Recess Chess
Main Report

Team 20:

Tobias M. Andersen, Kjerand Evje, Ole Jonas Liahagen,
Dilawar Mahmood, Håvard S. Markhus, Martin Johannes Nilsen

Table of Contents

Preface

This application has been developed as a project for the course System Development 1. This emphasizes the use of UML diagrams in projects. Therefore, the wiki contains several types of UML diagrams, such as sequence-, class- and database diagrams. In addition to this the course is parallel with a course on Databases and object-oriented programming, which is why using a database with SQL is part of the project, as well as inheritance and polymorphism. This course is a central part of our education as its specialization revolves around system development. A key takeaway for the students is how to work in teams and manage bigger projects than one individual can handle.

Introduction

Purpose

The main report is written to give an overview of the task, and reflections on how the team dealt with various problems. These things are specifically elaborated in chapters: "3. Assignment description", "4. How the assignment was solved", and "5. Implementation of the project".

Scope

This report concerns all parts of Team20s project for the course System Development 1. For further information on what that project revolves around, read "3. Assignment description".

Definitions, Acronyms, and Abbreviations

Salt-hashing: Encryption method used in this project to keep the user's password safe.
MVP: Abbreviation for "minimum viable product."
Java: Programming language.
HikariCP: Library used for connection-pooling in the database.
JRE: Abbreviation for "Java Runtime Environment." Software that is used to run Java programs.
JDK: Abbreviation for "Java Development Kit." Software that is used to develop and compile Java code.
SQL: Abbreviation for "Structured Query Language." Query language for communicating with the database.
IntelliJ: Development environment for Java products.
Git: Version control system.
JavaFX: Library in Java for creating and managing GUI (see GUI).
GUI: Abbreviation for "Graphical User Interface."

References

Below are the referenced documents with their report numbers in the Main report document.

1. Status reports with performance evaluations for each member.
2. Project plan with the schedule in the form of a Gantt chart
3. Timesheets with status reports for each team member
4. Meeting invitations and minutes
5. Vision document

Overview

The rest of the document is split into four parts describing some core parts of the Team20 Chess project. Assignment description recites the task the team was given, and what requirements they would have to keep in mind. "How the assignment was solved" is self-explanatory, but ranges from software usage, languages used to documentation overview. "Implementation of the project" is mainly comparison and reflection between our plans (in the form of vision document, gantt-diagram etc) and actual development. There are also some thoughts about our collaboration and use of collaborative tools.

Assignment description

The assignment is to create a multiplayer game using Java where each player uses their own laptop to play. Each player should have a personal account and should be able to log on with a username and password. The password should also be hashed with a salt. The multiplayer game should not contain any animations as that part of development lies outside the focus of the course. As we have to use databases to communicate between the players, games in real-time will probably be quite hard, so making a turn-based game was the recommended way to go.

How the assignment was solved

- ***Methods and standards used***
  - Gitlab issue-boards
  - UML-diagrams to keep track of program and database structure
  - Javadoc-documentation standards
- ***Use of literature and Internet***
  Use of sources was mostly limited to literature gathered on the internet, where we found inspiration from several sources. The most used sources were Bucky Roberts (www.youtube.com/thenewboston) and several other quoted sources from www.stackoverflow.com. These sources were, as stated, mostly used for inspiration and guidance in places where we were stuck. For example: the use of services to run threads were heavily influenced by this source: https://stackoverflow.com/questions/16708931/javafx-working-with-threads-and-gui
- ***Overview of the hardware used***
  - Database servers provided by NTNU.
  - Network location provided by NTNU.
  - Personal computer
- ***Brief description of standard software used***
  - IntelliJ: IDE for Java development. Licensed by JetBrains, full version provided by NTNU.
  - MySQL-driver (jdbc-connector): Driver allowing for communication via clients and the SQL-database, integrated in Java code.
  - HikariCP: Library allowing for connection-pooling. Made by Brett Woolridge: https://brettwooldridge.github.io/HikariCP/
  - Git: A version control system that provides an easy method for programmers to collaborate on separate computers.
- ***How the work was divided between the team members***
  The work was divided quite evenly between the team members. Each team member got their own task to complete, after which they could opt in to help other team members if they had completed their own task. Some tasks were quite short, so the team members provided these, often got a new task before the other members were finished. This made the work quite flexible and allowed the members already done with their tasks to choose between helping each other or opting in to assist the others. Naturally, the members helped each other as problems arose as well.
- ***An overview of the documentation***
  The documentation consists of:
  - Documentation of the source code
    - Using javadoc, every method and class is described and documented...
  - Status reports
    - Every member has a written status report for every week with a performance evaluation at the end. Status reports are updated at the end of every week and goes through the tasks the member has worked with and are working on currently, as well as problems that occurred and how we solved them.
  - Timesheets
    - Documentation of how and when the working hours were spent.
    - Timesheets were updated after each working day and says something about what the team member was working on during the work day.
  - Diagrams and charts (can all be found in the wiki --> https://gitlab.stud.iie.ntnu.no/martijni/team-20---spillprosjekt/wikis/home)
    - Class diagram
    - Sequence diagrams

- Domain model
- Use case diagrams
- Database model

Implementation of the project

We started out brainstorming which kind of game we could make right after we were given the assignment. Chess was the game the group decided to make, and we began creating some basic parts needed for the teamwork to work out, for example collaboration agreement, common google drive/GitLab and messenger/discord for communication. The highest priority features for the game was to have online play where the players could take turns on moving a piece.

Additional high priority planned features that were achieved:

- A chat system
  - the chat system was added and worked on relatively early and was completed in due time.
- Timer for the game, chosen by the player
  - The timer was not something we prioritized highly but was worked on as soon as we had the MVP and we could start thinking about adding additional features. This feature was kept in mind the entire project, but only initiated and implemented at the end.
- Ranking system
  - The ranking system was also kept in mind from the start and the system/formula for calculating ELO rating was implemented quite early in the project, but not realized before we had the MVP down.
- Alternative game modes
  - We initially planned to have extra game modes like Fischer random, but in the end, product ended up with several alternative game modes:
    - Fischer Random
      - Game mode where all the second-row pieces are placed randomly with two rules: the king has to be between the rooks and the bishops have to be placed on opposite colored tiles to avoid stalemate.
    - Farmer's chess
      - All the pieces are pawn except for the king
    - Peasants revolt
      - One side has 4 knights and a king, while the other side has a king and eight pawns
    - Horse attack
      - All the pieces except the pawns and the king are replaced with knights
  - The game modes were relatively easy and quick to implement as most of the work was done by changing how the board started out. The exception was Fischer random as the locally random chess game has to be the same for both players. The way we solved this was by passing a random seed from the creator of the game in the database for the player that joins it to use on their own chess board.
- Showing legal moves for a given piece
  - This feature was necessary for the game to actually be playable as we only want the user to be able to make legal moves. As this was a very crucial part of the MVP. We started early on working with this and kept perfecting it until the end to account for all the special rules.

Bonus features we did not plan:

- Skin system
  - The skin system is an idea that came up during the implementation part of the project and is a feature that helps our chess game be a bit more unique.
  - The implementation of the skin system was at first just options to change the color of the board, but we decided to add possibility to change skins on the pieces as well. The skins are all saved as images in the application, and the setting for the skin is saved in the database on "UserSettings" as the name of the skin in the column "SkinName". every time a game is started the game gets the

information about skinName from the database for both players in the game and sets up the skins for the pieces accordingly.

Features we planned to do if we had time, but did not achieve:

- Ability to see previous games
- Ability to use custom emotes in the chat
- Chess bot for single player
- Ability to see which opening is being played at a specific point in time

In the following weeks we started coding and after 8 weeks we stand here with a finished game. In this part we are going to describe what was successful or not, what we should have done differently, how the technical issues have been resolved and what limits the system:

- Git problems
    - During the project, we had some git problems with the merging of the code each member of the team wrote on their own laptop. The main problem was the merging of the xml-files in the project. These files originated from the IDE we used (IntelliJ). At first, we solved the problem by using a gitignore-file. When that didn't work, we used git revert when we accidentally pushed a xml-file.

We had four project phases:

- Planning:
    - The team set out to be done with planning 9th of march, roughly 3 weeks into the project, as most of us were not going to have a lot of time for the project during the first few weeks. The planning was mostly done a week before the deadline we set, as we did not use a lot of time on planning the classes for the game itself. The planning phase ended up being stretched out and finished '"on the go" instead of fully finishing the phase before we started implementing. The entire phase itself was not finished as planned. An example is that implementation of the game logic started before we planned the database system.
- Implementation:
    - The implementation was already planned to have the tasks running concurrently, with game logic as a task planned to last the entire "phase". The foundation of the game logic was finished a lot sooner than planned, as we had several people working on the task. However, the board graphics was not finished as soon as we'd expect as it had several complications and needed extra logic to function like we intended. The graphics consist of javafx panes and buttons and needed to communicate with the database and the game logic classes, a lot of special rules and cases took a lot more time to implement than expected, as we had to account for multiplayer and a "sandbox" version of the board as well.
- Testing:
    - The testing was not a single standalone phase, but rather a continuous part of every implementation of code in the project. After implementing a new component or a new feature, we tested that component along with how it interacts with the other components we have implemented earlier in the project. After we were finished with the program, we tested the application as a whole, and we did also conduct some user tests. The tests consisted of playing chess games with each other's and trying out different scenarios and edge-cases in the game.
- Documentation:
    - The documentation was planned to be executed in several separate phases to make for a smooth documentation process. We did not do it this way, however. Instead we decided to dedicate the last week to documentation and setting a deadline for coding work the week before. This way the whole team could focus on writing good documentation without any code being changed. This approach led us to start all the phases of the documentation process almost simultaneously. Therefore, we cannot say that we strictly followed the gantt-chart provided. Even though this approach did not follow the plan originally set, we feel like it was a decent solution overall.

Solutions and areas we were satisfied with:

- **Fischer Random board layout**

Image 1 below shows a snippet of the solution we had for the mode “Fischer Random” and how we made sure the randomly generated board was the same for both users. The creator of the game creates a random number between 1000 and 5000 and sets this number as the “mode” for the game in the database. The client joining the game checks the number for mode and knows it is Fischer random when the number is higher than 1000. The random number is used as a seed for the Random() object in the code that generates the board, so both clients (the creator and the joiner) use that same number between 1000 and 5000 as a seed for their Random() object and uses that object to generate new random numbers for where the pieces will be placed. As both objects have the same seed the same random numbers will appear for both, thus we save a lot of time and data in transferring what the board looks like and we only have to send the seed.

```
} else if (mode > 1000) {
    //fischer random
    //the value of mode is a randomly selected number which is used as a seed to create a random board setup
    //this seed is uploaded to the database so that both computers use the same seed so that the same board is generated
    Random random = new Random(mode);

    int whiteBishops = random.nextInt( bound: 4);
    position[whiteBishops*2 + 1][0] = new Bishop( color: true,  x: whiteBishops*2 + 1,  y: 0);
    position[whiteBishops*2 + 1][7] = new Bishop( color: false,  x: whiteBishops*2 + 1,  y: 7);

    int blackBishops = random.nextInt( bound: 4);
    position[blackBishops*2][0] = new Bishop( color: true,  x: blackBishops*2,  y: 0);
    position[blackBishops*2][7] = new Bishop( color: false,  x: blackBishops*2,  y: 7);
```

Image 1

- **The responsiveness and smoothness of the multiplayer**

We were struggling a lot with figuring out how to send queries to the database when awaiting a move from the enemy and uploading your own moves while keeping the software responsive and the upload and polling time fast. Since the chess board is just a lot of javafx items, in order for them to stay responsive, we cannot have loops running in the main thread as it has to be reserved for the key listeners. We needed the polling of enemy moves and gestures to run in another thread to keep the software responsive, as well as being able to change the layout of the javafx scene. We start a timer (Image 4 below) in the main thread that runs the method from *image 3* below, which keeps starting a new thread running the method in an interval we choose. Using booleans we make sure a new one is not started before the previous one is finished, so the timer does not call “enemyMovePollingService()” before the previous one is already finished, this makes it possible for us to poll sequentially as fast as we feel like, while still keeping the application responsive. The uploading of moves is also done within threads to keep the game responsive. Upping the polling rate to the max makes it possible to play blitz (very fast chess) without issues as the move transfer feels almost instantly.

- **The clever programming of the game logic**

Image 5 below describes an algorithm for checking if there is check on the king from all 8 directions. We are proud of this piece of code because of how compact and elegant it is, and this piece of code also reflects how complex the programming of the chess logic can be. Another thing we want to point out is the use of a HashMap for determining if there has been a threefold repetition in the game. We are pleased with this implementation since the HashMap is an unknown data structure that we have not learned about (Image 1). Overall, we are very pleased with the GameLogic-class, and it is the most logic-heavy class in our code.

```
public static boolean isMoveRepetition(HashMap<String, Integer> rep, Board board){
    if (rep.containsKey(board.toString())){
        int oldBoardState = rep.get(board.toString());
        rep.put(board.toString(), oldBoardState + 1);
    }
    else{
        rep.put(board.toString(), new Integer( s: "1"));
    }
    return rep.get(board.toString()).compareTo(2) == 1;
}
```

Image 2

```java
public void enemyMovePollingService() {
    //System.out.println("Started service: service = " + serviceRunning);
        Service<Void> service = () -> {
                return () -> {
                        //Background work
                        final CountDownLatch latch = new CountDownLatch(1);
                        //System.out.println("entered service");
                        serviceRunning = true;
                        Platform.runLater(new Runnable() {
                            @Override
                            public void run() {
                                //System.out.println("Starting myTurn check in service:");
                                pollEnemyMove();
                                latch.countDown();
                                serviceRunning =false;
                            }
                        });
                        latch.await();
                        //Keep with the background work
                        return null;
                };
        };
        service.start();
}
```

Image 3

```java
public void startTimer(){
    timer = new Timer( isDaemon: true);
    timer.scheduleAtFixedRate(() -> {
            if(isDone){
                timer.cancel();
            }
            else if(!polling && !serviceRunning && !myTurn) {
                enemyMovePollingService();
            }
    }, delay: 0, period: 250);
}
```

Image 4

```java
int[][] move =  {{1, 0, 0, 1, -1, 0, 0, -1}, {1, 1, -1, 1, -1, -1, 1, -1}};
boolean[][] dir = {{true, true, true, true}, {true, true, true, true}};
for (int r = 0; r < 2; r++){
    for (int i = 1; i < 8; i++){
        for (int k = 0; k < move[r].length; k += 2){
            if (x+i*move[r][k] >= 0 && x+i*move[r][k] < 8 && y+i*move[r][k+1] >= 0 && y+i*move[r][k+1] < 8){
                if (dir[r][k/2]){
                    if (board[x+i*move[r][k]][y+i*move[r][k+1]] != null){
                        if (board[x+i*move[r][k]][y+i*move[r][k+1]].getColor() != color){
                            if (r == 0 && (board[x+i*move[r][k]][y+i*move[r][k+1]] instanceof Queen || board[x+i*move[r][k]][y+i*move[r][k+1]] instanceof Rook)) {
                                return true;
                            }
                            if (r == 1 && (board[x+i*move[r][k]][y+i*move[r][k+1]] instanceof Queen || board[x+i*move[r][k]][y+i*move[r][k+1]] instanceof Bishop)) {
                                return true;
                            }
                        }
                        dir[r][k/2] = false;
                    }
                }
            }
        }
    }
}
```

Image 5

Solutions and areas we could have done better:
The team is overall satisfied with the end product, however there are some areas where we could have done better:

- Less static variables, more getting and setting:
    - This is an area we definitely should improve on; using java for what it is - an object-oriented language. Somewhere along the way while programming, static variables started to appear more and more frequently while nobody really pointed this out. These variables stuck and were used frequently throughout the source code. Eventually they were used so much that it would be pretty tedious to correct everything, so we just stuck with it. After all, it did work! However, changing a variable name, or moving a variable meant that we would have to change this name in every occurence of the variable, making this method not very efficient.
- More structured workflow
    - More elaborate planning and better structure in giving out tasks. In a lot of cases individuals within the team were not aware of what others were working on and often resulted in problems with version control.
- More dynamically programmed UI
    - Creating the UI in such a way that it is easy to edit and dynamically works well on most screen sizes and resolutions.
- Better usage of Git branches
    - This is our first project using Gitlab for collaborative programming, and we had as stated earlier on, some issues in regard to merging. On second thoughts, we should have been better at using different branches and merged these after finishing up parts of the code, as everyone used the master branch and merged at different times. We did make it work out quite decently, but this was heavily depending on us working on different classes and communicating about what we did from time to time.

Further work

Recess Chess is an online chess platform where it is possible to play chess with another player using their own laptops. There is also support for making your own user, chatting with other users, using custom skins and playing with different game modes.
Further work could have been to implement functionality to invite a user directly from their profile and possibility to take a rematch directly after a game has ended. With addition to that, we could also make support for seeing pieces that have been taken during the game. Further, we could also add a "forgot password" functionality for users who forget their own passwords. Another mode we could add is "bot mode", which would be a mode where you can play against the machine. That could be done by either downloading and implementing stockfish, or by programming our own AI by using Minimax or deep Q-learning.

Repository

Below you will find a link to out Gitlab-repository:
https://gitlab.stud.iie.ntnu.no/martijni/team-20---spillprosjekt/wikis/home
(No username/password needed to run the program, just create an account within the application).

Attachment 1: Collaboration agreement

Collaboration agreement - Team 20

Members:
Tobias Meyer Andersen, Kjerand Evje, Ole Jonas Liahagen, Dilawar Mahmood, Håvard Stavnås Markhus, Martin Nilsen

The purpose of this collaboration agreement is to determine guidelines and to give an overview of common rules and goals.

Our goals

Outcome-goals

1. Get to know each other and build trust.

To achieve this we will…

In the beginning, we focus on getting to know each other's interests, strengths and weaknesses. We believe that the trust in a group is the result of opening up and to give of one's self.

2. Collaborate effectively with good communication

To achieve this we will…

Start the workday with a brief meeting about work distribution; what we have done so far, what we should do now, and what should be done before our next meeting. We will also plan when during that day we will update each other with our current progress. For this purpose we have also created a messenger chat, and a discord server so we can reach one another outside of the classroom. All members of the team are obligated to check the discord server for upcoming meetings and other information on a daily basis.

3. *Flexible workload*

The different tasks will alternate between team members, so that no team member is stuck with the same type of work throughout the whole project.

If a member struggles with their task, then it is encouraged to seek help from other members so that the team can be as effective as possible. This alternation between tasks will also contribute to each members increased understanding of the project's different aspects. This makes it so everyone knows a little bit about everything, making it possible to help each other as we begin to specialize.

If a member is displeased with their tasks, then this alternation can will provide an opportunity to switch tasks and trying something new.

Result Goals

1. Stay within all deadlines

We will always strive to complete the tasks with time to spare. In this way we will have additional time to handle possible unforeseen problems. If the project is evaluated using grades, the team will come to an agreement on what grade to aim for as soon as possible. All members shall also keep track of the upcoming deadlines, and dare to be honest if work is inefficient.

2. *Deliver finished work that all members are happy with*

The completed work must be approved by all members as a finished product. If this is not the case, then everyone has a shared responsibility to do their best to improve their work so that everyone's approval is achieved.

Roles and distribution of workload

A. Teamleader: Dilawar Mahmood

The team leader is in charge of the distribution of workload. He also has the final say in matters where the group is unable to come to an agreement. The leader is also responsible for the well being of the team-members and ensuring that the assigned work is done.

B. *Organisation of meetings*

Meeting coordinator: Kjerand Evje
Meeting-moderator: Dilawar Mahmood

The date of the meeting is to be agreed upon in due time. The Meeting Coordinator is to inform the other team members of the agreed upon meeting date and place. He or she also has the responsibility of informing the team about the meeting's agenda.

The responsibility of the Meeting Moderator is to moderate/lead the meeting and ensure that each matter gets enough time to be discussed sufficiently. He is also responsible of keeping the meeting relevant and factual and to avoid the discussion of irrelevant or otherwise unnecessary discussions.

C. *Archivation and documents*

Archivist: Håvard Markhus

The archivist is responsible for keeping order of the documents.
The documents should be systematized and organized so that every member can orient themselves in the archive.
The archivist is the one responsible of setup and finalization of reports. Every document should be archived and systematized in our common Google Docs-folder.

D. *Reporter*

Responsible: Tobias Meyer Andersen

Writes reports from every meeting. The report should include the date, who's present, main issue and possible solutions. It should be short and factual. Reports should be stored in an own folder, and submitted in the common Discord-chat.

E. *The person responsible of submission, quality assurance of submissions*

Primarily responsible: Ole Jonas Liahagen
Secondarily responsible: Martin Nilsen

This person has the responsibility of reviewing reports/code.
Should look for errors and weaknesses. Be critical!
Has the responsibility of securing that everything regarding the task is delivered on time.

Expected behaviour and rules

A. Attendance and preparations

Team members should ideally meet 5 minutes before the agreed time for group meetings and work sessions.
Before group meetings the individual team members shall inform themselves about the topics that will be discussed in the meeting. This way everyone will be prepared for the meeting.
The meeting leader shall keep a spreadsheet where he keeps account of which team members attended the meeting and potentially information about reasons for their absence.
Absence from meetings shall be reported to the meeting leader as early as possible, at the latest 1 hour before the meeting. The absence shall be reported with a reasoning by messaging through either Facebook-chat or on the team-Discord.
If unexpected complications arise, eg. a bus is delayed, the absence will be reported the same way.

A formal warning is given immediately if a team member is late (10 minutes) for a meeting without having reported their absence beforehand. If a team member gets three warnings it will have further consequences.
There will be a meeting every week where the team members discuss what they have done the previous week and plan what they are going to do the next week. The meeting shall start at the agreed time even though all members are not present, unless a team member has reported he will be a couple minutes late.

B. *Presence and involvement*

When present on a workday, it is important to stay focused on the theme and the task at hand. In an effort to create a good working environment, induced procrastination should be avoided, but it is always allowed to agree on taking a break.

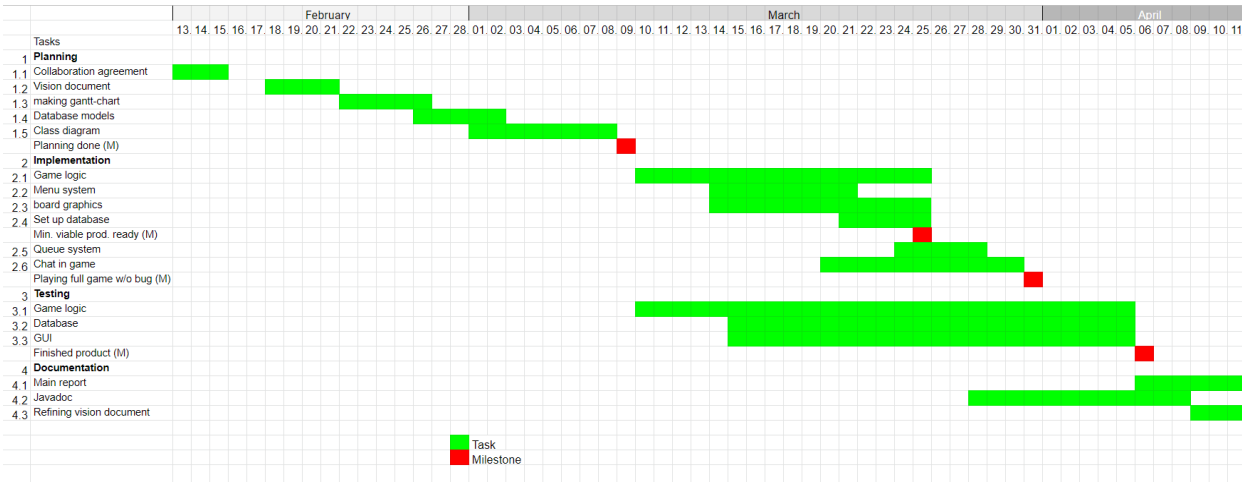
C. *Disagreement, violation of rules*

Disagreements that cannot be solved between to individuals within the group, should be presented to the rest of the group. If four people agree on a case, those four's decision will be the conclusion. If the team does not agree and is in a situation where two agree while the rest do not, a lecturer should be included in the discussion. If the team still cannot agree, the team leader will make the decision.

If a person within the team breaks the rules or exerts behaviour that does not correspond with the contract regularly, the rest of the team has the responsibility of conducting a meeting. In the meeting the person in mind will be confronted and informed that the rest of the team is not happy with his or her behaviour. During the meeting where he or she is confronted, an attempt to finding a solution to the source of the problem will be done. The confronted person will be given 2 weeks where an evaluation of his or her behaviour during these 2 weeks will be done. The evaluation will decide if they get to stay in the team.

Attachment 2: Gantt-chart

Here we have our project plan in the form of a Gantt-chart:



Attachment 3: Timesheets for each team member

Timesheet for Martin

Week 7	Start	End	Start	End		Description
Monday 11.02.2019					0,0	
Tuesday 12.02.2019					0,0	
Wednesday 13.02.2019	08:15	10:00	12:00	13:30	3,3	Kickoff, brainstorming and collaboration agreement
Thursday 14.02.2019	12:00	13:30			1,5	Translation of collaboration agreement to english
Friday 15.02.2019					0,0	
Saturday 16.02.2019					0,0	
Sunday 17.02.2019					0,0	
Total amount of hours:					**4,8**	
Week 8 - Studyweek	Start	End	Start	End		Description
Monday 18.02.2019					0,0	
Tuesday 19.02.2019					0,0	
Wednesday 20.02.2019					0,0	
Thursday 21.02.2019	08:50	09:50			1,0	Created this timesheet
Friday 22.02.2019					0,0	
Saturday 23.02.2019					0,0	
Sunday 24.02.2019					0,0	
Total amount of hours:					**1,0**	
Week 9	Start	End	Start	End		Description
Monday 25.02.2019	09:00	12:00			3,0	Vision document
Tuesday 26.02.2019					0,0	
Wednesday 27.02.2019	12:00	14:00			2,0	Vision document
Thursday 28.02.2019	10:00	12:00			2,0	Vision document
Friday 01.03.2019					0,0	
Saturday 02.03.2019					0,0	
Sunday 03.03.2019	10:30	18:30			8,0	Login.java and Javafx
Total amount of hours:					**15,0**	
Week 10	Start	End	Start	End		Description
Monday 04.03.2019	09:00	15:00	15:45	16:45	7,0	Further work with Login.java og javafx
Tuesday 05.03.2019	10:00	16:00			6,0	Redesign of Login.java
Wednesday 06.03.2019					0,0	
Thursday 07.03.2019					0,0	
Friday 08.03.2019	14:00	15:30			1,5	Created a mainScene and started creating new classes for new scenes
Saturday 09.03.2019					0,0	
Sunday 10.03.2019					0,0	
Total amount of hours:					**14,5**	
Week 11	Start	End	Start	End		Description
Monday 11.03.2019	10:00	16:00			6,0	Creating new scenes and general GUI
Tuesday 12.03.2019	10:00	17:00			7,0	Further work on scenes - GUI
Wednesday 13.03.2019	10:00	18:00			8,0	Working on design and new scenes, GUI
Thursday 14.03.2019					0,0	
Friday 15.03.2019	16:00	18:00			2,0	Cleanup of databaseconnections, using DBOps in all classes
Saturday 16.03.2019	12:00	16:00	17:00	20:00	7,0	Creation of sandbox chess game
Sunday 17.03.2019	21:30	23:00			1,5	Fixes for sandbox chess game
Total amount of hours:					**31,5**	
Week 12	Start	End	Start	End		Description
Monday 18.03.2019	10:15	17:00			6,8	GUI
Tuesday 19.03.2019	10:00	16:00	22:30	23:15	6,8	GUI
Wednesday 20.03.2019	10:00	18:20			8,3	GUI
Thursday 21.03.2019					0,0	
Friday 22.03.2019					0,0	
Saturday 23.03.2019					0,0	
Sunday 24.03.2019	14:30	16:30			2,0	clearboard fix and skinName in settings
Total amount of hours:					**23,8**	
Week 13	Start	End	Start	End		Description
Monday 25.03.2019	10:00	16:40			6,7	GUI, createGame- & joinGame-popup, and team meating
Tuesday 26.03.2019	10:00	14:00			4,0	GUI
Wednesday 27.03.2019					0,0	
Thursday 28.03.2019					0,0	
Friday 29.03.2019					0,0	
Saturday 30.03.2019					0,0	
Sunday 31.03.2019					0,0	
Total amount of hours:					**10,7**	
Week 14	Start	End	Start	End		Description
Monday 01.04.2019	10:00	17:00			7,0	Fix mainscene, optimalization of scenechange
Tuesday 02.04.2019	10:00	17:30			7,5	Further optimalization, better placements in grids etc.
Wednesday 03.04.2019					0,0	
Thursday 04.04.2019	11:00	14:30			3,5	Created methods for code in MainScene, and further optimalization
Friday 05.04.2019					0,0	
Saturday 06.04.2019					0,0	
Sunday 07.04.2019					0,0	
Total amount of hours:					**18,0**	
Week 15	Start	End	Start	End		Description
Monday 08.04.2019	10:00	18:00			8,0	Documentation, Main Report
Tuesday 09.04.2019	10:00	17:20			7,3	Documentation, Main Report
Wednesday 10.04.2019	10:00	17:20			7,3	Installation manual, User manual, WIKI, Main Report
Thursday 11.04.2019	07:00	08:40	12:00	14:00	3,7	Finished installation manual and user manual
Friday 12.04.2019	11:00	13:00	17:00	23:00	8,0	Finishing up documentation, main report. Prepare delivery
Saturday 13.04.2019					0,0	
Sunday 14.04.2019					0,0	
Total amount of hours:					**34,3**	
Total Prosjekttid:					153,6	

Timesheet for Dilawar

Week 7						Description of activity
Monday 11.02.2019					0,0	
Tuesday 12.02.2019					0,0	
Wednesday 13.02.2019	08:15	10:00			1,8	Kickoff, brainstorming og collaboration agreement
Thursday 14.02.2019	12:00	13:30			1,5	Translating collaboration of agreement from norwegian to english
Friday 15.02.2019					0,0	
Saturday 16.02.2019						
Sunday 17.02.2019					0,0	
Total amount of hours:					**3,3**	
Week 8 - Studyweek						
Monday 18.02.2019	09:00	12:00			3,0	Programming: Pieces basic logic
Tuesday 19.02.2019	09:00	12:00			3,0	Programming: Board basic logic and design
Wednesday 20.02.2019					0,0	
Thursday 21.02.2019	12:00	16:00			4,0	Vision document
Friday 22.02.2019	10:00	16:00			6,0	Programming: Finishing pieces and board
Saturday 23.02.2019					0,0	
Sunday 24.02.2019					0,0	
Total amount of hours:					**16,0**	
Week 9	Starttidspunkt	Sluttidspunkt	Starttidspunkt	Sluttidspunkt		
Monday 25.02.2019	09:00	12:00			3,0	Programming: Game logic
Tuesday 26.02.2019					0,0	
Wednesday 27.02.2019	12:00	14:00			2,0	Programming: Game logic
Thursday 28.02.2019	10:00	16:00			6,0	Programming: Game logic
Friday 01.03.2019					0,0	
Saturday 02.03.2019					0,0	
Sunday 03.03.2019					0,0	
Total amount of hours:					**11,0**	
Week 10						
Monday 04.03.2019	10:00	12:00			2,0	Programming: Movement of pieces
Tuesday 05.03.2019	10:00	17:00			7,0	Programming: Game logic: Movement of pieces
Wednesday 06.03.2019	12:00	14:00			2,0	Programming: Game logic: Valid moves
Thursday 07.03.2019	10:00	16:00			6,0	Programming: Game logic: Valid moves
Friday 08.03.2019	14:00	16:00			2,0	Programming and modeling database
Saturday 09.03.2019					0,0	
Sunday 10.03.2019					0,0	
Total amount of hours:					**19,0**	
Week 11						
Monday 11.03.2019					0,0	
Tuesday 12.03.2019	10:00	14:00			4,0	Programming game logic
Wednesday 13.03.2019	10:00	12:00			2,0	Programming special moves
Thursday 14.03.2019					0,0	
Friday 15.03.2019	13:00	15:00			2,0	Programming special moves
Saturday 16.03.2019					0,0	
Sunday 17.03.2019					0,0	
Total amount of hours:					**8,0**	
Week 12						
Monday 18.03.2019	12:00	16:00			4,0	Further work with finishing logic
Tuesday 19.03.2019	10:00	16:00			6,0	Programming ChessDemo
Wednesday 20.03.2019	10:00	14:00			4,0	Programming ChessDemo
Thursday 21.03.2019	12:00	14:00			2,0	Testing
Friday 22.03.2019					0,0	
Saturday 23.03.2019					0,0	
Sunday 24.03.2019					0,0	
Total amount of hours:					**16,0**	
Week 13						
Monday 25.03.2019	10:00	17:00			7,0	Programming timer
Tuesday 26.03.2019	10:00	14:00			4,0	Programming timer
Wednesday 27.03.2019	12:00	14:00			2,0	Testing timer
Thursday 28.03.2019	12:00	15:00			3,0	Fixing sync and fixing threefold repetition
Friday 29.03.2019					0,0	
Saturday 30.03.2019					0,0	
Sunday 31.03.2019					0,0	
Total amount of hours:					**16,0**	
Week 14						
Monday 01.04.2019	10:00	15:00			5,0	Programming: Working with sync
Tuesday 02.04.2019	10:00	16:00			6,0	Programming: Finished timer
Wednesday 03.04.2019	10:00	22:00			12,0	Programming graphics and logics for writing out moves
Thursday 04.04.2019	10:00	18:00			8,0	Documentation and testing
Friday 05.04.2019					0,0	
Saturday 06.04.2019					0,0	
Sunday 07.04.2019					0,0	
Total amount of hours:					**31,0**	
Week 15						
Monday 08.04.2019	10:00	16:00			6,0	Documentation and testing
Tuesday 09.04.2019	10:00	18:00			8,0	Documentation and testing
Wednesday 10.04.2019	10:00	17:00			7,0	Documentation: Main report and status report
Thursday 11.04.2019	10:00	14:00			4,0	Documentation: Main report
Friday 12.04.2019	10:00	21:30			11,5	Documentation and a lot of testing. Finishing the project.
Saturday 13.04.2019					0,0	
Sunday 14.04.2019					0,0	
Total amount of hours:					**36,5**	
Total project hours:					156,8	

Timesheet for Tobias

Week 7	Start	End	Start	End		Description
Monday 11.02.2019					0,0	
Tuesday 12.02.2019					0,0	
Wednesday 13.02.2019	900	1000			1,0	Kickoff, brainstorming and collaboration agreement
Thursday 14.02.2019	12:00	13:30			1,5	Translation of collaboration agreement to english
Friday 15.02.2019					0,0	
Saturday 16.02.2019					0,0	
Sunday 17.02.2019					0,0	
Total amount of hours:					**2,5**	
Week 8 - Studyweek	Start	End	Start	End		Description
Monday 18.02.2019	900	1200			3,0	Board Class
Tuesday 19.02.2019					0,0	
Wednesday 20.02.2019	900	1200	1300	1400	4,0	Piece Class
Thursday 21.02.2019					0,0	
Friday 22.02.2019					0,0	
Saturday 23.02.2019					0,0	
Sunday 24.02.2019					0,0	
Total amount of hours:					**7,0**	
Week 9	Start	End	Start	End		Description
Monday 25.02.2019					0,0	
Tuesday 26.02.2019					0,0	
Wednesday 27.02.2019					0,0	
Thursday 28.02.2019					0,0	
Friday 01.03.2019					0,0	
Saturday 02.03.2019					0,0	
Sunday 03.03.2019					0,0	
Total amount of hours:					**0,0**	
Week 10	Start	End	Start	End		Description
Monday 04.03.2019					0,0	
Tuesday 05.03.2019					0,0	
Wednesday 06.03.2019	900	1200			3,0	GameLogic
Thursday 07.03.2019	1100	1700			6,0	GameLogic
Friday 08.03.2019					0,0	
Saturday 09.03.2019					0,0	
Sunday 10.03.2019					0,0	
Total amount of hours:					**9,0**	
Week 11	Start	End	Start	End		Description
Monday 11.03.2019	1000	1600			6,0	GameLogic
Tuesday 12.03.2019	1000	1700			7,0	GameLogic
Wednesday 13.03.2019	1200	1600			4,0	GameLogic
Thursday 14.03.2019					0,0	
Friday 15.03.2019					0,0	
Saturday 16.03.2019					0,0	
Sunday 17.03.2019					0,0	
Total amount of hours:					**17,0**	
Week 12	Start	End	Start	End		Description
Monday 18.03.2019					0,0	
Tuesday 19.03.2019	1000	1800			8,0	Optimizing GameLogic
Wednesday 20.03.2019	900	1700			8,0	Documentation (Database & domain model)
Thursday 21.03.2019	1200	1500			3,0	Optimizing GameLogic
Friday 22.03.2019					0,0	
Saturday 23.03.2019					0,0	
Sunday 24.03.2019					0,0	
Total amount of hours:					**19,0**	
Week 13	Start	End	Start	End		Description
Monday 25.03.2019	1000	1600			6,0	Documentation
Tuesday 26.03.2019	1000	1400			4,0	Documentation
Wednesday 27.03.2019					0,0	
Thursday 28.03.2019	1000	1800			8,0	JavaFX(create ELO calculation and graph for player profiles)
Friday 29.03.2019					0,0	
Saturday 30.03.2019					0,0	
Sunday 31.03.2019					0,0	
Total amount of hours:					**18,0**	
Week 14	Start	End	Start	End		Description
Monday 01.04.2019	1000	1800			8,0	JavaFX (adding lists of moves on the game screen)
Tuesday 02.04.2019	1000	1800			8,0	JAVAFX (minor fixes, and adding lists of moves on the game screen)
Wednesday 03.04.2019	1200	1800			6,0	Testing
Thursday 04.04.2019	1000	1600			6,0	JavaDoc
Friday 05.04.2019	1000	1700			7,0	JavaDoc
Saturday 06.04.2019					0,0	
Sunday 07.04.2019					0,0	
Total amount of hours:					**35,0**	
Week 15	Start	End	Start	End		Description
Monday 08.04.2019	1000	1800			8,0	JavaDoc and database
Tuesday 09.04.2019	1000	1700			7,0	Usertests, preperation of final delivery
Wednesday 10.04.2019	1000	1700			7,0	Main report
Thursday 11.04.2019	1000	1500			5,0	Main report
Friday 12.04.2019	900	1100			2,0	Time Sheets, Performance evaluation
Saturday 13.04.2019					0,0	
Sunday 14.04.2019					0,0	
Total amount of hours:					**29,0**	
Total Prosjekttid:					136,5	

Timesheet for Jonas

Week 7	Start	End	Start	End		Description
monday 11.2.19					0,0	
tuesday 12.2.19					0,0	
wednesday 13.2.19	08:15	10:00	12:00	13:30	3,3	Kickoff, brainstorming and work agreement
thursday 14.2.19	12:00	13:30			1,5	Work agreement tasks
fredag 15.2.19					0,0	
lørdag 16.2.19					0,0	
søndag 17.2.19					0,0	
Totalt antall timer:					**4,8**	
Week 8	Start	End	Start	End		Description
monday 18.2.19					0,0	
tuesday 19.2.19					0,0	
wednesday 20.2.19	kl. 12.00	kl. 13.00			1,0	Miscellanious, brainstorming
thursday 21.2.19	08:50	09:50			1,0	Lage timelisteoppsett
friday 22.2.19	12:30	14:30	17:30	18:30	3,0	Vision document and some coding work
saturday 23.2.19					0,0	
sunday 24.2.19					0,0	
Total hours:					**5,0**	
Week 9	Start	End	Start	End		Description
monday 25.2.19					0,0	
tuesday 26.2.19					0,0	
wednesday 27.2.19	12:00	14:00			2,0	Work on vision document
thursday 28.2.19	20:30	22:10			1,7	Finishing vision document
friday 1.3.19					0,0	
saturday 2.3.19					0,0	
sunday 3.3.19					0,0	
Total hours:					**3,7**	
Week 10	Start	End	Start	End		Description
monday 4.3.19					0,0	
tuesday 5.3.19	10:15	16:15			6,0	
wednesday 6.3.19					0,0	
thursday 7.3.19	11:30	kl. 13.15			1,8	
friday 8.3.19	13:45	18:30			4,8	
saturday 9.3.19					0,0	
sunday 10.3.19					0,0	
Total hours:					**12,5**	
Week 11	Start	End	Start	End		Description
monday 11.3.19					0,0	
tuesday 12.3.19	10:00	15:00	15:10	16:30	6,3	
wednesday 13.3.19	11:30	19:00			7,5	
thursday 14.3.19					0,0	
friday 15.3.19	14:30	18:00			3,5	
saturday 16.3.19					0,0	
sunday 17.3.19	22:00	23:00			1,0	Troubleshooting
Total hours:					**18,3**	
Week 12	Start	End	Start	End		Description
monday 18.3.19	10:15	17:15			7,0	
tuesday 19.3.19	kl. 10.15	18:45			8,5	
wednesday 20.3.19	12:15	18:45			6,5	Chat system, some database work.
thursday 21.3.19	15:00	22:30			7,5	Testing, troubleshooting and JUnit testing
friday 22.3.19					0,0	
saturday 23.3.19					0,0	
sunday 24.3.19					0,0	
Total hours:					**29,5**	
Week 13	Start	End	Start	End		Description
monday 25.3.19	12:00	17:00			5,0	
tuesday 26.3.19	10:00	16:00			6,0	
wednesday 27.3.19					0,0	
thursday 28.3.19	kl. 11.00	kl. 17.00			6,0	
friday 29.3.19					0,0	
saturday 30.3.19					0,0	
sunday 31.3.19					0,0	
Total hours:					**17,0**	
Week 14	Start	End	Start	End		Description
monday 1.4.19	10:45	17:00	21:30	23:00	7,8	
tuesday 2.4.19	11:00	18:00			7,0	
wednesday 3.4.19	12:00	21:00			9,0	
thursday 4.4.19	11:00	16:00			5,0	
friday 5.4.19	10:30	12:00	18:00	19:00	2,5	
saturday 6.4.19					0,0	
sunday 7.4.19					0,0	
Total hours:					**31,3**	
Week 15	Start	End	Start	End		Description
monday 8.4.19	10:30	15:00			4,5	Brainstorming, problem solving, documentation
tuesday 9.4.19	10:15	16:00			5,8	Documentation
wednesday 10.4.19	11:00	15:00			4,0	
thursday 11.4.19	12:00	16:00			4,0	
friday 12.4.19	12:00	22:00			10,0	
saturday 13.4.19					0,0	
sunday 14.4.19					0,0	
Total hours:					**28,3**	
Total hours spent on project:					150,3	

Timesheet for Kjerand

						Activity
Week 7						
Monday 11.02.2019					0.0	
Tuesday 12.02.2019					0.0	
Wednesday 13.02.2019	08:00	10:00	12:00	13:30	3.5	Kickoff, brainstorming and collaboration agreement.
Thursday 14.02.2019	12:00	13:30			1.5	Translation of collaboration agreement to english
Friday 15.02.2019					0.0	
Saturday 16.02.2019					0.0	
Sunday 17.02.2019					0.0	
Total amount of hours:					**5.0**	
Week 8 - Studyweek						
Monday 18.02.2019					0.0	
Tuesday 19.02.2019					0.0	
Wednesday 20.02.2019	10:00	13:00			3.0	Coding game logic
Thursday 21.02.2019					0.0	
Friday 22.02.2019					0.0	
Saturday 23.02.2019					0.0	
Sunday 24.02.2019					0.0	
Total amount of hours:					**3.0**	
Week 9						
Monday 25.02.2019					0.0	
Tuesday 26.02.2019					0.0	
Wednesday 27.02.2019	12:00	16:00			4.0	Vision document
Thursday 28.02.2019	09:00	12:00			3.0	Coding game logic
Friday 01.03.2019					0.0	
Saturday 02.03.2019					0.0	
Sunday 03.03.2019					0.0	
Total amount of hours:					**7.0**	
Week 10						
Monday 04.03.2019	09:00	12:00			3.0	Modeling.
Tuesday 05.03.2019	09:00	15:00			6.0	Modeling and coding logic.
Wednesday 06.03.2019	09:00	10:00			1.0	Meeting planning.
Thursday 07.03.2019	10:00	16:00			6.0	Coding game logic.
Friday 08.03.2019	15:00	16:00			1.0	Modeling database.
Saturday 09.03.2019					0.0	
Sunday 10.03.2019					0.0	
Total amount of hours:					**17.0**	
Week 11						
Monday 11.03.2019					0.0	
Tuesday 12.03.2019	09:00	18:30			9.5	Coding logic for pawns, rook, bishop and knight.
Wednesday 13.03.2019	08:30	16:00			7.5	Coding logic for check and en passant.
Thursday 14.03.2019					0.0	
Friday 15.03.2019					0.0	
Saturday 16.03.2019	16:00	19:00			3.0	Coding logic for check and en passant.
Sunday 17.03.2019	17:00	18:00			1.0	Coding logic for check and en passant.
Total amount of hours:					**21.0**	
Week 12						
Monday 18.03.2019	09:00	19:00			10.0	Added graphics.
Tuesday 19.03.2019	09:00	20:00			11.0	Working with database connection.
Wednesday 20.03.2019	09:00	18:00			9.0	Working with queue system.
Thursday 21.03.2019	09:00	12:00			3.0	Testing.
Friday 22.03.2019					0.0	
Saturday 23.03.2019					0.0	
Sunday 24.03.2019					0.0	
Total amount of hours:					**33.0**	
Week 13						
Monday 25.03.2019	10:00	16:30			6.5	Fixed database-sync and had team meeting.
Tuesday 26.03.2019	08:30	17:00			8.5	Added more game-modes.
Wednesday 27.03.2019					0.0	
Thursday 28.03.2019	10:00	18:30			8.5	Added castling and en passant.
Friday 29.03.2019	10:00	12:00			2.0	Added ability to invite friends.
Saturday 30.03.2019					0.0	
Sunday 31.03.2019					0.0	
Total amount of hours:					**25.5**	
Week 14						
Monday 01.04.2019	10:00	18:00			8.0	Added ability to send remis.
Tuesday 02.04.2019	10:00	18:00			8.0	Added timer.
Wednesday 03.04.2019	09:30	18:30			9.0	Added moves to gamescene and timer
Thursday 04.04.2019	10:00	14:00			4.0	Cleaning folders.
Friday 05.04.2019					0.0	
Saturday 06.04.2019					0.0	
Sunday 07.04.2019					0.0	
Total amount of hours:					**29.0**	
Week 15						
Monday 08.04.2019	10:00	19:00			9.0	Documentation.
Tuesday 09.04.2019	10:00	17:00			7.0	
Wednesday 10.04.2019	10:00	12:00			2.0	Documentation, working on main report.
Thursday 11.04.2019					0.0	
Friday 12.04.2019	10:00	12:00	20:00	22:00	4.0	Documentation, finishing report
Saturday 13.04.2019					0.0	
Sunday 14.04.2019					0.0	
Total amount of hours:					**22.0**	
Total hours					162.5	

Timesheet for Håvard

Week 7						
Monday 11.02.2019					0,0	
Tuesday 12.02.2019					0,0	
Wednesday 13.02.2019	11:00	14:00			3,0	Collaboration contract
Thursday 14.02.2019					0,0	
Friday 15.02.2019					0,0	
Saturday 16.02.2019					0,0	
Sunday 17.02.2019					0,0	
Total amount of hours:					**3,0**	
Week 8 - Studyweek						
Monday 18.02.2019	09:30	12:00			2,5	User stories
Tuesday 19.02.2019					0,0	
Wednesday 20.02.2019	09:00	12:00			3,0	User stories
Thursday 21.02.2019					0,0	
Friday 22.02.2019					0,0	
Saturday 23.02.2019					0,0	
Sunday 24.02.2019					0,0	
Total amount of hours:					**5,5**	
Week 9						
Monday 25.02.2019	16:00	18:00			2,0	modelling
Tuesday 26.02.2019					0,0	
Wednesday 27.02.2019					0,0	
Thursday 28.02.2019					0,0	
Friday 01.03.2019					0,0	
Saturday 02.03.2019					0,0	
Sunday 03.03.2019					0,0	
Total amount of hours:					**2,0**	
Week 10						
Monday 04.03.2019					0,0	
Tuesday 05.03.2019	10:00	12:00	15:00	kl. 16.00.00	3,0	chess, gameengine-class
Wednesday 06.03.2019	09:00	10:00	13:00	kl. 17.00.00	5,0	Board class, studying javaFX
Thursday 07.03.2019	10:45	16:00			5,3	javaFX chess demo
Friday 08.03.2019	16:00	18:00	19:00	kl. 21.00.00	4,0	Chess demo programming
Saturday 09.03.2019					0,0	
Sunday 10.03.2019					0,0	
Total amount of hours:					**17,3**	
Week 11						
Monday 11.03.2019					0,0	
Tuesday 12.03.2019	10:00	18:30			8,5	Chessdemo, graphics
Wednesday 13.03.2019	10:30	19:00			8,5	implementing multiplayer
Thursday 14.03.2019					0,0	
Friday 15.03.2019					0,0	implementing multiplayer
Saturday 16.03.2019	13:00	14:00			1,0	implementing multiplayer
Sunday 17.03.2019					0,0	
Total amount of hours:					**18,0**	
Week 12						
Monday 18.03.2019	10:00	18:30			8,5	implemented skins, modeled database
Tuesday 19.03.2019	10:00	20:00			10,0	queue system
Wednesday 20.03.2019	11:00	20:00			9,0	skins and making it possible for the opponent to see your skin
Thursday 21.03.2019	12:00	14:00			2,0	testing
Friday 22.03.2019					0,0	
Saturday 23.03.2019					0,0	
Sunday 24.03.2019					0,0	
Total amount of hours:					**29,5**	
Week 13						
Monday 25.03.2019	10:00	17:00			7,0	team meeting, fixed polling ingame, stats update on profile
Tuesday 26.03.2019	10:00	17:00			7,0	worked with database classes, updating elo
Wednesday 27.03.2019	11:00	14:00			3,0	database classes, elo stuff
Thursday 28.03.2019	10:30	18:30			8,0	testing, fixing of stuff
Friday 29.03.2019	10:30	11:30			1,0	testing
Saturday 30.03.2019					0,0	
Sunday 31.03.2019					0,0	
Total amount of hours:					**26,0**	
Week 14						
Monday 01.04.2019	11:00	19:00			8,0	Draw function, friend invite, database methods
Tuesday 02.04.2019	10:30	19:00			8,5	timer ingame, stats, skin
Wednesday 03.04.2019	13:00	22:00			9,0	testing
Thursday 04.04.2019	16:00	17:00			1,0	cleaned program
Friday 05.04.2019	15:00	17:00	19:00	20:00	3,0	testing, javadoc
Saturday 06.04.2019					0,0	
Sunday 07.04.2019					0,0	
Total amount of hours:					**29,5**	
Week 15						
Monday 08.04.2019	10:00	20:00			10,0	javadoc
Tuesday 09.04.2019	10:00	18:00			8,0	documentation, reports
Wednesday 10.04.2019	10:00	17:00			7,0	reports
Thursday 11.04.2019	11:00	17:00			6,0	Main report
Friday 12.04.2019	12:00	16:00			4,0	Main report
Saturday 13.04.2019					0,0	
Sunday 14.04.2019					0,0	
Total hours:					**35,0**	
Total Prosjekttid:					165,8	

Status report for Martin

Week 7:

- Tasks completed:
  - Finished collaboration agreement
- Tasks doing:
  - N/A
- Problems during this week:
  - N/A

Week 8:

- Tasks completed:
  - Brainstorming ideas
- Tasks doing:
  - Create timesheet, and learn JavaFX
- Problems during this week:
  - N/A

Week 9:

- Tasks completed:
  - Created timesheet and watched a lot of Buckys videos about JavaFX.
- Tasks doing:
  - Vision Document
  - Begin creating first JavaFX-classes
- Problems during this week:
  - JavaFX and GridPanes (me being non-experienced)

Week 10:

- Tasks completed:
  - Finished first draft of vision document.
- Tasks doing:
  - Further work on Login and JavaFX
  - Redesign
  - New MainScene
- Problems during this week:
  - Switching scenes, from different classes

Week 11:

- Tasks completed:
  - Login-function
- Tasks doing:
  - Create new scenes
  - GUI
  - Cleanup of database-connections
- Problems during this week:
  - Database connection is working but it is quite slow!

Week 12:

- Tasks completed:
    - Switching between scenes from different classes.
    - Implementation of sandbox chess game
- Tasks doing:
    - Redesign of MainScene, new background and structure
    - Clearboard-button fix and create settings-scene
- Problems during this week:
    - Sandbox not working 100%

Week 13:

- Tasks completed:
    - Sandboxgame and functional main screen
- Tasks doing:
    - Create new game- and join game-popupbox.
    - Fix gamequeue and matchmaking
- Problems during this week:
    - GIT!

Week 14:

- Tasks completed:
    - Matchmaking now working!
- Tasks doing:
    - Fix mainscene to optimalize scene-change. Use same scene just replace grids
    - Create more methods for cleaner mainScene-class.
    - Finish GIT Wiki
- Problems during this week:
    - GIT again, do think we should have used more branches.

Week 15:

- Tasks completed:
    - Documentation to be finished this week
- Tasks doing:
    - Documentation
- Problems during this week:
    - N/A

Performance:

I do believe I have been an important part of this project, along with the other members. I started out with getting the responsibility of the GUI/JavaFx-part of the project, creating the interface that the user will see. I am very happy with how it turned out, and do think it is a good-looking application, even though we did not use CSS nor FXML. If I should have done something differently, I would have taken a larger role in the implementation of the chess-moves. We did have to split up the work though, for using less time developing as the team gathered experience in different fields. I have learnt to use git as a collaborative tool to develop code, learned to implement graphics, learned some backend, and to work with a team on a large project. Overall, I think that we have delivered a good application, and I am happy with the work we have done as a team.

Status report for Dilawar

Week 7:

- Tasks completed:
  - Collaboration agreement
- Tasks doing:
  - N/A
- Problems during this week:
  - Must translate collaboration agreement to english.

Week 8:

- Tasks completed:
  - Brainstorming
- Tasks doing:
  - Game logic, pieces and board
- Problems during this week:
  - N/A

Week 9:

- Tasks completed:
  - Finished first version of vision document.
- Tasks doing:
  - Game logic and modeling
- Problems during this week:
  - N/A

Week 10:

- Tasks completed:
  - Database model
- Tasks doing:
  - Modeling database.
  - Worked with game logic.
- Problems during this week:
  - Could optimize game logic

Week 11:

- Tasks completed:
  - Game logic
- Tasks doing:
  - Modeling
  - Working with Chess Demo and special moves
- Problems during this week:
  - Some bugs in the game logic

Week 12:

- Tasks completed:

        - Fixed some of the game logic
        - Testing.
- Tasks doing:
    - Timer.
- Problems during this week:
    - En passant problem.

Week 13:

- Tasks completed:
    - Game logic done
- Tasks doing:
    - Timer
- Problems during this week:
    - Timer synchronization

Week 14:

- Tasks completed:
    - Added timer to the game.
- Tasks doing:
    - Modeling.
    - Documentation.
    - Writing out moves
- Problems during this week:
    - Problems with draw and resign.

Week 15:

- Tasks completed:
    - Finished documentation.
- Tasks doing:
    - Main report
    - Vision document
    - Testing
- Problems during this week:
    - N/A

Performance:

I would say that my performance has been pretty good, and I have contributed a lot to the project. I must say that I have worked the most with the game logic, and how the chess system behaves, and implementing the rules in the game. I have also worked a little with JavaFX, and have built some components on the game scene, like the timer and move-table. I wish that I could have been more active on the backend-side of the application, working more with the database. One of my regrets with this application was that we did not have the time to implement a chess bot, because that would have been pretty cool. After this project I have learned to use git as a collaborative tool to develop code, learned to implement graphics, learned some backend, and to work with a team on a large project. Overall, I think that we have delivered a good application, and I am happy with the work we have done as a team

Status report for Tobias

Status report for Tobias

Week 7:

- Tasks completed:
  - Create collaboration contract
- Tasks doing:
  - N/A
- Problems during this week:
  - N/A

Week 8:

- Tasks completed:
  - Making user stories
  - Brainstorming ideas for chess app
  - Piece class
  - Board class
- Tasks doing:
  - Planning project
- Problems during this week:
  - Deciding on how to represent the board was somewhat problematic as we had to change it later.

Week 9:

- Tasks completed:
  - Finished first version of vision document.
- Tasks doing:
  - Game logic and modeling
- Problems during this week:
  - N/A

Week 10:

- Tasks completed:
  - Database model
- Tasks doing:
  - Modeling database.
  - Worked with game logic.
- Problems during this week:
  - Could optimize game logic

Week 11:

- Tasks completed:
  - All the rules of game in GameLogic
- Tasks doing:
  - Optimize and shorten the code of GameLogic
- Problems during this week:
  - N/A

Week 12:

- Tasks completed:
    - Optimize GameLogic
    - Database Diagram
    - Domain Diagram
- Tasks doing:
    - Documentation
    - Working with the gitlab wiki
- Problems during this week:
    - N/A

Week 13:

- Tasks completed:
    - ELO graph in JavaFX
    - meeting reports
    - update the Database & domain models
- Tasks doing:
    - Adding lists of moves to game screen
- Problems during this week:
    - To add the ELO graph I had to add another table in the database so that many elo values (all the values achieved by a player) could be connected to the same account. This mean that I also had to change the models I made last week.

Week 14:

- Tasks completed:
    - Adding lists of move to game screen
- Tasks doing:
    - Writing Javadoc
    - testing
    - minor fixes in Javafx
- Problems during this week:
    - I need a lot of help when I had already created the list of moves, on how to show it in a nice way next to the chessboard. The testing was also somewhat hard as bug recreation was really tedious.

Week 15:

- Tasks completed:
    - Javadoc
    - Reports and documentation
- Problems during this week:
    - N/A

Performance

All in all, I think my contribution to the team and performance was decent, but nothing special. From time to time it was unclear to me how I should contribute the most and spending over a month on the same project I grew somewhat tired of it by the end. That being said I have mainly contributed to the GameLogic class. I have written a big chunk of the logic itself, and also shortened my own and others' code in that class greatly. Additionally, I also wrote the ELO calculation, the ELO graph and the move history in the game. Something I would have done differently if i restarted the project today was to try and write more code in topics I am not familiar with, I did too many things that I knew how to do before this course began.

Status report for Jonas

Week 7:

- Tasks completed:
  - Collaboration agreement
- Tasks doing:
  - --
- Problems during this week:
  - Writing a proper and sufficient collaboration agreement.

Week 8:

- Tasks completed:
  - Brainstorming
  - Some work on the time-sheets
- Tasks doing:
  - Game logic and board
- Problems during this week:
  - --

Week 9:

- Tasks completed:
  - Finished first version of the vision document.
- Tasks doing:
  - Looking into timers and some JavaFX
  - Looking into some hashing with salt/working with Martin on making the hashing work.
  - Understanding salt-hashing
- Problems during this week:
  - Understanding how to implement timers in the background

Week 10:

- Tasks completed:
  - Made a timer class with various operations for future use.
  - Hashing completed
- Tasks doing:
  - Still experimenting with some timer-tasks and lambda expressions
  - Looking into connection pooling
  - Setting up a database class to handle sql-queries. (including prepared statements for user-input)
  - Looking into threads in conjunction with JavaFX and playing multiplayer
- Problems during this week:
  - Understanding connection-pooling

Week 11:

- Tasks completed:
  - Implemented connection-pooling with HikariCP
  - Database class completed (some optimization could be required)
- Tasks doing:

        - Making a better database-helper
        - Working on an in-game chat
        - Still looking into some threading/services
- Problems during this week:
    - Implementing the connection pool and making a decent database-helper.

Week 12:

- Tasks completed:
    - Database-helper has been improved
- Tasks doing:
    - Still working on the chat. Giving me some problems with messages being fetched several times from the database.
- Problems during this week:
    - Slow database

Week 13:

- Tasks completed:
    - Chat working!
- Tasks doing:
    - Looking more into timer/removing the old one
    - Separating JavaFX from database calls. Making them into separate classes, making the code more readable.
    - Cleaning up/restructuring code in the project
- Problems during this week:
    - Making everything work as it did while maintaining relatively clean code

Week 14:

- Tasks completed:
    - Finished cleaning up code
    - Separated JavaFX from database calls
    - Fixed a major flaw in the database helper (big misunderstanding on my part on how to use prepared statements.)
- Tasks doing:
    - Modeling.
    - Documentation.
    - Writing out moves
    - Fixing some test classes that don't work anymore after some changes in the source code.
- Problems during this week:
    - Problems with draw and resign.
    - Big flaw in database helper could potentially lead to SQL-injection vulnerability

Week 15:

- Tasks completed:
    - Finished documentation.
    - Main report
- Tasks doing:

    - Testing
- Problems during this week:
    - Making the jar file work on some computers

Performance:

In my opinion, my performance has been decent. I have not been overwhelmed with tasks, neither have I been without tasks. I was for some reason stuck on fixing an error in the chat for the longest time, making my performance in that particular week suboptimal. Other than that I feel like I have earned my place on the team. I may not be as experienced as others on this team, but I still feel like I was able to keep up with what was demanded of me. The moments where I had little to do, I took on sort of a consultant role, trying to look at problems other team-members had and come with (hopefully) helpful suggestions and ideas.

All in all, I feel like my overall performance and efforts have been adequate.

Status report for Kjerand

Week 7:

- Tasks completed:
  - Collaboration agreement
- Tasks doing:
  - N/A
- Problems during this week:
  - N/A

Week 8:

- Tasks completed:
  - Brainstorming ideas
- Tasks doing:
  - Project planning
- Problems during this week:
  - N/A

Week 9:

- Tasks completed:
  - First draft of vision document
- Tasks doing:
  - Working on diagrams
  - Working on the game logic
- Problems during this week:
  - N/A

Week 10:

- Tasks completed:
  - Database model
- Tasks doing:
  - Working on modeling database.
  - Working on game logic.
- Problems during this week:
  - Problems with the highlight boxes on the board (graphics)

Week 11:

- Tasks completed:
  - Main part of the game logic
- Tasks doing:
  - Working on modeling.
  - Working with logic for pawns, rook, bishop and knight.
- Problems during this week:
  - Database connection is working but it is quite slow.

Week 12:

- Tasks completed:

        - Queue system in the game.
        - Made database connection faster
    - Tasks doing:
        - Working on queue system and game logic.
    - Problems during this week:
        - Problems with the special move en passant

Week 13:

- Tasks completed:
    - More game modes
    - Friend invites
- Tasks doing:
    - Working on adding more features to the game (resign/remis and invites)
- Problems during this week:
    - Fixed problem with ending game/resigning.

Week 14:

- Tasks completed:
    - Ability to send remis and resign game.
    - Game timer
- Tasks doing:
    - Working on some documentation
- Problems during this week:
    - Problems with the offering draw system
    - Problems with timer not being synced with other player

Week 15:

- Tasks completed:
    - Documentation
- Tasks doing:
    - Working on main report
- Problems during this week:
    - N/A

Performance:

I am happy with my performance during this project. I worked a decent amount with the project and I think I got quite a lot done. I did a lot of the work on the logic behind the game and on special cases like castling, promotion and en passant. Other cases I added was checkmate, stalemate, not enough pieces to checkmate and 50 moves without a piece is taken. I also worked with some board graphics and the game timer. In addition I added the different features for the game, like different game modes, setting a timer and increment, choosing color and whether the game is rated, and more. I worked quite a lot on the queuing system and the invite system. I also did some work on the graphics for the user interface and I worked on the services for polling from the database. For the documentation I did quite a lot on the database model, the class diagram, the Gantt-diagram, the sequence diagram and the vision document.

I am very happy with the finished project and all the features we managed to add in such a short timespan. One thing that could be improved for next time is the orderliness of the code. We put a larger emphasis on adding new features and getting the program working as fast as possible, rather than using time on making the code cleaner and more orderly. We also had some problems with Git, which we could have avoided.

Status report for Håvard

Week 7:

Tasks completed:
- Create collaboration contract

Tasks doing:
- N/A

Problems during this week:
- N/A

Week 8:

Tasks completed:
- Making user stories
- Brainstorming ideas for chess app

Tasks doing:
- Planning project

Problems during this week:
- N/A

Week 9:

Tasks completed:
- Class diagram
- Vision document

Tasks doing:
- Game logic, chess classes.

Problems during this week:
- N/A

Week 10:

Tasks completed:
- Game logic classes

Tasks doing:
- Studying JavaFX, making functional graphical chess board that interacts with the interface for the game logic classes.

Problems during this week:
- N/A

Week 11:

Tasks completed:
- Creating a graphical, interactable chess board that uses the game logic to operate.

Tasks doing:
- Implementing additional chess rules to the board; promotion, castle, en passant.
- Adding skin system.
- System for multiplayer, and database.

Problems during this week:
- Threading issues with uploading and receiving moves. The polling system for checking if the opponent is making a move has to be on another thread than the

main thread in order for the game to be responsive while playing, but the other thread is not allowed to make changes to the graphical board.
    - Solved by using a timer that runs a service that runs a poll on the database in an interval. The service is allowed to make a change on the main javafx thread while the program stays responsive.

Week 12:

Tasks completed:

- Sequence diagram
- Skin system
- Implementing multiplayer play; database structure and system for sending and receiving moves within the game, syncing chess boards.

Tasks doing:

- Optimizing polling.
- Separating database queries from javaFX classes.

Problems during this week:

- Skins not showing for enemy.
    - Solved issue by saving the name of the skin in a “userSettings” table in the database, and assigning skins locally based on what skin name the opponent has on their settings.

Week 13:

Tasks completed:

- Optimized polling for checking if opponent made a move.
- Database classes, for abstracting the queries from the javafx-classes.

Tasks doing:

- Implementing draw offers.

Problems during this week:

- Errors when polling too fast, Game stops checking for moves after a while.
    - Solved by using booleans to make sure a new service that polls is not started before the next is finished, game now runs smoothly and without errors.

Week 14:

Tasks completed:

- Draw offers
- Cooperated on implementing and 100% optimizing timer
- Fixed player statistics

Tasks doing:

- Writing Javadoc

Problems during this week:

- Timer was not synced on both player’s side
    - Solved by uploading a timestamp every time someone makes a move and having the recipient update its enemy’s timer with the timestamp they received.

Week 15:

Tasks completed:

- Javadoc
- Reports and documentation

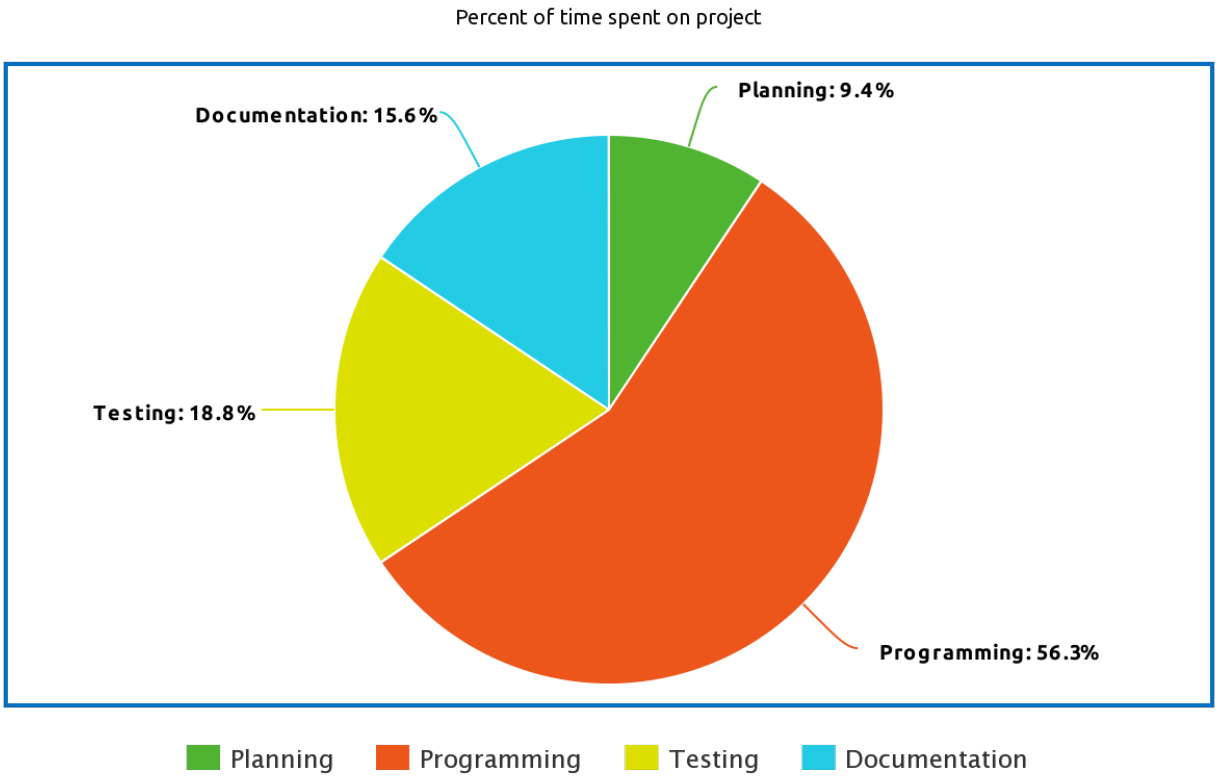
Problems during this week:

- N/A

Performance:
I am happy with my effort and the result of my actions in this project. I have been motivated and eager to work on this project during the entire timespan of the project. What I would do differently is to plan more thoroughly and divide tasks in a more structured manner, as on this project the tasks were mostly done and taken “on the fly”. Lack of structure has resulted in several minor problems with git/version control but has mostly not affected the result too much. My code has not been very readable for others during the project, and this is something I would like to improve and do better next time.

Division of labor

Here you see a pie-chart showing the percentage of the total amount of hours spent on the project, divided into the 4 different stages we have had:



Attachment 4: Meeting invitations and minutes

First team meeting invitation:

Notice of meeting for Team 20

6. mars 2019

Notice of meeting pertains to the following participants:

Tobias Meyer Andersen, Kjerand Evje, Ole Jonas Liahagen, Dilawar Mahmood, Håvard Stavnås Markhus, Martin Johannes Nilsen, Grethe Sandstrak, Nils Pettersen Tesdal

TIME AND PLACE: Thursday 07.03.2019, 15:50 – 16:15. Meeting is held at TBM 401.

Plan for meeting: Review and feedback on vision-document. Go through demo of first prototype.

Agenda:

Case nr:	Case	Time	Responsible
1	Confirmation of meeting	2 min	Meeting leader
2	Review of vision-document.	10 min	-
3	Demo of first prototype.	10 min	-
4	Eventual questions.	3 min	-

The meeting is held without any breaks.

Enclosed in the email is a copy of the vision-document.

If you are unable to meet contact the meeting organizer (Kjerand Evje, kjerande@stud.ntnu.no).

Best regards,

Kjerand Evje

Second team meeting invitation:

Notice of meeting for Team 20

21. mars 2019

Notice of meeting pertains to the following participants:

Tobias Meyer Andersen, Kjerand Evje, Ole Jonas Liahagen, Dilawar Mahmood, Håvard Stavnås Markhus, Martin Johannes Nilsen, Grethe Sandstrak, Nils Pettersen Tesdal

TIME AND PLACE: Monday 25.03.2019, 15:50 – 16:15. Meeting is held at TBM T04.541

Plan for meeting: Review and feedback on Gitlab Wiki. Present a demo of our prototype.

Agenda:

Case nr:	Case	Time	Responsible
1	Confirmation of meeting	2 min	Meeting leader
2	Review of Gitlab Wiki	10 min	-
3	Demo of prototype	10 min	-
4	Eventual questions.	3 min	-

The meeting is held without any breaks.

If you are unable to meet contact the meeting organizer (Kjerand Evje, kjerande@stud.ntnu.no).

Best regards,

Kjerand Evje

Report after the first meeting:

Team meeting (25.03.2019)

Place: TBM T04.541.
Present: Grethe, Jonas, Martin, Håvard, Dilawar, Kjerand, Tobias

Case 01, Meeting approval:

- Plan: review wiki, show demo

Case 02, gitlab wiki:

- Don't show update move to database or send message as their own use-case in the use-case diagram. It's too specific.
- Use extends instead of includes in the main menu use-case diagram.
- Add domain model
- Add vision document
- Use Javadoc for source code

Case 03, demo:

- Very good! (Extra points for doing something original with the skins)
- A few bugs are tolerated

Case 04, questions:

- Anything in particular we should add to the program?
    - The product is already good enough
- Is originality an important assessment criterium?
    - If the program is well made, you don't need a very original product.

End of meeting report

Report after the second meeting:

Team meeting (07.03.2019)

Place: TBE 404
Present: Grethe, Jonas, Martin, Håvard, Dilawar, Kjerand, Tobias

Case 1 - Vision document

- "double introduction", combine?
- Have a more specific target-audience, not just Grethe and Nils
- Remove "and more!", don't promise anything vague
- More effect-goals?
- Process-goal: could add "get better at using git" or "getting to know JavaFX better"
- Add the developers as stakeholders!
- Write that it will be evaluated at 4.5.1
- 4.7: include playability, how to find a game, what should the user be able to do in game?
- Chat?
- 5.4: Elaborate on the risks. How do you avoid said risks?
- 5.6: remove
- Product features: should reflect 4.7
- 6: foundation for user stories
- 10: Include the wiki-structure from tasks, password, connection pool, sql-injection, JUnit

Case 2 - Wireframe

- Add the features that are shown in the wireframe from the vision document!

End of meeting report

Attachment 5: Vision document

Team 20

Chess Project - Recess Chess Vision Document

Tobias Meyer Andersen, Kjerand Evje, Ole Jonas Liahagen, Dilawar Mahmood, Håvard Markhus, Martin Johannes Nilsen

Revision History

Date	Version	Description	Author
28/02/19	1.0	Finished first version of the document.	Kjerand, Tobias, Jonas, Dilawar, Martin, Håvard
26/03/19	1.1	Rewrote some parts after guidance from Grethe at the first meeting.	Martin, Dilawar, Jonas, Tobias, Kjerand

Table of Contents

1. Introduction

In the following months, we want to create a user-friendly online Chess game to fulfill the stakeholders needs of an online multiplayer game. The purpose of this document is to collect, analyze, and define high-level needs and features of the Chess Project. It focuses on the capabilities needed by the stakeholders and the target users, and why these needs exist. The details of how the Chess Project fulfills these needs are specified in the use-case-diagram and supplementary specifications. This vision document is associated with the Chess Project and all members of its developer team, Team 20. The document is divided into 12 chapters. Each of these will analyze a certain part of the project, for instance, constraints or risks.

2. Positioning

2.1 Business Opportunity

Our game provides a new online chess experience with unique twists, such as different game modes and different themes for your board and pieces. We believe this introduces a new experience standing out from the competition. This can be capitalized on and could introduce more sales opportunities.

2.2 Problem Statement

Problem	• Non-digitalized chess with physical chess boards can degrade and pieces are often lost. • Playing timed chess games usually also requires a separate clock to keep track of time. This can also be lost or just degrade over time. • Physical chess boards require opponents to be in the same location in order to play with each other. • Physical chess boards do not provide any hints as to where a given piece can move. Without a handbook, an inexperienced player could become confused.
Affects	The users of the system. In this case; our audience
The impact of which is	• Makes chess hard to play without previous experience. • Chess becomes hard to play for people living in separate places. • Equipment can degrade, making the game unplayable until a replacement is found. This could also result in increased expenses for the user.
A successful solution would be	A digitalized chess game which provides a timer and hints telling where each piece can move.

2.3 Product Position Statement

For	Board game enthusiast
Recess Chess	is a online chess game with different game-modes to challenge and entertain users.
That	is made by a group of students. It also has unique game modes not usually seen in traditional chess.
Unlike	Competitors like the physical board game or chess.com.
Our Product	Introduces a unique twist on a beloved classic. Extra features include: alternative game-modes, social chat system between opponents and a ranking system!

3. Project goals

3.1 Efficiency goals

- Increase the number of chess players in our class by 10%.
- Increase brand awareness by 300%.
- Earn money from the product (by adding ads or a onetime payment)

3.2 Result goals

- Introduce a web-based chess game with unique game-modes before May 2019.
- Present the finished product to the class.

3.3 Process goals

- Increase competency in database management.
- A in the subject
- Learn about JavaFX
- Learn/make use of Git
- Learn about security, with Salt Hashing and prepared statements
- Learn about threading in java

4. Stakeholder and User Descriptions

4.1 Market Demographics

We are not an established organization, therefore we do not have a reputation. Ideally we wish to be respected by the stakeholders. This product fulfills the requirements given by the stakeholders (make a turn-based game which communicates via a database).

4.2 Stakeholder Summary

Name	Description	Responsibilities
Employers • Grethe Sandstrak • Nils Tesdal	These employers are lecturers at NTNU.	Their responsibilities are to watch the project, have meetings with us, and provide resources and help with technical problems. They are also the ones assessing our final product.
Programmers • Dilawar Mahmood • Håvard Markhus • Kjerand Evje • Martin Nilsen • Ole Jonas Liahagen • Tobias Andersen	The programmers are students at NTNU	Their responsibilities are to create the game, test it and be prepared for the meetings and demands given to them by the employers.

4.3 User Summary

Name	Description	Responsibilities
Player	Clients, end-users	• Register themselves as users • Play/test the game • Give criticism and response

4.4 User Environment

The task of playing our game will be performed by two people at a time. The two users will be competing against each other for an indefinite amount of time. We will implement a timer for each player so that time eventually runs out if timed mode is chosen. The task cycle will therefore not be a set period of time. It will vary from cycle to cycle.

Our goal is to make the product free of constraints other than a required internet connection and a functional computer. Although unlikely, we might develop a version for the mobile platform later on to compete with rivals such as lichess.

4.5 Stakeholder Profiles

4.5.1 Clients

Representative	Grethe Sandstrak and Nils Tesdal
Description	University lectors
Type	Experienced lectors from NTNU. Unknown skill level and experience with chess.
Responsibilities	Help us with technical problems and provide guidance if needed.
Success Criteria	These stakeholders define success as a working game which communicates via a database. The stakeholders are rewarded by seeing and trying our online game.
Involvement	The stakeholders are involved by providing resources and helping us with technical problems. They will also be assessing the final product.

4.5.2 Programmers

Representative	Ole Jonas Liahagen, Dilawar Mahmod, Tobias Meyer Andersen, Håvard Stavnås Markhus, Martin Johannes Nilsen, Kjerand Evje.
Description	Programmers/Students
Type	Programmers from NTNU.
Responsibilities	Plan, create, document and test the application to be provided to the clients.
Success Criteria	The application must execute without bugs and give the user options to create a new user, pick an avatar, see own progress. The user must also be able to play either solo in sandbox-mode or against an opponent playing from another computer
Involvement	The programmers are involved by being the ones responsible for creating the program

4.6 User Profiles

4.6.1 Casual user

Representative	Family member, friend or classmate
Description	Beginner/intermediate-level chess player
Type	Casual user
Responsibilities	Produce feedback for the developers for use in future product development.
Success Criteria	Success for this user is defined by being able to choose a game mode and play it and always understanding how to operate the software for its most fundamental use.
Involvement	The role of this user will be to test the software after the MVP is made, and see if it is intuitive enough,
Deliverables	Written/oral feedback for the developer team.

4.6.2 Experienced user

Representative	Family member / classmate / friend
Description	Experienced online chess player
Type	This is a user that has spent many hours using various online chess applications, that will have more concrete demands from the applications behavior.
Responsibilities	Produce feedback to guide the developer team.
Success Criteria	This user will define chess as being able to play a chess game with specific restrictions and rules that satisfy the ruleset they desire.
Involvement	The role of this user will be to test the software after the MVP is made, and see if it is intuitive enough,
Deliverables	Written/oral feedback for the developer team.

4.7 Key Stakeholder or User Needs

Need	Priority	Concerns	Current solution	Proposed solutions
Database password-security	High	Passwords must be hashed for user-security reasons.	• Created a method returning a SHA-256 encrypted hex-string. • Created a method for generating a SALT for further encryption. • Created a method for combining a password and a SALT, and then encrypt this string. • Created method comparing two hex-strings to see if the user has entered a valid password.	Salt hashing
Opportunity to track progress and skill	Medium	Without a ranking system, some users could find it hard to track own progress and skill.	• Added an updateable elo-rating, with a graph presented at the user-profile screen. • Implemented a top-5 leaderboard.	Provide leaderboards and statistics stored in a mySQL database system which can then be transferred to something more visual in the final application.
Opportunity to create a game	High	Must be able to create a game so that other players can play chess against you.	• Created method for inserting your user-id to database with the criterias you wish to have in your game. • The game is set as active, and waits for another player to join, which in turn sets the game to inactive. This	Provide the possibility to add your user-id to a database with different criterias for the game you want to play.

			prevents other people from trying to join an ongoing game.	
Play game	High	Must be able to play the game. The main criteria of the assignment is to be able to play against each other.	• When you want to play a game, you click on the "new game" button. You are then presented with three options: create game, join game or invite friend. All of the three options presented will give you the opportunity to play a game.	The proposed solution is the same as the end-solution.
Join game	High	Must be able to join a game such that two players can play chess against each other	• When a player creates a game, their user-id along with game-criteria is sent to the database, and the game is set as active. • When a user wants to join a game, the user must match the criteria set by the creator of said game. • The user joins the game, and the game is set as inactive.	Provide possibility to join a game and choose criteria. You can either join a game that a user has created, or you can be invited to a game.
Register user	High	Must be able to register as a new user with their own username and password The password must be hashed with salt for security reasons.	• A user can create a user with a username and the password on the login screen. They can only register a username if the username does not already exist. The passwords are hashed with salt	Prove a UI where the user can register themselves with a username and a password in a textfield. The password should be hidden

4.8 Alternatives and Competition

4.8.1 Lichess

A popular online chess service. They provide the ability to play against your friends online. They also provide many other useful features like ELO-ratings, time restrictions, tactics and more.

4.8.2 Physical Chess Board

An alternative to our service is playing chess on a physical board. The disadvantage of this is not being able to play against each other over long distances. The advantage is that it is really easy to start playing and you get to play with real pieces.

4.8.3 Chess.com

Another popular online chess service. Provides different game-modes such as 4-player chess and the opportunity to play against friends.

4.8.4 Magnus Carlsen game on iPhone

A popular game on iPhone were you can allegedly play against Magnus Carlsen at different stages of his life. The opponent, Magnus Carlsen, has different difficulty based on the age of Carlsen chosen by the user.

5. Product Overview

5.1 Product Perspective

The finished product will be an independent, standalone java application.

5.2 Summary of Capabilities

Multiplayer Chess game

Customer Benefit	**Supporting Features**
Customers can enjoy online chess with their friends. Rendering a physical board unnecessary.	Our built in chess-engine shows legal moves so that even newcomers can play.
Timer already included - removing the necessity of buying one.	Automatic switching between turns, avoiding human mistakes when switching manually.
Integrated ranking system making it easy to track progress and skill.	Statistics are stored in the database system, making it easily accessible.

5.3 Assumptions and Dependencies

The factors that affect the Vision document and the game itself are as follows:

- MySQL Database:

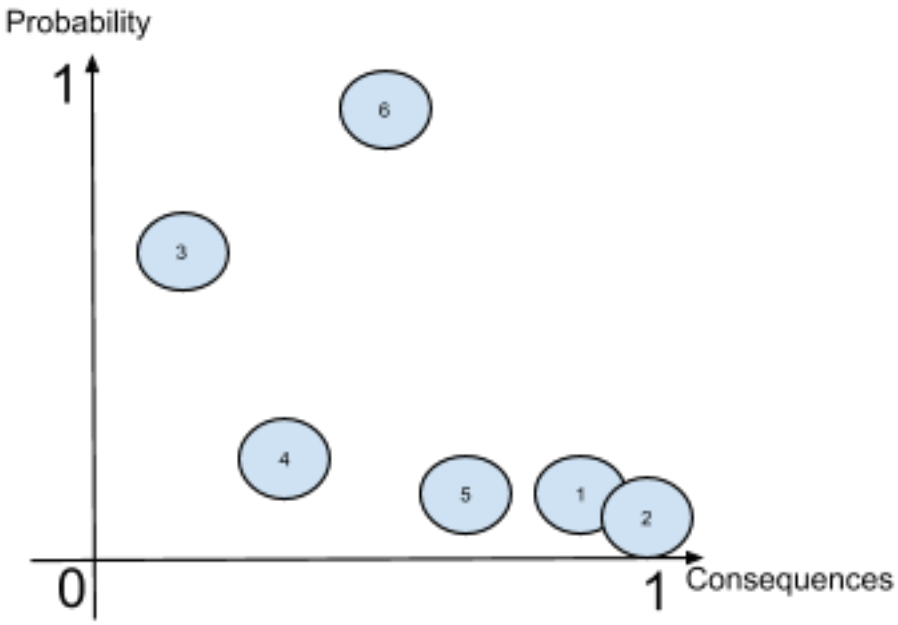
  We depend on the MySQL-database provided by the university. If we lose access to the database, we will have to find another provider of said service. If not, our multiplayer function will no longer be possible.

- Internet Access:

  We rely on internet access for the multiplayer-aspect of the game, which was an important part of the assignment from the key stakeholders.

5.4 Risk analysis

Possible risks:



1. **Deletion of the whole database**
   The database can be deleted if somebody decides to run a DROP DATABASE query. The only way to avoid this problem is to use prepared statements in our application, preventing users from doing so. Every team member also has to be careful and not joke around with such SQL-queries.

2. **SQL-Injections**

   SQL-injections are when a user inserts an SQL-string into a textfield provided by the program. This string is then passed to the database without any protection. This could potentially lead to SQL-queries being executed by the user where it is not intended.

   Therefore, we should protect every place where user-input is accepted, using prepared statements and not trusting user input.

3. **Sickness in teams**

   Sickness in the team could lead to some minor delays in the project. However this is not a very serious issue seeing as a member can work at home. The only situation where sickness could become a big problem, is in the case of severe sickness. This could incapacitate the team member, seriously reducing productivity of the team in the long term.

4. Loss of internet access

Loss of internet access is a relatively minor problem. Most of the work that is to be done on the project can be done offline. Loss of internet access is only a problem if we want to commit, pull, push or do anything Git-related. Testing of multiplayer or anything related to the database will also not be possible. A temporary backup solution to this problem is to use smartphones as a wireless hotspot until regaining access to the internet.

5. Unintended removal of any code

Unintended removal of any code could be a big problem, making all our previous progress non-existent. However, this problem can be solved by a simple git-rollback. To prevent this problem from ever occurring, we should be careful when handling folders or files in general, so that no mishaps occur.

6. Git problems

Git problems are likely to arise, but they are rarely very serious. Most of the time there will be merge conflicts or some user-branch being some steps behind or ahead of master. To proactively prevent this the team will split tasks and make sure we do not work on the same place at the same time. The team will also make sure to commit only the changes they actually want to keep, and actively try to individually stay updated on the branch before pushing commits. Using several branches is also recommended as big conflicts will only happen as soon as branches are merged.

6. Product Features

6.1 Play chess on different computers

We wish to make it possible for users to play against each other. They should be able to log in to our program using their own PC and play against each other online.

6.2 Registering a user

The user should be able to safely create an account on our service and use it to store a rating and statistics connected to their games.

6.3 ELO-rating

Integrated Elo-rating, a ranking system allowing users to see their progress and ranking.

6.4 Game timer
Implement timed mode with automatic switching of turns as soon as a move is made.

6.5 User profile
Being able to have your own personalised profile with statistics, ELO-graph and avatar.

6.6 Find User
Possibility to search for other users and see their stats.

6.7 Leaderboard
The game features a leaderboard showing the top players within our game; the 5 players with the highest ELO rating.

6.8 Alternate game modes
Include alternate game modes such as Peasants revolt, Fischer Random or Crazy House.

6.9 Live chat
We want to create a fast and responsive chat for the players to communicate during a game.

6.10 Sandbox
We want to create a sandbox chess game for playing on your own, either while waiting in queue for a game, or just to figure out some tactics by yourself.

7. Constraints
This projects constraint is the fact that we have no server to do calculations for us, so everything has to be done in front-end.

8. Quality Ranges

The team will focus on performance and robustness on the project, making the minimum viable product as good and efficient as possible before any expansions are made, while usability might suffer for it, as the team is not going to put a lot of resources into graphics and animations, the product itself will be very solid and fault tolerant.

9. Precedence and Priority

The team's first priority is to create an application where the user is able to play regular chess with another user on another device. Ranking comes next, allowing users to view different statistics from their previously played chess games. The application's interface will not be prioritized as highly as functionality and effectiveness. However, interface will be kept in mind when working on the application and also be fully fleshed out with the finished product.

10. Other Product Requirements

10.1 System Requirements

- CPU: 2 GHz Intel Pentium 4 or AMD Athlon or equivalent.
- OS: Windows 7/8/10, UNIX/MAC-OS
- VIDEO CARD: Intel HD graphique.
- SOUND CARD: All.
- FREE DISK SPACE: 100 MB.

10.2 Performance Requirements

- Program should run smoothly without noticeable lag, even on lower end computers.
- Less than a 0,5 s delay between sending a move and the opponent receiving it.
- No noticeable "input lag". The time it takes between the user clicking on the screen and the graphics being updated should not be noticeable.

11. Documentation Requirements

11.1 User Manual

We have to create an user manual that contains how to log in, and how to use our application.

11.2 Installation Guide

We also have to create an installation guide for how to download Java JRE and how to download and run our application.

12. Feature attributes

12.1 Status

Proposed	• Game mode with different characters replacing the king. • Chess bot for single player • Emotes • Ability to see which opening is being played at a specific point in time.
Approved	• Timer • Online play • Alternative game modes, eg. Fischer Random • Show legal moves for a given piece • Ranking system • Choosing amount of time for timed chess • Chat system
Incorporated	• Timer • Online play • Alternative game modes, eg. Fischer Random • Show legal moves for a given piece • Ranking system • Choosing amount of time for timed chess • Chat system

12.2 Stability

The only real discussion at this stage is whether or not we should make a chess bot. Most team members consider this to be a big task compared to the original purpose of this project. Therefore, this will only be an additional feature implemented if time is not scarce.

12.3 Target Release Date

Target release date for our final product will be April 12. 2019.

Attachment 6: Link to Gitlab WIKI

Here you have the link to our Gitlab WIKI:

https://gitlab.stud.iie.ntnu.no/martijni/team-20---spillprosjekt/wikis/home