Project Scoping Submission
VidSynth: YouTube Comment & Video Analysis Tool

Team Members:
- Ananya Mahesh Shetty
- Ganapriya Hiriyur Shivakumar
- Nicholai Platonoff
- Nidhi Surve
- Shreyas Somnath Jondhale
- Sree Ramya Pasala

1. Introduction
VidSynth is a tool, implemented as a browser extension, that aims to enhance the YouTube user experience by providing concise summaries of video content and comment sections. By leveraging a Large Language Model (LLM), the tool will analyze video transcripts and top comments to generate summaries, allowing users to quickly grasp the key points and overall sentiment of the discussion without needing to watch the entire video or read through hundreds of comments. Additionally, there are two unique features planned to improve user experience within the comment section. The first will allow filtering of the comment section by various sentiments. The second will attempt to extract novel insights provided by a commenter. The project will prioritize using a self-hosted open-source LLM for potential cost savings.

2. Dataset Information
- **Dataset Introduction:** The "dataset" for this project is not static; it consists of dynamic, real-time data retrieved directly from YouTube via its API. This includes video transcripts and user-generated comments for any given YouTube video. The purpose of this data is to serve as the direct input for our analysis and summarization pipeline, making its relevance paramount to the project's function.
- **Data Card:**
  - **Data Content:** Publicly available YouTube video transcripts and comments.
  - **Size:** Variable; depends entirely on the video's length and the number of comments it has.
  - **Format:** The data will be retrieved in JSON format from the YouTube API.
  - **Data Types:** Primarily strings (comment text, transcript text, user names), integers (like counts), and timestamps.

Examples Below:

Video Transcript API return format. Python module: "Youtube-transcript-api"

Video ID Type: string	Transcript Language Type: String	Transcript Type: List of Dictionaries

"12345"	"English"	<pre>[
		    {
		        'text': 'Hey there',
		        'start': 0.0,
		        'duration': 1.54
		    },
		    {
		        'text': 'how are you',
		        'start': 1.54
		        'duration': 4.16
		    },
		    # ...
		]</pre>

Official YouTube Data API to obtain comments (JSON format)
Only 3 out of roughly 25 fields displayed (other fields like publish time would be disregarded)

Video ID Type: string	Comment ID Type: string	Comment Text Type: string
"dQw4w9WgXcQ"	"UCuAXFkgsw1L7xaCfnd5JJOw"	"This is a comment"

- **Data Sources:** The sole source of data will be the official **YouTube Data API**. The python module, **youtube-transcript-api** , is built on top of the official YouTube API, and allows for simpler API requests. All data will be retrieved through authenticated API calls to the appropriate endpoints for transcript and comment data.
- **Data Rights and Privacy:** This project will operate exclusively on publicly available data. We will adhere strictly to the YouTube API Services Terms of Service. No personally identifiable information (PII) beyond what is publicly visible on YouTube (e.g., usernames) will be stored. The tool will process data on-the-fly and will not maintain a separate database of user comments, ensuring compliance with data protection regulations like GDPR.
    - **Note**, scope has been updated: YouTube blocks transcript requests from Data Center IPs to protect from large scale scraping of YouTube videos. As a solution, we will be required to use an IP migration tool that provides non-blocked IPs. We

will stay within the YouTube terms of service since this tool is not storing transcripts long term, nor will they be used for training any AI model on YouTube videos.

3. Data Planning and Splits

Our data plan focuses on a real-time processing pipeline rather than traditional model training splits.

1. **Loading:** The process begins when a user activates the tool on a YouTube page. An API call is made to the YouTube API to fetch the top comments and the video transcript.
2. **Preprocessing:** Before being sent to the LLM, the raw data undergoes crucial cleaning. This includes stripping any HTML tags, removing extraneous characters, and filtering to use only the relevant text from the comments and transcript. Tokenization or adding special prompts will be required for our own hosted LLM.
3. **Managing Data:** The cleaned text data is structured into a suitable format (e.g., a single text block or a structured JSON object) to be sent as a prompt to our LLM API.

4. GitHub Repository

- **Main GitHub Link:** https://github.com/ganapriyahs/VidSynth-YouTube-Comment-Video-Analysis-Tool
- **Extension GitHub Link:** https://github.com/ganapriyahs/VidSynth-Extension-YouTube-Comment-Video-Analysis-Tool.git
- **Folder Structure (Main GitHub):**
  - /VidSynth: Contains the source code for the Airflow DAG code, each Cloud Run (which is each DAG step), and Gateway service that mediates between front and backend
  - /Model_Dev_Pipeline: Contains code for validating the language model we use.
  - /BrainStorming_Docs: Includes project documentation, including setup and usage guides.
  - /Deployment_Scripts: For any automated utility scripts.
  - .github/workflows: CI/CD scripts
- **README:** The repository will include a comprehensive README.md file detailing the project's purpose, installation instructions, and clear usage guidelines.

5. Project Scope

This project will be broken down as follows:

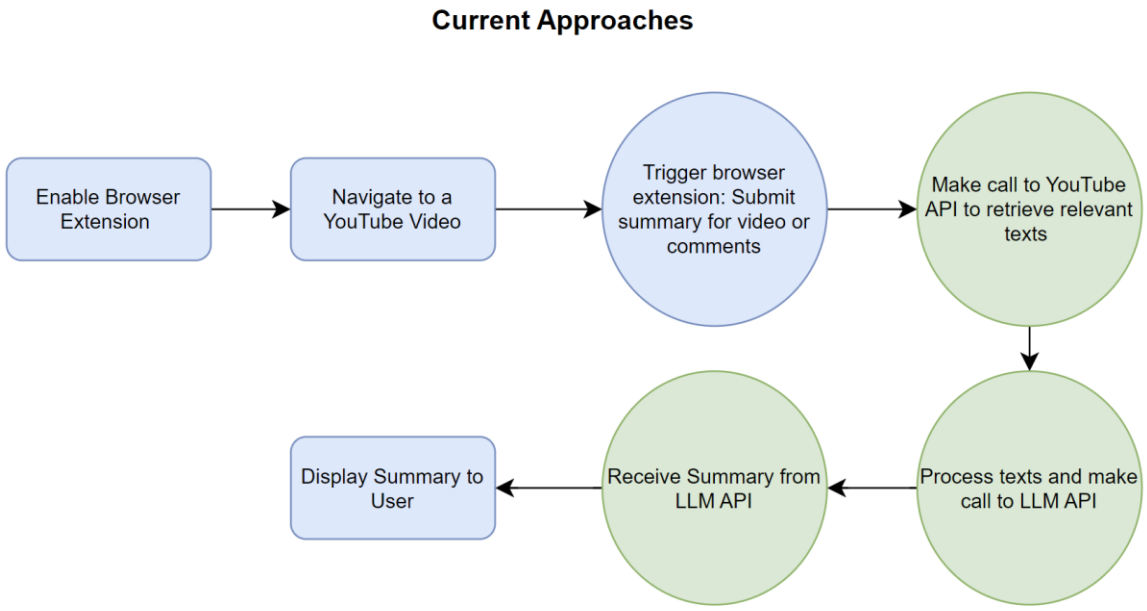
- **Problems:**
  - **Information Overload:** Reading through hundreds or thousands of comments to gauge public opinion is time-consuming.
  - **Time Commitment:** Users may not have time to watch a full video to determine if it's relevant to them. This especially applied for long-form content.
  - **Difficulty for Creators:** Content creators struggle to sift through repetitive or non-constructive comments to find valuable feedback.
- **Current Solutions:**

    - Using third-party summary services (extensions), which often rely on closed-source models like GPT.
    - Manually skimming top comments.
    - Relying on the video's like/dislike ratio and view count.
- **Proposed Solutions:**
    - **More Convenient Summarization:** Existing tools require navigation to the video itself. An interface would be added to the home page that submits a summarization request. Similar to video preview, generated summaries could help users find the best video for them.
    - **Intelligent Comment Filtering:** Unique features to filter comments, such as isolating **constructive criticism** or filtering out low-effort comments to highlight **unique, serious discussion points**.
    - **Open-Source Implementation:** A key innovation is the use of a self-hosted, open-source LLM on a cloud platform (e.g., GCP). This has potential of making the service cheaper

6. Current Approach Flow Chart and Bottleneck Detection

.

Flowchart:



The blue rectangles represent user actions or visibility, while the green represents back-end actions taken by the competitor browser extensions.

Potential Bottlenecks:

1. **API Rate Limits:** Both the YouTube API and our own LLM API will have request limits.
2. **LLM Inference Time:** The time it takes for the LLM to generate a summary could be slow.
3. **YouTube API latency:** There is an unavoidable bottleneck that waits for YouTube comments or video transcripts to be received.
4. **Preprocessing Efficiency:** Inefficient data cleaning could add to the overall delay.

Improvements:

1. Instead of making an API call to OpenAI or Anthropic for LLM summaries, we would host an open-source model ourselves on GCP. This has the potential to be cheaper.
2. To improve user experience, we would eliminate the requirement to navigate to a specific YouTube video. Instead, the summary request could be triggered from a user's homepage and also displayed there.
3. Novel ideas mentioned above like comment filtering could provide competitive advantage over similar products.
4. We could implement a prefetching system where a user selects a YouTube channel's new uploads or particular videos for background processing.

7. Metrics, Objectives, and Business Goals

- **Metrics:**
  - **Latency:** The time from user request to summary display (target: <15 seconds).
  - **Performance Degradation:** Due to simultaneous user requests, increase in latency will be measured.
  - **Summary Quality:** A user-facing feedback mechanism (e.g., a simple thumbs up/down) to qualitatively measure summary relevance.
  - **Comment Filter Quality**: Qualitative observation, or ability to filter on preselected examples (Recall/Precision).
  - **User Engagement:** Frequency of use per user, number of active installations.
- **Objectives:**
  - To successfully develop and deploy a functional browser extension.
  - To integrate a self-hosted open-source LLM as the core of the analysis engine.
  - To provide users with fast, accurate, and useful summaries of YouTube content.
- **Business Goals:**
  The measurable goal is reducing service cost via use of open-source models. If the product can deliver similar summaries compared to competitors at lower costs, this is a competitive advantage.
  Qualitatively, the goal is to make a better user experience through an improved interface and unique features. Generating summaries from the home page and performing unique comment filtering are differentiating factors.

8. Failure Analysis

We anticipate several potential risks and have planned mitigation strategies.

- **Potential Risks (During Project):**
  - **API Integration Issues:** Difficulties in setting up and managing the self-hosted LLM API on GCP.
  - **Scope Creep:** Adding too many features without completing the core functionality first.
- **Potential Risks (Post-Deployment):**
  - **API Changes:** YouTube could update its website or API, breaking the extension's functionality.
  - **Poor Quality Summaries:** The chosen open-source LLM may produce incoherent or biased results.
- **Pipeline Failure Mitigation:** The main points of failure are API calls. Failure should not cause browsers to stall or crash, but instead fail gracefully with a descriptive message to the user. Internally, API calls should be retried before a fail message is displayed to the user.
- **General Mitigation Strategies:** We will start with a Minimum Viable Product (MVP), use robust error handling for all API calls, and monitor for performance issues.

9. Deployment Infrastructure

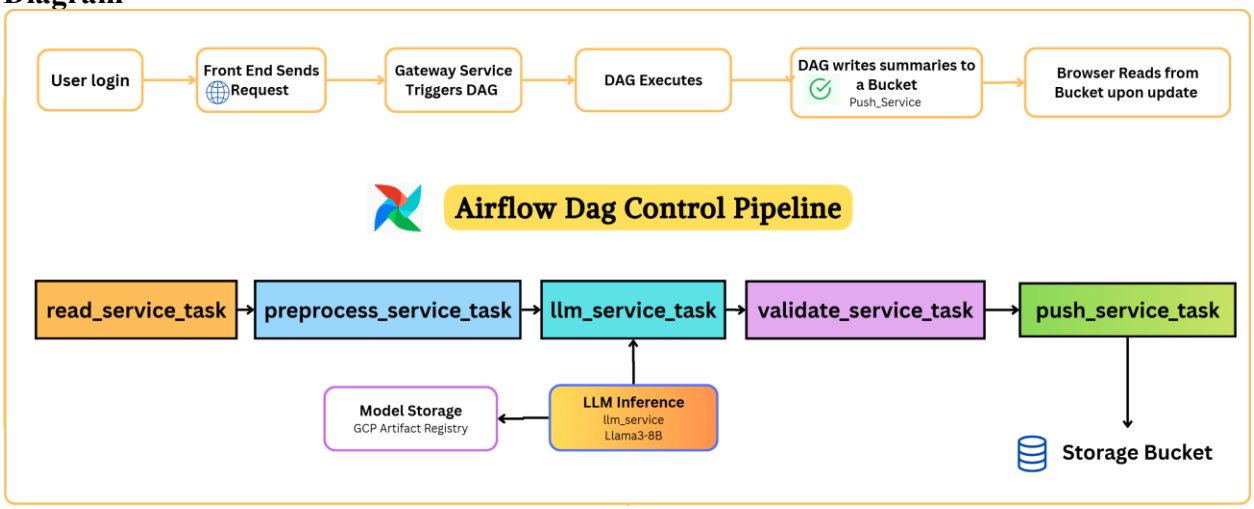
The deployment infrastructure will consist of two main components, and detailed flowchart diagrams will be included as necessary.

1. **Backend (LLM API):**

- **Infrastructure:** Airflow Cloud Composer to process the data in discrete steps. The Airflow UI also ensures proper communication, logging, retry on error, and timing for each step to identify bottlenecks.
- **Platform (GCP)**. Airflow will be hosted with GCP's Cloud Composer. Each DAG step will be a Cloud Run. Storage buckets will be needed to store the open-source model weights, and to store summaries that will be read from the front end (like a database).

2. **Frontend (Client):**
   - **Infrastructure:** A browser extension. Firebase DB to store user information and usage history.
   - **Supported Platforms:** Google Chrome, Mozilla Firefox, and other Chromium-based browsers.

Diagram



10. Monitoring Plan

Our monitoring plan will focus on ensuring service reliability and managing costs. Detailed documentation will be prepared.

- **What to Monitor:**
  - **API Endpoints:** Uptime, error rates, and latency.
  - **Cloud Infrastructure:** CPU, GPU, and RAM usage of the GCP server.
  - **API Key Usage:** Track calls to the YouTube API to stay within quotas.
  - **Model Monitoring**: The model will be monitored in a 'validation_service' as seen above. This service will validate the quality of the summaries generated. Since it is a real-time service, it must be quick. Therefore, a similarity score will be performed between the video title and the generated summaries. If the score is very low, it indicates that the summary quality is very poor, or there was some data flow error causing a mismatch. Alerts will be triggered on the event of low similarity score.

    Additionally, there will be an offline validation step. A test set of YouTube videos will be collected (either manually or passed automatically from videos users' request). Summaries for theses videos and comments will be generated and this test set will be evaluated via LLM as a judge (stronger Open AI model). This offline monitering will give stronger confidence that the model is meeting needs and is not

being biased towards certain demographics by providing worse summaries for certain languages or video genres.

- **Why Monitor:** To proactively detect issues, manage operational costs, and understand usage patterns.

11. Success and Acceptance Criteria

The project will be considered successful upon meeting the following criteria:

- The browser extension successfully installs and activates on a YouTube video page.
- The tool generates a coherent and relevant summary for a video and its comment section (>100 comments) in under 15 seconds.
- Advanced features (like sentiment filtering) are attempted and show some feasibility.

12. Timeline Planning

This is a preliminary project timeline, which can be modified based on given deadlines and constraints.

- **Weeks 1-2:** Project setup, tech stack finalization, acquiring API credentials, and initial GCP setup.
- **Weeks 3-4:** Development of the backend API to host and serve the LLM.
- **Weeks 5-6:** Development of the browser extension frontend.
- **Weeks 7-8:** Integration of the frontend with the backend API, end-to-end testing, and bug fixing.
- **Weeks 9-10:** Implementation of advanced features and final deployment preparation.

13. Additional Information

This project places a strong emphasis on the end-to-end deployment of a machine learning model. A key technical objective is the successful implementation of a self-hosted, open-source Large Language Model, which provides a valuable opportunity to navigate the practical challenges of MLOps. Future work could involve expanding the tool's analytical capabilities or adapting the core technology for other content platforms.