Domain-Specific AI Assistant Project Plan

Course: ITAI 2377 – Applied Artificial Intelligence

Group Name: Yunze Wu

Submission Date: 11/10/2025

Team Members: Yunze Wu

---

1. Executive Summary

This project aims to develop an intelligent Educational Assistant powered by Few-Shot Learning, designed to provide personalized and interactive learning support. The assistant uses a chat-based interaction model, allowing students to communicate naturally with the AI—asking questions, receiving explanations, and engaging in targeted practice sessions.

Unlike conventional template-driven systems, this assistant analyzes students' responses and linguistic cues to identify their strengths and weaknesses. With only a small number of examples, it can quickly learn each student's learning patterns and preferences, enabling it to dynamically adjust question direction and difficulty. For instance, if a student performs well in algebra but struggles with geometry, the assistant will automatically shift focus toward geometry exercises and provide step-by-step guidance with tailored improvement feedback.

In addition, the assistant adapts its strategy according to each student's interests and mastery level, creating personalized learning paths and motivational feedback loops that foster progress in a low-stress environment. By combining Few-Shot Learning with curated educational datasets, this project aims to deliver a deployable AI-powered learning companion for online learning platforms (e.g., Google Colab or institutional LMS), enhancing both engagement and learning outcomes.

---

2. Project Definition

The chosen domain of this project is Intelligent Education, focusing on personalized learning and adaptive instructional technology. With the widespread adoption of online education and the increasing diversity of learners, traditional one-size-fits-all teaching approaches can no longer accommodate the varied learning paces and cognitive differences among students. An AI-powered educational assistant has the potential to enhance both learning efficiency and user experience through real-time analysis and adaptive feedback, serving as an integral component of future learning environments.

Modern educational systems face several key challenges. First, the instructional content often lacks adaptability and fails to dynamically adjust to students' capabilities or goals. Second, students frequently lack timely feedback during the learning process, making it difficult for them to recognize areas where their understanding is incomplete. As a result, they may spend excessive time on repetitive practice without achieving substantial improvement. Finally, most AI-based educational products rely on large-scale datasets for training, making them ineffective in data-scarce or highly personalized learning scenarios. This project addresses

these issues by employing Few-Shot Learning to build an adaptive assistant capable of understanding individual learner differences with minimal examples, thereby enabling genuinely personalized and efficient learning support.

The proposed system integrates several core functionalities within a unified conversational framework. It provides personalized question generation and step-by-step explanations tailored to each student's recent performance and learning style. The assistant analyzes students' answers to detect recurring errors and offers targeted suggestions for improvement. Based on individual interests and mastery levels, it recommends adaptive learning paths and adjusts study pacing accordingly. Through natural, dialogue-based interactions, students can ask questions, receive explanations, and obtain motivational feedback in real time. Additionally, the system periodically generates visualized learning reports to help learners track their progress and identify weak areas, ensuring continuous growth.

The primary users of this system include secondary and college students, adult self-learners, and educational platform developers. The first two groups can directly benefit from personalized learning and targeted practice, while the latter can integrate the assistant into existing e-learning infrastructures to enhance adaptivity and engagement.

The project's core value lies in its ability to deliver personalized learning experiences with minimal data requirements, foster motivation in a low-stress environment, and dynamically adapt to learner interests and performance. Furthermore, the assistant's scalability allows for

seamless deployment across multiple platforms—such as LMS environments, Google Colab, and mobile applications—supporting multilingual education and future expansion.

---

3. Data Requirements Analysis

Since the proposed educational assistant is based on Few-Shot Learning, the focus of data requirements lies not in volume but in representativeness and diversity. The project primarily relies on three categories of data: problem-based textual data, student–AI interaction logs, and instructional feedback records.

In terms of data types, the system processes a combination of structured and unstructured data. Structured data include problem metadata such as difficulty levels, knowledge tags, and topic categories, as well as user profile attributes such as learning goals and interests. Unstructured data consist of conversational text between the student and the assistant, natural language explanations, and system-generated feedback. This hybrid data structure enables the model to capture not only what the student got right or wrong, but also how the student approached each problem.

The data will be sourced from three main channels. First, public educational datasets (e.g., MathQA from Kaggle, OpenAI CodeTutor corpus) will be used for initial fine-tuning and building general-purpose educational knowledge. Second, custom-created datasets will be developed by the team, combining textbook content with instructor-authored practice questions to produce a small yet high-quality sample set suited for few-shot learning. Third,

synthetic or augmented data will be generated through controlled transformations such as rule-based question generation and paraphrasing, increasing semantic coverage without compromising data integrity.

Regarding data volume, approximately 5,000 to 10,000 entries will be used for the initial model training, while each subject area will require only a few dozen curated examples for fine-tuning. The dataset will be refreshed once per academic term, gradually expanding through collected user interactions and feedback.

Data quality standards include accuracy, semantic coherence, and pedagogical relevance. Each question or dialogue sample will undergo manual review and automated filtering to ensure linguistic clarity and conceptual correctness. For user-generated data, the system will employ automatic spell-checking, duplicate detection, and noise filtering to maintain data consistency and ethical integrity.

Potential challenges in data acquisition and quality control include copyright and compliance issues related to educational materials, semantic variability among learners leading to interpretation bias, and overfitting risks caused by limited few-shot samples. To mitigate these issues, the project will implement data augmentation, semantic similarity validation, and iterative mini-batch validation cycles to continuously refine the dataset.

The Data Schema of the system can be conceptualized as a multi-layered framework consisting of three main components. The User Profile Table stores user information, learning objectives, and historical performance. The Learning Content Table contains problem statements, metadata tags, and correct answers. The Interaction Log Table records dialogues between the student and the AI, including question responses, detected error patterns, and feedback messages. These components are linked by unique user IDs and timestamps, enabling dynamic modeling and continuous tracking of each learner's progress.

---

4. Processing Pipeline Design

The processing pipeline of this project is designed to establish an efficient, modular system for data preprocessing and feature engineering to support the operation of the Few-Shot Learning model. The workflow consists of five major stages: data acquisition, cleaning, feature engineering, sample assembly, and model inference.

During the Data Cleaning stage, raw data—including problem statements, solutions, and dialogue records—are imported and processed using the Python Pandas library. Missing values, inconsistent formatting, and redundant entries are identified and resolved. Regular expressions are applied to remove HTML tags, special symbols, and noise. Duplicate entries are eliminated through hash matching and semantic similarity filtering (cosine similarity < 0.95), ensuring data uniqueness and semantic independence.

In the Data Preprocessing stage, all text data are converted to lowercase and tokenized. Stop words and non-semantic characters are removed to enhance signal clarity. For educational text, syntactic simplification and keyword extraction are performed to preserve core pedagogical meaning. For example, the prompt “Explain the difference between variables and constants in Python” is simplified into the key token set {“difference”, “variables”, “constants”, “Python”}.

The Feature Engineering component focuses on constructing high-value features required for few-shot tasks. Text vectorization will be implemented using either Transformer-based sentence embeddings (e.g., Sentence-BERT) or TF-IDF weighting to capture semantic similarity between student responses and reference answers. To enhance model generalization under limited data, data augmentation techniques—such as synonym replacement, sentence reordering, and semantic expansion—are applied to generate small but diverse training examples.

During Sample Assembly, each few-shot task is organized into a structured JSON template containing “example input + expected output” pairs. Each support set includes fields such as problem text, correct answers, error samples, and instructional feedback, forming a compact contextual prompt for model input.

Finally, in the Model Inference stage, formatted data are fed into the few-shot model (e.g., Hugging Face T5, GPT models, or OpenAI API). The model then generates educational

outputs such as hints, explanations, or improvement suggestions. These outputs are subsequently evaluated to assess their relevance and pedagogical quality.

The overall pipeline can be summarized as follows:

1. Import raw educational data (problem sets and dialogue logs)
2. Clean and normalize data (deduplication, tokenization, format standardization)
3. Extract features and generate embeddings (TF-IDF / Transformer-based)
4. Assemble few-shot templates and apply data augmentation
5. Perform few-shot inference and generate personalized responses
6. Output and evaluate educational feedback

This modular pipeline can be easily implemented and demonstrated in a Google Colab or Jupyter Notebook environment. It ensures both technical completeness and practical feasibility, aligning with the course's emphasis on reproducible, data-driven AI workflows.

Raw Data	Cleaning	Preprocessing	Feature Engineering	Few-Shot Template	Model Inference	Feedback Output

---

5. Implementation Strategy

The implementation strategy of this project aims to ensure that the educational assistant prototype can be completed within the constraints of time and computational resources, while successfully demonstrating the full workflow from data processing to few-shot inference. The team will utilize the Python ecosystem and open-source AI libraries to maintain reproducibility and scalability.

For the technical stack, the primary development environments are Google Colab and Jupyter Notebook, allowing cloud-based execution and interactive presentation of model results. Data processing and visualization will rely on Pandas, NumPy, and Matplotlib; text preprocessing and feature extraction will be implemented using NLTK, spaCy, and scikit-learn. The few-shot learning module will be developed with the Hugging Face Transformers framework, using pre-trained models such as T5, DistilGPT, or Sentence-BERT. For natural conversational responses, the OpenAI API may be integrated to improve dialogue quality. All experimental results and logs will be synchronized via Google Drive for version traceability.

The development timeline is divided into three weekly phases. During Week 1, the team will complete the data acquisition and preprocessing components, including data cleaning, normalization, and feature extraction. Week 2 will focus on implementing the few-shot prompting logic and dialogue prototype to test adaptive response behavior. Week 3 will emphasize visualization, evaluation, and the creation of presentation materials (PowerPoint and PDF report). Final deliverables include a runnable Jupyter Notebook, execution screenshots, and the comprehensive project report.

In terms of team member responsibilities, the project will be developed collaboratively by three members. Member A will handle data collection and preprocessing, including feature engineering and dataset formatting. Member B will develop the few-shot prompting and inference logic. Member C will oversee performance evaluation, visualization, and documentation, producing final figures and written materials. The team will use Google Docs and GitHub for version control and collaborative editing to maintain synchronization between code and documentation.

The resource requirements are minimal. The project can be executed entirely in CPU mode using the free Google Colab environment, with no need for dedicated GPU resources. Data will be stored on Google Drive and accessed via Python APIs. If the OpenAI API is used for response generation, the expected cost is under USD 5, shared among team members.

Potential implementation challenges include instability in generated outputs, limited sample diversity, and reduced interpretability of responses. To address these, the team will employ iterative prompt validation, qualitative output evaluation, and selective manual refinement to ensure educational relevance. If model outputs diverge from pedagogical objectives, rule-based correction mechanisms will be applied.

Finally, the integration approach will link all modules within a unified Jupyter Notebook framework. Each stage of the workflow will communicate through standardized interfaces

(e.g., JSON structures or Pandas DataFrames), ensuring compatibility between components. The final deliverable will demonstrate a fully functional pipeline—from data input to adaptive educational feedback—executed within Google Colab to validate the assistant's feasibility and pedagogical value.

---

6. Evaluation Framework

To comprehensively evaluate the performance and educational impact of the assistant, this project establishes a dual-layer evaluation framework encompassing both technical and pedagogical dimensions. The framework consists of five key components: success metrics, testing strategy, user feedback mechanism, performance benchmarks, and continuous improvement process.

Success Metrics are defined across quantitative and qualitative measures. On the technical side, evaluation will use accuracy, semantic similarity, and response consistency as core indicators. Semantic similarity is computed through cosine similarity between model outputs and reference answers, while response consistency measures the stability of generated outputs under similar prompts. On the pedagogical side, the assistant's effectiveness will be assessed through learning gain—the improvement in test performance before and after interaction—and user satisfaction, derived from feedback surveys and ratings.

The Testing Strategy includes modular unit tests and integrated scenario validation. Unit tests will verify the correctness of data preprocessing and inference pipelines. During integration

testing, simulated learning sessions will test the assistant's adaptability across different learner profiles (e.g., beginner vs. advanced). A small set of 5–10 unseen problem samples will be introduced to measure generalization and few-shot adaptability. Results will be reported through tables and visualizations such as confusion matrices and performance plots.

For the User Feedback Mechanism, the system will provide a simple feedback interface allowing users to rate each response with options such as "helpful," "irrelevant," or "needs more explanation." These responses will be logged for iterative improvement and model fine-tuning. Future versions could automate this feedback integration, enabling the assistant to adapt question difficulty and topic focus based on cumulative user trends.

The Performance Benchmarks for the prototype will focus on maintaining a high semantic alignment and user approval rate. The target is an average cosine similarity of 0.85 or higher between generated and reference outputs, and at least 80% positive feedback from users. While the project is exploratory, these thresholds provide a measurable foundation for assessing educational impact.

The Continuous Improvement Process centers on a feedback-driven data loop. Each interaction will feed into the augmentation and retraining cycle, enriching the dataset with validated examples and improving prompt templates. Post-project development may incorporate Reinforcement Learning from Human Feedback (RLHF) to refine adaptability and pedagogical sensitivity, thereby strengthening the system's long-term learning effectiveness.

References

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). *Attention is all you need.* In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS 2017)* (pp. 5998–6008). Curran Associates, Inc. https://doi.org/10.48550/arXiv.1706.03762

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of deep bidirectional transformers for language understanding.* In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2019)* (pp. 4171–4186). Association for Computational Linguistics. https://doi.org/10.48550/arXiv.1810.04805

Hugging Face. (n.d.). *Transformers documentation.* Hugging Face. https://huggingface.co/docs/transformers

Kaggle. (n.d.). *MathQA dataset.* Kaggle. https://www.kaggle.com/datasets

OpenAI. (n.d.). *OpenAI API documentation.* OpenAI. https://platform.openai.com/docs