Abstract

Our vision is to create an immersive text-based role-playing game. This report showcases the design and thought process from the initial idea to the conclusion of the project - a demo version of the game. The design process was done utilizing the knowledge we learned during the first semester of DAT18C at KEA. Such as Unified Processes and OOP in JAVA with JGrasp as the environment. With the short time frame taken into account, limitations were created and we chose to exclude several features from the demo. /written by Søren

Text Based RPG

Modul Opgave 4

Frederik Lundbeck, Marcus Bender, Nicklas List, Søren Carlsen
DAT18C

7 December 2018

Contents

1. Introduction

In this report we want to bring you through the different phases of our project and how we have documented our work process. First, a short introduction to the demands we had, which made the framework for software. Afterwards we will give you an insight into our thought process, together with the models we have created under the UML phase of our study. /written by Marcus

2. Design

2.1 Glossary

OOP : Object Oriented Programming
OS-type : Operating system
GPU : Graphic card
bug : An error, flaw, failure.
NPC : Non Player Character.
RPG : Role Playing Game.

2.2 ITO

2.2.1 Vision/Mission

Business vision

To create an immersive adventure experience.

Busines s Mission

To develop well documented OOP code as a solid foundation for a fun and exciting role playing game.

2.2.2 SWOT-analysis

Strengths
Strong communication and collaboration will ensure a productive way of tackling problems/challenges that might occur. The two supervisors available for support will guide us through the project. The participants within the group are all quite fond of adventure games. Minimal activity besides the study allows us to work on the project during 'freetime'.

Weakness
Short time to develop the game. We're not a well established game company. Breach of student contract. Unexpected sickness etc. To reach our goal we need everyone to work as planned Our end product is not good enough, the user wouldn't find the game interesting.

Opportunities
Further development after time frame is ceased. Develop our own skills within game development. Expansion to other platforms.

Threats
The population of people playing text-based adventure games is declining. It's a saturated marked. With a lot of competition in this niche.

2.3 FURPS

- **F**unctionality:
    1. **OS-Compatibility:**
        - When running the program your OS-type is stored and used for important operations such as writing and naming new files etc.
    2. **Save/Load:**
        - While playing the player is able to save/load old game states.
    3. **Options:**
        - The player will be able to customize settings such as volume, language and font size.
    4. **Commands:**
        - Commands will be how the user input text that will be parsed to actions and interact with the game
    5. **Item Handling:**
        - The player will be able to pick up items and store them in an inventory with limited space.
    6. **Exit:**
        - The player will be able to exit the game at any point.

- **U**sability:
    1. **Console-Prompt:**
        - The game is played in console, that will support basic things in the game such as taking input from the player and displaying it.
    2. **Configurability:**
        - Choosing to turn on music or not
        - Safe file handling
        - Color scheme
        - Language
        - Volume tweaking
        - Text size tweaking
    3. **Help:**
        - Players can input -help and receive all commands and tips.
- **R**eliability:
    1. **Crash Handling:**
        - In order to handle exceptions we will write safe code that will prevent a lot of otherwise game-breaking crashes, such as handling when a file doesn't exist
- **P**erformance:
    1. **Graphics rendering:**
        - Because the game is text based only, we won't need to 3D render graphics to screen, so the player will not need a powerful GPU in order to play the game.
- **S**upportability:
    1. **Crash/bug reporting:**
        - If the program throws an exception, it will be displayed so the user can report or look up possible solutions/fixes.
        - A customer support line.

2.4 Use Cases

UC#1 - New game
Player can choose to start a new game from the startup menu.

UC#2 - Load game
Player can choose to start an existing game from a save file.

UC#3 - Game
The main story of the game will be handled through game objects. The game holds the story logic.

UC#4 - Dialog
The dialog is a fundamental element within the game and will serve as the main communicator between it and the player.

UC#5 - Commands
Commands will be how the user input text that will be parsed to actions and interact with the game.
UC#6 - Item Handling
The player will be able to pick up items and store them in an inventory with limited space. Only allowing the player to carry a certain amount of items.

UC#7 - Description
Every item, location, NPC will have some sort of description to explain/immerse the player into the location/setting.

UC#8 - Light
Light will be a factor when deciding if an item, location, NPC is visible to the player.

UC#9 - Options
Options should be available to the player at all times during a session and will allow the player to turn up or lower sound volume, save game and exit game.

UC#10 - Save - (Fully dressed)

Property	Description
Use Case Name	UC#10 - Save
Scope	Application, save folder directory
Primary Actor	Player
Stakeholders and Interests	We as developers are stakeholders because this function working or not working could throw off potential players. Players are interests because, they expect a save/load function within the game.
Preconditions	Enough space on the hard drive to successfully save the game. Existing save file to override.
Success Guarantee	Player can save the game without any corrupted files. Player can load the game successfully and start where his last save left him.
Main Success Scenario	1. Player chooses between 3 possible save slots. 2. Game successfully saves game progress. 3. Player keeps playing or exits the game.
Extensions	**1.a Save file can not be found (missing)** **-** The player will need to start the game from beginning. **2.a When player starts from a saved game, it doesn't load.** **-** Prompt player to choose from latest working autosave (some progress might be lost). **3.a Player is not able to save the game when exiting program.** **-** To minimize losses we are implementing a autosave function that will automatically save the game whenever a mission is completed or milestone reached.

UC#11 - Volume
Extension of UC#9. The player will be able to turn up/down the volume.

UC#12 - Exit
Extension of UC#9. The player will be able to exit the game at any point.

UC#13 - Crash log
If the game crashes a log file will store the crash message.

2.5 Limitations

Shortly after we got our assignment we noticed that the size and extent could be huge, so we had to limit ourselves. We came up with a time schedule in form of a Gantt-chart that we followed almost all the way through. After forming the first iteration of our use cases, where ideas were thrown around and the bar sat high, we quickly found out that we had a too big scope and wouldn't be able to showcase a working product within the time limit. After the first iteration, we had to cut off some of our more extravagant ideas and focus on being more realistic. That's why we chose to make a demo game, where we could showcase how the game mechanics will work, without all the extravagant features we talked about, such as save game, load game, and options. In the second iteration, we got a better hold of the size and could shortly after start forming our adventure game. by Frederik

2.6 Diagrams

2.6.1 UC diagram

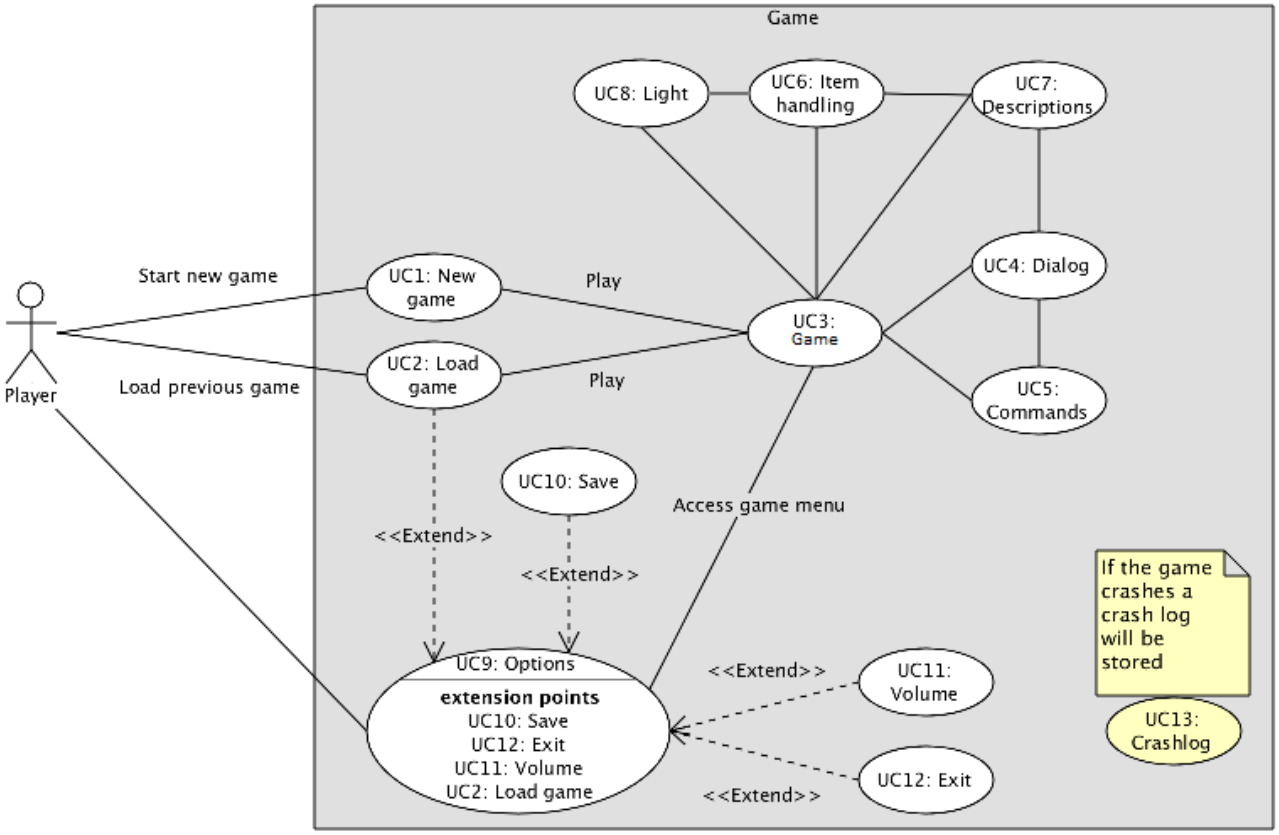

Figure 2.1: UC Diagram

2.6.2 Domain model

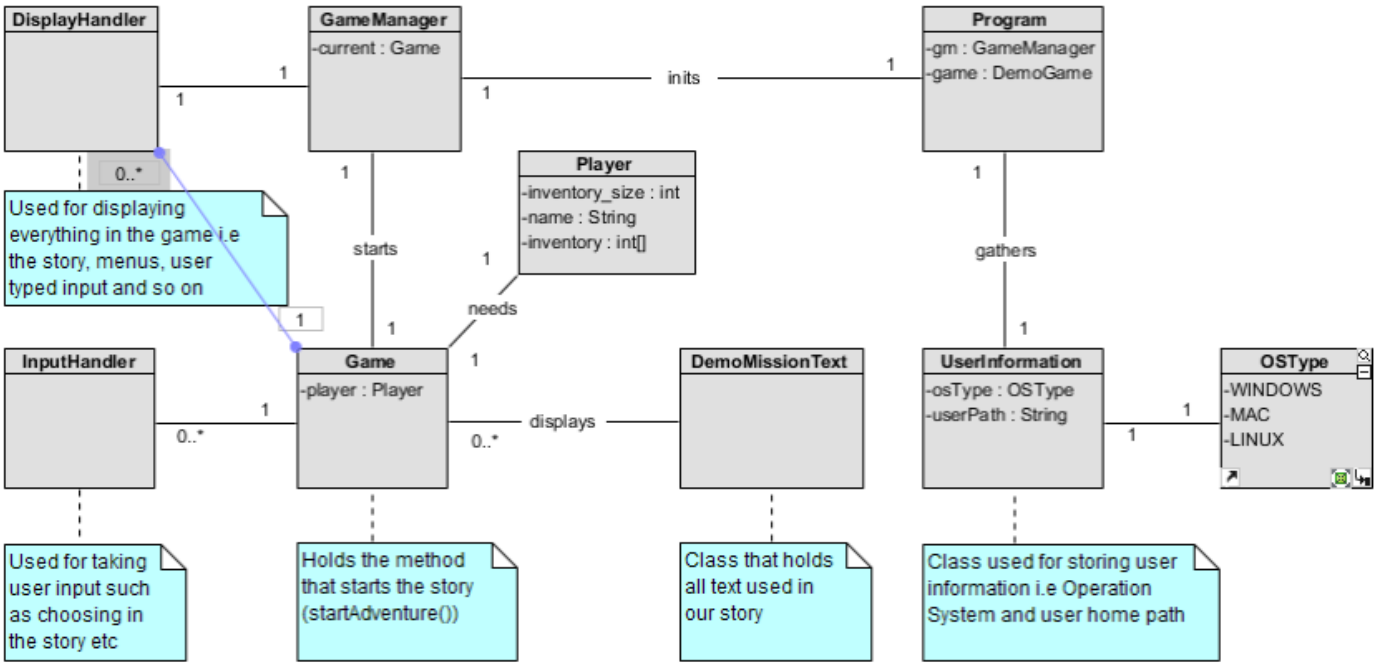

Figure 2.2: Domain Model

2.6.3 Class diagram

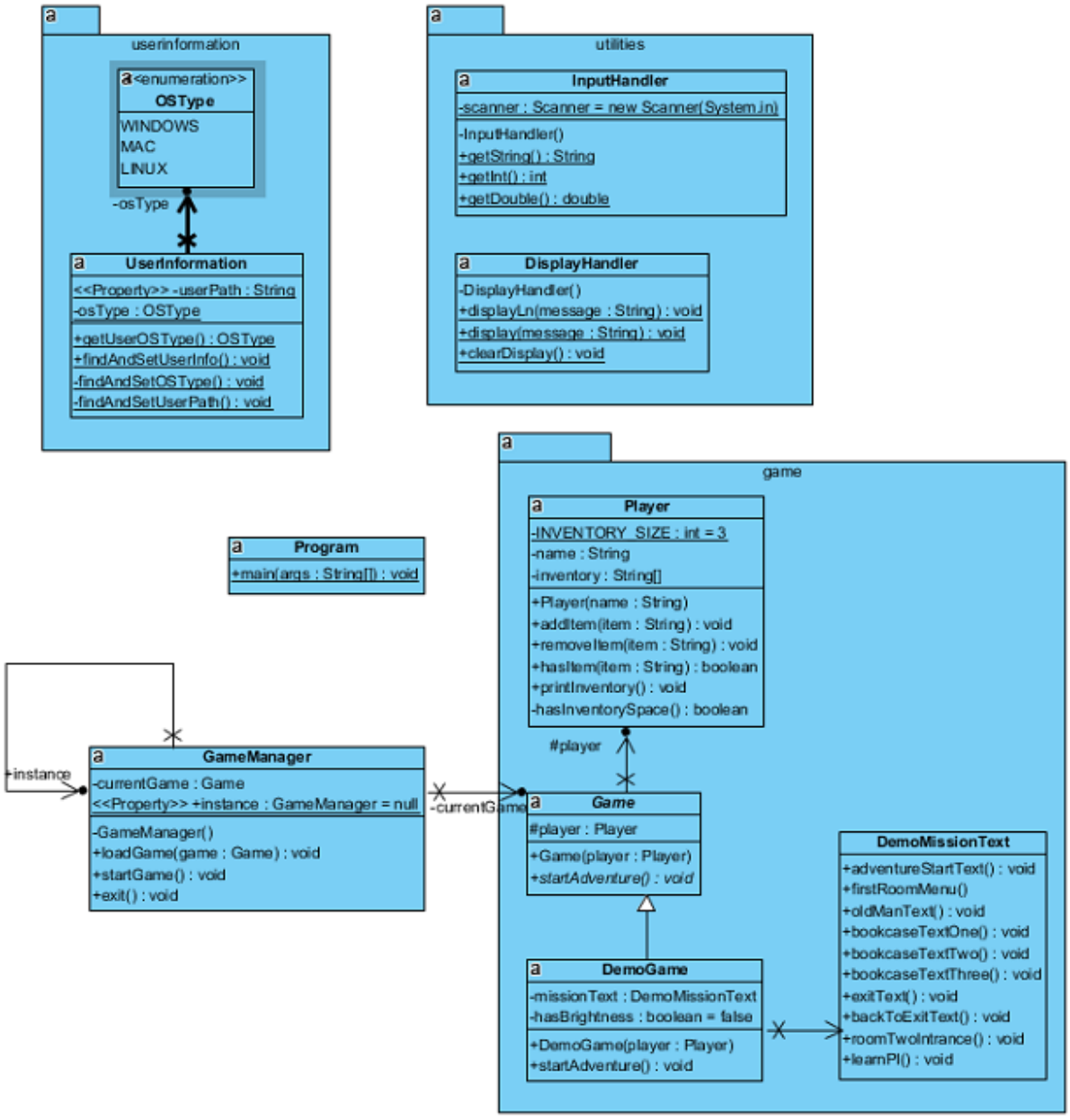

Figure 2.3: Class Diagram

2.6.4 SSD

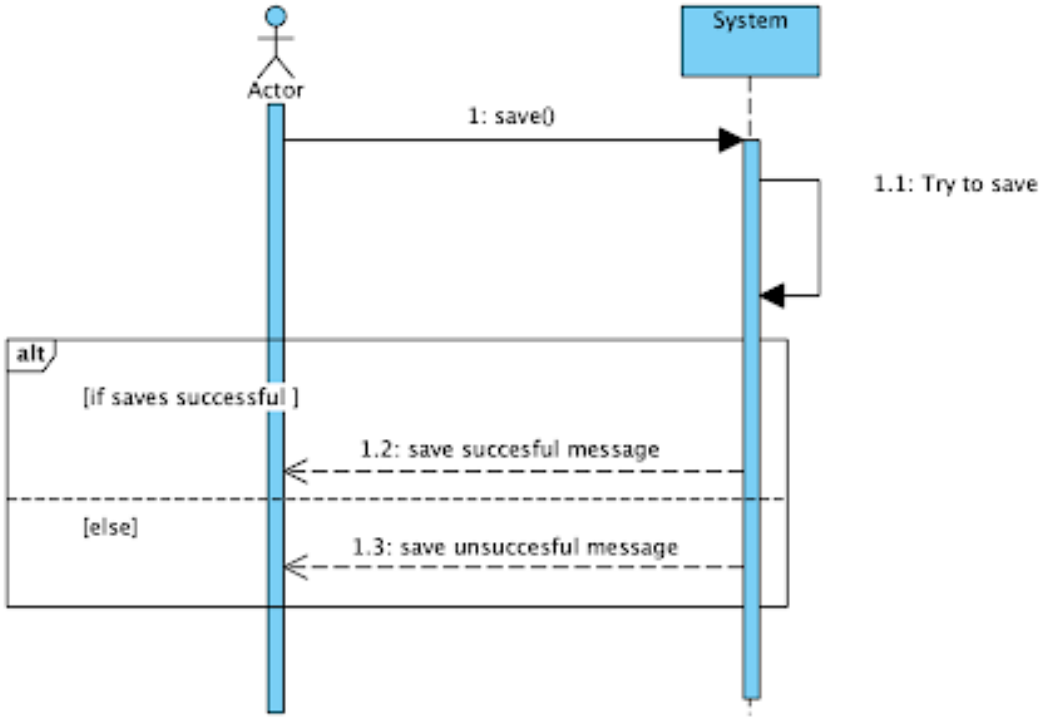

Figure 2.4: SSD

2.6.5 SD

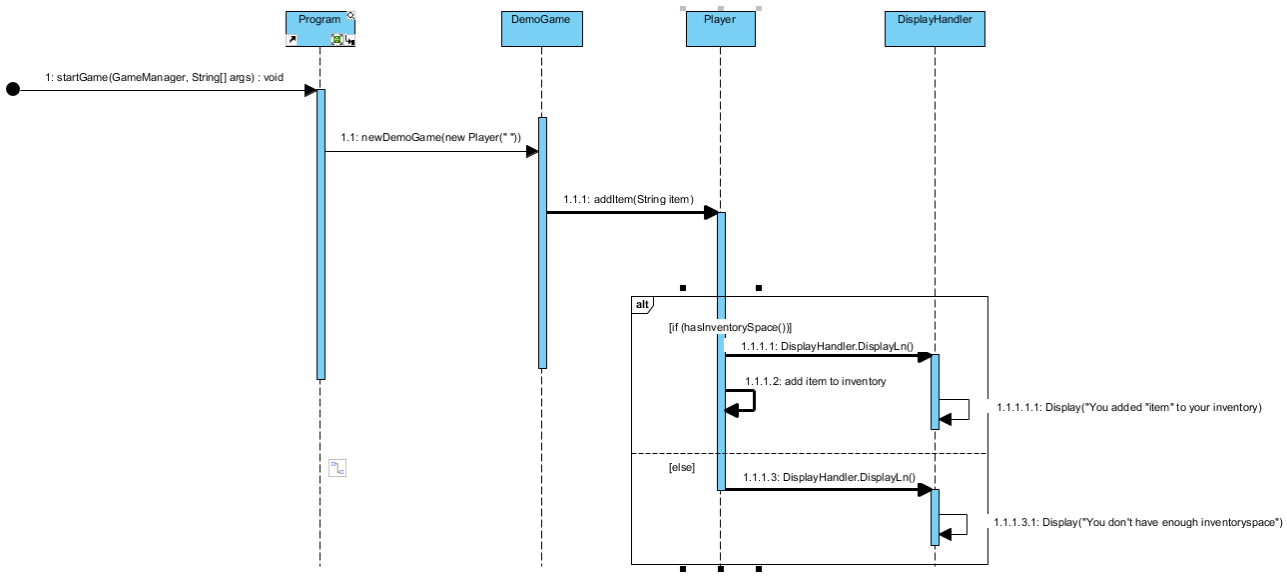

Figure 2.5: SD

3. Conclusion

Through this project, we have learned some important skills we can bring with us to our next project. We have learned how much time you actually have to use, just discussing your ideas and thoughts, about how the program should look like when we finish. From this aspect, we can conclude that structuring your time is key to making a good well-documented program. Other than that we can conclude that working in an agile environment have been essential to forming our program the best way we could. We have taken a huge leap in our personal development, especially when we look back at the UML phase, we've gotten a lot stronger using diagrams to explain our intentions the best way can. /written by Nicklas

4. Appendix

4.1 Gantt Chart

TASK NAME	START DATE	DAY OF MONTH*	END DATE	DURATION* (WORK DAYS)	DAYS COMPLETE*	DAYS REMAINING*	TEAM MEMBER	PERCENT COMPLETE
First Sample Project								
Gennemgå kontrakt	11/21	21	11/21	1	1	0		100%
SWOT analyse	11/21	21	11/22	2	2	0		100%
Buisness Vision	11/21	21	11/22	2	2	0		100%
Use cases	11/21	21	11/26	6	6	0		100%
Gantt-Chart	11/21	21	12/7	17	17	0		100%
Second Sample Project								
Use case Diagram	11/22	22	11/30	9	9	0		100%
Use case fully dressed	11/23	23	11/28	6	6	0		100%
Plan over første instans	11/26	26	11/27	2	2	0		100%
Domain model	11/26	26	11/28	3	3	0		100%
Third Sample Project								
SD diagram	11/27	27	11/28	2	2	0		100%
Klasse diagram 1. iteration	11/27	27	11/28	2	2	0		100%
SSD diagram	11/28	28	11/30	3	3	0		100%
Oversættelser	11/28	28	11/29	2	2	0		100%
Aflever 1 iteration af design	11/21	21	11/30	10	10	0		100%
Fourth Sample Project								
Kanaban på trello	11/22	22	11/30	9	9	0		100%
Udform og Færdiggør koden	11/30	30	12/7	8	8	0		100%
Færdiggør rapport	12/6	6	12/7	2	2	0		100%
	12/6	6	12/7	2	2	0		100%

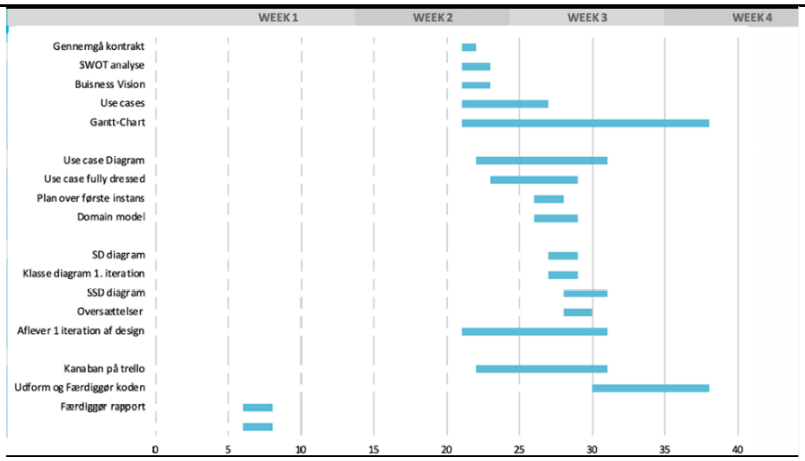

Figure 4.1: Gantt Chart

4.2 Kanban Iterations

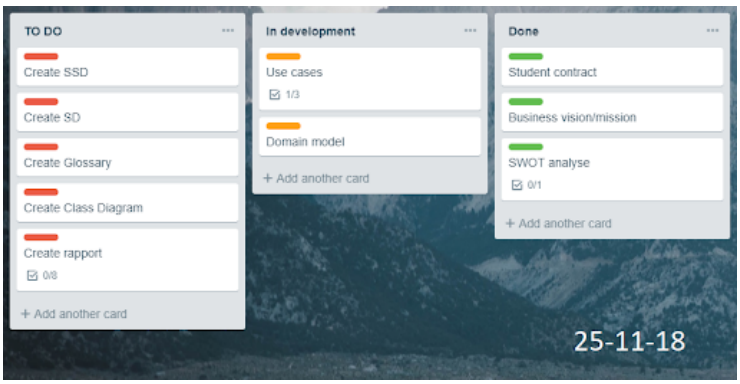

Figure 4.2: Kanban 1

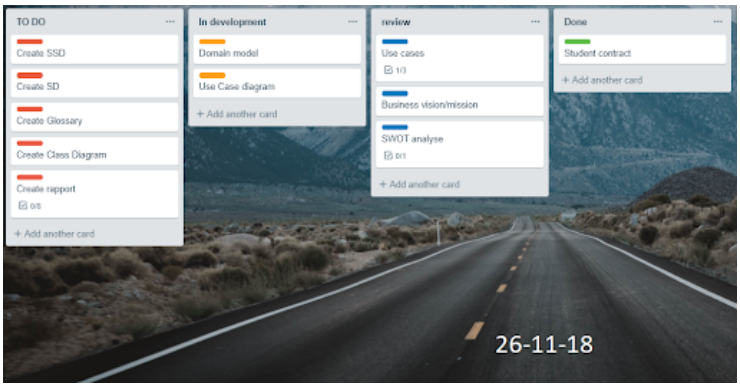

Figure 4.3: Kanban 2

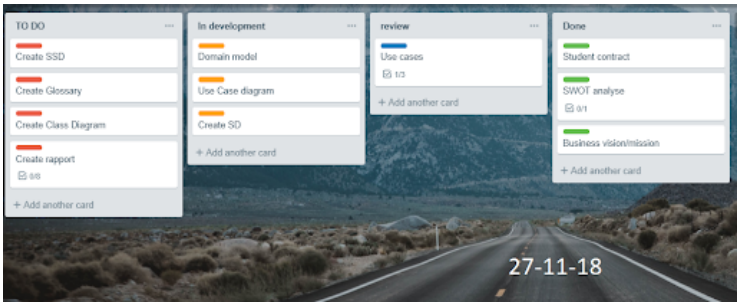

Figure 4.4: Kanban 3

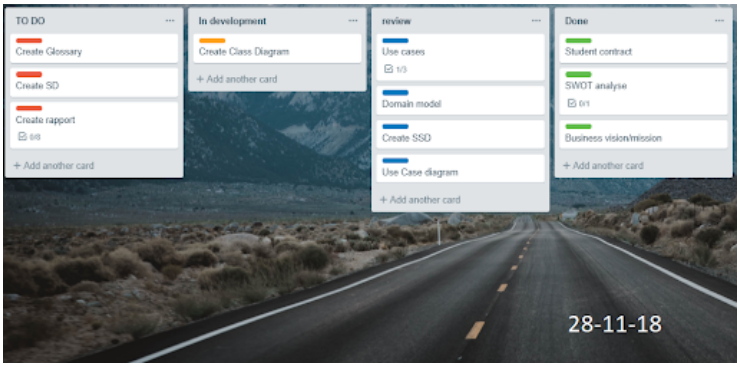

Figure 4.5: Kanban 4

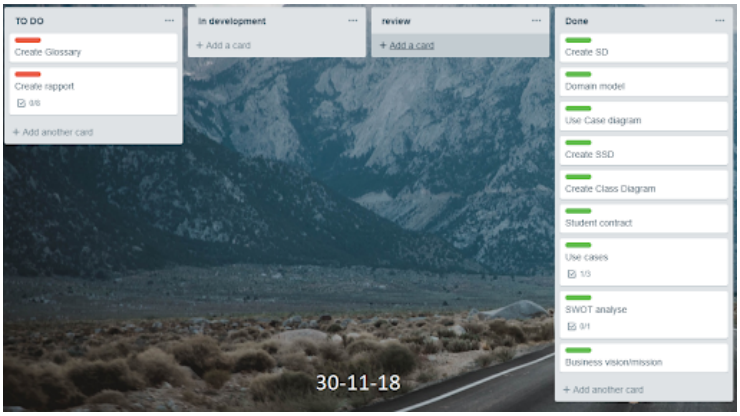

Figure 4.6: Kanban 5