RoServer

An Android App

By Joel Smith

04/29/2022

Introduction

Before getting into the details of my application, I would like to give a brief overview of what Roblox & Roblox Studio is. Roblox is a platform where solo game developers & teams of developers can create their games and publish them for the world to play. Roblox is geared toward children, although there are many games on the platform that teenagers and young adults would enjoy playing. Developers create their games on Roblox Studio. Roblox Studio is very similar to Android Studio since it gives you the resources to develop whatever you want! Roblox's primary scripting language is a Lual *(a subset of Lua)*. I have been developing games on Roblox Studio for over three years now, so it felt very natural that I'd try to connect Android Studio to it in one way or another.

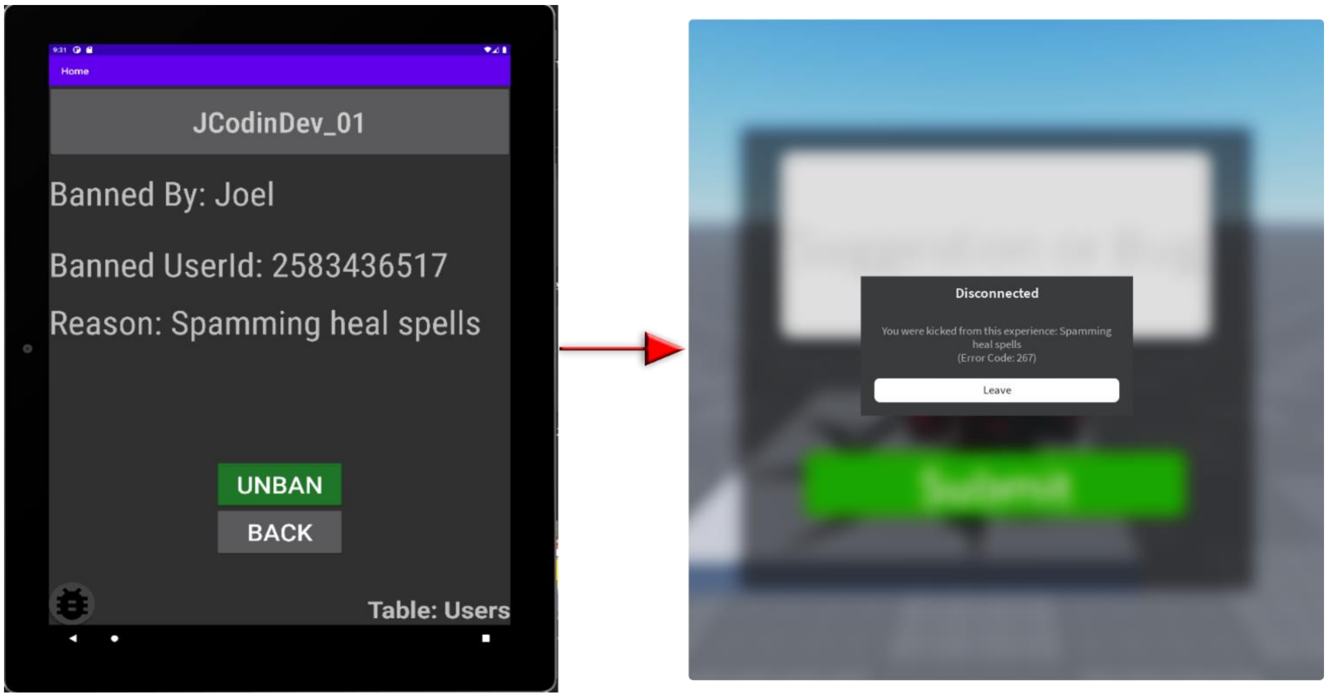
RoServer is a multi-purpose developer panel meant for streamlining user data so Game Developers & Administrators can view feedback quickly and at ease. Typically, logging user data is a time-consuming task that the developers could be using to add to their game.

I considered many thoughts while developing this app. Although, the most crucial feature that would make this process more at ease is, firstly, the user interface. When using an app meant for data access, you don't want to be overwhelmed with crowded UI or the trivialities that many apps face. Thus, I created an app with a very intuitive interface that anyone can understand in a few clicks.

The easiest way to display user information in Android Studio was through using Text Views and the utilization of Android ViewGroups such as Recycler Views, Fragments, and Activities. Recycler Views allowed me to dynamically insert data into a container capable of storing large amounts of views. Fragments were important for switching between frames and user information. Finally, activities allowed me to display the same view on the screen while traversing fragments.

The first significant design element is Database Management. To accomplish the goal of my app, I needed to explore many options for data storage. A handful of the choices were: Firebase, PHPMyAdmin, and MySQL. In my case, I needed a database that would allow me to connect through a web server. The most obvious choice here is MySQL. Making this decision was almost non-sensical since I knew SQL already. All that was left was to learn some PHP, and I was ready to start storing data.

The second significant design element is unique Animations. An easy way to make the User Interface feel clean and unique is to add animations. There are a few animations added throughout my app for effect. The first type of animation is my Splash Screen animation. The Splash Screen animation I made through a website called ShapeShifters. This website allowed me to morph a path from one shape to another. ShapeShifters also made it easy to translate the animation to an XML file. Transitioning also happened to be very important throughout my app as well. For example, one frame may transition from the left or right side of the screen to add a nice effect while complementing UI well.



JCodinDev_01

Banned By: Joel

Banned UserId: 2583436517

Reason: Spamming heal spells

Table: Users

Disconnected

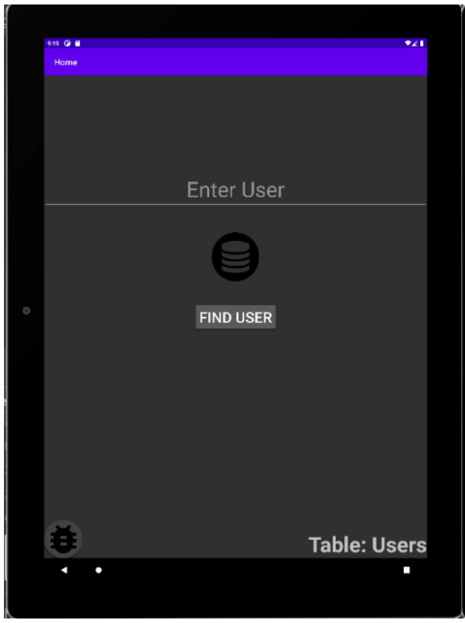
You were kicked from this experience: Spamming heal spells
(Error Code: 267)

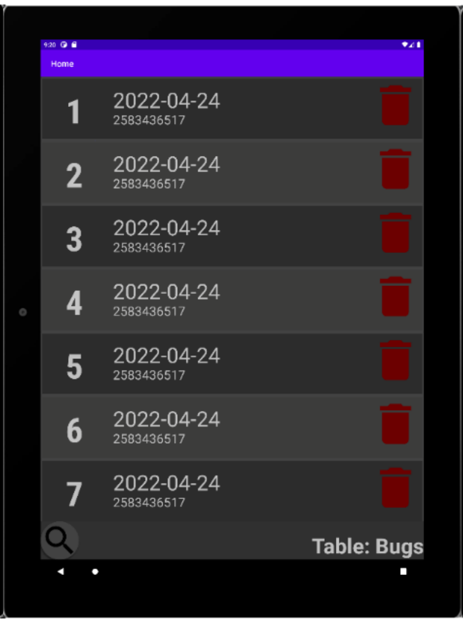
Here is an example of what it may look like if a player is banned on Roblox through Android Studio. On the left is our Android Studio tablet, and on the right is a Roblox game. As you can probably tell, the reason displayed on the right image is “Spamming heal spells” where the player banned is JCodinDev_01. If the player isn't banned, the disconnected notification would disappear. I’ll be going into more detail on this in the implementation of my app.

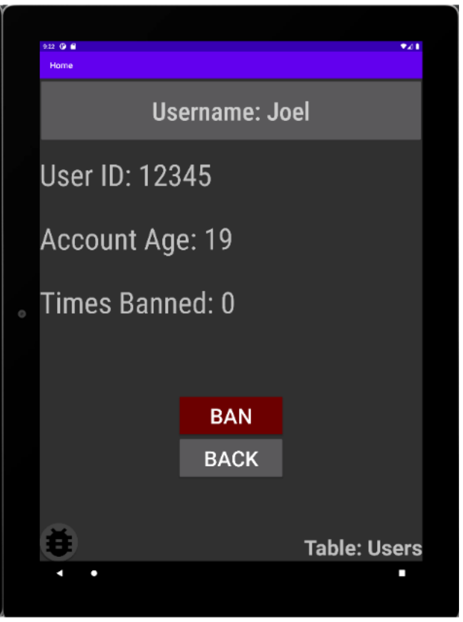
User Experience

Main Fragments

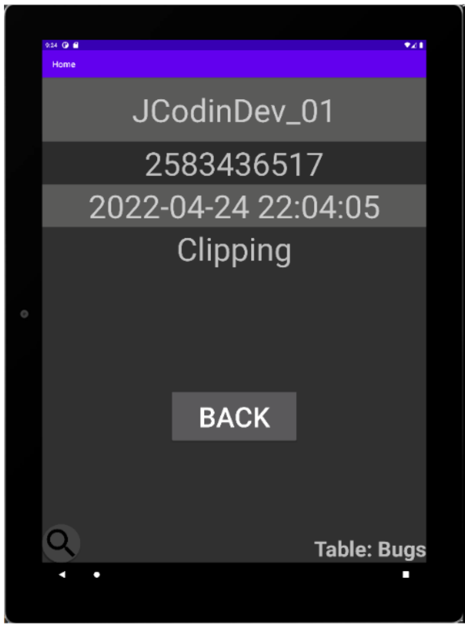
Figures 1, 2, and 3 Respectively

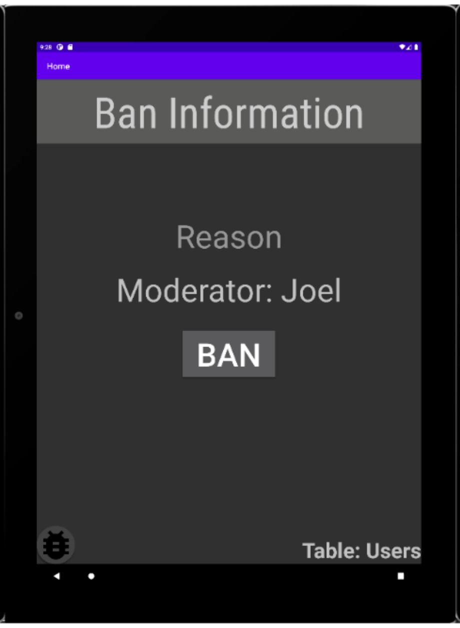


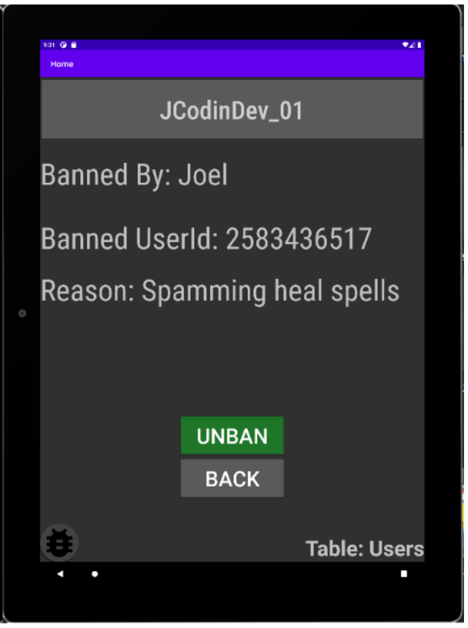




Figures 4, 5, and 6 Respectively

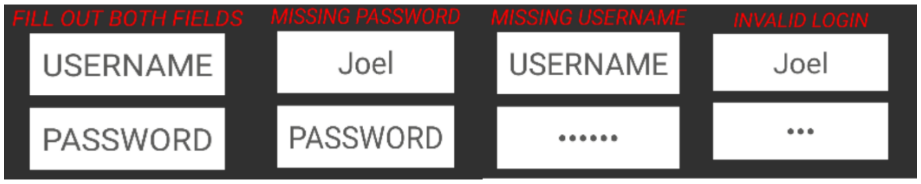






Firstly, during login, the administrator will be required to log in to their account. I have no create account feature in my app for security reasons. When a user attempts to make a login and fails, they will receive an alert notifying that to log in again. There are four alert types:

- Fill out Both Fields
- Missing Password
- Missing Username
- Invalid Login Attempt



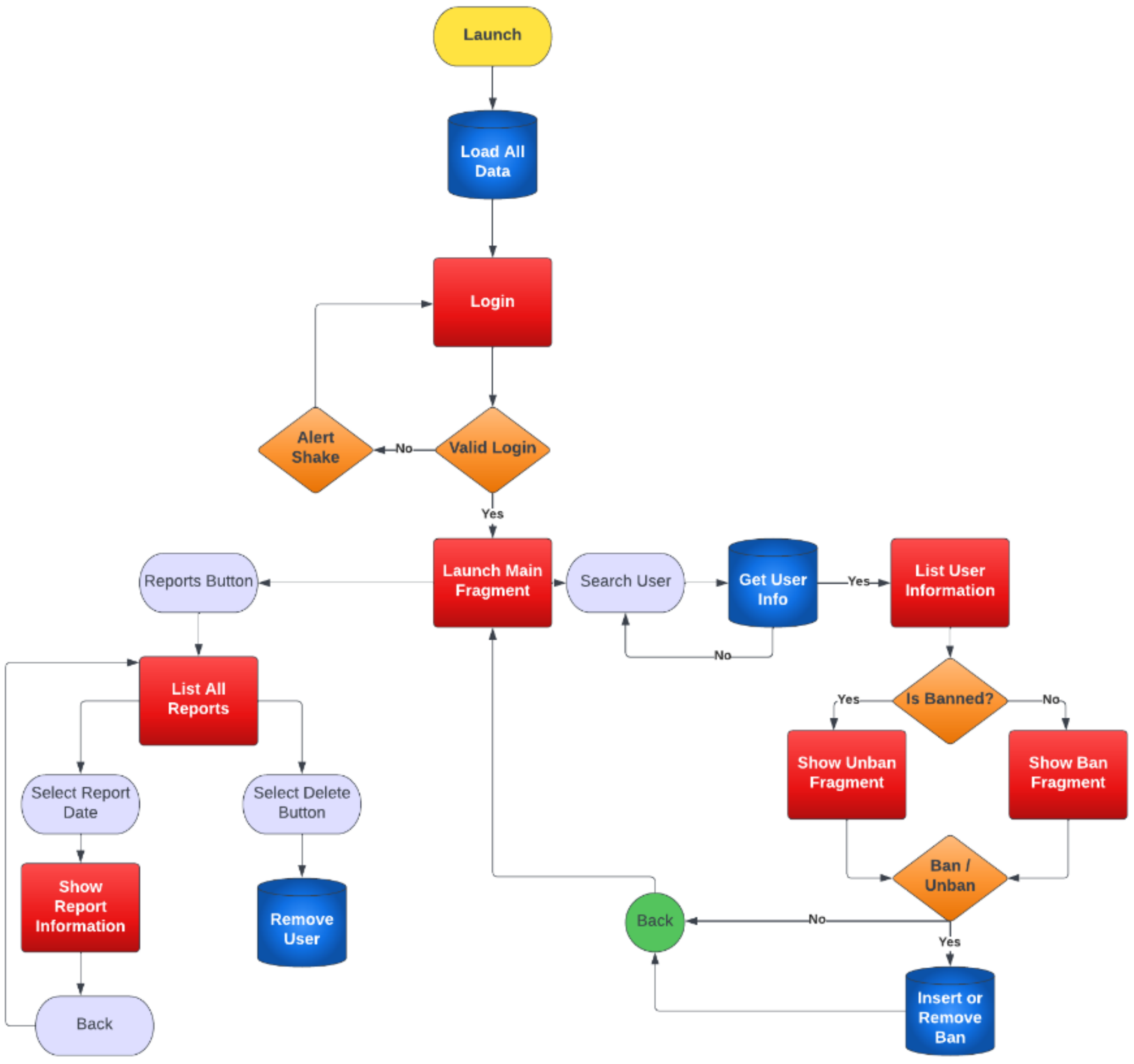
Immediately after login, administrators will have a Find User fragment displayed *(Figure 1)*. The Find User fragment is displayed first since administrators will most likely be looking for particular users rather than wanting to look at reports.

If the admin decides they want to search for a user, they can do it by typing in the player's userId or username. Although they have a choice, this will not alter the search. If the search is successful in the table, a new fragment will be displayed. From the user fragment *(Figure 3)*, you will either have the option to Ban or Unban the user. The choice is dependent on their previous moderated state. Take note that if you choose to ban a player, a reason will need to be provided *(Figure 5)*. If you decide to unban the player: the description, moderator, and user information will be displayed.

Finally, the user will have the option to display all of the reports in descending order from oldest to newest. All of the views are contained within a recycler view, where the user will then be able to select any of the dates they want. After the user selects a date, a new fragment is displayed *(Figure 4)*. This fragment display's all of the report's information. As of right now, the reports can only be displayed from oldest to newest, although I plan on having multiple sorting methods in the future.

Implementation *Android Studio*

Figure 7.



Shown above is a flow chart containing the flow of my app. The primary purpose of this flow chart is to show where Web Server requests occur. The Web Server connections are Blue Cylinders on the diagram. At the top of *Figure 7*, notice it says the launch is where most of the data is loaded. When data is loaded, everything gets stored within a companion object class, to which an Android Studio class has access. Companion objects allow me to dynamically retrieve & manipulate data without constantly needing to make Web Server requests. Some examples where I use companion objects are for retrieving login users and all of the player reports.

Immediately after an admin looks up a user from the Search User fragment, a Web Server request is delivered. Although, why might I make a Web Server request on the search and not just load it at launch? The simple answer is that player data is constantly changing, and the player's information is not always guaranteed to be the same as when you first launched the app. This allows the admin to always view up to date information on players.

Finally, Web Server requests are delivered when you delete a report from the search fragment and Banning / Unbanning a user. As shown in in Figures 3 & 6, the administrator can also ban & unban users. When editing a user's moderation status, the change needs to be input into the database. The Web Server will send the query to the database, and the Web request is complete.

One problem that frequently occurs when making Web Server requests from the main thread are app crashes. When handling Web Server connections, you are connecting through the internet. If not dealt with correctly, your Android Studio app can be left handing and cause a crash. So this doesn't happen, you need to run the connection through a separate thread. Coroutines were my goto in this case, for the server will not crash if a connection is unstable. Couroutines allow my app to run code off of the main thread, so if a server connectivity issue occurs, everything on the main thread isn't affected.

Implementation *Web Server*

Since I was unable use Firebase for a database, I needed to connect to a Web Server. I decided to use Blue Host to host my website. Although this wasn't a free method, I found my topic so interesting that I decided to splurge. BlueHost then provided me with a domain which I could use from Android Studio & Roblox Studio to make web calls. This was the most important step in the development process without a doubt. Without a proper Web Server, I wouldn't have been able to connect to a database, rendering my project impossible.

As for the scripting language I decided to use for the Web Server, I went with PHP. After doing research, I figured that PHP would be the easiest to use and implement. Once I figured out how to use PHP, all that was left was querying the database for necessary information, and echoing it out in JSON format as if it were part of the pages body. This made it very easy to make queries from Android Studio.

Implementation *Roblox Studio*

Roblox Studio I spent the least amount of time on during this project. From Roblox Studio I wrote a few scripts:

- When the player joins, add them to the database
- When the user submits a report, add them to the database.
- The ability to ban a user from Roblox. (Administrator only)

After I created the proper interface, I created a script that used Roblox's HTTPService. Thankfully, Roblox's API made this a simple process since they have a method: PostAsync() which will make an HTTP POST request to the Web Server.

The obstacle, in this case, wasn't using the API but finding the necessary information to retrieve data on the Web Server. Since storing data externally is an uncommon process in Roblox, I could only find 1-to 2 Forum posts that went over this topic. Although they barely scraped the surface with the information I needed, it pushed me in the right direction.

Miscalleneous

After completion, I learned so much in multiple different fields, it makes me very esstatic. The overall organization of this project happened to be a huge obstacle, which thankfully through my previous large scale projects have helped me significantly. I learned a lot about Web Servers and practical applications of making SQL calls to databases. This immediately allowed me to get access to this information remotely from Roblox Studio & Android Studio.

The first few days of me beginning this project I didn't touch any code. I was strictly doing research on if this was actually possible from Roblox. I was forced to change the path I underwent multiple times before I came to a conclusion.

The initial project that inspired this app is a game I'm currently working on with Roblox's platform:

https://www.roblox.com/games/6998582502/Dungeon-Crusaders-TESTING

The game I've been working on since June *(Still in Testing)* and has been a huge project since then.

Our next milestone is 1M+ plays.

References

https://developer.roblox.com/en-us/api-reference/class/HttpService

https://devforum.roblox.com/t/using-an-external-database-to-manage-your-data-and-more-part-1/1001029/3

https://devforum.roblox.com/t/httpservice-and-sql/10650/3

https://developer.android.com/guide/topics/ui/layout/recyclerview

https://developer.android.com/guide/fragments

https://developer.android.com/guide/components/activities/intro-activities