Middle East Technical University

Department of Computer Engineering

Adamadama Development

A Game of Multitasking

Software Requirements Specification Document

Arınç Elhan - 1819291

Halim Görkem Gülmez – 1819382

Muhammet Furkan Yılmaz - 1881630

Yiğit Kardelen – 1819424

November 30, 2014

Preface

This is the official document of Software Requirements Specification(SRS) of “A Game Of Multitasking”. This document is prepared according to the “IEEE Recommended Practice for Software Requirements Specification – IEEE Std 830 – 1998”. Main purpose of this document is to give detailed information about the functionalities, constraints and software requirements of the project.

A Game of Multitasking is a unique mobile game that aims to fuse multitasking and mobile gaming. Two main modes will be available when the game is released. The target audience who wants to make use of this system, can find all related information in this document. It assists the development team and the end users.

The first section gives the definition, purpose and scope of the project. Other sections include detailed information, descriptions and requirements of the project. All the design, implementation, and test/validation plans will be accessible in this specification document.

Table of Contents

1. Introduction

1.1 Problem Definition

There are some games on mobile markets that have the theme of multitasking but they are boring, simple and unsatisfying from the point of view of users. So, in order to create much more entertaining and unique game we have chosen our project.

There are several issues that needs to be handled while doing the project. First of all we did not implement any network project so developing an online multiplayer game will be a challenge. Using a game engine for the project and publishing the project on several mobile platforms is also another issue for the project members. Finally, combining and implementing three or more different games with different control mechanisms for the same screen is another issue.

1.2 Purpose

This is a software requirements specification document about “A Game of Multitasking”, which provides a complete description of all the properties that will be implemented and a complete definition of system attributes. Specialties of the system and functionalities of the project will be defined, explained and demonstrated by using some models.

The main purpose of the project is to create an entertaining and a unique game with a multitasking theme, which runs on the mobile platforms. There are several mini games in the project and each one will be played with a different mechanism of the device such as virtual joystick with one game or accelerometer with another game. This document’s audience is developers, testers, end users.

1.3 Scope

The project’s name is “A Game of Multitasking”. The scope of this project is playing a unique game that consists of multiple mini games, which are controlled by different kinds of mechanisms at the same time. The aim of this project is to create an entertaining and different mobile game by using the multitasking theme. This project consists of two basic modes, which are called “single player mode” and “multiplayer mode”, respectively.

In the single player mode, there are several different mini games (three or four) that will be played when the game starts. Player will play those mini games by using different control mechanisms of the device such as accelerometer or virtual joysticks.

In the multiplayer mode, players will connect to the same mini games and play at the same time in order to compete each other. One of the mini games is about disrupting other player's mini game.

1.4 Definitions

Term	Description
Database	A collection of related data about user scores.
DBMS	A software package/system to facilitate the creation and maintenance of a computerized database.
User	Person who can play the single player or multiplayer mode of the game.
SRS	Software Requirements Specification.
Class Diagram	A type of static structure diagram in UML that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationship among the classes.
Use Case Diagram	A type of diagram in UML that represents the user's interaction with the system.
Sequence Diagram	An interaction diagram that shows how processes operate with one another and in which order.
Unity 3D	A cross-platform game creation system developed by Unity Technologies, including a game engine and integrated development environment.
Game Engine	Software framework designed for the creation and development of video games.
Integrated Development Environment	Software application that provides some good and useful facilities to programmers for software development.
IEEE	The Institute of Electrical and Electronics Engineers
FPS	Frame per second
ms	millisecond
MB	megabyte
Scene	In unity every different screen is represented as scene.
Parse	A hosted service, providing backend services to end-user

	applications.

1.5 References

The resources listed below are the references that has been used during the requirements analysis; IEEE Standard Documents:

[1] IEEE. (1998). IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society.

[2] Unity UI. (n.d.). Retrieved November 28, 2014, from http://unity3d.com/

[3] Parse. (n.d.). Retrieved November 28, 2014, from https://www.parse.com/

[4] How can I connect Unity to an SQL database in order to implement an MMO? (n.d.). Retrieved November 28, 2014, from http://answers.unity3d.com/questions/15694/how-can-i-connect-unity-to-an-sql-database-in-orde.html

1.6 Overview

This document includes seven chapters:

- Overall Description
- Specification of Requirements
- Data Model and Description
- Behavioral Model and Description
- Planning
- Conclusion
- Supporting Information

The software requirements begin with the overall description of the project. The product perspective, major functions and constraints are covered briefly. The software and hardware interfaces are stated in the next section as well. The following chapter gives the detailed requirements and functionalities of the software product. The functional and nonfunctional requirements are explained broadly. The design model and design constraints

are mentioned extensively. One complete chapter is reserved for the data description and the behavioral model separately. SRS document involves UML diagrams for the project design. The use case, class and state diagrams are depicted in related sections. The documentation is finalized with the estimated schedule and team work for the project.

2. Overall Description

2.1. Product Perspective

A Game of Multitasking is a mobile game which will be made in Unity Game Engine. The game will have two modes. Single player mode will consist of three or more mini games and multiplayer mode will consist of two mini game. When the user enters, the game menu screen will be displayed. From menu screen the user can go to preference scene, single player scene, multiplayer scene, leader board scene and about scene. For the first visit of the user there will be tutorials for each game. As mentioned before single player mode will consist of three or more mini games. Initially only one game will be shown. After the user plays a specific amount of time second mini game will appear. After the user plays two mini games concurrently for a specific amount of time the third mini game will appear and so on. After the user plays four mini games concurrently for a specific amount of time the game difficulty will increase until the user loses one of the mini games. Single player modes goal is to get the highest score. Users can see their scores in leader board screen. Multiplayer mode consists of two mini game. The user will see both the opponent's game and their game real-time. The mini game in multiplayer mode will be designed such that the player can interact both their game and their opponent's games concurrently. The game can be won if the opponent loses the game. The connection between two devices will be established by peer-to-peer architecture. There will be a external server which stores current active connections. Clients (devices) will be in interaction with the server only for playing multiplayer mode.

2.1.1. System Interfaces

One system requirement is Internet Access for the multiplayer mode and checking leader boards. Also systems should be mobile platforms and have Android 4.0 (or later) or iOS 5 (or later) operating systems.

2.1.2. User Interfaces

Figure 1: Initial screen of the game

The Above screen is the initial menu of the our application. “Single Player” and “Multiplayer” buttons are for selection of game type. The logo (may not be last) of our project is placed at the left corner of the initial screen. Additionally, “Preferences” and “Leader Boards” can be reached using the buttons that are located at the top right corner. Finally, the background image currently is set as seen in above, however, it can be updated in the next steps of the project.

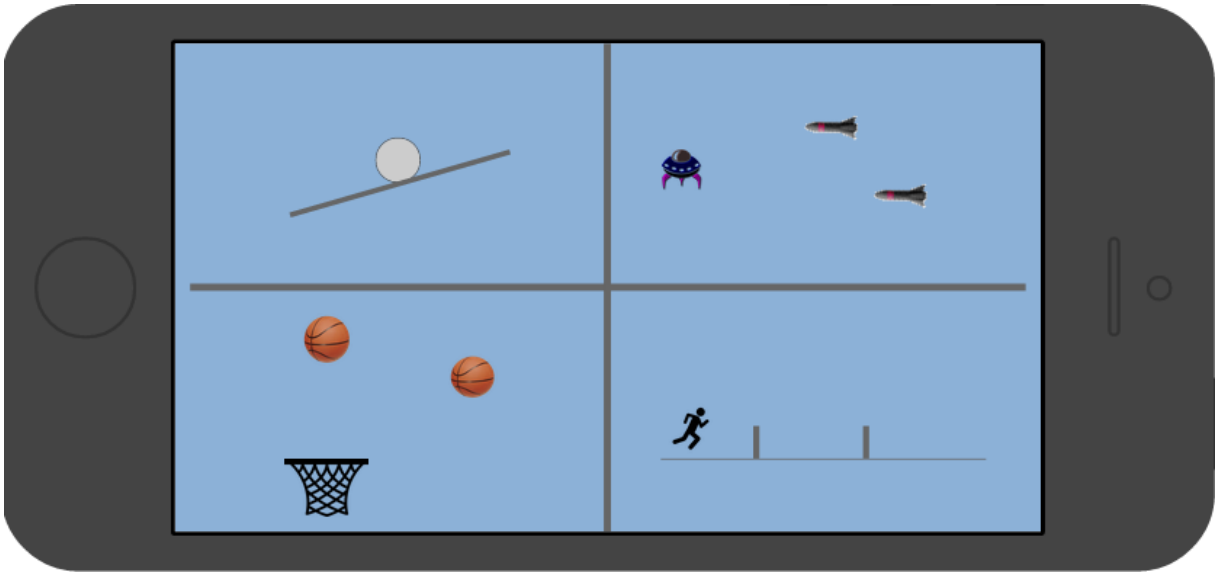

Figure 2: A mock screenshot for single player mode

In play screen, there are four different subscreens which contains four different mini games. The first one is a balance game. User should balance the ball on the bar and should not drop it. The second one is a retro space - alien game for which user is a spacecraft and should dodge all the missiles using up and down buttons. The third one is a simple basketball game. User should make a specific amount of points in order not to lose this mini game.

Finally, last one is an endless run game for which user should jump over the obstacles that comes randomly. In order to control these four games simultaneously, more than one virtual control mechanisms will be implemented.

2.1.3. Hardware Interfaces

Since our project is based on mobile platforms, a mobile phone with android or iOS operating system is a must to play the game. Also for the multiplayer mode of the game, there should be a smooth Internet connection, Wi-Fi or 3G, is necessary. Also, since we are using the Unity3D's API for the communication between the users in the multiplayer mode, Unity3D should be supported by the mobile device which the game will be played. For external server if found appropriate Parse servers will be used.

2.1.4. Software Interfaces

The game will work on all smart mobile devices with Android 4.0 (or later) or iOS 5 (or later) operating systems. There will be a graphical user interface for the player to interact with the game by using it. There will be another interface which will do database interactions. This interface will be invisible to the user. For external server if found appropriate Parse Unity API will be used.

2.1.5. Communication Interfaces

Project is a mobile application and in the single mode there won't be any communication. In the multiplayer mode, since Unity's multiplayer API is used, network connection which is used in Unity's API is necessary (called UNET). For external server if found appropriate Parse Unity API will be used.

2.1.6. Memory Constraints

Since we will be using Unity as a game engine, Unity will pack lots of memory for its own. The devices minimum available RAM should be 100 MB and internal memory 20 MB. External server will manage 20 GB of data.

2.1.7. Operations

As some fundamental operations were introduced in the section 2.1.2 (User Interfaces), this section will summarize the operations. Most of the operations are visible to anyone who will play the game. User can look the tutorial of the game, play multiplayer or single player mode or share his/her score on social platforms. After these interactions between the user and system, invisible operations shall take place between the subsystems. Such as after sharing the score, Facebook API will handle the connection protocols.

2.1.8. Site Adaptation Requirements

There will be no site adaptation requirements, because project's base is mobile platforms. Anyone with a smart phone can play on single player mode and if he/she has an active internet connection then he/she can play on multiplayer mode, too.

2.2. Product Functions

In this section, the main functions that the system will provide will be explained as a higher-level specification. Use case diagrams will be used in order to represent these main functions.

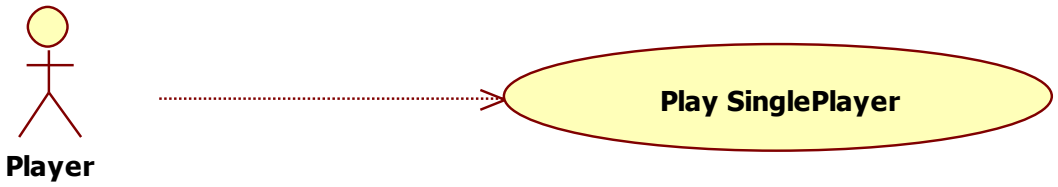

Figure 3: Use Case 1

Play Single Player: The user plays the single player mode.

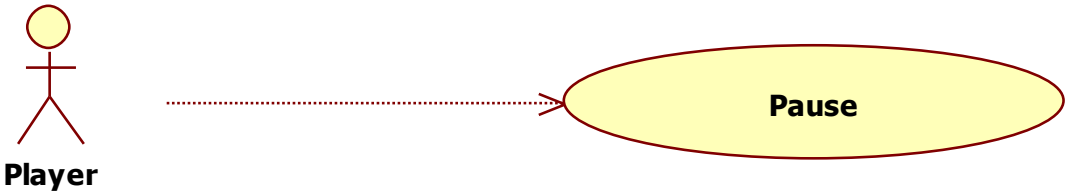

Figure 4: Use Case 2

Pause: The user pauses the game in single player mode.

Figure 5: Use Case 3

Continue: The user continues the paused game.

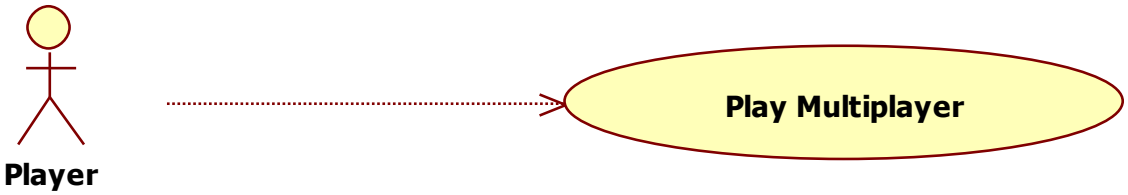

Figure 6: Use Case 4

Play Multiplayer: The user plays the multiplayer mode.

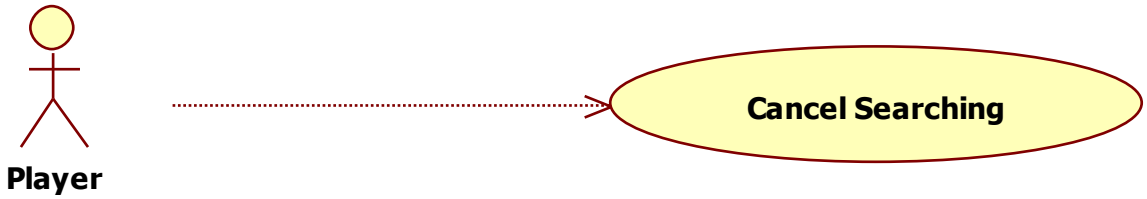

Figure 7: Use Case 5

Cancel Searching: The user stops searching for a game in multiplayer mode.

Figure 8: Use Case 6

Leader Boards: The user checks his/her score among with other people.

Figure 9: Use Case 7

Login: The user logs into the game by using his/her created account or another possible accounts (Facebook or G+).

Figure 10: Use Case 8

Register: The user registers to the game.

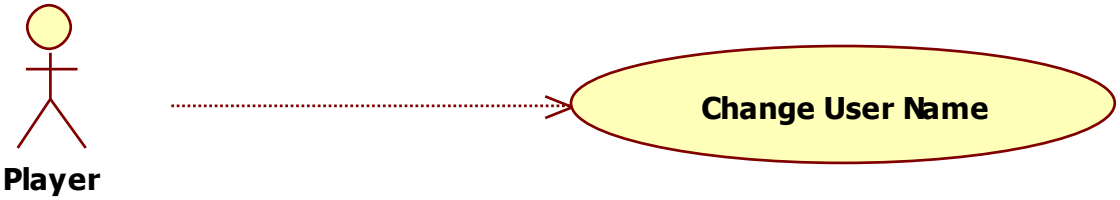

Figure 11: Use Case 9

Change Username: The user changes its username.

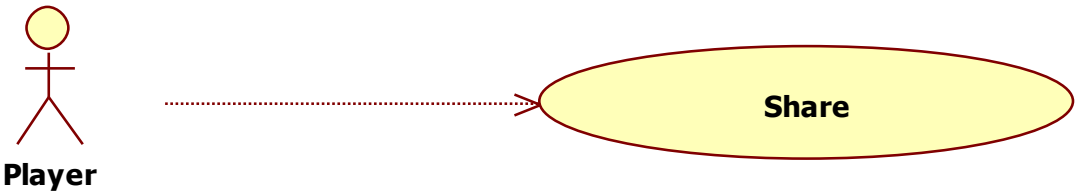

Figure 12: Use Case 10

Share: The user shares his/her score via Facebook and Twitter.

Figure 13: Use Case 11

About: The user sees information about developers.

Figure 14: Use Case 12

Steer: The user steers his/her device to control the game.

Figure 15: Use Case 13

Tap: The user taps to the screen to control the game.

Figure 16: Use Case 14

Swipe: The user swipes his/her finger on the screen to control the game.

Figure 17: Use Case 15

Remove Ads: The user removes the advertisements on the game by paying a certain amount.

Figure 18: Use Case 16

Preferences: The user changes its preferences (volume etc. here).

Figure 19: Use Case 17

See Tutorial: The user sees the tutorial to learn how to play the game.

Overall use case diagram is given below:

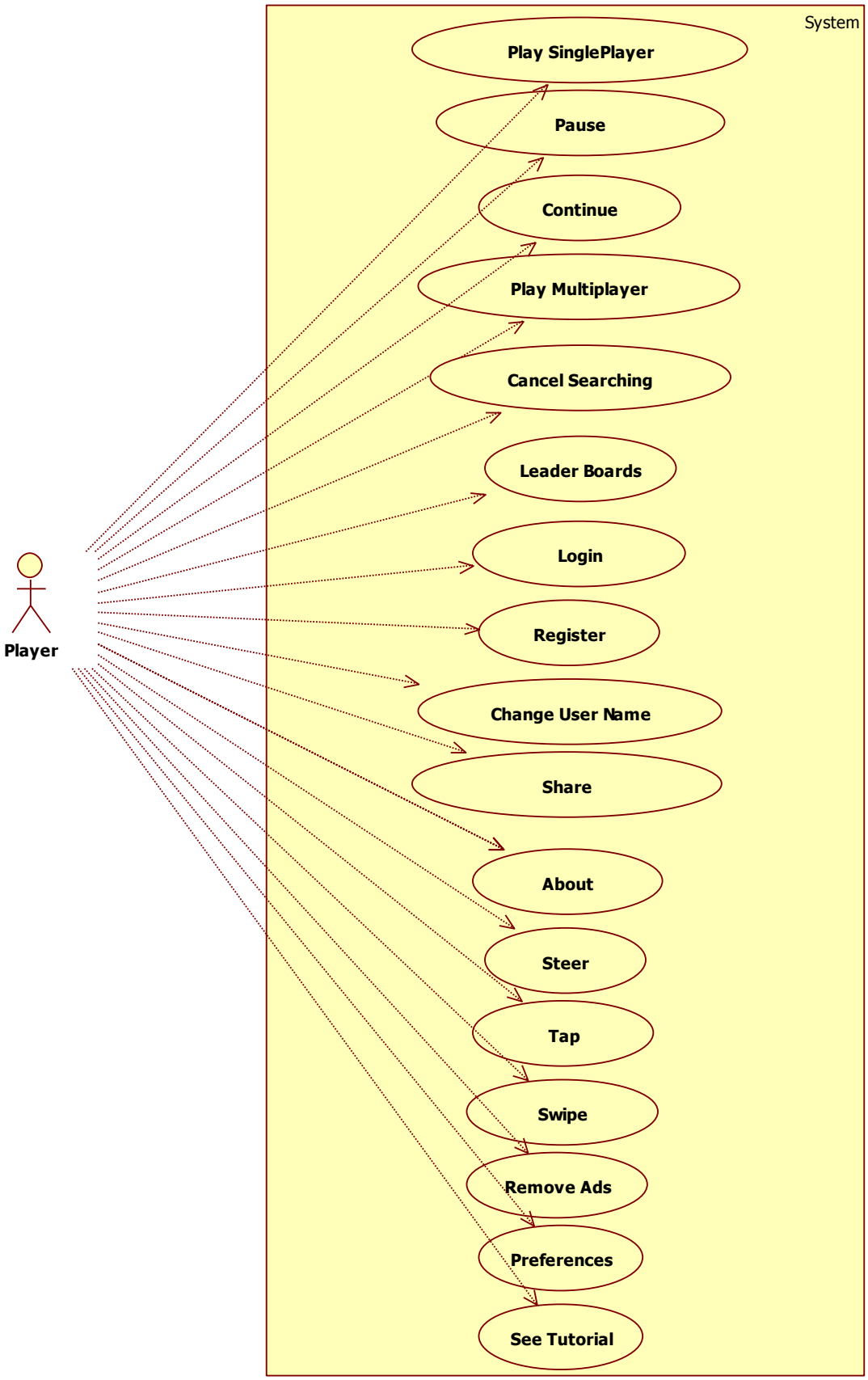

Figure 20: Use Case Diagram

2.3. User Characteristics

The user should be familiar to mobile apps and mobile games. User should know understand English and have interest in multitasking.

2.4. Constraints

We are planning to launch our product to both iOS and Android platform. We will support Android 4.0+ and iOS 5+. The device should have built in accelerometer as a sensor event. For multiplayer mode the devices should be connected to the internet via wireless network. Also Parse servers should be not be in offline state.

2.5. Assumptions and Dependencies

Apart from the OS/platform that the system will run on, Unity game engine will be our dependency. Also Parse servers will be a dependency for multiplayer mode.

2.6. Apportioning of Requirements

These are not strict requirements for the future releases, but rather the additional functionalities that may or may not be integrated in later versions.

- More than 4 games.
- For the sensor events custom calibration
- Facebook data fetch
- Showing thumbnail while searching opponents.

3. Specification of Requirements

3.1. Interface Requirements

The detailed user interface description is given in Section 2.1.2 with mock screenshots. The user interface shows how use cases will be implemented. Also, more mock screenshots about gameplay is given, too. The detailed information about the use cases, and their functionalities are given in Section 2.2 with diagram.

3.2. Functional Requirements

Section 2.2 has a detailed information about the functional requirements by explaining use cases in detailed manner, so there won't be any detailed information in here.

See section 2.2 for detailed explanations.

3.3. Non-functional Requirements

3.3.1. Performance Requirements

In single player mode, the game should work at least 30 FPS, so that the game will be fluent. Also, the game should not crash more than one per one hour playtime. Finally, loading phase of the game should not take more than 30 seconds; in other words the game should start in 30 seconds after the user opens it -this requirement is not just about the single player mode but about the game itself.

In multiplayer mode, the matchmaking system should match two players in at most 30 seconds. Additionally, both players should be able to play at most 150 ms latency. Also, our game should handle up to 500 players to play the multiplayer mode simultaneously for the beginning. If our game becomes popular and gets played by much more players, then we should handle more players with necessary upgrades. While playing multiplayer mode, there should not be more than 20 MB of data flow per one hour playtime for each player.

Finally, a player who has not played the game before should learn all of its functionalities in one hour.

3.3.2. Design Constraints

This section has five subsections which are security, usability, availability, maintainability, respectively and they are explained in detail on their own sections. Also, this section contain a brief information about the attributes of the software system by the terms of “good” software attributes.

3.3.2.1. Security

Database has to kept secured as it will contain e-mail addresses of the registered users. Also, if any cheating attempt exists, it has to be prevented. Leader boards will be kept in Apple’s own servers, therefore its security is not our concern.

3.3.2.2. Usability

The scope of the product is widespread. People from every age should play the game without any effort.

3.3.2.3. Availability

The application will be available unless it is removed from the mobile market.

3.3.2.4. Maintainability

The system should be maintainable in order to add new features to the newer versions. Also, the game should work on newer versions of the related operation systems (the game should be updated then if necessary). Hence, all design aspects should be well-documented and easily understandable.

3.3.2.5. Portability

The game requires minimum effort in terms of installation. Also it should work on any smart mobile devices with Android 4.0 or iOS 5.0 (or later).

4. Data Model and Description

4.1. Data Description

4.1.1. Data Objects

UML Class Diagrams will be presented in this section. The purpose is to introduce the objects that the system is required to have a representation for. There are 15 classes namely Player, GameStats, GameController, DBConnector, UI, Game, GameServer, Scene, GameObject, Component, AbstractGameScript, FirstGameScript, SecondGameScript, ThirdGameScript and FourthGameScript. Their explanation is given below.

Player: This class represents the player entity and it features.

GameStats: This class hold general statistics about the games played by the player.

GameController: This class is the main logical unit of the application. It has control over almost all object. It is capable of creating/joining rooms in Multiplayer mode.

DBConnector: This class is responsible for Database related operations. Player list , game statistics and all related objects will be stored at the Database. This operations will be handled in this class.

UI: The UI class will hold all functionalities (buttons, symbols) that the application provides to the users.

Game: The Game class holds reference to the scene object and contains properties of game.

GameServer: The GameServer contains all available rooms (both automatically generated and user created). Also, GameServer has functionality to search and prepare rooms for players.

Scene: The Scene class has list of GameObject. It contains physical and other futures about the scene.

GameObject: The GameObject class is one of the main class in our application. Every object that is seen on the game is actually a GameObject type.

Component: The Component class is attached to the GameObject instances. It provides extra functionalities.

AbstractGameScript: The AbstractGameScript class is the generic class for game script classes. It contains common functions that will be used in every game scripts.

FirstGameScript: The FirstGameScript class will extend the AbstractGameScript class. Moreover, it contains extra functions and members that is specially generated for First Game.

SecondGameScript: The SecondGameScript class will extend the AbstractGameScript class. Furthermore, it contains extra functions and members that is specially generated for Second Game.

ThirdGameScript: The ThirdGameScript class will extend the AbstractGameScript class. In addition, it contains extra functions and members that is specially generated for Third Game.

FourthGameScript: The FourthGameScript class will extend the AbstractGameScript class. Moreover, it contains extra functions and members that is specially generated for Fourth Game.

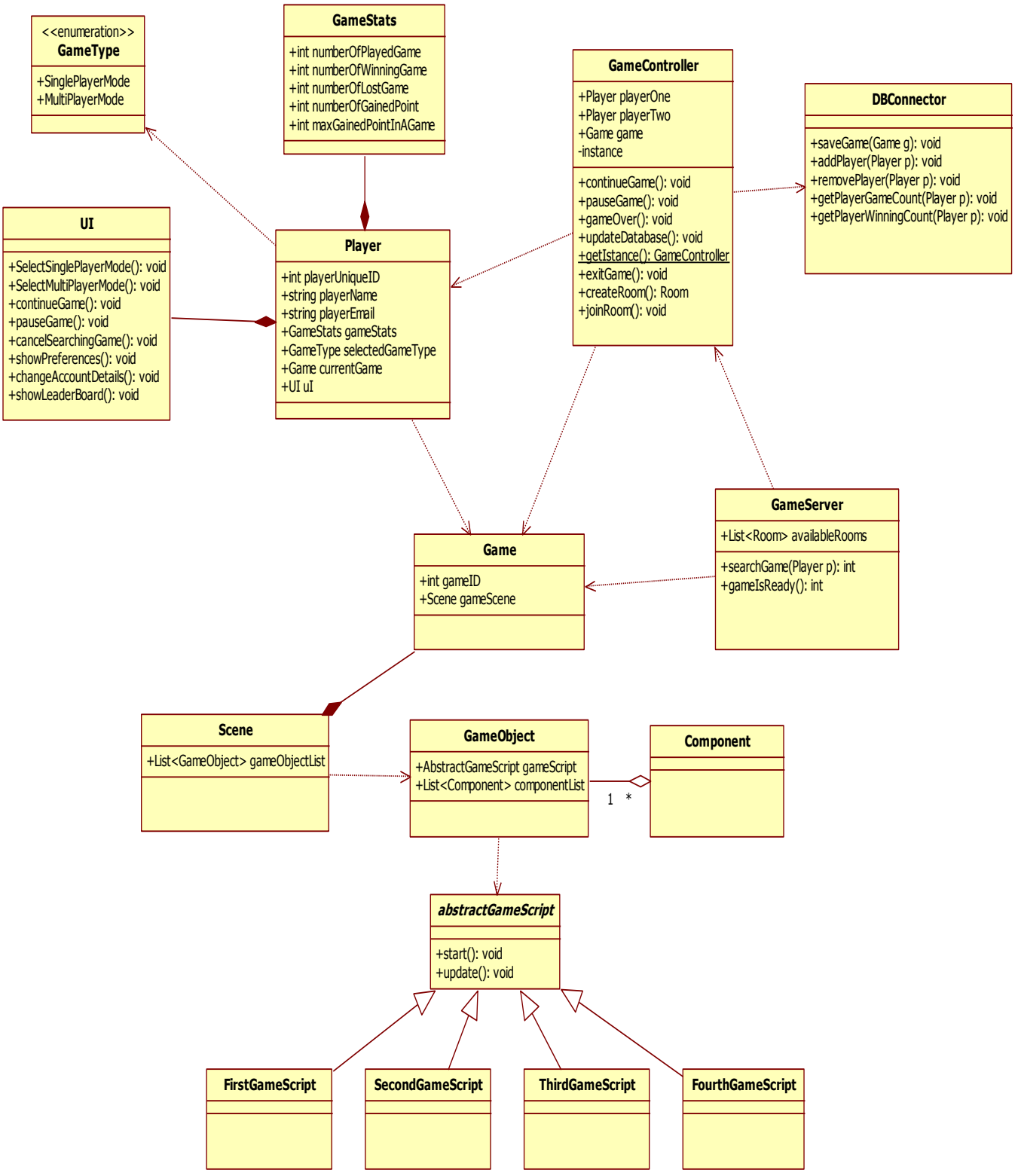

Figure 21: Class Diagram

4.1.2. Data Dictionary

Term	Definition
playerID	The unique identification number of a player.
gameID	The unique identification number of a game.
Room	Room represents a place where two players meet for multiplayer game.
playerName	The nick name that the player chooses.
playerEmail	The main (unique) contact address of player.

5. Behavioral Model and Description

This section provides overall behavior of the system. State chart diagram is given for describing events and states.

5.1. Description For Software Behavior

There are four main states in the system, namely initial state, menu, single player state, multiplayer state. In menu state player has some alternatives to choose from. According to the player's choice, the state will change into single player state or multiplayer state. In single player state, player will play the game until it is over. In this state player can also pause the game. The multi player state is similar to the single player state except it does not have a pause functionality. The user can exit from all states by using device's quit functionality (it is home button for most devices). Until then, the game will be in one of these three states.

5.2. State Transition Diagram

When the user starts the game, user will be in the initial state. In this state there are two options: login and register. A registered user can move onto menu state after login event. An unregistered user can register to the system in this state, then login automatically. Also the user can directly go to menu state if he/she will play the game in offline mode. By doing that, the player will not be able to use all of the functionalities of the game such as multiplayer mode.

In the menu state, there will be different alternatives. The player can choose a game mode in this state, check leader boards or set/mute volume in the preferences. If the player choose a game mode, then its state will be active state.

In the single player state, the player will be playing the game until it loses the game or pauses the game. Paused game can be unpaused in the same state. If the game is over, then the game will return to the menu state after asking the user whether he/she wants to replay or not.

In the multiplayer state, the player will be matched with another player. These two player will compete against each other until one of them loses the game. Then, the game will ask both players whether they want to rematch or not. If they do not want to rematch, then the game will return to the menu state.

The state diagram is given below.

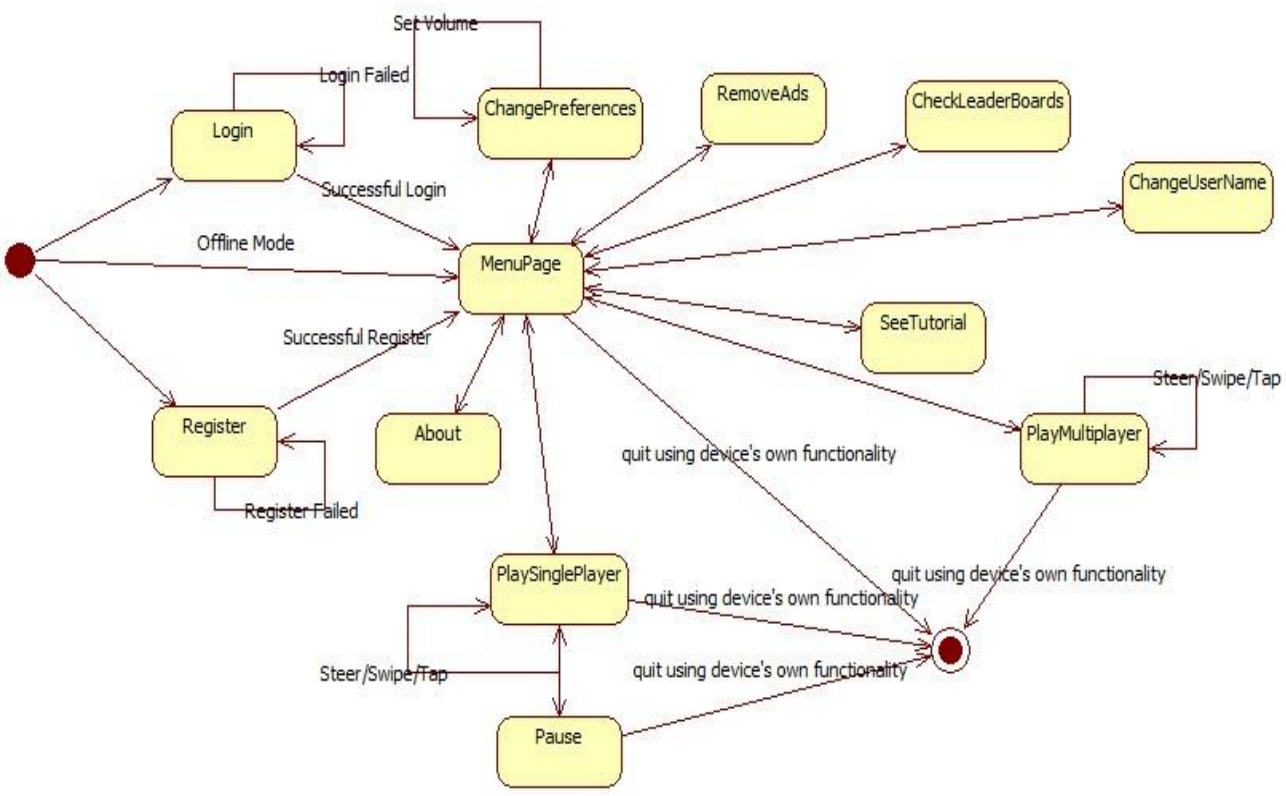

Figure 22: State Diagram

6. Planning

6.1. Team Structure

The four members of the Adamadama team are Arınç Elhan, Muhammet Furkan Yılmaz, Halim Görkem Gülmez, Yiğit Kardelen. The team structure is as follows:

Task	Members
Development of First Mini Game	Muhammet Furkan Yılmaz

Development of Second Mini Game	Halim Görkem Gülmez
Development of Third Mini Game	Yiğit Kardelen
Development of Fourth Mini Game	Arınç Elhan
Development of Initial Menu	Halim Görkem Gülmez, Muhammet Furkan Yılmaz
Tutorials of Mini Games	Arınç Elhan, Yiğit Kardelen
Connection with Social Platforms	Arınç Elhan, Halim Görkem Gülmez
Pause and Continue Attribute in Single Player Mode	Yiğit Kardelen, Muhammet Furkan Yılmaz
Server for Multiplayer Mode	Halim Görkem Gülmez, Yiğit Kardelen
Client for Multiplayer Mode	Arınç Elhan, Muhammet Furkan Yılmaz
General Game Structure	Arınç Elhan, Muhammet Furkan Yılmaz, Halim Görkem Gülmez, Yiğit Kardelen

6.2. Estimation

Estimation Date	Task
15.08.2014	Deciding Project
21.09.2014	Project Idea Request
17.10.2014	Proposal & User Story
9.11.2014	Retrospective Document 1
22.11.2014	Researching Game Engine Attributes
30.11.2014	Software Requirement Specification (SRS)
7.12.2014	Retrospective Document 2
28.12.2014	Retrospective Document 3
4.01.2015	Software Design Description (SDD)
6.01.2015	Making Mockup with Unity Game Engine
10.01.2015	Updated Reports
16.01.2015	Implementing Base Game Structure

6.3. Process Model

Agile model will be used for our project so that quick reviews and changes can be made. Also, prototypes can be available if agile model is applied and system can respond quickly to changing requirements without excessive documentation and work. Agile method is based on an iterative approach, each iteration involves planning, analysis of requirements, design, implementation and testing.

Each iteration takes approximately four or five weeks. After releasing the initial version of the game, adding new games or some features can be easily done.

7. Conclusion

This Software Requirement Specification document is prepared to give requirement details of the project “A Game of Multitasking”. The detailed functional and nonfunctional requirements, system, user, software and hardware interfaces, data and behavioral model are stated in an extended outline. This document will be helpful at constituting a basis for design and development of the system to be developed.

8. Supporting Information

There won’t be any appendixes for this document. Table of contents can be found entrance of Software Requirements Specification document of the project.