Milestone 1: Team Formation & Architecture (Design Only)

Project Title: Deepfake & Synthetic Media Detector
Team Name: WCU RAMS
Team Members: Kevin Ha, Isreal Adegbie, Thomas Burke,
Mo Abdularazzak, Alym Rejepov
Date: September 9, 2025

1 Overview

The **Deepfake & Synthetic Media Detector** is a Kubernetes-based pipeline designed to help users detect AI-generated or manipulated media, including images, videos, audio, and text. Users upload media, which is processed through a multi-pod pipeline to classify content as "Real" or "Fake," providing confidence scores and logging results. This project emphasizes infrastructure as code, multi-pod orchestration, and persistence handling in Kubernetes.

Problem

Misinformation from AI-generated or manipulated media is growing rapidly; users have no simple way to verify authenticity across text, audio, image, and video.

Solution

Users upload text, videos, images, or audio. The pipeline analyzes authenticity using anomaly detection and compares it against trusted sources, flagging suspected deepfakes.

2 Use Cases & Workflow

User Stories:

- As a user, I would upload my file (text, audio, image, or video) and receive a report within seconds stating if it is a deepfake or not.
- As a journalist, I would want to double-check my story's origins and its authenticity.
- As an older person or someone who is not technically advanced, I would want my phone calls, emails, texts, etc., to be checked to see if they are robocalls or scams.
- As a teacher, I would want to upload my students' assignments and check if they are AI-generated or not.
- As law enforcement, I would want to validate video/audio evidence or alibi authenticity before trial.

3 System Diagrams

Data-Flow Diagram

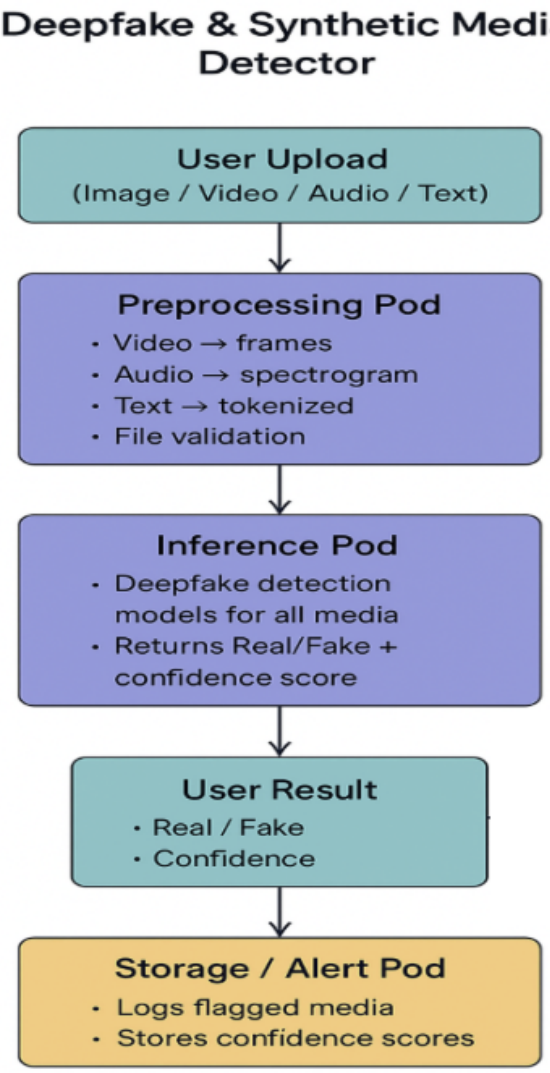

Figure 1: Data Flow Diagram of Media Processing Pipeline

Kubernetes System Architecture

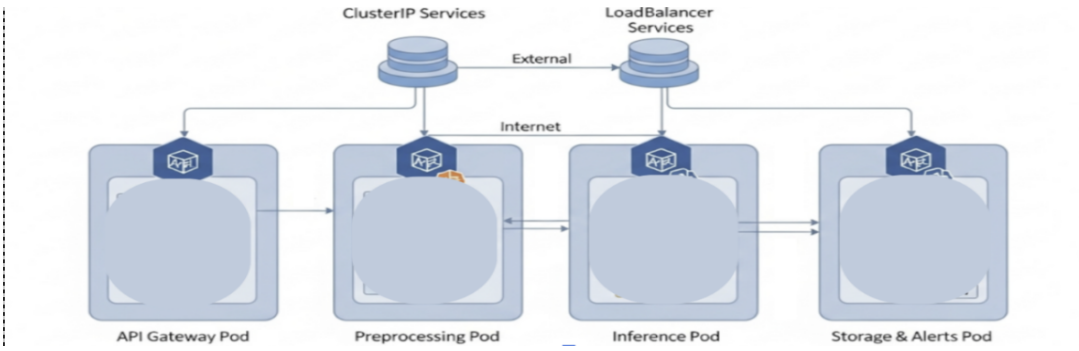

Figure 2: Kubernetes System Architecture for Deepfake Detection

4 Kubernetes Architecture Design

Pod Structure:

- **API Gateway (Single Pod)**: Handles all incoming requests from users. Kept separate to isolate the internet-facing component. Exposed externally via a LoadBalancer/Ingress with HTTPS.
- **Preprocessing (Single Pod)**: Cleans and normalizes uploaded media (images, audio, video, text). Sandboxes raw, untrusted user data.
- **Inference (Multi-container Pod)**: Runs the detection model with a metrics sidecar. Scales independently under load.
- **Storage & Alerts (Single Pod + Persistent Volume)**: Stores results, logs, user credentials, and flagged media. Encryption at rest enabled.
- **User Management (Single Pod)**: Handles sign-up/login; stores hashed credentials in Kubernetes Secrets.

Service Types:

- **ClusterIP**: Internal communication between pods.
- **LoadBalancer**: External secure access for users.

5 API Contracts

POST /auth/register

```
Input:  { "username": "alice", "password": "mypassword123" }
Output: { "status": "success", "message": "Account created" }
```

POST /auth/login

```
Input:  { "username": "alice", "password": "mypassword123" }
Output: { "token": "<JWT-token>", "expires": "3600s" }
```

POST /upload

```
Auth: Bearer JWT required
Input:  multipart form-data { file: <media>, user_id: <uuid> }
Output: { "job_id": "12345", "status": "processing" }
```

GET /result/job$_id$

```
Auth: Bearer JWT required
Output: { "status": "complete", "verdict": "fake", "confidence": 0.94 }
```

6 Authentication & Authorization

Authentication is required before any media upload or result retrieval. Users must first create an account with a username and password, which are hashed and stored in Kubernetes Secrets.

- **Normal Users** – can upload media and view their own results.
- **Admins** – can view all flagged content, system metrics, and manage users.

7 Risks

- Model size may cause slow startup.
- Large input files may exceed memory limits.
- Some file types may be unsupported.

8 Test Matrix

Scenario	Input	Expected Output
Happy Path	Valid image	Prediction + confidence score
Large File	Video >100MB	Error: "File too large"
Invalid Format	.exe file	Error: "Unsupported File Type"
Adversarial Case	Corrupted audio	Report: "Unable to process" + logs

9 Team Charter

Roles

- Lead / PM – Kevin
- DevOps – Isreal
- Backend / Interface – Thomas (frontend/interface), Mo and Alym (backend)
- Preprocessing – Thomas
- QA Docs – Kevin

Meeting Cadence

- Weekly on Wednesday at 1:00 PM (in person)
- Quick check-ins as needed (virtual)

Definition of Done

- Inference pod scales under load.
- Logs stored successfully.
- API passes happy-path and error-path tests.
- All services deployed in Kubernetes cluster.
- Program works as planned.
- Program passes all test cases.