EGR 404 Project Proposal

1. Name of Project Partners

   Suparna Veeturi (Working solo)

2. Title of the Project

   Conversational AI Assistant for IRB Submissions

3. Clear Description of the Project

   Problem:

   Preparing IRB submissions at URI involves extensive paperwork, including the main Application, Consent forms, and various supporting documents. Adhering to specific language and formats while managing multiple projects is time-consuming.

   Solution:

   This project will create an interactive chatbot assistant using generative AI. The chatbot will streamline IRB package preparation by leveraging past examples and institutional templates to suggest relevant content, while still requiring the researcher to provide the core, unique protocol details. It will also assist in structuring the research flow and managing documents per project.

   Core Features:

   Example-Based Suggestions: The chatbot will ask the user for the overall project type (e.g., BME device study, survey, etc.) and the core research protocol. It will then query its knowledge base (containing URI templates and user-uploaded past IRB packages) to retrieve and suggest relevant standard language, phrasing, and required sections for the Application and Consent forms based on similar, successful past projects.

   Conversational Refinement: Users can interact with the chatbot to refine the suggested text or request alternatives based on the examples in the knowledge base.

   User-Extensible Knowledge Base: Users can continuously upload more approved IRB packages (Applications, Consents, supporting docs). The tool will process these, allowing the system to provide increasingly relevant and tailored suggestions over time based on a larger corpus of examples.

Supporting Document Guidance: Based on the protocol and project type, the chatbot will identify potentially required supporting documents (e.g., recruitment flyers, screening forms) and can assist in drafting standard components of these.

Protocol Flow Outlining: Assists the user in structuring their research methodology by generating a textual outline or sequence of steps based on the provided protocol, suitable for guiding the creation of a protocol diagram.

Project-Based Context Management: Stores conversation history, the core protocol, generated drafts, and uploaded example packages separately for each research project, enabling easy context switching and iterative work.

Guideline Q&A: Allows users to ask questions about URI IRB guidelines (loaded into the vector store).

Impact & Innovation: By leveraging successful past examples and automating the generation of standard text components, this tool aims to drastically reduce drafting time, improve consistency, and lower the barrier to initiating research, while keeping the researcher in control of the unique scientific content.

4. Resources to Use

- Development Environment: Python, VS Code, venv.
- AI Models & APIs: OpenAI API (GPT-4o-mini or similar).
- Key Libraries/Frameworks:
    - openai Python library.
    - Vector store libraries (e.g., FAISS, ChromaDB) and embedding models.
    - Python libraries for parsing uploaded documents (PDFs, DOCX - e.g., PyPDF2, python-docx).
    - Python framework for chatbot interface (e.g., Streamlit, Gradio).
    - Libraries/methods for managing project state/data persistence.
- Data Sources:
    - Primary Templates: Downloaded from irbnet.gov.
    - Core Knowledge: User-uploaded examples of previously approved URI IRB packages (Applications, Consents, supporting docs).
    - Official University of Rhode Island IRB guidelines (to be loaded into vector store).
    - User's specific research protocol details (provided as input).
- Relevant Class Lab Assignments:
    - Vector store implementation and document uploading (Lab 6).

- Advanced prompt engineering for suggestion generation and protocol outlining (Lab 4, Lab 5).
- Python environment and modular design (Lab 5).
- Chatbot/UI concepts (Lab 6).

5. Deliverables

Functional MVP Chatbot Prototype: A demonstrable chatbot application showcasing the core suggestion functionality for selected key sections of the URI Consent Form(e.g., Risks, Procedures, Confidentiality) based on uploaded examples.

Core Code Repository: A GitHub repository containing the documented Python code for the prototype, including document parsing, vector store interaction, LLM calls, and basic project state handling.

Basic User Documentation: A README file in the GitHub repository explaining setup, how to upload initial examples, and how to run the core suggestion feature for the targeted sections.

Timeline:

- Week 1: Setup & Backend Core:
  - Setup Python environment, install core libraries (openai, vector store lib, PyPDF2/python-docx).
  - Implement and test document parsing (PDF/DOCX) for the templates and examples. Implement vector store population (indexing the extracted text).
  - Develop the core Python function to query the vector store based on input keywords/type and retrieve text snippets for the single target section. Test basic retrieval.

  Goal: Command-line script can load docs and retrieve relevant text snippets for one section.
- Week 2: Output & Cleanup:
  - Refine the retrieval logic slightly. Implement the code to simply print the retrieved snippets clearly to the console. (Optional: Add one basic LLM call to perhaps rephrase the snippets slightly).
  - Clean up the scripts, ensure they run reliably from the command line.
  - Write the basic README, prepare a quick command-line demo video or description.

  Goal: Demonstrable command-line proof-of-concept for snippet retrieval.

Demo plan: Demonstrate the ability to load IRB documents (templates + examples) into a vector store and retrieve relevant text snippets for one specific section based on project type/keywords via a command-line interface.

6. GitHub Account & Sharing

A GitHub repository will be created.

Access shared with sendag@uri.edu and justin_watkins@uri.edu.

README file (items 1–5) included.

Link to GitHub Repository will be shared!