Maturarbeit
Documentation

Game

2022

Version: 1.0
Date: 2022-07-27 16:51:23+02:00

Team: Marc Honegger
Elia Schürpf

Begleitperson: Martin Hunziker

Gymnasium
KZO - Kantonsschule Zürcher Oberland

Extract

This will be the best extract that ever existed. By Eli and Marc:

Inhaltsverzeichnis

Teil I

Ziel

Kapitel 1

Vision

JassTracker is a web app which allows tracking and analysis of the popular Swiss card game Jass. There are many forms of playing, but the one we're focusing on is called "Coiffeur". The following paragraphs expect the reader to know the basic rules and concepts of a "Coiffeur Jass".

What It Does

The goal of the JassTracker is to allow the players to focus on the game without having to worry about tracking points or whose turn it is. Instead of having to write down the scores using pen and paper, you can simply input them into the JassTracker digitally. The website automagically tracks whose turn it is and analyses your scores for you. You can also gain exhaustive insight into your play style by looking at personal or group statistics, such as average score by player or historic averages. The system is also very flexible and can be configured to your liking. For example, once you have configured a Jass table, you can easily start a new game with the same players.

How It Works

The first step is to create a new table and enter all team members for the game. Then you start playing the game physically, just like you would normally. Once a round is over, you input the score for the scoring member and "Trumpf". After entering the points, a few things happen automatically:

- Player & team score update
- The upcoming starting person is shown
- Statistics refresh with the latest information

When the game has finished, highlights of the game (e.g. the best player or highest round score) are displayed.

Extensions

Some potential ideas to expand on:

- Configurable Jass game (e.g. Coiffeur with 8 instead of 10 options)
- Prediction of scores based on historic performance
- Current game trajectory (win probability by team)
- Share the scoreboard in real-time with other members in case you want to show the scoreboard on a different device as well
- Associate game to a personal account to track personal statistics across games
- In-depth statistics like play style, favorite “Trumpf”, best “partner” by win probability etc.
- Charts in game to show statistics

Kapitel 2

Anforderungen

2.1 KZO

2.2 Notwendige Features

Info:

2.3 Nice to have

2.3.1 Online Multiplayer

ID	NFR-7
Requirement	Browser Support
Trigger(s)	User wants to use a specific browser
Measure(s)	Last 2 major version of any browser with more than 1% usage (except IE 11) is supported
Testing	Manual

ID	NFR-8
Requirement	Mobile Device Support
Trigger(s)	User wants to access application using a phone or tablet
Measure(s)	The application must support being used on mobile devices
Testing	Manual

ID	NFR-18
Requirement	Unicode Support
Trigger(s)	User wants to include special characters or emojis in his username
Measure(s)	Support umlauts and emojis by using utf-8 and unicode
Testing	Manual

2.3.2 Monetarisierung

ID	NFR-1
Requirement	Reasonable response time for the rendering of new scores
Trigger(s)	User doesn't want to wait to see scores
Measure(s)	Rendering of the new scores must be doable within 1 second
Testing	Manual

ID	NFR-2
Requirement	Authentication Response Time
Trigger(s)	User wants to be able to login in a timely manner
Measure(s)	Authentication of a user at login must be doable within 1 second
Testing	Manual

ID	NFR-3
Requirement	New Game Creation Time
Trigger(s)	User wants to start the game as soon as possible
Measure(s)	Setting up a new game must be doable within 1 second
Testing	Manual

2.3.3 Graphiken

ID	NFR-4
Requirement	Keyboard-only usability
Trigger(s)	User has no touch-screen or pointing device available
Measure(s)	Application must be usable with only a keyboard
Testing	Manual

ID	NFR-5
Requirement	General Usability
Trigger(s)	Users want to be able to use the application without getting stuck
Measure(s)	Hallway testing shows eight people used the app without getting stuck
Testing	Manual

ID	NFR-6
Requirement	User Guidance
Trigger(s)	User wants to be able to access a Help-Center
Measure(s)	Hallway testing shows eight people find the Help-Center and say it's helpful
Testing	Manual

2.4 Skizzen

Disclaimer: These are just Mock-ups. Design can still change, and colors aren't fixed and are just as a visual aid for where containers should go.

Kapitel 3

Skizzen

Kapitel 4

Risiko

4.1 Hoch

1. Used technologies are not well knows by all team members (certain, critical)
   (a) Mitigation: Every team member should create a PoC in the used technologies to ensure basic understanding is present
   (b) Mitigated risk: Low

4.2 Mässig

1. Inaccurate estimations (likely and critical)
   (a) Mitigation strategies: apply cone of uncertainty, apply definition of ready to ensure planning quality
   (b) Mitigated risk: Low
2. Poor risk management (likely and critical)
   (a) Mitigation strategies: likelihood calculation, risk mitigation plans and monitoring of risks every planning
   (b) Mitigated risk: Low
3. Project reviewer's expectations are not aligned with project (possible and critical)
   (a) Mitigation strategies: obtain frequent approval and acknowledgement (naturally happens for us with review meetings)
   (b) Mitigated risk: None
4. Unexpected absence of team member (unlikely and catastrophic)
   (a) Mitigation strategies: Code changes need to be pushed on a daily basis, stories could at any point be taken over by another team member
   (b) Mitigated risk: Medium

4.3 Tief

1. Insufficient code quality (possible and marginal)
   (a) Mitigation strategies: code reviews, clear coding standards, apply definition of done
   (b) Mitigated risk: None
2. Lack of ownership (possible and marginal)
   (a) Mitigation strategies: setting clear responsibilities for roles
   (b) Mitigated risk: None
3. Losing sight of documentation tasks (possible and marginal)
   (a) Mitigation strategies: documentation strategy, documentation part of definition of done
   (b) Mitigated risk: Low
4. Failure of hardware like personal devices, OST GitLab, Jira, hosted environment (rare, catastrophic)
   (a) Mitigation strategies: Code changes need to be pushed on a daily basis
   (b) Mitigated risk: Low

Kapitel 5

Roadmap

Teil II

Produkt

Project name: JassTracker

Team Members

1. Pascal Honegger (pascal.honegger1@ost.ch)
2. Marcel Joss (marcel.joss@ost.ch)
3. David Kalchofner (david.kalchofner@ost.ch)
4. Jamie Maier (jamie.maier@ost.ch)

Availabilities

Time slot	Mon	Tue	Wed	Thu	Fri	Sat
08h00-09h00	(XO)	-	-	-	(XO)	-
09h00-10h00	(XO)	-	-	-	(XO)	XO
10h00-11h00	(XO)	-	-	-	(XO)	XO
11h00-12h00	(XO)	-	-	-	(XO)	XO
12h00-13h00	(XO)	(XR)	(XR)	(XO)	(XO)	XO
13h00-14h00	(XO)	XR	-	(XO)	(XO)	XO
14h00-15h00	(XO)	XR	-	(XO)	(XO)	XO
15h00-16h00	(XO)	-	-	(XO)	(XO)	XO
16h00-17h00	(XO)	-	-	(XO)	(XO)	XO
17h00-18h00	-	-	-	(XO)	(XO)	-
18h00-19h00	-	-	-	(XO)	(XO)	-

Project Idea

JassTracker is a web app which allows tracking and analysis of the popular Swiss card game "Jass". There are many forms of playing, but the one we're focusing on is called "Coiffeur". The two teams have two players each and need to keep track of what they've already played and which options are available to them. They also track whose turn it is, apply the correct multiplication to the score and sum it all up in the end. To work around this, a project team member is currently using a excel spreadsheet, but this solution provides limited functionality and has many drawbacks.

To make the scoring easier JassTracker allows players to easily track, analyze and sync games digitally. In a first step you will be able to create and arrange team members for a given game. Then you start playing the game physically and assign scored rounds to the correct member. During this phase you'll also be able to see live stats such as average score by player so far. After the game some highlights (e.g. the best player)

are highlighted. You can also gain exhaustive insight into your play style by looking at personal or group statistics such as average score by player or historic averages. To enable this, other physical members are able to associate their game to their personal account to track personal statistics across games.

Some potential ideas to expand on: configurable Jass game (e.g. Coiffeur with 8 instead of 10 options), prediction of scores based on past performance, current game trajectory (win probability by team).

Proposed Realization

We plan on implementing a web app using Vue.js as a frontend library. For styling we plan on using Bootstrap for basic styles. The server will be implemented in Kotlin using the Ktor framework. Persistent data is stored in a PostgreSQL database and accessed using jOOQ. Development will be done locally in IntelliJ IDEA, production deployments will be using Docker containers. CI / CD will be implemented using the OST GitLab.

Kapitel 6

Architektur

Kapitel 7

Feedback

Time tracking is done exclusively in Jira. Detailed reports with hours spent per issue, epic and sprint can be found there. The following charts focus on the overall progress to notice early trends, in case certain members invest significantly more or less time than expected

Who	**Logged Hours**	**Expected Hours**
Pascal	135h	120h
Marcel	115h	120h
David	111h	120h
Jamie	106h	120h
Team	468h	480h

Kapitel 8

Zukunft

Teil III

Schwierigkeiten

Project name: JassTracker

Unity

1. Pascal Honegger (pascal.honegger1@ost.ch)
2. Marcel Joss (marcel.joss@ost.ch)
3. David Kalchofner (david.kalchofner@ost.ch)
4. Jamie Maier (jamie.maier@ost.ch)

Github

Time slot	Mon	Tue	Wed	Thu	Fri	Sat
08h00-09h00	(XO)	-	-	-	(XO)	-
09h00-10h00	(XO)	-	-	-	(XO)	XO
10h00-11h00	(XO)	-	-	-	(XO)	XO
11h00-12h00	(XO)	-	-	-	(XO)	XO
12h00-13h00	(XO)	(XR)	(XR)	(XO)	(XO)	XO
13h00-14h00	(XO)	XR	-	(XO)	(XO)	XO
14h00-15h00	(XO)	XR	-	(XO)	(XO)	XO
15h00-16h00	(XO)	-	-	(XO)	(XO)	XO
16h00-17h00	(XO)	-	-	(XO)	(XO)	XO
17h00-18h00	-	-	-	(XO)	(XO)	-
18h00-19h00	-	-	-	(XO)	(XO)	-

LaTex

JassTracker is a web app which allows tracking and analysis of the popular Swiss card game “Jass”. There are many forms of playing, but the one we’re focusing on is called “Coiffeur”. The two teams have two players each and need to keep track of what they’ve already played and which options are available to them. They also track whose turn it is, apply the correct multiplication to the score and sum it all up in the end. To work around this, a project team member is currently using a excel spreadsheet, but this solution provides limited functionality and has many drawbacks.

To make the scoring easier JassTracker allows players to easily track, analyze and sync games digitally. In a first step you will be able to create and arrange team members for a given game. Then you start playing the game physically and assign scored rounds to the correct member. During this phase you’ll also be able to see live stats such as average score by player so far. After the game some highlights (e.g. the best player)

are highlighted. You can also gain exhaustive insight into your play style by looking at personal or group statistics such as average score by player or historic averages. To enable this, other physical members are able to associate their game to their personal account to track personal statistics across games.

Some potential ideas to expand on: configurable Jass game (e.g. Coiffeur with 8 instead of 10 options), prediction of scores based on past performance, current game trajectory (win probability by team).

Proposed Realization

We plan on implementing a web app using Vue.js as a frontend library. For styling we plan on using Bootstrap for basic styles. The server will be implemented in Kotlin using the Ktor framework. Persistent data is stored in a PostgreSQL database and accessed using jOOQ. Development will be done locally in IntelliJ IDEA, production deployments will be using Docker containers. CI / CD will be implemented using the OST GitLab.

Kapitel 9

Personal Reports

9.1 Team Review

Generally, whenever we had a question, other members were happy to help and could provide helpful input. The team was motivated to work on the project and work together to get everything done. It was good that we invested extra time in the beginning, to set up a good architecture and have clean code, which made it very easy to create new features/endpoints. There were times when it was a little challenging working in a team because of waiting dependencies, where one had to wait until someone finished something before you could continue working, which caused some delays. Nevertheless, we managed very well, especially after noticing this issue and setting internal deadlines during the spring to prevent this, and it got much better. After the large documentation block, we all wanted to start coding and focused less on creating informative Jira tickets. Minimal task descriptions often lead to further questions or having to change something again. Pascal-dependency was something we could improve on in the future, as he was often the person we contacted to answer questions, give more input on tasks and review our changes. In the last weeks, we agreed on trying to reduce this dependency with the usage of public chats, asking questions as comments on issues instead of direct messages. If we'd start the project from new again, we'd invest time into frontend testing, as nearly all our bugs were caused by the frontend, and we don't have any automated testing there.

9.2 David

Overall I'm happy with the whole project and enjoyed working on it. I had the opportunity to learn new technologies (Kotlin, jOOQ, ktor, pinia) and improve my knowledge of already known ones (javascript, vue.js). In a future project, I'd plan out my time better to prevent it all heaping up on the weekend and try to spread it out more under the week, which makes achieving the deadlines easier. In the beginning, I had to ask for help regarding certain technologies, but it became less with time after working with them for a while. In these past weeks, I feel like I've improved my skill set and can take

my learnings with me to the next project. My highlight was seeing everything coming together from the original idea and then being able to play a round of Jass after a few weeks.

9.3 Pascal

Overall, I really enjoyed our project and have no major complaints. We chose a very adequate tech stack for our problem scope, and I wouldn't change any technology when starting over. My personal highlight was the last three weeks when everybody got to work on new features, and we were able to extend our software without any significant roadblocks. However, there is one thing I would do differently, the requirements engineering. I tried my best to present reasonable requirements in my role as product owner, but that wasn't enough. We should've prioritized the requirements from the beginning and spent more time questioning potential end-users to tailor our solution in a better way.

9.4 Jamie

With this project, I had the opportunity to get to know a lot of new technologies. At the beginning of the project, I was hesitant to ask for help, which created some delays. I learned that there is a balance of knowing when you should ask for help. For future projects, I think I would invest more time in the requirement engineering and time management aspect of the project.

9.5 Marcel

I enjoyed working on the project, and I am happy with the end result. I was able to deepen my knowledge of Kotlin, Vue.js and TypeScript and got to learn a cool framework with Ktor. I could also bring my knowledge of SQL into the project and implemented all the database migrations. In the middle of the project I did not always put enough time into the project, which meant not always meeting my sprint goals. I was able to correct this for the last half of the project and was almost always able to get my tasks done within a sprint. This project was also my first contact with the onion architecture style, as I was previously more familiar with layered architectures that are more tightly coupled to the database.

Teil IV

Reflexion

Project name: JassTracker

Team Members

1. Pascal Honegger (pascal.honegger1@ost.ch)
2. Marcel Joss (marcel.joss@ost.ch)
3. David Kalchofner (david.kalchofner@ost.ch)
4. Jamie Maier (jamie.maier@ost.ch)

Availabilities

Time slot	Mon	Tue	Wed	Thu	Fri	Sat
08h00-09h00	(XO)	-	-	-	(XO)	-
09h00-10h00	(XO)	-	-	-	(XO)	XO
10h00-11h00	(XO)	-	-	-	(XO)	XO
11h00-12h00	(XO)	-	-	-	(XO)	XO
12h00-13h00	(XO)	(XR)	(XR)	(XO)	(XO)	XO
13h00-14h00	(XO)	XR	-	(XO)	(XO)	XO
14h00-15h00	(XO)	XR	-	(XO)	(XO)	XO
15h00-16h00	(XO)	-	-	(XO)	(XO)	XO
16h00-17h00	(XO)	-	-	(XO)	(XO)	XO
17h00-18h00	-	-	-	(XO)	(XO)	-
18h00-19h00	-	-	-	(XO)	(XO)	-

Project Idea

JassTracker is a web app which allows tracking and analysis of the popular Swiss card game "Jass". There are many forms of playing, but the one we're focusing on is called "Coiffeur". The two teams have two players each and need to keep track of what they've already played and which options are available to them. They also track whose turn it is, apply the correct multiplication to the score and sum it all up in the end. To work around this, a project team member is currently using a excel spreadsheet, but this solution provides limited functionality and has many drawbacks.

To make the scoring easier JassTracker allows players to easily track, analyze and sync games digitally. In a first step you will be able to create and arrange team members for a given game. Then you start playing the game physically and assign scored rounds to the correct member. During this phase you'll also be able to see live stats such as average score by player so far. After the game some highlights (e.g. the best player)

are highlighted. You can also gain exhaustive insight into your play style by looking at personal or group statistics such as average score by player or historic averages. To enable this, other physical members are able to associate their game to their personal account to track personal statistics across games.

Some potential ideas to expand on: configurable Jass game (e.g. Coiffeur with 8 instead of 10 options), prediction of scores based on past performance, current game trajectory (win probability by team).

Proposed Realization

We plan on implementing a web app using Vue.js as a frontend library. For styling we plan on using Bootstrap for basic styles. The server will be implemented in Kotlin using the Ktor framework. Persistent data is stored in a PostgreSQL database and accessed using jOOQ. Development will be done locally in IntelliJ IDEA, production deployments will be using Docker containers. CI / CD will be implemented using the OST GitLab.

Kapitel 10

Project Plan

10.1 Relevant Links

The following chapters are referencing external tools. For convenience, those tools are also listed here:

10.2 Processes

Meetings

Meeting	Schedule	Subject	Timebox
Weekly	Every Saturday	Scrum Daily	15m
Planning	Every other Tuesday	Scrum Planning	45m
Retro	Every other Tuesday	Scrum Retrospective	45m
Review 1	07.03.2022	Project Plan	1h
Review 2	21.03.2022	Requirements	1h
Review 3	04.04.2022	End of Elaboration	1h
Review 4	02.05.2022	Quality	1h
Review 5	23.05.2022	Architecture	1h
Presentation	07.06.2022	Present the great JassTracker	30m

Roles

Because of the small team size some members have more than one assigned role.

Who	Roles	Responsibilities
Pascal	PO, DEV, Architect	Push project vision, provide business feedback
Marcel	DEVOPS	Setup and maintain operations, continuous delivery
David	SM, DEV	Ensure correct scrum procedure
Jamie	QA, DEV	Guarantee the final product has a good quality and works reliably

DEV

A developer is responsible for the development of the target application. Every developer is responsible for the project / code quality as a whole and should act accordingly.

PO

The product owner is responsible to push the project vision forward. In planning meetings, he is responsible to get as many features done as possible. The product backlog is mainly created by the PO, but the team as a whole is responsible for maintaining it. But the PO has full authority over the prioritization of user stories.

DEVOPS

A devops engineer extends all responsibilities from a regular dev. In addition, he is responsible for operating the app by ensuring it's working correctly. He manages and oversees the deployment process and tracks metrics such as uptime or hardware utilization.

SM

The scrum master is responsible for the correct implementation of the scrum process. He is in charge of organizing the scrum meetings and documenting them if needed (e.g. Retrospective)

QA

Quality Assurance is responsible to ensure a good quality of the final product. To achieve this, quality metrics should be set up and tracked by QA, such as test coverage. It is not the responsibility of QA to fix such issues or run manual tests, that honor belongs to the whole team.

Architect

The architect is responsible for architectural decisions regarding the project. He evaluates and compares different patterns and methodologies to figure out the best fit for the

project. It doesn't mean he's solely responsible for implementing these, but has the lead on them and ensures that the other members are following them correctly.

10.3 Collaboration

We're using MS Teams for internal communication. We plan on using the OST GitLab for version control. Changes are always done on separate branches and integrated through merge requests with a required code review.

10.4 Risk Management

10.4.1 Very High

1. Used technologies are not well knows by all team members (certain, critical)
   (a) Mitigation: Every team member should create a PoC in the used technologies to ensure basic understanding is present
   (b) Mitigated risk: Low

10.4.2 High

1. Inaccurate estimations (likely and critical)
   (a) Mitigation strategies: apply cone of uncertainty, apply definition of ready to ensure planning quality
   (b) Mitigated risk: Low
2. Poor risk management (likely and critical)
   (a) Mitigation strategies: likelihood calculation, risk mitigation plans and monitoring of risks every planning
   (b) Mitigated risk: Low
3. Project reviewer's expectations are not aligned with project (possible and critical)
   (a) Mitigation strategies: obtain frequent approval and acknowledgement (naturally happens for us with review meetings)
   (b) Mitigated risk: None
4. Unexpected absence of team member (unlikely and catastrophic)
   (a) Mitigation strategies: Code changes need to be pushed on a daily basis, stories could at any point be taken over by another team member
   (b) Mitigated risk: Medium

10.4.3 Medium

1. Insufficient code quality (possible and marginal)
   (a) Mitigation strategies: code reviews, clear coding standards, apply definition of done
   (b) Mitigated risk: None
2. Lack of ownership (possible and marginal)
   (a) Mitigation strategies: setting clear responsibilities for roles
   (b) Mitigated risk: None
3. Losing sight of documentation tasks (possible and marginal)
   (a) Mitigation strategies: documentation strategy, documentation part of definition of done
   (b) Mitigated risk: Low
4. Failure of hardware like personal devices, OST GitLab, Jira, hosted environment (rare, catastrophic)
   (a) Mitigation strategies: Code changes need to be pushed on a daily basis
   (b) Mitigated risk: Low

10.4.4 Low

1. Project idea, "Jassen", is not well known or understood by team members (possible, negligible)
   (a) Mitigation strategies: Play a "Jass" in the team to ensure everybody knows the basic rules and concepts
   (b) Mitigated risk: Low

10.5 Time & Issue Tracking

Our project plan is tracked on our Jira Board. All team members are already experienced with Jira and it provides many desired features (Epics, burndown charts, sprints) which were lacking in GitLab.

Kapitel 11

Time Tracking Report

Time tracking is done exclusively in Jira. Detailed reports with hours spent per issue, epic and sprint can be found there. The following charts focus on the overall progress to notice early trends, in case certain members invest significantly more or less time than expected

Who	**Logged Hours**	**Expected Hours**
Pascal	135h	120h
Marcel	115h	120h
David	111h	120h
Jamie	106h	120h
Team	468h	480h

Glossar

Acorns A suit in the Swiss playing cards, also known as Eicheln.

Bells A suit in the Swiss playing cards, also known as Schellen.

Bottoms-up A kind of Contract in the Coiffeur Jass which plays without a trump. The lowest card always wins, the 8 card counts as 8 points and the 6 is worth 11 points, but the ace is worthless.

Coiffeur Jass A variant of Jass which is played by four players. Two teams of two players each compete against each other, trying to get the higher score.

Contract A available option within a coiffeur. Possible contracts: any Trump, Bottoms-up, Tops-down, Slalom and Guschti.

Game A game starts by selecting the appropriate table and starting a new game. A game consists of 20 rounds in total, 10 for each team.

Guschti A kind of Contract in the Coiffeur Jass which plays without a trump. Starts with either Bottoms-up or Tops-down for the first 5 Tricks and then flips, scoring rules apply from the first chosen option.

Jass A Swiss card game, see Wikipedia for more details.

Jass table Physically Jass is usually played sitting around a table. In Coiffeur Jass, diagonally opposite players are in the same team.

Joker A special Contract which can be used to do any other valid contract, even if already used.

pass A player has the option to not choose a Contract but instead pass on the decision to the next player (Schiebe). This can only be done once per round.

Roses A suit in the Swiss playing cards, also known as Rosen.

Round A round starts by being dealt 9 cards and is done once all cards are played.

score To take a Trick. After every player played one card from his hand the player with the strongest card in the current Contract scores.

Scoreboard The board on which the scores (points) are tracked. Knows as "Jasstafel" in swiss german.

Shields A suit in the Swiss playing cards, also known as Schilten.

Slalom A kind of Contract in the Coiffeur Jass which plays without a trump. Alternates between Bottoms-up and Tops-down or vice-versa, scoring rules apply from the first chosen option.

Swiss-suited playing cards A big part of swiss german speaking switzerland uses the Swiss-suited playing cards. There are four suits: Bells (Schellen), Shields (Schilten), Roses (Rosen), Acorns (Eicheln).

Tops-down A kind of Contract in the Coiffeur Jass which plays without a trump. The highest card always wins, the 8 card counts as 8 points.

Trick A trick consists of four cards played in counter-clockwise order.

Trump A suit can be declared as the trump suit (Trumpf), which changes the card values within that suit for the round.

Literaturverzeichnis

[Jen19] Jensche. Medien in der kategorie “jass”. `https://commons.wikimedia.org/wiki/Category:Jass`, 2019. [Online; accessed 21-May-2022].