a big part of Kotlin. Jetpack compose is heavily reliant on state. We have used the MutableState in the viewmodels and this is a simple way of state hoisting. We keep the state at a higher level and pass it down to the composables that need it. This is also the case with our common composables. It is a common ancestor that holds the state.
As mentioned earlier, our API file has handled everything regarding getting information from the API, processing it and making it usable and accessible in other parts of the system, such as in the view models.

5 PROJECT MANAGEMENT

As mentioned previously we have not had too much experience with using the agile development process in the real world so this is our first attempt therefore also a decent amount of trial and error. We decided more specifically on trying the scrum process.
Doing it as if we were a company was difficult and we settled on a mid-road where we would act and assume when needed. For example we didn't specifically dedicate the roles, but took them upon us as needed and delegated the tasks depending on the task. We have not had a single product owner, scrum master or so on, but definitely had a person acting as one during the artifacts.
Our iteration process would go as follows:
Sprint planning
We would get some requirements for the iteration and these we would digest and atomize into tasks. These task would then fill our backlog. We would like to have a proper backlog and choose the tasks based on the time available, but since all the tasks in our backlog should be part of the present iteration we didn't really get that experience. Worth noting is that we did try doing planning poker on the tasks produces from iteration 3's requirements(seen in appendix). This we did despite we not have a clue how many story points we would be able to complete in one iteration.
Task Breakout
After the tasks had been assigned the iteration we would all pick a task. This was based on peoples forces but also on the critical path. We never succeeded in utilizing Jira to its fullest extend, more on this later, but we tried making a gannt-diagram in iteration 3. The diagram made it clear to us that the most important task in the beginning of that iteration was to get the API implemented.
Scrum meeting
During the iteration we held Scrum Meetings. For us this was the time where we took on new tasks, told the team how it went with the task/tasks we were doing and reached out for help if we needed it.
In hindsight we might have used these meetings a little to much to reflect on our work and status which took away focus from the meetings and the meaning of the later sprint review. We think this is in part caused by the time span of the iterations being too long compared to our learning curve.
Developing
During the iteration we had great experiences and utilized some specific coding practises. We did a lot of pair programming. This was highly effective for us since it made us catch mistakes before they caused hours of debugging.
Another thing we did was common review/introduction/tutorial. This was especially helpful when filling in the rest of the team on how an implementation was working and how it should be used by the rest of the team in the future. An example of this is the API. Anton and Esben spent time on getting to know it and setting it up with the most important data classes we needed. When they were ready they could introduce the rest of the team to the use of this without the whole team having to dive as deep as them. This approach of an "expert" teaching the rest also applied to other things such as cucumber, coroutines etc.
Sprint review and retrospective
We always finished the iterations with these two events. We kept it short and the documentation can be seen in the appendix under Sprint Retrospectives and Sprint Reviews.
We used the sprint review to talk about how the product was coming along. The retrospective we used to reflect on our own workflow, the procedure and most important for us how well we were doing the agile development. The end-goal was to improve our workflow and the process. We have used a simple and quick approach of talking and writing thing we should start doing, stop doing and continue doing.
In general we have down-prioritized doing the whole thing by the book. We wanted to get the full experience but there were too many obstacles holding us back. Therefore we decided to loosen the rules a bit and adapt the scrum process to better fit our needs and style. Overall this was a good decision.
The project has been organized in three iterations that is considered the projects major events. Every iteration and tasks that we had in the given iteration is documented in Jira. The first iteration: 14/9/2023 - 23/9/2023. In this iteration we had to choose what project we wanted to work with. The second iteration: 2/10/2023 - 29/10/2023. We had to make a high fidelity prototype in Figma and use Kotlin for the first time to make basic navigation in the app. The third iteration: 6/11/2023 - 30/11/2023. As the final iteration this had to deliver an app that was able to complete some user-stories from start to finish.

5.1 Project organization

We decided to use Jira as the management tool for the project. There is a lot of layers and a lot of complicated functionality therefore it was quite a mouthful to get started. We have and are still most definitely not using it the best way but it has still been a help and been an experience using an organization tool like this.

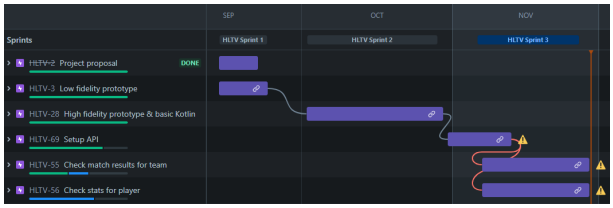

Figure 10: Project iteration overview

In the beginning we dedicated an epic to be a deliverable but figure out the better way to go was using the user stories as epics. Though we later learned that there is an issue type called a story. This was the general way we worked in Jira: Trial and error, or maybe Trial and optimization.
All epics then had relevant tasks assigned to it and the tasks could be assigned. Example of tasks for an epic:

The prioritization of the tasks in the third iteration was as mentioned done by planning poker. We used an online site for this where we could import our issues from Jira and it would go through them and assign a story point value based on the average of our suggestions. This was a good way to know which tasks would take longer, but what we afterwards feel was missing was the time-pressing aspect of the tasks. Which is why we as mentioned also tried figuring out which tasks/epics were linked and blocked by each other. Though visualising this correctly using the built in gannt-diagram feature in Jira was challenging.

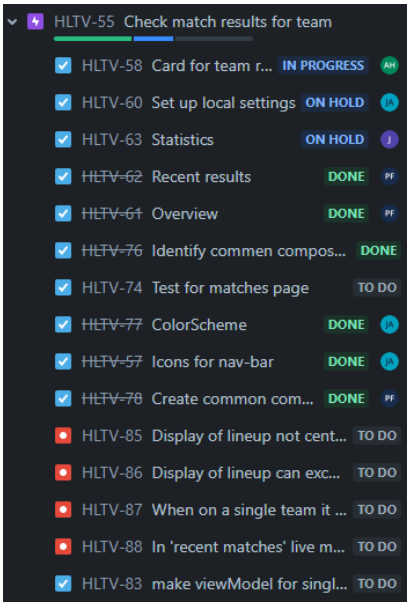

Figure 11: Example tasks for an epic

We also tried using confluence which can be in correspondence with Jira for all the documents and artifacts we have created through the project. We did not link this up properly but everyone on the team has access at all times and can for example create a new retrospective document straight from a template. This made sure we had all the documents together in one place. No asking around what was where.

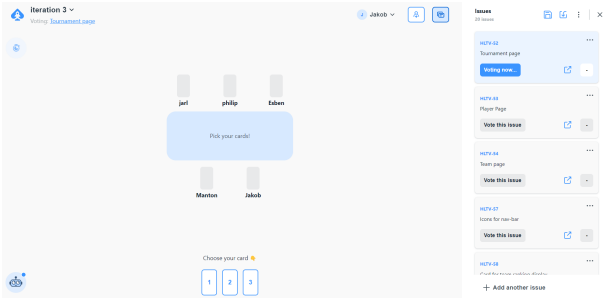

Figure 12: The planning poker session

Results of planning

The planning has been a helpful tool but as the agile manifesto states "Responding to change over following a plan". Therefore we have, as told, used a change-log where we have logged the major changes and pivots in the project. This change-log is visible in the appendix under Change log.

5.2 Team Coordination

As a whole team we have been meeting Mondays and Thursdays to work for 2-3 hours on the project. It would be during this time we would participate in the scrum activities.

When a deadline was closing in we would meet online besides the Mondays and Thursdays. Usually we would work between 4 and 6 hours and the weekly meetings would also be prolonged.

For the last week of the last iteration we have been meeting 5 days in sessions of 4 to 6 hours. The team has been using Facebook Messenger for casual communication and planning meeting times and so fourth. Furthermore Discord for the online meetings and Jira for the official communication of tasks, roles, status etc. Discord is a good way to make time for some work even under time-limited circumstances.It is also a great way to share your screen for pair programming.

6 TEAM

Backgrounds

We are five guys who likes gaming and studies IT and economics and Software on DTU. We all had some sort of interest in IT before we started studying, but have never tried to code on a high level before DTU. Since then we've learned to use the coding language Java and will be using the JVM-based Kotlin.

In terms of prior work experience we have all made a game called "Robo-Rally". This was in the second semester were we developed the game on top of a consigned skeleton. This was coded in java. Other than that it's safe to say that we're rather new in the environment.

Non of us have made an Application before this project or used Kotlin. As mentioned we are familiar with java which made the Kotlin language easier to learn and use.

The IDE we used was familiar since it was released by the same manufacturer that makes IntelliJ, which we have used before for java.

The development tools that we were meant to use are completely new to us and we have only been introduced to these and have never actually tried them in action. This extends from the project management aspect of the project all the way down to the test-processes

Roles

As we all were new to Kotlin and app development there was no steady pillar to lean on. Anton is the most experienced with development in general and the team was able to seek his advice on the procedure and execution of a lot of things but especially the coding do's and don't's.

Jakob were taking on the role as scrum master and made sure the team were following the agile development process we set out to do and controlling the project on Jira. Furthermore he contributed to the coding, testing of features and setup.

Esben similarly took on the role as scrum master and likewise contributed to the coding and implementation of the API.

Philip has been a work-machine and taking on a lot of tasks from prototyping and designing in Figma to coding logic and composables.

Jarl has been contributing with work on the prototypes and design in Figma like Philip. He has been creating composables in Kotlin and writing report.

In general the role distribution was very democratic and we are a group that is used to working together. We haven't needed for anyone to be above the others. Below a percentage contribution by each team member is shown:

Participant	Contribution (Total 1.00)
Philip	0.225
Jakob	0.225
Anton	0.25
Jarl	0.15
Esben	0.15

Table 5: Project Contribution

7 CONSTRAINS AND RISKS

Were there any social, ethical, policy, or legal constraints?

During the current state of our app, we haven't met any social, ethical, policy or legal constraints, however, in the following 3 week course, we will

be implementing US3, which focuses on being able to vote on a team to win, without risking money.
Although we don't plan on giving the user a reward for picking correctly, it may still be considered a form of gambling, which would introduce social, ethical, policy and legal constraints.
This is also a way of promoting gambling, which could be risky, as we don't have an age-rule on our app.
Did you have access to the data, services, and resources you needed?
In terms of the data for the project we were not satisfied. There are a very limited amounts of API that even gives some of the data we want. We think we found the best one but even this one was not good. Very weird data delivery structure and a lot of missing information on the keys in the JSON's we were getting. We also realized that it would be difficult to gather information about future events and games from the API. At this point we still don't know if it will be possible for us. This made us pivot the user stories to solely focus on live or past information. The API is able to gather information about other sports as well, which made a lot of the available API calls unusable as they didn't make sense for our sport

8 FINAL STATUS

We have made a video that demonstrates the app as it is right now. Link to Demonstration.
We have successfully run and completed one single cucumber test. We would have liked this number to be much higher but we had, as mentioned earlier, a lot of problems with it.
We know it would be a good idea also to write some unit tests but we haven't felt like we have developed a lot of logic to test yet.
We have run several 100 manual test cases by running the app on our phones and emulators. We didn't feel like the best use of our time was documenting these tests and therefore there is no evidence of these.
Altogether the team has written about 2500 lines of Kotlin code. There is a lot of hard work ahead of us in the 3-week period and we look forward to reap the rewards of all the hard we already have done setting up the basics of the app.

9 REFLECTION

We've learned a lot about how to make an application not only development wise but also how to design one. We've learned how easily problems occur and how the scrum method will help control and keep track of the problems, and mitigate their impact. Even though we, as first timers, has spent a lot of time on the scrum-management and it may take some more practice to get fully autonomous it has been a great tool. We are sure it has helped us manage a project of this size.
The adoption of Scrum and Jira also proved to be beneficial for the general documentation. The Scrum framework facilitated iterative development and adaptability to changing requirements, enhancing overall project flexibility. Jira's capabilities streamlined our backlog management, making it easier to prioritize and allocate tasks. Once we got the hang of it.
From what we've figured out, we would consider regular sprint retrospectives as a best SCRUM method. These sessions provided a forum for the team to reflect on successes and challenges, fostering continuous improvement. Additionally, maintaining a well organized backlog in Jira allowed for efficient sprint planning and execution. Another tool we are big supporters of is planning poker as it provides a common expectation for the tasks.
But as in any other project, everything didn't go according to plan. At this point we were very happy to have used SCRUM as a method and done our retrospectives. With retrospectives we could reflect over what to do better and that way figure out what wasn't working. Based on our retrospectives we haven't had any major problem that we should stop doing. The majority of the problems we found was things that we needed to start doing. So what we were doing was not wrong, we just did too little. After our retrospective we were all on the same page, and could start a new iteration on a clean slate with same expectation in the group. Our Retrospectives can be seen in the appendix G.
For the final sprint, we plan to allocate additional time for testing and debugging to ensure a more robust final product. We will prioritize the completion of critical features and focus on the usability in terms of time spend fetching data from the API.
We had a few features that didn't make the app as it is now. Some of those features were the burger menu, date filter and the old color scheme (can be seen in figure 2 - 4). The issue with the burger menu and color scheme was mainly design concerns, functionality wise we could make it work if we wanted. We didn't like how the Burger menu was used and it seemed like it just added more confusion than did any good. Therefor we preferred the bottom menu as it's simpler and is the most popular form of menu. The date filter was not included because we had troubles seeing the usage.

Recommendations

There are many thing we would like to do differently but taken into consideration that we were very time limited and couldn't meet every day, we think our job done is very satisfying. On that note it's hard to imagine what we could have done differently other than what we've already documented in our retrospective and sprint reviews. But if we had more time and only had that project to focus on, then have a daily stand-up meeting and try to role distribute more firmly.
If other teams would like to have had the same positive experience that we did. Our advice for other teams would first of all be regular usage of sprint retrospective. By using sprint retrospective continuously the team has a better chance to improve, since the group articulate on what can be improved and what is already going well. It is a very effective way for a group to provide constructive feedback to itself. Other than that, we recommend using sprint review as it gives a nice overview of what the current state of the app is in. And if possible then familiarization with the language that need to be used. Lastly we've had a very pleasant experience using Jira as platform for work organization, backlog and documentation. Though it may have too many, too advanced features for small teams like ours.

10 DEMONSTRATION

10.1 Video

Youtube video with walkthrough

10.2 GitHub repository

Github repository

REFERENCES

[1] Group 13. 2023. Project proposal. (October 2023). This has been a starting point for this report.
[2] Group 13. 2023. Status report. (October 2023). This has been a starting point for this report.

11 APPENDIX

A SCRUM BOARD

https://jakob-sa.atlassian.net/jira/software/projects/HLTV/boards/3
If access is needed contact Jakob at s224570@dtu.dk

B PROTOTYPE

Link to the prototype in Figma

C CHANGE LOG

	Date	Description	Motivation	Implications	Direct tasks (if any)
1	2. okt. 2023	Change the display options for events. There will no longer be complicated filters. (espesially by date)	Very few will use the feature and it is not really appropriate for the events	Plans have to change a bit. Figma design change. Caught it pretty early so nothing big	☑ Change Figma prototype
2	12. okt. 2023	The decision to not make a login screen and account creation	Out of scope for project and mostly unnessesary	Less work	
3	12. okt. 2023	Scratching the burger menu	We wont get funcionality enough to fill it out with different functions that are not in the bottom-bar menu	Less work	
4	6. nov. 2023	Scratching 'math result page'. Instead it will be integrated in the specific team page	Simpler navigationen bar and less pages	Change in code structure.	
5	26. nov. 2023	Having to change some of our thought out flow and components in the application due to lack of API-support. Maybe we will look at this at a later time and mock some data or implicate another API.	It was simply not possible to make what we wanted with the API we have been using. We need to deliver something for iteration 3 so we are pivoting a tiny bit.	Most of the matches, tournament etc. in our app that are to be played in the future are not available to us through the api.	☐ Remove all upcoming things in app ☐ Brainstorm on good alternatives to implementing the (now) missing features. ☐ Implement alternative

D STAKEHOLDER MATRIX

Stakeholder	Role	Interest	Impact
Users	Users will use the platform and it's features, that provides them with their individual value.	**High**. A customizable and intuitive user experience, that is better than the competitors	**Medium**. We will adapt the platform to the users needs and wants, through user-interaction, -observation and -tests.
Developers	Provide the users, with the desired product.	**High**. To profit of our application by providing value to our users and out-compete	**High**. As developers, but also product owners, we have the power to steer the project in the direction we would like.
Event organization	Organize events and make sure enthusiasts come to the event	**Medium**. They need people to participate in events. Platform could be direct marketing	Low. They have no way to affect the development.
Valve	Has no direct desire about the platform, but wants to protect the overall health of the game and game.	**Low**. Only cares about indirect impact on overall playerbase of CS, and not so much the specific third party data provider.	Unless Valve removes CS from the market, which would not be in their interest, they will likely have **very low impact**.
E-sport teams	Plays tournaments and are finansially invested in the game and their fan base.	**medium**. They want their fan base to know when they are playing and how it is going. Also want attendence to live matches.	**Low**. They have no way to affect the development.

Figure 13: Stakeholder analysis matrix showing the role, interest and impact of each stakeholder

E CUCUMBER RESULT

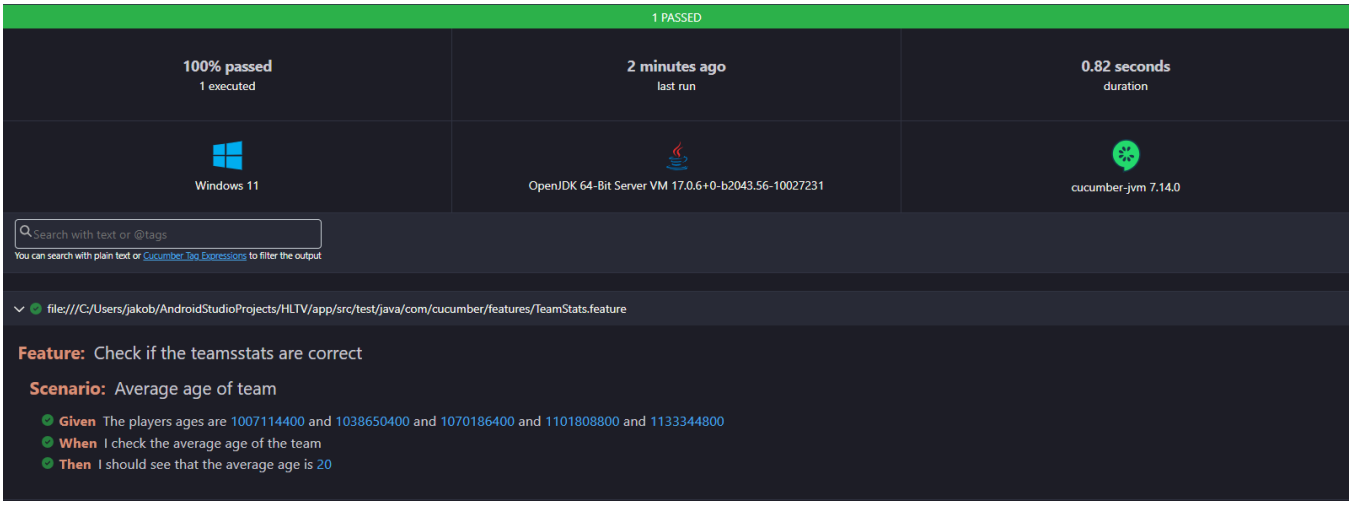

Figure 14: Result from cucumber showing input of 5 player ages (given in seconds, as that is what we receive from the API), that returns an average age in years

F BPMN DIAGRAM OF USE CASE 2

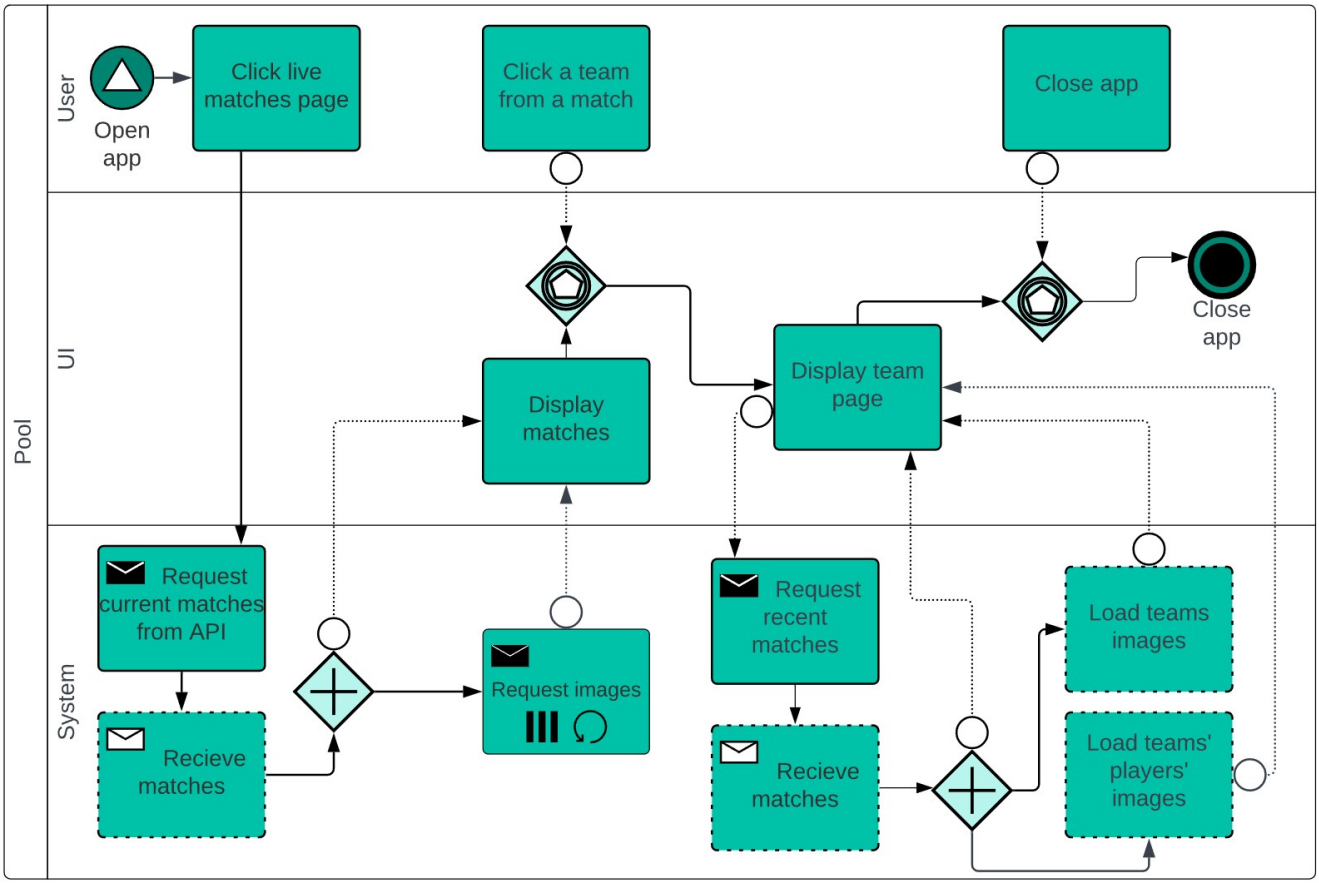

Figure 15: BPMN of the majority of UC2. Notice that the last part, where the user navigates to a single player is not included for simplicity. The image loading process is shown in higher detail in another diagram.

G SPRINT RETROSPECTIVES

Retrospective 1

JA Owned by Jakob Agergaard, created with a template •••
Last updated: just a moment ago • 1 min read

Date	Sep 25, 2023
Participants	@jarl @Anton Grandelag Helsgaun @Jakob Agergaard @Philip Muff Førrisdahl

Start doing	Stop doing	Keep doing
Use the task-system more		Scrum and documentation
Better progress indication		Attend lectures
Build up a product backlog		
More codelabs		

Retrospective 2

JA Owned by Jakob Agergaard, created with a template •••
Last updated: just a moment ago • 1 min read

Date	Nov 2, 2023
Participants	@Philip Muff Førrisdahl @Jakob Agergaard @Anton Grandelag Helsgaun

Start doing	Stop doing	Keep doing
More streamlined workload distribution	Placeholders in code	Use Jira for task and deligation
More artifacts and documentation		Meetings in person
Gant diagram feature in Jira		
Planning poker for task estimation		
Planning meetings in time so these can be used for sparring and decision making		

H SPRINT REVIEWS

Review 2023-10-02

Owned by Jakob Agergaard, created with a template •••
Last updated: just a moment ago • 1 min read

In general product is on schedule. Sprint 1 has turned out as planned.

Improvements
• Events description (maybe an entire page for each event) • Bottom menu bar • Team logo i match history • Predictions on match page • Settling on colour scheme • Details for more engagement • Variety in pages

Review 2023-10-30

Owned by Jakob Agergaard, created with a template •••
Last updated: just a moment ago • 1 min read

Pursuing	Not pursuing
• Tests -Junit -Test of user stories CUCUMBER og AQUSA -Test and documentation of API working • MVVM • API working? • Design patterns • Documentation • Proper color and typo scheme setup	• Placeholders in code • Laggy app •