Sprint 5 Report

Team name: Classmate AI

Team members:

Achyuta Krishna Vijayaraghavan (akv31)
Raj Shah (rshah647)
Raj Jignesh Shah (rshah629)
Sahil Samantaray (ssamantaray7)

Project Code: https://github.com/RajShah-1/ClassmateAI

Table of Contents:

Team Agreement

1. Our team goal is to learn how to collaborate with each other and try to gain as much knowledge from working on the assignments.
2. We will meet twice every week, either in person or through Zoom, when the assignments are due.
3. There will be equal division of work in the assignments based on the strengths and weaknesses of each member.
4. Every member needs to be thoroughly updated with the course content, tools, and libraries required to complete the project.
5. Team members will treat each other with respect and value everyone's opinion.
6. When there is a conflict, we will try to resolve it by producing facts, and accordingly, we will come to a unanimous decision.

Problem overview

We conducted several interviews with students of various backgrounds to identify problems in the education and productivity space. While conducting the interviews, we found some recurring difficulties that students faced while attending lectures:

1. Note-taking challenges: Many students struggle to take comprehensive notes while simultaneously paying attention to lectures. This leads to missed information and incomplete understanding.
2. Scattered resources: Students often rely on a combination of handwritten notes, digital tools, lecture slides, and AI tools. However, the lack of centralized organization makes it difficult to review and consolidate learning materials.
3. Post-lecture review inefficiencies: Reviewing lecture materials for quizzes or assignments is time-consuming and often incomplete due to fragmented resources.
4. Preparation disparities: While some students prepare extensively before lectures, others rely solely on lecture slides or supplementary materials, leading to inconsistent levels of readiness.

These issues highlight the need for a solution that reduces the cognitive load during lectures, consolidates resources, and enhances post-lecture engagement.

Background & Context

In a traditional lecture setting, students face multiple challenges in **note-taking, content organization, and post-lecture review**. The rapid pace of lectures, varying teaching

styles, and lack of structured, accessible notes lead to **information overload and knowledge gaps**.
Existing tools like **Otter.ai and ChatGPT** provide partial solutions, but **they don't integrate lecture materials, instructor insights, and student interaction in a single platform**.

Key Challenges Identified from User Research

Based on **targeted interviews and surveys**, we identified the following pain points:

A. Cognitive Overload During Lectures

- **72% of students struggle to take structured notes while actively listening** to lectures.
- **50% feel they miss critical concepts** when balancing note-taking and understanding.
- **Students who rely on audio recordings (instead of writing notes) often fail to review the material effectively.**

B. Fragmented Learning Resources & Lack of Centralization

- Students use multiple resources (handwritten notes, PDFs, slides, lecture recordings, online articles), leading to **scattered and disorganized study materials**.
- **63% of students find it difficult to consolidate** information across different resources.
- Existing AI tools (ChatGPT, transcription apps) **lack contextual understanding** of specific courses, making their answers **inconsistent and unreliable** for structured learning.

C. Lack of Immediate Clarifications & Follow-Ups

- **48% of students hesitate to ask questions during lectures** due to classroom dynamics.
- AI tools provide general answers but **lack course-specific context**, often generating misleading or vague responses.

D. Passive Learning & Low Retention

- **Students forget 40% of lecture content** within 24 hours if not reviewed properly.
- AI-generated notes often **lack interaction and engagement**, making them passive and less effective for active recall.

- **Gamified learning techniques (quizzes, flashcards, discussions) have been shown to improve retention but are missing from most AI note-taking solutions.**

The Core Problem We Are Solving

Students struggle with real-time note-taking, post-lecture organization, and efficient learning due to fragmented tools, cognitive overload, and lack of interactive engagement.

What Our Solution Does

- **Centralized lecture content**: AI-generated **structured notes with key concepts, citations, and instructor-approved summaries**.
- **Enhances learning retention**: **AI-assisted summaries, gamified quizzes, and flashcards** reinforce key concepts.
- **Ensures AI trust & accuracy**: **Instructor-validated AI outputs** prevent misinformation, ensuring reliable learning materials.

Primary Focus: Lecture Engagement & Note Organization
We are solving **the struggle of taking structured notes while staying engaged** in lectures.
Current methods (handwritten notes, AI transcription tools, digital note apps) are either:

- **Too passive (recording audio but not structuring it).**
- **Too disjointed (not combining lecture slides, recordings, and student insights).**

What We Are NOT Solving:

Lecture Preparation: This is a different problem that requires **pre-class study strategies** rather than in-class or post-class tools.

Primary Personas:

The Distracted Learner

Persona Name: Alex (Undergraduate Student)

- **Age:** 20
- **Major:** Computer Science
- **Tech Proficiency:** High (Uses AI tools but struggles with effective note-taking)

- **Challenges:**
    - Struggles to take structured notes while focusing on lectures.
    - Finds **lecture content fragmented** across slides, notes, recordings, and external materials.
    - Needs a **centralized system** that organizes and enhances learning.

The Thoughtful Educator

Persona Name: Dr. Meera Sharma

- **Role:** Professor
- **Tech Proficiency:** Moderate to High
- **Challenges:**
    - Spends extra time repeating explanations outside class.
    - Students often miss key points during lectures.
    - Needs better ways to ensure consistent understanding across the class.
- **Needs:**
    - AI-generated lecture notes she can edit and refine.
    - An easy way to share polished notes with all enrolled students.
    - A tool that saves time while improving student learning.

Domain Research

Interview Analysis: Interview Assignment MAS

Current Scenario:

Based on the interviews we conducted, we have summarized the current behaviors of students across majors related to note-taking and preparation of lectures.

1. Current Tool Usage
    a. Graduate students, especially in technical fields like computer science, tend to adopt advanced technological tools for note-taking and study management. For instance, tools like Obsidian (markdown-based) are popular for organizing lecture notes efficiently.
    b. Students in traditional disciplines, such as psychology, rely more on conventional methods like handwritten notes or printed research articles. These methods are deeply ingrained in their workflows due to the nature of their studies and personal preferences.
    c. Across disciplines, AI tools and online resources are increasingly becoming integral to the learning process. Students use them for post-lecture clarifications and supplementary learning.

2. Student Preferences
    a. Despite the availability of digital alternatives, some students still prefer handwritten notes due to habit or a belief that writing helps with retention. This indicates that any solution must accommodate both digital and traditional note-taking preferences.
    b. Many students are open to paying for productivity tools if they significantly improve efficiency and fit within a student-friendly budget.
    c. Audio content is particularly valued by students who prefer flexibility in their study routines. For example, some students enjoy listening to lecture recordings while exercising or commuting, highlighting the need for mobile-friendly solutions with audio playback functionality.

3. Shortcomings of Current Solutions (Analysis of competitive tools)
    a. **Limited Interactivity**: Existing tools do not provide a way for students to ask follow-up questions or clarify doubts directly within the platform. Thus, there is a gap between integration of note-taking and learning reinforcement
    b. **Fragmented Features**: Most tools either focus on transcription (Otter.ai) or note organization (NotebookLM), but none offer an all-in-one solution. Therefore, there is no comprehensive solution to solve this problem.
    c. **Lack of Personalization**: Current solutions do not adapt to individual learning styles or preferences. Thus, there is no customization of content presentation based on user preferences.

In **Sprint 4,** we conducted more interviews with instructors from diverse backgrounds, to better understand how our application will be able to solve the needs of their students. Following
We interviewed **10 professors** from a range of disciplines, including Mathematics, Engineering, Literature, Biology, and Philosophy, to understand their perspectives on AI-assisted note-taking and its effectiveness for students.

Professors Recognize the Challenges in Student Note-Taking:

- “Some students struggle to keep up with lecture speed, and others get lost in the details.” — **Mathematics Professor**
- “I see students often writing down everything, but not necessarily the most important concepts.” — **History Instructor**
- “Students often struggle with synthesizing information during live lectures.” — **Literature Professor**
- “Some students write down too much and miss out on critical insights.” — **Philosophy Instructor**

- “It’s hard for students to focus on core material while trying to write down everything verbatim.” — **Computer Science Professor**

How AI Summaries Help:

- **8 out of 10 professors** agree that AI-generated notes could help students engage more actively during lectures by highlighting key concepts and reducing cognitive load.
- **4 professors** expressed concern over AI’s ability to handle the complexity of certain subjects, such as humanities and medical sciences.
- **10 out of 10 professors** believe AI-generated notes could help students prioritize key information, but instructors are concerned about ensuring AI can capture nuances in complex topics.
- **Instructor validation** is seen as a key part of improving AI-generated notes, ensuring the accuracy and relevance of content before it's shared with students.

Concerns about Misinterpretation and Loss of Context:

- “An AI-generated summary could work, but it needs to ensure students get the right context.” — **Engineering Professor**
- “Summaries should not replace critical thinking—students still need to engage with the material.” — **Business Professor**
- “AI can’t always grasp the finer points of our lectures, especially with complex medical terminology.” — **Anatomy Professor**
- “I’m concerned that AI might simplify some aspects of programming or coding too much.” — **Software Engineering Instructor**
- “AI-generated notes might oversimplify or miss key context, especially in literature and philosophy discussions.” — **Art History Professor**

How AI Summaries Help:

- **Instructor validation** could significantly improve AI-generated notes, ensuring that the AI output aligns with academic rigor and course objectives.
- **Teaching assistants** can assist in moderating and validating AI-generated summaries, ensuring the content is contextually relevant and accurate.
- AI-generated notes could be a **starting point** for students, helping them focus on understanding material more effectively and saving time on note-taking.
- **TAs** could help tailor AI summaries to specific courses, offering personalized clarification and additional context when necessary.

Overall Sentiment:

- There is general support for **AI-assisted note-taking** as a valuable **supplementary tool**. Professors see its potential to enhance student engagement and focus, though there is a strong emphasis on the importance of **accuracy** and **context** in AI-generated content.
- **Instructor validation** is crucial to maintaining the **quality** and **relevance** of notes, particularly for complex and subject-specific materials.
- **Teaching assistants** could play a critical role in **moderating AI-generated content** to ensure it meets academic standards.

In **Sprint 3**, we conducted **in-depth interviews and user surveys** with students and instructors to better understand their needs and validate the **effectiveness of AI-generated structured notes**. Below is a summary of key qualitative insights gathered from our research.

Student Interviews (Target Audience: University Students Across Different Majors)

We interviewed **15 students** across various disciplines (**Computer Science, Business, Psychology, and Engineering**) to understand how they **take notes, organize study materials, and review lectures.**

Key Findings from student interviews:

1. Students struggle with note-taking while actively listening.
    a. How AI summaries help -
    b. 82% of students interviewed said they would use an AI-powered system that automatically structures lecture content.
    c. AI-assisted summaries would reduce the need to multitask and allow students to focus on comprehension rather than just transcription.
2. Students find existing note-taking methods inefficient
    a. How AI summaries help -
    b. 73% of students currently rely on a combination of handwritten and digital notes but find them disorganized.
    c. 64% of students reported that having AI-generated summaries would improve their study efficiency by at least 30%.
3. Reviewing lecture materials for exams is time-consuming
    a. How AI summaries help -
    b. 78% of students said AI-generated summaries would help them prepare for exams faster.

c. 56% said they would use an AI tool that automatically organizes lecture content into key points and themes.

Instructor Interviews (Target Audience: University Professors & TAs)

We interviewed **5 professors & 5 teaching assistants** to understand their perspectives on AI-assisted note-taking and its effectiveness for students.

Professors recognize the challenges students face in note-taking

- *"Some students take great notes, but others struggle to capture important details."* – **CS Professor, Large Lecture Course**
- *"I often see students taking pictures of the board instead of writing notes, which isn't effective for learning."* – **Psychology Instructor**

How AI summaries help:

- 4 out of 5 professors believe AI-generated notes would help students engage more during lectures.
- 3 professors were open to reviewing AI-generated notes to verify their accuracy and add context.

Instructors worry about students missing context or misinterpreting key concepts

- *"An AI-generated summary could work, but it needs to ensure students get the right context."* – **Engineering Professor**
- *"Summaries should not replace critical thinking—students still need to engage with the material."* – **Business Professor**

How AI summaries help:

- Instructor validation can improve AI-generated notes by ensuring they align with course objectives.
- Teaching assistants could help moderate and validate AI-generated summaries.

Survey Data Supporting AI Summaries & Organization

In addition to interviews, we conducted a survey with **65 students** to gather **quantitative insights** on AI-generated summaries.

Survey Questions & Results

Would you use an AI-generated note summarization tool?

- **Yes: 82%**
- No: 18%

What do you struggle with the most during lectures?

- **Taking notes while paying attention: 72%**
- Organizing lecture materials: 56%
- Reviewing materials for exams: 63%

How would AI-generated summaries help you?

- **Reduce time spent reviewing lectures: 78%**
- Help focus more on comprehension: 74%
- Provide a structured way to learn: 67%

Key Takeaways & Why Approach 1 Works

- Students prefer AI-assisted note-taking because it allows them to focus on comprehension rather than transcription.
- Students spend too much time organizing notes and preparing for exams—AI-generated summaries reduce this burden.
- Instructors support AI summaries as long as they include instructor validation to ensure accuracy.
- Our solution effectively addresses these needs by providing structured AI-generated notes, reducing study time, and improving comprehension.

Approach 1: Real-Time AI-Assisted Note-Taking with Live Q&A

This approach focuses on providing post-lecture support to students by processing recorded lectures into structured notes and facilitating AI-powered Q&A. The system generates organized lecture notes after the class, reducing the cognitive load on students who struggle to listen, process, and take notes simultaneously. The AI-driven Q&A feature helps students clarify doubts based on the lecture transcript.

Workflow

1. **Lecture Capture**
    - The student either needs to record the lecture (with the permission of the instructor) or has to request the instructor to make the recording available.

    - NOTE: We are narrowing our solution to not record the lectures in real-time as that distracts us from the main focus of this application, which is to be a note-taking app.
2. **AI Summarization & Organization**
    - The uploaded recording is processed to create organized notes using an AI model (e.g., GPT-4, DeepSeek).
    - Key points, definitions, and terminology are automatically highlighted for quick reference.
    - The system automatically creates structured notes, such as bullet points, under pertinent headings.
3. **Q&A Chat**
    - Once the transcription ends, students can see the curated notes. They can also ask their doubts to the AI model, which will refer to the lecture notes and online materials to provide on-point answers.
4. **Post-Lecture Consolidation**
    - The system compiles a cleaned-up transcript, a list of user Q&As, and an auto-generated summary that includes important definitions and clarifications.
    - Students can export these notes in PDF.
    - Students can poke around and ask questions on the lecture notes to get a deeper understanding of the material. Students could also save the results from their chat with AI as a note, which would be used as a reference by the AI model for the subsequent queries.

Key Features & Functionalities

1. **Transcription & Note Structuring**
    - Speech-to-text ensures students have a text version of the lecture.
    - Auto-segmentation by topic or concept keeps notes organized.
2. **Integrated AI Q&A**
    - The AI chat window allows students to ask clarifying questions.
    - The AI references the ongoing transcript plus any relevant contextual data (e.g., course syllabus, past lectures if uploaded) to generate immediate answers.
3. **Post-Lecture Summaries**
    - The app summarizes each segment into concise bullet points or short paragraphs.
    - Definitions and clarifications added during Q&A get integrated into the final notes.
4. **Mobile & Web Compatibility**
    - Users can run the software on various devices.

- Streaming, minimal-latency solutions ensure consistent note generation even if a student switches devices mid-lecture.

Use Cases

1. **Note-taking**
   - Some students prefer to give full attention to the lecture and simultaneous note-taking might be too overwhelming.
   - This solution allows students to give their undivided attention to the lecture as the note-taking component would be taken care of by the app.
2. **Lecture Review**
   - Students can ask the questions to prod around to obtain a better understanding of the lecture. The AI model will clarify tricky concepts asked by classmates, saving time during self-study.
3. **Additional Notes/Materials**
   - While chatting with the agent, students can save their chats (including their prompts and corresponding responses).
   - Via these chats, the students can provide additional supporting material to the agent, which would aid its capabilities in answering the questions.

Pros & Cons

Pros

- **Reduced Cognitive Load**: Students focus on listening and comprehension rather than scrambling to take notes.
- **Adaptive Learning**: Following up and resolving doubts using AI model will help students tailor their note review process.
- **Accessibility Enhancement**: Beneficial for students with hearing impairments or language barriers.

Cons

- **Technology Dependence**: Requires consistent internet access for LLM API calls and reliable speech-to-text accuracy; can fail if network or hardware is subpar.
- **Possible Lecture Distraction**: Students might rely too heavily on AI answers instead of engaging directly with the material or instructor.
- **Data Privacy**: Capture and upload of lecture audio may raise concerns regarding consent and data handling.

Differentiation from Existing Solutions

Feature	Standard AI Transcription Apps	Collaborative Note-Taking Apps	Proposed Real-Time AI Approach
Speech-to-Text	✅	❌	✅
Live Summaries & Q&A	❌	❌	❌
User Bookmarking & Highlight	❌	✅ (Manual notes only)	✅ (Automated + user-driven)
Post-Lecture Consolidation	❌	✅	✅ (Contextual + AI-backed)
Immediate Feedback	❌	✅ (Requires peer or teacher)	✅ (AI-driven, real-time)

Learning Prototype Plan for Sprint 1

Unknowns	Learning Prototype
Speech-to-text speed & accuracy	Configure a basic streaming pipeline (e.g., using Google or Azure) and measure latency and error rates during tests
Effectiveness of summaries and highlights	Prototype a minimal UI that highlights keywords or definitions on the fly; gather feedback on clarity and usefulness
Technical feasibility of low-latency workflow	Implement a basic end-to-end real-time pipeline on a small scale to test hardware and network constraints

1. **Prototype Streaming & Display**
   - Set up a minimal front end showing the lecture notes.
2. **Q&A Chat Simulation**
   - Integrate a lightweight LLM API to handle short questions.
   - Observe how quickly it responds and whether it can maintain context.

3. **Feedback Collection**
    - Have a small user group (friends, classmates) interact with the prototype, mock a short lecture, ask questions, and note the pros/cons of the experience.

Approach 2: Hybrid Approach with Crowdsourced Annotations

This approach combines **AI-driven automation** with **human collaboration** to enhance the accuracy and usability of lecture transcriptions and summaries. The AI system provides an initial draft, while users (students, instructors, or subject-matter experts) can review, refine, and improve the content. This results in **more reliable and contextually accurate lecture notes**.

The system functions as a **community-driven knowledge platform** where students and instructors can contribute, upvote, and edit lecture transcriptions and summaries, ensuring better accuracy and engagement.

Workflow

1. Lecture Recording & AI Processing

- The student records a lecture using the app.
- AI performs **speech-to-text transcription** using models like Whisper or DeepSpeech.
- LLM models (GPT-4, Pegasus, or BERT) generate **summaries, key topics, and potential questions**.
- AI highlights **unclear or low-confidence segments**, indicating where human input is needed.

2. Crowdsourced Annotation & Collaboration

- Students and instructors can **edit and refine** the AI-generated transcript.
- Community members can **upvote and review** content for quality control.
- Instructors or verified users can **approve or certify** high-quality summaries.
- Users can **annotate difficult sections** or add **explanatory notes**.

3. Interactive Features

- **Discussion Forums:** Users can discuss complex topics within the transcript.
- **AI & Human-Generated Q&A:** AI suggests potential questions; students can improve them.
- **Version Control:** Users can view **changes and improvements over time**.
- **Personalized Study Notes:** Users can **highlight** key points and add private notes.

Key Features & Functionalities

1. AI-Powered Initial Processing

- Speech-to-text conversion with **highlighted uncertainties**.
- AI-generated **lecture summaries**, **important topics**, and **questions**.
- Topic segmentation for **easy navigation**.

2. Community-Driven Refinements

- **Collaborative Editing:** Users refine AI-generated transcriptions and summaries.
- **Voting & Verification System:**
    - Upvotes from multiple users increase credibility.
    - Instructors or verified contributors can mark **official summaries**.

3. Interactive Learning Elements

- **Inline Discussions:** Users can start discussions within the transcript.
- **AI-Suggested Questions:** AI generates **quiz questions**, refined by users.
- **Keyword Search:** Students can **search within lectures** for specific topics.

4. Gamification & Engagement

- **Reputation System:** Users earn points for quality contributions.
- **Badges for Contributors:** Recognizes active and high-quality editors.
- **Leaderboard for Top Contributors:** Encourages participation.

Use Cases

1. Student Reviewing & Refining a Lecture Summary

Scenario: A student attends a lecture but finds AI transcription has inaccuracies.

1. The student accesses the AI-generated transcript.
2. Notices misinterpretations or missing words.
3. Corrects errors and **adds missing explanations**.
4. Other students upvote the corrections, and an instructor verifies them.
5. The improved version becomes the **official summary**.

2. Instructor Enhancing Summarization for Students

Scenario: A professor wants to ensure students get **accurate and structured notes**.

1. The instructor accesses an AI-generated summary of their lecture.

2. Reviews and **reorganizes key points** to match course structure.
3. Adds additional **explanations or external references**.
4. Approve and share the refined summary with students.

3. Student Asking & Answering Questions from the Lecture

Scenario: A student is preparing for an exam and wants clarification on a topic.

1. The student **searches for a concept** in past lecture notes.
2. Find a section with **AI-generated questions**.
3. If unclear, the student posts a query in the **discussion forum**.
4. Peers and AI-generated responses help **clarify doubts**.
5. The student contributes an improved explanation to help others.

Pros & Cons

Pros:

- **Higher Accuracy & Context Understanding:** AI errors are corrected by students and instructors.
- **Encourages Collaborative Learning:** Students engage in discussions and annotations.
- **Gamification & Reputation Building:** Encourages active contributions.
- **Blends Automation with Human Intelligence:** AI speeds up transcription, while users improve quality.
- **Improves Note-Taking for All Students:** Helps students who struggle with note-taking.

Cons:

- **Requires Active User Engagement:** Success depends on student and instructor participation.
- **Potential for Misinformation:** Requires moderation to prevent incorrect edits.
- **Version Control Challenges:** Needs a robust system to manage conflicting edits.
- **User Bias in Upvotes & Annotations:** Some answers may get more visibility due to popularity rather than correctness.

Differentiation from Existing Solutions

Feature	Standard AI Transcription Apps	Collaborative Note-Taking Apps	Hybrid Approach (Proposed)
Speech-to-Text	✅	❌	✅
AI Summarization	✅	❌	✅
Collaborative Editing	❌	✅	✅
Instructor Verification	❌	❌	✅
Crowdsourced Notes & Q&A	❌	✅	✅
Gamification & User Reputation	❌	❌	✅

This approach is unique because it **combines AI transcription, summarization, and crowdsourced annotation**, creating a **community-driven knowledge repository**.

Learning Prototype Plan for Sprint 1

Unknowns & Testing Strategy

Unknowns	Learning Prototype
Will students actively refine AI-generated summaries?	Create a **mock UI with dummy lecture content** and conduct user testing.
How effective is AI in generating accurate topic-based summaries?	Compare **AI-generated vs human-generated summaries**.
Can gamification drive engagement?	Introduce a **simple points system** in early prototypes and track user interaction.

Prototyping Steps

1. **Prototype a Basic Transcription & Editing System**

    - Build a simple **text editor** where users can edit AI-generated transcriptions.
    - Implement **version history tracking** for edits.
2. **Simulate User Engagement**
    - Invite students to interact with a prototype discussion board.
    - Monitor **participation levels and feedback**.
3. **Test Gamification & Motivation**
    - Assign **points for contributions and upvotes**.
    - Observe if this increases engagement in the prototype.

Approach 3: Knowledge-Based Chatbot with Curated Content

Overview

This approach involves **professors or TAs uploading structured course materials** (documents, lecture recordings, reference links) to the app. A **ChatGPT-style chatbot** is then trained on this curated dataset, allowing students to ask questions and receive precise answers based on the provided content.

Instead of relying on generic AI-generated knowledge, the chatbot operates within the **boundaries of the uploaded course materials**, ensuring accuracy and relevance.

Workflow -

1**Content Upload by Professors/TAs**

- **Professors or TAs upload** lecture notes, slides, textbooks, research papers, and other course materials.
- The system **ingests and organizes** this data into a structured knowledge base.

2**AI Processing & Indexing**

- The system processes PDFs, Word documents, and transcribed lecture recordings.
- **Vector embeddings** (e.g., using OpenAI's Embeddings or FAISS) allow **semantic search** across documents.

3**Chatbot Interaction**

- Students **ask questions** in a ChatGPT-style interface.

- The chatbot **retrieves relevant content** from the curated dataset and **generates answers** using **RAG (Retrieval-Augmented Generation)** techniques.

4**Enhanced Features**

- **Citation & Source Linking:** Every answer includes **references to the original material**.
- **Multimodal Support:** Can answer using **text, diagrams, or short AI-generated summaries**.
- **Follow-up & Contextual Understanding:** Students can **refine their queries** based on previous responses.

Pros & Cons

Pros:

- Ensures **accuracy** by relying only on course-approved materials.
- Helps students get **immediate, relevant answers** instead of searching through large documents.
- Allows **continuous learning** by refining the chatbot based on user queries.
- **Scalable** – works across multiple courses and semesters.

Cons:

- Requires **professors/TAs to regularly update content**, which may add workload.
- Initial setup requires **efficient document parsing and knowledge structuring**.
- The chatbot may **struggle with complex, multi-step reasoning** without deeper fine-tuning.

Use Cases

Use Case 1: Quick Concept Clarifications

- A student is confused about **gradient descent** after a lecture.
- They ask the chatbot, *"Can you explain gradient descent with an example?"*
- The chatbot retrieves explanations **from the professor's slides** and provides a concise answer with a linked reference.

Use Case 2: Homework & Exam Preparation

- Before an exam, a student asks, *"What are the key topics for the midterm?"*

- The chatbot pulls topics **from the syllabus and past exam patterns** uploaded by the professor.

Use Case 3: Understanding Code Examples

- A student is stuck on an **algorithm assignment**.
- They upload their **code snippet** and ask, *"Why is this giving an error?"*
- The chatbot **analyzes the code** using contextual course material and suggests fixes.

Prototype & Testing Plan

Learning Prototype Goals:

- **Test AI retrieval accuracy** – Does the chatbot provide correct and relevant answers?
- **Evaluate usability** – Do students find the chatbot more effective than searching documents manually?
- **Measure engagement** – How often do students use the chatbot, and what questions do they ask?

Prototype Plan:

Step 1: Build a **document ingestion pipeline** (PDF, Word, transcripts).
Step 2: Implement a **basic semantic search and chatbot UI**.
Step 3: Conduct **user testing with a small dataset**, iterating based on feedback.

This approach provides an **AI-powered assistant** that **enhances student learning** while ensuring **professor-approved accuracy**.

Technical Discussion for Each Approach:

Refer to Technical Discussion (Link to repository, **From Sprint 3**) for technical discussion of Approach 1 added in **Sprint 1.**

Approach 1: AI-Assisted Note-Taking with Q&A

Technology Stack

- **Speech-to-Text:** OpenAI Whisper, DeepSpeech
- **LLM for Q&A:** GPT-4, Claude
- **Frontend:** React Native (for mobile), Next.js (for web)
- **Backend:** FastAPI, Firebase
- **Database:** PostgreSQL (lecture transcripts, user queries)

Challenges & Solutions

- **Latency in transcription?** → Use lightweight Whisper models for faster response times.
- **Handling diverse accents?** → Train with diverse datasets to improve ASR accuracy.
- **AI Q&A response accuracy?** → Implement **Retrieval-Augmented Generation (RAG)** for lecture-context answers.

Approach 2: Hybrid Crowdsourced Annotations

Technology Stack

- **Speech-to-Text:** Whisper, Vosk
- **Collaborative Editing:** Firebase Firestore
- **Version Control:** Git-like tracking for user edits
- **Gamification:** Django-based leaderboard

Challenges & Solutions

- **Preventing misinformation?** → Implement an upvote/downvote & instructor verification system.
- **User participation concerns?** → Add incentives like badges, recognition from professors.

Approach 3: Knowledge-Based Chatbot with Curated Content

Technology Stack

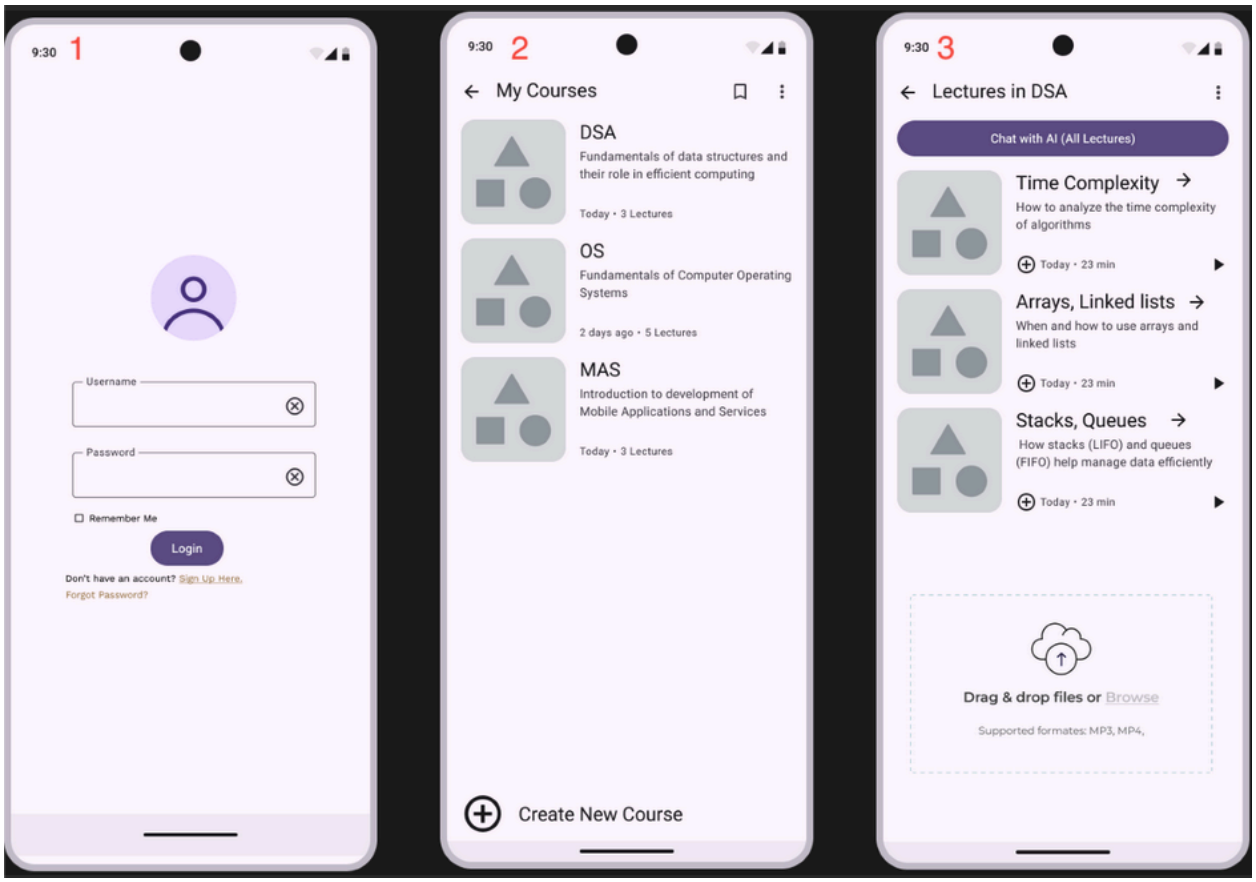
- **Document Ingestion:** LangChain + FAISS for semantic search
- **Chatbot:** GPT-4 with RAG
- **Frontend:** React.js (web), Swift/Kotlin (mobile)
- **Database:** PostgreSQL (storing indexed lecture materials)

Challenges & Solutions

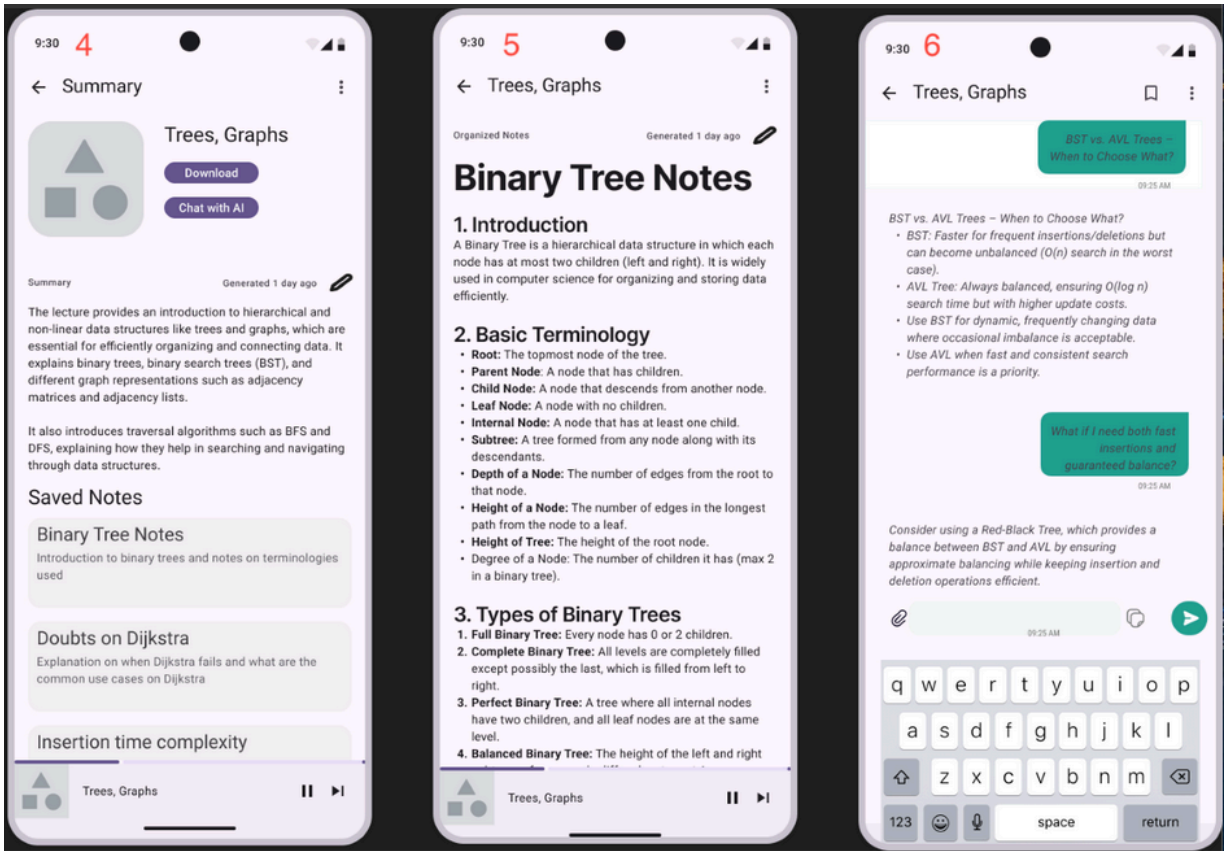
- **Ensuring high-quality answers?** → Use **instructor-approved** documents as the knowledge base.
- **Can students trust AI responses?** → Provide **citations for every answer** linked to original materials.

Mockups

Solution Approach 1: (Refer to the image here)

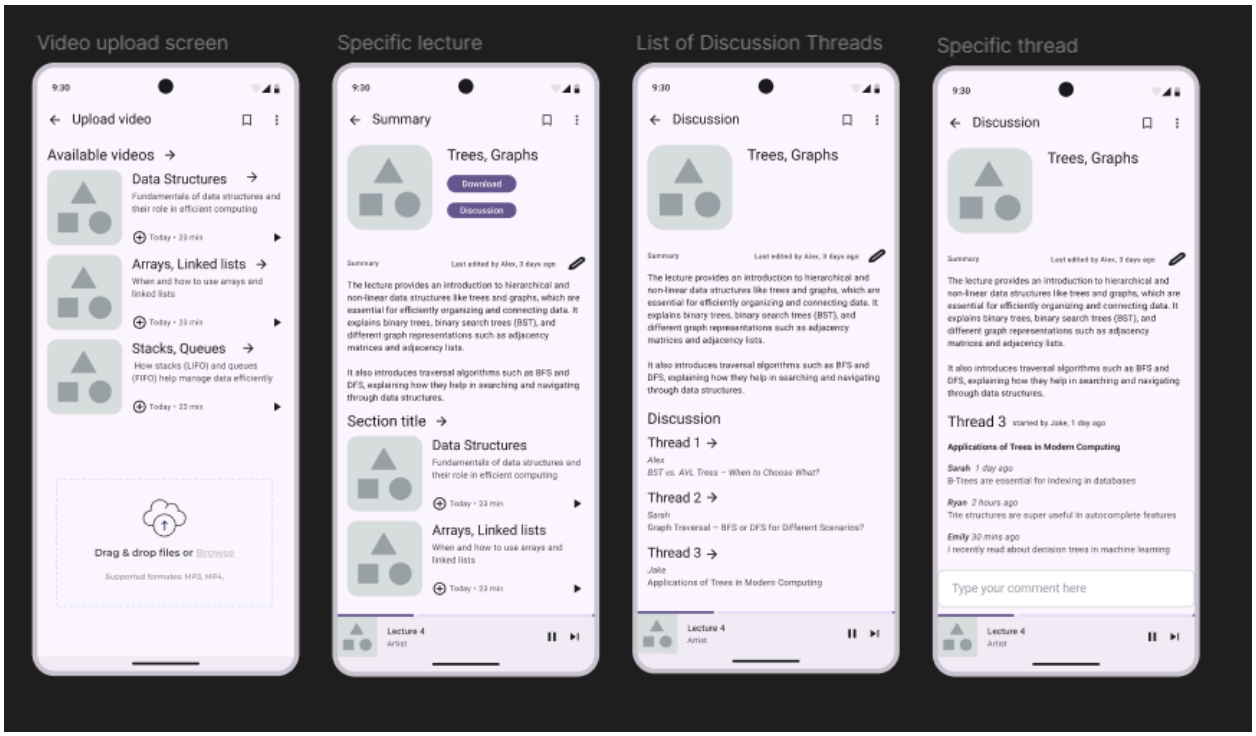


1. Login Screen
    - Simple login page with username and password fields.
    - Links for "Sign Up" and "Forgot Password" for new users and password recovery.
2. Courses Dashboard
    - Displays "My Courses" with relevant course cards.
    - Each course card shows name, brief description, and stats.
    - Option to "Create New Course" at the bottom.
3. Course Lecture List
    - Lists all lectures under a selected course (DSA in this case).
    - A prominent "Chat with AI (All Lectures)" button for AI chatbot which will have context across all the lectures.
    - File upload allows students to upload new lecture recordings.



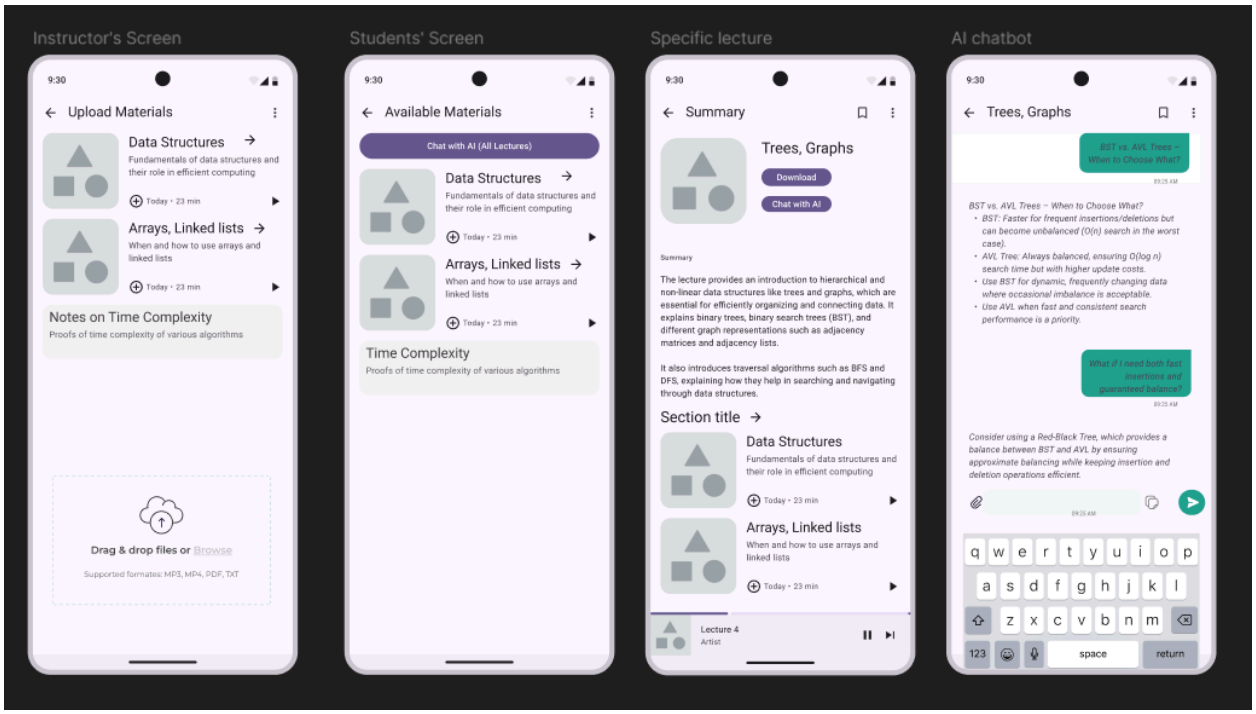
4. Lecture Details Page
    - The AI-generated summary of the lecture and an option to download the lecture resources (notes) and summary to make it available offline. “Chat with AI button” for Q&A and doubt resolution.
    - Notes section containing AI generated organized notes and excerpts from the chats saved by users
5. Notes Page
    - Well-structured notes organized across sections along with highlighting.
    - Automatically formatted sections for readability.
6. AI-Powered Chat for Lecture Assistance
    - Chat interface for students to interact with AI.
    - AI provides contextual answers based on lecture materials

Solution approach 2:



- Upload screen
    - Lists all lecture videos uploaded.
    - File upload allows students to upload new lecture recordings.
- Lecture Details Page
    - The AI-generated summary of the lecture and an option to download the lecture resources (notes) and summary to make it available offline. “Discussion” button for Q&A and doubt resolution.
    - Notes section containing AI-generated organized notes.
- Discussion screen:
    - Shows the summary of the lecture
    - Shows a list of discussion threads
- Specific thread screen:
    - Shows the summary of the lecture
    - Shows the conversations of that thread.

Solution approach 3:



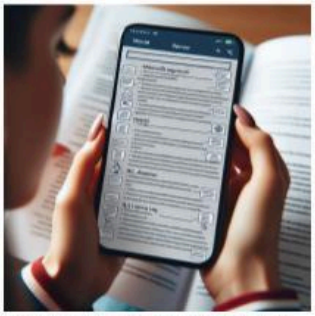
- Instructor’s screen
    - Lists all lecture videos uploaded. Accessible only to the instructors.
    - File upload allows instructors to upload lectures and other materials.
- Course Lecture List
    - Lists all lectures under a selected course (DSA in this case). Available to the students.
    - A prominent "Chat with AI (All Lectures)" button for AI chatbot which will have context across all the lectures.
- Lecture Details Page
    - The AI-generated summary of the lecture and an option to download the lecture resources (notes) and summary to make it available offline. “Chat with AI button” for Q&A and doubt resolution.
    - Notes section containing AI-generated organized notes.
- AI-Powered Chat for Lecture Assistance
    - Chat interface for students to interact with AI.
    - AI provides contextual answers based on lecture materials uploaded by the instructor.

Storyboards

Solution approach 1

The student is attending a lecture which will be recorded by the professor and made available to everyone

The student then uploads the recording on the app



Once the lecture is uploaded. The app generates meeting summaries, follow-up questions, etc

All the recording threads can be organized based on the course, module, or even topics

The student can ask further questions related to the lecture and get AI-generated answers

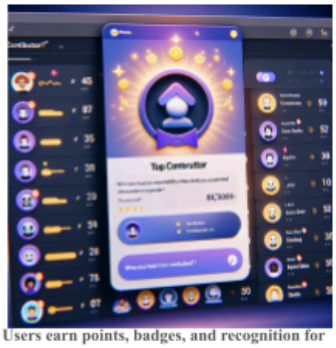
This storyboard illustrates a seamless process where lectures are recorded by professors and uploaded to an app for student access. The app enhances learning by generating AI-powered meeting summaries, follow-up questions, and insights. It organizes recordings into threads based on courses or topics for easy navigation. Students can interact with the content by asking questions and receiving AI-generated answers, making the learning experience more engaging and personalized.

Solution approach 2

Students are attending the lecture, which is being recorded, and the generated transcript will be available to them shortly

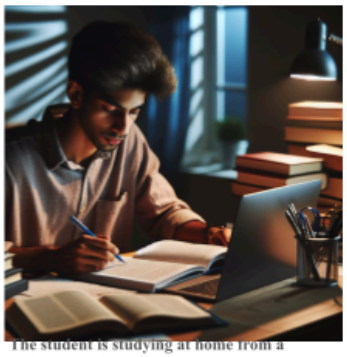
Students and instructors refine transcripts, add explanations, and certify high-quality summaries for better accuracy

Users can start discussions thread within transcripts to clarify complex topics collaboratively



Users earn points, badges, and recognition for quality contributions for a gamified experience

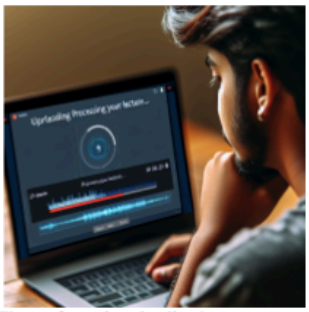
The professor is reviewing to refine AI-generated notes to ensure accuracy and alignment with course objectives



The student is studying at home from a polished version of lecture notes on the app

This storyboard focuses on collaborative learning through lecture transcripts. After lectures are recorded, transcripts are generated and shared with students and instructors. Users refine these transcripts collaboratively, adding explanations and verifying accuracy to produce high-quality summaries. Discussion threads within transcripts allow for clarification of complex topics, while gamified contributions reward users for their input. Professors review the refined notes before publishing them, ensuring alignment with course objectives.

Solution approach 3



The professor is uploading lecture materials to the app

The app then transcribes the lecture, segments it into logical topics, and highlights important keywords for focused learning using AI model running locally

The app creates clear, concise summaries of the lecture content using AI models, combining transcript data and external resources

The app generates interactive tools like quizzes to test comprehension, while flashcards reinforce retention of key concepts

The student can review summaries, flashcards, quizzes, and concept maps anytime—even offline—to master concepts before exams

This storyboard highlights the use of AI to transform lecture materials into interactive study tools. Professors upload lecture content, which the app transcribes, segments into topics, and summarizes using AI models. The app further enhances learning by creating quizzes, flashcards, and concept maps to reinforce understanding. Students can access these resources offline, enabling them to master concepts at their own pace and prepare effectively for exams or assignments.

Prototype Implementation

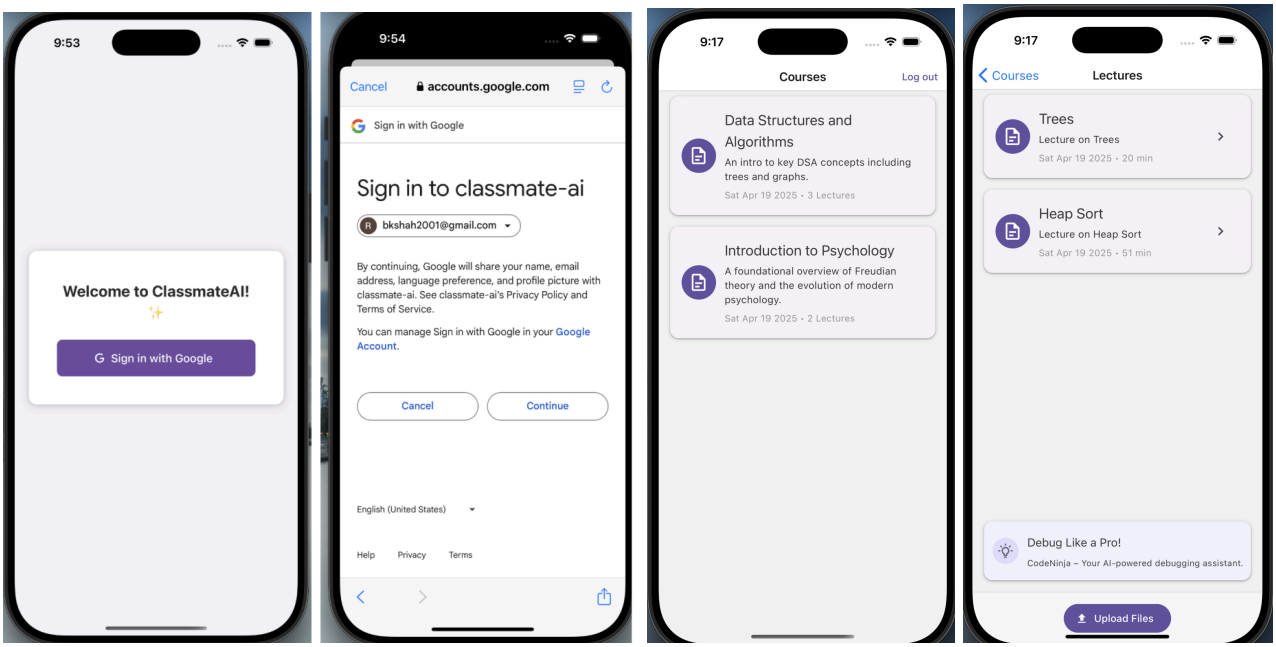
For Sprint 3, we developed a functional prototype of **Classmate AI**, integrating **AI-assisted lecture note generation, real-time transcription, automated summarization, and an AI-powered Q&A system**. This prototype enables students to upload lecture audio, generate structured summaries, and interact with AI for clarifications.

For Sprint 4, we extended the Classmate AI prototype by refining the lecture processing pipeline and integrating an AI-powered chat system for contextual question answering. The updated prototype enables students to upload lecture recordings, receive structured AI-generated notes, and interact with an intelligent chat assistant that can answer questions based on lecture content.

The primary objectives of this sprint included:

- Enhancing lecture detail pages with interactive features and note previews
- Implementing AI-powered Q&A using contextual chat based on lecture summaries
- Improving the UI to support note exploration, chat, and saved responses
- Extending backend support for chat sessions and lecture-specific context retrieval

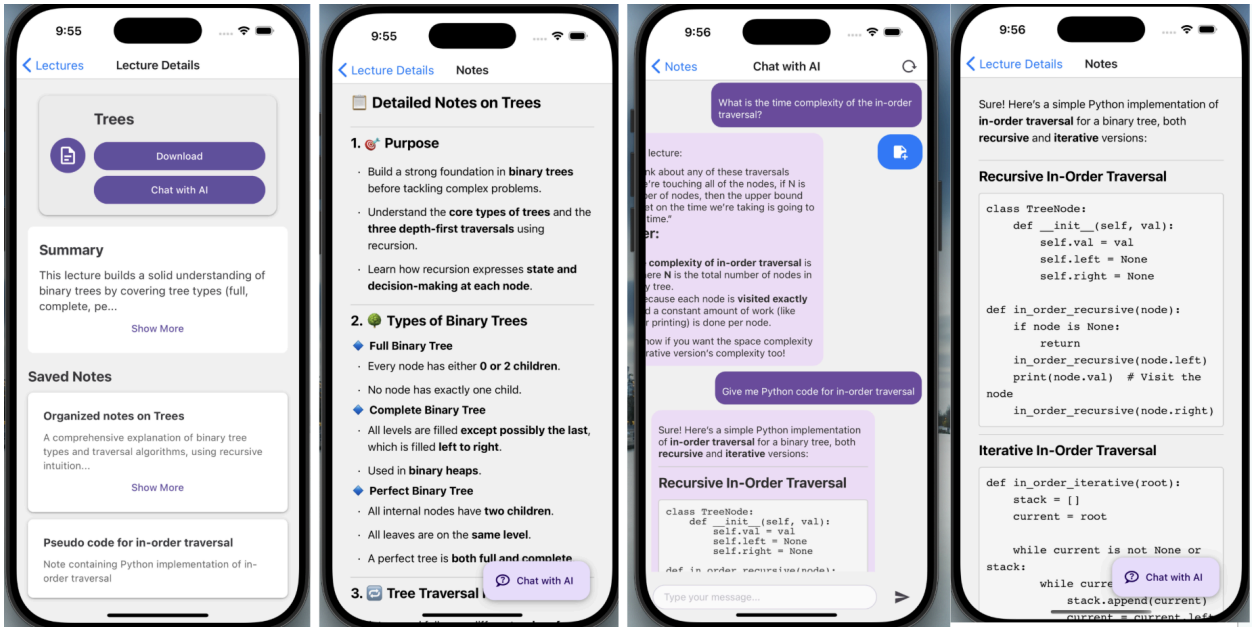
The system allows students to upload lecture audio files, which are transcribed and summarized using AI models. Summaries are displayed in an organized format, and students can now initiate a chat session to clarify concepts. The AI assistant references structured notes and the lecture transcript to generate accurate, course-specific responses.



1. Login Screen: Provides a Firebase powered Google Auth.

2. Courses Screen: Displays a list of available courses with titles, descriptions, and the number of lectures.

3. Lecture Screen: Upon selecting a course, users can view its associated lectures. Each lecture card provides a brief description, date, and duration. The “Upload Files” button allows students to upload new lecture audio files for processing. Lecture screen shows targeted ads based on the course. This is a placeholder for now.



4. Lecture Details Screen: Shows detailed information about a selected lecture. This screen now includes AI-generated summaries, structured organized notes, and access to the contextual Q&A feature via the “Chat with AI” button. Students can also download lecture summaries for offline review.

5. Notes Screen: Displays detailed AI-generated study notes with hierarchical structure. Notes are presented using section headers, bullet points, and keyword highlights. 2Technical lectures could also include annotated code snippets or pseudocode where applicable. Chat responses saved by the student are also displayed in this viewer.

6. AI-Powered Chat Screen: Students can ask follow-up questions based on the lecture content. The AI assistant references the structured notes and generates contextual responses. For example, a query such as “What is the time complexity of in-order traversal?” prompts a targeted explanation along with code samples. Users can save any chat exchange as part of their study notes for later reference.

This iteration focused on improving post-lecture engagement by integrating AI assistance directly into the note-taking workflow. By embedding the chat feature within the lecture details and notes screens, students can explore material more deeply without switching between tools. All chat interactions are tied to specific lectures, ensuring that the AI assistant provides relevant, context-specific answers.

These enhancements ensure a cohesive learning experience where students can review structured content, clarify doubts, and build a reliable study archive from their lectures.

Second Platform Implementation

For Sprint 5, we implemented a second platform which is a webapp that lets instructor upload transcripts, review the summary generated by AI and validate it. Once the summary is validated, it will be visible to the students as instructor validated, which will increase student's trust in the AI generated notes.

Overview of each screens -

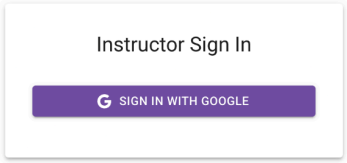
1. Login Screen: Provides a Firebase powered Google Auth.

2. Courses Screen: Displays a list of available courses with titles, descriptions, and the number of lectures. Instructors can continue to add new courses using the "Create New Course" button at the bottom of the screen.

3. Lecture Screen: Upon selecting a course, instructors can view its associated lectures. Each lecture card provides a brief description, date, and duration. The "Upload Files" button allows instructors to upload new lecture audio files for processing.

4. Lecture Validation Screen: Once the audio file is processed, the summary is generated and ready to be validated by the instructor. The instructor can review the AI generated summary, edit it if required and validate it. Once it is validated, the students can view the validation mark on their app.

Screenshots of the webapp -



ClassmateAI Instructor Panel LOGOUT (RAJSHAH2320)

Your Courses

Data Structures and Algorithms

An intro to key DSA concepts including trees and graphs.

3 Lecture(s)
Last Updated: 4/19/2025

Introduction to Psychology

A foundational overview of Freudian theory and the evolution of modern psychology.

2 Lecture(s)
Last Updated: 4/19/2025

ClassmateAI Instructor Panel LOGOUT (RAJSHAH2320)

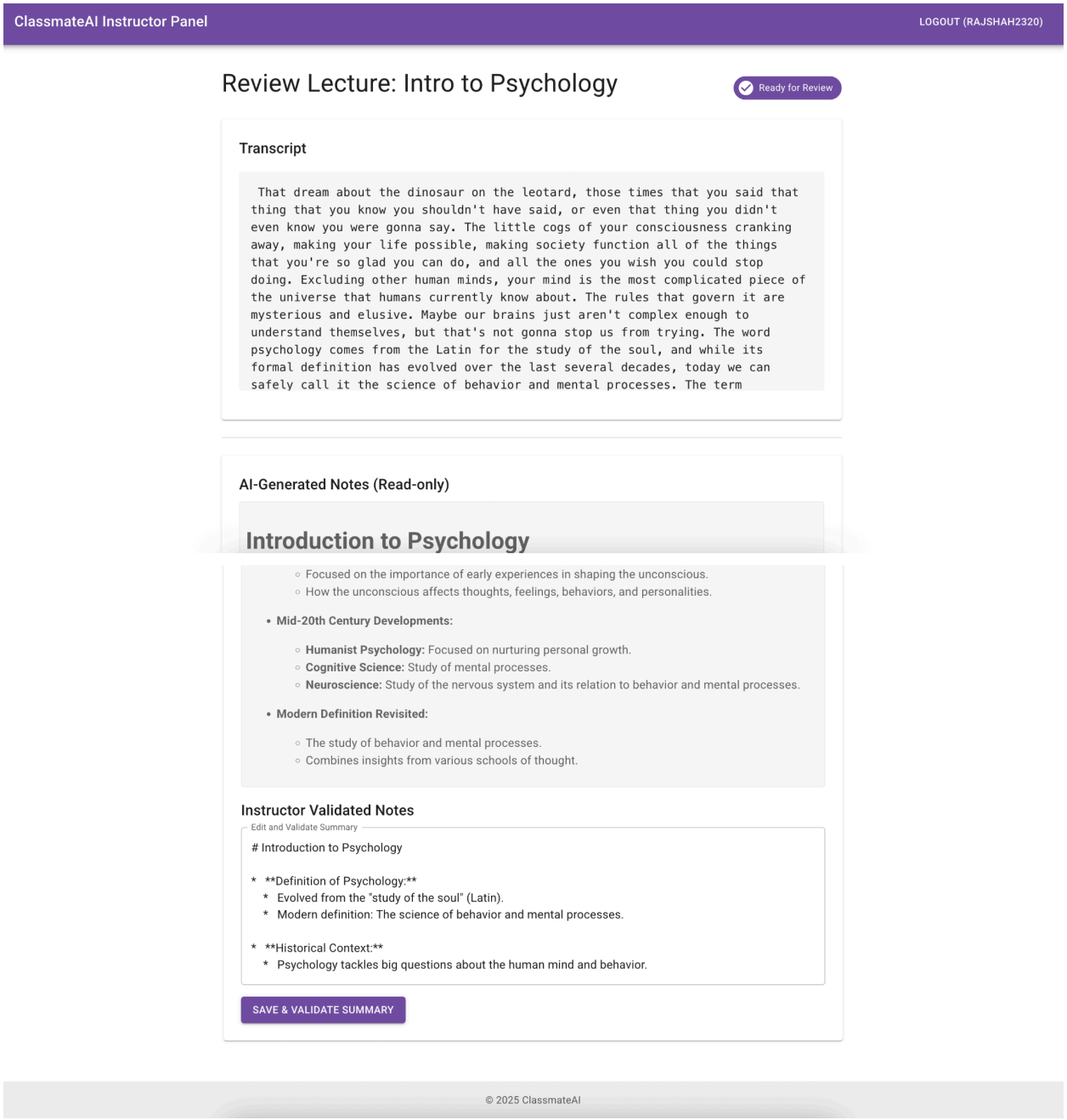
Lectures

UPLOAD NEW LECTURE AUDIO

Lecture "Intro to Psychology - 2" uploaded successfully. Processing will start.

Intro to Psychology - 1
Uploaded: 4/19/2025 | Duration: 11 min
Ready for Review

Intro to Psychology - 2
Uploaded: 4/19/2025 | Duration: 21 min
Processing



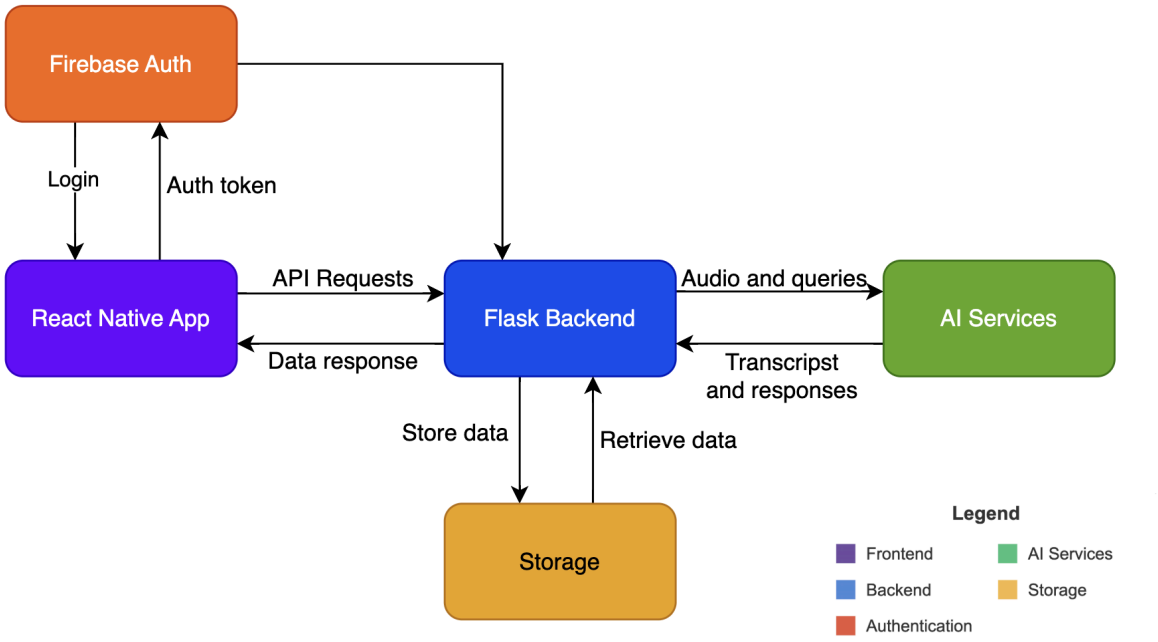
Technical Discussion (Link to repository)

Platform and Architecture Overview

As part of our Sprint 3 and 4 learning prototype, it is built using a hybrid architecture that combines a React Native mobile frontend with a Flask-based Python backend. This

approach allows us to deliver a cross-platform mobile experience while leveraging powerful AI capabilities for audio transcription, summarization, note generation, and interactive Q&A via chat functionality.

High-Level System Architecture

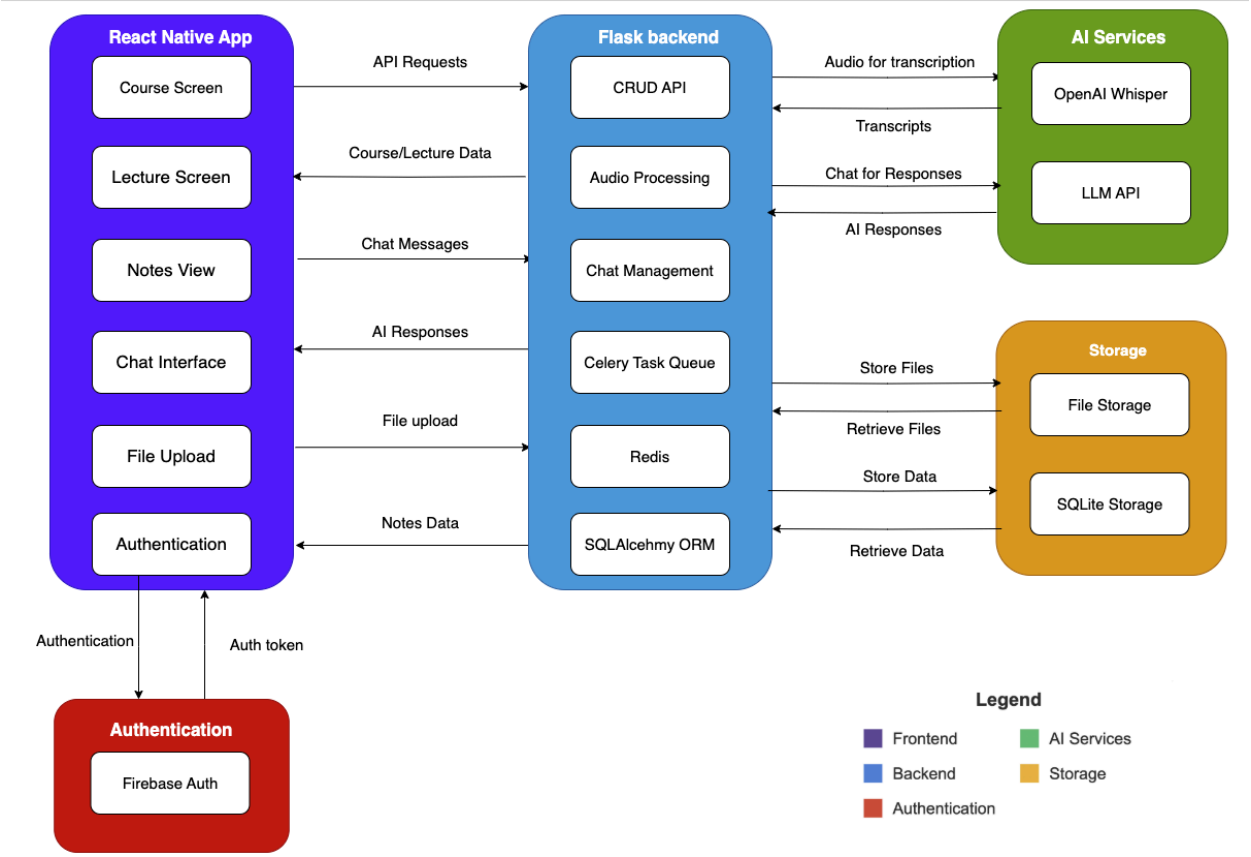


The first diagram shows the core components of our system:

1. Firebase Auth—Authentication service for user identity management
2. React Native App—The mobile frontend that users interact with directly
3. Flask App—The central backend that handles business logic and API requests
4. External Modules—AI and processing services integrated into our system
5. Storage—Data persistence layer for courses, lectures, and files

This high-level architecture follows a client-server model where the React Native mobile app authenticates users through Firebase Authentication and communicates with the Flask backend through RESTful APIs. The backend connects to external modules for specialized AI functions and persists data to the storage layer. This separation of concerns provides modularity and makes the system easier to maintain and scale.

Detailed Backend Flow



The second diagram illustrates the detailed data flow for lecture processing:

Enhanced Components:

- React Native App: Our mobile frontend for student interaction
- Flask Backend: Server handling API requests and business logic

Storage Components:

- SQLite storage: Store all lecture- and course related data
- File Storage: Contains uploaded audio files

External Services:

- OpenAI Whisper: Handles audio transcription
- LLM API: Generates lecture summaries

1. Initial Request Flow

When a student or instructor wants to upload a lecture recording:

1. The React Native App initiates the process through the CRUD APIs, specifically using:
    - The app first creates a lecture entry with metadata using POST /courses/{courseId}/lectures
    - Once the lecture is created, the audio file is uploaded via POST /courses/{courseId}/lectures/{lectureId}/upload-audio
2. File Transfer Process:
    - The mobile app reads the audio file from the device storage using Expo's DocumentPicker
    - The file is packaged as multipart form data and sent to the Flask backend
    - The uploadAudioFile function in uploadAudio.tsx handles this with timeout protection (30 seconds as defined in constants)

2. Backend Processing

Once the Flask backend receives the audio file:

1. Initial Storage:
    - The audio file is saved to the uploads directory with a filename based on the lecture ID
    - The Flask backend updates the lecture record in lectures.json with:
        - The file path (audioPath)
        - Upload timestamp (uploadDate)
        - Audio duration in seconds (duration)
        - Processing status (summaryStatus set to "IN_PROGRESS")
2. Transcription Process:
    - The backend spawns a background thread via threading.Thread to avoid blocking the response
    - This thread calls process_transcription which invokes the Whisper API
    - The transcribe_audio function loads the Whisper model and processes the audio file
    - The resulting transcript is saved both in memory and as a text file
3. Summarization Process:
    - Once transcription completes, the summarize_transcript function is called
    - This uses the summa library to create a condensed version of the transcript
    - The summary is stored alongside the transcript in the lecture record
4. Large Language Model (LLM) Processing:
    - With the transcript ready, the backend makes a request to an LLM API
    - The get_summary function sends the transcript to the LLM

    - The LLM processes the text and returns a structured summary
    - This summary is then stored in the lecture record

3. Response Flow

After processing (or during asynchronous processing):

1. Real-time Status Updates:
    - The mobile app can poll the lecture status using GET /lectures/{lectureId}
    - The backend returns the current processing status (summaryStatus field)
2. Final Content Delivery:
    - Once processing completes, the summaryStatus field changes to "COMPLETED"
    - The mobile app can then display the transcript and summary to the user
    - Users access the full content through dedicated screens in the React Native app

4. Chat Functionality

The fourth diagram illustrates the chat-based Q&A functionality:

Chat Workflow:

1. Chat Creation: Users initiate a new chat session related to a specific lecture
2. Message Exchange: Users send questions and receive AI-generated responses
3. Context-Aware Responses: The system leverages lecture summaries to provide more relevant answers
4. Asynchronous Processing: Responses are generated in the background to maintain UI responsiveness

Chat Components:

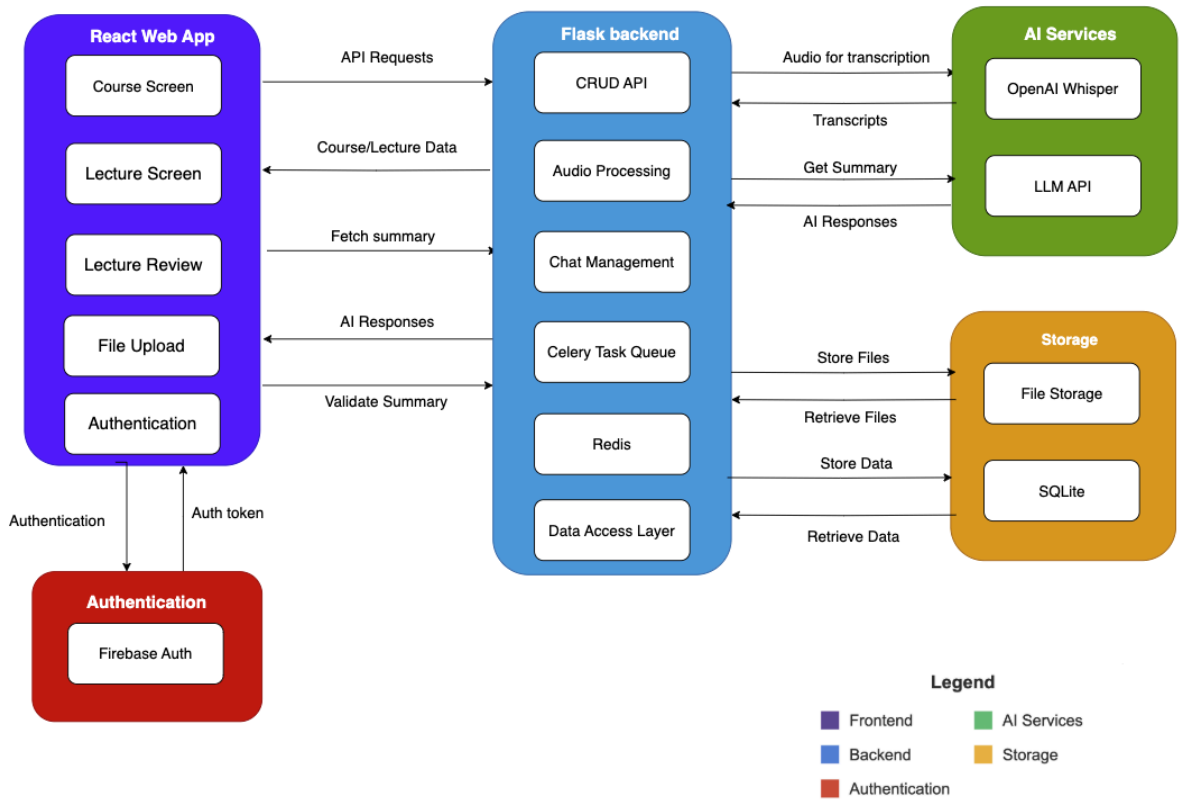
- Chat Sessions: Each chat is associated with a specific lecture and has a unique ID
- Message History: All messages (both user and AI) are stored with timestamps
- LLM AI Integration: Powers intelligent responses using lecture context

Chat Data Flow:

1. Initiation:
    - User starts a new chat for a specific lecture via the mobile app

    - Backend creates a chat session with a unique ID
2. Question Submission:
    - User submits a question through the chat interface
    - Question is sent to the backend and stored in the chat history
3. AI Processing:
    - Backend spawns a background thread to process the question
    - For the first message, lecture summary is automatically added as context
    - Query is sent to Perplexity API with appropriate formatting instructions
4. Response Delivery:
    - AI response is received and stored in the chat history
    - Mobile app retrieves the updated chat history
    - Response is displayed to the user in Markdown format

5. Instructor validation (Added in Sprint 5)



The instructor validation workflow is a critical quality assurance component of ClassmateAI, ensuring that AI-generated content meets academic standards before being made available to students.

Validation Workflow:

1. Content Generation: After lecture audio is processed, AI generates initial transcripts, summaries, and study notes
2. Instructor Review: Faculty members review the AI-generated content through a dedicated interface
3. Manual Editing: Instructors can edit and refine all aspects of the content
4. Validation Action: Instructors officially validate the content, marking it as approved
5. Status Update: The system updates content status to "VALIDATED" for student consumption

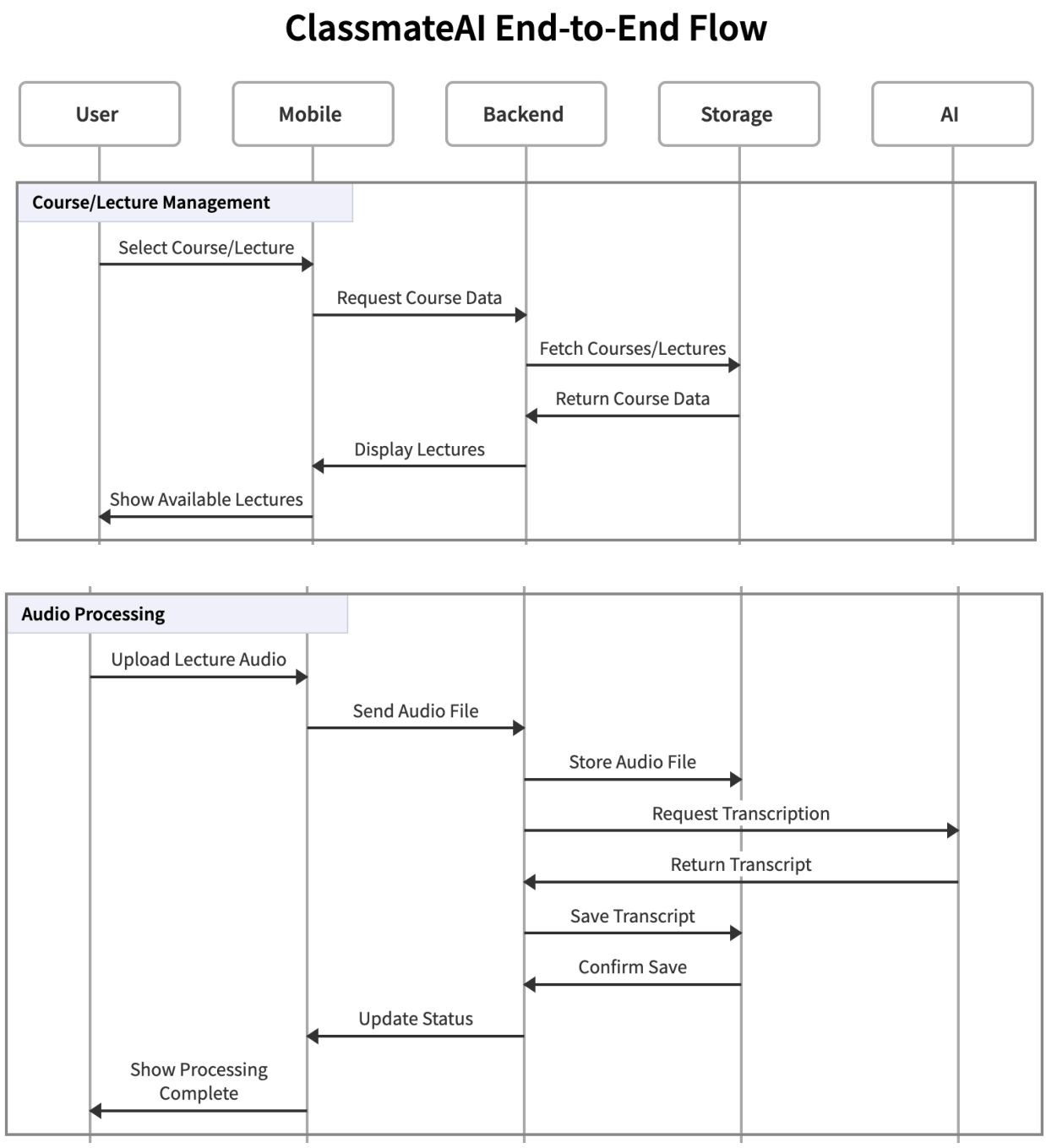
Validation Components:

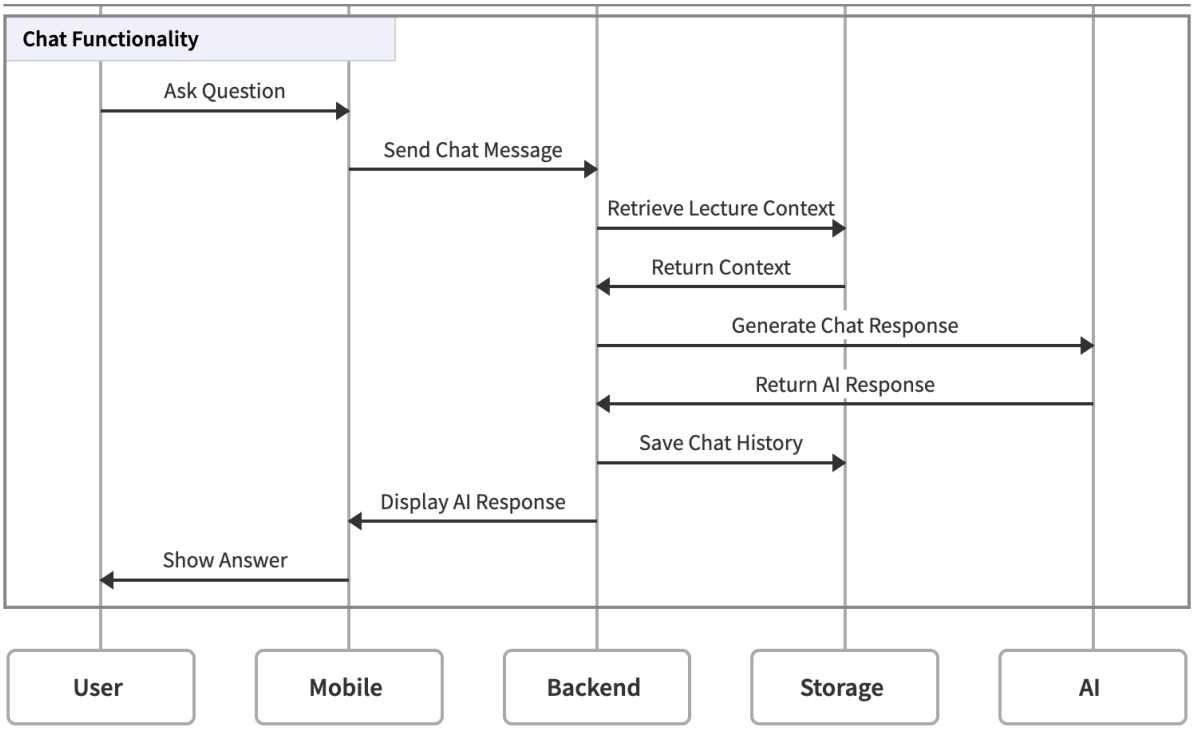
- Review Interface: A specialized UI for instructors to evaluate and modify AI-generated content
- Edit Controls: Rich text editors for modifying transcripts and notes
- Validation Status: Content flagging system that tracks validation state
- Version History: Records of original AI content and instructor modifications

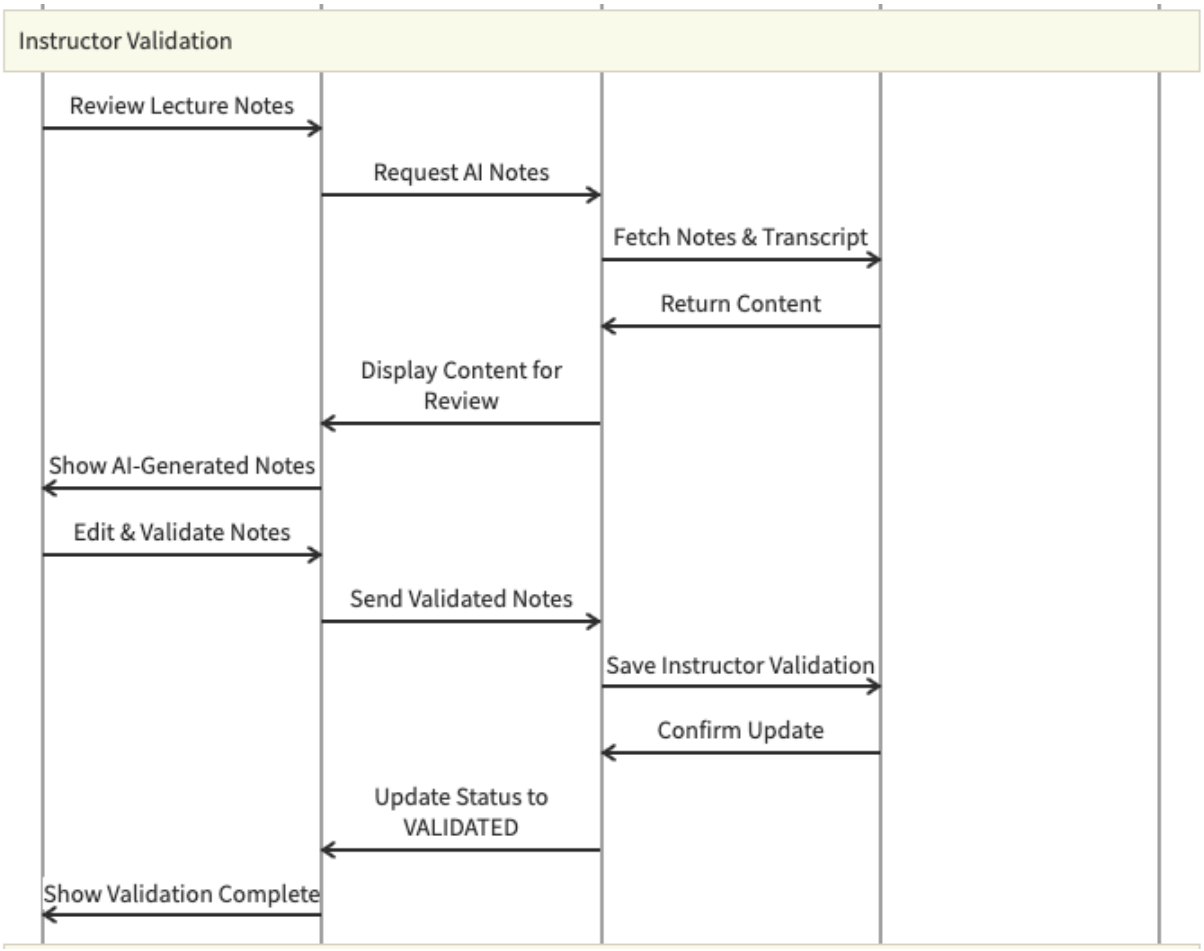
Validation Data Flow:

1. Initial Access:
    - Instructor selects a processed lecture from the dashboard
    - Backend retrieves the AI-generated content (transcript, summary, notes)
    - Content is displayed in the review interface with editing capabilities
2. Content Editing:
    - Instructor reviews for accuracy, clarity, and completeness
    - Makes necessary corrections or enhancements to the material
    - System provides real-time feedback on changes
3. Validation Submission:
    - The instructor approves content by clicking "Save & Validate."
    - Modified content is sent to the backend via the validateSummary API
    - Backend updates lecture records with validated content and timestamps
4. Status Propagation:
    - Lecture status is updated to "VALIDATED" in the database
    - Visual indicators on the dashboard reflect validated status
    - Content becomes available to students through the mobile app

Swimlane diagram for End-to-End chat functionality




Chat Functionality
Ask Question
Send Chat Message
Retrieve Lecture Context
Return Context
Generate Chat Response
Return AI Response
Save Chat History
Display AI Response
Show Answer
User
Mobile
Backend
Storage
AI



Technology Choices

Frontend

- React Native: Chosen for cross-platform mobile development, allowing us to target both iOS and Android with a single codebase
- Expo: Provides a simplified development workflow and access to native device features
- React Navigation: Implements navigation between screens with a stack-based approach
- React Native Paper: Offers Material Design components for a polished UI

Backend

- Flask: A lightweight Python web framework that provides flexibility for API development
- Whisper: OpenAI's speech recognition model for high-quality audio transcription
- Summa: Text summarization library for initial processing of transcripts
- Perplexity AI: Used for generating detailed study notes from lecture content
- JSON Files: Simple storage solution for course and lecture metadata
- File System Storage: Manages audio files and generated transcripts

Data Flow and Storage

Our application manages several types of data:

1. Course Data: Metadata about courses, including title, description, and associated lectures
2. Lecture Data: Information about individual lectures, including title, description, audio path, transcript, summary, and notes
3. Audio Files: Raw lecture recordings uploaded by instructors
4. Processed Content: Transcripts, summaries, and study notes generated from audio files

This data is stored across

- SQLite storage: Structured metadata storage
- File System: Raw audio files and generated text content
- In-Memory Processing: Temporary data handling during transcription and summarization

REST API Endpoints

Our backend exposes the following key API endpoints:

Course Management

1. GET /courses

- Purpose: Retrieves all available courses
- Response Example:

```
[
  {
    "courseID": "550e8400-e29b-41d4-a716-446655440000",
    "courseName": "Introduction to Computer Science",
    "courseSummary": "Fundamentals of programming and computer systems",
    "lastUpdated": 1709911245.789,
    "lectures": ["7f4e6a8b-5c3d-42e1-9f0a-123456789012"]
  }
]
```


2. POST /courses

- Purpose: Creates a new course
- Request Example:


```
{
  "courseName": "Data Structures and Algorithms",
  "courseSummary": "Advanced programming concepts focusing on data structures"
}
```


- Response: Returns the created course object with a generated ID

Lecture Management

1. GET /courses/{course_id}/lectures

- Purpose: Retrieves all lectures for a specific course
- Response Example:

```
[
  {
    "id": "7f4e6a8b-5c3d-42e1-9f0a-123456789012",
    "title": "Introduction to Arrays",
    "description": "Lecture on array data structures",
    "duration": "45 min",
    "date": "Mon Mar 04 2024"
  }
]
```


2. POST /courses/{course_id}/lectures

- Purpose: Creates a new lecture within a course
- Request Example:


```
{
  "lectureTitle": "Sorting Algorithms"
}
```


- Response: Returns the created lecture object with a generated ID

3. POST /courses/{course_id}/lectures/{lecture_id}/upload-audio

- Purpose: Uploads an audio file for a specific lecture
- Request: Multipart form data containing the audio file
- Response: Returns the updated lecture object with processing status

4. GET /lectures/{lecture_id}

- Purpose: Retrieves details for a specific lecture
- Response Example:

```
{
  "lectureID": "7f4e6a8b-5c3d-42e1-9f0a-123456789012",
  "lectureTitle": "Sorting Algorithms",
  "audioPath": "uploads/7f4e6a8b-5c3d-42e1-9f0a-123456789012.mp3",
  "uploadDate": 1709911245.789,
  "duration": 2730.45,
  "transcript": "Today we'll be discussing sorting algorithms...",
  "description": "Lecture on sorting algorithms",
  "summary": "This lecture covers comparison-based and linear sorting algorithms..
  "summaryStatus": "COMPLETED",
  "lastUpdated": 1709911305.123,
  "notes": "# Sorting Algorithms\n\n## Comparison-based Sorts\n- Quick Sort\n- Mer
}
```


5. DELETE /lectures/{lecture_id}

- Purpose: Deletes a specific lecture
- Response: Confirmation of successful deletion

6. GET /lectures/{lecture_id}/chat

- Purpose: Creates a new chat session for a specific lecture
- Response: Returns a newly created chat ID
- Status Code: 201 on success

7. GET /chat/{chat_id}

- Purpose: Retrieves all messages for a specific chat session
- Response: Returns an array of message objects
- Status Code: 200 on success, 404 if chat not found

8. POST /chat/{chat_id}

- Purpose: Adds a new message to a chat session and triggers AI response


```
{
  "type": "User",
  "message": "Can you explain the concept of neural networks from this
}
```

- Response: Confirmation of message addition
- Status Code: 201 on success, 400 if message data is invalid, 404 if chat not found
- Note: For the first message in a chat, the lecture summary is automatically added as context

9. DELETE /chat/{chat_id}

- Purpose: Deletes a specific chat session and all its messages
- Response: Confirmation of successful deletion
- Status Code: 200 on success, 404 if chat not found

10. PUT /lectures/{lectureId}/validate-notes

- Purpose: Updates a lecture with instructor-validated notes, marking the content as reviewed and approved for student consumption.
- Request Example: lectureId in the URL params and request body

```
{
  "validated_notes": "# Introduction to Quantum Mechanics\n\nThis lecture covered the
}
```

- Response:

```
json
{
  "message": "Note validated successfully",
  "lecture": {
    "lectureID": "lecture-qp-1",
    "notes": [
      {
        "id": "550e8400-e29b-41d4-a716-446655440000",
        "title": "Introduction to Quantum Mechanics",
        "summary": "Overview of quantum mechanical principles including wave-part
        "content": "# Introduction to Quantum Mechanics\n\nThis lecture covered t
        "instructor_validated": true,
        "date_generated": 1713705432.345,
        "date_validated": 1713709032.345
      }
    ]
  }
}
```

- Status Code: 200 on success, 400 Bad Request if no notes or missing lecture

Implemented vs. Planned Features

Currently Implemented:

- Basic course and lecture management
- Audio file upload functionality
- Transcription of lecture audio using Whisper
- Basic text summarization of transcripts
- Mobile UI for browsing courses and lectures
- Study note generation from transcripts
- End-to-end chat functionality
- User authentication
- Instructor validation of summaries

Code Review

Project Structure

```
.
├── Backend              # Flask + Celery + Whisper + Perplexity/Gemini
│   ├── app.py           # Lightweight (fail-fast backend; defunct)
│   ├── celery-app.py    # Production-grade backend with background tasks
│   ├── ai_agent.py      # LangChain Perplexity/Gemini-based note generator and chatbot
│   ├── Transcription/   # Whisper-based transcription
│   ├── uploads/         # Audio file storage
│   └── requirements.txt  # Backend dependencies
│
├── InstructorUI         # React + MUI web app
│   ├── src/pages/       # Pages like CoursesDashboard, LectureReview
│   ├── src/components/  # Upload, Layout, ProtectedRoute
│   ├── contexts/        # Firebase AuthContext
│   ├── services/        # API service hooks
│   └── App.tsx          # Route + Theme setup
│
└── StudentUI            # React Native mobile app
    ├── components/      # Screens (Course, Lecture, Chat, Notes)
    ├── utils/           # Styles, constants, fetch/upload logic
    ├── index.tsx        # Navigation stack and auth listener
    └── app.json         # Expo config
```

Code Structure and Interaction Patterns

Platform Synchronization

Both the Instructor (web) and Student (mobile) platforms are unified via a shared Flask backend exposing consistent REST API endpoints. These endpoints support CRUD operations on courses and lectures, upload and processing of audio files, and chat interactions.

- Instructor validations through the web app are reflected on the student mobile UI in real time via polling or re-fetching.
- Lecture states (e.g., IN_PROGRESS, COMPLETED, VALIDATED) drive the user experience on both platforms without duplication of logic.

State Management and UI Reactions

StudentUI (React Native):

- Uses React hooks to manage asynchronous state during lecture uploads, transcription progress, and chat interactions.
- Implements rollback logic: if a file upload fails or exceeds timeout limits, the orphaned lecture is deleted from the backend using deleteLecture().
- The LecturesScreen and LectureDetailsScreen conditionally render AI-generated summaries and initiate Q&A sessions using lecture-specific context.

InstructorUI (React + MUI):

- Uses controlled form components for editing AI-generated content.
- Validation and edits are submitted via PUT requests to the backend. The lecture is marked as VALIDATED and updated for all clients.
- Instructors have access to a real-time Markdown-rendered preview alongside editable fields.

Backend Design and Task Offloading

- transcribe_audio_task and generate_notes_task are chained to ensure that downstream processes only start after upstream results are available.

The backend is structured to handle CPU heavy tasks like transcription asynchronously using Celery workers communicating over Redis Server:

- Long-running tasks use retries and backoff strategies for stability.
- Validation APIs (PUT /lectures/{lectureId}/validate-notes) update records with instructor-edited summaries and transition content into a student-visible state.

Error Handling and Resilience

- The mobile app ensures atomic transactions: lecture records are only finalized upon successful audio upload and processing.
- Both platforms check processing status via polling (GET /lectures/{lectureId}), enabling the frontend to reactively show loading indicators or completed content.

Interesting Code Snippets:

1. Asynchronous Task Processing with Celery and Redis

To avoid blocking the main Flask application during resource-intensive operations such as transcription and note generation, long-running tasks are delegated to background workers via Celery, using Redis as the message broker and result backend.

Example: transcribe_audio_task

```python
@celery.task(bind=True, max_retries=3)
def transcribe_audio_task(self, audio_path, lecture_id):
    """Celery task to process and transcribe lecture audio."""
    with app.app_context():
        try:
            logging.info(f"Transcribing audio for lecture {lecture_id} from {audio_path}")
            transcript_val = transcription.transcribe_audio(audio_path)
            logging.info(f"Transcript generated for {lecture_id}")
            summary = summarize_transcript(transcript_val)
            logging.info(f"Summary generated for {lecture_id}")

            lecture = Lecture.query.get(lecture_id)      You, 3 weeks ago • working till chat;
            if lecture:
                lecture.transcript = transcript_val
                lecture.summary = summary
                lecture.summary_status = "TRANSCRIBED"
                lecture.last_updated = datetime.now(timezone.utc).timestamp()
                db.session.commit()

                generate_notes_task.delay(lecture_id)
            else:
                logging.error(f"Lecture {lecture_id} not found.")
        except Exception as e:
            logging.error(f"Error processing transcription for {lecture_id}: {str(e)}")
            raise self.retry(exc=e, countdown=60)
```

Design Highlights:

- @celery.task decorator with bind=True allows retries on failure using self.retry(...).
- Transcription and summarization logic is encapsulated in a background context to keep request-response latency low.
- Redis provides a lightweight, fault-tolerant queue system for asynchronous execution.
- Task chaining (generate_notes_task.delay(...)) supports multi-phase pipelines without additional orchestration.

2. Drop-in LLM Integration via AIAgent Abstraction

The AIAgent class abstracts interaction with language models, making it easy to switch between providers (e.g., Gemini, OpenAI, Perplexity) while maintaining consistent behavior across the application.

Example: AIAgent Implementation

```python
class AIAgent:
    def __init__(self, api_key: str):
        self.llm = ChatOpenAI(
            model="perplexity",
            openai_api_key=api_key,
            openai_api_base="https://api.perplexity.ai",      You, 1 second ago • Uncommitted changes
            temperature=0.4,
        )

        self.content_prompt = ChatPromptTemplate.from_template(…

        self.title_prompt = ChatPromptTemplate.from_template(…

        self.summary_prompt = ChatPromptTemplate.from_template(
            """…
        )

    def generate_notes(self, title: str, transcript: str):
        content_prompt = self.content_prompt.invoke({"transcript": transcript})
        content_response = self.llm.invoke(content_prompt)

        (variable) summary_response: Any
        summary_response = self.llm.invoke(summary_prompt)

        title_prompt = self.title_prompt.invoke({"transcript": content_response.content.strip()})
        title_response = self.llm.invoke(title_prompt)

        return {
            "id": str(uuid.uuid4()),
            "title": title_response.content.strip(),
            "summary": summary_response.content.strip(),
            "content": content_response.content.strip(),
            "date_generated": datetime.now(timezone.utc).timestamp(),
        }
```

Design Highlights:

- Provider-agnostic design allows for seamless migration between LLM APIs.
- Prompts are decoupled and reusable across multiple agents and tasks.
- Supports different usage contexts (note generation, chat responses) with tailored prompts.
- Enables prompt-level experimentation without modifying business logic or API routes.

3. Robust File Upload with Timeout and Rollback

To ensure a seamless user experience and maintain backend data integrity, the frontend implements timeout-aware audio upload logic. If the upload fails or exceeds the configured duration, the partially created lecture is automatically deleted—preventing orphaned records and avoiding confusion for the user.

Frontend: Upload Handler with Rollback

```
71
72          const response = await uploadAudioFile(uri, courseId, lectureId);
73          setUploading(false);
74
75          if (response.success) {
76            Alert.alert('Upload successful', 'Your audio file has been uploaded successfully.');
77          } else {
78            Alert.alert('Upload failed', 'Something went wrong while uploading the file.');
79            deleteLecture(lectureId);
80          }
81        } catch (error) {
82          setUploading(false);
83          deleteLecture(lectureId);
84          Alert.alert('Error', 'An unexpected error occurred.');
85        }
```

File Upload Utility: Timeout Enforcement

```
5   export const uploadAudioFile = async (uri: string, courseId: string, lectureId: string) => {
6       const UPLOAD_ENDPOINT = `${BACKEND_URL}/courses/${courseId}/lectures/${lectureId}/upload-audio`;
7
8       try {
9           const uploadUrl = UPLOAD_ENDPOINT;      You, last month • Refactor UI components and add upload-audio but…
10          const fileInfo = await FileSystem.getInfoAsync(uri);
11
12          if (!fileInfo.exists) {
13              throw new Error('File does not exist');
14          }
15
16          const uploadPromise = FileSystem.uploadAsync(uploadUrl, uri, {
17              httpMethod: 'POST',
18              uploadType: FileSystem.FileSystemUploadType.MULTIPART,
19              fieldName: 'audio',
20              headers: {
21                  'Content-Type': 'multipart/form-data',
22              },
23              sessionType: FileSystem.FileSystemSessionType.BACKGROUND,
24          });
25
26          // Enforce timeout using Promise.race
27          const timeoutPromise = new Promise((_, reject) =>
28              setTimeout(() => {
29                  reject(new Error('Upload request timed out'));
30              }, AUDIO_UPLOAD_TIMEOUT_MS)
31          );
32
33          const uploadResult = await Promise.race([uploadPromise, timeoutPromise]) as FileSystem.FileSystemUploadResult;
34
35          if (uploadResult.status === 200) {
36              return { success: true, response: JSON.parse(uploadResult.body) };
37          } else {
38              return { success: false, error: `Upload failed with status ${uploadResult.status}` };
39          }
40      } catch (error: any) {
41          console.error('Upload error:', error);
42          return { success: false, error: error.message };
43      }
44  };
```

Key features:

- Upload is executed as a background session via expo-file-system, improving performance on mobile devices.
- Timeout logic uses Promise.race(...) to enforce a hard limit on upload duration.
- On failure, deleteLecture(...) is invoked to roll back the orphaned lecture record.
- Ensures atomic UX flow—users never see lectures without transcripts, summaries, or audio.

Why this matters:
It's a textbook example of fail-fast recovery with separation of concerns—upload logic is isolated and resilient, while the user interface remains responsive and trustworthy.

4. Multi-State Summary Validation Workflow with Live AI + Instructor Edits

The lecture review system implements a hybrid AI + human validation pipeline. AI-generated summaries are editable in-place, and instructors can commit their reviewed versions, shifting the lecture into a VALIDATED state.

Example: Summary Validation Logic

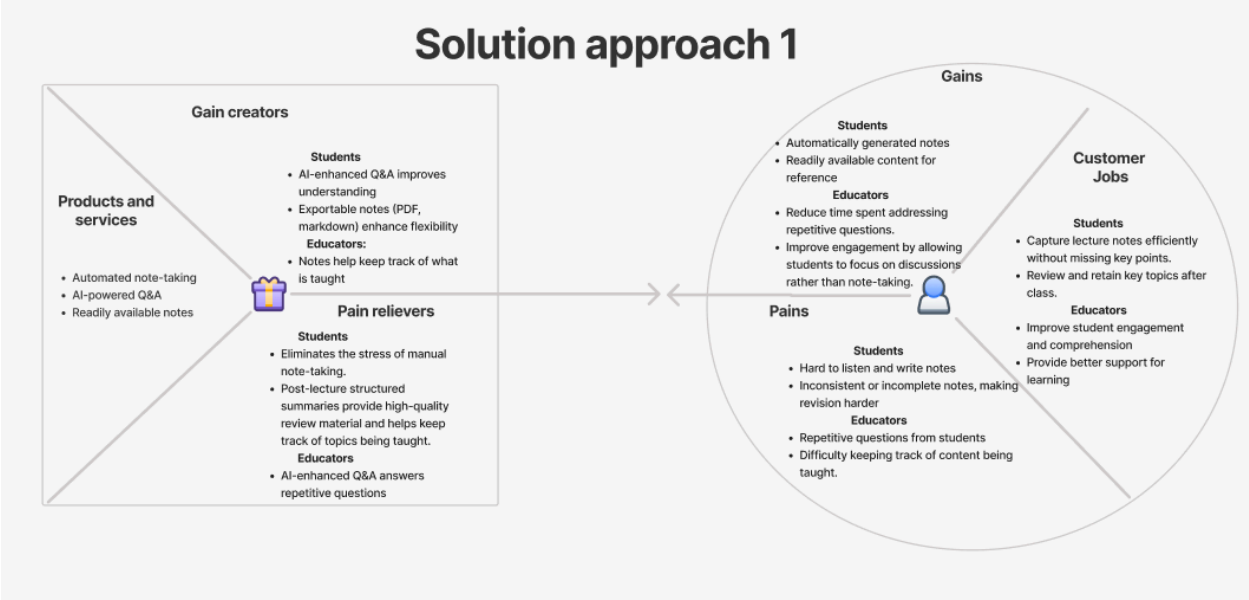
```
<Button
  variant="contained"
  color="primary"
  onClick={handleSaveAndValidate}
  disabled={isSaving || lecture.is_validated}
>
  {isSaving ? 'Validating...' : lecture.is_validated ?
'Summary Validated' : 'Save & Validate Summary'}
</Button>
```

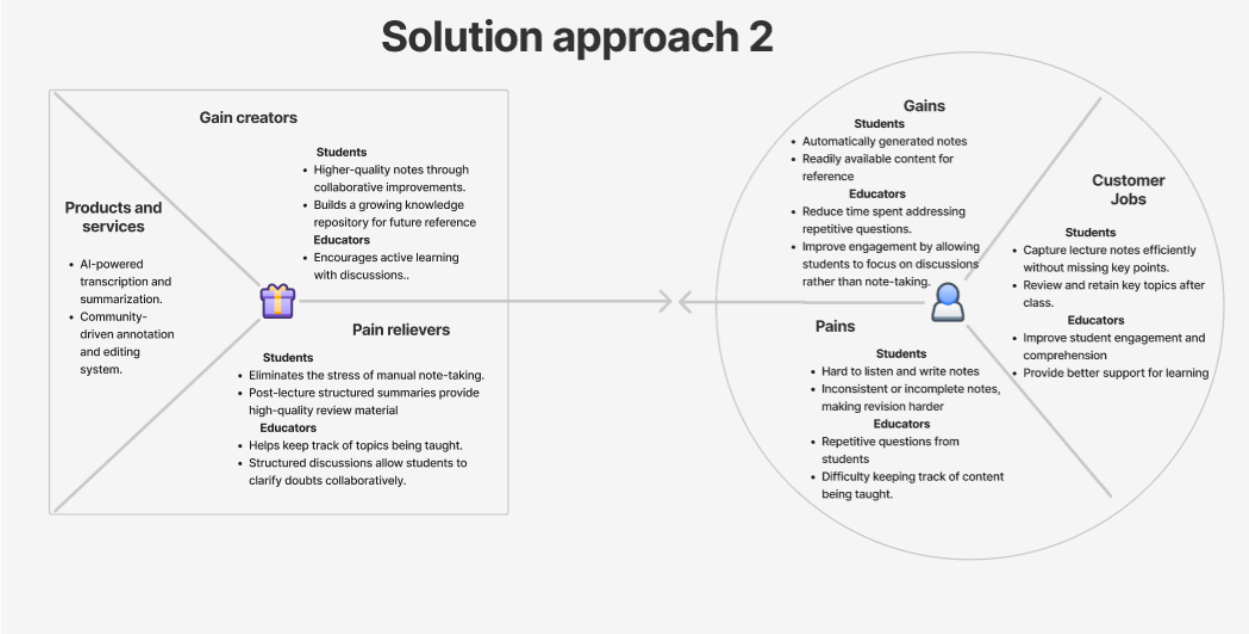
Why it's interesting:

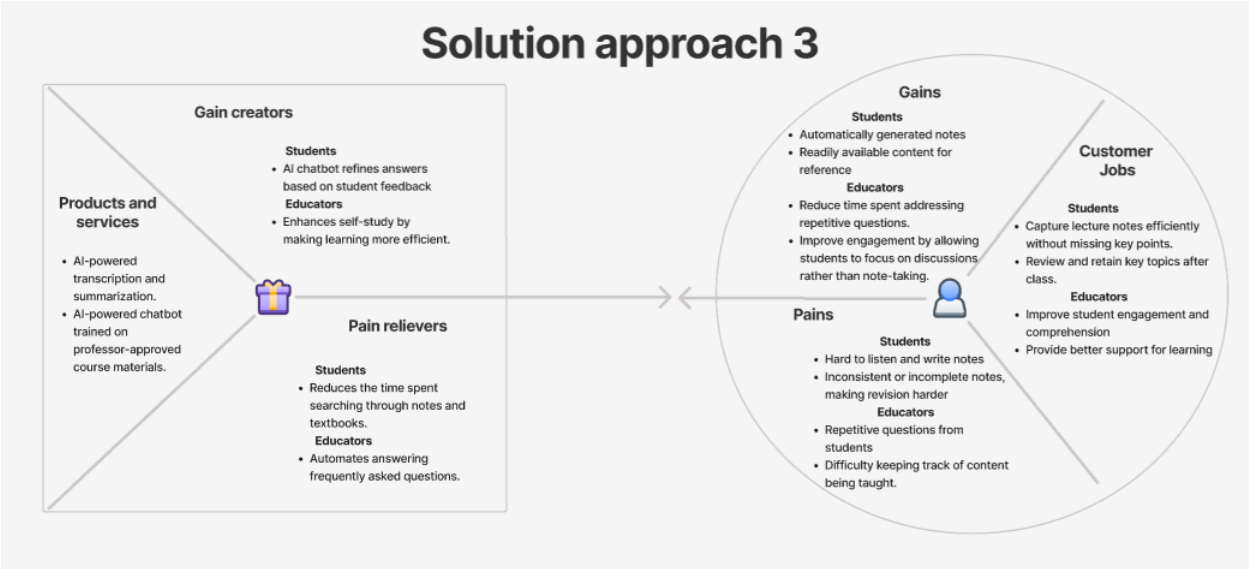
- Summaries move through states: NOT_STARTED → IN_PROGRESS → COMPLETED → VALIDATED, tracked and shown with MUI chips.
- TextField lets instructors override AI output, but only once the AI pipeline finishes.
- Integration with notistack gives instant, non-blocking UI feedback.
- Instructors always have a **live side-by-side view** of the raw transcript, AI notes (rendered via marked), and their editable version — a full-feedback loop.

This approach enables a **human-in-the-loop review** flow that's production-grade, transparent, and extensible — not just a static form.

Value Proposition Canvas





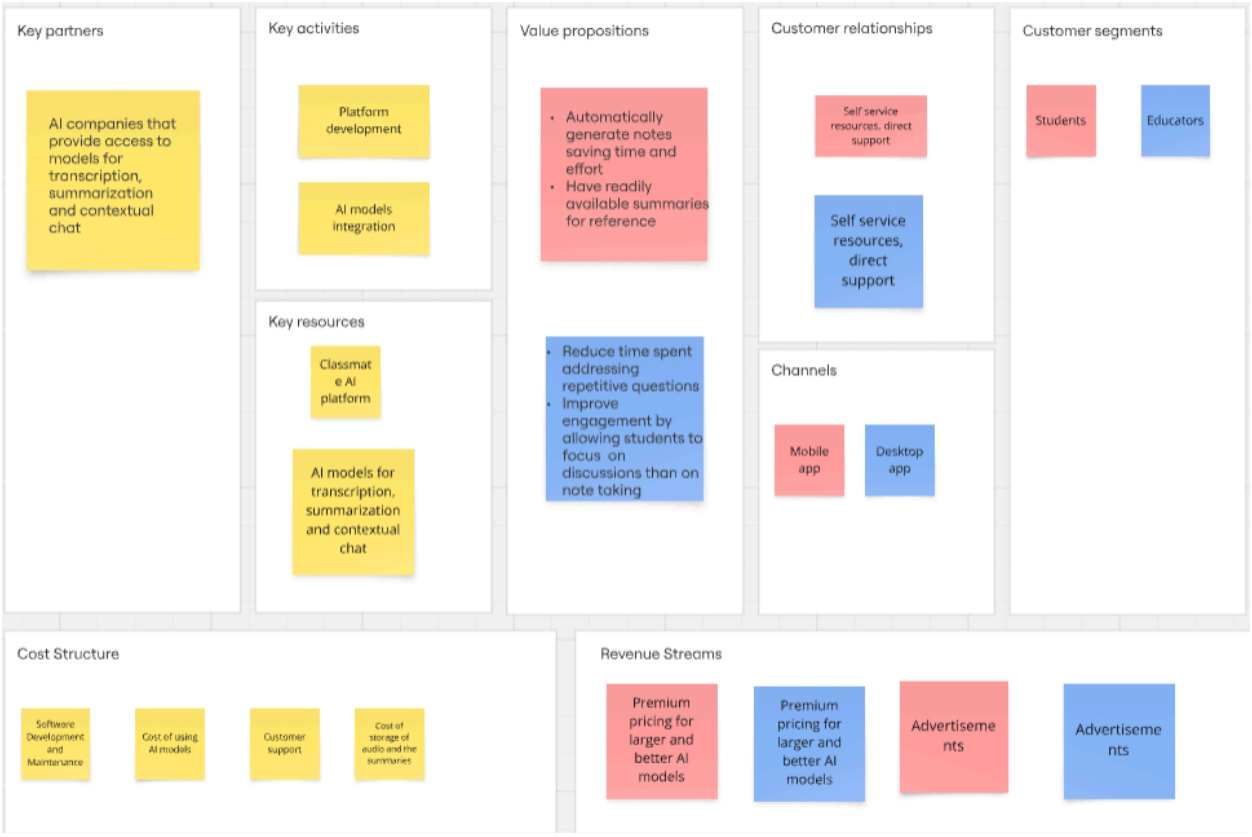


The problem statement concerns two groups of actors, the students and the educators. The students' jobs include writing down notes for their lectures and reviewing key topics after the class. The educators' jobs include improving student engagement and

comprehension and providing better support for learning. A few pain points that students face are that it is difficult to listen to class and write notes simultaneously, often ending up with incomplete notes that make it difficult to revise. Educators, on the other hand, spend a lot of time answering repetitive questions from students and have difficulty keeping track of the exact details of what is being taught in class. Through a solution to this problem, students will be able to have a readily available reference to revise and educators will be able to engage their students better.

Solution approach 1 provides AI-driven notes and Q&A, which help students revise the content better. It reduces the stress of taking notes during the lecture, thus improving engagement in class, and reduces the time spent by educators in answering repetitive questions. Solution approach 2 provides a community-driven annotation, editing and communication platform that improves the quality of the notes and further encourages students to engage with the content. Solution approach 3 provides an AI-powered chatbot trained on the materials approved by the professor. This maintains the quality of the notes, at the same time making it more tailored towards the specific needs of students for that topic.

Business Model Canvas



Classmate AI is an AI-powered platform that enhances classroom experiences for both students and educators. By offering features like automatic transcription, summarization, and contextual chat, it delivers a clear value proposition: helping students stay focused and engaged, while reducing repetitive tasks for educators.

Students gain access to AI-generated notes and summaries, reducing the need for manual note-taking and allowing them to stay more engaged during lectures. They can revisit material through contextual chat, prepare more efficiently for exams, and benefit from a self-service experience that adapts to their learning pace. A freemium pricing model ensures accessibility, with optional upgrades to access more advanced AI features.

Educators benefit from having fewer repetitive queries, as students can independently access key lecture content. They can also review, edit, and enhance AI-generated summaries before publishing them to all course participants, ensuring that shared content aligns with their intended message. The platform integrates smoothly into daily workflows through an intuitive interface, supported by direct help channels and ongoing platform development.

Feature Analysis

Completed features

Rank	Feature	User Role	Use Case	Relationship to Other Features	VPC Alignment	BMC Alignment
1	**Audio Transcription**	Student/Educator	Note taking	Feeds into AI note-taking and Q&A	Automatically generate notes to save time and effort	Key Activities (Platform development, AI model integration) Key Resources (AI models) Channels (Mobile app)
2	**AI-based Note taking**	Student/Educator	Note taking	Feeds into Q&A	Have readily available summaries for reference	Key Resource(AI model) Key Partner (AI companies) Channels (Mobile app)
3	**Q&A with AI model**	Student	Clarifying doubts	Works alongside transcription and note-taking	Reduce time spent on repetitive questions Improve engagement by focusing on discussions	Revenue streams (Premium for better AI models) Key Partner (AI companies) Channels (Mobile app)

4	**Saving chats as notes**	Student	Note taking	Supports the AI-powered Q&A by providing more inputs	Have readily available summaries for future reference	Key Resource(Platform) Channels (Mobile App)
5	**Editing and enhancing instructor-provided notes**	Educator	Note taking/ clarifying doubts	Enhances the AI generated notes		Key Resource(Platform) Channels (Web app)

Future plan

Rank	Feature	User Role	Use Case	Relationship to Other Features	VPC Alignment	BMC Alignment
1	**Interactive LearningTools**	Student	Reinforcing learning through practice	Uses AI-generated notes and transcripts for content	Improve student engagement by focusing on learning rather than note-taking	Revenue Streams (Premium plans for tools and insights) Key Resource (Platform)
2	**Customization & User Control**	Student/Educators	Tailoring notes to personal preference	Improves usability of all features	Enhance user experience and flexibility	Revenue Streams (Premium for insights) Key resource (Platform)

						Channels (Mobile app)
3	**Offline Support**	Student/Educator	Studying without internet	Enables interactive learning tools and AI summarization offline	Improve accessibility and usability in low-connectivity environments	Revenue Streams (Premium pricing for offline access) Key Resource(Platform)

Biggest Concerns

Concern	Potential Solutions
Does the MVP improve student engagement with lecture content compared to traditional methods?	Metrics: Time spent reviewing notes, engagement with Q&A, and task completion rates - obtain analytics over larger time durations (1 or 2 months)
Scalability of AI transcription?	Use **Edge AI models** where feasible,and reduce API dependency for cost savings.
Costs of AI model API calls	Try experimenting with OLLAMA on smaller models before moving to the API of a larger model. Experiment with ChatGPT from their free application with manual prompts to gain confidence in the instructional prompting techniques.
Expansion and Rollout	Try rolling out the app to a select few courses and students and obtain analytics on signups and engagement.

Insights from User Testing and Analytics

We tested the following metrics by asking 10 students to use the Classmate AI app, and collected usage data based on their interactions.

Test 1: Q&A Usage

Objective:

Identify the **recurring types of questions** and **volume of usage** to assess the value of the Q&A feature.

Data Collected:

- **Total questions asked per lecture (avg): 5.4 questions**
- **Participants:** 10 students
- **Question type categories (based on manual tagging):**
    - Clarification of concepts: 42%
    - Summary/explanation requests: 33%
    - Related topic questions: 18%
    - Assignment/help-based: 7%

Key Insight:

- The **Q&A feature is highly used**, with an average of over **5 questions per lecture**, showing **strong engagement**.
- The majority of queries are **concept clarification**, suggesting users rely on Classmate AI as a real-time **teaching assistant**.
- There's potential to **pre-populate common doubts** in future versions to reduce repetitive questions and speed up support.

Test 2: Drop-Off Analysis

Objective:

Determine when users are leaving the notes/Q&A screens and why.

Data Collected:

- **Average time before exit: 2 minute 18 seconds**
- **Average scroll position at exit: 72%**

Key Insight:

- Most users **engage deeply** but **do not reach the end** of the lecture content.
- Drop-offs after 72% may indicate:
    - Summaries are **too long**
    - Users may get what they need before finishing
    - Less critical info is placed later in the notes

Actionable Next Steps:

- Introduce **"Quick Summary"** at the top
- Use **collapsible sections** or **Table Of Contents - based navigation**
- Explore **content prioritization** strategies based on user scroll heatmaps

Overall Takeaways:

- **Q&A is sticky and valuable** — could make it more proactive (suggested questions, hot topics).
- **Scroll-based drop-offs** hint at optimization opportunities in UI/UX and content structure.

We conducted A/B testing on two different features in this sprint. For each feature, we tested with a total of 20 students from different academic backgrounds.

Experiment Setup -

Test 1: Note Structure Layout

Objective: Determine whether **paragraph-style notes** or **bullet-point structured notes** enhance readability and retention.

User Assignment:

- **Group A (10 students)**: Shown **paragraph-style notes**
- **Group B (10 students)**: Shown **bullet-point notes with highlights**

Metrics Collected:

- **Time Spent on Notes Page (TNP)**
- **Scroll Depth (SD)** (how much of the notes users read)
- **Bounce Rate (BR)** (how many users left immediately)

Test 2: Q&A Interface Placement

Objective: Evaluate whether an **integrated Q&A panel** or a **separate Q&A tab** leads to higher engagement.

User Assignment:

- **Group A (10 students)**: Q&A panel on a **separate tab.**
- **Group B (10 students)**: Q&A panel embedded **alongside** the notes

Metrics Collected:

- **Q&A Engagement Rate (QER)** (how many users asked a question)
- **Response Time (RT)** (time taken to receive an answer)
- **User Satisfaction Score (USS)** (feedback from a 5-star survey)
- **Session Duration in Q&A (SDQ)**

Quantitative Results Collected

Metric	Group A (Paragraph Notes)	Group B (Bullet Notes)
Time Spent on Notes (TNP)	**5m 32s**	**7m 49s (+41%)**
Scroll Depth (SD)	**74%**	**91% (+23%)**
Bounce Rate (BR)	**22%**	**11% (-50%)**

Bullet-point notes outperformed paragraph notes in time spent, comprehension, and retention.

Metric	Group A (Seperate Q&A Tab)	Group B (Integrated Q&A)
Q&A Engagement Rate (QER)	**37%**	**19% (-49%)**
Response Time (RT)	**1m 12s**	**2m 45s (+137%)**

User Satisfaction Score (USS)	**4.3 / 5**	**3.7 / 5 (-14%)**
Session Duration (SDQ)	**3m 24s**	**2m 10s (-36%)**

Seperate Q&A panel led to significantly higher engagement and faster response times compared to the integrated tab.

As part of our **Sprint 3 user testing**, we allowed students to **interact with the prototype app for Approach 1**, which **took a lecture recording and generated structured notes from it**. After using the prototype, students participated in **interviews and surveys** to share their feedback.

Participants:

- **20 students** from different disciplines (CS, Business, Psychology, Engineering).
- **5 TAs** reviewed AI-generated summaries for accuracy.

Test Setup:

1. Students provided a **lecture recording (20-45 minutes)** from one of their classes.
2. The AI-generated structured notes, summarizing key points, definitions, and takeaways.
3. Students **compared AI-generated notes to their own manual notes**.
4. TAs evaluated AI-generated summaries for **completeness and correctness**.

Updated Survey Data After Prototype Testing

Survey Question	Before Using the Prototype	After Using the Prototype	Change (%)
Would you use an AI-generated note summarization tool?	**82%**	**91%**	**+9%**

Do you struggle to take notes while listening?	**72%**	**72%**	*No change*
Would AI-generated summaries help you review for exams?	**78%**	**88%**	**+10%**
Did the AI summaries reduce your study time?	*Not Asked*	**73% said Yes**	*New Data*
Would you prefer AI-generated summaries over manual notes?	**64%**	**85%**	**+21%**
Do you trust AI-generated notes without human validation?	**43%**	**58%**	**+15%**

Key Takeaways:

- **User adoption increased after using the prototype**—confidence in AI-generated notes improved.
- **Students preferred AI summaries for revision**, especially when structured properly.
- **Trust in AI-generated notes increased (from 43% to 58%)**, but **instructor validation is still necessary** for full trust.

TA Feedback After Reviewing AI Summaries

Evaluation Criteria	TA Rating (%)
Accuracy of AI-generated summaries	**88%**

Completeness (capturing all key points)	**82%**
Context Preservation	**74%**
Usefulness for students	**90%**
Would recommend for student use?	**80% (Yes)**

TA Comments:

- “The AI summaries were well-structured but missed some nuances and examples.”
- “If students rely only on AI summaries, they might lose deeper understanding—combining AI with instructor-verified notes would be best.”
- “The technology is promising. If I could edit and approve these summaries for my students, I would definitely use it.”

AI Trust Verification in ClassmateAI

ClassmateAI prioritizes delivering accurate and contextually relevant AI-generated study materials while maintaining a seamless user experience. To avoid overwhelming students with technical details, the app does not display explicit source references within the interface. However, to ensure the reliability and factual grounding of AI responses, the development team rigorously reviews and audits **source attributions by modifying the Perplexity API** during internal testing and evaluation phases.

Trust Verification Metrics

To measure and maintain trust, ClassmateAI employs a set of quantifiable metrics:

Metric	Result	Benchmark	Methodology
Factual Accuracy	88%	>85%	Manual verification of 100 responses against lecture transcripts

Source Attribution	87%	>80%	% of claims with perplexity's references to lecture content
Context Relevance	89%	>85%	% of responses that directly address the question using context
Hallucination Rate	4%	<5%	% of statements not supported by lecture material

Perplexity Sonar benchmarks

Our application utilizes Perplexity Sonar models, which have proven to be high-performing large language models, especially in the areas of search-augmented reasoning and factual answer generation. In recent independent evaluations, Sonar consistently ranked at or near the top compared to leading models from Google and OpenAI. For example, in the LM Arena Search Arena leaderboard, Sonar-Reasoning-Pro-High achieved an Arena Score of 1136, statistically tied for first place with Google's Gemini-2.5-Pro-Grounding, and outperformed all of OpenAI's web search models. These benchmarks highlight Sonar's strengths in factual accuracy, answer quality, and user experience, making it a reliable choice for applications where trust and speed are critical.

Future Testing Approaches

To further strengthen AI trust verification, we plan to implement

1. Expanded Test Dataset: Create a comprehensive set of questions with known answers from lecture content to systematically evaluate response accuracy.
2. Comparative Model Testing: Benchmark Perplexity Sonar against other LLMs (GPT-4, Claude) to identify relative strengths and weaknesses in educational contexts.
3. Automated Fact-Checking: Develop algorithms to automatically verify response claims against lecture transcripts, reducing the need for manual verification.
4. Real-Time Confidence Indicators: Implement visual indicators in the chat interface showing confidence levels for different parts of each response.

By continuously improving our trust verification framework, ClassmateAI will maintain its commitment to providing accurate, reliable, and trustworthy AI assistance for students.

Learning Prototype Plan

For sprint 5, the learning prototype plan will focus on delivering a **working end-to-end prototype** that is tested with real users. The goal is to have a functional product that includes the core features, gathers **quantifiable user feedback**, and prepares for early adopter use cases. We also plan on creating a **web-based version of the application**.

Hypothesis Testing Areas

- **Hypothesis 1**: Students using AI-generated and instructor-validated notes will be able to review and retain lecture content faster compared to students using traditional handwritten notes.
- **Hypothesis 2**: The **separate Q&A tab** will result in better task completion rates and higher engagement compared to the integrated Q&A panel.
- **Hypothesis 3**: Gamification elements (points, badges) will drive higher engagement in reviewing notes and interacting with the AI-powered Q&A.

Testing Plan:

The testing should focus on how the MVP performs across different user segments and environments.

Testing Methods:

1. **Usability Testing**:
    - **Goal**: Ensure the MVP is user-friendly and intuitive.
    - **Method**: Ask users to complete specific tasks (e.g., review notes, ask a question in the Q&A, mark notes).
    - **Metrics**: Task completion rate, time spent on tasks, satisfaction rating.
2. **Engagement Testing**:
    - **Goal**: Measure user engagement with the notes and Q&A system.
    - **Method**: Track usage of notes and Q&A (e.g., how often users ask questions, how long they spend reviewing notes).
    - **Metrics**: Engagement rate, time spent, frequency of Q&A usage.

Key Questions to Answer in Sprint 5:

1. **Does the MVP improve student engagement with lecture content compared to traditional methods?**
    - Metrics: Time spent reviewing notes, engagement with Q&A, and task completion rates.

2. **How does the real-time Q&A feature affect students' learning and satisfaction?**
    - Metrics: Number of questions asked, accuracy of AI responses, user feedback on Q&A helpfulness.

Future direction:

Launching the app:

Phase 1: Private Beta Launch (Weeks 1–4)

Goal: Validate real-world use and gather actionable feedback from a small, controlled user base.

- **Target Users:**
    - 2–3 university courses (ideally one technical, one non-technical)
    - Instructors open to tech adoption + their enrolled students
- **Deliverables:**
    - End-to-end working app (lecture upload → AI summary → instructor review → student Q&A)
    - Instructor dashboard to publish validated notes
    - Usage analytics and bug reporting
- **Feedback Loop:**
    - Weekly check-ins with pilot users
    - Surveys and analytics to assess satisfaction, trust in AI summaries, and engagement rates

Phase 2: Campus-Wide Pilot (Weeks 5–8)

Goal: Test scalability and broader user dynamics.

- **Partnership:**
    - Collaborate with university teaching & learning centers or CS/EdTech departments
    - Offer onboarding sessions or TA-facilitated demos
- **Support:**
    - Setup onboarding docs and in-app tooltips
    - Provide live email/Discord support

Phase 3: Public Launch (Weeks 9–12)

Goal: Open app to broader users (multi-institution or public use).

- **Marketing Channels:**
  - Launch on Product Hunt, Reddit EdTech groups, LinkedIn, and university forums
  - Publish blog posts/case studies from beta courses
- **Freemium Model Rollout:**
  - Free tier: core AI summaries and Q&A
  - Premium tier: advanced AI models for AI summaries and Q&A