README

Specification for running/building
Java 8 or greater

Libraries
MySQL Connector Java
DV8FromTheWorld JDA

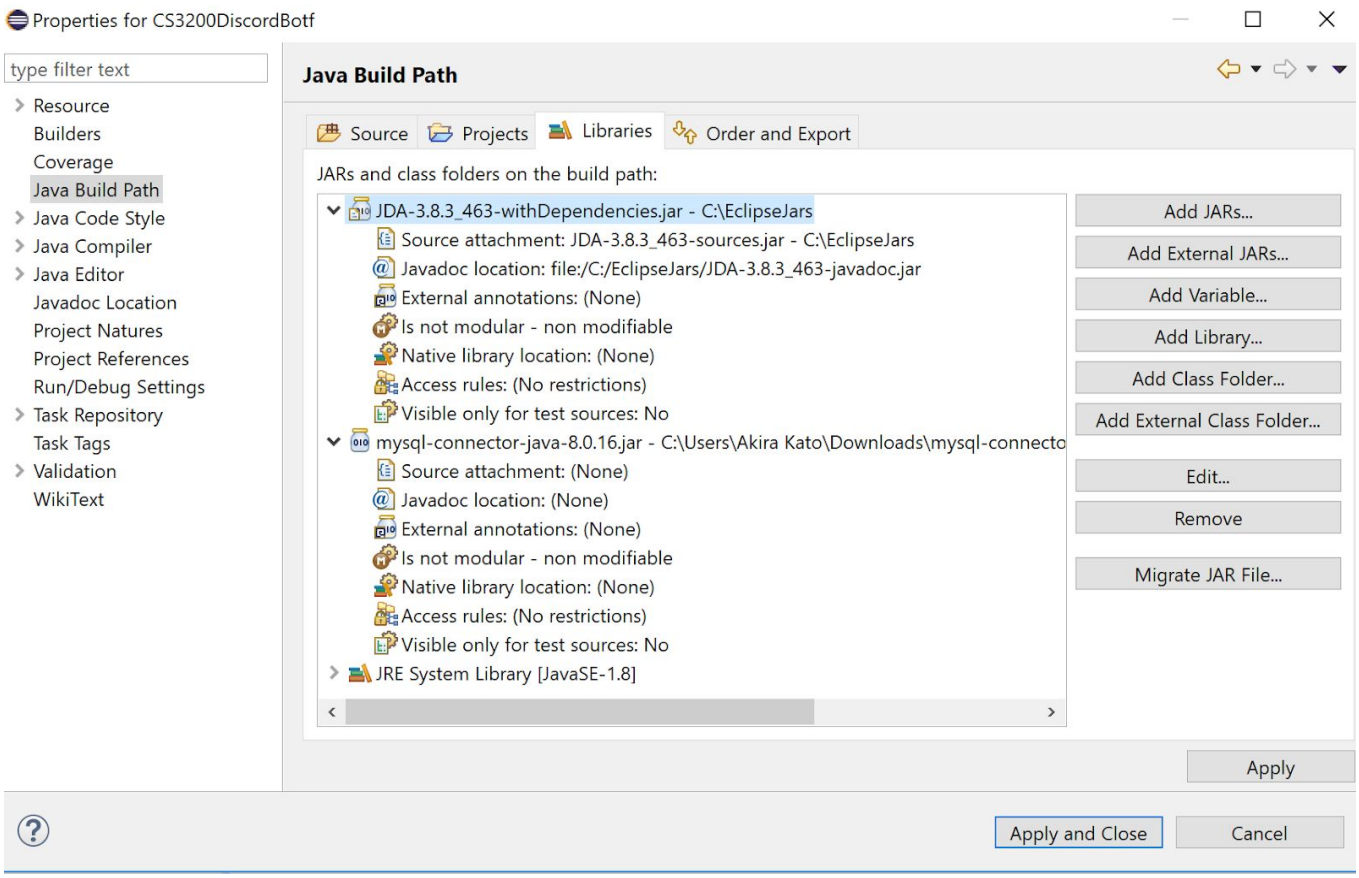
Installation Guide
Java Frontend
Go to https://github.com/DV8FromTheWorld/JDA/releases/tag/v3.8.3
Scroll down to download:
JDA-3.8.3_463-javadoc.jar
JDA-3.8.3_463-sources.jar
JDA-3.8.3_463-withDependencies.jar

Go to http://dev.mysql.com/downloads/connector/j/
Download the connector for your operating system

Either
Go to https://git-scm.com/downloads
Download and install git for your OS
Create a new java project
Open terminal and change directory to that project
Type:
Git init
Git remote add origin https://github.com/Akirakato1/CS3200DiscordBot.git
Git pull origin master
Or
Unzip and import the front end source code files into a new java project through IDE

Add external libraries
JDA-3.8.3_463-withDependencies.jar
My-sql-connector-java.jar

Add the javadoc and sources to JDAwithDependencies



Database installation
Run the sql file in backend to create database locally with MySQL workbench or online
Go to Main.java
Change the fields of
Database db=new
Database("sql9295993","NnAbW18udW","sql9.freemysqlhosting.net",3306,"sql9295993");
According to your database details:
Database(String username,String password,String server, int port, String database_name)

Just simply run Main.java and the bot is now online.

User interface installation
Create discord account at https://discordapp.com/
Either download/install the app through same link or access web version
https://discordapp.com/channels/@me

Create a server. Refer to the following link for assistance
https://support.discordapp.com/hc/en-us/articles/204849977-How-do-I-create-a-server-

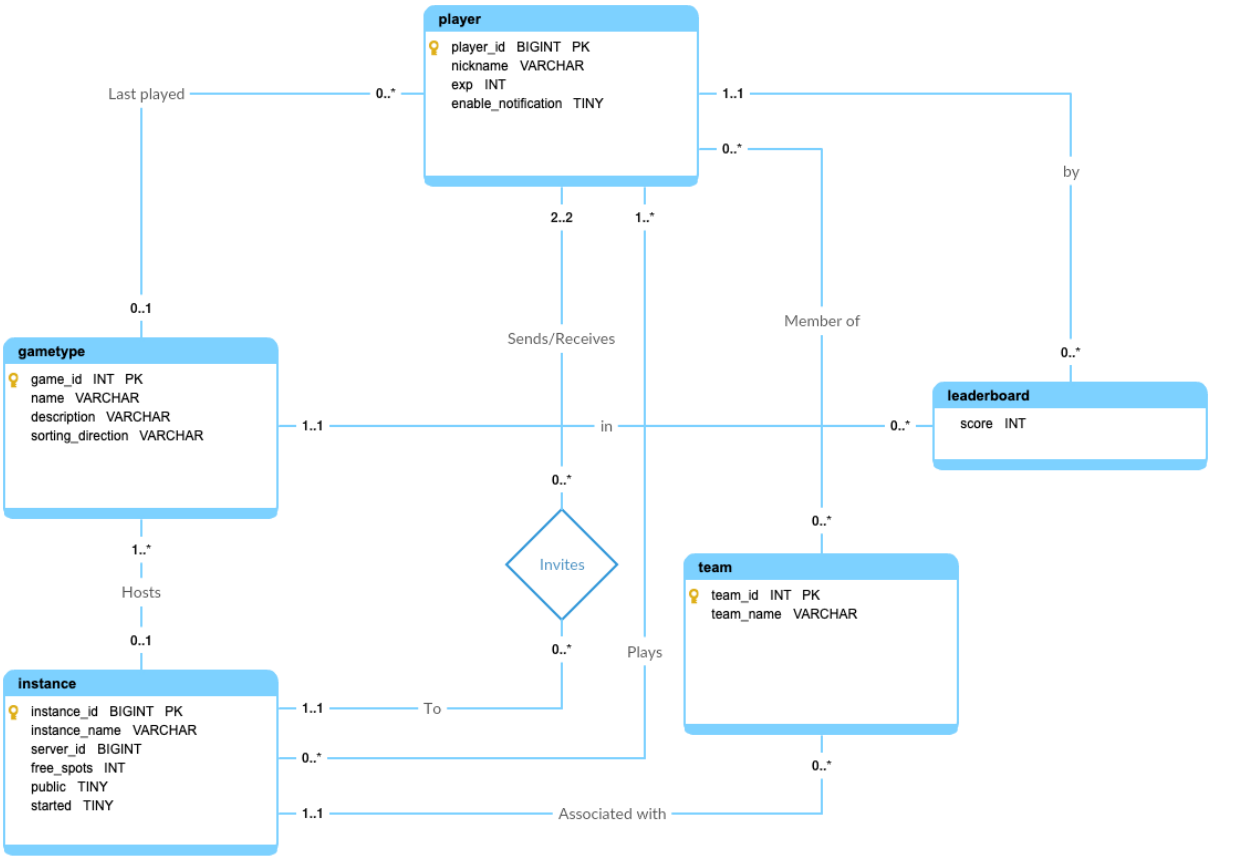
Go to
https://discordapp.com/api/oauth2/authorize?client_id=588425785920978969&permissions=8&scope=bot
select your server from the drop down window and invite the bot
If servers do not appear, make sure you are logged in on web version of discord

Upon successfully entering your server, the bot will prompt you to
!initbot

Technical Specs

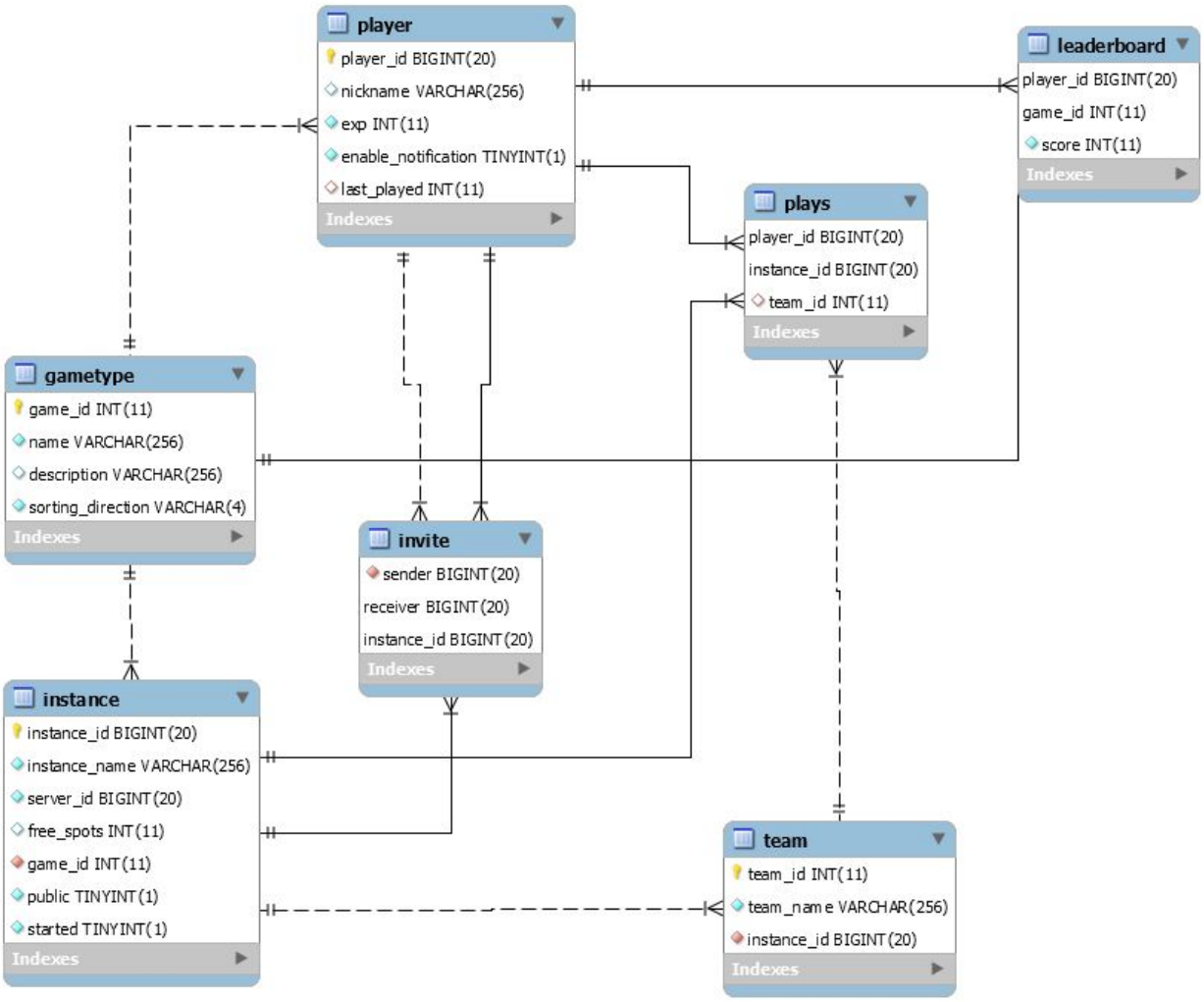
Database type: SQL storage
Software / Languages, Libraries, Hardware:
Languages: Java (ver 8) +MySQL
Software interface: Discord chat
Libraries:
DV8 FromTheWorld/JDA (Discord Java API)
Mysql connector java (jbdc)
Hardware requirements: Any computer able to handle Java, server load.

UML

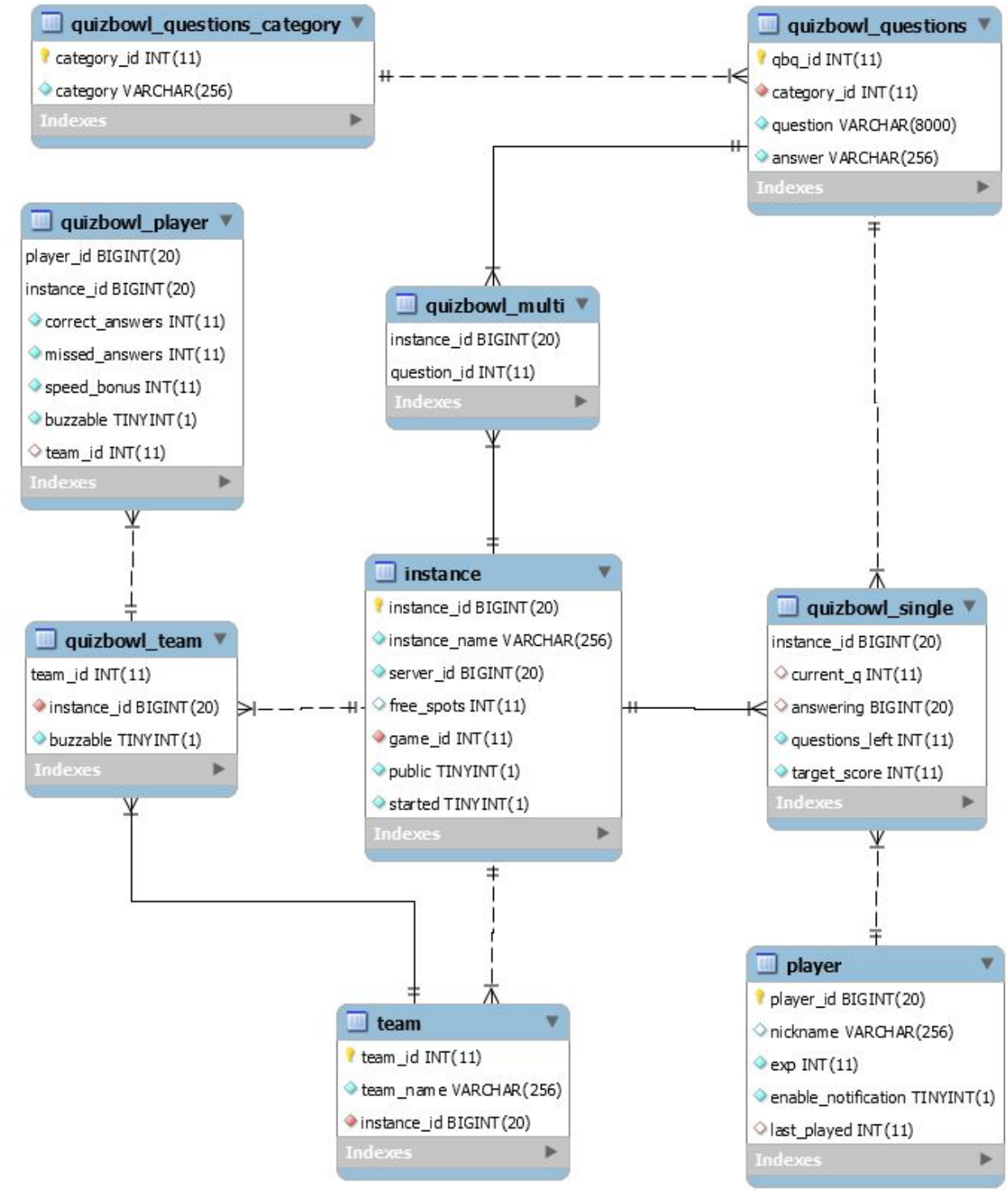


EER

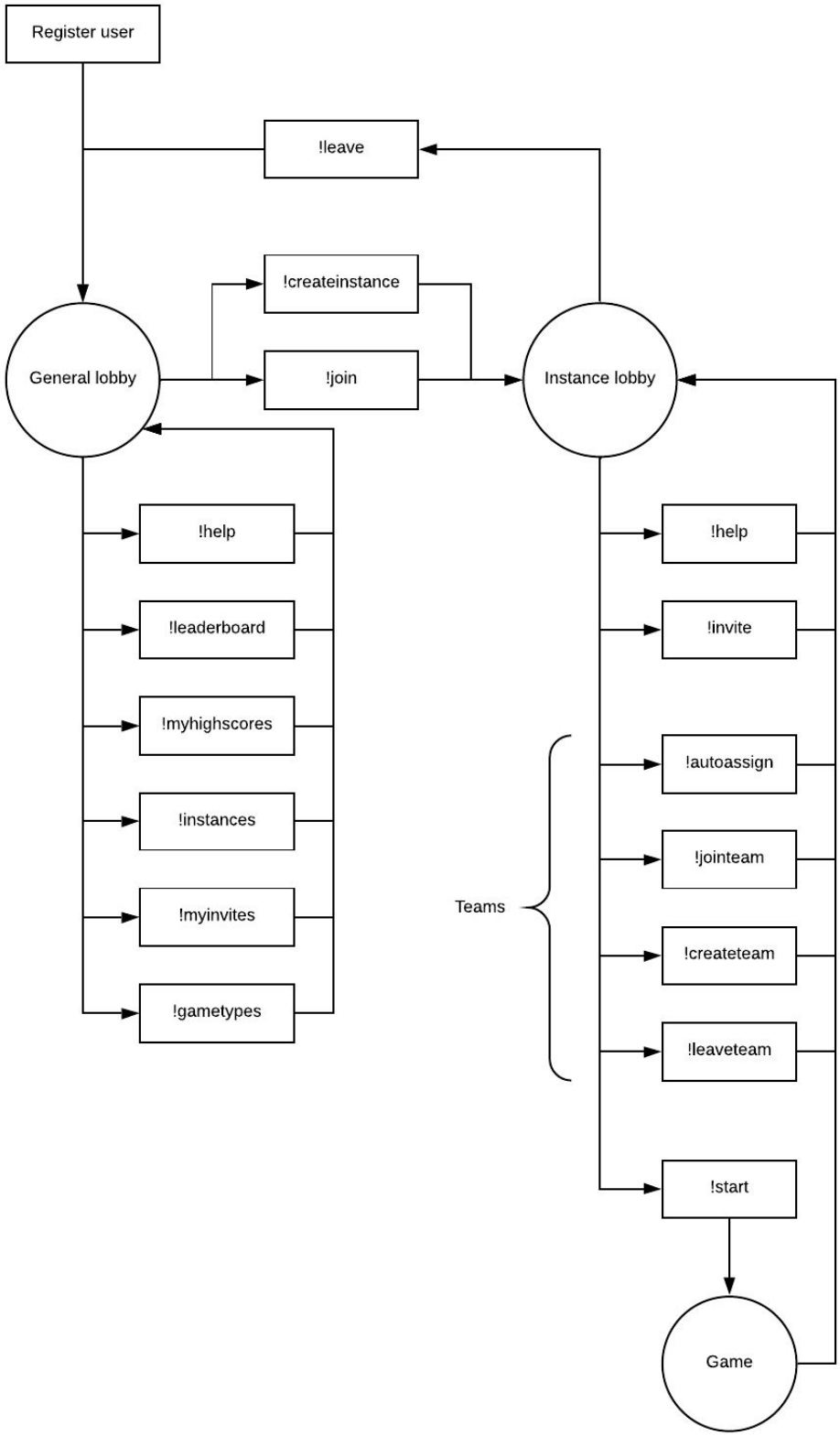
General bot EER



EER with Quiz Bowl tables



User flow


Register user
!leave
!createinstance
General lobby
!join
Instance lobby
!help
!leaderboard
!myhighscores
!instances
!myinvites
!gametypes
!help
!invite
Teams
!autoassign
!jointeam
!createteam
!leaveteam
!start
Game

Commands
For commands with no arguments type !command
For commands with arguments type !command: arg1, arg2, arg3
Game functionality actions do not require “!” prefix
Corresponding SQL operation is listed (C, R, U, or D)
Use edit Distance algorithm to suggest a command closest to the user input given the user input is invalid.

General Commands:

help context(optional)
Provides the user with information on available commands in general or detailed information on the command specified in the context.

createinstance [public | private], gametype, capacity, instance_name
Creates an instance/channel of the provided gametype, privacy, capacity, and name. If the name is not used.

Creates an instance tuple.

join instance_name
Adds user to the instance if all conditions are met.

Check if instance is valid by reading from database. Upon joining creates a Plays tuple for the player and updates the number of free_spots by subtracting one. Deletes invite for this player to this instance.

instances
Displays all instances in this server and their privacy, capacity, gametype, instance_name.

Reads the database for instance tuples.

gametypes
Displays all available game types with description.

Reads the database for gametype tuples.

myhighscores
Displays personal best for each game type the user has participated in through any server.

Reads the Leaderboard tuples.

leaderboard gametype, limit_number(optional)
Displays player and their personal best for the given gametype up to 10 by default. If wish to retrieve more tuples provide a second argument for limit.

Reads the Leaderboard tuples.

myinvites
Sends a private message to the user with information regarding all invitation for this player.
An invitation contains the server,instance_name and who invited the user.

Reads the Invite tuples.

Instance Commands:

help context(optional)
Provides the user with information on available commands in instance or detailed information on the command specified in the context.

leave
Leave the instance the user is in right now

Deletes the tuple in Plays corresponding to the user and instance and updates freespots by adding one.

invite @recipient_name1 @recipient_name2 …
Sends an invitation to the recipient(s).
An invitation contains the server,instance_name and who invited the user.

Creates a tuple in Invite.

teams
Displays all teams for this instance and the members in them.

Read Team tuple

leaveteam
The user leaves the current team.

Update team_id of Plays tuple for this player in this instance to null.

createteam team_name
Creates a team in the instance with given team name if it does not exist yet

Read team tuple for this instance from database to make sure no duplicate name.
Create Team tuple.

Jointeam team_name
Joins the given team if it exists. Overwrites the current team the user is in to the given one.

Read Team table to make sure the team exists. Update team_id of Plays tuple for this player in this instance to the team's id.

autoassign
Automatically adds all players in the instance without a team yet into teams in the instance as evenly as possible in a random fashion.

Reads players already in teams in the instance. Calculates how many players need to go to each team to evenly distribute. Read all players not in a team yet and update their team_id field in Plays table to the randomly assigned team.

start config1(optional), config2(optional) ...
Starts the game of this instance's gametype. If the gametype for the instance requires configuration before starting, enter those arguments. What fields to configure can be seen using !help: start.

Update started field in instance to true. Creates and updates all tuples and fields specific to that gametype.

1 d 100 Actions:

roll
The user rolls a d100 dice

Quiz Bowl Actions:

buzzing
The user may buzz by sending any message either while the question is being read or within five seconds of completion. Once buzzed, the bot will give the buzzer 10 seconds to answer the question. If correct, the user gains points and moves on to the next question. If incorrect, the other team gets to answer the same question.

answering
Once you have buzzed, you will be given 10 seconds to answer the question. This is done by simply sending a message containing the answer.

Edit Distance algorithm to allow small typos to answer. Value used to assess is the ratio of (edit distance of user input to correct answer) to (the length of correct answer).

Use Of Stored Procedures (SQL)

Stored procedures were created and used for well defined Database.java methods. Many of the methods in Database class sets custom table/field names so benefits from stored procedures are minimal to none.

Stored Procedures in Database:

- createPlays(in playerid bigint, in instanceid bigint)
- deletePlays(in playerid bigint, in instanceid bigint)
- joinInstance(in playerid bigint, in instanceid bigint)
- deleteInstance(in instanceid bigint)
- createInstance(in privacy boolean, in gameid int, in capacity int, in instanceid bigint, in serverid bigint, in instancename varchar(256))
- leaveInstance(in playerid bigint, in instanceid bigint)

Lessons learned

Technical Expertise Gained:
To create the timer functionality on the front end, we learned how to use ScheduledFuture.
We learned how to use discord java api and how discord is structured and how to communicate between java and mysql database using the connector.
Alternative Design:
Design choices between adding specific instances to handle all requests of a given type, vs. one per ‘user’ (might try one GameManager per “instance” if redoing front end from scratch)
Group work insights
Github proved to be extremely useful in collaborating and git was mostly new to us. It keeps track of all commits and aided in propagating code changes between our computers.

Future work

The database will continue to support the functionalities of this bot as we use it and work on it as personal project.

- Quiz Bowl: Add complexity to answer response according to official rules
    - If the first answer is incorrect, any other member of the answerer’s team has 5 seconds to buzz and answer. If they too are incorrect, the full question is read again, and only the other team may buzz.
- Fully implement pausing and resuming games
- Make use of lavaplayer api for audio related functionalities
- Make use of discord content network for image related functionalites
- Multiplayer visual novel
- Add more games?