CA326

Alexander Norton
Ryan Byrne

Software Requirements Specification Document

CLANBUTTON

06/03/2019

Table of Contents

Section 1: Introduction **3**

1.1 Purpose 3

1.2 Scope & Problem Statement 3

1.3 Definitions 4

1.4 References 4

1.5 Overview: 5

Section 2: General Description **6**

2.1 Product 6

2.2 Interfaces 6

2.3 Hardware Interfaces 6

2.4 Software Interfaces 7

2.5 Product Functions 8

2.6 User Characteristics 8

2.7 Constraints 9

Section 3: Requirements **10**

3.1 User Interfaces 10

3.2 Hardware Interfaces 10

3.3 Communications Interfaces 10

3.5 Database Requirements 13

3.6 Availability 13

3.7 Security 13

Section 4: High-level Design **14**

4.1 Class Diagram 14

Section 5: Preliminary Schedule **15**

5.1 Preliminary Schedule 15

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 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 people to find other gamers who want to play the same game.

For example, some games require teamwork, such as tactical games like Rainbow Six Siege, or Counter-Strike: Global Offensive. An advantage is given to those who use verbal communication.

- All app users will register or log in to their profile.
- A search bar will be available on the main menu where they will search for any game (game suggestions will also be provided as they type).
- 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).

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. 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 appear in your follower's 'activity' feed on the main page, 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. 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 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 added 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 edit their public details such as their Discord username or other socials.
When searching for a game, they will be given a list of suggested games as they type (obtained from the Steam API) 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 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.
2. User profiles
    a. Tap the profile picture on the top left of the app to visit your profile page.
    b. Collect information from the 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. Show the user the profile’s socials.
    f. Include a ‘visit Steam profile’ option to visit the user’s Steam profile.

3. Searching for a game
    a. Suggestions will be loaded from the Steam API as the users type.
    b. The suggestions will be based on the games the user owns.
    c. Suggestions can be tapped and searched for, or the player may choose to search for any game they want 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 country flag next to their name.

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 message sender’s profile.
6. Beacons feature.
    a. You will be able to create a beacon, sending out a notification to all of your followers and can be viewed on the front page.

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.

Section 3: Requirements

3.1 User Interfaces

When a user opens the app they will be prompted to create a user account. The user will then sign in using their Steam account. If the user has made a account previously they can choose to log-in instead of creating an 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

GameSearch:

Attributes:

GameName: Name of the game
UserId: Id of user searching
UserName: Username of user searching

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 name

UserAccount:

Methods:

FillSteamData(): Gets the user data from Steam
Update() : Update the user (when a change is made to any of its attributes).
Create() : Used for when the account is to be created and sent to the database.
isFollowingUserAccount(): Checks if a user follows a specified user.

Attributes:

Username : Username taken from Steam
Following : An array of users who this user follows
UserId: Assigns the user a unique ID
Origin: Customizable field for username on Origin
Uplay: Customizable field for username on Uplay
Discord: Customizable field for username/server ID on Discord
SteamId: The steam ID which corresponds to the users Steam Account

Functional Requirements:

The UserAccount class will be initiated when the the user 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.

Beacon:

Methods:

Create(): Insert a new beacon object to the Firebase API.

Attributes:

GameName : Game object the beacon will be created from.
CreationTime: When the beacon was created
UserId: User who created the beacon.

Functional Requirements:

Creates a new beacon object which will trigger a function, sending a activity item to all players’ activity feeds who are in the User’s followers list.

MessageContent:

Attributes:

Message : A unique message to the chat.
Time: The time the message was created.
Email: Clanbutton email bound to the user
Game: Game which the message is for

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.

DatabaseHander:

Methods:

CreateAccount(): Creates an account which has gathered user information from the Steam login
GetAllAccounts(): returns all the accounts present
GetAllChatMessages(): returns all present chat messages
GetAllActivities(): return all present search and beacon activity

Attributes:

FirebaseClient : The Firebase database we are using

Functional Requirements:

Allow interaction with the database for the other classes and also request summaries of data such all all Chat messages etc.

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 UserAccount object (which will contain their followers and players they follow, username, etc).

3.6 Availability

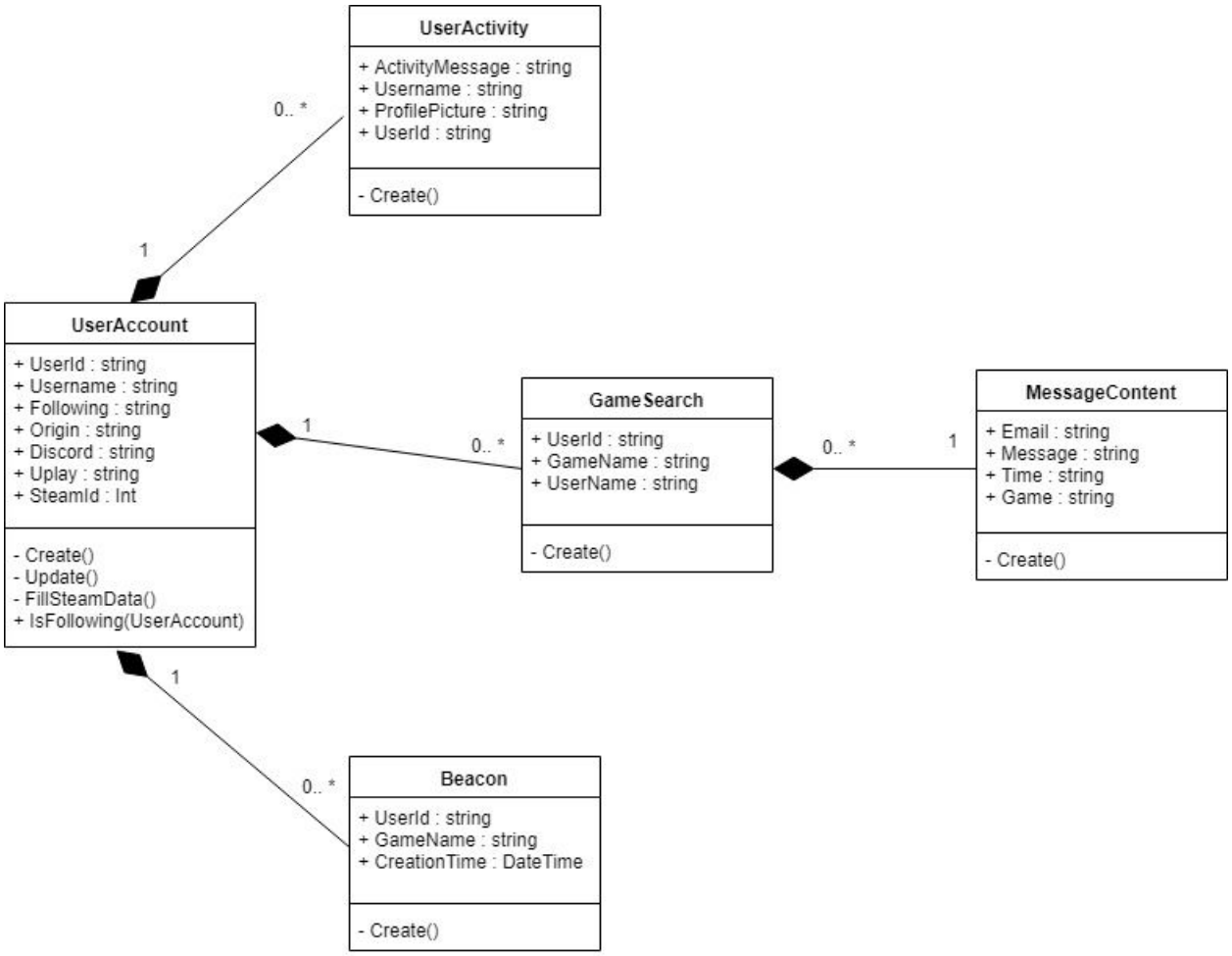
Our use of APIs (Steam) means that if Steam is down either for planned maintenance or otherwise, our users will be unable to access Steam Profiles or register their accounts. They will however still be able to log in to their existing account and use the app as normal (with limited fully functioning features).

3.7 Security

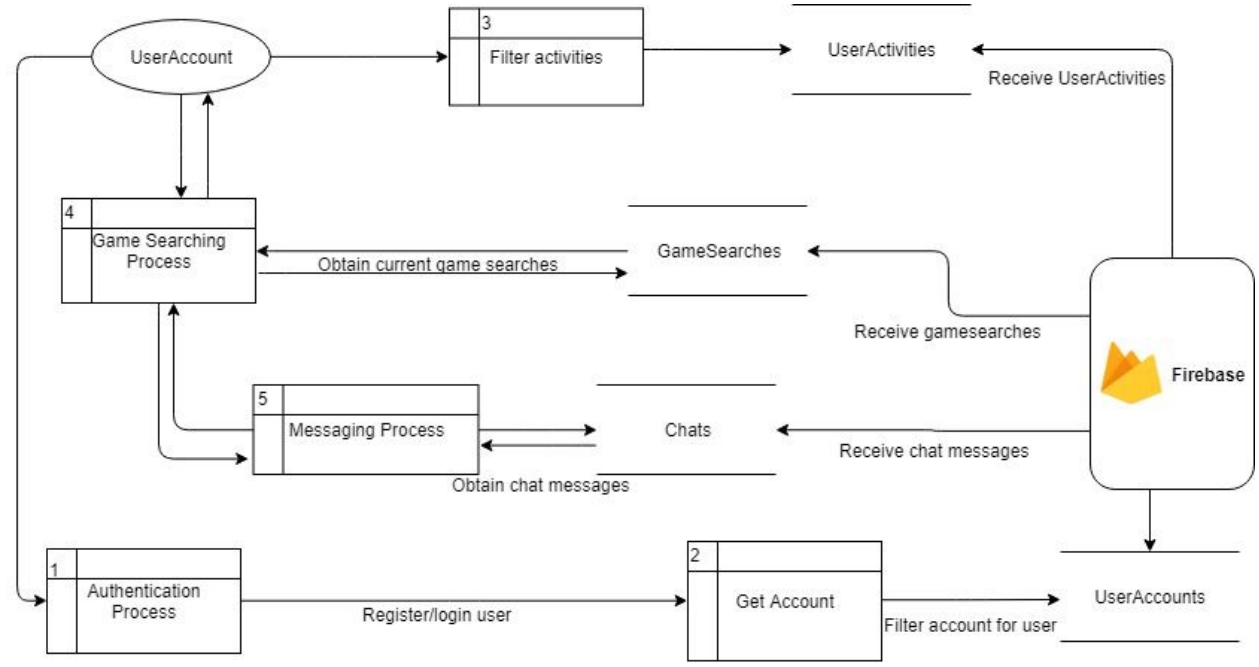
User data will be stored in a Firebase Database. Firebase has a built in authentication system to protect user data.

Section 4: High-level Design

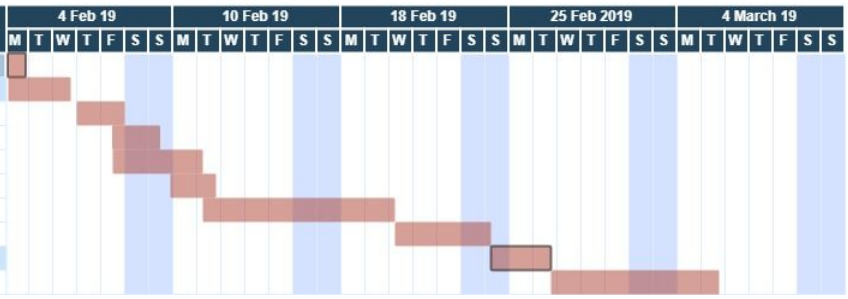
4.1 Class Diagram



4.2 Data Flow Diagram



Section 5: Preliminary Schedule

5.1 Preliminary Schedule

	Task Name	Duration
1	Setting up of Xamarin	1
2	Creation of required classes and functions	3
3	Writing functions for Firebase	2
4	Authentication system	3
5	Steam and Discord API	4
6	User profiles	2
7	Search suggestion system	8
8	Chat room system	4
9	Beacon system	3
10	UI Design + Refactoring	7