Requirements Analysis Document (RAD)

Introduction

This project consists of the development of a Discord bot that can automate the management and organization of Capture The Flag (CTF) events within a Discord server, relying on data provided by the CTFTime API. This bot will allow server administrators to easily create channels, roles, and notifications for each CTF event, simplifying communication and engagement with community members.

Project goals

- **Automatize CTF management** within Discord servers, reducing the workload for administrators and increasing efficiency in organizing CTF events.
- **Create and manage channels, roles, and Discord** events specific to each CTF, facilitating communication and coordination among participants.
- **Integrate with the CTFTime** platform to retrieve and use accurate and up-to-date information about CTF events.

Functional requirements

1. **Automatic creation of dedicated channels**:
    - Given a CTF event URL on CTFTime, the bot creates a dedicated channel for the event in the previously configured “CTF Active” category.
    - The bot sends an embed message to the channel, containing detailed information about the CTF event, including title, description, start and end dates, logo, and link to the CTFTime site.
2. **Discord event management**:
    - The bot creates a Discord event for each CTF with start and end dates retrieved from CTFTime.
    - At the beginning and end of the event, the bot automatically sends notifications to participants, updating the status of the event in the server.
3. **Role Management**:
    - Creating a role associated with CTF for participants, with customizable name and permissions.
    - Automatic role assignment to members who react to the embed related to the CTF event in the dedicated channel, simplifying the joining process.
4. **Notifications for the start and end of each event**:
    - When the event starts, the bot automatically tags members with the associated role in the CTF channel, sending a start message.
    - When the event ends, the bot sends an alert message and automatically moves the channel to the “CTF Archived” category, organizing past events.
5. **Configuration of categories and roles**:
    - Configuration command to define the category of active CTFs, the category of archived CTFs, and the role required to manage CTFs (based on role position).
    - Only users with the role manager can create new CTF events.

Non-functional requirements

1. **Scalability**:

    - The bot must be able to handle multiple CTFs simultaneously on different servers, allowing for large-scale use.
2. **Reliability**:
    - Robust error handling during channel, event and role creation. Any problems should be reported in real time to the server administrator with appropriate error messages.
3. **Security**:
    - Limited access to CTF management commands, configurable via roles. Only users with the correct role can execute commands for creating and managing CTFs.
    - Secure management of data received from CTFTime to prevent possible exploits.
4. **Maintenance and ease of configuration**:
    - Clear documentation for administrators on how to configure and use the bot.
    - Bot architecture that easily allows changes and updates, ensuring compatibility with future CTFTime or Discord API changes.
5. **Response time**:
    - The bot must respond to commands quickly, minimizing wait time for users, especially when creating channels and events.

Feature definition (commands)

Command `/init`

- **Description**: Initialization command that configures basic settings for managing CTFs in the server.
- **Parameters**:
    - `category_ctf_active`: Category in which the channels of active CTFs will be placed.
    - `category_ctf_archived`: Category into which the channels of archived CTFs will be moved.
    - `role_manager`: Role required to be able to create and manage CTFs, based on the location of roles in the server.
- **Functionality**:
    - Sets the categories and the role required for future CTF creations.
    - If the configuration is completed correctly, the bot confirms with a message; if there is an error, it reports the problem to the user.

Command `/create_ctf`

- **Description**: Command to create a CTF event within the server.
- **Parameters**:
    - `url`: URL of the event on CTFTime in the format `https://ctftime.org/event/<id>`.
- **Functionality**:
    - Retrieves CTF information from the CTFTime API, checking the validity of the URL.
    - Creates a dedicated channel in the “CTF Active” category, with an embed containing event details.
    - Create a Discord event for the CTF with start and end dates, notifying participants at the beginning and end of the event.
    - Creates a dedicated role for the CTF and assigns it to members who react to the embed.
    - If the event already exists in the server, sends an alert message informing that the CTF already exists.

Flowchart of main operations

1. **Configuration (`/init`)**:
    - A user with administrator permission in any channel sends the `/init` command with the necessary parameters.
    - The bot saves categories and the role to manage CTFs.
    - The bot confirms with a message or reports errors in case of missing parameters.
2. **Create CTF Event (`/create_ctf`)**:
    - A user with the role defined in the `/init` command sends the `/create_ctf` command in any channel with the CTFTime event URL.
    - The bot verifies the URL and retrieves the event data.
    - The bot creates the channel in the “CTF Active” category.
    - The bot creates the Discord event and role for participants.
    - The bot sends an embed into the channel, allows role assignment with reaction, and notifies at the beginning and end of the event.

Use Cases

Use Case 1: Initial setup of the bot

- **Actors**: Discord server administrator.
- **Description**: The server administrator runs the `/init` command to configure basic settings for handling CTFs.
- **Main Flow**:
    1. The administrator runs the `/init` command specifying the categories and the role to manage CTFs.
    2. The bot saves the settings and confirms the configuration.
- **Extensions**:
    - If the `/init` command does not include all the necessary parameters, the bot sends an error message and requests the missing information.
- **Pre-conditions**: The one sending the message must have the necessary permissions to execute the command.
- **Post-conditions**: The bot configuration is completed, ready for use.

Use Case 2: Creating a CTF event

- **Actors**: Administrator of the Discord server.
- **Description**: Administrator creates a new CTF event using the `/create_ctf` command.
- **Main Flow**:
    1. The administrator sends the `/create_ctf` command with the event URL to CTFTime.
    2. The bot retrieves the event information from the CTFTime API.
    3. The bot creates a channel in the “CTF Active” category and publishes an embed with the event details.
    4. The bot creates a Discord event for the CTF, defining the start and end dates.
    5. The bot creates a role for participants and assigns the role to users who react to the embed in the channel.
    6. The bot schedules notifications for the start and end of the event.
- **Extensions**:

    - If the event URL is invalid or the CTFTime API is not responding, the bot sends an error message to the administrator.
    - If the CTF event already exists in the server, the bot sends an alert without duplicate channels or roles.
- **Pre-conditions**: The bot must have been configured with the `/init` command.
- **Post-conditions**: CTF event is created with channel, information embed, Discord event and participant role.

Use Case 3: Member participation in a CTF

- **Actors**: Member of the Discord server.
- **Description**: A server member joins the CTF event by reacting to the information embed in the dedicated channel.
- **Main stream**:
    1. The server member displays the embed with the CTF information in the channel.
    2. The member reacts to the embed with the appropriate emoji.
    3. The bot automatically assigns the role created for the CTF to the member.
- **Extensions**:
    - If the member does not have the necessary permissions to access the channel or react to the embed, the bot reports the error and does not assign the role.
- **Pre-conditions**: The CTF event must have been created and the role configured for reactions.
- **Post-conditions**: Member gets the CTF role and full access to the dedicated channel.

Use Case 4: Event Start Notification

- **Actors**: Discord bot.
- **Description**: When the start date of a CTF event is reached, the bot automatically sends a notification in the dedicated channel.
- **Main Stream**:
    1. At the start time of the event, the bot sends a message in the CTF channel, tagging members with the associated role to inform them of the start of the event.
- **Pre-conditions**: The CTF channel and role must have been created, and the bot must have permissions to send notifications.
-     - **Post-conditions**: Participants are notified of the start of the CTF.

Use Case 5: Conclusion and archiving of the event

- **Actors**: Discord bot.
- **Description**: At the end of a CTF event, the bot notifies participants and archives the channel.
- **Main Flow**:
    1. On the end date of the event, the bot sends a message in the CTF channel informing participants that the event has ended.
    2. The bot automatically moves the CTF channel to the “CTF Archived” category to organize past events.
- **Pre-conditions**: The event must be active and the bot configured with categories for archiving.
- **Post-conditions**: The channel is archived, members are notified of the end of the event, and the role is made light gray (all roles created later by the bot will be placed on top of the previous role so that the role of the active ctf is always foregrounded).

Additional considerations

- **Integration API**: Ensure that the CTFTime API is accessible and manage any rate limits to avoid blockages.

Conclusion

The proposed Discord bot will automate the management of CTFs, simplifying the creation and management of complex events and improving the experience of server users. Through its integration with CTFTime, the bot will provide up-to-date information on CTFs, while the automation of channels, events, and roles will make the organization efficient and professional.