

# Gamedb



Group: f11b405a  
BaIT4/INF4 Project May 2011  
Department of Computer Science  
Aalborg University

Morten Nielsen  
Nirojan Srikandarajah  
Umachanger Brinthaparan  
Lars Christian Vagner Lichon  
Stephan Vinther Smedegaard Rasmussen



**Title:** GameDB  
**Theme:** Modelling IT Systems  
**Project period:** 01/02/2011 - 27/05/2011

### Synopsis

This report contains both modeling of an IT system and software engineering. The product is constructed using an agile approach. The report is structured as an ongoing development, ending with a final prototype. The prototype is created using theory obtained from the database course and previous semesters design principles. The system is tested and concluded on with the problem statement in mind where the result is evaluated.

### Participants:

Lars Christian Vagner Lichon  
Morten Nielsen  
Nirojan Srikandarajah  
Stephan Vinther Smedegaard Rasmussen  
Umachanger Brinthaparan

**Group:** f11b405a  
**Supervisor:** Hua Lu  
Ralf Rantzau

**Page numbers without appendixes:** 87

**Number of appendixes:** 2

**Closed:** 24/05-2011

**Handed in:** 27/05-2011

Signed:

---

Lars Christian Vagner Lichon

---

Stephan Vinther Smedegaard Rasmussen

---

Morten Nielsen

---

Nirojan Srikandarajah

---

Umachanger Brinthaparan

The content of the report is freely accessible, but publication (with source) may only occur by an agreement with the authors.



## PREFACE

This report covers the construction of a database which purpose is to deliver reliable data to the end user. The report will contain the written documentation of the process using the relevant theoretical tools available to construct the database.

Several database genres were considered before the video game database designed for mobile devices was chosen. The reason is that the market for mobile optimized web-applications is relatively new.

The following two case-modules will form the theoretical basis upon which this report is built:

- DB – Databases
- SOE – Software Engineering

These abbreviations will be used throughout the rest of the report along with other case-specific terminology.

This project will be constructed using the agile development method.

We would like to thank Ralf Rantzau and Hua Lu for counselling throughout this semester as well as all teachers of the courses given.

The web-application can be accessed online at <http://gamedb.xtreemhost.com>, preferably using a smartphone. An account has been registered for test purposes and can be accessed with the following information:

Username: AAU\_Test

Password: 123

For further guidance on accessing the web-application, see appendix 1

## READING GUIDANCE

This report is to be read from top to bottom. It will start with some introduction to the topic and some preliminary analysis in order to establish the needs and demands for the system. Hereafter the construction phase will be described and coding examples will be given. A chapter covering the development process will describe the iterations. The testing phase will then follow, and documentation for the testing will be written in this section. Conclusively a reflection, discussion and conclusion will end the report.

The whole report, and the construction of the database, will be constructed by using the techniques obtained from the case module of SOE. The log of this method is included in the report.

Throughout the report the end users will be described as “he” or “him” courtesy of the reader, but not because the end users couldn’t be female. This is the chosen writing technique in this report.

The term “The group” will be used throughout the report and consist of the same people who have written this report and whose names are on the front page. Also the term “system” will be used for the product.

A “rating” will be used as a term that covers a text review and a numeric value assignment which indicates the user’s view of the quality of the game in question.

The source reference will be done by the Harvard method.

## INDEX

|            |   |           |
|------------|---|-----------|
| <b>1</b>   | <b>INTRODUCTION</b>                       | <b>10</b> |
| <b>1.1</b> | <b>FOCAL POINTS IN THIS PROJECT</b>       | <b>11</b> |
| <b>1.2</b> | <b>PROBLEM STATEMENT</b>                  | <b>11</b> |
| <b>1.3</b> | <b>DELINEATION</b>                        | <b>11</b> |
| <b>1.4</b> | <b>REPORT STRUCTURE</b>                   | <b>12</b> |
| 1.4.1      | DEVELOPMENT METHOD                        | 12        |
| 1.4.2      | ANALYSIS PHASE                            | 12        |
| 1.4.3      | DESIGN PHASE                              | 12        |
| 1.4.4      | IMPLEMENTATION PHASE                      | 13        |
| 1.4.5      | DEVELOPMENT PROCESS                       | 13        |
| 1.4.6      | TEST PHASE                                | 13        |
| 1.4.7      | CLOSURE                                   | 13        |
| <b>2</b>   | <b>DEVELOPMENT METHOD</b>                 | <b>14</b> |
| <b>2.1</b> | <b>SCRUM</b>                              | <b>14</b> |
| 2.1.1      | USING SCRUM                               | 16        |
| <b>3</b>   | <b>ANALYSIS</b>                           | <b>19</b> |
| <b>3.1</b> | <b>SYSTEM DEFINITION AND REQUIREMENTS</b> | <b>19</b> |
| <b>3.2</b> | <b>RISK ANALYSIS</b>                      | <b>20</b> |
| 3.2.1      | PRE-CONSTRUCTING RISKS                    | 20        |
| 3.2.2      | CONSTRUCTING RISKS                        | 20        |
| 3.2.3      | POST-CONSTRUCTING RISKS                   | 21        |
| <b>3.3</b> | <b>PRODUCT BACKLOG</b>                    | <b>22</b> |
| <b>3.4</b> | <b>ACTORS</b>                             | <b>23</b> |
| <b>3.5</b> | <b>USE CASES</b>                          | <b>24</b> |
| <b>3.6</b> | <b>STATE DIAGRAMS</b>                     | <b>25</b> |
| <b>3.7</b> | <b>ER-DIAGRAM</b>                         | <b>29</b> |
| <b>4</b>   | <b>DESIGN</b>                             | <b>31</b> |
| <b>4.1</b> | <b>PRIORITIZING SCHEMA</b>                | <b>31</b> |
| <b>4.2</b> | <b>DATABASE STRUCTURE</b>                 | <b>33</b> |
| 4.2.1      | GAME                                      | 34        |
| 4.2.2      | GENRE                                     | 36        |
| 4.2.3      | USER                                      | 36        |

|            |                               |           |
|------------|-------------------------------|-----------|
| 4.2.4      | RATING                        | 37        |
| 4.2.5      | COMPANY                       | 38        |
| 4.2.6      | RELATIONS WITHIN THE DATABASE | 39        |
| <b>4.3</b> | <b>INTERACTION</b>            | <b>40</b> |
| <b>4.4</b> | <b>WINDOW DIAGRAMS</b>        | <b>41</b> |
| 4.4.1      | GENERAL DESIGN                | 41        |
| <b>4.5</b> | <b>NAVIGATION DIAGRAM</b>     | <b>47</b> |
| 4.5.1      | NEWS                          | 47        |
| 4.5.2      | SEARCH                        | 47        |
| 4.5.3      | GAMES                         | 48        |
| 4.5.4      | LOGIN                         | 48        |
| 4.5.5      | PROFILE                       | 48        |
| 4.5.6      | ERROR MESSAGE:                | 48        |
| <b>4.6</b> | <b>GESTALT THEORY</b>         | <b>50</b> |
| 4.6.1      | LAW OF PROXIMITY              | 50        |
| 4.6.2      | LAW OF SIMILARITY             | 50        |
| 4.6.3      | LAW OF CLOSURE                | 51        |
| 4.6.4      | LAW OF CONTINUITY             | 51        |
| 4.6.5      | EXAMPLES                      | 52        |
| <b>4.7</b> | <b>FOUR DESIGN PRINCIPLES</b> | <b>53</b> |
| 4.7.1      | AFFORDANCE                    | 53        |
| 4.7.2      | MAPPING                       | 54        |
| 4.7.3      | CONSISTENCY                   | 54        |
| 4.7.4      | FEEDBACK                      | 55        |
| <b>5</b>   | <b>IMPLEMENTATION</b>         | <b>56</b> |
| <b>6</b>   | <b>DEVELOPMENT PROCESS</b>    | <b>60</b> |
| <b>6.1</b> | <b>SPRINTS</b>                | <b>60</b> |
| 6.1.1      | SPRINT 1                      | 61        |
| 6.1.2      | SPRINT 2                      | 62        |
| 6.1.3      | SPRINT 4                      | 68        |
| <b>7</b>   | <b>TEST</b>                   | <b>71</b> |
| <b>7.1</b> | <b>HEURISTIC EVALUATION</b>   | <b>71</b> |
| 7.1.1      | THE HEURISTIC EVALUATION      | 74        |
| 7.1.2      | RESULTS                       | 75        |
| 7.1.3      | FURTHER DEVELOPMENT           | 76        |
| <b>7.2</b> | <b>BLACK BOX TEST</b>         | <b>76</b> |
| 7.2.1      | EXAMPLES OF AN ERROR          | 77        |



|            |  |           |
|------------|--|-----------|
| 7.2.2      | SEARCH FUNCTION  | 77        |
| 7.2.3      | LOGIN DIFFICULTIES   | 77        |
| <b>8</b>   | <b>REFLECTION</b>  | <b>78</b> |
| <b>8.1</b> | <b>APPROACH</b>  | <b>78</b> |
| <b>8.2</b> | <b>ANOTHER AGILE APPROACH</b>  | <b>79</b> |
| <b>8.3</b> | <b>FUNCTIONALITY</b>   | <b>79</b> |
| <b>8.4</b> | <b>CONSTRUCTION</b>  | <b>79</b> |
| <b>8.5</b> | <b>TESTING</b>   | <b>79</b> |
| <b>8.6</b> | <b>USE OF USER DATA</b>  | <b>80</b> |
| <b>8.7</b> | <b>USER INNOVATION</b>   | <b>80</b> |
| <b>9</b>   | <b>DISCUSSION</b>  | <b>81</b> |
| <b>9.1</b> | <b>USING THE AGILE METHOD</b>  | <b>81</b> |
| <b>9.2</b> | <b>SWOT ANALYSES</b>   | <b>82</b> |
| <b>9.3</b> | <b>CONSEQUENCES FOR FUTURE DEVELOPMENT PRACTICE AND ENGINEERING PRACTICES TO BE IMPROVED</b> | <b>83</b> |
| <b>10</b>  | <b>CONCLUSION</b>  | <b>84</b> |
| <b>11</b>  | <b>BIBLIOGRAPHY</b>  | <b>86</b> |
| <b>12</b>  | <b>APPENDIX INDEX</b>  | <b>87</b> |

## 1 INTRODUCTION

The market share for video games is on a steady rise and is expected to reach yearly revenue at around 70 billion US dollars in 2015, making it a major global market (Takahashi, 2010). This makes it an attractive market for developers since it also offers a vast series of niche markets. The niche markets can consist of such genres as Indie or self-produced flash games. There are many different types of platforms on which the games can be played on, such as consoles (PlayStation, Xbox, Wii, etc.), PC's, mobile devices and tablets (iPhone, Android devices, iPad) or portable devices primarily designed for games (PlayStation Portable, Nintendo DS).

A popular trend in the industry is to make a spin-off of an already popular society trend. This can be seen in popular games such as the Lego Harry Potter game series or Facebook's Farmville game. This trend is also seen in the market for mobile phones where some games have reached an enormous popularity. The game Angry Birds made by the Finish game developers Rovio, has sold over 75 million copies (Roelsgaard, 2011). There are already several planned spin-offs such as board games, a TV-series, a Facebook application, a movie and further development to suit the game for different platforms such as the Wii and Xbox 360.

This shows that a small game can reach enormous success and therefore helps raise the markets popularity since there is a great potential profit, although the competition is fierce and the entry barriers can be high.

This complicated, yet attractive market has made it hard for the end-user to choose which games suit him best. The video games have many genres and sub-genres which affects the choices of the end-user. The user often has to consider platform, single- or multiplayer, price, type of game and so on before choosing whether or not to purchase a game. A popular method of sorting many entities from another is by popularity, which is particularly handy when it comes by internet media, where many users can give ratings, which helps the user see what other people think of the entity.

This is a helpful tool for the end-user to learn about what fellow peers think of a given game and together with a helpful synopsis of the game the user has a better chance of choosing a well-suited game.

This project's goal will be to create a video game database on which the user easily can read some short information about a given game or see the newest games. The database will be accessed by the user through a web-application optimized for mobile devices. By having the opportunity to look at game information and ratings on the phone, tablet etc. right in the store where the games are, the user is more likely to come home with the right game. This is because the user can read more detailed information on the web-application instead of only have the short information on the back of the game cover. The opportunity to read and write ratings of the game makes it even more interesting to make a bit of researching when buying a game as the "human" content on the site will make it more vibrant with the constant rating activity.

All in all this product will be a help to the user who does not want to spend hours in front of the computer just to decide which game to go and buy, but instead want simple and quick accessible information about the games available.

## 1.1 FOCAL POINTS IN THIS PROJECT

This report will focus mainly on the development of a database and how this works. This will result in diagrams and descriptions showing the internal structure of the database together with the theoretical basis on which it is built upon.

The other main focal point will be the development method used to create the database. This will include the description of the process together with the theoretical thesis compared with the database and project.

Therefore, the object of the report is to describe the process of which a database can be constructed and functions hereof.

## 1.2 PROBLEM STATEMENT

The following problem statement will be answered throughout this report and summarized at the end. The statement will help serve as a guideline in order to keep the report on track in regards of the chosen subject.

The statement also involves the theoretical thesis on which the report will be formed upon: The teachings of Software Engineering (SOE) and Databases (DB).

The statement is as follows:

*“How can we develop a web based database application, and which processes and challenges do we face when using an agile development method?”*

## 1.3 DELINEATION

This report will cover the following only partially or not at all, due to the reasons submitted.

- User analysis will not be described in depth, because in this development phase the users will primarily be the group itself.
- The database will not consist of every possible genre or platform since there are too many and there are always new added. This project will instead focus on some of the most popular genres and platforms.
- The user-interface (UI) will not be described in depth since it is not a major focal point in this project.
- The project will only include the traditional development methods to compare with the chosen agile approach as written in the preface.
- The only code examples that will be described in this report will be those relevant to the database.

## 1.4 REPORT STRUCTURE

The following will be an overview of the whole report. The relevant method theories will be described in the individual chapters.

- Development method
- Preliminary analysis
- Design phase
- Implementation phase
- Development process
- Testing
- Closure

### 1.4.1 DEVELOPMENT METHOD

This chapter will describe the chosen development method on which the database will be constructed. It will contain both the theoretical basis and adaptations for use on this specific project.

The theory will be based on the teachings from the course of SOE.

### 1.4.2 ANALYSIS PHASE

The analyses phase will determine the general structure of the database. It will help the developers structuring their work efforts and ensures that every developer has a clear overview of the database in terms of functionality and structure.

This chapter will cover some preliminary introduction to the idea behind the database.

A risk analysis will be made to ensure that the group knows which risk factors needs to focused on before, during and after the construction of the database.

The structure of the database will be formed by the product backlog which will help form an overview of which functions are necessary to be implemented in the database. These functions will also be described in detail.

It will also contain the important Entity-Relationship model (ER-model) which is a conceptual model that illustrates the database structure.

Furthermore there will be a brief user-analysis in this chapter, although not in explicit detail since this report does not focus on user-experience or innovation, using use-cases and actor description.

### 1.4.3 DESIGN PHASE

Here the basic form of the UI will take form. The inspiration of this design will come from what is already available (and commonly used) on the chosen platform combined with the experience gathered from the previous semester. This includes the use of relevant design laws such as the Gestalt laws, the law of

closure, mapping, feedback etc. These laws will only be briefly substantiated with examples because this neither is major focal point in the report. Furthermore the interaction forms will be described.

Besides the graphical aspects of the system, the design of the database structure will be described as well. This will include a diagram of the database along with descriptions of tables, attributes etc.

#### 1.4.4 IMPLEMENTATION PHASE

Based on the analysis and design phases, the database itself can be constructed. The chosen tools for this will be PHP, MySQL and HTML. Furthermore jQuery Mobile is used to design the web-application, as this is already optimized for mobile device use. The database will be small in size and complexity in order to support the chosen platform and in order to secure reliable data for the user.

The code which is attached as appendix 2 (website files) and appendix 3 and 4 (SQL), have comments describing the code itself in it. The comments will have the distinction “//” and “<!-- -->” before it. Some interesting code examples (which will have relevance to the database) will be included in the report and explained.

#### 1.4.5 DEVELOPMENT PROCESS

This chapter will include sprint-documentation and sprint-backlog used to develop this project. The chapter will also cover the iterations used in detail, describing each prioritized sprint used to make this project.

This will be structured by the bi-weekly ordering by which the group has incorporated during the process.

#### 1.4.6 TEST PHASE

This chapter will cover testing of the database and its user interface. This chapter will contain a heuristic test which will be performed and documented. Following the test, the results will be described as well as further development ideas. Furthermore a short black box test will be described and used.

The flaws found in the test will be described but not implemented in the system from this point.

#### 1.4.7 CLOSURE

The final chapters in this report will consist of the answer to the problem statement and further reflection and discussion based on the experiences gathered in the development process. This section will also contain the source and appendix list.

## 2 DEVELOPMENT METHOD

The development approach for this project will be agile since the requirements for the final product is unknown. The agile approach will be beneficial since it involves small steps in the development process which are helpful given the many uncertainties. Because of these uncertainties, the goals and focal points of the development process can and will change, which responds well to the agile approach.

Since this project will have a lot of focus on trial-and-error programming, and not a lot of research, the agile methods are appropriate since it gives the developers more individual freedom instead of a fixed framework where the developers are inhibited by following a sequential model.

### 2.1 SCRUM

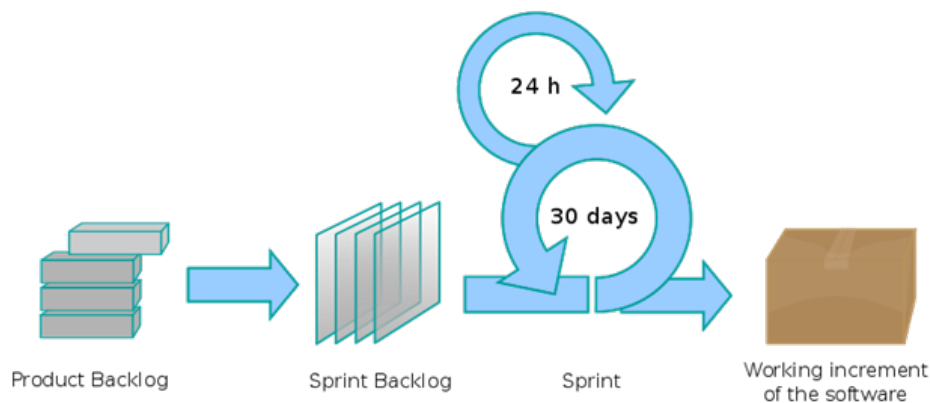


Figure 1

The chosen agile development method is the SCRUM mode (Scrum (development)). This will describe the general theory about SCRUM. Figure 1 is an illustration of the SCRUM model.

SCRUM is an iterative and incremental project management method. It is seen as an agile software development method. The definition of “agile” is that it is flexible to error handling and changes raised during the development process as it relays more on experience than analysis.

SCRUM contains sets of practices and predefined roles. The main roles are a “SCRUM Master”, “Product Owner” and the “Team”.

The “SCRUM Master” maintains the processes and is also called the product manager. This role is accountable for removing impediments to the ability of the team to deliver in time. It is important to notice that the SCRUM master is not the team leader but acts as a buffer between customer and the team. If anything changes, the SCRUM master has to make sure to solve it with both of the parties. This role also has to make sure to keep the team focused.

The “Product Owner” is seen as the stakeholder. A product owner can be a member of the developing team but it is recommended that this role does not be combined with the SCRUM master. The product owner usually writes customer-critic items which could be the user stories.

The “Team” is a cross-functional group of about seven people who do the actual analysis, design, implementing, testing, etc. It is recommended that the team is self-organizing and self-led and does the work with a team management.

In this method there is a term called “sprint” which means a period of typically two to four weeks where the team creates a potentially shippable product increment. An example of this could be a working search function. The set of features that gets into a sprint comes from the “backlog” which is a prioritized set of requirements of work to be done. These features are determined by the sprint planning meetings. On these meetings the product owner informs the team of the items in the product backlog that should be completed. The team then determines how much is possible to be done and how it should be handled. This ends in the sprint backlog. When a sprint is started no one is allowed to change the backlog. A sprint must end on time and if some features can’t be done on time, it ends in the product backlog for another sprint meeting.

SCRUM makes it possible to make self-organizing teams by encouraging colocation of all team members. It will also give the teams a chance to discuss and communicate through the project.

SCRUM has a key principle that a customer is allowed to change his mind after the team has started on a project. When using an agile method you are aware of the unpredicted challenges. The focus is to get the team to deliver quickly and respond to emerging requirements.

The SCRUM method does also contain two secondary roles which are the stakeholders and the managers. The stakeholder will typically be the customer that enables the project and make sure that the product will end up as what have been agreed. They are only directly involved when a process is during the sprint reviews. The manager role is to set up the environment for product development.

There are different kinds of meetings which are a daily meetings, sprint planning, sprint review and sprint retrospective.

The daily meeting is done during the sprint to get a status of the process. The meeting will start precisely on time and does normally contain the core roles. It will last for about 15 minutes and should held at the same location every time. The questions that each team member answers are, “What have you done since yesterday?”, “What are you planning to do today?” and “Do you have any problems that would prevent you from accomplishing your goal?” The SCRUM master has to resolve these problems.

The sprint planning meeting is done when a sprint cycle is about to start (typically every 7-30 days). In these meetings the work that has to be done will be selected and a sprint backlog will be prepared. In this sprint backlog it will be determined how long it will take. You also determine how much of the work that is likely to be done during the current sprint. The meeting has an eight hour time limit. The first four hours priorities the product backlog and the plan for the sprint, resulting in a sprint backlog is made in the last four.

At the end of a sprint cycle there will be a sprint review meeting. In this meeting the production team reviews the work that has/hasn't been completed. The completed work is presented to the stakeholders as a demo. These meetings have a four hour limit.

The last meeting is the sprint retrospective meeting where all team members reflect on the past sprint and makes continuous process improvements. There are two main questions that have to be asked which are, "What went well doing the sprint?" and "What could be improved in the next sprint?" These meetings have a four hour limit.

The SCRUM method contains three artefacts which are the product backlog, sprint backlog and a burn down chart. The product backlog is a high-level list that is maintained throughout the entire project. It contains the prioritizing and a broad description of all the features. This backlog has all the rough estimates of both business value and development effort which helps the product owner to make the timeline and to limit the product features and how it should be done.

The sprint backlog is a list of work that must be completed before the next sprint cycle. Features are then broken down into tasks which should normally last for 4 to 16 hours of work. With a high level of details all of the team members understand exactly what to do. Tasks on the sprint backlog will never be assigned but rather signed up for by the team members. This helps the team to use their best competences for the different task which will give a high productivity. This backlog is the property of the team and a task board will usually be used as a supplement. The task board contains the state of the different tasks like "to do", "in progress" and "done".

The sprint burn down chart is a publicly displayed chart that shows remaining work in the sprint backlog. This is updated every day and gives a simple overview of the progress.

### 2.1.1 USING SCRUM

The following will describe how the group will make use of the SCRUM model. The purpose is to ensure that every developer has an overview of the current state of the project and which challenges has to be overcome in the next time period.

In this project the SCRUM model will be used to:

- Arrange meetings in the group with an task specific agenda
- Make a complete list of the necessary tasks regarding the project.
- Make sure that everyone in the group knows the current state of the project
- A fair delegation of the different work tasks, both writing documentation and coding.
- Make deadlines for every task in order to ensure on-time delivery.

In order to maximize the efficiency of the SCRUM model, it will be adapted for this project, which is described below.



#### 2.1.1.1 MEETINGS

The group has meetings every second Monday morning at 9.00 am where the tasks of the week, according to the sprint backlog, are discussed. The individual tasks are assigned to the group members and a deadline for each one is estimated. This meeting also sums up the experience gathered from the previous weeks, which forms a realistic time estimation of the upcoming tasks.

There will also be short meetings daily, where the individual tasks statuses are discussed. If there are any challenges associated to these, resources may be reorganized.

Every second Friday there will be a meeting regarding task statuses and weekend planning regarding these. This may help if some tasks demanded higher resources than anticipated, and reorganizing resources is the only solution in order to ensure on time delivery of the sprint.

#### 2.1.1.2 ROLES

The associated roles with this project will mainly be the “SCRUM master” and the “team”. The SCRUM master will have the responsibility that the tasks are completed on time and that every group member knows their task. This role will be rotational, meaning a new person will be assigned this role every week, because this ensures that everyone will experience how it is to be a SCRUM master. The SCRUM master will also have one or more tasks himself within the production.

The team will consist of the rest of the group, and their main function is to complete the tasks given, e.g. writing documentation or coding a specific function.

#### 2.1.1.3 PRODUCT BACKLOG

Here follows a complete list of all of all tasks associated with the project in regards of coding.

The analysis phase is set before the product backlog is created in order to ensure the basic formalities are present in order to create a functional overview. The functions will be prioritized in this phase in order to secure which functions needs to be developed first.

The individual functions made in this chapter will form the base for the sprint backlog.

#### 2.1.1.4 SPRINTS

The sprints will last two weeks each and will be documented in prioritized order. The sprints will be assigned to the individual group members and a deadline for its completion established.

The sprint backlog takes the individual functions and converts them into tasks, which are to be assigned to each group member during the project period. The sprints will have a fixed time limit assigned and will be of variable length due to the various complexities of each task. Because it is an iterative process, more tasks will appear in the sprint backlog that didn't originate from the product backlog.

After the sprint is finished and the task is completed, the tasks will be marked as completed. Some tasks may be assigned to two or more of the group members if deemed necessary.

The testing will be done throughout the whole development process using iterative trial-and-error programming and therefore it won't appear as a task sprint.

#### 2.1.1.5 DOCUMENTATION

The use of the SCRUM model will be documented in this report. Every second Monday during the project period, the upcoming two-week plan for sprints will be planned. Each sprint will be documented when completed.

The documentation will be inserted every two weeks starting from week 11 of 2011, because the previous time period was used to plan and develop the concept of the database.

This will include:

- Status of pending tasks.
- Current status.
- Assignment of individual tasks for the upcoming two-week sprint.
- Results of the sprint.

The daily meetings will not be documented as they are seen as minor meetings which purpose only is to come up-to-date with the current status.

## 3 ANALYSIS

This chapter will cover the requirements and description of the product. It will contain the functionality in detail that the product needs to cover the basic needs. This will appear in a product backlog where the functionalities. Every one of these functions will be given a state diagram that shows the steps that the users have to go through to solve a task and the system statements.

The user will be described through use cases and actor tables which help the developers to keep the users in mind while the system is designed and produced.

The chapter will also contain a risk analysis that will be divided into three sections. This will help the group to understand which risks that could be in the development and how to solve these.

Before the database can be created and contain any information, an ER-diagram has to be made. This helps the developers knowing what the tables should contain and which relates to each other. The ER-diagram will be used as a conceptual model.

### 3.1 SYSTEM DEFINITION AND REQUIREMENTS

This system will consist of a database containing data of video games. The system's purpose is to give the user a quick overview of relevant game data such as the title, release date, developer, rating, etc.

The games will be suited for various platforms, but the database itself will be developed for a mobile device as a website optimized for mobile devices. The user can access the site from his mobile phone using the internet.

The system will be developed using MySQL, HTML and PHP.

The database will be developed for the mobile market, since there are only few applications like this on a rising market, and there are a lot of forums and sites on the internet like this system. The advantage of this system is the flexibility the platform gives the user. E. g. a user accesses the webpage from his mobile device when in a store that sells video games. He can then read ratings of the game and see ratings before deciding whether or not to purchase the game.

The users are all people with a mobile device and an interest for video games. The user interface will be suited for mobile phones and other handheld devices such as tablets. The age of the users are primarily 10-40 years old and will typically be males (ESA, 2010). It is important to notice that this does not mean that females or others can't use the system, but they are not the primary target users.

Using a login function, the user can interact with the system, give ratings, making the database interactive. This function is of major importance since interaction has become one of the major buzzwords in recent internet applications.

The general purpose of the system is to give the user a quick basic overview of the most relevant facts and comments regarding video games such as genre, platform, synopsis and playability in regards of single- and multiplayer etc. Therefore there will be no in-debt analysis of the games, but rather minor ratings, because

of the chosen platform which is not well-suited for reading long texts. Simplicity is a keyword in the database so the user can make a quick search for a game and read the simplest information regarding the game.

## 3.2 RISK ANALYSIS

This chapter will cover the various risks, surrounding the development of the database and the various risks succeeding its production in order to maintain it. Please note that there are many more risks than what is stated in this section, but the risks described below are the most relevant for this project.

The risk analysis will be divided into three sections:

- Pre-construction risks.
- Construction risks.
- Post-construction risks.

### 3.2.1 PRE-CONSTRUCTING RISKS

Before the actual construction of the system, an overview of the database must be established in order to ensure a reliable system.

A system overview must be constructed so the database can be built upon it. Basic items such as which objects should be set as primary keys and which attributes each entity should have are essential for an effective system.

Therefore it is a considerable risk if these initial steps are not well executed. To ensure this will not fail, several models will be used to construct a visual overview of the database before construction. These models are the relational model, the ER-model, navigation diagram etc.

There are several unknown factors regarding the development of this database such as the programming abilities of the group members, the complexity of constructing the database, and technical challenges during the process. These are all risks which can't be fully determined before the development process is initiated. Therefore the agile development method fits well, because it adapts to the situation and makes use of the trial-and-error sort of development.

### 3.2.2 CONSTRUCTING RISKS

There are several risks during the construction of the database which must be considered in order to create a reliable and stable database.

During construction of the database there are several programming risks that must be accounted for in order for the database to run as intended. The programmers must secure that the users can't type in a text string where a numeric value is needed. This risk is overcome by choosing the proper data type, such as int, string or varchar. When general errors are not corrected the system will not run as intended.

The programmers also need to be aware of default values and null values. According to the ER-diagram, the game table consists of such attributes as publisher and release date. But if the admin has to type in a game where the release date is unknown, there has to be a default value of *null* in order to secure the correct data output.

An important development tool is version controlling. There will always be risks of systems failures when the code is altered or some code is added to the existing. Therefore the programmers need to have version control in order to perform a rollback to when the last time the system was working correctly.

The mathematical functions in the system (e.g. the rating system or the top 25 listings) has to be finite in functions yet flexible when altered as more games will be added to the database and users give more ratings. The functions have to give a precise result each time in order not to make the system crash as more entries are added.

### 3.2.3 POST-CONSTRUCTING RISKS

After the construction there are still considerable risks that need to be precaution against. There are some external factors as malware infections (spyware, viral infections etc.) that the system needs to be protected from.

Power failure at the webhost could be catastrophic because it can corrupt the data. In order to overcome this risk, a backup system needs to be established. This backup system could take backups of the database daily, bi-daily or weekly, but not rarer than that. The choice of backup delay depends on the cost versus the effect of loading a database a couple of days old.

Another external risk is change in technology, e.g. new browsers or operating systems. These can be hard to predict, but therefore it is necessary to choose a flexible and adjustable webhost.

Since the database contains user-login features, there has to be a backup system if the user forgets his login information. The solution could be an automatic generated e-mail with the login information sent to the user-specified email address which is noted during the registration process.

### 3.3 PRODUCT BACKLOG

The following product backlog will contain the main functions of the product together with a short description. The description creates a general overview of the systems need-to-have functions and will form the basis upon which the sprints can be executed by.

The purpose is to create a general overview of the system and help the design phase.

| Features (program)      | Description  | Prioritizing |
|-------------------------|--|--------------|
| <b>Database</b>         | Making the entire database structure in MySQL.   | 1            |
| <b>Design</b>           | The overall design of the website  | 2            |
| <b>Log in and out</b>   | The user should have the opportunity to log in and out. This gives the users the opportunity to write reviews and rate games.  | 3            |
| <b>Registration</b>     | If the user does not have any account, he should be able to register as a new user via a registration form. This form will contain a username, name, birthday, email and a password (which has to be encrypted). It is possible to change this information later in the user profile.<br>The reason for this is that the system should be taken seriously when the user has to give some personal information. Without giving any personal information, the ratings given could be taken less serious. | 4            |
| <b>User information</b> | User information is the person relevant data such as ratings, email address, birthday, etc. The user should be able to view and edit his personal information and ratings.   | 5            |
| <b>Game information</b> | This is the display of each game that contains title, genre, release date, developer, publisher, summary, cover, rating, reviews etc. The game information can only be edited by the administrator, the user is only able to view game information.  | 6            |
| <b>Front page</b>       | This page shows the latest entries in the database as a list, and it is the first page user should see when he runs the system.  | 7            |
| <b>Rating</b>           | The rating function is only possible if the user is logged in to the system. It is possible to read ratings without being logged in to the system. When the user rates a game, he can chose from one to five stars. The user cannot give a game a rating without typing a short review. This is because other users should have the option to know why the game has been given the specific rating.<br>It is possible to edit or delete a rating in the user profile.                                  | 8            |
| <b>Search</b>           | The user should be able to search with typing and selecting some attributes like platforms and genres. If the system comes with a result it will show a list of games and a short description and picture. If the system doesn't find any result the user should be informed that no result was found.<br>The user should also be able to search by different lists. This means by genre, platform, highest rated etc.   | 9            |

Table 1

### 3.4 ACTORS

The following describes the actors within the system and what their role is. The reason to do this is to get a better understanding of how the final users will use the system. The areas that will be covered in the tables are purpose, general characteristics and finally some examples of how they will use the system. These data will help to keep the users in mind when designing the system. Furthermore this will help to create the use cases that show what aspects of the system they have access to.

| Administrator   |
|---|
| <p><b>Purpose</b><br/>                     This person's purpose is to manage the website and software running behind the scenes, such as managing the MySQL database(s). The administrator also has the responsibility of adding new information to the site.</p>  |
| <p><b>Characteristics</b><br/>                     The administrator should be an experienced IT user and should be familiar with MySQL, PHP, CSS and HTML and should at least possess the basic knowledge on how to use these, in some cases more than the basics will be necessary.</p>   |
| <p><b>Examples</b></p> <ul style="list-style-type: none"> <li>A. The Administrator wants to add a new game to the database with the information that's tied to it. The Administrator goes to the MySQL login page, logs in successfully and navigates to where information is added to the database. The administrator adds the wanted information and logs out when done. The information will now be shown on the website.</li> <li>B. Another Administrator is new to the system and how it works; he needs a lot of help or seeks help online in order to get the work done.</li> </ul> |

Table 2

| User  |
|---|
| <p><b>Purpose</b><br/>                     The user's purpose is basically to use the system to get information about different video games. The user can look up games and he can search for games and read more information for a single game at a time. The user will get more options if he makes a profile for the homepage.</p>   |
| <p><b>Characteristics</b><br/>                     The user must have some knowledge about video games, and smartphones while interested in getting more information about different video games.</p>   |
| <p><b>Examples</b></p> <ul style="list-style-type: none"> <li>A. The user wants quick information about a game. He goes to the website and search for the game, and without any big hazels he finds the game he is looking for.</li> <li>B. A user has just played a game which in his opinion was really bad, so he wants to warn other users from buying it and potentially wasting their money. The user enters the websites, navigates to the game he is looking for and writes a rating for the game.</li> </ul> |

Table 3

### 3.5 USE CASES

The picture below illustrates the use cases within the system. It shows which tasks the user is able to and which the administrator is able to. As seen on the illustration the user has the ability to search through games, write/edit/delete ratings, read game info, edit user information and register to the system.

The administrator can also search through games, read game information and delete data (if e.g. a user writes something inappropriate). This means that the administrator has the ability to delete data from all the users while the individual user only has the right to edit his data. Besides the tasks that the user and the administrator has in common, the administrator can also add and delete games in the database.

The administrator has access to all of the tasks because he makes sure that the system works as it should. The reason why this is not illustrated is because the administrator shouldn't write/edit ratings because it is the user's job but this does not imply that the administrating person can't be a user. The administrator shouldn't be able to view user information either, unless absolutely necessary.

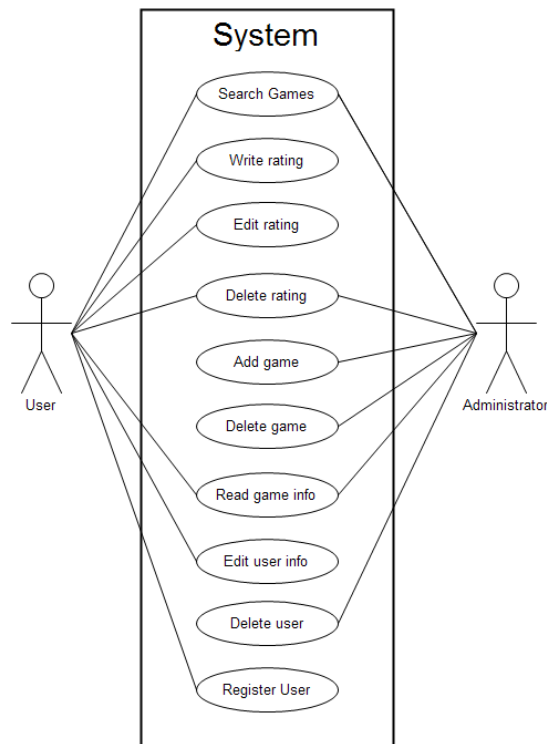


Figure 2



### 3.6 STATE DIAGRAMS

#### REGISTER NEW USER

When a user wants to register into the system, he has to choose the profile button in the menu. The log in page will then show where the user can choose to register or cancel. Afterwards the register form page appears and the information has to be entered (username, name, birthday, email etc.). The system checks if the necessary information has been filled or typed wrong and determines if the user has to return to the form page or be verified. If the system verifies the information, the user is redirected to the login page. The system will also check whether the submitted username or email already exists in the database – if so, the user will be sent back to the register form.

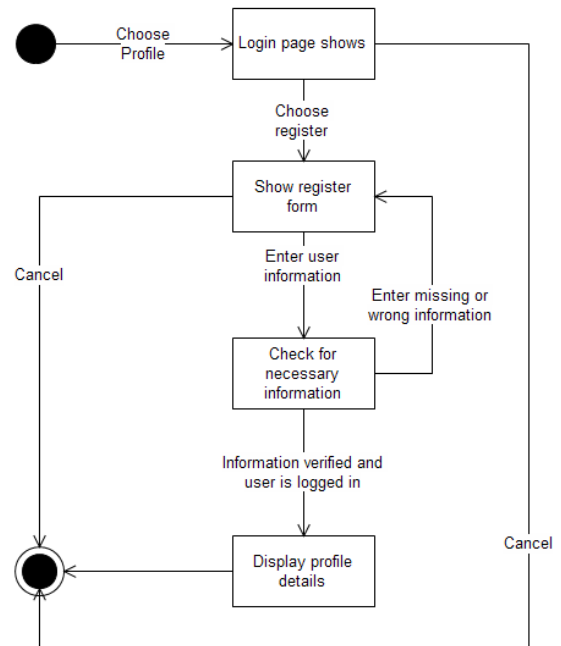


Figure 3

#### LOG IN AND OUT

When the user wants to log into the system, the “Profile” button has to be clicked and the login form shows. The necessary information then has to be entered and the system checks if the user exist or some information are missing or typed wrong. If there is something wrong, the user returns to the login form, otherwise the user will be logged in. The system can then be used until the user wants to log out where the profile page has to be clicked. On this page the user can log out with the log out button, and the user session ends. When logged out of the system, the user will be redirected to the news page.

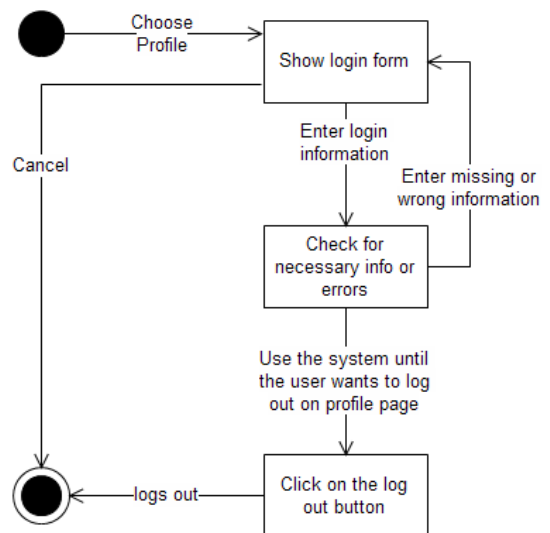


Figure 4

### ADD AND EDIT RATING

To add or edit a rating, the user has to be logged in to the system and choose a specific game. If the user is not logged in to the system he will be redirected to the login page. After the user has logged in to the system, it will show the game information page where the user has the opportunity to write or edit a game rating or cancel. When the rating form is shown he can choose to cancel the task or enter the information (title, review, rating etc.). The system checks for necessary information and if something is missing the user is sent back to the rating form. If the information is verified, the rating will be added to the game and the user will be redirected to where he came from.

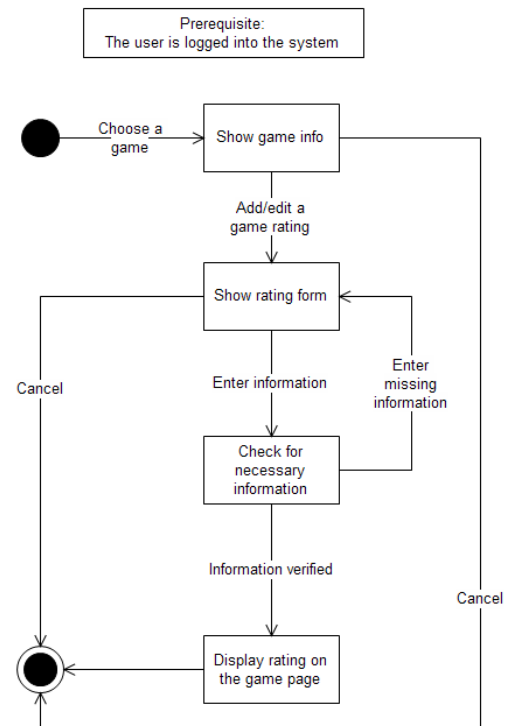


Figure 5

### EDIT AND DELETE RATINGS FROM PROFILE

To edit or delete a rating the user has to be logged into the system and choose the profile option in the menu. If the user is not logged in to the system he will be redirected to the login page. After the user has logged in to the system the system will show the profile page where the user has the opportunity to see the list of his ratings or cancel. On this list there will be two buttons at every rating so the user can choose to either delete or edit a rating.

If the user chooses to delete a rating, a new site will appear asking if the user is sure that he wants to delete the rating. If the user chooses “No”, the system returns to the rating list where the user can cancel the task and return to the profile or delete/edit another rating. If the user chooses “Yes”, the rating will be deleted, and the user will be redirected to the list of ratings.

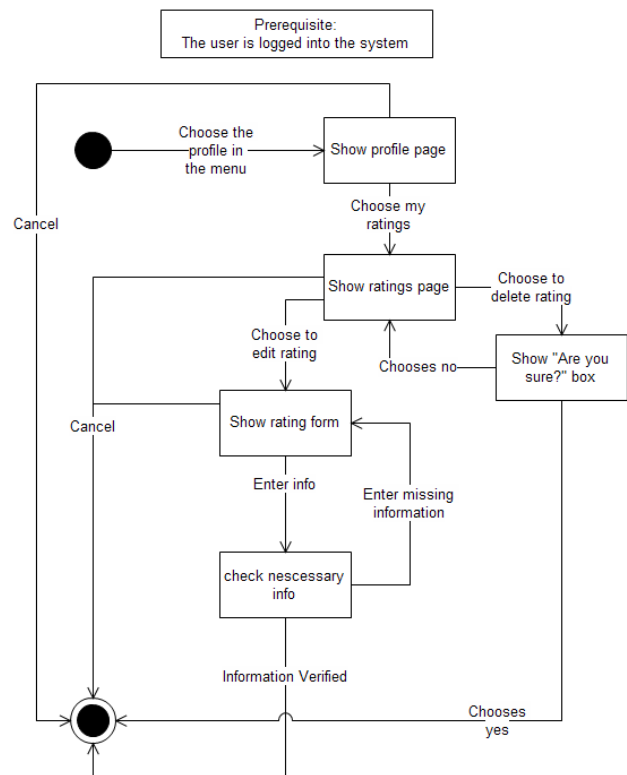


Figure 6

In the case where the user chooses to edit a rating, the system shows the rating form with all the information that has to be filled out like title, review, rating etc. The user can then choose to proceed entering the information or cancel the task. If the user chooses to enter the information and clicks save, the system checks if the necessary information has been filled. In the case where information is missing the user is sent back to the rating form to type the right information. The system then checks if the user has filled out the form again. If the information is verified, the system displays the rating on the specific game page. If the user has to edit some of the content, he can choose to edit again, otherwise the rating can be saved to the system.

## SETTINGS

The user has to be logged in to the system to change user setting which can be found on the profile page. If the user is not logged in to the system, he will be redirected to the login page. After the user has logged in to the system, the system will then show the profile page. The user can also choose to cancel on this page. When the settings option has been chosen, the system displays the form that has to be filled out. The user then has three options which are to cancel the task (and return to the profile page), choose to change the information on that page or go to another page to change the password.

If the user chooses to continue changing the information on the current page and enters the information, the system will check for necessary information (name, birthday, email etc.). The information can either be verified or else the user will be sent back to enter missing information. The profile page will then show, and the task is finished.

If the user chooses to change the password, a new site will appear with a password form that has to be filled. Like the rating, form the system checks for necessary information (previous password, new password and retype new password) and then determines if the user has to go back to type the right information or if the information is verified and shows the profile page.

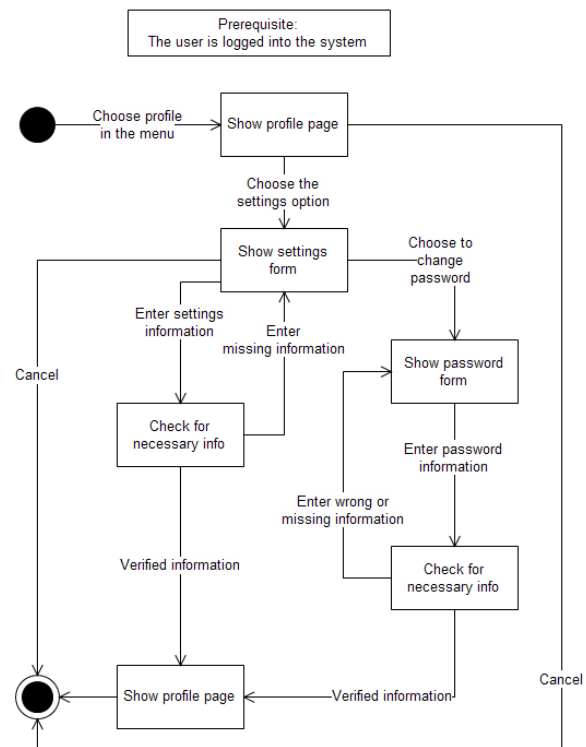


Figure 7

## SEARCH

The system has two ways of searching. The user can either choose to search manually by typing a keyword or choose to search by lists.

In the case where the user chooses to search by typing, the “Search” button has to be clicked and a search page will appear. The user can then type in a keyword he is looking for and use advanced search options if needed. The system will then search for results in the database. If the system does not find any results the user can go back to search again. If the system finds any results a list will show and the user can choose a specific game. The game side will show and the task has ended. Notice that the user can choose to cancel at any moment doing the task.

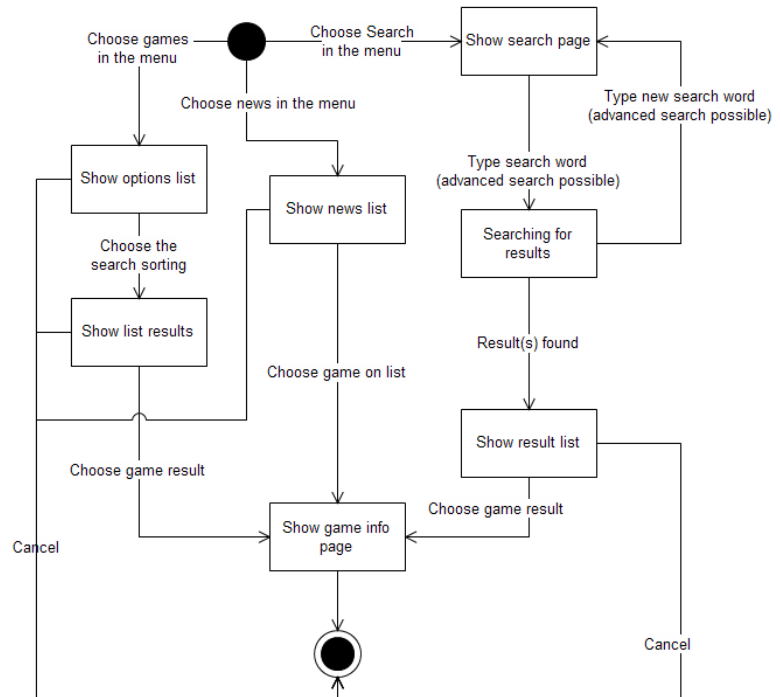


Figure 8

In the case where the user chooses to search by lists, he can either choose the “News” or the “Games” option in the menu. If the news page has been chosen the user sees a list of new games where he can choose to see a game or cancel. Afterwards a game page will show and the task has been finished. If the user chooses the game page, an option list will show where the user has to pick the one he is looking for. He can also choose to cancel. The options are genre, platform, top 25 and all games. When the user has chosen the way he wants to sort his search a list of results will show and he can choose to view a game on its specific page or cancel the operation.

### 3.7 ER-DIAGRAM

As part of the early analysis, the group made an Entity Relationship Diagram. The diagram is in this context was made as a conceptual model of the database and was made to help the group get a visual picture of the database and to get a better overview of it. This diagram was the basic for the start of the database, so there might be minor changes from this layout till the final layout. The diagram will now be described. If the reader wants further instructions on how to understand the following diagram, please proceed to appendix 5 (Korth & Sudarshan).

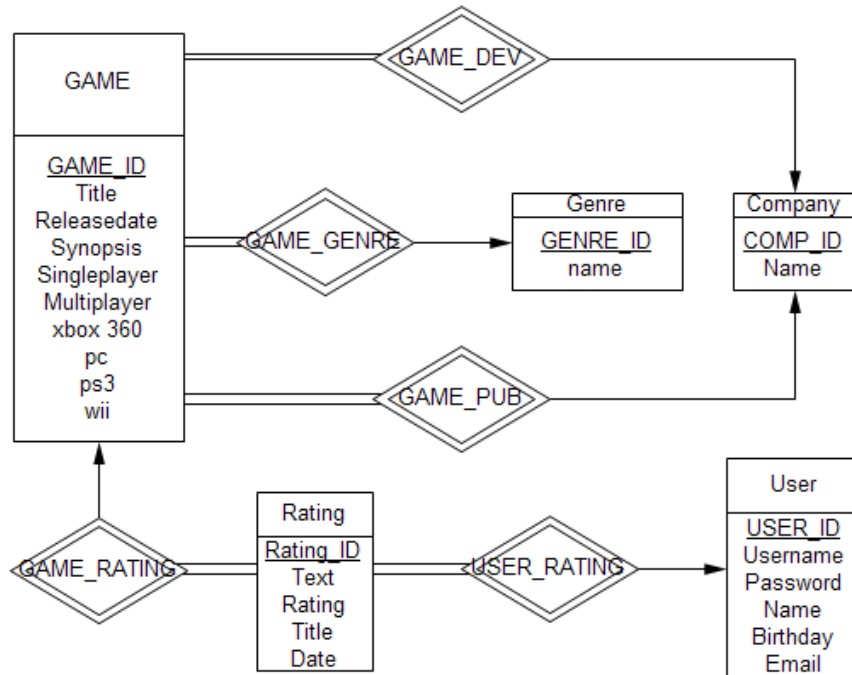


Figure 9

Figure 9 is the ER-Diagram of this database. There are a total of 5 tables in the database which are User, Rating, Genre, Company and Game. These tables are all connected to each other with relations. The relations are as following:

**Rating and User:** A Rating is associated with at most one User via USER\_RATING and a User is associated with several (including 0) Ratings via USER\_RATING. Every Rating must have a User.

This means that a user can write as many ratings as he wants to, and these ratings will only be tied to this user.

**Rating and Game:** A Rating is associated with at most one Game via GAME\_RATING and a Game is associated with several (including 0) Ratings via GAME\_RATING. Every Rating must have a Game.

This means that a game can have as many ratings as the system allows, but one rating can only be tied to one game and one user.

**Game and Genre:** A Game is associated with at most one Genre via GAME\_GENRE and a Genre is associated with several (including 0) Games via GAME\_GENRE. Every Game must have a Genre.

This means that a game can only be tied to one genre, but a genre can exist in more than one game.

**Game and Company:** A Game is associated with at most one Company via GAME\_PUB/GAME\_DEV and a Company is associated with several (including 0) Games via GAME\_PUB/GAME\_DEV. Every Game must have a Company via GAME\_PUB and GAME\_DEV.

This means that a game has one publisher and one developer, but a company can be tied to more than one game. The reason for this structure is that one company can be both developer and publisher for one game, so this was thought as the best solution in order to avoid duplicated information.

## 4 DESIGN

In this chapter, the design of the system will be sketched and described in detail using different design principles. The design will be formed by following the prioritizing schema, which describes the focus areas during the development process.

Based on the prioritizing schema and product backlog, window diagrams will be formed and thereby a navigation diagram will be created.

To ensure that the end-users can understand and use the UI, several design laws will be implanted and accounted for in the design, which will be at the closure point of this chapter.

Besides graphical design and interactions, the database structure will be described in depth such as tables, relations between tables and attributes.

### 4.1 PRIORITIZING SCHEMA

In the prioritizing schema you'll see the focus areas of this project. It shows how the product should be prioritized for the user which helps to the designing process and function implementation.

|                   |                          | Very Important | Important | Less important | Irrelevant |
|-------------------|--------------------------|----------------|-----------|----------------|------------|
| <b>Usability</b>  | Effectiveness            | X              |           |                |            |
|                   | Efficiency               | X              |           |                |            |
|                   | Safety                   |                |           | X              |            |
|                   | Utility                  |                | X         |                |            |
|                   | Learnability             |                |           | X              |            |
|                   | Memorability             |                |           | X              |            |
| <b>Experience</b> | Satisfying               |                | X         |                |            |
|                   | Enjoyable                |                | X         |                |            |
|                   | Fun                      |                |           | X              |            |
|                   | Entertaining             |                |           | X              |            |
|                   | Helpful                  | X              |           |                |            |
|                   | Motivating               |                |           | X              |            |
|                   | Aesthetically pleasing   |                | X         |                |            |
|                   | Supportive of creativity |                |           |                | X          |
|                   | Rewarding                |                | X         |                |            |
|                   | Emotionally fulfilling   |                |           | X              |            |

Table 4

The system is planned to be a tool that can help the user to learn more about different games. It is very important that that it is easy to use. The results in the database have to be valid for the user.

The schema shows that effectiveness and efficiency are keywords for this system. This means that it has to find the result fast and easy so it can help the user as quick as possible. When the user finds what he is searching for, it has to feel like some kind of reward and emotionally fulfilling. In that way it also has to be satisfying and enjoyable to use so the user wants to come back and use it again. This does not mean that the system has to be fun and very entertaining because the system is an information tool. If it had trailers and screenshots, it would be more entertaining but it is seen as a tool to find quick information. The motivation of this system will appear automatically when the user finds himself in need of some information in a hurry.

Because this system is not a very important safety system like a bank account, it does not have to be safe with a lot of security. Since this system is small and does not have a lot of function it doesn't need to be secured as much. That is why it does not appear to be that important.

This mobile web-application is not a big system which means that is easy to understand and to use. This is because the user does not have to learn how to use it, because it should be intuitive. This does also mean that memorability is prioritized as less important because the system is easy to remember.

The system does not contain any kind of support of creativity because the user does not have a lot of functions that can do this. The user only has the ability to write a game rating and create an account. If the system should support some kind of creativity it has to have more functions that can help the user to personalize the system to fit his needs. This system has a solid framework which doesn't give the user a lot of opportunities to think different ways to use the system.



## 4.2 DATABASE STRUCTURE

In this section the structure of the database will be described based on the ER-Diagram. The database is the part of the system that is responsible for containing the data that is displayed on the website; this information is e.g. data about each game. The database consists of 5 different tables; all of which will be described. These descriptions will cover the attributes and data types of each table (attributes will be written in **bold**, data types in *italic*).

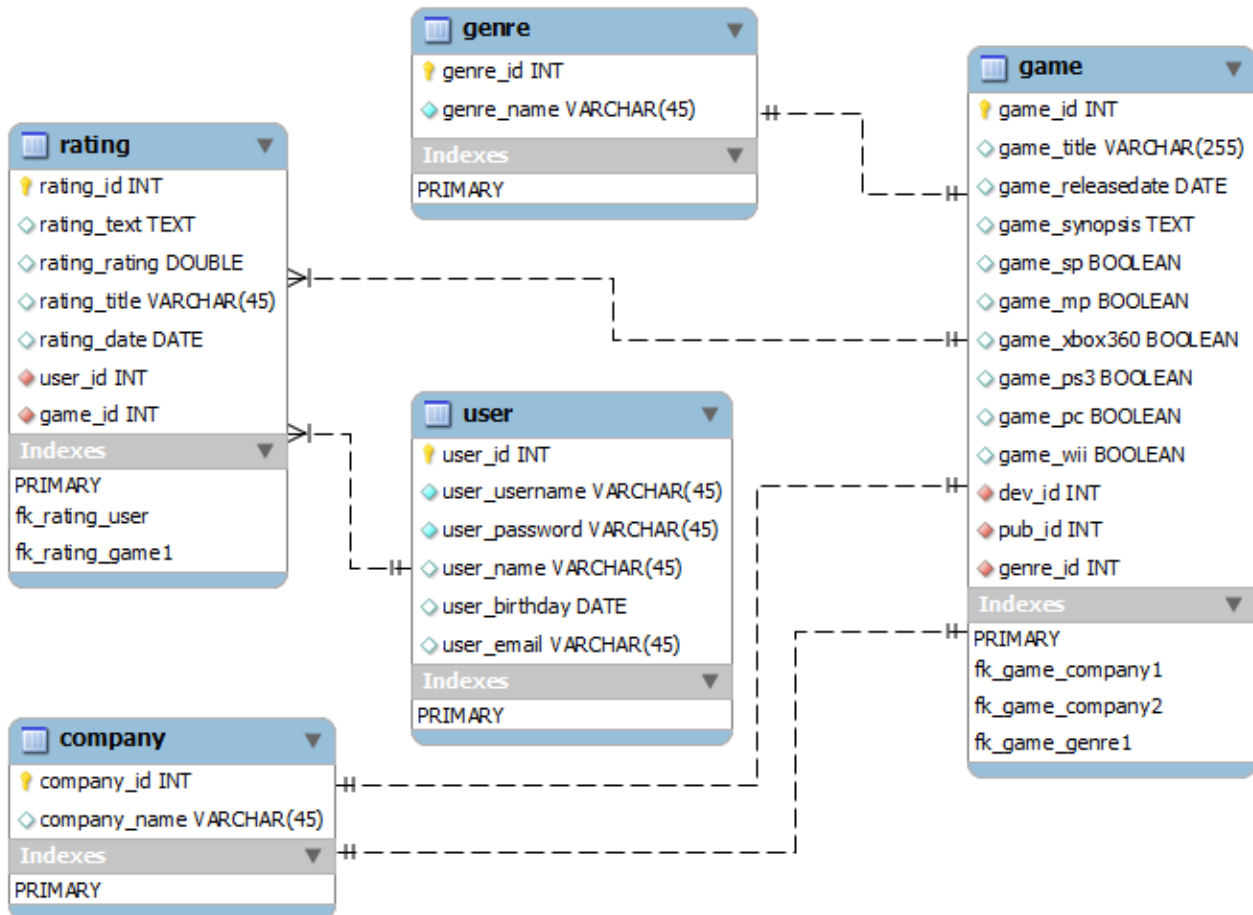


Figure 10

The schema above (figure 10) shows the tables of the database and what each of these tables contains. These tables have a primary key and some of them use foreign keys and there will be some relations between the tables. The database contains five different tables as seen in the diagram. The tables are company, user, rating, game and genre. The primary keys of the table are illustrated by a yellow key and the foreign keys contain an “fk” in front of the name. The line between the tables describes the relation between them. Each table has different attributes. These attributes are all named after the table they derive from. Each table and its attributes will now be described; furthermore the relations will be described at the end of this chapter

#### 4.2.1 GAME

The game table contains the information that is important to know about a game. The information is title, release date, synopsis, single player, multiplayer, platforms, developing company, publishing company and genre.

**game\_id:** This is the unique ID of a game. Every game must have an assigned unique ID; this way there will always be a clear distinction between games. This ID was made the unique ID because some of the other attributes in the table might be duplicated, e.g. two different games might have the same name which makes the game title unsuitable as a primary key. This attribute may not be NULL either, due to the fact that the primary key is used as a unique identifier. The attribute is set to auto increment so it will automatically set it to the next available value. The chosen type for the attribute is *INTEGER(11)* the length is set to 11, this is big enough for this system for the time being. And the max value of an *INTEGER* is 2.147.483.647, which is more than plenty. This kind of ID is used for each of the tables in the database.

**game\_title:** The **game\_title** column stores the game title of each game, for this attribute the type *VARCHAR(255)* has been chosen, with the default length of 255. This specific type was the most logical pick in this case, because *VARCHAR* is best when the amount of characters needed is few.

**game\_releasedate:** This attribute determines the date where the game has been released on the market. The most logical type for this attribute was picked which was the *DATE* type. The *DATE* type saves information in the format Years-Months-Day (YYYY-MM-DD). All dates throughout the system will also use this type.

**game\_synopsis:** This attribute contains a short synopsis about the game, what the game is about etc. The type that has been picked for this attribute is the *TEXT* type. This specific type is suitable for this attribute as a synopsis is generally very short and the *TEXT* type can be up to 65536 characters long which are more than ever needed for a short synopsis.

**game\_sp:** This attribute determines if the game has a single player part. On figure 14 it shows that this attribute has been picked as the type *BOOLEAN*. This type is basically a synonym for *TINYINT(1)*, where a value of zero is considered false and a nonzero value is considered to be true. What this means is that if the value is set to 0 it means the game doesn't have a single player part and vice versa.

**game\_mp:** This attribute determines if the game has a multiplayer part. The attribute has the same type as picked for **game\_sp** and for the same reasons.

**game\_xbox360:** This attribute determines if the game is available on Microsoft's Xbox 360. The attribute has the same type as picked for **game\_sp** and for the same reasons.

**game\_ps3:** This attribute determines if the game is available on Sony's PlayStation 3. The attribute has the same type as picked for **game\_sp** and for the same reasons.

**game\_pc:** This attribute determines if the game is available on the Windows PC platform. The attribute has the same type as picked for **game\_sp** and for the same reasons.

**game\_wii:** This attribute determines if the game is available on Nintendo Wii. The attribute has the same type as picked for **game\_sp** and for the same reasons.

**dev\_id:** This attribute refers to the primary key **company\_id** in the company table; it is used to determine which company is the developer for each specific game. The type picked for the attribute is *INT(11)* and is picked for the same reasons as the **game\_id**. More about this attribute and its table will be described later in this chapter.

**pub\_id:** This attribute is basically the same as **dev\_id**, with the exception that this attribute is used to determine the publisher of each specific game.

**genre\_id:** This attribute refers to the primary key **genre\_id** in the genre table; it is used as a relation to determine which genre the game is under such as: Action, Racer, and MMO etc. The attribute has the same type as picked for **game\_id** and for the same reasons.

#### 4.2.1.1 CREATING THE TABLE IN THE DATABASE:

When the table has been designed, the following SQL command was used in MySQL:

```
CREATE TABLE `game` (  
  `game_id` int(11) NOT NULL auto_increment,  
  `game_title` varchar(255) default NULL,  
  `game_releasedate` date default NULL,  
  `game_synopsis` text,  
  `game_sp` tinyint(1) default NULL,  
  `game_mp` tinyint(1) default NULL,  
  `game_xbox360` tinyint(1) default NULL,  
  `game_ps3` tinyint(1) default NULL,  
  `game_pc` tinyint(1) default NULL,  
  `game_wii` tinyint(1) default NULL,  
  `dev_id` int(11) NOT NULL,  
  `pub_id` int(11) NOT NULL,  
  `genre_id` int(11) NOT NULL,  
  PRIMARY KEY (`game_id`),  
  KEY `fk_game_company1` (`dev_id`),  
  KEY `fk_game_company2` (`pub_id`),  
  KEY `fk_game_genre1` (`genre_id`)  
);
```

## 4.2.2 GENRE

The genre table describes the different game genres such as action, racer, MMO etc. These genres are e.g. used when a game is looked up and it is possible to look for games based on specific genres, which are picked by the development team.

**genre\_id**: This is the unique ID for each genre. This kind of id is the same as seen in **game\_id**.

**genre\_name**: This attribute contains different genre names, such as action and racer. The type picked for this attribute is *VARCHAR(45)* has been chosen, with the length of 45. This specific type was the most logical pick in this case, because *VARCHAR* is best when the amount of characters needed is few and it was reduced to 45 from its default which is 255.

### 4.2.2.1 CREATING THE TABLE IN THE DATABASE:

When the table has been designed, the following SQL command was used in MySQL:

```
CREATE TABLE `genre` (  
  `genre_id` int(11) NOT NULL auto_increment,  
  `genre_name` varchar(45) NOT NULL,  
  PRIMARY KEY (`genre_id`)  
);
```

## 4.2.3 USER

The following table contains all the information about each user, when someone creates a user on the website, he will fill out some information according to the attributes in this table. A user is necessary if a user wants to rate a game.

**user\_id**: This is the unique ID for each user. This kind of id is the same as seen in **game\_id**.

**user\_username**: When a user creates a user, he will be asked to make a username. This username is saved in this attribute and the user will have to login when he wants to access user-restricted content on the website. The type picked for this attribute is *VARCHAR(45)*, for the same reason as described in **genre\_name**.

**user\_password**: Alongside with the username, a password has to be provided when a user wants to log onto the website to see restricted content or to get extra permissions on the website. The type picked for this attribute is *VARCHAR(45)* with the length of 45; this type seemed like the most logical pick for this attribute. A noticeable element about this password attribute is that they are all MD5 decrypted, which means that admins won't be able to read users passwords at a first glance.

**user\_name**: This attribute is used to store the user's real name, right now this information isn't used for a whole lot, but the information could be useful in further development of the system. The type picked for this attribute is *VARCHAR(45)*, for the same reason as described in **genre\_name**.

**user\_birthday:** This attribute is used to store the user's birthday, right now this information isn't used for a whole lot, but the information could be useful in further development of the system, this information could e.g. restrict users from viewing certain games according to the PEGI ratings, which is a European video game content rating system. As the **game\_releasedate** this attribute is also of the type *DATE*.

**user\_email:** This attribute is used to store the users email, the user's email can be used as a link between the system and admins to the user, e.g. if a user gets locked out of the system. The type picked for this attribute is *VARCHAR(45)*, for the same reason as described in **genre\_name**.

#### 4.2.3.1 CREATING THE TABLE IN THE DATABASE:

When the table has been designed, the following SQL command was used in MySQL:

```
CREATE TABLE `user` (  
  `user_id` int(11) NOT NULL auto_increment,  
  `user_username` varchar(45) NOT NULL,  
  `user_password` varchar(45) NOT NULL,  
  `user_name` varchar(45) default NULL,  
  `user_birthday` date default NULL,  
  `user_email` varchar(45) default NULL,  
  PRIMARY KEY (`user_id`)  
);
```

#### 4.2.4 RATING

The information in this table is used when a user wants to write or read a rating. The information is used to give the users of the website an idea of how good a game is based on opinions of other people. Everyone can always read ratings, but you can only write them if you are logged into the system. This is created so that the system won't get an overflow of content that is of a lower quality, so the user has to be logged in to make a rating and a user can only make one rating for each game.

**rating\_id:** This is the unique ID for each rating. This kind of id is the same as seen in **game\_id**.

**rating\_text:** This attribute contains the short review of each rating, as each time a user rates a game he has to give a small review for the game for other users to read. The type picked for this attribute is *TEXT* this type has been picked because the length of *TEXT* can be up to 65536 characters long, which is more than ever needed, even when it is considered that this system has its main focus on smart phones.

**rating\_rating:** This attribute is the rating a user can give. The user will be able to give a game stars, the user can give 1,2,3,4 or 5 stars to a game, where 1 star is the lowest and 5 stars is the highest. The type picked for this attribute is *DOUBLE*, The reason that *DOUBLE* has been used is because it allows the use of comma which is needed when an average of all ratings for a game has to be found.

**rating\_title:** When a user rates a game he also has to provide a small title for his rating, this attribute saves this information. The type picked for this attribute is *VARCHAR(45)*, for the same reason as described in **genre\_name**.

**rating\_date:** This attribute is used to see what day a rating by a user has been made as information to other users and admins. As the **game\_releasedate** this attribute is also of the type *DATE*.

**user\_id:** This attribute refers to the primary key **user\_id** in the user table; along with **game\_id** this attribute is made to make sure that only one rating for a game by a user can exist. The type picked for the attribute is *INT(11)* and is picked for the same reasons as the type picked for **game\_id**. More about this attribute in the section on the table from where this attribute originates from.

**game\_id:** This attribute refers to the primary key **game\_id** in the game table; along with **user\_id** this attribute is made to make sure that only one rating for a game by a user can exist. The type picked for the attribute is *INT(11)* and is picked for the same reasons as the type picked for **game\_id**. More about this attribute in the section on the table from where this attribute originates from.

#### 4.2.4.1 CREATING THE TABLE IN THE DATABASE:

When the table has been designed, the following SQL command was used in MySQL:

```
CREATE TABLE `rating` (  
  `rating_id` int(11) NOT NULL auto_increment,  
  `rating_text` text,  
  `rating_rating` double default NULL,  
  `rating_title` varchar(45) default NULL,  
  `rating_date` date default NULL,  
  `user_id` int(11) NOT NULL,  
  `game_id` int(11) NOT NULL,  
  PRIMARY KEY (`rating_id`),  
  KEY `fk_rating_user` (`user_id`),  
  KEY `fk_rating_game1` (`game_id`)  
);
```

#### 4.2.5 COMPANY

The genre table describes the different game companies that can be tied to a game. These companies can either be developer or publisher or both for a game. This information is used when a single game is looked up and it is possible to find games by searching on company name.

**company\_id:** This is the unique ID for each company. This kind of id is the same as seen in **game\_id**.

**company\_name:** This attribute contains different company names, such as Valve and Electronic Arts. The type picked for this attribute is *VARCHAR(45)*, for the same reason as described in **genre\_name**.

#### 4.2.5.1 CREATING THE TABLE IN THE DATABASE:

When the table has been designed, the following SQL command was used in MySQL:

```
CREATE TABLE `company` (  
  `company_id` int(11) NOT NULL auto_increment,  
  `company_name` varchar(45) default NULL,  
  PRIMARY KEY (`company_id`)  
);
```

#### 4.2.6 RELATIONS WITHIN THE DATABASE

As seen on figure 14, there are a total of 5 lines between the different tables, these lines describes the relations between the tables. In this section each of these relations will be looked upon.

##### 4.2.6.1 GENRE AND GAME RELATION:

This relation between game and genre can be seen in the game table, because in the game table you can see the foreign key **genre\_id**. This relation is used when a game needs to get a genre defined which is being used when a user e.g. want to sort games by genres. How this is being implemented in the system is that a game can e.g. have the **genre\_id** 1, which on the genre table has the **genre\_name** "Action". The reason why this has been implemented this way is because the design team saw it as an easier way for the admins to change genre information and add new genres.

##### 4.2.6.2 USER, RATING AND GAME RELATION:

This relation is essentially being created to enable us to see who has made each rating in accordance to the related game. In the rating table the foreign keys **game\_id** and **user\_id** exist and by implementing these keys correctly in the system it has been made sure that not one user can write two ratings for the same game.

##### 4.2.6.3 COMPANY AND GAME RELATION:

The last important relation is the company and game table relation. The reason why this relation has been made is because one game has a developing company and a publishing company behind it. But in some instances the same company can be a developer on one game and a publisher on another or in some cases both for a single game. So to make sure that there was no unnecessary duplication of data, this setup was made.

## 4.3 INTERACTION

To better understand the interaction between the user and the system there are used different kinds of interactions. This should help to determine how an object is picked and activated. It also shows how data is read and placed on the page.

The system uses menu-structuring that helps the user to navigate through the system. Some of the menu choices will contain submenus that lead to other pages that are relevant. An example of this is the profile page that contains buttons for other pages like settings or my ratings. This helps the user to organize information which gives a controlled information- and functioning overview. The menus advantage is that it does not require a lot of training to understand how to navigate throughout the system. The disadvantages of menus are the risk of having too many menus which slow down the process. Menus also use a big space on the screen.

Another kind of interaction that is used in this system is dialog. It appears when the user e.g. wants to delete a rating. A dialog box will ask if the user is sure that he wants to delete the current rating. There is also dialog every time the system is loading a new page which is important so that the user knows that the system is processing the user's request. When new users have to register or write a rating, the system checks if something is missing or typed wrong. If an error occurs, the system will inform the user what the problem is. The user will always know if a task is complete or something has gone wrong.

When the user has to fill a form in situations like registration or edit a rating, the system uses a form that tells the user what to type in. This helps the user to enter the information faster, and it is easy to understand. This means that the user does not have to train to use a form interaction. The user will always know what to enter till the task has been completed.

Another interaction that is used in this system is post-WIMP (Windows, Icons, Menus and Pointers). The reason for this is that the system has icons that lead to different pages. This is also called direct manipulation because the user chooses what the system should do. An example of this is the moment when the user has to make a rating for a specific game. The user can then choose directly how well a game should be rated. Because the system has a touch-interface, the system does not have a cursor as known from e.g. Microsoft Windows interfaces. This means that he has to use his finger to interact with the system.

The last interaction that can be found is the printing of data which appears when a list of search results is shown or a specific game has been chosen. Because the system uses a database, the different data that are found in the tables will be written in all situations where needed.

All these interactions should be intuitive which is also called NUI (Natural User Interface) where the user always knows how to interact with the system.



## 4.4 WINDOW DIAGRAMS

These sketches form the basis of upcoming user-interface for the end-user.

Although crude in appearances, they have most of the functions illustrated in them. The sketches will be showed individually accompanied by a short description of design and purpose.

The inspiration of the chosen design comes from the general designs already commonly used on the internet for mobile devices. The sketches will also form the base for the navigational diagram, which will be presented afterwards.

The program used to create these sketches is Balsamiq Mockups and the sketches are presented in no particular order.

### 4.4.1 GENERAL DESIGN

Included in most of the sketches, there will be a navigation bar at the top of the screen with four action-buttons:

- News
- Search
- Games
- Login/profile

This will help the user to navigate quickly to the four most common used functions throughout most of the system. This will however be replaced by information when the user activates specific buttons. If the user wants to return to the previous page, he will use a return button instead, as known from web-browsers.

Note that the search function has been marked with an “S” in these sketches due to space-limitations in these sketches.

Use of common known design will be used, in order for easy recognizable operation of the system, which is important for the development of this system, as seen on the prioritizing table.

These commonly known design factors are the scrollbars, checkboxes, search fields etc., which will be used throughout the system. The goal is to be an intuitively fusion between web browsers and smartphone applications, which the user is already using, since he accesses the system from the phone.

## SEARCH WINDOWS

This window provides a basic search field and a coherent action button aligned with it. Furthermore, the user can use the check boxes provided in order to specify results.

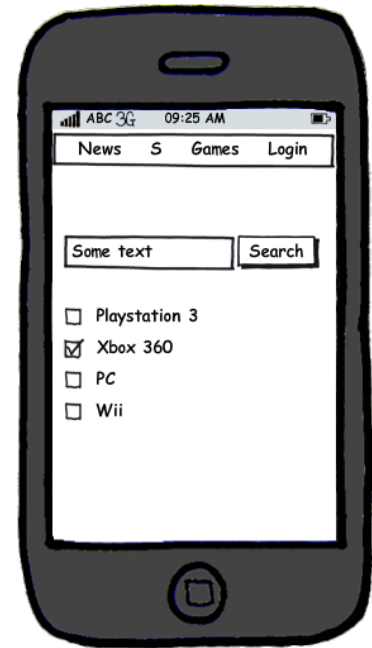


Figure 11

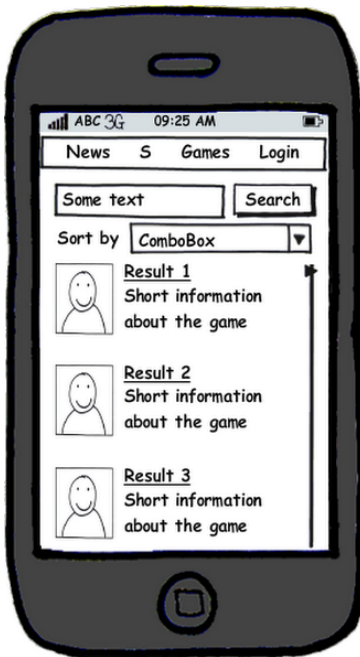


Figure 12

This window shows the results found in the search. An important thing is the use of pictures, since this is one of the first eye-catchers when skimming through the results. In order to sort the results, the user is given a drop-down box in which he can sort by some predetermined preferences.

The scrollbar helps the user to assess the quantity of results.

## NEWS

When the user activates the “News” action button from the navigation bar, he will be redirected to this page. Here the user can see which games are the newest added to the game database. The games shown will be sorted by date, starting with the newest. A predetermined limit of shown results will be made in order to limit results to a fixed number. This is due to the internet-traffic consumption of the user’s subscription. If the user pays his phone service provider by traffic use, it could be quite costly if the results shown are too many.

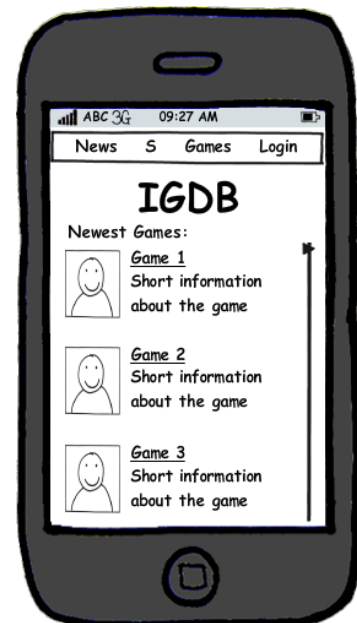


Figure 13



Figure 14

## PROFILE

When the user has logged in, he will be redirected to this page, where his information will be displayed in the top of the screen. An important feature is that the top navigation bar has changed the last action button from “Login” to “Profile”. Furthermore this screen contains three large action buttons, which are deemed the most relevant for the user in order to use the “logged in” functions.

This screen shows the settings menu. It is a typical settings menu, in which the user can change data such as e-mail address or password. The save button in the bottom is put there as the last step after editing the data needed.

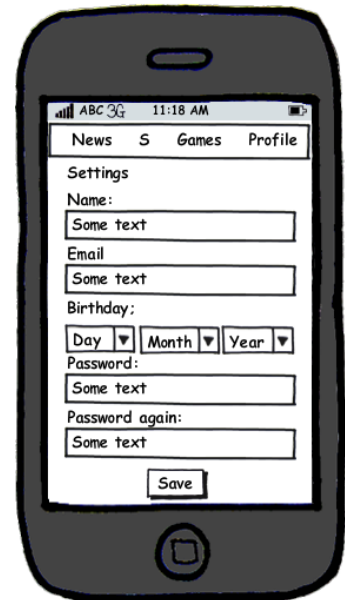


Figure 15

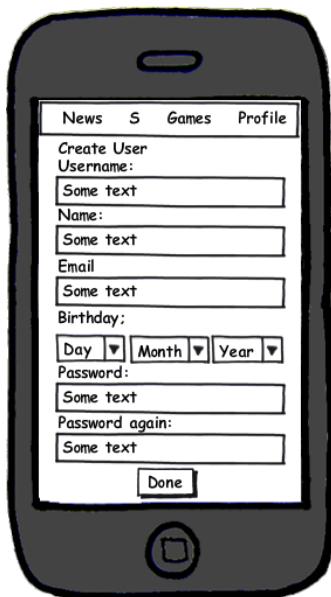


Figure 16

If the user is new to the system and he wants to rate a game, he must create a user. He will only have to provide basic information such as shown on this screen. The password has to be typed in twice in order to minimize the chance of misspelling it. The “Done” button is used to create the user with the information given.

If the user presses "Login" at the top of the navigation bar, he will have to type in a username and password or create a new user.

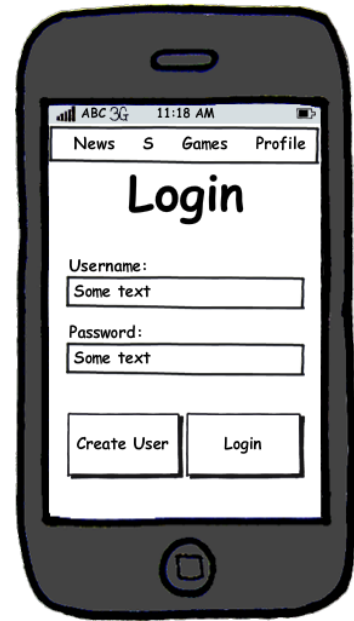


Figure 17



Figure 18

## GAMES

Pressing the "Games" button in the navigation bar, the user will be prompted with this window, giving him several sorting options, which are listed below.

- Genre
- Popular
- All Games
- Platform options

This gives the user quick options to see which sort of games he wants to see.

If the user uses the "Genre" action, he will be prompted with this screen, where he has to choose which genre of games he wants to see.

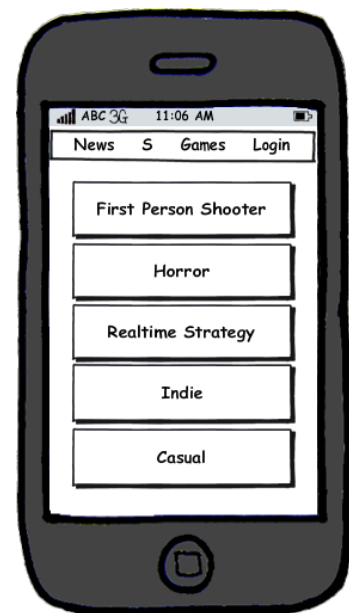


Figure 19

This screen shows the results derived from the users actions. The results view are the same whether the user pressed “Popular” or specified a specific genre, only the results them self are different.

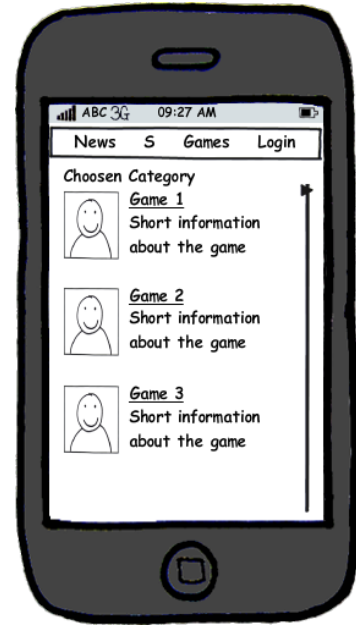


Figure 20

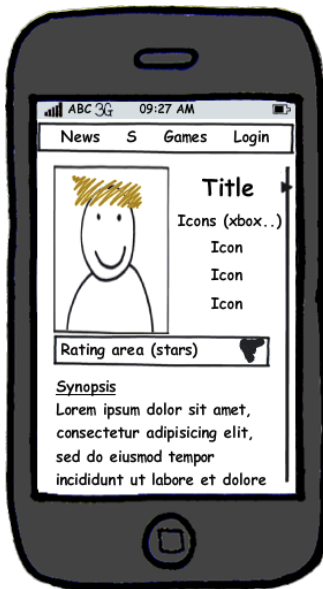


Figure 21

When the user chose a game from any of the previous screens, he will be prompted with this screen. A picture, the cover of the game, will be shown at the top of the page along with basic information such as platforms, title and user-ratings. The purpose is to give a quick overview of the game before reading some of the ratings that will appear when the user scrolls down.

If the user has successfully logged in, he can now give a rating of the game he has chosen. He has to fill in basic information such as a rating title and a text description of his views upon the game. He can then press “Send” in order for the rating to be added to the database. All fields are mandatory to fill in and the user will be asked to fill everything in before sending, if he tried otherwise.

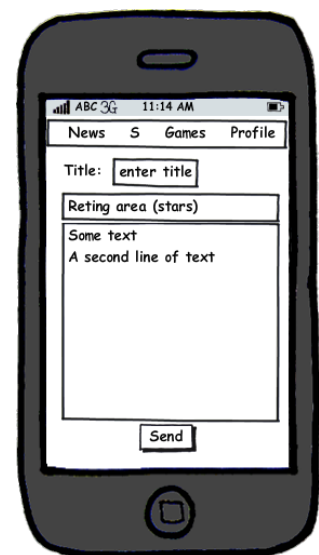


Figure 22

## ERROR WINDOWS

If an error occurs in the system, an error screen will be presented.

It will have an error statement, with a short description of why the error occurred, e.g. the typed in password was wrong. It will only contain a “Back” action button, since the user has to return to the previous screen and correct the mistake or try something else.

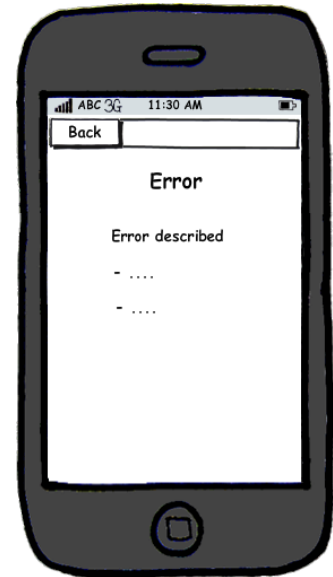


Figure 23

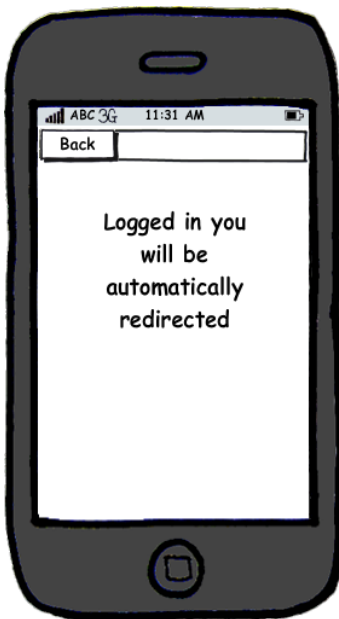


Figure 24

## REDIRECTION WINDOWS

When the user has logged in, he will be redirected to the page containing the newest games. He will see this screen in order to confirm that he has successfully logged in.

## 4.5 NAVIGATION DIAGRAM

The navigation diagram helps the group to get the same understanding of how a user would navigate throughout the system plus the navigation diagram gives the reader an idea of how the user navigates throughout the system. The reader should be aware that there may be some minor changes in the final system compared to the navigation diagram.

In this section of the report the navigation diagram will be described in depth to help the reader get a better understanding of the system and how you navigate in it. As the navigation diagram is being described the reader will be advised to look at the navigations diagram from time to time when reading the descriptions.



Figure 25

The reader will notice that the bar in figure 25 is the centre of the navigation diagram. This bar is on almost every single window so instead of drawing arrows each time this bar appears; the bar has been taking out of the windows and will act as the centre of the navigation diagram. Each time the bar will appear in a window, the user will be able to click and too see where the user will head when a button has been clicked, the reader should look at the centre of the navigation diagram to see what the outcome of this click will be. The buttons that appears in this bar are from the left: News, Search, Games and login/profile. The reason that the button on the right is named "Login/Profile" is because depending on the user is logged in or not that button will have a different caption.

### 4.5.1 NEWS

When the user clicks the news button, he will get a list of games where the newest addition to the database will show on top. The user can then click on a game on the list to get more information about the game. This information includes a short synopsis, developer and publisher information and the rating of the game. If the user scrolls down a list of short ratings will appear. From here the user can also make a rating to the game, but only if the user is logged in and haven't already made a rating for the specific game, because then the user will simply be editing the already written rating. When the user is done creating or editing a rating he will be redirected back to the game page.

### 4.5.2 SEARCH

The search function will help the user find a specific game quickly. He can write a keyword to search for and use some of the functions to help finding the exact game he is looking for. E.g. the user can specify which platforms he wants results shown for. When the user have written in the keyword and perhaps specified his search by using one of the advanced search functions. He can press the button search and a list of search results will appear, from here the user can edit his search keyword a bit and the results can be

sorted for convenience. When the wanted game has been found the user can click on it and gets information about that game, this information was described in the previous section called news.

#### 4.5.3 GAMES

When the user clicks the games button a window full of button will appear these are: Genre, popular, pc Xbox 360, Wii, PlayStation 3 and all games. If the user clicks any button except genre he or she will get a list of games that fits into the chosen category. If the user picks the button genre a list of buttons with genres will appear, such as First person shooter, Real-time strategy, indie etc. When one of these genres is picked, a list will appear of games that fit into the chosen genre. No matter what the user will always get a list of games and when the user clicks on any one game, further information will be presented; this information was described in the previous section called news.

#### 4.5.4 LOGIN

When the bar shows the button named “login” it means that the user hasn’t logged into the system yet. When the user then clicks this button, the user will be able to login to the system or create a profile. If the user chooses to create a profile he will have to fill out some information and if some of the information isn’t filled out when the user presses send an error page will be represented specifying what the error is. If the registration is successful, the user will be sent back to the login page. The user can also choose to log in with an existing account. When this is done the user will be redirected to the front page of the website.

#### 4.5.5 PROFILE

When the bar shows the button named “profile”, it means that the user is already logged into the system. On this page the user will be presented with information about his profile and will be able to see ratings, settings and log out. If the user chooses to logout the front page will be presented and the user has to log back in again if the need be. The user can also from the profile page see his ratings and edit them or see his settings and change some of them, e.g. password.

#### 4.5.6 ERROR MESSAGE:

As the reader may have noticed, is that no arrows point to the error window, this was done to ensure readability throughout the navigation diagram, as the error message can be shown in many cases, like when the user forgets to type in information when he creates a profile or when he tries to log in and types in the wrong password the user will get an error message.

The navigational diagram will now follow.



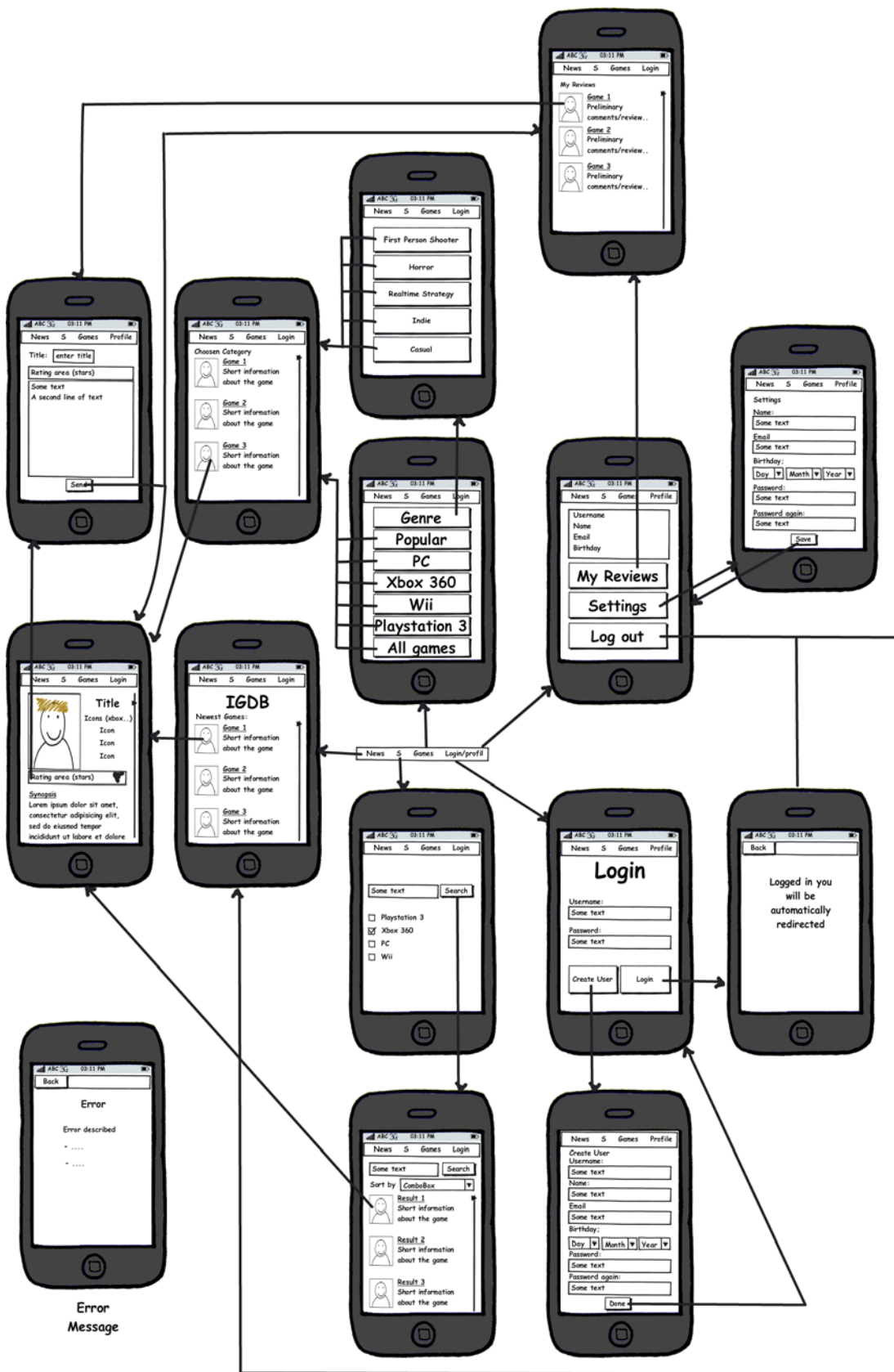


Figure 26

## 4.6 GESTALT THEORY

Following this navigation diagrams the final design can now be made. To do this the gestalt theory put to use which the following two examples will illustrate (Gestaltlovene, 2010).

Humans are different, but they have some things in common. One of these things is the way things are being observed. People look at things in patterns and wholes. The definition of gestalt theory is wholes, form and experience as a whole. It involves how visual input is being perceived by people. The gestalt law says the whole is more than the sum of the parts. When you look at a gallery, a picture, a website or other kinds of combination of these elements is the whole seen before the different parts.

These design laws have all been implemented in the system.

### 4.6.1 LAW OF PROXIMITY

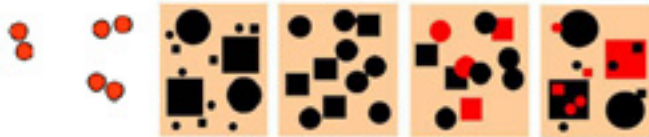


Figure 27

The law of proximity is shown when an amount of objects are closer to each other than others. The law says that object that is placed close to each other has a context. This means that the distance between objects and elements can say which belongs together.

### 4.6.2 LAW OF SIMILARITY



Figure 28

When looking at some objects that look alike does people often see these as identical. When grouping elements it will be done after form, colour and size. This law works best if it is combined with the law of proximity.

### 4.6.3 LAW OF CLOSURE

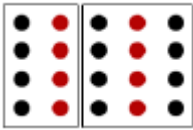


Figure 29

Objects are seen belonging together when a frame is used around them. It can be a physical frame of lines or a frame created by invisible lines.

### 4.6.4 LAW OF CONTINUITY



Figure 30

This law is about how people can separate continuity lines and curves where figures and elements on these lines or curves belong together.

These are just a few of the gestalt laws. They are seen as the most recognized of the laws. The reason to use these laws while designing interfaces is because the user will get a more intuitive experience. The different gestalt laws help organizing information so that the user better can manage the containing of the page. It makes sure that functionality and design are working together which is the reason why it is prioritized high in the developing industry.

#### 4.6.5 EXAMPLES

Based on this theory and the sketches, the UI have been created. To illustrate that we have used this theory, two screenshots will now be shown in order to show how the laws are implemented.

The figures are all actual screenshots from the game database.

Figure 31 shows use of the laws of proximity and similarity. The top navigation bar has 4 action buttons, which have the same form, shape, colour and form of icons. The only button which is altered is the current active button, illustrating that the user is current looking at the profile screen. This helps the user to understand that these buttons are connected together, as a quick navigation tool.

These laws are also implemented in the remainder of the screen, since both the “username” and “password” fields have the same size, font and have same alignment in the screen. Same thing goes for the “login” and “register”

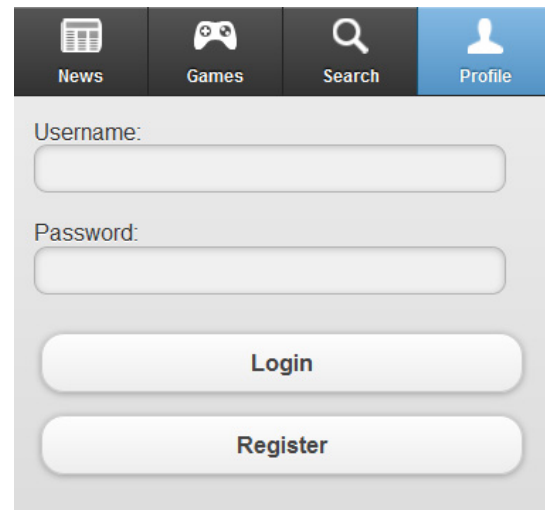


Figure 31

The law of continuity is also used, since the navigation is shown at the same place, with the same buttons, throughout many of the screens in the UI.

Figure 32 also shows several of the design laws used.

The law of similarity is used in the icons below publisher information. The have same size and aligned which gives the user a sense of similarity. Since these buttons are placed close together it also helps the user to regard them as a whole, due to use of the proximity law.

The law of closure law is also discretely implemented by use of an almost invisible line, separating each game from one another.

This whole screen is also using the law of continuity, which can be seen in the placement of each button/icons/pictures. This helps the user to gain a quick overview of the screen, using some basic structure.

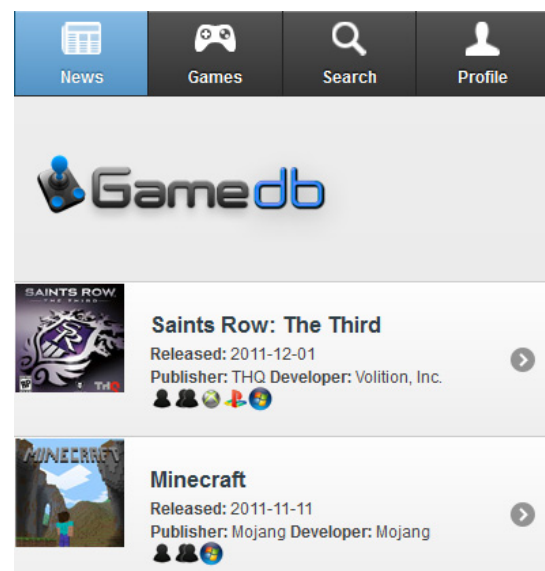


Figure 32

## 4.7 FOUR DESIGN PRINCIPLES

(Skov, 2010) In this chapter the focus will be on the four design principle which contains "affordance", "mapping", "consistency" and "feedback". These principles determine how good the design is. It is important as a developer to have a focus on these to build the best design as possible.

The human abilities and habits should be taking into consideration in order to create as good a design as possible. If this happens, there will be a risk that the system would not be used. That is the reason why the product has been designed with the principles in mind. Every principle will be described in detail and how it is used in the final product. A system should be flexible in the way that the system is useful for both experienced and non-experienced users.

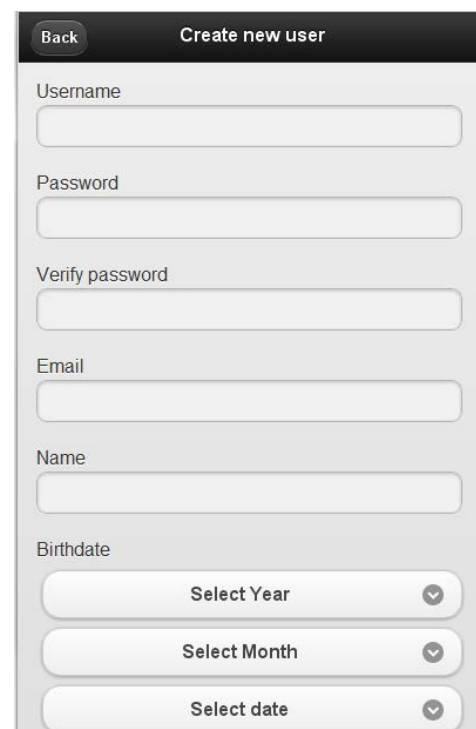
### 4.7.1 AFFORDANCE

The definition of affordance covers both visibility and constraints. Visibility is a description that shows how elements could be presented compared to how it should be used. The user shouldn't end up in a situation where they can't figure out what to do to solve a task. Unlike visibility, the definition of constraints covers the area where the system makes sure that the user does not interact with the system in a way that is not intended.

The user control and freedom are quite important in a system because it is possible that they can make a mistake that was not intentional. That is why it is important to make a sign that shows the user how to cancel so that the user can return to the task they were working on without walking through a long dialog with the system. It is also important to make objects, functions and opportunities clear so that the user can easily learn and remember how to solve a specific task.

In the product are the tools, known from Windows interfaces and smartphone applications, which helps the user to recognize functions they seem familiar. This means that the user knows how to use checkboxes, scrollbars, menus etc. without any doubt. In this system simplicity has been categorized as important so that the different pages don't contain too many information which gives a bad overview. The buttons has a clear text that tells the user what function it does.

The system strives to ensure that the user always knows where the different buttons or links lead. An example of this is the "submit" and "back" buttons that tells the user what function that will be activated. The user will always know what it means in the context. When a user wants to create a new account, he knows that when pressing the "submit" button, his account information will be stored.



The image shows a mobile application interface for creating a new user. At the top, there is a dark header bar with a 'Back' button on the left and the title 'Create new user' on the right. Below the header, the form consists of several input fields: 'Username' (a text box), 'Password' (a text box), 'Verify password' (a text box), 'Email' (a text box), and 'Name' (a text box). The 'Birthdate' field is implemented as three stacked dropdown menus: 'Select Year', 'Select Month', and 'Select date', each with a downward arrow icon on the right side.

Figure 33

The user does also know what has to be filled in the form when registering because there is a text close to every textbox or dropdown functions.

#### 4.7.2 MAPPING

The purpose of mapping is to show the context between the control and its effect. When context and control has a good link between them, it is called direct mapping and results in a more intuitive conception for the user. That is why it is important that the system does not contain indirect mapping where he control and effect can't be linked together. The user shouldn't have to use long time to solve a simple task. A heuristic rule says that the system should use words that the user already knows from reality instead of system orientated language that could be misunderstood.

A good of mapping in the system is where the user has to register or edit user information. The user fills in the forms and know that the submit button saves all the information on the current page. When the user is logged in to the profile page he knows that the settings options belong to the current logged in user.

#### 4.7.3 CONSISTENCY

The consistency means that the system has to use consistent objects and functions throughout the whole system. It means that it looks almost the same on every page and uses the same functions so that the user can reuse experiences that he has tried before. It saves the user from spending a lot of time learning to use the system. It does also mean that the user don't have to remember the system and how to use it, because it has to be intuitive.

This has been implemented throughout the system because the windows are closely alike. An example of this is the buttons, colours, typography and the different forms. The menu will always be found in the top of the screen. The back buttons are also placed at the same location.

The system uses consistency every time a user goes to a page with a list which is almost the same on every page.

Another place where consistency is used is every time the user has to add or edit some information. The same form is used to help the user to interact faster with the system. The form uses textboxes, dropdown select boxes and buttons that the user already knows.

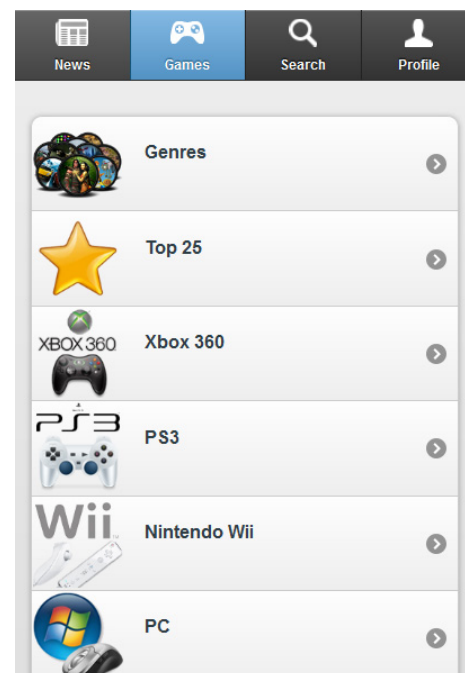


Figure 34

#### 4.7.4 FEEDBACK

Feedbacks are found in situations where the system has to tell the user what has happened or what could happen if he continues. If the system is working on a task, it has to inform the user.

Feedback happens when the user have forgotten some information or typed something wrong in a form. The system then tells the user where the problem is so that the user can go back and change it.

When a new page is loading, the system tells that it is currently loading. When the user logs in or out the system tell whether the user has been logged in or out of the system.

If a user is not logged in to the system wants to create a rating to a game, the system tell the user that he has to be logged in to use this function.

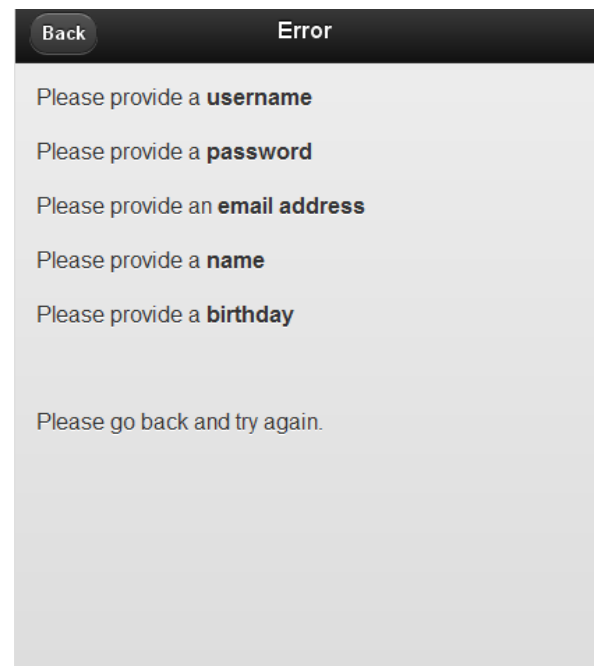


Figure 35

## 5 IMPLEMENTATION

This chapter will cover the most important and interesting parts of the code and how it works. The parts will be described in detail and explained why. These code fragments have all been chosen with the database in mind.

### UPDATE.PHP

```
8 <body>
9
10 <div data-role= "page" >
11
12 <div data-role= "header" >
13 <h1>Error </h1>
14 </div>
15
16 <div data-role= "content" >
17
18 <?php
19 session_start ();
20 $login = $_SESSION['username' ];
21
22 $navn = $_POST['name'];
23 $email = $_POST['email'];
24 $year = $_POST['year'];
25 $month = $_POST['month' ];
26 $day = $_POST['day'];
27
28 $birthday = "year-$month-$day" ;
29 $errors = array ();
30
31 if($navn == "" ) {
32 $errors []= "Please provide a <b>Name</b><br/>" ;
33 }
34 if($email == "" ) {
35 $errors []= "Please provide an <b>email address</b><br/>" ;
36 }
37
38 if (sizeof ($errors) == 0) {
39 mysql_query ("UPDATE user SET user_name='$navn', user_birthday='$birthday', user_email='$email' where user_username = '$login' " );
40 header ('location: update-success.php' );
41 }else {
42 for($i = 0; $i <= sizeof ($errors); ++$i) {
43 $temp = $errors [$i];
44 echo "$temp<br>" ;
45 }
46 echo "<br>Please go back and try again." ;
47 unset ($errors);
48 }
49 ?>
50 </div>
51 </div>
52
53 </body>
```

This piece of code is the update function which changes settings data the user enters into the system. It also checks if the data has been filled in correctly.

At line 19 the session is set to start. Sessions are only used on the pages where needed to keep a user logged in to the system. With the use of `$_SESSION` command, the system ensures that only data which can be changed, are the data of the user which is currently logged in.



In the lines between 22 and 29 variables are declared which gets the data from the specific user. The `$_POST` command writes the information. To make sure that the birthday is written correctly it combines information from year, month and day at line 28.

The error variable has been declared to be an array which helps getting more errors written at the same time in the cases where more errors are found. The error-check between line 31 and 36, checks if nothing has been written in the given fields. Then the user has to provide with the information needed to continue. There is no error check to the birthday because it is not possible to give an empty data because the user can only change the numbers.

If no errors have been found, the `MySQL_query` updates the new information that has been given where `user_username` is the user logged in to the system. The user will then be transferred to a new page where it tells that the information has successfully been changed. If there has been found at least one error a for-loop will be activated to post all the errors that were found. This means that if the user has not provided with a new name and a new email at the same time, the system will tell the user both the errors. The last thing in this code is that the error array is unset and gives the user an instructing text to provide the missing information.

## REGISTER-CONTROL.PHP

```
49     if($Username != "") {
50         $qry = "SELECT * FROM user WHERE user_username='$Username'";
51         $result = mysql_query($qry);
52         if($result) {
53             if(mysql_num_rows($result) > 0) {
54                 $errors[] = "The <b>username</b> you have picked is already user by someone else<br/>";
55             }
56             @mysql_free_result($result);
57         }
58     }
59
60     ...
61
62     if (sizeof($errors) == 0) {
63         $qry = "INSERT INTO user(user_username, user_password, user_name, user_birthday, user_email) VALUES('$Username','".md5($Pa
64         ssword)."', '$Name', '$Birthday', '$Email')";
65         $result = mysql_query($qry);
66         header("location: register-success.html");
67     }else {
68         for($i = 0; $i <= sizeof($errors); ++$i) {
69             $temp = $errors[$i];
70             echo "$temp<br/>";
71         }
72
73         echo "<br/>Please go back and try again.";
74         unset($errors);
75     }
76     ?>
```

In this code section some error checks and adding will be covered. The code will not cover the variables and includes at the beginning because it has previously been covered.

In the first error check the system makes sure that if a textbox is not empty then new username can be added. The system gets contact to the user table in the MySQL where the username gets added. If the username already exist in the system an error comes up when trying to submit where the user gets a feedback telling that the username has been picked by another one.

If no errors have been found the information gets inserted to the user table which is shown in line 73. If an error has been found the for-loop will be activated to write all of the errors.

## CONNECTION.PHP

```
1 <?php
2 $db_hostname = 'sql106.xtreemhost.com';
3 $db_username = 'xth_XXXX';
4 $db_password = 'XXXX';
5 $db_name = 'xth_XXXX';
6
7 $connection = mysql_connect($db_hostname, $db_username, $db_password)
8
9 or die("It seems this site's database isn't responding.");
10
11 $db = mysql_select_db($db_name)
12
13 or die ("It seems this site's database isn't responding.");
14 ?>
```

This file contains relevant information needed to connect and communicate with the database. Line 2-5 defines variables containing required information to connect to the database. Line 7 and 10 then opens up the connection and selects the right database to communicate with. This piece of code is needed every time the website wants to communicate with the database. Therefore it is placed in a separate file, which then is included in all files which communicate with the database. Please note that the “XXXX” notation used in the example above is only for viewer discretion purposes.

## TOPGAMES.PHP

```
11 <?php
12
13 $result=mysql_query ("SELECT * FROM game ORDER BY game_id ASC" );
14 $num=mysql_numrows ($result);
15 mysql_query ("CREATE TABLE tempGames (game_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT, game_title TEXT, game_
rating DOUBLE, PRIMARY KEY (game_id))" );//__%%Temporary Table%%__
16
17 ?>
...
32 $n=1;
33 while ($n < $num+1) {
34 mysql_query("INSERT INTO tempGames (game_title, game_rating) SELECT game_title, avg(rating_rating) FROM rating NATURAL
JOIN game WHERE game_id = $n");
35 $n++;
36 }
37
38 $result2=mysql_query("SELECT game_id FROM tempGames ORDER BY game_rating DESC, game_title ASC");
39 $num2=mysql_numrows($result2);
...
95 <?php
96 mysql_query ("DROP TABLE tempGames" );
97 ?>
```

These pieces of code are taken from the topGames.php file which lists the highest rated games from the database. The first part selects all games and their information from the database, and saves the information into the \$result variable. After this a new table is created in the database. In an optimal environment a temporary table would have been used, as this automatically will be deleted when the queries has been executed.

This was not possible on the chosen server, and therefore this solution with creating a permanent table and then deleting it, has been necessary. The *temporary* table called tempGames will contain id, title and rating from all games. This information is inserted with the while loop at line 33. For every single game found in the database, the **id** and **title** will be selected from the **game** table and **rating** from the **rating** table. This is done at line 34 by the MySQL\_query. A natural join command is used to get information from both the **game** and **rating** table by joining them where game\_id from both tables matches. By this query the games which have one or more rating is inserted into the **tempGames** table.

After this these game will be selected and ordered first by rating and then by title at line 38.

## 6 DEVELOPMENT PROCESS

Using the SCRUM method has resulted in several logs which will be documented in the chapter. It contains a detailed log of how Scrum has been implemented throughout this project, so that the reader will get a comprehensive overview of what has been made and when it was made over the course of the project. The following chapter will describe every sprint in the project with a sprint backlog and in depths description of each sprint and the results here from.

### 6.1 SPRINTS

Every sprint in the developing method has a specific focus. The sprints are an overall framework for developing work, which gives some deadlines and focus points to work with to optimize the work and the developing time.

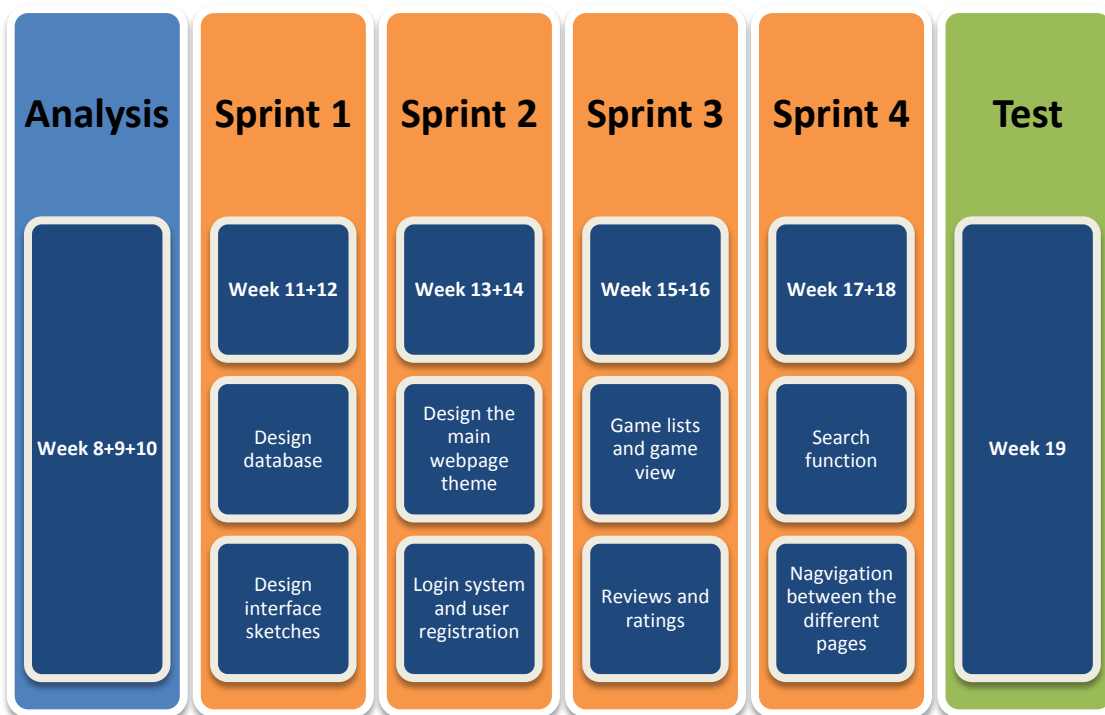


Figure 36

A sprint will last two weeks and for each sprint there will be a sprint backlog with a detailed list of the assignments that are still pending; the sprint backlog will have a status for the beginning of the sprint and then again in the end of the sprint (these tasks can even come from previous sprints). This way the team will be able to see what will have to be pushed to the impending sprint. The sprint backlogs will weigh heavily on the product backlog, so it would be advisable to refer to the product backlog in order to gain a more comprehensive and thorough understanding of the sprint backlogs.

Besides the sprint backlog to each sprint, there will also be an in depth description of each sprint, the group will describe what was the major tasks what the results of these tasks were and if the results somehow was different from what was initially planned for the system.

### 6.1.1 SPRINT 1

Sprint Backlog for the first sprint:

| From product backlog | Key                   | Summary  | Status at the start | Status at the end | Prioritization |
|----------------------|-----------------------|--|---------------------|-------------------|----------------|
| <b>Database</b>      | Database Schema       | Make a schema over the database and explain the database in depth. | TO DO               | DONE              | 1              |
|                      | Database construction | Get a database up and running with the wanted architecture         | TO DO               | DONE              | 2              |
|                      | Table Values          | Insert values into tables (companies and games).                   | TO DO               | DONE              | 5              |
| <b>Design</b>        | Window diagrams       | Make and decide what should be on each window of the website.      | TO DO               | DONE              | 3              |
|                      | Navigation diagram    | Make and decide the navigation on the website.                     | TO DO               | DONE              | 4              |

Table 5

The objective of the first iteration has been designing the database model for the system and drawing rough sketches of the user interface. Based on the system requirements, it has been decided which entities the database should contain, then a schema has been modelled, which contains Entities, attributes and relationships between them. The database has been developed in a MySQL database based on the ER-diagram.

A second aim of this iteration was to decide and draw sketches of the user interface and design window diagrams for each window in the system so that everyone in the group has a common idea of what each window should look like.

#### 6.1.1.1 RESULT

To see and understand the results of this sprint the reader should read the chapter on the design of the system.

## 6.1.2 SPRINT 2

Sprint Backlog for the second sprint:

| From product backlog            | Key  | Summary  | Status at the start | Status at the end | Prioritization |
|---------------------------------|--|--|---------------------|-------------------|----------------|
| <b>Log in and out</b>           | Log in                                       | Make it possible for the user to log in.   | TO DO               | DONE              | 4              |
|                                 | Log out                                      | Make it possible for the user to log out.  | TO DO               | DONE              | 7              |
|                                 | Sessions for user login                      | Make sessions for when a user logs in to the system and keep him logged in.                                | TO DO               | IN PROGRESS       | 4              |
| <b>Registration</b>             | Registration                                 | Make it possible for the user to register to the website.  | TO DO               | DONE              | 2              |
|                                 | Error feedback                               | Make sure that the user gets feedback when he does something wrong.  | TO DO               | DONE              | 2              |
|                                 | Encrypt password                             | The passwords should be encrypted so that the administrator cannot see the user's password in the database | TO DO               | DONE              | 3              |
| <b>Design</b>                   | Design theme                                 | Make a header and each page general design.  | TO DO               | DONE              | 1              |
|                                 | Design Icons                                 | Design icons for header menu items   | TO DO               | DONE              | 8              |
|                                 | Design template                              | Create a template with design codes that every page should include to make the programming process easier. | TO DO               | DONE              | 1              |
| <b>User Information Display</b> | User information                             | Make it possible for the user to see his owns information.   | TO DO               | DONE              | 5              |
|                                 | Change user information                      | Make it possible to change password, email, name and birthday.   | TO DO               | IN PROGRESS       | 6              |
|                                 | Error feed back in changing user information | Make sure that the user gets feedback when he does something wrong.  | TO DO               | DONE              | 6              |

Table 6

The major tasks for the second iteration can be seen in figure 40. The major tasks were as following: Design the main webpage theme and login system and user registration. These major tasks have both been split up into smaller and more manageable tasks as seen in table 6.

### 6.1.2.1 RESULT

On figure 41 you can see the result of the login page. As seen there are four menu items on the header bar: “News”, “Games”, “Search” and “Profile” and every item has its own icon. The register window where the user can create an account is shown on figure 38. As shown, the header change to “Create new user” and a back-button which takes the user

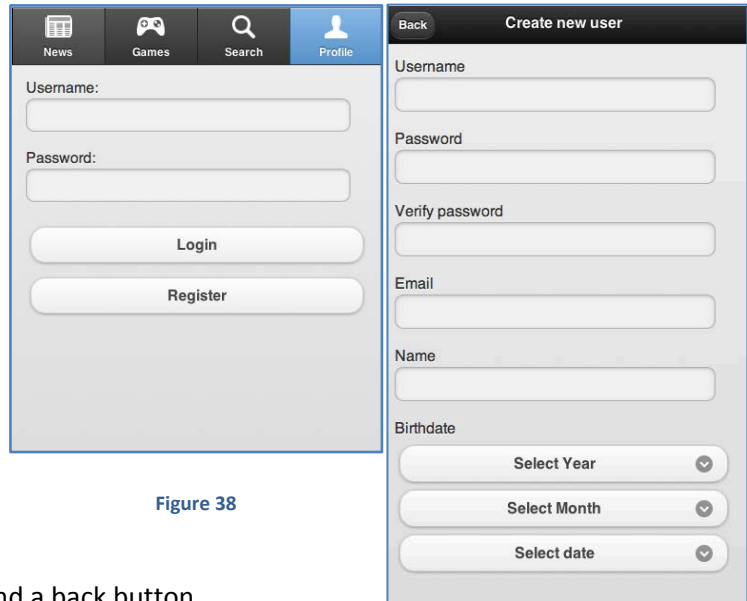


Figure 38

back to the previous page appears on the left. Only the 4 main sites will contain the header bar with the 4 icons, every other sub-menu gets its own header with a title and a back button.

Figure 37

If the user enters invalid data or leaves the input fields empty in the registration page, an error message will appear as seen on figure 39.

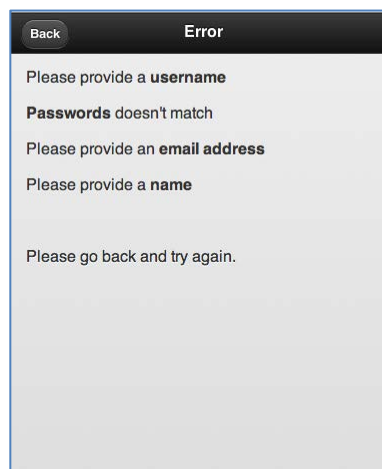


Figure 40

When registration is completed without any errors, the success screen will appear (Figure 40).

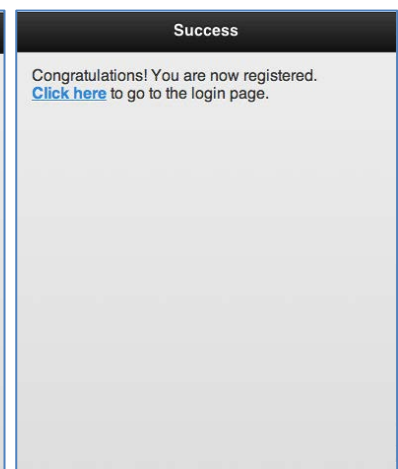


Figure 39

Earlier in the sketch phase it was described that the success window (figure 40), automatically should redirect to the profile page. This function is changed so that the user now has to click the link “Click here to go to the login page”. Then the user will be transferred to the login page where he has to use the created login information to login.

When the user logs in, the profile page with the user information (Figure 41) appears. It was planned that the “Profile” button on the header should change caption to “Log in” if the user was not logged in. This however has been revised and the caption stays the same. There are three buttons on the profile page. “My rating” should direct the user to a page where he can view, edit and delete his ratings.

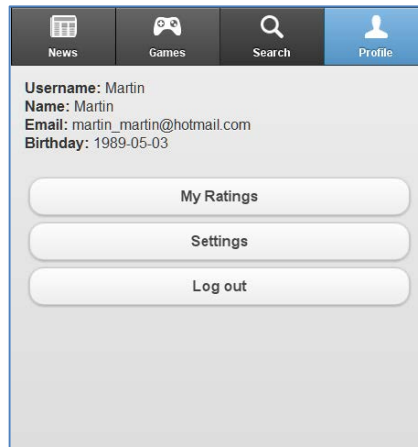


Figure 42

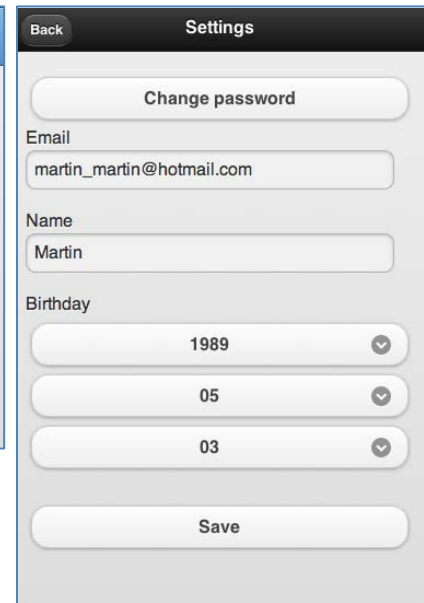


Figure 41

The “Log out” button logs the user out of the system.

“Settings” button goes to a page (Figure 42) where the user is able to change his profile information. On the “Settings” page it was initially planned that the user should be able to change all his information from the same window. This included password changeability as well however this was changed during the iterations so that the user has to access another page to change the password (figure 43). This is due safety precautions regarding the password.

The result of this sprint is a full working login system with the right design, where a user is able to create an account, login, view/edit personal information and log out.

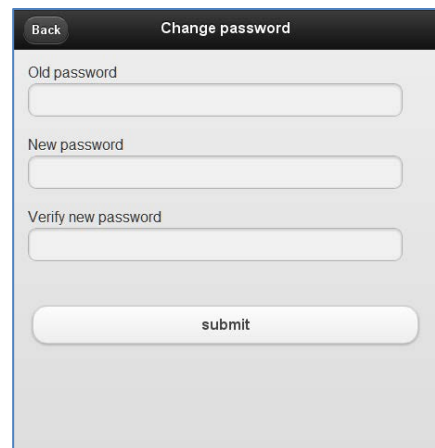


Figure 43



6.1.2.2 SPRINT 3

Sprint Backlog for the third sprint:

| From product backlog            | Key                     | Summary   | Status at the start | Status at the end | Prioritization |
|---------------------------------|-------------------------|---|---------------------|-------------------|----------------|
| <b>Log in and out</b>           | Sessions for user login | Make sessions for when a user logs in to the system and keep him logged in.     | IN PROGRESS         | DONE              | 1              |
| <b>User Information display</b> | Change user information | Make it possible to change password, email, name and birthday.                  | IN PROGRESS         | DONE              | 2              |
| <b>Game information display</b> | Game list               | Make it possible for the user to get a list of games.                           | TO DO               | DONE              | 3              |
|                                 | Game information        | Show a single game's information.   | TO DO               | DONE              | 4              |
|                                 | Ratings list            | Make a list of ratings made by the user to display in the game information page | TO DO               | DONE              | 7              |
|                                 | Icons                   | Design icons to genres, submenus in games page and game information page        | TO DO               | IN PROGRESS       | 9              |
| <b>Front page</b>               | Front page              | Constructing the front page of the program                                      | TO DO               | DONE              | 1              |
| <b>Rating</b>                   | Add rating              | Make it possible for the user to add a new rating to a game.                    | TO DO               | DONE              | 5              |
|                                 | Edit/delete rating      | The user should be able to view/edit/delete his ratings.                        | TO DO               | DONE              | 6              |
| <b>Design</b>                   | Logo                    | Design logo to the website.   | TO DO               | DONE              | 8              |

Table 7

Table 7 illustrates it can be seen how the sprint backlog for the second sprint looks like. All tasks are new except from the first two tasks. The reason for that is because of it was not accomplished during previous sprint so it is moved to this sprint.

The overall purposes with this sprint are to design and code the game information page and the different game lists and to integrate rating function.

### 6.1.2.3 RESULT

After the third sprint the password has been encrypted so that it cannot be seen in the database. The user information is now saved with PHP SESSION as long the user has to log in. That means the system will remember a user's when he is logged in, so that he should not log in again every time he navigates away from the profile page.

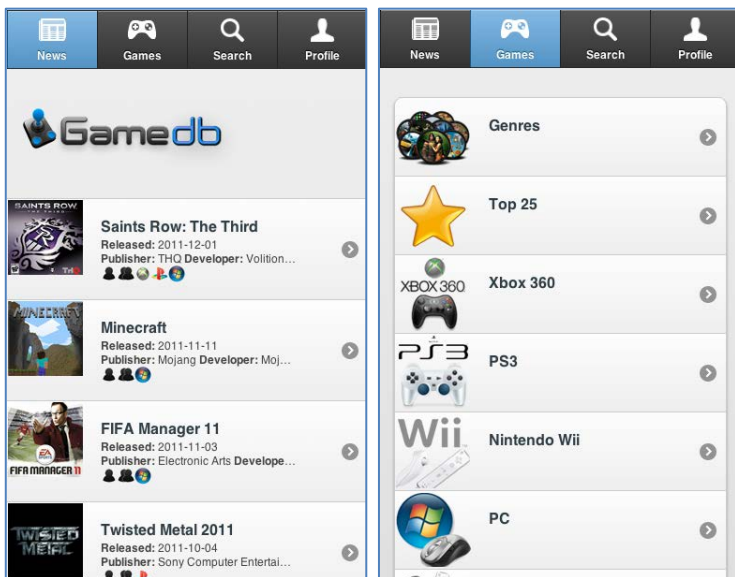


Figure 44

Figure 45

The front page is seen on figure 44. This is the first page the user meets when he visits the website. It lists the newest games. A thumbnail of the game cover, Game title, release date, publisher and developer name and icons that indicate whether the game supports multi and single playability is shown on every list item.

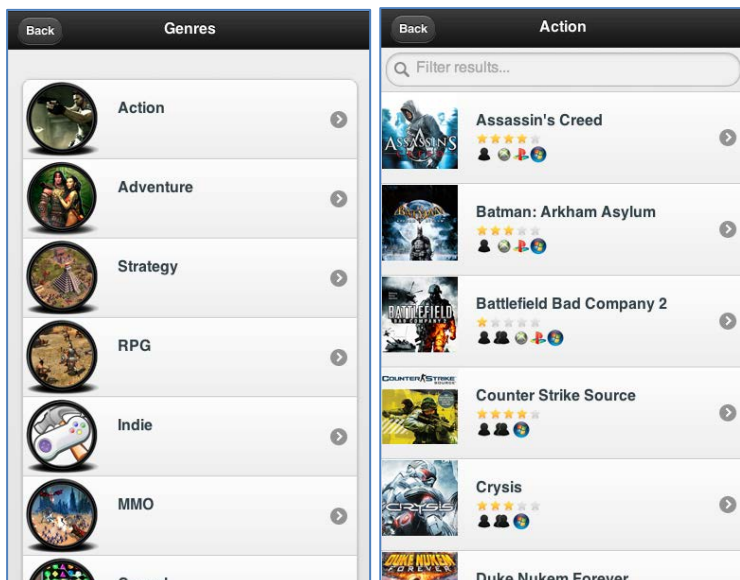


Figure 46

Figure 47

The game page is shown in figure 45 and it contains a list of submenu items, where the user can select how the game list should be sorted. If he selects "Genres" a list of genres will appear (Figure 46). If the user chooses "Action", a list with action games (Figure 47) will appear. In every list item a thumbnail of the game cover, game title, multi/single player icons and rating stars that indicate how many stars the game has in average. On the top there is a filter bar where it is possible to filter the result by typing in the search field. The other lists e.g. "Top 25" look similar.

The reason for that the game list on the "News" page does not contain rating stars is that the newly released and arrived games often have none or few ratings.

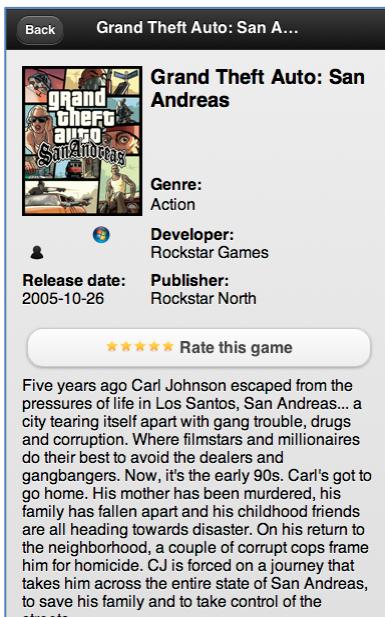


Figure 49

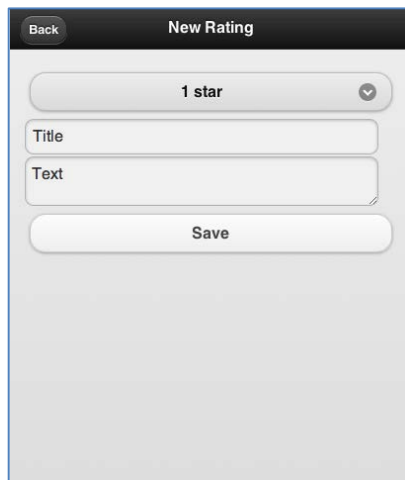


Figure 48

The single game information page is seen on Figure 48 and it appears when the user chooses a game from a list. In the bottom the user rating and short reviews are listed, sorted by date.

The user can rate a game by clicking on “Rate this game”, and a rating form will appear (Figure 49). When he clicks save he will be sent to the game information page where he came from and the rating lists at the bottom will be updated so that his recently added rating will be the first in the list.

After this sprint the profile page is updated so the “My Rating” button works. It sent the user to a page (Figure 50) where he can view, edit and delete his ratings.

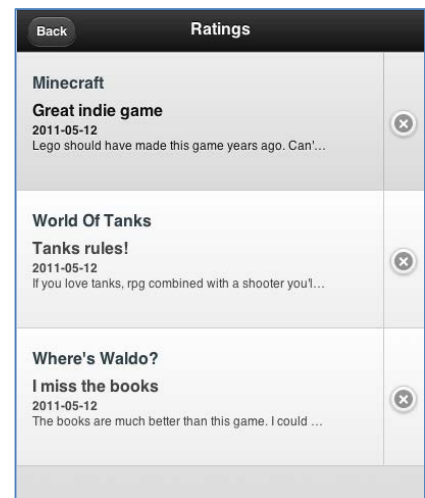


Figure 50

### 6.1.3 SPRINT 4

Sprint Backlog for the fourth sprint:

| From product backlog            | Key   | Summary   | Status at the start | Status at the end | Prioritization |
|---------------------------------|---|---|---------------------|-------------------|----------------|
| <b>Game information display</b> | Icons   | Design icons to genres, submenus in games page and game information page            | IN PROGRESS         | DONE              | 1              |
| <b>Search</b>                   | Search function                                 | Constructing the search function.   | TO DO               | DONE              | 2              |
|                                 | Search message                                  | Make sure that the user gets a small message if no results were found.              | TO DO               | DONE              | 3              |
|                                 | Search results                                  | Get a list displayed of games if search was successful.                             | TO DO               | DONE              | 2              |
| <b>Overall</b>                  | Back-button issue                               | Fix missing back-button issue.  | TO DO               | DONE              | 4              |
|                                 | Link issue                                      | Fix the link issue where the address bar keeps getting bigger which gives an error. | TO DO               | DONE              | 5              |
|                                 | Redirect from log out                           | Redirect the user to the front page when logged out.                                | TO DO               | DONE              | 6              |
|                                 | Redirect from rating                            | Redirect to specific place when a rating has been submitted.                        | TO DO               | DONE              | 6              |
|                                 | Optimization                                    | Optimize the code to make it better and more logical.                               | TO DO               | DONE              | 8              |
|                                 | Error message                                   | If the user types a page that does not exist they will see an error                 | TO DO               | DONE              | 7              |
| <b>Registration</b>             | Remove back button when registered and redirect | Removing the back button when a user has registered and redirect to the login page. | TO DO               | DONE              | 6              |

Table 8

The tasks for the fourth iteration can be seen on the previously mentioned figure 40 and as mentioned earlier all the tasks originate from the product backlog. One of the major tasks in this iteration was to get a search function up and running, it should be possible for the user to search with a keyword and the user should be offered advanced search options. The other major task in this iteration was to look at the navigation aspects of the system, e.g. when the user has finished rating a game, where should the system

then direct him. Regarding the navigation the group would also look on navigation errors that should be corrected. All the major tasks have all been made into smaller manageable tasks which are all in the backlog for this sprint.

### 6.1.3.1 RESULT

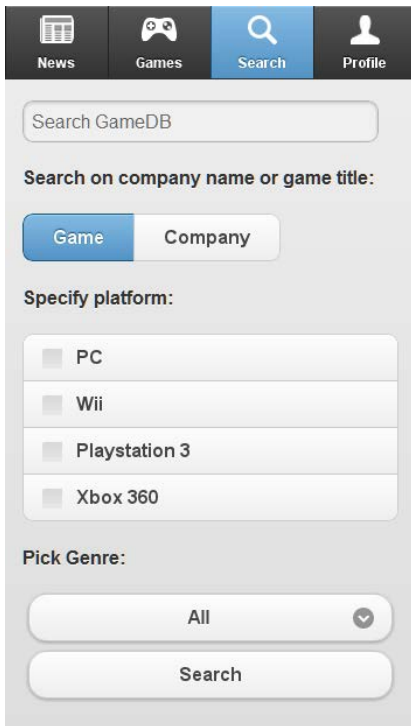


Figure 51

On figure 51 the reader can see an image of the final result of the search page as mentioned in previous sprints it has the header bar that will only be shown on the main pages of the website. The search page has a bit more advanced functions than initially planned. It is possible for the user to pick a genre to search by and the user can search by company name instead of game title. When the user types in a search keyword he will be met by one of the following figure 52 or 53.

If the search was successful the user will be presented with a list of games (figure 52) much like the list as described in the third sprint (in this specific search the keyword was "red"). If the search was unsuccessful the user will be met by the window on figure 53 and he now has to go back and change some information and try again.

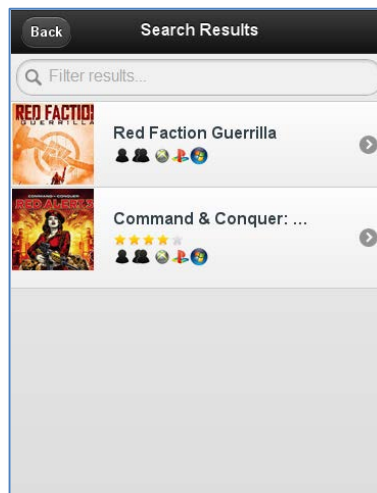


Figure 53

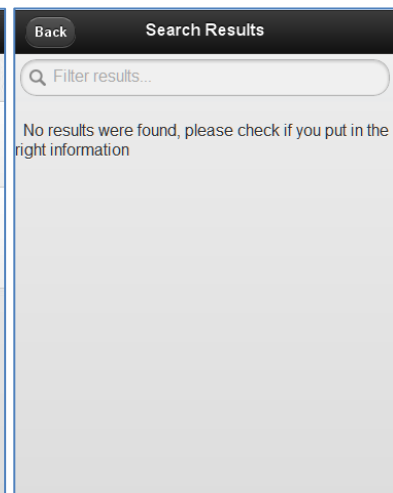


Figure 52

Regarding the navigational aspects of the website some minor changes was made from what the group had initially planned. At first it was planned that the user should be redirected automatically from time to time, but this idea was abandoned.

Throughout the website there were also some navigational errors that had to be fixed, e.g. some of the redirection functions were incorrect. If the user weren't logged in and wanted to create a rating he would be redirected to the login page but not back to the correct page. All these errors were corrected by e.g. making manual back buttons and make sure that they would always redirect the user correctly. This was ensured by going through scenarios on the web-application to cover every navigational aspect of the website.

After this sprint the system is ready to get a final test.

## 7 TEST

This chapter will cover the test of the system which contains both a black-box test and a heuristic evaluation. Both these test will be described in detail and how they are done. Besides the theory and practical parts, this chapter also shows the results from the tests. The results will be discussed and why they work or is considered a problem. At the final part of this chapter the further development will be covered. The section will be based on what was learned from the tests.

Throughout the development of the system, the database has been tested to make sure that that it performs as intended. The functions that have been tested are whether it is possible to add and delete data to the tables, that it searches correctly and have the correct data attached etc. The group did also test if the right information got written on the pages where needed.

Users can access the web-application using <http://gamedb.xtreemhost.com> on any device which has internet access. The UI is designed for mobile devices, but can also be accessed through a PC.

### 7.1 HEURISTIC EVALUATION

A heuristic evaluation (Nielsen, Heuristic Evaluation, 2005) is a quick, cheap and easy method for testing a user interface design. It is the most popular usability inspection method. This is done by using a systematic inspection of usability in an interface design. It involves having a small set of evaluators that examines the interface and judges its compliance with the usability principles, which is also called the heuristics.

There are ten general principles for user interface design (Nielsen, Ten Usability Heuristics, 2005). The reason why they are called heuristics is because they are more rules of thumb than specific usability guidelines. The principles are as follows:

- The system has to inform the user at every time so that the user knows what is going on. This can be done by appropriate feedbacks within reasonable time.
- The system has to speak the users' language. This means that the system has to use words, phrases and concepts that the user is familiar with instead of using system-oriented terms. It is a good idea to follow real-world conventions which makes the information appear natural and logical. The system has to make sure that the user always understands what to do in different situations so that there can be no misunderstandings.
- Users often choose system functions by mistake which means that the system has to provide some support to the user. The user shouldn't have to go through a long dialogue to go back. The system should also support with an undo and redo function.
- It is important that the system uses consistency and standards. The users should not wonder if different words, actions or situations mean the same thing.
- The best thing would be to make sure that an error does not occur. If that is not possible the system has to inform the user with good error messages so that the user knows what has gone wrong. It is also a good idea to give the user a confirmation option before the action is committed.
- The user's memory load should be minimized as much as possible by making objects, actions and options visible. It is important that the user should not have to remember information from one

part to another part. The users should always have an introduction option so they can get help if needed.

- The system should be flexible and efficient to use. This means that the system should have accelerators that is unseen by the novice user but speeds up the process for the expert user. The system can be used by both experienced and inexperienced users.
- A good design of the system is to be aesthetic and minimalistic. It is important that dialogues do not contain information that is irrelevant or rarely needed.
- It is important that the system helps the user to recognize, diagnose, and recover for errors. This means that error messages should be expressed through plain language and not codes that indicates the problem precisely and shows a suggested solution.
- The best system is if the system can be used without documentation. It can be necessary to provide the user with help and documentation which should be easy to search through.

It is difficult to make a heuristic evaluation (Nielsen, How to Conduct a Heuristic Evaluation, 2005) using a single individual because one person will never be able to find all the usability problems in an interface. This is the reason why these test should use more people because different people find different usability problems. Another reason is that the method's effectiveness gets improved significant by involving multiple evaluators. With multiple evaluators it is possible to determine the size of a problem. It is recommended to use between three and five evaluators since the developers do not gain that much additional information by using a larger number.

The evaluation is done by having each evaluator inspect the system alone. Every single evaluator is not allowed to communicate until every evaluation has been completed. The reason for this is that the evaluators can affect the result by telling what problems they found through there inspection. The results of the evaluation can be recorded by written reports for each of the evaluators or by having an observer to get all the verbalize comments through the inspection.

The advantages of written reports is that it presents a formal record of the evaluation, but requires a big effort by the evaluators which has to be read and aggregated by an evaluation manager. If an observer is used, it will give an overview of each session and reduces the workload on the evaluators. The result of the evaluation are also available soon after the last session since the observer only has to organize and understand the personal notes and not understand reports written by others. The observer does also have the ability to assist the evaluators in operating the interface in case of problems and further explanation of the prototype.

In the situation where users are involved in the evaluation, the observer has the responsibility of interpreting the actions of the user in order to relate these to the usability issues in the interface design. This makes it possible to do the testing even if the users do not know everything about user interface design. In contrast, the responsibility for analysing the user interface lays with the evaluator. This means that the observer only has to take notes from the evaluators actions and not the reactions.

There are two further differences between heuristic evaluation and traditional user testing which are the willingness of the observer to answer questions from the evaluators during the session and the extent to provide the evaluators with hints on how to use the interface. In a traditional user testing the developers



wants to discover the mistakes the users make when using the system. This means that the observer does not provide with more help than absolutely necessary. Furthermore the users are requested to discover the answers to their own questions by using the system instead of having them answered by the observer.

In a heuristic evaluation it would be unreasonable to refuse to answer the evaluators' question about the system, especially if the nondomain experts are the evaluators for the sessions. By answering the evaluators' questions will enable them to better assess the usability with respect to the characteristics of the domain. The evaluators can also be given hints if problem should occur during the session in order to not waste precious evaluation time by struggling with the mechanics of the system. It is important to notice that the evaluators should not be given any help until they are in trouble and have commented on the usability problem.

Every session in a heuristic evaluation will typically last one or two hours depending of the size of the system. Larger sessions might be necessary if it is a complicated and large system. It is a good idea to split the evaluation up into several smaller session where every of these session concentrates on a part of the interface.

While doing these sessions, the evaluator goes through the interface several times and inspects the elements comparing to the previous mentioned heuristics. The evaluators are allowed to consider any additional usability principle or result that comes to mind that can help the developers. It is also possible to develop category-specific heuristics that apply to a specific class of products as a supplement. One way to do this is to perform a competitive analysis and user testing to help explain the usability problems that were found. The evaluators decide for themselves how they want to proceed with the evaluation. It is recommended that they go through the interface at least twice. The first pass would be to get a feel for the interaction while the second pass allows the evaluator to focus on specific interface element while knowing how it fit into the larger whole. As a supplement to the evaluators they can be given usage scenarios so they can walk through the system as a user would to solve different tasks.

The output for using the heuristic evaluation method is to get a list of usability problems with references to the usability principles in the opinion of each of the evaluators. It is not possible for the evaluators to simply say that they dislike something. They have to explain why they don't like it and be very specific about it. Every usability problem should be listed separately. If three things are wrong with a certain dialogue, all of the problems should be listed with references to the heuristics which explains why it is a problem. The reason for doing this is that there is a risk of repeating some problematic aspects, even if it were replaced with a new design, unless the developers are aware of the problem. It may not be possible to fix all the problems, but it helps the developers to fix some of them if they are known.

This evaluation does not provide a systematic way to generate fixes or assess any redesign for the problems found doing the session. It gives the explanation for the usability problems which makes it easier to generate a new design according to the guidelines provided for a good interactive system. Many problems have an obvious fix as soon they have been identified.

It is possible to extend the heuristic evaluation with some design advice which can be used in a debriefing session after the last evaluation session. This session should include all that has been involved during the

evaluations and the developers. The debriefing will be a brainstorming with focus on the discussions of redesigns to address the major usability problems. It is also a good idea to discuss the things that work and why, since the heuristic method does not address this important issue.

### 7.1.1 THE HEURISTIC EVALUATION

The test in this project will be performed by three evaluators, each a member of the group. The test will be using wireless internet access on the university to connect to the system and will be tested on a mobile device. The evaluators' assignment is to cycle through the system twice and compares every function and page with the heuristics. A system does not need to have all the mentioned things that the checklist contains, e.g. if a system does not contain a buying option you should not test if that works.

The heuristics sheet, by which the test has been carried out, is enclosed as appendix 6, which also contains the results of the test.

If a problem is found, the evaluator writes it down with the reason why it was deemed a problem. No observer will be used because the group knows how the system works and does not need to have an extra person to take notes of the test.

When the test has been completed, a schema (Pierotti) will be made to show all the problems which can be seen in appendix 6. Afterwards an evaluation meeting will be established to walk through all of these problems to discuss why it is a problem and find the best way to solve it. Beside the problems the team will also discuss the good things about the system.

It is expected to find some design errors throughout the program that the developers wouldn't have noticed while constructing the system. Another expectation is to learn why there is a usability problem and how it should be fixed. It is important that every problem is found and discussed throughout the whole evaluation.

## 7.1.2 RESULTS

In this chapter the results of the tests will be described.

In general the dialogs have been found very neutral and informing in the situations where needed. The system tells the user what has gone wrong without accusing the user. The system does also immediately tell the user that it is loading when the user wants to enter a new page. Each page had a title in the header that showed the user what page that has been entered. The menu bar shows on which page the user is currently at by using colour distinction.

The system uses consistency and reuse function types to solve different tasks. The navigation of the system does always give the user the opportunity to go back.

There have been found some elements that could be improved but does not necessarily mean that it is an error. One of these situations where found in the setting and edit password form. The system tells the user to write a password and then verify the password. If the user is not familiar with this term, he would not know what to do. To help the user, a small description could be added telling the user to type the password again.

When the user enters the rating page, the header says "New rating" but does not tell which game is getting rated.

Another situation is found when the user wants to rate a game and is not logged in to his own profile. When the user presses the rate button the system redirects the user without telling the user why. The system could give the user feedback that saying: "You have to be logged in to rate a game".

On the "Register" page each field is not granted with any direct feedback when filled in. A user-friendly register form live-checks for errors whilst the user is typing. It then shows a little icon that tells the user whether or not it is correct and will be accepted. It helps to spare time going back and forth changing data if something has gone wrong. Another feedback example is found when the user adds a rating and gets redirected to the game page. The system could give the user a message telling that the rating has been added, instead of the user having to find out himself.

When entering the "My ratings" page, every rating contains a small "x" icon. There is no description on what it does. When the button is clicked, the user gets redirected to a dialog page where he will be asked if he is sure he wants to delete the specific rating. It is not clear to the user that it is possible to edit the rating by clicking on them. The page could either contain a button saying, "Edit" or have a text on the page, explaining the layout.

When the user is navigating through the system an indication of the previous visited pages could be useful. The only navigation is the main menu and the back buttons.

Throughout the whole system there are no help option provided.

The sorting functions of the "Top 25" list needs to be corrected in order to give the most relevant output. This could be done by accounting for attributes like date and number of ratings given.

There has also been found two errors which are both found in the URL. The first problem is that the address keeps getting longer when the user goes back and forth or uses the search function. This will result in the system not working as intended. The second problem is that the user has the ability to change other users' rating by changing the "ID" in the address bar.

### 7.1.3 FURTHER DEVELOPMENT

From the results of the test, listed above, several items can be improved, by which some of them are listed below.

- Communication from system to end user must be improved, in order to secure satisfactory use of the database. This includes both error-handling and informative communication. The test showed that the system lacked these elements several places in the system.
- The security of the database needs improvement. It was discovered that there was a security breach in the URL setup, which made it possible to log in as another user and editing/rating games. This could have catastrophic consequences if need fixed prior to a launch.
- The UI needs further testing to conclude whether or not it is logical. This is due to the fact that the test was performed by the group itself, and therefore the UI naturally seemed logical to them. The test showed that many of the visual characteristics needed in a good UI were present.
- There was only one possible system crash errors found in the test, which was the URL problem, on which it kept extending the URL, instead of using the short URL. This will, by continuous use, result in redirection error and system failure.
- Several functions need to be implemented in order to ensure a satisfying user experience, such as a help function, by which the user can get his questions answered. The test focused a lot on "Q/A" sections, which is not implemented at the current state of the system, but this could also be a good idea to implement in the system.
- Minor error correction is also needed. The "error" header appears when loading a new page, even though there are no errors. But to ensure satisfactory use this needs to be corrected as well.
- The top 25 list needs to be optimized in order to ensure the most relevant output accounting for popularity, rating and date.

## 7.2 BLACK BOX TEST

In a black box test all inputs and outputs is tested to make sure that it work as it should. The idea of a black box test is to consider the system as a black-box where valid and invalid data is entered to check what the corresponding output will be.

A black box test is a basic testing philosophy where it is needed to determine on what level it is used. The chosen method is to do the test based on the requirements to ensure that they are fulfilled. The test uses input and controls the output, making sure it works as intended. The system will be tested on the same level as the user. The test will be divided into different areas of the system.

It helps determine errors and to see if data gets transferred correctly. The test has been done throughout the development of the system and some scenarios has been done more than once to make sure it worked as it should. The following examples are not shown in any particular order.

## 7.2.1 EXAMPLES OF AN ERROR

The following will be a description of some errors found through the black-box test.

### 7.2.1.1 PASSWORD SETTING ERROR

While constructing the password system the developers found a bunch of errors. The system didn't check if the old password were correct which meant that the password would be changed whether the user made an error or not. This means that the system did not come up with the errors messages that were expected.

The system did also have an error while checking if the new password and the re-typing of the password were identical. It turned out that it did not check for this and only changed the password according to the re-typed one. This was not intended because the system should make sure that the password could only change when these are identical.

## 7.2.2 SEARCH FUNCTION

Before constructing this, it was suspected to be one of the more complicating functions to implement, due to the many ways it can be constructed in. It was critical to choose the right way to search since it is critical to the user experience. Some games have used special signs in the title such as "Assassin's Creed", which use an apostrophe. There were also several issues regarding how the search function should work in regard to searching for parts of the title, the whole title or the start of the title. This had the consequence that much testing has been done before a tolerable solution was found.

## 7.2.3 LOGIN DIFFICULTIES

Several minor setbacks occurred when operating with the login or session part of the system. The idea of the login function is that the user can rate games, for others to use. Some coding setbacks arose, so that the user could make these ratings without logging in, which took some time to correct. Furthermore there were some redirecting issues regarding the login feature. When pressing the "back" button, the session ended, and later on there were issues regarding the log out function as well. They were, after several attempts, corrected using iterative trial-and-error testing.

## 8 REFLECTION

This chapter will cover some interesting reflections the group has made throughout this report and project.

All scenarios are estimations and thereby fictional.

### 8.1 APPROACH

The approach used to construct the system, as well as this report, is the agile approach to software engineering. The whole construction phase is documented in the log, using the SCRUM model. It would have been interesting to see the consequence if a traditional approach was used instead, like the Waterfall model. It is hard to say which direct and indirect consequences it would have had if the other model had been used, but some consequences could have been:

- Planning and structuring would have been very different. A long term plan would have been established instead, and there would have been a lot more focus on the analysing part. Instead, using the SCRUM model, only the basics of the system has been analysed and the rest has been built along with the construction process.
- Workwise it would also have been different. In the waterfall model all the participants of the group have the same tasks at a given time in the process. This is because each member of the group works together with each phase of the process, right from early analysing to testing and delivery. This counteracts flexibility and hinders specialization, but helps in securing that everyone in the group has tried to work with all work necessary to construct the database and report.
- The database itself may have been altered in some way, due to what would have been planned from the early start. This could mean other-, less- or more functions compared to what has been realized using the SCRUM methods, where just the basic functions were planned and other functions constructed using the experience gathered along the construction.
- The user-interface of the database could also have been altered. Using the SCRUM, the interface has been inspired by what is already available and commonly used on the internet. If the group had used the waterfall model, perhaps a different design had been chosen in order to accommodate the expected functions of the database.
- The reason of choosing an agile approach was primarily due to the many uncertainties surrounding the whole planning and construction phase. Neither member of the group had previous experience on construction databases and were only slightly experienced in creating computer systems. This made it complex to estimate many things concerning the build of the database. It would have been interesting though, to see how these obstacles would have been managed using the Waterfall model.

## 8.2 ANOTHER AGILE APPROACH

This report and database has been constructed using primarily a SCRUM method. Agile developing has some other well renowned models such as the XP-model or the agile modelling method. SCRUM was chosen due to the team based situation the group was in, as well as the previous mentioned uncertainties regarding the development process. The SCRUM model allowed the use of specialization and smaller teams within the group, which seemed to work fine during the construction phase. However, another agile approach might have been more beneficial to help some of the challenges which rose during the process. Perhaps the XP-model would have helped with a better working system as well as given a better end result if used instead or together with the SCRUM model, although the XP-model uses a lot in user-interaction along the construction process, in which there are virtual none in this process. This is due to the fact, that the users of the system are the group itself.

## 8.3 FUNCTIONALITY

The functionality of the system itself were debated amongst the group members, before a basic set of functions were determined. These functions were primarily chosen in order to set up a basic user interface with the most common functions available, such as displaying the newest games, the opportunity to search for games and make a rating. Most of the data output are derived from a mathematical point of view, meaning that e.g. the rating system is based upon a mathematical deduction from user ratings. If the functions were determined from a social point of view instead, the functionality would have been different. Functions like posting a reply to a rating should have been constructed and another way of showing news would have been made. There could also be some control functions, which e.g. checks whether or not a given has been released yet, before letting the user rate it.

## 8.4 CONSTRUCTION

There were several ways that the database itself could have been constructed in ways of determining which tables to use, attributes to include, schemas constructed and so on. Alternating these in some way, would have consequences for the system, but are considered of minor importance in this report and system. This is due to the facts that it is no long term plan of releasing the database, making flexibility almost irrelevant from post the delivery date.

## 8.5 TESTING

Throughout the whole construction process, testing has been made by continuously trial-and-error programming. After the system was constructed a detailed heuristic- and black box test were used. It would have had a major impact if the system was not tested along the process since it could be hard to correct the code as a whole, when not having a full overview of which functions impacts others in the system. The chosen final test, the heuristic, was done by the developers (the group) themselves. It would certainly have given other results if performed by use of external testers, which had no relation to the development process. Other forms of testing would also have provided other results, but the heuristic was deemed most suitable due to the level of detail included in the test.

## 8.6 USE OF USER DATA

At this time, the user data such as age not used other than to give the user a sense of “commitment” to the system in order to make more serious ratings. Data such as age could be used to sort which games should be shown to the user, when making a search. This could present the most relevant games for his or hers specific age group, or sort out games for which the user is too young to play due to age-restrictions.

## 8.7 USER INNOVATION

It would have been interesting to construct the database by innovation from the end users point of view. User innovation can be a powerful tool, and can be better suited to constructing a system which the user finds interesting and useful. This could have had a major impact on such things as the prioritizing table, if constructed from what the users desired. The subject for the database, video games, may have been altered or perhaps the platform for accessing the database would have been changed. Instead, the whole system was designed from the group’s point of view.



## 9 DISCUSSION

In this chapter some of the most important choices the group has made while constructing the system and report will be discussed. The discussions that will be covered are the choice of software engineering method, the structure of the database and overall goals for the project. The discussion part will mainly use the SOE Mini-project 2, in which the group critiques the choice of development method.

### 9.1 USING THE AGILE METHOD

After doing some research, the group decided to use the SCRUM model as primary development method. It seemed well suited due to the many uncertainties regarding the project and the lack of experience among the group members in creating a database.

At the beginning, the group had difficulties of shedding the traditional development methods, since it had been used the prior two semesters. It was hard not to plan ahead making in-depth analysis of the whole program, but instead just using basic analysis and progressing step-by-step from this point. The basic analysis was mainly the product backlog and the ER-diagram.

At the positive side, programming was started almost immediately and experience was quickly gathered using trial-and-error style of programming. Although the sprints at first were delayed, due to difficulties of estimating the time consumption for each sprint, many small tasks were quickly completed. The daily SCRUM meetings were another success due to the fact that each group member knew the current status of the sprints and tasks needed to be completed. The trial-and-error style of programming caused delays to the sprints due to needed error correction, which at first, was not planned in the sprints. Later on, the time estimation for the individual sprints was prolonged in order to plan ahead for needed error search and correction. Using the sprints was also well used during the vacation period of Easter, where each member had their own tasks to complete.

The SCRUM method also had some negative consequences for the project development. It was rather quickly established that working individually with the sprints caused working difficulties and delays, since there was no knowledge sharing, a tool which is most helpful when working with new systems. Each group member has unique qualities which were polarized in the early development process where some members only coded and other members only wrote documentation. This caused misunderstandings and difficulties when arranging further sprints. This polarization also caused periods where the documentation writing part of the group only had minor sprints to complete, due to need of a coding sprint before continuing to the next level in the project.

Furthermore, it was hard to have an overview of the system as a whole, because the group only had the product backlog to suffice where in the progress the system and report was.

The overcome these difficulties, the second mini-project of SOE helped, using a SWOT analysis of the group.

## 9.2 SWOT ANALYSES

|  |  |
|--|--|
| <b>Strengths</b><br>Flexible.<br>Working individually and in groups.<br>Each member's skills.<br>Social activities.                                    | <b>Weaknesses</b><br>Estimation and planning skills.<br>Allocation of roles.<br>Discipline.                      |
| <b>Opportunities</b><br>Programs that help the group build the product.<br>Getting help online.<br>Getting help from other teams.<br>Getting equipment | <b>Threats</b><br>IT problems.<br>Webhost problems.<br>University Administration.<br>Time consuming obligations. |

Table 9

In the SWOT analysis the team wrote down what they believed was their strengths and weaknesses of the group. The team wrote down what they had to do in order to try and make their weaknesses into strengths. This was to try and work more in teams instead of individually. This way the team members could more easily take advantage of each other's strengths and perhaps try and make weaknesses to strengths.

Working in small teams of two or three members resulted in more minor errors were discovered faster. Generally the group thinks the teamwork approach is more effective than working individually.

### 9.3 CONSEQUENCES FOR FUTURE DEVELOPMENT PRACTICE AND ENGINEERING PRACTICES TO BE IMPROVED

During this project the group experienced that the best thing was to work in smaller groups where it is possible to discuss the different tasks and decisions. It is also important to use the different core competences where needed which makes it both easier and saves time. It has been found that the SCRUM method has been a helpful tool to help organize the project. The big strength of scrum is the daily meetings where all the members say how far he/she is with the tasks. If a problem comes up the group discusses the best solution.

In this semester all group members has experienced how it is to be a SCRUM Master which was very educational for everyone. It also means that there hasn't been a single scrum master who had the overview which could have been helpful in some situations.

A thing that has to be improved for future projects is to think less traditional when using an agile approach. It is important that the group is prepared for changes and errors throughout the process. Another thing that has to be better is to split the group into smaller groups so that no one is sitting alone.

An alternative way for the future is to use the XP method that is different by being more program orientated than SCRUM that is known for its project management suitability. XP uses test-driven development, refactoring, pair programming, simple design etc. XP is more acceptable to changes during the iterations.

It could also be possible to combine these methods because it has both the project management and the program development that at all time will be ready to change.

## 10 CONCLUSION

After these experiences, analysis and results we can now try to answer our problem statement which was as follows:

*“How can we develop a web based database application, and which processes and challenges do we face when using an agile development method?”*

The development of the database has been an exciting journey using a lot of new tools to analyze, shape and design it. The tools taught in the course of SOE helped the progress by organizing and executing throughout the timespan in which the database has been constructed.

At first, it needed to be established which development method was most suitable for developing this database. The agile method was chosen due to its suitability with development of systems which have many uncertainties. Before this project, neither of the group members had experience in constructing databases and only minor experience in creating net based software. Therefore, the group was in need of a step-by-step method, which made use of task-dividing while creating room to work individually or in teams.

Although several agile methods seemed suitable for this development, the SCRUM method was chosen, since its use of product backlog and sprint backlog seemed appealing. Using these individual sprints as a way of learning how to create the system proved a challenge, because the group members quickly became polarized in task distribution. This resulted in some members almost only wrote documentation while others only focused on coding.

A meeting, together with a mini-project in SOE, helped with this and small teams were more often used in order to secure a more reasonable division of labour in the group, making sure that everyone knew the current status of both coding and writing.

Using the experience gathered throughout the process, time estimations became more stable and realistic as the project progressed. At first, there was almost no time scheduled for error handling, which, especially at first, took a long time to handle. This resulted in delays, but later on, the group adjusted this in sprint time estimation.

A traditional development method would have required a lot more analysis before construction of the database could have been built, which could have resulted in major delays since neither member of the group had experience in constructing the database. If time had to be scheduled for both major analysis and thereafter the learning process of coding databases, the result could be lack of time which would give a product of lesser quality.

Compared to a traditional development method, the agile proved well suitable for this project. Only rudimentary analysis was done, and programming commenced quickly after, which form the basis of knowhow upon which this system is built.

The database itself, has been constructed using this development method, and is currently running at <http://gamedb.xtreemhost.com> at the delivery date (27/05/2011).

Through testing, several flaws of minor importance were discovered, but it is the group's opinion that the system delivers reliable data, which means that the database part of the system is working. The focal point on constructing the database was that it had to deliver reliable data by use of correct tables, schemas and search functions. The flaws of the system are mainly on communication and design.

Using an agile approach is to adapt to the situation as it proceeds. When facing challenges, it is important to adapt right away, using previous obtained experience and strength to overcome it. No matter what system that needs to be constructed, the process is the same, which is to adapt and overcome.

## 11 BIBLIOGRAPHY

- Gestaltlovene*. (23. December 2010). Accessed 13. May 2011 at nielsgamborg.dk:  
<http://www.nielsgamborg.dk/?p=gestaltlovene>
- ESA. (2010). *Essential Facts about the computer and video game industry*. Accessed 13. May 2011 at ESA:  
[http://www.theesa.com/facts/pdfs/ESA\\_Essential\\_Facts\\_2010.PDF](http://www.theesa.com/facts/pdfs/ESA_Essential_Facts_2010.PDF)
- Korth, H., & Sudarshan, S. (u.d.). *Chapter 7: Entity-Relationship Model*. Accessed 19. May 2011 at Yale University: <http://codex.cs.yale.edu/avi/db-book/db6/slide-dir/PDF-dir/ch7.pdf>
- Nielsen, J. (2005). *Heuristic Evaluation*. Accessed 13. May 2011 at Uselt.com:  
<http://www.useit.com/papers/heuristic/>
- Nielsen, J. (2005). *How to Conduct a Heuristic Evaluation*. Accessed 13. May 2011 at Uselt.com:  
[http://www.useit.com/papers/heuristic/heuristic\\_evaluation.html](http://www.useit.com/papers/heuristic/heuristic_evaluation.html)
- Nielsen, J. (2005). *Ten Usability Heuristics*. Accessed 13. May 2011 at Uselt.com:  
[http://www.useit.com/papers/heuristic/heuristic\\_list.html](http://www.useit.com/papers/heuristic/heuristic_list.html)
- Pierotti, D. (u.d.). *Heuristic Evaluation - A System Checklist*. Accessed 24. May 2011 at Society for Technical Communication: <http://www.stcsig.org/usability/topics/articles/he-checklist.html>
- Roelsgaard, P. (7. Marts 2011). *'Angry Birds' flyver ind på Facebook*. Accessed 13. May 2011 at Ekstra Bladet: <http://ekstrabladet.dk/kup/elektronik/spil/article1514405.ece>
- Scrum (development)*. (u.d.). Accessed 13. May 2011 at Wikipedia:  
[http://en.wikipedia.org/wiki/Scrum\\_\(development\)](http://en.wikipedia.org/wiki/Scrum_(development))
- Skov, M. (Spring 2010). Design implementering og evaluering af brugergrænseflader.
- Takahashi, D. (25. May 2010). *Video game industry to hit \$70 billion by 2015, but growth will slow*. Accessed 13. May 2011 at GamesBeat: <http://venturebeat.com/2010/05/25/video-game-industry-to-hit-70-billion-by-2015-but-growth-will-slow/>

## 12 APPENDIX INDEX

|   |                        |
|---|------------------------|
| <b>APPENDIX 1: TEST TUTORIAL</b>                                      | <b>REPORT &amp; CD</b> |
| <b>APPENDIX 2: WEBSITE FILES (ONLY ON CD)</b>                         | <b>CD</b>              |
| <b>APPENDIX 3: SQL FILE (PDF - ONLY ON CD)</b>                        | <b>CD</b>              |
| <b>APPENDIX 4: SQL FILE (ONLY ON CD)</b>                              | <b>CD</b>              |
| <b>APPENDIX 5: ER-DIAGRAM EXPLAINED</b>                               | <b>REPORT &amp; CD</b> |
| <b>APPENDIX 6: HEURISTIC EVALUATION TEST AND RESULTS (ONLY ON CD)</b> | <b>CD</b>              |
| <b>APPENDIX 7: THE REPORT IN PDF FORMAT (ONLY ON CD)</b>              | <b>CD</b>              |

## APPENDIX 1: TEST TUTORIAL

Please remark, that the user-interface has been designed for mobile devices. The database can be accessed using a regular laptop/desktop computer but the user-interface will appear distorted. This can, in some way, be regulated by adapting the web-browsing window to a smaller size, fitting the user-interface.

In order to test the system, simply follow these steps:

1. Access <http://gamedb.xtreemhost.com> preferably by using a mobile device.
2. Log in by pressing “Profile” at the top navigation bar.
3. Type “AAU\_Test” as username (please remark that it is case sensitive and without quotation marks).
4. As password type in “123” – also without quotation marks.
5. You are now free to navigate the system as you please.



## APPENDIX 5: ER-DIAGRAM EXPLAINED

This appendix has been made to ensure that the reader understands the terminologies in the ER-Diagram.

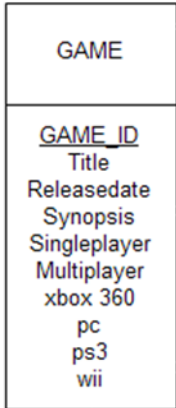


Figure 54

The figure 1 on the left is a visual presentation of an entity in the database, the box on the top is the name of the entity and in the box below you can see all the attributes in the entity. The attribute that is underlined is the primary key in the table.



Figure 55

The diamond shapes as seen on figure 2 describes relationship sets, the reason for it to be double lined will be described further down.



Figure 56

Figure 3 is an example of one of the many relations in the ER-Diagram. First of is this a many-to-one relation, which is described with the lines. In the example on the left the lines indicate that, a Rating is associated with at most one User via USER\_RATING and a User is associated with several (including 0) Ratings via USER\_RATING.

The double lines in the as seen on figure 3 is a total participation, it means that every entity in the entity set participates in at least one relationship in the relationship set. In the example above it means that every Rating must have an associated User.