DOKUZ EYLÜL UNIVERSITY

ENGINEERING FACULTY

DEPARTMENT OF COMPUTER ENGINEERING

CME 2210

Object Oriented Analysis & Design

Project Report

DIGITAL GAME MARKETING

By

Nazmican DİKMEN

Kasım GÖKMEN

Vedat ÖZCAN

27/05/2020

Table of Contents

INTRODUCTION

1.1 Purpose

The project is based on selling, renting and marketing video games digitally. Besides these main features, the project will also consist of including a basic social system and rating system. These features will be explained thoroughly in the upcoming titles.

1.2 Scope

The project aims to present a digital video game marketing for users to purchase, rent and do several other activities. A social friending system is also implemented using specific data structures. Users are now able to purchase their desired games, as well as rent them. This recommendation process is implemented by taking insight from the social system.

Besides digital marketing, the project aimed to sell physical copies of the games with a special shipping system. There was going to be subtle differences between digital and physical marketing, but these features will be explained why it was not completed in detail in the following chapters.

1.3 References

Digital Game Marketing project has taken insight from a variety of popular programs that lots of users use in a daily basis. Project team desired to make a project that works just as fine as the other popular programs but in a better and friendlier way. While aiming these goals, it's safe to say that the project resembles small parts of the mentioned popular programs which is quite liked by the users.

1.4 Overview

The project crew cares about the importance of back-end coding. Hence the crew was concentrating on the background to provide better and more efficient program. However, it doesn't necessarily mean that the front-end was neglected. The project crew was confident to be able to ensure an appealing, yet basic; elaborate, yet pure user interface.

Analysis & Design

2.1 Functional Requirements

2.1.1 The Social System

As it is mentioned before, social system is implemented to the project for several reasons to enhance immersion and credibility. Users will be able to add and remove other users as friends however they wish. This friending system is also helpful when it comes to searching for games. Users will be able to see recommended games which are the purchased or rented games of the users friends. Thus users will have an insight of the games which they would like to purchase.

The social system also has an Interface which contains specifically designed functions in which you can add, remove or send a request. It is also possible to display friendslist as well as the pending requests.

2.1.2 Market & Library

After a brief design, this part will be explained in one title, while both are different parts of the project. Although they're different parts, their goals and purposes are somewhat the same. This part is the main database of the project. Every product will be stored in this section.

- o Market:

  The market section is self-explanatory. Every product in this section is presented to users. In this section, users are going to be able to purchase, rent, review and sort every game by their preferences. The market class also has its own interface like the social system, which is also designed specifically for its own usages.

  Along with the database of the market, the functionality of the market carries a great importance as well. As mentioned above, an elaborate interface requires functionality which makes it easier for users to use to program.

- o Library:

  The library section acts like the market but it's exclusive to the user. It contains games that the user have purchased or rented. It uses the same types of attributes that the market uses but the functionality differs a bit from the market. While the function design remains the same, the contents change from the market's functions.

2.2 Non-Functional Requirements

2.2.1 The User Interface

The project crew always tried to base the project around the user. The user interface is the most important part of the project thus it requires the most effort and thinking. There is several features and attributes which enhances the user interface.

o **Sign-in/Sign-up System:**

The first and the foremost feature of the program is naturally the login system. Several attributes are held in a user class to ensure a safe login and register.

o **Social Interactions:**

It is believed amidst the crew that the social interaction system is a neat feature for the project. Under the user interface, the program holds publicly open information to be able to operate social interactions.

o **Game Library:**

One of the most crucial and basic part of the project. This is a digital game marketing project and every user needs a library to store their purchased or rented games.

2.2.2 Games & Products

The digital game marketing system can not sell anything if there is nothing to sell basically. While the games and products part of the project is quite self-explanatory, it is an important part that needs to be pointed out.
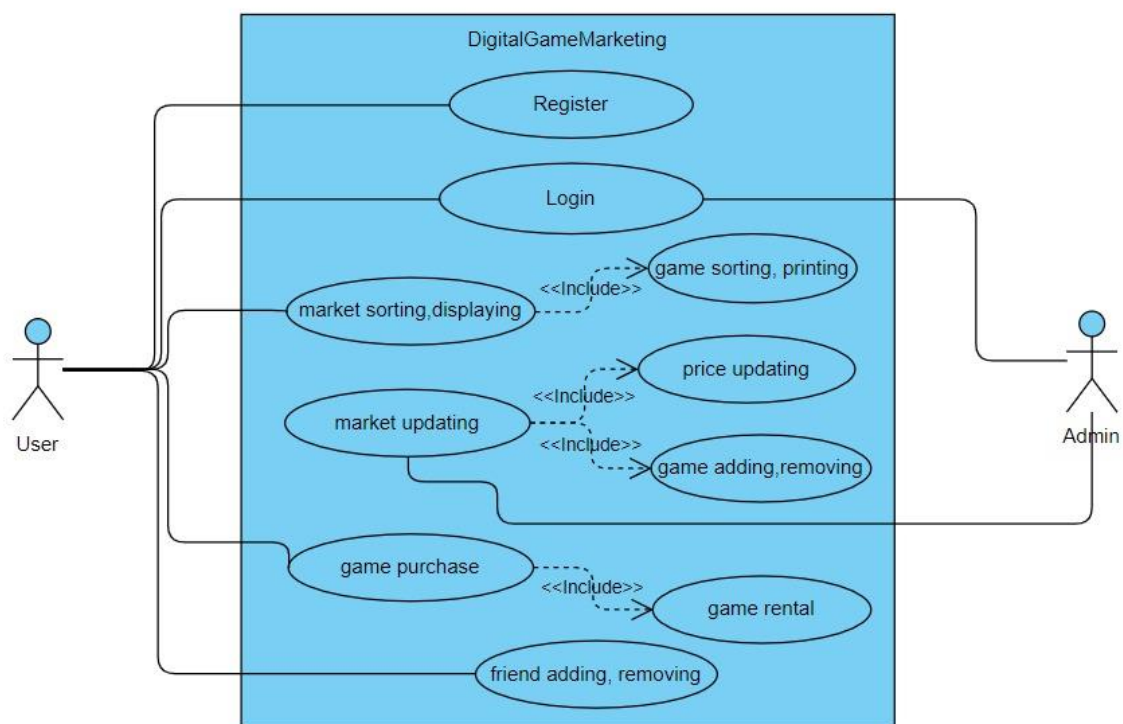
2.3 Use Case

Diagram 1. : Use Case

User registers to the system initially, admin does not require any registration. Only after registration, login is possible for the user. The user can display, sort market and purchase or rent games. User can also add and remove friends. On the other hand, Admin has the permission to update prices, add game stocks to the market along.
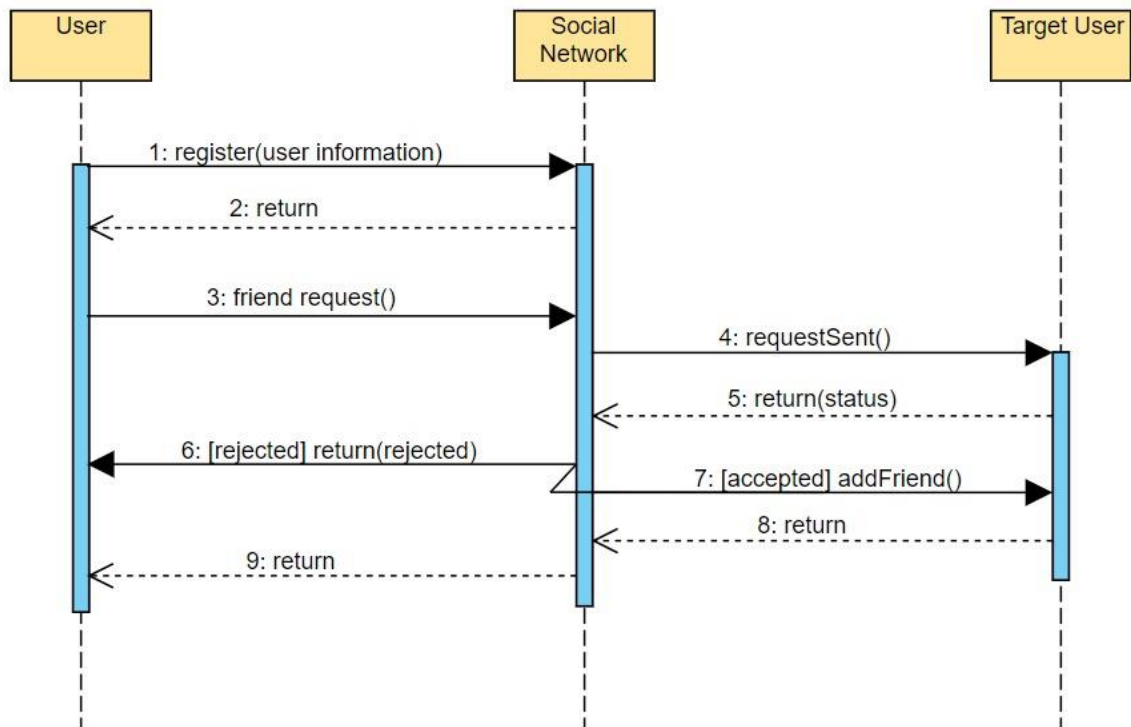
2.4 Models

2.4.1 Sequence Diagram

Diagram 2. : Sequence Social Networking

This sequence diagram shows the process of the Social Network System. After a successful registration, the user also automatically signs up to the Social Network. The diagram only presents a "friending" action. A friend request is processed through the Social Network System (SNS), directed to the target user. After a response from the target user, it returns the status of the response. Based on the response, the pending request will be handled.
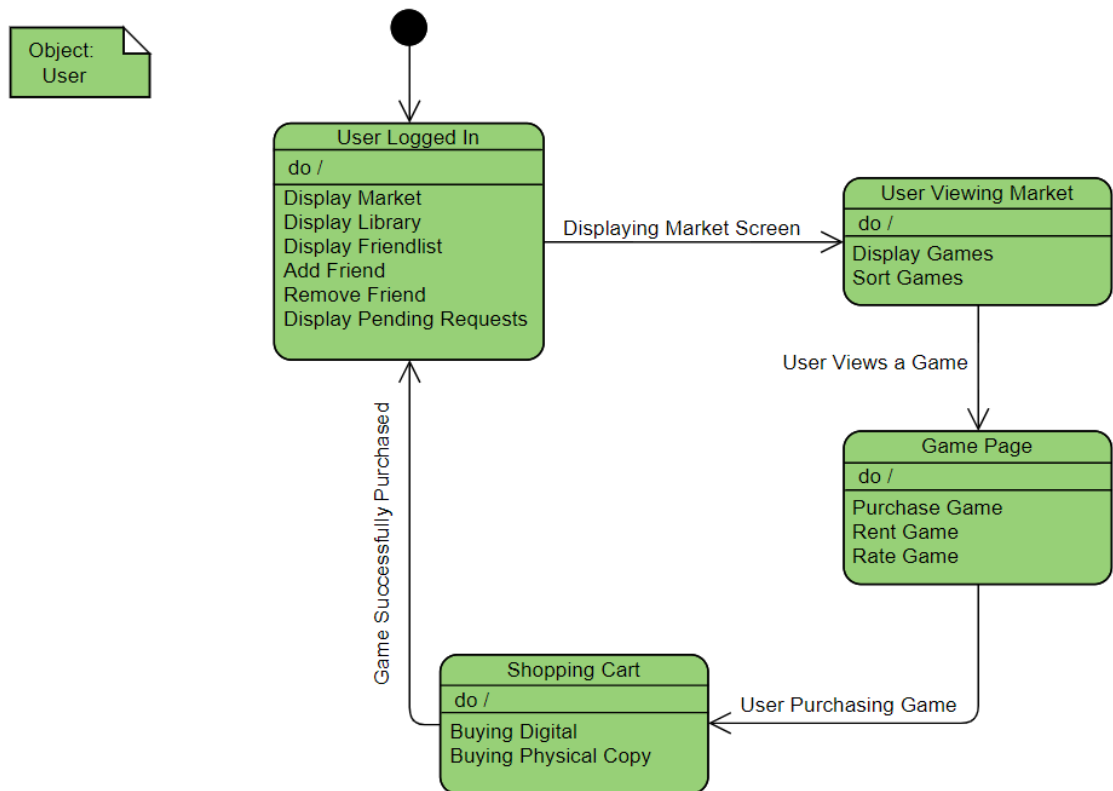
2.4.2 State Diagram

Diagram 3. : Purchasing a Game, State Chart

The diagram shows a successful purchase process. After a log in, in order to purchase a game, the user should display the market page first. After that the user can either display the entire market, sort the market or search for a game specifically. Once the game has found, the user can purchase or rent the game but can not rate unless it is already bought. If the user choses to purchase the game, it can be done in 2 ways: buying a digital key or a physical copy. If the game is successfully purchased and the transactions are made, the user will return to the main menu.
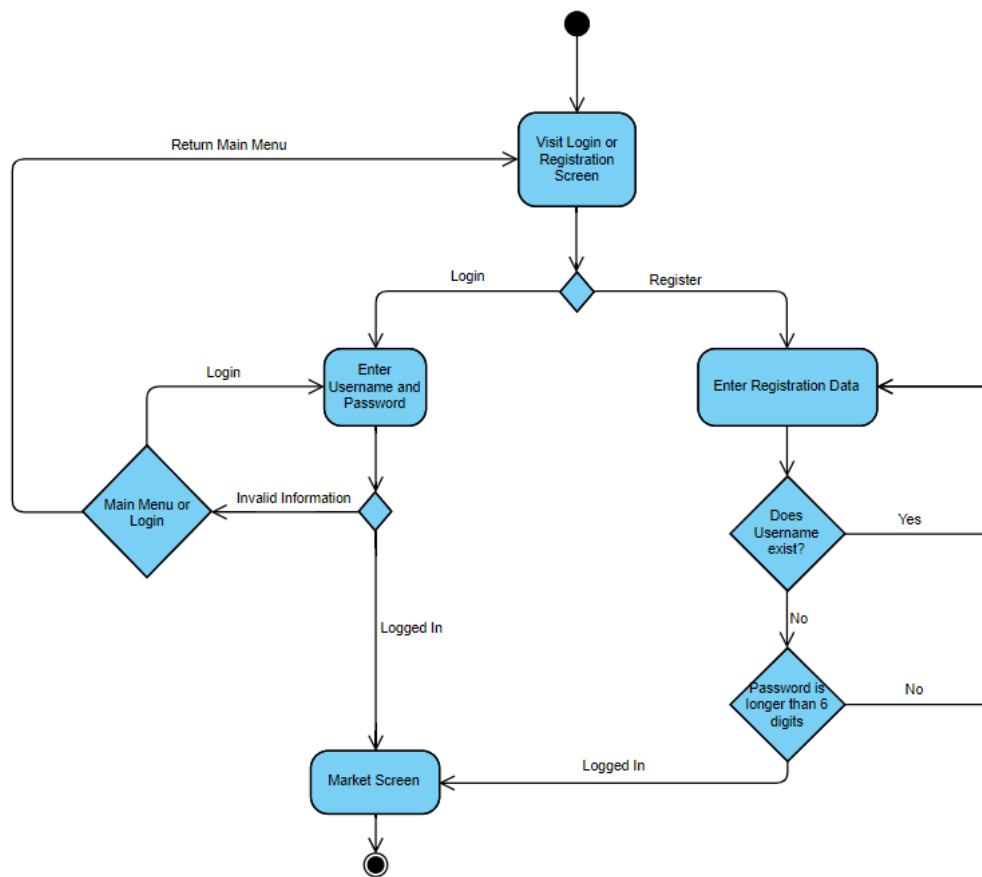
2.4.3 Activity Diagram

Diagram 4. : Login Screen, Activity Diagram

After running the program, it initially starts from a login screen. Users can either login using their information or register to the system. If the user decides to login, it will require a username and a password. If an information is invalid, the user can decide whether to re-enter the information or return to the main login screen. If the user is not yet registered and chooses to sign up, a registration screen appears. The user have to submit registration data to the system in order to register successfully. If the chosen username already exists in the system, the program asks for a new username or the given password is less than 6 digits, the program naturally asks for a new password. After these processes, the user logs in to the system and a market screen appears.
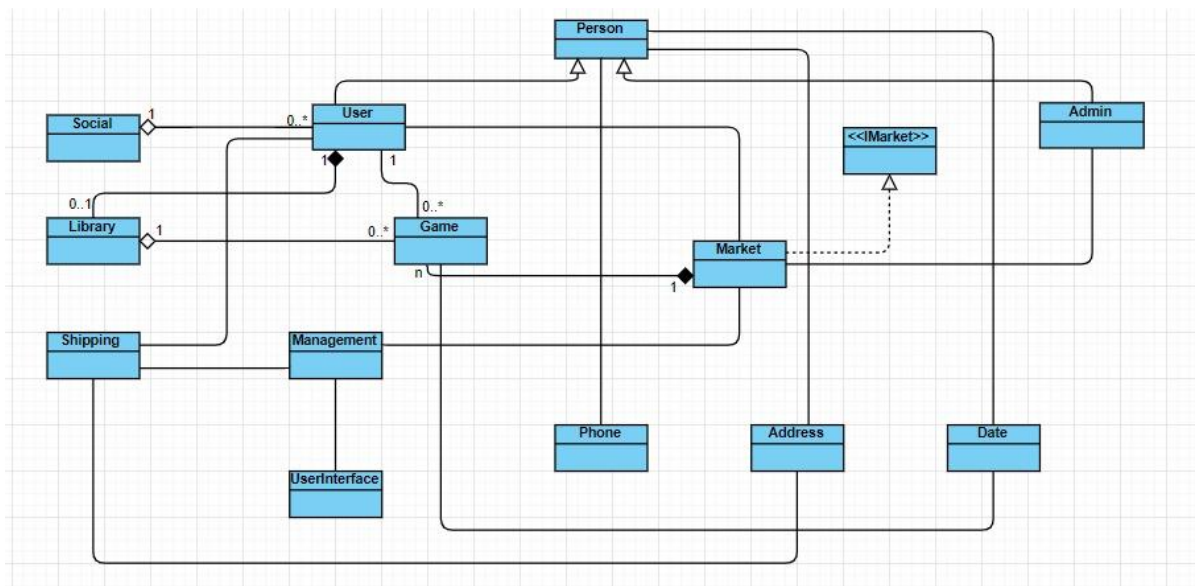
11

2.4.4 Relation Diagram

Diagram 5. : Relations

In this diagram, classes are shown briefly without functions, methods and attributes. There is primarily one super class which is extended by User and Admin classes. Besides there is one Interface class (IMarket) which is implemented by Market and this Market class consists of n amount of games. Additionally each user has a friendlist and a game list which is called the Library.

2.4.5 Class Diagrams

Person

-name:String
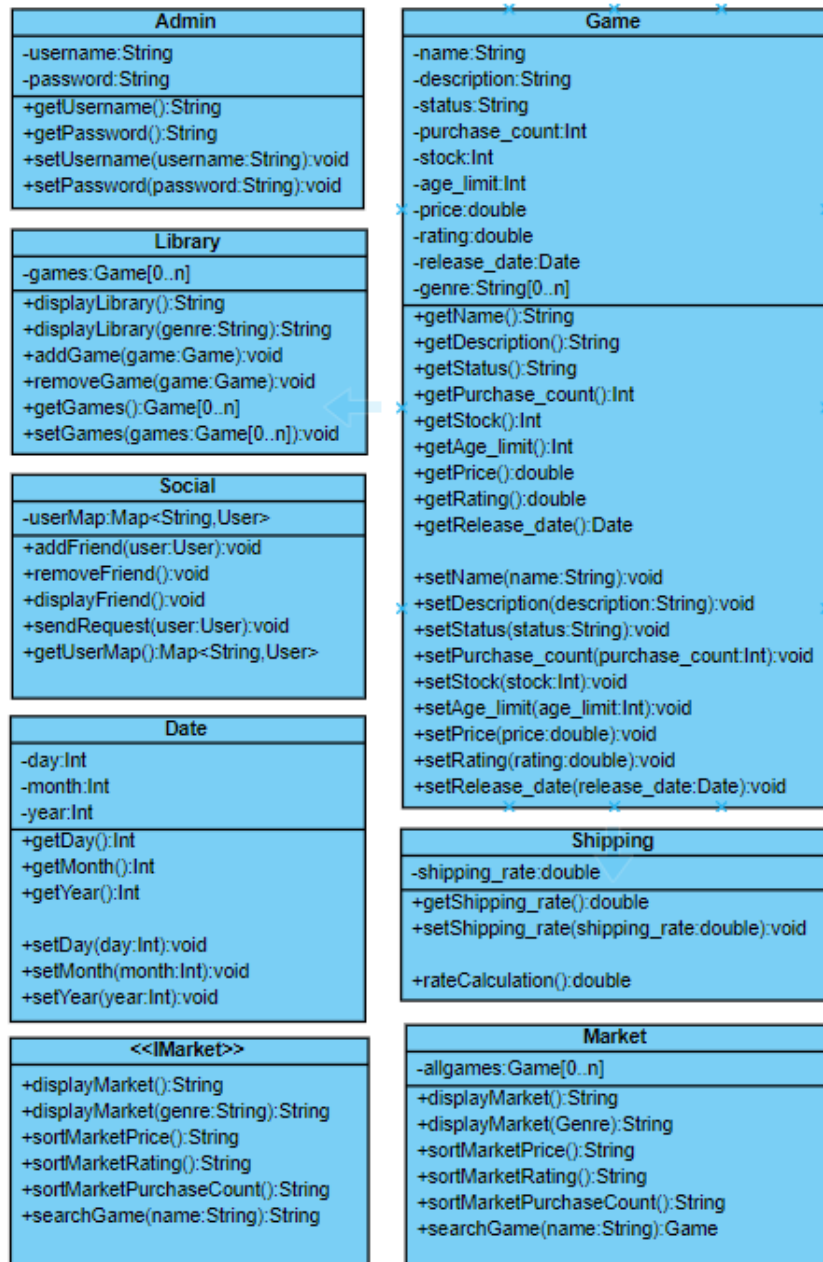-lastname:String
-email:String
-address:Address
-phone:Phone
-birthdate:Date

+getName():String
+getLastname():String
+getEmail():String
+getAddress():Address
+getPhone():Phone
+getBirthdate():Date

+setName(name:String):void
+setLastname(lastname:String):void
+setEmail(email:String):void
+setAddress(address:Address):void
+setPhone(phone:Phone):void
+setBirthdate(birthdate:Date):void

User

-ID: Int
-username:String
-nickname:String
-password:String
-library:Library
-friendlist:User[0..n]
-pending_request:User[0..n]

+getID():Int
+getUsername():String
+getNickname():String
+getPassword():String
+getLibrary():Library
+getFriendlist():User[0..n]
+getPending_request():User[0..n]

+setID(ID:Int):void
+setUsername(username:String):void
+setNickname(nickname:String):void
+setPassword(password:String):void
+setLibrary(library:Library):void
+setFriendlist(friendlist:User[0..n]):void
+setPending_request(pending_request:User[0..n]):void

Phone

-countrycode:String
-citycode:Int
-number:Int
-type:String

+getCountrycode():String
+getCitycode():Int
+getNumber():Int
+getType():String

+setCountrycode(countrycode:String):void
+setCitycode(citycode:Int):void
+setNumber(number:Int):void
+setType(type:String):void

Address

-streetname:String
-town:String
-city:String
-country:String

+getStreetname():String
+getTown():String
+getCity():String
+getCountry:String

+setStreetname(streetname:String):void
+setTown(town:String):void
+setCity(city:String):void
+setCountry(country:String):void

Management

+purchase():void (?)
+rent():void
+login():void
+register():void
<none yet>

UserInterface

<none yet>

Diagram 6.1. : Class Diagram 1

13

Admin

-username:String
-password:String

+getUsername():String
+getPassword():String
+setUsername(username:String):void
+setPassword(password:String):void

Library

-games:Game[0..n]

+displayLibrary():String
+displayLibrary(genre:String):String
+addGame(game:Game):void
+removeGame(game:Game):void
+getGames():Game[0..n]
+setGames(games:Game[0..n]):void

Social

-userMap:Map<String,User>

+addFriend(user:User):void
+removeFriend():void
+displayFriend():void
+sendRequest(user:User):void
+getUserMap():Map<String,User>

Date

-day:Int
-month:Int
-year:Int

+getDay():Int
+getMonth():Int
+getYear():Int

+setDay(day:Int):void
+setMonth(month:Int):void
+setYear(year:Int):void

<<IMarket>>

+displayMarket():String
+displayMarket(genre:String):String
+sortMarketPrice():String
+sortMarketRating():String
+sortMarketPurchaseCount():String
+searchGame(name:String):String

Game

-name:String
-description:String
-status:String
-purchase_count:Int
-stock:Int
-age_limit:Int
-price:double
-rating:double
-release_date:Date
-genre:String[0..n]

+getName():String
+getDescription():String
+getStatus():String
+getPurchase_count():Int
+getStock():Int
+getAge_limit():Int
+getPrice():double
+getRating():double
+getRelease_date():Date

+setName(name:String):void
+setDescription(description:String):void
+setStatus(status:String):void
+setPurchase_count(purchase_count:Int):void
+setStock(stock:Int):void
+setAge_limit(age_limit:Int):void
+setPrice(price:double):void
+setRating(rating:double):void
+setRelease_date(release_date:Date):void

Shipping

-shipping_rate:double

+getShipping_rate():double
+setShipping_rate(shipping_rate:double):void

+rateCalculation():double

Market

-allgames:Game[0..n]

+displayMarket():String
+displayMarket(Genre):String
+sortMarketPrice():String
+sortMarketRating():String
+sortMarketPurchaseCount():String
+searchGame(name:String):Game

Diagram 6.2. : Class Diagram 2

14

Specific Requirements

3.1 External Interfaces

The program is designed using JavaSwing GUI. The project team discussed which UI to use at the first glance, contemplated between JavaSwing and JavaFx and finally decided to go with JavaSwing.

Using this interface, the project presents the best and the simplest user interface with a good design. Implementation is easy but the design was a bit difficult. The problems that have been encountered will be mentioned in the next chapters.

3.2 Performance Requirements

The project does not require a high end system to run by any means yet since it is decided to use visual images in the project, refreshing the page might require a bit time. This runtime also may extend according to the size, resolution and the amount of the images that are and will be used.

3.3 Database Requirements

The program fetches its database from *.csv* files. Although the project members would have loved to implement much more complex and useful database systems (e.g. an SQL database), it was simply not possible to invest enough time to get a grip of the implementation and the logic behind it.

Although the project database was not based on a high-functioning system, the current system works just as fine as it would be. File operations were done with great finesse and without any errors or problems that may or may not occur in the future.

The project team stands with its database system and understand the importance of it for it's a crucial part of the project.

3.4 Design Constraints

Since it's a digital game marketing program, it was neither logical nor viable option to design a physical copy shipping system. Although this idea did not carry a great importance for the project, the team insisted on creating one and worked on it until figuring out that it's simply not possible.

This shipping system was mentioned in the previous progress reports as well but it included no further information besides the idea of it. The project team was aware and also is aware of the abstractness of such idea, but not enough to actually remove it from the contents of the project. Further explanation of the shipping system failure will be made in the next chapters.

There was no "constraints" that has set boundaries to the project team other than the desire of creating a physical system in a project that couldn't get as much more abstract as it is originally.

Conclusion

4.1 Problems & Solutions

- ○ Asynchronization:

  The general problems of the project was mentioned briefly before. First and foremost problem of the project was the asynchronized coding platform choices. The members decided to use different *IDE's* which caused a problem on the *GitHub* side which caused a huge time penalty. Rather than time issues, no other problems occurred due to the sync issue.

- ○ Shipping:

  Another problem that has been mentioned is the *physical shipping system*. The problem that it has caused is exactly as it's mentioned before, it is simply not possible to create a phsyical system in a digitally made project. But a solution was brought up recently. The team members suggested to change the idea of it, the products now have alternative versions to buy.

- ○ Rent Date Calculation:

  It is still obscure who has come up with the idea of renting games, but once it is implemented into the project, it attracted a great attention among the members. Although it's a nice feature, the requirements it would bring were not to be ignored.

17

When a user rents a game, system saves the date that it has been rented to calculate the remaining time. The concept is quite simple, the program simply has to compare two different times and do a basic math calculation. But the comparison somehow didn't work at the first phase.

o Imprisoning Layout:

The main *IDE* that the project team used did not support an abstract layout in the *JavaSwing GUI*. Which means the design could not be made freely, every component needed to be in a specific, bordered location. This problem was solved via simply using another *IDE*.

4.2 Teamwork & Task Management

Due to the global state that currently is going on, it was challenging to properly work as a team. That's why during the first months of the project, *GitHub* commits were empty, neglected and desynched.

Although it was challenging to team up, the coding and the designing part were never done by only one member seperately. Each part was designed after gathering through a social platform and the overall designed has the contribution of every team member.

However, coding is another level of the task management. While the design was made all together, the project was parsed into different sections to be coded in a faster and quicker phase. While parsing the code, the team parsed it as equal as possible so that the design contributions will not be unjust.

4.3 Achievements

- ✓ Fully functioning database system.

- ✓ Fully functioning file operations.

- ✓ Variation in products.

- ✓ Fully functioning, appealing GUI.

- ✓ Correct price and time calculations.

- ✓ Better visualization and image implementation.

4.4 Additional Comments

The project team is satisfied with the final achieved program, but desires to step it up a bit further. Even after the deadline, the plan of making it better and better still remains.

User Manual

5.1 Software Description

The program starts with a login page. One must either sign-in or sign-up to enter to the system. There's no close button, the program can be closed by pressing the X button on the top right corner on the <u>main panel</u> which means closing the auxiliary panels won't close the program.
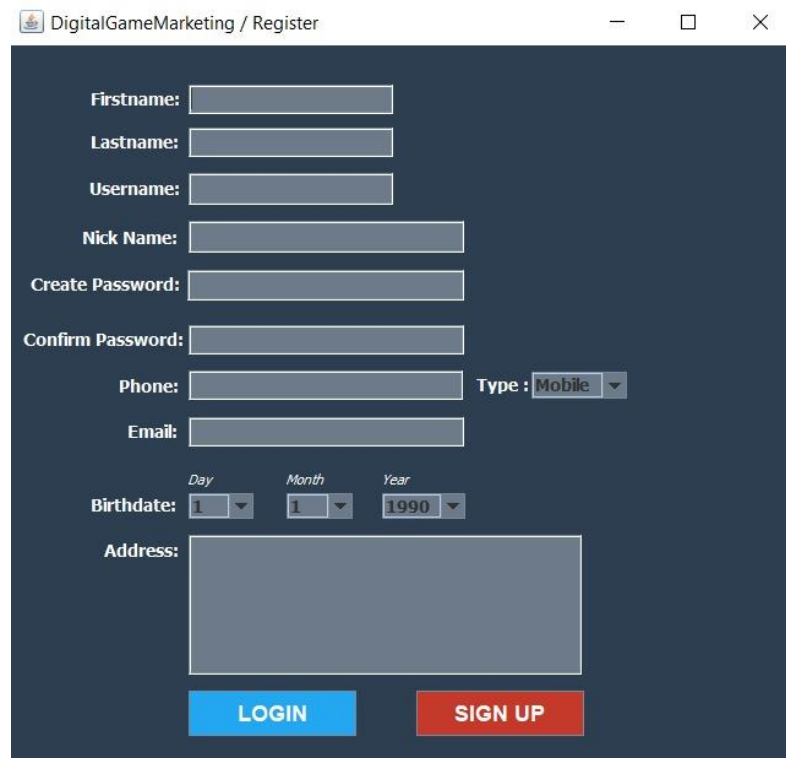
5.2 How To Use

5.2.1 Login Page

An examplary login page with admin's info. (password: admin35) The admin uses polymorphism with a normal user. (Admin functions can be tested by logging in with admin.)

5.2.2 Register Page

An examplary register page.

5.2.3 Main Menu
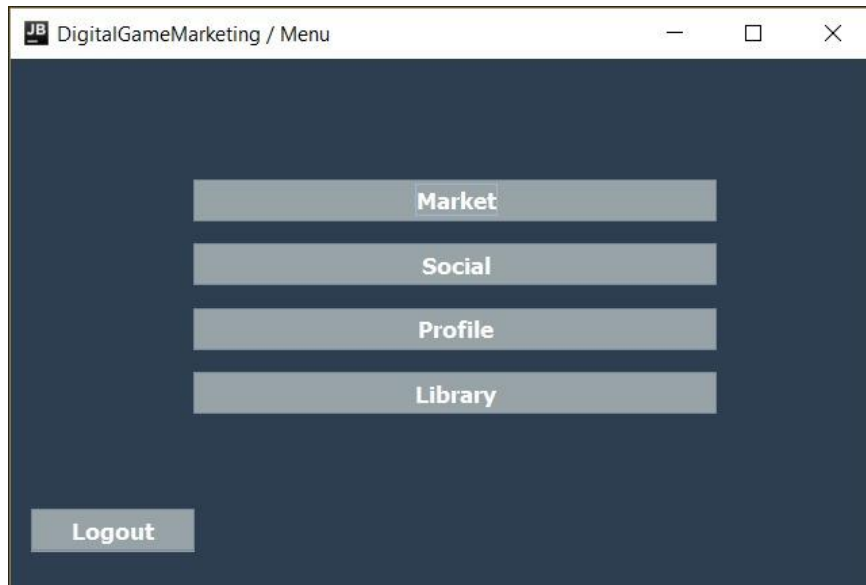
5.2.3 Market
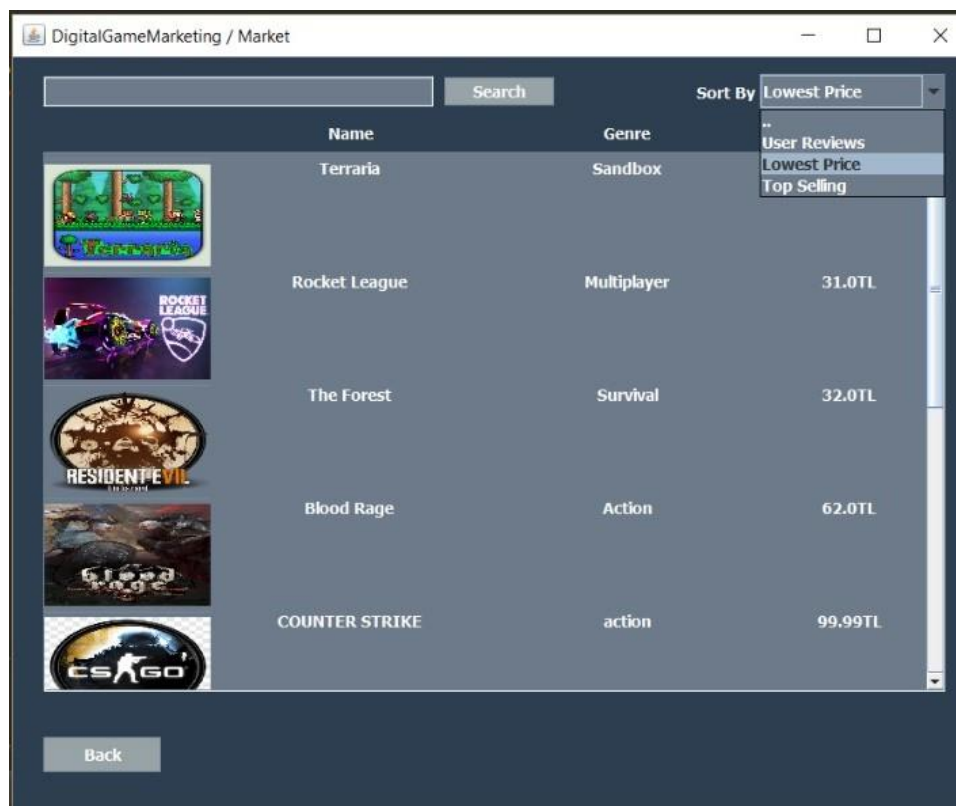
Users can search any game they wish to find by the search bar. Also it is possible to sort the market by either price, review or selling amount.

5.2.3 Game Page
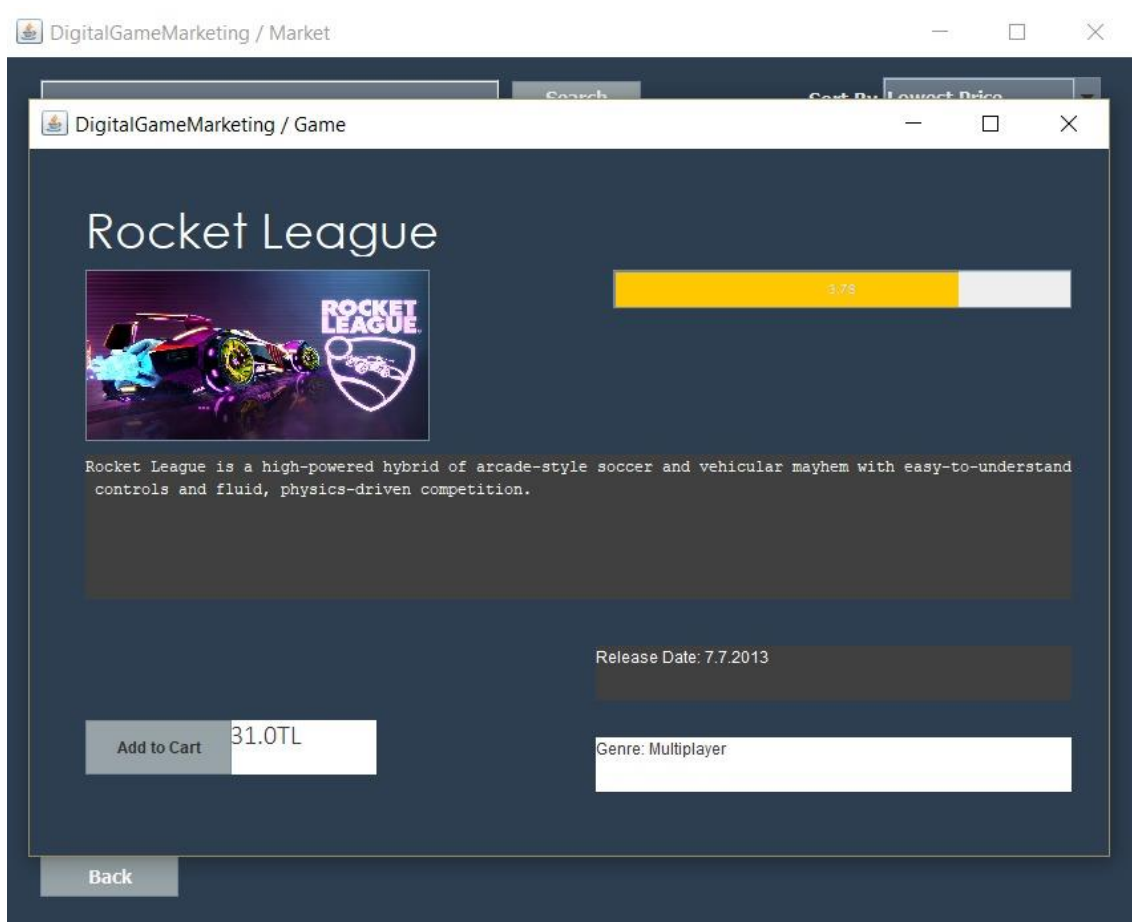
An examplary game page with Add to Cart option and its information.

5.2.4 Transaction
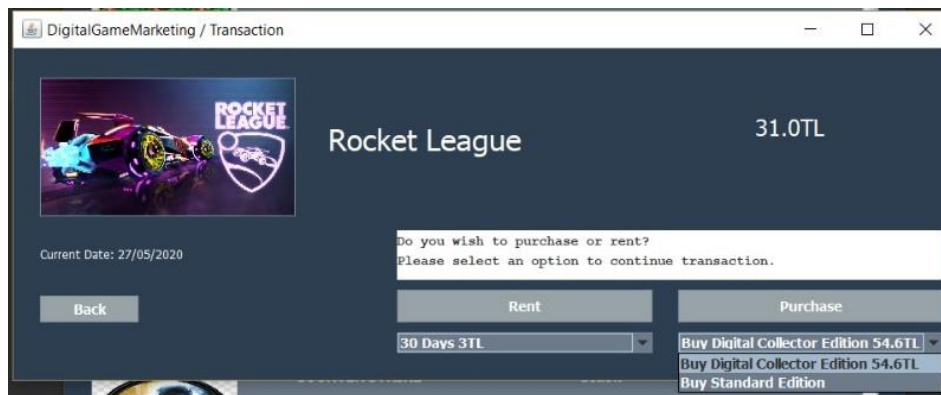
The transaction page offers two options, rent and purchase. The user can only rent the standard version of the game for a month, 2 months or 6 months if desired. Also it is possible to purchase different versions of the game with extras.

5.2.4 Social
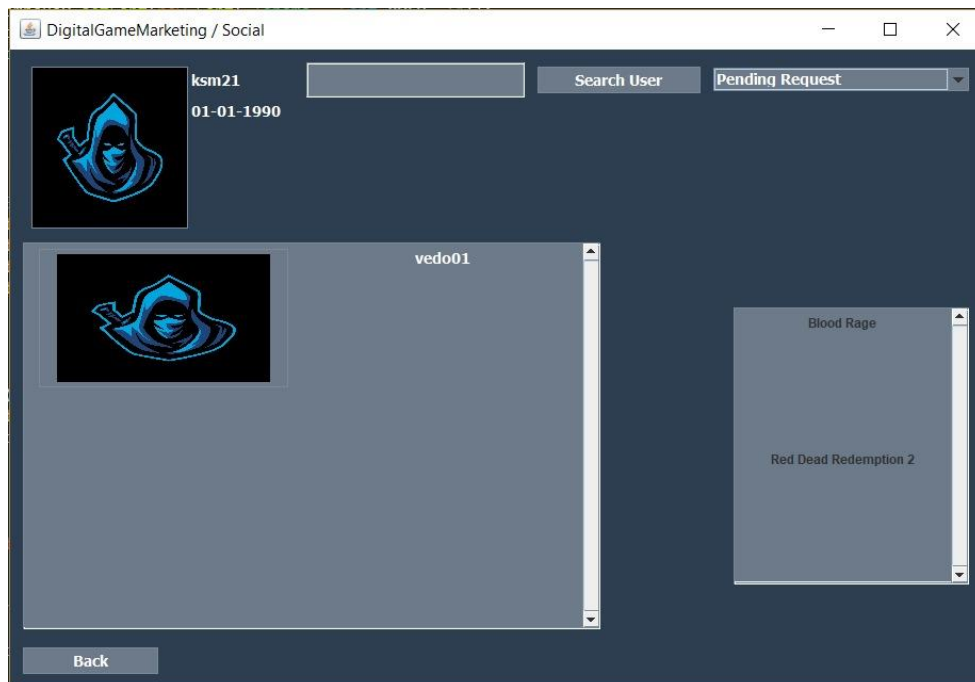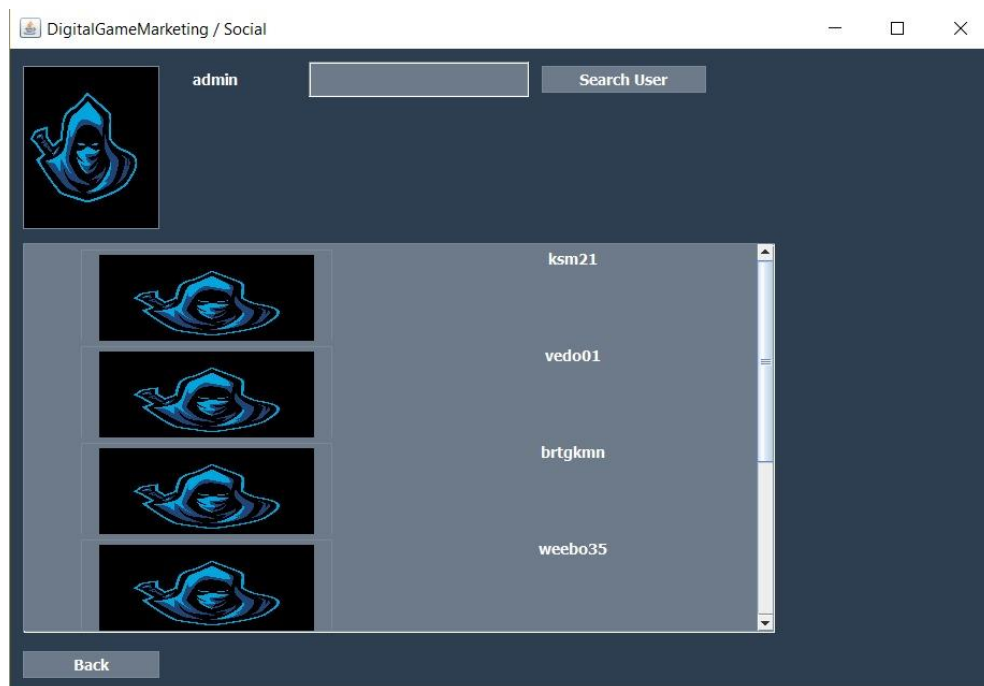
5.2.4 Library

5.2.4 Profile Page

5.2.4 Admin Pages

Admin can add any game by specifying its details. The image of the game will be set when a new image named with the exact same name of the game into the images folder.

Admin can also change the specs of the product.

Index

---

o Date Comparison, Stack Overflow

https://stackoverflow.com/questions/14618608/calendar-issue-in-adding-month-1-to-to-calendar-month-in-android

o ComboBox, Stack Overflow

https://stackoverflow.com/questions/11999560/get-combobox-value-in-java-swing

o JScrollPane, Stack Overflow

https://stackoverflow.com/questions/48615579/how-to-dynamically-add-components-to-java-jscrollpane

o Panel Layouts, Geeksforgeeks

https://www.geeksforgeeks.org/java-swing-scrollpanelayout-class/

o JPanel Examples, Geeksforgeeks

https://www.geeksforgeeks.org/java-swing-jpanel-examples/?ref=rp

o JButton Examples, Stack Overflow

https://stackoverflow.com/questions/43225745/how-to-add-jbuttons-to-jscrollpane