LottoBot

Abstract

Discord is a free video, voice and text chat app that is used by millions of people around the globe. A great functionality of discord is their concept of servers, channels and bots. Servers are basically chat groups where small communities are formed. Within these servers, you can invite your friends and family to join. Channels exist within each server where you create sub groups. For example, a group leader might create a server for the entire group and channels for each task the group needs to complete. These subgroups/channels could be private to only some people or public to everyone in the server. There are a lot of features to customize your own server. Bots in discord are basically applications that you can add to your server. For example, there are some bots that help manage your discord server and other bots that add entertainment. There are even music bots that just play music in the background of your server.

What we wanted to create was a bot that would allow people of a server or community to join and play in community games. The game that we decided to build was a raffle/lottery type game, where a user would join live raffles or lotteries and win prizes from whom the game was hosted by.

What is LottoBot

LottoBot is a lottery and raffle game type bot that utilizes Discord as its medium due to the multiplatform support of Discord. Also, it's simple and appealing interface, to reach our target audience, which is small communities. Discord's large player base can improve our potential to grow. LottoBot will be able to allow users to create private, individual games and events for small groups or a large community. Our intention is to tie the virtual currency to real world currencies or even utilize blockchain technology and have the value linked to a crypto currency.

Motivation

Our team's motivation in the development of this project was to fill the niche of lottery game Discord bots and challenge ourselves in order to gain experience. As it stands right now and from researching other bots in the public market, there are none that are completely developed and the ones that exist are buggy, not well featured and have very minimal support. This project utilizes different areas of code and allows us to get a further understanding on how applications work and communicate with each other in the real world.

Competitors

RaffleBot

RaffleBot(1) is a bot that only has one main function. It allows users to create a raffle and join it. There's no currency within this bot nor any variety of prizes, each user would join and a winner would be randomly chosen. One feature that caught our eye when researching was the bots ability to give access and permissions to anyone within the server. This feature is called “Deputies” in which “Deputies” have access to control live raffles. “Deputies” have the ability to add and remove people from raffles as well as gain permissions to any commands the bot has. This was interesting as it allows for moderation and a dedicated team to work on live raffles. Unlike RaffleBot(1), LottoBot users would wager x amount of currency to partake in the raffle. Winnings would also be paid out in currency according to raffle conditions. LottoBot also has a cloud database whereas RaffleBot(1) does not.

Lottery Bot

Lottery Bot(2) is a supposed lottery bot that allows users to bet with in-game currency. According to the bot’s description page(Lottery Bot(2), top.gg), this bot is supposed to have an economy system and have lottery games. However, the reviews of the bot show that it’s ‘incomplete and not working. Currently, the only features this bot has working is the ability for a user to wager money for a coin flip of double or nothing. Lottery Bot(2) has a daily rewards function in which users would get daily rewards for checking in on the server. This is an assumption as the Lottery Bot(2) main page has no description on what the rewards function

actually entails. Lottery Bot(2) gave inspiration for having an economic system within the raffles and not just a winner out of a pool of people.

LottoBotto

LottoBotto(3) is a bot that's meant to replicate the casino. LottoBotto(3) currently has two game types, flip and spin. Flip is a gamemode in which users wager money for a coinflip to double or lose their money. Spin is a gamemode in which the user plays a game that replicates a slot machine. Payouts would be paid based on what combination the player lands on the “slot machine.” This bot has nice features and ideas like the slot machine, however, the bot does not have any cloud data. Currency and experience does not transfer over from different servers. LottoBotto(3) also does not host the same games that LottoBot wish to hold; raffles and powerball.

Our Project

How we will stand out

What our bot has that other bots don’t is the ability to play a variety of lottery games with all of its data saved to the cloud. A user's account along with its winnings will be saved so bots in any server can access this data. The bot will have it’s own virtual currency that acts as a user's entry into lottery games. The bot will also grant fully customizable features in an easy to understand gui. For example, a user will enter a command and the bot will prompt them with information about the raffle such as prize pool, who can enter, how long the raffle will last for and etc.

Project Details

So how does this bot work? To run and create a game we would first need to build using gradle to implement dependencies and libraries.

Next go into the code of our project and enter our discord bots token. A bot token is essentially the key to our bot. When a bot is created inside discord servers the only way to connect and talk to that bot is with this key.

Once gradle is done building, Firebase is connected and the bot token is verified, we can now run our program. Currently with what we have, the lottery games are hard coded in, when we run the program a live game will automatically be sent to a certain channel in a certain server and run

based on the parameters given. For example, in Fig 2. you can see our lottery object and the parameters.

When a game is sent to the discord server it is sent in what we call a “panel.” A panel is basically an embedded message within a discord text chat. It allows for customization of what we embed and a whole slew of other functions. In Fig1, you can see the “panel” or embedded message. It contains who the lottery was started by, what the winnings of this lottery is, who is allowed to join this lottery (in this case everyone), time left of the lottery (the timer counts down within that embedded message), what type of lottery it is (there’s two options, power ball or raffle), a cool gif that plays and the unique game id at the bottom. All of this information is tied to a single object/panel. Code in Fig 3.

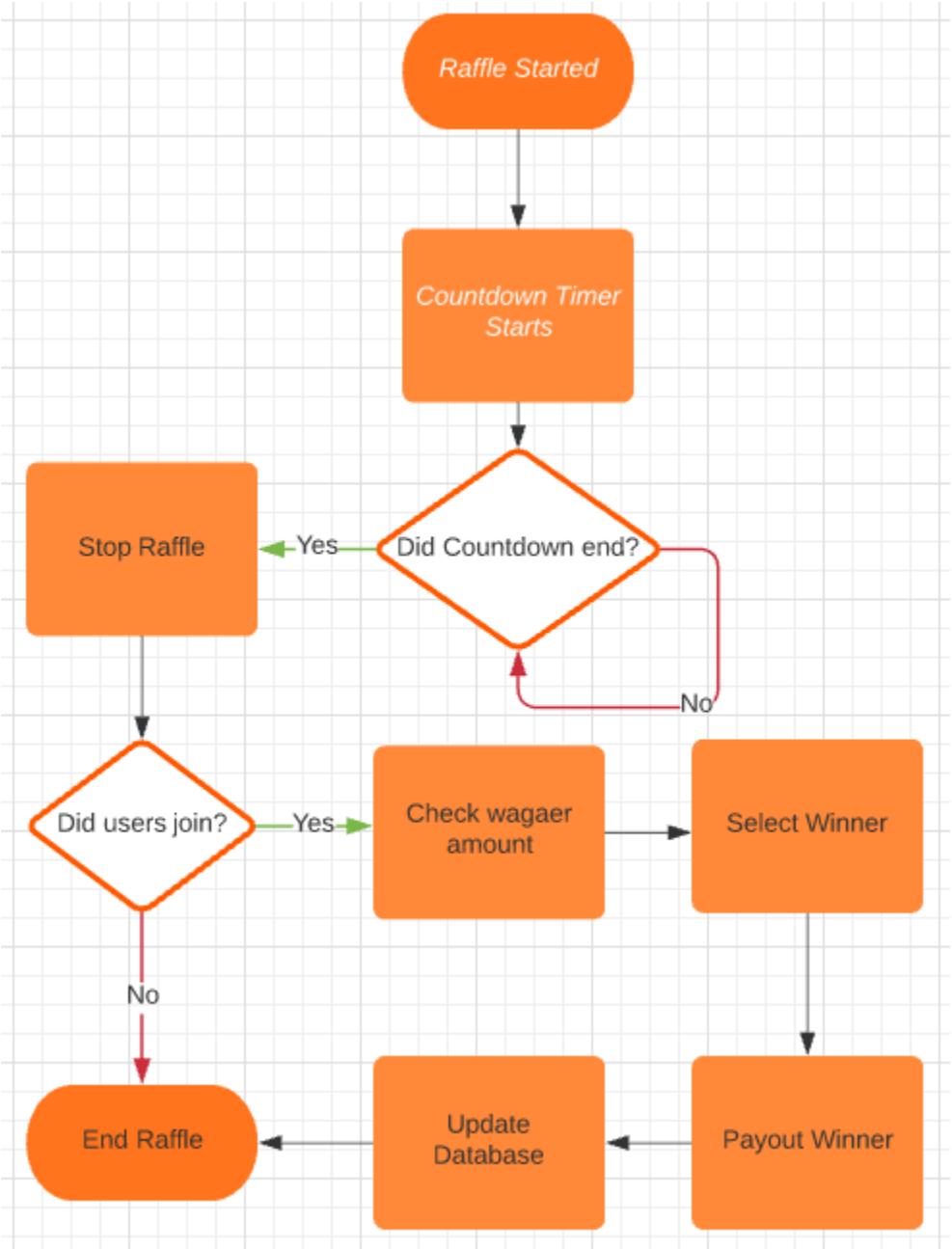
Underneath the panel is a button. Button is currently what we’re working on and trying to implement. An early version of this bot required users to enter text commands to interact with the games. For example, when we would start a game and send it to a server, users would need to reply with “$join 40.” The dollar sign indicates that what the user is about to type is a command and not just text. Join is the function that would actually enter the user into the game and 40 is how much the user would like to wager. We realized that this is not user friendly and could easily get confusing.

What we decided to do was implement buttons that would trigger the same events as text commands. Buttons were only recently added to discords jda so prior to this update, creation and implementation of buttons were extremely difficult and not built in. Luckily, discord pushed an

update that allowed us to easily implement buttons. Code for the implementation of buttons can be found in Fig 4. and the handler in Fig 5. Discords built in buttons have its limitations however, when you create a button object it has to be within an ActionRow object and each ActionRow can only have 5 buttons and each button can only have 5 attributes.

Once a raffle is live and started users can join with the command mentioned above. When they join the raffle their discord ID will be automatically logged and saved to Firebase. This is how raffle data, winners and their earnings or losses will be logged. Discord ID cannot be changed and it is hard coded to every account created on Discords platform. Even if a user were to change their name their Discord ID remains the same. When a raffle ends and a winner is chosen the wager amount will be calculated and the winnings will be saved to the Discord ID on Firebase.

Flowchart


Raffle Started
Countdown Timer Starts
Did Countdown end?
Yes
Stop Raffle
No
Did users join?
Yes
Check wagaer amount
Select Winner
No
End Raffle
Update Database
Payout Winner

Technologies

- Firebase (realtime database)
- Github (collaboration and version control)
- Java (primary language)
- JDA (discord rest API wrapper)

Features

1. Create panels to host games
2. Allow for text commands
3. Allow for button commands
4. Live timer for duration left of game
5. It’s own type of currency for entering the raffles/lotteries
6. Database management (users and their respective currency amount)
7. Gradle
8. Easy implementation into servers and channels
9. Raffle code
10. Lottery seed

Testing

The creation of a Discord bot requires a lot of steps. There's the creation of the bot itself on Discord's servers, accessing the bot with credentials, creating the functionality of the bot and creating a database for the bot to access. Naturally with all these steps each process had to be double checked for its functionality and any problems before moving on to the next. The first test happened with the creation of the Discord bot. Each bot that is created on Discord's platform has a token. This token acts as a key for its creator to access and tweak any functionality they would want with the bot. Initially there was a small hiccup in which connection to the bot couldn't be made but was quickly solved with the issue being syntax errors. Next was to verify the functionality of the bot itself. Could it create games? Would it send the message to the correct channel? Would it listen to players actions? Could you wager successfully? These were all functionalities that needed to work in order for the bot to work properly. Testing in this section was quite easy. First we start up the bot and run a mock game. A test user would enter commands and try to join a live game. Any errors that happened here would be handled in the code and reruns of mock games would continue until the issue was resolved. The next check in the functionality would be the database storage. During each game, someone would monitor Firebase and the values shown to see if it matches up with what's happening with the game. If the values matched up that would mean the bot is functioning correctly with successful connections to both Discord and Firebase servers.

References

Figure 1

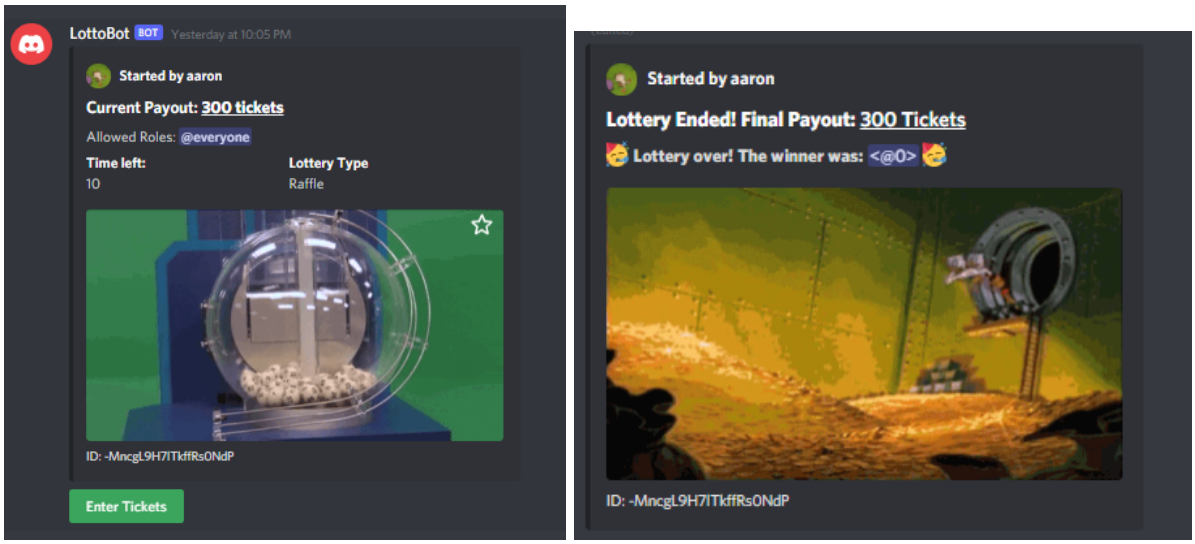


Figure 2

```
RaffleLottery raffleLottery = new RaffleLottery( guildID: 895321750932447263L, botChannelID: 899675807490932817L
        , startedBy: 117648536350359555L, prizePool: 300, time: 20
        , (ArrayList<Long>) null, main); //Create the RaffleLottery object
raffleLottery.print(); //Print the object to the channel
raffleLottery.start(); //Start the timer on the game
```

Figure 3

```
EmbedBuilder embedBuilder = new EmbedBuilder()
        .setTitle("Current Payout: **__" + prizePool + " tickets__**")
        .setAuthor("Started by " + user.getName(), null, user.getAvatarUrl())
        .setFooter("ID: " + uniqueKey)
        .addField("Time left:", timeLeft + "", true)
        .addField("Lottery Type", lotteryType.toString(), true)
        .setImage("https://media2.giphy.com/media/Ps8XflhsT5EVa/giphy.gif");
if (allowedRoles == null) {
    embedBuilder.setDescription("Allowed Roles: @everyone");
} else {
    StringBuilder sb = new StringBuilder();
    for (Long l : allowedRoles) {
        sb.append("<@&").append(l).append(">, ");
    }
    sb.deleteCharAt(sb.length() - 1);
    sb.deleteCharAt(sb.length() - 1);
    embedBuilder.setDescription("Allowed Roles: " + sb);
}
//Above code creates the Embed message see
```

Figure 4

```
MessageBuilder mb = new MessageBuilder();
mb.setEmbeds(embedBuilder.build());
mb.setActionRows(ActionRow.of(
        Button.of(ButtonStyle.SUCCESS, "enterTickets", "Enter Tickets")
));
```

Figure 5

```
public void onButtonClick(@NotNull ButtonClickEvent event) {
    if (event.getComponentId().equals("enterTickets")) {

        event.reply("test").setEphemeral(true).queue();

    }
    super.onButtonClick(event);
}
```

Citations

- Jack#5904. “Rafflebot.” Discord Bots | Top.gg, https://top.gg/bot/722816933996789812.
- GabHas#8277. “Lottery Bot.” Discord Bots | Top.gg, https://top.gg/bot/715282360815452202.
- YGOLG#8005. “Lottobotto.” Discord Bots | Top.gg, https://top.gg/bot/425441496158830593.
- Austin Keener. “JDA.” Github | github.com, https://github.com/DV8FromTheWorld/JDA
- “JDA 4.3.0_340 API.” *Overview (JDA 4.3.0_340 API)*, https://ci.dv8tion.net/job/JDA/javadoc/index.html.
- “Community Home.” *Gradle Docs*, https://docs.gradle.org/current/samples/sample_building_java_applications.html.
- “Bots: Buttons - Discord.” *Bots: Buttons*, https://support.discord.com/hc/en-us/articles/1500012250861-Bots-Buttons.
- “Class EmbedBuilder.” *Embedbuilder (JDA 4.4.0_351 API)*, https://ci.dv8tion.net/job/JDA/javadoc/net/dv8tion/jda/api/EmbedBuilder.html.
- “Class MessageEmbed.” *MessageEmbed (JDA 4.4.0_351 API)*, https://ci.dv8tion.net/job/JDA/javadoc/net/dv8tion/jda/api/entities/MessageEmbed.html.