**Complexities**

**Team 3**

**Sec - 005**

**Team Members: -**

* Gitansh Mittal
* Dhrumit Patel
* Alakh Bharatkumar Patel
* Divya Agrawal
* Henil Rajeshkumar Patel
* Khushboo Koshti
* Shubham Shivaji Savalajkar

**1.0 Users of the Proposed System:**

**Primary Users Description:**

The users of NotivAI, our AI Video Summarizer, encompass various roles with a focus on enhancing accuracy and predictability while ensuring user-friendliness. The primary users are:

**User Profiles:**

**Students:**

**Frequency of Use:** Regularly, especially during academic sessions**.**

**Expertise with the Domain:** Varied, limited to their educational content**.**

**Proficiency with Computers and Software:** Moderate**.**

**Proficiency with the Software Being Developed:** Varied**.**

**Goal for Using the Software:** Efficiently review lengthy educational videos, saving time in the learning process.

**Teachers:**

**Frequency of Use**: Periodically for reviewing and customizing summaries.

**Expertise with the Domain**: Moderate to high, depending on their subject matter.

**Proficiency with Computers and Software**: Generally proficient.

**Proficiency with the Software Being Developed**: Basic to moderate.

**Goal for Using the Software**: Tailor content summaries to suit students' comprehension levels, enhancing the teaching experience.

**Educational Content Creators (Potential Secondary Users):**

**Frequency of Use**: Occasionally for evaluating and improving content.

**Expertise with the Domain**: High, particularly in content creation.

**Proficiency with Computers and Software**: High.

**Proficiency with the Software Being Developed**: Moderate to high.

**Goal for Using the Software**: Analyze the effectiveness of video summaries, refine content based on user engagement.

**Administrators (Potential Secondary Users):**

**Frequency of Use:** Periodically for system management.

**Expertise with the Domain**: Moderate to high.

**Proficiency with Computers and Software**: High.

**Proficiency with the Software Being Developed**: High.

**Goal for Using the Software**: Ensure the smooth operation of NotiveAI, handle technical aspects.

**Overall System Goal:**

Our aim is to make NotivAI easy and friendly for everyone to use. Even if you're not very technical, we want you to find it simple and straightforward. While making sure it's accurate and works well, we also want to empower users, helping them use NotivAI efficiently in their roles. We care about making it user-friendly and easy to use, so everyone can benefit from its accuracy without feeling overwhelmed by technical stuff.

**2.0   Front-End User Interface (25 pts.)**

For each user of the proposed system, provide mockup user interfaces.

*25 pts. deduction for incorrect, incomplete, nonsensical, or non-submission per occurrence.*

**Identification of users:**

Students and Educators: Primary end-users who will utilise the NotivAi system to access summarised educational video content and review structured notes.

**Mockup Interface:**

Landing Page: The user will land on to this page after a search on internet for NotivAI and this page will have info regarding the application tutorial how to use it. and will have the direct link for the testing of the application that will take user to Login Page.

Login Page: The user will have an option to either login or register. This can be done manually by entering email & password or the user can do it through google.

Home page: The user will have a prompt to upload the video. Additionally, the user has access to a history of the videos they have submitted as well as the previously generated synopsis. (In the future, this can be applied to premium user accounts with proper funding.)

User Profile Page: The user may easily access this page to view account details, modify their name, password, and profile picture.

Uploading Video: The user has two options on the home page, they can select a file from their computer directory or upload it directly using the YouTube link.

Video Player: To ensure they have entered the correct video, users can preview the one they wish to submit on the home page before uploading it.

Loading Page: This page appears after the user uploads the video and informs them that it is taking a while to load. In the meantime, the AI backend summarises the video and sends the transcript and summary to the next page. The user will be automatically redirected to the next page when results are ready.

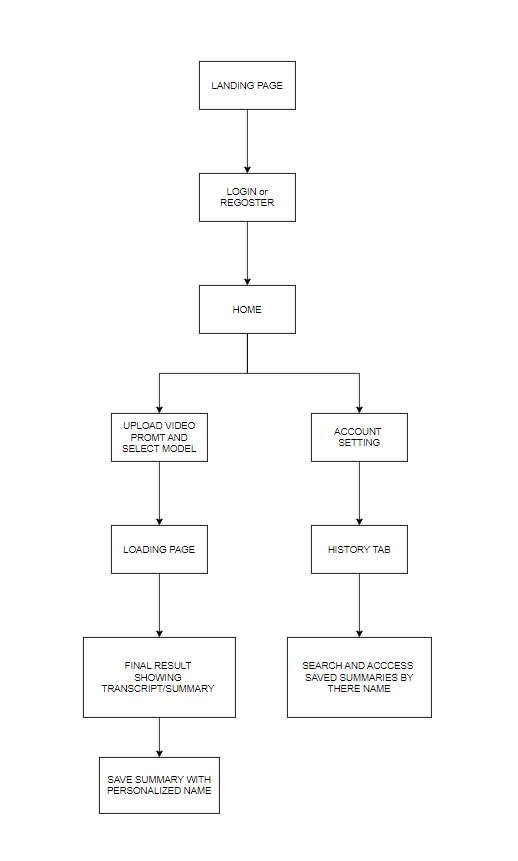
Summary Page: Presents the summarised notes and transcripts generated by the

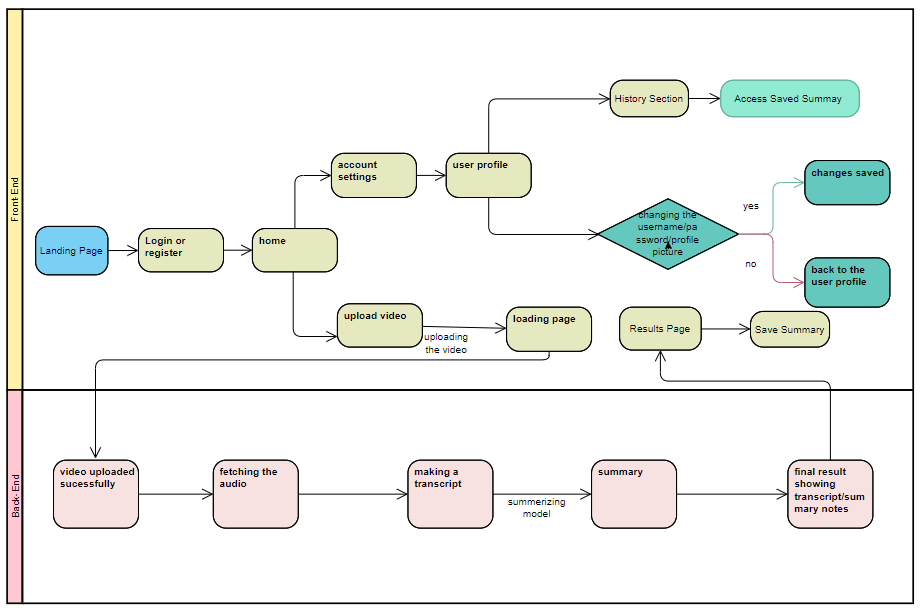
system for each video.

**FUTURE UPDATES:**

1. User Feedback Page
2. Admin Dashboard
3. Notes of Summary
4. **cross-platform integration**

**USER FLOW:**





**Flow:**

**Landing Flow:**

User will find the info for application.

Tutorial for how to use it

The Info for the company

The direct link to try the application.

**Login Flow:**

User enters login credentials.

If credentials are valid, user is redirected to the Home page.

If credentials are invalid, appropriate error message is displayed.

**Navigation Flow:**

From the Home page, users can navigate to the Video Upload section, User Profile section.

After completing an action (e.g., changing password), users can navigate back to the Home page or continue to other sections as needed.

**Video Upload Flow:**

User navigates to the Video Upload section.

User selects a video file or provides a YouTube link.

User initiates the upload process.

System processes the uploaded video and redirects user to the Loading page.

**Change Password or any other relevant sources Flow:**

User navigates to the User Profile section.

User selects the option to change password/username/profile picture.

User did the relevant changes.

User saves the changes, and is updated in the system.

**Video Processing Flow:**

After video upload, system extracts audio and transcribes it into text.

Text is passed to the summarization model, which generates a summary.

Summary and original transcript are displayed to the user for review.

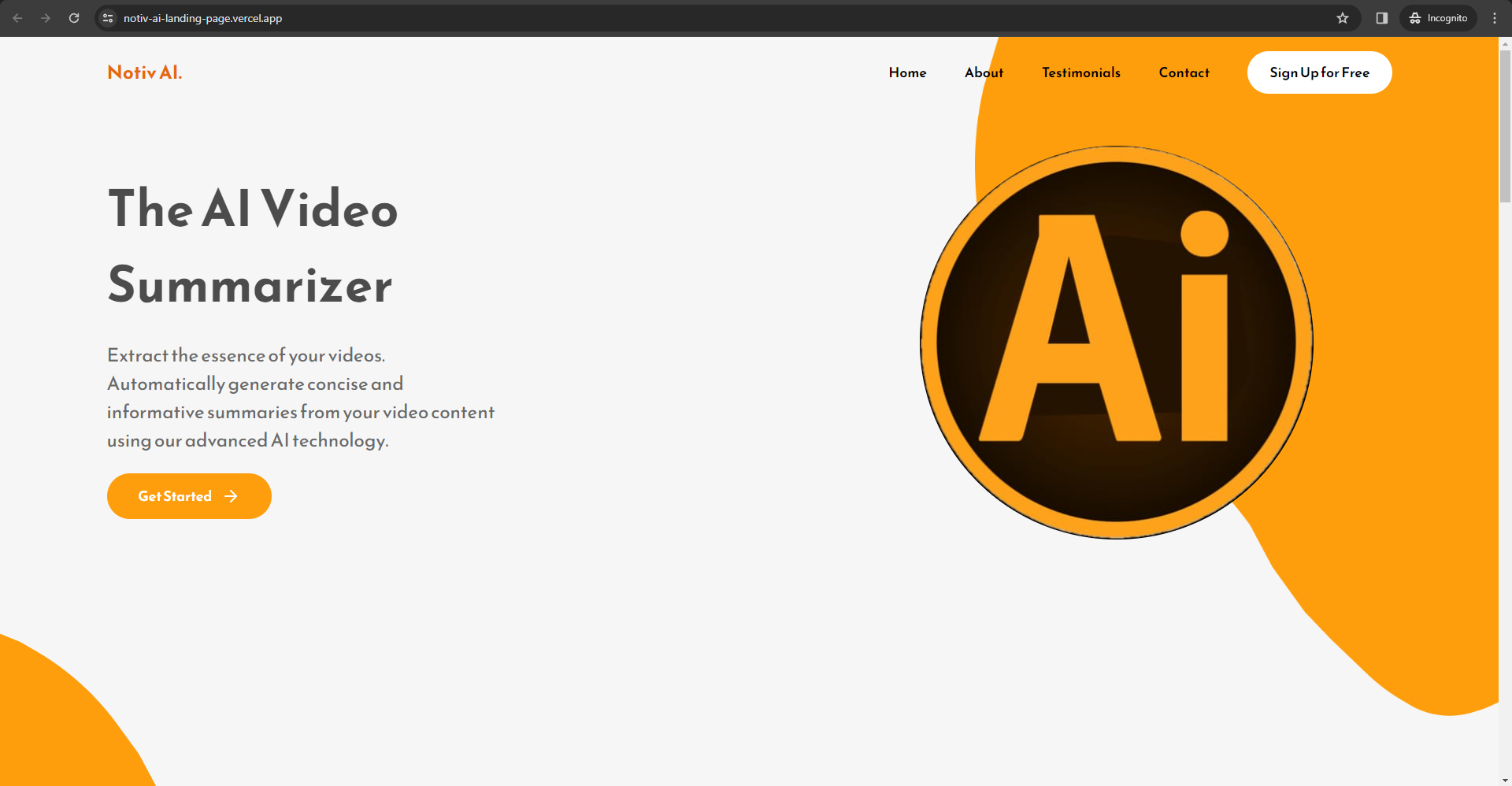
**History Flow:**

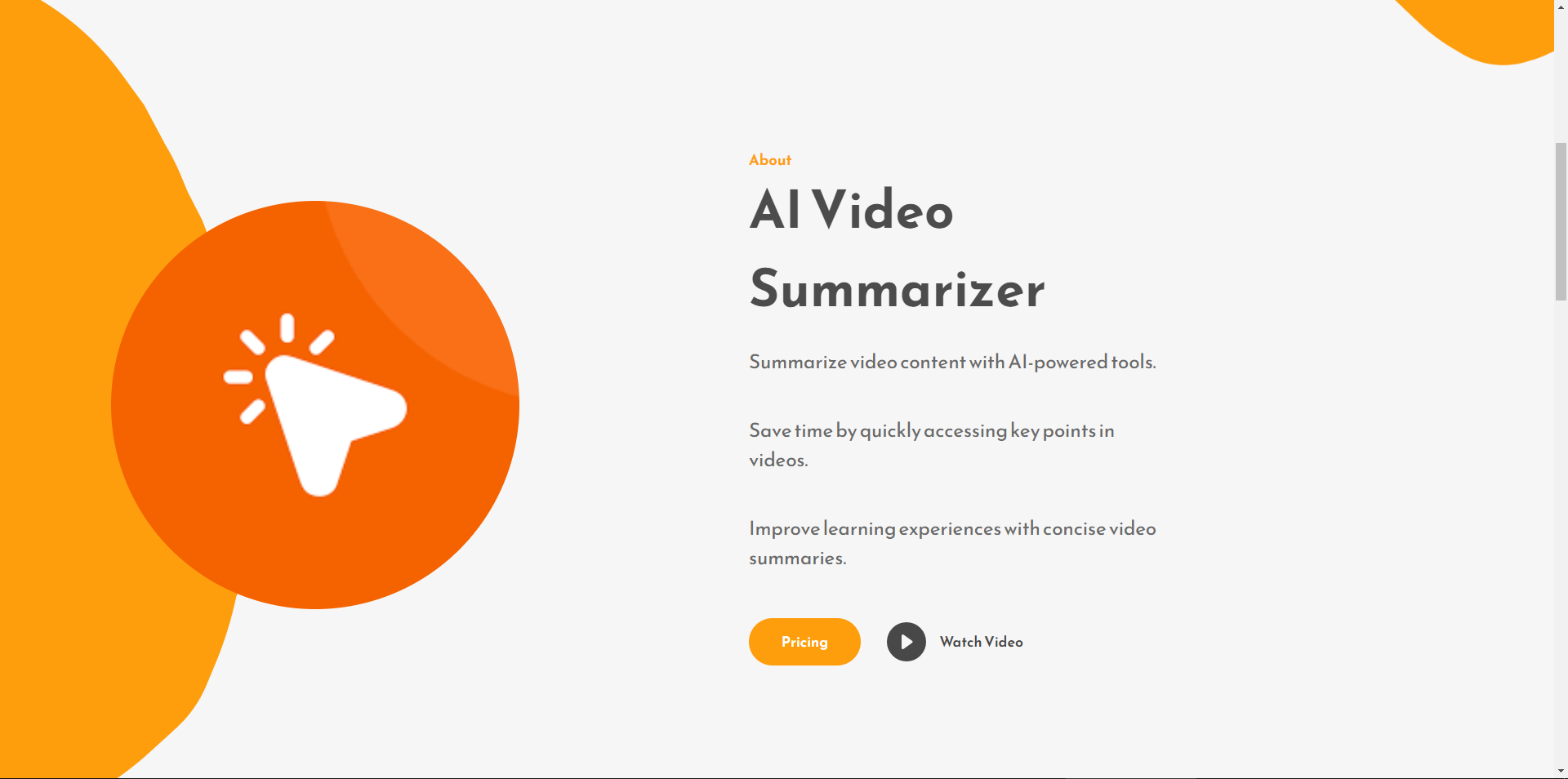
User will save the summary with a Name to access it in future

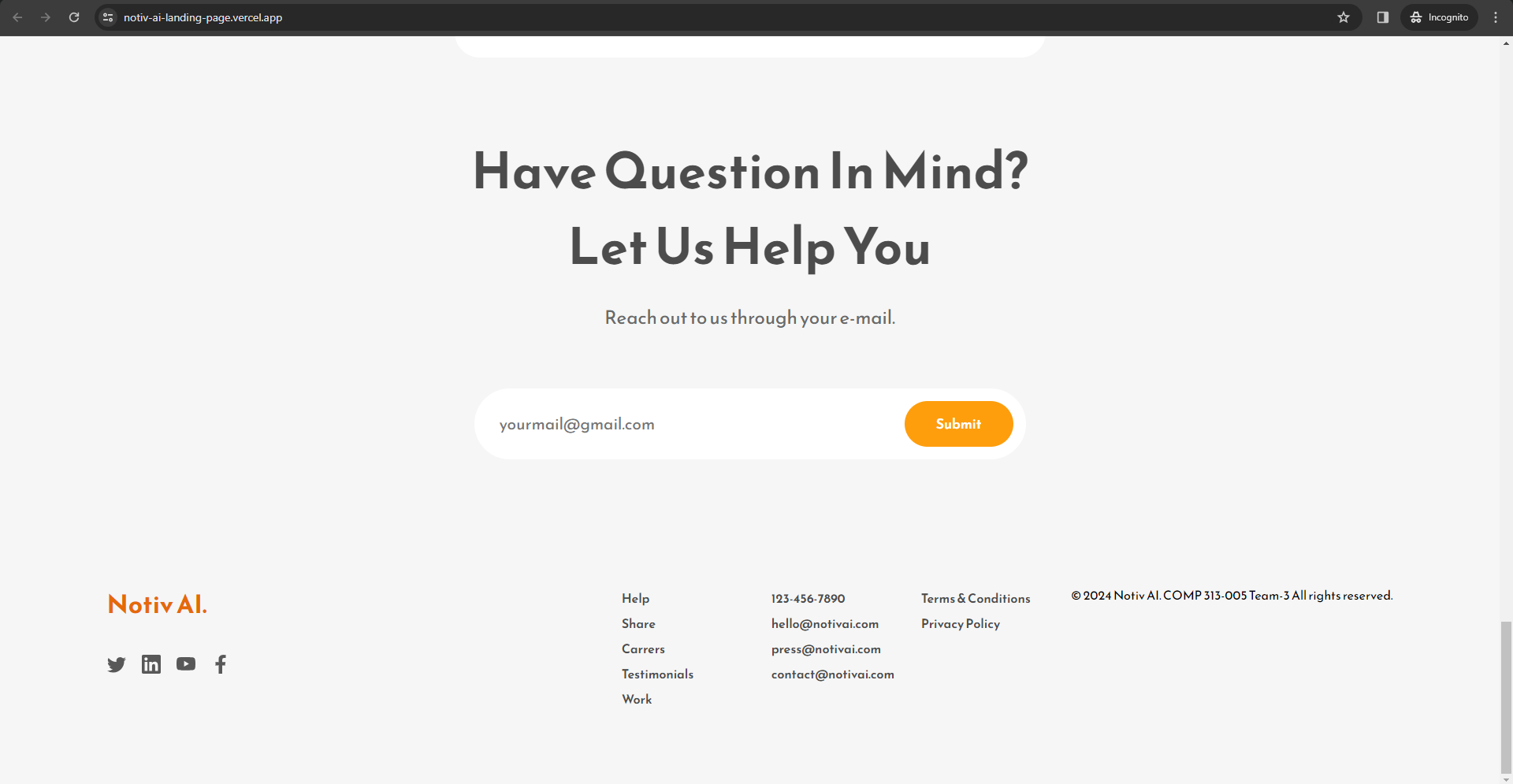
User will go to profile section switch to History tab.

User can search and access the later saved summary by there name.

**LANDING PAGE:**







**LOGIN PAGE:**

A screenshot of a login screen

Description automatically generated   
   
 

**REGISTER:**A screenshot of a login form

Description automatically generated

**HOME PAGE:**

A screenshot of a computer

Description automatically generated

**USER PROFILE:**

A screenshot of a computer

Description automatically generated

**VIDEO PREVIEW AFTER UPLOAD:**

A screenshot of a computer

Description automatically generated

**LOADING STATE WHILE VIDEO IS BEING PROCESSED:**

A screenshot of a computer

Description automatically generated

**SUMMARY PAGE (1):**

A screenshot of a computer

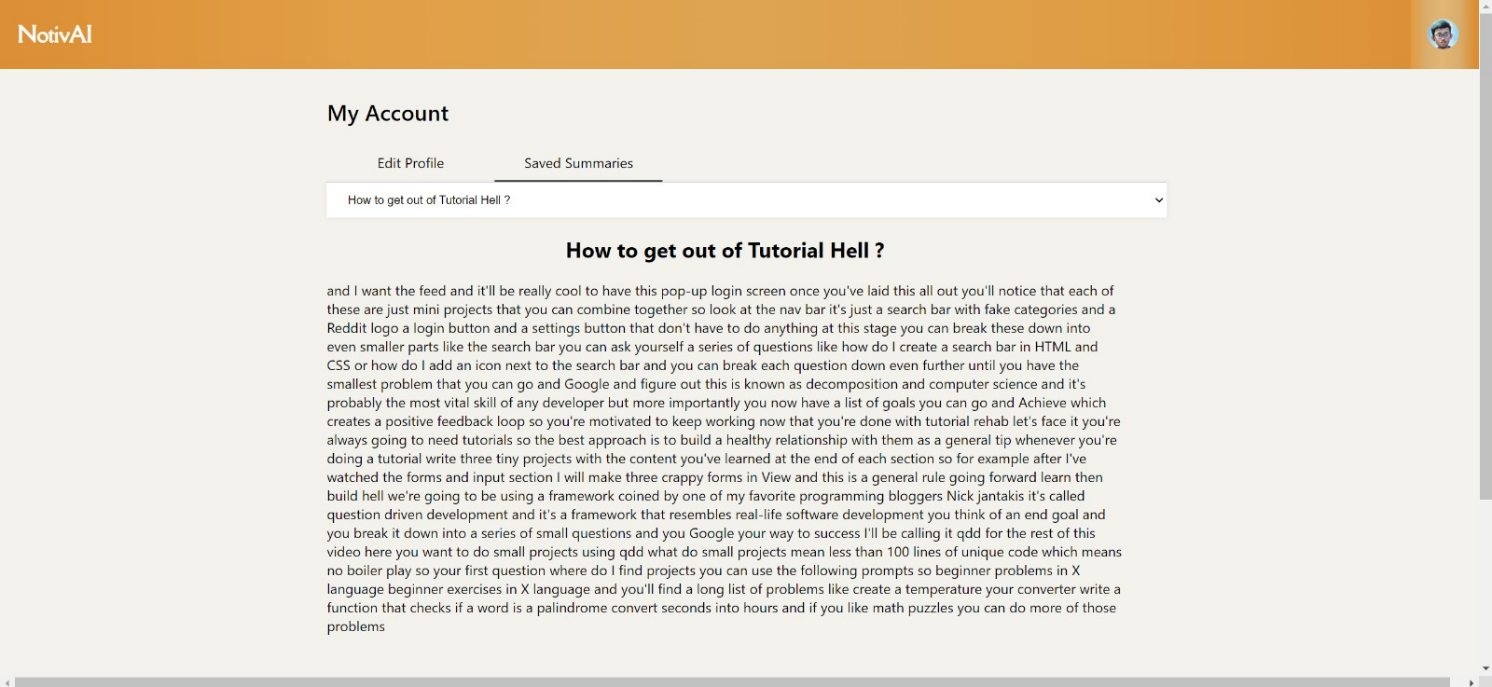
Description automatically generated

**SUMMARY PAGE (2):**

A screenshot of a computer

Description automatically generated

**HISTORY PAGE:**



**3.0** **Cloud Back-End (25 pts.)**

Our project focuses on transforming video content into text transcripts and generating concise summaries. The cloud back-end plays a critical role in achieving this goal. Here’s how it fits into the system:

Input: We receive video content (e.g., YouTube videos) that needs processing.

Processing: The cloud back-end handles the heavy lifting:

* Extracts closed captions or transcripts from the video.
* Applies summarization techniques to condense the content.

Output: The summarized text or transcript is then made available to users via an API.

**Flask Framework**

We’ve chosen Flask as our API framework.

* *Flask: A lightweight web framework for Python, Flask simplifies creating RESTful APIs. Its ease of use makes it an excellent fit for our project, especially when handling HTTP requests and responses.*

**Workflow**

1. *API Calls: Clients (such as our web application or other services) make API calls to our Flask back-end.*
2. *Processing Request: Flask receives the API request and processes it.*
3. *Transcript Extraction: For YouTube videos, Flask extracts well-formatted closed captions.*
4. *Summarization: Using various summarization techniques, Flask generates a concise summary of the video content.*
5. *API Response: Flask sends the summarized text back to the client.*

**Cloud Deployment**

Frontend Deployment on Vercel:

* The front-end interface of the text summarization system is hosted on Vercel.
* Vercel's platform offers seamless and accessible user experiences.
* Leveraging Vercel's capabilities allows for rapid deployment and easy management of the user interface.
* Hosting the front end on Vercel enhances accessibility and usability for end-users.

Backend Functionality on Hugging Face's Platform:

* Core functionality, including models and the Flask API (app.py), is hosted on Hugging Face's platform.
* Centralization of backend components on Hugging Face streamlines model management.
* Ensures consistent performance across different modules of the system.
* Hugging Face's platform provides a robust infrastructure for model hosting, enabling efficient inference and scalability.

Codebase Management on GitHub:

* The system's codebase is stored on GitHub.
* GitHub facilitates version control, collaboration, and seamless integration with CI/CD pipelines.
* Acts as a central repository for the project, enabling developers to collaborate and track changes effectively.
* CI/CD pipelines automate testing, building, and deploying changes, ensuring reliability and stability.

Advantages of Distributed Architecture:

* Offers scalability, reliability, and ease of maintenance.
* Optimizes resource utilization by leveraging specialized platforms for different components.
* Ensures high availability of the system.
* Use of Git and CI/CD pipelines enhances development workflows, enabling rapid iteration and deployment of new features or updates.

Conclusion:

* The deployment architecture of the text summarization system maximizes efficiency, reliability, and scalability.
* Provides a seamless user experience and facilitates collaborative development.
* Combining front-end deployment on Vercel, backend functionality on Hugging Face, and codebase management on GitHub creates a robust and flexible system architecture.

**Use Case Scenario**

* *Long Videos: Summarization proves especially valuable for lengthy videos. Viewers can save time by focusing only on essential content.*
* *Improved Productivity: Users benefit from efficient consumption of video information.*

We’re building functionality to summarize YouTube videos with owner-added captions. Our goal is to provide users with concise text responses using advanced summarization techniques.

Remember, this back-end also hosts a web version of the summarizer, making API calls straightforward and displaying output within a webpage.