Proposal

Alex Burroughs - JJ Hamaoui - Ahmed Farah - Nathanael Maher

Introduction

For our final project, we decided to make a Minecraft server manager with social networking features, than can be used by both server owner/moderators and players on the server.

The goal of our app is to make it easier to become a part of, and manager, a minecraft community, all on the go. Minecraft server owners will be able to manage there servers without needing to establish an SSH connection (as is usually done) and minecraft players can join and chat on servers they are a part of, allowing them to continue the minecraft experience and not miss a beat, no matter where they are.

Motivation

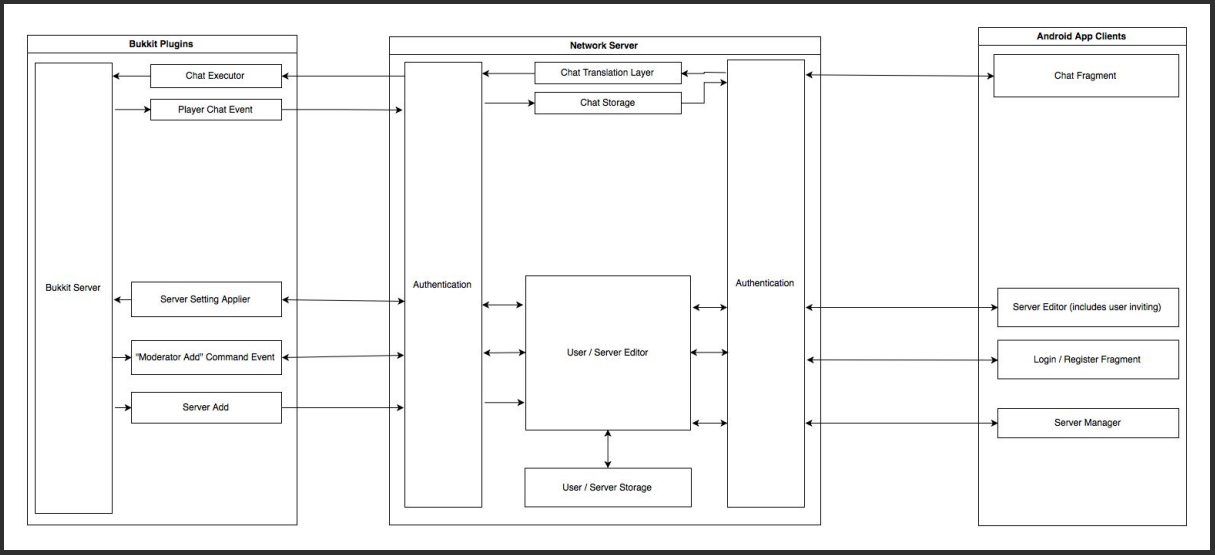
Managing a minecraft server without tools can be tedious and difficult. Accessing the console to run commands, being able to talk with the players on the server, scheduling events, while all relatively easy to do on a computer, can be a challenge when you're not near one. Developing an android application you can have on your phone will solve all of these problems while adding extra features to your minecraft server.

There are some existing apps for managing a minecraft server on android, but those focus more on the technical side of server management. They monitor cpu usage, error messages etc. Our app adds the options for all users, not only administrators to connect and chat with players on the server and others using the app. It also adds the option to connect to multiple servers.

This solution allows the players accessing a particular server to be in close interaction with the administrators of a server, allowing them to take part in pre-scheduled events. To prevent major security risks, the regular members will not be capable of running remote commands from the in-app command line.

Architecture and Environment

Our project consists of 3 components; the Bukkit[1] plugin, the Android app, and the main component, the network server(running Spring.io). The Android app and Bukkit plugin never interact with each other directly, but use the network server as a bridge, allowing us to use the data being sent in between for authentication and implementation of social networking features.



The network server is 1 central server that manages all minecraft servers and android users. The minecraft servers connect to the network server to add itself to the service, add moderators and send player chat messages to be saved. The android app communicates with the network server to send chat messages, edit server settings (if they are a moderator), login/register accounts and join/leave servers that they are a part of.

All security is handled by the network server. The android clients and minecraft servers cannot be trusted as anyone can open up accounts on these services, but the network server is managed by us. Minecraft servers can of course manage which android users can make changes to it.

[1] https://bukkit.org/pages/about-us/

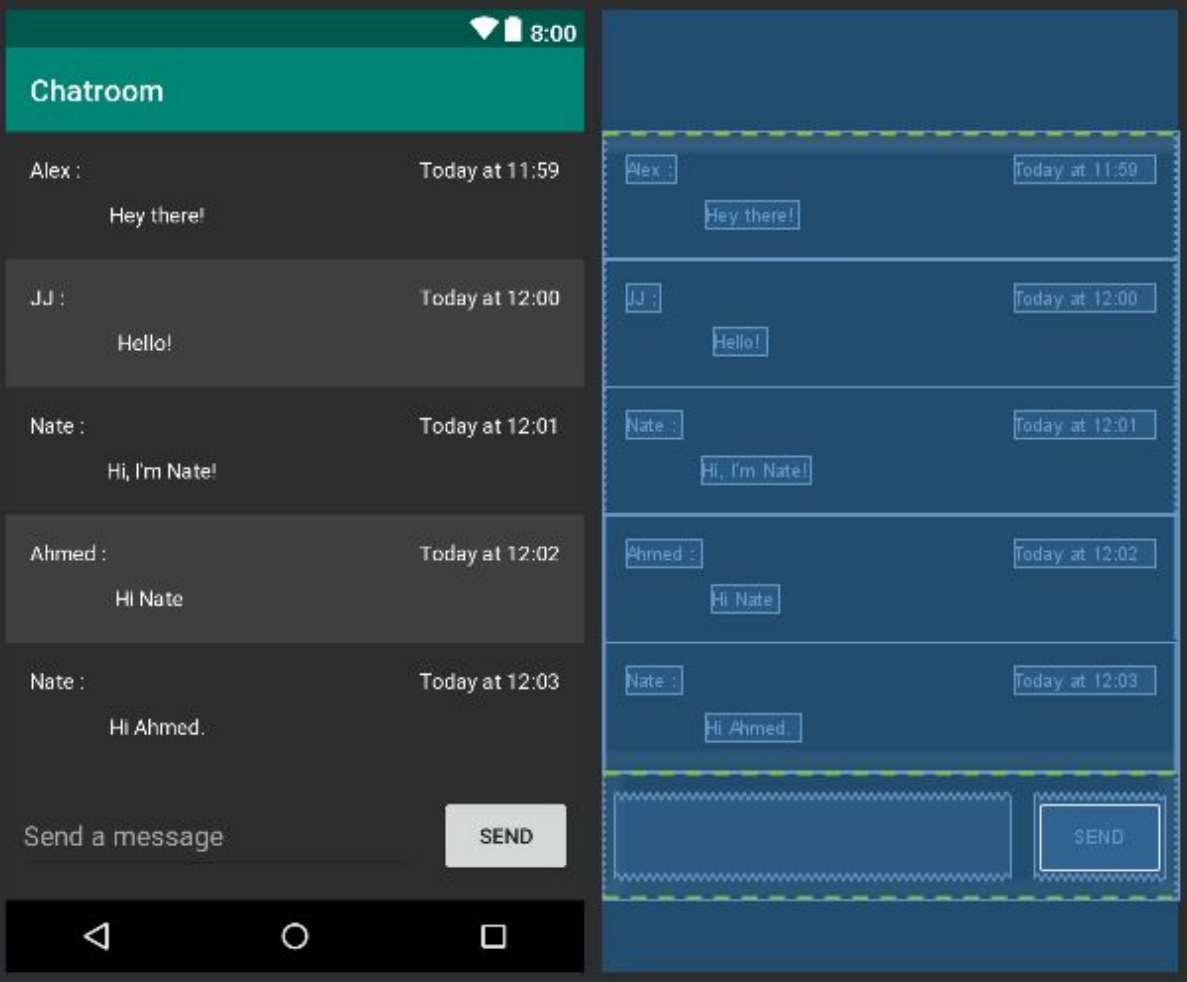
Deliverables

Chatroom Fragment

The goal of the chatroom fragment is to have a chatroom on the app and have it connected to the minecraft server's chat feature. When players type messages on the minecraft server it will be sent to the app, and when the user sends messages on the app it is sent to the minecraft servers text feature. The chatroom fragment is how the user will interact with the chatroom by sending and receiving messages.

The chatroom fragment will interact with the spring IO server by storing and retrieving messages to and from the messages table. It will also use the users table to get the username and see if they are logged in on the app. If not it will use another field in the messages database to get the players name. The app will continuously query the the spring IO server for new messages, and will send messages to the spring IO server when the user presses the share button.

The fragment will have a recycler view that will show all messages for the currently connected server. Each message will have a sender which it will show whether they are connected to the app or they are a player in the game, a timestamp and the actual message text itself. It will have an edit text and a button send messages to the chatroom.

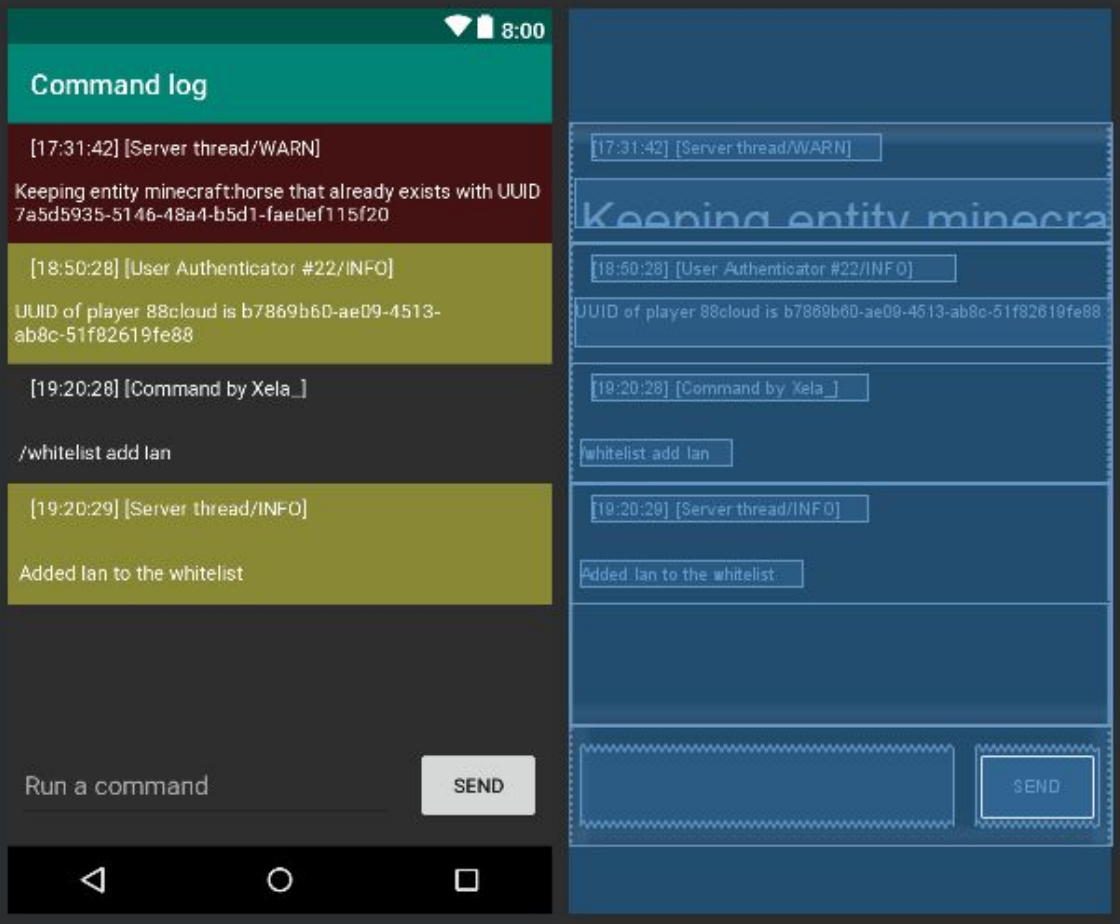


Server Commands Fragment

The server commands fragment will enable moderators of servers to send server commands to the minecraft server they are connected to. It will only be available if the user of the app is a moderator on the currently connected minecraft server. If the app user is not a moderator then this fragment will be disabled so that a regular user won't be able to send moderator commands to the server.

The only communication this fragment has with the spring IO server is to send commands and to receive warning messages. The command is sent and then validated, which checks if the user is a moderator on the minecraft server, and then is sent to the bukkit plugin. Once the bukkit plugin receives the command, it will process the json string and execute the command on the minecraft server. When a warning message is displayed on the server console, the bukkit plugin will send the message to the spring IO server and then to all moderators currently connected to the server.

The fragment will have a edit text for the app user to type in the server commands and a button to send the commands to the server. It will also have a recycler view to show all server warnings and info and all sent commands with their results.



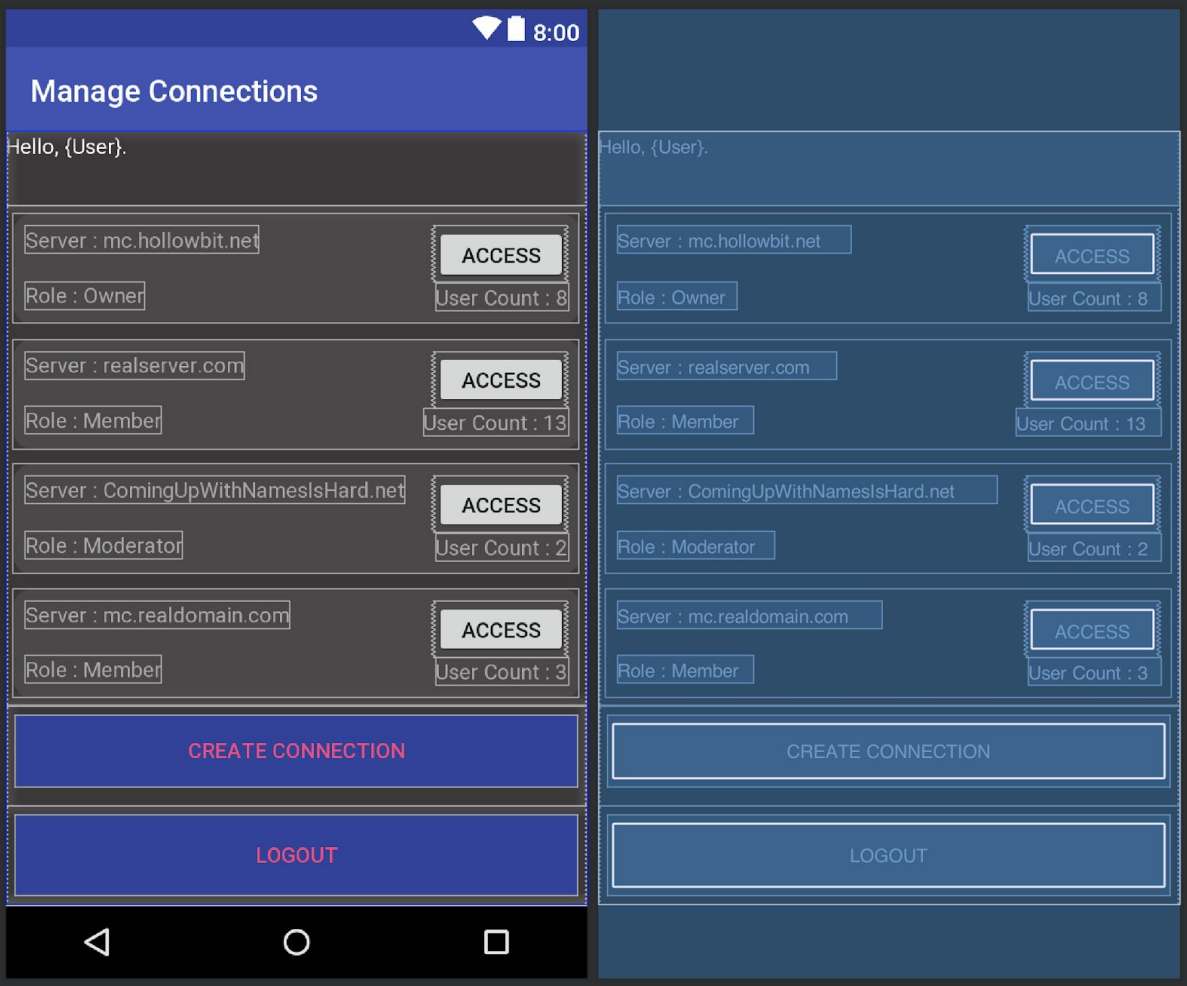
ServerManager Fragment

The ServerManager activity is one of the key activities making up this project. The ServerManager Activity will allow users to create remote connections to their Minecraft Server Client, this will be done assuming the minecraft server is compatible with this application. A minecraft server is considered compatible if it contains a particular bukkit plugin which will allow us to retrieve and send information to and from the server. Upon creating a connection, the database will be updated containing a new connection hooked up with the user who just created the connection. Once a connection is created, a recycler view containing the user's connections will be updated. Through this Recycler View, the user can decide which connection he wishes to access.

This fragment is one of the required base fragments, in that it provides one of the main features of our application. It is where we will be attempting to connect to remote servers and register

these server connections on our databases. Without this fragment, the rest of the application would be meaningless, there is no server management without first being capable of forming a connection to the server itself.

To begin, we will need a base ServerManager fragment completed first, this fragment would include the ability to make a connection to a remote minecraft server. Given we end up with a fully functioning remote connection to the minecraft server, we will then implement storing the connections in both the spring.io database and sqlite database. This will reduce the redundancy of attempting to make connections, and will save successful connections created by the user.



Calendar Fragment

The calendar fragment will allows the users , both MC server admins as well as minecraft players to set events on their calendar relating to the server. These events will be written to the calendar on the users phone but will also send a message to the server about that event

with the options to echo existing events to the server for all the players to see. Events set on the calendar will be read by the fragment so that it can automatically read in already set events and echo them out to the server. The calendar will also interact will the calendar app made by google on android phones to include any minecraft events to their phone's calendar when they set an event using the calendar inside the app.

The first iteration of the calendar would just have the basic functionality of setting event with an in app calendar, witch would be stored locally. Moving that up being able to store server events on a database and echoing out to the MC server when events are set would be the 2nd build. And finally interactions with the calendar on the users phone and automatic messages sent to the players of upcoming events would be the final phase of the fragment. Upon achieving all of this, we would also consider notifying users phones when a event is approaching it's given date, and a notification for when a new event is marked.

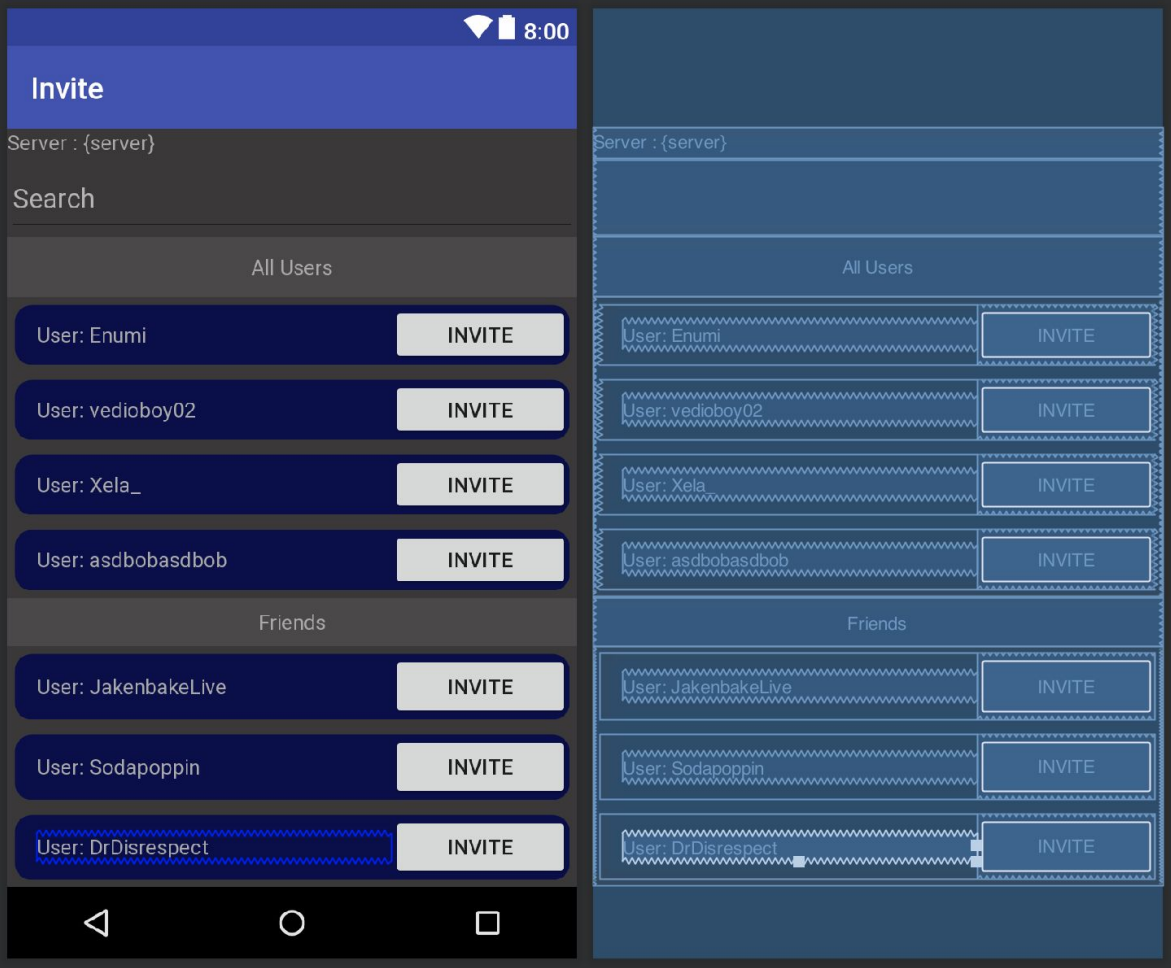
The fragment would consist of a calendar view as the calendar where users could set the events and a button where users could echo out all upcoming events to the server. The calendar view will be used to set the date and the button will be to set and echo the event to all the players on the server.

InviteManager Fragment

The InviteManager fragment is a fragment that will be implemented after a functioning ServerManager fragment is created. The invite manager will allow an owner or moderator of a particular connection invite other users to take part in their connection. This will be conducted by having a RecyclerView with all the users that can be invited with a search bar at the top. The search bar will be used to filter through the users if you wish to invite someone in particular. Upon inviting a user, they will receive a notification telling them they've been invited a connection, it is then up to them to accept this invitation or not. This fragment will act as both an fragment to invite other users to your connection, while also being used to accept invitations which were sent.

The InviteManager will be a simple fragment containing a TextBox and two Recycler views. The TextBox will programmatically filter through the users which can be invited to your connection. The users will be displayed in a recyclerview containing all users. This table will be stored on the spring.io server, and is populated whenever a user registers an account. The second recyclerview will be a list of "friends" the user has added, this can be interpreted as a "favorites" category for users, just as a quick search. To lessen the load on the spring.io server, we can limit the retrieval of all users to only occur when a button is pressed. Note, the "favorite" users are contained on the sqlite database on the users phone.

The InviteManager isn't one of the base fragments, however it allows us to provide the functionality we are intending to give the users. That is to have multiple people connecting to a single connection so that they may use the other fragments to interact with a connection while also being included with connection specific events and the like.



Conclusion

To conclude, our project is an application hosted on a mobile device to create a remote connection to a minecraft server command-line while also providing opportunities to schedule events and interact with currently logged in users. This allows the user to run commands when they were otherwise busy and incapable of manually accessing their minecraft server command line. This app, upon completion will allow the user to schedule events with everyone who has access to the server, to notify a users phone when they have been invited to a server connection they aren't a part of, and it will use websockets in the form of connecting multiple minecraft servers to our main server.

References

The Mojang API: https://wiki.vg/Mojang_API

Bukkit documentation: https://bukkit.gamepedia.com/Main_Page