# Software Requirements Specification

**for**

# Double Six

**Version 1.0 approved**

**Prepared by K.O.D.**

**K.O.D**

**December 9, 2018**

# Table of Contents

**1. Introduction ..... 1**
1.1 Purpose ..... 1
*1.2* Document Conventions ..... 1
*1.3* Intended Audience and Reading Suggestions ..... 1
*1.4* Product Scope ..... 1
1.5 References ..... 2
**2. Overall Description ..... 2**
*2.1* Product Perspective ..... 2
*2.2* Product Functions ..... 2
*2.3* User Classes and Characteristics ..... 3
*2.4* Operating Environment ..... 3
*2.5* Design and Implementation Constraints ..... 3
*2.6* User Documentation ..... 4
*2.7* Assumptions and Dependencies ..... 4
**3. External Interface Requirements ..... 5**
3.1 User Interfaces ..... 5
*3.2* Hardware Interfaces ..... 19
3.3 Software Interfaces ..... 19
*3.4* Communications Interfaces ..... 20
**4. System Features ..... 20**
4.1 Signing In ..... 20
4.2 Create Game ..... 21
4.3 Join Game ..... 22
4.4 Exit Game ..... 22
4.5 Start Mining ..... 23
4.6 Stop Mining ..... 23
4.7 Top Up ..... 24
4.8 Withdrawal ..... 24
4.9 Refresh ..... 25
4.10 Disconnect ..... 25
4.11 Exit ..... 26
**5. Other Nonfunctional Requirements ..... 27**
5.1 Performance Requirements ..... 27
5.2 Safety Requirements ..... 27
5.3 Security Requirements ..... 27
5.4 Software Quality Attributes ..... 28
5.5 Business Rules ..... 29
**6. Other Requirements ..... 29**

# Revision History

| Name | Date | Reason For Changes | Version |
|---|---|---|---|
| Double Six | 09/12/2018 | Initial Release | 1.0.0 |
| | | | |

# 1. Introduction

## 1.1 Purpose

As a gaming simple gaming mobile application, the Double Six is intended to provide entertainment to users who play. The game is made more interesting through the use of in-game tokens called “Virtual Quarters” which are transferred in between players based on their ranks in rounds.

## *1.2* Document Conventions

This document conforms to the IEEE Software Requirements Specification Template. All headings are bolded and larger than the body text in addition to being in the Times New Roman font whereas the body text is in the Arial font.

## *1.3* Intended Audience and Reading Suggestions

This document is intended for programmers as it includes many contains jargons and snippets of code that may not be understood by other audiences. To a lesser extent it may also appeal to product managers as some UML is used which is also used in product management.

## *1.4* Product Scope

Double Six is a gaming application enabling users to experience a real time dice toss with other players on their mobile devices.

Disclaimer and Business Strategy

Due to the Google’s recent anti-mining policy that strictly prohibits mining application from the Google Play Store, two versions of the Double Six app will be created. The two versions are one with mining functionality and one without mining functionality. This document will cover the Double Six app that includes the mining feature as this would be a more feasible approach. Users may optionally include cash in the games. The business strategy to be deployed consists of three phases. Phase one consists of uploading the Double Six App without mining to the Google Play Store. Phase two consists of waiting for a user base to use the app offered on the Google Play Store in addition to persuading users to use the app. Once a large enough user base has been gathered, Phase Three would be initiated. This phase includes informing the user base through indirect means (means that do not included Google) of a new, branched version of Double Six that includes the mining feature and hosting the Double Six App with the mining functionality

on an official Double Six server for users to access. This will allow users to download the application and use the new feature, which will generate a mining pool.

### 1.5 References

IEEE SRS Document Template

*https://web.cs.dal.ca/~hawkey/3130/srs_template-ieee.doc*

Google's Policy on Gambling Apps

*https://play.google.com/about/restricted-content/gambling/*

The URL above states that Google will not support apps that allow for mining on the mobile device. The criterion for gambling type apps is also provided.

# 2. Overall Description

### *2.1* Product Perspective

The combination of the simplicity of the Double Six game in addition to the rewards conforms to the stereotypical, simple addictive games. These are games that are both simple in nature require little system resources and as such makes them very addicting. The total downloads for these types of games dwarfs the total downloads for more complex and resource intensive games. The Double Six app would also be mutually beneficial for both the user and its creators. Users can earn money by trading in their in-game tokens called Virtual Quarters or VQs. As the name suggests one VQ is equal to USD$0.25. Users can earn Virtual Quarters through buying, mining, or by scoring them in matches. The creators would earn money through ads and a percentage of income from the mining pool.

### *2.2* Product Functions

The main functionalities of the Double Six application are:

- User Authentication
- Creating a Game
- Playing the Game
- Topping Up
- Withdrawals
- Mining

## *2.3* User Classes and Characteristics

The main incentive for using the Double Six app is earning cash. As such the user classes are based on this incentive.

The user classes to use the application are:

- Gamers – These users would be primarily engaged in dice tossing to earn tokens
- Miners – These users would be primarily engaged in mining to earn tokens
- Full-fledged Users – These users are engaged in a mixture of Gaming and Mining to earn tokens

## *2.4* Operating Environment

Double Six is intended for on mobile devices running Android OS, however an iOS version will be available in a future release. The Double Six Application, being a hybrid application, runs on most Android Versions such as Android Froyo, however the recommended version of Android is 4.4 (KitKat) API level 19.

Double Six would be built using Cordova and would make use of the following plugins to achieve the necessary native functionalities:

- cordova-plugin-background-mode 0.7.2 "BackgroundMode"
- cordova-plugin-badge 0.8.7 "Badge"
- cordova-plugin-device 2.0.2 "Device"
- cordova-plugin-dialogs 2.0.1 "Notification"
- cordova-plugin-local-notification 0.9.0-beta.2 "LocalNotification"
- cordova-plugin-statusbar 2.4.2 "StatusBar"
- cordova-plugin-whitelist 1.3.3 "Whitelist"
- cordova-plugin-x-toast 2.7.2 "Toast"

## *2.5* Design and Implementation Constraints

In order to provide an exchange service for users to buy or sell Virtual Quarters a payment service would have to be utilized. In order to provide a familiar and secure payment environment PayPal's Checkout and Payment is to be utilized. PayPal is also supported internationality in many countries. As such it would save time in implementing each card company's API (such as VISA and MasterCard) separately. Also PayPal has very good documentation, and PayPal's API has been heavily tested over the years making it a very reliable option. In terms of server hosting, the server would be hosted on Amazon's Cloud Platform called Amazon Web Services on a cloud server in the United States, since PayPal has the best support there.

## *2.6* User Documentation

The components of the User Documentation would be:

- User Manual – Contents of the User Manual would be included on the Google Play Store as the App description and on the Double Six website hosting the App with mining capabilities.

- FAQ – This would be a page on the Double Six official website

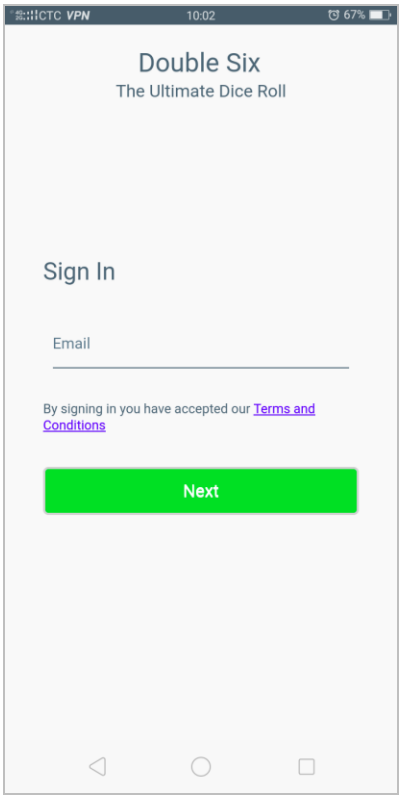
## *2.7* Assumptions and Dependencies

In order to access certain functionalities of the Double Six Application it is dependent upon whether user has a PayPal account or not. Since only adults are able to create a PayPal account it would be more difficult for a younger audience to access all the features of the App which is important because the App may attract unwanted skepticism if that were the case as a result of possible gambling. Users are assumed to have an email address as this is necessary for the verification process.

# 3. External Interface Requirements

## 3.1 User Interfaces

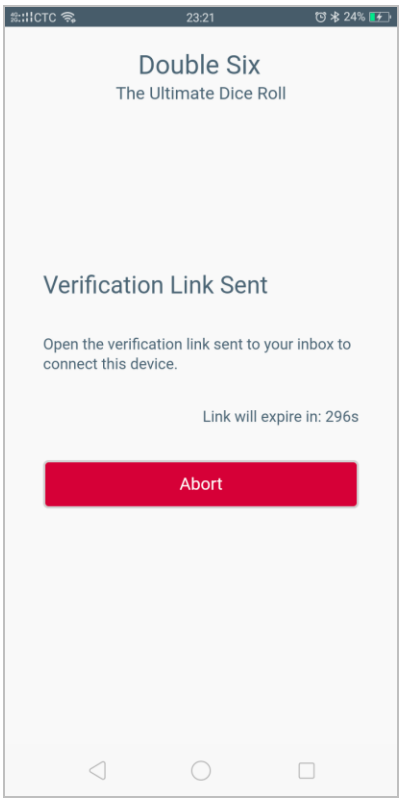
The Sign-Up Screen




*Sign-Up Screen*

The sign-up screen provides the user with an interface for providing their email address to connect their device. Once the user enters a valid email address and clicks next, an email containing a verification link will be sent to the user's email address. The application will then navigate to the verification page, awaiting the user to open the verification link. As you may have noticed there is no link for user registration. This enables user to log in without the hassle of remembering a password in addition to the security of using their existing email address.

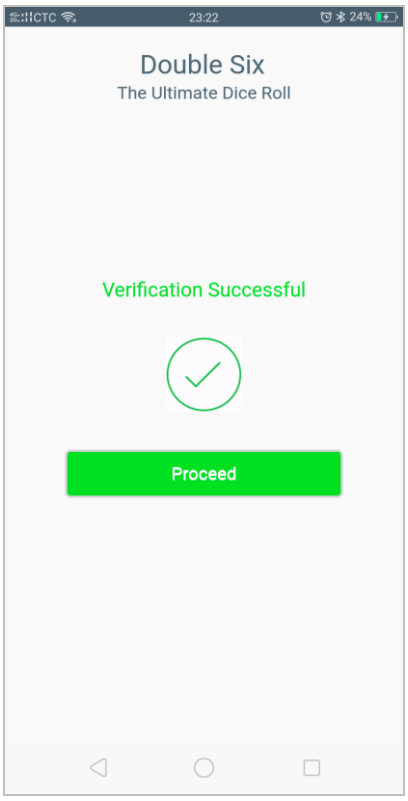
## The Verification Process Screen




*Verification Process Screen*

The verification screen is shown to the user upon submission of their email address. A countdown timer is shown to indicate the validation time of the verification link sent to the provided email address. If the verification link sent is opened before the timer elapses then the user will be taken to the verification success screen where they can proceed to the home screen. If the timer elapses then the verification link will become invalid and the user will be redirected to the sign in screen.

The Verification Success Screen



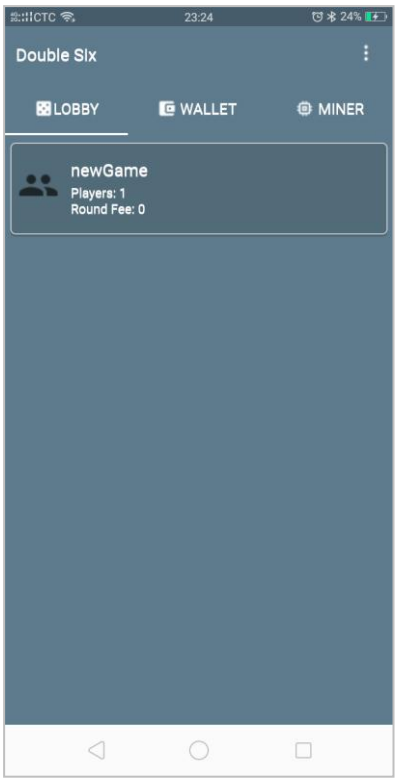

*Verification Success Screen*

Once a device connecting to the server with a specific email address has been successfully the verified the user will be redirected to the verification success screen where they can proceed to the home screen when desired.

```
const connectionRequest = function(email, poll_token){
    return new Promise(function(resolve, reject){
        $.ajax({
            "url": serverDomain+"connect",
            "type": "post",
            "data": {
                "email": email,
                "pollTkn": poll_token
            },
            "success": function(response){
                response = JSON.parse(response);
                if(typeof(response["pollTkn"]) != "undefined"){
                    localStorage["pollTkn"] = response["pollTkn"];
                }
                timer = parseInt(response["timeout"]);
                if(typeof(poller) != "number"){
                    poller = setInterval(function(){
                        connectionRequest(email, localStorage["pollTkn"]);
                        $("#timerHolder").html(timer);
                        timer--;
                    }, 1000);
                }
                $.mobile.navigate("#pending", { transition: 'slide', reverse: false});
                try{
                    if(timer <  4){
                        localStorage.clear();
                        poller = "";
                        abortConnect();
                    }
                    if(typeof(response["session_id"]) != "undefined" && typeof(response["session_tkn"]) != "undefined"){
                        //CONNETION SUCCESSFUL
                        localStorage["sess_id"] = response["session_id"];
                        localStorage["sess_tkn"] = response["session_tkn"];
                        localStorage.removeItem("pollTkn");
                        $.mobile.navigate("#success", { transition: 'none', reverse: false});
                        clearInterval(poller);
                        resolve(true);
                    }
                    else{
                        resolve(false);
                    }
                }
```

*Code snippet of how the verification process would be implemented*

The Home Screen




*Home Screen*

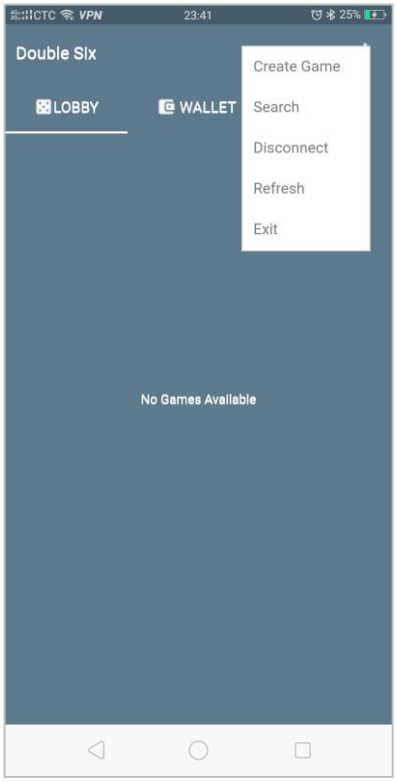
Once the user has successfully logged in, they can access the Home Screen which defaults to the Lobby Tab. The verification process would only have to be done once. Therefore each time they use the application they will have access to the home screen unless they remotely disconnect the device. For this process to occur however the user requires internet connection as their session token has to be verified.

```
const checkSession = function(){
    return new Promise(function(resolve, reject){
        $.ajax({
            "url": serverDomain+"checksession",
            "type": "post",
            "data": {
                "sess_id": localStorage["sess_id"],
                "sess_tkn": localStorage["sess_tkn"]
            },
            "success": function(response){
                resolve(JSON.parse(response));
            },
            "error": function (x, y, z) {
                toastMessageBottomShort("Could not connect to server");
                resolve(false);
            },
            timeout: 5000
        });
    });
}
```

*Code snippet of how the user session verification would be implemented*

Session information as expected would be stored on the user's device. As such only a single login is required.

## Application Menu




*Application Menu on the Home Screen*

The application menu would list the main functionalities of the application. Details on each function are provided below:

Create Game
This option allows the user to create a new game. The user is prompted to provide the game information including the game name, password and round fee (Figure 5).

Search
This option allows the user to search for specific hosted games with names that match a particular search query.

Refresh
This option refreshes variables through the application such as the list of games that shows the 5 most active and 5 latest created games, the wallet balance and the bucket balance.

Disconnect
This option remotely disconnects all devices excepts the current device from the user's account using the same email address.

Exit

The Exit option disconnects the user's current device from the server. In other words the user's session key is deleted from the server database preventing the user from connecting.

```
const createGame = function(){
    return new Promise(function(resolve ,reject){
        //VALIDATE FORM INPUTS
        $("#createGameFormDialog").toggle(false);
        $("#creatingGameProcessDialog").toggle(true);
        showClouder();
        var gamename = $("#createNewGameNameContainer").val();
        var pwd = $("#createNewGamePasswordContainer").val();
        var fee = $("#createNewGameFeeContainer").val();
        var error = false;
        if(gamename.length < 3){
            toastMessageBottomShort("The game name set is too short");
            error = true;
            $("#createNewGameNameContainer").focus();
        }
        else
        if(checkForSpecialCharacters(gamename)){
            toastMessageBottomShort("Please remove special characters from game name");
            error = true;
            $("#createNewGameNameContainer").focus();
        }
        else
        if(checkIfNumber(fee) == false || fee < 0){
            toastMessageBottomShort("Please enter a valid number");
            error = true;
            $("#createNewGameFeeContainer").focus();
        }
        if(!error){
            $.ajax({
                "url": serverDomain+"creategame",
                "type": "post",
                "data": {
                    "sess_id": localStorage["sess_id"],
                    "sess_tkn": localStorage["sess_tkn"],
                    "gname": $("#createNewGameNameContainer").val(),
                    "pw": $("#createNewGamePasswordContainer").val(),
                    "f": $("#createNewGameFeeContainer").val()
                },
                "success": function(response){
                    response = JSON.parse(response);
                    console.log(response);
```

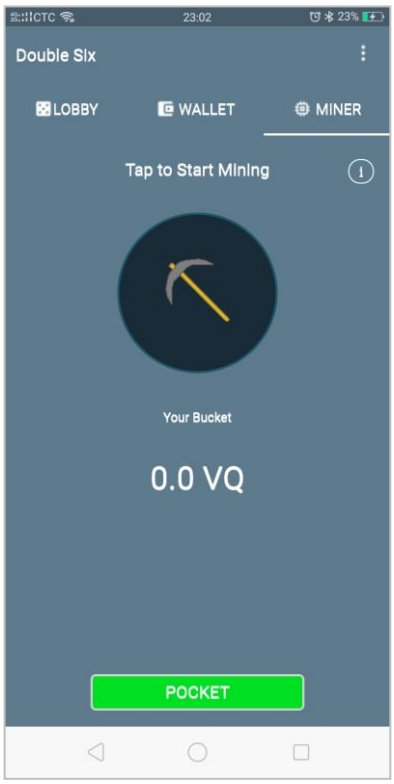
*Code snippet of how the create game function would be implemented*

Wallet Tab


*Wallet Tab on the Home Screen*

The Wallet Tab provides the controls that allow the user to top up and withdraw from their account. Text displaying the User's Balance is also provided. If the user selects the top up button they will be redirected to a PayPal checkout page (Figure 4.4) on which they can purchase their game tokens known as Virtual Quarters (i.e. 1 Virtual Quarter = USD$0.25). If the user select the withdraw button they will be prompted to provide their email address and the amount they wish to withdraw. The server will then send this amount from its PayPal account to the user's account using the email address.

Miner Tab




*Miner Tab on the Home Screen*

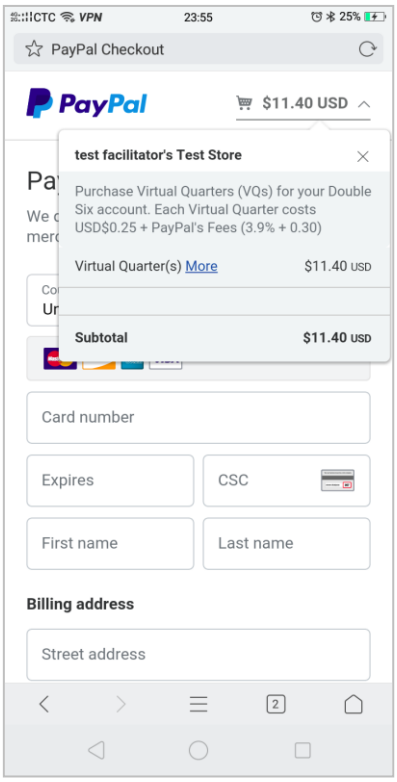
Mining is a reserved feature in the application in the sense that it will not be included in the application developed for the Google Play Store as the Google Terms prohibits applications that include mining. Instead the version of the application to include mining will be hosted on the Double Six official Webpage for user download. The Miner Tab allows users to mine. The App utilizes the devices processing power in exchange for Virtual Quarters. Once Mining has started the App will start a new background process that will enable it to continue the mining process even while the app is minimized.

```
const topup = function(){
    return new Promise(function(resolve, reject){
        $("#toppingUpProcessDialog").toggle(true);
        var amount = $("#topupFieldHolder").val();
        $("#topupFieldHolder").val('');
        if(!isNaN(amount) && amount > 0){
            $.ajax({
                "url": serverDomain+"topup",
                "type": "post",
                "data": {
                    "sess_id": localStorage["sess_id"],
                    "sess_tkn": localStorage["sess_tkn"],
                    "amt": amount
                },
                "success": function(response){
                    try{
                        response = JSON.parse(response);
                        if(typeof(response) == "boolean"){
                            toastMessageBottomShort("An error occurred dugin the topup process. Please try again later");
                        }
                        else{
                            window.location.href = response;
                        }
                    }
                    catch(err){
                        window.location.href = response;
                    }
                },
                "error": function (x, y, z) {
                    hideClouder();
                    toastMessageBottomShort("Please check your internet connectivity");
                    resolve(false);
                }
            });
        }
        else{
            toastMessageBottomShort("Please enter a valid topup amount");
        }
    });
}
```

Code snippet of how the top up function would be implemented

As shown in the code snippet, once the transaction details have been verified by the server the user is sent a PayPal Checkout URL which the app opens. A sample PayPal Checkout webpage is shown below:

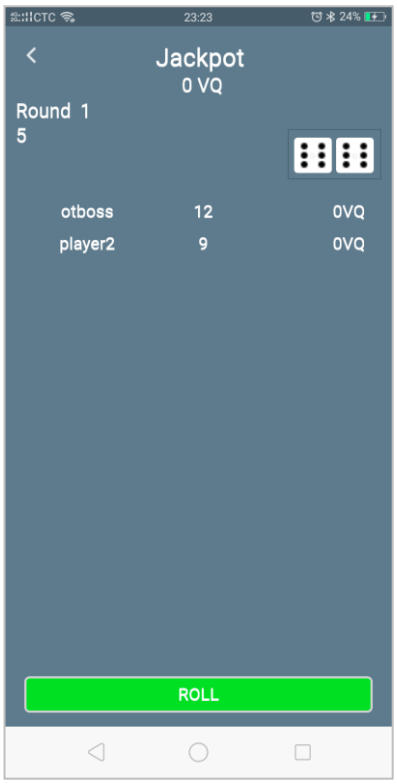
## Checkout Page




*PayPal's Checkout Page Opened in the Devices Web Browser*

Once the user has purchased the Virtual Tokens a confirmation request containing a token will be sent to the Double Six server. The server uses this token to reference the users account's information in addition to the transaction details in the MySQL database. The Server will then allocate the bought Virtual Quarters to the user's account.

## The In-Game Screen




*Game Screen*

If the user creates a new game or joins an existing game they will be redirected to the in-game screen where other users will be able to join.

```
const rolldice = function(){
    $.ajax({
        "url": serverDomain+"roll",
        "type": "post",
        "data": {
            "sess_id": localStorage["sess_id"],
            "sess_tkn": localStorage["sess_tkn"]
        },
        "success": function(response){
            var diceroll = JSON.parse(response);
            try{
                if(typeof(diceroll) != "boolean"){
                    if(diceroll[0] == "0" || diceroll[1] == 0){
                        $($(".dicesRolled")[0]).attr("src", "img/dice0.png");
                        $($(".dicesRolled")[0]).attr("src", "img/dice0.png");
                    }
                    else{
                        $($(".dicesRolled")[0]).attr("src", "img/dice"+String(diceroll[0])+".png");
                        $($(".dicesRolled")[1]).attr("src", "img/dice"+String(diceroll[1])+".png");
                    }
                    $($(".playerScoreContainer")[x]).html(diceroll[0]+diceroll[1]);
                    for(var x = 0; x < $(".dicesRolled").length; x++){
                        if($($(".dicesRolled")[x]).attr("src") == "" || $($(".dicesRolled")[x]).attr("src") == "img/dice0.png"){
                            $($(".dicesRolled")[x]).toggle(false);
                        }
                        else{
                            $($(".dicesRolled")[x]).toggle(true);
                        }
                    }
                }
            }
            catch(err){
                $($(".dicesRolled")[0]).attr("src", "img/dice0.png");
                $($(".dicesRolled")[1]).attr("src", "img/dice0.png");
            }
        },
        "error": function (x, y, z) {
            toastMessageBottomShort("Please enable internet connectivity");
        }
    });
}
```

*Code snippet of how the roll dice function would be implemented*

As shown in the snippet of code the roll dice function would have to be handled on the server. If the function was handled on the user's device a misuser may edit the application's source code and specify what the output of the roll function would be.

## *3.2* Hardware Interfaces

A Mobile Hybrid Application, such as Double Six, runs on a series of Stack Layers. As such all software interfaces of the application has to pass through these layers before the hardware interfaces can be accessed. Below is a graphical representation:


In terms of any specific hardware components of the device used, the Double Six application only uses an inbuilt file storage interface called “localStorage” to store the user’s session information and other configurations.

## 3.3 Software Interfaces

The Double Six Application will make use of several software interfaces to achieve all the required functionalities. They are listed below:

MySQL Database
User information will be stored in a MySQL database running on the Server’s Computer. The server uses a MySQL API to communicate to the database. Once a connection has been established the server uses the interface to run queries and parse and handle the data returned.

PayPal’s Checkout API
The server would require PayPal’s Checkout API in order to generate the checkout URL that will be sent to the user’s device.

PayPal's Adaptive Payments API
The server uses the Adaptive Payments API to complete the withdrawal functionality. When a user requests to withdraw Virtual Tokens for Cash The server will send the cash from its PayPal account to the user's PayPal account and deduct the amount in Virtual Quarters from the users account in the MySQL Database.

### *3.4* Communications Interfaces

The communication standard utilized by the Double Six Application and Server is HTTP implementing the secure socket layer for a secure transaction experience.

Server API
The Double Six application would require an API from the server in order to communicate with the server and necessary HTTPS requests for the server to handle.

Sparkpost SMTP API
In order for the server to send emails. It can either run its own SMTP server or use an SMTP API. The Double Six server would use Sparkpost's SMTP API to send verification emails for users to verify their accounts.

# 4. System Features

## 4.1 Signing In

#### 4.1.1 Description and Priority

This function enables a user to connect a device to their user account. This process is critical because users would be unable to use the entire application if it fails to function.

Priority: 9

#### 4.1.2 Stimulus/Response Sequences

Preconditions:
1. User is not logged in

Process:

| | Stimulus | Response |
|---|---|---|
| 1 | User launches the Application | Application Launches |
| 2 | User enters email address on the sign in screen | Email address field contains email address |
| 3 | User taps next button | Email address sent to the Server, Server sends verification URL to email address |
| 4 | User opens verification URL sent the entered email address | User verified successfully |

4.1.3 Related Functional Requirements

None

## 4.2 Create Game

4.1.1 Description and Priority

This function is responsible from enabling the user to create games

Priority: 7

4.1.2 Stimulus/Response Sequences

Preconditions:
1. User is logged in
2. Game name is available
3. User has enough Virtual Tokens

Process:

| | Stimulus | Response |
|---|---|---|
| 1 | User taps menu icon | Menu opens |
| 2 | User taps create game option | App prompts user for game parameters |
| 3 | User selects OK | Game is created, User redirected to in-game screen |

4.1.3 Related Functional Requirements

REQ-1: Exit Game
REQ-2: Join Game

## 4.3 Join Game

### 4.1.1 Description and Priority

This function is responsible from enabling the user to join games

Priority: 7

### 4.1.2 Stimulus/Response Sequences

Preconditions:
1. A game has been created
2. User has enough Virtual Quarters
3. User knows game password
4. Game is not full

Process:

| | Stimulus | Response |
|---|---|---|
| 1 | User taps game | App prompts user for game password |
| 2 | User provides game password | Server verified game password, App redirects user to in-game screen |

### 4.1.3 Related Functional Requirements

REQ-1: Exit Game
REQ-2: Create Game

## 4.4 Exit Game

### 4.1.1 Description and Priority

This function is responsible from enabling the user to exit games

Priority: 4

### 4.1.2 Stimulus/Response Sequences

Preconditions:
1. User is currently in a game

Process:

| | Stimulus | Response |
|---|---|---|
| 1 | User taps back button | App prompts user for exit confirmation if round currently in progress |
| 2 | User selects OK | Server removes user from game |

4.1.3 Related Functional Requirements

REQ-1: Exit Game
REQ-2: Create Game

## 4.5 Start Mining

4.1.1 Description and Priority

This function is responsible from enabling the user to start the mining process

Priority: 1

4.1.2 Stimulus/Response Sequences

Preconditions:
1. User is currently not mining

Process:

| | Stimulus | Response |
|---|---|---|
| 1 | User taps the mining icon | Mining process started |

4.1.3 Related Functional Requirements

REQ-1: Stop Mining

## 4.6 Stop Mining

4.1.1 Description and Priority

This function is responsible from enabling the user to stop the mining process

Priority: 1

4.1.2 Stimulus/Response Sequences

Preconditions:
1. User is currently mining

Process:

| | Stimulus | Response |
|---|---|---|
| 1 | User taps the mining icon | Mining process stopped |

4.1.3 Related Functional Requirements

REQ-1: Start Mining

## 4.7 Top Up

4.1.1 Description and Priority

This function is responsible from enabling the user to top up their Double Six Wallet

Priority: 8

4.1.2 Stimulus/Response Sequences

Preconditions:
1. User is has cash on their PayPal account

Process:

| | Stimulus | Response |
|---|---|---|
| 1 | User taps the “Top Up” button on the Wallet Tab | App prompts user for amount |
| 2 | User enters amount | Input field displays amount |
| 3 | User Confirms Top Up | User redirected to PayPal Checkout Website |
| 4 | User completes checkout | Server tops up user’s account with Virtual Quarters |

4.1.3 Related Functional Requirements

REQ-1: Withdrawal

## 4.8 Withdrawal

4.1.1 Description and Priority

This function is responsible from enabling the user to withdraw Virtual Quarters from their Double Six Wallet

Priority: 8

4.1.2 Stimulus/Response Sequences

Preconditions:
1. User is has Virtual Quarters in their PayPal wallet

Process:

| | Stimulus | Response |
|---|---|---|
| 1 | User taps the “Withdraw” button on the Wallet Tab | App prompts user for email address and amount |
| 2 | User enters their email address and the amount | Input field display the values |
| 3 | User Confirms Withdrawal | Server sends the cash equivalent to the user’s PayPal and deducts the Virtual Quarters from their account |

4.1.3 Related Functional Requirements

REQ-1: Top Up

## 4.9 Refresh

4.1.1 Description and Priority

This function is responsible from enabling the user to refresh components of the app such as the game list, wallet balance and bucket balance.

Priority: 7

4.1.2 Stimulus/Response Sequences

Preconditions:
1. User is logged in

Process:

| | Stimulus | Response |
|---|---|---|
| 1 | User taps the “Refresh” option in the main menu | App sends request to the server, and loads the response into the interface. |

4.1.3 Related Functional Requirements

None

## 4.10 Disconnect

4.1.1 Description and Priority

This function is responsible from enabling the user to disconnect all other devices from their account.

Priority: 9

4.1.2 Stimulus/Response Sequences

Preconditions:
1. User is logged in

Process:

| | Stimulus | Response |
|---|---|---|
| 1 | User taps the “Disconnect” option in the main menu | App sends request to the server, Server removes all sessions of other devices connected to the user’s account |

4.1.3

Related Functional Requirements

REQ-1: Exit

## 4.11 Exit

4.1.1 Description and Priority

This function is responsible from enabling the user to disconnect all other devices from their account.

Priority: 9

4.1.2 Stimulus/Response Sequences

Preconditions:
1. User is logged in

Process:

| | Stimulus | Response |
|---|---|---|
| 1 | User taps the “Disconnect” option in the main menu | App sends request to the server, Server removes all sessions of other devices connected to the user’s account |

4.1.3

Related Functional Requirements

REQ-1: Disconnect

# 5. Other Nonfunctional Requirements

## 5.1 Performance Requirements

The performance of lower end devices may plummet during the mining process. As such it should be made optional how much of the devices processing power is directed towards mining. For instance, upon tapping the mining icon the user is prompted to select the desired processing power. That way the user can mine without noticeably affecting their device's performance.

To prevent hanging of the app's user interface during asynchronous request, timeouts for each request function should be specified. In some cases the network speed may be very slow and as such the desired action may take some time to complete, instead of being instantaneous. To the user it may seem as if the app itself is slow. After the timeout period a toast message can simply be user to state that the app could not connect to the server and kill the request.

## 5.2 Safety Requirements

In mobile application development, whether native or hybrid, it is important that all vulnerable functions that may affect the user experience for other users are handled on the server side and not within the application as the application can be decompiled and modified. Below are a few components that require strict server handling:

1. User's Wallet – The Virtual Quarters for each user should be stored on a wallet on the server. If this is not the case users can modify the app's source code an place any amount of tokens desired

2. Dice Rolling – It is crucial that the actual rolling of the dice is carried out by the server and not a function on the user's device. If this is not the case then users may edit the app's source code and specify what the roll outcome will be for every turn.

## 5.3 Security Requirements

In order to ensure that all communication between clients and the server is secure, HTTPS is utilized over regular HTTP. This will prevent any communication from being sniffed.

For clarification, since users are verified through their email addresses passwords are not stored on the server. This will also eliminate the hassle of endured by users having to remember their passwords.

## 5.4 Software Quality Attributes

Adaptability
The Double Six App contains some level of adaptability as it is not affected by the user environment, specifically geographic location. Information acquired from the users is also of a fixed format and would affect the system regardless of what is inputted by the user. Strings are converted into UTF-8 before being handled by the server.

Availability
The Services offered by the Double Six server would have a high level of Availability as all requests are handled asynchronously using JavaScript. The server would also utilize the forever module which automatically restarts the server in the event of a crash.

Correctness
The Double Six client and server is to respond correctly to each input provided by the user. Since Virtual Currency is being handled, Correctness is of utmost importance.

Flexibility
The Double Six App possesses some level of flexibility, for instance when the user loses internet connection the internet the app still runs however most if not all functionalities will become inaccessible.

Interoperability
In order for the Double Six application to achieve all the necessary functionalities it needs to interface with many other servers through API's. As such it contains some interoperability.

Maintainability
The Double Six App would have to be maintainable, as new features an updates will have to be included as the app scales.

Portability
As a mobile App that runs on the Android OS, the Double Six app is portable by default.

Robustness
JavaScript by nature has great error handling capabilities. If an asynchronous function throws an exception it will only affect the Promise that encapsulates it, in addition to logging the exception. As such the App will continue to run

Usability
Double Six confirms to certain features of the Material Design which according to Google, is aesthetically pleasing.

### 5.5 Business Rules

As a centralized client server system server security is important for the server logic to provide a concise flow of operations when handling user transactions.

The Client – The client is only required to specify the amount of Virtual Quarters they wish to top up or withdraw from their Double Six wallet. This is then sent to the server as a POST request.

The Double Six Server – The server parses this request and extracts the amount of Virtual Quarters and the operation. It then executes mathematical functions to calculate how much money should be involved and then communicates with PayPal's Server using the PayPal API.

The PayPal Server - receives this request and executes the necessary operations specified in the code used for the API entry point.

## 6. Other Requirements

Pertaining to the MySQL database it is important that it conforms to at least the third normal form (3NF) standard to achieve optimal performance as the application scales.

## Appendix A: Glossary

HTTP – Hyper Text Transfer Protocol

MySQL – A type of relational database.

Mining – The process of verifying transactions contained within blocks in a blockchain in order to earn a reward.

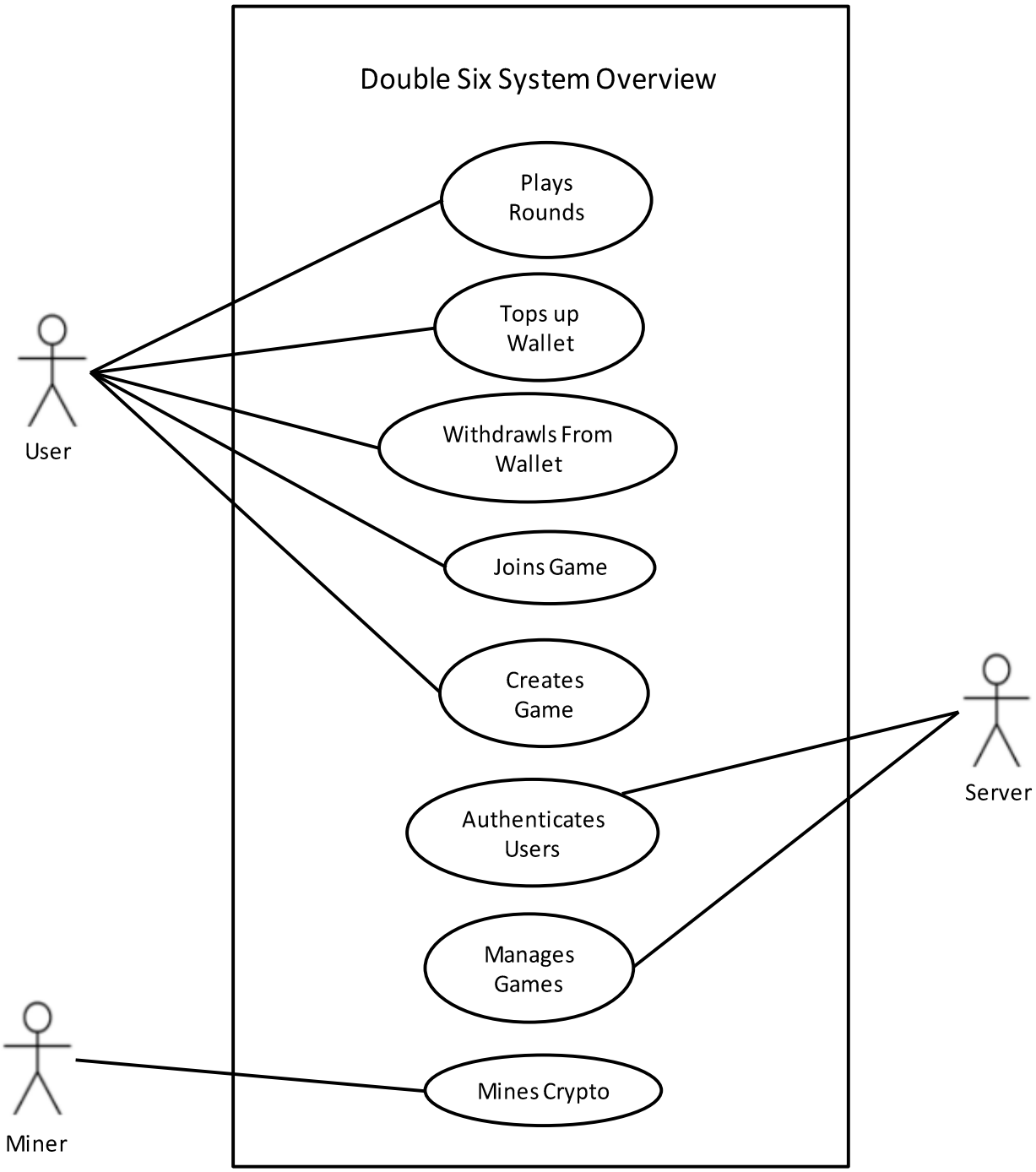
Miner – An actor who engages in Mining

Mining Pool – A mining pool is the pooling of resources by miners in an effort to collaboratively mine. Mining rewards are shared equally among all miners.

Promise – A function in JavaScript used to encapsulate asynchronous code.

SSL – Secure Sockets Layer

# Appendix B: Analysis Models



# Appendix C: To Be Determined List

The following features are to be determined and included in future releases:

- CAPTCHA Challenge after opening the verification URL as extra defense
- Circumvention of PayPal checkout fees
- Sending of Funds between players within the Double Six App
- Instant messaging in rounds

# Percentage Work Completion

Below is a table showing the percentage effort contributed by each member of the group:

| Name | ID | Effort (%) |
|---|---|---|
| Ottor Mills | 180917 | 30 |
| David Thomas | 180912 | 23.33 |
| Nicoy Smith | 180902 | 23.33 |
| Kenneth Anglin | 180907 | 23.33 |