Drawing Based Game Project

Software Requirements Specification

(In accordance with IEEE 830-1998)

v1.0

DynaDraw

Barış ÖZÇELİK	**1881804**
Bünyamin SARIGÜL	**1881440**
Cemil Kocaman	**1881333**
Turan SOYUER	**1881481**

January 8, 2016

Preface

This document contains the software requirements specification for the “ Drawing Based Android Game” project. The document is prepared according to the “830-1998 IEEE Recommended Practice for Software Requirements Specifications” , IEEE Computer Society, 1998.

In this Software Requirements Specification, a complete description of all the software requirements and views of the “ Drawing Based Android Game” project can be found.

While the first section of this document includes a complete description about scope and purpose of the project, the other sections give detailed description and explain requirements of the project.

Table of Contents

1. Introduction 6
1.1 Problem Definition 6
1.2 Purpose 6
1.3 Scope 6
1.4 Definitions, Acronyms and Abbreviations 7
1.5 References 7
1.6 Overview 7
2. Overall Description 7
2.1 Product Perspective 8
2.1.1 System Interfaces 8
2.1.2 User Interface 8
2.1.3 Hardware Interface 9
2.1.4 Software Interfaces 9
2.1.5 Communication Interfaces 9
2.1.6 Memory 10
2.1.7 Site Adaptation Requirements 10
2.2 Product Functions 10
2.2.1 Starting Scene 10
2.2.2 Offline-User Use Case 11
2.2.3 Online-User Use Case 12
2.3 User Characteristics 13
2.4 Constraints, Assumptions & Dependencies 13
3. Specific Requirements 13
3.1 Interface Requirements 13
3.2 Functional Requirements 19
3.2.1 Game Device Component Requirements 19
3.2.2 Server Component Requirements 20
3.3 Non-Functional Requirements 20
3.3.1 Performance Requirements 20
3.3.2 Safety 20
3.3.3 Security 21

3.3.4 Software Quality Requirements 21
3.3.5 Design Constraints 21
4. Data Model and Description 21
4.1 Data Objects 21
4.2 Data Dictionary 22
5 Behavioral Model and Description 24
5.1 Description for Software Behavior 25
5.2 State Chart Diagram 25
6. Planning 26
6.1 Team Structure 26
6.2 Schedule 26
6.3 Process Model 27
7. Conclusion 27

Table of Figures

Figure 1: Start Scene Use Case 10
Figure 2: Offline User Use Case 11
Figure 3: Online User Use Case 12
Figure 4: Start Scene Mock-Up 14
Figure 5: Login Scene Mock-Up 15
Figure 6: Register Scene Mock-Up 16
Figure 7: Drawing Scene Mock-Up 17
Figure 8: Play Scene Mock-Up 18
Figure 9: Profile Menu Mock-Up 19
Figure 10: Class Diagram 22
Figure 11: State Chart Diagram 25
Figure 12: Agile Method 27

1. Introduction

This document is a Software Requirements Specification for a “Drawing Based Android Game” for children. In this document, definition of the problem and introduction of the purpose and scope of this document is stated first. Then, an overall description is given. After that, specific requirements, data models and behavioral models are explained. Finally, process model, team structure and the planning of the project with a basic schedule is provided.

1.1 Problem Definition

Today, there are many computer and mobile games in the world. However, many of these games are harmful and not good for children. They steer children to violence as showing it like an entertaining thing. Also, they restrict children’s freedom and do not care about children’s imagination. With this senior project, we will deal with this problem.

1.2 Purpose

This software requirements specification document includes detailed description of the system architecture, functionalities and requirement specifications of the project. Thus, this document is a guideline for both the developer team and the users.

There may be changes in this document in the future in order to show the changes related to requirements and specifications with a new version number on the first page.

1.3 Scope

The final product of this project is an Android Application which is a game for children. The game provides children a place to use their creativity without restrictions. Children will be able to draw their own heroes, companions or animals.

Firstly, game asks children to draw a character in the Drawing Screen. After that drawing part, the game identifies a rigid body and creates a skeleton schema for it. Then, joints are specified. By using image processing techniques, game identifies the parts of body as head, hands, legs, wings, etc. Then, the drawing is added to Game Screen. Finally, the drawn character moves according to its joint points that we have identified and body parts that we have specified.

The project will be done by a team of 4 people in 2 semesters.

1.4 Definitions, Acronyms and Abbreviations

Term	**Definition**
Joint	The movable part where two bones or elements of a skeleton join
IEEE	The Institute of Electrical and Electronics Engineers
Rigid Body	An idealization of a solid body without deformation
Unity	A game engine for multi-platform game developing
MySQL	A database querying language

1.5 References

[1] American Academy of Child & Adolescent Psychiatry, Children and Video Games: Playing with Violence (Facts for Families, updated Aug. 2006). American Academy of Pediatrics. “Policy Statement—Media Violence,” Pediatrics (Nov. 2009): Vol. 124, No. 5, pp. 1495–503.

1.6 Overview

This document has 6 main parts hereafter. These are respectively Overall Description, Specific Requirements, Data Model and Description, Behavioral Model and Description, Planning and Conclusion.

2. Overall Description

Overall description of this project is explained in below sections with detail.

2.1 Product Perspective

In this game project, users can create account and login by using their accounts. By using their accounts, they can share characters they created or they can see high scores and ranking tables. Also they will also be able to play without login in offline mod.

Firstly, the game asks user to draw a character in “Drawing Scene”. It offers tools namely brush and eraser in order to create a drawing. There are also color palette, “Undo” and “Clear” buttons. After finished drawing, user finalizes character by tapping “Next” button. Then, we process this character and create an animation for it.

In the “Playground Scene”, user can move the drawn character by tapping on the screen. Also, user can add physical objects by drawing in this scene.

2.1.1 System Interfaces

The system has two components.

2.1.1.1 Game Device Component

Unity is multi-platform friendly. Thus, we are able to create the game for PC, Android, iOS and Web. Having a device of these kinds are enough to run application.

2.1.1.2 Server Component

Our system has a server component which keeps user information within the server component. Users interact with each other by using this server.

2.1.2 User Interface

User interface of the game will be simple and easy to use.

In the first screen, there are 3 buttons namely “Login”, “Register” and “Play Offline”. User can create an account or login with account information by using these buttons.

Other two screens are about login and register which have input fields for information.

Another screen provides a drawing area and tools which help drawing. User create a hero in this screen with the help of brush and eraser and also “Clear” and “Undo” buttons. Then the drawing is finalized after tapping “Next” button.

Also, there is a “Profile Screen” that shows user’s information and statistics. User can also see ranking tables and high scores of his/her friends in this screen.

Finally, there are several “Playground” screens which has different game environment for each different level of game. User interacts with his/her character in this screen by tapping or adds objects to the game by drawing.

2.1.3 Hardware Interface

The system works as mobile, desktop and web application without any specific hardware requirement. However, a reasonable CPU speed and sufficient amount of memory is expected for the efficiency of computations and smooth flow of user interactions.

2.1.4 Software Interfaces

Name	**Version**	**Source**
Windows	7/8/10	windows.microsoft.com
Android	From 4.2	android.com
iOS	From 8.0	developer.apple.com
Web Player	Web Browser	chrome.google.com
Unity	5	unity3d.com
Visual Studio	2015	visualstudio.com
OpenCV (EmguCV)	3.1 (3.0)	opencv.org

2.1.5 Communication Interfaces

By using TCP, data will be exchanged between server and client. Server will use MySQL while getting information from database.

2.1.6 Memory

Game will make calculations and inferences based on data collected from previously executed operations, so a reasonable amount of memory shall be used. Also data related to each registered user will be kept in the database so, a memory proportional to the user number shall be used in the database.

2.1.7 Site Adaptation Requirements

Game requires just an OS on an environment to work. There is no other specific requirement for site adaptation.

2.2 Product Functions

2.2.1 Starting Scene

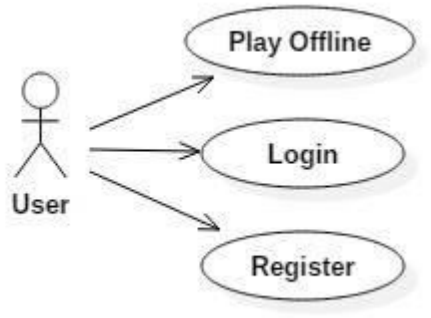

Figure 1: Start Scene Use Case

Use Case	Description
Play Offline	User can choose the playing in offline mode within this option
Login	If user has an account and wants to play in this mode, user can choose this option
Register	If user wants to play in online mode and he/she doesn't have account, user can choose this option to create new account

2.2.2 Offline-User Use Case

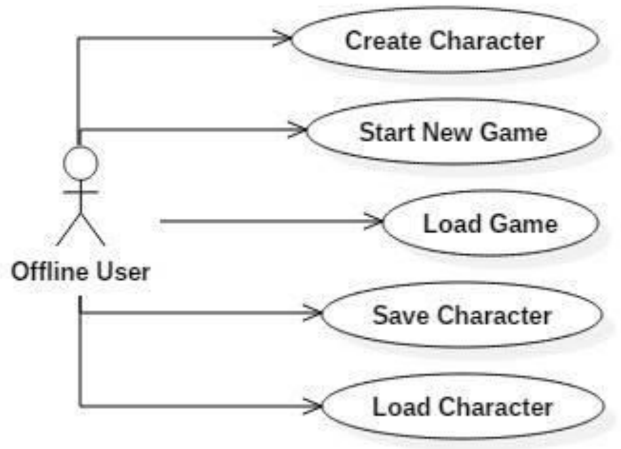

Figure 2: Offline User Use Case

Use Case	Description
Create Character	User has to create character for playing game
Start New Game	User has to start new game if there isn't any game that started before, or user can choose this option to start new game
Load Game	User can load old game to continue
Save Character	User can save character to characters. Within this option, character can be choose again in next games. In addition to that, if character is saved, user can share it with user's friends.
Load Character	User can load character from characaters ,that saved before, to play again.

2.2.3 Online-User Use Case

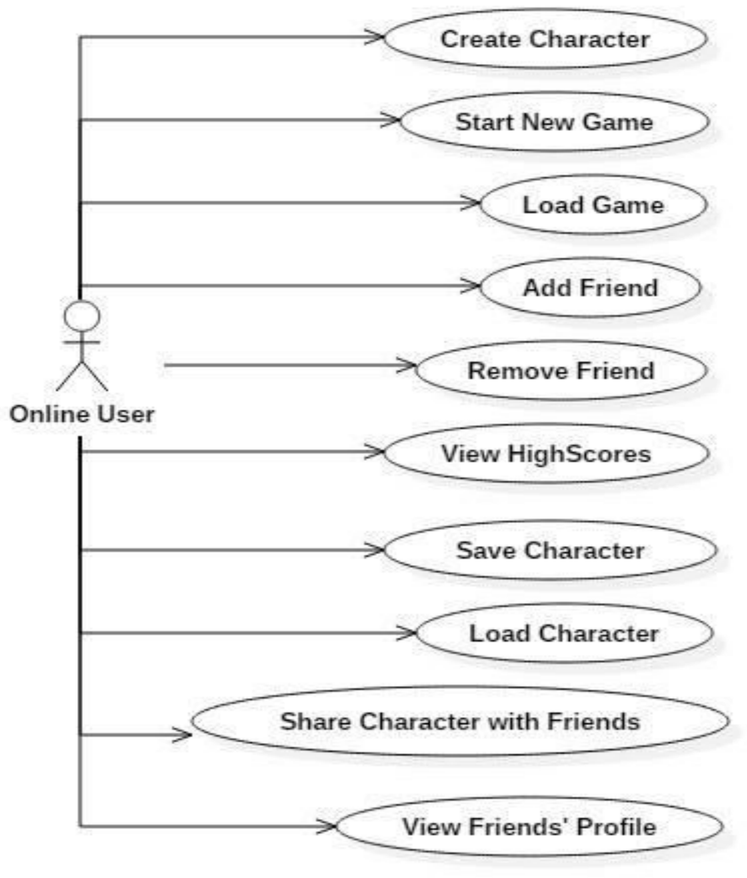

Figure 3: Online User Use Case

Same options that offline users had will be same with online users.

Use Case	Description
Add Friend	User can add friends
Remove Friend	User can remove friends
View HighScores	User can see high score table amoung friends
Share Chracter with Friends	User can share character with user's friends, so other users can play with this shared character
View Friend's Profile	User can view friends' profiles and can see their scores and characters.

2.3 User Characteristics

Our target audience is the student level children. We choose this group, because our game is depend on nature physic system and our target audience can perceive the nature laws more clearly. Again we choose this age level, because our goal is increase children creativity. Our expectation from children is to use their creativity for character creation and success at game levels. In this way, can can improve their creativity.

2.4 Constraints, Assumptions & Dependencies

- Although our game is multi-cross platform, it is best suitable for desktop and tablets. Because, game is depend on drawing skills and drawing on this devices is more convenient.
- Android version 4.2 or upper versions are needed.
- Increasing of number of users will lead to more server capacity in later.

3. Specific Requirements

3.1 Interface Requirements

The user interfaces described in section 2.1.2 can be found in below figures.

Start Scene

Component	Definition
Login	User logs in to his/her existing account
Register	User creates a new account
Play Offline	User can play without internet access

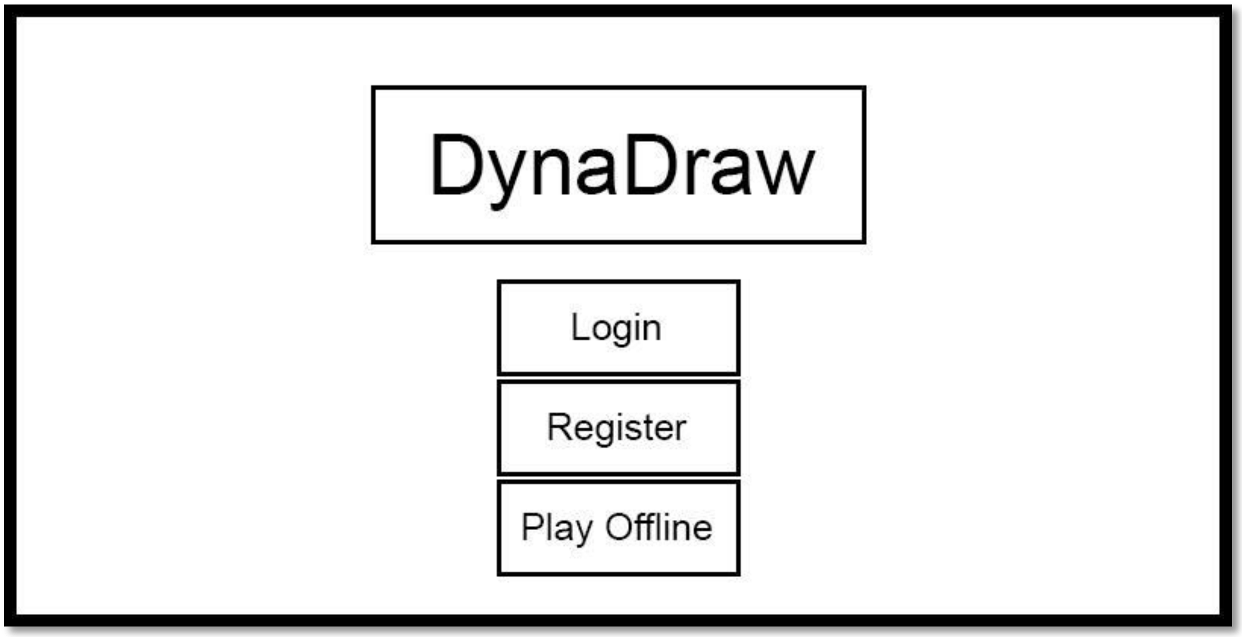

Figure 4: Start Scene Mock-Up

Login Scene

Component	Definition
User Name	Unique user name chosen by the User
Password	User's specific password
Login	Button that validates the username/password combination from the database and proceeds to the game if successful.

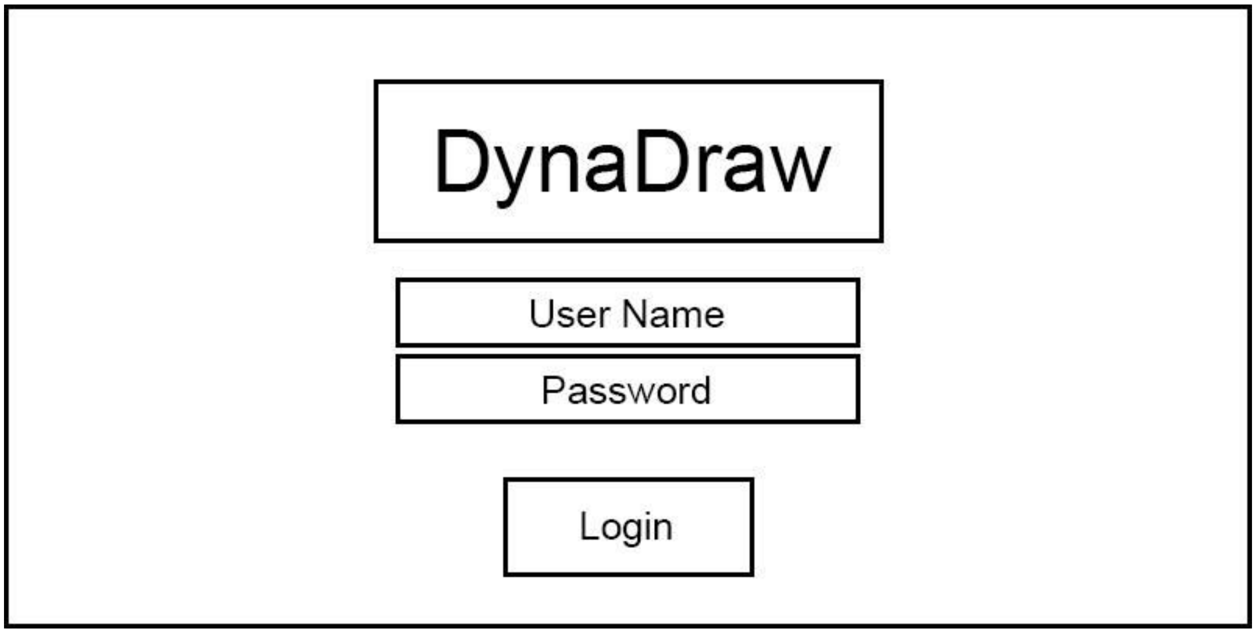

Figure 5: Login Scene Mock-Up

Register Scene

Component	Definition
User Name	Unique user name chosen by the User
Password	User’s specific password
E-mail	User’s e-mail address
Register	Button that validates the uniqueness of the user name and correctness of the e-mail address. Then, creates a new account if validation is succeeded.

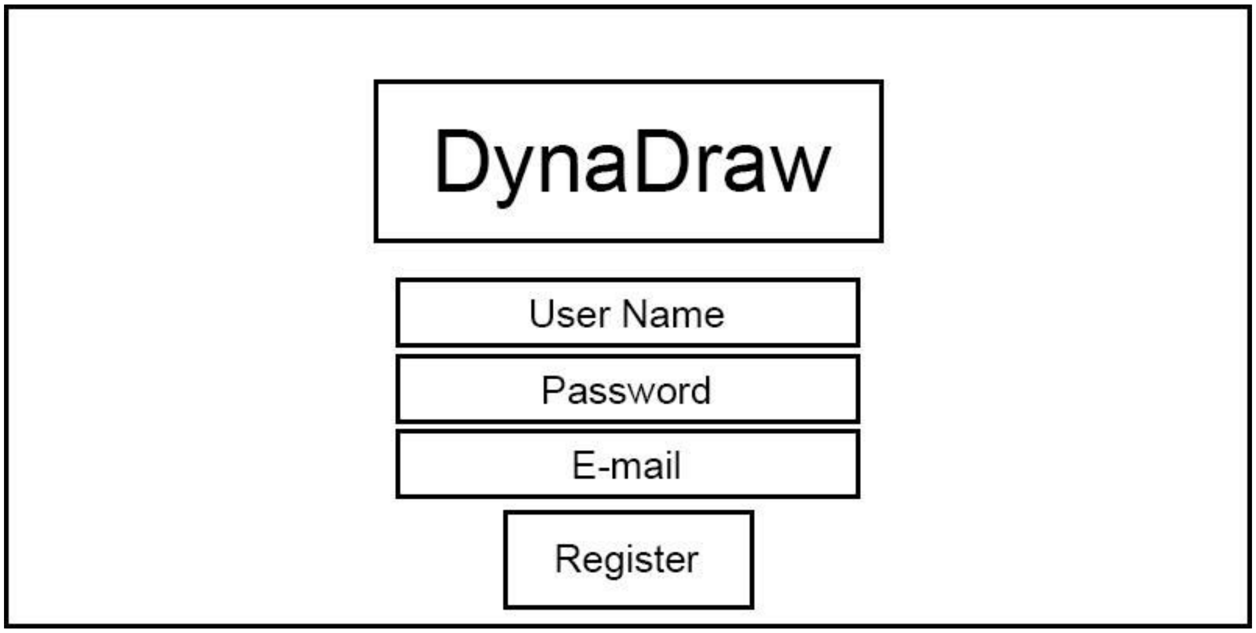

Figure 6: Register Scene Mock-Up

Drawing Scene

Component	Definition
Draw Line Button (Brush)	Switches to the drawing mode.
Color Buttons	Sets the color of the brush.
Remove Line Button (Eraser)	Switches to the erasing mode.
Clear	Clear all the lines drawn.
Undo	Undo the last action done by the user.
Next	Process the drawing and proceed to the game
Character Drawing Area	User draws his/her character on this area

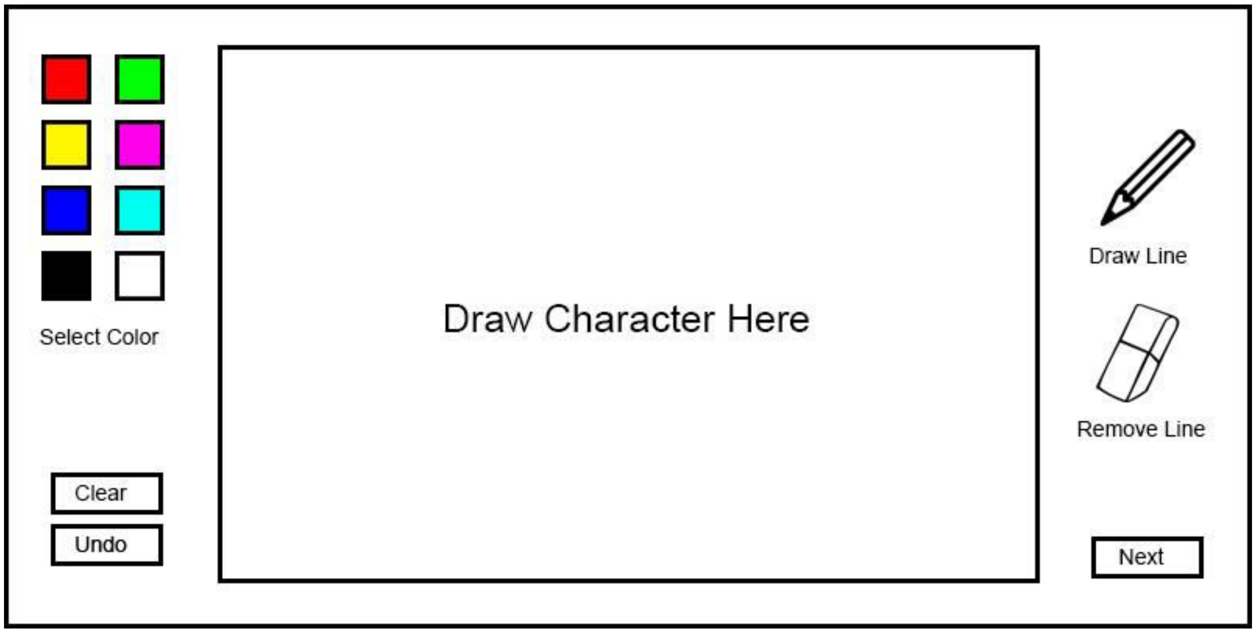

Figure 7: Drawing Scene Mock-Up

Play Scene

Component	Definition
Character	Animated character that drawn by the user previously.
Ground	Platform that have a collider. Characters move over these grounds.
Draw Object Button	Switches to drawing mode. User can draw lines on this mode to progress on the game like drawing bridges between grounds and moving over these bridges.
Move	Switches to moving mode. In this mode, user can control his/her character by touching the edges of the screen.

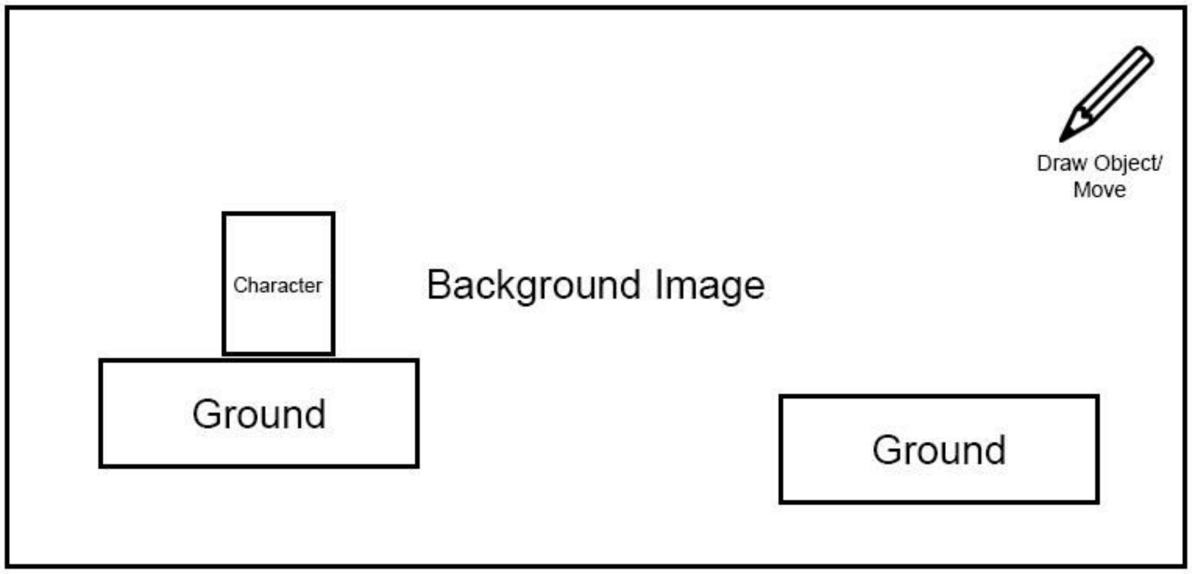

Figure 8: Play Scene Mock-Up

Profile Menu

Component	Definition
User Name	Unique user name chosen by the User
Points	Total points of the user
Level	Last level reached by the user
Characters	All stored characters f the user will be shown here
Friends	Friend list of the user
Back	Back to the game

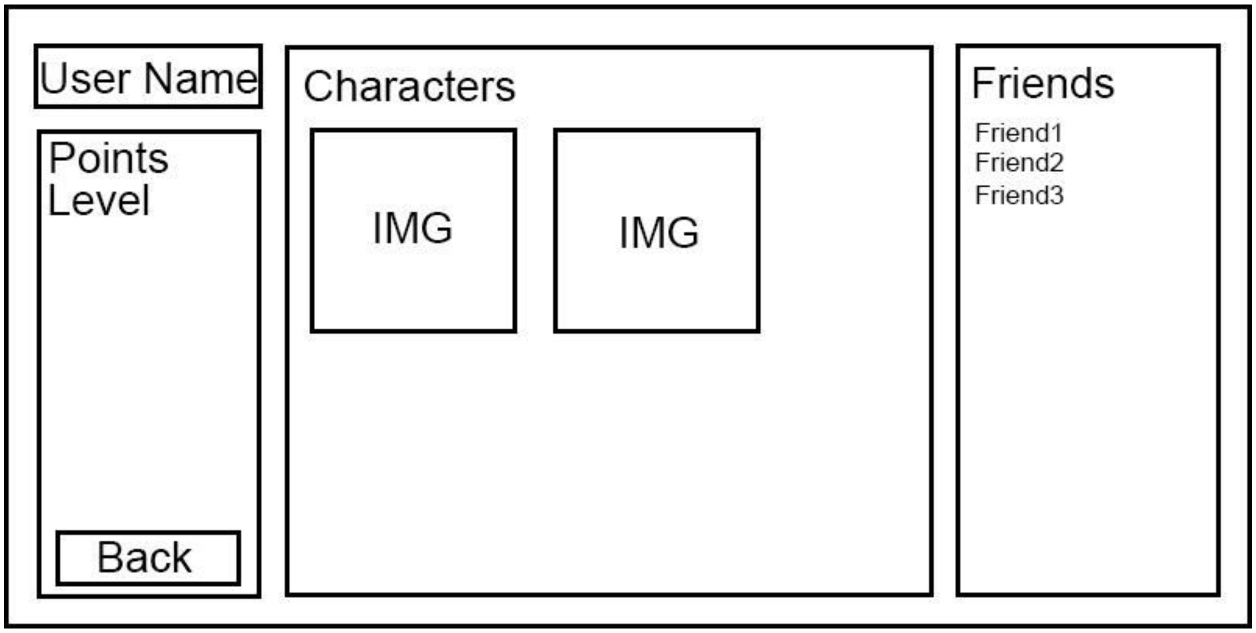

Figure 9: Profile Menu Mock-Up

3.2 Functional Requirements

3.2.1 Game Device Component Requirements

- User should create an account if he/she wants to interact with other users online.
- User may play offline if he/she does not have internet access.
- User should be able to draw his/her character.
- User should be able to choose a tool on drawing scene.
- User can choose colors while drawing.
- User can store his/her character.
- User can choose characters from the stored characters to play with it.
- User can add friends by their user names.
- User should be able to draw objects on the game scene to proceed in the game.
- User can check his/her score and high scores of his/her friends.

3.2.2 Server Component Requirements

- Server should assign specific user ID to all users.
- Server should store unique ID, unique name, password, e-mail, characters, score, level and ID of friends for each user.
- Server should check the username/password combination on every login.
- Server should be able to add/remove friends from the user.

3.3 Non-Functional Requirements

Major requirements are separated into performance, safety, security requirements subsections. Other related requirements are explained in software quality subsection.

3.3.1 Performance Requirements

- Server should response to login request immediately.
- Collecting point and color data should work parallel with drawing action.
- Edge detection and joint analysis should be done within 2 seconds for character animation.
- Since the game will not be multiplayer, there is not large bandwidth requirement.
- Server should be able to store at least 20 characters for each user.
- Server should collect score and level data after each level.
- On offline mode, user data and character storing should be limited to maintain the performance of the device.

3.3.2 Safety

- Since the targeted user profile is children, the project is not containing violence.
- Game music is suitable for children.
- The project will not contain any safety risks for children’s psychology.

3.3.3 Security

There is no security concern in this project since any critical information is not stored on the database. There will be only password protection for user profiles.

3.3.4 Software Quality Requirements

In sense of portability, we are able to publish the project on multiple platforms like Android, iOS and Windows.

System should be reusable since user should be able to load the game after re-entering the game.

3.3.5 Design Constraints

Since the aim of the project is improving the creativity of children, there is no visible limitation for user. On the other hand, it is not possible to identify all drawn characters but still there will be a sufficient animation for the character.

4. Data Model and Description

4.1 Data Objects

Class diagram of the project is demonstrated below:

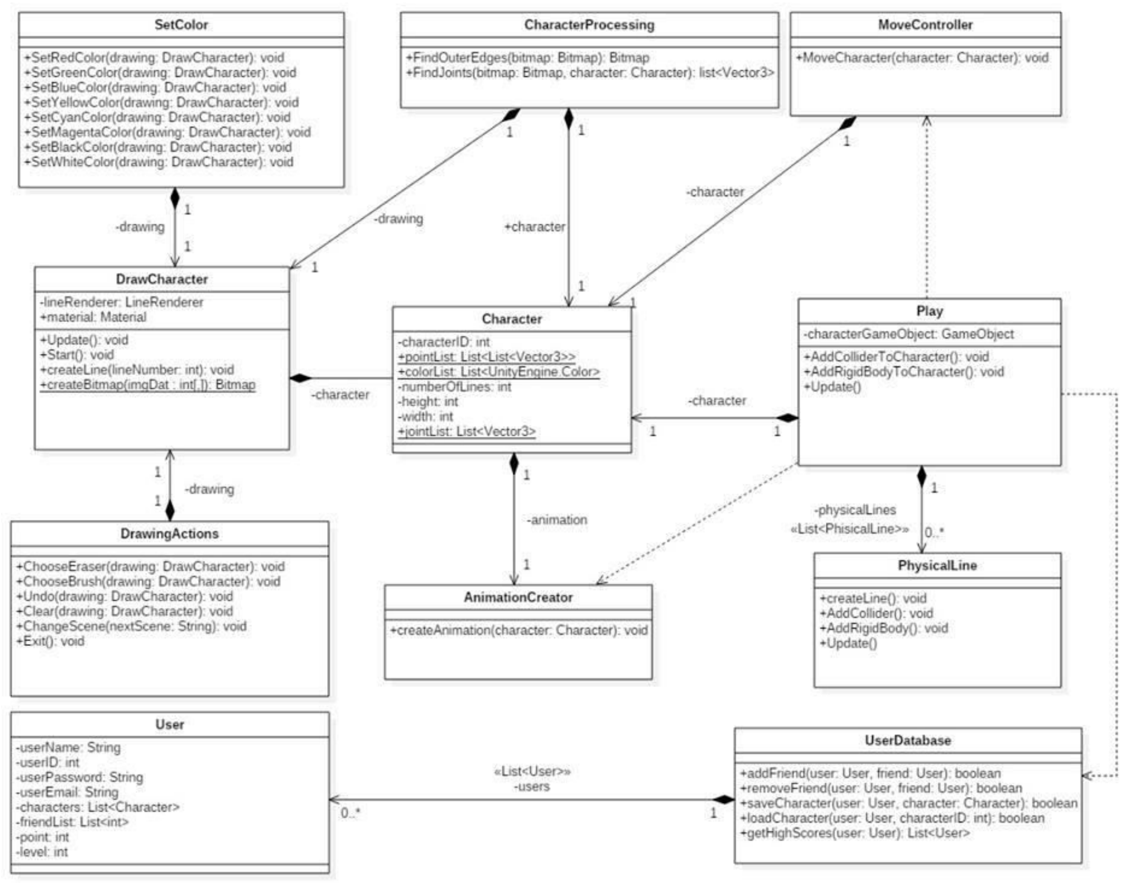
SetColor
+SetRedColor(drawing: DrawCharacter): void
+SetGreenColor(drawing: DrawCharacter): void
+SetBlueColor(drawing: DrawCharacter): void
+SetYellowColor(drawing: DrawCharacter): void
+SetCyanColor(drawing: DrawCharacter): void
+SetMagentaColor(drawing: DrawCharacter): void
+SetBlackColor(drawing: DrawCharacter): void
+SetWhiteColor(drawing: DrawCharacter): void
CharacterProcessing
+FindOuterEdges(bitmap: Bitmap): Bitmap
+FindJoints(bitmap: Bitmap, character: Character): list<Vector3>
MoveController
+MoveCharacter(character: Character): void
-drawing
-drawing
+character
-character
1
DrawCharacter
-lineRenderer: LineRenderer
+material: Material
+Update(): void
+Start(): void
+createLine(lineNumber: int): void
+createBitmap(imgDat : int[,]): Bitmap
-character
Character
-characterID: int
+pointList: List<List<Vector3>>
+colorList: List<UnityEngine.Color>
-numberOfLines: int
-height: int
-width: int
+jointList: List<Vector3>
-character
Play
-characterGameObject: GameObject
+AddColliderToCharacter(): void
+AddRigidBodyToCharacter(): void
+Update()
-drawing
-animation
-physicalLines
«List<PhisicalLine>»
0..*
DrawingActions
+ChooseEraser(drawing: DrawCharacter): void
+ChooseBrush(drawing: DrawCharacter): void
+Undo(drawing: DrawCharacter): void
+Clear(drawing: DrawCharacter): void
+ChangeScene(nextScene: String): void
+Exit(): void
AnimationCreator
+createAnimation(character: Character): void
PhysicalLine
+createLine(): void
+AddCollider(): void
+AddRigidBody(): void
+Update()
User
-userName: String
-userID: int
-userPassword: String
-userEmail: String
-characters: List<Character>
-friendList: List<int>
-point: int
-level: int
«List<User>»
-users
0..*
UserDatabase
+addFriend(user: User, friend: User): boolean
+removeFriend(user: User, friend: User): boolean
+saveCharacter(user: User, character: Character): boolean
+loadCharacter(user: User, characterID: int): boolean
+getHighScores(user: User): List<User>

Figure 10: Class Diagram

4.2 Data Dictionary

Class Name	Attiributes & Methods
General Explanation	✓ **Start ():** This method exists some classes which is required to make initialization. ✓ **Update ():** This method exists in every class and is called once per frame.
DrawCharacter	Attributes: ✓ **lineRenderer:**It is a game object component in Unity and is used to draw lines into scene. ✓ **material:** We use unlit/color material to determine color of the line renderer. Methods: ✓ **createLine(int lineNumber):** It creates a game object for the given line number and attach a line renderer component to it. ✓ **createBitmap(int [,] imgData):**It creates bitmap image according to character points and colors.

SetColor	Methods:The methods in this class changes the color of brush. ✓ **SetRedColor(DrawCharacter drawing)** ✓ **SetGreenColor(DrawCharacter drawing)** ✓ **SetBlueColor(DrawCharacter drawing)** ✓ **SetYellowColor(DrawCharacter drawing)** ✓ **SetCyanColor(DrawCharacter drawing)** ✓ **SetMagentaColor(DrawCharacter drawing)** ✓ **SetBlackColor(DrawCharacter drawing)** ✓ **SetWhiteColor(DrawCharacter drawing)**
DrawingActions	Methods: ✓ **ChooseEraser (DrawCharacter drawing):** It allows the user select eraser while drawing character. ✓ **ChooseBrush (DrawCharacter drawing):** It allows the user to select brush and to continue draw lines while drawing character. ✓ **Undo (DrawCharacter drawing):** It deletes the last drawn line. ✓ **Clear (DrawCharacter drawing):** It deletes the all lines in the drawing scene. ✓ **ChangeScene (String nextScene):** It changes the current scene to next scene given as argument. ✓ **Exit ():** It exists from the game when pressing Esc on computer or back button on the phone.
Character	Attributes: ✓ **characterID:** This is the unique character id for characters of each user. ✓ **pointList:** It stores all points for each line renderer which form the character. ✓ **colorList:** It stores color of each line renderer. ✓ **numberOfLines:** It is the count of line renderers. ✓ **height:** The height of the character. ✓ **width:** The width of the character. ✓ **jointList:** It stores the points which are the joints of the character.
CharacterProcessing	Methods: ✓ **FindOuterEdges(Bitmap bitmap):** It finds the outer edges of the character by using bitmap and return a bitmap. ✓ **FindJoints(Bittmap bitmap, Character character):**It finds the joint points of the character and return a list of Vector3 for each joint.
AnimationCreator	Methods: ✓ **createAnimation(Character character):** While the character is moving, this method is change the points of character lines to give an animation to it.
Play	Attributes: ✓ **characterGameObject:** It is a game object. All game objects which have line renderer components are child of this game object.

	Methods: ✓ **AddColliderToCharacter():** It adds box collider 2D to all lines of the character. By doing this, the character has the ability of making physical interactions. ✓ **AddRigidBodyToCharacter():** It adds the physical aspects to the character such as mass, gravity.
PhysicalLine	Methods: ✓ **createLine():** It creates a game object and attach a line renderer to it when the user draw a line to play scene. ✓ **AddCollider():** It adds polygonal collider 2D to drawn lines in game sceen. By doing this, the lines have the ability of making physical interactions. ✓ **AddRigidBody():** It adds physical aspects to the character such as mass, gravity.
User	Attributes: ✓ **userName** ✓ **userID** ✓ **userPassword** ✓ **userEmail** ✓ **characters:** It stores the characters drawn by this user. ✓ **friendList:** It stores the userID of the friends. ✓ **point:** The total game point of the user. ✓ **level:** User's current game level.
UserDatabase	Methods: ✓ **addFriend(User user, User friend):** It adds userID of the friend to friendList of the user. ✓ **removeFriend(User user, User friend):**It removes userID of the friend from the friendList of the user. ✓ **saveCharacter(User user, Character character):**It adds the character to the characters of the user. ✓ **loadCharacter(User user, int characterID):** It sets the current character of the user. ✓ **getHighScores(User user):** It returns the users have highest points among the user's friends.

5 Behavioral Model and Description

This section provides the overall behavior of the system. Major events and states are displayed in a state chart diagram.

5.1 Description for Software Behavior

Project has two options as online and offline playing. In both modes, the main progress of the game is same. On online mode, user has some additional functionality like adding friends or viewing highscores as stated earlier. These functions not added to state chart diagram for simplicity.

5.2 State Chart Diagram

The state chart diagram of the project is demonstrated below:

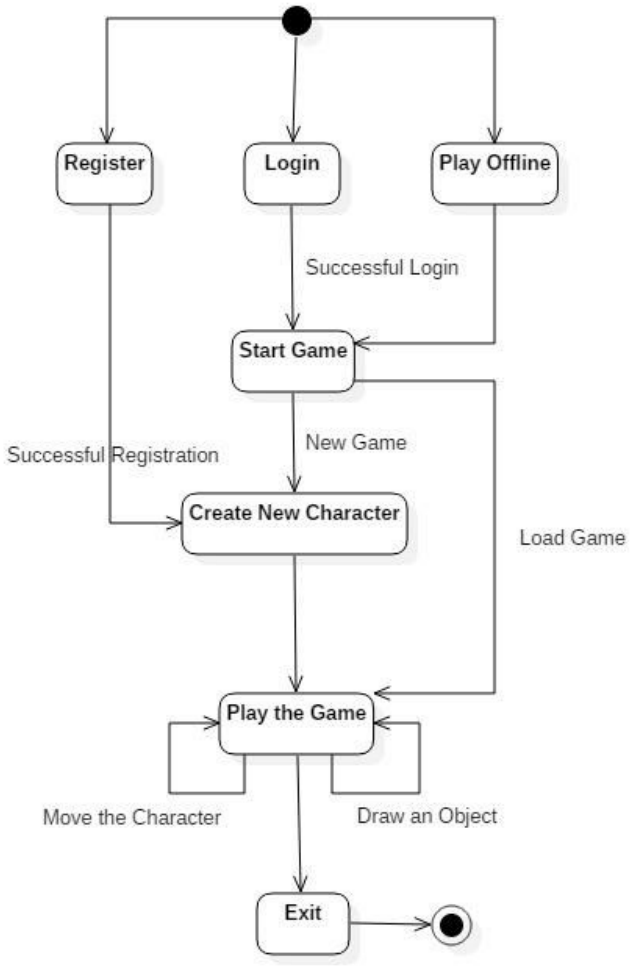

Figure 11: State Chart Diagram

Initially, in both online and offline modes, new players must create a character. Then, users can play with same character or draw new character each time.

For online mode, user should register with a unique user name and a valid e-mail address. Then, user can log in to game and pull his/her data from the database.

In the game playing, user should move the character and draw objects to solve problems and proceed in the game.

6. Planning

Planning about this project is explained in below sections.

6.1 Team Structure

We meet with Dr. Sibel Tarı and Serdar Çiftçi once a week to give information about project's current state and to get feedback. Our members are:

Bünyamin Sarıgül – Developer, Character Animation& Designer

Barış Özçelik –Developer, Game Physics

Turan Soyuer –Developer, Character Animation& User Interface

Cemil Kocaman – Developer, User Interface& Effects

6.2 Schedule

This project is planned to be finished by June 2016. This process is divided into parts according to major requirements of the project.

In two months character animation and improvements in game physics will be done.

By June 2016, the content of the game will be enriched.

6.3 Process Model

We use agile model for our project in order to make the system respond quickly respect to changes. This saves us from doing excessive work every time a requirement or specification changes. Agile method is an iterative approach which has planning, requirement analysis, design and testing. Each iteration takes about 3 weeks. At the end of the each iteration, testing is done and some changes are applied if tests fail or a new iteration starts if tests succeed.

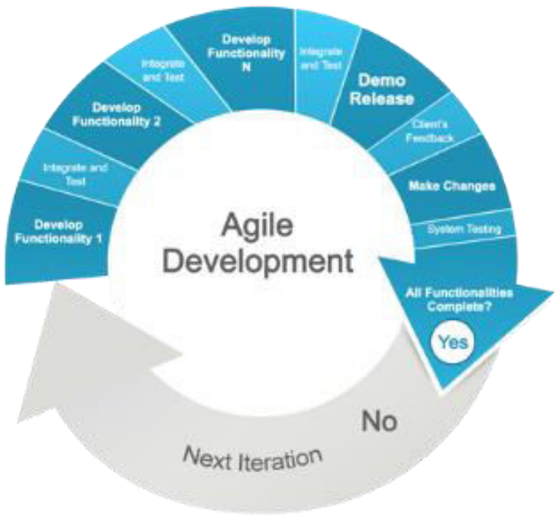

Figure 12: Agile Method

7. Conclusion

This Software Requirements Specification document is prepared to give a total understanding of “Drawing Based Game” project. Detailed system specification, functional and nonfunctional requirements, interfaces, data and behavioral models are stated clearly. Thus, this document explains the design and development aspects of the system to be developed.

This Software Requirements Specification document is a guideline for developers and users. Requirements are listed with detail in previous parts of this document.