Monopolish
Team 4
2019

Version 1.5

Bård Hestmark
Eirik Hemstad
Mikael Karlstad
Torbjørn Bøe Lauvvik
Lisa Willa

Revision History

Date	Version	Description	Author
01/04/2019	0.1	Started on main report	Team 4
09/04/2019	0.2	Created template with headers, started filling out sections	Team 4
10/04/2019	1.3	Further filling out sections	Team 4
11/04/2019	1.4	Finalizing report	Team 4
12/04/2019	1.5	Finalizing report, part 2	Team 4

Table of Contents

Main report

1. Preface

The study program Computer Engineering (NTNU, 2019) specializes in software development and focuses on practical development through the use of practical projects. This project is designed to help teach the students to work well in teams, to work on bigger development projects with databases, and to document them expediently.

During the project we have learned how to use the JDBC API in practice, procedures in SQL and the use of DAO classes. Furthermore, we have learned how to develop larger applications in JavaFX and connecting the front-end with the back-end and the database. Additionally we learned how to make executable java applications. We have also worked on how to implement user security with hashing and salting passwords.

Moreover, we learned how to plan, document and develop a project of this magnitude, and have a better understanding of the value of these parts of the project process. In addition to this we have learned to work together as a team..

We would like to thank Grethe Sandstrak and Nils Tesdal for being available and giving good feedback and help.

We would also like to thank our test subjects for being willing to help test the application and give us feedback.

2. Introduction

2.1 Purpose

The purpose of this main report is to document and evaluate our project. This report will also give an overview of the assignment description, how we chose to solve it and what we could have been done differently.

2.2 Scope

This document is in relation to the main project in the subject Systemutvikling 1 (TDAT1006). It covers the entire team work, development process and other documentation regarding the project.

2.3 Definitions, Acronyms, and Abbreviations

Term	**Definition**
Player	An entity that interacts with the in-game environment, representing a user
User	The identity of any person that interacts with the application
Server	A computer that serves other computers(clients), in our case a database server, that provides database services to the users of the application. And houses our database.
Logic	The logical composition of a program code.
GUI	The Graphical User Interface allows users to see and interact with the game
IDE	Integrated Development Environment
MySQL	A database management system, it has its own query syntax which is also referred to as MySQL
Stored procedure	A pre-compiled SQL query that is stored in the database, that can be called by an application. They protect against SQL injection.

SQL injection	A security breach when working with SQL queries in applications, that allows a user to modify the query with the input to the query. This can give an user access to things they should not have access to.
DAO	Data Access Object. Class for handling communication between the application and the database.
Kanban	A signboard system for keeping track of tasks that need to be done in a project
View/Scene	A graphical user interface section of the application. E.g. the login view, which is the GUI for the login screen of the application.
I/O	The means a physical user can use to interact with the system, like mouse, keyboard or simply the computer screen
Concurrency	An aspect of computer science where various processes run out of order in parallel to improve work speed
UX	User Experience. The experience of the target user when using the application.
API	Application Programming Interface. A set of classes and functions that can be used.

2.4 References

Gitlab Repo (2019) Available from https://gitlab.stud.iie.ntnu.no/team04/monopolish (Retrieved: 12.04.2019)

GitLab Wiki (2019) Available from https://gitlab.stud.iie.ntnu.no/team04/monopolish/wikis/home (Retrieved: 11.04.2019)

Monopolish JavaDocs (2019) Available from: http://team04.pages.stud.idi.ntnu.no/monopolish (Retrieved: 11.04.2019).

Swaldman/c3p0 (2019) Available from: https://github.com/swaldman/c3p0 (Retrieved: 12.04.2019).

NTNU Computer Science study program description (2019) Available from: https://www.ntnu.edu/studies/ithingda (Retrieved: 12.04.2019)

Sandstrak, G. Tesdal, N (2019) *TDAT1006 Software Engineering 1 with Database Project Assignment* Available from: https://ntnu.blackboard.com/bbcswebdav/pid-541629-dt-content-rid-19759130_1/courses/194_TDAT1006_A_2019_V_1/SU1-prosjekt-2019-Game.pdf (Retrieved: 11.04.2019)

Team 4 *Vision document* (2019) https://gitlab.stud.iie.ntnu.no/team04/monopolish/wikis/Home/Vision-document (Retrieved: 11.04.2019)

JDBC - Statement, PreparedStatement and CallableStatement (2019) Available from: https://www.tutorialspoint.com/jdbc/jdbc-statements.htm (Retrieved: 12.04.2019)

2.5 Overview

This Main report contains further information about the Monopolish project structured according to the table of contents. Each section is a new subject and the subheadings further explains the main points of the section.

Main points from each section of the report:

- The preface section explains the purpose of the project and what the team has learned from it.
- The second section of the report is an introduction to the project and it explains whether our objectives where completed.
- The third section describes the assignment and the nature of the work.
- The fourth section explains how the assignment was solved, some problems the team stumbled upon and how these were solved. Further detail include how the front-end, back-end and database portions of the project was done.
- Fifth section gets more in depth about the implementation of the project, which results were achieved in terms of what the team initially intended. Problems that occured and solutions to these problems. Limitations of the system explains our applications main limitations.
- The sixth section of the report lists further work to be done regarding product perspective, product functions and dependencies.
- The seventh section has information about the repository and how to use the application, including usernames and passwords for test users.
- This file contains all of the required documentation for the assignment.

3. Assignment

A brief description of the assignment that was given and the nature of the work is specified in this section. This section also includes a interpretation of the assignment since it is important to clarify how we as a team analyzed the assignment. The basis of this analysis shapes the foundation of our ideas and the work we have achieved.

3.1 Assignment description

The assignment was to create a turn based game that used Java and MySQL database to communicate. Requirements for the project (Sandstrak, Tesdal, 2019) stated that JavaFX or Swing should be used to develop the GUI. The nature of the work was developing an application with a GUI suitable for multiple players at a time.

3.1.1 Technical requirements

1. The game must be made as a standalone Java application.
2. Use the MySQL database at the university to store game state and game history/statistics.
3. User passwords should be hashed and salted. Find theory and examples of this online.
4. Use Prepared Statement (or stored procedures) when making database calls to prevent SQL injection.
5. Use a connection pool with one connection(or more if you have multiple threads). This is to improve performance and avoid using too many database connections. Find theory and examples of this online.
6. Test all the classes that define the game logic with JUnit(be sure to isolate the game logic in dedicated classes instead of spreading this around).

3.2.2 Client information

Grete Sandstrak: Assistant Professor at Department of Computer Science at NTNU

E-mail: grethe.sandstrak@ntnu.no

Phone: (+47) 735 59 561

Nils Tesdal: Assistant Professor at Department of Computer Science at NTNU

E-mail: nils.tesdal@ntnu.no

Phone: (+47) 734 12 112

3.2 Assignment interpretation

Underneath is a brief description of the main points we focused on when interpreting this assignment.

3.2.1 Project difficulty

After some discussion we figured out from the assignment and the time given to complete the task, that we had to balance the project difficulty. The game we chose for the assignment had to be challenging enough, but also comprehensible enough so we could manage to develop it in time. There were some discussions within the team to make a type of card game, but we found out that this would not be challenging enough. Therefore, we later decided to make our own version of monopoly.

We decided on monopoly since it is a game that has a lot of rules and logic that can be challenging to implement. Especially the front-end would likely be more challenging than a card game for instance. The game is turn based, but we wanted the users to be able to see the events that occured in the game even if it was not their turn.

3.2.2 User experience

The assignment was to make a multiplayer game for people who wanted to play the game. Therefore, we quickly decided that UX is important for this project. We also decided that it was necessary to create a platform around the actual game to give a better UX. We wanted to implement features that users expect to see, such as a login-, register-, dashboard-, and lobby-screen. In addition to this, we wanted to implement extra features that would make the experience more enjoyable for the user.

This also implied that we wanted to focus on developing GUI with the UX in mind. We decided that spending some extra resources and time on the front-end would make the application better for our users. In total we wanted to balance the workload on the front- and back-end, since both are important for the application and the UX.

3.2.3 Documentation

We understood that documentation would be an important aspect of the process in this project, and wanted to make sure that we could document everything as good as possible. We therefore made sure to complete prototype drafts of all class diagrams, and comment all of the code from the beginning. This would also help much on communication during development, so that everyone is well informed on what the code does and how the application is structured. We made sure to read all of the requirements regarding documentation (Sandstrak, Tesdal, 2019) beforehand, so that we knew what we needed to get working on right away.

3.2.4 Learning Outcome

The team expected to learn more about building a Java application that uses a database to hold information about users and game data. Equally important we expected to gain additional experience with working as a team, and completing a project of this magnitude. In addition to this, we expected to learn more about front-end development with JavaFX.

4. How the assignment was solved

4.1 Methods and standards

We mainly used the usual development practices for developing in a team environment. A remote repository was set up on GitLab (GitLab Repo, 2019), where team members could clone the repository onto their own systems. When a work task was done, the team member performing the task could upload this to the remote repository once they could make sure that the changes were working. This made it easier to work together as a team, and perform changes in different areas of the source code. We kept track of issues and tasks that needed to be done through a Kanban-style board on the GitLab project site (GitLab Repo, 2019). Though workload was evenly distributed and allround, there were some team members who gained certain areas of expertise that made them more suitable for completing certain tasks.

The project structure was based on standard conventions, where we separated the source code, resources, test classes and libraries from each other. The source code is based on a package-by-feature structure, where classes are grouped by what practical feature they add to the application.

4.1.1 Back-end

Before starting code implementation, we wanted to make diagrams that could give us an idea on how to structure the code. Naturally, major and necessary changes became more apparent as we progressed through development. We tried to stick to standard object-oriented conventions, where we separated functional classes, GUI-elements and database-related functionality. This would also mean creating extra layers to abstract away certain problems and methods, preparing easy-to-use methods for a variety of operations that would appear later in development. Various utility classes for data sources and settings handling were introduced to make it easier for other class to retrieve global attributes. The most important element in this system is the Handler, which is a static class used for storing global variables that nearly all classes in application may use, like the id of the currently logged in user, id of the current game being played and all of the application's DAOs.

4.1.2 Front-end

FXML files were used for the GUI elements, to exclude as much GUI from the Java classes themselves. We used Scene Builder to design and structure the graphical elements instead of editing the FXML files directly. This makes design a lot more streamlined. Controller classes were then used to assign functional operations to buttons and fields, as well as updating the graphical elements in regards to changes from the database and other user actions.

We also created a few classes which draw/rendered different GUI elements that needed more dynamic rendering than the static elements created in SceneBuilder. A more detailed description of the different methods we used to dynamically update the GUI elements can be found in section 5.2.1.

4.1.3 Database

Data Access Objects (DAOs) were used to create a layer between the game logic and the stored procedures in the database. This means that all database operations were abstracted into simple methods that could perform those operations with ease. That way, we always know which classes to look up once a database operation goes awry.

We chose to use stored procedures, when interacting with the database. This is often done by professionals as it lets them to have the data/database layer separate from GUI, and the application logic. Layering allows a developer to change one layer of the application without having to change the rest. Meaning that one can use a stored procedure in many different applications. A stored procedure is also stored fully compiled and optimized in the database saving both time and and space for the application. Stored procedures also protect against SQL injection the same way a stored procedure does. Another advantage is that it simplifies/allows for more complicated SQL queries and efficient reuse.

A connection pool class was used for connecting to the database. This means that an instance of this class holds a pool of open connections throughout the lifetime of the application. Data Access Objects that wants to connect to the database has to retrieve a connection from this pool, then put it back once it's done. This is a regular practice to prevent opening and closing multiple connections over a short time, which again prevents strain on the database server. We used C3p0 (Swaldman/c3p0, 2019), which is an external library for JDBC, for setting up a connection pool.

4.2 External resources

During development the team faced some challenges with database connections and JavaFx. Google was used to search for solutions to our problems as it is a good way to search through big amounts of information. Many of the answers to our questions were found on StackOverflow, W3Schools, Oracle Docs, MySQL docs and YouTube.

- StackOverflow was used a lot when searching for answers to MySQL or Java issues.
- The MySQL docs was used to find answers to tricky SQL questions.
- W3Schools has a lot of useful examples on the various SQL questions.
- Oracle Docs has documentation for all JavaFX classes and methods that we used to solve GUI issues.

4.3 Hardware

During this project the only hardware we used was our own computers, NTNU's GitLab and NTNU's MySQL - server.

4.4 Software

During the development, the team used a lot of different software. IntelliJ from JetBrains was chosen as the main IDE because of its ease of use and the fact that the team already was familiar with it. JetBrains offers IntelliJ Ultimate for students free of charge. Other software used in the development:

- Git was used to communicating with the GitLab-server.
- GitLab was used as a remote version control system as the assignment required it.
- MySQL was used for a database system because it was required for the assignment.
- Lucidchart was used to design UML-diagrams and database models as it works quite similar to Google Docs in terms of collaboration.
- Google Drive / Docs was used to keep all documents in one place because it simplifies collaboration.
- Balsamiq Mockups was used to make the first design prototypes and as a guide when designing the GUI in Scene Builder.
- SceneBuilder was the natural choice for assistance in building a JavaFX application as it allows for easier implementation and design of GUI.
- Facebook Messenger was used for team communication because the team members already uses it for the most of their communication.
- Slack was used for communication with the project owners.
- Blackboard used for delivering required documents.
- Everyone in the team used Windows 10 during development.

Java libraries used in the project:

- JUnit used for unit testing during development.
- MySQL-connector, a connector used for communicating with the database
- JDK - Java Development Kit
- JRE - Java Runtime Environment

4.5 How the work was divided between team members

All of the team members participated in all of the different tasks but the different aspects of the development tasks were divided into three different categories, front end (GUI), backend (Java), database(MySql). This section will give a brief description of the responsibilities that the different categories involved. It will also give information about what each team member was responsible for during development.

4.5.1 Front-end (GUI)

The team members responsible for the GUI had the responsibility to research and develop the graphical interfaces in the application. This involved developing the Java Controller classes for the GUI and the FXML (GUI layout). We decided to split the GUI into two different sections; *Game* and *Platform*:

Game Section

Responsible: Bård Hestmark

The game section is about the game view in the application. Early in the project, we realised that this view/scene would be complicated since it needed many components and difficult logic. One of the team members, Bård Hestmark, took responsibility for this section. Although at the end of the project several other team members joined to mainly work on this.

Platform Section

Responsible: Mikael Kalstad

The platform section is more abstract than the game section, it is about all the GUI that is around the game section. This section will give the application a structure surrounding the game. Sub-sections in this section are:

- Login
- Register
- Dashboard
- Lobby

4.5.2 Back-end (Java)

The team members responsible for the back-end of the application had the responsibility to research and develop the basis structure for the application. This involves creating Java classes for different purposes that can be used in different parts of the application. We decided to split this section into these sub-sections: *Game Logic, Game Entities* and *DAO*:

Game Logic

Responsible: Eirik Hemstad

Game Logic was mostly compressed into a single class that could use entities to perform all logical game operations. This means structuring the game, player, property and other such objects in a way that made them easy to access and edit. We tried abstracting all game logic operations in a single class, so that all other classes only needed to use methods from this class to perform operations on player- and property-classes. The Game Controller created by Bård was used as a interaction layer between I/O and the game operations.

Game entities

Responsible: Eirik Hemstad

The game entities are the different entities that are used in the game. This work involves creating useful methods for handling communication between different entities. Entities must also be incorporated with the database if needed, therefore it is important to work with the team member responsible for the DAO. Game Logic works directly with these entities, so they need to be updated consistently.

DAO

Responsible: Lisa Willa

The DAO classes work as the communication layer between the application and the database. This responsibility involves creating different classes with useful methods that is used in the application. The person responsible must also make sure the DAO methods correlate with the procedures created. By creating these DAO classes we can abstract the layer between the database and the application.

4.5.3 Database (MySql)

Responsible: Torbjørn Bøe Lauvvik, Lisa Willa

The people responsible for procedures worked closely with the database structure, and was responsible for creating useful procedures that was needed in the application. It is also important that they work along with the people responsible for the DAO, since the DAO classes uses these procedures. Eirik and Lisa were mostly responsible for database-related structure and methods. Torbjørn also took a big part in creating certain procedures. Lisa made sure to update the database according to any requirements resulting from changes in the database.

4.6 Documentation

The GitLab Wiki contains the majority of the documentation, a description of these can be found underneath. JavaDocs for the application is generated into a website which can be found in the WIKI. The link to the WIKI can be found in section 7.

4.6.1 Wiki

Structure of the documentation in the Wiki

Main Wiki page

    Vision document

Extra features

Requirements

- Use-case diagrams
- Domain model
- Sequence diagram
- Wireframes

System

- Project structure
- Class diagram
- Database model
- Source code
- Installation manual
- User manual

The main wiki page contains the vision document and links to “Requirements” and “System”.

The requirements part of the documentation contains:

- Use-case diagrams (GitLab Wiki, 2019) that shows two use-cases where the user wants to do stuff…
- The domain model shows how the different parts of the application works together and is sort of similar to the class diagram but it does not show classes or database tables, rather the parent structure.
- The sequence diagram shows how different methods collaborate to do a certain task.
- The wireframes are a mockup of the GUI design and used as a basis for the JavaFX design.

The system part of the wiki contains:

- Class diagram of the most important classes in the application.
- Database model which shows the relation between different database tables.
- Link to GitLab repository for the source code.
- Installation manual for the application.
- How to play / game rules.
- User manual that explains how to use the application.

Team 4

5. Implementation of the project

Early on in the project, we set some objectives that we wanted to achieve:

- Thorough planning to make development more efficient and straightforward.
- A stable platform around the main game.
- Focus on basic functionality before adding extra features.

The planning at the start of our project went well. Before we started to develop the application we spent some time planning on how we wanted to structure our system. We also planned how we were going to work together as a team, and the different roles each team member would have during development.

These roles were main responsibilities; which suggests that the person would research and distribute work within this domain. However, this does not imply that the person should only work his or her responsibility domain. By doing this we had more control over the different aspects of the system, and at the same time, each team member would work on every aspect of the system.

Although we planned thoroughly we had to make some changes to our original plans during the project. Some aspects of the application and the game could not be implemented the way we thought. Therefore we had to make some small changes to the system structure. Also during the project, small changes to the design mockups were made to improve on the original design. We wanted to focus on the user experience, therefore we made changes where we found better solutions.

When we started to develop, we knew what we wanted to make, but some aspects of our vision required some research. Especially GUI was a bit intricate to implement in the beginning. To implement some features that we had envisioned we needed more knowledge and experience with JavaFX and SceneBuilder then we received in lectures and assignments in the subject “TDAT1006 Systemutvikling 1 med databaseprosjekt”. After some time researching and trying out different solutions, we became more familiar and experienced with this library and software.

5.2 Problems that occurred and our solutions

5.2.1 GUI dynamic update

We had some problems with dynamically updating the GUI in different parts of the application with data from the database. We needed to create the different elements in the GUI in such a way that it could easily be created, updated and removed dynamically. We found three different methods of doing this:

1. **Draw methods**

   This solution uses JavaFX code to create the element that needs to be rendered. This element is then placed in a fixed container which is created in SceneBuilder and compiled to a .fxml file. Removing this element can be easily done by removing all elements in the container or by setting an id and finding that specific element. By doing this the element can be dynamically created, rendered and removed based on data from the database.

2. **Static elements**

   Some elements do not need to be re-rendered each time, and sometimes we only need one instance of the element. In this situation, the element can be created in SceneBuilder and get an fx:id which can be used in a controller class for the .fxml file. This enables us to easily dynamically display and hide the element.

3. **View elements**

   To have more control- and to make it easier to create complex layouts, SceneBuilder can be used to create elements. These elements can then be rendered inside a container in the same way as the draw method. The different elements inside this parent can easily be manipulated by searching with an index or by setting an id.

In our application, we used all three of these different methods. We found that in different situations, different methods are more suited for the solution and are faster and easier to develop.

5.2.2 Player pieces movement on the board

This is another GUI issue that we had some problems solving. The player pieces on the board should dynamically update and move to a position. The main issue was finding and placing the player piece on the specified position on the board. To solve this, we made a layer on top of the actual board which only held the player pieces. Later on, we also made a similar layer to hold the houses and hotels on the properties.

The position also needed to be converted to an X and Y coordinate which could be used with the GUI elements. In addition to this, we needed to dynamically place the player piece on the tile along with other player pieces. The solution to this was to check how many pieces were on the board and then set standard layouts for the different number of players on a single tile.

5.2.3 Connection Pool

We decided early on to create a custom class for connection pooling, as this seemed like a simple class to make from scratch. It seemed to work fine from the start, but we started meeting some challenges as soon as we introduced timers running on separate threads. This seemed to create some concurrency issues, as the pool would increase quickly, and eventually filling up. We couldn't find any solution to this, so we decided to convert to C3p0, which is an external library for connection pooling. Converting to this system seemed to solve the problems. This is most likely because C3p0 is a more sophisticated library more suited for concurrent use (Swaldman/c3p0, 2019).

5.2.4 Switching views

An issue that appeared when making several differents GUI views/scenes, is to easily switch between them. A scene is inside a window/stage, if you made a new window/stage for every new scene, a new window would pop up above the previous one. This is a bad solution, therefore we made a Scene Manager. This manager will handle all view switching within the application. This manager has one window/stage which will switch contain the actual scene. Switching a scene will only happen inside this window/stage.

5.3.4 Synchronizing forfeit

A forfeit can be initiated by a player in the game. When a player forfeits a popup dialogue will appear to all players. The players can then vote if they want to continue playing the game, or if the game should finish and a winner is declared. The issue with this feature is that there is no guarantee that players are synchronized before, after and when voting. During testing, we found that the forfeit would work the first time, but if a player immediately initiated a new forfeit right after the last one, the other players would not see the forfeit dialogue.

The solution to this problem was implementing a method of making sure all players have read the votes for the current forfeit before resetting the votes in the database. By creating a "read-check" entry in the database, players can confirm that they have read the votes. When all players have read the votes and set their "read-check", all entries regarding forfeit in the database will be reset. The players will not be able to initiate a forfeit before this reset is performed.

5.3 Limitations of our system

5.3.1 Database connection

Our system relies heavily upon a fast database connection. If a user has a slow connection it will impact the user experience of our application. Especially the GUI can be unresponsive if the connection is slow. This is due to our solution regarding the updates and refreshes. We wanted our application to feel responsive to improve user experience, therefore we are pulling data from the database frequently. A solution to this issue could be to restructure our algorithms to check the database less frequently to decrease traffic and refactor our GUI to be less dependent on frequent updates and re-renders.

5.3.2 Amount of players in each game

A different limitation that we chose to set was the maximum amount of players in each game. This limitation was due to the nature of the game and design we chose to implement. After some discussion and trial, we limited each game to have a maximum of four players. We found that adding more players would make the game less exciting for the users since the time between each round would increase as we added more players. The user cannot do many things while waiting for their turn, which is why we chose this limitation.

Adding more players would also break our design. We had some ideas on how to implement a design that supported more players in each game, but this was not needed when we limited it to four players. Due to our limitations of players in each game, we decided that we wanted a lobby system where the users could choose who they wanted to play with. This lobby system should also not have any limits to the number of lobbies that could be created.

5.4 What could have been done differently

5.4.1 Earlier development of the game

In the start of the project after we had spent some time planning, we started by developing the platform around the game. This involves a platform where users can log in, register an account, dashboard/home screen and a lobby. Creating of all this took some time and effort, therefore we did not start fully to develop the game until after this platform was stable. Though some members of the team worked on the game from the beginning, but the rest of the team did not join in until later.

The actual game needed more features than we first expected to be an enjoyable experience for the players. Therefore, the game took a longer time to develop and finish than we first expected according to our Gantt-diagram (Team 4, 2019).

5.3.2 Overview concerning game rules

During the development, we had some discussion regarding the game rules. Each team member had different rules for different aspects of the game. During the planning, we could have gone through all the different aspects of the game and agreed on rules. This would have coordinated the team better when developing the different parts of the actual game. There were moments where a team member implemented different rules in the game than the rest of the team wanted, therefore setting the rules before development could have made us more efficient.

5.3.3 Naming and structure convection

Variable-, class- and function names could have been more consistent within the team by implementing a naming convention. This also applies to some structure in the code. During development, there were moments where it could be confusing to use or refactor code that a different team member had made. Some team members used a different structure when coding than others, therefore some structure in the code was inconsistent.

Also, this could be applied to the procedures. Some team members had established a naming convention for the procedures. Though the other team members who had not worked on the procedures previously did not implement this naming convention.

5.5 An analysis of planning and time usage

5.5.1 what was planned

Planning:

The planning phase for the project started 13th of february. We had planned on being finished planning before 1. of March. This would mean getting an overview of functionality and create a Gantt-diagram.

Modeling:

We had planned to begin modelling around 21st of february, wireframing february 18th, and finish all of this work by march 8th.

Development:

The development process was planned to begin march 4th, and be completed by the end of march, so that we could use the last couple of weeks to complete all diagrams, reports and other necessary documentation.

Documentation and post-development:

We planned for the work on documentation to begin march 27th, and persist to the end of the project period (12.04.2019). Here, we would finish the GitLab Wiki, finish all remaining diagrams and do some bug fixing.

5.5.2 What was achieved

Planning:

The planning period went along as planned. We quickly found out what game we wanted to make and what functionality we wanted it to have, and the vision document was delivered on time.

Modelling:

The modelling phase got quite delayed, due to some other work that got in the way of the project in the start. Modelling was started officially march 4th, and we finished all the diagrams and wireframes within the same week. Wireframes got started in time, and was also finished the same week. At the end of the process, we ended up with complete class diagram, activity diagrams and use case diagrams which would prove very useful on implementing the code.

Development:
Start of development got delayed due to the delay of the modelling phase. We started on development march 11th. The completion of this process was also delayed due to the many technical issues that occured during development. We hoped to be able to plan the structure of the application with class diagrams and domain models, but we strayed a bit from the original plans of the structure, due to some unforeseen issues and aspects of the game logic that occured. In regards to the aforementioned class diagram, we had to make a lot of changes upon learning the functional structure of JavaFX and how the game should operate.

Documentation and post-development:

We had finished some documentation before the end of the project period, but the official start time for documentation was delayed quite a bit, due to the delay of previous phases. We started officially on the documentation april 8th, and finished in time by april 12th.

5.5.3 Requirements fulfilled (Tesdal, Sandstrak, 2019)

1. The game must be made as a standalone Java application.

The game is programmed in IntelliJ as a standard Java application, and can be exported as a Jar file.

2. Use the MySQL database at the university to store game state and game history/statistics.

The application uses the JDBC library to establish a connection to the university database server where we have created a database system with stored procedures.

3. User passwords should be hashed and salted. Find theory and examples of this online.

We have used stored procedures to hash and salt passwords before storing them in the database.

4. Use Prepared Statement when making database calls to prevent SQL injection.

We have used Callable Statements to call stored procedures stored in the database. This method is also good for preventing SQL injection, as they work on the same premise as Prepared Statements.

5. Use a connection pool with one connection(or more if you have multiple threads). This is to improve performance and avoid using too many database connections. Find theory and examples of this online.

We have a dedicated Data Source class that uses a pooled data source for distributing open connections throughout the application.

6. Test all the classes that define the game logic with JUnit(be sure to isolate the game logic in dedicated classes instead of spreading this around).

We have used a dedicated project directory for storing test classes made in JUnit. These classes test a variety of game logic methods.

5.5.4 Time accounting

You can see correlation between the time usage of the Gantt-diagram (Team 4 vision document, 2019) and the time usage displayed in the time sheets. The time spent on performing the different tasks ended up coming very close to the distribution we expected from the start. We could have delegated some tasks more properly, for example testing, to get a better overview on how much time we spent on work like this. The different types of tasks were mainly distributed evenly, as expected. Still, there were some more specific areas, like front-end, database and game logic that had more variety on how much time a team member spent doing them.

5.6 What solutions we are satisfied with

Some solutions within our application that we are specially satisfied with:

5.6.1 Front-end design

We are very satisfied with the graphical outcome of the application. Those who have been in charge of designing the graphical interfaces has done a great job creating a good and clean design, with satisfactory reaction from user interaction. The game looks vivid, with good colors and icons.

5.6.2 Procedures

Also, we are happy with the decision to use Stored Procedures in interacting with the database, as we can shape the way we retrieve and update data in the database. This is also faster than using regular Prepared Statements (JDBC, 2019).

5.6.3 Game

We are also generally very happy with the resulting game. We feel that the game has all the functionality that would be expected of a regular Monopoly-type game, and that the game feels fun and engaging to play.

5.7 Teamwork

The teamwork has functioned very well, and everyone has played their part in making the project a true collaborative effort. Everyone has displayed enthusiasm and made sure to voice their opinions on every matter. This has ensured good communication and a result that everyone can be happy with. Everyone has also been open to new ideas, and there has been few conflicts in reaching an agreement in every decision.

Collaboration over Gitlab has worked very well, as this has allowed us to properly work in parallel, thus increasing productivity in the development. We have done some major refactoring during the project, and Git as a version-control system has made it easier to attempt large changes without affecting the latest stable build. Working together on Git has worked better than anticipated, and we imagined that there would be more merge issues and concurrency issues than what became apparent.

6. Further work

Some further work could be to make a Java backend server to run on a dedicated machine. This would make the updating of the board and the game in general much more responsive as the database could just handle storing and updating of the data.

If we had more time the team would like to implement some more features, such as a proper trading functionality and the ability to reset your password. This was dropped due to lack of time. In addition to this, the application would need more bug fixing and refactoring to be more optimal for a stable release.

7. Repository

Link to GitLab Wiki: https://gitlab.stud.iie.ntnu.no/team04/monopolish/wikis/home

An installation manual can be found in the WIKI under the "System" section.

DO THIS BEFORE TRYING TO RUN APPLICATION:

First: Locate the game.properties file in the project root-folder

The file should already contain the db_url:

db_url=jdbc:mysql://mysql.stud.iie.ntnu.no:3306/eirikhem

The following should be added to the file:

db_user=eirikhem

db_pass=xFAdqcan

7.1.1 Test accounts

The application has a register feature with a GUI view, therefore to test the game a user can be created with this feature. However this is optional, we have two different test users that is pre-made and can be used for testing purposes of the application.

Test user 1

Username: test1

Password test123

Team 4

Test user 2

Username: test2

Password: test123

Note that if someone else is currently logged in with one of these users, you will not be able to use this test user. This is to prevent a user logging in several places which can cause a lot of problems in the application. The best solution to prevent this problem is to create a new account. The email does not have to be an actual email address, therefore creating accounts with fake emails is no problem.

Vision Document

Monopoly

IDI, NTNU

REVISION HISTORY

Date	**Version**	**Comment**	**Author**
13.02.2019	0.1	Initial structuring	
14.02.2019	0.2	First draft	
27.02.2019	1.0	First draft final version	
26.03.2019	1.1	Reduced text-size	Torbjørn B. Lauvvik

TABLE OF CONTENTS

1. Introduction

This document describes the specifications given in the project in the subject *Systemutvikling 1 med Databaseprosjekt*. The assignment is to make a multiplayer game where each player uses their own laptop to play. Each player should have a personal account and should log on with a username and password. Animated graphics are not a part of the assignment, and shall therefore not be specifically included in the game.

On the basis of our assignment, we chose to make a game replicating the board game monopoly. It is expected that some functionality of the game might not be implemented due to time constraint. The focus of this project is implementing the platform for the game and then implement as much functionality as possible.

1.1 References

2. Summary issue and product

2.1 Issue summary

This application will be a product of a major project assignment in the subject *Systemutvikling 1 med Databaseprosjekt.* This course focuses heavily on practices used in developing sophisticated software and data-based applications. Therefore, the resulting product is expected to follow proper programming practices and structures, and that each step of the way is documented properly with UML diagrams, ER-diagrams and wireframe models to show a proof-of-concept. Proper documentation of the entire process is also expected, with team meetings and a wiki page overviewing the project.

2.2 Product summary

The product will be a Java-based desktop application installed on the user’s computer. All online play is based around an online database, where all game sessions are stored during runtime. All players use a personal account for logging into the system, which will be used for tracking player scores and joining games. Each game is turn-based, and players will move in a fixed order, writing to the database only on their own turn. All clients will be updated to reflect changes in the database each turn. The database server should be able to handle numerous games simultaneously.

The gameplay and game design will be largely dictated by the official rules of Monopoly.

3. Stakeholder and user descriptions

3.1 Stakeholder summary

Name	Description	Role
Team 4	Students at IDI NTNU	Software development, testing

		and documentation
Teachers/mentors	Employee at IDI NTNU, responsible for the subject the project is a part of.	Mentoring, and grading
Faculty of Information Technology and Electrical Engineering at NTNU	Economic support, as a supplier of hardware and software	Administration and maintenance

3.2 User summary

Name	Description	Role	Stakeholder
People affiliated with IDI in Trondheim	End-user	None	None/developers

3.3 User Environment

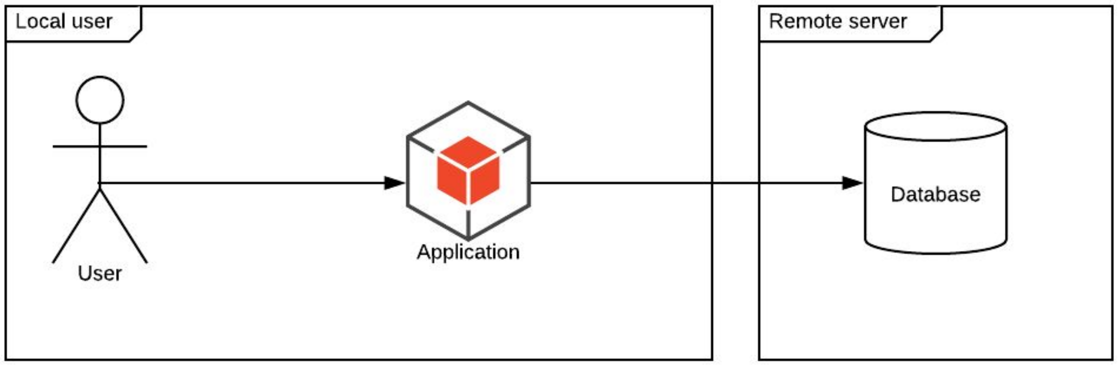
This is a desktop application, you need JRE(java runtime environment), or jPortable and jPortable launcher to run the application. The user also needs to be connected to the internet to properly use the application.

3.4 Summary of user needs

Need	Priority	Concerns	Current solution	Proposed solutions
Register user	high	Login	none	Create a user when entering the system.
Change password	medium	Login	none	Allow the user to change the password when logged in.
Play the game	high	The game	The board game monopoly	
Show high score	medium/high	End-user	none	Display the high score
Chat	Low	End user experience	none	Have a chat in the Lobby

4. Product Overview

4.1 Product role



4.2 Dependencies and assumptions

Anyone with access to the system can register with email and password.

One presupposes that the game covers the curriculum in Java and SQL.

There are no conditions regarding the registration of users. Anyone can register with an e-mail.

4.3 Risk Analysis

	Highly unlikely	Unlikely	Possible	Likely	Very likely
Extensive	🟨 1.	🟥 5.	🟥	🟥	🟥
Major	🟨	🟨	🟨 2.	🟥	🟥
Medium	🟩	🟨	🟨	🟨 4.	🟥 6.
Minor	🟩	🟩	🟨	🟨	🟥 3.
No impact	🟩	🟩	🟩	🟩	🟨

1. Database server unavailable
2. Sickness
3. Bad Git merge
4. Stagnation from lack of competence
5. Computer failure
6. Underestimate time-use

4.4 Prevention of risky situations

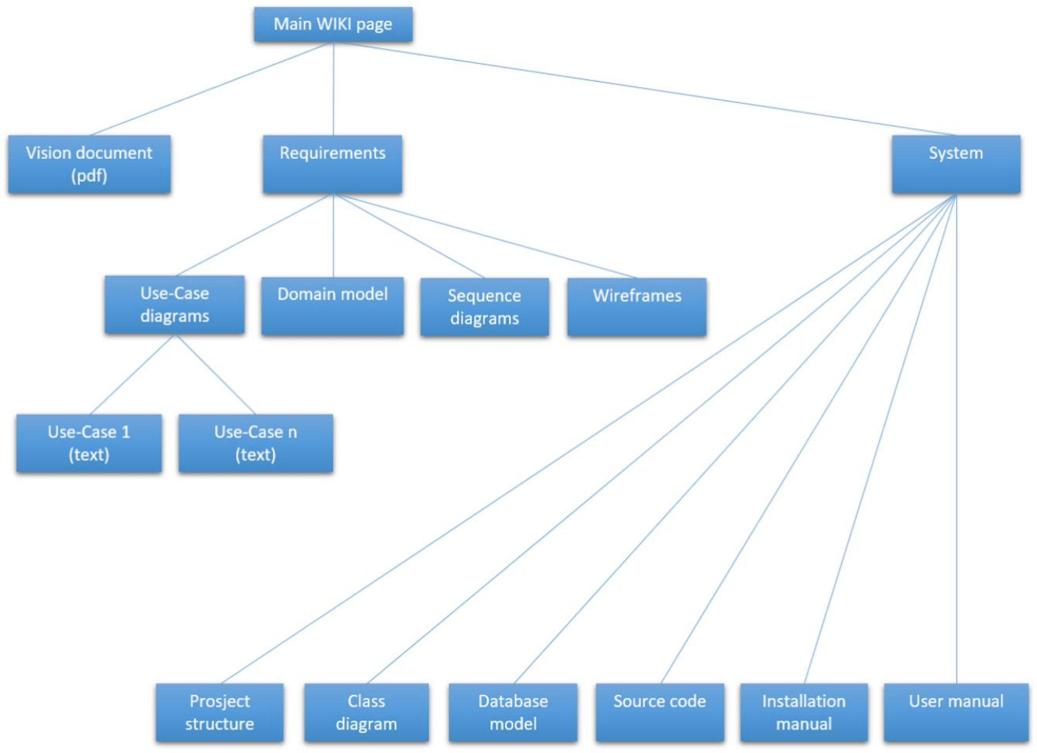
To prevent risky situations we will incorporate good routines:

1. For Database problems, the team has access to a second database to use while developing the application. That way the probability of database-problems due to faults with the server is less likely, as there are two.
2. For sickness, the team has more than one member involved in all aspects of the development so that the team will be able to progress even if someone is sick. Making it a smaller problem.
3. The team members will make sure that everything works when they push commits in git, and will update and push often. In addition to this there will be continuous communication between team members. The team members will also try to avoid working in the same class simultaneously, to reduce the risk further.
4. The team will cooperate, and research to avoid stagnation, and work together when things become complicated.
5. The team will try to take care of their equipment, but has no other way to prevent computer failure or damage.
6. The team will try to keep aware of the deadlines, and try to overestimate time-use instead of underestimating it.

5. Functional features

Login / register	When opening the app users will be prompted to either login or make an account. To make an account the user will need a unique username, email address and a password. Lost password sent via email is an extra feature that can be added if there is time.
Requirements	The game requires users to make an account to keep their score and to keep track of where each player are in case of disconnecting. Each game allows 2-4 players to participate.
GUI	The GUI will contain two “dices” and a picture of the Monopoly game board with dots showing where each player is located on the board. A text box will display currently relevant information like whose turn it is and what the user has to do next.
Highscore	The application will also include a list of top 10 high scores of all registered users. The score will be based on the total amount of points at the end of a round. The high score data will consist of the number of points and username. The high score will be calculated using the value of each “street” the user owns and the amount of money he got.

6. Documentation requirements:



7. Non-functional characteristics and other specifications

Users and score will be stored in a database. The game will be run through a desktop application connected to a database.

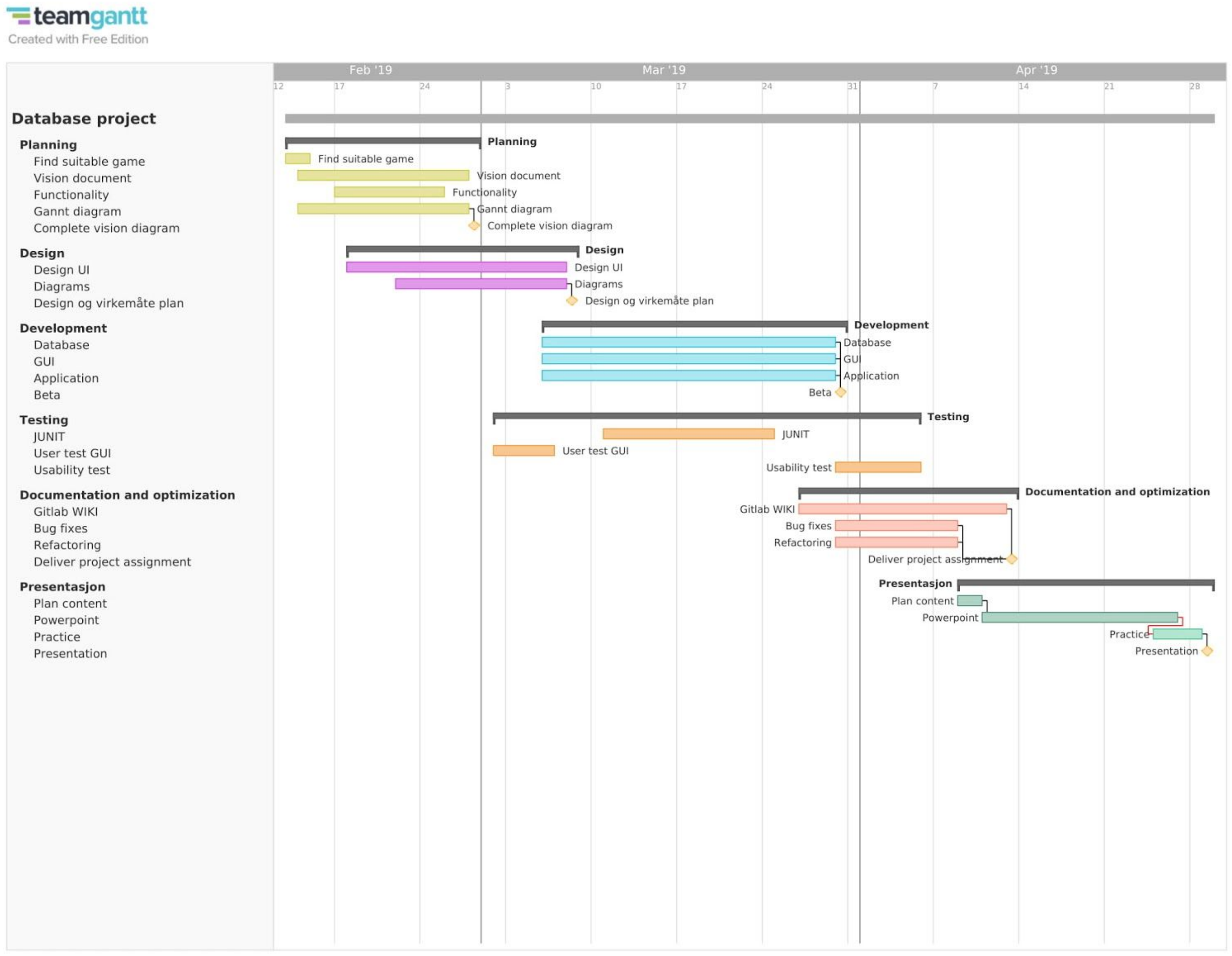
Even under load, the response time of the application should be no more than one second.

These are the technical requirements for the finished project:

1. The game must be made as a standalone Java application.
2. Use the MySQL database at the university to store game state and game history/statistics.
3. User passwords should be hashed and salted. Find theory and examples of this online.
4. Use the Prepared Statement when making database calls to prevent SQL injection.
5. Use a connection pool with one connection(or more if you have multiple threads). This is to improve performance and avoid using too many database connections. Find theory and examples of this online.
6. Test all the classes that define the game logic with JUnit(be sure to isolate the game logic in dedicated classes instead of spreading this around).

Attachments

Gantt diagram



Kunnskap for en bedre verden

Meeting agenda 1/2019 for Team 4 Trondheim, 28.02.2019

This summons goes out to:
Torbjørn B. Lauvvik, Lisa Willa, Bård Hestmark, Mikael Kalstad, Eirik Hemstad

Time and date: Thursday 07.03.2019 at 13.30 – 13.55. TBM 401, 4th floor at Akrinn

Agenda:

Case no.	Cases	Duration	Decision	Responsible
01/2019	Vision document	12 min		Team leader
02/2019	Demo 1.prototype wireframes	8 min		Team leader
03/2019	Other business	5 min		Supervisor

There will no breaks or service during the meeting.

Contact Torbjørn B. Lauvvik (torbjbla@stud.ntnu.no) if you are unable to participate.

Welcome!

Torbjørn B. Lauvvik

NTNU
Kunnskap for en bedre verden

Meeting 07.03.2019, TBL401

Entire team 4 is present

Case 01/2019

Recommended improvements for Vision document

1. Too large font
2. Need to include author in versions
3. Include teachers in stakeholders
4. We need to be included in stakeholders
5. Target demographic, more specific
6. Nils wants chat
7. Is mediocre too offensive?
8. Database server must be represented by a cylinder
9. Recommendations for prevention of risky situations
10. More structure in Functional Features (put it into a table?)
11. Remove wireframe screenshots from Vision Document, need it in Specifications Document instead
12. Include pictures of Wiki

Case 02/2019
Case dismissed. Teacher approves Wireframe.

Case 03/2019
Wanted to present class diagrams. No specific notes.

Meeting agenda 2/2019 for Team 4 Trondheim, 25.03.2019

This summons goes out to:
Torbjørn B. Lauvvik, Lisa Willa, Bård Hestmark, Mikael Kalstad, Eirik Hemstad, Grethe Sandstrak

Time and date: Monday 25.03.2019 at 13.30 – 13.55. TBM 401, 4th floor at Akrinn

Agenda:

Case no.	Cases	Duration	Decision	Responsible
04/2019	Feedback gitlab	12 min		Team leader
05/2019	Demo 2. prototype	8 min		Team leader
06/2019	Other business	5 min		Supervisor

There will no breaks or service during the meeting.

Contact Torbjørn B. Lauvvik (torbjbla@stud.ntnu.no) if you are unable to participate.

Welcome!

Torbjørn B. Lauvvik

NTNU
Kunnskap for en bedre verden

Meeting 25.03.2019, TBL401

Entire team 4 is present

Case 04/2019 - GitLab feedback
USE CASE:
What about buying houses/hotels. When is this action available?
Remove “unnecessary” use cases e.g. very short use cases
Properties: Change Trigger to “Want to buy property”
Expand main flow in Buy Property to include all automatic operations in the transaction

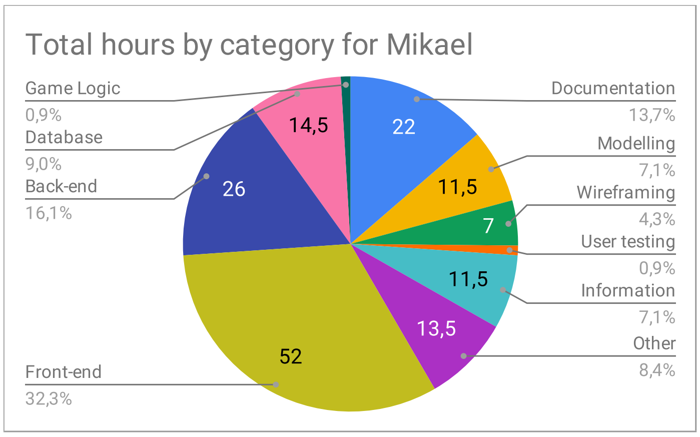
DOMAIN MODEL:
Remove application entities
Change arrows from inheritance arrows to normal relational arrows
Domain model should display the practical communication between entities in the diagram, how the game plays out and which rules are present in gameplay

SEQUENCE DIAGRAM:
Can simplify with further abstraction to remove.

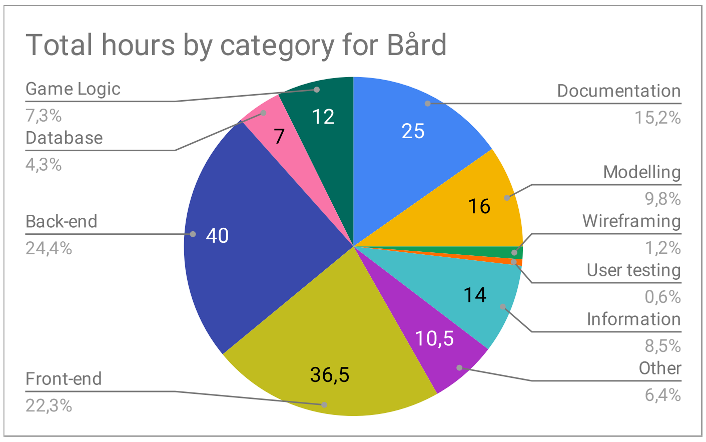
Case 05/2019
Demo looks good, no specific notes.

Case 06/2019
No further business

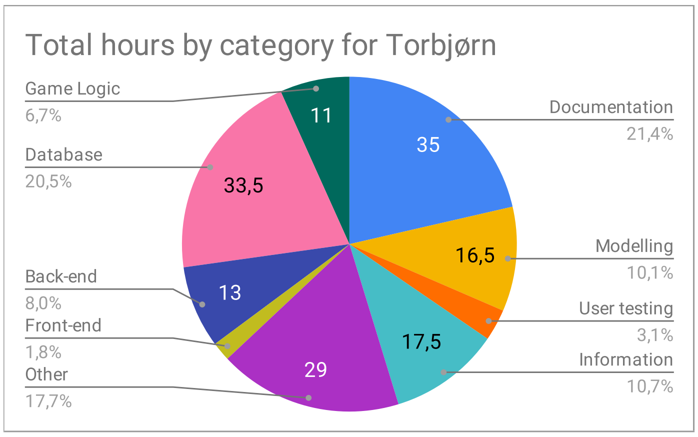
Mikael	
Documentation	22
Preparing presentation	0
Modelling	11,5
Wireframing	7
User testing	1,5
Information search	11,5
Other development	13,5
Front-end	52
Back-end	26
Database	14,5
Game Logic	1,5
Total	161



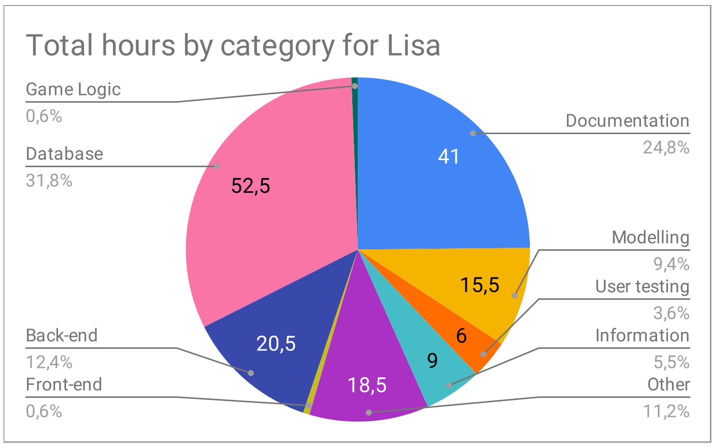
Bård	
Documentation	25
Preparing presentation	0
Modelling	16
Wireframing	2
User testing	1
Information search	14
Other development	10,5
Front-end	36,5
Back-end	40
Database	7
Game Logic	12
Total	164



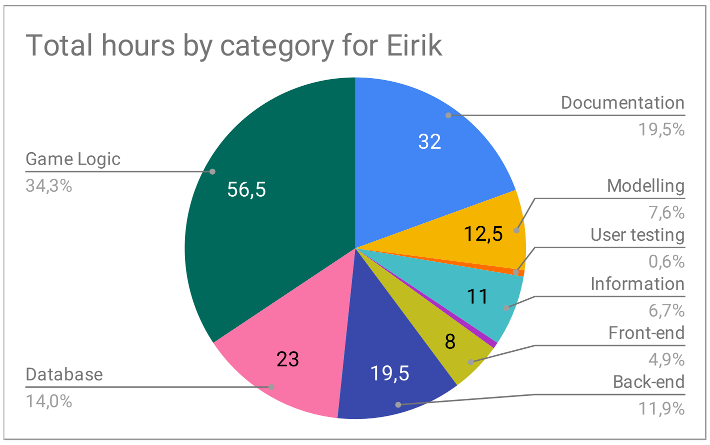
Torbjørn	
Documentation	35
Preparing presentation	0
Modelling	16,5
Wireframing	0
User testing	5
Information search	17,5
Other development	29
Front-end	3
Back-end	13
Database	33,5
Game Logic	11
Total	163,5



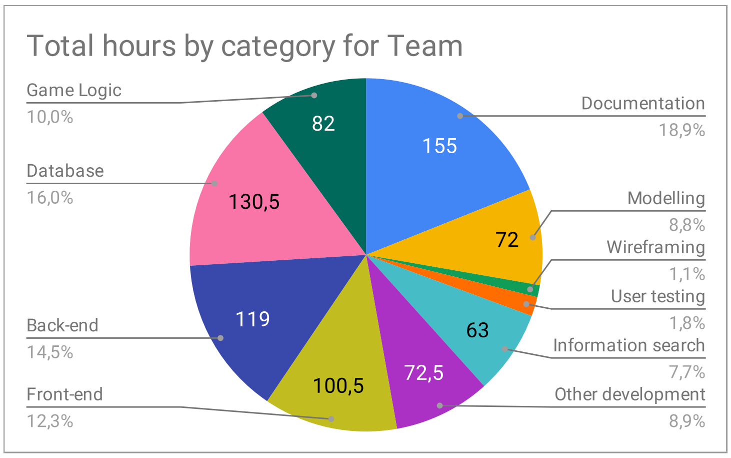
Lisa	
Documentation	41
Preparing presentation	0
Modelling	15,5
Wireframing	0
User testing	6
Information search	9
Other development	18,5
Front-end	1
Back-end	20,5
Database	52,5
Game Logic	1
Total	165

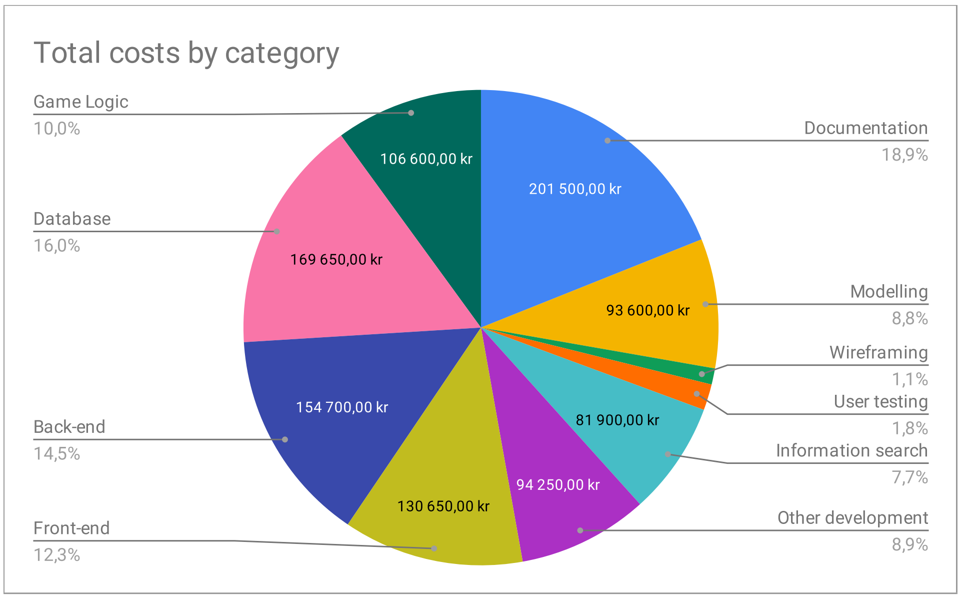


Eirik	
Documentation	**32**
Preparing presentation	**0**
Modelling	**12,5**
Wireframing	**0**
User testing	**1**
Information search	**11**
Other development	**1**
Front-end	**8**
Back-end	**19,5**
Database	**23**
Game Logic	**56,5**
Total	164,5



Team		Costs
Documentation	155	201 500,00 kr
Preparing presentation	0	0,00 kr
Modelling	72	93 600,00 kr
Wireframing	9	11 700,00 kr
User testing	14,5	18 850,00 kr
Information search	63	81 900,00 kr
Other development	72,5	94 250,00 kr
Front-end	100,5	130 650,00 kr
Back-end	119	154 700,00 kr
Database	130,5	169 650,00 kr
Game Logic	82	106 600,00 kr
Total	818	1 063 400,00 kr
		Salary: 1300kr/h





Week 2

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A								0	
B								0	
C								0	
D								0	
E								0	
F								0	
G								0	
H								0	
I								0	
J								0	
K								0	
							Total	0	

Week 3

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A		1						1	
B								0	
C								0	
D								0	
E								0	
F		0,5						0,5	
G								0	
H								0	
I								0	
J								0	
K								0	
							Total	1,5	

Week 7

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A			3	1				4	
B								0	
C								0	
D				2				2	
E								0	
F								0	
G								0	
H								0	
I								0	
J								0	
K								0	
							Total	6	

Week 9

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A			3					3	
B								0	
C								0	
D								0	
E								0	
F								0	
G								0	
H								0	
I								0	
J								0	
K								0	
							Total	3	

Week 10

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A								0	
B								0	
C	6	5	2	1				14	
D								0	
E								0	
F		1	2	0,5				3,5	
G								0	
H				3				3	
I								0	
J								0	
K								0	
							Total	20,5	

Week 11

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A								0	
B								0	
C								0	
D								0	
E								0	
F	1	1						2	
G								0	
H	4	3	3	1				11	
I	2	3	3	1				9	
J								0	
K								0	
							Total	22	

Week 12

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A								0	
B								0	
C								0	
D								0	
E								0	
F	1							1	
G								0	
H	2	3	2	2				9	
I	2	4	2	2				10	
J	3		1					4	
K		1	1	1				3	
							Total	27	

Week 13

		9							
Cat.	**Monday**	**Tuesday**	**Wednesday**	**Thursday**	**Friday**	**Satuday**	**Sunday**	**SUM**	**Notices**
A								0	
B								0	
C	1	1						2	
D								0	
E	0,5							0,5	
F	0,5	1	1					2,5	
G								0	
H	2	3	2	1		1		9	
I	2	2	2	2		1		9	
J	1	1						2	
K		1	1					2	
							Total	27	

Week 14

	9,5	8	4,5	7					
Cat.	**Monday**	**Tuesday**	**Wednesday**	**Thursday**	**Friday**	**Satuday**	**Sunday**	**SUM**	**Notices**
A								0	

B								**0**	
C								**0**	
D								**0**	
E			0,5					**0,5**	
F	1	2	0,5	1				**4,5**	
G	1			1				**2**	
H	2,5	1	0,5					**4**	
I	3	3	2	3				**11**	
J	1							**1**	
K	1	2	1	2				**6**	
							Total	29	

Week 15

8 7 6

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A	4	3	4	6				**17**	
B								**0**	
C								**0**	
D								**0**	
E								**0**	
F								**0**	
G	1,5	4	2	1				**8,5**	Bugfixing & testing
H	0,5							**0,5**	
I	1							**1**	
J								**0**	
K	1							**1**	
							Total	28	

Week 2

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A		0,5						0,5	
B								0	
C								0	
D								0	
E								0	
F		1						1	
G								0	
H								0	
I								0	
J								0	
K								0	
							Total	1,5	

Week 3

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A		1						1	
B								0	
C								0	
D								0	
E								0	
F		0,5						0,5	
G								0	
H								0	
I								0	
J								0	
K								0	
							Total	1,5	

Week 7

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A			3					3	
B								0	
C								0	
D								0	
E								0	
F								0	
G								0	
H								0	
I								0	
J								0	
K								0	
							Total	3	

Week 9

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A			2					2	
B								0	
C								0	
D								0	
E								0	
F			1					1	
G								0	
H								0	
I								0	
J								0	
K								0	
							Total	3	

Week 10

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A	1	2	1					4	
B								0	
C	5	3,5	1	3				12,5	
D								0	
E								0	
F		0,5		0,5				1	
G								0	
H								0	
I								0	
J				1				1	
K								0	
							Total	18,5	

Week 11

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A								0	
B								0	
C								0	
D								0	
E								0	
F		1						1	
G								0	
H								0	
I	2	5						7	
J	5	1		1				7	
K			6	3				9	
							Total	24	

Week 12

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A								0	
B								0	
C								0	
D								0	
E								0	
F								0	
G								0	
H				2				2	
I	1	2	1					4	
J	2	1	2					5	
K	5	5	4	3				17	
							Total	28	

Week 13

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A								0	
B								0	
C								0	
D								0	
E								0	
F	1	0,5	1					2,5	
G								0	
H	2	1		0,5				3,5	
I				1				1	
J	1	1,5						2,5	
K	4	5	5	1,5				15,5	
							Total	25	

Week 14

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A			0,5	1				1,5	

B								**0**	
C								**0**	
D								**0**	
E		1						**1**	
F	0,5			1				**1,5**	
G			0,5	0,5				**1**	
H		1	0,5					**1,5**	
I	0,5	2	3	0,5				**6**	
J		2	0,5	2				**4,5**	
K	1,5	2	4,5	2				**10**	
							Total	**27**	

Week 15

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A	3	4	3,5	8	1,5			**20**	
B								**0**	
C								**0**	
D								**0**	
E								**0**	
F	1		1,5					**2,5**	
G								**0**	
H	1							**1**	
I	1		0,5					**1,5**	
J	1	2						**3**	
K	1	2	2					**5**	
							Total	**33**	

Week 2

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A								0	
B								0	
C								0	
D								0	
E								0	
F								0	
G								0	
H								0	
I								0	
J								0	
K								0	
							Total	0	

Week 3

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A		1						1	
B								0	
C								0	
D								0	
E								0	
F		0,5						0,5	
G								0	
H								0	
I								0	
J								0	
K								0	
							Total	1,5	

Week 7

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A			3	3				6	
B								0	
C								0	
D								0	
E								0	
F								0	
G								0	
H								0	
I								0	
J								0	
K								0	
							Total	6	

Week 9

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A			3					3	
B								0	
C								0	
D								0	
E								0	
F								0	
G								0	
H								0	
I								0	
J								0	
K								0	
							Total	3	

Week 10

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A								0	
B								0	
C	6	4,5	4	1				15,5	
D								0	
E								0	
F		1,5		0,5				2	
G								0	
H								0	
I								0	
J				3				3	
K								0	
							Total	20,5	

Week 11

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A								0	
B								0	
C								0	
D								0	
E								0	
F								0	
G	1	1	1	0,5				3,5	
H								0	
I	6	3	2					11	
J		2		3,5				5,5	
K								0	
							Total	20	

Week 12

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A			1	3				4	
B								0	
C								0	
D								0	
E	1	0,5						1,5	
F	1	1						2	
G	1	0	3					4	
H								0	
I	2	2		2				6	
J	3	4,5	2					9,5	
K								0	
							Total	27	

Week 13

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A	3	1		0,5				4,5	
B								0	
C								0	
D								0	
E	1	0,5	0,5	0,5				2,5	
F								0	
G								0	
H		1						1	
I	1,5	1		1				3,5	
J	2,5	4,5	4,5	1				12,5	
K			1					1	
							Total	25	

Week 14

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A								0	

B								**0**	
C								**0**	
D								**0**	
E	1	1						**2**	
F	1,5	1	1	1				**4,5**	
G				1				**1**	
H								**0**	
I				0				**0**	
J	7	6	3,5	5		0,5		**22**	
K								**0**	
							Total	**29,5**	

Week 15

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A	2	2	9	8	1,5			**22,5**	
B								**0**	
C								**0**	
D								**0**	
E								**0**	
F								**0**	
G	6	4						**10**	
H								**0**	
I								**0**	
J								**0**	
K								**0**	
							Total	**32,5**	

Week 2

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A		0,5						0,5	
B								0	
C								0	
D								0	
E								0	
F		1						1	Searching for inspiration
G								0	
H								0	
I								0	
J								0	
K								0	
							Total	1,5	

Week 3

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A		1						1	
B								0	
C								0	
D								0	
E								0	
F		0,5						0,5	
G								0	
H								0	
I								0	
J								0	
K								0	
							Total	1,5	

Week 7

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A			0,5	0,5				1	
B								0	
C								0	
D			2,5	2				4,5	
E				0,5				0,5	
F								0	
G								0	
H								0	
I								0	
J								0	
K								0	
							Total	6	

Week 9

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A			3					3	Vision document
B								0	
C								0	
D								0	
E								0	
F								0	
G								0	
H								0	
I								0	
J								0	
K								0	
							Total	3	

Week 10

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A								0	
B								0	
C	4	4	2	1				11	
D	1,5							1,5	Final changes
E								0	
F	0,5	1		0,5				2	
G								0	
H		1	2	3			2	8	
I								0	
J								0	
K								0	
							Total	**22,5**	

Week 11

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A								0	
B								0	
C								0	
D								0	
E								0	
F			0,5	0,5				1	
G								0	
H	4	6	5	3				18	
I	0,5	0,5		0,5				1,5	
J	1,5	1,5	0,5					3,5	
K								0	
							Total	**24**	

Week 12

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A								0	
B								0	
C								0	
D	1							1	New lobby feature
E			0,5					0,5	
F	0,5	1						1,5	
G								0	
H	4	4	2,5	1,5				12	
I	0,5	2,5	2	3,5				8,5	
J	2	1	1					4	
K								0	
							Total	**27,5**	

Week 13

	8	7	6	3					
Cat.	**Monday**	**Tuesday**	**Wednesday**	**Thursday**	**Friday**	**Satuday**	**Sunday**	**SUM**	**Notices**
A								0	
B								0	
C	0,5							0,5	
D								0	
E	0,5							0,5	
F	2							2	
G			2	2				4	
H	2	0,5	2	1				5,5	
I	2	4	2					8	
J	1	2						3	
K		0,5						0,5	
							Total	**24**	

Week 14

	9,5	8	4.5	7					
Cat.	**Monday**	**Tuesday**	**Wednesday**	**Thursday**	**Friday**	**Satuday**	**Sunday**	**SUM**	**Notices**
A		0,5		0,5				1	

B								0	
C								0	
D								0	
E								0	
F		1	1	0,5				2,5	
G	1	2		2				5	
H	4	1	0,5	2				7,5	
I	3	2	1	2				8	
J	1	1	2					4	
K	0,5	0,5						1	
							Total	29	

Week 15

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A	4	4	5,5	2				15,5	
B								0	
C								0	
D								0	
E								0	
F		1						1	
G	3	1	0,5					4,5	
H	1							1	
I								0	
J								0	
K								0	
							Total	22	

Week 2

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A			1					1	
B								0	
C								0	
D								0	
E								0	
F			0,5					0,5	
G								0	
H								0	
I								0	
J								0	
K								0	
							Total	1,5	

Week 3

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A		1						1	
B								0	
C								0	
D								0	
E								0	
F		0,5						0,5	
G								0	
H								0	
I								0	
J								0	
K								0	
							Total	1,5	

Week 7

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A			3	3				6	
B								0	
C								0	
D								0	
E								0	
F								0	
G								0	
H								0	
I								0	
J								0	
K								0	
							Total	6	

Week 9

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A			3					3	
B								0	
C								0	
D								0	
E								0	
F								0	
G								0	
H								0	
I								0	
J								0	
K								0	
							Total	3	

Week 10

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A								0	
B								0	
C	4	4	4	4,5				16,5	
D								0	
E								0	
F	2	1						3	
G								0	
H								0	
I								0	
J								0	
K								0	
							Total	19,5	

Week 11

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A								0	
B								0	
C								0	
D								0	
E								0	
F								0	
G	2	2	2	2				8	
H								0	
I	3	2						5	
J	2	3	4	2				11	
K								0	
							Total	24	

Week 12

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A	2							2	
B								0	
C								0	
D								0	
E		2	1					3	
F		2	2	1				5	
G	2	2	2	2				8	
H								0	
I		2	1	1				4	
J	4			1				5	
K								0	
							Total	27	

Week 13

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A	1							1	
B								0	
C								0	
D								0	
E								0	
F	1			1				2	
G	3	2		2				7	
H		1	2					3	
I		2	2					4	
J								0	
K	2	3	2					7	
							Total	24	

Week 14

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A								0	

B								**0**	
C								**0**	
D								**0**	
E	1	1						**2**	
F	1,5	1	1	1				**4,5**	
G				1				**1**	
H								**0**	
I				0				**0**	
J	5	4	3,5	5				**17,5**	
K	2	2						**4**	
							Total	**29**	

Week 15

Cat.	Monday	Tuesday	Wednesday	Thursday	Friday	Satuday	Sunday	SUM	Notices
A	3	**3**	7,5	6	1,5			**21**	
B								**0**	
C								**0**	
D								**0**	
E								**0**	
F		2						**2**	
G	3	2						**5**	
H								**0**	
I								**0**	
J								**0**	
K								**0**	
							Total	**28**	

Status Report

For Bård Hestmark, project TDAT1006

Week 2 (07.01.2019 - 13.01.2019)

Work that was done:
This week the team was assembled.
We prepared a collaboration agreement, discussing what should be in it and such.

Results that were achieved:
Team building, we got to know each other and what each of us want out of this project.

Week 3 (14.01.2019 - 20.01.2019)

Work that was done:
We got a new member and had a brief meeting to introduce each other.

Results that were achieved:
The team got even more familiar, and we made progress on the collaboration agreement.

Week 7 (11.02.2019 - 15.02.2019)

Work that was done:
We got the assignment and started discussing/brainstorming what application to make. It didn't take long before we agreed on making monopoly. We started work on the vision document.

Results that were achieved:
We all agreed on what to do and were all seemingly happy with the decision. We got started working on the project.

Week 8 (18.02.2019 - 22.02.2019)

(Study week) Started learning some javafx since i figured it was beneficial to the project. I started designing a mockup of the board in fx.

Week 9 (25.02.2019 - 03.03.2019)

Work that was done:
We continued to work on and finished the vision document that was due this week. Wrote an agenda for the meeting next week.

Results that were achieved:
Working on the vision document helped us to get a better overview on the project and what to do further.

Plans for next week:
Start modeling the game. We need class & activity diagrams before starting implementation to be able to collaborate optimally. We also have a meeting scheduled.

Week 10 (04.03.2019 - 10.03.2019)

Work that was done:
Began work on different diagrams to get an overview for easier code implementation. We all did these together on a whiteboard, working on the basic diagrams we needed to start coding, and got these finished pretty quickly. Me and Mikael started work on the basic gui. This was mostly a basic game board layout, login and registration. As we had almost no previous experience with javafx and gui on this level in general, this took some time at first, but we quickly picked up the pace.

Results that were achieved:
We ‘drafted’ the diagrams to know what to do and expect, an overview of what to do. We got started coding.

Plans for next week:
We will start coding as much as possible, and get git up and running for everyone so we can collaborate optimally.

Week 11 (11.03.2019 - 17.03.2019)

Work that was done:
This week, we began work on the code implementation. Everyone chose a class from the diagram to construct, and we created these classes based on these classes. Some changes were made underway, as we discovered ways to improve the code structure.

I mostly continued work on the main game board and some other classes related to game gui and logic. I made methods to be able to draw players on the gameboard, and methods to be able to move player pieces around on the boards gridpane. Meanwhile i was making sure to pay attention to and understand what the other members were doing.

Results that were achieved:
We really got started on the application, both gui, game logic and database.

Plans for next week:
We have a meeting with a stakeholder the week after next week. The plan is to have a ‘functional beta’ by then. For that we need to establish communication between gui and logic, and also have multiple players being able to in turn move around the board and updating via the database.

Week 12 (18.03.2019 - 24.03.2019)

Work that was done:
Continued work on implementation. I helped with some database stuff, but mostly continued with making gui and controllers for views which took some time. We also started connecting the gui and game logic classes so we could start running and testing the game. This included making and connecting buttons in the gui to their respective logic, and finishing methods to move players around board when buttons are clicked and updating this to database.

Results that were achieved:
- We were able to make a working prototype
- We now have a lobby allowing for ease of entering games

Problems that occured:
1. Had to find a way for the client to know when it’s their turn, without interfering with game controls.
2. The movement of other players would not update on your own system
3. The JavaFx game board is made up of a gridpane, and to draw objects on this we needed X and Y position for the player moving along the sides of the board.

Solutions:
1. We used a timer thread to check the database and update the gui.
2. I wrote a method to translate a single integer position variable into X and Y position used only by the board.

Plans for next week:
We need to finish the second prototype for the stakeholder meeting, and make as many improvements as we have time for to have it running smoothly. After that we plan on starting finishing/implementing some transactions and property logic.

Week 13 (25.03.2019 - 31.03.2019)

Work that was done:
We had a the stakeholder meeting, and got some ideas on what to to moving forward. I continued adding game/gui functionality like player info, a view containing info about every player in a game and their money and eventually owned properties. This view also contains buttons for rolling dice and thus moving the player piece. And also a button to end the players turn hopefully allowing the user to feel more in control of the game. Onward i started making a static method for drawing property cards to be drawn several placed in the gui. At the end of the week i made views(gui windows) for trading between players, in case we got time to implement this.

Results that were achieved:
- We got the second prototype approved
- Several gui elements finished
- Most of the property logic completed

Problems that occured:
1. We had some trouble with the connection pool filling up and giving exceptions.
2.

Solutions
1. We were not able to locate the problem yet.

Plans for next week:
Our goal for next week is again to finish the implementation of the application, so that we can focus on finishing documentation the last week. I will be away on monday, but I'm planning to spend the evening that day to refactor a lot of the code regarding game logic, as this is quite messy as of now.

Week 14 (01.04.2019 - 07.04.2019)

Work that was done:
I continued my work on the trading screens with controllers, getting them ready for further implementation. I then started working on drawing houses in the gui, which was much harder than expected. We finished the game board and screen, some finishing touches to the board and adding some additional property types like boats and trains. We started implementing rent levels so that rent levels will be different according to e.g how many houses a property has. At the end of the week i made views and controllers for buying houses, implementing houses to the game.

Results that were achieved:
- A lot of cleaning up in the code

- New types of properties
- Completed rent logic
- Jail functionality completed

Problems that occured:

1. The trading data needs to go through the database, showing a view on both the sending and the receiving players' sides.
2. Making boxes for drawing houses on the game board and then 'locating' them by their id in the controller did not work as expected.
3. We still had issues with the connection pool.

Solutions

1. Torbjørn volunteered to give this a try. Although we have a lot more stuff to implement and polish so this function might not make it to the final version of the game.
2. There was too much stuff in the gridpane i originally tried to draw the houses in. I ended up making an entirely new gridpane on top of and the exact same size as the original one, which made drawing houses much easier.
3. This was still not solved.

Plans for next week:
Next week is the last week of the project, so we first and foremost have to finish the application. This means adding remaining functionality; house/hotel purchase and trading of properties. We also have to start writing the main report and other remaining documentation necessary that's necessary.

Week 15 (08.04.2019 - 14.04.2019)

Work that was done:
As this was the last week, we decided we didn't have time to add more functionality to the game. As a replacement for trading i quickly made a view and controller for sending properties and money to another player. The rest of the week mainly consisted of writing the remaining documentation such as the main report, putting some small finishing touches to the game, and bug fixing.

Results that were achieved:

- We finished a version of the game that we were happy with.

Problems that occured:

1. We still had issues with the Connection Pool.
2. Trading like we originally envisioned it would not be finished in time, we got too many exceptions that we were not sure how to handle to make for a smooth experience.

Solutions:

1. Eirik solved this.
2. As trading can be a necessity for a good game of monopoly(ish), i made an alternative not as reliant on the database. As i now had a lot more experience in javafx i got this done very quickly. As a replacement for ‘conditional‘ trading, the players can instead communicate what to trade through the chat.

Status Report

For Eirik Hemstad, project TDAT1006

Week 2 (07.01.2019 - 13.01.2019)

Work that was done:
Team was assembled.
The team met up for the first time and prepared a collaboration agreement. The collaboration agreement was not completed fully.

Results that were achieved:
The team got to know each other, and had some brief discussions about what they wished to achieve with the project in the future.

Week 3 (14.01.2019 - 20.01.2019)

Work that was done:
Lisa Willa was added to the team, and we had a meeting to introduce her to the team. The team spent more time on translating and completing the collaboration agreement.

Results that were achieved:
The team got even more familiar, and we finished the collaboration agreement.

Week 7 (11.02.2019 - 17.02.2019)

Work that was done:
This week, we got the project assignment. We spent most of the time going over the assignment and brainstorming what kind of application we can make. We quickly settled on making a clone of Monopoly, and immediately started working on the Vision Document, to get our ideas into writing. I was away for a day due to sickness.

Results that were achieved:
The team agreed on an overall idea, and started off the design process of the project to a good start.

Week 9 (25.02.2019 - 03.03.2019)

Work that was done:
We worked more on the vision document, as this was due this week. We managed to complete the Vision Document in time, and deliver it within the deadline. We also have a

meeting with one of the stakeholders next week, so we wrote a meeting agenda for this meeting.

Results that were achieved:
The Vision Document was completed and delivered, and we could now start the modelling process.

Plans for next week:
Next week, we wish to start the modelling process. This means we will start creating class diagrams and activity diagrams before beginning the code implementation. A meeting is scheduled next week, for going through the Vision Document with a stakeholder.

Week 10 (04.03.2019 - 10.03.2019)

Work that was done:
We began work on all the diagrams used in planning the code implementation. We quickly completed a temporary class diagram, use case diagram and ER model for the database, so that we had a good starting point for the code and database structure. The activity diagram was also finished to get a good grip of how the main game loop should function. We did all of this together on a whiteboard. We also had a meeting where we got some feedback on the Vision Document and some of the diagrams. Some team members have also begun work on the GUI windows of the game.
Me and Lisa also created the database tables and scripts for intializing these.

Results that were achieved:
We completed the first revision of diagrams, and could finally start coding the solution. We got a good overview of how the game should be designed.

Plans for next week:
Next week, we start on implementing the code, building the database and add functionality to the GUI objects we have so far.

Week 11 (11.03.2019 - 17.03.2019)

Work that was done:
This week, we began work on the code implementation. Everyone chose a class from the diagram to construct, and we created these classes based on these classes. Some changes were made underway, as we discovered ways to improve the code structure. I mostly worked with some of the DAO classes, as well as some utility classes for reading game properties. I also created some pseudocode for the main game loop, to further get some understanding how this process should be programmed. There was a lot of unit testing.

Results that were achieved:

Got the main application structure going, as well as a good database connection foundation for further use in game logic and other operations.

Plans for next week:
The goals for next week are to complete a functional prototype, as we have a meeting with a stakeholder where we are to give a short demonstration of the prototype. This means that we have to create some form of communication between the game GUI element and the logic.

Week 12 (18.03.2019 - 24.03.2019)

Work that was done:
This week, we continued work on the main game functionality. Some of us worked with the main game process, while some worked on the lobby functionality for the game, so that we could launch the game properly with more players. I had finished a prototype for the main game loop, so Bård and I worked together on making the game logic and GUI classes work together so that you could actually run the game in the game window. Our goal for the second prototype was to make up to 4 players be able to play a game together where they could take turns throwing the dice, so we mainly worked on implementing the right buttons and operations to make this possible.

Results that were achieved:
- Got a working prototype in order
- Made it possible for players to join games through the lobby

Problems that occured:
1. Had to find a way for the client to know when it’s their turn, without interfering with game controls.
2. Upon creating a game, all clients would attempt to create a new game, although only one clients needs to do it.
3. The movement of other players would not update on your own system

Solutions:
1. Used a Java Timer thread to run in the background and check for updates in the database
2. Used an algorithm to check if game already is created. If so, just join that game instead
3. Changed the way movement worked on the game board to reflect game data instead of moving the pieces directly

Plans for next week:
Next week we have a prototype meeting with a stakeholder. We may need to do some slight improvements on the prototype before that time. Our goal for next week is to finish a sort of beta version with functional purchasing of properties, rent transactions and jail functionality.

Week 13 (25.03.2019 - 31.03.2019)

Work that was done:
We had a meeting with one of the stakeholders. We got some constructive criticism on some of the diagrams, and our prototype was approved. We spent most of the week improving on the game and adding more functionality. I worked on polishing money transactions and such, as well as completing jail functionality. After some considerations, we decided to add a specific button for ending your own turn, which gave us much more room for performing actions before your turn is over. I worked on implementing this. I spent rest of the week implementing functionality for property purchasing, rent transactions and free parking. Unfortunately, we were not able to reach our goal for a beta version this week, as we still had some work that needed to be done on the game to be playable.

Results that were achieved:
- Prototype got approved
- Polished money transactions
- Property logic got completed
- Support for different rent levels on streets (not implemented yet)

Problems that occured:
1. Connection Pool started getting full for no apparent reason, resulting in a lot of exceptions during runtime. Did not seem to have any direct consequence on the gameplay.
2. There occurred some uncertainty on how the players should be able to perform operations like buying houses and such in the future.
3. Not properly quitting a game, then trying to join a new game would result in you ending up in a game alone

Solutions
1. Tried some debugging on the pool, but the problem is still persisting as of now
2. Added the ‘End turn’ button to give players more freedom during their own turn
3. Made some more changes to the algorithm to ensure a correct create-or-join-game procedure, but there are still some unpredictable problems

Plans for next week:
Our goal for next week is again to finish the implementation of the application, so that we can focus on finishing documentation the last week. I will be away on monday, but hoping to still get some work done.

Week 14 (01.04.2019 - 07.04.2019)

Work that was done:
I started the week by fixing some issues with the rent transactions in the game. Bård and I

also worked on implementing new types of properties, Boats and Trains, that would have more unique functionality from typical streets. We also added support for the different rent levels that each property would have, so that players could upgrade their properties to get more rent from other players. We finished the logic, but the GUI still needs some work to be fully functional. I worked on polishing some game elements, like adding the functionality to pay bail to get out of jail and a unique tile for paying income tax. I had decided that I wanted to refactor all of the game logic aspects of the game, as this area of the application was getting quite messy. This process made the code a lot more understandable, and could be seen as a big improvement.

Results that were achieved:

- A lot of cleaning up in the code
- New types of properties
- Completed rent logic
- Jail functionality completed

Problems that occured:

1. Free parking was not working, as landing on free parking would not let you get out of paying rent next time
2. Issues with joining new games after improperly leaving previous games could still cause issues
3. We still had issues with the connection pool

Solutions

1. Free parking was not stored in database, and would be overwritten every time. Added this attribute in database to fix the issue
2. This was fixed by introducing randomly generated session codes to bind lobbies and games together. We had to use these codes since the lobby database table often gets deleted, thus excluding the option of using foreign keys.
3. We tried increasing the pool size, but this still did not solve the issue. We started thinking that the issues could come from the concurrent threads running in the game.

Plans for next week:

Next week is the last week of the project, so we first and foremost have to finish the application. This means adding remaining functionality; house/hotel purchase and trading of properties. We also have to start writing the main report and other remaining documentation necessary that's necessary.

Week 15 (08.04.2019 - 14.04.2019)

Work that was done:

This week was the last week, so we had put in a last effort of work. Remaining work was mostly finishing the property logic completely, along with trading. We also had to start writing the main report and go over other documentation like user manuals and diagrams. In regards to coding, I was mostly done with my work. Therefore I only spent time refactoring the connection pool to fix its issues, and assisting others in completing their work and testing. I also spent a lot of time writing the user manual and contributing to writing the main report.

Problems that occured:

1. We still had issues with the Connection Pool.

Solutions:

1. I refactored the connection pool to use a built-in pool from the library C3p0, which seemed to fix all issues with the connection pool. This was likely a much better solution instead of the pool we built from scratch.

Status report Lisa Willa

Week 3 (14.01.2019 - 20.01.2019)

Work that was done:
I was added to the team, and we had a meeting in which I was introduced to the team. The team worked on translating and completing the collaboration agreement.

Results that were achieved:
I got to meet the team, and we finished the collaboration agreement.

Plans for next week:
we wait until we have something to do, as we have yet to see the assignment.

Week 7 (11.02.2019 - 17.02.2019)

Work that was done:
This week, we got the project assignment. We spent most of the time going over the assignment and brainstorming what kind of application we wanted to make that was doable. We quickly settled on making a Monopoly like game, and quickly started working on the Vision Document, to get started, on the real planning.

Results that were achieved:
The team agreed on an overall idea, and started off the design process of the project to a good start.

Plans for next week:
in week 9 we will try to complete the vision document and deliver it before the deadline.

Week 9 (25.02.2019 - 03.03.2019)

Work that was done:
We worked more on the vision document, as the deadline is thursday this week. We managed to complete the Vision Document in time, and deliver it within the deadline. We have a meeting with one of the stakeholders next week, so we wrote a meeting agenda for this meeting.

Results that were achieved:
The Vision Document was completed and delivered, and we are now ready to start modelling.

Plans for next week:
Next week, we hope to start the modelling. This means we will start creating domain models, class diagrams and activity diagrams before we start writing code. A meeting is scheduled next week, for going through the Vision Document with a stakeholder.

Week 10 (04.03.2019 - 10.03.2019)

Work that was done:
We began work on all the diagrams used in planning the code implementation. We quickly completed a temporary class diagram, use case diagram and ER model for the database, so that we had a good starting point for the code and database structure. The activity diagram was also made to get a good grip of how the main gamelogic loop should function. We did all of this plenary on a whiteboard. We also had a meeting where we got some useful feedback on the Vision Document and some of the diagrams. Some team members have also begun work on the GUI windows of the game.
Eirik and I also created the database tables and scripts for initializing these.

Results that were achieved:
We completed the first revision of the diagrams, and could finally start developing our game. We got a good overview of how the game should be designed.

Plans for next week:
Next week, we hope to start implementing code, building the database and add functionality to the GUI objects we have so far.

Week 11 (11.03.2019 - 17.03.2019)

Work that was done:
This week, we began work on the code implementation. Everyone chose a class from the diagram to construct, and we created these classes based on these classes. Some changes were made underway, as we discovered ways to improve the code structure. I mostly worked with some of the basic classes like Player and Property. I also started to create some procedures for Property and Player, and started work on their DAO's. There was a lot of unit testing. There was also a lot of testing.

Results that were achieved:
The others got the main application structure going, as well as a good database connection foundation for further use in game logic and other operations. Some entities like Player, and Property, where ready to be used, but are not yet implemented in the GUI. And the the Logging functionality was well under way.

Plans for next week:
The goal for next week is to complete a functional prototype, as we have a meeting with a stakeholder where we are to give a short demonstration of the prototype. This means that

we have to create some form of communication between the game GUI element and the logic.

Week 12 (18.03.2019 - 24.03.2019)

Work that was done:
This week, we continued work on the main game functionality. Some of us worked with the main game process, while some worked on the lobby functionality for the game, so that we could launch the game properly with more players. I mainly worked on game entities and DAO's and procedures so that they would be ready for use. This made me useful, while not "coding in everyone's way".

Results that were achieved:
- We got a working prototype in order
- It is now possible for players to join games through the lobby

Problems that occured:
1. We had to find a way for the client to know when it's their turn, without interfering with game controls.
2. Upon creating a game, all clients would attempt to create a new game, although only one client needs to do it.
3. The movement of other players would not update on your own system

Solutions:
1. They used a Java Timer thread to run in the background and check for updates in the database
2. Used an algorithm to check if game already is created. If so, just join that game instead
3. Changed the way movement worked on the game board to reflect game data instead of moving the pieces directly

Plans for next week:
Next week we have a prototype meeting with a stakeholder. We may need to do some slight improvements on the prototype before that time. Our goal for next week is to finish a sort of beta version with functional purchasing of properties, rent transactions and jail functionality.

Week 13 (25.03.2019 - 31.03.2019)

Work that was done:
We had a meeting with one of the stakeholders, so I fixed some of the documentation for the meeting, mainly textual use-case diagrams. We got some constructive criticism on some of the diagrams, and our prototype was approved. We spent most of the week improving the game and adding more functionality. After some considerations, we decided to add a specific button for ending your own turn, which gave us much more room for performing

actions before your turn is over. I worked on implementing this in the DB. Mikael and I made the chat functionality, he fixed GUI, and I fixed DAO and procedures. I spent rest of the week helping on implementing functionality for property purchasing, and free parking in DB as well as fix stuff that needed to be done or was asked for, mainly in regards to DB. Unfortunately, we were not able to reach our goal for a beta version this week, as we still had some work that needed to be done for the game to be playable, and have the necessary functionality.

Results that were achieved:
- Prototype got approved
- Polished money transactions
- Property logic got completed

Problems that occured:
1. Connection Pool started getting full for no apparent reason, resulting in a lot of exceptions during runtime. Did not seem to have any direct consequence on the gameplay though.
2. There was some uncertainty on how the players should be able to perform operations like buying houses and such in the future.
3. Not properly quitting a game, then trying to join a new game would result in you ending up in a game alone

Solutions
1. Eirik tried some debugging on the pool, but the problem is still there unfortunately
2. Added the ‘End turn’ button to give players more freedom during their own turn
3. Made some more changes to the algorithm to ensure a correct create-or-join-game procedure, but there are still some unpredictable problems

Plans for next week:
Our goal for next week is again to finish the implementation of the application, so that we can focus on finishing documentation the last week, and not be forced to stress.

Week 14 (01.04.2019 - 07.04.2019)

Work that was done:
I worked on Lobby and account functionality, as well as DAO’s. I also spent some time trying to fix the trade functionality. I included the “special” properties in the createPropertys script. I also worked on getting the player scores, with the property values in a procedure and fixing faulty procedures.

Results that were achieved:
- A far cleaner code
- New types of properties
- Complete rent logic
- Jail functionality completed

Problems that occured:

1. Free parking was not working, as landing on free parking would not let you get out of paying rent next time
2. Issues with joining new games after improperly leaving previous games could still cause issues
3. We still had issues with the connection pool

Solutions

1. Free parking was not stored in database, and would be overwritten every time. Added this attribute in database to fix the issue
2. This was fixed by introducing randomly generated session codes to bind lobbies and games together. We had to use these codes since the lobby database table often gets deleted, thus excluding the option of using foreign keys.
3. We tried increasing the pool size, but this still did not solve the issue. We started thinking that the issues could come from the concurrent threads running in the game.

Plans for next week:
Next week is the last week of the project, so we first and foremost have to finish the application. This means adding remaining functionality; house/hotel purchase and finnish trading of properties. We also have to start writing the main report and other remaining documentation necessary that's necessary.

Week 15 (08.04.2019 - 14.04.2019)

Work that was done:
This week was the last week, so we had put in a last effort of work. Remaining work was mostly finishing the property logic completely, along with trading. We also had to start writing the main report and go over other documentation like user manuals and diagrams. Therefore we gave up on completing the trading functionality. I spent a lot of time commenting the sql procedures, and assisting others with their problems. I also spent time contributing to writing the main report, use-case, and fixing the domain model, and class diagram.

Problems that occured:

1. We still had issues with the Connection Pool.
2. The documentation combined with the final fixes unfortunately took as much time as I had feared.

Solutions:

1. Eirik refactored the connection pool to use a built-in pool from the library C3p0, which seemed to fix all issues with the connection pool. This was likely a far better solution instead of the pool that was built from scratch.

Plans for next week: the team will assemble some time before the presentation to prepare.

Status Report

For Mikael Kalstad, project TDAT1006

Week 2 (07.01.2019 - 13.01.2019)

Work that was done:
This week the team was founded. On our first day as a team, we created and agreed on a collaboration agreement. We did not fully complete the agreement since we had other work that had higher priority. Also, the project was a couple of weeks away so we had a good time to finish this agreement.

Results that were achieved:
- A team for the project was founded.
- A partial collaboration agreement.

Week 3 (14.01.2019 - 20.01.2019)

Work that was done:
We got a new team member this week; Lisa. We also finished the collaboration agreement. This was done during one meeting, where we also introduced Lisa to our team.

Results that were achieved:
- The team members became more familiar with each other.
- A new team received a new team member.
- We finished the collaboration agreement.

Week 7 (11.02.2019 - 17.02.2019)

Work that was done:
This week marked the start of the project. The team got introduced to our project assignment during the project kickoff. After some discussions and brainstorming, we decided to make our own version of monopoly. After this, we began working on the vision document. I also started to sketch a design for our application. We needed design

Results that were achieved:
- The team decided what kind of application we wanted to make; Monopoly.
- We began working on the vision document.
- I began sketching designs.

Week 9 (25.02.2019 - 03.03.2019)

Work that was done:
We finished the vision document since it was due this week. The team did work on anything else since we were busy with other subjects.

Results that were achieved:
- The team finished the vision document.

Plans for next week:
The plans for next week is to begin the process of developing different schematics for our system and application. We also have a meeting with a stakeholder where we will show our vision document and the schematics we will work on next week.

Week 10 (04.03.2019 - 10.03.2019)

Work that was done:
This week we began the planning for the project. This involved discussing features, structures, team roles, time management and creating diagrams. During this week we created a class diagram, use case diagram and an ER model for the database. We also created an activity diagram.

After we had created these diagrams and discussed different aspects of the game and structure of the code, I started working on the GUI for the application. We had a vision of how we wanted the application to look with the design sketches, but actually creating these views was difficult. I, therefore, started to research JavaFX and how to use SceneBuilder.

We also had a meeting with the stakeholder were we showed our vision document and some of the other diagrams we made.

Results that were achieved:
- The team started planning.
- We made diagrams (first revision).
- I started to research into how to create the GUI.

Plans for next week:
Our plans for next week are to start the development of the system. I will be focusing on implementing the GUI for the system.

Week 11 (11.03.2019 - 17.03.2019)

Work that was done:
I mainly worked on the GUI this week as planned. I learned how to use Scene Builder and how to connect a .fxml file to a controller class. I also found a smart solution to switching scenes/views, by creating a scene manager. The rest of the team started developing other aspects of the system. Some started with the database and the DAO classes. We also implemented an prototype of the login feature which is connected to the database.

Results that were achieved:
- Basic GUI for the application with scene switching.
- Basic database structure.
- Database connection.
- An early prototype of the login feature.

Problems and solutions:
1. Switching scenes in the GUI without creating new windows for each scene/view.
2. Run some startup code when scene mounts/loads.

Solutions:
1. Created a scene manager that handles all scene switching without creating new windows.
2. Use a @FXML method called initialize, which will always run upon mount/load. It will only run once.

Plans for next week:
Next week our main goal is to finish a prototype of the application. We have a meeting with the stakeholder where will give a presentation of this prototype.

Week 12 (18.03.2019 - 24.03.2019)

Work that was done:
This week I worked on implementing a lobby in the application. I wanted to get more experience with the SQL, procedures and DAO classes therefore I implemented most of the logic for this feature back- and front-end. The other members on the team have started working more on the game logic and the GUI for the game.

The lobby feature was buggy in the start of the week, but I managed to make it stable towards the end of the week. The platform around the game is now getting along nicely, there are still some bugs, but it starting to be stable. I also worked on several minor bugs in the front-end of the application that needed to be fixed.

Problems and solutions:

1. Dynamically update lobbies with changes in the database.
2. Change status and buttons dynamically based on if the lobby is open, full or in-game.

Solutions:

1. Create a draw method and periodically update all lobbies with new data from database with a timer. A refresh button was used originally, but was removed since it was inconvenient and not useful with a timer.
2. Create methods for changing the buttons and logic for when the lobby is open, full or in-game. This is updated in the same time as the lobbies refresh.

Results that were achieved:

- A working lobby where players can join each other.
- Bug fixes in front-end.
- An early prototype of a working game with several players.

Plans for next week:
The main event that we need to prepare for next week is the meeting with the stakeholder. We need a working prototype to be ready for this meeting. I also plan to start working more on the game since the platform around the game is now beginning to be finished.

Week 13 (25.03.2019 - 31.03.2019)

Work that was done:
At the start of the week we had the meeting with the stakeholder. The meeting went well, and our prototype worked just as expected. After this we continued to add features. At the meeting we received some feedback, one the things that they mentioned was a chat feature. Therefore, I began to work on this feature.

I have become more comfortable and experienced with front-end and JavaFX, so it was easy and fast to implement this feature in the application. I spent some time polishing this chat feature to make sure it was working optimally with the database. In addition to this, I started to work more on the game. I redesigned some aspects in the game view, and added a dynamic GUI logic for the opponents in the sidebar.

Problems and solutions:

1. Make the newest chat messages show, rather than the ones that were first added.
2. Having the ability to hide the chat if the user does not want to see the messages.
3. Synchronize the player colors between different users.

Solutions:

1. Made sure to always scroll down to the bottom of the ScrollPane where the newest messages would appear.
2. By using a translate method, I made the chat move up and down when clicking on the top.

3. Save a list of the different colors that each user had in the lobby, so they would have the same color when they started the game.

Results that were achieved:
- First prototype was finished and approved by stakeholder.
- Chat feature was added.
- The game logic and GUI has been improved.

Plans for next week:
Next week we want to finish the main features of the application. We want to focus on documentation and bug fixing the last week of the project. Therefore we have set a goal to have a stable and working version of the application by the end of the week. We have also planned to maybe meet and work in the weekend to be able to accomplish this goal.

Week 14 (01.04.2019 - 07.04.2019)

Work that was done:
This week I only worked on the game part of the application. There were a lot of GUI that was not finished or not working properly in the game, therefore a lot of time was spent in fixing minor GUI elements. I worked on a message popup feature for the game. It was a bit tricky to get this to work properly, but after some refactoring the feature was stable and working.

I also worked on some game logic, especially the game logic interconnected with the GUI. Buttons, text and other different elements needed to update dynamically according to the game logic. The code for doing was working at some point, but I wanted to refactor this to make it more understandable and easier to add new features and fix bugs.

Results that were achieved:
- More game features were added, the game logic was updated.
- More GUI elements in the game that is dynamic.
- Better and more structured code in game controller.

Problems that occured:
1. Timer issues where some timers would not stop. Also some methods created new timer periodically, which filled our connection pool size.
2. Show more clearly that a player, or opponent, is in prison or bankrupt.

Solutions
1. Removed timers from methods that was called multiple times, or making sure they were stopped before initiating a new one. I also added timer cancel and purge functions to every timer with issues.
2. Added a descriptive image on top of the color for each players that dynamically appears if a player e.g. is in jail.

Plans for next week:
The goal for last week was not accomplished, so we have to focus on getting the application finished. However, we also need to start working on the different types of documentation that is required.

Week 15 (08.04.2019 - 14.04.2019)

Work that was done:
Since this is the last week of the project we had to really work hard to finish the application and the required documentation. We started writing the main report for the project. In addition to this we looked over the WIKI on gitlab and added/updated documents. I spent some time formatting our time distribution sheet to make it more clear how many hours we have spent on each category during the project.

I also spent a lot of time fixing bugs in the application. Furthermore, I helped implement the GUI feature for pawning properties, and buying houses and hotels on properties. The last GUI I added to application was a dialog showing who won the game when the game was finished.

The rest of week I worked with the rest of the team to finish all the documentation that is required. I mostly worked on writing the main report, and also refactoring the time distribution sheet to display the data more visually with diagrams. I also refactored the gui package in the application. I separated components and views, making the package structure better overall.

Results that were achieved:
- Completed the main report
- Completed all documentation, javadocs and comments in code.
- Changed package structure in gui.
- Final release of the application.
- The assignment was ended/finished.

Problems that occured:
1. Forfeit feature was not working properly. It worked fine the first time, but the players were not synchronized as expected the second time.

Solutions:
1. I updated the synchronization between players when forfeiting. There were also an issue in a procedure that Lisa Willa fixed.

Status Report

For Torbjørn Bøe Lauvvik, project TDAT1006

Week 2 (07.01.2019 - 13.01.2019)

Work that was done:
The team was compiled. We started on the collaboration agreement.

Results that were achieved:
We got to know each other

Week 3 (14.01.2019 - 20.01.2019)

Work that was done:
Lisa joined the team and we got to know each other.

Results that were achieved:
Lisa joining the team.

Plans for next week:
Teamwork is postponed until project kickoff week 7

Week 7 (11.02.2019 - 17.02.2019)

Work that was done:
Project kickoff. We did a lot of brainstorming and decided what game we should make and we started on the documentation.

Results that were achieved:
Agreed to make a board game somewhat based on Monopoly

Plans for next week:
Winter break

Week 9 (25.02.2019 - 03.03.2019)

Work that was done:
We finished the vision document.

Results that were achieved:

Finished vision document

Week 10 (04.03.2019 - 10.03.2019)

Work that was done:
We made a lot of diagrams such as class diagram, use-case diagram, ER diagram and more.

Results that were achieved:
Made diagrams
Did modelling

Week 11 (11.03.2019 - 17.03.2019)

Work that was done:
I added account procedures and did some testing regarding hashing and salting passwords and handling authentication. We started using a standard naming scheme for all stored procedures to keep it organized. Tried using error handling in the database but it did not function as intended.

Results that were achieved:
Password hashing completed

Week 12 (18.03.2019 - 24.03.2019)

Work that was done:
This week, we continued work on the main game functionality.

Results that were achieved:
The prototype was finished.

Week 13 (25.03.2019 - 31.03.2019)

Work that was done:
Status meeting with one of the stakeholders. Our prototype got approved and we continued working on the main product.

Results that were achieved:
Prototype approved

Plans for next week:
Get a good start on the reports and documentation.

Week 14 (1.04.2019 - 07.04.2019)

Work that was done:
This week we started to get good control of the documentation. I started to work on adding a trading function to the game using the database to store the money and transactions to be done.

Results that were achieved:
Got a good start on the documentation.

Plans for next week:
Complete the product and deliver it.

Week 15 (08.04.2019 - 14.04.2019)

Work that was done:
I added a GitLab CI file for automatic publishing of the JavaDoc when changes were made to it. We worked on finishing the documentation and make the application into a final product. The trading was dropped due to it not functioning as intended.I tried to pack the application into a jar-file.

Results that were achieved:
Game was completed
Documentation was completed
Project was delivered

Plans for next week:
Easter egg hunting.

Usability test 1 Mockups

Date: 17.02.2019
Name: Kristian Hansen
Age: 22
Field of study: Communication Technology at NTNU

Tasks to complete:
Note: this test is conducted with the mockups, there is no application made for the test subject.

Task number	Description	Details
Task 1	Register new user	User tries to register a new user
Task 2	Login	User tries to login with his registered user
Task 3	Start a new game	User tries to start a new game
Task 4	Roll dice	User tries to roll dice
Task 5	See property information	User tries to see his property information
Task 6	Buy property	User tries to buy a property

Tasks	1 - Very Hard	2 - Hard	3 - Medium	4 - Easy	5 - Very Easy	Comments
Task 1			X			Make button more visible
Task 2					X	Good overview
Task 3			X			Add text to show where to start
Task 4				X		
Task 5		X				Add icon to indicate property
Task 6					X	

Additional comments	Good	Bad
Comments on GUI	Design looks nice	Some features are unclear

Usability test 2 MVP

Date: 15.03.2019
Name: Jon
Age: 22
Field of study: Chemistry

Tasks to complete:
Note: this test is conducted with the application the team has developed. The goal for this test is to optimize our design for the minimal viable product.

Task number	Description	Details
Task 1	Register new user	User tries to register a new user
Task 2	Login	User tries to login with his registered user
Task 3	Start a new game	User tries to start a new game
Task 4	Roll dice	User tries to roll dice to move piece

Tasks	1 - Very Hard	2 - Hard	3 - Medium	4 - Easy	5 - Very Easy	Comments
Task 1				X		Design is easy to follow
Task 2					X	
Task 3			X			Could add lobby feature
Task 4				X		
Task 5				X		

Additional comments	Good	Bad
Comments on GUI	Good overview	No lobby feature
Comments on Game Logic		Unclear when turn ends