Gürkan KARAHAN

Onur Can ÜNER

Serkan TUNA

Kader BELLİ

SOFTWARE DESIGN DESCRIPTION

HEDEF4YILDIZ

Table of Content

Table of Figures

1. Overview

1.1 Scope

This Software Design Description (SDD) document provides necessary information about the project REVENGE. This document includes design principles of the software with its requirements, functionalities and necessary definitions. These information is aimed to guide any programmer to understand our design and be an assistant in the development phase.

1.2 Purpose

The purpose of this document is to outline the design views of the project which will satisfy functional and nonfunctional requirements stated in the SRS Document of Revenge. Purpose of this document is serving as a guideline throughout development phase of the project for developers.

1.3 Intended audience

The main audience of this document is the developers of this project. They are supposed to use this document in the development phase to the structure and design of each component.

2. Definitions

SRS : Software Requirement Specification

SDD : Software Design Description

UML : Unified Modelling Language

TCP/IP : Transmission Control Protocol/Internet Protocol

E-R Diagram : Entity Relationship Diagram

API : Application Programming Interface

3. Conceptual model for software design descriptions

Basic terms, concepts and context of SDD will be given in this part.

3.1 Software design in context

The aim of this project is to provide people a new context in which they can play a multiplayer online fighting game with their own movements. This movements will be captured by using Kinect and its integration will be done in Unity3D game engine. Also real time constraints will be satisfied with this project.

This project will be designed with modular approach. By obeying this approach it will be easy to respond quickly and efficiently to the changing demands' of the customers. Since this game will be played with numerous people this respond time is very critical. Also the content of the game may change and updates may be required. In this case, the updates will be done easily with our approach.

3.2 Software design descriptions within the life cycle

3.2.1 Influences on SDD preparation

This document is prepared by considering the Software Requirement Specification (SRS). The requirements in the SRS (interface, functional, non-functional and logical database requirements) and also the demands of the candidate users specify the design of the project.

3.2.2 Influences on software life cycle products

The visual side and the client-server side of the project should be implemented as paralled. Since this project is a game project which includes dynamic scenes we need make realistic movements or collisions. We are trying to achieve this in our single player game mode and then same implementations will be adapted to the multiplayer mode. On the other hand, the client-server side is trying to send/receive messages through the network. For now, server and client side is working fine for connection and sending/receiving arbitrary messages. Next improvement on the client side will be sending actual joint (small parts on the body) positions which can be obtained by using some Kinect SDK or by using Unity's built-in classes and their functions. Also the next improvement on the server side is parsing this messages properly, then making necessary calculations and responding with new positions.

3.2.3 Design verification and design role in validation

Verification and validation will be tested after the preparation of the test cases but an outline of these test cases will be real time satisfiability, connection and physic reliability, cheating inhibition. All system parts will be tested against these cases. Also in this process, it will be checked whether the requirements in the SRS are fulfilled or not.

4. Design description information content

4.1 Introduction

This part of the Software Design Document provides information about how the design description will be explained in the later parts of this document. In this section, the information about SDD identification, design stakeholders and their concerns, design views, design viewpoints, design elements, design overlays, design rationale and design languages will be given.

4.2 SDD identification

This application is being built by Hedef4Yildiz and the date of issue is 04.01.2015 for this version of the document. The game 'Revenge' will be a multiplayer online game which can be played with Kinect on PC. Every user will have their own profiles and they can have access with given password to the game. Users will be able to play the game with their own movement and we hope that this will be very realistic and enjoyable.

The project group is responsible for the project implementation and the project is owned by the group.

The Unified Modelling Language (UML) is used throughout the document for the design views and design viewpoints.

4.3 Design stakeholders and their concerns

Design stakeholders Revenge project is the team Hedef4Yildiz, their advisors, Dr. Selim Temizer, Prof. Dr. Veysi İşler and their assistant Serdar Çiftçi. Stakeholders' main concern is completing the project until the deadline which is the end of the 2nd semester of 2014. The team and the assistant Serdar Çiftçi meet once in a week and the team gets feedback from him to realize their situation and improve it.

4.4 Design views

The project will be implemented with modular approach. The team can respond changing needs by updating only necessary modules and fix the bugs in small modules. Object oriented principles will be applied so that new features can be integrated without much effort. Users will download the game from our website and while doing this they will obtain their user names and passwords. In order to play the game, users will login to the system with specified user name and password. Then they will select a game mode such as training, tournament of match mode. And then necessary server connections will be satisfied and game will begin. A logical view of the product is explained and also it is supported by diagrams in the upcoming sections.

4.5 Design viewpoints

In this Software Design Descriptions document; context, logical, interaction, interface, composition and state dynamics viewpoints are given. These viewpoints are explained in details with required UML diagrams in Section 5.

4.6 Design elements

4.6.1 Design entities

In context viewpoint there are 2 design entities Application and Game Website. In

logical viewpoint there are 7 design entities User, Task, Character, CharType, Item, Game and Position. In interaction viewpoint there are 9 design entities User, Game Website, Web Service, Server, Database, Download Manage, Operating System, Application Interface and Player.

The details about these design entities such as its variables, methods or meanings will be given in the related topics of Section 5.

4.6.2 Design constraints

4.6.2.1 Performance

Revenge will calculate collisions in server which is a hard task to do and it should do it 10 times in a second since Kinect's sample rate is 10. Therefore, efficient algorithms should be used in order to prevent delays caused by computational issues

4.6.2.2 Real-Time

Since the game will be played in real-time, delays caused by network should be minimized.

4.6.2.3 Security

Revenge will be a multiplayer online competition game. It means that player will have a point or degree and they will try to improve it and they may want to cheat in this situation. The developer team should develop smart algorithms to prevent this.

4.7 Design rationale

Design choices are made by taking some issues in consideration like maintaining accuracy, real time efficiency and reduced design complexity. According to the users' and stakeholders' requirements, it can be updated in the future. Since, we are a team and we may switch the subjects between us, each function in the software will be commented in terms of readability, and understandability.

4.8 Design languages

Unified Modeling Language (UML) is selected as a part of design viewpoint and it will be used for clarifying design viewpoints.

5. DESIGN VİEWPOINT

5.1. INTRODUCTION

In this section following viewpoints are illustrated;

- Context viewpoint
- Logical viewpoint
- Interaction viewpoint
- Interface viewpoint
- Composition viewpoint
- State dynamics viewpoint

5.2. CONTEXT VİEWPOINT

5.2.1. User Use Cases

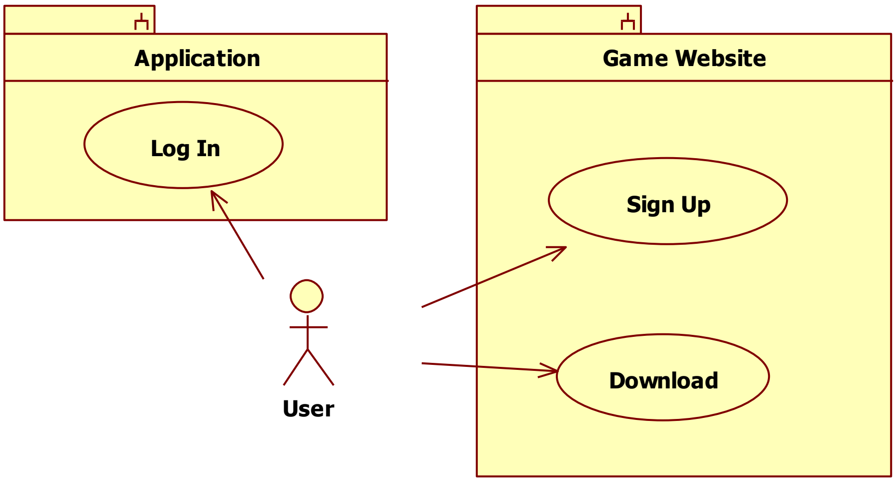

Figure 1 - General Use Case Diagram for User

5.2.1.1. Download

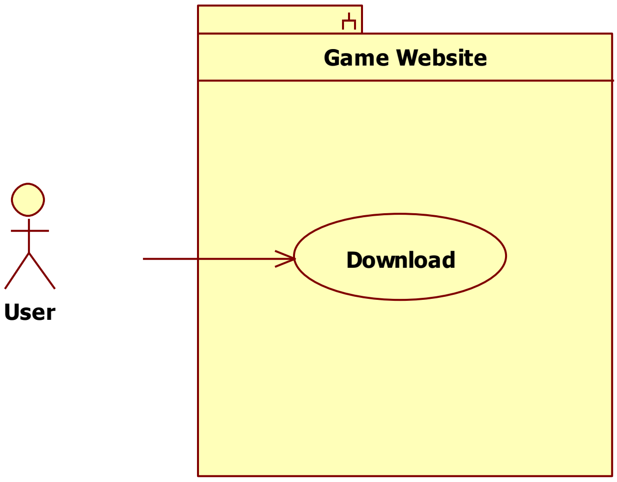

Figure 2 - Use Case Diagram for User - Download

The only source of our game is our website, so user have to download our game by clicking the download session provided in our website.

Preconditions: None

5.2.1.2. Sign up

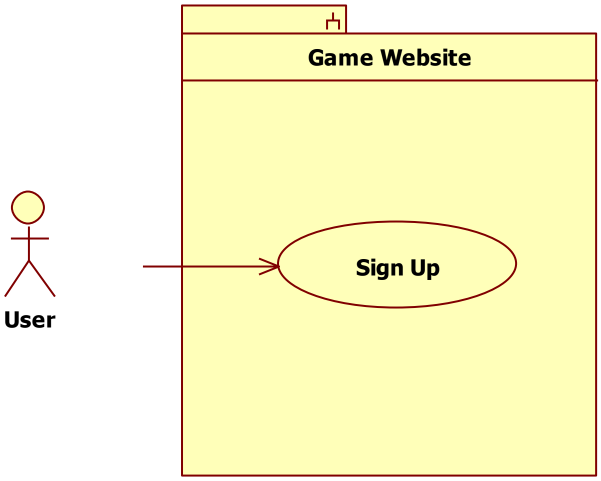

Figure 3 - Use case Diagram for User - Sign Up

User should have an account in order to play our game so user has to sign up by using game website. The required information that user have to provide to sign up are; name & surname, nickname (has to be unique), password, gender and e-mail.

Preconditions: None

5.2.1.3. Login

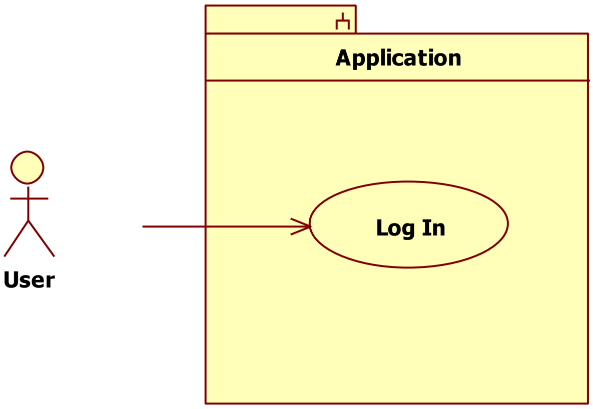

Figure 4 - Use Case Diagram for User - Login

There is a login screen which is opening screen of our game. User have to provide his/her unique nickname (username) and password to reach the interior content of game.

Preconditions: download, sign up

5.2.2. Player Use Cases

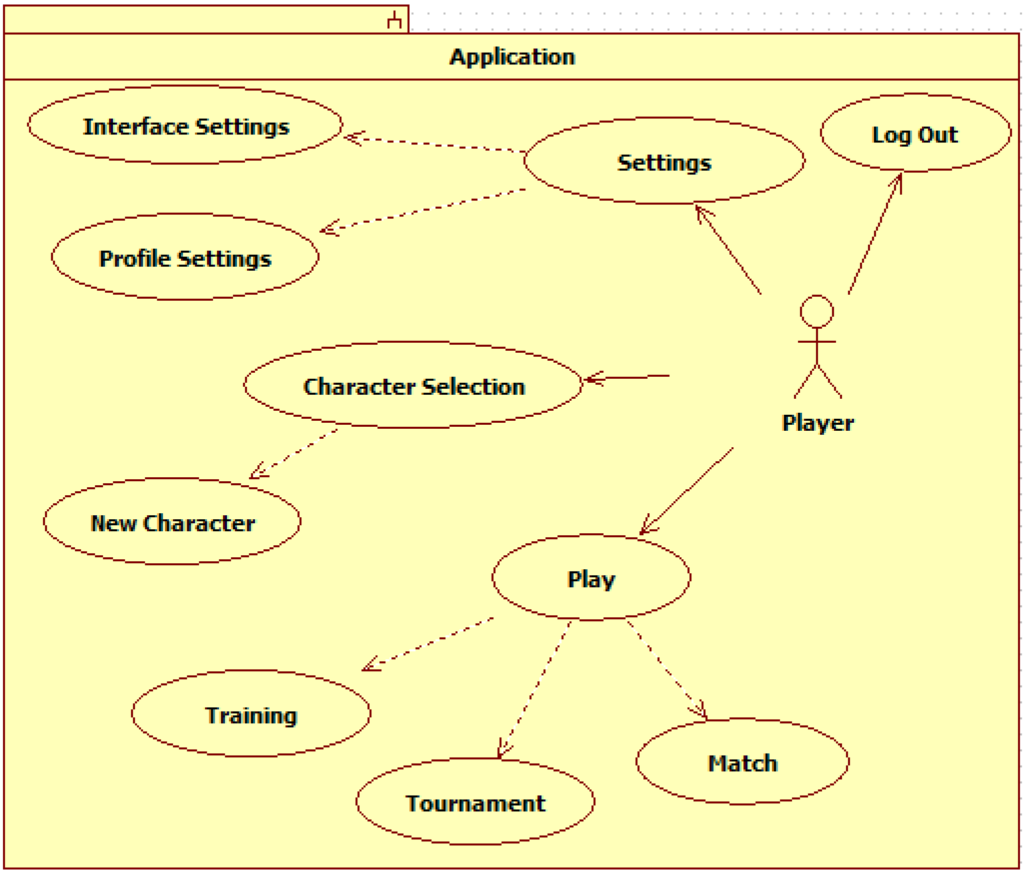

Figure 5 - General Use Case Diagram for Player

5.2.2.1. Settings

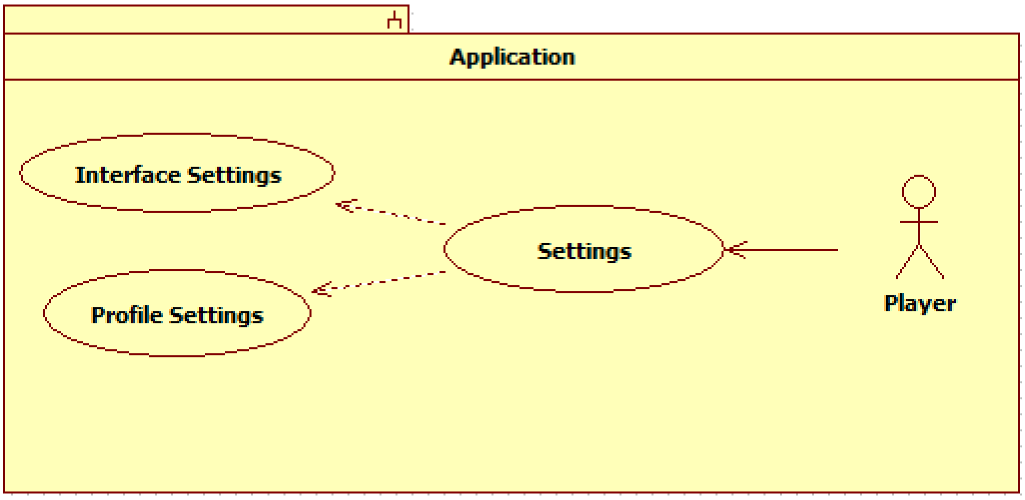

Figure 6 - Use Case Diagram for Player - Settings

5.2.2.1.1. Interface Settings

Player can change the display settings (resolution) and audio settings by clicking the interface settings which is under the settings button.

5.2.2.1.1. Profile Settings

Player can change his/her account settings under profile settings section. There is only information that cannot be changed by player is username (nickname).

Preconditions: download, sign up, login

5.2.2.2. Character Selection

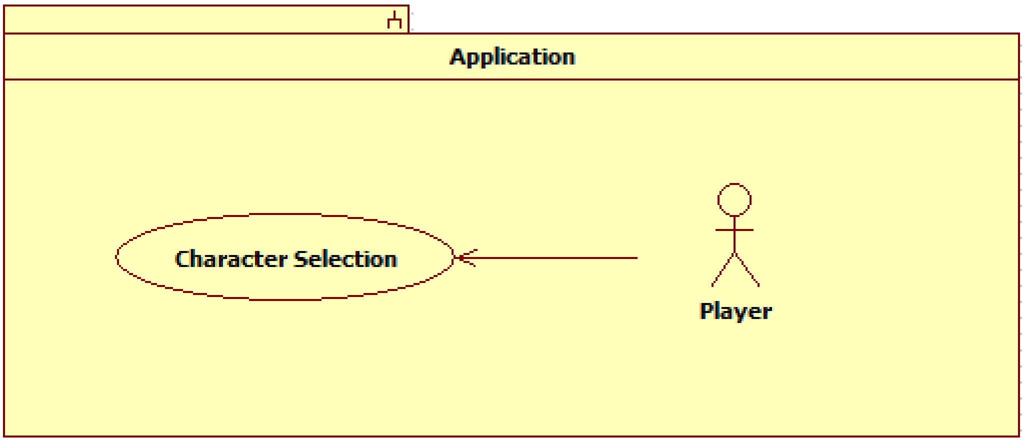

Figure 7 - Use Case Diagram for Player - Character Selection

Player chooses one character among his/her characters in the game. The selected character is used in the following game play. The player can also purchase new characters.

Preconditions: download, sign up, login

5.2.2.3. Play

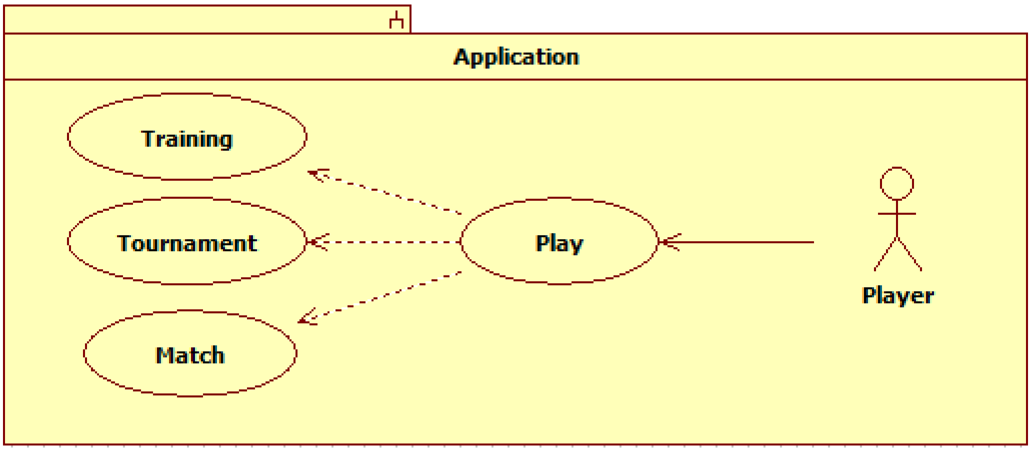

Figure 8 - Use Case Diagram for Player - Play

5.2.2.3.1. Training

Player can select training mode to get familiar with game or improve his/her skills.

Preconditions: download, sign up, login, character selection

5.2.2.3.2. Tournament

Player can choose tournament mode to attend a tournament in order to get medal, cup and some special gifts.

Preconditions: download, sign up, login, character selection

5.2.2.3.3. Match

This mode is provided to find an opponent quickly for a single match. After match there will be a rematch option.

Preconditions: download, sign up, login, character selection

5.3. LOGICAL VIEWPOINTS

The classes and the relations amongst them will be explained here. All classes involved in the project and their relations can be seen in Figure 9.

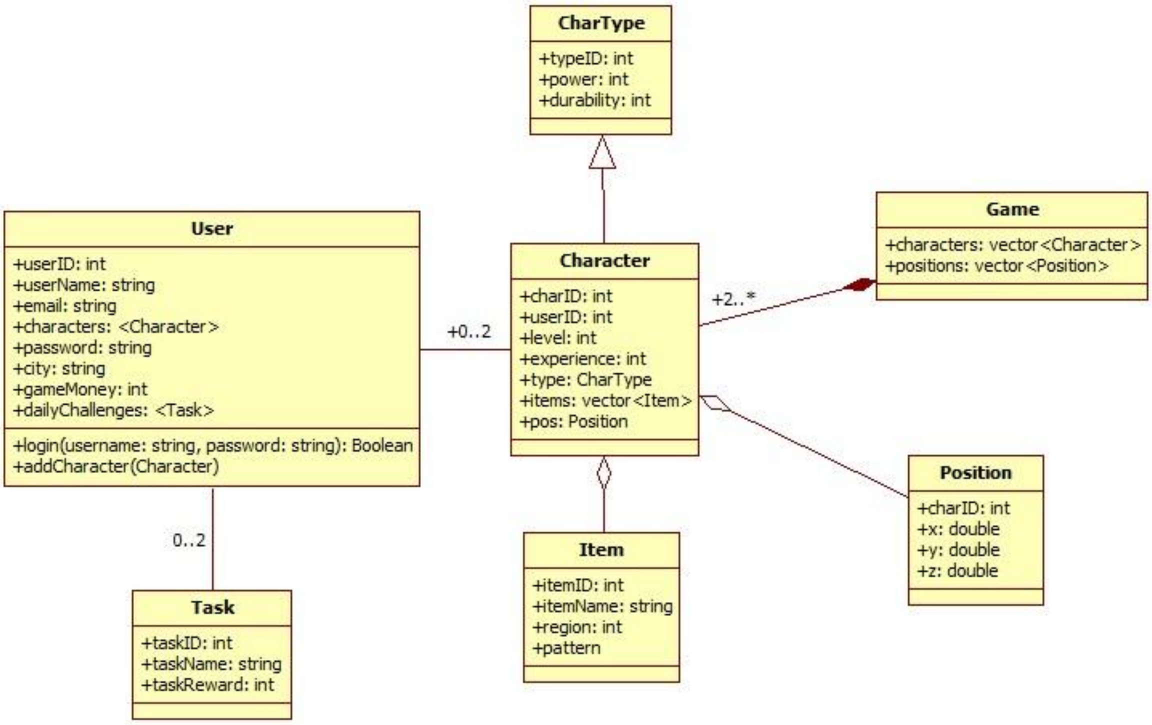

Figure 9 Revenge Class Diagram

5.3.1. User Class

This class holds information of players. Also it has the main functions which players can do.

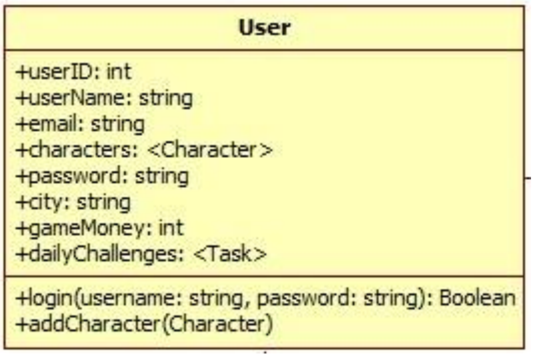

Figure 10 User Class

5.3.1.1. User Class Fields

- userID: Unique integer value for all user.
- userName: String value which is defined by user.
- email: String value which is defined by user. This field is required for verifying of his/her registration.
- password: Every user has a password which is defined by himself/herself.
- city: String value which shows user location.
- gameMoney: It has a default float value for every user. It can be change during gameplay if players buy virtual revenge money or players use it. It shows virtual money amount for shopping in gameplay.

- dailyChallenges: List of daily tasks which are needed to be done in order to gain experience points.

5.3.1.2. User Class Functions

- login: This function is used for authentication for user when login to game. If parameters is true for user authentication then return true value. Otherwise this function return false value.
- addCharacter: This function is used for adding game character which is created by player during gameplay to player character list.

5.3.2. Task Class

This class holds information for tasks which are needed to be done daily.

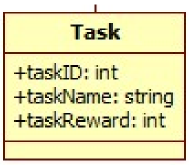

Figure 11 Task Class

5.3.2.1. Challenges Class Fields

- taskID : Unique integer value for all challenges.
- taskName: Name of task which as assign to player.
- taskReward: Experience point amount.

5.3.3. CharType Class

This class is base class of character class. This class holds common attributes for characters information.

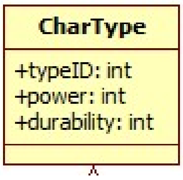

Figure 12 CharType Class

5.3.3.1. CharType Class Fields

- typeID: Unique integer value for each default character(no modified character).
- power: Integer value to represent power of character.
- durability: Integer value to represent durability of character.

5.3.4. Character Class

This class holds information about character modified by player.

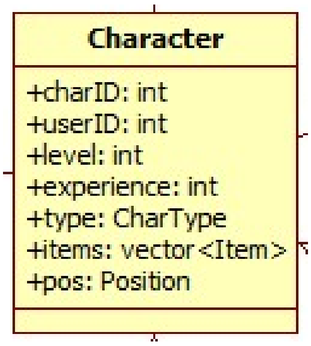

Figure 13 Character Class

5.3.4.1. Character Class Fields

- charID: Unique integer id. Each modified character information is stored according to their charID value.
- userID: Unique integer value of player. This field indicates that this character belongs to which player.
- level: Integer value to represent level of this character.
- experience: Integer value to effect level of character.
- type: Char value to represent type of character.
- items: List of items that belongs to character.
- pos: Instance of position class. This instance shows character current place during the game.

5.3.5. Item Class

This class holds information about items that is used for modification appearance and abilities of character.

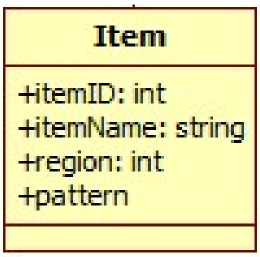

Figure 14 Item Class

5.3.5.1. Item Class Fields

- itemID: Unique integer value for item.
- itemName: String value. This name is showed in gameplay screen when player wants to add item to his/her character.
- region: Enumerable integer value. This value shows that this item where should be on character body.
- pattern: Texture image file. This field contains appearance of this item.

5.3.6. Game Class

This class contains current match information such that players information and their current positions.

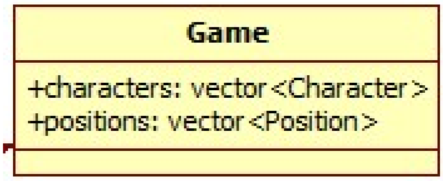

Figure 15 Game Class

5.3.6.1. Game Class Fields

- characters: This field contains list of players which are pair off this match.
- positions: This fields contains positions of players relatively.

5.3.7 Position Class

This class holds player position during match.

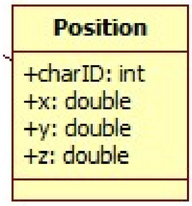

Figure 16 Position Class

5.3.7.1. Position Class Fields

- charID: This field shows that which player has this position.
- x: Player current position on ‘x’ coordinate.

- y: Player current position on 'y' coordinate.
- z: Player current position on 'z' coordinate.

5.4. INTERACTION VIEWPOINT

In this section, system users' interaction with the system and system objects in terms of messages among objects in fulfilling required actions is described. UML sequence diagrams are used to visualize data and work flow for the use cases which are described in Context Viewpoint section. Seven diagrams are drawn to describe the most fundamental use cases. Indeed, a sequence diagram is constructed as a compound use case consisting of several use cases.

5.4.1. Registration Sequence Diagram

Registration sequence diagram defines data flow between the user, game website, game server and the database during the registration process of a new user. Registration is carried out on the game website, before downloading the application.

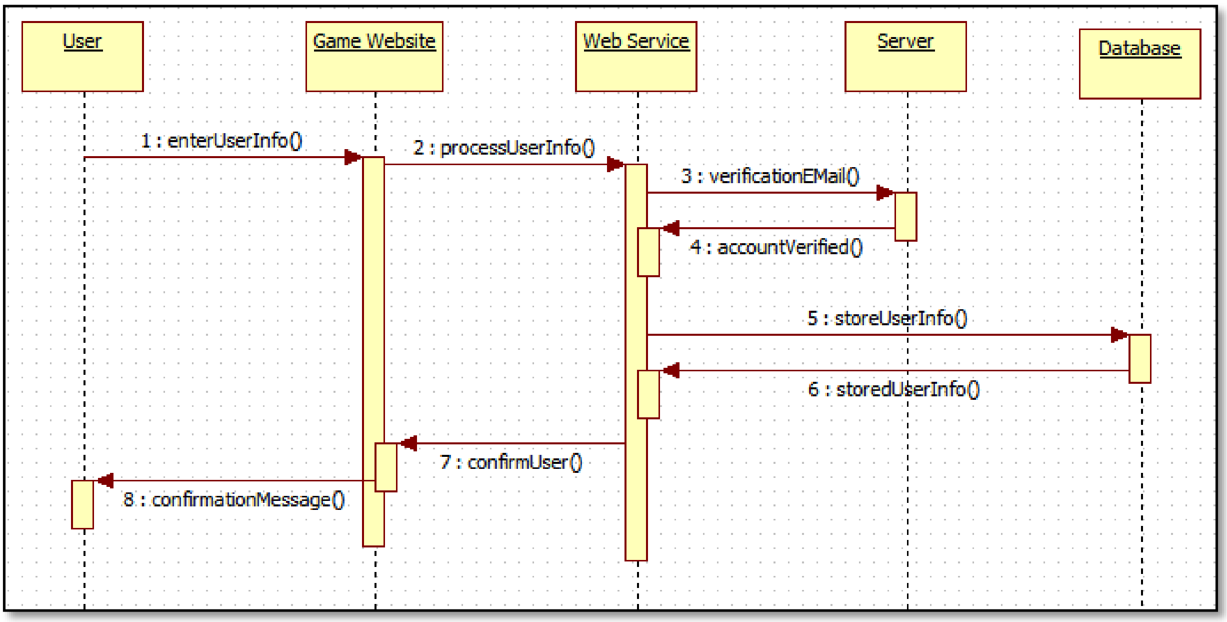

Figure 17 – Sequence Diagram for User Registration Process

5.4.2. Application Setup Sequence Diagram

Application setup sequence diagram describes the process of user's downloading and installing the application to his/her personal computer.

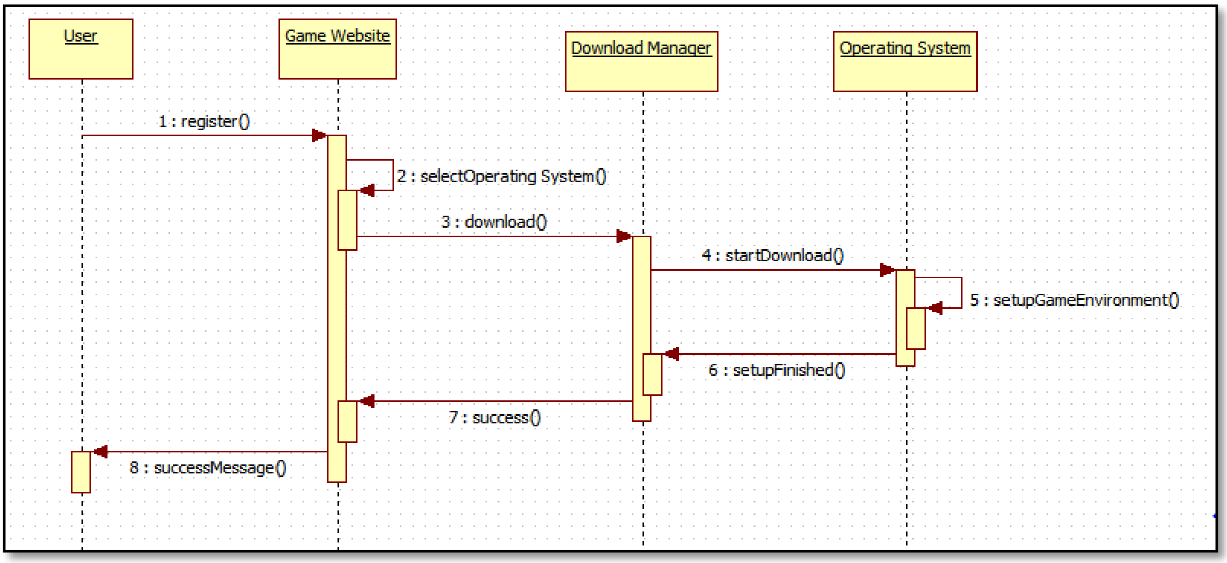

Figure 18 – Sequence Diagram for Application's Download and Installation

5.4.3. Login / Logout Sequence Diagram

Login / Logout sequence diagram describes the data control and flow during the process of login and logout. Login and logout operations are performed on application interface after the application is installed and run.

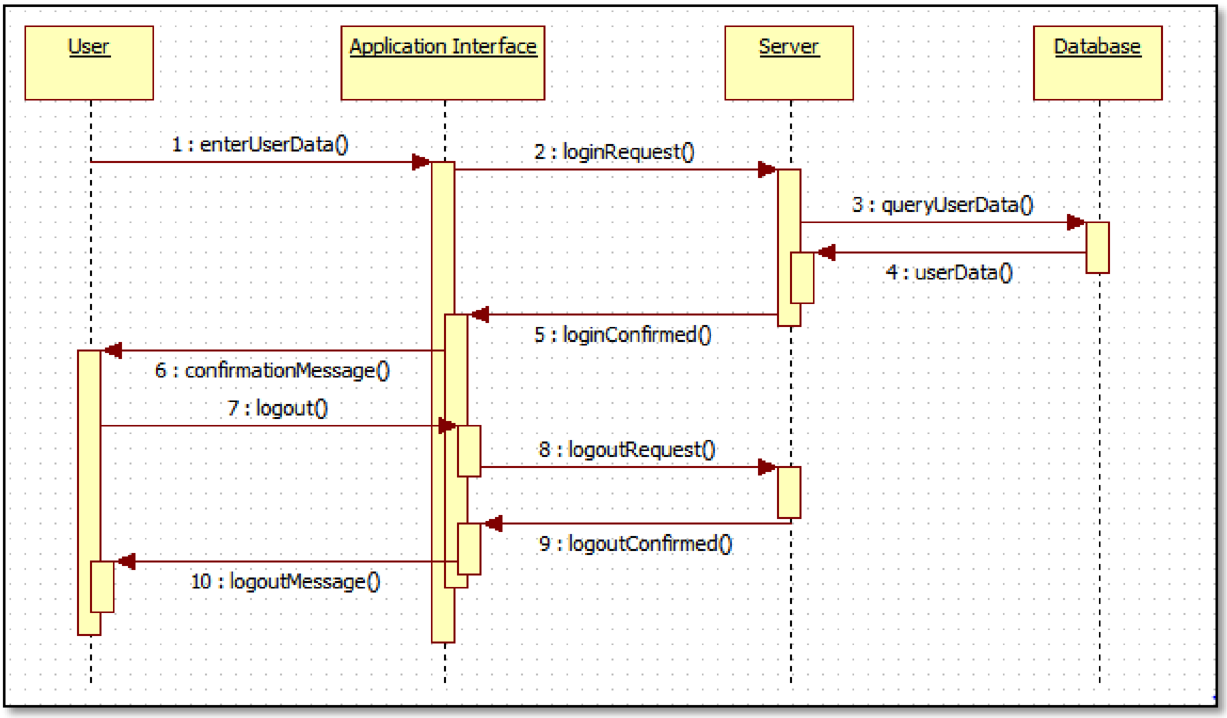

Figure 19 – Sequence Diagram for User's Login and Logout on the Application Interface

5.4.4. Display Settings Sequence Diagram

Display settings sequence diagram describes the work flow between the player, application interface and the operating system after the player's request to update visual and audio-related properties of the application, such as switching to full screen, turning game sounds on and off etc.

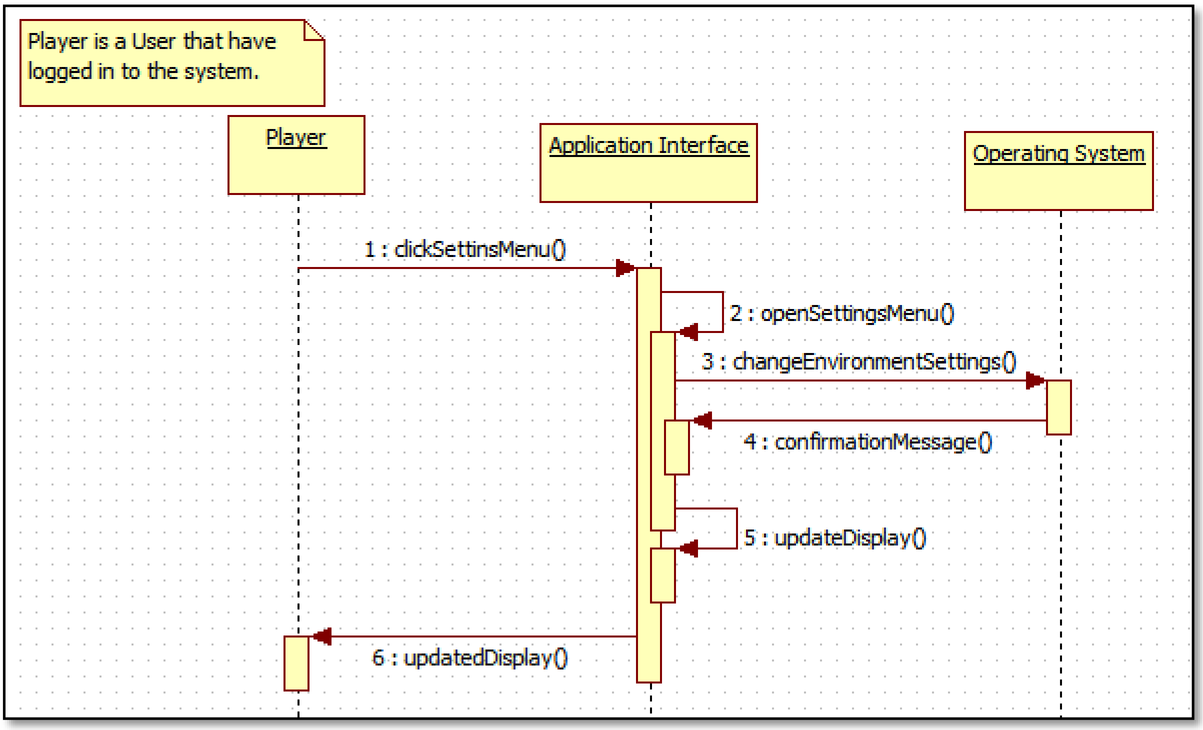

Figure 20 – Display Settings Sequence Diagram

5.4.5. Profile Settings Sequence Diagram

Profile settings sequence diagram describes the data flow after the player's request to update his/her personal profile details.

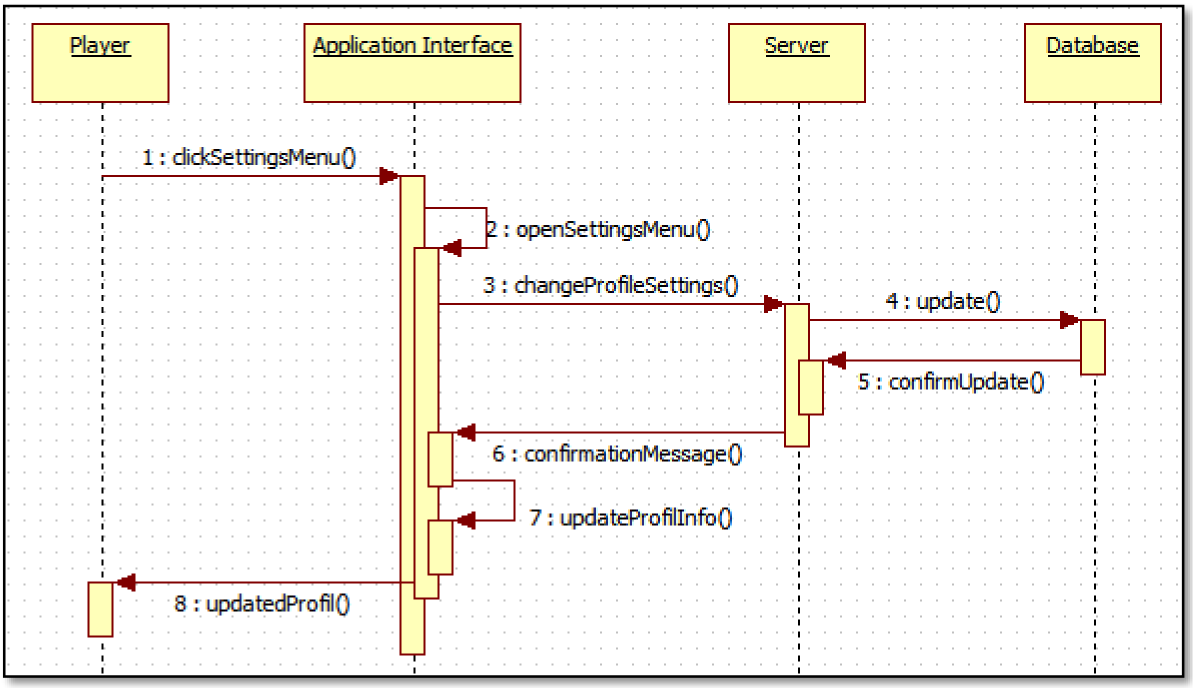

Figure 21 – Profile Settings Sequence Diagram

5.4.6. Training Mode Sequence Diagram

Training mode sequence diagram describes flow of data between application interface, server and database from the start till the end of a game in training mode. In training mode, player tries to hit a punch bag to gain experience points. Since the mode is single-player, there is no heavy communication between the application and server. Most interactions are performed within the application itself.

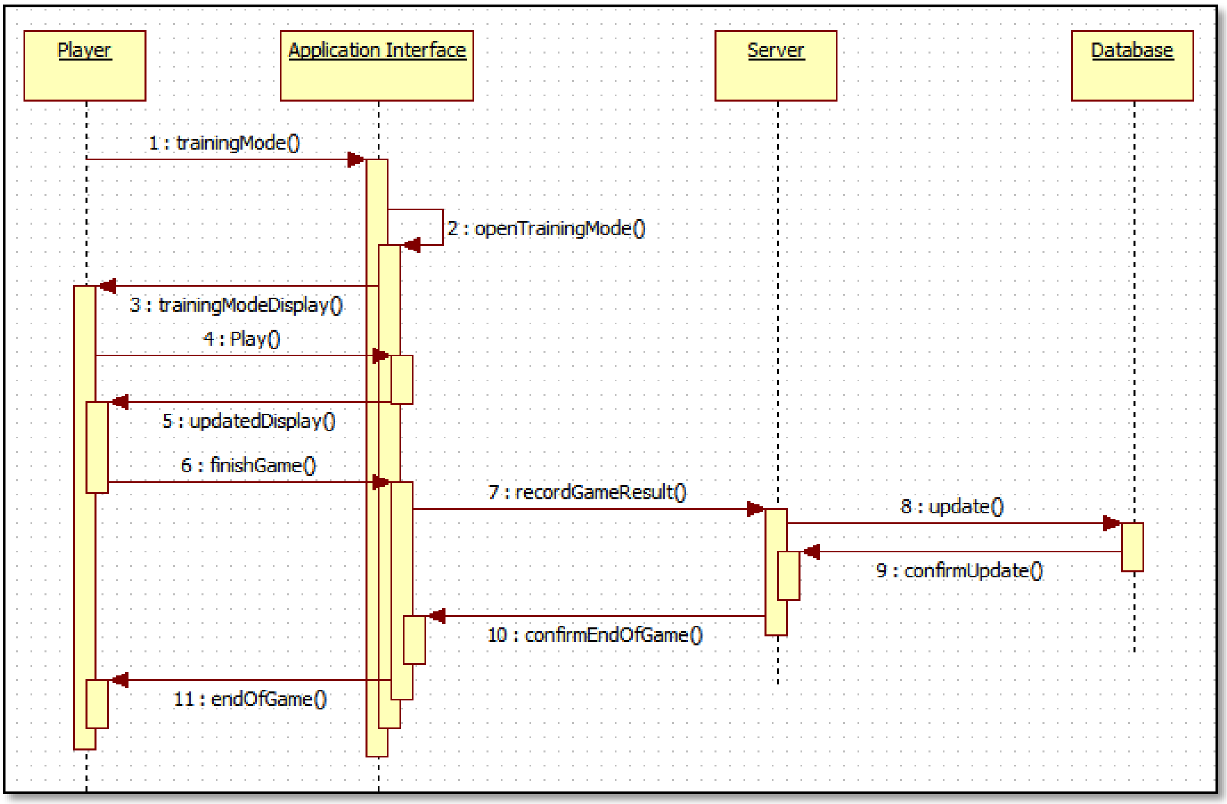

Figure 22 – Training Mode Sequence Diagram

5.4.7. Match Mode Sequence Diagram

Match mode sequence diagram describes interaction of the application with the game server during a game in match mode. In match mode, there two players each connected to server start a game. The players fight against each other with the aim of being the winner. Since the mode is multiplayer, communication and data flow goes through the game server, so that the game runs in synchronization on both displays.

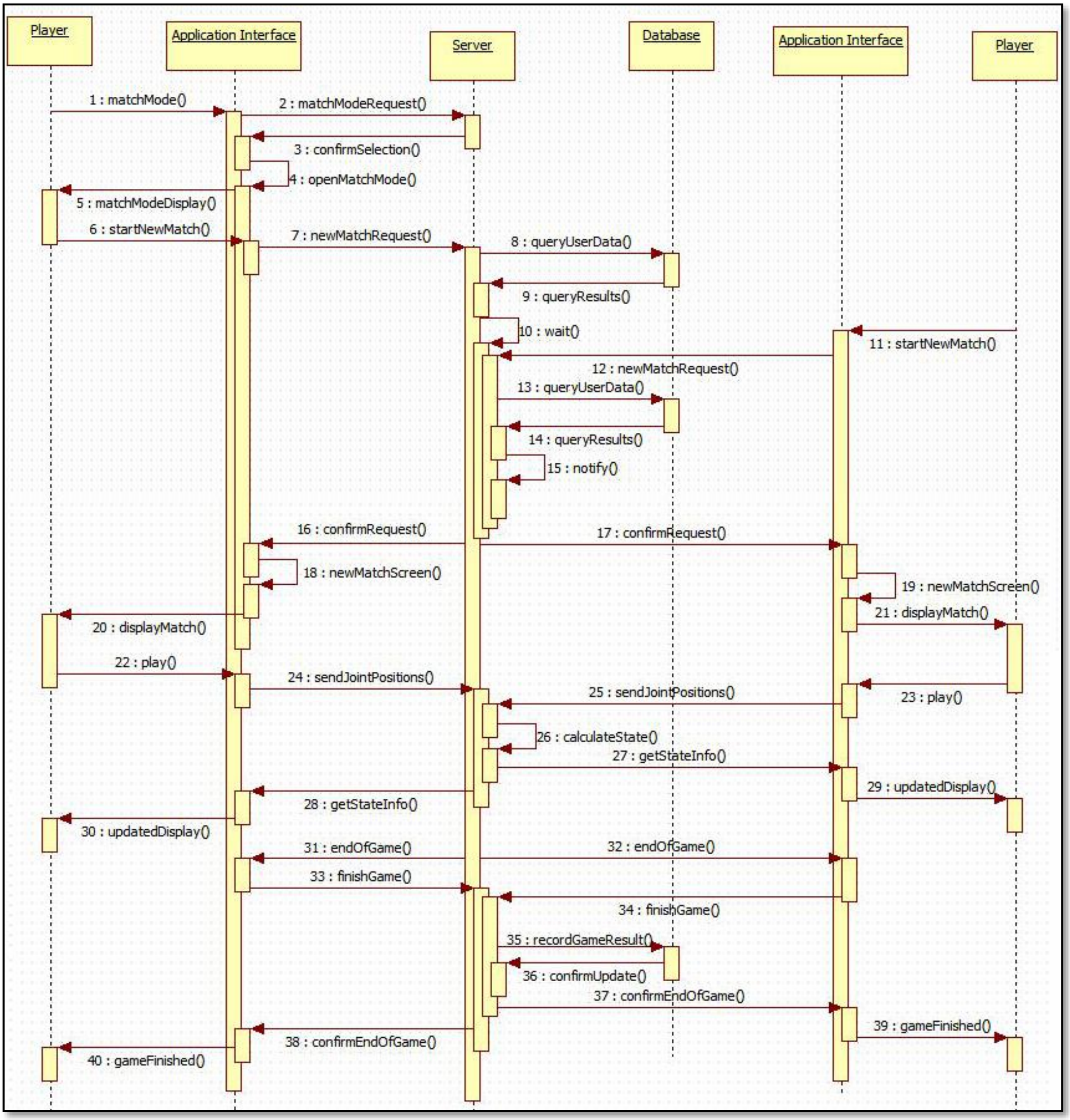

Figure 23 – Match Mode Sequence Diagram

5.5. INTERFACE VIEWPOINT

The interface viewpoint provides information to designers, programmers and testers the means of how to correctly make use of individual components of the system, and the way they communicate. In this section, details of external and internal interfaces are explained. A component diagram representing the ways of communication between system interfaces is available in Figure 27, under the *Composition Viewpoint* section (Section 5.6).

5.5.1. User Interfaces

There are two main environments using which users interact with the system: Game website and application platform. Within the application platform, there are several interfaces which provide user required services to control the application. User interfaces are designed using several facilities and methods provided by the Unity3D game engine.

Details of the user interfaces are explained in the SRS document. Thus, their specifications are just briefly mentioned in this document. It is worth to be noted that the described interfaces are just templates at this stage of the project. Although the functionality can be expected to be preserved, visual elements may be modified.

5.5.1.1. Game Website

Game website is where a new user can register to the application, find detailed information and instructions related to game, and download the game platform. For an already registered user, game website is where he/she can follow updates and news related to the game. User interact with the website using keyboard and mouse, and by clicking buttons and links provided on the interface.

5.5.1.2. Application Welcome Screen

The welcome screen is the main screen that appears once the user runs the application on his/her personal computer. The screen has user name and password fields which user needs to fill in order to login to the game.

5.5.1.3. Application Main Menu

Once the user signs in, the application's main menu is displayed. The main menu has buttons which forward the user to screens for game settings, profile settings, game play in different modes and game exit.

5.5.1.4. Application Settings Menu

The settings menu provides the user several options for updating his/her personal profile, and changing display and audio settings in the game.

5.5.1.5. Training Mode

In the training mode of the game, there is a box ring with the game character and a punch bag on it. The game character tries to hit the punch bag in order to gain experience points. There are small figures at the sides of the screen that shows the character's progress during the game.

While interaction of the user with other menus of the application is achieved by simple hardware like keyboard and mouse, user controls the game character via Kinect sensor in this mode. The Kinect sensor perceives movements of the user, and the game character's pose is updated accordingly. A sample image from the training mode is available in Figure 24.

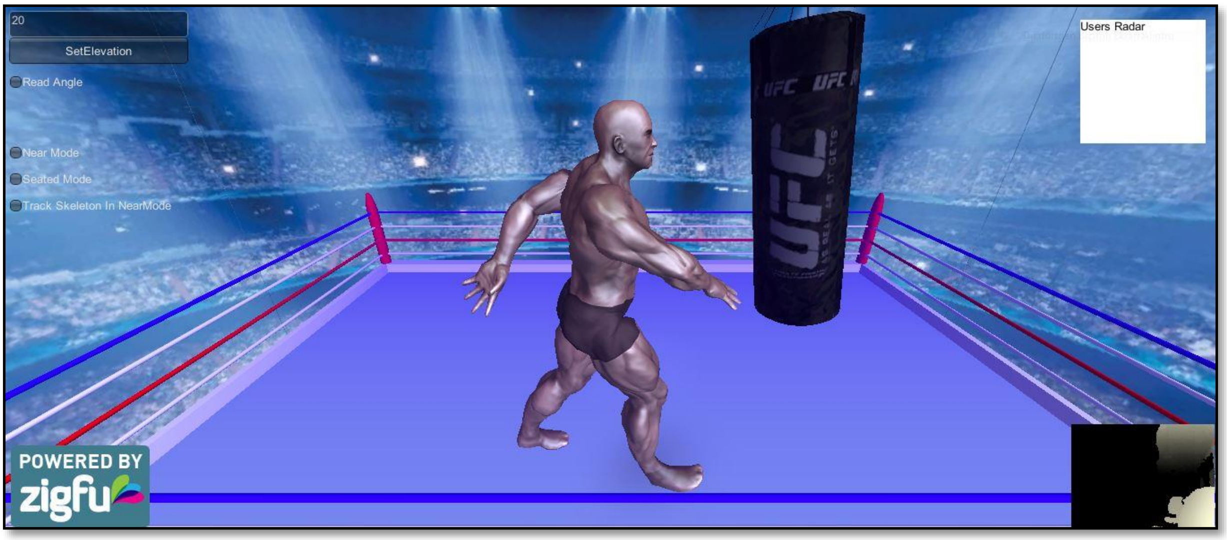

Figure 24 – Training Mode Sample Interface

5.5.1.6. Match Mode

In the match mode, there is an underground-like environment with two fighters and audiences around them. The fighters fight against each other trying to be the winner. There are simple figures at the sides of the screen that show characters' health and progress during the game. Similar to the training mode, characters are controlled by Kinect sensor in the match mode. A sample image from the match mode is available in Figure 25.

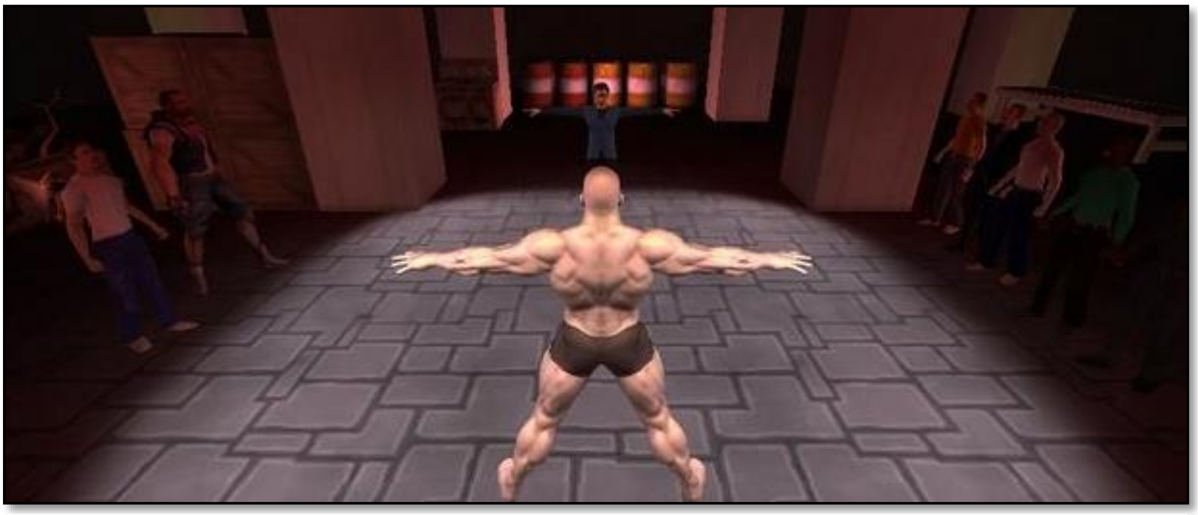

Figure 25 – Match Mode Sample Interface

5.5.2. Kinect – Application Interface

Kinect sensor is required to control the game character in several game modes. Kinect has position and depth sensors that can perceive movements of the user and report them as joint coordinates. In this project, methods provided by Zigfu Kinect SDK are used in order to be able read joint coordinates and use them in controlling the character. Received coordinates are mapped to several points in the character's skeleton and the point positions are updated accordingly.

5.5.3. Application – Server Interface

Communication between the application and game server is achieved by sending and receiving data packages via TCP/IP protocol. The game server is written in Python, and the client is in C#. Each client sends data packages to server, and server sends packages to each client after making several computations and comparisons. By this way, synchronized game with more than one clients can be achieved.

5.5.4. Server – Database Interface

One of the responsibilities of the game server is updating game database, inserting new entities, and executing queries. Communication between the game server and the database is achieved using MySQL statements.

5.6. COMPOSITION VIEWPOINT

We are aiming to explain physical and logical components of our system by introducing composition viewpoint. UML is used under this title. There is a deployment diagram below which illustrates the physical structure of our components and connections between them.

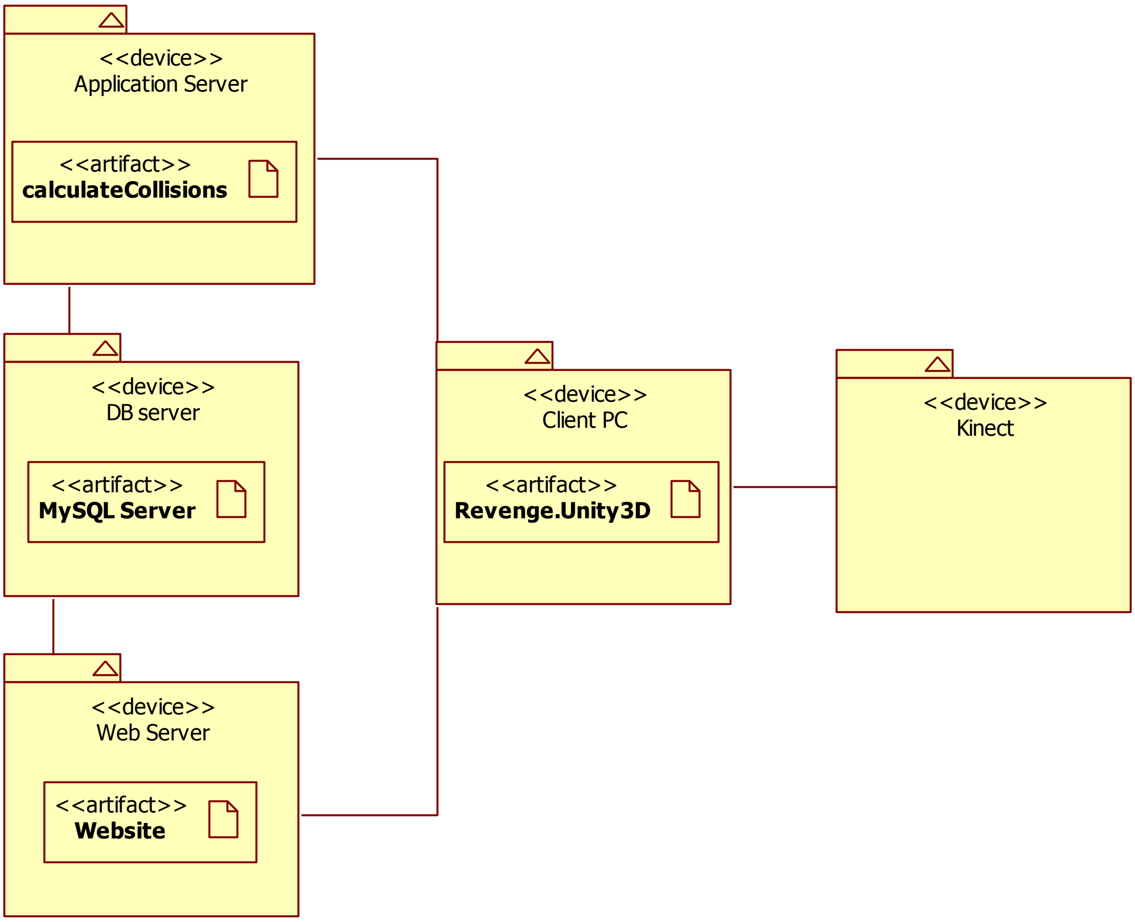

Figure 26 - Deployment Diagram

Aplication server : Basicly calculates the collusions and returns the results to the clients.

DB server : Data of every client is hold at DB server. MySQL Server is used as DBMS.

Web server : Web side of our game which is used to sign up and download the game.

Logical relation between components are showed the component diagram below.

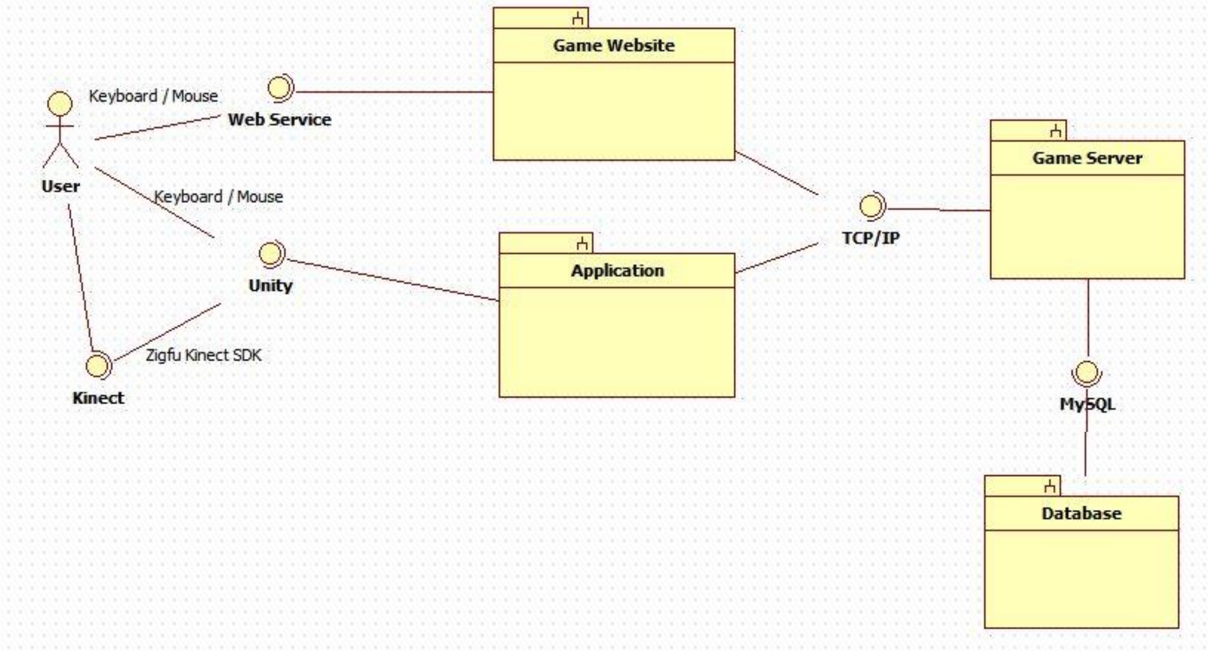

Figure 27 - Component Diagram

5.7. STATE DYNAMICS VIEWPOINT

There are eight several states in Revenge. All of them is in gameplay scenario. We assume that user access to our web site for registration our game. After registration, player download game from our web site. This states are included in web page. States flows of Revenge gameplay is illustrated in figure 28.

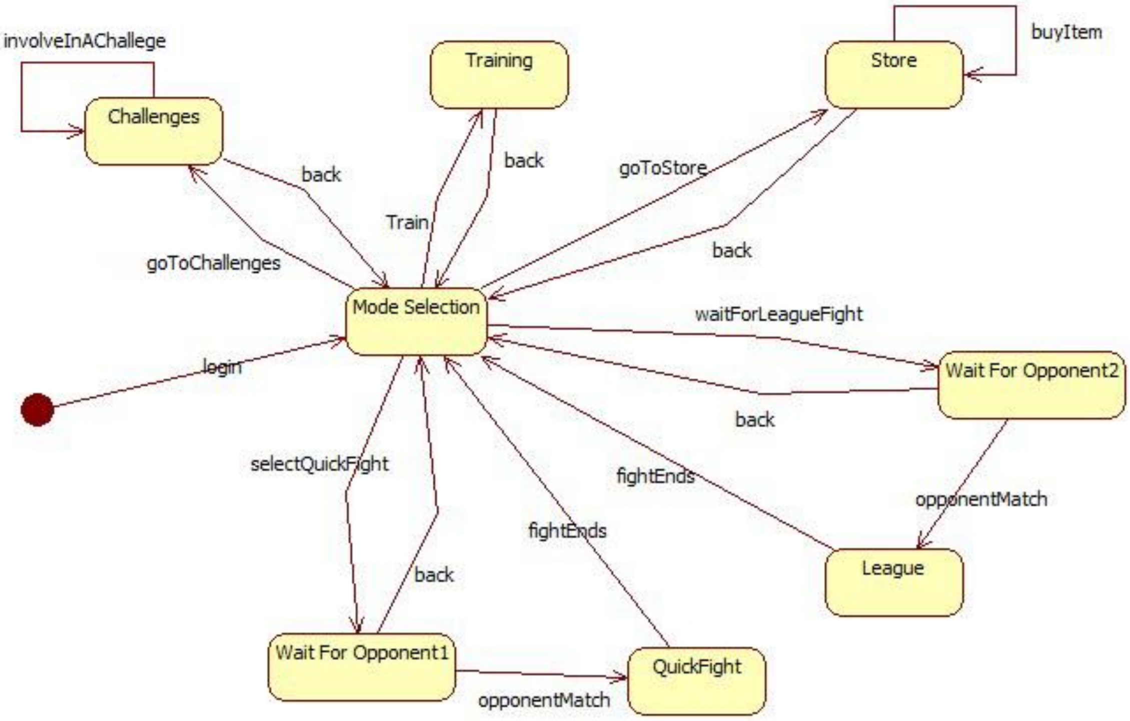

Figure 28 State Chart Diagram of Revenge Gameplay