

**MICHIGAN STATE**  

---

**UNIVERSITY**

# Project Plan Presentation

## Automated Content Editor

### The Capstone Experience

Team TechSmith

Joe Baran

Emily Feuer

Justin Masters

Gabriel Sotelo

Riley Tucker

Department of Computer Science and Engineering  
Michigan State University

Fall 2023



*From Students...  
...to Professionals*


# Project Sponsor Overview

- Software company focusing on visual media
- Established in East Lansing in 1987
- Flagship products are Camtasia and Snagit
- Serve 73 million users worldwide



# Project Functional Specifications

- Video content creation is essential for modern companies to communicate with their customers.
- However, creating video content can be time-consuming and difficult for those without experience.

 ACE (Automated Content Editor) allows quick and simple video editing for creators of all experience levels by allowing users to simply tell the application what editing operations they want carried out.

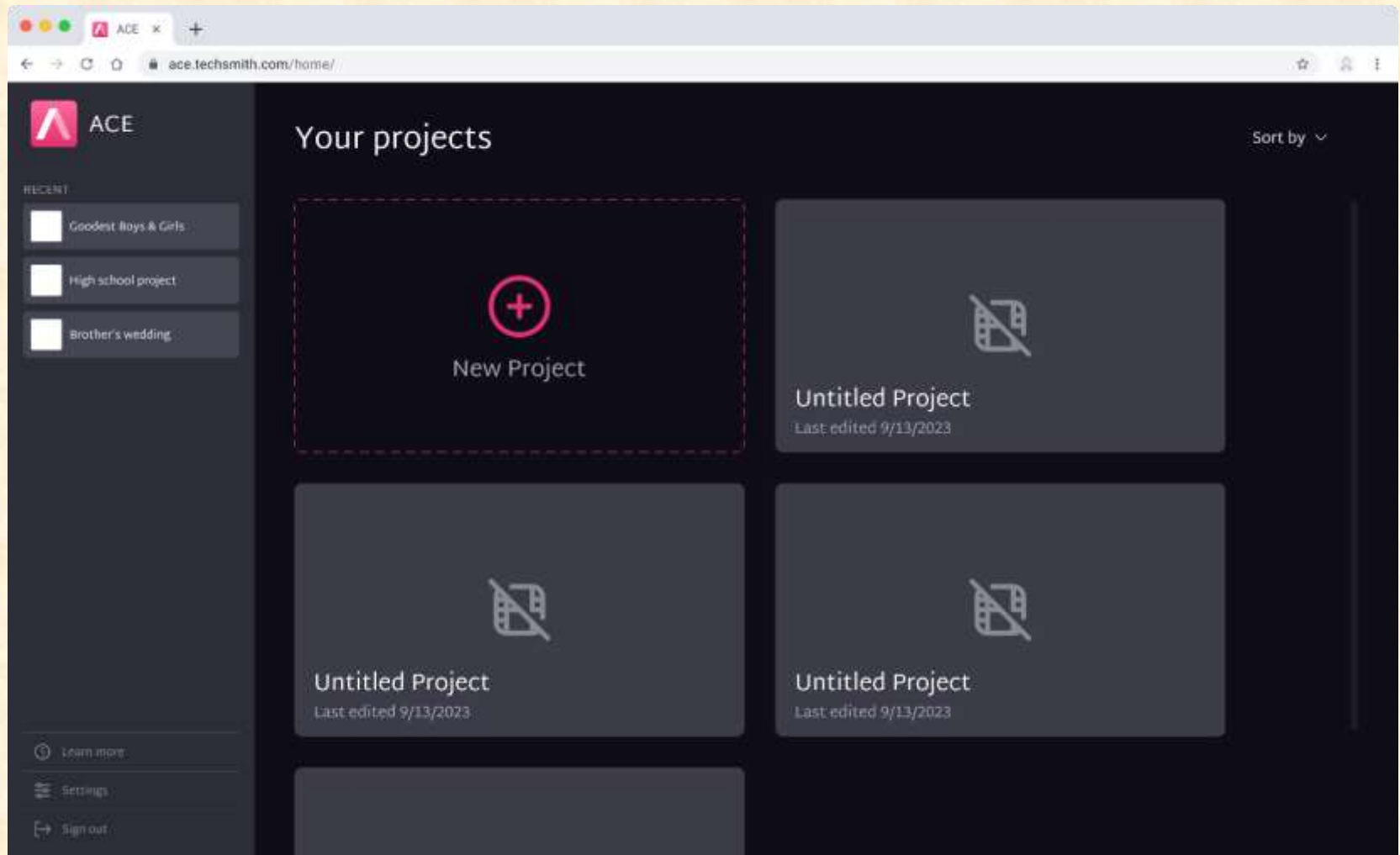


# Project Design Specifications

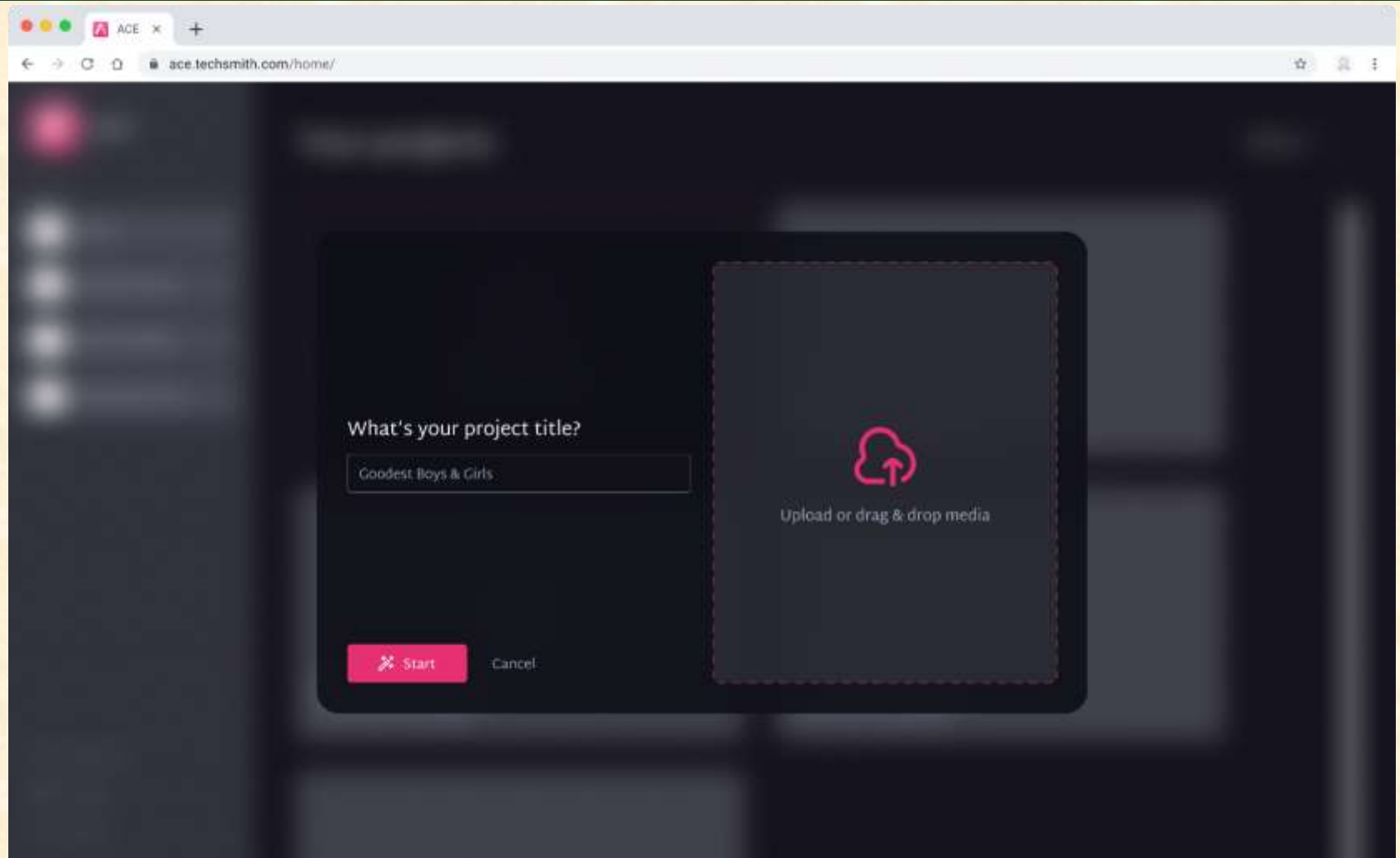
- Web application designed to facilitate video editing for users of all experience levels.
- Users upload audio, images, and video or select from provided stock media.
- Users communicate with an AI chatbot to generate video editing commands.
- Simple readable UI to make editing as straightforward as possible.
- Users create projects to keep files for different editing projects separate.
- Projects can be managed across sessions with state being maintained.



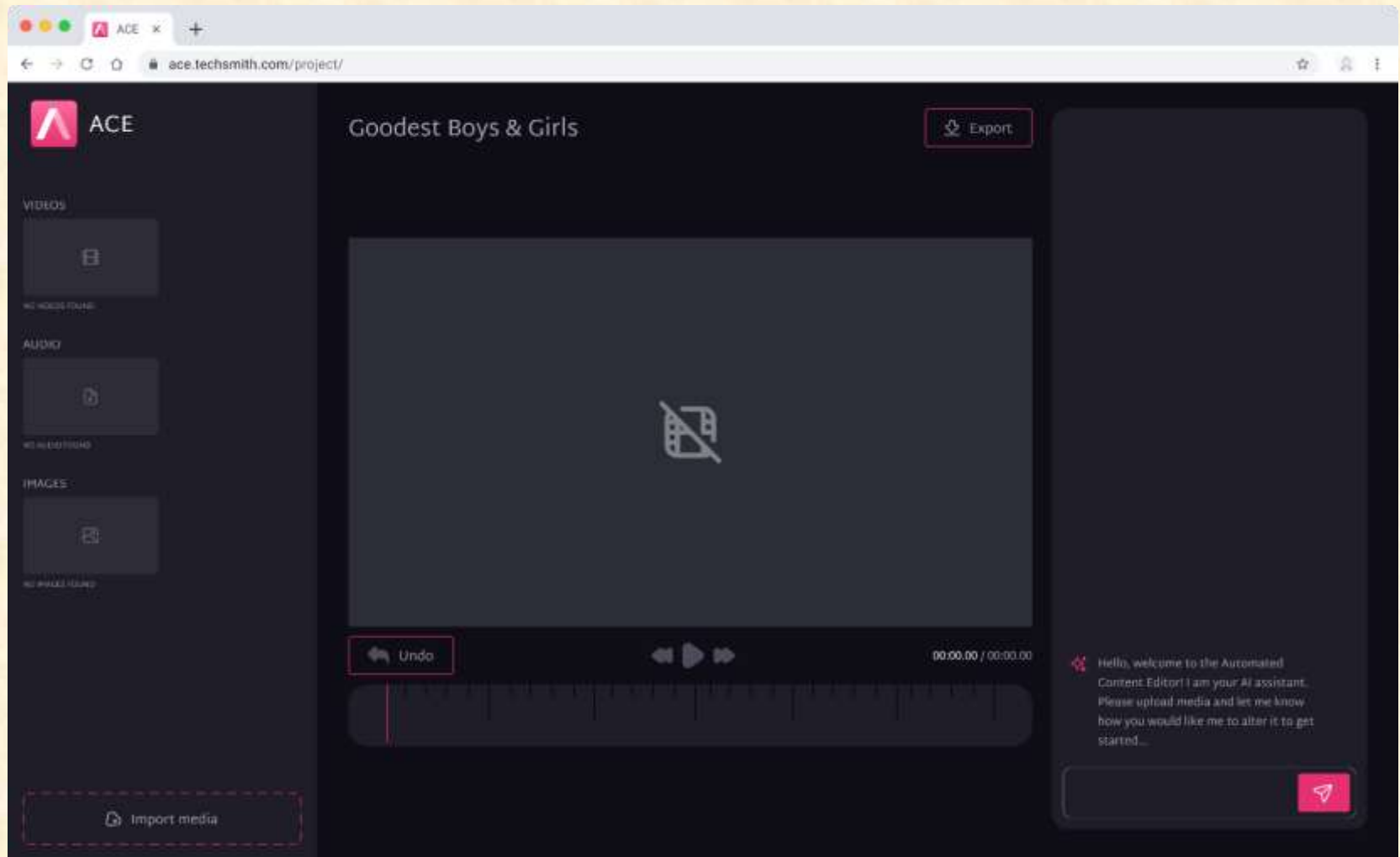
# Screen Mockup: Project Dashboard



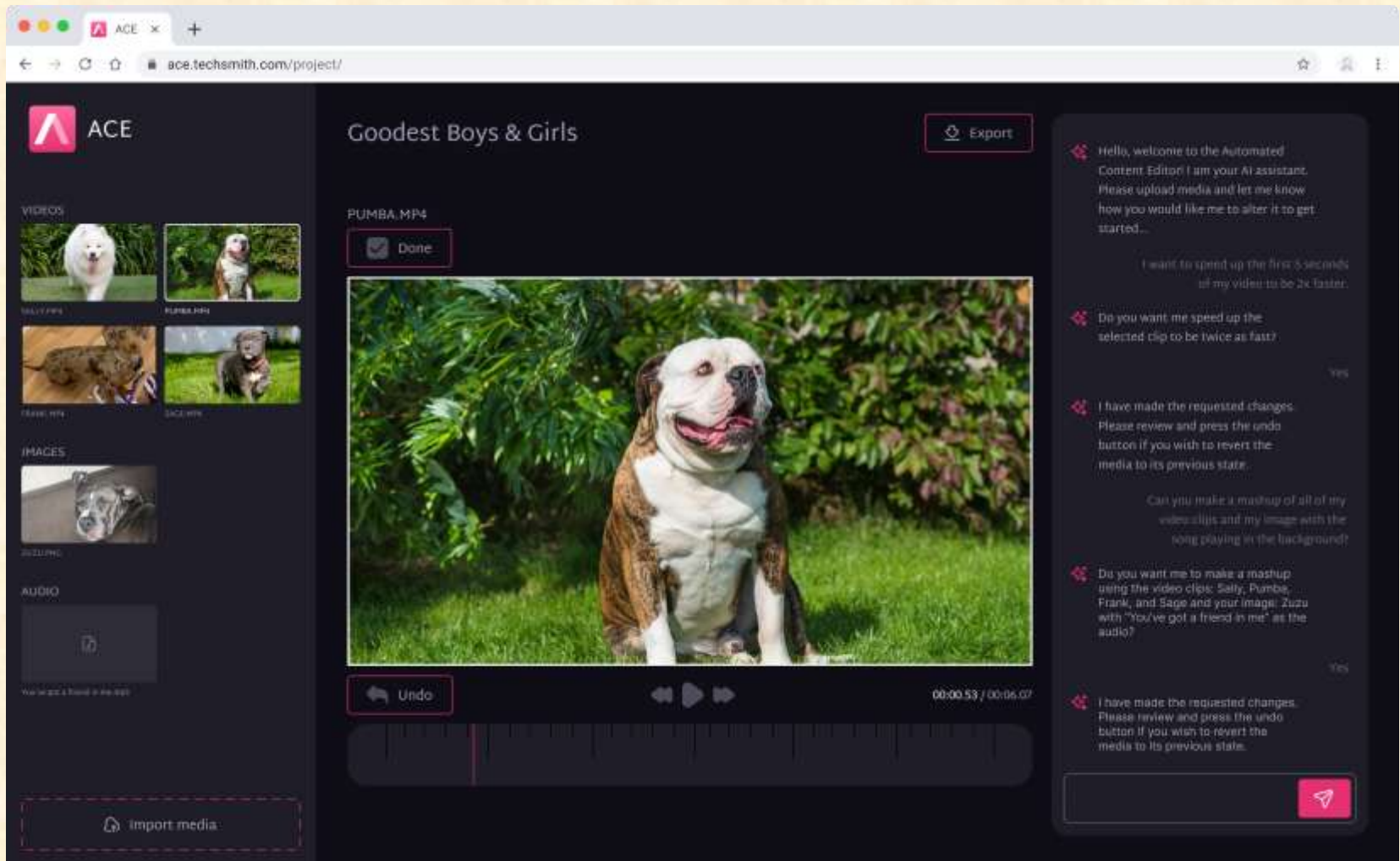
# Screen Mockup: Project Creation



# Screen Mockup: Blank Project



# Screen Mockup: Project Editing



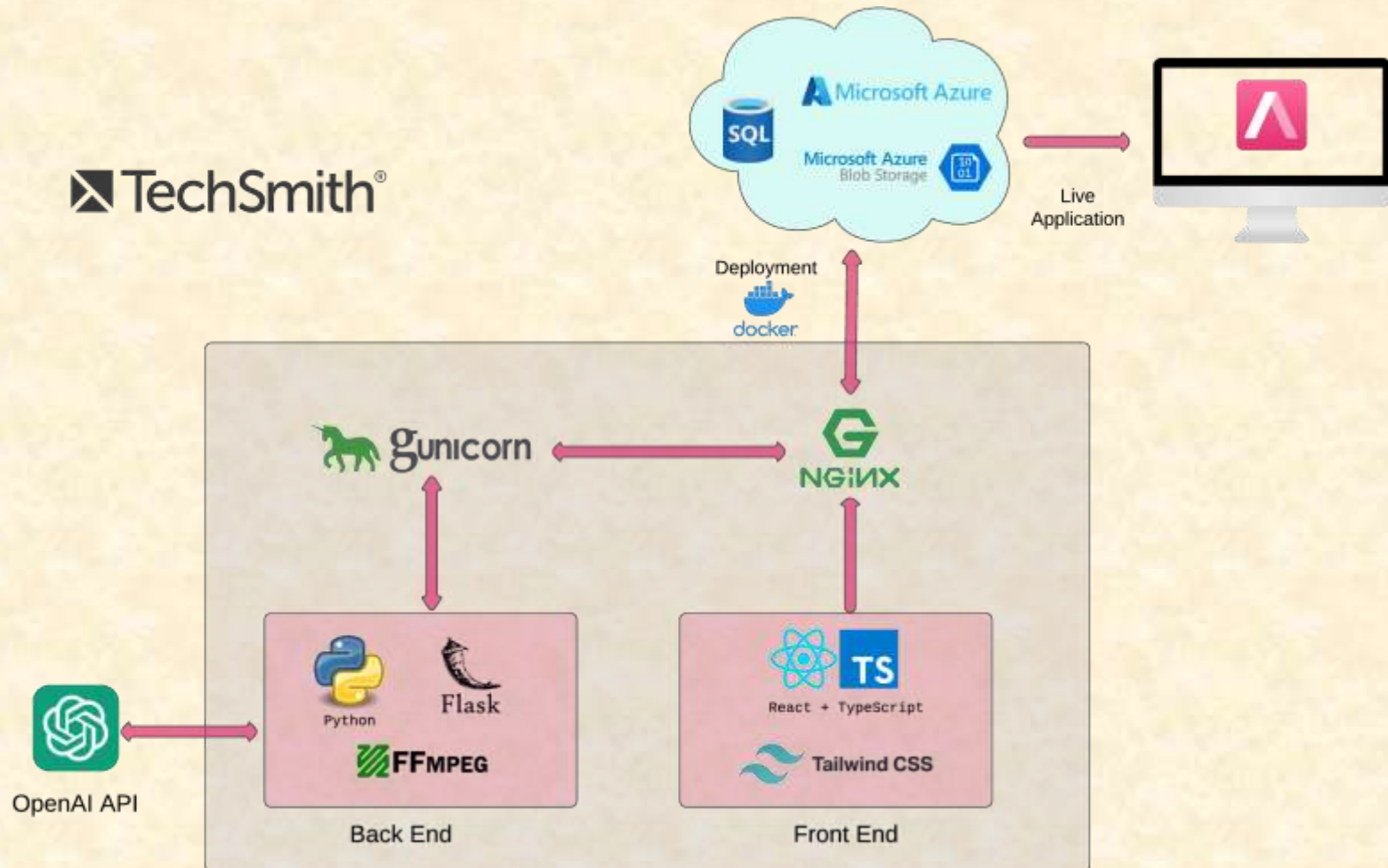


# Project Technical Specifications

- ACE's is a web application with a front end implemented using the React library's Create React App framework with a Typescript template.
- ACE's backend is implemented as a Python Flask application to create an API, manage the underlying database, and facilitate WebSocket communication.
- FFmpeg is used to facilitate media processing operations.
- Docker is used to containerize the application and allow for a simple, single-container deployment to Azure web services where the live application and related databases are hosted.
- The application is served from an Nginx web server with a Gunicorn server gateway to ensure production-quality request-handling and security.



# Project System Architecture



# Project System Components

- **Software Platforms / Technologies**
  - **React (+ TypeScript + Tailwind CSS)**
    - Popular JavaScript library for building UI of dynamic web applications
  - **Flask (+ Gunicorn)**
    - Micro web framework used to process incoming user requests to backend API
  - **FFmpeg**
    - Multimedia framework used extensively for media processing applications
- **Cloud System**
  - **Azure Blob Storage**
    - Object storage platform designed to hold unstructured binary data (media files)
  - **Azure SQL**
    - Storage platform used for structured data (user profiles, settings, edit history)
  - **Azure OpenAI -> GPT-3.5 Turbo**
    - GPT-3.5 is a massive language model that is accessible through Azure OpenAI via its API



# Project Risks

- Responsiveness of live web application
  - Video editing is a computationally-expensive operation, so handling those operations in the application backend could negatively impact performance and become expensive to host.
  - *Mitigation:* Experiment with downscaling videos and front-end media processing to exploit user hardware.
- Interpreting user input
  - User input may be difficult to interpret, especially if they are not aware of the applications limitations.
  - *Mitigation:* Prompt the AI to ask for clarification from the user and offer guidance to users to assist them in providing interpretable instructions.
- Database implementation
  - The web application requires the storage of a lot of data to save uploaded media, project metadata, state information, etc. Our implementation will use different databases to store different kinds of data, which risks “wire crossing” and complications in implementation.
  - *Mitigation:* Get a working database prototype running and progressively add functionality (Azure SQL -> Azure SQL & Blob -> Azure SQL and Blob working in tandem).
- Compatibility of ChatGPT with FFmpeg
  - ChatGPT may have difficulty consistently creating commands for FFmpeg that can be interpreted correctly by Automated Content Editor.
  - *Mitigation:* Use prompt engineering to constrain and simplify the output of ChatGPT by making the initial prompt as detailed and granular as possible.



# Questions?

---

?

?

?

?

?

?

?

?

?

