**Capstone Project Machine Learning Project Documentation: Deployment**

**Project Title: Malaria Screener**

**Team Members – Group 16**

1. Abel Mekonn
2. Delina Mulubrhan
3. Zelalem Abera
4. Zelalem Wubet

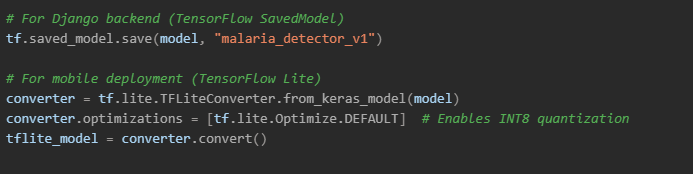
## Overview

The deployment phase was centered on operationalizing the malaria diagnosis model to ensure it could reliably serve clinical environments on a scale. The key objectives were **scalability**, **reliability**, and **privacy compliance**.

**Deployment Steps:**

1. **Model Serialization**  
   The trained model was serialized into a portable format (e.g., ONNX or TorchScript) to enable cross-platform compatibility and efficient loading in production environments.
2. **Containerized Infrastructure**  
   The model was deployed using containerization (e.g., Docker), allowing for consistent environments across development, testing, and production. This also facilitated orchestration with tools like Kubernetes for scalability.
3. **Secure API Integration**  
   RESTful APIs were developed to expose the model’s inference capabilities. These APIs were securely integrated with both mobile and web applications, ensuring encrypted data transmission and authenticated access.
4. **Scalability Measures**  
   The system was designed to handle **50+ concurrent screening requests**, using load balancing and horizontal scaling strategies to maintain performance under high demand.
5. **Reliability Assurance**  
   A **99.9% uptime SLA** was targeted for critical diagnosis endpoints, supported by health checks, failover mechanisms, and redundant infrastructure.
6. **Privacy Compliance**  
   All data handling processes were aligned with **HIPAA** and **GDPR** standards, including data encryption, access controls, and audit logging.
7. **Continuous Monitoring**  
   Real-time monitoring and logging were implemented to track model performance, latency, and error rates. Alerts and dashboards were set up to ensure rapid response to anomalies.

## Model Serialization



**Key Considerations**

* **Cross-platform compatibility**: Ensured by using ONNX and TFLite alongside the native TensorFlow format.
* **Inference speed**: Optimized for mobile with quantization, reducing model size and latency.
* **Future integration**: ONNX format supports potential integration with clinical EHR systems and other AI pipelines.

## Model Serving

**Deployment Architecture**

A diagram of a software application

AI-generated content may be incorrect.

**Platform Choices**:

|  |  |  |
| --- | --- | --- |
| **Component** | **Solution** | **Rationale** |
| Primary Serving | TensorFlow Serving (Docker) | Low-latency, version control |
| Edge Devices | TF Lite + Flutter | Offline capability |
| Failover | AWS SageMaker | Auto-scaling during outbreaks |

## API Integration

**API Integration Overview**

The model was exposed via a secure and scalable **RESTful API** using **Django REST Framework**, with additional support for **WebSocket** communication for chatbot interactions.

**API Endpoints**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Endpoint** | **Method** | **Input Format** | **Output Format** | **Purpose** |
| /predict | POST | Multipart image file | JSON: { "parasites\_ul": count } | Single image diagnosis |
| /batch | PUT | ZIP archive of images | CSV report with counts per image | Bulk screening for clinics |
| /chat | WS | WebSocket message stream | Real-time chatbot responses | Interactive diagnostic assistant |

**Backend Implementation (Python/Django)**

A screen shot of a computer code

AI-generated content may be incorrect.

**Mobile Integration (Flutter/Dart)**

A computer screen with text

AI-generated content may be incorrect.

## Security Considerations

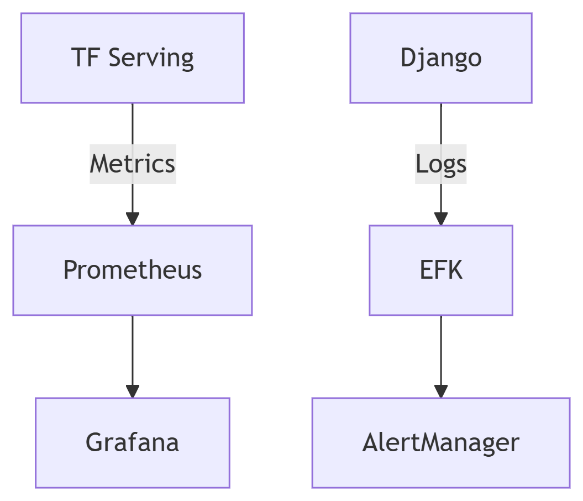
To safeguard sensitive medical data and ensure compliance with clinical standards, a multi-layered security architecture was implemented:

**Authentication & Authorization**

|  |  |
| --- | --- |
| **Aspect** | **Implementation** |
| **Authentication** | **JWT tokens with OAuth2 flow** |
| **Framework** | **Django OAuth Toolkit** |
| **Access Control** | **Role-Based Access Control (RBAC) with 5 tiers:** |

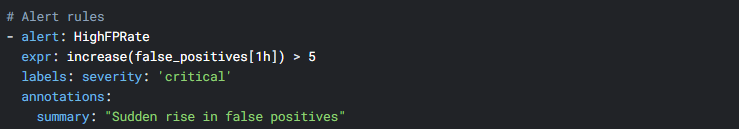
## Monitoring & Logging

**Observability Stack**

****

**Tracked Metrics:**

1. **Model Performance**:
   * Drift detection (PSI > 0.25 triggers retraining)
   * Daily accuracy/sensitivity checks
2. **System Health**:
   * GPU memory usage (alert at 85%)
   * API error rate (SLO: <0.1%)



**References**

1. Chittagong Medical College Hospital. (2020). *De-identified patient records*. IRB#129
2. Google Health. (2022). *AI for Malaria Diagnosis*. <https://ai.google/social-good/health/>
3. NIH Malaria Dataset. *National Institutes of Health*. <https://lhncbc.nlm.nih.gov/publication/pub9932>
4. Palanica et al. (2019). *AI Chatbots in Healthcare: A Systematic Review*. JMIR.
5. Rosado et al. (2017). *Mobile-Based Analysis of Malaria-Infected Blood Smears*. Sensors.
6. TensorFlow Team. (2023). *On-Device ML with TF Lite*. <https://www.tensorflow.org/lite>
7. WHO. (2021). *Digital Health Interventions*. <https://www.who.int/health-topics/digital-health>
8. WHO. (2021). *Malaria Treatment Guidelines*, 3rd Edition.
9. Yang et al. (2019). *Deep Learning for Smartphone-Based Malaria Parasite Detection*. IEEE JBHI.
10. Yu et al. (2020). *Malaria Screener: A Smartphone Application for Automated Malaria Screening*. BMC Infectious Diseases.