



**FINAL SEMESTER ASSESSMENT (FSA)
B.TECH. (CSE)
VI SEMESTER**

**UE18CS355 – OBJECT ORIENTED ANALYSIS AND DESIGN
WITH SOFTWARE ENGINEERING LABORATORY**

**PROJECT REPORT
ON**

Discord bot for CTFd

SUBMITTED BY

NAME

SRN

- | | |
|------------------|---------------|
| 1) Ajitesh Nair | PES2201800681 |
| 2) Shashank G S | PES2201800706 |
| 3) Aayush Kapoor | PES2201800211 |

**JANUARY – MAY 2021
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
EC CAMPUS,
BENGALURU – 560100, KARNATAKA, INDIA**

TABLE OF CONTENTS		
Sl.No	TOPIC	PAGE No
1.	ABSTRACT	2
2.	SOFTWARE REQUIREMENTS SPECIFICATION	2
3.	PROJECT PLAN	15
4.	DESIGN DIAGRAMS	21
5.	MODULE DESCRIPTION	26
6.	TEST CASES	28
7.	SCREENSHOTS OF OUTPUT	35

1. ABSTRACT :

The purpose of the developed CTFd bot is to connect Discord server to CTFd framework with a centralized system for real-time management of the competition by the organizers. CTFd bot can monitor real-time solves, submissions and filter out valid ones, sends notifications on each solves with beautiful and colorful embeds with current rank and score of the person who solved with a timestamp in it. We have a live scoreboard that notifies the participants and the organizers when there is any change in the ranks.

The benefit of this bot is that it becomes much easier for the organizers to perform all the most performed tasks with one command on Discord. This would reduce the need to move to the CTFd web app. This would also help the participants to know where they stand without having to go to the website

This product is a follow on to the CTFd platform. CTFd platform is a web application to help organizers to easily manage and conduct Capture the Flag events. With the existing CTFd framework, it is a paid web app, or it can be freely installed and hosted on our own servers. We plan to make use of the Discord API to combine it with the CTFd web application so that every task that an organizer and a participant would do for a CTF, can now directly be done as a command to that bot on Discord.

2. SOFTWARE REQUIREMENTS SPECIFICATION :

Introduction

Purpose

This document aims to give a brief description about the **Eye of Providence- A discord bot for CTFd, v1.0**. This SRS document describes the entire functions of the bot that both the participants and the organizers can use to effectively take part in or conduct CTF events. With the help of this document, the needs of the company and the solution that will be provided to these needs will be clearly presented. In other words, this document will provide a basis for validation and verification.

Intended Audience

This document will be of interest to the following audience:

- **Developers** – Those building the application need to have access to the document to see what they are building and to have a clear understanding of the product.
- **Project Managers** – Managers need to refer to this document to analyze the requirements of the application, the features and the course of action involved in building this application.
- **Testers** – Testers will be required to write unit tests and various other tests to verify that this application can handle all (un-intended as well) use cases.
- **Open-Source contributors** – As the source code of this project will be live on

GitHub, anyone who needs to contribute to this project i.e., add additional features, improve on the existing features, improve the efficiency, fix security issues and so on can refer to this document.

- **Organizers of CTF competitions** who wish to get a deeper understanding of how this bot works.

These readers will get a clear understanding of the basic functionalities or the framework and can explore further expansion possibilities onto the existing model on reading this SRS document.

The rest of this document contains more information about the Discord Bot like the product scope, references used to build it, user classes, constraints, software, and hardware interfaces needed, and the communication specifications.

Product Scope

The purpose of the developed CTFd bot is to connect Discord server to CTFd framework with a centralized system for real-time management of the competition by the organizers. CTFd bot can monitor real-time solves, submissions and filter out valid ones, sends notifications on each solves with beautiful and colorful embeds with current rank and score of the person who solved with a timestamp in it. We have a live scoreboard that notifies the participants and the organizers when there is any change in the ranks.

The benefit of this bot is that it becomes much easier for the organizers to perform all the most performed tasks with one command on Discord. This would reduce the need to move to the CTF web app. This would also help the participants to know where they stand without having to go to the website.

References

The following websites refer to some of the terms and conditions of the tools and software that we will be using to build this bot.

Discord API Documentation - [Discord Bots | API \(top.gg\)](#)

Terms of Service, Discord - [Terms of Service | Discord](#)

CTFd framework - [CTFd : The Easiest Capture The Flag Platform](#)

CTFd Terms of Use - [CTFd : Terms of Use](#)

Overall Description

Product Perspective

This product is a follow on to the CTFd platform. CTFd platform is a web application to help organizers to easily manage and conduct Capture the Flag events. With the existing CTFd framework, it is a paid web app, or it can be freely installed and hosted on our own servers. We plan to make use of the Discord API to combine it with the CTFd web application so that every task that an organizer and a participant would do for a CTF, can now directly be done as a command to that bot on Discord.

Product Functions

This Discord bot has two classes of users that can access it with commands to this bot and a set of functions that it performs in the background.

Tasks that the bot will do in the background, automatically:

- Manage authentication of participants and organizers.
- Take backups every hour.
- Display first bloods – the first person to solve a challenge.
- Notify when a challenge is created.
- Notify when a challenge is deleted.
- Create and assign channels for team chat and provide roles.
- Monitor participant activity.

Tasks that an organizer can perform with this bot:

- View the help/manual page.
- See list of challenge categories.
- See list of challenges under a category.
- Can add a challenge.
- Can delete a challenge.
- Can add a new category.
- Remove a category and all challenges associated with it.
- Can ping a problem author to notify any error in the challenge.
- Can view the status of a particular challenge – see number of solves and other such statistics.
- Can view the flag for a challenge.

Tasks that a participant can perform with this bot:

- Can Register for the event.
- Can create a new team.
- Can join a team.
- View challenge list.
- View the scoreboard.
- View challenges solved.

User Classes and Characteristics

The types of users of this bot are broadly classified into two classes i.e., **organizers** and **participants** of the Capture the Flag event. To perform any task with this bot, they first must be authenticated and registered with this bot on the Discord server. This bot can differentiate between the classes of users based on the roles system used by Discord. Each role has a different set of permissions. These users and the roles are described in detail below.

- **Organizers** – These classes of users have full access to the bot. These are the users who are conducting the competition.
- **Participants** – These class of users have partial access to the bot. These users are taking part in the CTF event as participants. They do not have access to the administrative commands of the bot.

Operating Environment

The bot is specifically designed to operate on the discord software environment, the functionality of this bot can be hosted/deployed on various platforms like **Heroku**, **Repl.it** etc. The model is capable of operating under all OS's, Linux (Debian) being an example of one such OS in which it is very frequently used. This works based on the Discord API and accordingly can work on any machine which supports Discord.

Design and Implementation Constraints

CTFd is a paid service. To get access for free, it must be self-hosted for free on Heroku or AWS or any other such cloud providers. These services offer a free tier having a limited amount of storage, processing power and memory. If we need a higher amount of any of these, we will have to pay. This is a freemium model i.e., it can be used for free with limitations, but for full access the user will have to pay a small amount.

Assumptions and Dependencies

We have made the assumption that the Discord API would continue to support its working with the CTFd framework. Any updates to the API should not affect the working of this bot.

External Interface Requirements

User Interfaces

The user interface design is intended to serve the needs of important entities of the CTF event for the users and the moderators/organizers.

The interface provided for the event participants is a command driven interaction environment, in which the users(participants) can access the various functionalities of the bot using the specified commands for the respective functionalities. The command based interface is available to the moderators/organizers as well. The commands are typed as messages on the Discord server chat with the bot.

The moderators/organizers are provided with an additional tkinter based graphic user interface to monitor the activities and progression of the contestants. This is in addition to the existing web application provided by the CTFd framework and the message based communication with the bot.

Software Interfaces

The software consists of the following elements:

1. Web socket API by Discord (<https://discord.com/developers/docs/intro>)
2. Repl.it's built in Database (<https://blog.repl.it/database>)
3. Python3 (version greater than 3.7)
4. Heroku/Repl.it (The latest build for hosting the bot and the

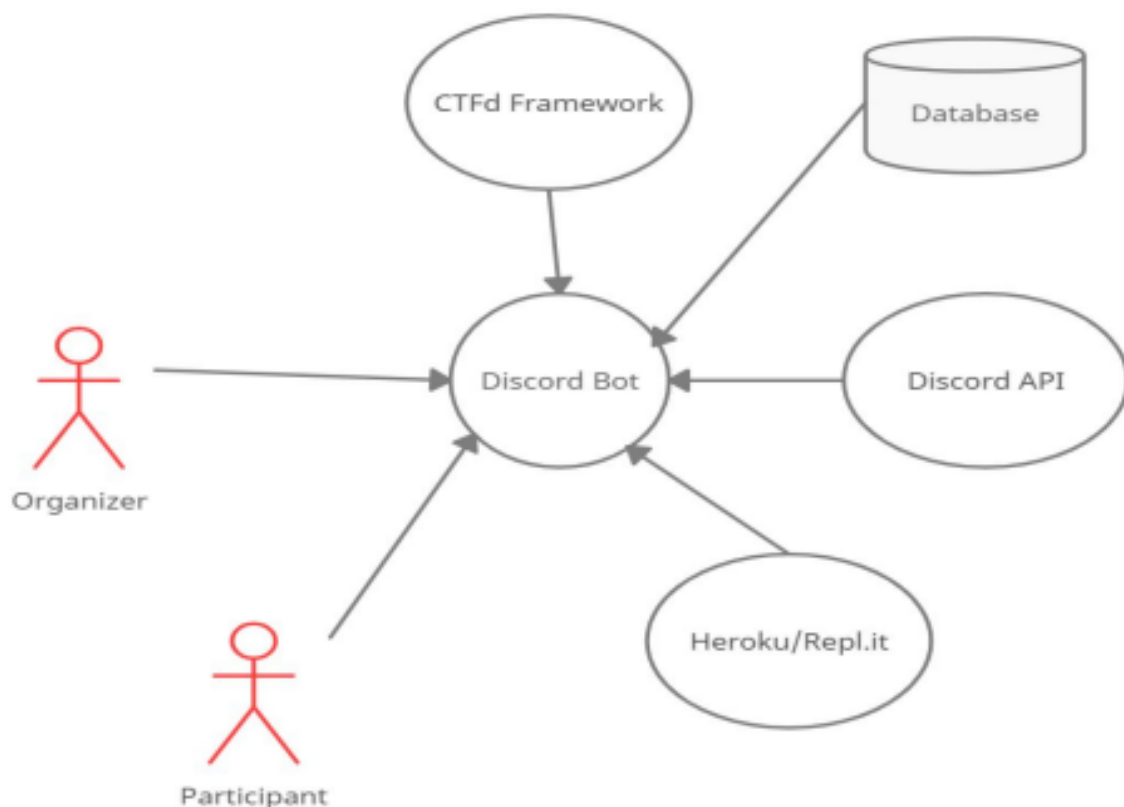
CTFd framework)

Communications Interfaces

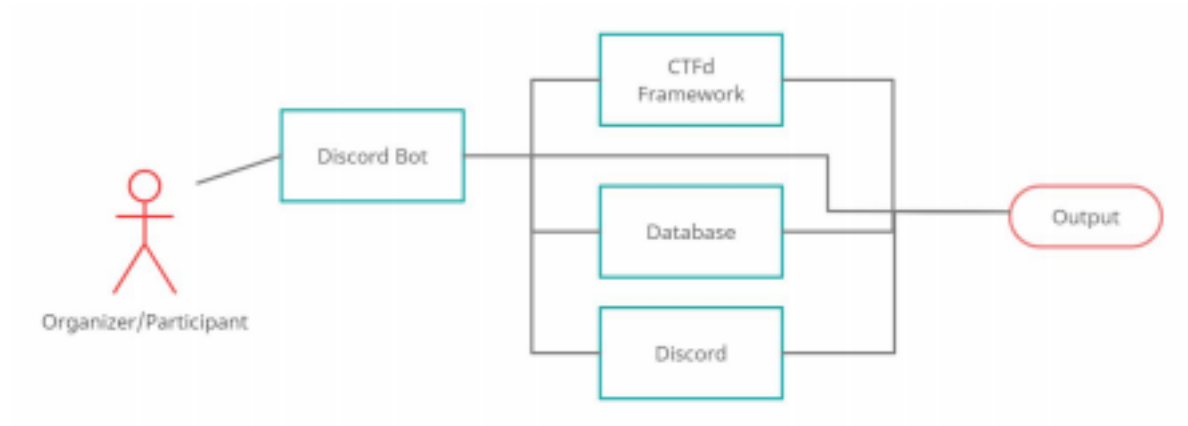
Discord's API is based around two core layers, a HTTPS/REST API for general operations, and persistent secure WebSocket based connection for sending and subscribing to real-time events. The most common use case of the Discord API will be providing a service, or access to a platform through the OAuth2 API.

Analysis Models

The context diagram for a Discord bot based on the CTFd framework is shown below. It interacts with the participant, organizer, CTFd framework, Discord API, Hosting Provider and the database.



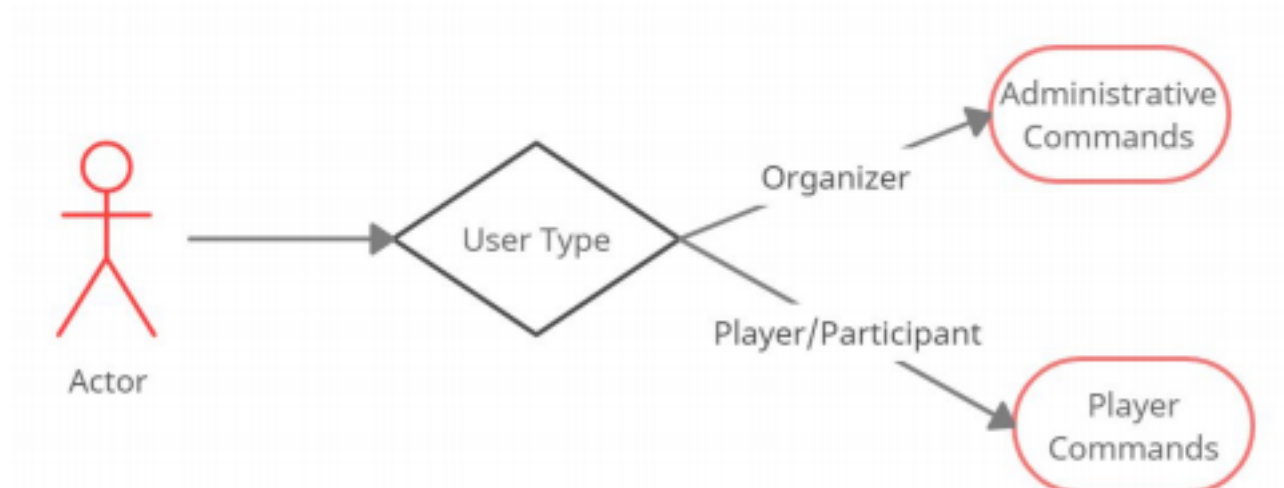
The users i.e. the participants and organizers interact with the Bot through Discord, which intern works on the CTFd framework, the database and the Discord API to provide us with the output.



Use case diagram for the bot is shown below.



Each class of users have their own commands. This is shown in the following image.



System Features

This section describes the most important features of the Discord bot in detail. These features include registration of participants, first-blood announcements, backups and commands.

Registration of Participants

Description and Priority

This feature involves registration of the participants and the creation of teams for the CTF event. Will involve a username (which could be their Discord username as well), email address and a password. This will help to keep track of all the participants and teams participating in the event.

Priority: 10

This has a high priority of ten because this is the most important aspect with regards to the competition. To register to be a part of the competition, this step is mandatory.

Stimulus/Response Sequences

Once a user joins the Discord server of the event, we will ask him/her to register for the event through the bot. As mentioned earlier, the interaction with the bot will be through commands/messages. The user would have to give the register command, and then the bot responds with a series of questions asking for the username, email ID, and a password.

After an account has been created, the next step is to join a team or to create a team. This will also be done by answering questions asked by the bot.

Functional Requirements

REQ-1: Selenium

REQ-2: bs4

First Blood Announcement

Description and Priority

First bloods are a common term in CTF competitions. They are given to the first person to solve a challenge. They usually come with a few benefits like additional points to the team/individual. The bot will scan through all the challenges at regular intervals and find the first solver for each challenge. After the list is found, an announcement is made and also points are added to a tally which will be added to the final points at the end of the CTF.

Priority: 7

It is a common trend/culture to have first bloods in tournaments as it encourages the participants to try and think of ways to solve a challenge faster thereby having more engagement in the competition.

Stimulus/Response Sequences

As a person solves the challenges, they are added to the list where organisers can see the people who have solved a particular challenge. The bot will check this list for each challenge at regular intervals to find first bloods. Upon finding one, the announcement is made on the discord server's channel and a score(for first blood) is updated. Once all challenges have first blood, this process is stopped as it won't be required until newer challenges are added.

Functional Requirements

REQ 1 - Python modules - selenium, discord.py, bs4

Automated Backups

Description and Priority

The bot will make backups of the CTFd instance at regular intervals. By default this is set to once every hour. These backups will be mailed to the organisers via a Gmail account assigned to the bot. There will be an option to either have every backup sent out or just the latest one right before a CTF server crash.

Priority:9

Stimulus/Response Sequences

As soon as the CTF event is started, backups will start to take place. As soon as the CTF website crashes, the organisers will be notified. Based on the severity of

the crash, if the website cannot be restored, a rebuild will be necessary. For this, the old instance is replaced with a new one and the backups are uploaded to the new instance which resumes where the previous one left off. So a downtime of anywhere between 10 - 20 mins is expected for the rebuild to finish.

Functional Requirements

REQ 1 - Gmail account with 'Enable less secure apps' feature turned on.

REQ 2 - Python modules for sending email - smtplib, email, selenium, base64, datetime, bs4

REQ 3 - Heroku/Repl.it - To be able to store and send files.

Commands For Challenges

Description and Priority

The bot allows organisers and players to run commands from the discord server for challenges present in the CTF. An organiser will have commands to add, delete, and change challenges. Whereas the player will have the option to list the challenges and their categories along with a few more player oriented functionality.

Priority: 8

Stimulus/Response Sequences

Once a command is entered, the first step is to check who is running the command. If it's a user, all permissions are granted. Players will have a subset of commands to run. Once this is done, the bot will process the command and take the required actions against the challenges. The output is printed back into the same channel from which the command was given. In case of errors, based on the type of error, it is either printed on the channel or in the logs for organisers to go through.

Functional Requirements

REQ 1 - Python modules - Discord.py, selenium, bs4, async

Other Nonfunctional Requirements

Performance Requirements

The system is meant to be interactive and likewise the delays (if any) involved must be at a minimum, but if the load on the server is high, some delays are acceptable as we are providing a freemium model. This application does not have to be real-time though real-time is ideal.

Safety Requirements

With great power comes great responsibility.

There's a huge responsibility in being a place where millions of people talk to each other every day, and we want Discord to always be a welcoming place for you and your friends. We do not sell your data, nor do we share it with third parties for advertising purposes.

We take public safety very seriously.

We put your privacy first and as a rule do not scan or read your messages. However, we can do so for safety reasons. If an issue arises, we will investigate, review the situation, and take proper action. We know there are cases where bad actors are acting to cause violence or harm in the real world, and we make it clear to them that they don't have a place on Discord.

We give you the tools to control your experience.

We've created tools for server owners to moderate their spaces. These tools allow you to enforce your rules and norms for your unique community.

Security Requirements

Your identity is yours.

Your account doesn't have to be tied back to your real world identity — we want to make sure that everyone can be their true self and feel comfortable when they talk on Discord. But that doesn't mean that someone can avoid consequences for their actions.

Software Quality Attributes

Adaptability: The bot is highly adaptable because it is built on the Discord API and CTFd framework, under the assumption that the API will continue to support bots built on the older versions. If not, we will have to update the bot in a timely manner to keep up with the latest frameworks and APIs.

Availability: The bot deployment happens on the discord server, and the server and functionalities are active 24/7 to all the users in need, therefore ensuring availability assurance to everyone in need.

Interoperability: Multiple users can operate on the bot from different machines, which could possibly be running on different operating systems and having different processing, computational powers.

Portability: This bot can run on a variety of end devices.

Correctness: The bot is guaranteed to be correct. The bot retrieves all data from the CTFd framework, which is a tried and tested tool used by most CTF competitions around the world.

Reusability: This bot is highly reusable, by making small changes to the configuration, it can be added to any Discord server.

Reliable: The fairly straightforward functionalities simplify user tasks and yields desired results at all times, making this a fairly reliable product.

Business Rules

Organizers:

The organizers can perform organizational tasks like creating/deleting challenges, categories, view the help/manual page, see list of challenge categories, see list of challenges under a category, can add a challenge, can delete a challenge, can add a new category, remove a category and all challenges associated with it, can ping a problem author to notify any error in the challenge, can view the status of a particular challenge – see number of solves and other such statistics and can view the flag for a challenge.

Contestants:

A participant can perform with this bot, view the challenge list, view the scoreboard and view the challenges solved by him/her.

A domain name would be needed for the bot if we would like to advertise it. For now, the bot can be published on Discord without having a domain.

Appendix A: Glossary

Discord: The application where the bot is launched and maintained.

CTF- (Capture The Flag) is a kind of information security competition that challenges contestants to solve a variety of tasks ranging from a scavenger hunt on wikipedia to basic programming exercises, to hacking your way into a server to steal data.

CTFd - It is a framework designed for the ease of use for both administrators and users(participants). Each of them can perform their specific tasks with a significant degree of ease.

Heroku: Heroku is a cloud platform as a service (PaaS) that lets companies build, deliver, monitor, and scale apps. Heroku bypasses infrastructure headaches.
Software Requirements Specification for a CTF Discord Bot Page 12

Repl.it: Repl.it allows users to write code and build apps and websites using a browser. Additionally, Repl.it allows users to share projects.

Organizer: The people monitoring the challenges posed to the contestants and also the categories that they fall into.

Participant/Contestants: The participants are the ones who attempt to conquer the challenges and aspire for validation of their results.

Appendix B: Field Layouts

Information required to register the customer

Field	Requirements	Data Type	Description	Is Mandatory
Username	max 15 characters	string	contestant name	Y
Email	must contain @	Varchar	contestant email	Y
Password	min 12 characters	Varchar	password	Y

Sample Report Requirements: Include the fields to be included in the report

Registration Report	Transaction Report
Team Name	Team Name
Team Lead	Rank
Team Members	Won/Lost
Email	Team Members
	Problems Solved
	Percentile Cash prize(for specific positions, null if not applicable)

Appendix C: Requirement Traceability Matrix

Sl. No	Requirement ID	Brief Description of Requirement	Architecture Reference	Design Reference	Code File Reference	Test Case ID	System Test Case ID
01.	1	Bot creation and it's functionality	V architecture	Sequence, Activity, State Diagram	main.py, command.py, handlers	BT-01--10, OT-01--10, PT-01--10	1

02.	2	Database	V architecture	Component diagram, System architecture	databases	NA	2
03.	3	Backup and GUI	V architecture	Sequence, Activity, State Diagram, System architecture	Backup and GUI	BT-03	3

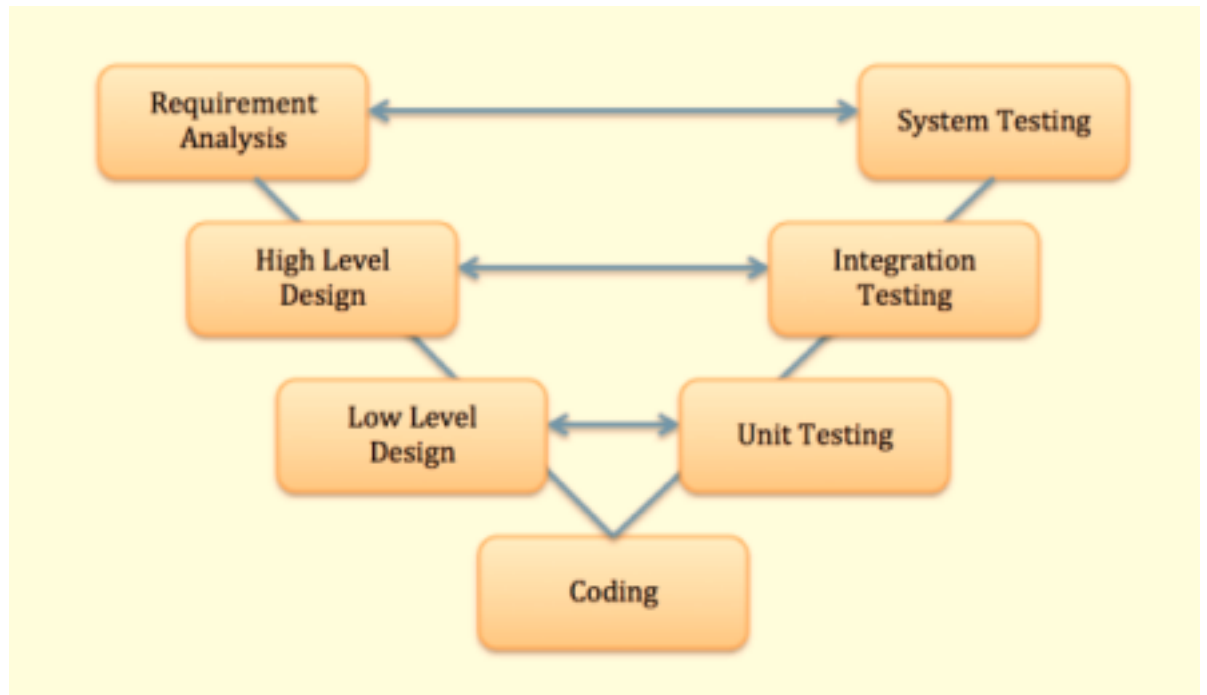
3. PROJECT PLAN :

Project Lifecycle

From what we have seen in the SRS document, we have a rather good set of requirements in a detailed manner. We do not have to do this iteratively, as in, we do not have to keep updating features one after the other. We can do all that is planned at once in an incremental format.

For the above-mentioned reasons, V-Model lifecycle seems to be a particularly good fit for this project. This is also called the verification and validation model. Our requirements being well known, we cannot make use of the iterative lifecycle model. This model is highly scalable and ensures that phases are completed one at a time, increasing the granularity making it easy to divide the tasks among the members of the team.

The below image shows an example of the working of this Lifecycle model/methodology.



Tools

The tools that we are planning on using throughout the lifecycle of the product development are listed below.

Planning Tool

We plan on using Trello (<https://trello.com/en>) for planning all the tasks to be done in the development of this project. Trello allows us to create boards for each project containing cards, which can be filled with different tasks. It allows us to granulate our tasks, supports checklists making it extremely easy to manage all the tasks.

We can add attachments to cards, which makes it easier to communicate with the team members for new iterations and for getting feedback. We can also add due dates to cards which come in handy when we must keep up deadlines.

They follow a freemium model, i.e., most features are free but for a few additional features we will have to pay. For the current project that we are working on, and considering that the team is small, we will not be requiring the premium version.

Design Tool

There is not really a lot of design required for our project because we are having a message-based interface. But we might need to design the workflows for the working of the bot, and these can be done with the help of StarUML.

We will need to make use of a logo design software like Canva to build a logo/display picture for the bot. We will need this logo for marketing purposes as well.

Version Control

For version control, we will be making use of **GitHub** which is a Git repository hosting service. The free version should be more than sufficient to work on this project, but because they offer the pro version for free of cost to students, we have access to that as well. Meaning we can make our repository private if we choose to.

Development Tool

We will be making use of VS Code from Microsoft for the development of this Discord bot. VSC is a very lightweight integrated development environment which can run on most machines. We will also be making use of code formatting tools that come installed on VSC. This will help maintain uniformity when multiple members are working on the same code.

Bug Tracking

We will be making use of Trac (<https://trac.edgewall.org/>) for bug tracking. This is an opensource tool for more than just bug tracking. It supports end-to-end project planning, writing Wikis, integrated Git and so on. This is the most widely recommended bug tracking tool and is in use by major projects like bbPress, GeoGebra, HTTPS Everywhere, jQuery, etc.

Testing Tools

There are not many tools out there for testing Discord bots because the interaction with Discord bots is in the form of messages. But among them, the most used testing tools are listed below.

1. <https://pypi.org/project/distest/>
2. <https://github.com/lucasgmagalhaes/corde>

We plan to use these to test out our project before making a public version available.

Deliverables

The final deliverables for this project are mentioned below.

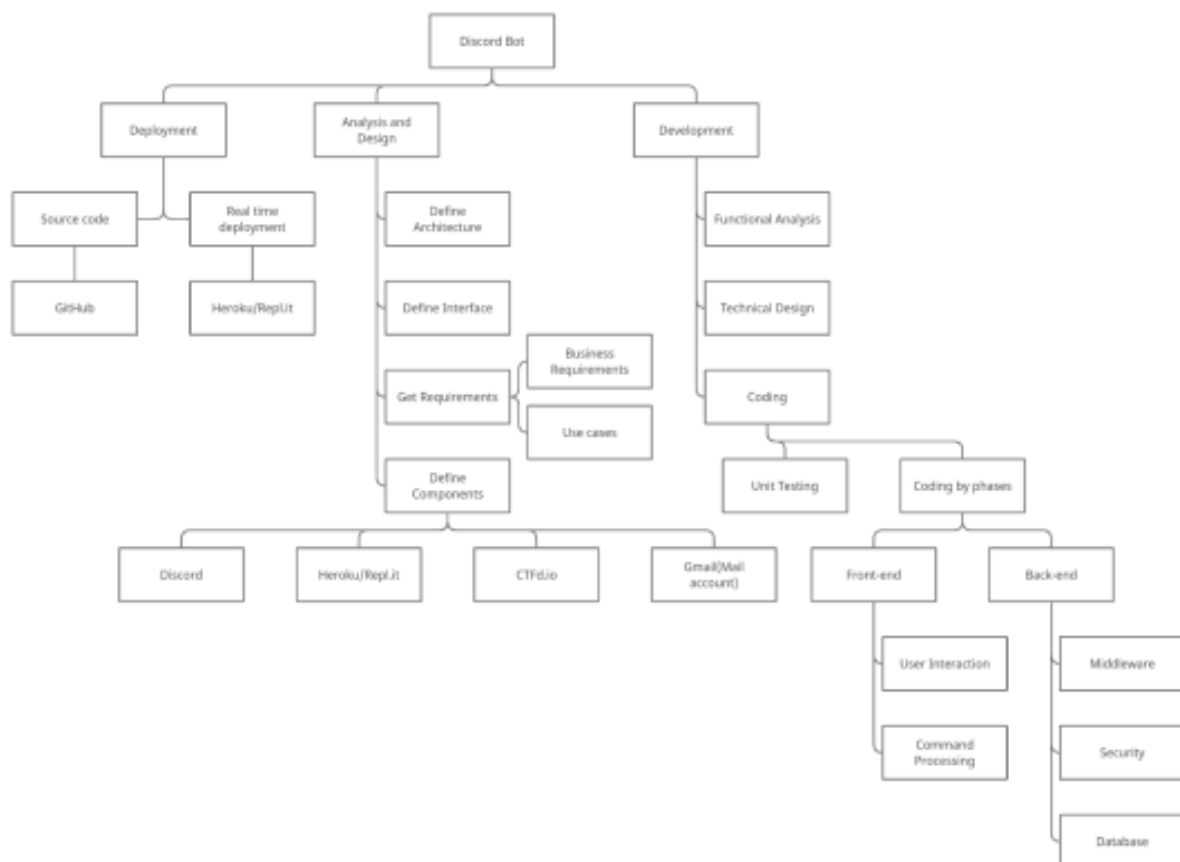
- API – We will be making use of the API given by Discord to communicate with it. This is a **reuse component**. It is readily available for free to cost to anyone willing to work on building Discord bots.
- CTF Discord Bot – This is a Discord bot to help organizers effectively conduct Capture the Flag competitions. It provides a message/command-based access to the CTFd framework i.e., the user can access the CTFd framework by sending small messages to

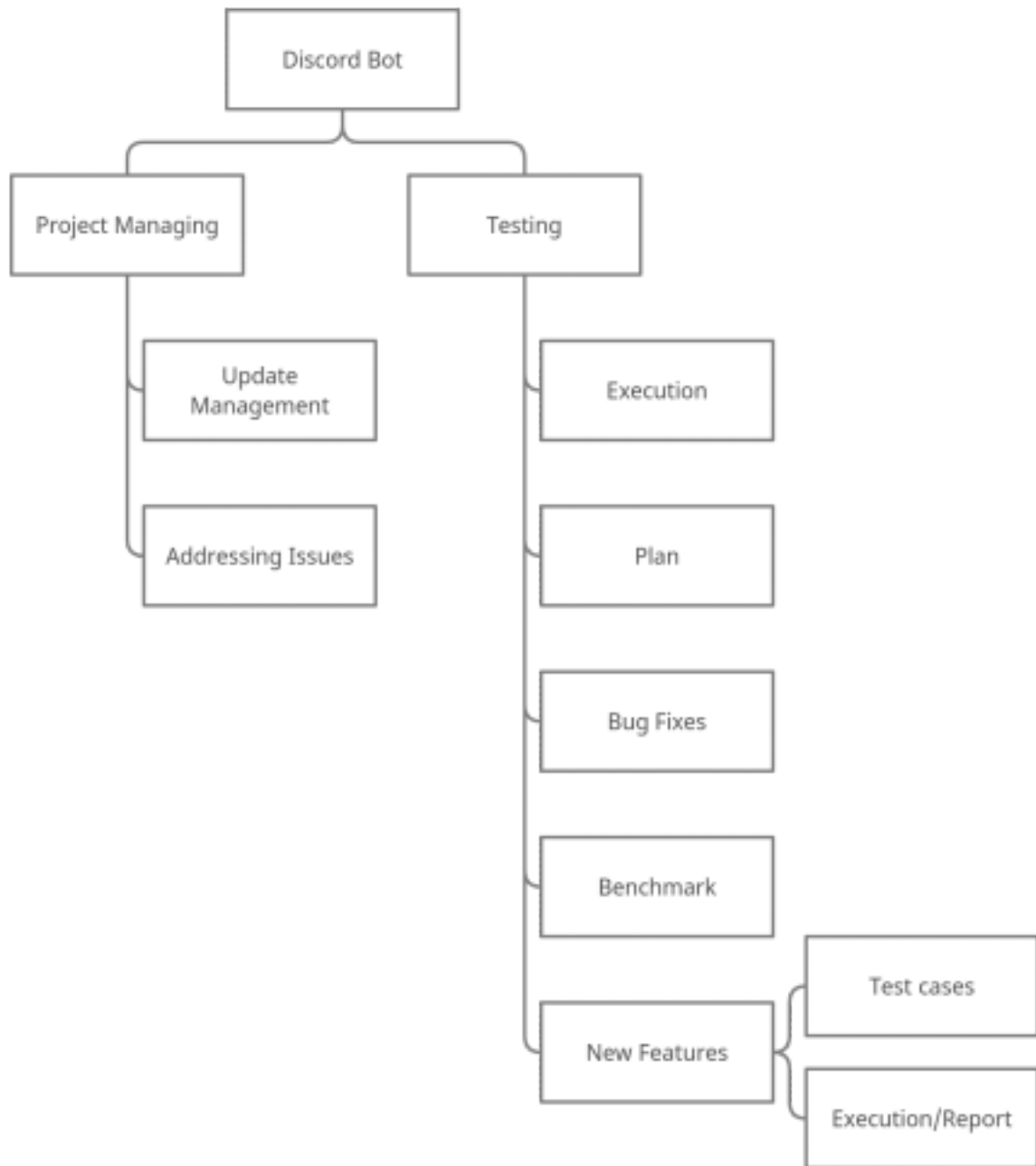
the bot from his/her Discord server. **This is a build component.**

- A dashboard for all stats – We plan to provide a GUI based dashboard for the admins to see every important information regarding the competition they are conducting. This would include the leaderboard, the challenge list, the teams, and suspicious activity and other such important details. This is a **build component** as well.
- Open Source the source code on GitHub – We will be putting the source code on GitHub for public access with certain licenses so that the open-source community can contribute and make it better.

WBS

The images shown below are the Work Breakdown Structure we have planned for the Discord Bot. Because of space constraints, we have divided this into two images. The first part deals more with development, while the second image deals more with testing and product launch.





Effort Estimate

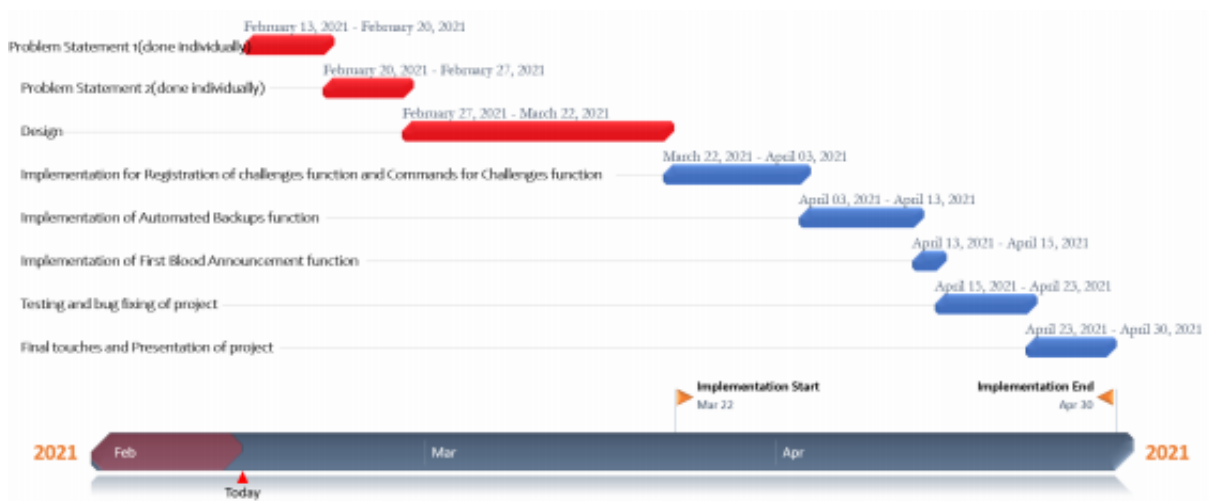
Using the constructive cost model, we get around 2.4 man-months for this project with a time of 3.5 months. The below table shows the constants involved in this formula.

Project Type	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

According to this estimate, it will take one organic developer 3.5 months to code this project. We have a team of 4, so this should technically speed up the time needed to implement this project.

Using other methods, we got Registration of Participants - 6 days for 1 person, first Blood announcement - 4 days for 1-person, automated Backups - 7 days for 2 people, commands for challenges - 13 days for 3 people for the development phases. All in all, this seems to sum up to the same as the previous method.

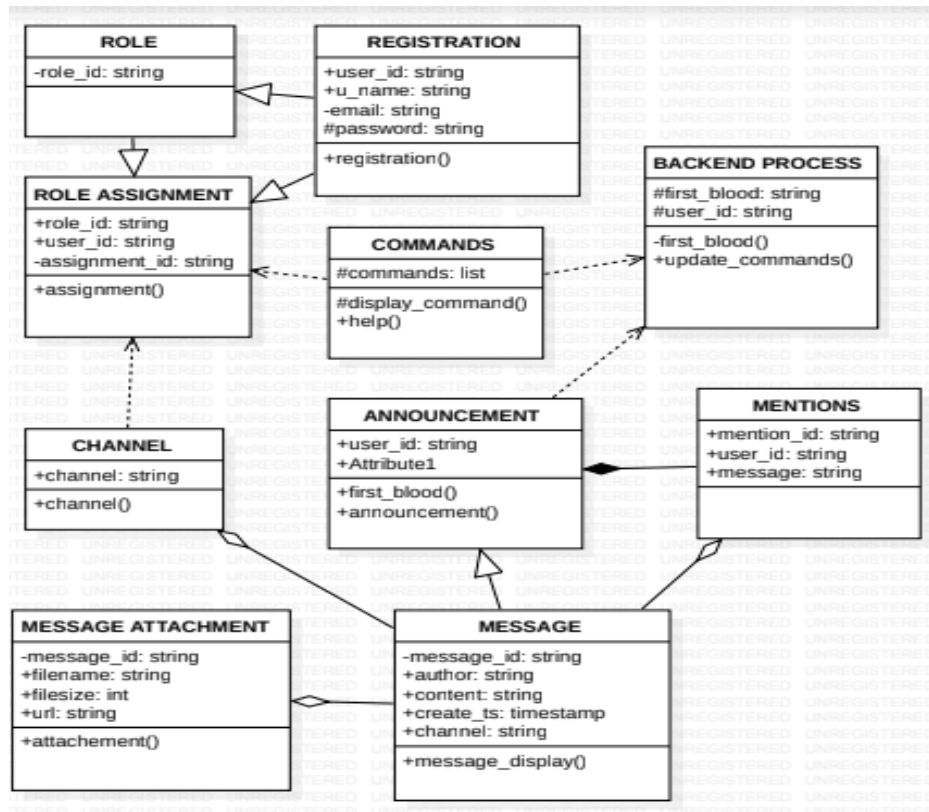
Gantt Chart



In the above image you can see the Gantt chart showing the timelines planned for development, implementation, testing and presentation of this project.

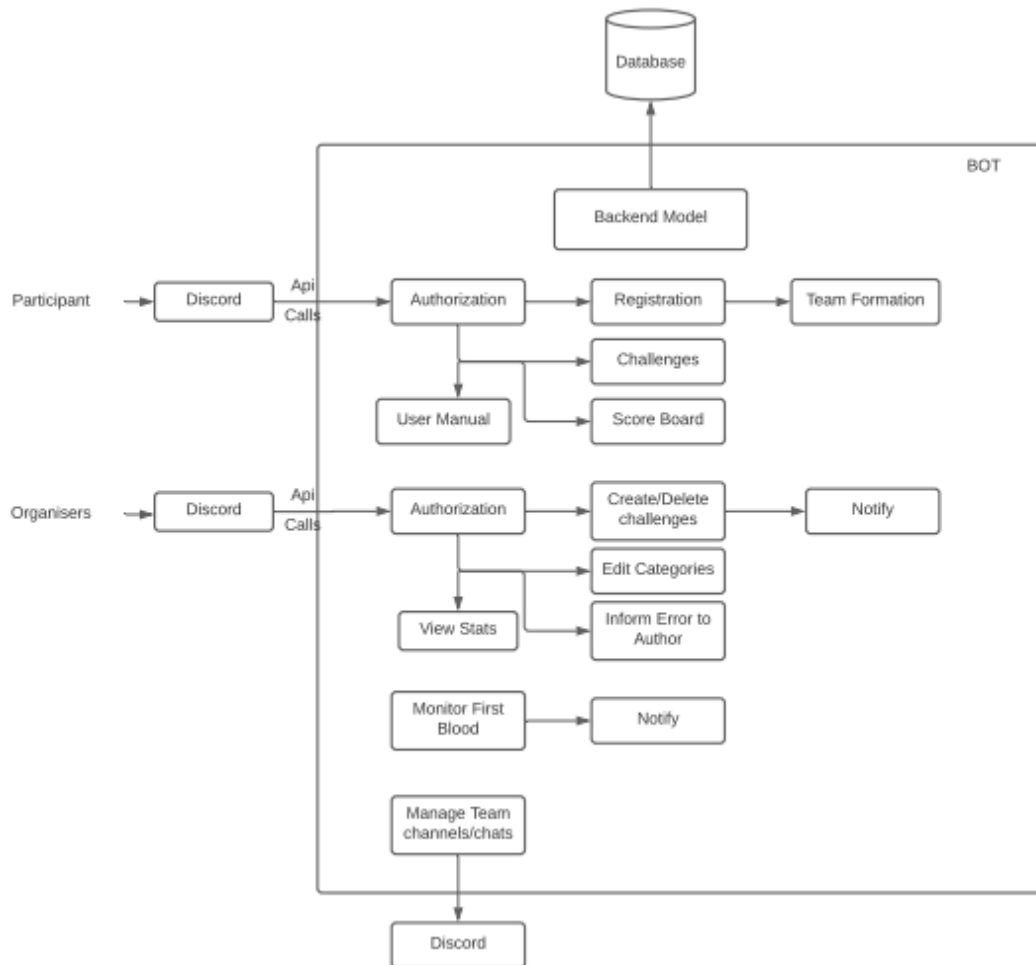
4. DESIGN DIAGRAMS :

- CLASS DIAGRAM :



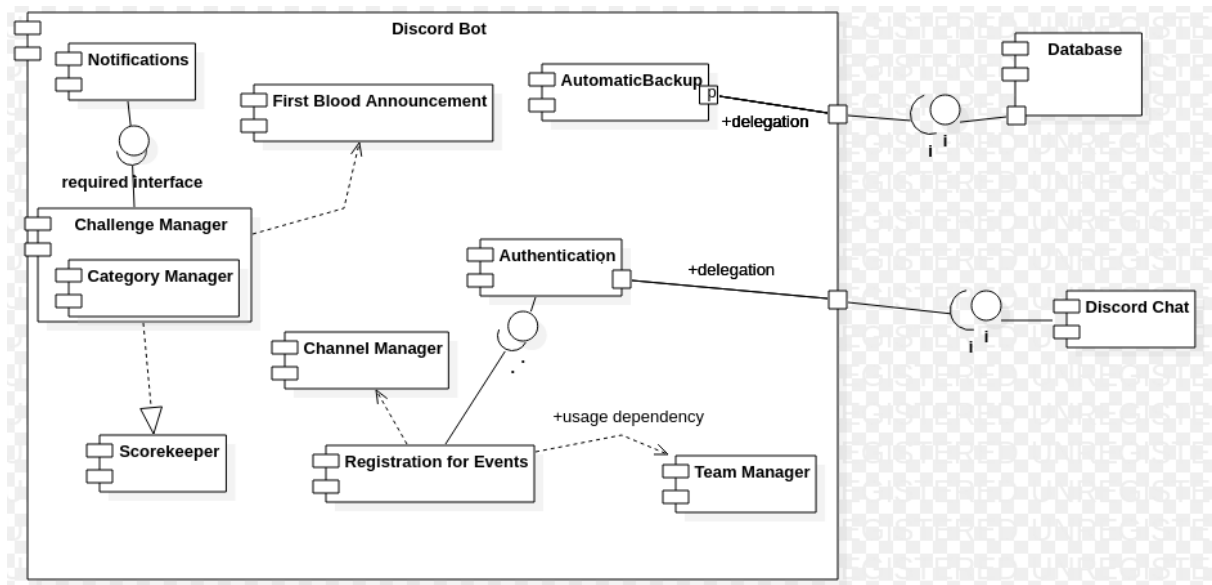
The class diagram is a static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations, and the relationships among objects. The class diagram displays the interaction between bot and the discord api and how the bot reacts to the message conversed with it.

- SYSTEM ARCHITECTURE :



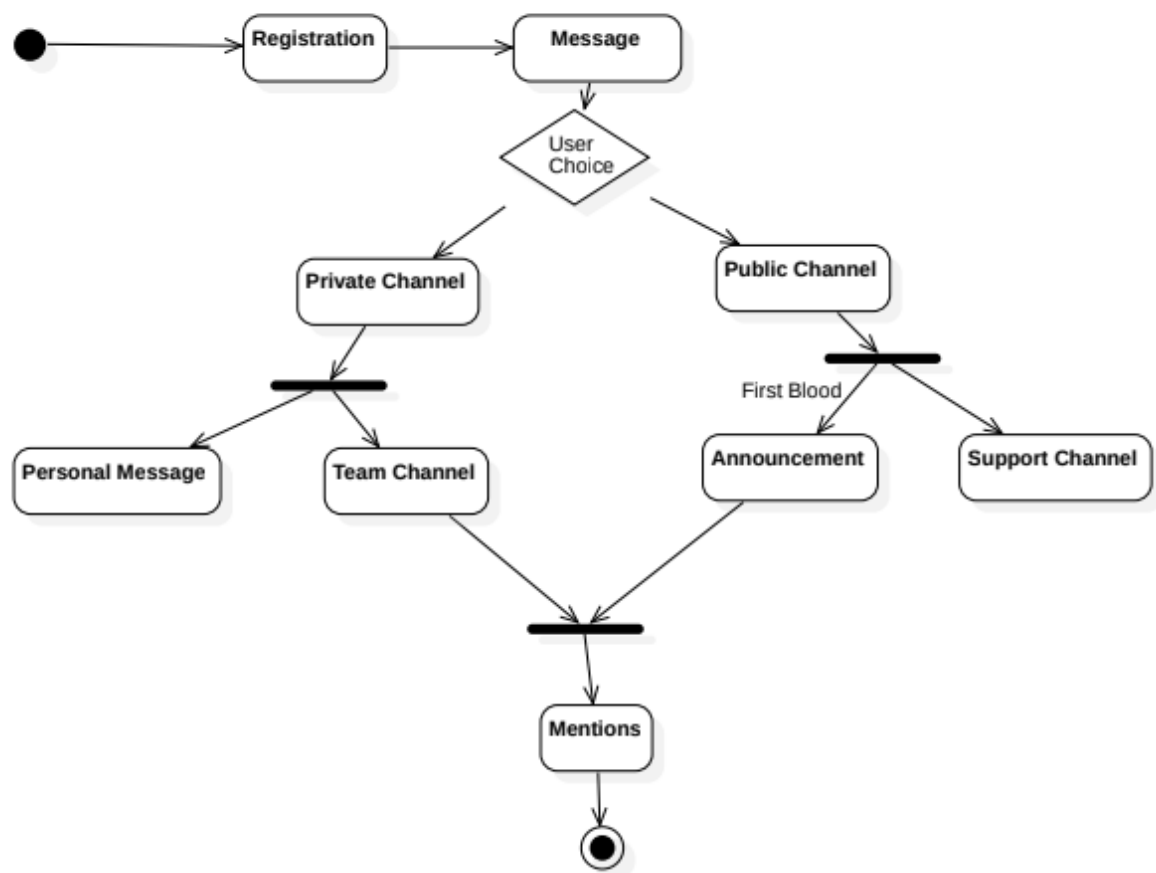
The system architecture provides the general overview of the components of the system and how it interacts with the outside world. The organisers and participants will interact with the bot through the discord app or website. The bot will read the commands posted on the channel and perform the appropriate task. The bot will also make use of a database to store all the details related to the CTF.

- **COMPONENT DIAGRAM :**



The component diagram shows the different components that are part of the system and how they interact with each other using interfaces. The different components that are part of the system are Discord chat, database and Discord bot which consists of Challenge manager, category manager, scorekeeper, channel manager, Team manager, Automatic Backup, authentication, First blood announcement and notifications.

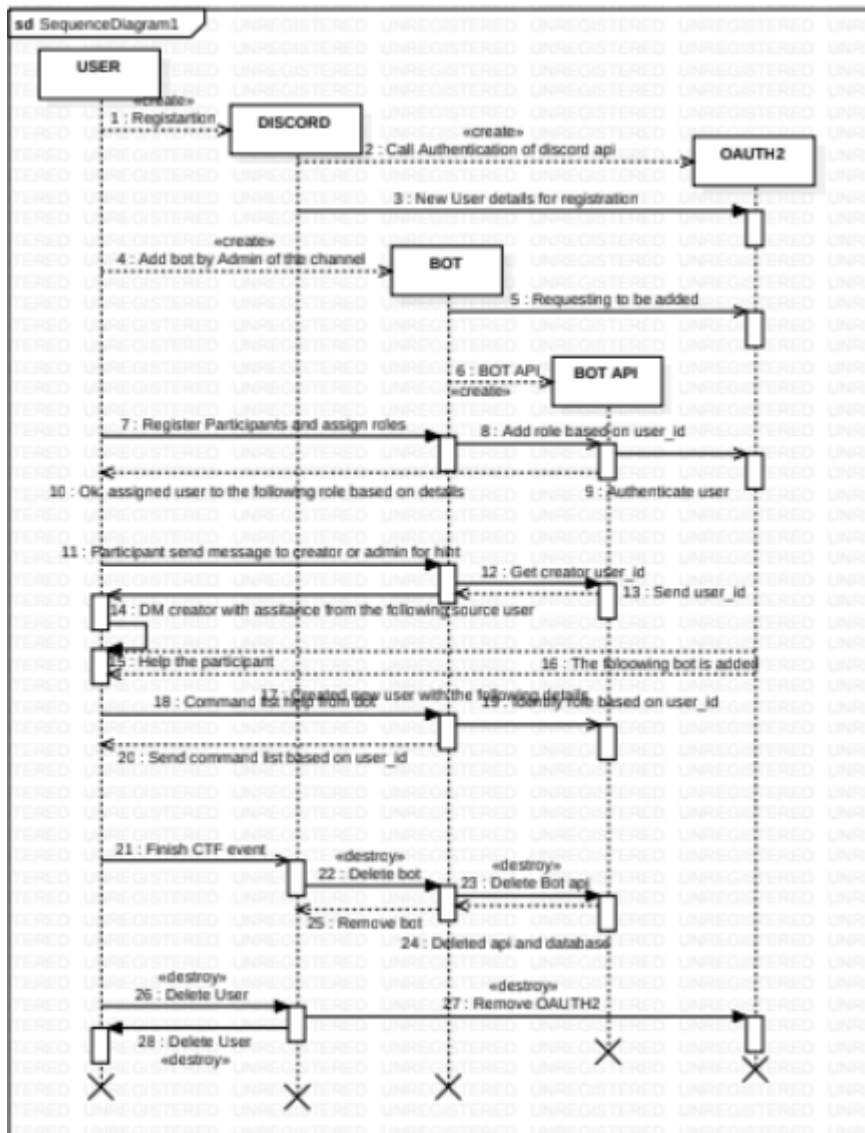
- **STATE DIAGRAM :**



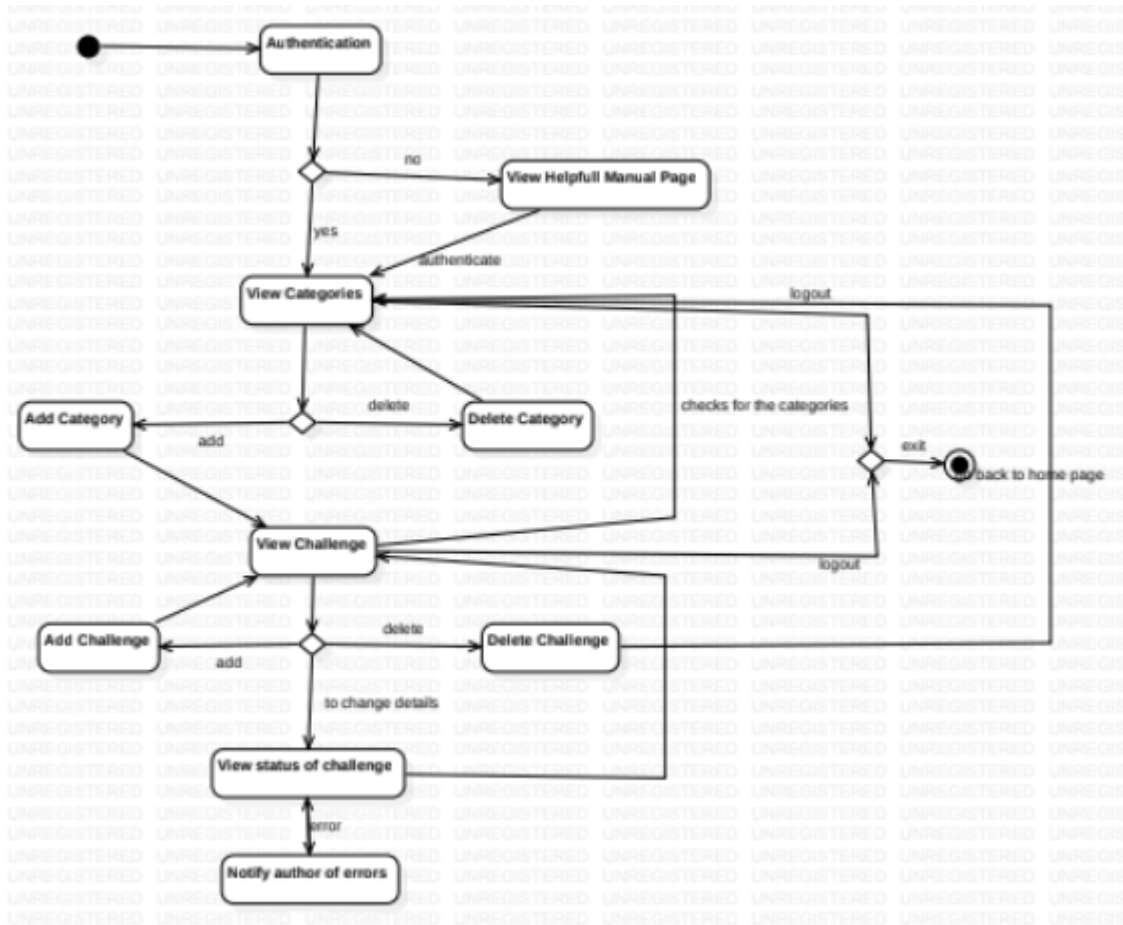
The state diagram represents the different states our system is in at different points of time. The first state is the registration where the user registers his team. Next for interacting with the bot, the user has two main choices of private or public channel. The private channel itself could either be his personal/direct message to the bot or the private team channel which the team has access to. The public channel will be used to make announcements or for displaying the help/support message. Finally the bot could be accessed from anywhere by mentioning/tagging it.

- **SEQUENCE DIAGRAM :**

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Here it shows the interaction sequence how bot is added to the discord server via Discord OAUTH2 and how the user on sending a message to the bot contacts its api part to display the message according to the test case.



- **ACTIVITY DIAGRAM :**



The activity diagram indicates the sequence of actions the user would perform during the course of interaction with the system. First is the authentication part, where the user has to register a team and login as part of that team. The user could then use the help option to view the commands it can ask the bot to perform. The user can then view the challenges and categories. The organisers will have the option to add/delete a challenge/category. In case of any problem with the challenge, the organiser can also notify the author of the error. Finally, the user can log out to exit the system.

5. MODULE DESCRIPTION :

- **FIRSTBLOOD :**

Firstblood functionality displays whether the submitted flag is the first time or not, otherwise on the flag submission the bot displays “Correct Flag”.

- **BACKUP :**

Backup functionality backup's the postgres database every 30min. The database contains the participant team and the points collected by the same. Screenshot in scoreboard section.

- **TEAM REGISTRATION :**

This functionality is for registering teams participating in the CTF. If two teams try to register with the same name, it will display the team already registered and also provide a token so that other team members can join the team.

- **LOGIN TOKEN :**

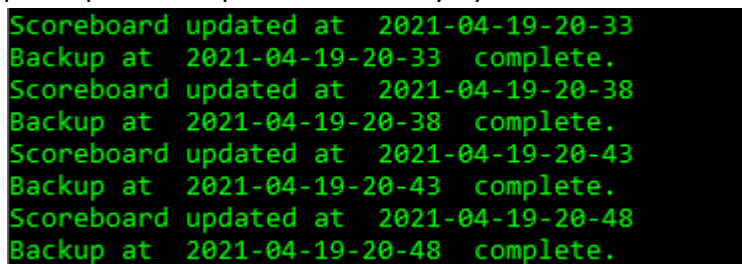
The token provided during team registration allows the team members to join the same team. It's like an authentication step so that there is no confusion in team and team members.

- **POINT REDUCTION :**

This function is implemented as part of HTB type functionality where the points awarded for solving the challenge is reduced if the team is not in top 5.

- **SCOREBOARD :**

The web interface will be provided to the team during the event. It displays the ranking of the team and number of challenges solved by them. Scoreboard is updated every 10 minutes, otherwise the participant can update it manually by ">>scoreboard command".



```
Scoreboard updated at 2021-04-19-20-33
Backup at 2021-04-19-20-33 complete.
Scoreboard updated at 2021-04-19-20-38
Backup at 2021-04-19-20-38 complete.
Scoreboard updated at 2021-04-19-20-43
Backup at 2021-04-19-20-43 complete.
Scoreboard updated at 2021-04-19-20-48
Backup at 2021-04-19-20-48 complete.
```

- **BOUND TIME :**

Bound time is implemented on challenges, where the challenges get hidden/deleted after a certain time of the release.

- **SUBMIT FLAG :**

Participants use this command to submit a flag to the bot, who verifies the flag and displays the output accordingly.

- **LIST CHALLENGES :**

List challenges functionality displays the released challenges by the organizers.

- **CREATE CHALLENGES :**

Create challenges helps the organizer to create the challenges with specific id, name and so on.

- **RELEASE CHALLENGES :**

Release challenges are used to display the created challenges to the participant.

- **HIDE CHALLENGES :**

Hide challenges are quite opposite to release challenges and it hides the challenges by the command of the organizer.

- **DELETE CHALLENGES :**

Delete challenges deletes the hidden challenge from the discord server.

- **GUI :**

GUI is the html dashboard of the ranking/scoreboard system and also contains a help manual.

6. TEST CASES :

Functionality 1:

Tasks that the bot will do in the background, automatically(any 3 tasks according to the SRS)

BOT TEST CASES:

Test Case ID	Name of Module	Test case description	Pre-conditions	Test Steps	Test data	Expected Results	Actual Result	Test Result
BT-01	First blood	Test first blood functionality.	Bot should be online .	>>register <team_name> >>login TOKEN >>submit flag	>>submit flag	First blood	First blood	PASS
BT-02	First blood	Test first blood functionality when the flag is already submitted by some other user.	Bot should be online and the flag is already submitted by some other user.	>>register <team_name> >>login TOKEN >>submit flag	>>submit flag	Correct Flag submitted	Correct Flag submitted	PASS
BT-03	Backup	Bot should backup instance automatically	Bot should be online	Automatic backup	Automatic backup	Backup completed on server	Backup completed on server	PASS

BT-04	Team registration	Test registration functionality.	Team name must not exist.	1.>>register<team_name>	>>register Anonymous	Registered successfully	Anonymous successfully registered, here is your auth token.	PASS
BT-05	Team registration	Test registration functionality with duplicate name.	Team name must exist.	1.>>register<team_name>	>>register Anonymous	Team name already exists.	Team name already exists.	PASS
BT-06	Points Reduction	Test Points reduction functionality based on number of solves.	Flags must be submitted by many users.	>>submit flag	>>submit 12fd	Scoreboard updated with less points.	Scoreboard updated with less points.	PASS
BT-07	Live scoreboard	Test scoreboard functionality to update and display it every 10 minutes.	Flags must be submitted by many users.	>>submit flag	>>submit 12fd	Updated scoreboard.	Updated scoreboard.	PASS
BT-08	Bound on end time	Test that flags are not accepted after end	Current time must be greater	>>submit flag	>>submit 12fd	Challenge has reached bound time.	Challenge has reached bound time.	PASS

		time.	than end time					
--	--	-------	---------------------	--	--	--	--	--

Functionality 2:

Tasks that an organizer can perform with this bot(any 3 tasks according to the SRS)

ORGANIZER TEST CASES:

Test Case ID	Name of Module	Test case description	Pre-conditions	Test Steps	Test data	Expected Results	Actual Result	Test Result
OT-01	List challenges	Test list challenges with the aid of help	Bot should be online	1.>>help	>>help	Display all the commands	Display all the commands	PASS
OT-02	Create Challenges	Test create challenge functionality	Bot should be online.	1.>>help create-challenges 2.>>create-challenge challenge- <id> category- <id> description- <id> files-# flag-#	>>create-challenge challenge-1 category-1 description-1 files-1 flag-1	Challenge created	Task with id 1 successfully created	PASS
OT-03	Create Challenges	Test create challenge functionality for duplicate challenge	Bot should be online. Challenge should be already created	1.>>help create-challenges 2.>>create-challenge challenge- <id> category- <id> description- <id>	>>create-challenge challenge-1 category-1 description-1 files-1 flag-1	Task already exists	Task already exists	PASS

				files-# flag-#				
OT-04	Release challenge	Test release challenge functionality	Challenges should be created.	1.>>help release-c challenge 2.>>release-challenge <id>	>>release-challenge 1	Release challenge	Task with id 1 successfully released	PASS
OT-05	Release challenge	Test release challenge functionality with invalid challenge id.	Challenges with given id must not exist.	1.>>help release-c challenge 2.>>release-challenge <id>	>>release-challenge 10	Task does not exist	Task with id 10 did not exist	PASS
OT-06	Hide challenge	Test hide challenge functionality.	Challenges must be created and released.	1.>>help hide-challenge 2.>>hide-challenge <challenge-id>	>>hide-challenge 1	Task successfully hidden	Task successfully hidden	PASS
OT-07	Hide challenge	Test hide challenge functionality with invalid id.	Challenges with given id must not be released.	1.>>help hide-challenge 2.>>hide-challenge <challenge-id>	>>hide-challenge 10	Task does not exist	Task with id 10 did not exist	PASS
OT-08	Delete challenge	Test delete challenge	Challenges must be created	1.>>help delete-c challenge 2.>>delete-c	>>delete-c challenge 1	Task successfully delete	Task successfully delete	PASS

		functionality.		te-challenge <id>		d	d	
OT-09	Delete challenge	Test delete challenge functionality with invalid challenge id.	Challenges with given id must not exist.	1.>>help delete-c challenge 2.>>delete-c challenge <id>	>>delete-c challenge 10	Task does not exist	Task with id 10 did not exist	PASS
OT-10	Scoreboard using GUI	Test scoreboard on webpage .	Bot is online, challenges created and released and participants submitted flags.	1. Go to dashboard.html	1. Go to dashboard.html	Current scoreboard displayed	Current scoreboard displayed	PASS

Functionality 3:

Tasks that a participant can perform with this bot(any 3 tasks according to the SRS)

PARTICIPANT TEST CASES:

Test Case ID	Name of Module	Test case description	Pre-conditions	Test Steps	Test data	Expected Results	Actual Result	Test Result
PT-01	Team registration	Test the register functionality	Bot should be online	1.>>help register 2.>>register <team_name>	>>register Anonymous	Registration of team with token for joining the team	Anonymous successfully registered, here is your auth token.	PASS

PT-02	Login team	Test the login functionality	Team should be registered	1. >>help login 2.>>login <token>	>>login TOKEN	Successfully logged-in	Successfully logged-in as Anonymous	PASS
PT-03	Login team	Test the login functionality with invalid token	Team should not be registered	1. >>help login 2.>>login <invalid token>	>>login INVALID TOKEN	Invalid token	Invalid token	PASS
PT-04	Login team	Test the login functionality	Team should be registered and logged in	1.>>login <token>	>>login TOKEN	Already logged-in	You already logged-in	PASS
PT-05	View Challenges	Test the view challenges feature	Participants should be logged in and challenges should exist.	1.>>help challenges 2.>>challenges 3.>>challenges-info name	>>challenges	View the released challenges	View the released challenges	PASS
PT-06	Add challenge	Test to add challenges	Participants should be logged in.	1.>>create-challenge challenge-1 category-1 description-1 files-1 flag-1	1.>>create-challenge challenge-1 category-1 description-1 files-1 flag-1	Role admin is required to run this command	Role admin is required to run this command	PASS
PT-07	Submit flag	Test submit flag feature	Participants should be logged in and the	1.>>help submit 2.>>submit	>>submit flag-1	Submit the flag to obtain	1.First blood for first submission	PASS

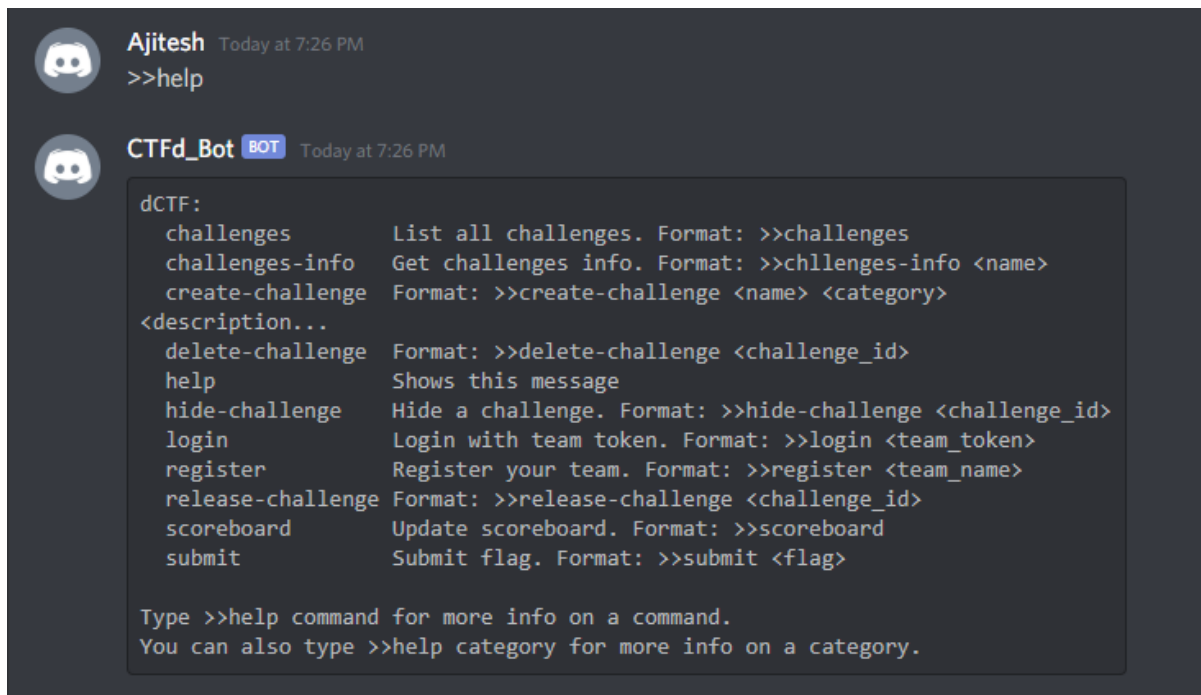
		with correct flag.	flag is correct.	<flag>		points	on 2. Correct Flag	
PT-08	Submit flag	Test submit flag feature with incorrect flag.	Participants should be logged in and flag input is incorrect flag	1.>>help submit 2.>>submit <flag>	>>submit ^Acasd25@4_ak8sdllss	Wrong flag	Wrong Flag	PASS
PT-09	Submit flag	Test submit flag feature with duplicate correct flag.	Participant had already submitted	1.>>help submit 2.>>submit <flag>	>>submit flag-1	Already solved this	Already solved this	PASS
PT-10	Scoreboard	Test to view scoreboard.	Participants should be logged in and at least one challenge should be solved	1.>>help scoreboard 2.>>scoreboard	>>scoreboard	Display scoreboard	Display scoreboard	PASS

7. SCREENSHOTS OF OUTPUT :

AJITESH - ADMIN/ORGANIZER

SHASHANK/AAYUSH - PARTICIPANT

- **HELP FROM ADMIN**



A screenshot of a Discord chat interface. At the top, a user named 'Ajitesh' (ADMIN/ORGANIZER) sends a message '>>help' at 7:26 PM. Below it, a bot named 'CTFd_Bot' (BOT) responds with a detailed help message. The bot's message lists various commands and their formats, including 'challenges', 'challenges-info', 'create-challenge', 'delete-challenge', 'help', 'hide-challenge', 'login', 'register', 'release-challenge', 'scoreboard', and 'submit'. It also includes instructions on how to use the help command for more information.

Ajitesh Today at 7:26 PM
>>help

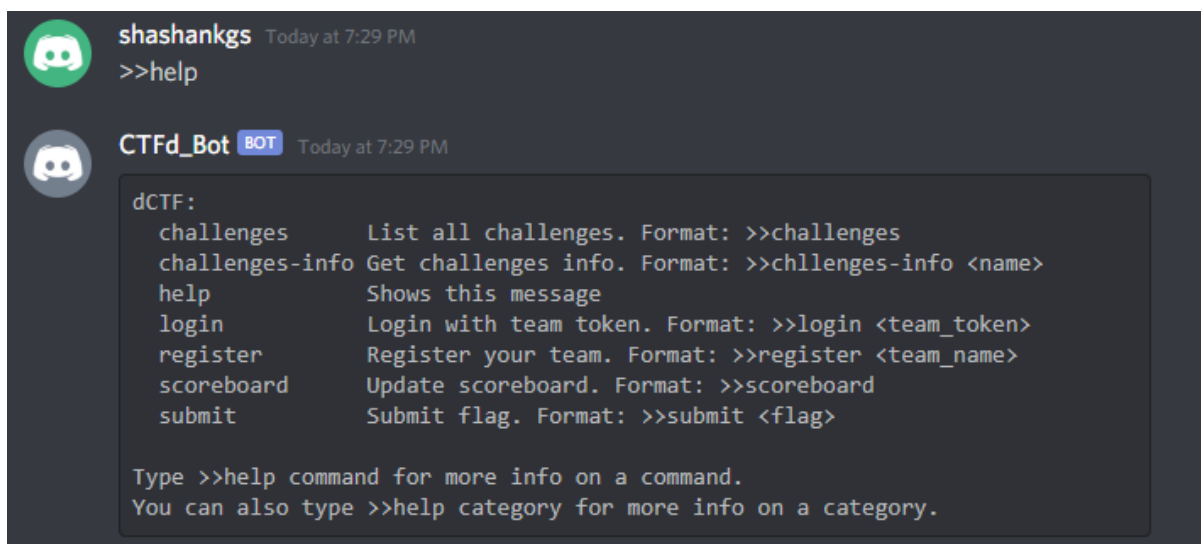
CTFd_Bot BOT Today at 7:26 PM

dCTF:

challenges	List all challenges. Format: >>challenges
challenges-info	Get challenges info. Format: >>challenges-info <name>
create-challenge	Format: >>create-challenge <name> <category> <description>...
delete-challenge	Format: >>delete-challenge <challenge_id>
help	Shows this message
hide-challenge	Hide a challenge. Format: >>hide-challenge <challenge_id>
login	Login with team token. Format: >>login <team_token>
register	Register your team. Format: >>register <team_name>
release-challenge	Format: >>release-challenge <challenge_id>
scoreboard	Update scoreboard. Format: >>scoreboard
submit	Submit flag. Format: >>submit <flag>

Type >>help command for more info on a command.
You can also type >>help category for more info on a category.

- **HELP FROM PARTICIPANT**



A screenshot of a Discord chat interface. At the top, a user named 'shashankgs' (PARTICIPANT) sends a message '>>help' at 7:29 PM. Below it, the same bot 'CTFd_Bot' (BOT) responds with a help message. This message is similar to the one in the previous screenshot but lists fewer commands, omitting 'create-challenge', 'delete-challenge', 'hide-challenge', 'release-challenge', and 'scoreboard'.

shashankgs Today at 7:29 PM
>>help

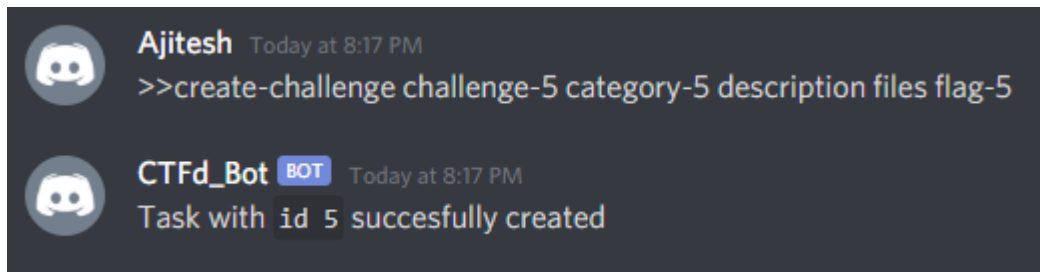
CTFd_Bot BOT Today at 7:29 PM

dCTF:

challenges	List all challenges. Format: >>challenges
challenges-info	Get challenges info. Format: >>challenges-info <name>
help	Shows this message
login	Login with team token. Format: >>login <team_token>
register	Register your team. Format: >>register <team_name>
scoreboard	Update scoreboard. Format: >>scoreboard
submit	Submit flag. Format: >>submit <flag>

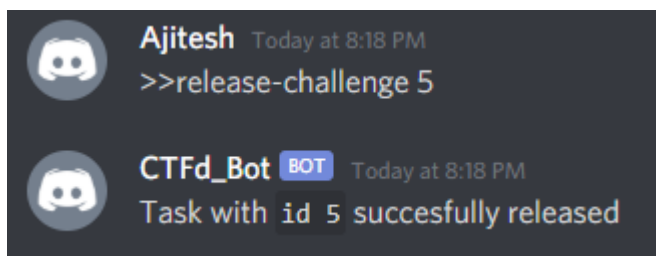
Type >>help command for more info on a command.
You can also type >>help category for more info on a category.

- **CREATE CHALLENGE**



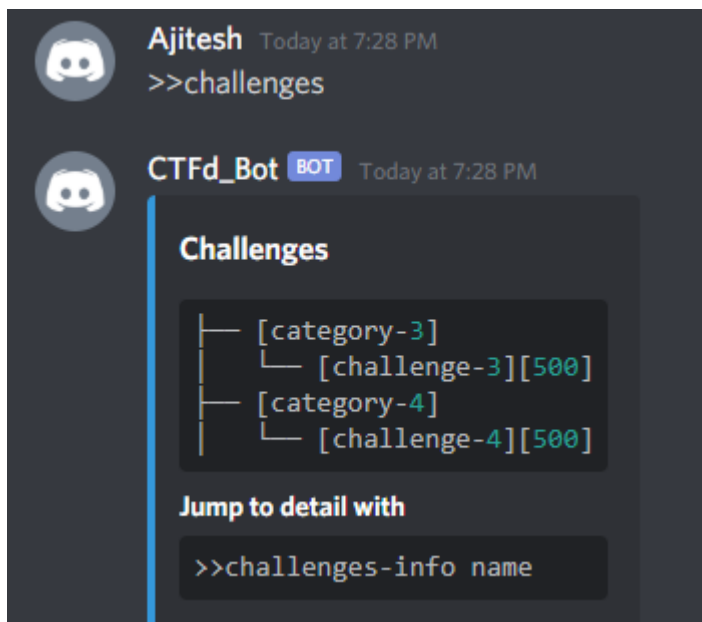
A screenshot of a Discord chat interface. The first message is from user 'Ajitesh' at 8:17 PM, containing the command '>>create-challenge challenge-5 category-5 description files flag-5'. The second message is from the bot 'CTFd_Bot' at 8:17 PM, responding with 'Task with id 5 succesfully created'.

- **RELEASE CHALLENGE**



A screenshot of a Discord chat interface. The first message is from user 'Ajitesh' at 8:18 PM, containing the command '>>release-challenge 5'. The second message is from the bot 'CTFd_Bot' at 8:18 PM, responding with 'Task with id 5 succesfully released'.

- **LIST OF CHALLENGES**



A screenshot of a Discord chat interface. The first message is from user 'Ajitesh' at 7:28 PM, containing the command '>>challenges'. The second message is from the bot 'CTFd_Bot' at 7:28 PM, displaying a list of challenges in a code block. The list shows two categories: 'category-3' with challenge 'challenge-3' having a value of 500, and 'category-4' with challenge 'challenge-4' having a value of 500. Below the list, there is a section titled 'Jump to detail with' followed by the command '>>challenges-info name'.

- **HIDE CHALLENGE**



Ajitesh Today at 8:18 PM

>>hide-challenge 5



CTFd_Bot BOT Today at 8:18 PM

Task succesfully hidden

- **DELETE CHALLENGE**



Ajitesh Today at 8:18 PM

>>delete-challenge 5



CTFd_Bot BOT Today at 8:18 PM

Task succesfully deleted

- **REGISTER TEAM**



Aksil Today at 8:30 PM

>>register Team_Anonymous



CTFd_Bot BOT Today at 8:30 PM

Team_Anonymous succesfully registered, here is your auth token

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ0ZWFTX2lkIjoxLCJ0ZWFTX25hbWUiOiJlUzZWFtX0Fub255bW91cyJ9.47Vh1JOBEUdgpxCeUMtovAId83xnoGlqKzmBplBtgbs
```

- **LOGIN PARTICIPANT TO TEAM**



Aksil Today at 8:30 PM

>>login

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ0ZWFTX2lkIjoxLCJ0ZWFTX25hbWUiOiJlUzZWFtX0Fub255bW91cyJ9.47Vh1JOBEUdgpxCeUMtovAId83xnoGlqKzmBplBtgbs
```



CTFd_Bot BOT Today at 8:30 PM

Succesfully Logged-in as Team_Anonymous

- **CHALLENGES**



Aksil Today at 8:24 PM

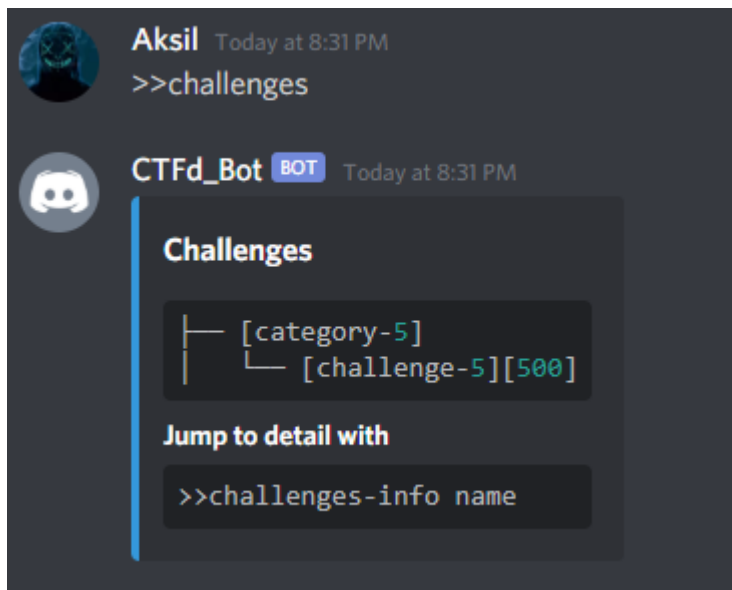
>>help challenges



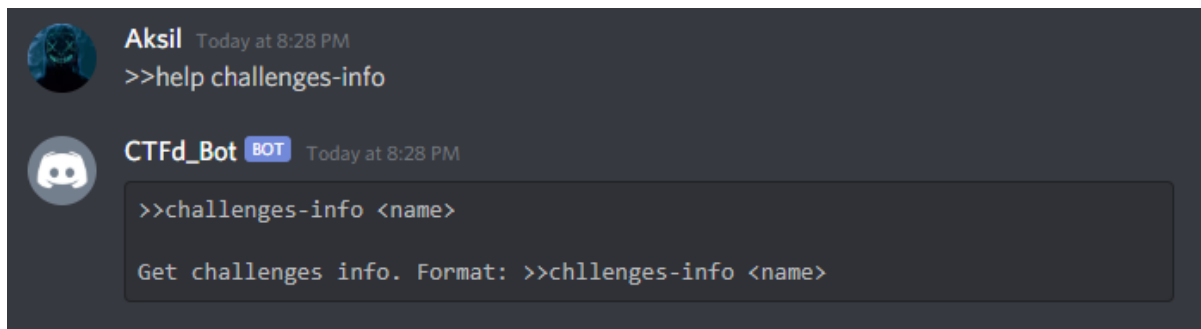
CTFd_Bot BOT Today at 8:24 PM

```
>>challenges
```

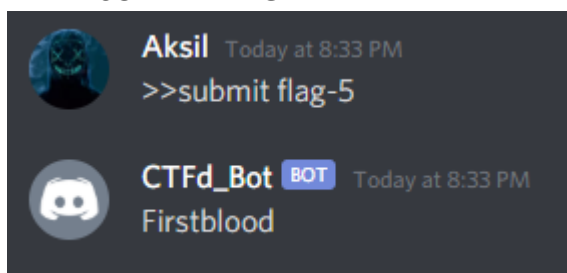
```
List all challenges. Format: >>challenges
```



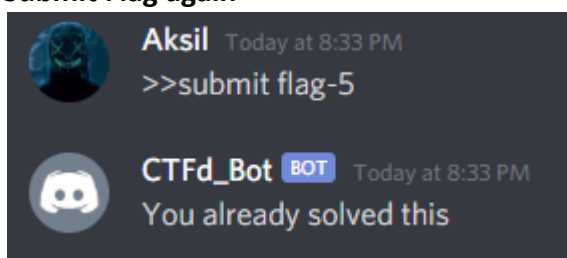
- **CHALLENGES-INFO**



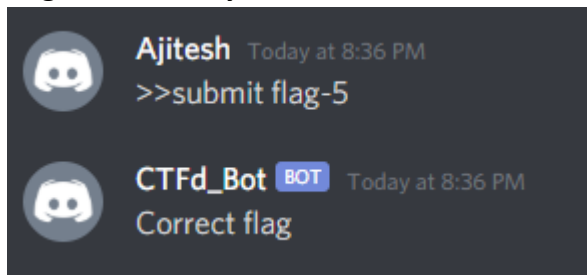
- **SUBMIT FLAG**



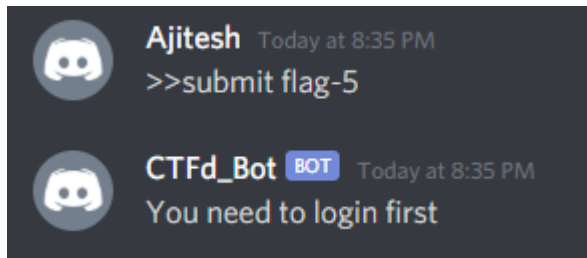
Submit Flag again



Flag submitted by other team



- Participant is not logged in



- SCOREBOARD



● LOGIN/SIGNUP

The image displays two screenshots of a web application running on a local host. The top screenshot shows the 'SIGN UP FORM' at the URL 'localhost/GUI/signup.html'. The form includes fields for 'Username' (with the value 'Online'), 'Password', and 'Confirm Password', followed by 'Sign Up' and 'Sign In' buttons, and a 'Forgot Password' link. The bottom screenshot shows the 'SIGN IN FORM' at the URL 'localhost/GUI/login.html'. This form includes fields for 'Username' (with the value 'username') and 'Password', followed by 'Sign In' and 'Sign Up' buttons, and a 'Forgot Password' link. Both forms are set against a background of glowing blue and purple geometric lines. The browser's taskbar and address bar are visible in both screenshots.

SIGN UP FORM

Username
Online

Password

Confirm Password

Sign Up

Sign In

[Forgot Password](#)

SIGN IN FORM

Username
username

Password

Sign In

Sign Up

[Forgot Password](#)

• DASHBOARD GUI

The image shows two screenshots of a web application interface for a Discord bot dashboard. The top screenshot displays the 'Help Manual' page, and the bottom screenshot displays the 'CTF Dashboard Score card' page. Both pages feature a sidebar with navigation links and a main content area with a table of commands or scores.

Help Manual

Dashboard [Log Out](#)

Command	Description
register	Register your team. Format: >>register <team_name>
login	Login with team token. Format: >>login <team_token>
create-challenge	Format: >>create-challenge <name> <category> <description> <files> <flag>
release-challenge	Format: >>release-challenge <challenge_id>
hide-challenge	Hide a challenge. Format: >>hide-challenge <challenge_id>
delete-challenge	Format: >>delete-challenge <challenge_id>
submit	Submit flag. Format: >>submit <flag>
challenges	List all challenges. Format: >>challenges

CTF Dashboard Score card

[Log Out](#)

Rank	Team Name	Number of Solves	Points
1	Team_Anonymous	1	499
1	Team-test	1	499