Vision Document

Monopoly

IDI, NTNU

REVISION HISTORY

Date	Version	Comment	Author
13.02.2019	0.1	Initial structuring	Team 4
14.02.2019	0.2	First draft	Team 4
27.02.2019	1.0	First draft final version	Team 4
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

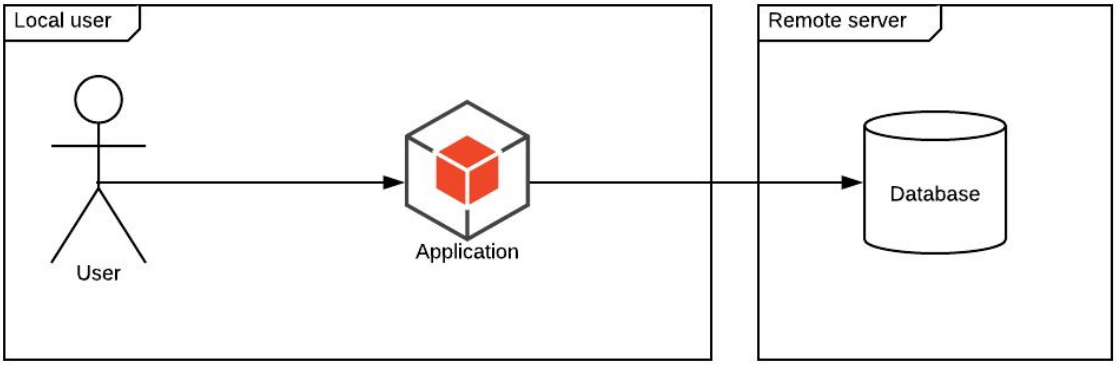
This is a desktop application, you need JRE(java runtime environment), or jPortable and jPortable launcher to run the application. The user also needs to be connected to the internet to properly use the application.

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

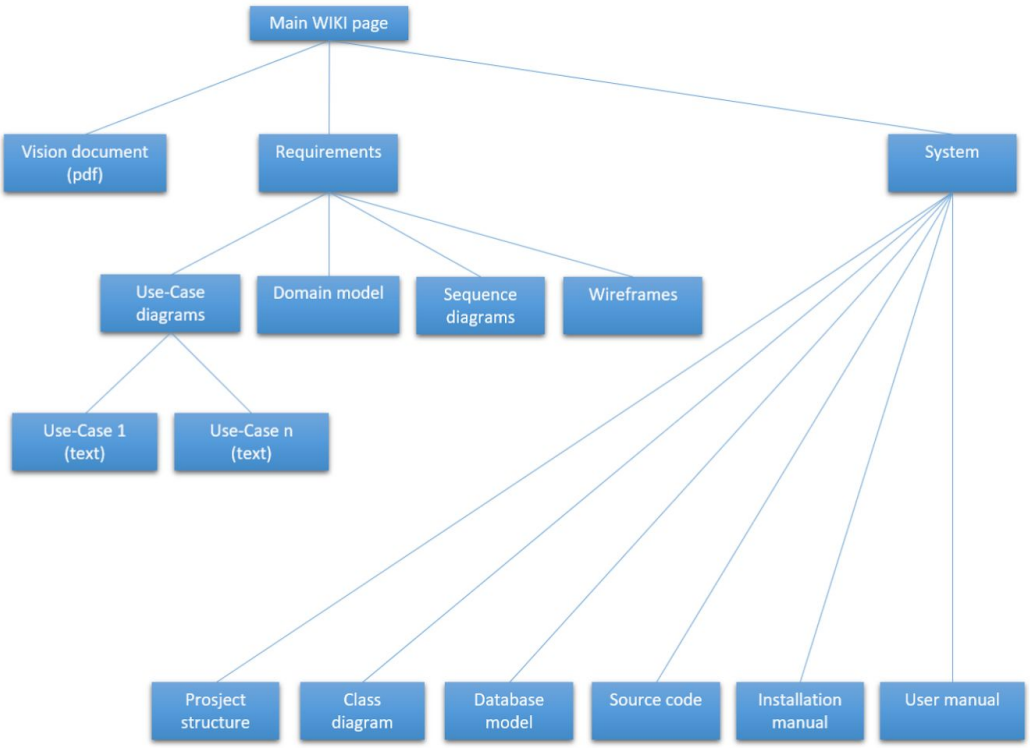
To prevent risky situations we will incorporate good routines:

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