NoteScribe - Notes Taking Application

Joseph Jang - jyj3
Kiruthika Janakiraman - kj25
Sai Shreya Kumar - ssk12
Sudarshni Ramesh - sr89

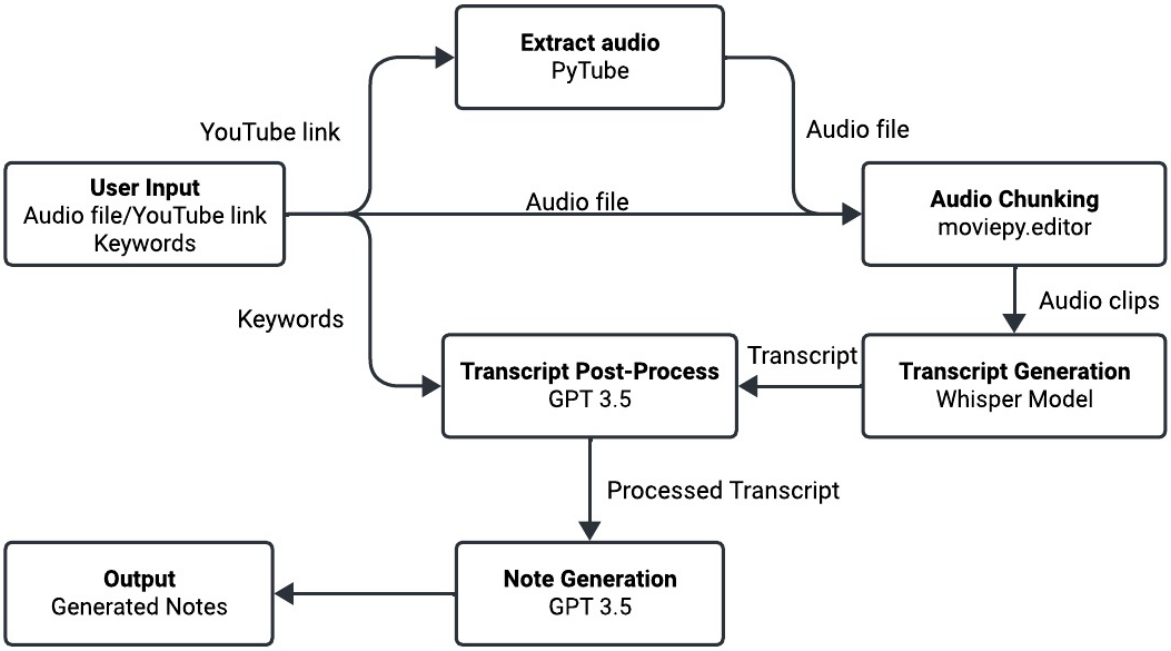

Figure 1: Architecture

ACM Reference Format:
Joseph Jang - jyj3, Kiruthika Janakiraman - kj25, Sai Shreya Kumar - ssk12, and Sudarshni Ramesh - sr89. 2018. NoteScribe - Notes Taking Application. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 6 pages. https://doi.org/XXXXXXX.XXXXXXX

1 GITHUB LINK

https://github.com/Spoon7227/CS510-Project

Unpublished working draft. Not for distribution.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference acronym 'XX, June 03–05, 2018, Woodstock, NY
© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

2 MOTIVATION

The idea for creating this stand-alone artificial intelligence notes-taking web application came from realizing how time-consuming and inefficient it is to manually extract important points from audio and video content. This is a problem that is common to traditional note-taking processes. The difficulty of condensing vast volumes of material from lectures, seminars, and internet resources into brief and insightful notes is especially evident in professional and educational contexts. The application intends to drastically speed up this process by utilizing cutting-edge AI technologies, like OpenAI's API and the Whisper model for audio-to-text conversion. By automating note-taking tasks and including features like post-processing, keyword assistance, and context-awareness, the application solves the drawbacks of time-consuming human transcribing and comprehension, giving users with a more efficient and effective solution. Furthermore, the application's ability to accommodate uncommon words and acronyms, as well as its support for retrieving audio transcripts from popular media

streaming platforms such as YouTube, increase its utility and relevance in meeting the needs of users looking to improve their note-taking workflows.

3 INTENDED USERS

The intended users of this AI notes-taking web application are diverse, spanning both educational and professional fields. Students at all stages of education, from high school to university, stand to gain greatly from the application's capabilities. For students, the program is an invaluable study tool, allowing them to create concise and detailed notes from lecture videos, online tutorials, and instructional content available on sites such as YouTube. By automating the note-taking process and providing features like keyword assistance and context awareness, the application enables students to easily record and retain critical information, improving their learning outcomes and academic achievement. In the professional realm, the application is designed for professionals who attend meetings, conferences, or training sessions that require note-taking. Business professionals, researchers, and knowledge workers can use the program to extract insights, action items, and key takeaways from audio and video content, allowing for better decision-making, collaboration, and knowledge management within organizations. Overall, the application's user-centric design, adaptability, and scalability make it an invaluable tool for a diverse spectrum of users looking to optimize their note-taking processes and increase productivity in both educational and professional settings.

4 MAJOR FUNCTIONS

In our project "NoteScribe", we develop a new standalone AI-driven web application to automate the process of note-taking from Audio inputs and Text transcripts. This application also leverages the modern web technologies and Machine Learning Models to provide a flexible, user-centered Tool useful for note-taking. Which supports a variety of user needs starting from academic learning to professional meeting documentation. The Major Functionalities of our project "NoteScribe" are as follows

3.1 Audio and Text Transcript Processing: Functionality: "NoteScribe" is created to handle audio inputs from multiple sources, including direct uploads from users and also from streaming platforms like YouTube. It supports a wide range of audio formats, which are automatically processed through advanced speech-to-text engines. The application utilizes the Whisper API for accurate and efficient transcription. This API is known for its robust performance across different languages and dialects, ensuring broad usability.

3.2 Dynamic Note Generation: The system generates customized notes based on user-selected parameters. It can produce various outputs such as detailed summaries, lists of action items, and records of key decisions. Making use of OpenAI's GPT models, "NoteScribe" transforms raw transcriptions into structured formats. The model is fine-tuned with prompts that guide the generation of specific types of notes, allowing for flexibility and relevance to the user's needs.

3.3 Keyword Assisted Transcript Correction: The accuracy of transcripts generated by speech-to-text models in recognizing domain-specific entities underrepresented in training data is improved by performing post-processing with user-provided keywords. Correcting key phrases and terms relevant to the context helps highlight significant concepts that could be missed during downstream note generation.

3.3 User-Centered Customization Users can customize their note-taking experience by setting preferences for the format, level of detail, and handling of specialized keywords. The frontend interface allows users to specify these preferences, which are then stored in a user profile managed by the backend. These preferences are then applied dynamically to the note generation process.

5 IMPLEMENTATION

Figure 1 shows the architecture of our system. "NoteScribe" begins its process by offering a user-friendly frontend, developed using React, which allows users to manage inputs through interactive form controls. Users can upload audio files or enter URLs from platforms like YouTube, which are then processed. The system's state management capabilities ensure that user preferences and sessions are dynamically handled, also enhancing the personalization of the experience.

Once the audio is uploaded or fetched from a URL, the backend (Node.js), takes over. It manages essential tasks such as session management and file storage, and begins the workflow of audio processing. The backend also handles the integration with third-party APIs, ensuring that data flows smoothly from the frontend to the processing modules.

In the audio processing stage shown in Algorithm 1, it undergoes naive chunking where it's divided into equal-length intervals, facilitating straightforward and efficient processing.

Audio is then fed into OpenAI's Whisper model to generate transcript as shown in Algorithm 2. Since speech-to-text models often struggle with domain-specific entities that are underrepresented in training data, we also perform post-processing on the raw transcript using user-provided keywords. We pass the transcript and keywords as a prompt to OpenAI's GPT-3.5-turbo model to correct transcription errors as shown in Algorithm 3.

The keyword corrected transcript is then fed into the GPT model as shown in Algorithm 4. Here, based on the user's predefined preferences and the type of notes required like summaries or concise action items, the GPT model then tailors the output. It uses various prompt strategies to generate notes that are not only relevant but also formatted according to the user's needs.

The integration with external APIs, like the YouTube API for sourcing audio directly from video content, and the efficient transcription and text generation is facilitated by the Whisper and GPT APIs, respectively, ensures that the entire process is seamless. These APIs are crucial for real-time processing.

Unpublished working draft. Not for distribution.

Algorithm 1 Process Audio File

Require: *audio_clip*: Audio file, *notes_type*: Type of notes, *keywords*: Keywords for processing, *chunk_length*: Length of audio chunks (default=60)
Ensure: Processed transcript and notes

```
chunks_transcript ← []
total_duration ← audio_clip.duration
num_chunks ← ⌊total_duration/chunk_length⌋ + 1
for i = 0 to num_chunks − 1 do
    start_time ← i × chunk_length
    end_time ← min((i + 1) × chunk_length, total_duration)
    chunk_audio_clip ← audio_clip.subclip(start_time, end_time)
    chunk_transcript ← transcribe_audio_with_openai(chunk_audio_clip, keywords)
    Append chunk_transcript to chunks_transcript
end for
Concatenate all chunks_transcript into a single transcript

if len(keywords) > 0 then
    Post-process transcript with keywords
end if
Generate notes based on notes_type
return transcript and notes
```

Algorithm 2 Transcribe Audio with OpenAI

Require: *audio_file_path*: Path to the audio file, *keywords*: Keywords for transcription
Ensure: Transcribed text

```
Initialize the OpenAI client
Load audio file
with open(audio_file_path, "rb") as audio_file:
    Transcribe
    transcription ← client.audio.transcriptions.create(model="whisper-1", file=audio_file, prompt=keywords)
return The transcribed text
```

Algorithm 3 Correct Keywords

Require: *user_input*: User input text, *keywords*: List of keywords
Ensure: Text with corrected keywords

```
Initialize the OpenAI client
prompt ← "Please ensure that all occurrences of the following keywords are spelled exactly as given, maintaining exact case sensitivity: " + keywords + ". For example: : 'I like apple and water guns.': [APPLE, WaterGuns]Output: 'I like APPLE and WaterGuns.'"
Define the prompt with the system message and user query
prompt ← ["role": "system", "content": prompt, "role": "user", "content": user_input]
Generate completion
completion ← client.chat.completions.create(model="gpt-3.5-turbo", messages=prompt)
return The generated notes
```

Algorithm 4 Generate Notes

Require: *user_input*: User input text, *prompt*: Prompt for generating notes
Ensure: Generated notes

```
Initialize the OpenAI client
Define the prompt with the system message and user query
prompt ← ["role": "system", "content": prompt, "role": "user", "content": user_input]
Generate completion
completion ← client.chat.completions.create(model="gpt-3.5-turbo", messages=prompt)
return The generated notes
```

Unpublished working draft. Not for distribution.

6 STEPS TO USE NOTESCRIBE

(1) **Installation:**

- Clone the repository using:
  git clone https://github.com/Spoon7227/CS510-Project.git
- Navigate to the backend directory:
  cd backend
- Install backend dependencies: **npm install**
  pip install -r requirements.txt
- Navigate to the frontend directory:
  cd ../frontend
- Install frontend dependencies:
  npm install
- Install `react-scripts`:
  npm install react-scripts

(2) **Configuration:** Configure the OpenAI API key in the `config.js` file. This key is required for authenticating with the OpenAI API and accessing language models used for transcription and note generation.

(3) **Usage:**

- Start the backend server:
  node server.js
- Start the frontend server:
  npm start

- Access the application using **http://localhost:3000**
- Choose prompt choice: Meeting notes or lecture notes
- Choose input type: Enter link or upload audio file.
- Figure 2 and Figure 3 show the input and output for a meeting video.
- Figure 4 and Figure 5 show the input and output for a lecture video.
- You can pass keywords to the input as shown in Figure 6.
- Figure 7 shows the transcript without using keywords. It is seen that the word "Truity" is misspelled as "Trudy". Figure 8 shows the transcript when keywords are given and "Truity" is spelled correctly.

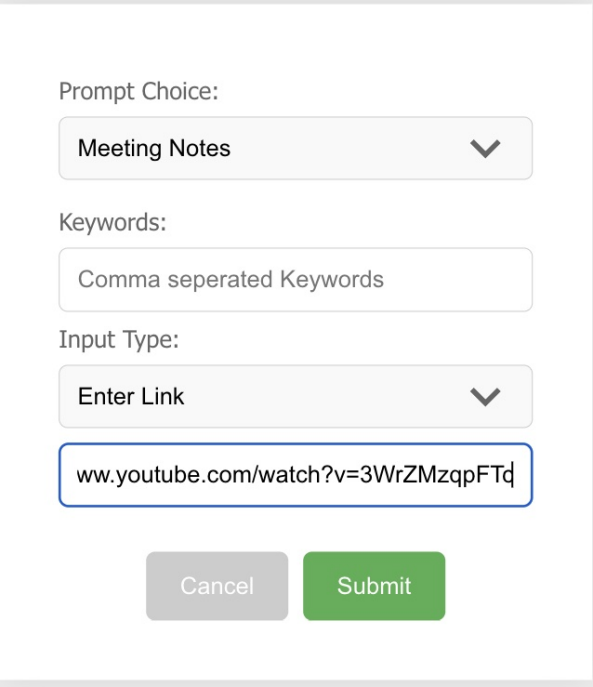

Figure 2: Input for meeting notes (YouTube link)

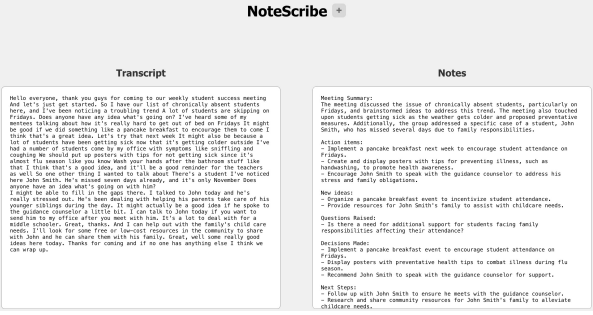

Figure 3: Output for meeting notes

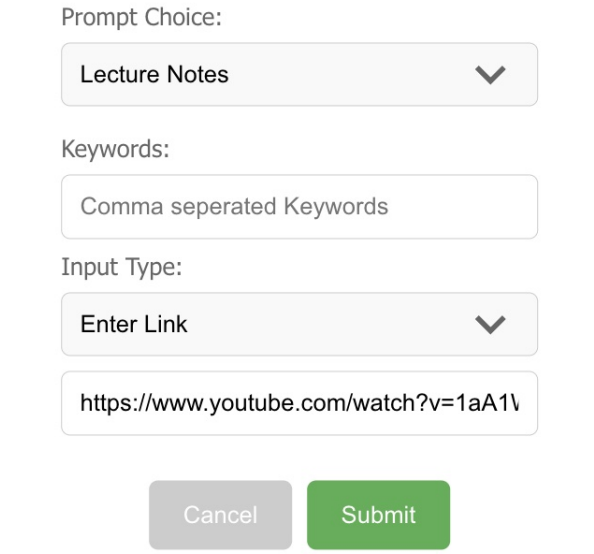

Figure 4: Input for lecture notes (YouTube link)

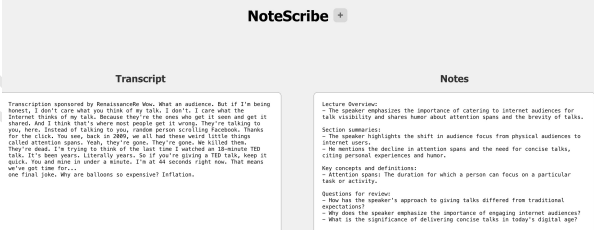

Figure 5: Output Notes for lecture notes

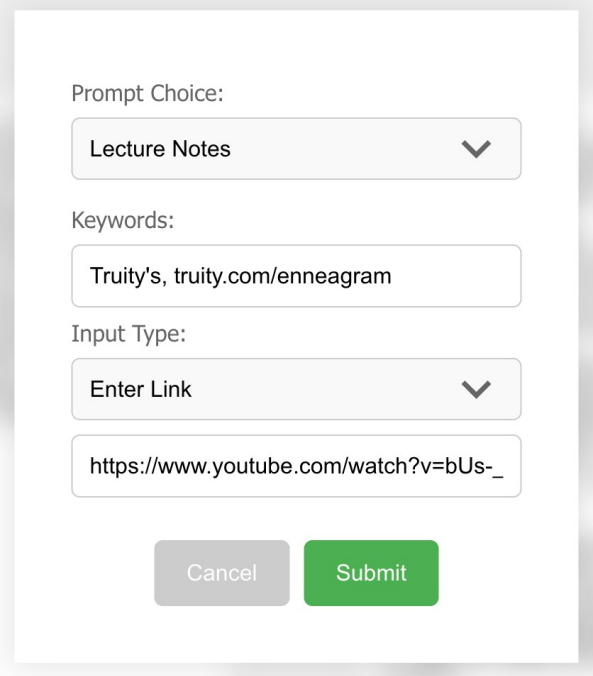

Figure 6: Input with keywords

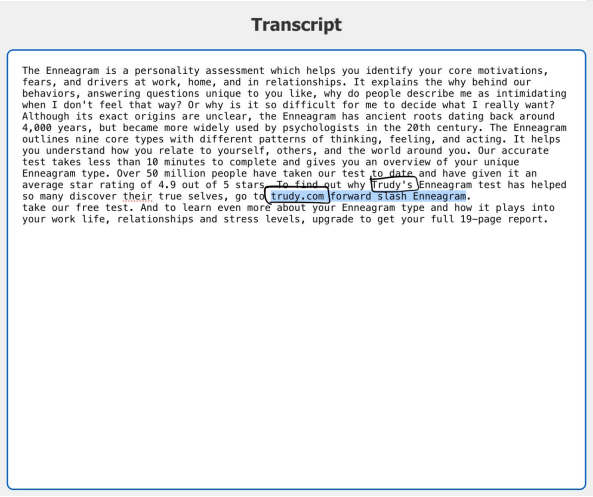

Figure 7: Output without keywords

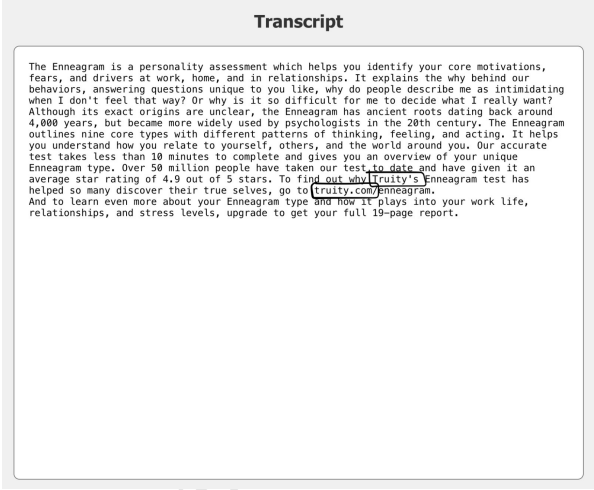

Figure 8: Output with keywords

Unpublished working draft. Not for distribution.

7 INDIVIDUAL CONTRIBUTIONS

(1) Joseph Jang - Planned the project's technical design, developed the front-end landing page and output display, managed API requests and file upload middle-ware, and implemented transcript post-processing with keyword enhancement.
(2) Kiruthika Janakiraman - Developed the backend script for generating meeting/lecture notes from the generated transcript using OpenAI's Chat Completion API and integrated the backend and frontend.
(3) Sai Shreya Kumar - Designed the input system for direct audio uploads and extracting audio from YouTube Links, implemented audio chunking to efficiently process long recordings, and integrated Open AI's Whisper model for accurate transcript generation.
(4) Sudarshni Ramesh - Implemented several key features including choice input for links and audio files, a panel with three fields, a prompt choice dropdown menu, and the ability to input keywords as a list of strings. Also developed the functionality for uploading files or links and displaying the corresponding upload field based on user selection and integrated with the main p.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009

Unpublished working draft. Not for distribution.