Documentation covering the design, implementation details, and user guide of Quizmaster Extravaganza

Main.py:

The code provided is a Python script for a quiz application built with Streamlit. It allows users to create and take quizzes on various topics. The code consists of several functions that handle different aspects of the quiz application, such as preparing the main screen, generating quizzes, navigating between questions, computing scores, and displaying feedback.

The prepare_main_screen() function clears the session state and sets the status to 'main', which is the initial state of the application.

The prepare_config_screen() function sets the status to 'config', which is the screen where users can adjust the quiz settings.

The prepare_score_screen() function sets the status to 'score', which is the screen that displays the user's score after completing the quiz.

The prepare_feedback_screen() function sets the status to 'feedback', which is the screen that displays the user's answers and the correct answers for each question.

The generate_quiz() function generates a quiz based on the specified number of questions and quiz context. It calls the prepare_quiz() function from the quiz module to retrieve the questions, choices, and answers for the quiz. If the input is invalid, it sets the invalid_input flag to True.

The go_to_next_question_button() function handles the logic for moving to the next question in the quiz. It updates the session state, generates a new quiz if necessary, and handles invalid input.

The got_to_previous_question_button() function handles the logic for going back to the previous question in the quiz. It updates the session state and stores the user's previous answers.

The compute_score() function calculates the user's score based on their answers and the correct answers.

The create_main_screen() function creates the main screen of the quiz application, displaying a welcome message and instructions.

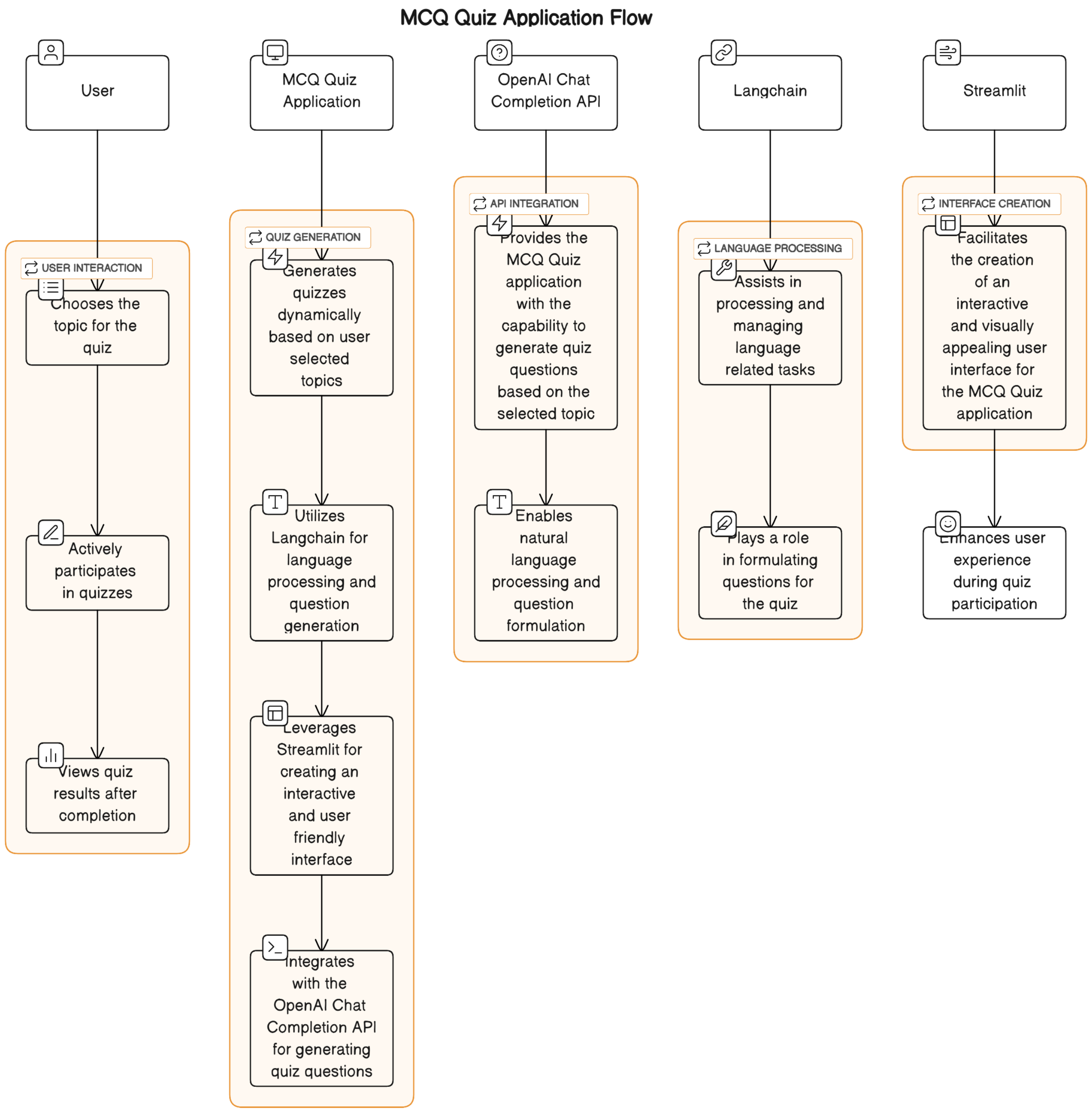
The create_config_screen() function creates the configuration screen, where users can adjust the quiz settings such as the number of questions and the quiz topic.

The create_score_screen() function creates the score screen, displaying the user's score and the highest score achieved so far. It also writes the score to a file.

The create_feedback_screen() function creates the feedback screen, displaying the user's answers and the correct answers for each question.

The create_question_screen() function creates the question screen, displaying a question and radio buttons for the user to select their answer. It also handles navigation between questions.

The code uses the st module from Streamlit to create the user interface and manage the session state.


MCQ Quiz Application Flow
User
MCQ Quiz Application
OpenAI Chat Completion API
Langchain
Streamlit
USER INTERACTION
Chooses the topic for the quiz
Actively participates in quizzes
Views quiz results after completion
QUIZ GENERATION
Generates quizzes dynamically based on user selected topics
Utilizes Langchain for language processing and question generation
Leverages Streamlit for creating an interactive and user friendly interface
Integrates with the OpenAI Chat Completion API for generating quiz questions
API INTEGRATION
Provides the MCQ Quiz application with the capability to generate quiz questions based on the selected topic
Enables natural language processing and question formulation
LANGUAGE PROCESSING
Assists in processing and managing language related tasks
Plays a role in formulating questions for the quiz
INTERFACE CREATION
Facilitates the creation of an interactive and visually appealing user interface for the MCQ Quiz application
Enhances user experience during quiz participation

quiz.py:

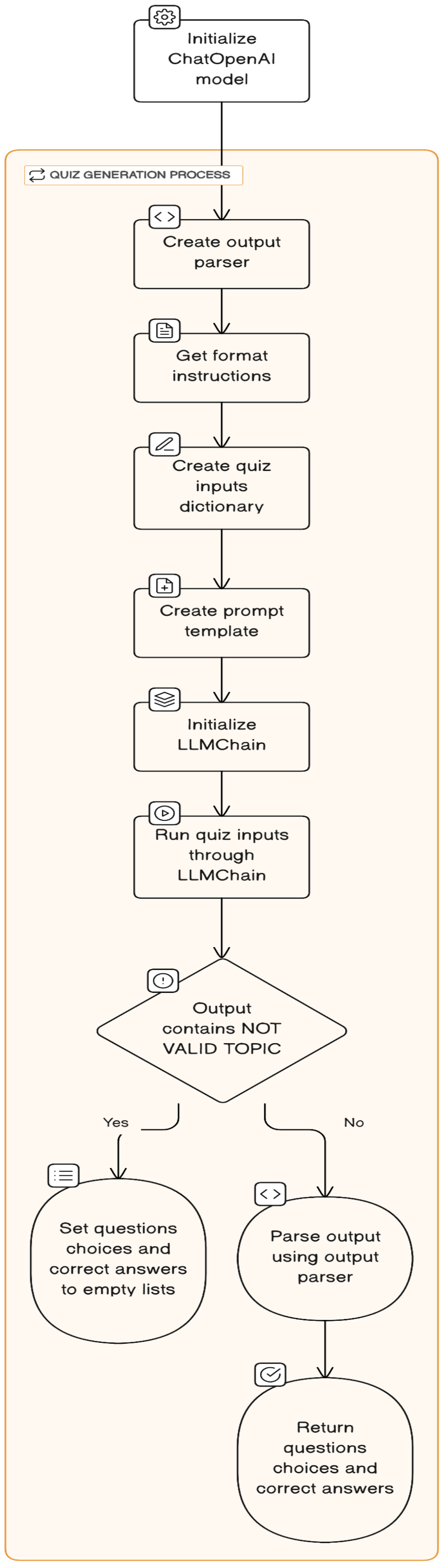
The code provided is a Python function called prepare_quiz. This function prepares a quiz by interacting with the OpenAI language model. It takes in three inputs: num_questions (the number of questions in the quiz), topic (the topic of the quiz), and key (the API key for accessing the OpenAI language model).

Here is a breakdown of the code:

1. The function initializes the ChatOpenAI model and a StructuredOutputParser.
2. It creates a prompt template using the format instructions from the output parser.
3. The function initializes the LLMChain with the ChatOpenAI model and prompt template.
4. It runs the quiz inputs through the LLMChain to generate the quiz output.
5. If the output contains the string "NOT-VALID-TOPIC", the function sets the questions, choices, and correct answers to empty lists.
6. Otherwise, it parses the output using the output parser and extracts the questions, choices, and correct answers.
7. Finally, the function returns the questions, choices, and correct answers.

The function uses the following classes from the langchain module: LLMChain, ChatOpenAI, StructuredOutputParser, ResponseSchema, and PromptTemplate. It also relies on the schemas and raw_prompt variables defined outside the function.

Flow Chart


Initialize ChatOpenAI model
QUIZ GENERATION PROCESS
Create output parser
Get format instructions
Create quiz inputs dictionary
Create prompt template
Initialize LLMChain
Run quiz inputs through LLMChain
Output contains NOT VALID TOPIC
Yes
No
Set questions choices and correct answers to empty lists
Parse output using output parser
Return questions choices and correct answers

logs_utils.py:

The code provided is a Python class called LoggerCreator that is used to create rotating loggers. The class has a constructor __init__ that takes a logs_dir parameter and creates the directory if it doesn't exist.

The class also has a method called create_rotating_logger that takes parameters such as log_name, log_dir, and level to create a rotating logger. It sets the logger's level, creates the log directory if it doesn't exist, and configures the log file names and rotation settings using TimedRotatingFileHandler.

The log files are created with names like log_name_info.log and log_name_error.log, and they rotate daily at midnight. The log messages are formatted with a timestamp, log level, and message content. The log files are encoded in UTF-8.

The method sets the log handlers' levels and suffixes, and adds them to the logger. Finally, it returns the logger object.

Overall, this code snippet provides a convenient way to create rotating loggers with customizable settings. Rotating loggers are useful when you want to manage the size of log files and prevent them from growing indefinitely.

With rotating loggers, log files are rotated or archived based on certain criteria, such as time or size. This helps in maintaining a manageable log file size and prevents it from consuming excessive disk space.

User Guide

Welcome to Quizmaster Extravaganza! 🎉

Where excitement and knowledge collide! 🚀 I am thrilled to curate an electrifying quiz for you.

Here you will find an interactive question and answer challenge to test your knowledge. Built with Streamlit, Langchain, and chat-GPT.

Instructions:

1. input preferred quiz topic
2. input number of questions
3. Press Start
4. Enjoy your Quiz!