**Project Proposal**

**“Doc Assist: A Domain-Specific Question- Answering System for Software Industry Documentation Dissertation”.**

**Name**: Amadi Bright  
**SID**:   
**Email**: @uni.coventry.ac.uk  
**Supervisor**: Micheal England

**“Doc Assist: A Domain-Specific Question- Answering System for Software Industry Documentation Dissertation”.**

☐ **I submitted my ethics application, and my application has been approved. I include my ethics certificate in the appendix as evidence.**

☑ **I submitted my ethics application, and my application is currently under review.**

☐ **I have not submitted my ethics application**

**Section A – Ethics Application**

**Section B – Project Proposal**

This project proposes the development of **Doc Assist**, a domain-specific question-answering (QA) system designed to help software developers find accurate and context-aware answers within technical documentation efficiently. Current keyword-based search tools are inadequate for navigating complex software documentation, as they fail to understand user intent, code syntax, and API-specific terminology. This leads to significant productivity loss, as evidenced by industry surveys like the Stack Overflow Developer Survey (2023), which indicates developers spend up to 30% of their time searching for information. Doc Assist addresses this gap by leveraging modern natural language processing techniques within a Retrieval-Augmented Generation (RAG) architecture to retrieve and generate precise answers paired with relevant code snippets. The system will be specifically tailored to the software domain, enabling it to interpret technical queries and provide responses grounded in official documentation and validated sources.

The core of this research lies in its data-driven methodology, utilizing the **StaQC dataset**, the largest available corpus of semantically aligned question-code pairs mined from Stack Overflow. A streamlined RAG architecture will be implemented to ensure answers are both accurate and contextually relevant. System performance will be rigorously evaluated against the **CoNaLa benchmark**, a human-annotated dataset, using metrics such as BLEU, ROUGE, and F1 scores. The final deliverable will be a functional prototype demonstrating measurable improvements in retrieval accuracy and answer quality over traditional search methods, validating the practical value of domain-specific QA systems for software engineering.

**1. Research Question, Problem Statement, or Topic for Investigation**

**Research Question**

What is the most effective methodology for designing and evaluating a domain-specific question-answering (QA) system to optimize the retrieval of accurate and contextually relevant answers from software documentation for developers? How does the quality and composition of training data (e.g., web-mined versus human-annotated datasets) directly impact on the performance and accuracy of such a domain-specific QA system?

**Problem Statement**

Software developers spend a significant portion of their time up to 30% searching through documentation for solutions to technical challenges. Existing tools often rely on simplistic keyword matching, which fails to capture contextual nuances, programming intent, or semantic relationships between queries and content. This inefficiency leads to prolonged development cycles, reduced productivity, and potential implementation errors. General-purpose QA systems are ill-suited for software-specific queries due to the highly technical nature of programming languages, API structures, and domain-specific terminology. This project addresses these gaps by developing a specialized QA system tailored to the unique demands of software documentation.

**Topic for Investigation**

The focus of this research is the design, implementation, and evaluation of a domain-specific QA system for software documentation, leveraging curated datasets such as **StaQC** and **CoNaLa**. The investigation will explore the role of data quality and relevance in training effective models, assess the applicability of retrieval-augmented generation (RAG) architectures in technical contexts, and measure improvements in accuracy, efficiency, and user satisfaction compared to conventional search methods.

**2. Intended User**

The primary intended user group for this research is software developers, who face significant information retrieval challenges due to the limitations of existing tools such as keyword-based documentation search engines (e.g., Elasticsearch-powered platforms), general-purpose web search engines, and non-specialized conversational agents. These tools lack the semantic and contextual understanding necessary for processing technical queries, often resulting in inefficient or inaccurate responses. This study addresses these shortcomings by developing a domain-specific question-answering system tailored to the precise needs of developers, with the goal of enhancing both the accuracy and efficiency of accessing complex technical documentation.

**3. Systems Requirements, Project Deliverables, and Final Project Outcome**

**3.1 Systems Requirements:**

**Software & Libraries:** The development will be conducted in Python, utilizing key data science and NLP libraries including PyTorch (for model training and fine-tuning), Transformers (Hugging Face) for pre-trained models and tokenization, sentence-transformers and FAISS for efficient semantic search and retrieval, and NLTK/spaCy for text preprocessing. Version control will be managed via Git.

**Hardware:** Model training and evaluation require GPU acceleration to handle transformer-based architecture efficiently. An NVIDIA GPU (e.g., V100, A100) with CUDA support is essential for optimizing training time and inference performance.

**Data:** The core datasets include:

**• StaQC**: For large-scale, domain-specific question-code pairs.

**• CoNaLa**: For high-quality, human-annotated intent-code examples, used for validation and testing.

• A custom-curated subset of software documentation (e.g., from GitHub or API docs) to enhance domain relevance.

**3.2 Project Deliverables:**

**Data Focus**

**• Process:** Immediately acquire and preprocess a focused subset of the **StaQC** dataset (e.g., 10k Python Q&A pairs). Clean, tokenize, and structure the data for modeling. Concurrently, prepare the **CoNaLa** test set for evaluation.

**• Verifiable Deliverable:** A clean, ready-to-use .csv dataset and a script that reproduces the entire preprocessing pipeline.

**3.3 Final Project Outcome:**

• A validated, domain-specific QA system prototype that demonstrates measurable improvements in accuracy and relevance for software documentation queries, contributing to both developer productivity and academic research on data-quality-aware NLP systems.

**4. Primary Research Plan**

This plan outlines a streamlined, data-centric research methodology designed to be executed by a single researcher to enhance the existing QA system.

**1. Data Collection:**  
The foundation of this work is the strategic use of existing, high-quality datasets to ensure efficiency and validity. The primary data sources are:

**• StaQC (Python):** Used as the main source for training data. A focused subset of approximately 10,000 high-quality question-code pairs will be systematically extracted based on code completeness and question clarity.

**• CoNaLa:** Serves as the gold-standard test set for evaluation. Its human-annotated quality provides a reliable benchmark for measuring performance improvements.

**2. Feature Engineering:**  
The focus will be on preparing the text and code data for the model:

**• Text Preprocessing:** Question text will be cleaned through tokenization, lowercasing, and removal of non-alphanumeric characters.

**• Code Normalization:** Code snippets will be processed by parsing and standardizing formatting (e.g., unifying variable names where possible, removing comments) to reduce noise and help the model learn semantic patterns rather than stylistic variations.

**• Embedding Generation:** For the retrieval component, text and code will be converted into dense vector representations using a pre-trained sentence transformer model to enable semantic similarity search.

**3. Model Development:**  
The approach prioritizes leveraging pre-trained models for efficiency:

**• Architecture:** A Retrieval-Augmented Generation (RAG) pipeline will be implemented. This involves a **retriever** (using FAISS for efficient similarity search on pre-computed embeddings) and a **generator** (a fine-tuned CodeT5 or T5 model to generate answers conditioned on the retrieved context).

**• Training:** The generator model will be fine-tuned on the curated StaQC training subset. Hyperparameter tuning will be limited to a few key parameters (e.g., learning rate, number of epochs) to optimize performance within a feasible timeframe.

**6. Evaluation:**  
Performance will be measured rigorously against the baseline system:

**• Quantitative Metrics:** The primary evaluation will use standard NLP metrics on the CoNaLa test set, including **BLEU** (for answer fluency), **ROUGE** (for recall of key information), and **F1 Score** (for accuracy). The goal is a measurable improvement over the baseline scores.

**• Qualitative Analysis:** A sample of outputs will be manually inspected to identify common failure modes and areas for future improvement, providing context to the quantitative results.

**5. Initial/Mini Literature Review**

Domain-specific question answering (QA) for software documentation sits at the intersection of natural language processing (NLP) and software engineering. Early work by Tian et al. (2017) established the value of domain focus with APIBot, a system designed for API documentation that demonstrated superior performance over general-purpose QA models using only a small, curated dataset of 92 pairs. This foundational work underscores a critical insight: quality and specificity of data can outweigh sheer volume.

Subsequent research has emphasized scaling data collection while grappling with quality challenges. Yao et al. (2018) created StaQC, the largest dataset of its time, by automatically mining question-code pairs from Stack Overflow. While invaluable for scale, automatically mined data like StaQC inherently contains noise and misaligned pairs, which can limit model performance. Conversely, CoNaLa (Yin et al., 2018) offered a smaller but human-annotated benchmark, highlighting the trade-off between scale and precision. This tension was further explored by Bansal et al. (2021), who demonstrated that carefully compiled, high-quality data triples significantly outperformed larger but noisier datasets in a neural QA system for code subroutines.

The evolution of model architecture has also been pivotal. The shift towards pre-trained transformer models like BERT and T5 revolutionized general NLP tasks, but their application to technical domains requires adaptation. More recently, Chen et al. (2023) showcased the effectiveness of a specialized QA system, QAssist, for software requirements analysis, significantly outperforming general-purpose models. Their work confirms that domain-specific fine-tuning of modern architecture is essential for high performance. The Retrieval-Augmented Generation (RAG) paradigm has emerged as a powerful framework for knowledge-intensive tasks, though its application to software documentation remains underexplored.

Despite these advances, significant research gaps persist. First, there is a lack of systematic studies that explicitly measure the *causal impact* of training data quality on model performance within this domain. Second, many existing systems are evaluated solely on automated metrics, lacking validation through user studies with actual software developers to assess practical utility. This research directly addresses these gaps. It will quantify the performance gain achieved by curating a high-confidence subset of StaQC, apply a RAG architecture specifically tailored for software documentation, and extend evaluation beyond standard metrics.

**6. Bibliography**

• Bansal, A., Eberhart, Z., Wu, L., & McMillan, C. (2021). A neural question answering system for basic questions about subroutines. *Proceedings of the 43rd International Conference on Software Engineering (ICSE)*, 1-12.  [PDF] A Neural Question Answering System for Basic Questions about Subroutines | Semantic Scholar

• Chen, S., Abualhaija, S., Arora, C., Briand, L. C., & V, F. (2023). AI-based question answering assistance for analyzing natural-language requirements. *Proceedings of the 45th International Conference on Software Engineering (ICSE)*, 1-13. https://arxiv.org/pdf/2302.04793

• Tian, Y., Thung, F., Sharma, A., & Lo, D. (2017). APIBot: Question answering bot for API documentation. \*2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)\*, 153-158. https://doi.org/10.1109/ASE.2017.8115628

• Yao, Z., Weld, D. S., Chen, W. P., & Sun, H. (2018). StaQC: A systematically mined question-code dataset from Stack Overflow. *Proceedings of the 27th International Conference on World Wide Web*, 1693-1703. https://doi.org/10.1145/3178876.3186081

• Yin, P., Deng, B., Chen, E., Vasilescu, B., & Neubig, G. (2018). Learning to mine aligned code and natural language pairs from Stack Overflow. *Proceedings of the 15th International Conference on Mining Software Repositories*, 476-486. https://doi.org/10.1145/3196398.3196408