CS682 Project Report

Course Title: CS682 - Software Development Laboratory I, Fall 2024

Project Name: Development of an AI-Driven Chatbot

Team Members:

Sathvik Rao Nemmani – Frontend, Backend, Testing, Documentation
Saketh Tadepalli – Frontend, backend, research, Documentation.

GEIR/Department Information	
Company Name	**SCREEN WITH PEEK**
Industry	**HEALTHCARE, MEDICAL DEVICES AND DIAGNOSTICS.**
Company Contact (Name, Phone, Email)	Krispian Lawrence (Founder & CEO) **,** +1 617 852 6152 (US) **,** **Email:** contact@screenwithpeek.com
Brief Overview of Venture	PEEK is an AI-powered tool for easy and non-invasive colon screening. It uses a small capsule that you can swallow, which takes clear images of your gut and sends the information to an app on your phone. This makes it possible to do the screening comfortably at home. PEEK's goal is to help people detect problems early and make colon screening available to everyone.
Scope of Work	Development and distribution of the PEEK endoscopic capsule, a non-invasive, swallowable diagnostic device. Providing a AI chatbot which acts a first response to the customer who is in any sort of trouble understanding about the procedure (ex: Bowel Preparation) and other elements about the product.

1.EXECUTIVE SUMMARY:

The "Development of an AI-Driven Chatbot - ScreenWithPeek" project aims to help patients prepare better for medical procedures like capsule endoscopy. Many patients struggle to understand and follow preparation guidelines, especially related to diet, which can affect the success of the procedure. This project focuses on creating a chatbot that gives patients easy-to-understand, personalized, and real-time guidance, making it easier for them to follow the necessary steps.

The main goals of this project are:

1. **Build a Responsive Chatbot**: Create a chatbot that quickly answers patient questions about bowel preparation for the procedure, such as what to eat, what to avoid, and when to take medicine.

2. **Ensure Accurate Responses**: Use advanced language processing so the chatbot gives accurate and useful information.

3. **Make User Interaction Easy**: Design the chatbot to provide simple, step-bystep instructions that are easy for patients to follow.

The chatbot includes features like natural language processing for better understanding, dietary advice, and a user-friendly interface.

This solution provides patients with clear and reliable help, reducing confusion and improving how well they follow medical guidelines.

2.PROJECT BACKGROUND:

Medical procedures such as capsule endoscopy require patients to adhere to specific bowel preparation, often involving dietary restrictions and timing instructions. Adhering to these requirements is crucial to ensure the success of the procedure. However, patients frequently struggle to find reliable, clear, and consistent information about these preparations. Conflicting online resources and generalized advice can lead to confusion, non-compliance, and ultimately, compromised procedural outcomes. This project aims to address these challenges by leveraging AI technology to offer personalized and precise guidance, thus reducing patient uncertainty and enhancing adherence to medical protocols.

Project Goals:

The primary goal of this project is to develop an AI-driven chatbot capable of guiding patients through the preparation process for medical procedures such as Bowel Preparation. This bowel preparation is an important step before the actual colonoscopy process and it is a key step in the whole procedure.

The key objectives include:

- Designing a responsive chatbot that efficiently answers user questions about procedure preparations, such as dietary restrictions and the timing of laxatives.
- Integrating comprehensive dietary recommendations, safe recipes, and specific instructions into the chatbot's response system to provide personalized guidance.
- Utilizing Natural Language Processing (NLP) techniques to ensure the chatbot understands diverse queries and provides accurate, contextually relevant responses.

The main deliverables are a fully functional chatbot deployed on a web platform, a detailed report outlining the development process, and a presentation summarizing the outcomes, challenges, and future work

3. TEAM CONTRIBUTIONS:

Team Member	Role Contribution	Summary
Sathvik Rao Nemmani	Backend & Frontend Testing	Implemented data ingestion, chunking logic, and integrated the vector database for efficient retrieval. Handled prompt construction and ensured coherent user interactions.
Saketh Tadepalli	Backend & Frontend Research	Developed the Streamlit-based user interface and integrated the LLM response generation module with the retrieval chain. Research and data preparation.

4.PROJECT TIMELINE:

Requirements & Planning :
During the initial phase, we defined the project scope, gathered all relevant domain documents, selects the LLM model, and establishes the architectural blueprint for the retrieval-augmented generation pipeline.

Data Preparation :
The next step focuses on data ingestion and cleaning. Documents are processed, chunked, and embedded. A vector database is set up to store and index these embeddings for efficient retrieval

Model & Retrieval Setup :
In this stage, the chosen LLM is integrated, and the retrieval chain is created. Finetuning is performed if required, and initial end-to-end tests are run to ensure that the system can retrieve relevant context and generate coherent responses.

Frontend & API Integration :
A user-facing interface, such as a Streamlit app, is developed. This frontend is connected to the backend, which handles retrieval and LLM inference. Alternatively, a REST API can be implemented using Flask or FastAPI for third-party integrations.

Testing & Iteration :
Unit tests and integration tests are performed to validate each component's correctness and coherence. User acceptance testing with a sample audience helps refine responses, ensure usability, and confirm that the chatbot meets domainspecific requirements.

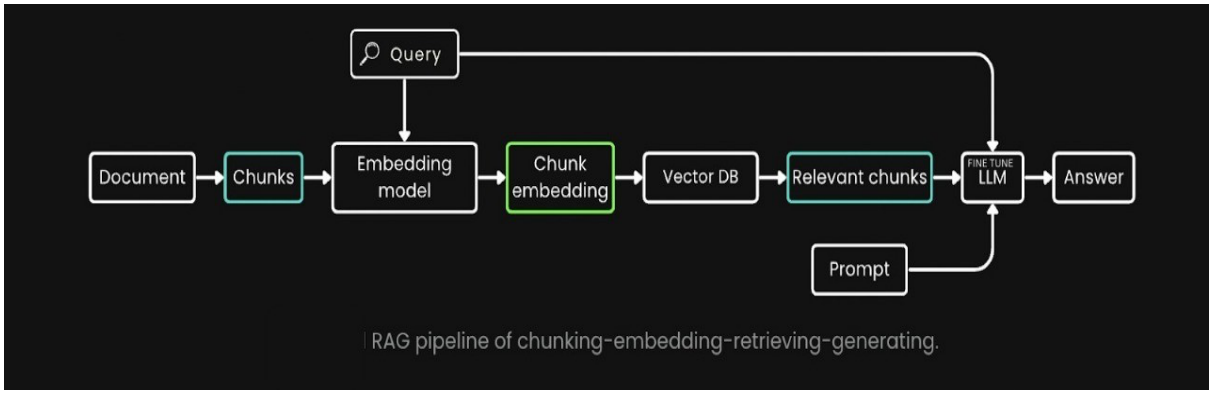
Deployment & Documentation:
The final stage involves deploying the system to the chosen environment, whether GPU-enabled or CPU-based. Documentation is finalized, including instructions for maintenance and updates, ensuring a smooth handover to end-users and stakeholders

5.TECHNICAL IMPLEMENTATION

Technology Stack

- **Programming Language:** Python (for backend logic and local LLM integration), JavaScript (Streamlit components).
- **Frameworks and Libraries:**
    - **Streamlit:** For building the user interface. ○ **LangChain:** For retrieval-augmented generation (RAG) pipeline and query handling.
    - **PyPDF2, pandas:** For document ingestion and preprocessing (PDF and CSV).
    - **SentenceTransformers:** For embedding generation using the BAAI/bge-base-en-v1.5 model.
    - **ChromaDB:** For vector database management and semantic search.
    - **Pyngrok:** For exposing the local deployment to the internet.
- **Models:** Locally fine-tuned LLM for endoscopy-related queries.
- **Development Environment:** Kaggle notebooks for initial implementation and testing.

6.ARCHITECTURE OVERVIEW



RAG pipeline of chunking-embedding-retrieving-generating.

1. DATA PREPARATION:

- The dataset for this project includes a combination of structured questions and answers related to endoscopy procedures, focusing on patient preparation, dietary restrictions, medication, and procedural guidance. we jotted down about 300 questions on various sections of the capsule colonoscopy and focused more on bowel preparation which is an important step in the endoscopy process.

- Preprocessing:

  The preprocessing pipeline involved: I. **Text Cleaning**: ○ Removed extraneous spaces, special characters, and inconsistent formatting.

  ○ Ensured uniform formatting for conversational and structured responses.

  II. **Standardization**:

  ○ Converted dataset content into a structured prompt-response format compatible with the fine-tuning requirements of the LLM

```
<s>[INST] <<SYS>>
System prompt
<</SYS>>

User prompt [/INST] Model answer </s>
```

- Challenges:

  Managing repeated questions or overlapping topics (e.g., dietary restrictions for different days) while ensuring concise, non-redundant content in the final dataset

  Truncated responses and system prompts dynamically to ensure they fit within the LLM's maximum token limit without losing critical information.

2.DATA CHUNKING:

- **Function**: chunk_documents.
- **Process**:
  - Splits large documents into smaller, overlapping chunks to facilitate better embedding and retrieval.
  - Uses the CharacterTextSplitter to maintain logical divisions while ensuring overlap for context.
- **Output**: A list of smaller text chunks.

3. EMBEDDING MODEL:

- Function: load_embedding_model.
- Process:
  - Loads a pre-trained embedding model (BAAI/bge-base-en-v1.5) using Lang Chain.
  - Converts each text chunk into a vector representation (embedding).
  - In RAG pipeline of the chatbot, the embedding model plays a critical role in converting the document chunks and user queries into comparable formats, enabling the vector database to find the most relevant information. ○ The embedding model converts both the document chunk and the user query into numerical vectors. For example: Document Chunk: [0.12, 0.78, -0.45, 0.33, ...] User Query: [0.15, 0.79, -0.43, 0.34, ...]

- The individual token embeddings are combined into a single embedding for the entire input sequence.
- Methods include:

  Mean Pooling: Take the average of all token embeddings.
  CLS Token: Use the special [CLS] token embedding from transformer-based models.

- Example:
  Sequence Embedding:
  Mean ([0.2, 0.4, 0.8, -0.1], [0.1, 0.6, 0.7, -0.3], [0.9, -0.2, 0.4, 0.3])
  = [0.4, 0.26, 0.63, -0.03]

4. VECTOR DATABASE CREATION:

- **Function**: create_vector_db.
- **Process**:
  - Stores embeddings in a vector database using Chroma.
  - The vector database allows fast and accurate similarity search. ○ The Chroma library is used to create and manage the vector database. ○ Each embedding is stored in the database, along with metadata (e.g., the chunk's text, file ID, or other identifiers).
  - The persist() method ensures that the vector database is saved to disk at the specified VECTOR_DB_PATH.
  - Indexing is performed to make retrieval fast and scalable.
  - Implements **Maximum Marginal Relevance (MMR)**, which ensures diversity in retrieved chunks by balancing relevance and novelty. It ensures that the returned items are not only highly relevant to the query but also provide varied information, avoiding redundancy.

Example:

Input:

- Query: "What is endoscopy?"
- Candidate Items (chunks):
    1. "Endoscopy is a procedure to examine the digestive tract."
    2. "Endoscopy is commonly used for diagnosing stomach issues."
    3. "An endoscope is a tool used in endoscopy procedures."
    4. "Endoscopy techniques have evolved in recent years."

Without MMR:

A traditional similarity-based approach might return:

- "Endoscopy is a procedure to examine the digestive tract." ○ "Endoscopy is commonly used for diagnosing stomach issues." ○ Both are relevant, but they overlap significantly in content.

With MMR (λ=0.7\lambda = 0.7λ=0.7):

MMR ensures: ○ Select the most relevant chunk first: "Endoscopy is a procedure to examine the digestive tract."

- Then choose a more diverse chunk: "An endoscope is a tool used in endoscopy procedures."

In chatbot's RAG pipeline, **Chroma** uses MMR during the retrieval process: ○ It retrieves the top K chunks based on the query embedding. ○ Ensures that the selected chunks are both relevant to the query and diverse.

The retrieval code in your pipeline specifies:

```
retriever = vectorstore.as_retriever
(
   search_type="mmr",
   search_kwargs={'k': 4}
)
```

search_type="mmr": Enables MMR for chunk retrieval.

k=4: Retrieves 4 chunks, balancing relevance and diversity.

- **Output**: A vector database persisted in a directory (VECTOR_DB_PATH).

5.LARGE LANGUAGE MODEL:

- First the LLM loads a local model **(HuggingFaceH4/zephyr-7b-beta**) using HuggingFace's AutoModelForCausalLM and AutoTokenizer.
- Sets up the tokenizer and model so that the LLM can be called upon to generate text.
- This initialization step ensures that the LLM is ready to process text input and generate outputs. Without this, no language generation could occur.
- Generating Responses from Prompts

  Code Reference: The generate_response method

Process:

- Takes a list of messages (system prompt, context, user query) and converts them into a model-compatible format.
- Encodes these messages using the tokenizer to produce input tensors.
- Calls the model.generate function with parameters like max_length, temperature, and do_sample to produce tokens for the answer. ○ Decodes the generated tokens back into human-readable text. ○ Extracts the final response from the generated string, removing special tokens and other prompt artifacts.

```
def generate_response (self, messages,
max_length=600, temperature=0.8,
do_sample=True):

 # Prepares the prompt and uses
self.model.generate to produce an answer  #
Decodes the output and returns the final
response string
```

This is the core function of the LLM—transforming the given prompt (including instructions, user questions, and context) into a coherent, contextually relevant answer. By adjusting parameters like max_length and temperature, we influenced how long and how creative the LLM's answer is.

- Currently, the model uses fixed response-generation parameters (max_length=600, temperature=0.8, num_beams=5). These values were chosen for balanced response quality, coherence, and diversity.

  However:

- Dynamic adjustment can be implemented by analyzing query complexity, such as: ○ Longer queries or highly technical questions triggering increased max_length or reduced temperature.
- Ambiguous queries lowering num_beams for faster generation.
- This can be achieved by introducing a lightweight complexity scoring mechanism (e.g., token count, presence of medical terms).
- If the combined input (context + query) is too large, the system enacts truncation before feeding it to the LLM. Within the generation call, `max_length` ensures that the model does not produce overly long responses, thus preventing excessive computational cost and staying within the model's operational limits.

6.MODEL INFERENCE:

In the chatbot project, **model inference** refers to the process of generating an appropriate, contextually grounded answer to a user's query using the trained LLM and the retrieval-augmented pipeline. Here's how inference unfolds step-bystep within this particular system:

1. **User Query Intake**:
   The inference process begins when the user enters a query into the chatbot interface.
   For example, the user might ask:
   "What should I eat the day before an endoscopy?"

2. **Query Embedding and Retrieval**:
   Before the LLM is consulted, the user's question is converted into an embedding, a numerical representation capturing its semantic meaning. This embedding is then used to query the vector database, which stores embeddings of various document chunks. The system retrieves the top relevant chunks—text snippets that contain instructions, dietary guidelines, or explanations related to the query. These retrieved snippets provide external knowledge and help ground the LLM's final answer in factual, domain-specific information.

3. **Prompt Construction**:
   With the user's query and relevant retrieved chunks in hand, the system constructs a prompt. This prompt combines: ○ **System Instructions**: High-level guidelines defining the tone, style, and role of the LLM as a medical assistant.
   ○ **Contextual Snippets**: The retrieved document chunks that are directly relevant to the user's question. ○ **User Query**: The question itself, which the user initially asked.

7.CHALLENGES FACED:

Technical Challenges

1. **Natural Language Understanding (NLU) Complexity** ○ **Challenge**: Designing the chatbot to understand diverse user inputs, including typos, slang was difficult.

    - **Resolution**: Used a embedding model to convert the documents into embeddings where we used RAG Architecture and the prompt that we pass to the LLM to understand the intent of the user query.

2. **Fine-Tuning Without GPUs** ○ **Challenge**: Training and fine-tuning the NLP model on a large dataset without access to high-performance GPUs was time-consuming and resource intensive.

    - **Resolution**:
        - **Parameter Tuning**: Reduced the model's complexity by optimizing parameters and using techniques like mixedprecision training and quantization to reduce resource demands.
        - **Kaggle**: Utilized the Kaggle free GPUs provided and trained the model.

3. **Data Privacy Compliance** ○ **Challenge**: Ensuring adherence to data privacy regulations (HIPAA, GDPR) for handling sensitive user information.

    - **Resolution**:
        - Integrated encryption for data in transit and at rest.
        - Employed data anonymization techniques to protect user identity.
        - Regularly audited compliance with regulations, seeking guidance from legal experts to refine privacy protocols.

Non-Technical Challenges

1. **Domain Knowledge**

   - **Challenge**: Understanding medical terminology and workflows related to endoscopy and bowel preparation.

     - **Resolution**:
       - Collected research papers and read online articles with medical professionals to build a comprehensive knowledge base.

2. **User Feedback Collection**

   - **Challenge**: Gathering feedback from a diverse set of users, including patients and medical staff, was logistically challenging.

     - **Resolution**:
       - Conducted user surveys through online forms and in-app feedback prompts.
       - Organized focus groups and pilot testing sessions to capture insights and actionable suggestions.

3. **Team Coordination**

   - **Challenge**: Ensuring effective communication between developers, medical experts, and UX designers.

     - **Resolution**:
       - Adopted Agile methodology with daily standups to align progress and priorities.
       - Used tools like Jira and Slack for task management and realtime communication.

4. **Time Constraints**

   - **Challenge**: Balancing tight deadlines with the need for a high-quality product.

     - **Resolution**:
       - Prioritized deliverables based on a clear roadmap, focusing on high-impact features.

Streamlined workflows by automating repetitive tasks such as testing and data preprocessing.

8.FUTURE WORK AND RECOMMENDATIONS:

Improvements

With more time and resources, several enhancements could be made to improve the chatbot's performance and functionality:

1. **Expanded Dataset**: Incorporate a larger, more diverse dataset for Natural Language Processing (NLP) training to handle a broader range of patient inquiries and medical procedures.
2. **Multi-Language Support**: Add multi-language support to cater to non-English speaking users, enhancing the accessibility of the chatbot.
3. **Integration with Health Systems**: Integrate the chatbot with Electronic Health Record (EHR) systems to personalize recommendations further based on the patient's medical history and specific needs.
4. **Voice Interaction**: Implement voice interaction capabilities to make the chatbot more user-friendly, especially for elderly patients or those with limited typing abilities.
5. **Advanced Error Handling**: Improve error-handling mechanisms to manage complex or ambiguous questions, ensuring the chatbot remains helpful even when encountering unexpected input.

Recommendations for Future Teams

1. **User Testing and Feedback**: Conduct extensive user testing early on, involving diverse patient groups to identify usability issues and better tailor the chatbot to real-world scenarios.
2. **Collaboration with Medical Professionals**: Work closely with healthcare providers to ensure the chatbot's responses are accurate, up-to-date, and compliant with medical standards.
3. **Scalable Architecture**: Design a scalable backend that can support increased user traffic and easy updates, allowing the chatbot to grow and adapt as more features are added.
4. **Continuous Learning Integration**: Implement a continuous learning mechanism for the chatbot, where feedback from users can be incorporated into ongoing NLP training, allowing the system to improve over time.
5. **Focus on Privacy and Security**: Given the sensitive nature of medical information, prioritize privacy and security throughout the project, including data encryption, user consent management, and compliance with healthcare regulations like HIPAA.

Future teams should build on these suggestions to create a more robust and accessible tool that effectively supports patient preparation for medical procedures.

9.CONCLUSION:

Project Journey and Accomplishments

The development of the AI-driven chatbot for medical procedure preparation was a comprehensive journey that involved multiple stages, each contributing significantly to the final outcome. The project began with an understanding of the challenges faced by patients in adhering to complex pre-procedural guidelines. Motivated by these challenges, the team set out to create a solution that could provide personalized, accurate, and user-friendly guidance to patients.

The initial phase involved extensive research on the specific requirements of medical preparation procedures and gathering datasets to be used for Natural Language Processing (NLP) training. This led to the development of a robust chatbot prototype capable of understanding patient inquiries regarding dietary restrictions, timing for medications, and other preparation steps for capsule endoscopy. The chatbot leveraged NLP models to provide personalized responses, ensuring that patients received clear and actionable advice tailored to their needs.

Major accomplishments include:

1. **Successful Implementation of NLP**: The integration of NLP capabilities enabled the chatbot to understand and respond effectively to diverse user queries. This significantly enhanced the chatbot's ability to provide accurate and relevant information.
2. **Comprehensive Data Integration**: The chatbot was equipped with a comprehensive database of dietary guidelines, recipes, and medical instructions to offer personalized recommendations.
3. **User-Centric Design**: The creation of an intuitive user interface ensured a smooth interaction experience, making the chatbot accessible and easy to use for patients of all backgrounds.
4. **Rigorous Testing and Evaluation**: Extensive testing phases were conducted to refine the chatbot’s accuracy and reliability, incorporating user feedback to improve performance continuously.

The journey from concept to completion was marked by challenges, such as ensuring the chatbot’s responses were both accurate and comprehensible, especially given the critical nature of medical advice. Despite these challenges, the project achieved its primary objective: developing a reliable and effective tool to guide patients through their medical preparations, ultimately improving patient compliance and outcomes. The accomplishments made during this project provide a strong foundation for future advancements and expansions.

10.APPENDIX:

Here is the github repository link which includes the required files:
https://github.com/Sathvikrao-5/ChatBot

references:
- American Society for Gastrointestinal Endoscopy (ASGE) Guidelines
https://www.asge.org
- Mayo Clinic Endoscopy Procedures Overview
https://www.mayoclinic.org