Final Technical Report

Project Title: Verus – AI/ML Media Detector

Team Name: WCU RAMS

Team Members:

Kevin Ha
Isreal Adegbie
Thomas Burke
Mo Abdularazzak

Date: December 7, 2025

Institution: West Chester University

Contents

1 Executive Overview

Verus is a cloud-native, containerized microservices platform designed to detect AI-generated or manipulated content across images, text, and video. It leverages Flask APIs, GPU-enabled inference pods, Kubernetes orchestration, and dynamic routing for real-time, scalable analysis.

1.1 Purpose

The system addresses the growing need for automated verification of digital media, mitigating misinformation risks for journalists, educators, law enforcement, and the public.

1.2 Key Features

- Modular, containerized microservices for each media type, allowing selective scaling.
- Flask-based API gateway and routing service for unified user access.
- Pretrained AI/ML inference models for images, text, and video classification.
- Persistence layer for optional user storage of results and metadata.
- Kubernetes-managed self-healing, logging, and operational monitoring.

1.3 Benefits

- Reduces manual verification effort for high-volume media workflows.
- Enables secure and private result storage for authenticated users.
- Provides a foundation for future scaling, automated retraining, and model expansion.
- Extensible architecture supports integration of new AI models and services.

2 Use Cases & Workflow

2.1 User Stories

- Users can upload media and receive an authenticity verdict with confidence scores.
- Journalists can validate digital content before publication.
- Educators can detect AI-generated assignments and submissions.
- Law enforcement can verify digital evidence.
- Users can view previously analyzed media stored under their account.

2.2 System Diagrams

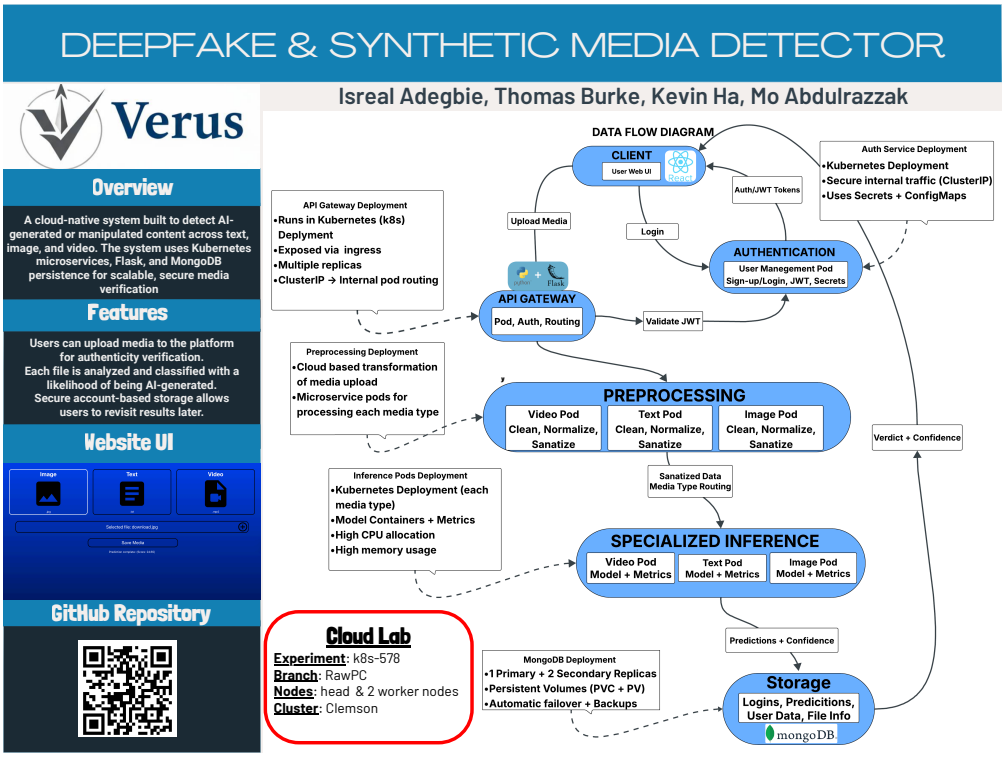

Figure 1: Data Flow: User uploads routed through Process Service to Predict Pods and returned.

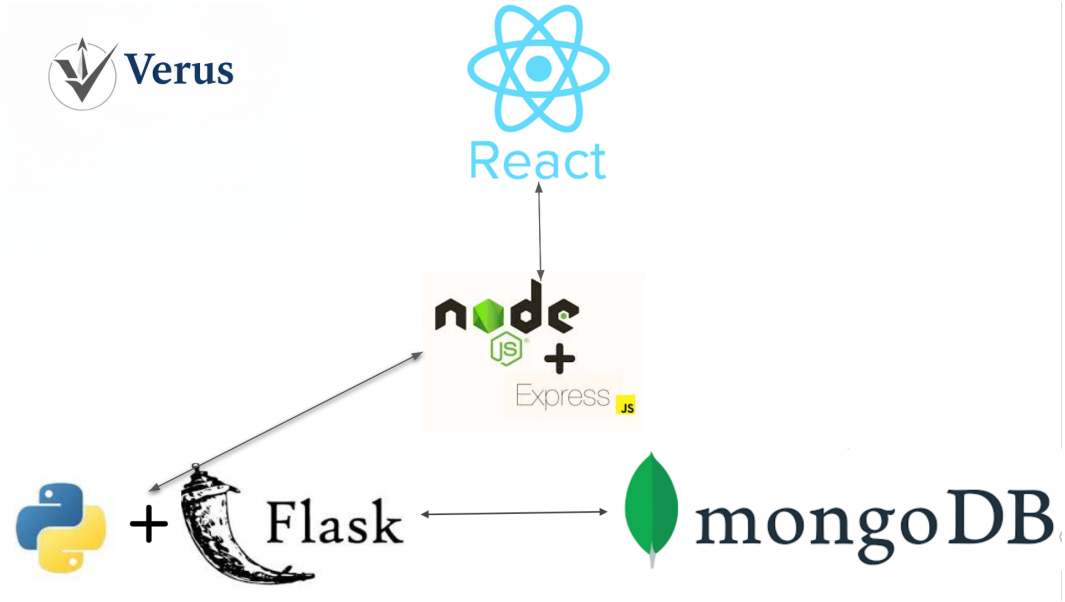

Figure 2: Kubernetes Architecture: Pod grouping, services, routing, and persistence.

3 Architecture Recap

3.1 Microservices & Pods

- **Flask API / Process Service:** Handles all uploads, routes requests to the appropriate Predict Pod (Image, Text, Video), and exposes API endpoints for authentication, upload, and result retrieval.
- **Predict Pods (Image, Text, Video):** Host respective AI models; include optional sidecars for logging, metrics, and performance monitoring.
- **Persistence Pod:** MongoDB StatefulSet for secure result storage and metadata; leverages PVCs for binary files.
- **User Management Pod:** JWT authentication, account creation, and login verification.

3.2 Service Types

- **ClusterIP:** Secure internal communication between pods.
- **Ingress:** Public exposure of Flask API for external users.

3.3 Data Flow

1. Users upload media through the Flask API endpoint.
2. Process Service validates the upload and forwards to the correct Predict Pod.
3. Predict Pod executes AI/ML inference using pretrained models.
4. Results returned to Process Service and optionally persisted in MongoDB.
5. JSON verdict containing label and confidence delivered to the user.

3.4 Code Snippet: Image Classification

```
from flask import Flask, request, jsonify
from transformers import AutoImageProcessor, SiglipForImageClassification
import torch
from PIL import Image

app = Flask(__name__)
processor = AutoImageProcessor.from_pretrained("prithivMLmods/deepfake-detector-model-v1")
model = SiglipForImageClassification.from_pretrained("prithivMLmods/deepfake-detector-model-v1")

def classify_image(img):
    image = Image.open(img).convert("RGB")
    inputs = processor(images=image, return_tensors="pt")
    with torch.no_grad():
        outputs = model(**inputs)
        probs = torch.nn.functional.softmax(outputs.logits, dim=1).squeeze().tolist()
    labels = {"0":"Fake", "1":"Real"}
    result = {labels[str(i)]: round(probs[i]*100,2) for i in range(len(probs))}
    return result
```

4 Scaling and Self-Healing

4.1 Scaling

Currently, scaling is implemented manually by increasing replicas in Deployment manifests. For example, multiple Flask API or Predict Pod replicas can handle concurrent uploads. While automatic Horizontal Pod Autoscaling (HPA) based on CPU/memory is **not yet implemented**, the system is designed to be **HPA-ready** for future load-based scaling.

4.2 Self-Healing

- Liveness and readiness probes automatically detect unhealthy pods.
- Kubernetes restarts failed pods to maintain service availability.
- Sidecars monitor pod performance and generate logs for operational awareness.

5 Persistence Design & Verification

5.1 Data Policy

- Users may store inference results securely under account-specific quotas.
- MongoDB document model handles heterogeneous AI inference outputs efficiently.
- Binary files stored on PVCs for durability.
- Access restricted to authenticated users only.

5.2 Backup & Verification

- MongoDB deployed as a 3-node replica set for failover.
- PVCs provisioned with dynamic storage classes for resiliency.
- Verification performed via test uploads, cluster restarts, and pod failover simulations.

6 Known Limitations

- Model accuracy may vary for unusual or adversarial inputs.
- Process Service introduces minor network latency.
- Persistent storage is limited by user quota.
- Routing depends on correct Base URL configuration for mirror/proxy service.
- No automated retraining or active learning pipeline is currently implemented.

7 Future Work

- Integrate additional AI/ML models to increase accuracy and coverage.
- Implement automatic retraining and model versioning.
- Develop operational dashboards for metrics, logs, and alerts.
- Introduce HPA for dynamic scaling under real-world loads.
- Support multi-cloud deployments and geo-redundancy.

8 Test Matrix

Table 1: Functional and Stress Test Matrix

Test Case	Input	Expected Outcome
Valid Image	2MB JPEG	Prediction + confidence; stored if chosen.
Large File	200MB video	"File too large" error.
Invalid Format	.exe	"Unsupported file type".
Corrupt Media	Damaged audio	"Unable to process".
Concurrent Uploads	50 simultaneous uploads	Manual replica scaling; no loss.
Unauthorized Access	Invalid JWT	"Unauthorized".
Persistence Test	Restart cluster	Data persists in MongoDB + PVC.
Pod Failure	Delete API pod	Auto-restart; service resumes.
DB Failover	Restart primary Mongo pod	Replica promotes; data intact.

9 Team Charter

9.1 Roles

- Lead / PM: Kevin Ha – Project oversight, documentation.
- DevOps: Israel Adegbie – Kubernetes, Flask API, authentication, and monitoring.
- Backend/API: Mo Abdularazzak – Storage, inference integration, verification.
- Frontend/Preprocessing: Thomas Burke – React UI, preprocessing scripts, and AI/ML integration.

9.2 Definition of Done

- Secure API with JWT authentication.
- Predict pods scale manually; HPA-ready.
- Persistence verified under failover scenarios.
- Functional, edge-case, and stress tests passed.
- Logging, metrics, and monitoring implemented.

References

1. Hugging Face. `https://huggingface.co/models`
2. Flask Documentation. `https://flask.palletsprojects.com/`
3. Gunicorn Documentation. `https://gunicorn.org/`
4. PyTorch Documentation. `https://pytorch.org/docs/stable/index.html`