AI-Powered Teaching Assistant

Digvijay Sanjay Ingole, Amit Anantrao Patil
California State University East Bay, Hayward
Department of Computer Science

January 8, 2025

Problem Statement

The goal of this project is to develop an AI-powered Teaching Assistant (TA) that enhances the learning experience for students by providing personalized, interactive, and accessible educational tools.

This project is motivated by our shared experiences as students, where we often faced challenges in finding answers within lengthy video lectures or extensive course materials. These difficulties can hinder learning, reduce engagement, and leave gaps in understanding. By addressing these pain points, we aim to create a solution that empowers students to learn more effectively and efficiently.

The proposed system is designed to complement existing Learning Management Systems (LMS) like Canvas by offering innovative features such as intelligent chatbots, interactive flashcards, dynamic quizzes, and summarized study notes. These tools will streamline learning processes and make advanced educational materials more accessible without duplicating existing LMS functionalities.

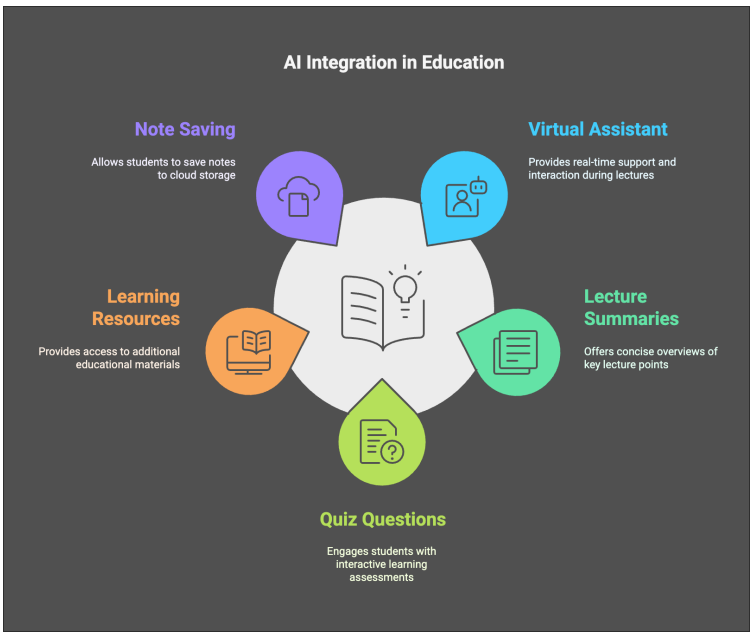

Figure 1: AI Integration in Education: Virtual Assistant, Lecture Summaries, Quiz Questions, Note Saving, and Learning Resources.

Introduction and Motivation

Current Learning Management Systems (LMS) such as Canvas offer robust tools for managing courses and materials but lack personalized features that cater to diverse student needs. Our project seeks to bridge this gap by focusing on the following student-centered features:

- **AI-Powered Chatbot:** A real-time assistant for topic-specific queries, providing detailed explanations and resources.
- **Interactive Flashcards:** AI-generated flashcards featuring spaced repetition and multimedia support to enhance concept retention.
- **Dynamic Quizzes:** Custom quizzes generated from uploaded materials to reinforce understanding and provide real-time feedback.
- **Concise Notes Generator:** Topic-specific notes extracted from course materials for quick reference and revision.

This project builds upon the success of our prior work, the Zoom Learning Assistant, completed in the previous semester. The Zoom Learning Assistant integrated AI capabilities directly into the Zoom platform, offering features such as lecture summaries, quiz generation, additional learning resources, and automated note-saving to Google Drive. User feedback highlighted the significant potential of AI in improving student learning efficiency and inspired us to expand the concept into a more comprehensive solution.

Through research and surveys conducted with students, we identified that no existing platform comprehensively addresses their need for personalized, real-time, and interactive learning aids. While some tools focus on individual features like flashcards or quizzes, there is no integrated solution that holistically supports students across their learning journey. This gap motivates us to develop an AI-powered Teaching Assistant tailored to enhance student engagement, simplify access to educational content, and provide meaningful insights for effective learning.

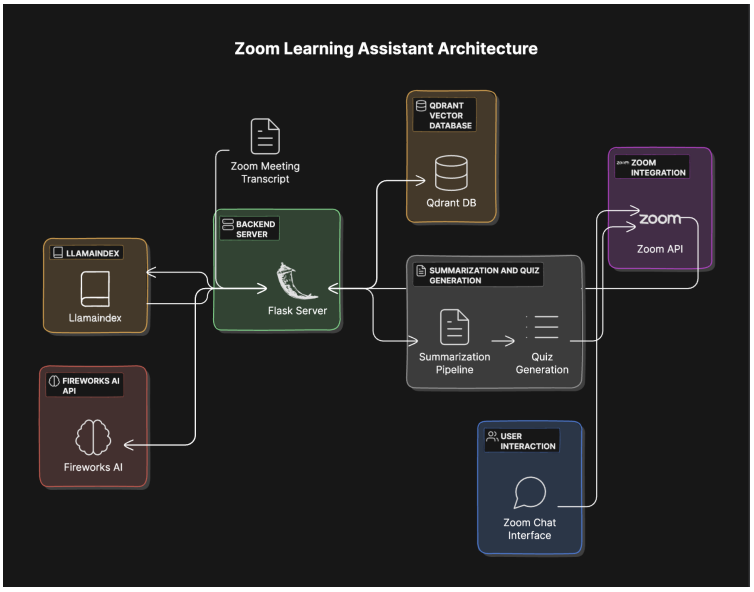

Figure 2: Flowchart illustrating the architecture and workflow of the previously developed Zoom Learning Assistant

Proposed Features

- **AI-Powered Chatbot**: A 24/7 assistant that answers topic-specific queries using natural language processing, providing tailored explanations and additional resources.
  - **Example:** A student asks, "Can you explain the difference between dynamic programming and divide-and-conquer?" The chatbot provides an explanation with examples, such as the Fibonacci sequence (dynamic programming) and merge sort (divide-and-conquer).
- **Interactive Flashcards**: AI-generated flashcards with features like spaced repetition and multimedia support to enhance concept retention.
  - **Example:** After uploading lecture notes on graph algorithms, the system generates flashcards like "Define Dijkstra's algorithm and its use case" or "What is the time complexity of Kruskal's algorithm?" for better understanding.
- **Dynamic Quizzes**: MCQ-based quizzes generated dynamically from uploaded course materials, targeting specific topics or concepts in advanced algorithms.
  - **Example:** After uploading slides on "Divide and Conquer Algorithms," the system generates:
    - **Easy Question:** "Which of the following is an example of a divide-and-conquer algorithm?"
      - A. Bubble Sort
      - B. Merge Sort (Correct Answer)
      - C. Linear Search
      - D. Dynamic Programming
    - **Medium Question:** "What is the recurrence relation for Merge Sort?"
      - A. $T(n) = 2T(n/2) + n$ (Correct Answer)
      - B. $T(n) = T(n-1) + n$
      - C. $T(n) = T(n/2) + \log(n)$
      - D. $T(n) = 2T(n-1) + n$
    - **Hard Question:** "What is the time complexity of finding the inversion count of an array using a divide-and-conquer approach?"
      - A. $O(n)$
      - B. $O(n \log n)$ (Correct Answer)
      - C. $O(n^2)$
      - D. $O(\log n)$
  - **Notes**: Summarized, topic-specific notes extracted from uploaded course materials for quick and easy reference.
    - **Example:** A professor uploads a lecture on "Greedy Algorithms." The system generates notes like "Key Examples: Huffman coding, Minimum Spanning Tree (Prim's and Kruskal's algorithms), and Activity Selection Problem" for students to review.

Implementation Approach

- **AI-Powered Chatbot**:
  - The chatbot will utilize a natural language processing (NLP) model such as OpenAI's GPT for generating intelligent, context-aware responses.
  - **Backend:** Flask APIs will handle user queries and pass them to the AI model via Groq. Groq will perform contextual analysis using embeddings stored in Qdrant.
  - **Database:** Qdrant, a vector database, will store embeddings of course materials for quick retrieval. When a query is received, the chatbot will search the embeddings to retrieve the most relevant sections of the content.
  - **Frontend:** The chatbot interface will be developed in Next.js, providing real-time interaction through WebSocket or REST API communication with the backend.
  - **Workflow:**
    1. Students type a question.
    2. The frontend sends the query to the backend.
    3. The backend searches the Qdrant database for context, processes the query with Groq, and returns a response.
    4. The response is displayed to the user in the chatbot interface.
- **Interactive Flashcards**:
  - **Backend:** Flask APIs will analyze uploaded teaching materials (PDFs, PowerPoints, etc.) using AI to extract key concepts and summaries.
  - **AI Model:** NLP models (e.g., Groq) will identify essential topics and convert them into flashcards with concise text. Media analysis tools can extract relevant images or diagrams for multimedia support.
  - **Database:** Generated flashcards will be stored in Firestore, categorized by topic and difficulty level. Progress data (e.g., which flashcards have been reviewed) will also be tracked.
  - **Frontend:** The flashcard interface, built in Next.js, will include spaced repetition algorithms to determine which flashcards to show based on user performance.
  - **Workflow:**
    1. Course material is uploaded via the frontend.
    2. The backend processes the material and generates flashcards using AI models.
    3. Flashcards are retrieved from Firestore and displayed in a swipeable or clickable interface.
    4. User interactions (e.g., "I know this card") are tracked and used to optimize the review schedule.
- **Dynamic Quizzes**:
  - **Backend:** Flask APIs will generate quizzes by analyzing uploaded content, using algorithms to create questions of varying complexity (e.g., multiple choice, short answer).

    - **AI Model:** Groq will analyze course materials and generate questions based on Bloom's taxonomy levels. The system will use AI to balance difficulty based on user performance metrics.
    - **Database:** Questions, scores, and progress data will be stored in Firestore. Metadata like timestamps and attempt history will also be logged.
    - **Frontend:** The quiz interface, built in Next.js, will support features such as timers, immediate feedback, and progress tracking. Adaptive quizzes will adjust the difficulty dynamically based on real-time responses.
    - **Workflow:**
        1. The Student selects a topic or quiz type.
        2. The backend fetches questions or generates new ones based on course material.
        3. Questions are delivered to the frontend, where users can answer them.
        4. User responses and scores are recorded and used to adjust future quizzes.
- **Notes**:
    - **Backend:** Flask APIs will process uploaded teaching materials to extract key points and generate concise, topic-specific notes using NLP models like Groq.
    - **Database:** Notes will be stored in Firestore, categorized by subject, topic, and priority level (e.g., essential vs. supplementary).
    - **Frontend:** The notes interface, developed using Next.js, will display the extracted notes in a clean and readable format. Features such as search, filtering, and bookmarking will enhance usability.
    - **Workflow:**
        1. Teacher upload course materials such as PDFs, slides, or lecture recordings.
        2. The backend analyzes the content and generates summarized notes using AI.
        3. Notes are categorized and stored in Firestore.
        4. The frontend retrieves and displays the notes with options for filtering and searching by topic or keyword.

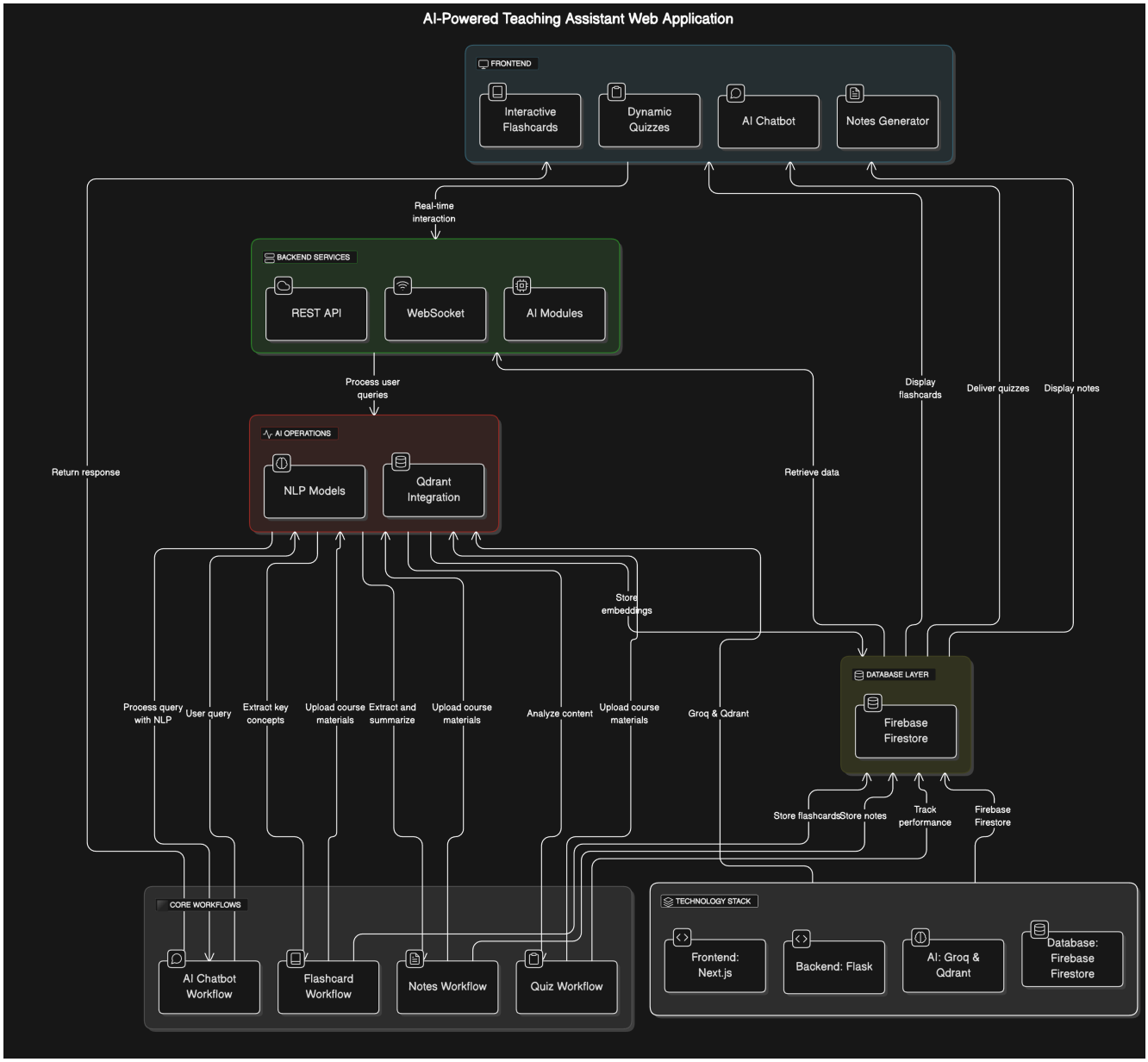

Figure 3: AI-Powered Teaching Assistant Proposed Architecture.

Tech Stack

The project will use the following technologies:

- **Frontend:** Next.js for building a fast and responsive web application.
- **Backend:** Flask for creating REST APIs.
- **AI Operations:** Groq and Qdrant for AI capabilities and vector database.
- **Database and Authentication:** Firebase Backend Services and Firestore.

Project Timeline

- **Week 1: System Architecture Design**: Develop the overall system blueprint, including data flow diagrams, component architecture, and integration points for frontend, backend, and AI features. Ensure scalability and alignment with the planned deliverables.
- **Weeks 2-4: Dashboard Development (Frontend)**:
  - **Week 2**: Design the user interface for the student and teacher dashboards using Next.js, focusing on layout, navigation, and placeholders for key features.
  - **Week 3**: Develop the basic frontend components for flashcards, quizzes, notes, and analytics. Create interactive mockups for user validation.
  - **Week 4**: Finalize the frontend functionality, including responsive design and UI/UX optimizations for accessibility and ease of use.
- **Weeks 5-7: Backend Development and Integration**:
  - **Week 5**: Set up the Flask backend, configure Firestore for database management, and develop basic APIs for dashboard data handling.
  - **Week 6**: Build and integrate APIs for flashcards and quizzes, ensuring data consistency and performance optimization.
  - **Week 7**: Develop backend functionality for analytics and notes generation. Ensure proper integration with Firestore for secure and efficient data storage.
- **Weeks 8-10: AI Feature Development and Integration**:
  - **Week 8**: Integrate the AI-powered chatbot using Grok for NLP and Qdrant for context-based data retrieval.
  - **Week 9**: Implement AI-based algorithms for flashcards and quiz generation from course materials, focusing on content accuracy and relevance.
  - **Week 10**: Develop the notes generator to summarize and extract key points from uploaded teaching materials.
- **Week 11: Functional Testing**: Conduct comprehensive testing of all features:
  - Validate student-facing functionalities, including flashcards, quizzes, notes, and the chatbot.
  - Test teacher-facing functionalities, including analytics and dashboard tools.
  - Identify and fix critical bugs, ensuring smooth functionality across all components.
- **Week 12: User Feedback Integration**: Collect feedback from beta testers, including students and teachers. Refine features based on feedback to address usability and functionality issues.
- **Week 13: Final Report Preparation**: Consolidate all project documentation, including:
  - Objectives and system design.
  - Implementation details for each feature.

    - Testing results and user feedback analysis.
    - Challenges encountered and solutions applied.
    - Future scalability and improvements.
- **Week 14: Final Presentation Preparation**: Develop a presentation summarizing the project, with a focus on key achievements, a live demo of the web application, and user impact.
- **Week 15: Final Presentation and Submission**: Deliver the final presentation to stakeholders, demonstrating the completed web application, its features, and its potential for scalability and impact.

Deliverable

The primary deliverable is a fully functional web application that integrates all the proposed features into a seamless platform. This application will serve as a comprehensive tool for both students and educators, providing the following functionalities:

- **AI-Powered Chatbot**: A 24/7 assistant embedded within the web application to answer topic-specific queries and provide additional learning resources.
- **Dynamic Flashcards**: A module for AI-generated flashcards featuring spaced repetition and multimedia support to aid in concept retention.
- **MCQ-Based Dynamic Quizzes**: A feature for generating quizzes on-the-spot based on uploaded course materials, with real-time feedback and progress tracking.
- **Concise Notes Generator**: A tool to extract and summarize key information from uploaded teaching materials, creating topic-specific notes for quick reference.

Expected Outcomes

- **Enhanced Learning Outcomes**: Students will benefit from interactive tools such as flashcards, quizzes, and topic-specific notes that adapt to their learning needs, making advanced algorithm concepts more accessible and engaging.
- **Reduced Teacher Workload**: Automation of routine tasks, such as quiz generation and participation analysis, will enable teachers to focus on more impactful activities like personalized instruction and curriculum design.
- **Real-Time Learning Support**: Instant generation of MCQ-based quizzes and real-time feedback will encourage active learning, allowing students to self-assess and address gaps immediately.
- **Seamless Usability and Integration**: The system will complement existing LMS platforms like Canvas, ensuring ease of adoption for educators and students without disrupting established workflows.

- **Data-Driven Teaching Insights**: Teachers will gain actionable insights through analytics, helping them identify struggling students, adjust instructional strategies, and track overall class performance.
- **Scalability and Versatility**: While the system is tailored for advanced algorithm courses, its design is scalable for broader applications, including corporate training, certification programs, and online education.
- **Support for Complex Topics**: The tools will address advanced algorithm topics like dynamic programming, NP-completeness, and graph algorithms, providing targeted support to students tackling challenging material.

References

- Jill Watson: Developed by Georgia Tech's Design Intelligence Lab, Jill Watson is an AI virtual teaching assistant that engages students in meaningful conversations about course materials, leveraging advanced AI techniques to provide accurate and context-relevant assistance. Available at: `https://dilab.gatech.edu/jill-watson`
- TAI: Teaching Assistant AI: A personalized AI teaching assistant that answers questions about classes using a combination of GPT-4 and the Canvas API. It integrates with course materials to provide tailored support to students. Available at: `https://devpost.com/software/tai-teaching-assistant-ai`
- Artificial Intelligence Teaching Assistant (AITA): A system designed to enhance student engagement and instructor presence in online courses by automating the creation of weekly announcements, including details about upcoming assignments and due dates. Available at: `https://github.com/jwadec/AITA`
- Pruju AI: An AI teaching assistant that allows students to interact with the teacher's course materials, facilitating a more engaging learning experience. Available at: `https://github.com/jaluoma/pruju-ai`
- Eduaide.Ai: An AI-powered teaching workspace that helps educators create, adapt, and refine their instructional materials, including lesson plans, quizzes, and slides. Available at: `https://www.eduaide.ai`
- AI Teaching Assistant by Code.org: An AI Teaching Assistant designed to assist teachers in managing classroom tasks and providing instant information to students, enhancing the overall learning experience. Available at: `https://code.org/ai/teaching-assistant`
- LittleMu: A virtual MOOC teaching assistant that provides question answering and chit-chat services by integrating heterogeneous sources and utilizing advanced prompting techniques. Available at: `https://arxiv.org/abs/2308.05935`
- AI-TA: An intelligent question-answer teaching assistant using open-source large language models to provide scalable and intelligent question-answering in educational settings. Available at: `https://arxiv.org/abs/2311.02775`

– Generative AI Teaching Assistant: A project aimed at combining generative AI into the learning process to benefit both professors and students by providing AI-driven educational support. Available at: https://www.fau.edu/engineering/senior-design/projects/fall2024/generative-ai-teaching-assistant
– Teacher Assistant: A virtual lesson generator that helps teachers produce lesson plans, quizzes, slides, and more, instantly generated to aid in instructional design. Available at: https://aiteacherassistant.vercel.app