ARCADE VERSUS

Realized by:

Andrea Giordano, Giulio Bollea, Jie Huang

Academic year 2023-2024

Software engineering course, spring semester

Professor: Gong Xiaoliang

Contents

About Us

Our Team

Andrea Giordano

giordano.andy03@gmail.com

Responsibility: 33.33%

Game Coding
Connection Setup Coding

Jie Huang

jennifer03huang@gmail.com

Responsibility: 33.33%

Game Coding
Connection Setup Coding

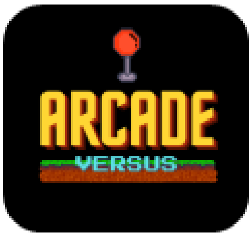
Giulio Bollea

bollea.giulio@gmail.com

Responsibility: 33.33%

Game Coding
Website Coding

The Logo of our APP:



Requirement analysis

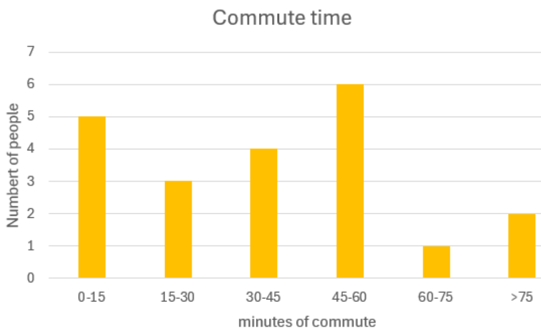
1. Introduction

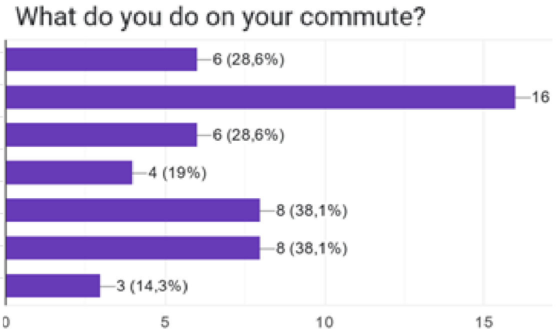
The mobile application being discussed and analysed in this project documentation is the end result of the Software Engineering course taught by Professor Gong Xiaoliang (龚晓亮) throughout the 2024 Spring Semester at Tongji University.

1.1 Background

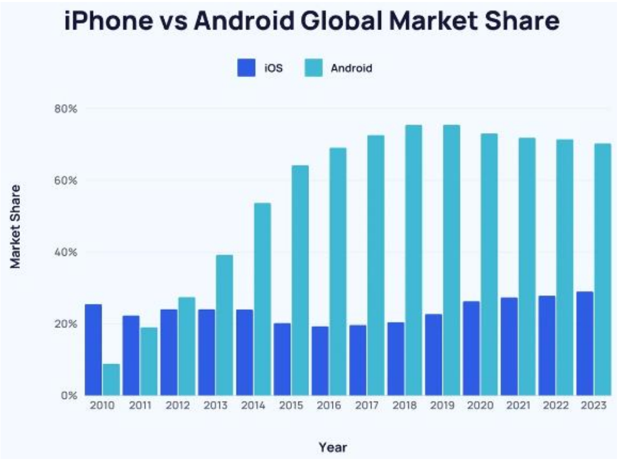
At first, a preliminary analysis of the current daily habits of people was conducted. It has been highlighted that most people nowadays study or work far from where they live, meaning that many need to rely on public transportation or cars to move around. Amongst those who don't drive, it has emerged that the great majority of people consider the time they spend commuting as a waste of time, and thus they wish there were some better ways of employing their daily commute.

A survey was conducted. As it turns out, most people commute for about 45-60 minutes, and while doing so, they are highly likely to listen to music, as shown in the bar charts below.

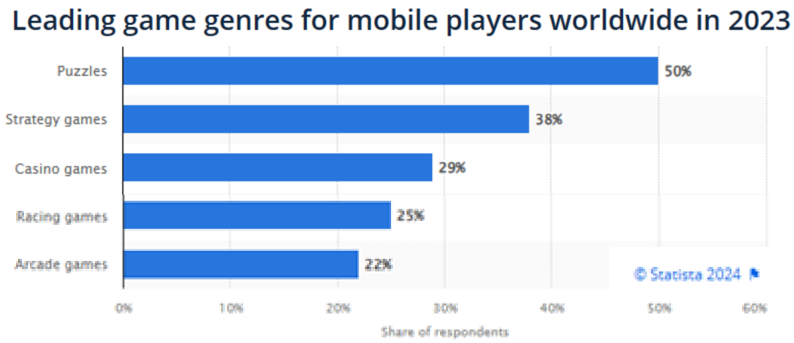




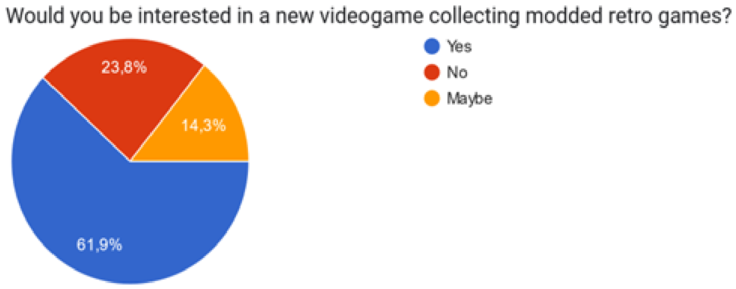
As on public means of transport, especially during rush hours, it is usually difficult to have enough space to entertain oneself in recreational activities which go beyond reading, listening to music or in general the usage of one's own smartphone, the solution to their problem ought to be tailored for users with little to no room to move. Also considering the current trends in worldwide mobile app downloads, which show a staggering +80% since 2016 (Statista, 2024), it is clear that also mobile games are bound to become more and more popular, even among the elder. For this reason, a mobile game has been thought of as a suitable software-based approach to the needs of commuters.



By looking into the market share of the most popular operating systems, it has been discovered that, although the popularity of Android devices has undergone a slight decline in the recent years, it is by far the most common OS worldwide (explodingtopics.com). Hence, in order to address the largest number of potential users, an Android application has been agreed on being the best choice. In addition, an ad-hoc web page allows users to easily download the game.



For what concerns the kind of gameplay experience users are offered, a study from Statista shows that approximately half of the players like to play simple puzzle games characterized by captivating music and designs. Furthermore, the survey shows that arcade and retro games have proven to be appealing to more than 1 in 5 players. Thus, the style of the developed game is directed towards that way. As a matter of fact, our survey shows that more than 3 in 4 users would play a collection of revised retro games, as shown in the pie chart below.



1.2 Purpose

The project aims at providing users with a game:
- which can be easily installed and played.
- which is fun and easy to play.
- which can evoke nostalgia, by including retro games with their original soundtracks.
- which lets users connect and play with friends.
- with a wide variety of games.

An essential part of the project is a website:
- which allows users to download the game.
- which makes it easy to share the game with others.
- which allows users to contact the developers.
- which displays all the information users may need.

2. Requirement Gathering

2.1 User story table with duration with priority estimation

The survey we conducted has highlighted many issues our potential users deal with on a daily basis. From the users' feedback, we have inferred what our clients' perspective of their daily commute is. Our job is to create some software-based tool to help them relieve their stress. After thoroughly going through all the users' suggestions for our game, we have managed to synthesize their needs into a table of requirements in the form of a user story table, as shown right below.

Identifier	User story	Size	Duration
ST-1	As someone who travels with my friends, I find that competitive game apps are very fun to play as a group to pass time. We are always searching for the newest multiplayer games to try.	10	5
ST-2	As a solo gamer, I prefer to play games by myself or against bots.	9	4.5
ST-3	As a player, I want to be able to invite my friends to join multiplayer sessions and enjoy playing together.	9	4.5
ST-4	As a commuter, the application should have an intuitive interface, which is easy to use even after a long and tiring day spent at work.	4	2
ST-5	As a commuter, I like listening to something while commuting, so the game soundtrack has to be catchy and memorable.	2	1
ST-6	As a user I want to be able to create a profile within the game, to track my progress.	5	2.5
ST-7	As a non-experienced gamer, I would be interested in playing a collection of retro games, but it should have clear instructions on the rules and controls of each game.	2	1
ST-8	As a casual gamer, I want to be able to suggest new games to be added to the list of games already present, to add variety to my gaming experience.	1	0.5
ST-9	As a user, I want to be able to provide feedback to the developers, so that the game can be constantly maintained and updated.	2	1

2.2 Functional Requirements table with priority estimation

From the user story table above, we have categorized the user requirements into two main classes: the functional requirements and the non-functional requirements. The former concern themselves with the basic functionalities of our application, without which our potential users would not be able to relieve their pain and experience a nice and smooth gaming experience, or no experience at all. Here, in the next table, one can find such essential functionalities our game must provide the users with.

Identifier	Functional requirement	Priority
REQ-1	The system shall include an intuitive game selection screen, that displays icons for each available arcade game	3
REQ-2	The system shall include single player mode, which enables each user to play solo or against bots	5
REQ-3	The system should include multiplayer mode: – On one device (splitting the screen): ○ allowing player to play taking turns (turn-based gameplay) ○ allowing playing at the same time ○ organizing a tournament – On different devices: ○ turn-based gameplay ○ real-time synchronisation	5
REQ-4	The system shall allow each user to create a profile	1
REQ-5	The system shall provide options to invite friends to join multiplayer sessions: – Enable players to create multiplayer room – Implement a tournament organiser feature, generating tournament brackets	5
REQ-6	The system should include a game soundtrack to enhance the experience, giving the user the possibility to disable this functionality	2
REQ-7	To download the APK file directly from the official web page	5

2.3 Non-Functional Requirements

The non-functional requirement table, instead, lists those functionalities that indeed bring the user experience to the next level, but are not part of the set of essential functionalities of the system. For this reason, in the table right below this paragraph one can easily notice that the priorities associated to these requirements are generally low. Nonetheless, it's important to keep in mind that these very non-functional requirements may be exactly what could make our game become popular, so we ought not to underestimate their importance.

Identifier	Non-functional requirement	Priority
NFREQ-1	The system shall provide updating and maintenance	3
NFREQ-2	The website shall include an interface to allow user to express feedback and suggestions to developer team	2
NFREQ-3	The games in the system should be updated frequently	2
NFREQ-4	The system shall include games settings, so that the user may control certain functionalities	1
NFREQ-5	The system shall include a way to share the game with possible other users	1
NFREQ-6	The system should provide for each game a set of instruction on the games rules and controls	4
NFREQ-7	The system should protect user data and privacy	2

3. Requirement analysis

3.1 Use case diagram of the system

After studying the needs of our potential customers by listing them in the requirement tables, it has become necessary to further investigate the relationship between the many actors that interact through our application and deepen our understanding in the way the use cases are interconnected to one another.

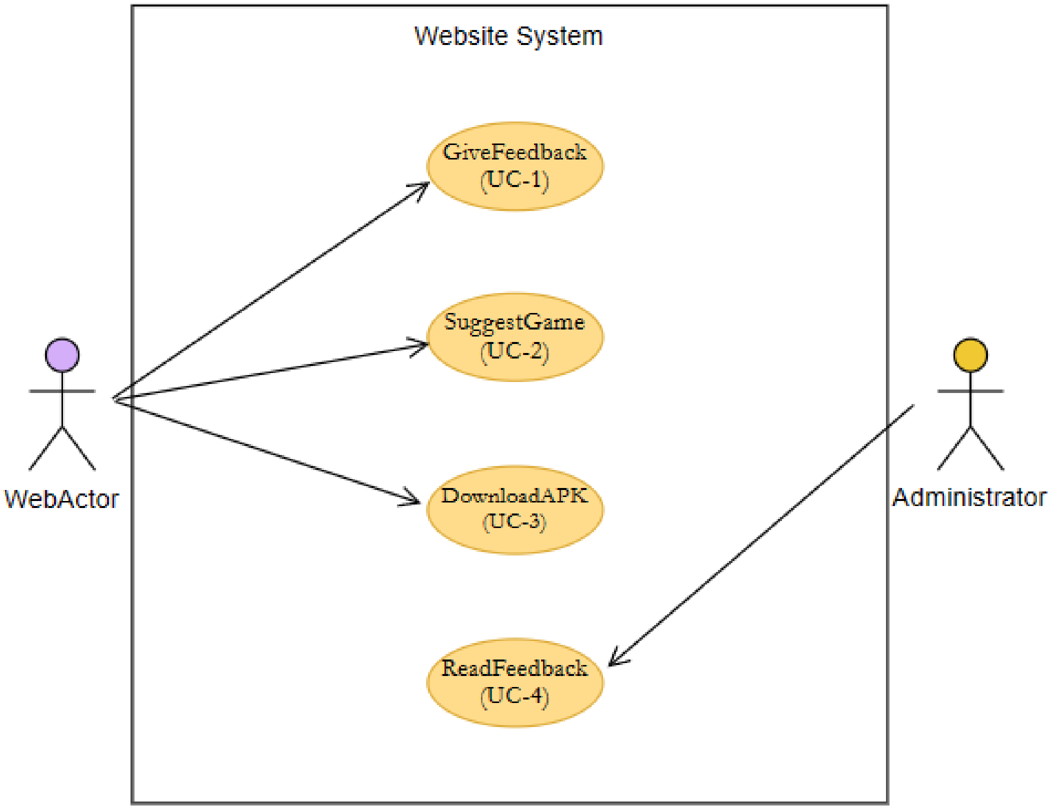
For the sake of easiness, we have decided to report here the use cases for the website system and the mobile application system in two separate paragraphs, in order to allow the readers of this software documentation to more easily have a clear understanding of the use cases of our project.

Here are listed the use cases for the website system, and then its use case diagram is shown to provide readers with a visual representation of the actors' goals.

Website use cases:

- Web user
- Website administrator

ACTOR	ACTORS' GOAL	USE CASE NAME
Web user	To provide feedback to the developers	GiveFeedback (UC-1)
Web user	To suggest new games	SuggestGame (UC-2)
Web user	To download the application	DownloadAPK (UC-3)
Administrator	To interact with the users	ReadFeedback (UC-4)

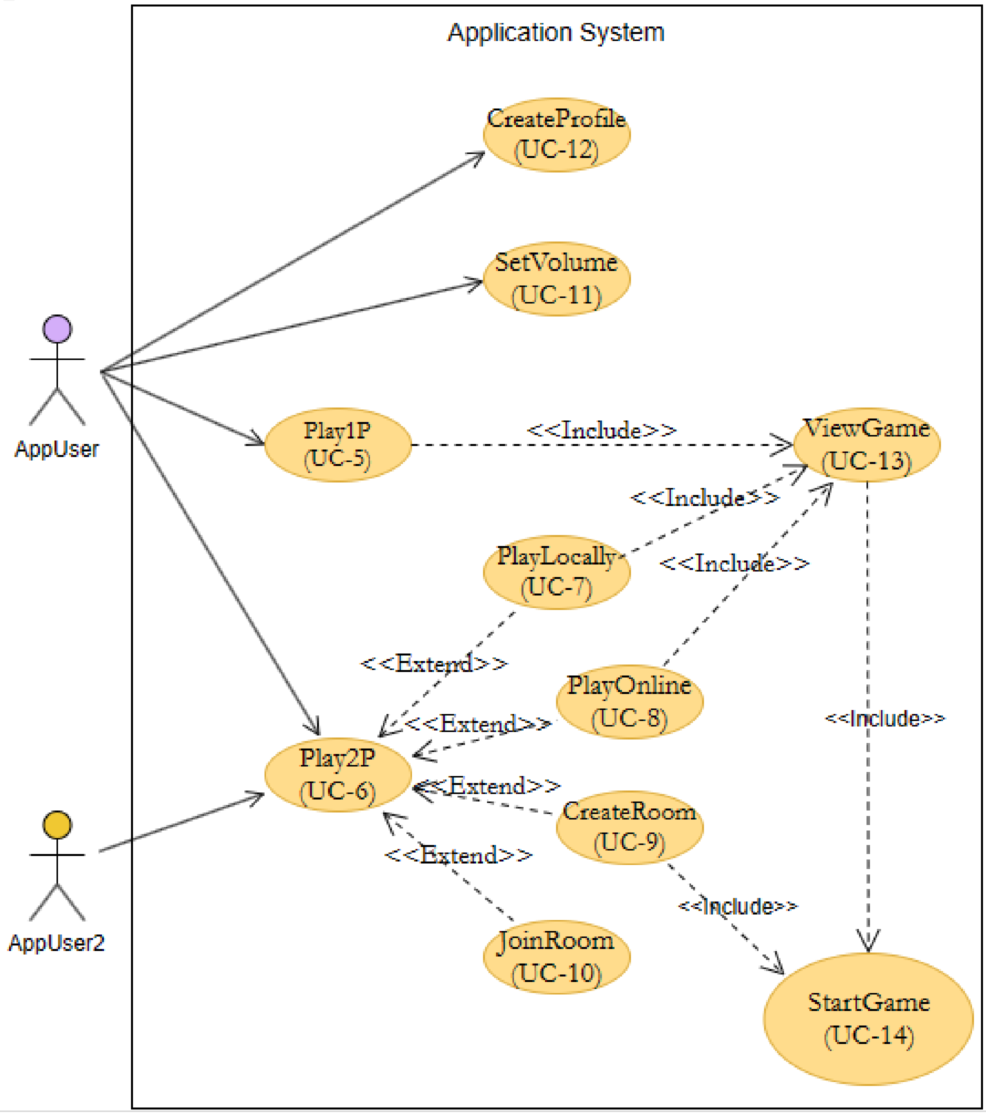


Analogously, the application use cases are again listed in the table below, next to their corresponding actors' goals and then again in the use case diagram for the application system, as shown below.

Application use cases:

- App user
- App user 2

ACTOR	ACTORS' GOAL	USE CASE NAME
App user	To play in single player mode	Play1P (UC-5)
App user	To play in multiplayer mode	Play2P (UC-6)
App user	To play on one device	PlayLocally (UC-7)
App user	To play on or multiple devices	PlayOnline (UC-8)
App user	To create sessions with friends	CreateRoom (UC-9)
App user	To join sessions with friends	JoinRoom (UC-10)
App user	To choose to switch off or turn on the background in app music	SetVolume (UC-11)
App user	To create a new profile	CreateProfile (UC-12)
App user	To view the game between the list of proposed ones	ViewGame (UC-13)
App user	To start playing the selected game	StartGame (UC-14)



3.2 One example of Use case narrative

Despite having listed the use cases and plotted the corresponding diagrams, sometimes this is not enough to fulfil the needs of the reader of this documentation. In other words, we ought to give some additional explanation to the entries of the use case tables above. Here is one detailed example of a use case narrative, which we have selected based on the crucial role it plays in our application system.

Use case UC-6: Play 2P
Related requirements: REQ1, REQ3, REQ4 and REQ5 Initiating actor: App user **Actor's goal:** to play in multiplayer mode **Participating actors: a**pp user1 and app user2 **Preconditions:** the app is downloaded on the user's Android device. The user should have created a profile **Postconditions:** the user is redirected to a new menu, with different multiplayer game modes to choose from. **Flow of events for main success condition:** ← 1. System redirect the app user to the multiplayer games mode menu → 2. App user chooses between the different game modes 3. extend: PlayLocally (UC-7) 3. extend: PlayOnline (UC-8) 3. extend: CreateRoom (UC-9) 3. extend: JoinRoom (UC-10)

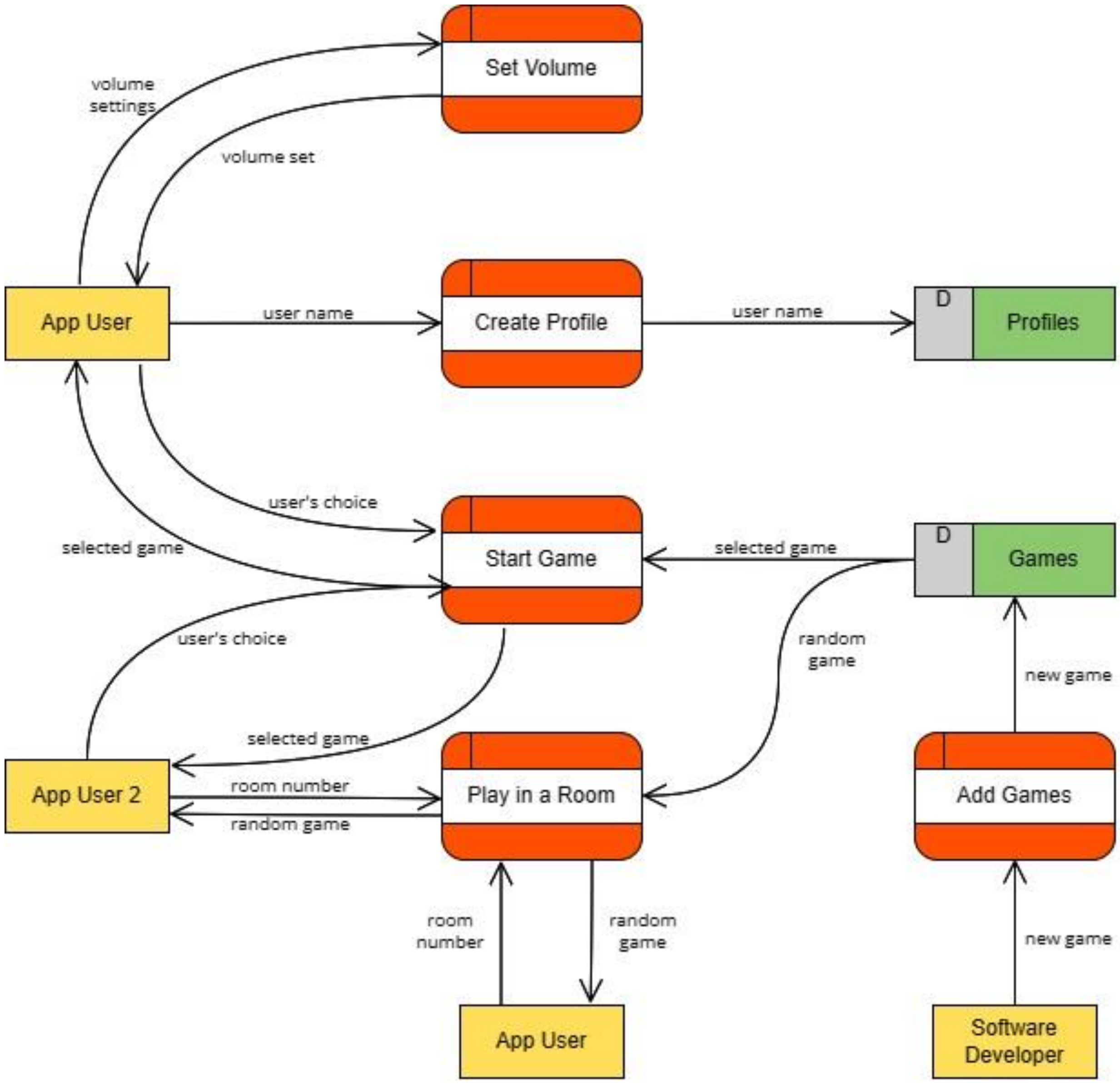
3.3 Use case Priority Matrix

Now that the reader has a complete understanding of the processes of our system, we ought to move forward and try to understand the priority of each of the use cases. In the table below, one can find a list of all the use cases, the importance of each of them under different points of view (in the Ranking Criteria column of the table). The scores are then summed up and depending on it, a use case is assigned a certain level of priority, which is then reflected in the build cycle we should implement the functionality in.

Use case	Ranking criteria from 1 to 5						Total score	Priority	Build cycle
	1	2	3	4	5	6			
GiveFeedback (UC-1)	3	2	1	1	2	2	11	Low	3
SuggestGame (UC-2)	3	2	1	1	2	2	11	Low	3
DownloadAPK (UC-3)	1	5	1	1	5	5	18	Medium	2
ReadFeedback (UC-4)	1	1	1	1	2	1	7	Low	3
Play1P (UC-5)	4	3	3	2	3	3	18	Medium	2
Play2P (UC-6)	5	2	5	2	4	4	22	High	1
PlayLocally (UC-7)	4	2	4	2	3	3	18	Medium	2
PlayOnline (UC-8)	5	2	5	5	4	4	25	High	1
CreateRoom (UC-9)	4	2	4	5	4	4	23	High	1
JoinRoom (UC-10)	3	2	3	5	4	4	21	High	1
SetVolume (UC-11)	1	4	1	1	1	1	9	Low	3
CreateProfile (UC-12)	1	5	1	2	3	1	13	Low	3
ViewGame (UC-13)	5	5	2	1	1	1	15	Medium	2
StartGame (UC-14)	5	5	1	1	4	4	20	High	1

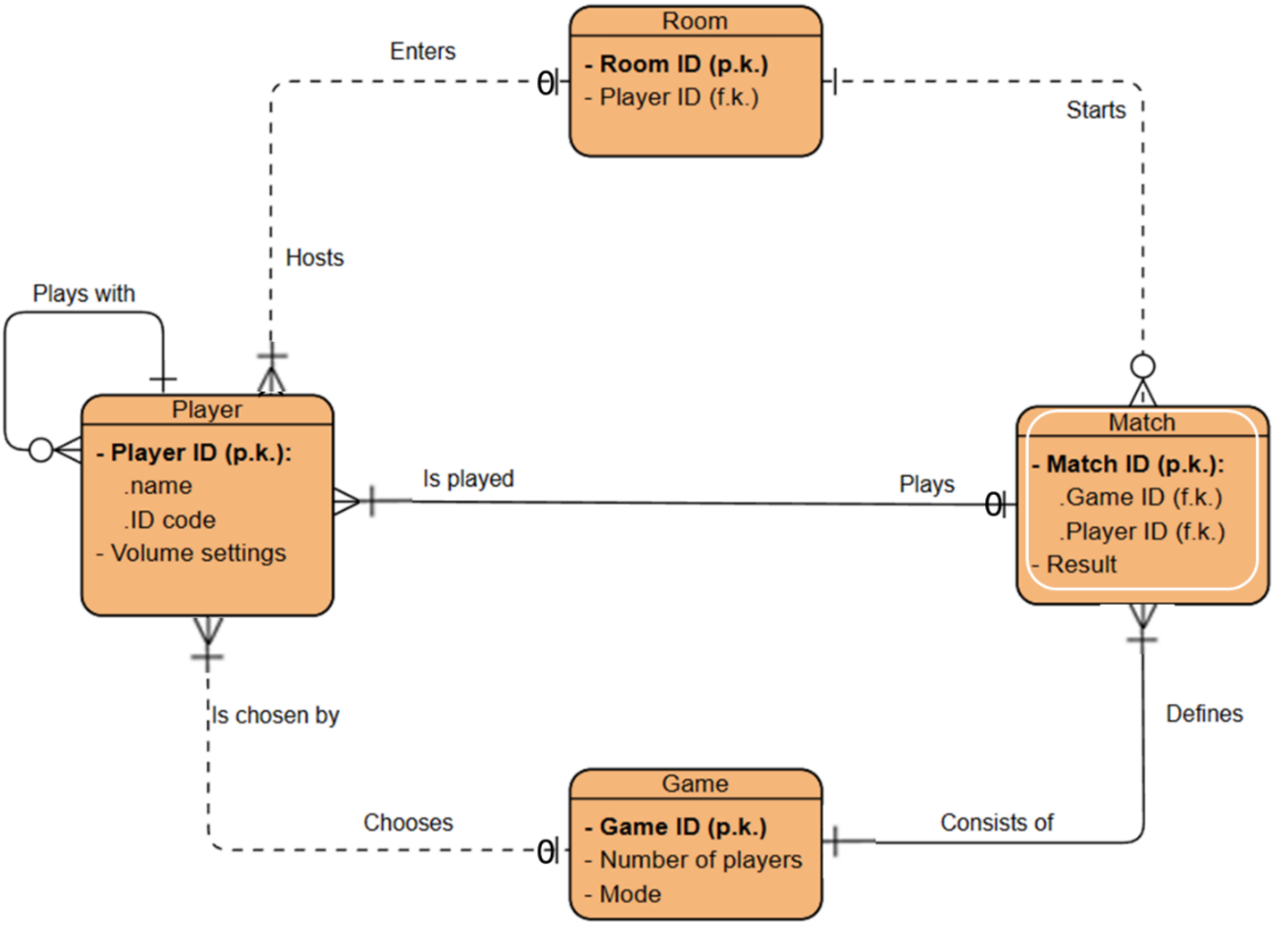
3.4 Data Flow Diagram（DFD）of the system

As our system is rather complex, just the mere listing of the use cases and their pictorial description in the use case diagrams is perhaps not enough. We ought to further explore the way data is exchanged, stored and processed by all the relevant entities playing a role in our system. Hence, we agreed that a Data Flow Diagram was the suitable tool to graphically design the way data flows in our mobile application system. As always, this has helped us in refining our understanding of the way our project works in different scenarios.



3.5 Entity relationship diagram (ERD) of the system

Finally, the way the entities take part in our system, their relationship and dependencies are addressed in the Entity Relationship Diagram. Its aim is to explicitly tell the data uniquely defining each entity and the additional data they are characterized by. In the following diagram, notice the relationship between each two connected entities, whether each entity is strong or weak, and the cardinality associated with each relationship. Immediately after the ERD, one can read a textual description of the diagram, providing additional explanation to the relationships involved in the pictorial representation.



One PLAYER plays with zero, one or more PLAYERS. This is because a player can decide to play in single player mode, multiplayer mode or enter a room and play with many players.

1. **One PLAYER enters zero or one ROOM. One ROOM hosts one or more PLAYERS.** In fact, a player can decide not to enter a room, and at the same time the player can decide to join at most one room at the same time. Multiple players can join a room.
2. **One ROOM starts zero, one or more MATCHES. A match is started by one ROOM.** Players in a room can decide whether to start a tournament. Once they do start the tournament, multiple matches are started by the game, to allow players to play against one another.

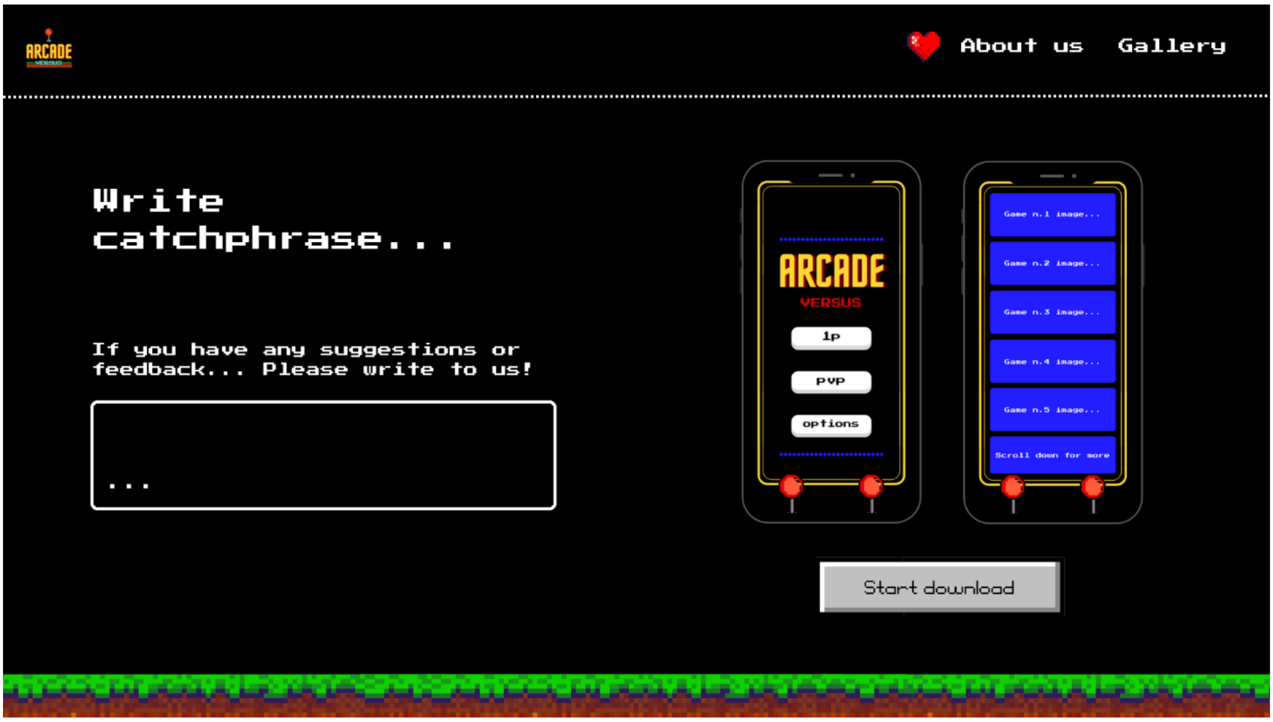
3. **One PLAYER plays zero or one MATCHES. One MATCH is played by one or more PLAYERS.** A player can be playing at most one match at any given time.
4. **One PLAYER chooses zero or one GAMES. One GAME is chosen by one or more PLAYERS.** A player can choose to play at most one game at any given time. However, a game can be played by multiple players at the same time.
5. **One GAME defines one or more MATCHES. One MATCH consists of one GAME.** For example, in the tournament started in a room, multiple players (playing in different matches), play the same game.

4. On-screen appearance requirement (UI)

Our idea for the user-interface was to make it clear and intuitive for new users, but we also wanted it to have that 'retro' feel and aesthetic that make the mobile application more visually appealing.

Not only we designed the interface for the application but also the website, since it's the first thing that people see before installing our system, so it's crucial that it displays what the mobile game offers to new potential players.

4.1 Website user interface:

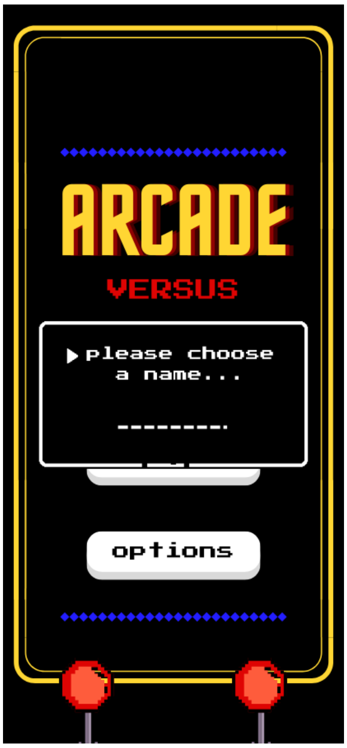


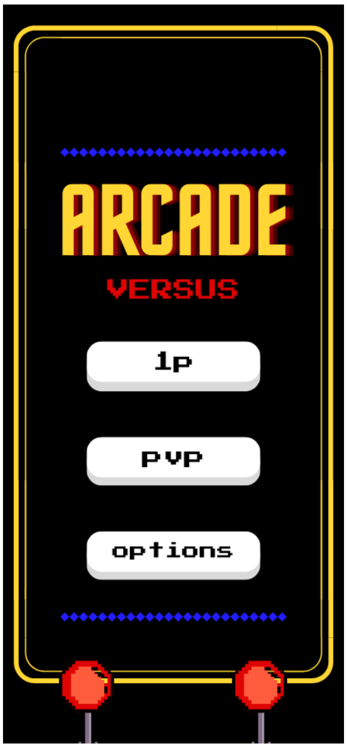
Write catchphrase...

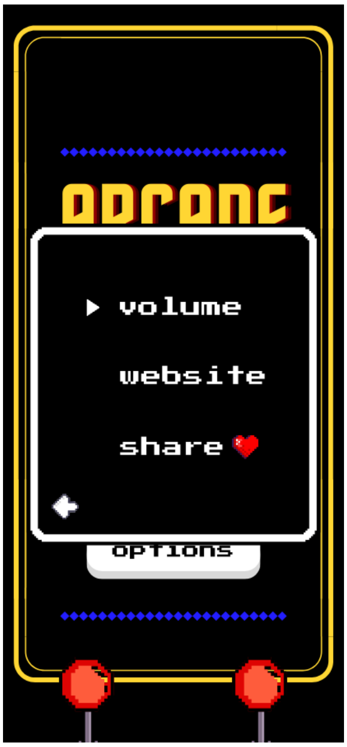
If you have any suggestions or feedback... Please write to us!

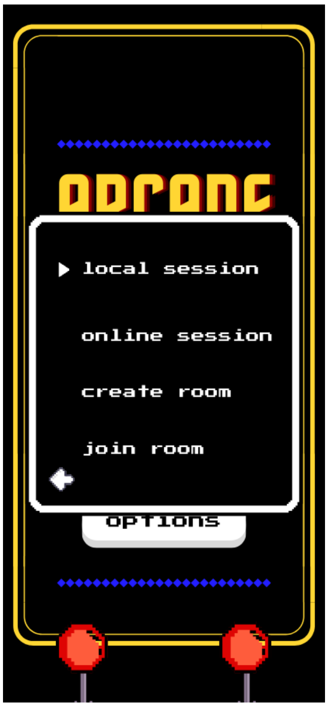
...

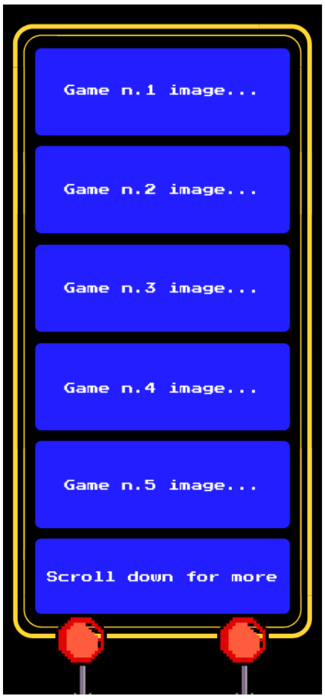
4.2 Application user interface:







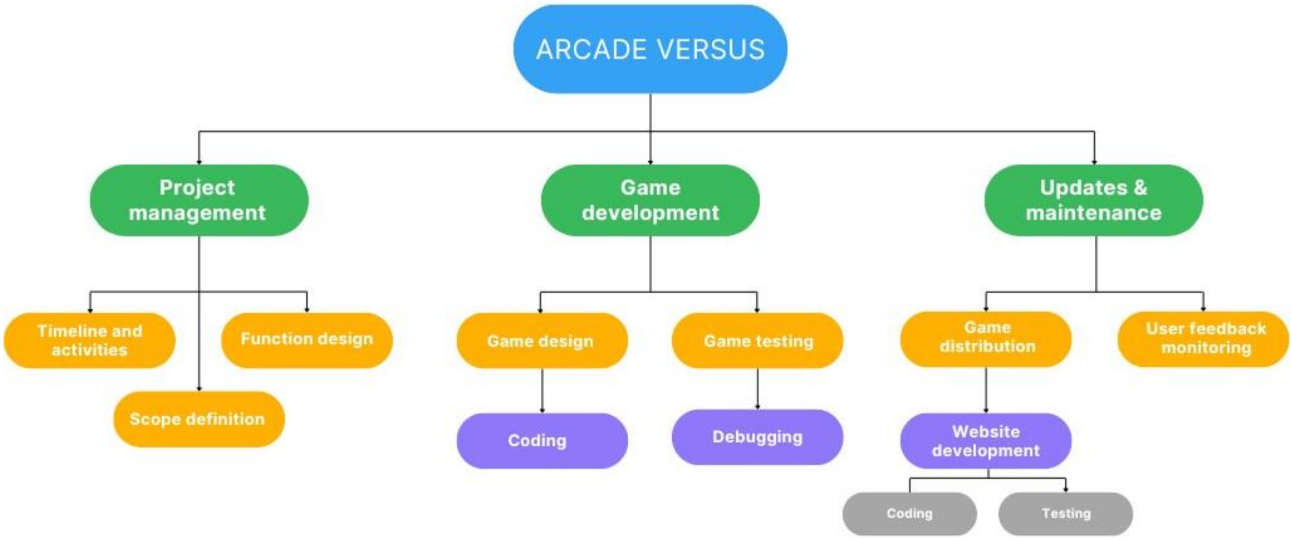




5. Project time management

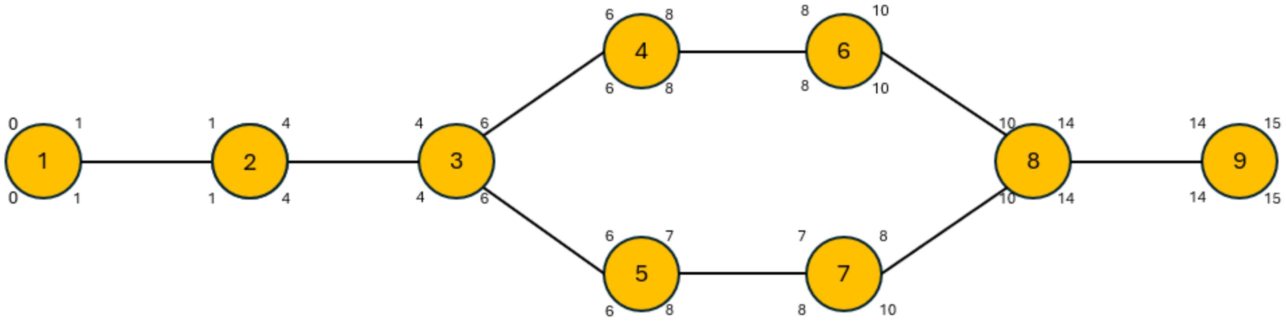
5.1 Work Break Structure of our system

In order to provide readers with a visual tool to keep track of the project development, we have defined a work break structure of our system. It contains the main tasks and activities we outlined in the roadmap, as shown in the critical path analysis shown in section 5.2.



5.2 The critical path of our system

To define the critical path of our software application we outlined a list of activities, each one consisting of one of the main tasks of our application. After we found the main activities, after studying the relationships among them, we were able to illustrate them. From the network, it is clear what the critical path and the non-critical path are (the latter has slack time, while the former has not). It is also possible to see that the first and the last activities are in common both to the critical and the non-critical path.



5.3 The slack time of each non-critical activity

By looking at the critical path network above, one can easily infer the slack time associated to each activity.

Activity ID	Description	ES	EF	LS	LF	SLACK
1	Proposal	0	1	0	1	-
2	Requirement analysis	1	4	1	4	-
3	System design	4	6	4	6	-
4	Connectivity between devices	6	8	6	8	-
5	UI design	6	7	6	8	1
6	Application coding	8	10	8	10	-
7	Web page coding	7	8	8	10	2
8	Testing	10	14	10	14	-
9	Conclusion	14	15	14	15	-

6. Summary

We started from a broad market research and user survey to define the requirements of our software application. After that, we analysed the requirements further, which permitted us to clearly define what was needed by our mobile application.

In fact, ideas that maybe were not so clearly defined before are now explicated and organized in the form of diagrams and tables, documenting in depth each part of our system, and not leaving any doubts of our future activities.

Starting from the user stories we extracted the functional and non-functional requirements needed to define each use case (the actions performed by our potential players) needed in the use case diagram. We also defined the priority matrix, which gives us an initial input of the importance of each use case to understand how to plan the project. The use case diagram is then used for the system analysis, where we define the data flow diagram (DFD) and the entity relationship diagram (ERD). The DFD shows how data is transported through our system, while the ERD depicts relationships between entities (anything that needs to store data in our system).

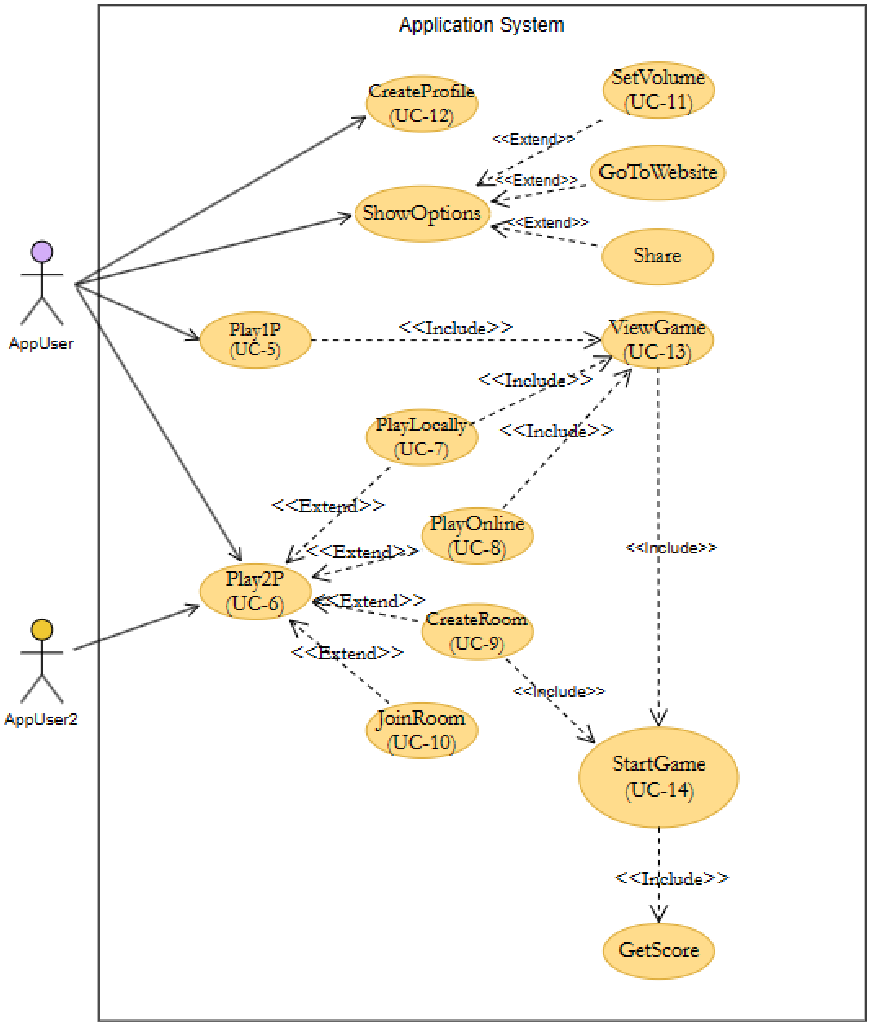
Since our users feel the need of an intuitive and responsive user experience, we also designed a user interface to cater those needs.

Finally, after defining in detail the aspects of our system we planned the path of our activities, and we show the critical path of the whole project.

Requirement design

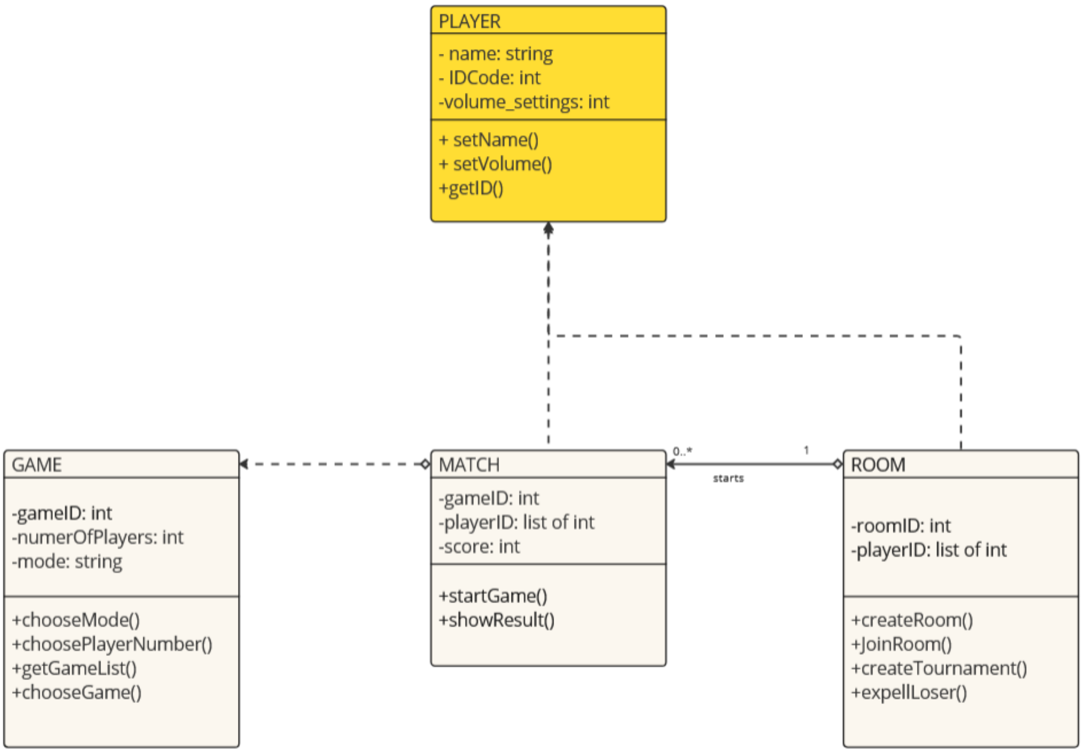
1. Design Use Cases

A further analysis in the functionalities our system provides the users with has highlighted the importance of the share button and a way to go to our website from within the mobile application. The user interface shows the existence of a panel containing the “Options”, that is the tools to modify the volume, to go to the website or to directly share the application with one’s friends. The use case diagram of the application system hence becomes as shown below.



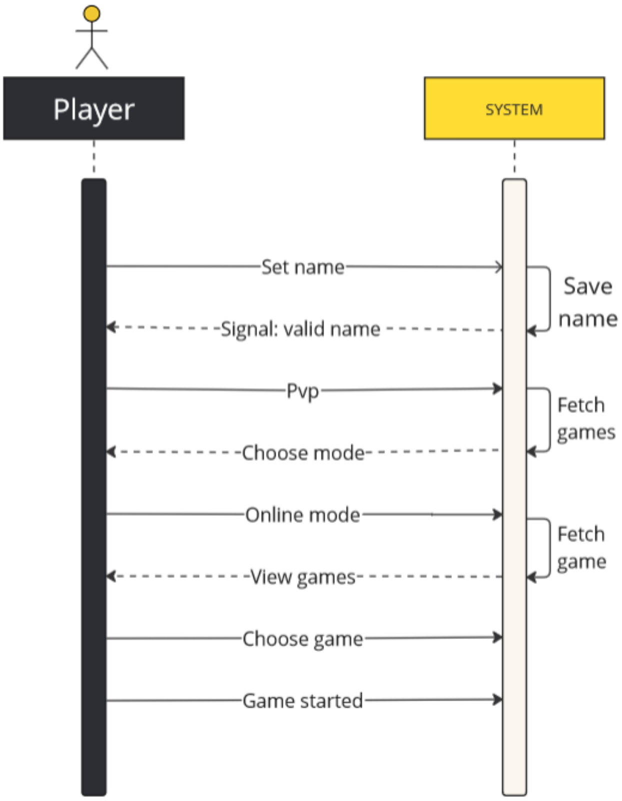
2. Design class diagram

Once the use case diagram has been refined to reflect the implementation environment, we move on to object modeling, aimed at displaying the object interactions and behavior that support the use case scenario. In the following class diagram, the software components which are used to build the software applications are represented as classes. Each class is made up of a name thanks to which it can be easily identified, a few attributes and some methods, which can be called from outside to let classes interact with one another. In addition, the dependency of one class from one or more other classes are represented using the arrows.



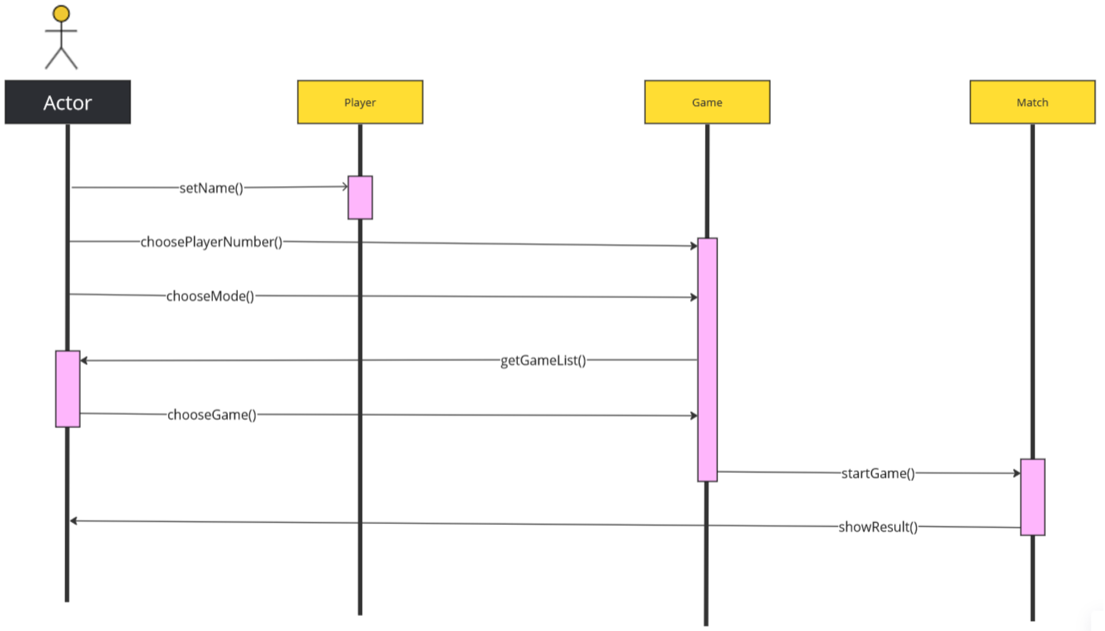
3. System Sequence Diagram

The system sequence diagram is a powerful tool as far as the interaction between actors is concerned. In fact, one can easily grasp the way any actor interacts with the system and/or with other actors. The following is a system sequence diagram displaying the information traveling from the player of our mobile application to the system, and vice versa, for a generic use case (which was called PlayOnline in our requirement analysis) in which a player executes the app for the first time and decides to play one of the games of our application.



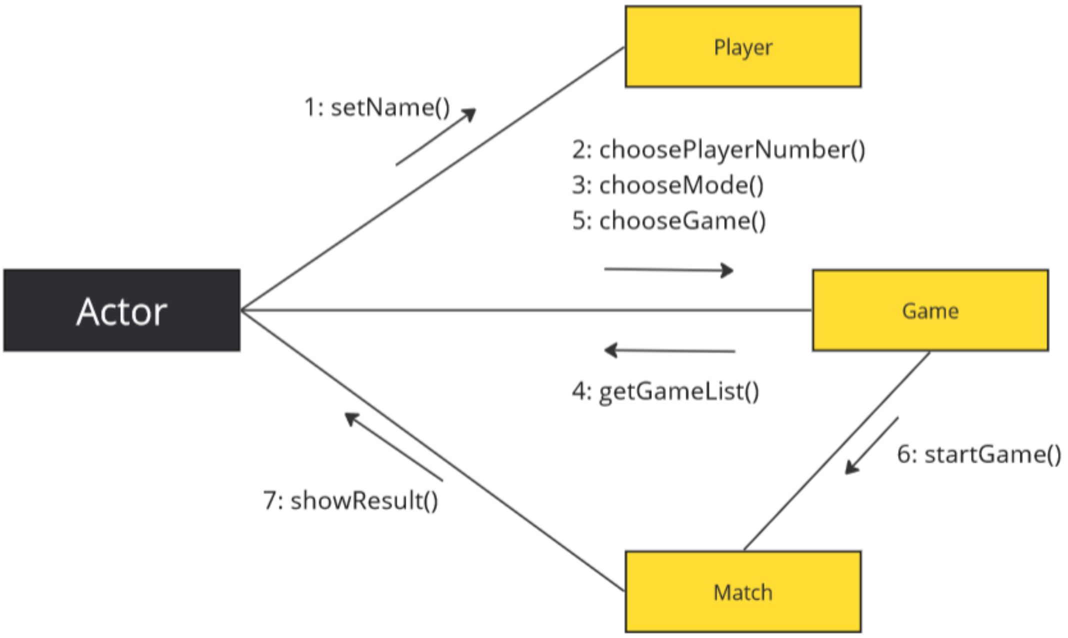
4. Sequence Diagram

The same use case analyzed in the system sequence diagram is represented below in its corresponding sequence diagram. This allows one to have a deeper understanding of how the system work. In fact, a sequence diagram shows how objects interact with one another and the order those interactions occur. The processes are represented vertically, and the interactions are depicted by arrows.



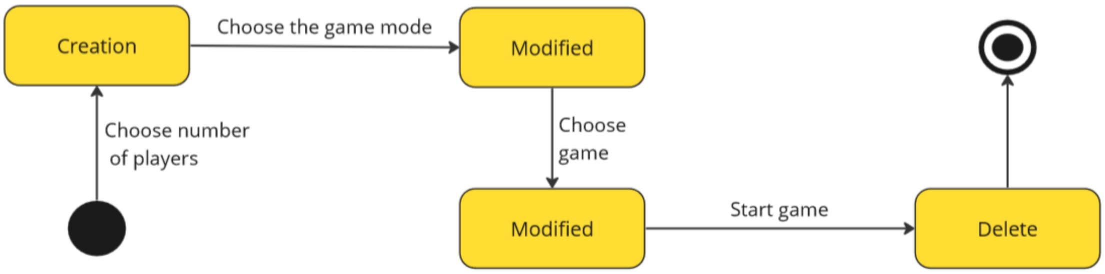
5. Collaboration Diagram

The same use case analyzed in the previous steps is now represented as a collaboration diagram. The collaboration diagram is a tool used to show the flow of messages between different objects in the system. It highlights both the order of the exchanged messages and the direction of the flow. The sequence diagram shows the interactions one after the other, resulting in a vertical representation in time. On the other hand, in the collaboration diagram we focus on the data moving from/to different modules rather than its time sequence.



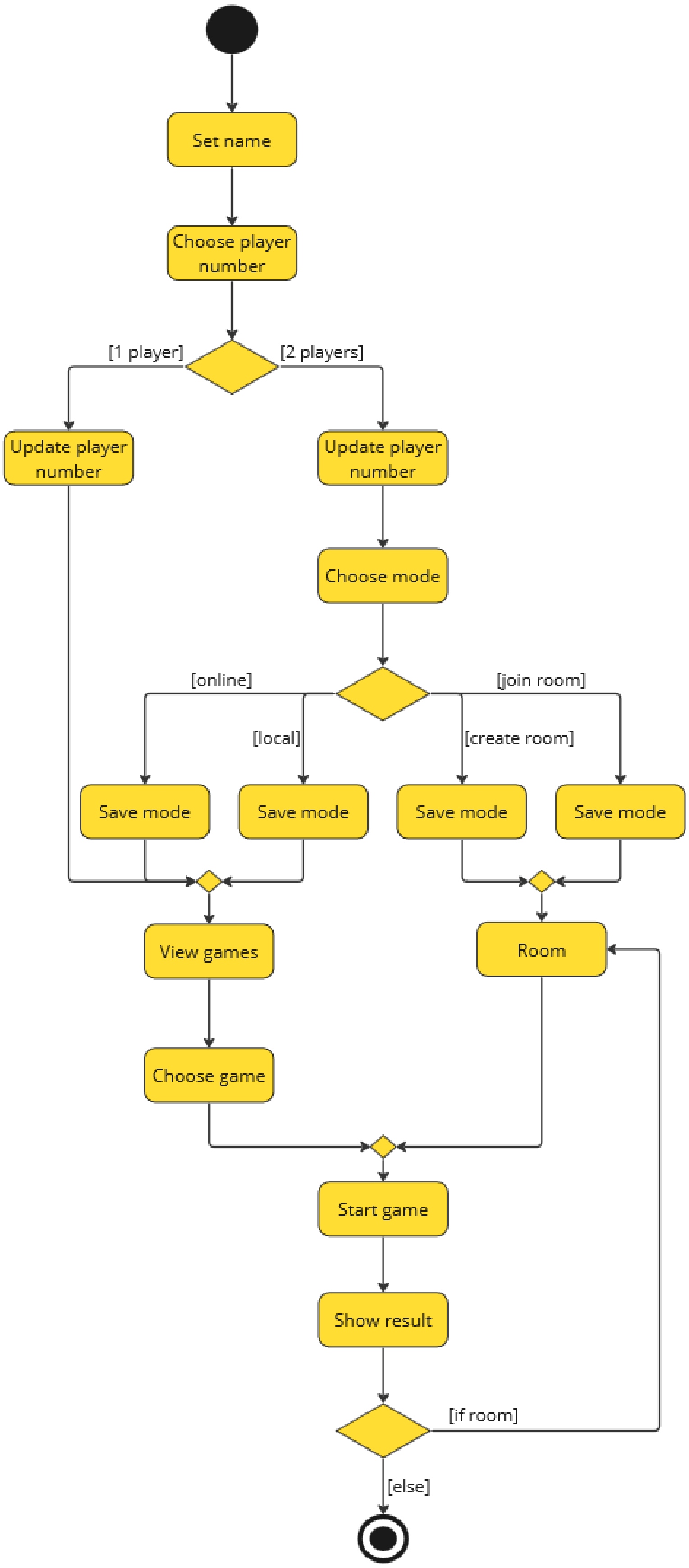
6. State chart diagram

The state chart diagram is a powerful tool to display the evolution of an object, as it depicts all the possible states of an object belonging to a certain class and the events that trigger the transitions between the states. Below is the state chart diagram of an object belonging to the Game class. Once a player chooses the number of players of the game he is about to start, a Game object is created. As soon as the player decides the game mode, it is modified (in fact, the game mode will be stored inside the object). Finally, when the player chooses the game to play, the ID of the game is stored inside the object, the game starts, and the object is deleted.



7. Activity diagram

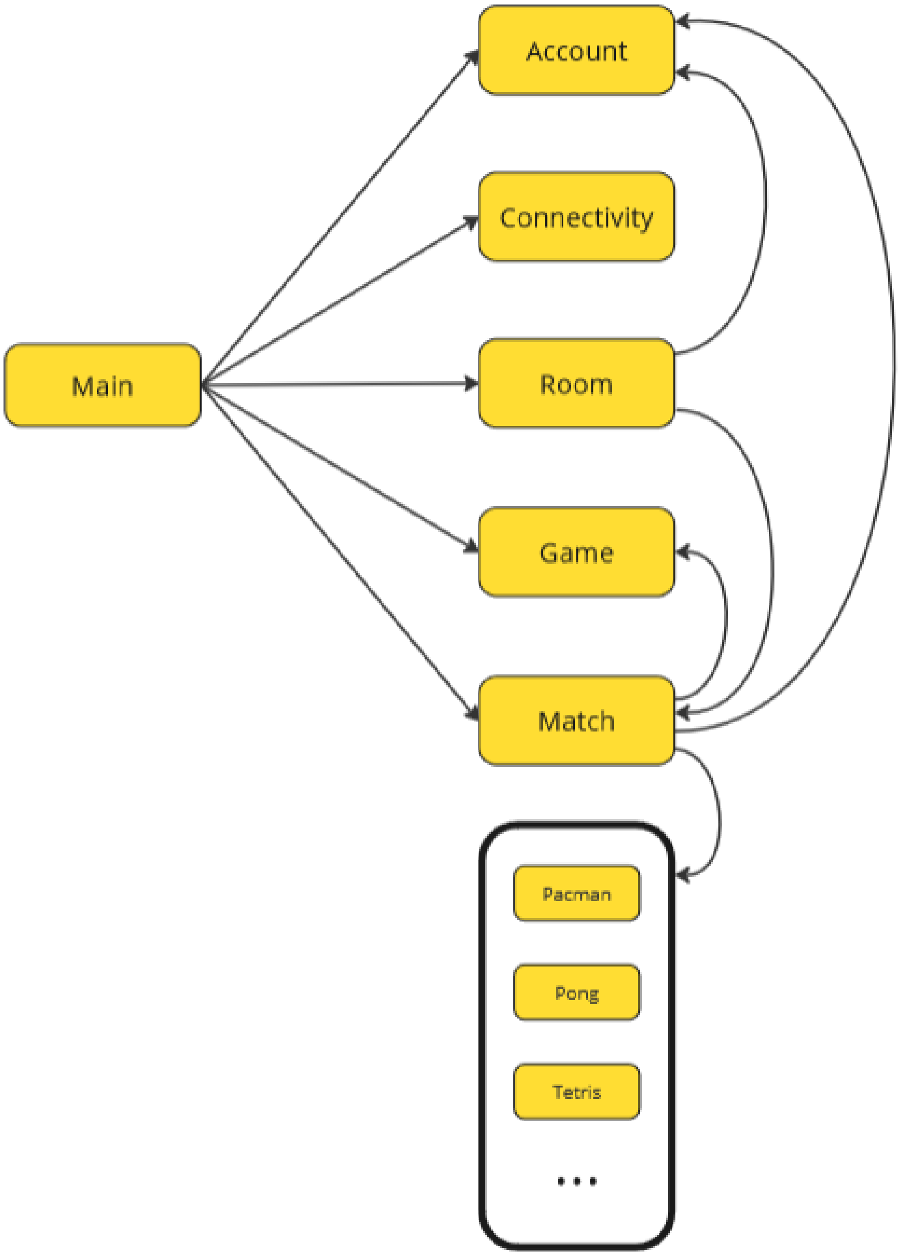
An activity diagram can be regarded as one of the most effective ways of conveying the activity flow related to one or more use cases. We show below the activity diagram of the same PlayOnline use case analyzed in the previous sections of this document. After the player has chosen a name, he is prompted to select the number of players of the game he is going to play. According to this choice, the left or right branches of the activity diagram are chosen. If he wants to play against another player, then he will have to choose one among the multiplayer options our app provides, which will then be stored in a Game object. Then, the list of the suitable games is shown to the user, who will have to choose one and start playing.


Set name
Choose player number
[1 player]
[2 players]
Update player number
Update player number
Choose mode
[online]
[local]
[create room]
[join room]
Save mode
Save mode
Save mode
Save mode
View games
Room
Choose game
Start game
Show result
[if room]
[else]

8. Component Diagram

A component diagram is used to graphically implement the physical architecture of our software application. It can be used to show how programming code is divided into modules (or components) and the dependencies between those modules.

Because our mobile application offers multiple games, we need to include game files for each one of them. To avoid having to specify the connections between the Match module and every game file, we show that a database containing our game files is connected to the Match module instead. Also, we added a connectivity module to support the online connectivity between players. This module is called in the main module, and it is the basis for game modes such as creating tournaments, and online PVP matches.



9. Summary

Last time we concluded the requirement analysis of our system, and now we are interested in defining the system design (object-oriented analysis), with the goal of studying the objects of our mobile application and the relationships between them.
We started by refining the use-case model, which now also reflects our implementation environment. To do so, we added additional use cases to describe new characteristics of our system.
After that, we modelled the class diagram to show how objects interact with each other and to specify the attributes and methods that distinguish them.
Furthermore, for the specific use case 'PlayOnline' we also designed three diagrams: the system sequence diagram to show actor interactions in time sequence, the sequence diagram to show software object interactions in time sequence and finally the collaboration diagram to depict the flow of messages between objects in message sequence.
We also focused our attention on a specific class of our system, the 'game' class, and defined the state chart diagram, which is a diagram that shows the state transitions triggered by certain events in our system.
Finally, we update the object model to also reflect our implementation environment by realizing the activity diagram and the component diagram.

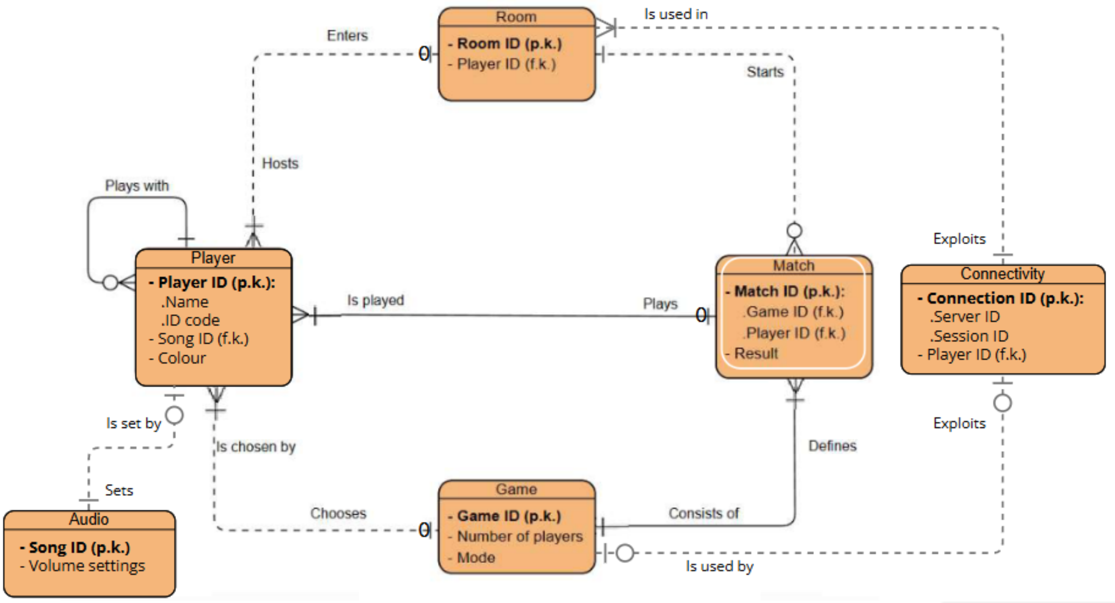
This was a good experience because it allowed us to translate what we already know from the requirement analysis into an actual framework for our project, which takes into consideration how we are going to organize our code in the chosen implementation environment. This will help us to be more efficient when we'll start coding and avoid unwanted doubts and misunderstandings due to poor prior knowledge on the

Product testing report

1. Data design

1.1 Refined ERD of our system

During the practical implementation of our design in the programming environment, we have found out that there were some things we had overlooked while defining the initial entity relationship diagram in the requirement design part. Here is shown a new detailed version of the ERD of our project. The main changes have been made by adding a Audio entity and a Connectivity entity, along with their primary keys and their attributes.



1.2 Relational Model

By looking at the refined Entity Relationship Diagram, we can easily infer the relationships and dependencies of the entities of our system. Below, one can find the relational model of our system.

Player (Name, IDcode, Audio.SongID, Colour, Room.RoomID, Game.GameID)
Audio (SongID, VolumeSettings)
Room (RoomID, Player.Name, Player.IDcode, Connectivity.ServerID, Connectivity.SessionID)
Game (GameID, NumberOfPlayers, Mode, Connectivity.ServerID, Connectivity.SessionID)
Match (Game.GameID, Player.Name, Player.IDcode, Result, Room.RoomID)
Connectivity (ServerID, SessionID, Player.Name, Player.IDcode)

1.3 SQL code to create the tables

```
CreateTable Player
(
        Name VARCHAR(20) NOT NULL,
        IDcode INTEGER NOT NULL UNIQUE,
        SongID INTEGER UNIQUE,
        GameID INTEGER NOT NULL UNIQUE,
        RoomID INTEGER NOT NULL UNIQUE,
        Colour INTEGER NOT NULL
        Constraint fk_SongID ForeignKey (SongID)
                References Audio (SongID)
        Constraint fk_GameID ForeignKey (GameID)
                References Game (GameID)
        Constraint fk_RoomID ForeignKey (RoomID)
                References Room (RoomID)
        PrimaryKey (Name, IDcode)
)
```

```
CreateTable Audio
(
        SongID INTEGER NOT NULL UNIQUE,
        VolumeSettings FLOAT NOT NULL,
        PrimaryKey (SongID)
)
```

```
CreateTable Room
(
        RoomID INTEGER NOT NULL UNIQUE,
        Name VARCHAR(20) NOT NULL,
        IDcode INTEGER NOT NULL UNIQUE,
        ServerID VARCHAR(30) NOT NULL UNIQUE,
        SessionID INTEGER NOT NULL UNIQUE,
        Constraint fk_Name ForeignKey (Name)
                References Player (Name)
        Constraint fk_IDcode ForeignKey (IDcode)
                References Player (IDcode)
        Constraint fk_ServerID ForeignKey (ServerID)
                References Connectivity (ServerID)
        Constraint fk_SessionID ForeignKey (SessionID)
                References Connectivity (SessionID)
        PrimaryKey (RoomID),
)

CreateTable Game
(
        GameID INTEGER NOT NULL UNIQUE,
        NumberOfPlayers INTEGER     NOT NULL,
        Mode VARCHAR(20) NOT NULL,
    ServerID VARCHAR(30) NOT NULL UNIQUE,
        SessionID INTEGER NOT NULL UNIQUE,
        Constraint fk_ServerID ForeignKey (ServerID)
                References Connectivity (ServerID)
        Constraint fk_SessionID ForeignKey (SessionID)
                References Connectivity (SessionID)
        PrimaryKey (GameID)
)
```

```
CreateTable Match
(
        Name VARCHAR(20) NOT NULL,
        IDcode INTEGER NOT NULL UNIQUE,
        GameID INTEGER NOT NULL UNIQUE,
        Result INTEGER,
    Room ID INTEGER,
    Constraint fk_Name ForeignKey (Name)
                References Player (Name)
        Constraint fk_IDcode ForeignKey (IDcode)
                References Player (IDcode)
    Constraint fk_GameID ForeignKey (GameID)
                References Game (GameID)
    Constraint fk_RoomID ForeignKey (RoomID)
                References Room (RoomID)
        PrimaryKey (Name, IDcode, GameID)
)

CreateTable Connectivity
(
        ServerID VARCHAR(30) NOT NULL UNIQUE,
        SessionID INTEGER NOT NULL UNIQUE,
    Name VARCHAR(20) NOT NULL,
        IDcode INTEGER NOT NULL UNIQUE,
    Constraint fk_Name ForeignKey (Name)
        References Player (Name)
    Constraint fk_IDcode ForeignKey (IDcode)
        References Player (IDcode)
    PrimaryKey (ServerID, SessionID)
)
```

2. Testing report

2.1 Two test cases

From the traceability matrix we had created while carrying out the requirement analysis of our system, which we have enhanced to make it more suitable for our testing purposes (as shown at the end of this report), we have selected what we agreed on being two of the most important test cases to verify the correct execution of one of the functional requirements of our application, namely the play online functionality and the Connect tournament.
The former test was carried out by test engineer Giordano. The results he has retrieved are as follows:

Test case ID	TC-8	**Test Engineer**	Andrea Giordano
Product module	PlayOnline	**Testing date**	26/05/2024
Product Version	1.0.4	**Testing cycle**	1
Revision history	/	**Status**	Approved
Purpose			To verify that two app users, after choosing the same game to play online, can actually connect and have fun together playing.
Assumptions			The user has a working internet connection.
Preconditions			The application is downloaded on the user Android device, the user has set a username.
Steps to reproduce			User from the “mainMenu” loads the “pvpMenu” and chooses “PlayOnline” and from the proposed list of games, chooses the games he/she wants to play.
Expected result			The system loads a “loading” page, establishes a connection with the Photon master server and then the game scene is shown to the player, and the game won’t start until another player connect to the Photon server.
Actual outcome			The system first displays a "loading" page and connects to the Photon master server. After establishing the connection, the game scene is presented to the player. The game won’t begin until another player connects to the Photon server.
Post conditions			The score is shown to both the players, so it is possible to see who the winner and the loser are. The connection to the Photon server is terminated and the user redirected to the “mainMenu” scene.

The latter test was executed by engineer Huang. Here are her results:

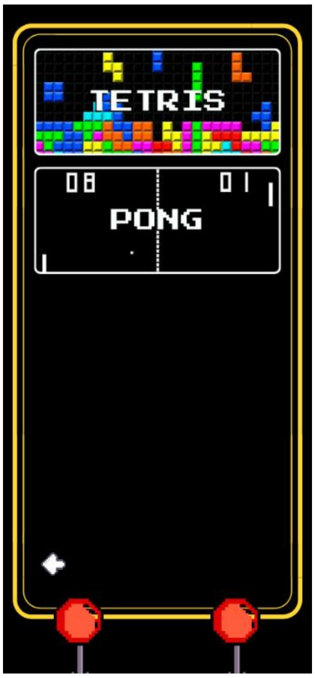
Test case ID	TC-11	**Test Engineer**	Jie Huang
Product module	Connect Tournament	**Testing date**	26/05/2024
Product Version	1.0.4	**Testing cycle**	1
Revision history	/	**Status**	Pending
Purpose	To verify that a room can be created, can be joined by a group of people and a tournament can be organized.		
Assumptions	Multiple users have a working internet connection.		
Preconditions	The application is downloaded on the all the user Android devices, the users have set a username.		
Steps to reproduce	All the users from the "mainMenu" load the "pvpMenu".		
Expected result	One user chooses "CreateRoom" and the others "JoinRoom", using as code number what is outputted on the screen of the room's creator page. Once everyone is inside the room, the creator clicks the button "Play" to create and start the tournament.		
Actual outcome	The room is correctly created, and it is joinable for all the users that have the room number. Once the creator clicks on the "Play" button the room's creator (MasterClient) correctly sees the game scene, the other players get an error message, and the game scene is not loaded (though for what concerns the Photon server they are connected and have started playing).		
Post conditions	The score is shown to both the players, so it is possible to see who the winner and the loser are. The players that lose the game are then expelled from the room and are disconnected from the server, while the other players continue with the tournament until only two players remain, and the winner will be the winner of the entire tournament.		

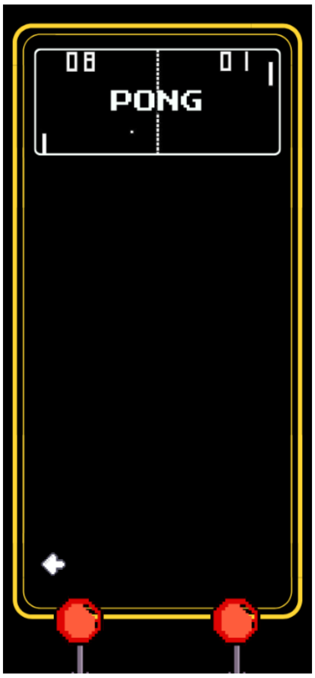
2.2 Descriptions of our modifications after the ‘Testing Meeting’

We met with the other group for the “Testing Meeting”, and they gave us useful feedback aimed at making the gaming experience of our mobile application smoother. As they tried the game for the first time, they had complaints about the clarity of the user interface for the list of games. It wasn’t that clear where the buttons to choose a specific game were, therefore, we plan to add some sort of frame to the game icons to show where the game buttons are located.
Next, they encountered some issues when trying different games (“Tetris” and “Pong”) since there isn’t an option to leave the match once the game starts. We decided that we will add buttons that go back to the main menu, so that the player doesn’t have to wait until the end of the match.
Furthermore, our testers also tried the “CreateRoom” functionality, but that leads to additional problems since the connection is not ended whenever the player decides to leave the room, making it impossible for them to start other matches in other game modes. That will be an easy fix, since we just need to tell the Photon server to disconnect the player if they decide to leave the room.
Additionally, we saw that the paddle for the second player in “Pong2POnline” doesn’t work properly, so in our testing phase we plan to fix this issue and debug other games to make the gaming experience more polished.

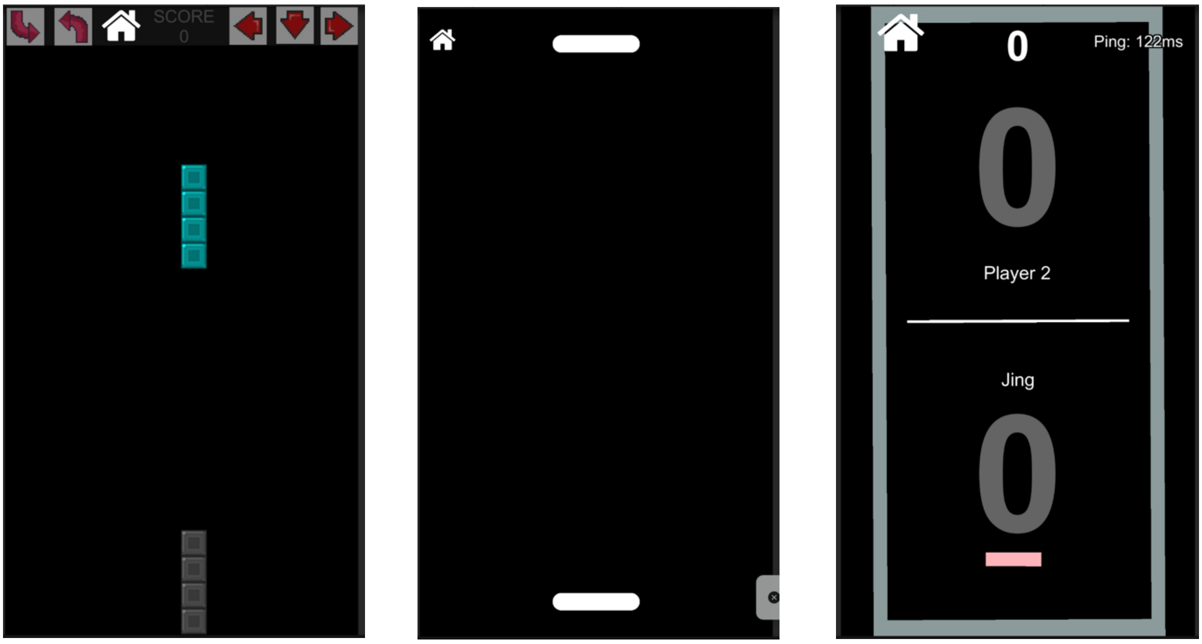
UI modifications after testing:

Frame to contain the game icons

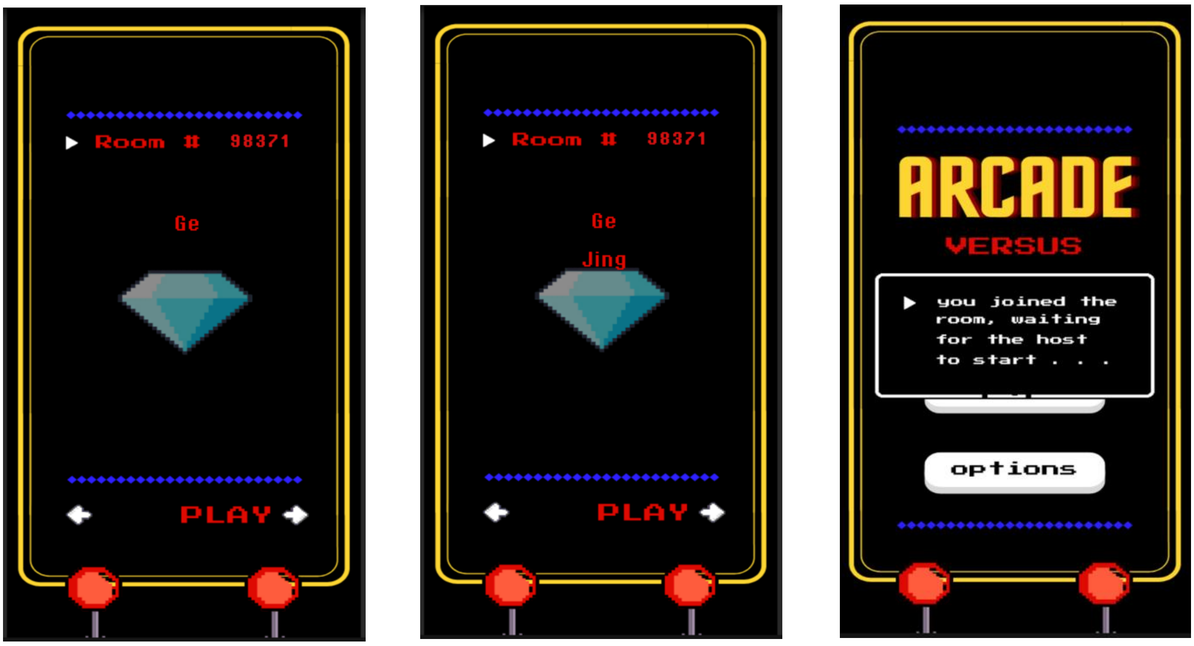




Home button added to games:



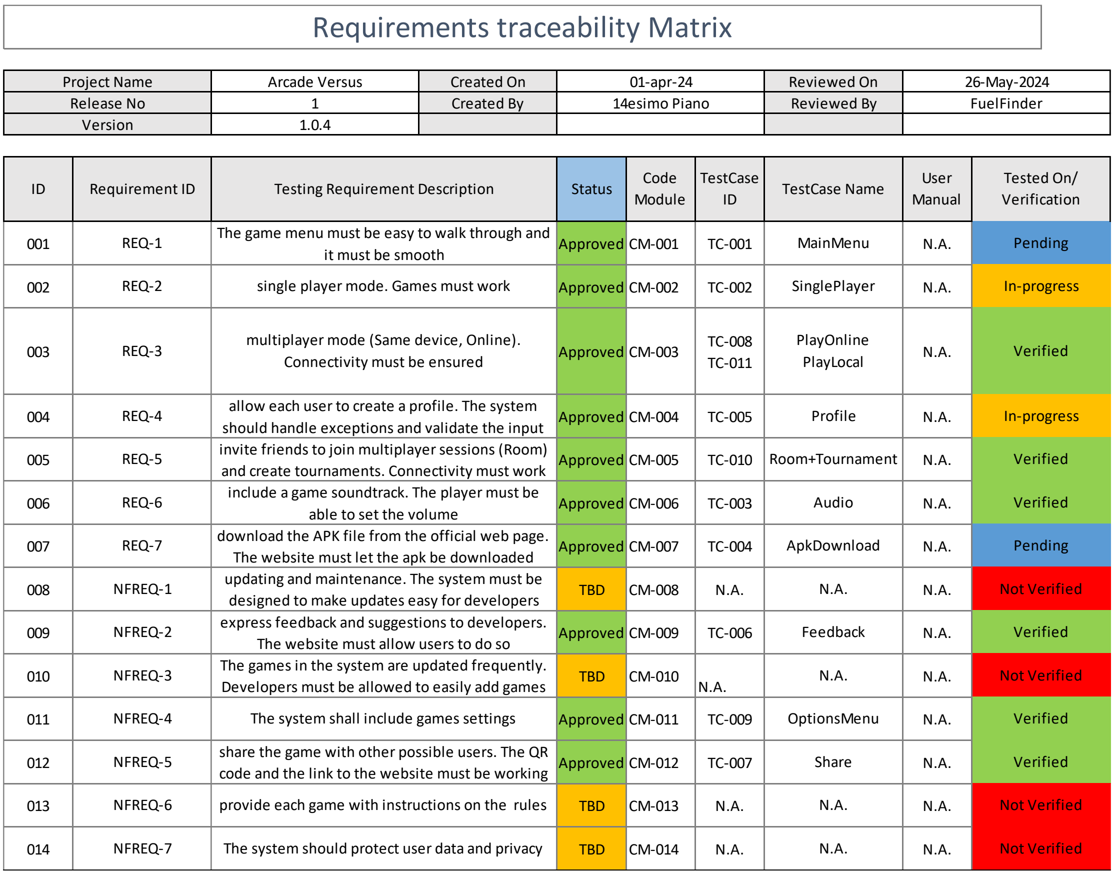
What the host sees when entering the room/when other players join/what the other players see:



you joined the room, waiting for the host to start . . .

3 Traceability matrix

In order for us to easily keep track of the progress of our testing activities, we have taken the traceability matrix we had created during the first phases of the requirement analysis to ensure that all the use cases were mapped to one or more functional or non-functional requirements. Here, we have modified said traceability matrix so as to make it more suitable for our testing activities. Along with the requirement ID, one can find the testing requirement description, the status of the testing, the code module, the test case ID and its corresponding name and whether the test case has already been carried out or it is pending, in progress or not verified.



Requirements traceability Matrix

Project Name	Arcade Versus	Created On	01-apr-24	Reviewed On	26-May-2024
Release No	1	Created By	14esimo Piano	Reviewed By	FuelFinder
Version	1.0.4				

ID	Requirement ID	Testing Requirement Description	Status	Code Module	TestCase ID	TestCase Name	User Manual	Tested On/ Verification
001	REQ-1	The game menu must be easy to walk through and it must be smooth	Approved	CM-001	TC-001	MainMenu	N.A.	Pending
002	REQ-2	single player mode. Games must work	Approved	CM-002	TC-002	SinglePlayer	N.A.	In-progress
003	REQ-3	multiplayer mode (Same device, Online). Connectivity must be ensured	Approved	CM-003	TC-008 TC-011	PlayOnline PlayLocal	N.A.	Verified
004	REQ-4	allow each user to create a profile. The system should handle exceptions and validate the input	Approved	CM-004	TC-005	Profile	N.A.	In-progress
005	REQ-5	invite friends to join multiplayer sessions (Room) and create tournaments. Connectivity must work	Approved	CM-005	TC-010	Room+Tournament	N.A.	Verified
006	REQ-6	include a game soundtrack. The player must be able to set the volume	Approved	CM-006	TC-003	Audio	N.A.	Verified
007	REQ-7	download the APK file from the official web page. The website must let the apk be downloaded	Approved	CM-007	TC-004	ApkDownload	N.A.	Pending
008	NFREQ-1	updating and maintenance. The system must be designed to make updates easy for developers	TBD	CM-008	N.A.	N.A.	N.A.	Not Verified
009	NFREQ-2	express feedback and suggestions to developers. The website must allow users to do so	Approved	CM-009	TC-006	Feedback	N.A.	Verified
010	NFREQ-3	The games in the system are updated frequently. Developers must be allowed to easily add games	TBD	CM-010	N.A.	N.A.	N.A.	Not Verified
011	NFREQ-4	The system shall include games settings	Approved	CM-011	TC-009	OptionsMenu	N.A.	Verified
012	NFREQ-5	share the game with other possible users. The QR code and the link to the website must be working	Approved	CM-012	TC-007	Share	N.A.	Verified
013	NFREQ-6	provide each game with instructions on the rules	TBD	CM-013	N.A.	N.A.	N.A.	Not Verified
014	NFREQ-7	The system should protect user data and privacy	TBD	CM-014	N.A.	N.A.	N.A.	Not Verified

4 Summary

At first, we refined the ERD by adding new entities that we realized were needed in our system. Those new entities are the “Audio” and “Connectivity” entities, which are needed in order to cover certain functionalities of our system such as setting the volume of the background music and allow for connection between players. The connection is finalized by connecting to an external server and supports both the online connection for the game mode “Play Online” and for the tournament organization once the players join a room together.

After that, we wrote down both the relational model as well as the create tables for our database definition. A database is used because, as data flows in the system, there is a need to store it somewhere and access it whenever needed.

The second part consisted of testing our application and for that we’ve shown two different test cases of our system. One successful test case concerning the game mode “Play Online”, while the other test case is still pending due to unresolved issues with our code and concerns the tournament creation from a room of players.

With the feedback that we extracted from the user testing meeting, we applied some changes to our system to reflect our users’ needs.

Finally, we show our traceability matrix, which kept track of the testing process, and shows which user requirements are satisfied in our application and whether the testing of those requirements was successful or not.

GitHub

Our application is fully open source and available for everyone who is interested in looking at the code. There is a repository on GitHub that contains all the resources of our mobile application: codes, UI images, Unity scenes and connectivity settings. In the file ReadME.md one can find more information about our aim and our reality.

A feature that can be interestingly carried out using the GitHub platform is to allow app users (or more in general) whoever has an interest in trying to add new games to our application. Designing games is a time-consuming task and, before launching our app, we were able to finish only four different games, of which only one is an online game. To become a successful game application and beat the possible competitors, it is important for the game selection to be wide and frequently updated.

On our website, reachable both from the mobile application and the GitHub page, there is an option to contact the developers. By reaching us who is interested in trying to design a new arcade game can have more detail about how the application is designed, which tool have been used and how to best deal with the connectivity within different players (in the case they are interested in making a multiplayer game).

When someone begins programming a new arcade game for our application, the programmer/user should create a GitHub account, so that to him it is possible to make a pull request on our GitHub page. Doing so, not only his work would be visible to us, but also to all the other app users that are redirected to our GitHub. That means that other users can help him integrate new functions in his game. Once the game is complete and is correctly working, let us know that it is done, we (the developers) can merge the work of the programmer/user to the main (principal branch of the GitHub repository). After merging the branch to the main, it is necessary to create a new APK file, to upload on our website; in such a manner the new game becomes visible to all the users that download the new APK file.

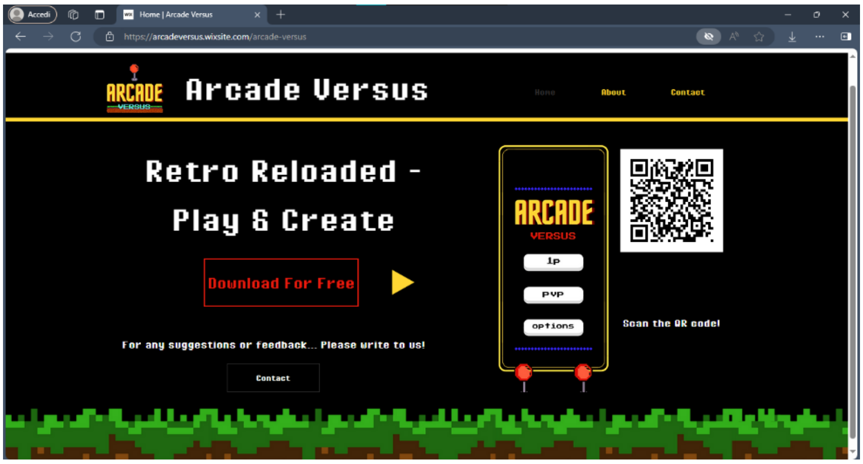
User manual

Great effort was made in making the application as user friendly as possible. In order to further increase the simplicity a user can play our game with, this section of the software documentation aims at providing readers with a basic yet thorough introduction on the app and website functionalities.

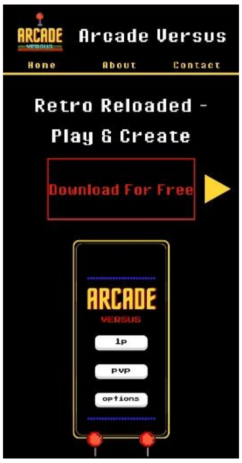
1 Website user manual

Our website can be found at https://arcadeversus.wixsite.com/arcade-versus.
It has been developed using online tools provided by wix.com.

2.1 Home

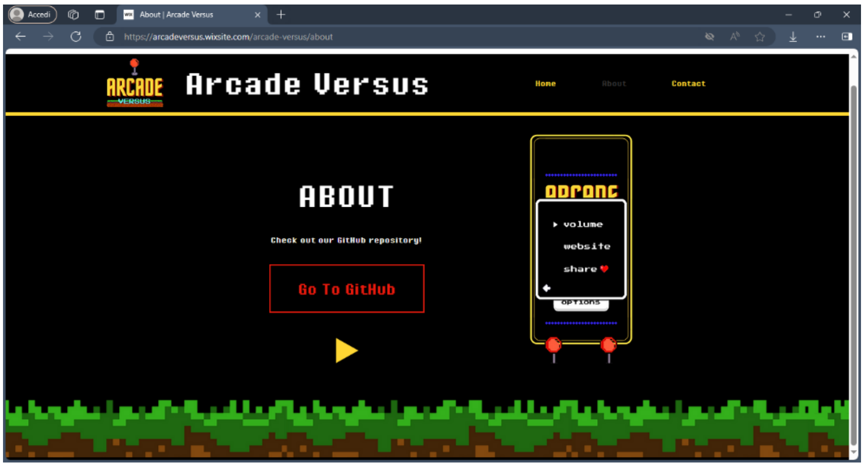


For any suggestions or feedback... Please write to us!



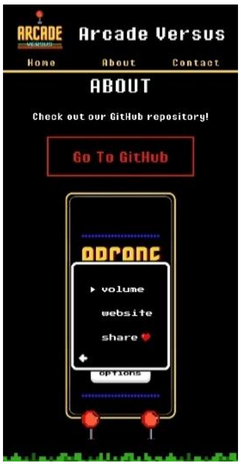
The home page of our website contains a horizontal menu in the heading of the website, which allows users to navigate the website. In the home page users are welcomed with the slogan of the project, as well as a screenshot of the UI of the mobile application.
The QR code can be easily scanned from a mobile device to allow a user to conveniently download the apk file directly on their smartphone, where it can be installed. The download operation is triggered by the click on the main red button ‘Download for free’.
The yellow triangular button can be clicked to play the game soundtrack while surfing the website.
The ‘contact’ button at the bottom of the page redirects the user to the contact page of the website.

2.2 About



ABOUT

Check out our GitHub repository!

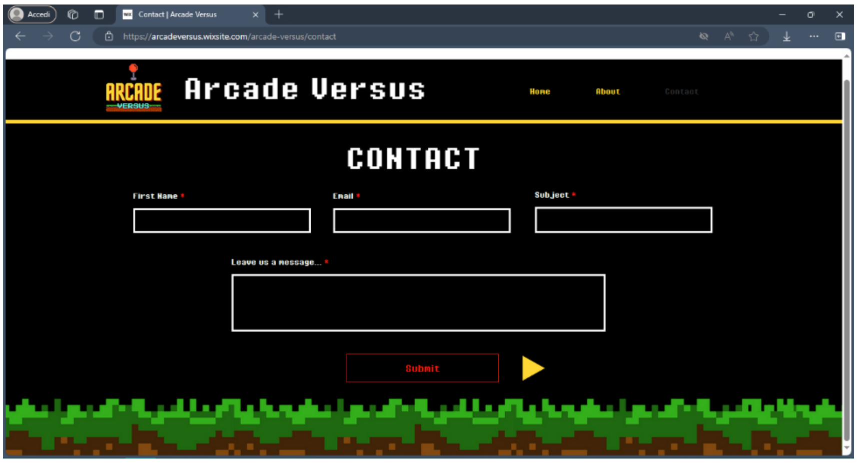


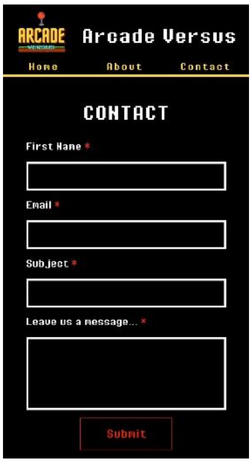
ABOUT

Check out our GitHub repository!

The about page displays another screenshot of the mobile application and a red ‘Go To GitHub’ button which redirects the user to our official GitHub repository, described in the ad hoc section of this document.

2.3 Contact





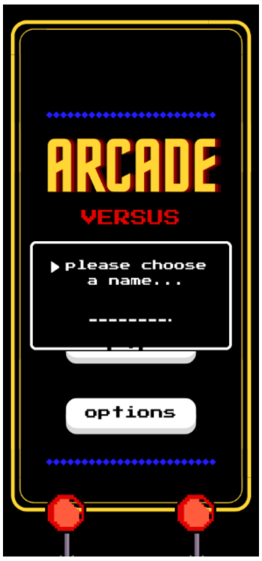
The Contact page lets users leave feedback and suggestions to the game developers. The fields one must fill before submitting the feedback are Name, Email address, Subject of the message and the message itself. After the message is sent, a confirmation message appears on the screen to inform the user about the positive outcome of the feedback submission.

The contact form can be used to leave feedback, complaints, suggestions or to get in touch with the developers for partnership purposes.

2 Mobile application user manual

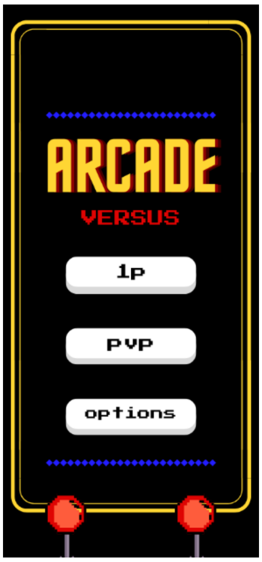
Our application has been developed in the Unity environment. It has been entirely coded in C#. For the adopted coding standard, check out our GitHub repository. For the online connectivity (both for the online session and the rooms) a third-party server, provided by Photon Engine, has been exploited. Photon is a professional tool, with various master servers in different geographical regions, to provide a service with as little delay as possible to the clients. It is possible to manage directly in the C# the connectivity and the synchronization between players.

2.1 Username Selection Screen



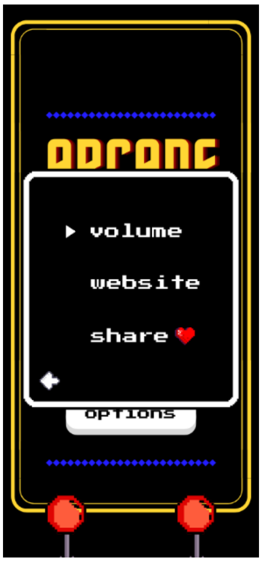
When the application is launched for the first time, the username selection screen is loaded. The user is prompted to choose a nickname which will be shown in the online games against other players. Once the user has finished writing their username, they can click the 'ok' button next to the input field and proceed to the Main Menu. The username cannot be changed, so choose the name that suits you best!

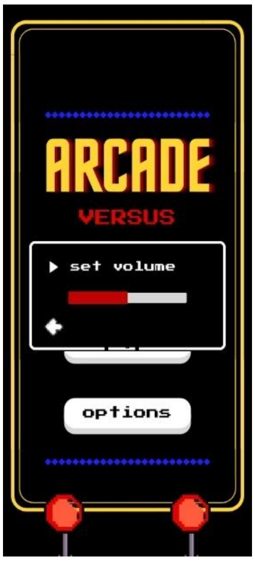
2.2 Main Menu

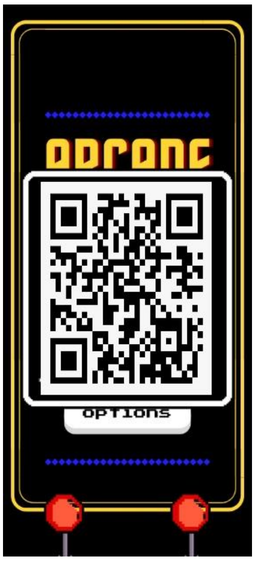


The main menu looks as shown in the figure on the left. The player can choose to go to the options menu, to start a single player game by clicking the '1p' button or a multiplayer match ('pvp' button).

2.3 Options Menu

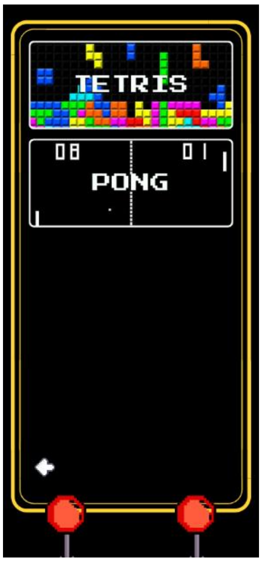






Within the Options Menu, the player can set the volume of the music to best match their needs, by moving the level of the red slider which can be seen in the picture in the middle above. Also, users can click on the 'website' button to let the application load our website home page in the default browser on the mobile device. Finally, the 'share' button allows users to see the QR code of the link of our website on their screen to easily let their friends go on the arcadeversus website and download the game. In the Options Menu, a click on the arrow-shaped button will let the user go back to the Main Menu. From the QR code screen, the user can simply click outside the white box to go back to the Main Menu.

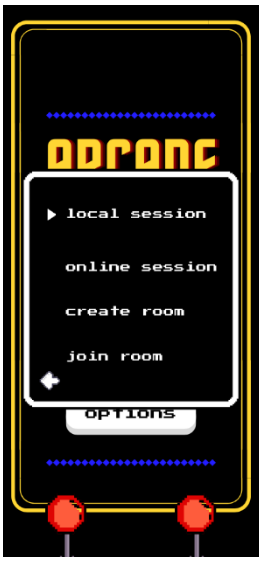
2.4 1p Game Selection Screen



Once the player has chosen to play a single player game by clicking on the '1p' button in the Main Menu, the list of single player games appears on screen. Once the user has made their mind up about which game to play, a click on the corresponding image determines the begin of a new match. The arrow shaped button can be clicked to go back to the Main Menu.

The 1p games provided by our application are Tetris and Pong. In the 'Games' section of the user manual, the game mechanics are described and the user is given some tips about how to play.

2.5 PvP Mode Selection Screen

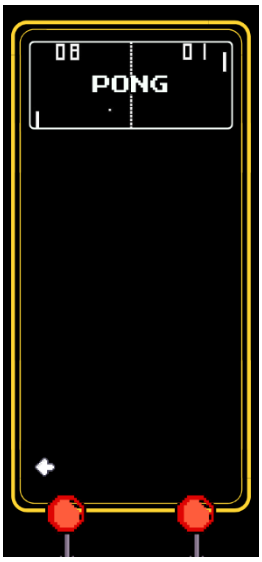


Our mobile application shows a great variety of multiplayer games. The menu shown on the left can be accessed form the Main Menu by clicking on the 'pvp' button. Here the user can decide whether to play:

- locally, i.e. to play with a friend of theirs on the same device,
- online, i.e. against a random player from all around the world,
- in a room with friends.

These game modes will be better analysed in the following sections of the user manual. The arrow shaped button lets the user go back to the Main Menu.

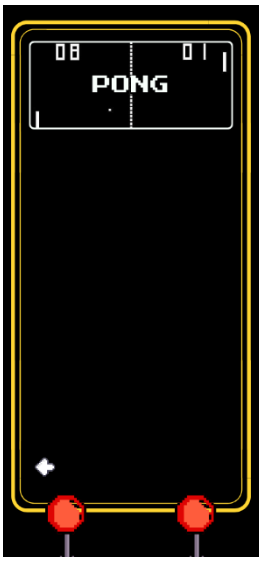
2.6 PvP local session



Once the player has chosen to play a multiplayer game on a local device (i.e. two players compete against each other on the same device), by clicking on the 'local session' button in the PvP Mode Selection Screen, the list of suitable games appears on screen. Once the user has made their mind up about which game to play, a click on the corresponding image determines the begin of a new match. The arrow shaped button can be clicked to go back to the PvP Mode Selection Screen.

The local PvP game provided by our application is Pong. In the 'Games' section of the user manual, the game mechanics are described and the user is given some tips about how to play.

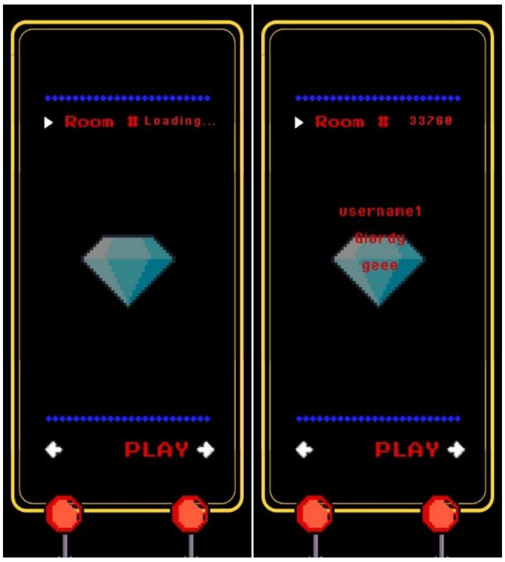
2.7 PvP online session



Once the player has chosen to play an online multiplayer game (i.e. two players compete against each other on different devices connected through the internet), by clicking on the ‘local session’ button in the PvP Mode Selection Screen, the list of suitable games appears on screen. Once the user has made their mind up about which game to play, a click on the corresponding image determines the begin of a new match. The arrow shaped button can be clicked to go back to the PvP Mode Selection Screen.

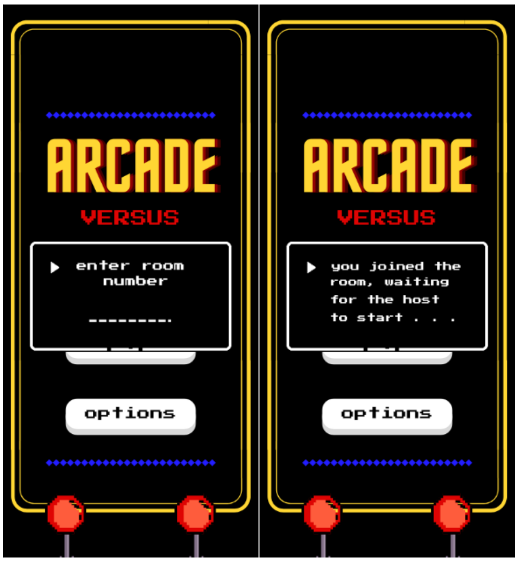
The online PvP game provided by our application is Pong. In the ‘Games’ section of the user manual, the game mechanics are described and the user is given some tips about how to play.

2.8 Room Creation Screen



Once the player has chosen to create a room by clicking the ‘create room’ button in the PvP Mode Selection Screen, the game will try to connect to an external server and a “#Loading…” text is displayed. As soon as the room is set up, the room number is shown in the top right corner of the screen. The user can share the room ID with their friends to let them enter the room. In the centre of the screen, a list of the players in the room appears. The arrow button pointing to the left can be used to delete the room and go back to the PvP Mode Selection Screen; while the arrow pointed to the right can be used to start a random tournament among the players in the room.

2.9 Join Room Screen



enter room number

----

you joined the room, waiting for the host to start . . .

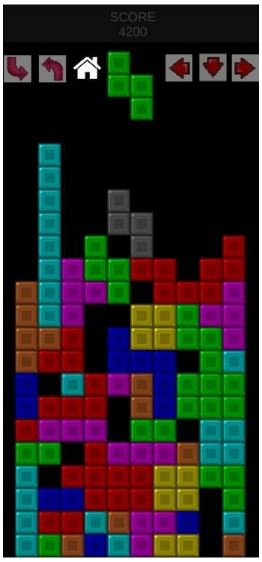
Once the player has chosen to join a room created by a friend by clicking the ‘join room’ button in the PvP Mode Selection Screen, the user will be prompted to write the room ID shared by their friend. Once the game has successfully made a connection with the external server and the room has been joined, a new screen appears, as shown on the left. If the host of the room deletes the room, all the guest players are automatically disconnected from it.

These screens can be exited by clicking outside the white rectangles.

3 Games

Here the game mechanics and the basic player inputs of the games of our application are described.

3.1 1p Tetris



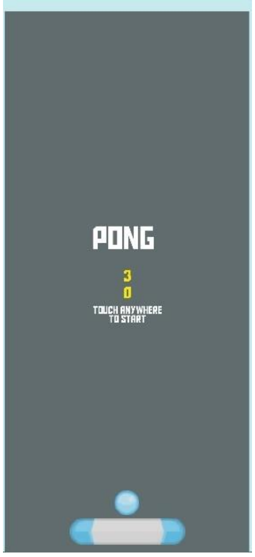
Game mechanics:

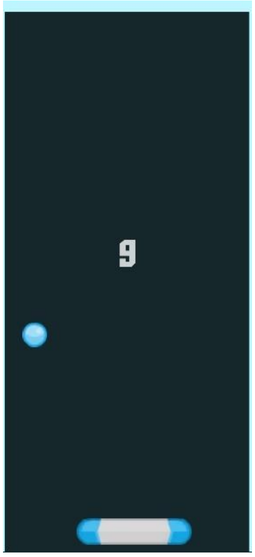
Tetrominos fall one by one from the top-centre of the screen. Players should use the arrows to place the falling piece in such a way to complete one or more horizontal lines. When a horizontal line is complete, it will disappear and the blocks placed on it will move down by one line. The score, shown in the top part of the screen, is incremented by 100 every time a row is completed. The game ends when a new tetromino can’t move downwards as soon as it is created.

How to play:

Use the arrows to move the falling tetromino leftwards, rightwards, downwards. Use the curved arrows to rotate the tetromino clockwise or counterclockwise. Prevent empty slots from being created between new rows!

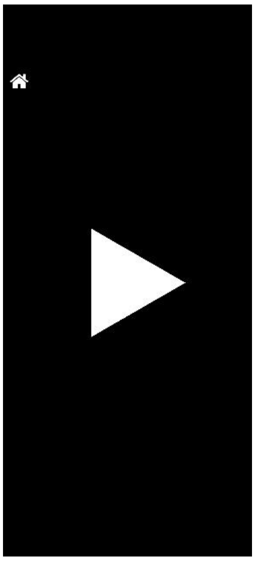
3.2 1p Pong

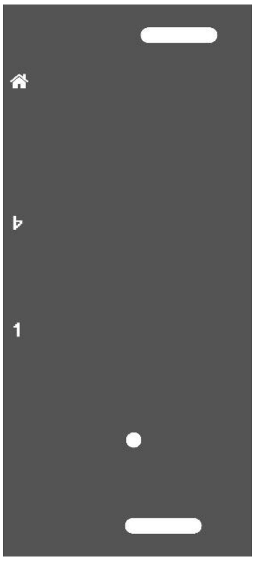


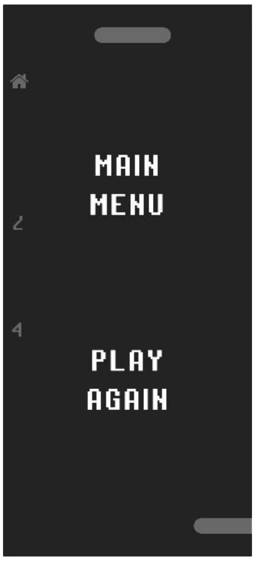


Game mechanics:
The picture in the right shows the starting screen of single player Pong. The high score is written in yellow. As soon as the player touches the screen, the ball starts moving. Every time the ball bounces on the paddle in the bottom part of the screen, the score, which can be seen in the middle of the screen, increments by one.
How to play:
Click on the left side of the screen to make the paddle go to the left; click on the right side of the screen to make the paddle go to the right. Don’t let the ball fall below the paddle!!

3.3 PvP Local Session Pong

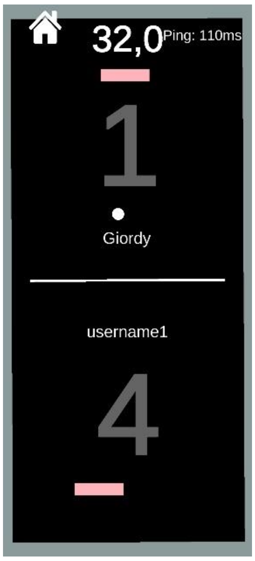






Game mechanics:
The picture in the right shows the starting screen of PvP local session Pong. The home button allows the player to go back to the Main Menu. The big triangular button makes the match begin. Each player has one assigned paddle, either in the top portion of the screen or in the lower part. The aim of the game is to make the ball fall over the screen on the side of the opponent. Every time a player makes the ball go off the screen on the end of their enemy, their score is incremented by one. The first player to get 7 points, wins. Then, the game over screen pops up and the players can decide whether to rematch or to head to the Main Menu.
How to play:
Hold your finger on your paddle and slide it across the screen to move it. Don’t let the ball fall off your side of the screen!

3.4 PvP Online Session Pong



Game mechanics:
The home button allows the player to go back to the Main Menu. The ping text shows the real time latency of the connection between the host and the server. The game begins as soon as two players are connected. Each player has one assigned paddle, either in the top portion of the screen or in the lower part. The aim of the game is to make the ball fall over the screen on the side of the opponent. Every time a player makes the ball go off the screen on the end of their enemy, their score is incremented by one. The game ends when the 60s countdown shown on top of the screen expires. At that point, the player who has scored the most points wins.

How to play:
Click on the left side of the screen to make the paddle go to the left; click on the right side of the screen to make the paddle go to the right. Don’t let the ball fall below the paddle!!

Conclusions

1. Personal Software Process

PSP2.1 (Personal Software Process)	Estimated the time required to complete and why (day)	Actual completion time and why (day)
Planning		
· Estimate（To Estimate how much time this task will take and plan the general steps）	18 weeks. See the slack time and critical path in the requirement analysis section	11 weeks. We did not have enough time to go through all the tasks we planned to do
Development		
·· Analysis（Demand Analysis (including learning new technology)）	3 weeks. See the slack time and critical path in the requirement analysis section	3 weeks. We sticked to the plan.
· Design Spec （Generate design document）	1 week. See the slack time and critical path in the requirement analysis section	1 week. We sticked to the plan.
· Design Review（Design review (and co-worker review and design document)）	3 days. See the slack time and critical path in the requirement analysis section	2 days. We were feeling we did not have enough time to complete the following tasks.
· Coding Standard (Code specification (develop appropriate specifications for current development))	1 day	1 day. We sticked to the plan.
· Design (detailed design)	1 week	4 days. We more or less sticked to the plan.
· Coding	5 weeks. See the slack time and critical path in	2 weeks. We did not have enough time to complete all the

	the requirement analysis section	functionalities of our application. Still, we have managed to implement all the functional requirements.
· Code Review	1 week	3 days. Lack of time
· Test (Testing (self-testing, modifying code, submitting changes))	4 weeks. See the slack time and critical path in the requirement analysis section	1.5 weeks. We did not have enough time
Reporting		
·· Test Report	1 week	1 week
· Size Measurement	2 days	2 days
· Postmortem & Process Improvement Plan	1 week	2 days

2. Conclusion of the software development experience

2.1 Good experiences

The course has left us with a new understanding of the world of software development. Not only it provided great insight to what software engineering is, it has also made us have fun while experimenting new ways to expand our knowledge in the ICT area, in a sector, that of software, which we had never encountered before in our academic career. Before attending this course, we hardly had a bare understanding of what software engineering is and we used to mistakenly associate it with coding of applications or other software products. During the unfolding of the course, we started to realize that the software engineering world is completely different -and far more complex- than that of mere programming. We took the chance to appreciate the beauty of doing things in a rigorous and schematic way, giving life to diagrams, documentation and designs which have allowed us to accomplish what we had never before dreamt of doing: develop a perfectly operational application from scratch. All just by ourselves. Our joy is immense. Our self-confidence is through the roof.

We had a lot of fun in making our mind concerning what piece of software to develop, because we had too many ideas, and then planning our app functionalities and squeezing our brains to design the app as best as we could, trying not to overlook anything. We have enjoyed working together as a team, as if we were the limbs of a single entity focused on a single target. Working together has made us grow closer together, learn to put ourselves in somebody else's shoes, discuss about things, compromise, turn to others when our skills were not enough to solve a problem, … which are skills we will carry with us in our future working careers, and which will make us better employees.

Finally, especially during the diagram making activities and during the coding of our application, we had to constantly face new and more and more complex challenges. This has made us seek new knowledge and explore new tools to deal with them (e.g. generative AI). Overall, the development of this project has made us become better engineers, but also better people.

2.2 Bad lessons

Throughout the unfoldment of the project, we were required to cope with a handful of issues and unexpected problem that would arise from time to time. Amongst those, the ones which challenged us the most were:

- Lack of experience. Before this course had begun, never in our lives had we had anything to do with software development. This has then proved to be a serious issue in following stages of the software development. At first, we had no idea of what the IDEs for application coding were, and once we had decided to use Unity, we had no idea of how it worked as well. We spent countless hours in trying to grasp its basic functionalities, which means that a great amount of time has not been devoted to coding during the alpha sprint development. As a result, we had to continue the coding process also after the alpha sprint development for about another week.
- Lack of time. Due to external circumstances, such as the short duration of the course, we have had very little time with respect to what we would need to develop our project idea in full.