## horizontal line



SRS for Mr. Tai (M)ulti-media (R)eal (T)ime AI

03.015.2025

**─**

Tristian Sellers

CPSC 4800

# Table Of Contents

[**Table Of Contents 1**](#_yyrhu7ml5bea)

[**Overview 3**](#_au51mny0sx6)

[**Goals 3**](#_3at9u9s4e0vp)

[**Specifications 3**](#_4p7xi5bvhxdr)

[(M)inimum (V)alue (P)roduct 3](#_56kfpodyq5td)

[**Product Scope 4**](#_vkcx2hr40rdf)

[Business Goals 4](#_aesxnld89zsy)

[Key Benefits 4](#_9utppe6yp7aq)

[Objective & Goals 4](#_cu69xeasulls)

[**Product Value 5**](#_lznn0jjyakv6)

[Why is Mr. Tai Important? 5](#_ip3648eofl0i)

[How Will It Help the Intended Audience? 5](#_rvha2eu2m00)

[What Problem Does it Solve? 5](#_rsnj0r4djb1e)

[**Intended Audience 6**](#_jv3k7sueuc70)

[**Intended Use 7**](#_eh91pf44hh3d)

[Primary Use Cases 7](#_7ub0zfkpotiz)

[**Definitions & Acronyms 8**](#_w44yephqmbvu)

[General Terms 8](#_898ie6xo0o23)

[Project Specific Terms 8](#_np81qqf9nl3l)

[Technical Terms 8](#_7ed1wz8snv5j)

[**System Requirements & Functional Requirements 9**](#_zf0l82d8m141)

[Development Considerations 9](#_mksw7ovmmhh9)

[Performance Optimization 9](#_qo1njhi0yme7)

[**External Interface Requirements 10**](#_owwep4lyovmr)

[User Interfaces 10](#_vgmq4gen0ncl)

[Hardware Interfaces 10](#_rz4k9o67kve6)

[Software Interfaces 11](#_4u3d673cfe8k)

[Communication Interfaces 11](#_8xctp3o9z1er)

[**Non-functional Requirements (NFRs) 12**](#_evy4ecfpmdvh)

[Security 12](#_arv5g5ktioba)

[Capacity 12](#_n9eayw28s269)

[Compatibility 12](#_yiwslxtbt6h4)

[Reliability and Availability 13](#_uu365f4pdl89)

[Scalability 13](#_yt419iw61jqp)

[Maintainability 13](#_uhkawzx4for5)

[Usability 14](#_pzj3ith5zror)

[Performance 14](#_49j5wz1q49)

[Specific Development Tools 15](#_6nkej1eyal3n)

[Start With: 15](#_45qfn1o79ife)

[Flow Chart 16](#_l6kkr8rrarda)

[Wireframe 17](#_5w79d7rgfxm3)

[Milestone Timeline 22](#_3l4exkw81l0v)

# 

# Overview

***Mr. Tai*** aims to revolutionize the gaming world through dynamic voice acting, which allows for a fresh experience each time one plays, boosting a game’s immersiveness and replayability.

# Goals

1. An AI-powered system that generates real-time voice-over commentary on football game clips
2. A website where users can upload *Madden* clips, select a commentator style, and receive AI-generated commentary based on that style.

# Specifications

***Mr. Tai*** will be focused on voice acting as a commentator, reacting to Madden footage received via user. It will generate dynamic voice commentary based on in-game events. Users can access ***Mr. Ta*i** through a website.

## (M)inimum (V)alue (P)roduct

***Mr. Tai*** will be able to receive at most 90-second *Madden* gameplay videos from the user through a file upload button on the website. Users must select a commentary type from a dropdown button that presents the options. Once the previous two criteria are met, a button that says “Hear from Mr.Tai” shall pop up, and once pressed, ***Mr. Tai*** shall play the video and begin its real-time commentary.

# Product Scope

***Mr. Tai*** is designed to revolutionize voice acting in gaming by providing real-time, dynamic narration, starting with football video game clips. The platform aims to enhance gaming immersiveness by automating play-by-play analysis with AI-generated voice-overs, eliminating repetitive commentary, starting with sports games.

## Business Goals

* Develop an AI-powered system that can analyze football gameplay and generate realistic voice commentary.
* Provide a scalable, web-based solution for users to upload gameplay clips and receive narrated versions.
* Lay the foundation for future real-time AI commentary that could be integrated into live gameplay.

## Key Benefits

* **Enhanced User Experience**: Reduces repetitive, pre-recorded commentary in sports games.
* **Content Creation**: Helps gamers and streamers produce unique game highlights with AI-driven voiceovers.
* **Scalability**: Designed as a web service, enabling future expansion to other sports and real-time applications.
* **Cost Efficiency**: Reduces the need for extensive human voice-over work for dynamic commentary.

## Objective & Goals

* Build a functional prototype to process 90-second *Madden* gameplay clips and generate AI-driven commentary.
* Develop an intuitive web interface to upload, process, and retrieve narrated videos.
* Utilize machine learning models to detect key in-game events and generate contextually accurate commentary.
* Optimize text-to-speech (TTS) output to produce natural and engaging narration.
* Establish a scalable framework that allows future enhancements, including real-time integration in live sports gaming.

# Product Value

***Mr. Tai*** addresses a major limitation in sports video games: repetitive and predictable commentary. In traditional sports games, pre-recorded lines from commentators quickly become repetitive, diminishing the immersive experience. This project leverages AI to generate dynamic, play-by-play narration, ensuring a fresh and engaging commentary experience for every game.

## Why is **Mr. Tai** Important?

* **Eliminates Repetitive Commentary** – Unlike pre-recorded lines that repeat after multiple play sessions, ***Mr. Tai*** generates unique, context-aware narration for each uploaded clip.
* **Enhances Immersion** – By reacting to in-game events in a dynamic way, the AI commentator makes gameplay highlights feel more natural and exciting.
* **Empowers Content Creators** – Streamers and gamers can generate creative, narrated highlights without the need for expensive editing or voiceover work.
* **Lays the Foundation for Future AI Commentary** – While starting with post-game analysis, ***Mr. Tai*** serves as a stepping stone for real-time AI-driven sports commentary.

## How Will It Help the Intended Audience?

* **Gamers & Streamers** – They can easily create engaging, narrated highlight clips for social media and streaming platforms, as well as enjoy a fresh, immersive gaming experience each time they play.
* **Sports Game Developers** – This technology could eventually be integrated into sports games to provide dynamic in-game commentary.
* **Esports & Sports Media** – AI-generated sports commentary could be used to analyze and narrate gameplay from various perspectives, offering unique insights.

## What Problem Does it Solve?

* **Prevents Repetitive Dialogue in Sports Games** – Current game commentary relies on pre-recorded lines, making the experience predictable.
* **Immersion & Replayability** – Dynamic commentary helps each playing experience feel unique and exciting.
* **Optimizes Voice Actor Workflows** – By automating repetitive lines and filler commentary, ***Mr. Tai*** allows voice actors to focus on more creative, high-impact voice work.

# Intended Audience

The intended audience for ***Mr. Tai*** includes gamers, content creators, sports game developers, esports organizations, and voice actors. Each group benefits from AI-generated, dynamic sports commentary in different ways:

* **Gamers & Streamers** – Players who want to enhance their gaming experience with unique, AI-driven commentary. Content creators can use ***Mr. Tai*** to generate high-quality, engaging narration for gameplay clips shared on social media and streaming platforms.
* **Sports Game Developers** – Game studios working on sports titles can use ***Mr. Tai*** as a prototype for AI-driven in-game commentary, replacing static pre-recorded dialogue with a more responsive system.
* **Esports Organizations & Sports Broadcasters** – AI commentary can provide alternative perspectives, highlight key plays, and enhance live esports coverage with real-time analysis.
* **Voice Actors & Broadcasters** – ***Mr. Tai*** doesn’t replace voice actors but offers a way for them to scale their work, licensing their voices for AI-powered narration while reducing the need for repetitive recording sessions.

This diverse audience influences the design of ***Mr. Tai***, ensuring an intuitive, scalable, and accessible experience for both casual users and industry professionals.

# 

# Intended Use

***Mr. Tai*** provides AI-generated voice commentary for sports video game footage, allowing users to upload gameplay clips and receive dynamically narrated play-by-play analysis. The product serves multiple use cases depending on the user’s role:

## Primary Use Cases

1. **Gamers & Content Creators**

* Upload a gameplay clip (e.g., a recorded Madden highlight).
* Select a commentary role (e.g., play-by-play announcer, color commentator).
* Receive an AI-generated narration of the clip for use in highlight reels or social media posts.

1. **Sports Game Developers**

* Integrate ***Mr. Tai*’s** AI model as a prototype for real-time in-game commentary.
* Test AI-generated voice-overs in different in-game scenarios to explore dynamic commentary options.

1. **Esports Organizations & Broadcasters**

* Generate AI commentary for esports matches and highlights.
* Provide alternative commentary styles or perspectives for live sports coverage.

1. **Voice Actors & Broadcasters**

* License voice samples to enhance AI-generated commentary.
* Use AI-driven narration tools to streamline voiceover production for sports content.

**Example Use Case: Gamer Uploading a Highlight Clip**

1. A user records a one-minute highlight from *Madden NFL*.

2. They upload the clip to ***Mr. Tai’s*** website.

3. They choose a role for the AI announcer (e.g., energetic play-by-play announcer, strategic analyst, or comedic color commentator).

4. ***Mr. Tai*** processes the video, detects key plays, and generates a narrated voiceover.

5. The user downloads the narrated video and shares it on social media.

# Definitions & Acronyms

To ensure clarity and consistency throughout the Software Requirements Specification (SRS) for ***Mr. Tai***, the following terms and acronyms are defined:

## General Terms

* **AI (Artificial Intelligence)** – The simulation of human intelligence in computers, allowing them to process information, recognize patterns, and generate responses.
* **Machine Learning (ML)** – A subset of AI that enables computers to learn and improve from experience without being explicitly programmed.
* **Natural Language Processing (NLP)** – A field of AI that enables machines to understand, interpret, and generate human language.

## Project Specific Terms

* **Mr. Tai (Multimedia Real-Time AI)** – The name of the AI-driven sports commentary project, focused on dynamically generating narration for Madden NFL gameplay footage.
* **Play-by-Play Announcer** – A commentator who provides a detailed description of the game’s actions as they happen.
* **Color Commentator** – A commentator who provides in-depth analysis, storytelling, and insights alongside the play-by-play announcer.
* **Dynamic Commentary** – AI-generated voice-overs that change based on in-game events, preventing repetitive and predictable narration.

## Technical Terms

* **Speech Synthesis (TTS - Text-to-Speech)** – The process of converting text into spoken audio using AI models.
* **Computer Vision (CV)** – A field of AI that enables computers to interpret and process visual data from images and videos.
* **Real-Time Processing** – The ability of a system to analyze and generate output instantly as events occur.
* **Post-Processing** – AI analysis that occurs after an event, such as generating narration for pre-recorded game clips.

# System Requirements & Functional Requirements

***Mr. Tai*** will use machine learning models to analyze video game footage and generate real-time or post-processed commentary. Since the development environment is based on an Apple Mac with an M2 chip, the AI processing workflow must be optimized for macOS hardware while leveraging cloud resources when necessary.

## Development Considerations

* **Local Development**
  + Apple’s **Metal Performance Shaders (MPS)** will be used to optimize PyTorch/TensorFlow models for macOS.
  + **CoreML** conversions may be explored for efficient inference on Mac hardware.
  + **FFmpeg** will handle video pre-processing tasks such as format conversion and frame extraction.
* **Cloud-Based AI Processing**
  + Since Macs do not support **CUDA (NVIDIA GPUs)**, heavy AI tasks like training and complex inference will be offloaded to cloud services.
  + Potential solutions include **Google Colab Pro, AWS EC2, or Hugging Face Inference APIs**.
  + The web-based system will allow users to upload Madden clips, which can then be processed on a cloud server before returning generated voice-overs.

## Performance Optimization

* **Latency-sensitive tasks** (e.g., real-time inference) may require cloud-based GPU acceleration.
* **Batch processing** will be used for post-game voiceover generation to reduce local computational strain.
* **Asynchronous processing** will ensure efficient handling of multiple user requests without overloading the system.

# External Interface Requirements

## User Interfaces

* **Interface Design**: The application will feature a **web-based interface** for users to upload Madden game clips, set narration roles, and receive generated voice-overs. The UI will be lightweight and optimized for both desktop and mobile access.
  + **Screen Layouts**: Simple layout with a file upload section, role selection dropdown, and an action button to process the video.
  + **Navigation**: Clear and intuitive navigation with an easy flow from uploading the video to receiving and playing back the generated commentary.
  + **User Assistance**: Contextual help for video upload (supported formats) and role explanation for narration.
* **Performance Considerations**: As the Mac M2 is a lower-powered device for AI tasks, the UI will be designed to handle only the frontend aspects of the application. AI-heavy tasks (like model inference) will occur on cloud servers to reduce strain on the local machine.

## Hardware Interfaces

* **Supported Device Types**:
  + MacBook (M2 chip) or any system running macOS for local development.
  + Remote servers with **GPU capabilities** (AWS, Google Cloud, or custom cloud providers) for cloud inference and heavy processing tasks.
  + **Mobile devices** for users accessing the web-based app. The UI will be mobile-friendly, but processing will still rely on cloud infrastructure for complex tasks.
* **Communication Protocols**:
  + **Web API calls** (RESTful APIs) for communication between the frontend and backend.
  + **File Uploads**: The web application will support file uploads of video files via a common **multipart/form-data** format.

## 

## Software Interfaces

* **Video Processing:**
  + **FFmpeg** will be used for video preprocessing tasks such as converting files, extracting frames, and resizing. It runs natively on macOS and will also be used in cloud environments.
  + **PyTorch/TensorFlow**: These libraries will load pre-trained AI models for real-time and post-processed narration. They are optimized for Apple’s **Metal Performance Shaders** (MPS), which allows inference on the M2 chip.
  + **CoreML**: Models will be converted to CoreML for better efficiency on macOS when interference occurs locally.
* **Cloud Services:**
  + For training and inference, external cloud providers (Google Cloud, AWS, or Hugging Face APIs) will be used to process large tasks and return results to the user interface.
  + The backend will be built using a **Flask** or **FastAPI** server. It will host the AI models, handle file uploads, and serve responses.

## Communication Interfaces

* **Video Processing Requests:**
  + Once a video is uploaded, the backend requests cloud-based services (Google Colab, AWS EC2) for heavy model inference tasks.
  + Responses from the cloud will be sent back as a **JSON** object, including the generated voiceover and any metadata (e.g., timestamps of narration).
* **User Notifications:** 
  + Email notifications (or optional web notifications) will inform users when their video has been processed and the results are ready for download or playback.
  + The backend may use **SMTP** or **API-based services** (e.g., SendGrid or Twilio) to send these notifications.

# Non-functional Requirements (NFRs)

## Security

* **Data Protection**: Implement basic encryption for video file uploads and downloads (e.g., using **HTTPS**). At this stage, we may not need to implement advanced encryption methods for rest, but we should ensure that any sensitive data is transferred securely.
* **Authentication**: If authentication is needed, use a simple authentication method such as **email and password login** or **OAuth** for ease of integration with minimal setup. **(Start with OAuth)**
* **User Privacy**: We can focus on ensuring that user data (video files) is deleted after processing, but avoid diving into complex compliance standards (e.g., GDPR/CCPA) unless it’s critical for our use case.

## Capacity

* **Video Storage**: For the scope of our project, temporary storage using **cloud storage services** (e.g., **AWS S3** or **Google Cloud Storage**) should suffice. For faster processing, limit the video clip size to 500MB or less.
* **Scalability Considerations**: Rather than building a full auto-scaling system, we can focus on using cloud resources effectively, but with manual scaling during testing and early use.

## Compatibility

* **MacBook (M2 chip)**: Since we are using a Mac M2 with 8GB of RAM, ensure that our local environment and any development work (e.g., frontend or AI model testing) runs smoothly on this hardware. We must optimize any local processing to avoid memory overload.
* **Cloud Infrastructure**: Focus on integrating with **Google Cloud** or **AWS** for cloud processing, but don’t worry about handling cross-platform compatibility extensively for our project scope. We can focus on a specific backend (e.g., **Google Cloud AI** for voice processing) and test only on macOS or a specific OS during development.

## Reliability and Availability

* **Uptime**: Aim for high availability, but for a prototype, consider a **99% uptime** goal. Also, consider maintenance and testing windows during which the system may be temporarily unavailable.
* **Redundancy**: We can manually manage redundancy (e.g., using a backup system in cloud storage or local backups), but don’t need to over-engineer this aspect for a prototype.

## Scalability

* **User Load**: The system will initially handle a small number of users, so we don’t need to focus heavily on scalability features. We can design our system to scale on demand using cloud services if needed, but we should avoid building an auto-scaling solution at this stage.
* **Processing Power**: Focus on optimizing your AI inference pipeline for the most common video file types and sizes. Focus on using **Google Cloud AI** or **AWS Lambda** to offload heavy processing to the cloud instead of running inference on our local machine.

## Maintainability

* **CI/CD**: While a full-fledged CI/CD pipeline might be overkill, you can set up a simple **GitHub Actions** workflow to automate testing and deployment to your cloud service.
* **Code Quality**: Maintain good coding practices (e.g., unit tests for key parts) but avoid implementing a fully-fledged testing suite unless it’s essential. Prioritize writing clean, understandable code that can be easily modified as we develop the project.

## 

## Usability

* **User-Friendly UI**: The UI should be simple and intuitive, but we don’t need advanced design elements. Focus on basic functionality like:
  + Upload video
  + Select the role for narration
  + Play audio narration
* **Feedback Mechanisms**: Use basic notifications to inform the user when their video is being processed and when it’s ready. Notifications can be simple in the UI rather than relying on email or text alerts.
* **Mobile-Friendly Design**: If time allows, make the web app responsive, but prioritize desktop compatibility initially and consider adding mobile optimization later.

## Performance

* **Inference Speed**: Aim for inference times under **1 minute** for typical video clips. However, more complex or longer clips might require additional processing time (e.g., 2-3 minutes). Use cloud computing resources to handle processing efficiently.
* **Video Upload Speed**: Focus on allowing video uploads for clips under **500MB**, with clear progress bars showing upload status. Optimize for an average upload speed of **1-2 Mbps**, but don’t stress if upload performance is not perfectly optimized.

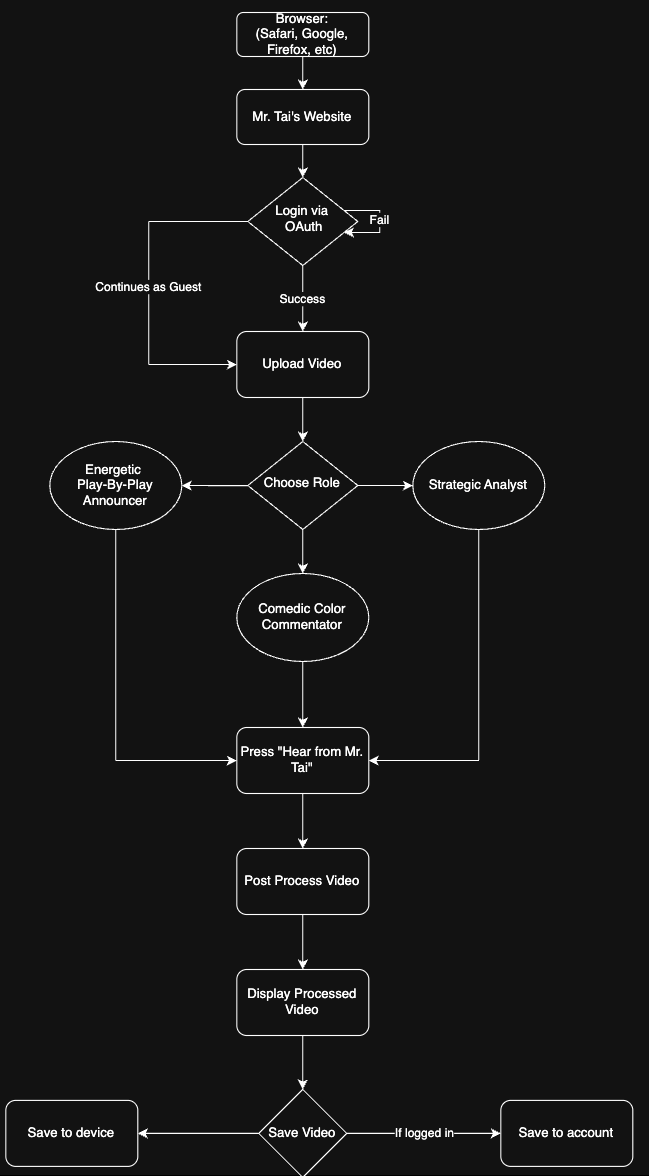
# Specific Development Tools

* **Python with TensorFlow or PyTorch** for AI modeling and inference
* **CoreML** for optimized AI inference on macOS
* **AWS EC2** or **Google Cloud AI** for inference and storage using S3 or **Google Cloud Storage**
* **Flask** or **FastAPI** for the backend
* **REST API** endpoints that will be used for video uploads and receiving AI results
* **React** or **Next.js** for the web interface
* **GitHub Actions** for CI/CD
* **Hugging Face** or **TensorFlow Model Garden** for pre-trained models

## Start With:

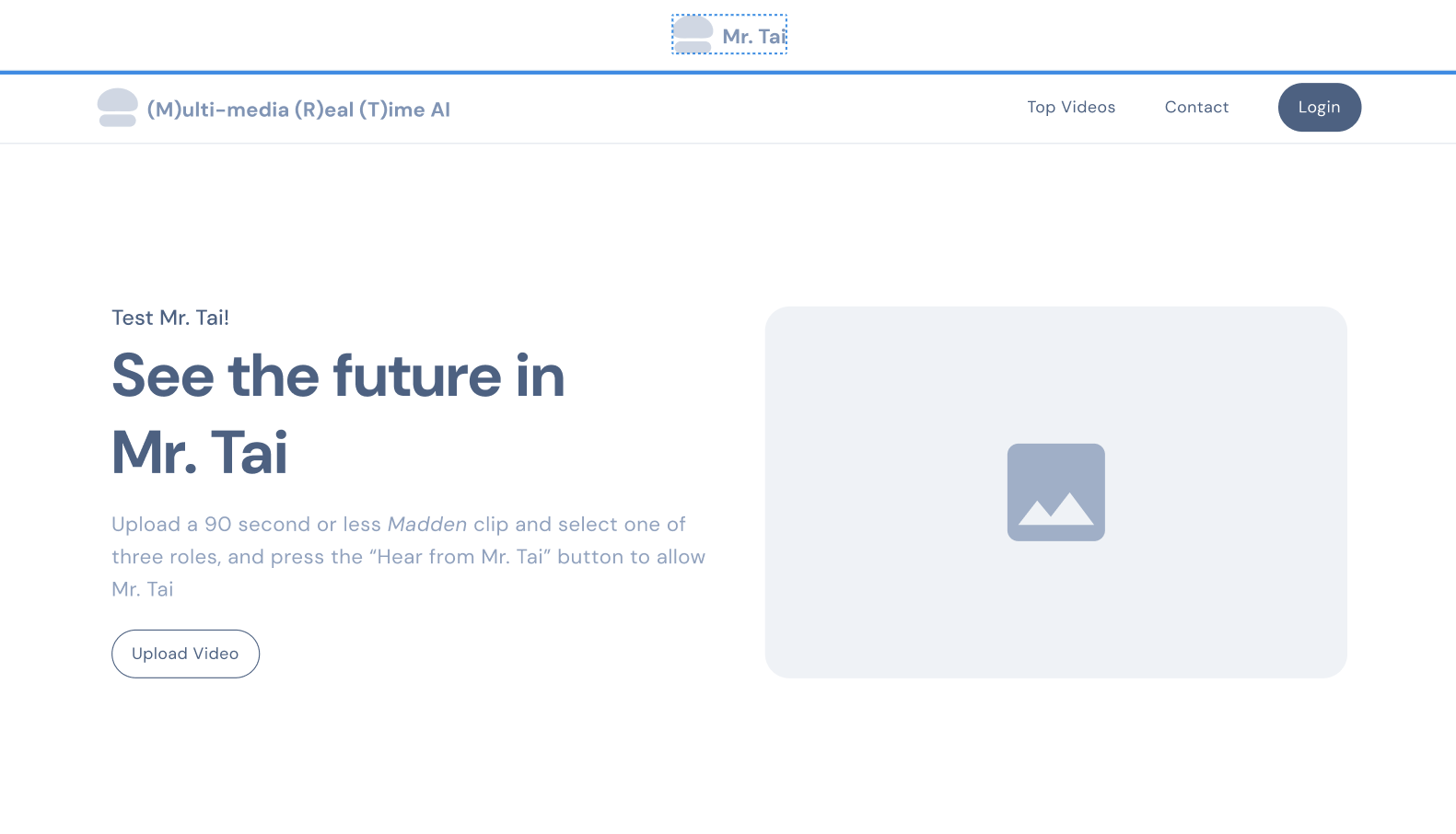
* **Python 3.11+** for Programming Language
* **PyTorch** for AI Framework
* **CoreML via PyTorch → CoreML converter** for Mac Inference Optimization
* **Google Cloud AI + Google Cloud Storage** for Cloud Inference & Storage
* **FastAPI** for Backend Framework
* **REST** for API Protocol
* **React** for Frontend Framework
* **GitHub Actions** for CI/CD
* **Hugging Face** for Pre-trained Model Source

# Flow Chart

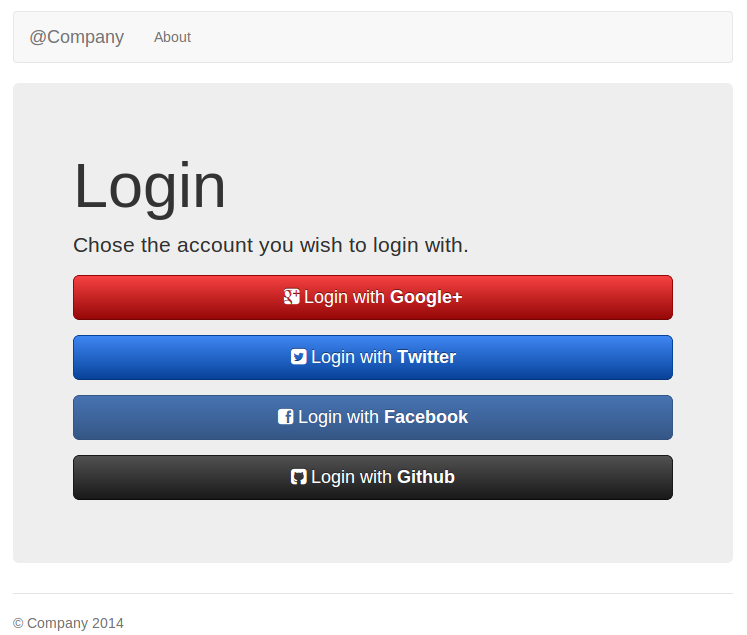


# Wireframe

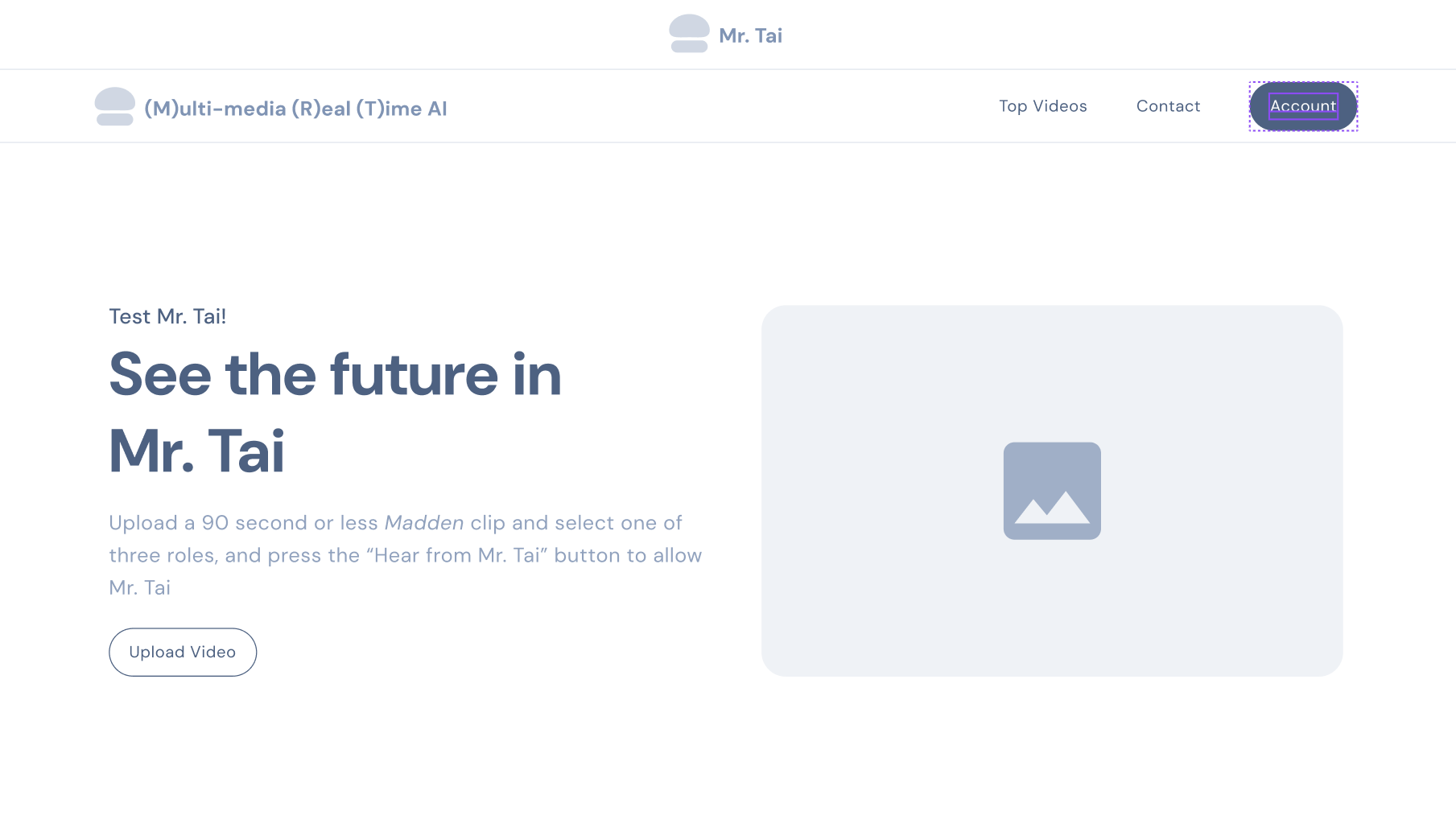
Home Page (Not Logged In)



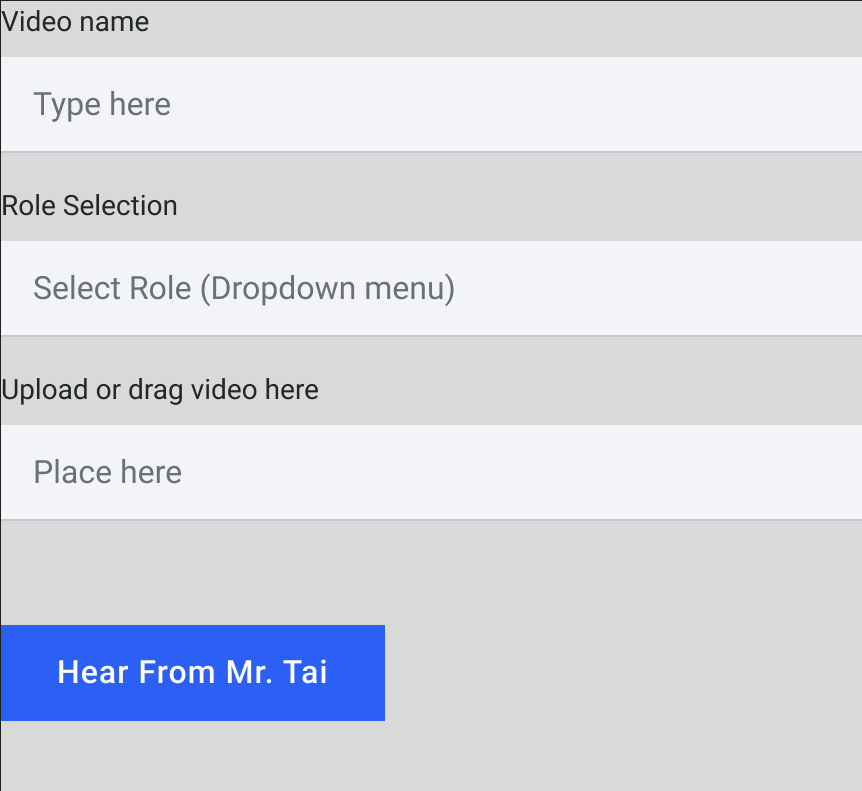
OAuth Login



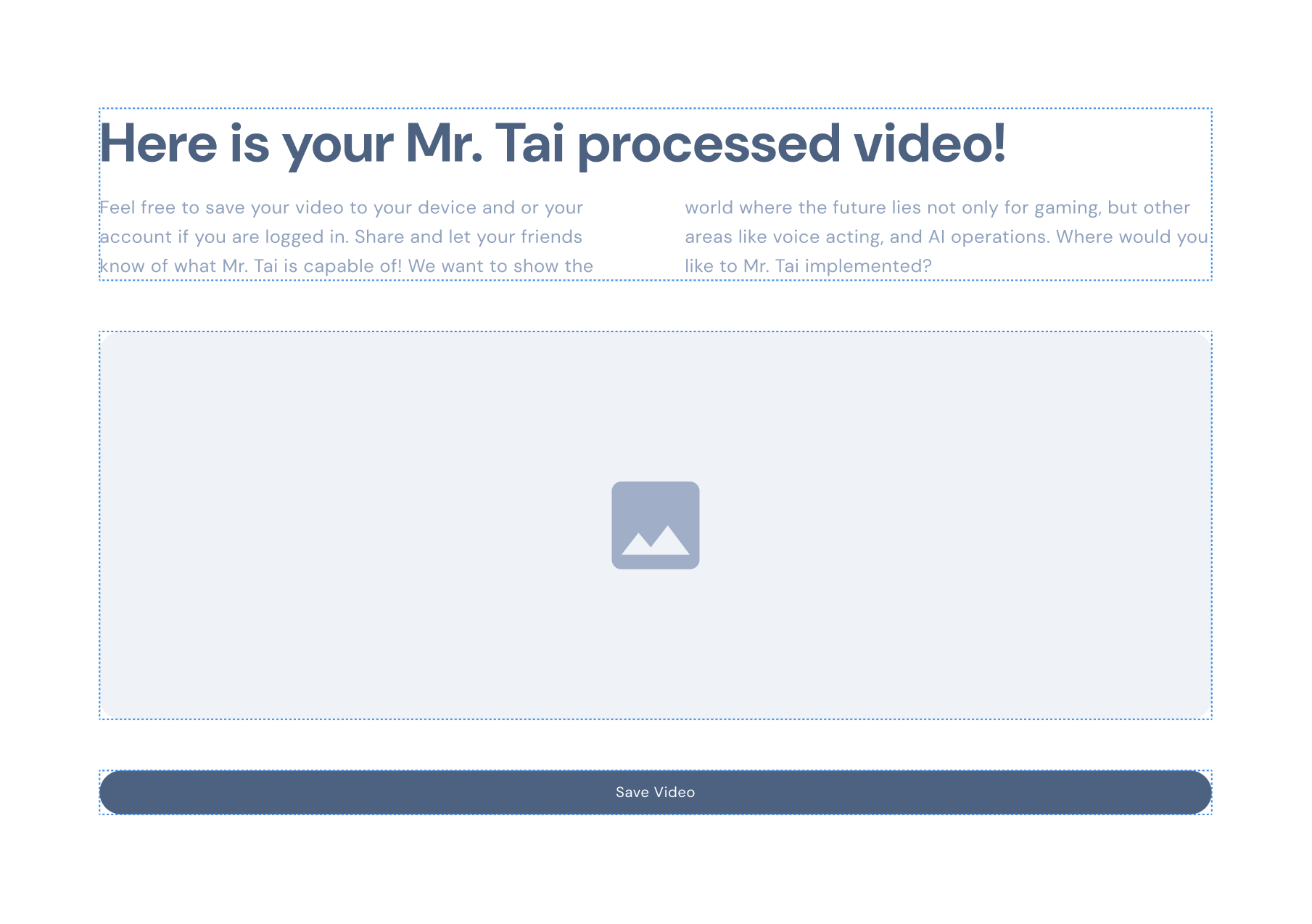
Home Page (Logged In)



Upload Video Page



Display processed Video



# Milestone Timeline

* Month 1– Foundation & Setup
* Month 2 – MVP Core Features
* Month 3 – UI Polish, Testing & Account System
* Month 4 – Final Polish, Deployment, Demo