DISCORD CHAT BOT

Report submitted in partial fulfilment of the requirement for the degree of

B.Tech.

In

Information Technology

Under the supervision of

Dr. Madhur Jain

By

Mohammad Aasim Kaif
(01520803118)
Vansh Verma
(03420803118)

Bhagwan Parshuram Institute of Technology
PSP – 4, Sector – 17, Rohini, Delhi – 89

September 2021

DECLARATION

This is to certify that Report titled "**Discord Chat-Bot**", is submitted in partial fulfilment of the requirement for the award of degree of B.Tech. In Information Technology to **BPIT** Rohini Delhi affiliated to **GGSIP University**, Dwarka, Delhi. It comprises of my original work. The due acknowledgement has been made in the report for using other's work.

Date : 1st October, 2021

Mohammad Aasim Kaif (01520803118)

Vansh Verma (03420803118)

Certificate by Supervisor

This is to certify that Report titled “Discord Chat-Bot” is submitted by **Mohammad Aasim Kaif** (01520803118) and **Vansh Verma** (03420803118) in partial fulfilment of the requirement for the award of degree of B. Tech in Information Technology to BPIT Rohini affiliated to GGSIP University, Dwarka, Delhi. It is a record of the candidates own work carried out by them under my supervision. The matter embodied in this Report is original and has not been submitted for the award of any other degree.

Date :

Signature

(Supervisor)

Training Coordinator Certificate

Training Coordinator Certificate This is to certify that Report titled "**Discord Chat-Bot**" is submitted by **Mohammad Aasim Kaif** (01520803118) and **Vansh Verma** (03420803118) in partial fulfilment of the requirement for the award of degree of B. Tech in Information Technology to BPIT Rohini affiliated to GGSIP University, Dwarka, Delhi. The matter embodied in this Report is original and has been dully approved for the submission.

Date :

Signature

(Coordinator)

udemy

Certificate no: UC-f0f3c6f3-5cd3-4abe-9fd9-719f6788ff0c
Certificate url: ude.my/UC-f0f3c6f3-5cd3-4abe-9fd9-719f6788ff0c
Reference Number: 0004

CERTIFICATE OF COMPLETION

Complete Core Java In Simple Way

Instructors **DURGASOFT NAGOOR BABU**

Vansh Verma

Date **Aug. 28, 2021**
Length **100.5 total hours**

INTERNSHALA TRAININGS

Certificate of Training

MOHAMMAD AASIM KAIF

from **Bhagwan Parshuram Institute of Technology** has successfully completed a six weeks online training on **Core Java** from 2nd June, 2021 to 14th July, 2021. The training consisted of Getting Started with Java, Leveraging Basic Concepts, Object Oriented Programming and Java App Development modules. MOHAMMAD AASIM scored 100% marks in the final assessment and is a top performer in the training. We wish MOHAMMAD AASIM all the best for the future.

Sarvesh Agrawal
Founder & CEO, Internshala

Date of certification: 2021-06-17 Certificate no. : 3C928EC7-A514-C8D2-E359-C7EAC89776D8

For certificate authentication, please visit https://trainings.internshala.com/verify_certificate

ACKNOWLEDGEMENT

Every work accomplished is a pleasure – a sense of satisfaction. However, it would not have been possible without the kind support and help of many individuals in the organization. We would like to extend our sincere thanks to all of them.

We would like to give our sincere and heartfelt thanks to our esteemed guide for providing us with the right guidance, advice at the crucial junctures & also for his support in completing the project.

We would like to express my gratitude towards our parents for their kind co-operation and encouragement which help me in completion of this project. We would like to express our special gratitude and thanks to industry persons for giving us such attention and time.

In preparing the project report, we had to take the help and guideline of some respected persons, who deserve my greatest gratitude. The completion of this project gives us much pleasure. We would like to show our gratitude to our **HOD Mr. Abhishek Swaroop** and Professor **Dr. Madhur Jain**, for suggesting us to take the course on **Discord Chat-Bot** and to help us successfully complete our major project.

(Signature of the student with Date)

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

This project consisted of creating a Discord Bot for our Discord server which can perform some operation on taking some commands form the user. We created code that set a Discord Bot using Java and JDA. To use our Bot we just have to type some command as a message in the server and our bot is smart enough to perform appropriate operation on the basis of command.

Bots on Discord, the group messaging platform, are helpful artificial intelligence that can perform several useful tasks on your server automatically. That includes welcoming any new members, banning troublemakers, and moderating the discussion. Some bots even add music or games to your server.

The bot's purpose is to make the Discord server a bit more fun by making it interactive and interesting. This bot brings together various people from all over the world and helps them remain connected to each other. Once people join the server they can use all the features of the bot and join voice channels to communicate or share ideas thoughts or just have a chat. This is a better version of WhatsApp, Microsoft teams and has surpassed both of them in the same field. Discord is a VoIP, instant messaging and digital distribution platform designed for creating communities. Users communicate with voice calls, video calls, text messaging, media and files in private chats or as part of communities called "servers". Servers are a collection of persistent chat rooms and voice chat channels. Discord runs on Windows, macOS, Android, iOS, Linux, and in web browsers.

The objective of this project is to create a smart Discord Bot which can behave intelligently on user commands. Our Bot can perform various function on discord server like playing music on any voice channel, kicking out any user, banning any user, share memes or jokes, tells all the details about an user, deleting messages etc.

Chapter 1
INTRODUCTION

Discord is a free voice, video, and text chat app that's used by tens of millions of people ages 13+ to talk and hang out with their communities and friends.

People use Discord daily to talk about many things, ranging from art projects and family trips to homework and mental health support. It's a home for communities of any size, but it's most widely used by small and active groups of people who talk regularly.

The vast majority of servers are private, invite-only spaces for groups of friends and communities to stay in touch and spend time together. There are also larger, more open communities, generally centered around specific topics such as popular games like Minecraft and Fortnite. All conversations are opt-in, so people have total control over who they interact with and what their experience on Discord is.

People love Discord because it’s a home for all their communities and groups of friends. It's a place where they can be themselves and spend time with other people who share their interests and hobbies. There's no algorithm deciding what they should see, no endless scrolling, and no news feed. Conversations on Discord are driven by shared interests.

1.1 History

So, in 2015, Stan and Jason started to bring Discord to life. People from all over the world loved it. Discord made it easy to genuinely communicate with friends, going beyond casual talking. Friends were staying in touch with their various communities. Discord was making it easy to participate in conversation, hopping around text, voice and video to talk. The technology was complex but the goal was simple: make Discord an inviting and comfortable home to jump into with your communities and friends. Within a few years, Discord began to take off, and devoted people who loved our product sprung up around the world.

In March 2021, Discord announced it had hired its first finance chief, former head of finance for Pinterest Tomasz Marcinkowski. An inside source called this one of the first steps for the company towards a potential initial public offering, though co-founder and CEO Jason Citron stated earlier in the month he is not thinking about taking the company public. Discord doubled its monthly user base to about 140 million in 2020. The same month, Bloomberg News and The Wall Street Journal reported that several companies were looking to purchase Discord, with Microsoft named as the likely lead buyer at a value estimated at $10 billion. However, they ended talks with Microsoft, opting to stay independent. Instead, Discord launched another round of investment in April 2021. Among those investing into the company was Sony Interactive Entertainment; the company stated that they intended to integrate a portion of Discord's services into the PlayStation Network by 2022.

1.2 Discord Features

Discord has its own vocabulary. You might hear your teen or students using these words when talking about Discord.

Server: Servers are the spaces on Discord. They are made by specific communities and friend groups. The vast majority of servers are small and invitation-only. Some larger servers are public. Any user can start a new server for free and invite their friends to it.

Channel: Discord servers are organized into text and voice channels, which are usually dedicated to specific topics and can have different rules.

- In text channels, users can post messages, upload files, and share images for others to see at any time.
- In voice channels, users can connect through a voice or video call in real time, and can share their screen with their friends - we call this Go Live.

DMs and GDMs: Users can send private messages to other users as a direct message (DM), as well as start a voice or video call. Most DMs are one-on-one conversations, but users have the option to invite up to nine others to the conversation to create a private group DM, with a maximum size of ten people. Group DMs are not public and require an invite from someone in the group to join.

Go Live: users can share their screen with other people who are in a server or a DM with them.

Nitro: Nitro is Discord’s premium subscription service. Nitro offers special perks for subscribers, such as the option to customize your Discord Tag, the ability to use custom emotes in every server, a higher file upload cap, and discounted Server Boosts.

Server Boosts: If your teen is a big fan of a community, they might want to boost the community’s server (or their own). Like Nitro, Server Boosts give servers special perks like more custom emotes, better video and voice quality, and the ability to set a custom invite link. Server Boosts can be bought with Nitro or purchased separately.

1.3 Discord Bot

Bots on Discord, the group messaging platform, are helpful artificial intelligence that can perform several useful tasks on your server automatically.

That includes welcoming any new members, banning troublemakers, and moderating the discussion. Some bots even add music or games to your server.

Don't worry, you don't have to be a coding genius to add an automaton to your server. You just download pre-made bots and customize what they do and say.

We'll cover how to download and use the popular MEE6 bot as a welcoming bot, a moderating bot, and a bot that alerts your server when you're streaming on Twitch.

How to add a bot to Discord

You can add a number of popular, pre-made bots to your server. One of

the most popular is MEE6.

1. Visit this page to download MEE6.
2. Click the button that says "Add to Discord."
3. Provide your Discord login to authenticate your account. If a pop-up asks for permission to use your account, click "Authorize."
4. Click "Set up MEE6" next to your server name.
5. Select your server in the pop-up window and click "Continue."
6. Select or deselect the permissions you'll give this bot.

7. Scroll down and click "Authorize."

1.4 Top Bots to have on your Server

Discord is one of the most popular ways gamers communicate and organize with one another, but sometimes the systems and settings in the application aren't enough.

By using third-party bot programs, Discord users can enhance their experience by recording voice chats, playing music for friends, organizing large groups, and moderating channels.

While many bots have a specific purpose, some are especially well known and used by users on Discord for a variety of purposes.

1.4.1 Mee6

Mee6 is one of the premier Discord bots used for moderation. With the ability to craft customized moderation settings and commands, you'll be able to rest easy without needing to watch your server like a hawk.

Meanwhile, the bot can help you make automatic alerts from social media or Twitch streams as well as give users a way to give themselves appropriate roles within the server without needing the assistance of an admin.

1.4.2 Groovy

Groovy is one of the more popular ways that people share music with one another in Discord. The bot uses a /play command that lets people select YouTube music videos and plays them in voice channels.

This is especially useful when you're playing games with your friends and want to have a shared inspirational music experience. The bot lets you queue multiple items, skip songs you're not interested in, and pause tracks when necessary.

1.4.3 Rythm

Rythm is also a music bot that you can add to your Discord, and it has many of the same functionality of Groovy. Selecting one over the other will largely be based on personal preference.

If your server is particularly congested on a regular basis, you may have a good reason to invite both to your server. Having both bots will allow two separate voice channels to listen to music at the same time if coordinated properly.

1.4.4 Raid-Helper

Raid-Helper is a third-party Discord bot used to help people schedule and organize events. This is used especially frequently to coordinate things like World of Warcraft raids, but it can also be used to do straw polls or make customized events.

While Discord servers are useful when it comes to text communication, these event posts make organizing as easy as reacting to a post that edits itself to show all of the events expected attendance, among other things.

1.4.5 Craig

For those who use Discord for more formal purposes, Craig is a way for you to easily record and keep conversation in voice chat. This can be especially useful if you're a journalist or content creator conducting an interview via Discord. It can also be useful if you're holding a meeting and want to make sure that the contents of the meeting are archived.

The bot doesn't need to record everything in your voice channels. It will only record if you use a specific command to have it join, and it will stop recording if you type a command to have it leave your voice channel.

Chapter 2

RELATED WORK

When we planned this program, we looked around to see if something similar had been done.

2.1 Androz2091/AtlantaBot:

A discord bot developed using JavaScript.

Atlanta is a open source Discord bot coded in JavaScript with Discord.js and Mongoose by Androz2091.

Complete Bot

- Atlanta offers (non-exhaustive list):
- Support for commands in direct messages
- Support for translations (unlimited languages)
- Guild configuration (prefix, ignored channels, etc...)
- Commands made pleasant thanks to the many emojis
- Support for Discordbots.org votes with rewards

Atlanta also adds a new mention like @everyone and @here, the @someone, which allows you to pick a random member of the server!

2.2 Egglord:

Another bot developed using JavaScript.

Egglord is an open source, fully customized Discord bot that is constantly growing. You can invite it to your Discord server using this link! Also, you can join the official Egglord Support Server for all questions, suggestions, and assistance! It comes

packaged with a variety of commands and a multitude of settings that can be tailored to your server's specific needs.

Egglord also comes packed with a variety of features, such as:

- Welcome messages and farewell messages.
- Extensive Logging for 37 events.
- Slash Commands
- Advanced auto-moderation.
- Audio filters for music plugin.
- Custom playlist support.
- Multi-language support.
- Giveaways
- Reaction roles
- And much more! There are over 40+ settings to tweak!

2.3 Red Discord Bot:

Red is developed using Python.

Red is a fully modular bot – meaning all features and commands can be enabled/disabled to your liking, making it completely customizable. This is a self-hosted bot – meaning you will need to host and maintain your own instance. You can turn Red into an admin bot, music bot, trivia bot, new best friend or all of these together!

Installation is easy, and you do NOT need to know anything about coding! Aside from installing and updating, every part of the bot can be controlled from within Discord.

The default set of modules includes and is not limited to:

- Moderation features (kick/ban/softban/hackban, mod-log, filter, chat cleanup)

- Trivia (lists are included and can be easily added)
- Music features (YouTube, SoundCloud, local files, playlists, queues)
- Stream alerts (Twitch, YouTube, Picarto)
- Bank (slot machine, user credits)
- Custom commands
- Imgur/gif search
- Admin automation (self-role assignment, cross-server announcements, mod-mail reports)
- Customisable command permissions

2.4 Onyx Discord Bot

A discord bot created using Node.js

Onyx is a multi-purpose media and utility bot, containing over 50 commands to help you bring the best of the web to your server.

Watch YouTube videos, search for GIFs and stickers, get answers to everything, and more, with Onyx. This bot is powered by Node.JS, and coded using the Discord.JS library.

Make electronic music, memes, edit images, get answers to everything, and more, with Onyx, a Discord bot powered by Node.JS, and coded using the Discord.JS library.

Onyx has over fifty commands, including those for:

- meme generation
- gif and sticker searching
- searching YouTube
- music production
- fetching info/live imagery from NASA
- interacting with IBM's Watson API

getting information on any topic

Chapter 3

PROBLEM STATEMENT

3.1 Motivation

We all love to spent time with our family, friends etc. But since the COVID era we weren't able to do so. As a result we came close more to social media and eventually came to know about Discord, and we loved it. Discord is a free voice, video, and text chat app that's used by tens of millions of people ages 13+ to talk and hang out with their communities and friends.

"Discord is WhatsApp for cool people"

But it still had it flaws. We have to handle a channel on our own which was literally so annoying. Like adding a new comer, banning someone, muting playing music, sharing memes/jokes and the list goes on. So we decided to create a bot who will perform our job on just taking some command, so that we don't get interrupted while having some cool session with our friends or family.

We made everything automated on our channel. You just have to enter some command and our bot will take care of the rest.

Discord is growing in popularity. As such, automated processes, such as banning inappropriate users and reacting to user requests are vital for a community to thrive and grow.

Automated programs that look and act like users and automatically respond to events and commands on Discord are called bot users. Discord bot users (or just bots) have nearly unlimited applications.

With a bot, it’s possible to automatically react to the new member joining your guild. You can even customize its behaviour based on context and control how it interacts with each new user.

3.2 PROBLEM STATEMENT

Various trends depicting the importance of chatbots have been adopted by modern organizations. For instance, messaging apps have turned out to be one of the effective means of communication across mobile devices where users access their business services through message apps. Now, chatbots are embedded within these apps as an interface between users and businesses.

Major technological developments have been seen in deploying bots to support this paradigm shift. Bots provide conversational interfaces with simple UI, resulting in reduced development and maintenance costs. Undoubtedly, this technology holds the potential to impart engaging workforce experiences for people of all ages. The fact that most of the organizations embrace chatbots reinforces the belief that they will reshape the future of business-to-customer interactions.

3.3 OBJECTIVES

We intend to develop a Discord bot named “Hokage” in which user can perform actions related to Discord server using commands only. The platform that will be used for developing the project will be Java IDE IntelliJ IDEA. We prefer IntelliJ IDEA as the operating platform because of its ease of use and easy debugging abilities. We will be implementing classes (number will depend on the algorithm developed) as Java is mainly an OOP(object oriented programming) language. We also make use of the inbuilt classes defined in Java library and the various methods in them. We plan to use the Input Output functions and also the JDA library. JDA strives to provide a clean and full wrapping of the Discord REST api and its Websocket-Events for Java. This library is a helpful tool that provides the functionality to create a discord bot in java.

CHAPTER 4

SYSTEM ANALYSIS AND DESIGN

Requirement analysis results in the specification of software's operational characteristics indicates software's interface with other system elements and establish constraints that software must meets. Requirement analysis allows the software engineer (sometime called Analyst or Modeler in this role) to elaborate on basis requirements during earlier requirement engineering task and build models that depict user scenarios, functional activities, problem classes and their relationships, system and class behavior and the flow of data as it is transformed.

The requirements analysis task is a process of discovery, refinement, modeling and specification. The scope, initially established by us and refined during project planning, is refined in details. Model of the required data, information and control flow and operations behavior are created.

4.1 SOFTWARE REQUIREMENT SPECIFICATION

4.1.1 Functional Requirement:

The functional requirements for a system describe what system do.

• The developed system should update its User Interface as soon as any player inserts Disc.

• System shall take both player name as Input and assign different coor Disc to them.

• System must provide the quality of service to user.

• System must insert in only those column which the player has clicked.

4.1.2 Non-Functional Requirement:

As the name suggest these are the requirements that are not directly interacted with specific functions delivered by the system.

• Performance: Quite high. You can play your turn as soon as you mouse got clicked.

• Functionality: This software will deliver on the functional requirements.

• Availability: The game can be played only on desktop as it is an desktop software.

• Flexibility: You don't have to install it in your system as it is available in .jar format.

4.1.3 Hardware Requirements:

• Processor- Dual core processor @ 1.65 GHz or above

• RAM- 256 MB or above. • Hard Disk- 10 GB or above.

• Monitor- 14" VGA

• Mouse, Standard 104 enhanced keyboard

4.1.4 Software Requirements:

• Operating System- Windows 7/8/10

• Language Used: Java

• JDK 11 Version or above

• Documentation- MS-Word (latest version MS-Word 2007 & above)

4.2 USE CASE DIAGRAM:

A use case defines a goal-oriented set of interactions between external users and the system under consideration or development. Thus, a Use Case Scenario is a description that illustrates, step by step, how a user is intending to use a system, essentially capturing the system behavior from the user's point of view. A software development life cycle goes through several stages: design, analysis, implementation, and close out. Use case analysis can happen at any stage. A use case analysis is used to design a system from the viewpoint of the end user, the person actually using the site or software. The use case analysis attempts to convey information on the system requirements and usage, the role of the user, the system actions in response to the user, and what the user will

receive from the system. Due to their simplistic nature, use case diagrams can be a good communication tool for stakeholders. The drawings attempt to mimic the real world and provide a view for the stakeholder to understand how the system is going to be designed. The use case diagrams convey the intent of the system in a more simplified manner to stakeholders and that they were "interpreted more completely than class diagrams". The purpose of the use case diagrams is simply to provide the high-level view of the system and convey the requirements in laypeople's terms for the stakeholders. Additional diagrams and documentation can be used to provide a complete functional and technical view of the system. The use cases are analyzed so that they can be converted into more technical requirements for the software developers. Later, when the software is developed, the use case is analyzed to develop the testing scenarios, also referred to as test cases, and so that they can be included in the user documentation.

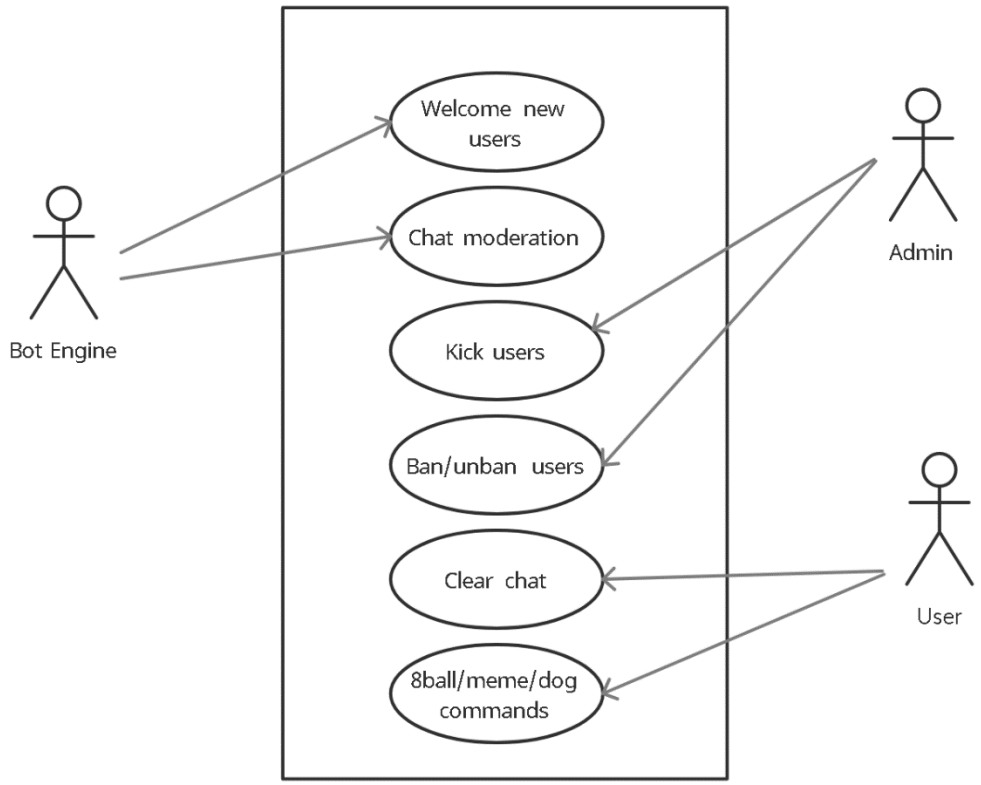

Fig 4.1 Use-case Diagram

4.3 PROCESS MODEL:

Process Model are processes of the same nature that are classified together into a model. Thus, a process model is a description of a process at the type level. Since the process model is at the type level, a process is an instantiation of it. The same process model is used repeatedly for the development of many applications and thus, has many instantiations. One possible use of a process model is to prescribe how things must/should/could is done in contrast to the process itself which is really what happens. A process model is roughly anticipation of what the process will look like. What the process shall be determined during actual system development. The goal of a process model is to be:

• Descriptive

1.Track what actually happens during a process.

2.Take the point of view of an external observer who looks at the way a process has been performed and determines the improvements that must be made to make it perform more effectively or efficiently.

• Prescriptive

1.Define the desired processes and how they should/could/might be performed.

2.Establish rules, guidelines, and behavior patterns which, if followed, would lead to the desired process performance. They can range from strict enforcement to flexible guidance.

• Explanatory

1.Provide explanations about the rationale of processes.

2.Explore and evaluate the several possible courses of action based on rational arguments.

3.Establish an explicit link between processes and the requirements that the model needs to fulfil.

4.Pre-defines points at which data can be extracted for reporting purposes.

4.3.1 Incremental Model :

Incremental model is used as the process model in our system. To save actual problems in an industry setting, Software Engineering must incorporate a development strategy that encompasses the process, method and the tool layers; this strategy is often referred as process model. A process model for Software Engineering is chosen base on the nature of the Project and its application. For our project, we have selected Incremental Model.

1. Using these models, a limited set of customer requirements are implemented quickly.

2. Modified and expanded requirements are implemented step by step.

3. It combines elements of linear Sequential Model with the iterative Philosophy of prototyping.

4. Each linear sequence produces a deliverable Increment of the Software. 5. Each linear Sequence is divided into 4 parts: -

• Analysis

• Design

• Code

• Testing

1. Analysis: It includes understanding of information domain, required functions, behavior, performance and interface. Requirements for the system and software are documented.

2. Design: It is multiple processes that include four attributes of program data structure, software architecture, interface representation and procedural detail.

3. Coding: Translation of design to machine code is done by this step.

4. Testing: It focuses on Logical internals of Software and ensures that all statement is correct to uncover all hidden errors. For an incremental model, the first Increment is developed as a core model, which is used by the customer. Then as things are added after the first delivery, product gets and better.

4.3.2 Why we use Incremental Model?

The main aim of using the model is the reason that we have to add more features in the existing modules to increase project reliability and usability. Using this model, we can adapt to the changing requirements of the customer which helps in developing the project in relatively small amount of time. The next increment implements customer suggestions plus some additional requirements in the previous increment. The process is repeated until the project is completed.

4.3.3 Characteristics of Incremental Model

1. Using this model, a limited set of customers' requirements are implemented quickly and are delivered then modified and expanded requirements are implemented step by step.

2. Each increment produces the product which is submitted to customer and suggests some change and increment implements that changes with some extra requirements to previous.

3. Incremental model does not facilitate the development of project in one go. This is useful for developing modules and then testing them which helps us to modularize the entire project for better handling. So finally, it is easier to develop project in increments. We can develop a working prototype 1st with just basic functions and then build upon this prototype in later increments. This will help to reduce the complexity of system by dividing entire system in different levels of priority.

JAVA projects are highly iterative; as you progress through the lifecycle, you'll find yourself iterating on a section until reaching a satisfactory level of performance, then proceed forward to the next task (which may be circling back to an even earlier step). Moreover, a project isn't complete after you ship the first version; you get feedback from real - world interactions and redefine the goals for the next iteration of deployment.

4.4 FUCTIONAL MODELLING

4.4.1 Data Flow Diagram

Data flow diagram (DFD) is also called as Bubble Chart is a graphical technique, which is used to represent information flow, and transformers those are applied when data moves from input to output. DFD represents system requirements clearly and identify transformers those becomes programs in design. DFD may further partitioned into different levels to show detailed information flow e.g. level 0, level 1 etc

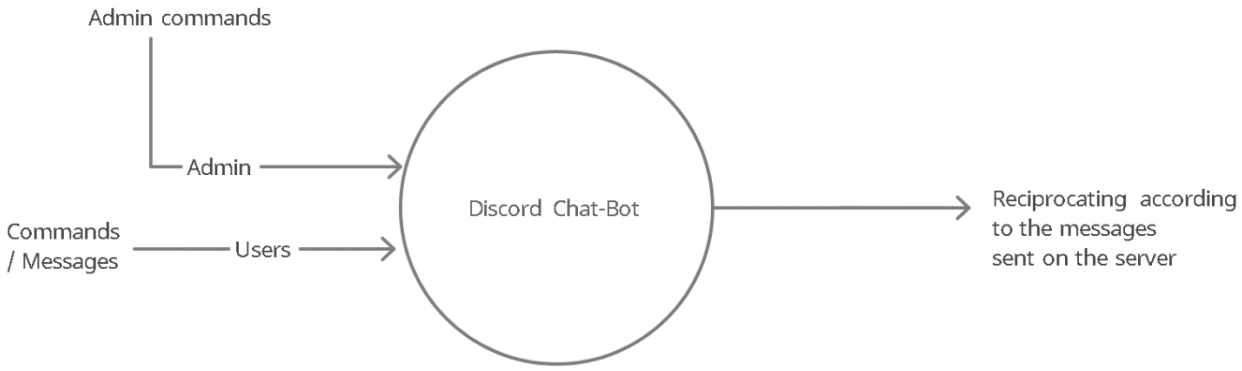

Fig 4.2 Data Flow Diagram

CHAPTER 5

PROPOSED WORK

5.1 DATASET

We decided to develop the Chat bot in June 2021. As we had some knowledge in developing field at that time. We asked our Professors, Seniors on how to implement the Idea. What we have to learn in order to make this chat bot.

We eventually came to know about Java, how this language is rich in API. As it has its own User Interface scheme, we thought it would be easy if we learn Java. Then we opt for the Java Language which we eventually found so interesting.

Java is one of the most preferred programming language being used in IT industry. It is rich in API so it helped us saving a lot of time throughout the development.

We wanted to develop a Chat bot to be used in discord servers. We decided to use Java for internal programming logic. We also wanted to add more features but there were some limitations that occurred such as limited libraries coded inside of JDA. Although there were 100s of libraries already implemented inside the JDA but still somethings like accessing third parties to some extent was limited. The other limitation of this system is that it isn't the API can make up to 50 Requests per second, since if our bot get popular enough it may be impossible to stay below 50 requests per second during normal operations.

We studied a lot of projects on discord bots implemented by different developers using different languages. Some of them were implemented on javaScript, some were on python. The issue we found out using all these versions was that all API's had some restrictions and due to the restrictions some times the program doesn't respond. One of the restrictions that we came through was the policies of the third parties such as Youtube policies which block the discord bots from playing their content through them but we found some ways to overcome that such as using short links. Sometimes songs were not played due to youtube policies but non-license songs were played without any difficulties.

5.2 INSTALLATIONS

5.2.1 JDK

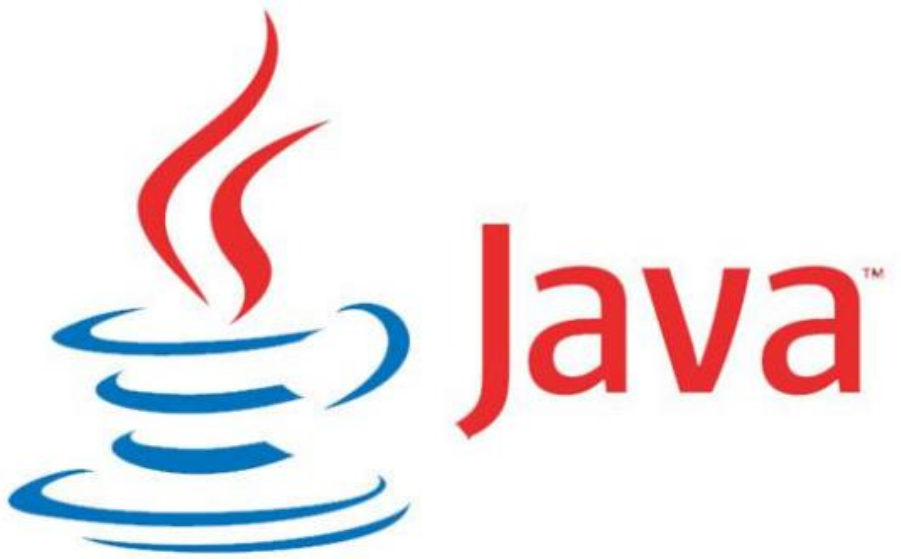

Fig 5.1 Java

The Java Development Kit (JDK) is one of three core technology packages used in Java programming, along with the JVM (Java Virtual Machine) and the JRE (Java Runtime Environment). It's important to differentiate between these three technologies, as well as understanding how they're connected:

- The JVM is the Java platform component that executes programs.

- The JRE is the on-disk part of Java that creates the JVM.

- The JDK allows developers to create Java programs that can be executed and run by the JVM and JRE.

Developers new to Java often confuse the Java Development Kit and the Java Runtime Environment. The distinction is that the JDK is a package of tools for developing Java-based software, whereas the JRE is a package of tools for running Java code.

The JRE can be used as a standalone component to simply run Java programs, but it's also part of the JDK. The JDK requires a JRE because running Java programs is part of developing them.

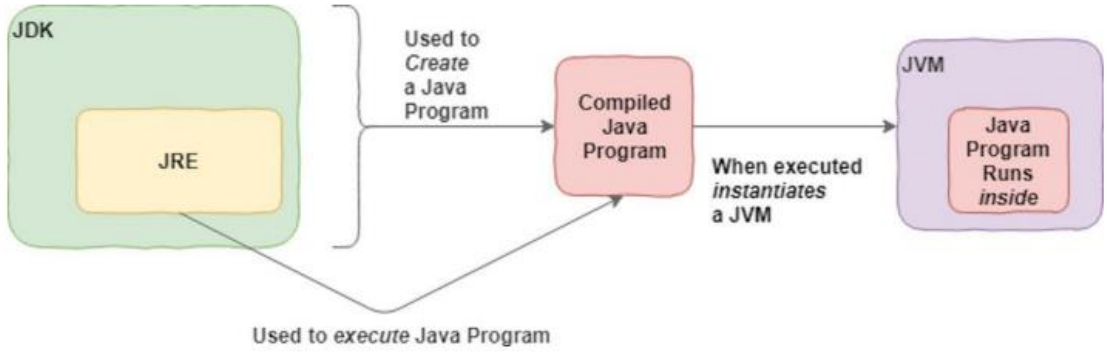

Fig 5.2 JDK Architecture

5.2.2 History of JDK:

¬ JDK Alpha and Beta (1995)

The Java Alpha and Beta was the first releasesbut they have highly unstable APIs and ABIs. The supplied Java web browser was named WebRunner.

¬ JDK 1.0 (January 23, 1996)

It was the first stable released version of Java. Its codename was **Oak**.The first stable version of JDK was JDK 1.0.2 and it was called Java 1.

Up to JDK 1.0.1, private and protected keywords could be used together to create yet another form of protection which used to act as a restriction to methods or variables mainly to subclasses of a given class. In JDK 1.0.2, this capability has been removed.

¬ JDK 1.1 (February 19, 1997)

Some additions were included to this version. i.e.

o The concept of Inner Class

o JavaBeans

o JDBC

o RMI

o AWT event model was totally reshaped.

o Reflection(which supported Introspection only, modification was not possible at runtime).

o JIT(Just In Time) compiler on Microsoft Windows platforms, produced for JavaSoft by Symantec

o Internationalization and Unicode support originating from Taligent.

¬ J2SE 1.2 (December 8, 1998)

Its codename was **Playground**. First time, it was called **J2SE (Java 2 Platform, Standard Edition)** .It replaced JDK to recognize the base platform from **J2EE (Java 2 Platform, Enterprise Edition) and J2ME(Java 2 Platform, Micro Edition)** .It was a very important java release as it tripled the size of the Java platform to 1520 classes in 59 packages.

Some additions were included to this version. i.e.

o Java plug-in

o Java IDL, an IDL implementation for CORBA interoperability

o Collections framework

o the Swing graphical API was integrated into the core classes

o Sun's JVM was equipped with a JIT compiler for the first time

¬ J2SE 1.3 (May 8, 2000)

Its codename was **Kestrel**. Some additions were included to this version. i.e.

o HotSpot JVM included.

o RMI was modified to support optional compatibility with CORBA. o JNDI (Java Naming and Directory Interface).

o Java Platform Debugger Architecture (JPDA) included.

o JavaSound.

o Synthetic proxy classes.

¬ J2SE 1.4 (February 6, 2002)

Its codename was **Merlin**. It was the first Java platform which was released under the Java Community Process. Some additions were included to this version. i.e.

o Improved libraries.

o Perl regular expressions included.

o Provided exception chaining (It allows an exception to encapsulate original lower-level exception).

o IPv6 support (Internet Protocol version 6).

o Logging API (Specified in JSR 47.) o Image I/O API for reading and writing images in formats like JPEG and PNG. o XML parser and XSLT processor integrated.

o Security and cryptography extensions (JCE, JSSE, JAAS) integrated. Support and security updates for Java 1.4 ended in October 2008.

¬ J2SE 5.0 (September 30, 2004)

Its codename was **Tiger**. It was originally numbered 1.5, which is still used as the internal version number. So, it was changed to 5.0 to "better reflect the level of maturity, stability, scalability and security of the J2SE". This process also was released under the Java Community Process.

Support and security updates for Java 5.0 ended on November 3, 2009 but updates were available to paid Oracle customers until May 2015.

J2SE 5.0added some significant new language features:

o It provided compile-time (static) type safety for collections and eliminates the need for most typecasts.

o Used Metadata or annotations.

o Autoboxing/unboxing.

o Enumerations.

o Enhanced for each loop.

o Improved semantics of execution for multi-threaded Java programs.

o Static imports. There were also some improvements in standard libraries:

o Automatic stub generation for RMI objects.

o Swing: It provided a skinny look and feel.

o The concurrency utilities in package java.util.concurrent.

o Scanner class for parsing data from various input streams and buffers.

Java 5 was the last release of Java which officially supportedthe Microsoft Windows 9x line (Windows 95, Windows 98, Windows ME).

Windows Vista was the last version of Windows that J2SE 5 supported before going to end in October 2009.

Java 5.0 is the default version of Java installed on Apple Mac OS X 10.5 (Leopard). Java 6 can be installed

¬ Java SE 6 (December 11, 2006)

Its codename was **Mustang**. After the release of this version, Java replaced the name J2SE to Java SE and dropped the .0 from the version number.

Some additions were included to this version. i.e.

o Dropped the support for older Win9x versions.

o Scripting Language Support.

o Generic API for tight integration with scripting languages.

o Improved Web Service support.

o JDBC 4.0 support.

o Use a Java Compiler API to invoke a Java Compiler programmatically. After the release of Java 6, Sun released many updates to fix bugs.

¬ Java SE 7 (July 28, 2011)

Its codename was Dolphin. It was launched on 7, July 2011 but was made available for developers on July 28, 2011. Some additions were included to this version. i.e.

o JVM support for dynamic languages.

o Compressed 64-bits pointer.

o Strings added in switch.

o Automatic resource management in try-statement.

o Underscores allowed in numeric literals.

o Binary integer literals.

o Improved type interface for creating generic instance. (also called diamond operator <>)

o Improved catching and throwing. (catch multiple exceptions and rethrow with improved type checking)

o Provided Java Deployment rulesets.

It was the default version to download on java.com from April 2012 up to the release of Java 8.

¬ Java SE 8 (March 18, 2014)

Its codename was **Spider**. Although, codenames have been discontinued, but the codename **Spider** is common among java developers.

It includes some features which were proposed for Java SE 7 but added in Java SE 8.

o Language-level support for Lambda expressions.

o Allowed developers to embed JavaScript code within applications.

o Annotation of Java Types.

o Provided Date and Time API.

o Repeating Annotations.

o Launching of JavaFX applications.

o Removal of permanent generation.

Java SE 8 is not supported in Windows XP but after JDK 8 update 25, we can install and run it under Windows XP.

Java 8 is set as a default version to download from java.com from October 2014.

¬ Java SE 9 (September 21, 2017)

In 2016, Oracle discussed some features to release in Java 9. It was hoped that Java 9 would include better support for multi-gigabyte heaps, better native code integration, a different default garbage collector and a self-tuning JVM. The release of Java 9 was postponed many times and finally it was released on September 21, 2017.

It includes some specific features:

o Modularization of the JDK under Project Jigsaw.

o Provided Money and Currency API.

o Tight integration with JavaFX.

o Java implementation of reactive streams.

o More Concurrency Updates.

o Provided Java Linker.

o Automatic scaling and sizing.

¬ Java SE 10 (March, 20, 2018)

Java SE 10 was released to remove primitive data types and move towards 64-bit addressable arrays to support large data sets.It was released on 20 March 2018, with twelve new features confirmed. These features are:

o Local-Variable Type Inference

o Experimental Java-Based JIT Compiler This is the integration of the Graal dynamic compiler for the Linux x64 platform

o Application Class-Data Sharing This allows application classes to be placed in the shared archive to reduce startup and footprint for Java applications

o Time-Based Release Versioning

o Parallel Full GC for G1

o Garbage-Collector Interface

o Additional Unicode Language-Tag Extensions

o Root Certificates

o Thread-Local Handshakes

o Heap Allocation on Alternative Memory Devices

o Remove the Native-Header Generation Tool - javah

o Consolidate the JDK Forest into a Single Repository.

¬ Java SE 11

It is currently open for bug fixing.

Features of JDK:

Object Oriented

In Java, everything is an Object. Java can be easily extended since it is based on the Object model.

Platform Independent

Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform-independent byte code. This byte code is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.

Simple

Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master.

Secure

With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.

Architecture-neutral

Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.

Portable

Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. The compiler in Java is written in ANSI C with a clean portability boundary, which is a POSIX subset.

Robust

Java makes an effort to eliminate error-prone situations by emphasizing mainly on compile time error checking and runtime checking.

Multithreaded

With Java's multithreaded feature it is possible to write programs that can perform many tasks simultaneously. This design feature allows the developers to construct interactive applications that can run smoothly.

Interpreted

Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an incremental and light-weight process.

High Performance

With the use of Just-In-Time compilers, Java enables high performance.

Distributed

Java is designed for the distributed environment of the internet.

Dynamic

Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry an extensive amount of run-time information that can be used to verify and resolve accesses to objects at run-time.

5.2.3 IntelliJ IDEA

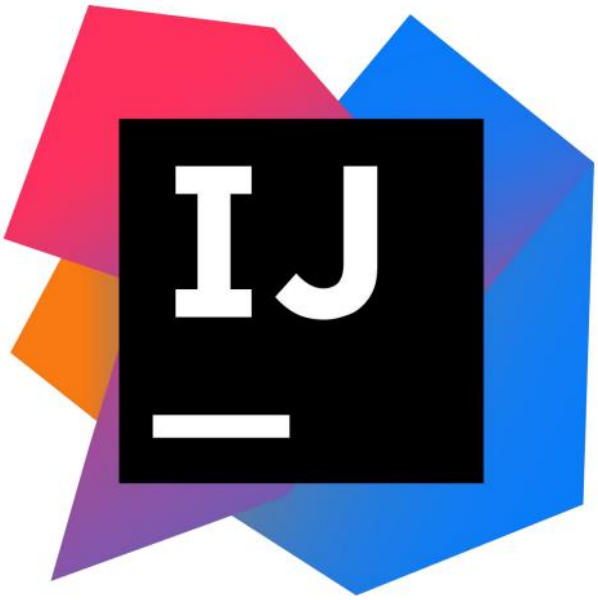

Fig 5.3 IntelliJ IDEA

IntelliJ is a powerful IDE designed by **JetBrains**. This particular IDE was created for importing, developing, modelling, and deploying computer software.

IntelliJ IDEA IDE for **Java** is the most powerful Integrated Development Enterprise. It is developed and maintained by JetBrains. This IDE is loaded with rich features and functionalities that, one can possibly imagine. In this tutorial, I'll walk you through its important capabilities.

It is available in two versions, namely:

• **Community Version** (Free and Open-Source)

• **Ultimate Version** (Licensed version)

Features of IntelliJ IDEA IDE :

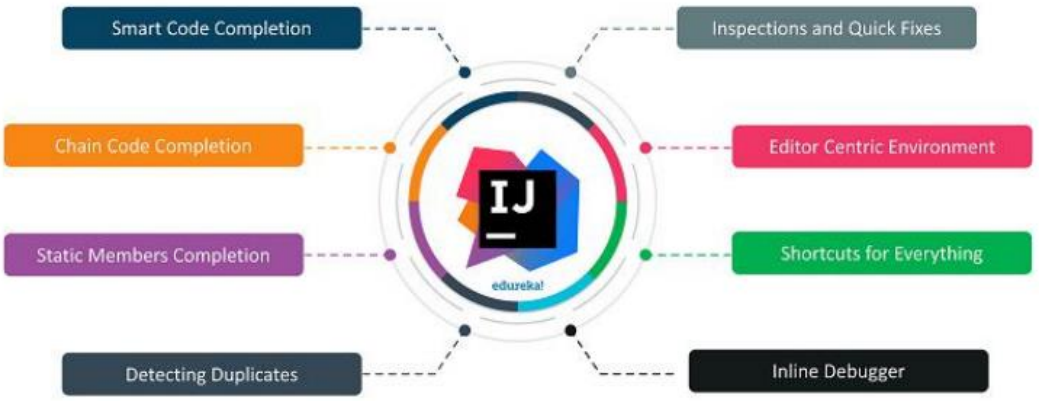

Fig 5.4 Features of IntelliJ IDEA

Smart code completion: IntelliJ can predict exactly what are you trying to **type**. It follows **context-based** code completion.

• **Chain code completion**: It is an advanced feature, designed for auto code competition. This lists the applicable symbols through methods and getters in the present context.

• **Static member's completion**: IntelliJ enables you to use static methods and constants that automatically adds the needed statements to avoid errors.

• **Detecting duplicates**: IDEA searches the duplicate code fragments on the go and gives suggestions to the user.

• **Inspections and quick-fixes**: When IntelliJ detects a mistake, a notification bulb pops up on the same line. Clicking on it gives you suggestions that will avoid you from making a mistake.

• **Editor-centric environment**: Instant pop-ups guide in checking additional information without leaving the present context.

• **Shortcuts for everything**: IDEA has shortcuts for nearly everything, including rapid selection and switching between tool windows.

• **Inline debugger**: Inline debugger enables the user to debug the application in IDEA itself.

5.3 IMPLEMENTATION :

5.3.1 Main class :

Our Main class will extend Listener Adapter class which is present in net.dv8tion.jda.api package present in the JDA. To build the Discord chat bot, we need to inherit this class and implement its abstract method createDefault(). In this method we need to mention the bot token code to connect the Code to the bot created through discord developer application option. After entering token inside of the method we need to call the method build() in order to build the bot and connect the code to the bot. and for that you need to follow the steps below :

- Create a application on discord developer option and mark it as bot
- Get the token code of the bot
- Put the token code inside fo createDefault method
- And use method build() to implement the bot

Important Functions/Constructors used inside Main Class

- **getPresence()** : This method is used to set the presence of the bot such as if bot if online , idle , offline etc. To set the status of the bot setStatus() method is used with getPresence() to manually set the status of the bot so that users can know If the bot is online idle offline or etc.
  To set what activity is bot doing to let the users know the current activity of the bot setActivity() method is used to set if bot is watching something, playing or some other activities present in the options.

- **registerCommands()** : This method is made to register all the commands in a separate class to avoid clustering and a positive point towards the clean code.
  All the registering of commands are done here and called through command manager.

- **eventListener()** : This method is called to keep an eye on the events happening as well, main agenda was to let moderation and welcome class call the eventListener() to greet the new comers as well as moderate the chat separately from the CommandManager class which also keeps an eye on the events happening in chat to get to know about the commands being called.

- **getAudioPlayerManager / getAudioPlayer** : These methods are responsible for the multimedia commands that is the music playing function of the chat bot.

- **getJDA()** : This method returns the referencing variable of the JDA Builder this is responsible for all the functioning between the code and the Java Discord API(JDA).

5.3.2 Command Manager

This class is the most important part of our discord chat bot. This class handles the internal implementations of our commands.

How things really going to work behind the commands being called in the chat is done by the Command Manager.

Command Manager class tells each of the commands when they are being called and calls the methods that are implemented inside each and every commands and hence this class controls all the commands implementations.

5.3.3 pom.xml file

A project object model or POM is the fundamental unit of work in Maven. It is an XML file that contains information about the project and configuration details used by Maven to build the project. It contains default values for the projects.

The POM was renamed from project.xml in Maven 1 to pom.xml in Maven 2. Instead of having a maven.xml file that contains the goals that can be executed, the goals or plugins are now configured in the pom.xml. When executing a task or goal, Maven looks for POM in the current directory. It reads the POM, gets the needed configuration information then executes the goals.

By using Maven, we have imported all the dependencies of the JDA so that hence when the JDA is used Maven import all the dependencies that are inside the API and get them ready to be used.

5.4 CODING PART

5.4.1 Main Class

(Building of the bot and registering of the commands as well as listening activities have been done in the main class)

```
public class Bot extends ListenerAdapter {
   public static JDA jda;
   public static JDABuilder jdaBuilder;
   public static AudioPlayerManager audioPlayerManager;
   public static commands.AudioManager audioManager;
   public static void main(String[] args) throws LoginException {

   /* Building Discord bot and token is used to connect back-end to front-end(BOT) */
    jda =
JDABuilder.createDefault("ODgzNjc4MDM1MTkzMjUzOTE5.YTNbTQ.05YtcaM
OdrBrCiRycEDeIZGAWXI").build();

      /* Status of BOT ( Online, idle , offline, Invisible) */
      jda.getPresence().setStatus(OnlineStatus.IDLE);

      /* Specifying what activity is bot doing right now : watching playing etc */
      jda.getPresence().setActivity(Activity.watching("The server"));

      /* Event Listener is responsible of handling events */
      jda.addEventListener(new Moderation());
      jda.addEventListener(new WelcomeMessage());


      audioPlayerManager = new DefaultAudioPlayerManager();
      audioManager = new AudioManager();

      AudioSourceManagers.registerRemoteSources(audioPlayerManager);
      jdaBuilder.enableIntents(GatewayIntent.GUILD_MEMBERS);

      registerCommands();
   }

   public static void registerCommands(){
      CommandManager commandManager = new CommandManager();
      jda.addEventListener(commandManager);
   }
```

```
    public static JDA getJda(){
        if(jda!=null){
            return jda;
        }
        return null;
    }
    public static AudioPlayerManager getAudioPlayerManager(){
        if(audioPlayerManager != null){
            return audioPlayerManager;
        }
        return null;
    }
    public static AudioManager getAudioManager(){
        if (audioManager != null)
            return audioManager;
        return null;
    }
}
```

5.4.2 Moderation Class

(All the messages are monitored using Listener Adapter and the chat is moderated whenever the bot is online. We have used HashSet of the Java Collections as the data structure to store the String values not allowed to be used in the server.)

```
public class Moderation extends ListenerAdapter {
    @Override
    public void onGuildMessageReceived(@NotNull GuildMessageReceivedEvent
event) {
        /* Main.Moderation Strings */
        HashSet<String> set = new HashSet<>();
        set.add("bad");
        set.add("abusive");
        set.add("looser");

        String receivedMessage = event.getMessage().toString();
        /* Getting each message for various checks */
        String[] message = event.getMessage().getContentRaw().split(" ");

        Member member;
        /* Main.Moderation */
        for (String bad: set) {
            if (receivedMessage.contains(bad.toLowerCase())) {
                member = event.getMember();
                event.getChannel().sendMessage(member.getAsMention() + "Please do not
use bad words. We do not tolerate such behaviour!!!").queue();
```

```
            event.getChannel().sendMessage("Moderators please look into this
matter").queue();
            break;
          }
       }
   }
}
```

5.4.3 WelcomeMessage Class

(In this class through enabling the Gateway intents of the bots we can access the activities of the users entering and leaving the server. Based on the activities users are greeted when they join the server.)

```
public class WelcomeMessage extends ListenerAdapter {


   String[] messages = {
         "[member] joined. You must construct additional pylons.",
         "Never gonna give [member] up. Never let [member] down!",
         "Hey! Listen! [member] has joined!",
         "Ha! [member] has joined! You activated my trap card!",
         "We've been expecting you, [member].",
         "It's dangerous to go alone, take [member]!",
         "Swoooosh. [member] just landed.",
         "Brace yourselves. [member] just joined the server.",
         "A wild [member] appeared."
   };

   public void onGuildMemberJoin(GuildMemberJoinEvent event) {


      Random rand = new Random();
      int number = rand.nextInt(messages.length);

      EmbedBuilder join = new EmbedBuilder();
      join.setColor(0x66d8ff);
      join.setDescription(messages[number].replace("[member]",
event.getMember().getAsMention()));

      event.getGuild().getSystemChannel().sendMessage(join.build()).queue();
   }
}
```

5.4.4 Server Command Interface

(This interface is implemented to get all the required variables such as textchannel : where the command is being called, member : who is calling the commands, arguments : the message through which specific command is triggered and etc.)

```
import net.dv8tion.jda.api.entities.Guild;
import net.dv8tion.jda.api.entities.Member;
import net.dv8tion.jda.api.entities.Message;
import net.dv8tion.jda.api.entities.TextChannel;

public interface ServerCommand {

    public void performCommand(String[] arguments, Guild guild, Member member,
TextChannel textChannel, Message message);
}
```

5.4.5 Command Manager Class

(This class is the most important part of our discord chat bot. This class handles the internal implementations of our commands.
How things really going to work behind the commands being called in the chat is done by the Command Manager.
Command Manager class tells each of the commands when they are being called and calls the methods that are implemented inside each and every commands and hence this class controls all the commands implementations.)

```
public class CommandManager extends ListenerAdapter {

    public final HelpCommand helpCommand;
    public final ClearCommand clearCommand;
    public final KickCommand kickCommand;
    public final BanCommand banCommand;
    public final UnBanCommand unBanCommand;
    public final CooldownCommand cooldownCommand;
    public final MemeCommand memeCommand;
    public final DogCommand dogCommand;
    public final PlayCommand playCommand;
    public final EightBallCommand eightBallCommand;
    public final UserInfoCommand userInfoCommand;

    public CommandManager(){
        this.helpCommand = new HelpCommand();
        this.clearCommand = new ClearCommand();
```

```
        this.kickCommand = new KickCommand();
        this.banCommand = new BanCommand();
        this.unBanCommand = new UnBanCommand();
        this.cooldownCommand = new CooldownCommand();
        this.memeCommand = new MemeCommand();
        this.dogCommand = new DogCommand();
        this.playCommand = new PlayCommand();
        this.eightBallCommand = new EightBallCommand();
        this.userInfoCommand = new UserInfoCommand();
    }

    @Override
    public void onGuildMessageReceived(GuildMessageReceivedEvent event){

        if(!event.getMember().getUser().isBot()){
            String[] arguments = event.getMessage().getContentRaw().split(" ");

            Guild guild = event.getGuild();
            Member member = event.getMember();
            TextChannel textChannel = event.getChannel();
            Message message = event.getMessage();

            switch(arguments[0]){
                case "!help" :
                    helpCommand.performCommand(arguments, guild,member, textChannel,
message);
                    break;
                case "!clear" :
                    clearCommand.performCommand(arguments,guild,member, textChannel,
message);
                    break;
                case "!kick" :
     kickCommand.performCommand(arguments,guild,member,textChannel,message);
                    break;
                case "!ban" :

banCommand.performCommand(arguments,guild,member,textChannel,message);
                    break;
                case "!unban" :

unBanCommand.performCommand(arguments,guild,member,textChannel,message);
                    break;
                case "!cooldown" :

cooldownCommand.performCommand(arguments,guild,member,textChannel,messag
e);
                    break;
                case "!hello" :
                    textChannel.sendMessage("Hello, Greetings for the day! Type !help for
available commands. " + member.getAsMention()).queue();
```

```
                break;
            case "!mention" :
                member = event.getMessage().getMentionedMembers().get(0);
                //message.delete().queue(); To delete the message sent then mentioning
                if (!member.getUser().isBot()) {
                    event.getChannel().sendMessage(("You have been mentioned through
me : " + member.getUser().getAsMention())).queue();
                } else {
                    event.getChannel().sendMessage("Sorry!  I  can't  mention  the
bots!").queue();
                }
                break;
            case "!meme" :

memeCommand.performCommand(arguments,guild,member,textChannel,message);
                break;
            case "!dog" :
                dogCommand.performCommand(arguments, guild, member, textChannel,
message);
                break;
            case "!play" :

playCommand.performCommand(arguments,guild,member,textChannel,message);
                break;
            case "!8ball" :

eightBallCommand.performCommand(arguments,guild,member,textChannel,
message);
                break;
            case "!userinfo" :

userInfoCommand.performCommand(arguments,guild,member,textChannel,
message);
                break;
        }
    }
  }
}
```

5.4.5.1 HelpCommand

(When !help is written on a text channel, the methods of this class gets triggered and execute themselves accordingly)

```
public class HelpCommand implements ServerCommand {
   @Override
   public void performCommand(String[] arguments, Guild guild, Member member, TextChannel textChannel, Message message) {
      EmbedBuilder embedBuilder = new EmbedBuilder();


      embedBuilder.setColor(0xC4E538);
      embedBuilder.setTitle("Some Helpful Commands");
      embedBuilder.setDescription("**----------------------------------------------------------
-----------------------------------------**\n" +
            "\n**!help** - *To display all the usefull commands available*\n" +
            "\n**!userinfo <user mentioned>** - *To display a detailed overview of a user*\n" +
            "\n**!clear <Amount>** - *To delete the specified amount of messages from a text channel*\n" +
            "\n**!kick <user> <reason>** - *To remove a user from the server \n(You must have the admin role to use this!)*\n" +
            "\n**!ban <user> <reason>** - *To Ban a user from the server and they cant join until you unban them \n(You must have the admin role to use this!)*\n" +
            "\n**!unban <userID>** - *To Unban a user from the server so they can join back \n(You must have the admin role to use this!)*\n" +
            "\n**!meme** - *To display a funny meme from reddit*\n" +
            "\n**!dog** - *To display a cute dog picture from reddit*\n" +
            "\n**!cooldown** - *To give a 30 second cooldown to the user*\n" +
            "\n**!8ball <Question>** - *To get the future prediction about the question asked*\n" +
            "\n**!play <link/name>** - *To play the specified song/video in your connected voice channel \n(You must be present in the voice channel to let the bot know about your presence)*\n" +
            "\n**----------------------------------------------------------------------------------
----------------**");
      textChannel.sendMessage(embedBuilder.build()).queue();
```

```
    }
}
```

5.4.5.2 Clear Command

(When !clear <Amount> written is sent on the text channel, the amount of messages specified are clear from the chat, This is done by storing the history amount of messages in list collection and then deleted accordingly)

```
public class ClearCommand implements ServerCommand {
    @Override
    public void performCommand(String[] arguments, Guild guild, Member member, TextChannel textChannel, Message message) {
        /* Clear <count> */
        if(arguments.length < 2){    //not working check
            textChannel.sendMessage("Sorry! This is not the right usage\nPlease use -*!clear <amount>*").queue();
        }
        else{
            try{
                try{
                    message.delete().queue();
                }catch (InsufficientPermissionException exception){
                    textChannel.sendMessage("Sorry but i don't have permissions to delete the messages!").queue();
                }
                message.delete().queue();
                List<Message> messageList = textChannel.getHistory().retrievePast(Integer.parseInt(arguments[1])).complete();
                textChannel.deleteMessages((messageList)).queue();

                textChannel.sendMessage("**Successfully clear "+arguments[1]+" message(s) in "+textChannel.getAsMention()).queue();

            }
```

```
        /* Cant delete more than 100 messages and messages dated more than 2 weeks
from present */
        catch(IllegalArgumentException exception){
            if(exception.toString().startsWith("java.lang.IllegalArguementException:
Message retrieval")){
                textChannel.sendMessage("Sorry but you cant delete more than 100
messages at once").queue();
            }
            else{
                textChannel.sendMessage("Sorry but you cant delete messages that are
older than 2 weeks").queue();
            }
        }
    }
}
```

5.4.5.3 Kick Command

(When !kick <user> <reason> is written on the text channel by the admin the user targeted will be removed from the server with specifying the reason. Note : This command can only be used by the user with admin permission only. It will show error if you don't have Admin permissions)

```
public class KickCommand implements ServerCommand {
    @Override
    public void performCommand(String[] arguments, Guild guild, Member member,
TextChannel textChannel, Message message) {
        /* !kick <user> <reason> */
        if(arguments.length == 3){
            Member target = message.getMentionedMembers().get(0);
            if(target !=null){
                String reason = arguments[2];
                if(reason != null){
                    if(member.hasPermission(Permission.KICK_MEMBERS)){
                        textChannel.sendMessage("The user : "+target.getUser().getName()+ "
was kicked for "+reason).queue();
```

```
                target.kick(reason).queue();
            }
            else{
                textChannel.sendMessage("You don't have permissions for this
command.").queue();
            }
        }
    }
  }else{
      textChannel.sendMessage("Sorry! This is not the right usage\nPlease use -
*!kick <user> <reason>*").queue();
  }
 }
}
```

5.4.5.4 Ban Command

(When !ban <user> <reason> is written on the text channel by the admin the user targeted will be removed from the server with specifying the reason and cannot rejoin the server until the ban is revoked from the user. Note : This command can only be used by the user with admin permission only. It will show error if you don't have Admin permissions)

```
public class BanCommand implements ServerCommand {
 @Override
 public void performCommand(String[] arguments, Guild guild, Member member,
TextChannel textChannel, Message message) {
  /* !ban <user> <reason> */
  if(arguments.length == 3) {
   Member target = message.getMentionedMembers().get(0);
   if(target != null){
    String reason = arguments[2];
    if(reason != null){
     if(member.hasPermission(Permission.BAN_MEMBERS)){
```

```
                textChannel.sendMessage("You have successfully banned
"+target.getUser().getName()+ " from the server for "+reason).queue();
                target.ban(0,reason).queue();
            }
            else{
                textChannel.sendMessage("You don't have permissions for this
command.").queue();
            }
        }
    }else if(arguments.length < 3 || arguments.length >3){
        textChannel.sendMessage("Sorry! This is not the right usage\nPlease use -
*!ban <user> <reason>*").queue();
    }
  }
}
```

5.4.5.5 Unban Command

(When !unban <userId> is specified why the admin the ban on a user will be revoked and the user can join the server again. Note : This command can only be used by the user with admin permission only. It will show error if you don’t have Admin permissions)

```
public class UnBanCommand implements ServerCommand {
  @Override
  public void performCommand(String[] arguments, Guild guild, Member member,
TextChannel textChannel, Message message) {
    /* !unban <userID> */
    if(arguments.length == 2){

      String id = arguments[1];
      if(id!=null){
        guild.unban(id).queue();
```

```
            textChannel.sendMessage("You have successfully unbanned user with ID : "+id).queue();
        }
    }else{
        textChannel.sendMessage("Sorry! This is not the right usage\nPlease use - *!unban <userID>*").queue();

    }
  }
}
```

5.4.5.6 Cooldown Command

(This command is triggered when !cooldown is mentioned by a user. Main motive behind this command is to activate a cooldown process for specific commands so that they cannot be spammed by the users and user will have a countdown i.e. user can re use the command after a specific cooldown period.)

```
public class CooldownCommand implements ServerCommand {

  /*Key is the member */
  public HashMap<Member,Long> cooldown = new HashMap<>();
  /* 30 sec timer */
  @Override
  public void performCommand(String[] arguments, Guild guild, Member member, TextChannel textChannel, Message message) {
    if(cooldown.containsKey(member) && cooldown.get(member) > System.currentTimeMillis()){
        long longRemaining = cooldown.get(member) - System.currentTimeMillis();
        int intremaining = (int)(longRemaining/1000);  // 1 seconds = 1000 ticks
        textChannel.sendMessage("You must wait " + intremaining + " second(s), before executing the command again").queue();
    }else{
        textChannel.sendMessage("Successfully executed the command, you have to for 30 seconds!").queue();
        cooldown.put(member,System.currentTimeMillis()+(30*1000));
```

```
        }
    }
}
```

5.4.5.7 Hello Command

(This command is used to greet the users mentioning the bot and help then to know which command they must use to display the available commands with their functions specified)

```
case "!hello" :
            textChannel.sendMessage("Hello, Greetings for the day! Type !help for available commands. " + member.getAsMention()).queue();
            break;
```

5.4.5.8 Mention Command

(This command's purpose is to mention a targeted user without letting them know who tagged them to avoid conflicts and successfully mentioning the user if they are required at the text channel)

```
case "!mention" :
            member = event.getMessage().getMentionedMembers().get(0);
            //message.delete().queue(); To delete the message sent then mentioning
            if (!member.getUser().isBot()) {
                event.getChannel().sendMessage(("You have been mentioned through me : " + member.getUser().getAsMention())).queue();
            } else {
                event.getChannel().sendMessage("Sorry! I can't mention the bots!").queue();
            }
            break;
```

5.4.5.9 Meme Command

(This command's purpose is entertainment of the users, when users want to see memes and don't have proper sources they can use !meme to trigger this command and through the api connected to the command random funny jokes from reddit get posted in the chat for the user)

```
public class MemeCommand implements ServerCommand {
   @Override
   public void performCommand(String[] arguments, Guild guild, Member member, TextChannel textChannel, Message message) {
      JSONParser parser = new JSONParser();
      String postLink ="";
      String title ="";
      String url ="";

        try {
          URL memeURL = new URL("https://meme-api.herokuapp.com/gimme");
          BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(memeURL.openConnection().getInputStream()));
          String lines;
          while ((lines= bufferedReader.readLine()) != null){
            JSONArray array = new JSONArray();
            array.add(parser.parse(lines));
            for (Object o : array) {
              JSONObject jsonObject = (JSONObject)o;
              postLink = (String)jsonObject.get("postLink");
              title =(String)jsonObject.get("title");
              url = (String)jsonObject.get("url");
            }
          }
          bufferedReader.close();

          EmbedBuilder builder = new EmbedBuilder()
```

```
                .setTitle(title,postLink)
                .setImage(url)
                .setColor(Color.ORANGE);
            textChannel.sendMessage(builder.build()).queue();
        }catch (Exception e ){
            textChannel.sendMessage(":no entry: **Hey, Something went Wrong. Please try again later**").queue();
            e.printStackTrace();
        }
    }
}
```

5.4.5.10 Dog Command

(This command let the user see cute pictures of dogs. To trigger this command !dog is used and random dog picture is posted in the chat for the user to get overwhelmed with their cuteness Note : This is also an alternate of the meme command since when presenting the project meme command is avoided because sometimes unethical memes also gets posted since we are getting the memes from reddit)

```
public class DogCommand implements ServerCommand {
    @Override
    public void performCommand(String[] arguments, Guild guild, Member member, TextChannel textChannel, Message message) {
        JSONParser parser = new JSONParser();
        String postLink ="";
        String title ="";
        String url ="";

        try {
            URL memeURL = new URL("https://random.dog/woof.json");
            BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(memeURL.openConnection().getInputStream()));
            String lines;
```

```
            while ((lines= bufferedReader.readLine()) != null){
               JSONArray array = new JSONArray();
               array.add(parser.parse(lines));
               for (Object o : array) {
                  JSONObject jsonObject = (JSONObject)o;
                  postLink = (String)jsonObject.get("postLink");
                  title =(String)jsonObject.get("title");
                  url = (String)jsonObject.get("url");
               }
            }
            bufferedReader.close();

            EmbedBuilder builder = new EmbedBuilder()
                  .setTitle(title,postLink)
                  .setImage(url)
                  .setColor(Color.CYAN);
            textChannel.sendMessage(builder.build()).queue();
         }catch (Exception e ){
            textChannel.sendMessage(":no entry: **Hey, Something went Wrong.
Please try again later**").queue();
            e.printStackTrace();
         }
   }
}
```

5.4.5.11 Play Command

(This command is used by user to play songs in voice channel so that multiple users can listen to same song and have a good session while sitting at different locations. To use this command !play <link> is used. Note : You must be present in the voice channel so that bot can connect to the same voice channel)

```
public class PlayCommand implements ServerCommand {
```

```
    @Override
    public void performCommand(String[] arguments, Guild guild, Member member,
TextChannel textChannel, Message message) {
        if(arguments.length==2){ //!play <url>
            GuildVoiceState voiceState;
            if((voiceState = member.getVoiceState())!=null){
                VoiceChannel voiceChannel ;
                if((voiceChannel= voiceState.getChannel())!=null){
                    MusicController musicController =
Bot.getAudioManager().getMusicController(voiceChannel.getGuild().getIdLong());
                    AudioPlayer player = musicController.getAudioPlayer();
                    AudioPlayerManager audioPlayerManager =
Bot.getAudioPlayerManager();
                    AudioManager audioManager =
voiceState.getGuild().getAudioManager();
                    audioManager.openAudioConnection(voiceChannel);


                    StringBuilder builder = new StringBuilder();
                    for (int i = 1; i < arguments.length; i++) builder.append(arguments[i]+ "
");
                    String rawLink = builder.toString().trim();
                    if(!rawLink.startsWith("http")){
                        rawLink ="ytsearch: "+rawLink;
                    }
                    final String url = rawLink;


                    audioPlayerManager.loadItem(url, new AudioLoadResultHandler() {
                        @Override
                        public void trackLoaded(AudioTrack audioTrack) {
                            musicController.getAudioPlayer().playTrack(audioTrack);
                            textChannel.sendMessage("Playing : "+ url).queue();
                        }


                        @Override
```

```
                public void playlistLoaded(AudioPlaylist audioPlaylist) {

                }

                @Override
                public void noMatches() {

                }

                @Override
                public void loadFailed(FriendlyException e) {

                }
            });

        }else{
            textChannel.sendMessage("You need to be in a voice channel to execute this command").queue();
        }
    }else{
        textChannel.sendMessage("You need to be in a voice channel to execute this command").queue();
    }
  }
}
```

AUDIO MANAGER CLASS

```
public class AudioManager {
  public HashMap<Long, MusicController> controllerHashMap;

  public AudioManager(){
    this.controllerHashMap = new HashMap<>();
  }
```

```
    public MusicController getMusicController(long guild){
        MusicController musicController = null;
        if (this.controllerHashMap.containsKey(guild)){
            musicController = this.controllerHashMap.get(guild);
        }else{
            musicController = new MusicController(Bot.getJda().getGuildById(guild));
            this.controllerHashMap.put(guild, musicController);
        }
        return musicController;
    }
}
```

MUSIC CONTROLLER CLASS

```
public class MusicController {
    private Guild guild;
    private AudioPlayer audioPlayer;

    public MusicController(Guild guild){
        this.guild = guild;
        this.audioPlayer = Bot.getAudioPlayerManager().createPlayer();
        this.guild.getAudioManager().setSendingHandler(new
MusicCommand(audioPlayer));
        this.audioPlayer.setVolume(15);
    }
    public Guild getGuild(){
        if (this.guild != null)
            return guild;
        return null;
    }
    public AudioPlayer getAudioPlayer(){
        if(this.audioPlayer != null)
            return  audioPlayer;
```

```
        return null;
    }
}
```

5.4.5.12 8ball Command

(This command is used to know the horoscope/future of the questions you have asked from the bot. When !8ball <question> is send on the text channel bot will randomly select a string from multiple strings and decide your fortune)

```
public class EightBallCommand implements ServerCommand {


    @Override
    public void performCommand(String[] arguments, Guild guild, Member member, TextChannel textChannel, Message message) {
        if(arguments.length <2){
            textChannel.sendMessage("Please mention the question as well!").queue();
        }
        else if(arguments[0].equalsIgnoreCase("!8ball")){
            EmbedBuilder embedBuilder = new EmbedBuilder();
            embedBuilder.setTitle("Your Horoscope");


            embedBuilder.setColor(0x2ecc71);
            embedBuilder.setThumbnail("https://cafeastrology.com/wp-content/plugins/magic-answers/images/ball.png");
            embedBuilder.setDescription(randomString());
            embedBuilder.setFooter("Hope this prediction will help you :)");
            textChannel.sendMessage(embedBuilder.build()).queue();
        }
    }
    public static String randomString(){
        Random r = new Random();


        int choice = 1 + r.nextInt(15);
```

```
String response = "";
if ( choice == 1 )
   response = "It is certain";
else if ( choice == 2 )
   response = "It is decidedly so";
else if ( choice == 3 )
   response = "Without a doubt";
else if ( choice == 4 )
   response = "Yes - definitely";
else if ( choice == 5 )
   response = "You may rely on it";
else if ( choice == 6 )
   response = "As I see it, yes";
else if ( choice == 7 )
   response = "Most likely";
else if ( choice == 8 )
   response = "Outlook good";
else if ( choice == 9 )
   response = "Signs point to yes";
else if ( choice == 10 )
   response = "Yes";
else if ( choice == 11 )
   response = "Reply hazy, try again";
else if ( choice == 12 )
   response = "Ask again later";
else if ( choice == 13 )
   response = "Better not tell you now";
else if ( choice == 14 )
   response = "Cannot predict now";
else if ( choice == 15 )
   response = "Concentrate and ask again";
else
```

```
            response = "8-BALL ERROR!";


        return  "MAGIC 8-BALL SAYS: " + response;
    }
}
```

5.4.5.13 User Info Command

(This command is used to get the user info of the targeted user such as when did they joined the server, when they made the account , is the user a bot or not, what roles they have and etc. and presented in a embedded message)

```
public class UserInfoCommand implements ServerCommand {
    @Override
    public void performCommand(String[] arguments, Guild guild, Member member, TextChannel textChannel, Message message) {
        if (arguments.length==2){
            Member target =message.getMentionedMembers().get(0);
            DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
            if(target!=null){
                EmbedBuilder embedBuilder = new EmbedBuilder();
                embedBuilder.setColor(Color.ORANGE);
                embedBuilder.setTitle("User - "+target.getUser().getName());
                embedBuilder.setThumbnail(target.getUser().getAvatarUrl());
                embedBuilder.addField("Time Joined ",target.getTimeJoined().format(formatter),true);
                embedBuilder.addField("Time Created ",target.getTimeCreated().format(formatter),true);
                embedBuilder.addField("Discord Bot" , String.valueOf(checkForBot(target)),true);
                embedBuilder.addField("User Roles", getRoles(target.getRoles()),true);
                textChannel.sendMessage(embedBuilder.build()).queue();
            }
        }
```

```
    }
    private boolean checkForBot(Member member){
        if (member.getUser().isBot()){
            return true;
        }
        return false;
    }
    public String getRoles(List rolesList){
        String roles = null;
        if(!rolesList.isEmpty()){
            Role tempRole = (Role) rolesList.get(0);
            roles = tempRole.getName();
            for (int i = 1; i < rolesList.size(); i++) {
                tempRole = (Role)rolesList.get(i);
                roles = roles+", "+tempRole.getName();
            }
        }else{
            roles = "No Roles";
        }
        return roles;
    }
}
```

CHAPTER 6

RESULTS

6.1 Help Command

Help command describes all the command available to the user and its function.

Syntax : *!help*

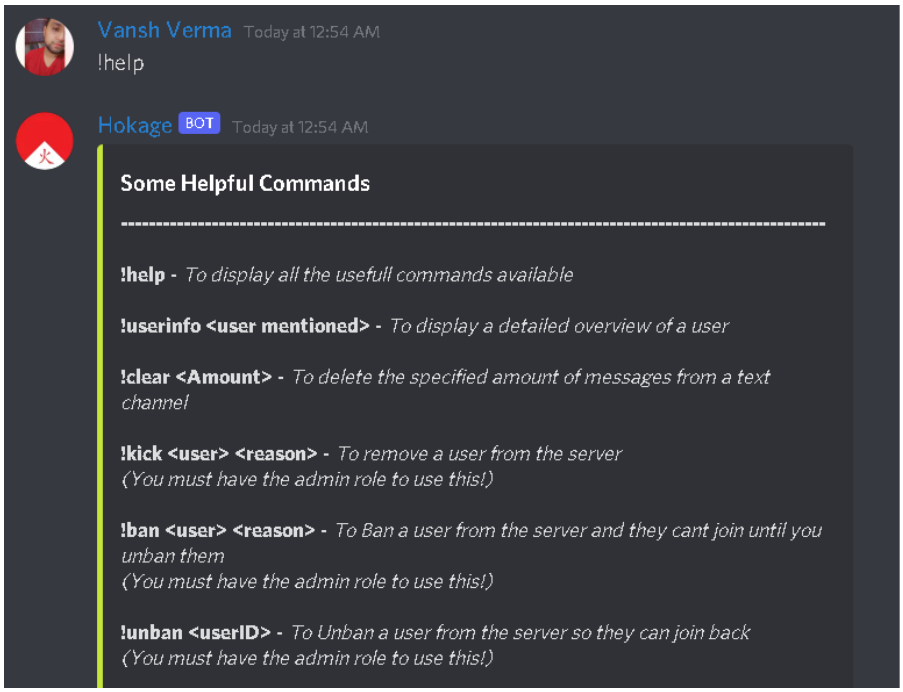

Fig 6.1 Help command-1

Fig 6.2 Help command-2

6.2 UserInfo Command

UserInfo command shows all the details about the mentioned user like when user has joined the server, what are his/her roles etc.

Syntax : *!userinfo <mention user>*

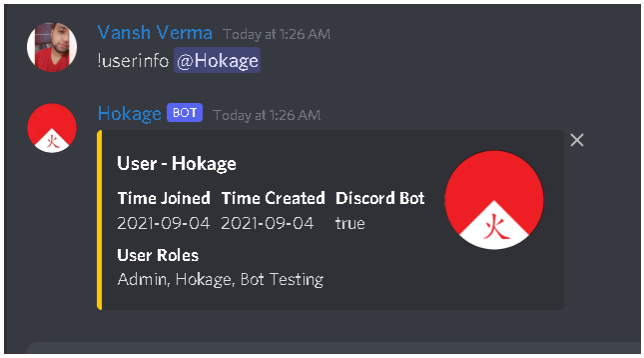

Fig 6.3 UserInfo command-1

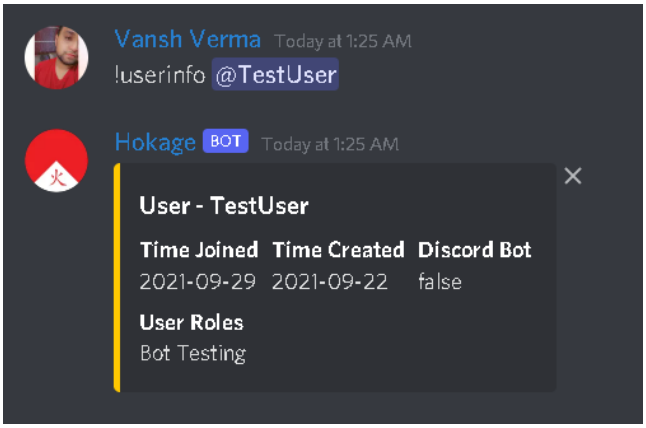

Fig 6.4 UserInfo command-2

6.3 Unban Command

Unban command is used to unban an user from the server.

Syntax: !unban <userID>

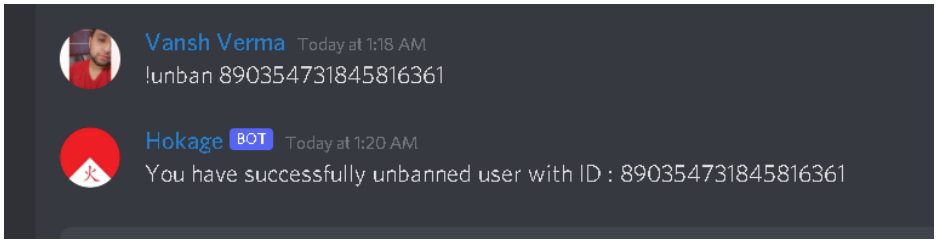

Fig 6.5 Unban command-1

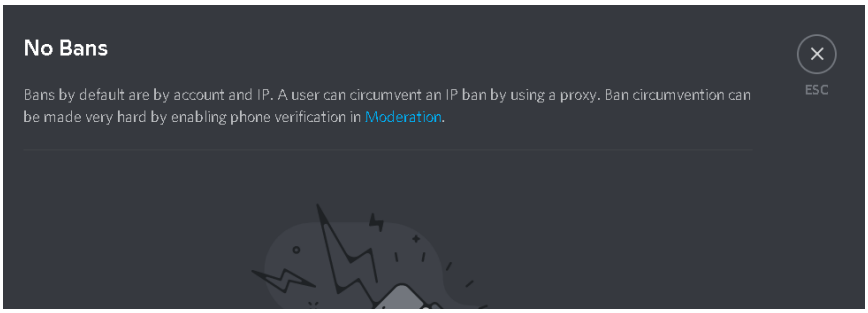

Fig 6.6 Unban command-2

6.4 Ban Command

As the name suggest ban command is used to ban an user from the server. Keep in mind that not everyone can ban anyone. You must have access first to doing ban someone.

Syntax: *!unban<mention user> <reason>*

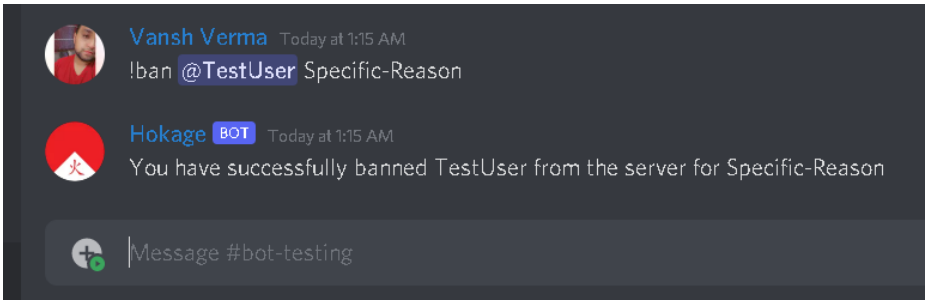

Fig 6.7 Ban command-1

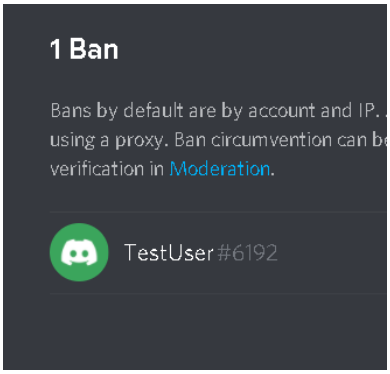

Fig 6.8 Ban command-2

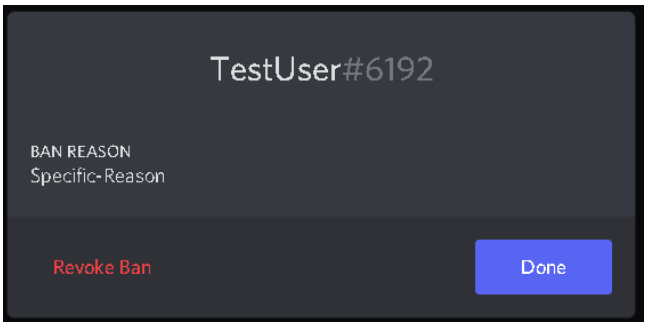

Fig 6.9 Ban command-3

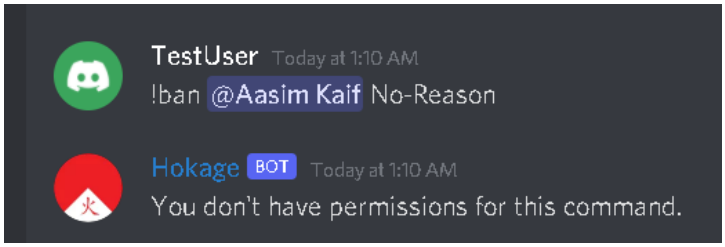

Fig 6.10 Ban command-4

6.5 Kick Command

Kick command is used to remove someone from the server.

Syntax: !kick <mention user> <reason>

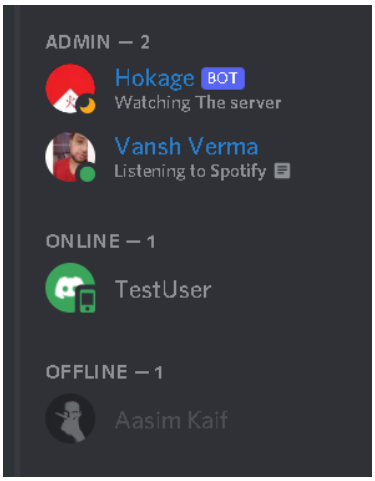

Fig 6.11 Kick command-1

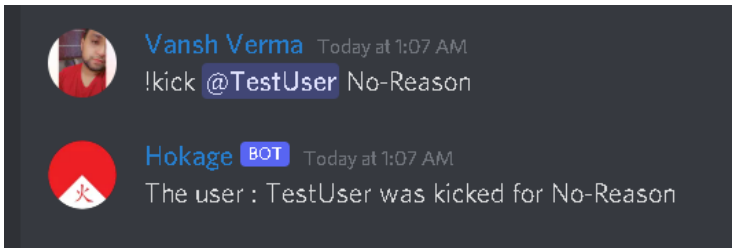

Fig 6.12 Kick command-2

Fig 6.13 Kick command-3

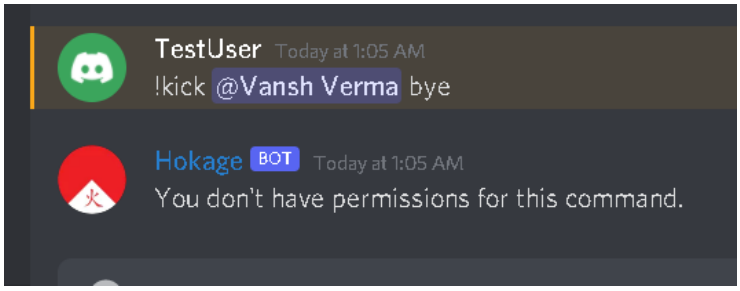

Fig 6.14 Kick command-4

6.6 Meme Command

Meme command can be used to share meme on the channel. It's just something we need to lighten up the server environment.

Syntax: *!meme*

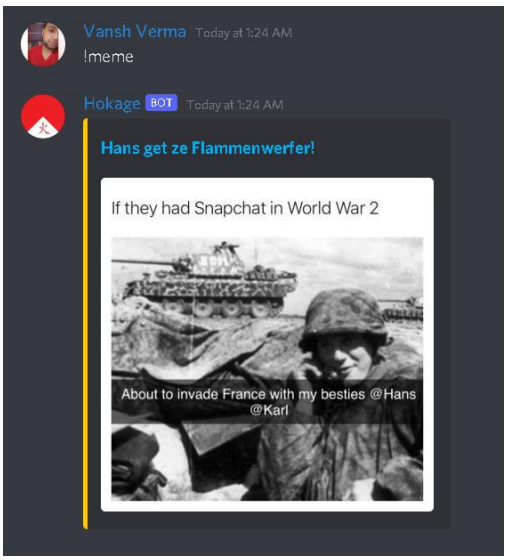

Fig 6.15 Meme command

6.7 Dog Command

Dog command is used to share cute dogs picture on the server,

Syntax: *!dog*

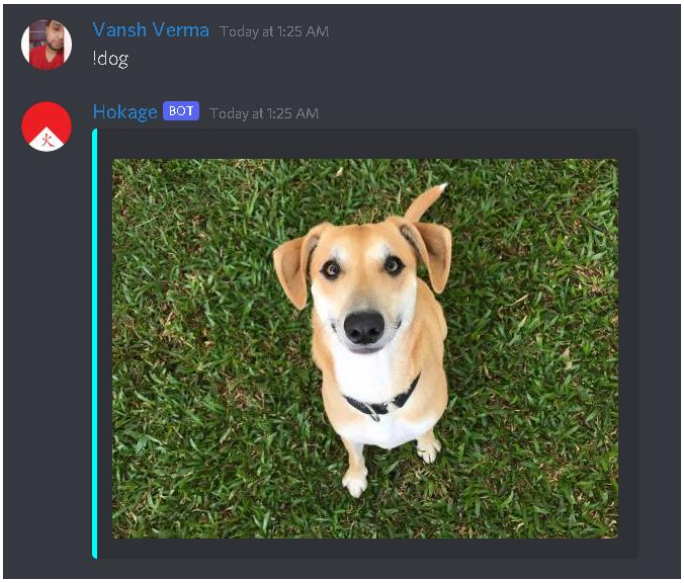

Fig 6.16 Dog command

6.8 Cooldown Command

Cooldown command to give cooldown to user for 30 seconds. After using this command user will not be able to execute any command for 30 seconds.

Syntax: !cooldown

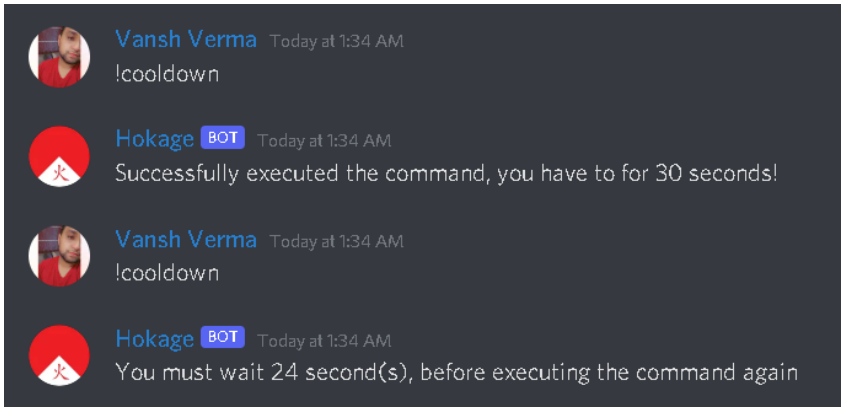

Fig 6.17 Cooldown command

6.9 8Ball Command

To get the future prediction about the question asked.

Syntax: *!8ball <Question>*

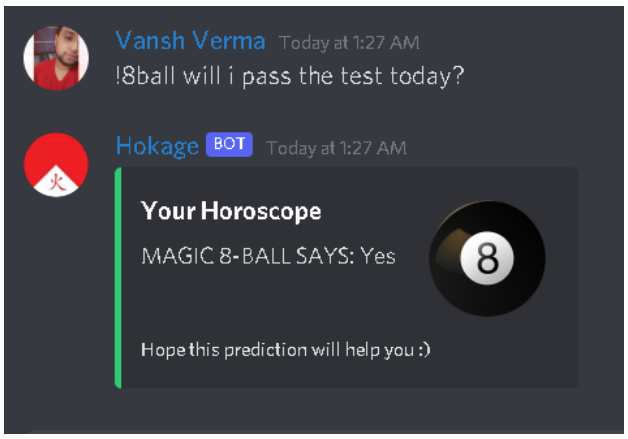

Fig 6.18 8Ball command-1

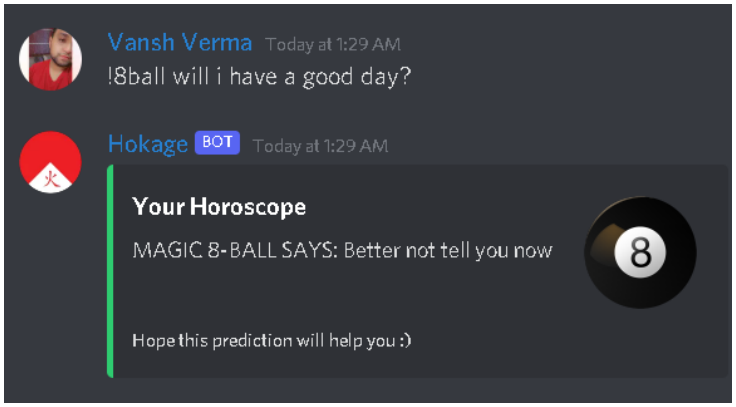

Fig 6.19 8Ball command-2

6.10 Play Music Command

Play command is used to play music on any voice channel. Play command takes a YouTube link and play a audio on the voice channel.

Syntax: *!play <YouTube link>*

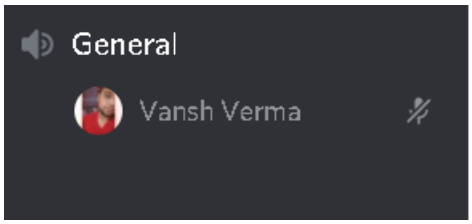

Fig 6.20 Play command-1

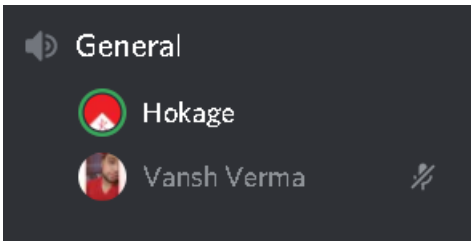

Fig 6.21 Play command-2

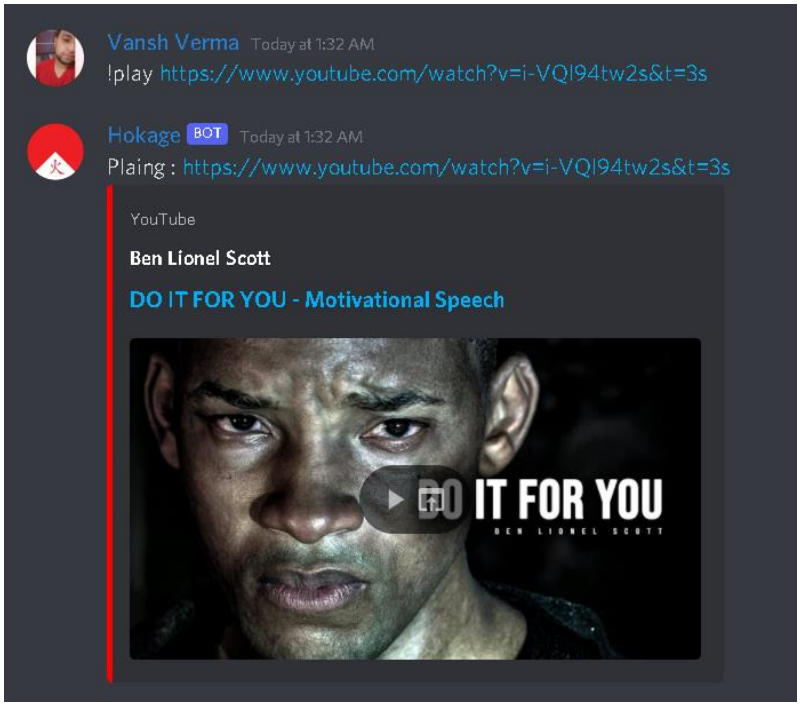

Fig 6.22 Play command-3

6.11 Clear Command

Clear command is used to clear any amount of messages mentioned by the user.

Syntax : *!clear <Amount>*

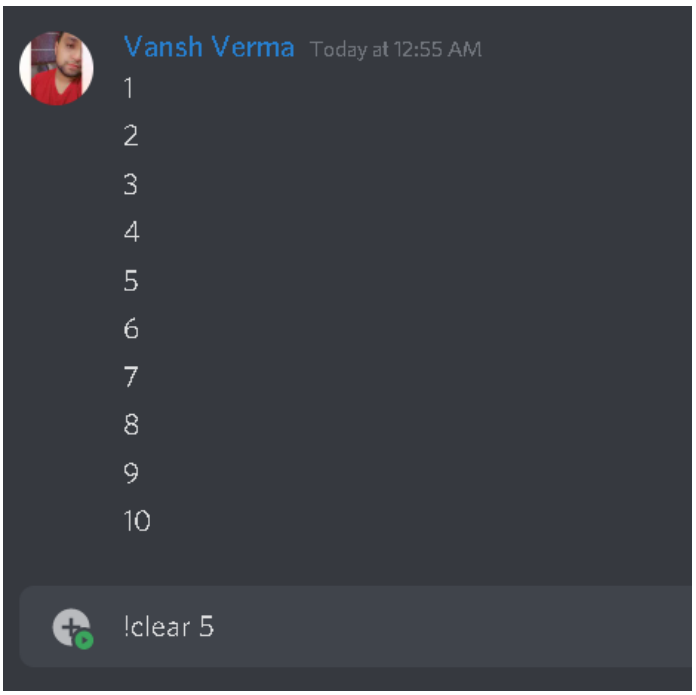

Fig 6.23 Clear command-1

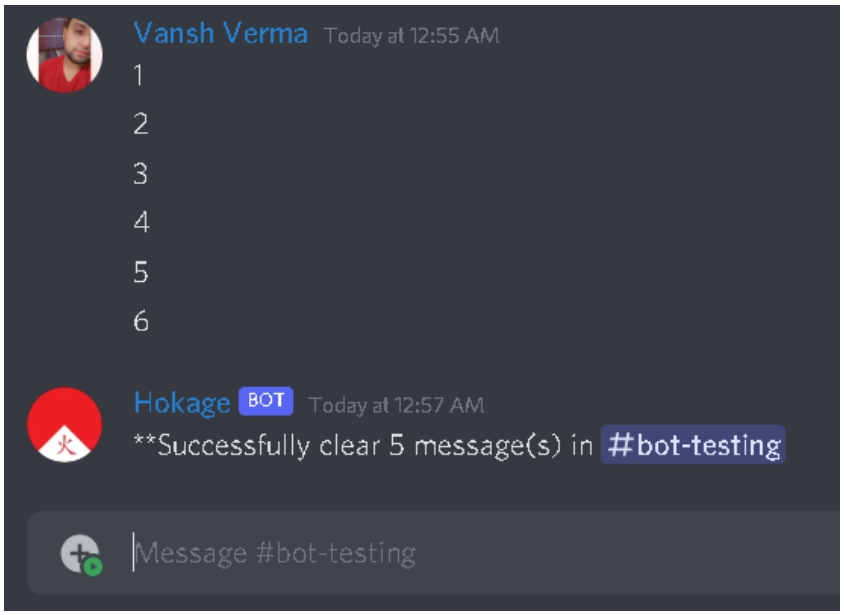

Fig 6.24 Clear command-2

6.12 Hello command

Greets the user and navigates them to the path of knowing all available commands.

Syntax : *!hello*

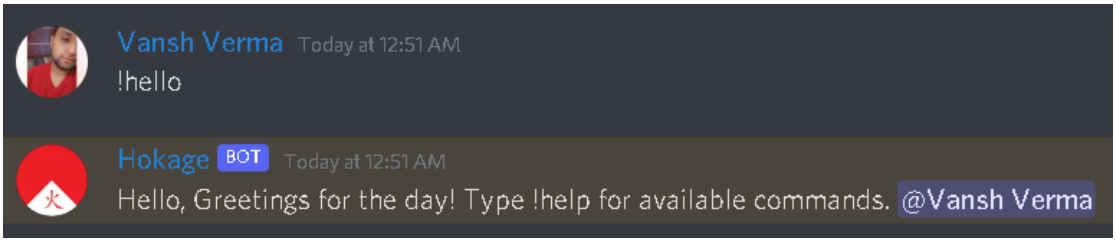

Fig 6.25 hello command

6.13 Mention Command

Mentions the targeted user without letting them know who tagged them.

Syntax : *!mention <userMentioned>*

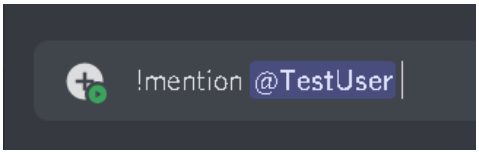

Fig 6.26 Mention command -1

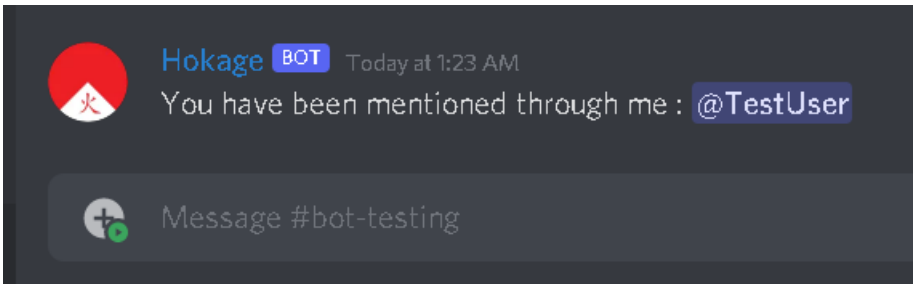

Fig 6.27 Mention command – 2

CHAPTER 7

FUTURE SCOPE & CONCLUSION

7.1 Future Scope

Our discord bot also known as the “Hokage” is a universal tool that can be used by anyone that uses discord as a platform. It allows people to fetch data from youtube to play music and other sites, provide them with information, entertainment and better accessibility. The bot can be deployed on any server and can setup itself as long as the admin gives it enough rights. The channels it sets up are all optional and can be selected or deselect as per choice. There are no restrictions on the number of servers it can join or on the number of people it can respond to. It can further in future be enhance for more activities and responses thereby allowing better personalisation for each individual server.

The discord bots do not interfere with the regular working of a server and are secure. They increases interactivity and provide a better experience for all the users.Updates can be constant and will reflect on all the servers no heavy maintenance is required hence making it very robust. The discord bots can change the whole platform and can bring several enhancements while offering ease of use. They are viable and easy to maintain.

This bot is unique in purpose and has several functions as well. The bot provides us with functions such as viewing memes, jokes, images and polls.It also does not read any personal information from the users present in the server and only gets activated when the trigger command is used. The bot also has some witty responses for the users just to make the platform more friendly and light hearted. It allows the users to manually select the channels and information they want. The “Hokage” is deployable on any server and helps set itself up, moreover the bot doesn’t require much hardware and is self sufficient for the most part

7.2 Conclusion

When testing the last prototype we got findings suggesting that the participants did not have a problem with getting information from a chatbot instead of a human. The information that they got was not seen as less trustworthy, this could be supported by the fact that the chatbot provided a source for the information it gave. It has been interesting to investigate how the participants interacted with the chatbot and how they reported on it afterwards. Our findings have some indicators leading towards that a chatbot could be a good alternative for acting as a helpful friend for freshmans at a new school. Still we have to stress the fact that the chatbot was not very intelligent and that the evaluators had to adjust their language to match the chatbots. 11 Because of the scope of the project we did not have time to conduct as much user testing and re-design to the chatbot as we would have liked. This has an impact on the validity of our research. Through the project we have touched on some theory when making the chatbot, but this should also have a larger focus for higher validity. Even though the participants trusted the information given in this project we cannot say that people trusts a chatbot as much as they trust a human being. There are also biases in our project, one of them is that all the students that we included in the project already knew a lot of the answer the prototype could provide. Another bias is that the information the chatbot provides could be seen as “casual” and are not crucial and/or vital This could have had an impact on the results regarding trustworthiness. With that being said we also think that some of our findings could give some insights into how a very small group of people think about using a chatbot to gain information in a school context. Some of the characteristics of our chatbot was viewed as appropriate for the given context, like “casualness” and links to where the information was gathered. If the IFI chatbot is to be furthered developed, this could be something to draw upon.

CHAPTER 8

REFERENCES

- https://github.com/DV8FromTheWorld/JDA (API)
- https://www.udemy.com/course/create-your-own-discord-bots-using-java/
- https://www.geeksforgeeks.org
- https://www.javatpoint.com
- https://www.youtube.com