# HEEDDATA INTERNSHIP

A AFSHEEN ZAAHRAH – 22BAD005

PROJECT – PDF READER CHAT BOT

**DAY-1:** This code employs object-oriented programming (OOP) in Python to structure the code into classes and objects, enhancing modularity and reusability. It also establishes a database connection to a server using SQLAlchemy as an Object-Relational Mapping (ORM) tool, enabling efficient and seamless database operations. This integration is beneficial for AI applications, as it facilitates managing and manipulating large datasets, supports complex queries, and ensures data integrity. By leveraging OOP and SQLAlchemy, this code promotes scalable, maintainable, and efficient AI development, allowing for streamlined data handling and processing.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

**Day-2:** This code leverages object-oriented programming (OOP) in Python to organize the code into classes and objects, enhancing modularity and reusability. It connects to a MySQL database using SQLAlchemy as an Object-Relational Mapping (ORM) tool and executes CRUD (Create, Read, Update, Delete) operations. It encapsulates the setup of the database connection, session management, database interactions, and file processing for user management. This approach is especially beneficial for AI applications as it enables efficient data handling, supports complex queries, and ensures data integrity, facilitating scalable and maintainable AI development.

A screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

**DAY-3:**  The code handles currency conversion using exchange rates from the Open Exchange Rates API. The class includes methods for reading exchange rates from a JSON file, calling the API to get the latest rates, saving the API response to the file, updating the stored rates, performing currency conversion, and saving the updated rates to a specified JSON file. The main execution of the code creates an instance of `CurrencyConversion`, updates the exchange rates, converts a specified value from INR to other currencies, and saves the updated exchange rates to a new JSON file. This setup facilitates the retrieval, conversion, and storage of exchange rate data in an organized manner.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

**DAY-4**: In this, I worked on a Python-based currency conversion application that integrates with MongoDB for data storage. I implemented a class `CurrencyConversion` that handles reading exchange rates from a file, calling an API to get the latest rates, saving this data to a file, and converting currencies based on the retrieved rates. The class also includes methods to save the conversion data into MongoDB. Key steps involved setting up MongoDB client connections, fetching and parsing JSON data from the Open Exchange Rates API, handling file operations for saving and loading data, and performing currency conversions. Through this project, I enhanced my skills in API integration, data manipulation, and database operations using MongoDB, and learned the importance of handling exceptions and maintaining code readability for easier debugging and maintenance.

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

A screen shot of a computer program

Description automatically generated

**DAY 5:** This code involves the extraction of text from a PDF document, conversion of the text into embeddings using a BERT model, and the implementation of a question-answering pipeline. Using the `fitz` library, text is extracted from the PDF, which is then saved to a text file. The BERT model (`bert-base-uncased`) from the Hugging Face `transformers` library, along with PyTorch, tokenizes the extracted text and generates embeddings that represent the semantic content of the text. Additionally, a question-answering pipeline is established using the `distilbert-base-uncased-distilled-squad` model. This pipeline takes a question and the extracted text as context, providing answers based on the text. For instance, when asked, "Can digital documents be uploaded with new SmallPDF?", the pipeline processes the question and context to deliver a relevant answer. This project exemplifies the integration of text extraction, embedding generation, and question-answering using advanced NLP models, offering a powerful tool for automated document processing and information retrieval.

A screenshot of a computer program

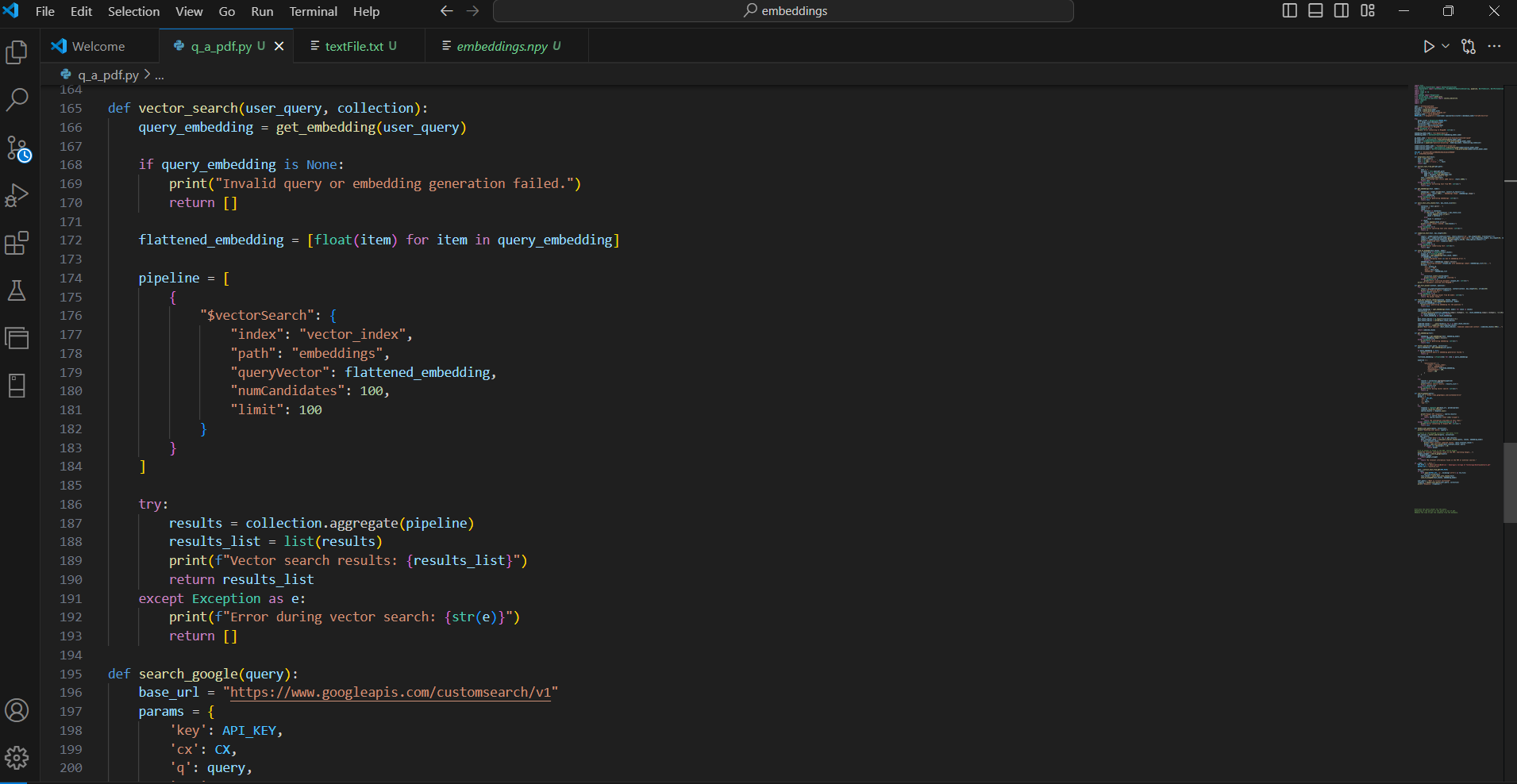
Description automatically generated

A screenshot of a computer program

Description automatically generated

**WEEK 2**

I explored the integration of diverse artificial intelligence techniques to create a sophisticated system for text analysis and information retrieval. I utilized the `SentenceTransformer` to generate embeddings, which capture semantic meanings of text segments. This was complemented by a pre-trained question-answering model, `bert-large-uncased-whole-word-masking-finetuned-squad`, enabling precise extraction of answers from context-rich documents. For summarizing lengthy texts, I employed the `facebook/bart-large-cnn` model, demonstrating its capability to distill information into concise summaries. Additionally, I implemented MongoDB to store and query vector embeddings, applying vector search to find relevant text chunks effectively. The integration of Google’s Custom Search API further enriched the system by allowing it to fetch supplementary information from the web. This highlights the power of combining various AI tools and models to build a comprehensive and efficient text analysis solution.



A screenshot of a computer

Description automatically generated

I have gained hands-on experience with integrating OpenAI’s API for advanced conversational capabilities and file management. I learned to initialize the OpenAI client and manage conversation threads effectively to interact with a virtual assistant. By implementing functions to handle file uploads and updates, I demonstrated how to enhance the assistant’s knowledge base with user-provided documents. Learnt the importance of handling asynchronous operations and polling for results to ensure timely responses. Additionally, I practiced secure management of sensitive API keys and explored file handling techniques for downloading and saving PDFs from URLs. Overall, this part of the project highlighted the practical application of API integration, file handling, and error management in developing sophisticated AI-powered solutions.

I am developing a Streamlit web application to interact with a PDF Insight Assistant. The application allows users to upload PDF files or provide links to PDFs, which are then processed to update the assistant’s knowledge base. Key functionalities include file uploading, downloading PDFs from URLs, and updating the assistant with new information. Users can submit queries via a text input box, and the assistant responds with relevant information extracted from the uploaded or linked PDF. This is highlighting my ability to integrate file handling with web-based user interactions and manage asynchronous tasks effectively using Streamlit’s interactive components. I am gaining practical experience in building user-friendly interfaces for AI applications, handling file operations, and enhancing AI model capabilities through dynamic content updates.