CA326

Alexander Norton
Ryan Byrne

Software Requirements Specification Document

07/12/2018

Table of Contents

Section 1: Introduction

1.1 Purpose

The purpose of this document is to clearly explain all aspects of the project from the functionality to the estimated time schedule for the project. It should also provide a detailed look into the hardware and software requirements along with the technical side of how the project will be developed. The intended audience of this document is the gaming community (Playstation, Xbox, PC, Nintendo Users) and technical minded individuals who are interested in the potential of this project.

1.2 Scope & Problem Statement

The Clanbutton is an Android application which aims to make game matchmaking an easier process. Currently, matchmaking is done in two ways: 1. Join a game with players you already know and 2. Join a game with random players. We would like to make point 1 easier by allowing gamers to find other gamers with the same game.

For example, some games require teamwork, such as tactical games like Rainbow Six Siege, or Counter-Strike: Global Offensive. A clear advantage is given to those who use verbal communication. Therefore players will be able to choose to be matched with only the players who have microphones.

- All app users will create an account and a profile. These profiles will be stored in a database.
- A search bar will be available on the main menu where they will search for any game (Game suggestions will also be stored on the database).
- A chat feature will be available for those who want to message other players they found for that game.
- You will also be able to follow each other to send out a ‘beacon’ for when you want to play a game with any of your followers. This will in turn notify your followers.

The app is essentially a ‘social network’ for gamers.

1.3 Definitions

Steam:

A digital distribution platform on PC used primarily for purchasing & playing games. Also serves as a social hub for gamers to add and play with their friends.

Discord:

Discord is a proprietary freeware VoIP application designed primarily for gamers/ their communities. Users can set up their own server, for free, where they can invite others to join and can then talk in text, voice, and video in channels. Currently available on mobile (App Based) and PC (Browser and Application).

TeamSpeak:

TeamSpeak is a proprietary voice-over-Internet Protocol application for audio communication between users on a chat server. While it is free to join a pre-existing server, there is a charge to set up a server.

Firebase:

A mobile and web application development platform, owned by Google.

Xamarin for Visual Studio:

Enables us to implement native Android user interfaces, owned by Microsoft. Would allow for possible expansion to IOS market in the future due to it's cross-platform implementing features.

Android SDK:

We'll be using the Android SDK which contains the emulators we will use to test the application.

1.4 References

Steam:

Steam: https://store.steampowered.com

SteamAPI: https://steamcommunity.com/dev

Firebase: https://firebase.com/

Xamarin: https://visualstudio.microsoft.com/xamarin/

1.5 Overview:

The Clanbutton aims to make game matchmaking easier. Simply search for a game, hit the button, and you'll be placed into a chat room with those who want to play the same game. A microphone indication will light up letting you know they are ready to use vocal communication during gameplay. You can also tap into their profiles to quickly add them on Steam (using the Steam API) where you can invite them to a game and start playing. Or, you may choose to not use Steam and include the platform of your choice on your profile.
Players may also follow other players they find along their way. Having many followers allows you to send out a 'beacon' which will send a notification to your follower's phones telling them you'd like to play a specific game.

When you're ready to play games, you may want to talk to them through Discord or TeamSpeak. Discord has a nice API which will allow you to show off your Discord server to other users allowing them to join your server by tapping the 'join' option, provided by the Discord API. This makes matchmaking even more simple.

We gathered our requirements for this app by creating a survey to determine the demand for an app such as this. We noted that a number of players found it difficult to find others to play games with - so we decided to go forward with this solution to the problem.
We then tried to find other ways of making the matchmaking process easier by using the Steam/Discord APIs.

In **section 2**, we will cover a general description of the application and the operations that are expected to be carried out.
Section 3 is aimed at development. Essentially, we would like to be able to hand section 3 to a developer and allow them to get to work on this app with a complete overview of what needs to be done.
In **section 4**, we'll cover a high level overview using models such as class models and data-flow diagrams.
And finally, in **section 5**, we'll present a schedule which describe the time estimations for each task represented by a Gantt chart.

Section 2: General Description

2.1 Product

The Clanbutton is essentially a ‘social network’ for gamers. Find new friends to play games with, let others know when you’d like to play a game and set up your own profile. Vocal communication apps such as Discord or TeamSpeak are different as they do not have the essential feature of finding others to play games with. They are at their core a chat application and not a network to connect the players. It would however be great to see the Clanbutton’s matchmaking feature into the Discord app.
Steam is where you can buy and play these games. You can invite others to your game through Steam by going into the Steam menu while playing a game. There is also no feature to find others who want to play the same game.
When you begin playing a game, and start the in-game matchmaking, you’ll be matched with random players. These could be players who speak a different language. Therefore having an app such as the Clanbutton where you can communicate before you start a game is essential for games that require vocal communication. There is a high demand for this.

2.2 Interfaces

Users will have their own profile page where they can upload a profile picture (to Firebase).
When searching for a game, they will be given a list of suggested games as they type (also stored on Firebase) before hitting search.
When users search for a game, they will be added to a chat room for that game, where they can send and receive messages. They can tap into this message and view the player’s profile, or click from the drop-down menu of users who want to play the current searched game.

2.3 Hardware Interfaces

This will be an Android based application, with the ability to scale it to iOS in the future with help from Xamarin for Visual Studio.
Xamarin allows for cross-platform UI development. However, we are limited to development solely for Android as iOS requires development to be done through a mac machine. We will therefore write the XML code for the Android app and later, when possible, port it over quite easily to iOS.

2.4 Software Interfaces

We will be using Firebase, which allows for a realtime no-sql database (something we are looking forward to delving into) and an authentication system which means we won't need to reinvent the wheel when it comes to user authentication.
We can also allows users to authenticate and register using Steam by using the Steam API. This is means any information stored in Steam (such as their Steam profile picture, description, username) will be transferred over to their Clanbutton profile, making it easier to add each other on Steam at the tap of a button.
We can also use the Discord API for the same reason. But instead, players can go to other players' profiles and join their Discord server (to communicate) at the tap of a button.

2.5 Product Functions

1. User authentication/registration
    a. Register/login through Steam to collect your Steam profile data and place it on your Clanbutton profile.
    b. Register/login through Facebook or email (to make future logging in much faster).

2. User profiles
    a. Swipe right at the home page (the search bar) to visit your own user profile.
    b. Collect information from database and place it on their profile based on their user ID.
    c. Include an ‘edit’ option if the user is the current user.
    d. Include a ‘follow’ option if the user is not the current user.
    e. Include a ‘join Discord server’ option if the user has made it available.
    f. Include an ‘add on Steam’ option if the user has made it available.

3. Searching for a game
    a. Suggestions will be loaded from the database as players type.
    b. Suggestions can be tapped and searched for, or the player may choose to search for their own game by tapping the Clanbutton.

4. Display list of users who want to play the same game.
    a. Users can be tapped on to display their profiles.
    b. Users will have a microphone symbol next to their name if they are available to communicate via Discord/TeamSpeak/other.

5. Chat feature.
    a. A box next to the list of players can be clicked to join the chat room.
    b. Messages sent to the chat room.
    c. Messages can be tapped to visit the user’s profile who sent it.

6. Beacons feature.
    a. On the home page, swipe left to visit the beacons feature, displaying all players with a beacon.
    b. You will be able to create a beacon, sending out a notification to all of your followers.

2.6 User Characteristics

The user will be expected to be a gamer who wants to play games while being able to play with other people and communicate with those people via voice communication. As our app will support searching for other players on various gaming platforms (Consoles / PC / Mobile), it is only required simply that the user owns one of any of the many gaming compatible devices in existence.

2.7 Constraints

There will need to be a sufficient number of players searching for the selected game in order to find another player to match up with.

Game suggestions will need to be updated frequently with any newly published multiplayer games. Otherwise they will not be suggested.

Section 3: Requirements

3.1 User Interfaces

When a user opens the app they will be prompted to create a user account, the user can choose to sign up by use of their email, Facebook or Steam account. If the user has made a account previously they can choose to log-in instead of creating a account.

3.2 Hardware Interfaces

The touchscreen of the user's phone.
Android device.

3.3 Communications Interfaces

The Firebase API will handle communication to and from the database. We will have specific functions set up to post and get the required JSON data and will finalize them into objects.

3.4 Classes & Objects

Game:

Methods:

None

Attributes:

GameId: Used to give the Game object a unique ID to be stored in the database.
GameName: Name of the game.

Functional Requirements:

A Game class object will be used during a game search or when a beacon is created. For example, when a player searches for a game, we will want to give the player suggestions as they type. This will be received from the database as Game object and sent to the user.

GameSearch:

Methods:

Delete() : Remove the Gamesearch object from the Firebase database
Insert() : Insert a new gamesearch object into the database
Update() : Update the gamesearch object when a change has been made to it.

Attributes:

Game: Game object.
CurrentSearchers: Users currently searching for this game.

Functional Requirements:

A new GameSearch object will be created when a user starts to search for a game. This will allow the user to find other game searches based on the Game in the SearchGame object.

User:

Methods:

Delete() : Remove the User object from the Firebase database.
Update() : Update the user (when a change is made to any of its attributes).
Insert() : Used for when the account is to be created and sent to the database.

Attributes:

Username : Username chosen by the user to identify themselves
Following : An array of users who this user follows
Description : A description added by a user to its own profile
AccountId : Assigns the user a unique ID
Gamesearches : The game which the user is currently searching for.

Functional Requirements:

The User class will be initiated when the 'Account' logs in or registers. We can use the user class methods to update the current state of the object and send it to the database. For example, we can send to the database the user's current GameSearch when they begin searching for a game, making it possible for other users to see them. Or, we can update who they are following when the user taps 'Follow' by simply calling the Update method after a new user is added to their 'Following' list.

Account:

Methods:

None

Attributes:

AccountId: Assigns a unique account ID to the account.
Email : The account's email address.
Password : User's password.

Functional Requirements:

The account class is created when the user registers. This will then be used to model the User class from the Account until the 'Account' user creates their profile.

Beacon:

Methods:

Delete() : Remove the beacon using the Firebase API.
Insert() : Insert a new beacon object to the Firebase API.

Attributes:

Game : Game object the beacon will be created from.
Expires : Expiry date (10 minutes from creation)
BeaconId: Unique beacon ID number.
User: User who created the beacon.

Functional Requirements:

Creates a new beacon object which will trigger a function, sending a notification to all players who are in the User's followers list.

Chat:

Methods:

Delete(): Remove from database using the Firebase API.
Update(): Update the Firebase database with the new Messages objects.
Insert(): Insert a new chat object using the Firebase API.

Attributes:

ChatId : Assigns the Chat a unique ID.
Messages : A list of messages sent to this chat.
Game : Game associated with this chat.

Functional Requirements:

A chat class containing all messages sent to this specific game chat room. The Delete method can be called when there are no players left in the chat room and it will be updated or created using the respective methods.

ChatMessage:

Methods:

Insert (): Creates ChatMessage in the database using Firebase API.

Attributes:

MessageId : A unique message ID to the chat.
Date : The date/time the message was created.
User: The user who created the message.

Functional Requirements:

A new chat message is created with the Insert method and is added to the Chat object ‘messages’ list. This will come with the user who sent the message and the date it was sent.

3.5 Database Requirements

The database will contain information about what games are currently being searched for, the messages sent to users using the chat feature and the user profiles which will be modelled from the user profile object (which will contain their followers and players they follow, username, etc).

3.6 Availability

Our use of APIs (Discord & Steam) means that if either of Steam or Discord is down either for planned maintenance or otherwise, our users will be unable to access Steam Profiles or add other users to a discord server using the API.

3.7 Security

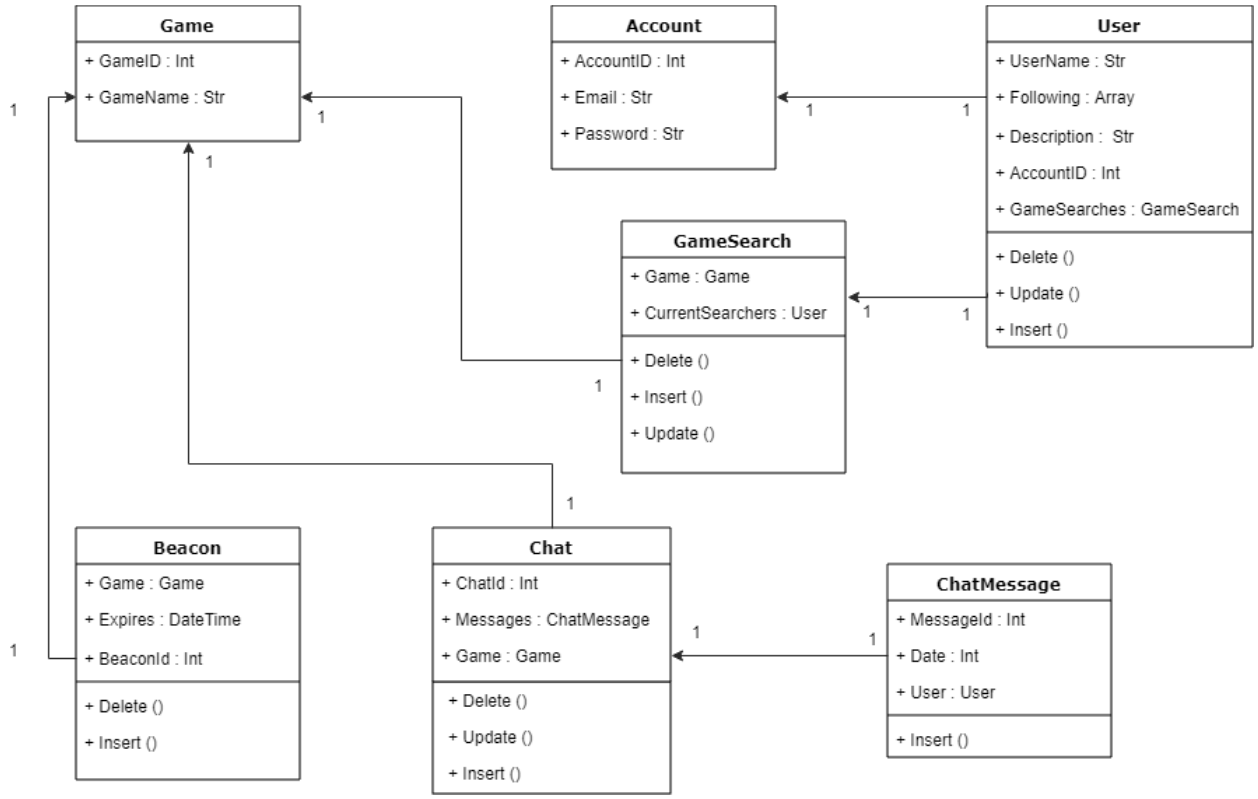
User data will be stored in a Firebase Database. Firebase has a built in authentication system to protect user data.

3.8 Maintainability

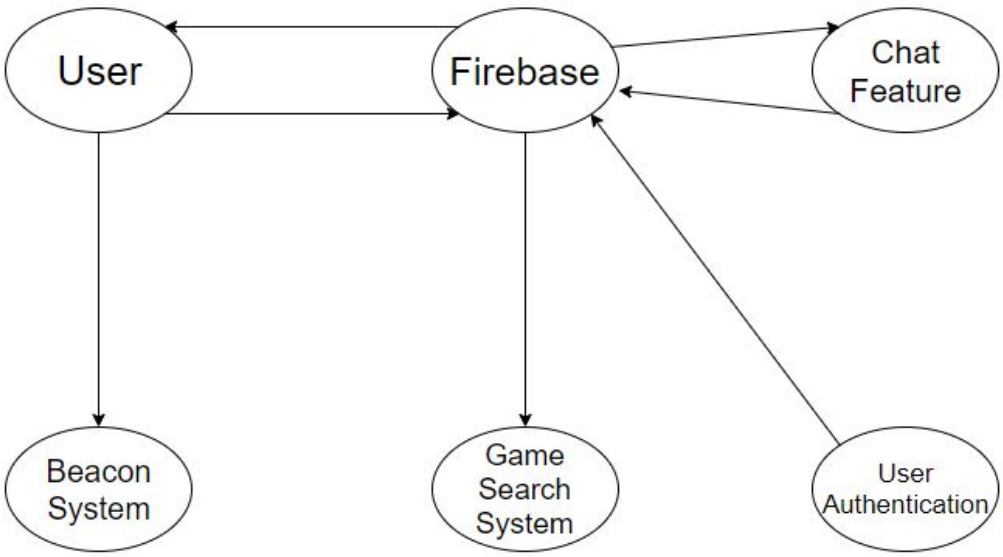
Our game class makes it easy to add a new game to the list of games which are searchable, as many popular team-based games are released daily on platforms such as steam. We need to be able to add games to our database of games efficiently to keep up with the ever-growing catalogue of popular games on all platforms.

Section 4: High-level Design

4.1 Class Diagram



4.2 Data Flow Diagram



Section 5: Preliminary Schedule

5.1 Preliminary Schedule