**Awesome Ticket Challenge Documentation**

**Overview**

The Awesome Ticket Challenge is designed to streamline the process of managing unanswered questions (tickets) in a Discord server. This solution allows moderators to easily view, access, and manage these tickets through a web interface.

**Backend**

**Technologies**

* **Framework**: FastAPI
* **Language**: Python

**Endpoints**

**GET /tickets**

* Description: Fetches a list of tickets.
* Query Parameters:
  + limit (int, optional): The number of tickets to retrieve.
* Response: JSON list of tickets.

**DELETE /tickets/{ticket\_id}**

* Description: Deletes a ticket by ID.
* Path Parameters:
  + ticket\_id (str): ID of the ticket to delete.
* Response: Confirmation message.

**Logging**

* Logs are stored in app.log, containing information about operations and errors.

**Summary**

The main.py file underwent several enhancements to improve functionality and interoperability of the backend service. These changes include the addition of logging, the implementation of CORS (Cross-Origin Resource Sharing) middleware, and the introduction of a new ticket deletion endpoint.

Logging was configured using Python's logging module, enabling the application to record informational messages, warnings, and errors in a file named app.log. This aids in debugging and monitoring the application's behavior.

CORS middleware was introduced to allow requests from different origins, addressing the need for the front-end and back-end to communicate seamlessly, especially when they are hosted on different domains.

A new DELETE endpoint /tickets/{ticket\_id} was created to facilitate ticket deletion. This endpoint enhances the application's functionality by allowing moderators to remove tickets that are no longer needed. Error handling was incorporated into this endpoint to manage cases where a ticket ID does not exist, with appropriate logging of such events for maintenance and monitoring.

The TicketRepository class in ticket\_repository.py was enhanced to include a new method for ticket management. This enhancement is focused on extending the functionality of the ticket system by allowing the deletion of tickets.

A method named remove\_ticket was added to the TicketRepository class. This method enables the removal of a ticket based on its ID. It includes a logging mechanism to track the process of ticket removal, enhancing the traceability of actions performed on the ticket data. The method first checks if the ticket with the specified ID exists; if not, it logs a warning and raises a ValueError. If the ticket exists, it is removed from the dataset, and a log entry is made to confirm successful removal.

This addition to TicketRepository significantly improves the application's capabilities, allowing for better management of tickets.

**Frontend**

**Technologies**

**Framework**: React

**UI Library**: Material-UI

**Pages**

**Index Page**

* Path: /
* Description: The home page with navigation buttons.

**Tickets Page**

* Path: /tickets
* Description: Displays a list of tickets with options to view or delete them.

**Components**

**Ticket**

* Description: Represents a single ticket.
* Props:
  + ticket: The ticket data.
* Features: View message, Delete ticket.

**Summary**

The revised IndexPage now includes three distinct buttons, each representing different functionalities: Knowledge Base, Tickets, and FAQ Insights. Each button is styled using Material-UI components and icons (LibraryBooks for Knowledge Base, SupportAgent for Tickets, and Lightbulb for FAQ Insights), providing a more intuitive and visually appealing user interface. Additionally, the 'Tickets' button is linked to the /tickets route, facilitating navigation within the application.

In the tickets.tsx component of the front-end application, a functional and interactive interface is provided for moderators to manage Discord tickets. This is what the component does:

* Fetching and Displaying Tickets: Upon loading, the component makes a GET request to the backend (http://localhost:5001/tickets) to fetch the tickets. These tickets are then displayed in a list format. Each ticket item shows essential information like the message ID and timestamp and includes two buttons - one to open the ticket in Discord and another to delete the ticket.
* Deleting Tickets: A DELETE request is sent to the backend when a moderator chooses to delete a ticket. The front end reflects this change immediately by filtering out the deleted ticket from the state, ensuring the list is up to date.
* Error Handling and User Feedback: The component includes error handling for both fetching and deleting tickets, logging any issues that occur. Moreover, if there are no tickets available, a message is displayed to inform the user.

**Repository**

URL: [GitHub Repository Link](https://github.com/korabs-x/awesomeqa-example)

**Screenshot of Results**

The homepage:

A screenshot of a computer

Description automatically generated

The tickets page:

A screenshot of a computer

Description automatically generated