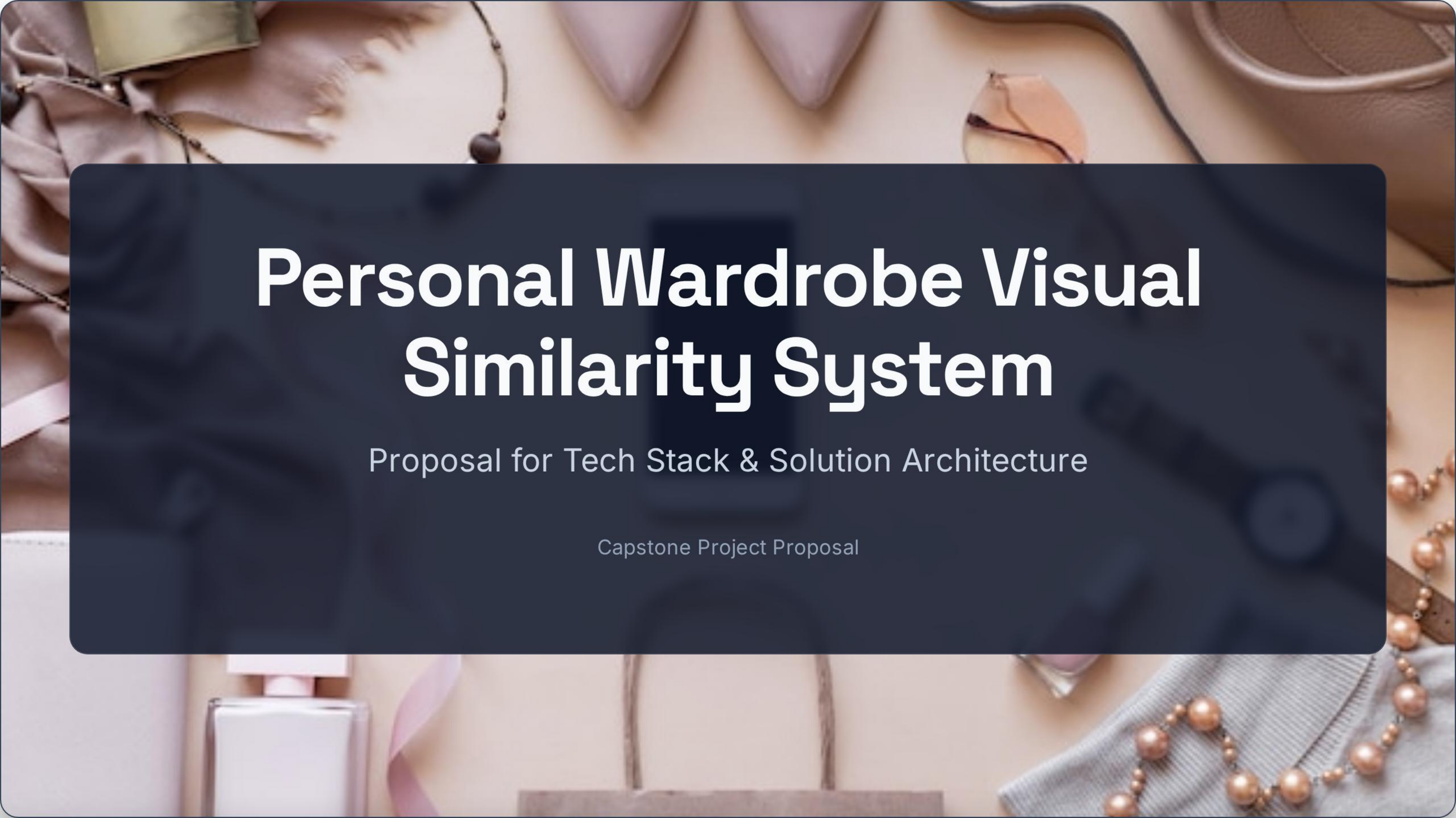


# Personal Wardrobe Visual Similarity System

Proposal for Tech Stack & Solution Architecture

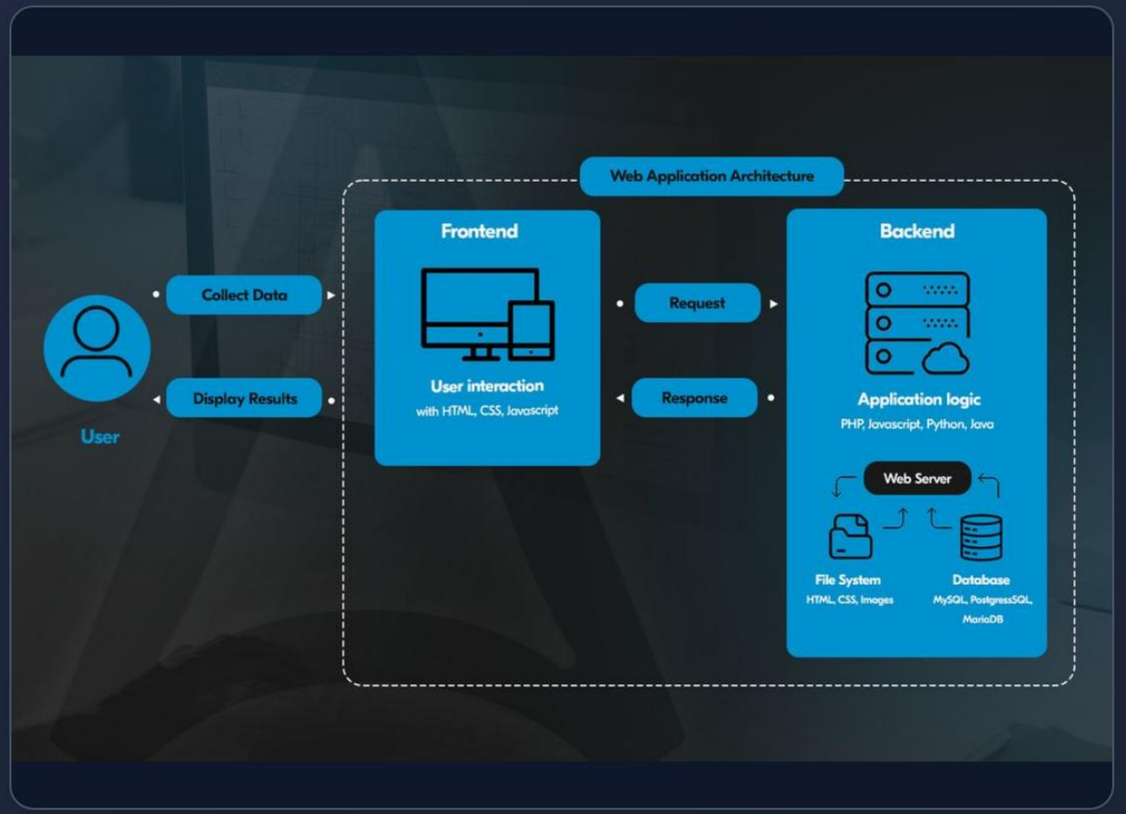
Capstone Project Proposal

# System Architecture Overview

## Dual-Interface Strategy

The system adopts a scalable, dual-interface architecture designed for both rapid prototyping and real-world simulation.

- **Unified Backend:** Python-based API serving both web and mobile clients.
- **Web Application:** Streamlit-based interface for instructor review and deep model inspection.
- **iOS Application:** Native SwiftUI client for simulating realistic consumer usage.



# Frontend Layer: Web Application


## Framework: Streamlit

Selected for its ability to turn data scripts into shareable web apps in minutes, focusing on pure Python development.

- **Rapid Prototyping:** Allows for fast iteration of the UI without complex frontend code.
- **Model Inspection:** Visualizes similarity scores and embeddings directly, ideal for evaluation.
- **Interactive Demo:** Provides a transparent interface for capstone presentation.

# Frontend Layer: iOS Application

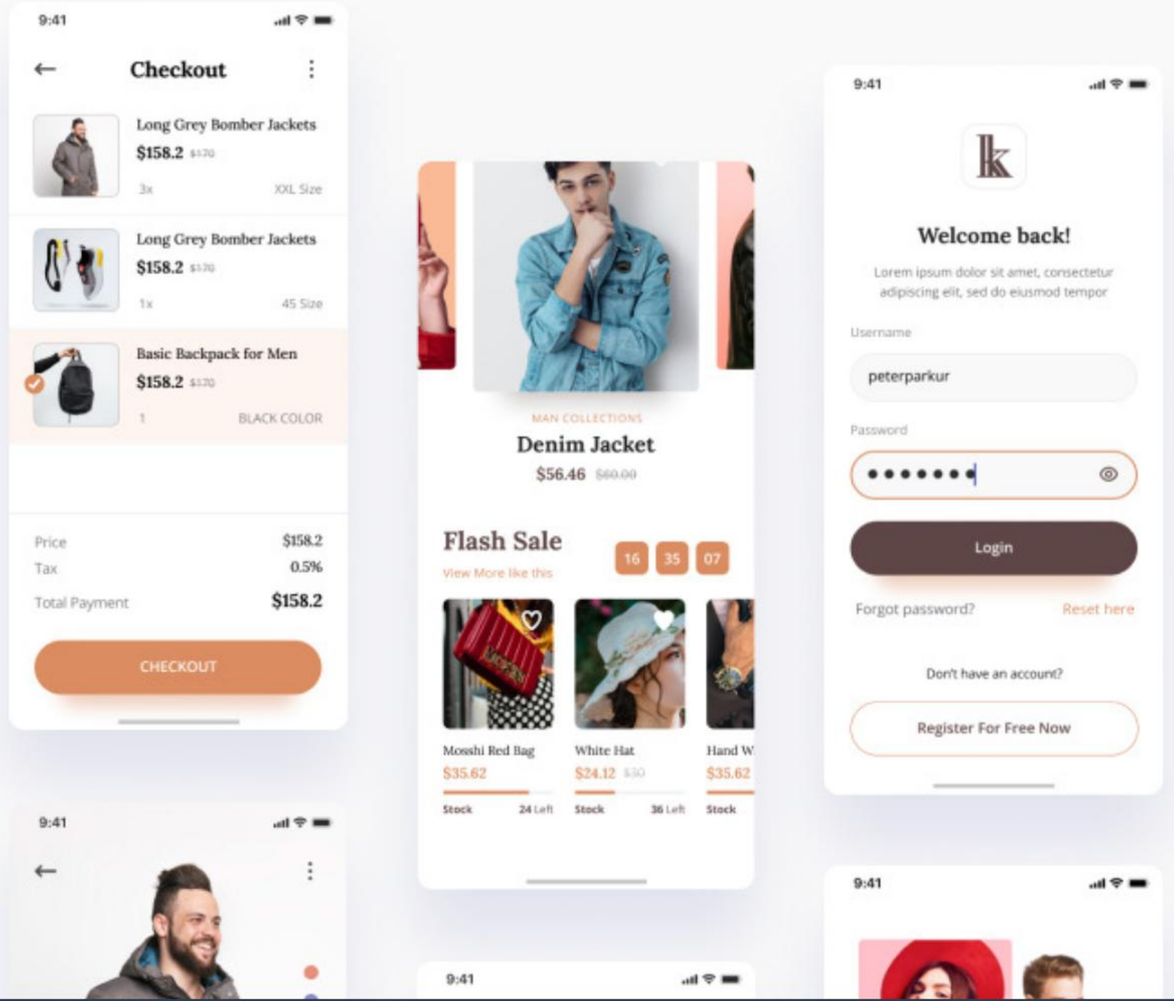
## Language: Swift / SwiftUI

A native mobile client designed to demonstrate the system's viability in a consumer context.

**Key Features:**

- **Mobile-First Experience:** Leverages native UI components for a smooth user experience.
- **Camera Integration:** Allows users to capture clothing items directly from their wardrobe.
- **Real-World Simulation:** Communicates with the backend API just like a production app.

## Designed for iOS, iPhone & Mobile Devices



# Backend & API Layer

## FastAPI Framework

A modern, fast (high-performance) web framework for building APIs with Python 3.7+.

- **Asynchronous:** Native support for async/await, critical for handling concurrent image uploads.
- **Auto-Documentation:** Generates interactive Swagger UI for easy testing and debugging.
- **Type Safety:** Leverages Pydantic for robust data validation.

## Core API Endpoints

RESTful architecture designed for stateless communication.

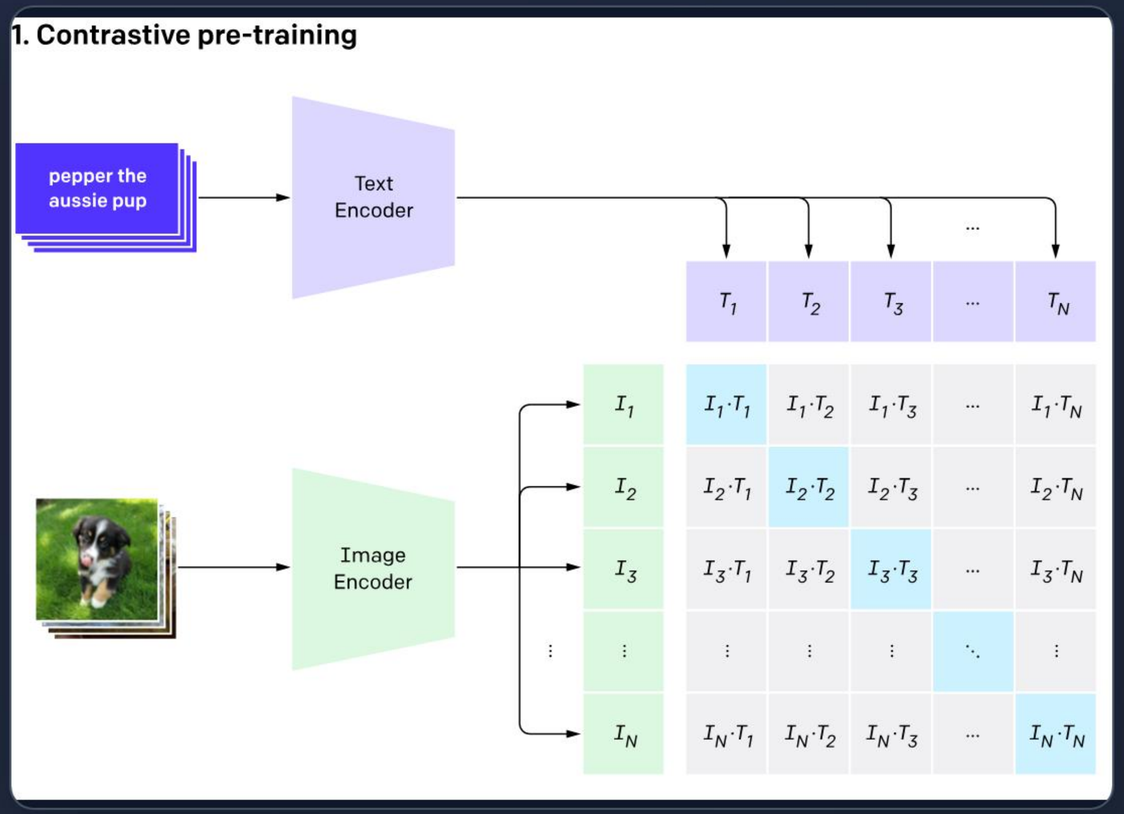
```
POST /upload
Accepts image -> Returns embedding

GET /similar
Query ID -> Returns top-k matches
```

# Machine Learning: Visual Embeddings

## Model: CLIP (OpenAI)

**Contrastive Language–Image Pretraining** is chosen over traditional CNNs for its semantic understanding capabilities.

- **Dual-Encoder:** Maps images and text to a shared latent space.
- **Semantic Search:** Enables finding items that are semantically similar, not just visually identical (e.g., "red floral dress").
- **Similarity Search:** Uses Cosine Similarity to measure distance between garment vectors.



# Data Storage Architecture

## Metadata

**SQLite / Cloud SQL**

Stores structured data: File paths, User IDs, Upload timestamps, and Category tags.

## Embeddings

**Faiss / ChromaDB**

Specialized vector database for high-dimensional vectors, enabling sub-millisecond similarity search.

## Image Assets

**Object Storage**

Local file system (Dev) or S3-compatible storage (Prod). Decoupled from the database for performance.

# Deployment Strategy

## Backend API

**Platform: Render / Fly.io**

Containerized deployment (Docker) of the FastAPI application. Supports auto-scaling and HTTPS.

## Web Frontend

**Platform: Streamlit Cloud**

Direct integration with GitHub for continuous deployment. Instant updates upon code push.

## iOS Client

**Platform: TestFlight**

Distribution to limited internal testers. Connects to the cloud-hosted API backend.

# Rationale & Alignment

- **End-to-End ML Deployment:**
  Demonstrates the full lifecycle from model selection (CLIP) to API serving and client consumption.
- **Interactive Demonstration:**
  Streamlit provides immediate visual feedback, crucial for validating visual similarity results during the demo.
- **Real-World Relevance:**
  The inclusion of an iOS app bridges the gap between academic theory and consumer-facing product design.
- **Scalable Foundation:**
  Using a Vector DB and Async API ensures the system can handle growth, satisfying "Scalability" requirements.

# Implementation Timeline

## Phase 1

**Setup & Backend**

FastAPI Setup, DB Init

## Phase 2

**ML Integration**

CLIP Model, Vector Search

## Phase 3

**Frontend Dev**

Streamlit & iOS UI

## Phase 4

**Deployment**

Cloud Hosting, Testing

# Q & A

Thank you for your attention.

Proposal for Personal Wardrobe Visual Similarity System

# Image Sources

https://acropolium.com/img/articles/modern-web-app-architecture/img02.jpg

Source: acropolium.com

---

https://prescienceds.com/wp-content/uploads/2025/05/Streamlit-Dashboards-scaled.jpg

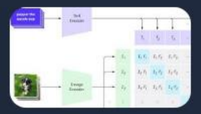
Source: prescienceds.com

---

https://yi-files.yellowimages.com/products/965000/965688/1628024-full.jpg

Source: yellowimages.com

---



https://images.ctfassets.net/kftzwdyauwt9/fbc4f633-9ad4-4dc2-3809c22df5e0/0bd2d5abf90d052731538613e4a42668/overview-a.svg

Source: openai.com

---

https://img.freepik.com/premium-photo/mock-up-blank-empty-smart-phone-screen-beige-fashion-women-stylish-accessories-outfit-glamour-set-gifts-sale-flat-lay-background-online-app-shopping-mobile-ecommerce-technology-concept-top-view_203461-747.jpg

Source: www.freepik.com