**AI Intern Assignment: RAG-Based Chatbot Development**

**📋 Assignment Overview**

Develop a **Retrieval-Augmented Generation (RAG) based chatbot** that can answer queries from multiple knowledge sources while maintaining conversational capabilities for general questions.

**🎯 Objective**

Create a context-aware chatbot that can:

1. Answer general conversational queries using its base knowledge
2. Provide specific information from NEC code guidelines when asked
3. Answer company-specific questions using Wattmonk information
4. Seamlessly switch between different contexts based on user intent

**📊 Data Sources**

You will work with two primary knowledge bases:

* **NEC Code Guidelines**: Technical electrical code standards and regulations
* **Wattmonk Company Information**: Company policies, services, and internal documentation

**🛠️ Technical Requirements**

**Frontend Options (Choose One)**

* **React.js**: For custom UI development
* **Streamlit**: For rapid prototyping and deployment
* **AI-based tools**: Lovable, Firebase, or similar platforms
* **Alternative**: Any modern frontend framework

**Backend Options (Choose One)**

* **Django**: Python-based web framework
* **FastAPI**: Modern, fast Python API framework
* **Node.js**: JavaScript runtime for backend development

**AI/ML Models**

* **Requirement**: Use **API-based models only** (no self-hosted models)
* **Suggested Providers**:
  + OpenAI (GPT-3.5/GPT-4)
  + Anthropic (Claude)
  + Google (Gemini)
  + Cohere
  + Hugging Face Inference API

**Vector Database Options**

* Pinecone
* Weaviate
* Chroma
* FAISS (with cloud storage)

**🏗️ Architecture Requirements**

**RAG Pipeline Components**

1. **Document Processing**
   * Text extraction from PDF/document files
   * Text chunking and preprocessing
   * Metadata extraction
2. **Vector Store**
   * Document embeddings generation
   * Similarity search implementation
   * Context retrieval optimization
3. **Query Processing**
   * Intent classification (general vs. domain-specific)
   * Context-aware response generation
   * Source attribution
4. **Response Generation**
   * Context injection into prompts
   * Multi-turn conversation handling
   * Fallback mechanisms

**✅ Functional Requirements**

**Core Features**

* [ ] **Multi-Context Handling**: Distinguish between NEC, Wattmonk, and general queries
* [ ] **Source Attribution**: Clearly indicate information sources
* [ ] **Conversation Memory**: Maintain context across multiple exchanges
* [ ] **Fallback Responses**: Handle queries outside knowledge base gracefully
* [ ] **Search Functionality**: Allow users to search specific topics

**Advanced Features (Bonus)**

* [ ] **Query Refinement**: Suggest related questions
* [ ] **Document Citations**: Link to specific document sections
* [ ] **Confidence Scoring**: Display confidence levels for responses
* [ ] **Multi-language Support**: Handle queries in different languages

**🚀 Deployment Requirements**

**Hosting Platforms (Choose One)**

* **Hugging Face Spaces**
* **Streamlit Cloud**
* **Railway**
* **Vercel**
* **Heroku**
* **Google Cloud Run**

**Deployment Checklist**

* [ ] Working deployed application with public URL
* [ ] Environment variables properly configured
* [ ] API keys secured (not exposed in code)
* [ ] Responsive design for mobile devices
* [ ] Error handling and user feedback

**📁 Deliverables**

**1. Source Code**

* **GitHub Repository**: Complete, well-organized codebase
* **ZIP File**: Backup of the entire project
* **Requirements**: requirements.txt or package.json
* **Configuration**: Environment setup instructions

**2. Documentation**

* **README.md**: Project setup and running instructions
* **API Documentation**: If backend APIs are created
* **Architecture Diagram**: System design overview
* **User Guide**: How to use the chatbot

**3. Deployment**

* **Live Demo**: Working chatbot accessible via public URL
* **Performance Metrics**: Response times, accuracy measures
* **Usage Examples**: Screenshots/videos of the chatbot in action

**📝 Evaluation Criteria**

**Technical Implementation (40%)**

* Code quality and organization
* RAG pipeline effectiveness
* Error handling and edge cases
* API integration and security

**Functionality (30%)**

* Multi-context query handling
* Response accuracy and relevance
* User experience and interface design
* Conversation flow management

**Documentation & Deployment (20%)**

* Code documentation quality
* Setup and deployment instructions
* Architecture clarity
* User guide completeness

**Innovation & Extras (10%)**

* Creative features or improvements
* Performance optimizations
* Advanced RAG techniques
* UI/UX enhancements

**🎯 Success Metrics**

**Minimum Viable Product (MVP)**

* Chatbot can distinguish between general and domain-specific queries
* Provides relevant answers from both knowledge bases
* Successfully deployed with public access
* Basic documentation provided

**Excellent Implementation**

* Sophisticated context switching
* High response accuracy and relevance
* Polished user interface
* Comprehensive documentation
* Advanced features implemented

**📋 Submission Guidelines**

**Required Submissions**

1. **GitHub Repository URL**
2. **ZIP file of complete project**
3. **Deployed Application URL**
4. **Documentation (README + User Guide)**
5. **Brief video demo (2-3 minutes)**

**Submission Format**

Subject: AI Intern Assignment Submission - [Your Name]

GitHub Repository: [URL]

Deployed Application: [URL]

Demo Video: [URL/Attachment]

Additional Notes: [Any important information]

**🔧 Technical Hints**

**RAG Implementation Tips**

* Use appropriate chunk sizes (typically 200-500 tokens)
* Implement hybrid search (semantic + keyword)
* Consider query expansion techniques
* Use context window efficiently

**Performance Optimization**

* Cache frequently accessed embeddings
* Implement request rate limiting
* Optimize API call patterns
* Use streaming for real-time responses

**Security Best Practices**

* Never hardcode API keys
* Implement input sanitization
* Use HTTPS for all communications
* Validate user inputs

**📚 Recommended Resources**

**RAG & LLM Resources**

* LangChain documentation
* OpenAI API guides
* Vector database tutorials
* RAG evaluation metrics

**Development Resources**

* React/Streamlit tutorials
* FastAPI/Django documentation
* Deployment platform guides
* Git best practices

**❓ FAQ**

**Q: Can I use multiple LLM providers?** A: Yes, but ensure consistent performance across providers.

**Q: What if the provided documents are too large?** A: Implement intelligent chunking and summarization strategies.

**Q: How should I handle API rate limits?** A: Implement proper error handling and user feedback for rate limit scenarios.

**Q: Can I add additional features not mentioned?** A: Absolutely! Innovation and additional features are encouraged.

**Good luck with your implementation! We're excited to see your creative approach to this RAG chatbot challenge.**