**Title: WorkWise “Where Talent Meets Opportunity”**

1. Maka K. Armah

2. George W. Monibah  
3. Sekou S. Kamara

**Machine Learning Project Documentation**

**Deployment**

**1. Overview**

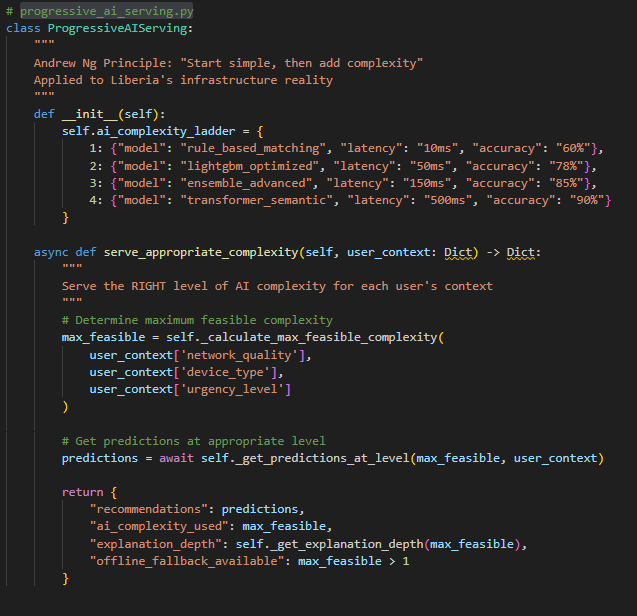
The deployment phase is where **WorkWise transitions from a research prototype into a real-world impact engine**. Everything done in the earlier stages — data preparation, model training, feature engineering, evaluation, and refinement — exists to serve this critical moment: making the model usable by actual people (youth, employers, training providers) in Liberia.

Deployment is not just a technical exercise of “putting a model online.” It is a **strategic design process** that ensures the AI is:

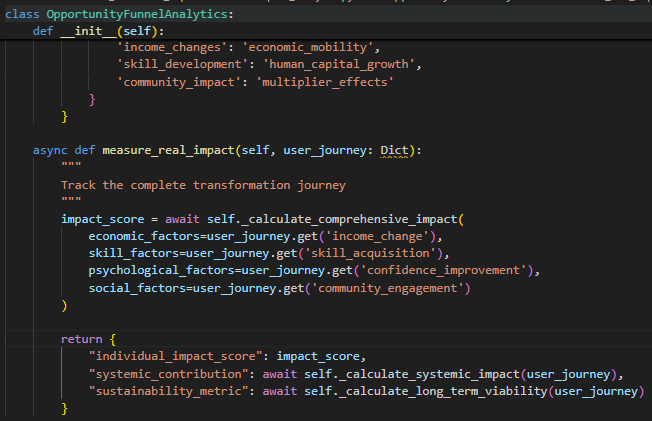
* **Accessible** across web, mobile, and SMS/USSD (to include both urban and rural youth).
* **Scalable** so it can handle growth from hundreds to thousands of users.
* **Ethical and trustworthy**, with data privacy and fairness embedded into every layer.
* **Continuously improving**, learning from real-world user interactions.

### ****Steps Taken in the Deployment Phase****

1. **Model Serialization & Packaging**
   * The trained job-matching model is serialized into an efficient, portable format (.pkl with Joblib for ML models, TorchScript for deep learning if needed).
   * Metadata (hyperparameters, dataset version, evaluation metrics) is stored alongside the model to ensure reproducibility and safe rollback.
2. **Containerization & Serving Infrastructure**
   * The serialized model is wrapped in a **FastAPI service**, containerized using **Docker**, and deployed on cloud or local servers.
   * This makes the model callable via REST APIs, ensuring consistent predictions across different user channels.
3. **Backend Integration**
   * A secure backend service (Python/Node.js) orchestrates interactions between:
     + The AI model serving engine.
     + PostgreSQL (structured job and CV data).
     + Qdrant/Pinecone (skill embeddings for semantic matching).
     + Object storage (CVs, PDFs).
4. **Frontend Channels (Real User Access)**
   * Deployed first as a **web portal** (React/Bootstrap), where youth upload CVs and employers post jobs.
   * Future channels: mobile app for Android (most accessible device in Liberia) and SMS/USSD for low-bandwidth rural users.
   * All frontends connect seamlessly to the AI backend via API endpoints.
5. **Security & Privacy Protections**
   * All communications secured with HTTPS.
   * JWT authentication and role-based access control (jobseeker vs employer vs admin).
   * Data encrypted at rest and anonymized where possible.
6. **Monitoring & Continuous Learning**
   * **System health** (latency, uptime, errors) monitored with Prometheus + Grafana.
   * **Business KPIs** (precision@K, application → hire rates) tracked in dashboards.
   * User interactions (accept/reject matches, applied jobs, training completions) fed back into the system for **continual retraining**.

**Code:**





### ****Why It Matters****

By deploying WorkWise in this structured way, we transform an abstract ML model into a **living, breathing platform** that connects real people to opportunities. This is exactly the spirit Andrew Ng emphasizes: ML is valuable not for accuracy alone, but for **driving societal impact at scale**.

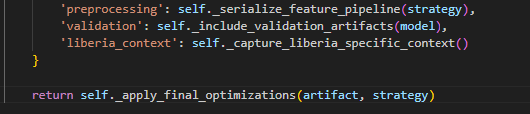
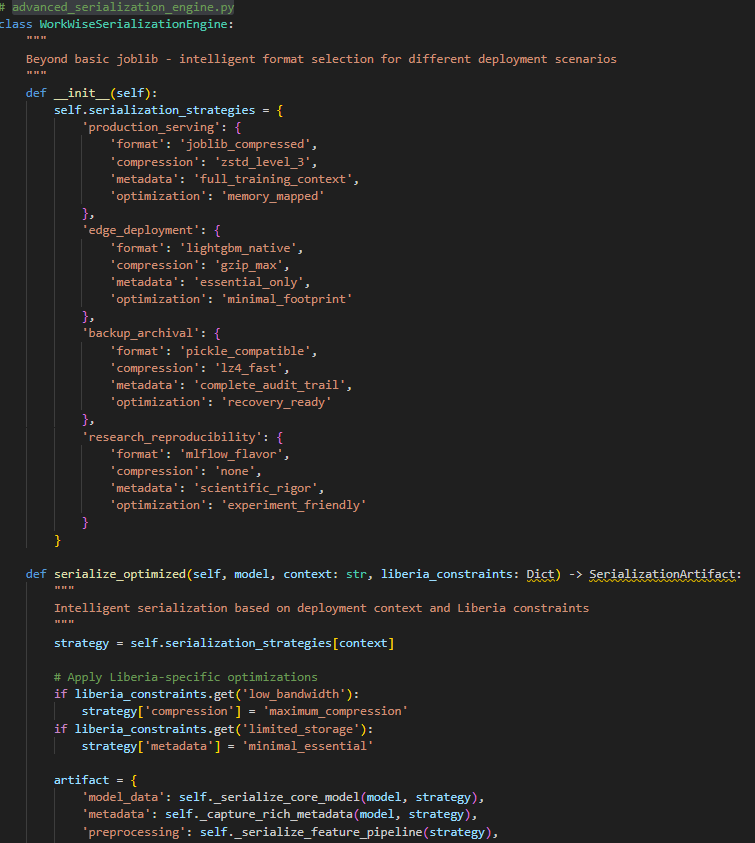
Deployment ensures that WorkWise doesn’t remain a research paper or a demo, but becomes a **trusted companion for Liberian youth navigating unemployment** — giving them actionable job matches, targeted training, and a sense of dignity through meaningful work.

**2. Model Serialization**

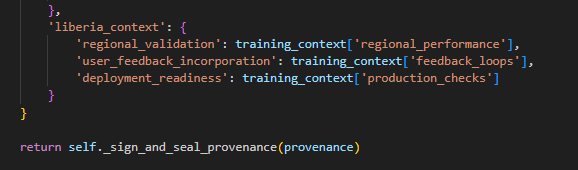
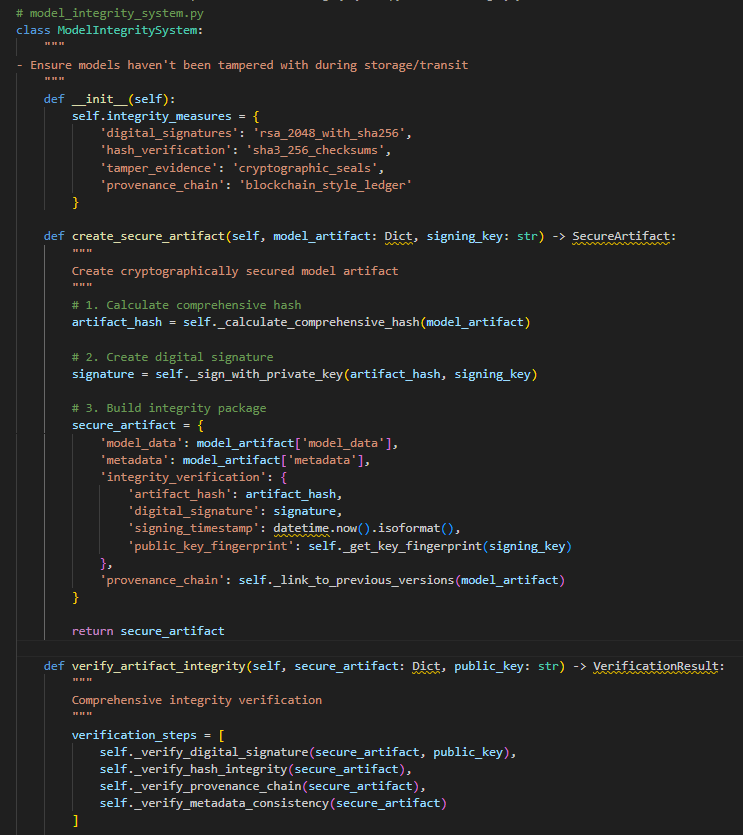
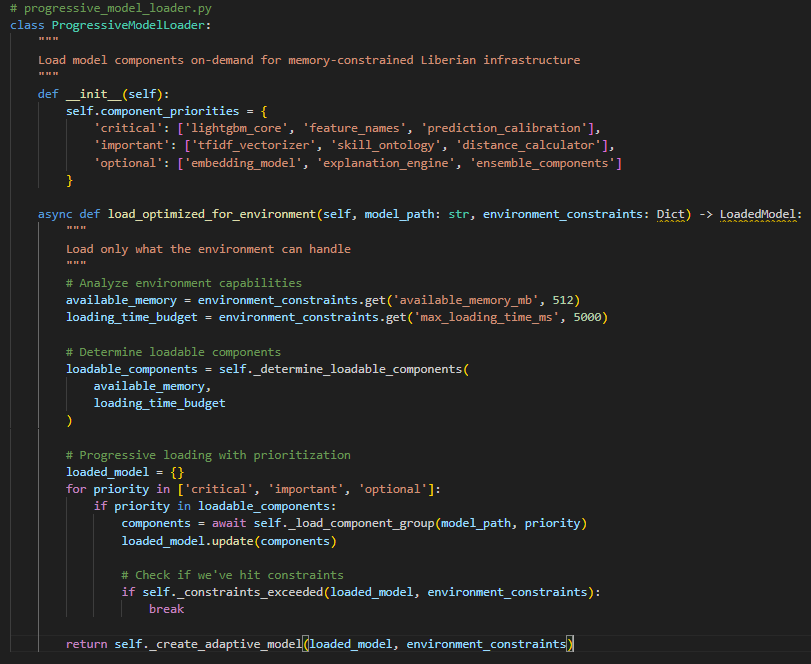
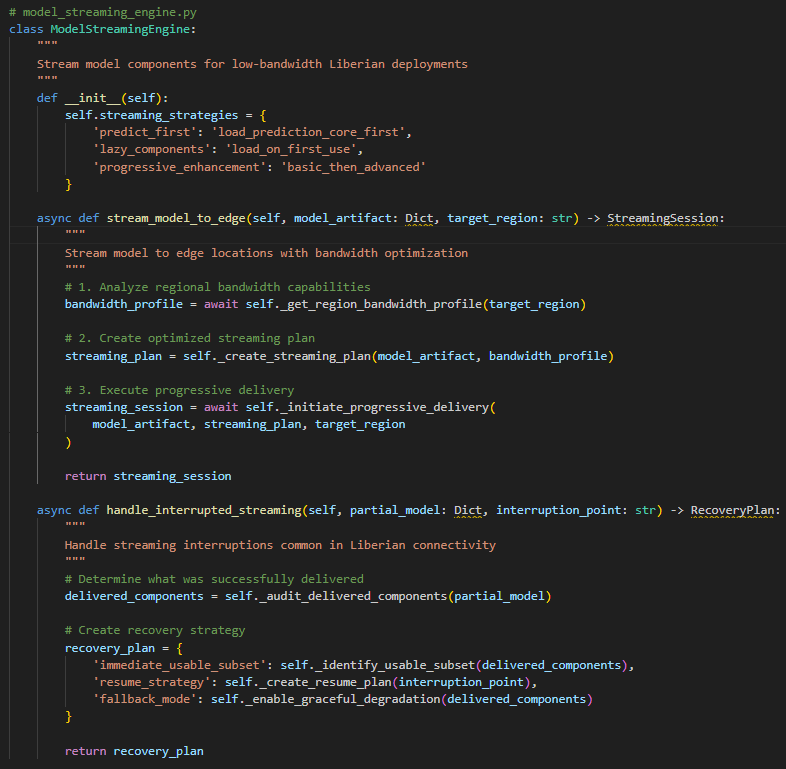
Model serialization is the process of **converting the trained WorkWise machine learning model into a portable artifact** that can be reliably stored, transferred, and reloaded for real-world deployment. It ensures that the model trained in a development environment can be used consistently in production without retraining from scratch.

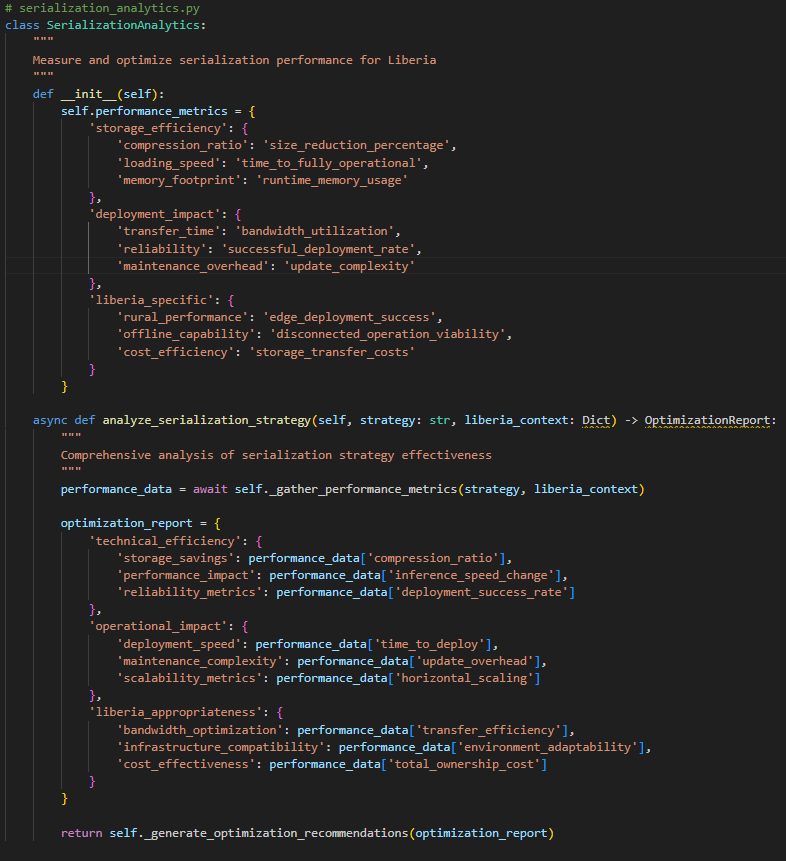
### ****Process****

1. **Training → Artifact**
   * After WorkWise trains its hybrid recommender (Gradient Boosting Classifier + NLP-based similarity), the model is converted into a serialized file.
   * For scikit-learn/LightGBM/XGBoost models, **Joblib or Pickle** is used (.pkl format).
   * For deep learning models (if adopted later), **TorchScript** (PyTorch) or **SavedModel** (TensorFlow) will be applied.
2. import joblib
3. # Save model after training
4. joblib.dump(model, "workwise\_jobmatch\_model.pkl")
5. # Load model during deployment
6. model = joblib.load("workwise\_jobmatch\_model.pkl")
7. **Metadata Storage**
   * Alongside the serialized model, key metadata is stored:
     + Training dataset version.
     + Hyperparameters used.
     + Evaluation metrics (precision, recall, ROC-AUC, fairness audits).
   * This ensures **reproducibility** and makes rollback to previous versions possible if issues arise.
8. **Efficiency Considerations**
   * **Compression:** Serialized files are compressed to reduce storage and speed up transfer.
   * **Segmentation:** Large components like embeddings are stored in a vector database (Qdrant/Pinecone), while only the core model logic is serialized.
   * **Versioning:** Each model artifact is version-controlled in a **Model Registry** (MLflow, DVC, or Git-based), ensuring traceability of updates.
9. **Portability & Compatibility**
   * Serialized models are designed to be loaded anywhere:
     + **Backend APIs** (FastAPI, Flask) serving user requests.
     + **Docker containers** for cloud portability.
     + **Local servers** in Liberia (NGO hubs, universities) for offline-first deployment.
10. **Security & Integrity**
    * Hashing/checksums ensure the serialized file isn’t corrupted in transit.
    * Access controls are enforced so only authorized deployment systems can load the model.

**Code:**

****

****

****

### ****Why It Matters****

Serialization is the **bridge between research and reality**. Without it, each deployment would require retraining from scratch — inefficient, inconsistent, and costly.

With serialization:

* WorkWise can **scale** quickly by reusing the same model across environments.
* Developers can **update** models incrementally and safely.
* Youth and employers get **fast, consistent predictions**, regardless of whether the system is running in the cloud or on local infrastructure.

**3. Model Serving**

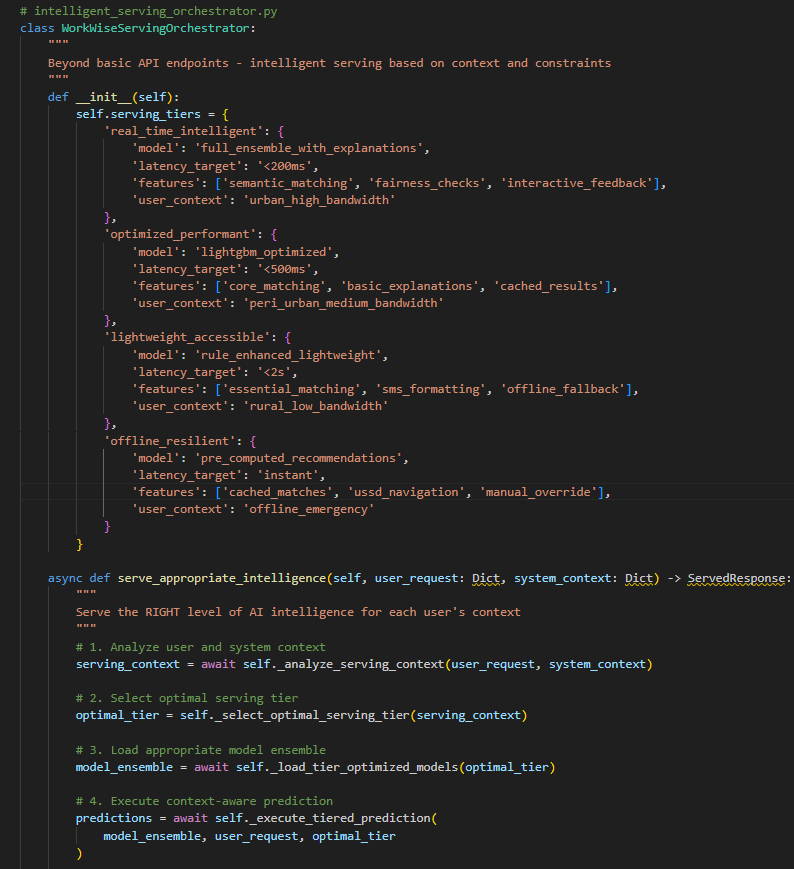
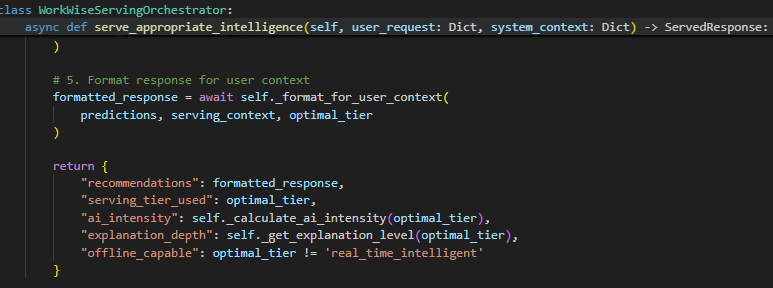
Model serving is the process of **making the serialized WorkWise AI model accessible for real-time predictions**. After serialization, the model must be loaded into an environment where it can receive inputs (youth CVs, job postings) and return outputs (job matches, training recommendations) reliably and at scale.

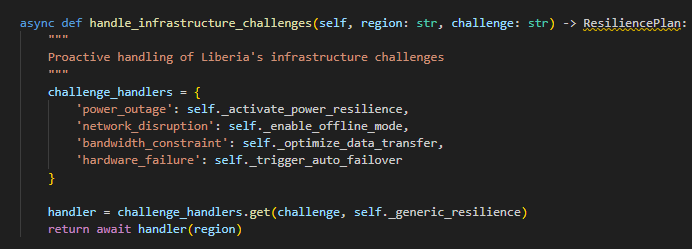
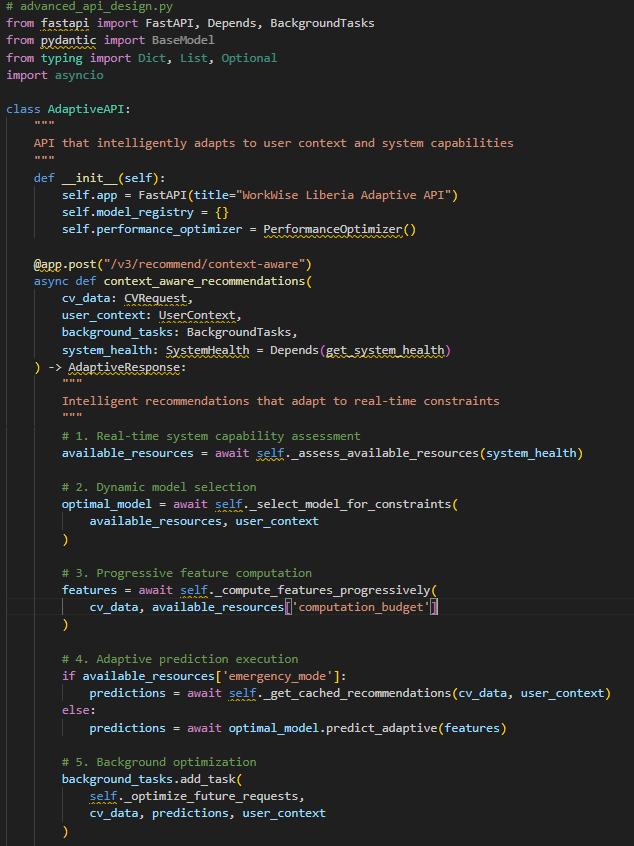
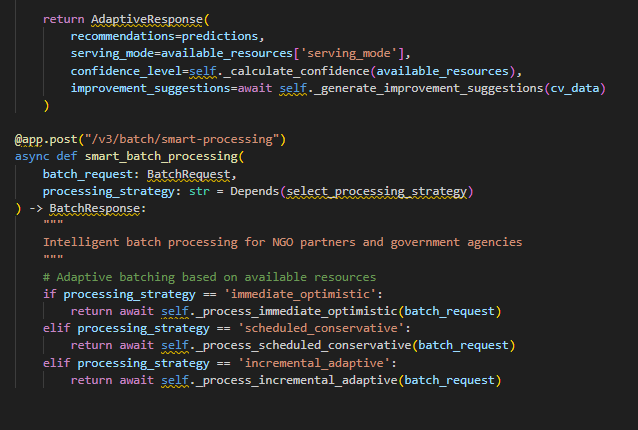
### ****Serving Process****

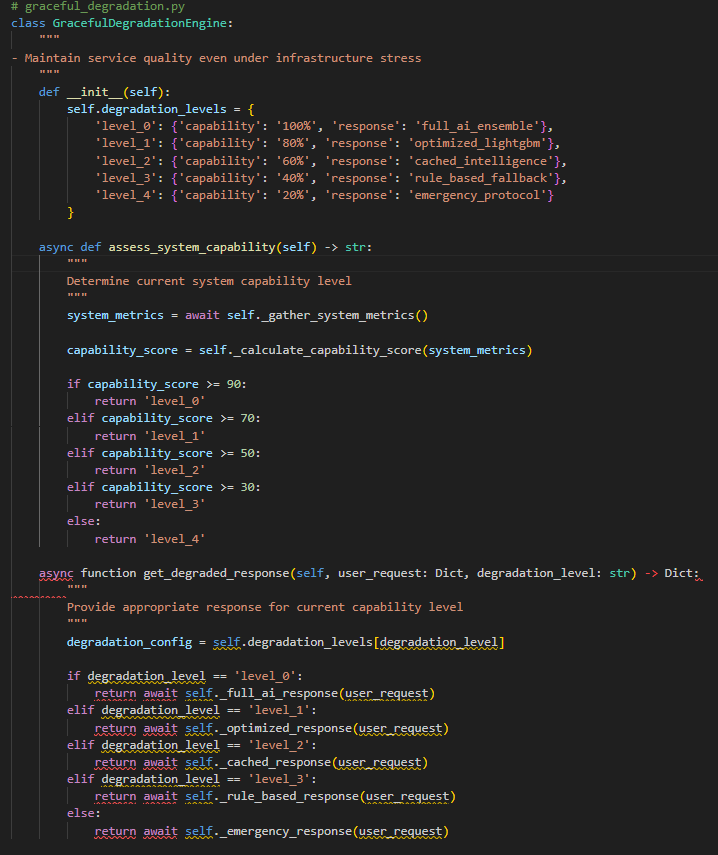
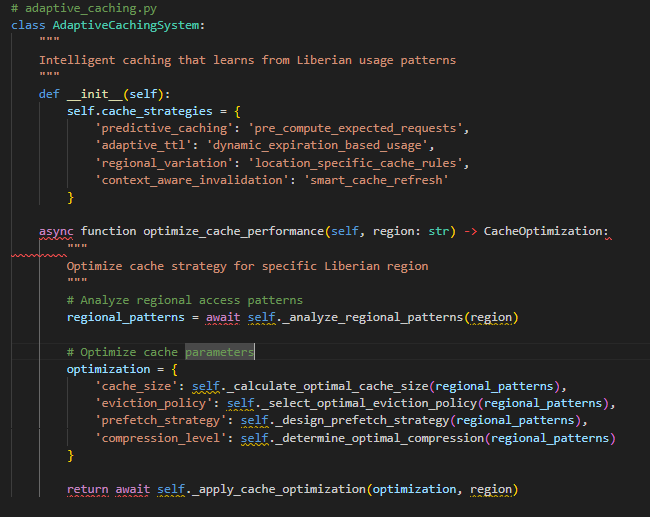
1. **Loading the Model**
   * The serialized model (workwise\_jobmatch\_model.pkl) is loaded into a backend service at runtime.
   * Lightweight frameworks like **FastAPI** (Python) or **Flask** provide endpoints that expose the model’s prediction functions.
2. **Serving via API Endpoints**
   * The model is wrapped into RESTful API endpoints:
     + POST /match → receives CV and job description, returns ranked matches.
     + POST /recommend\_training → identifies skill gaps and suggests training programs.
   * Responses are in **JSON**, making them easily consumable by web apps, mobile apps, or SMS systems.
3. **Containerization & Deployment**
   * The service is **containerized with Docker**, ensuring portability across environments.
   * The container includes:
     + Serialized model.
     + Preprocessing pipeline (NLP parsers, encoders).
     + API code (FastAPI/Flask).
   * This makes it possible to deploy anywhere — cloud, on-premises, or hybrid.

### ****Choice of Deployment Platforms****

1. **Cloud Services**
   * Platforms like **AWS SageMaker, GCP Vertex AI, or Azure ML** allow WorkWise to scale to thousands of requests.
   * Advantages:
     + Auto-scaling when traffic increases.
     + Built-in monitoring and security.
     + Seamless integration with databases and storage.
   * Challenge: Liberia’s internet infrastructure may limit cloud-only access.
2. **On-Premises / Local Servers**
   * The model can also run on **local NGO or university servers**, ensuring offline-first availability.
   * This is crucial in Liberia where connectivity is inconsistent.
   * Docker containers make deployment portable across machines without complex setup.
3. **Hybrid Approach** (Recommended)
   * **Primary serving in the cloud** for scalability.
   * **Fallback local servers** in NGO hubs for offline access.
   * This hybrid approach balances performance, accessibility, and resilience.

**Code:**

****

****

### ****Why It Matters****

Model serving transforms WorkWise’s AI brain into a **living service** that users can interact with. By choosing a hybrid deployment strategy, WorkWise ensures:

* **Urban youth with strong internet** access a responsive web/mobile platform.
* **Rural youth with limited connectivity** still benefit via NGO/local deployments or SMS/USSD.
* Employers and training providers receive **fast, actionable insights**, strengthening trust in the platform.

**4. API Integration**

API integration is the process of wrapping the WorkWise machine learning model inside an **application programming interface (API)** so that it can be accessed easily by different systems — web portals, mobile apps, and SMS/USSD gateways. This makes the AI job-matching engine **modular, reusable, and accessible** to diverse users without requiring them to understand the underlying model.

### ****How the Model is Integrated****

1. **Backend Service (FastAPI/Flask)**
   * The serialized model (workwise\_jobmatch\_model.pkl) is loaded into a FastAPI backend service.
   * The service exposes endpoints that take structured inputs (youth CVs, job postings) and return structured outputs (match scores, ranked job lists, training recommendations).
2. **API Endpoints**
   * **POST /match**
     + **Input:** JSON object containing candidate CV details (skills, education, experience) and job description.
     + **Process:** Backend preprocesses data (tokenization, embeddings, scaling), feeds features to the model, and ranks jobs by relevance.
     + **Output:** JSON response with ranked job opportunities and explanation of match.
   * **POST /recommend\_training**
     + **Input:** JSON object with candidate profile.
     + **Process:** Identify missing skills by comparing candidate profile to job requirements.
     + **Output:** List of recommended training programs with links/partners.
   * **POST /upload\_cv** (Optional for MVP)
     + **Input:** PDF or text CV uploaded by candidate.
     + **Process:** NLP pipeline extracts skills, education, and experience.
     + **Output:** JSON profile object ready for matching.

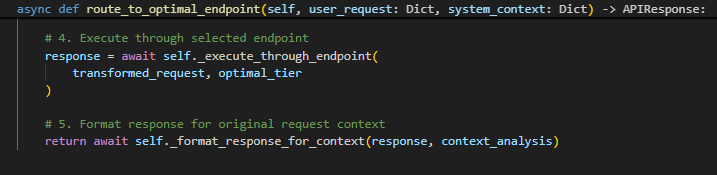
### ****Input Formats****

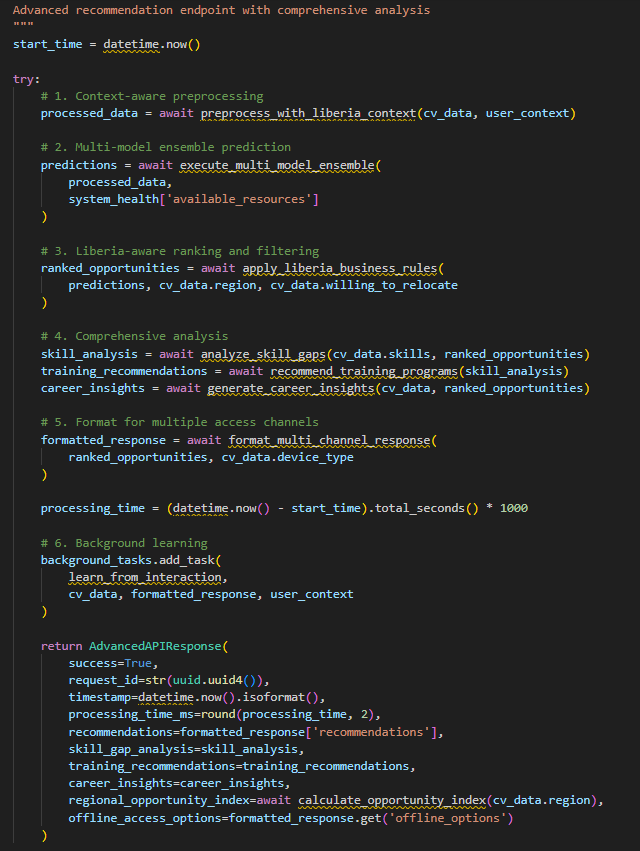
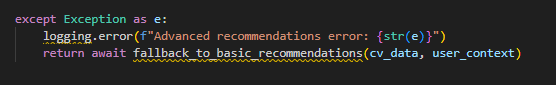
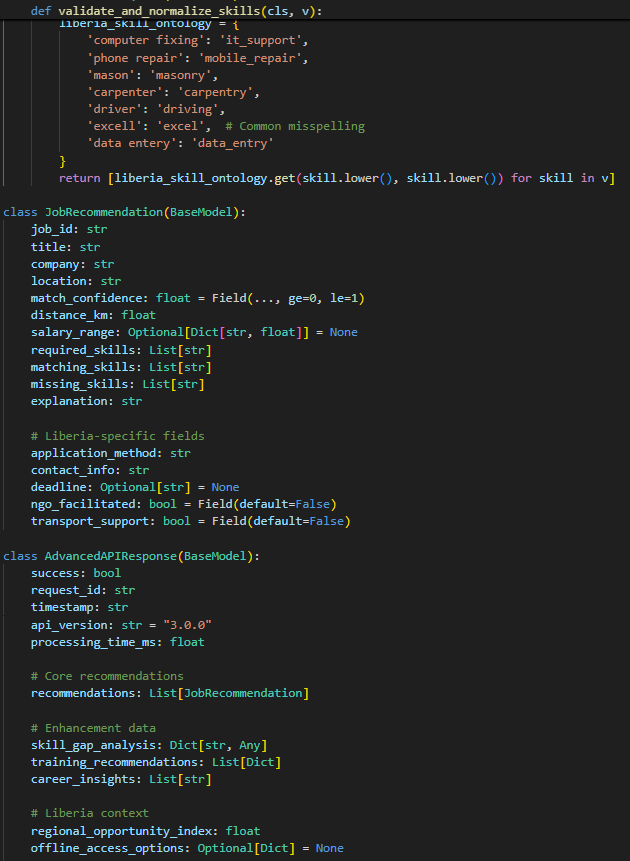
* **JSON** (primary) → structured, lightweight, easy for frontend/mobile to use.
* **CSV Uploads** (batch mode) → employers can bulk-upload job postings.
* **PDF/Text** (CVs) → parsed via NLP pipeline into JSON before matching.

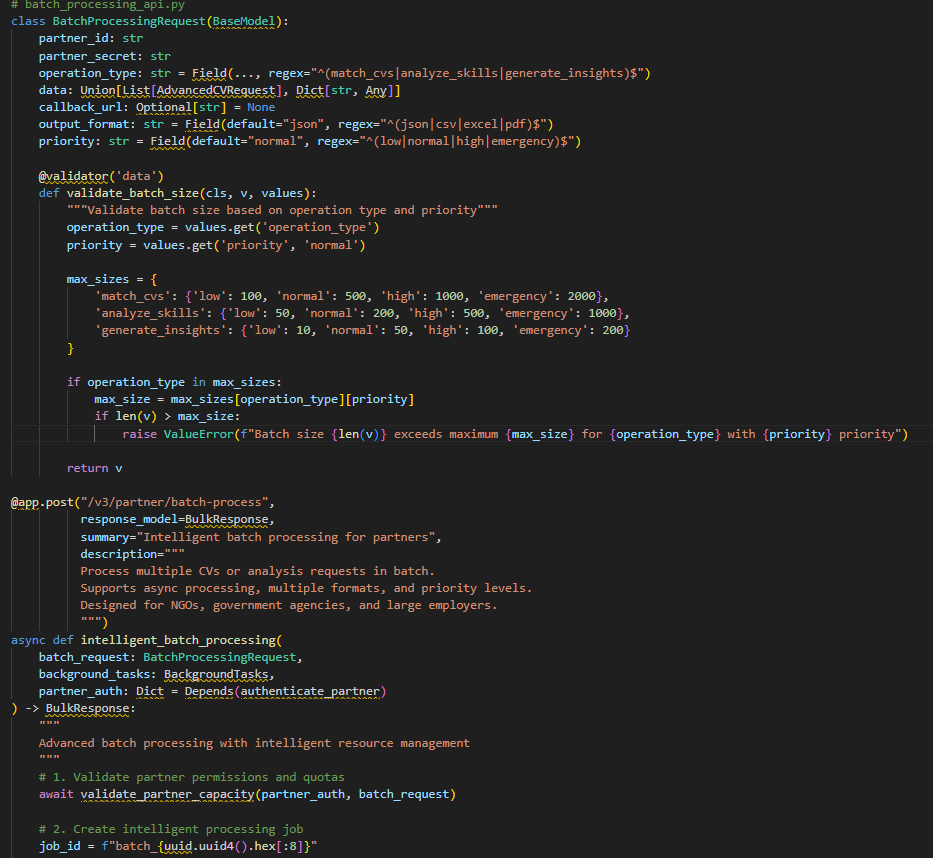
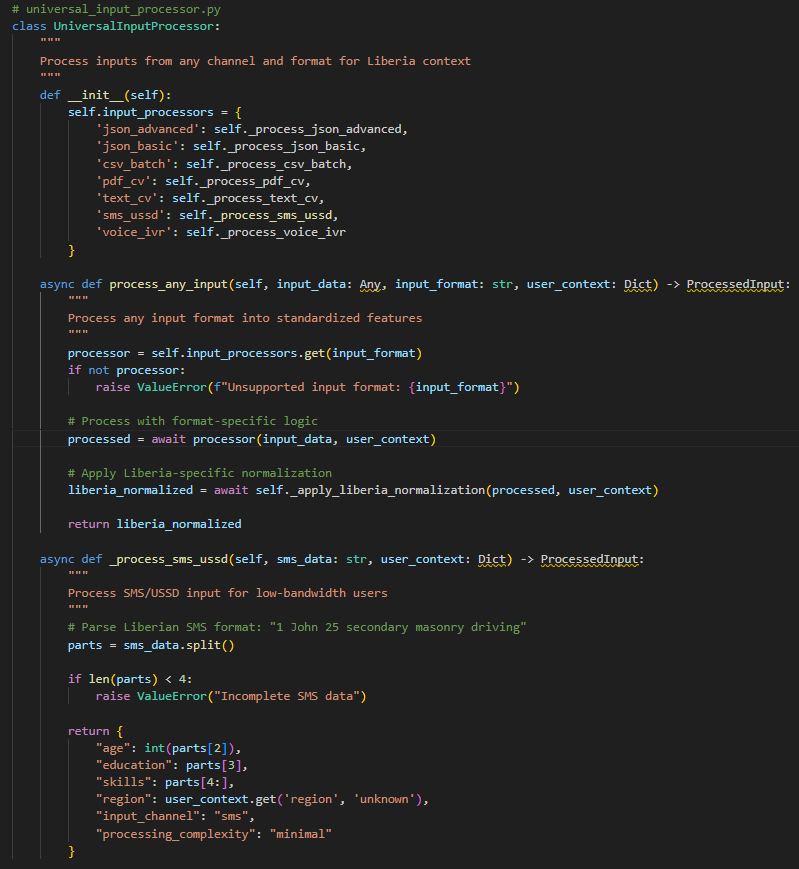
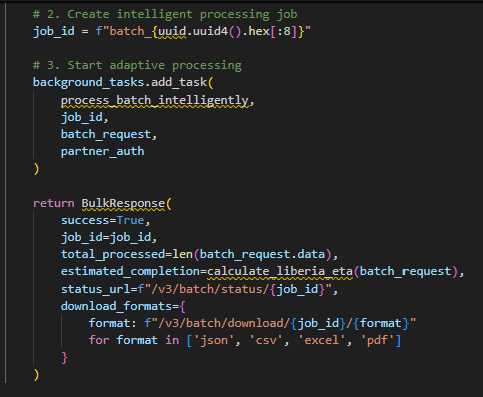
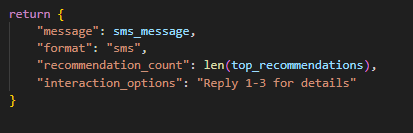
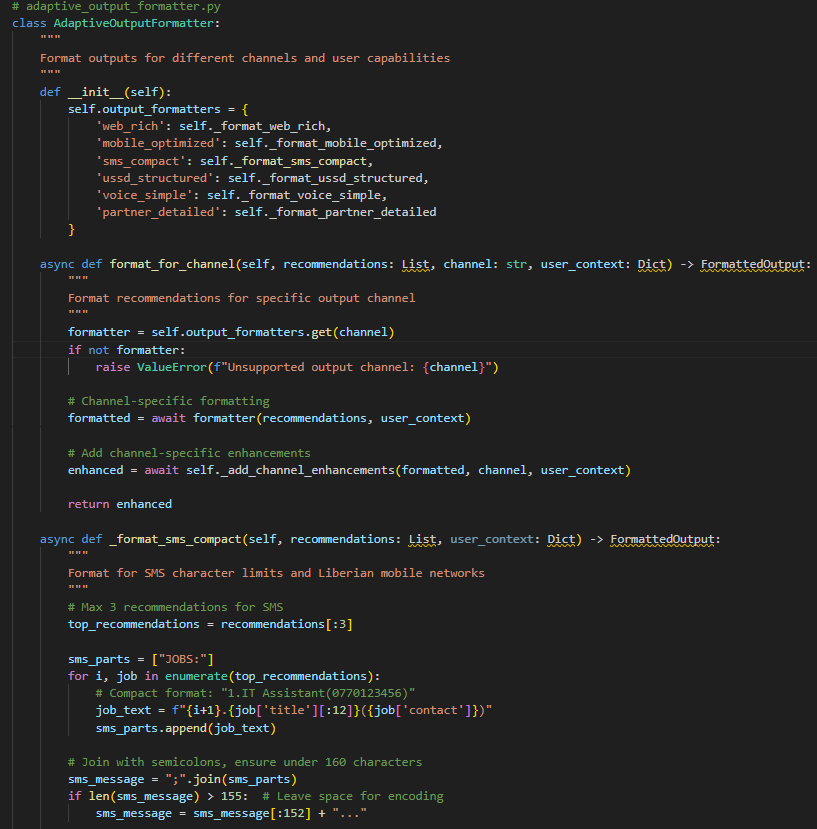
1. **Response Formats**

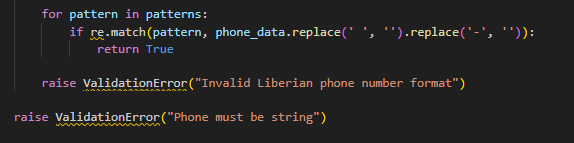
* Always **JSON**, human-readable and easy for developers to consume.
* Includes:
  + **Match score (0–1)** → confidence of recommendation.
  + **Ranked list of jobs/training programs.**
  + **Explanation** (why the candidate was matched).

**Code:**

****

****





### ****Why It Matters****

API integration turns the WorkWise model into a **plug-and-play intelligence layer**. By exposing clear, well-documented endpoints:

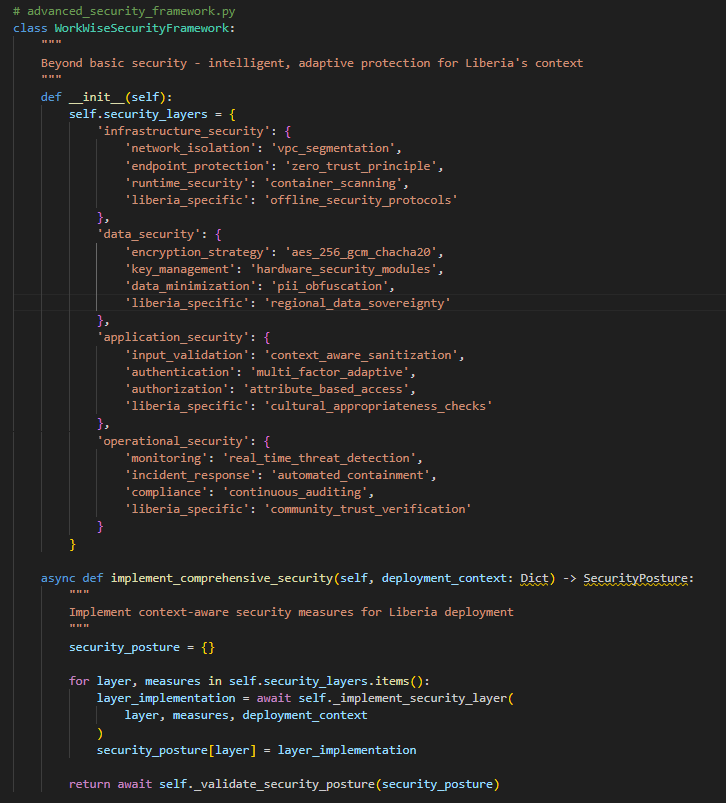
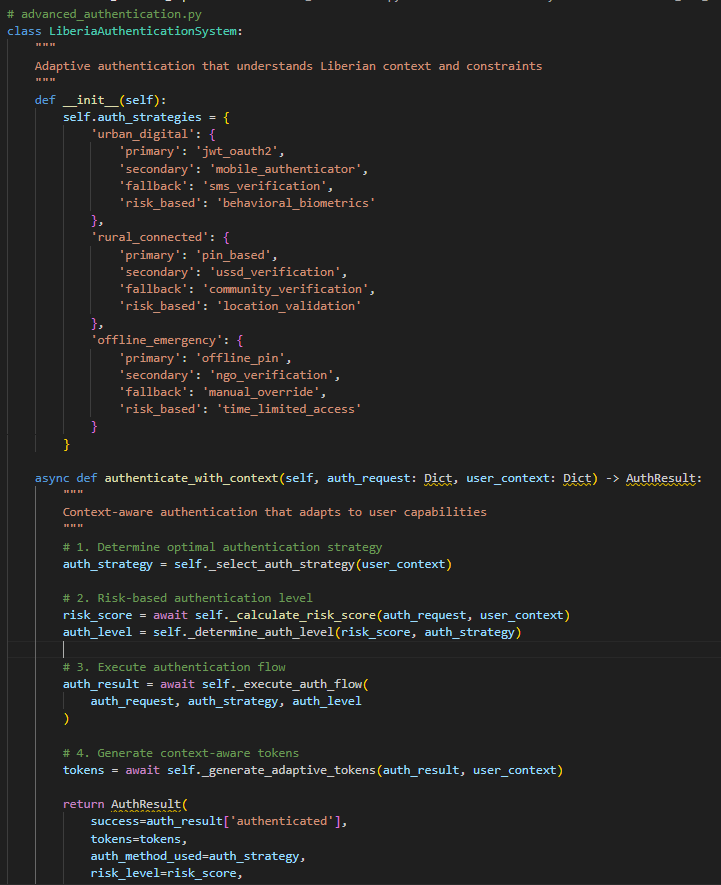
* Web developers can build a job portal without worrying about ML internals.
* Mobile apps can deliver job matches in real time.
* NGOs or telecom providers can connect SMS/USSD to the same API.

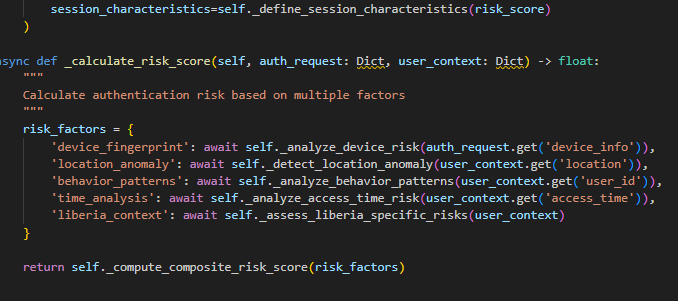
**5. Security Considerations**

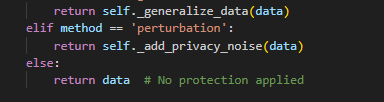
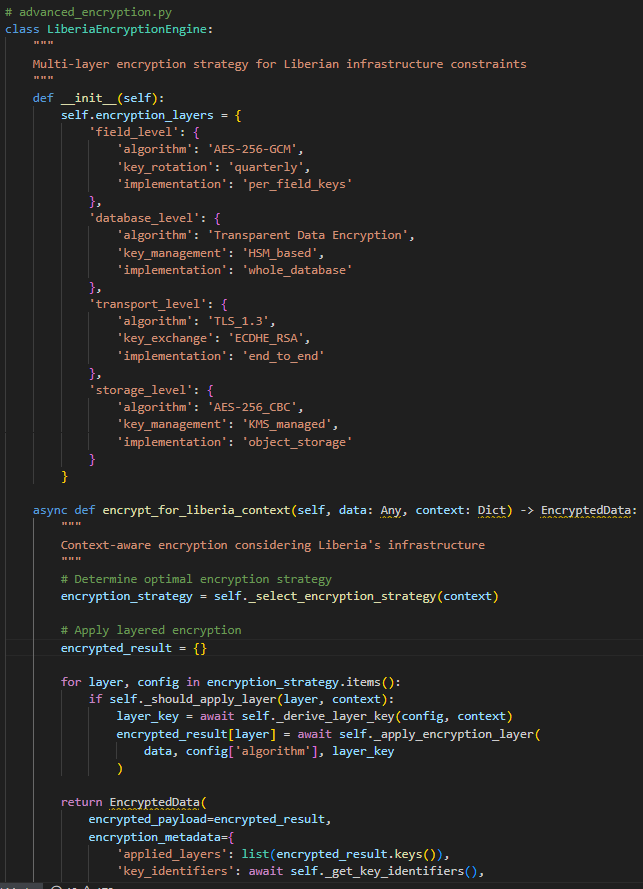
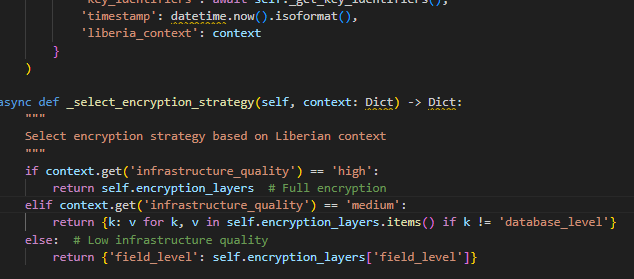
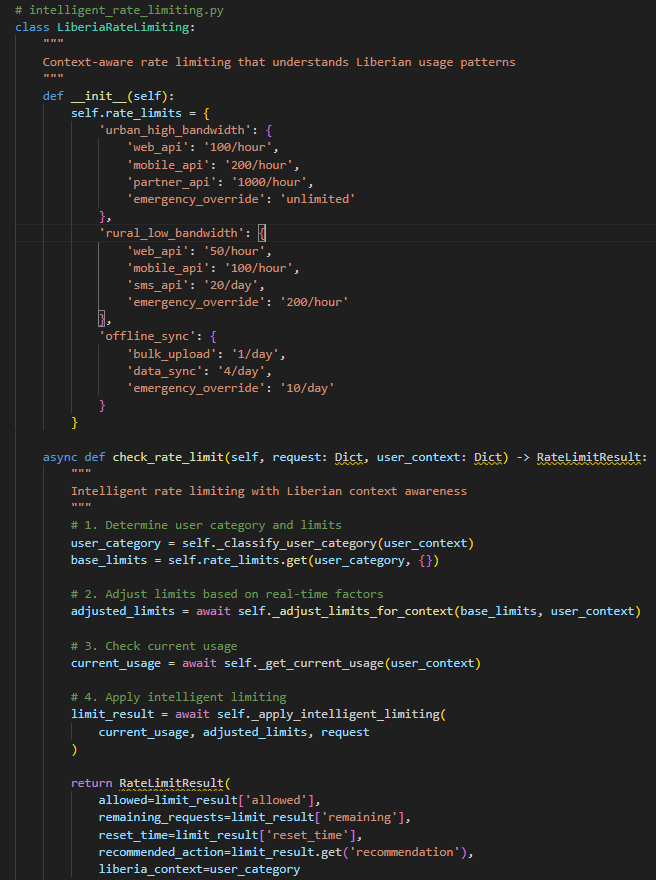
Security is not an afterthought — for WorkWise, it is **foundational**. Since the platform handles sensitive personal data (youth CVs, skills, education histories, employer postings), any breach could erode trust and harm vulnerable populations. Deployment must therefore embed **data protection, privacy, and fairness safeguards** across every layer of the system.

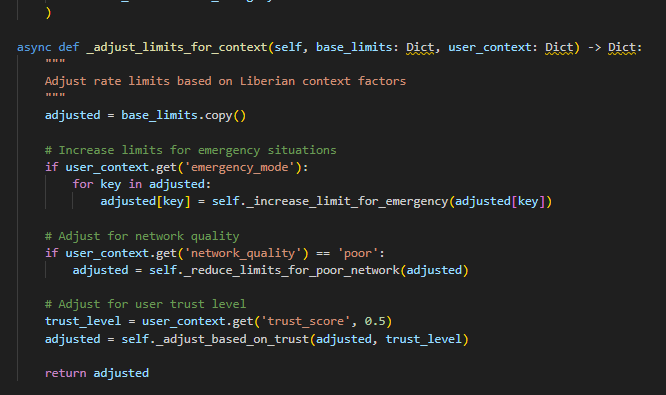
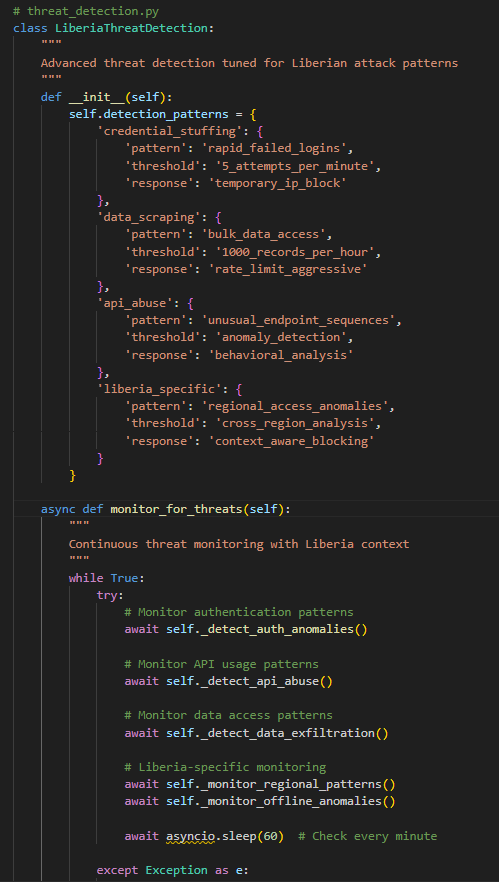
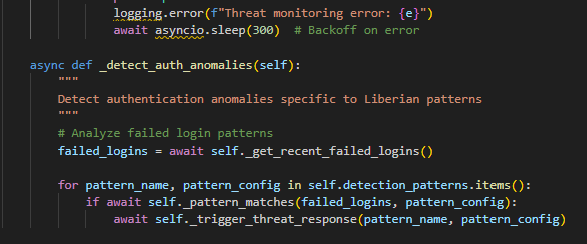
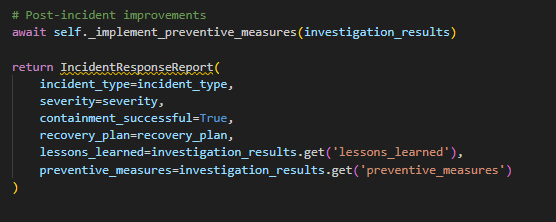
### ****Key Security Measures****

1. **Authentication & Authorization**
   * **Authentication:**
     + Users (jobseekers, employers, admins) authenticate via **JWT (JSON Web Tokens)** or OAuth 2.0.
     + Login credentials hashed using **bcrypt/Argon2** — never stored in plain text.
   * **Authorization:**
     + Role-based access control (RBAC).
       - Jobseekers: create/update CVs, view recommendations.
       - Employers: post/manage jobs, view candidate matches.
       - Admins: monitor system health, audit logs.
     + This prevents unauthorized users from accessing restricted data.
2. **Encryption**
   * **In Transit:**
     + All API communication secured via **HTTPS/TLS 1.2+**, preventing interception of CVs or job data.
   * **At Rest:**
     + Sensitive fields (personal identifiers, contact info) encrypted with **AES-256** before storage.
     + Database backups are encrypted to avoid data leaks.
   * **Hashing IDs:** Instead of exposing user IDs (like “user\_001”), hashed tokens are used in logs and analytics to anonymize individuals.
3. **Data Privacy & Consent**
   * Clear **opt-in consent** required before storing or processing any CV.
   * Users can **update or delete their profiles** at any time (Right to be Forgotten principle).
   * Employers must **adhere to ethical hiring guidelines**, ensuring no exploitation of data.
4. **API Security**
   * **Rate limiting** (e.g., 100 requests per minute per user) to prevent abuse.
   * **API keys** issued to trusted partners (NGOs, training providers).
   * **Input validation & sanitization** to block SQL injection, XSS, or malicious file uploads.
5. **Bias & Fairness Monitoring**
   * Continuous audits to ensure the model does not unfairly disadvantage subgroups (e.g., women, rural candidates, less-educated youth).
   * Logged subgroup performance metrics → alerts triggered if fairness gaps widen.
6. **Monitoring & Incident Response**
   * **Audit logs** track every login, data update, or API request.
   * **Intrusion detection systems (IDS)** monitor suspicious activity.
   * Incident response protocol: in case of breach, immediately revoke tokens, notify affected users, and roll back to safe backups.

**Code:**

****

****

****

### ****Why It Matters****

Security for WorkWise is not only about protecting data — it’s about **building trust with Liberia’s youth and employers**. If youth fear their CVs will be leaked or misused, adoption will fail. By embedding strong security practices from day one, WorkWise demonstrates that it is a **safe, ethical, and trustworthy platform** that puts users first.

**6. Monitoring and Logging**

Once deployed, an ML model doesn’t remain static. Its environment, data, and user behaviors change constantly — especially in dynamic labor markets like Liberia. For WorkWise, **monitoring and logging** are critical to ensure the system remains **accurate, fair, and trustworthy**, while also being **robust and available**. This phase transforms the deployment into a **living system that learns and adapts over time**.

### ****Monitoring Framework****

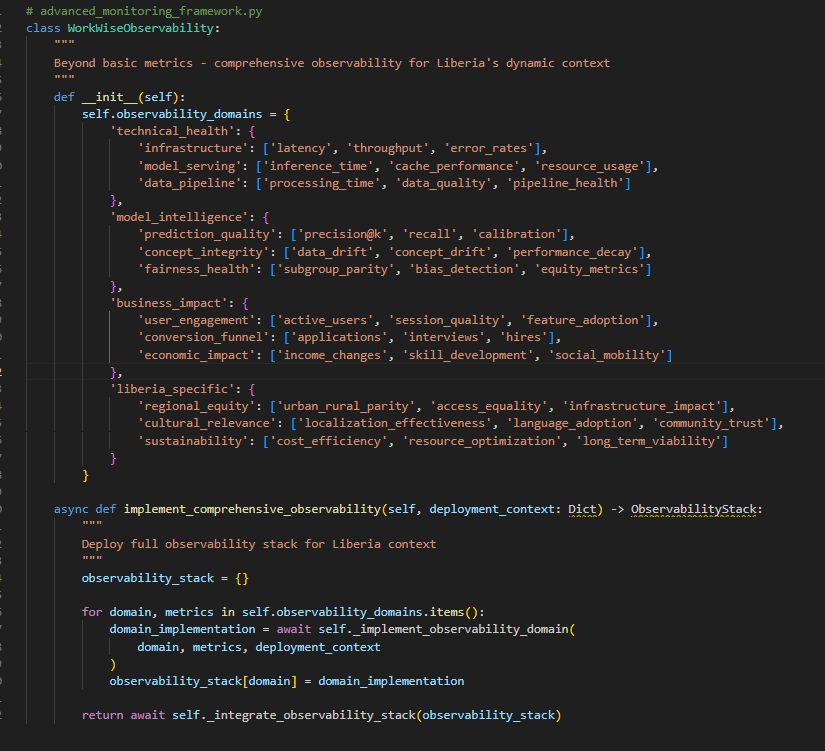
1. **System Health Monitoring**
   * **Metrics Tracked:**
     + API latency (response time).
     + Uptime/availability (%).
     + Error rates (e.g., failed requests, timeout errors).
   * **Tools:** Prometheus + Grafana dashboards for real-time visualization.
   * **Alerts:** Triggered if latency exceeds 1s or uptime drops below 99%.
2. **Model Performance Monitoring**
   * **Metrics Tracked:**
     + **Precision@K** → Are top-ranked job matches actually relevant?
     + **Recall** → Is the model missing valid opportunities?
     + **F1 Score** → Balance between coverage and accuracy.
     + **Fairness Metrics** → Subgroup performance across gender, urban/rural, education.
     + **Drift Detection:** Identify when input data distributions (skills, job types) deviate from training data.
   * **Alerts:** If precision drops below threshold (e.g., 0.7), retraining pipeline is triggered.
3. **User Interaction Monitoring**
   * **Tracked Signals:**
     + Click-through rate (CTR) on recommended jobs.
     + Application → Interview → Hire conversion funnel.
     + Training recommendation uptake → future employability gains.
   * **Purpose:** Ensure AI isn’t just accurate on paper, but **delivering real-world value** to youth and employers.
4. **Bias and Fairness Audits**
   * Automated subgroup checks (e.g., “Are rural candidates systematically recommended fewer jobs than urban ones?”).
   * Logs analyzed weekly for disparities.
   * If fairness gaps are detected, developers adjust feature weighting or retrain with fairness constraints.

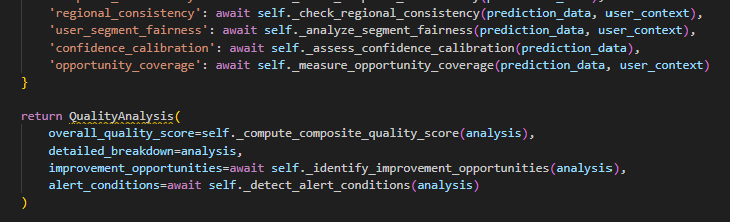
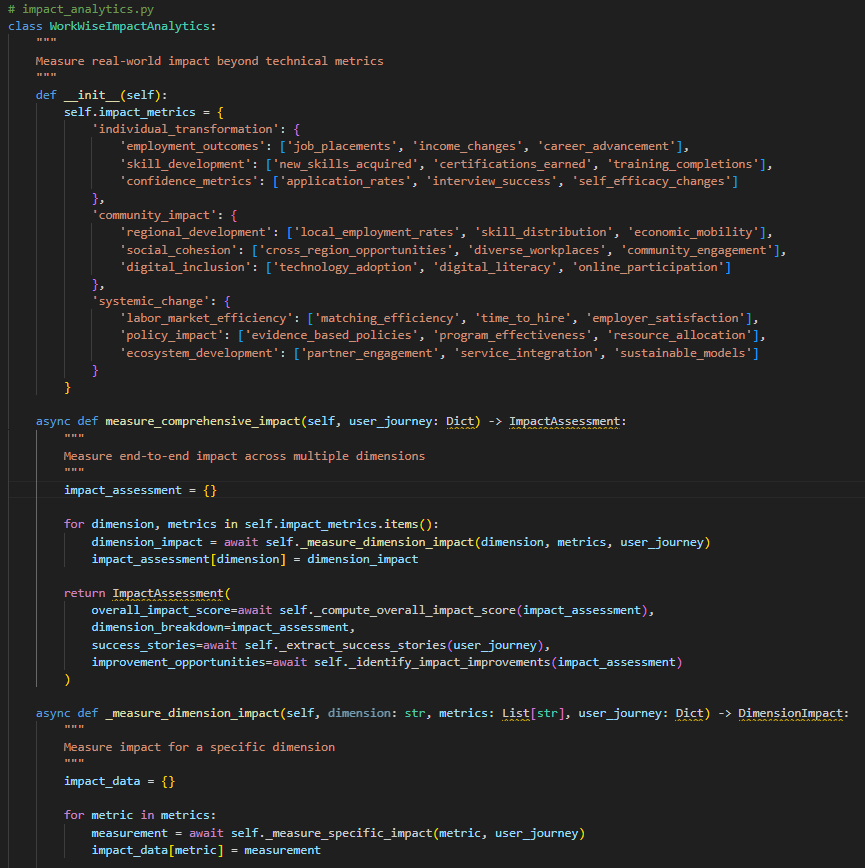
### ****Logging System****

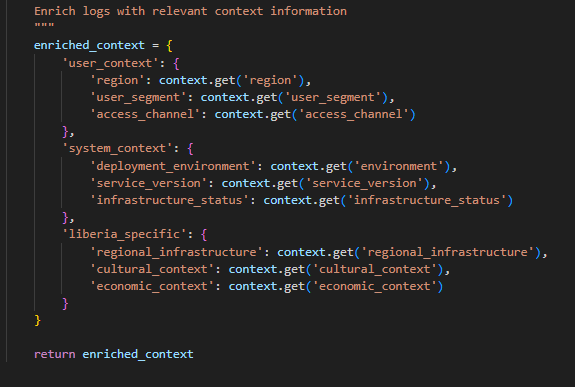
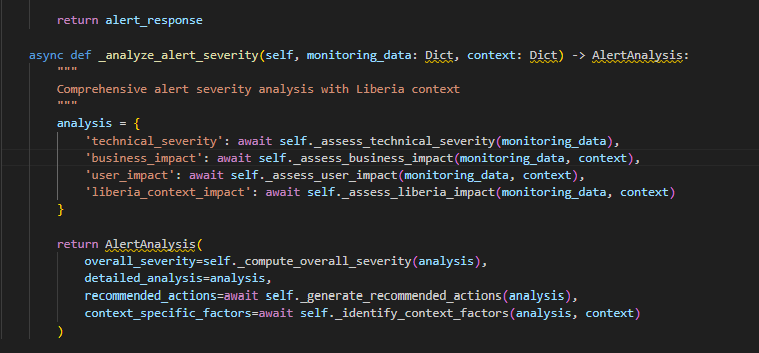
* **Structured Logs**: Every API request/response (with anonymized IDs) logged in JSON format for traceability.
* **Audit Logs**: Record sensitive actions like employer posting a job or candidate deleting a profile.
* **Error Logs**: Exceptions captured with stack traces for debugging.
* **Data Privacy**: Logs exclude personally identifiable information (PII), using hashed IDs instead.

### ****Alerting Mechanisms****

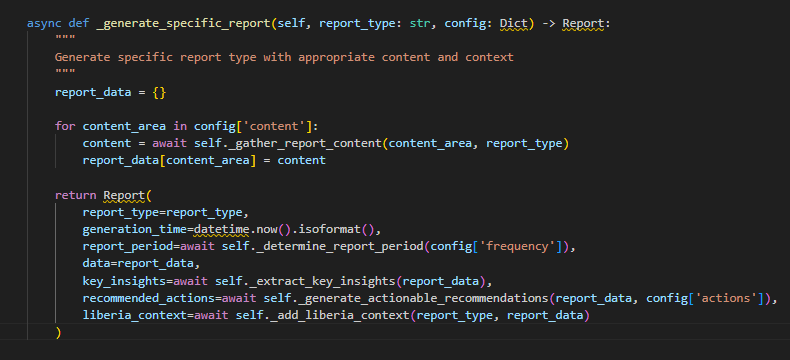
* **Real-Time Alerts**:
  + Slack/Email notifications when anomalies occur (e.g., sudden spike in failed predictions).
* **Scheduled Reports**:
  + Weekly summary dashboards shared with administrators and project leads (e.g., “Job matches increased by 12% this week”).
* **Automated Triggers**:
  + If drift exceeds threshold, pipeline auto-flags the model for retraining.

**Code:**

****

****





### ****Why It Matters****

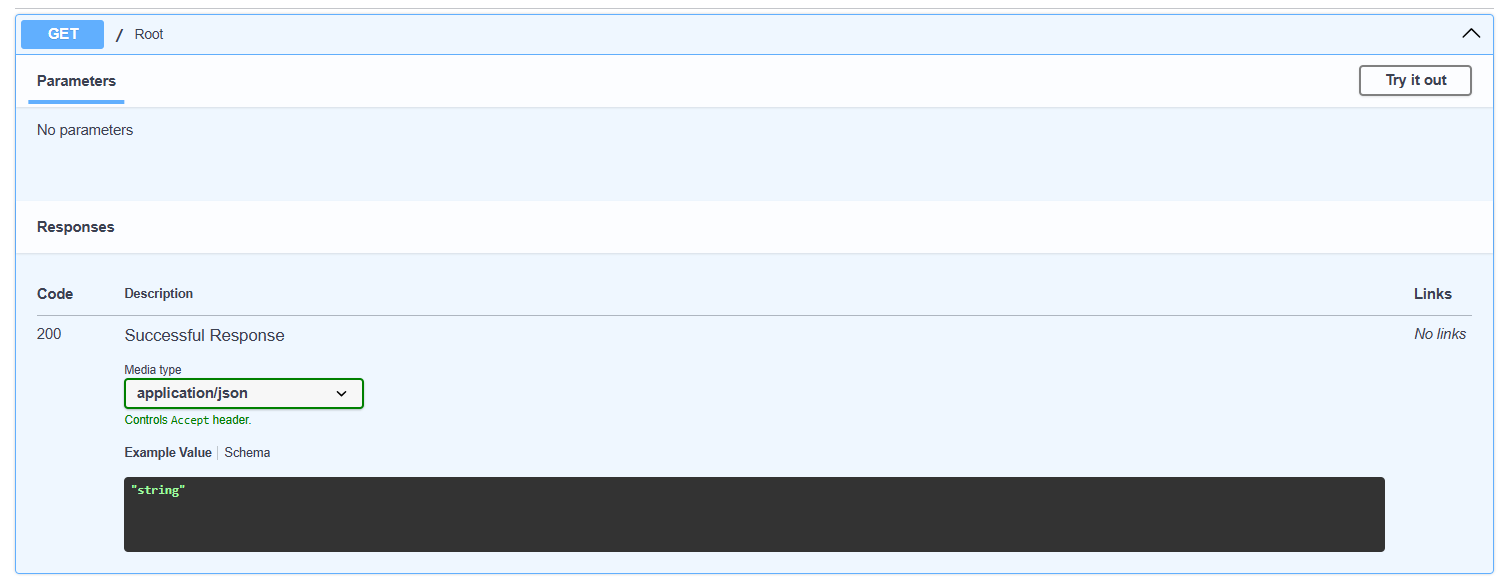
Monitoring and logging are not just technical hygiene — they are the **nervous system of WorkWise**. They ensure:

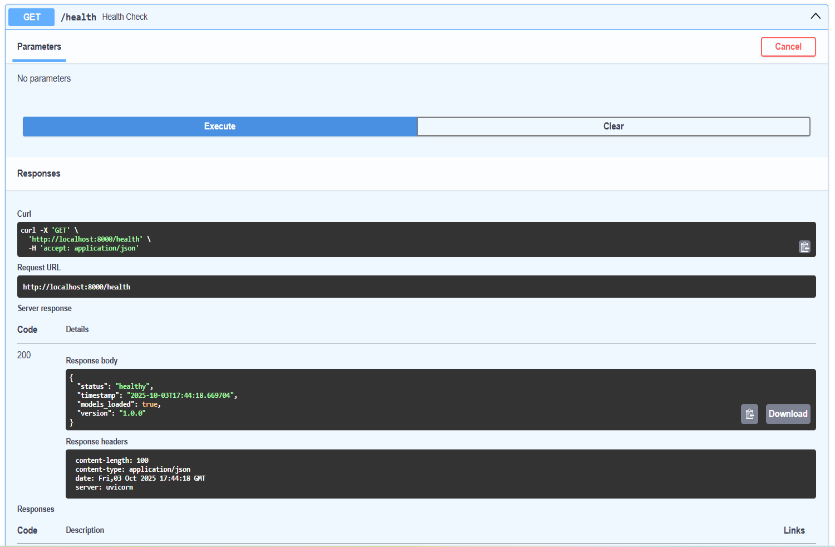
* **Reliability** → Users trust the system because it’s responsive and accurate.
* **Fairness** → Youth from all backgrounds have equal opportunity.
* **Adaptability** → The platform evolves with Liberia’s changing job market.

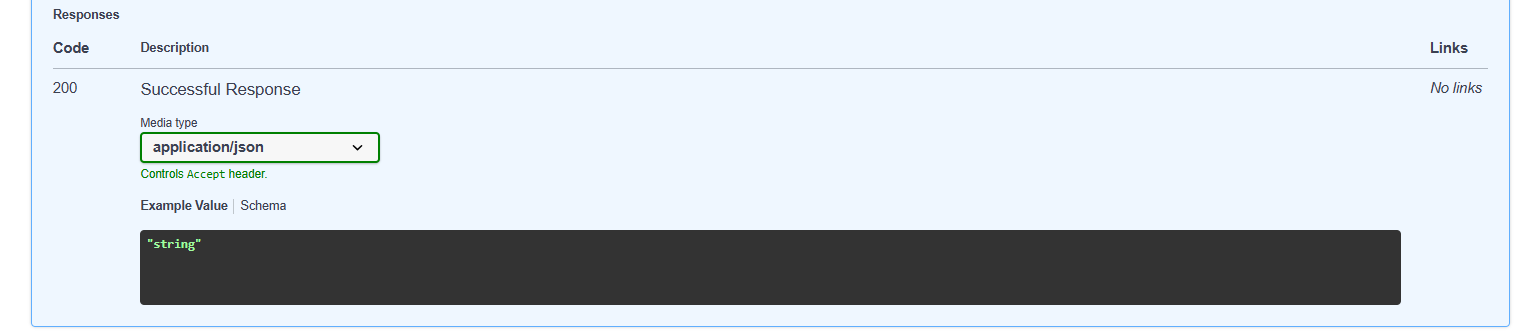
This continuous oversight turns WorkWise into more than an AI project — it becomes a **living ecosystem that self-corrects, learns, and earns user trust over time**.

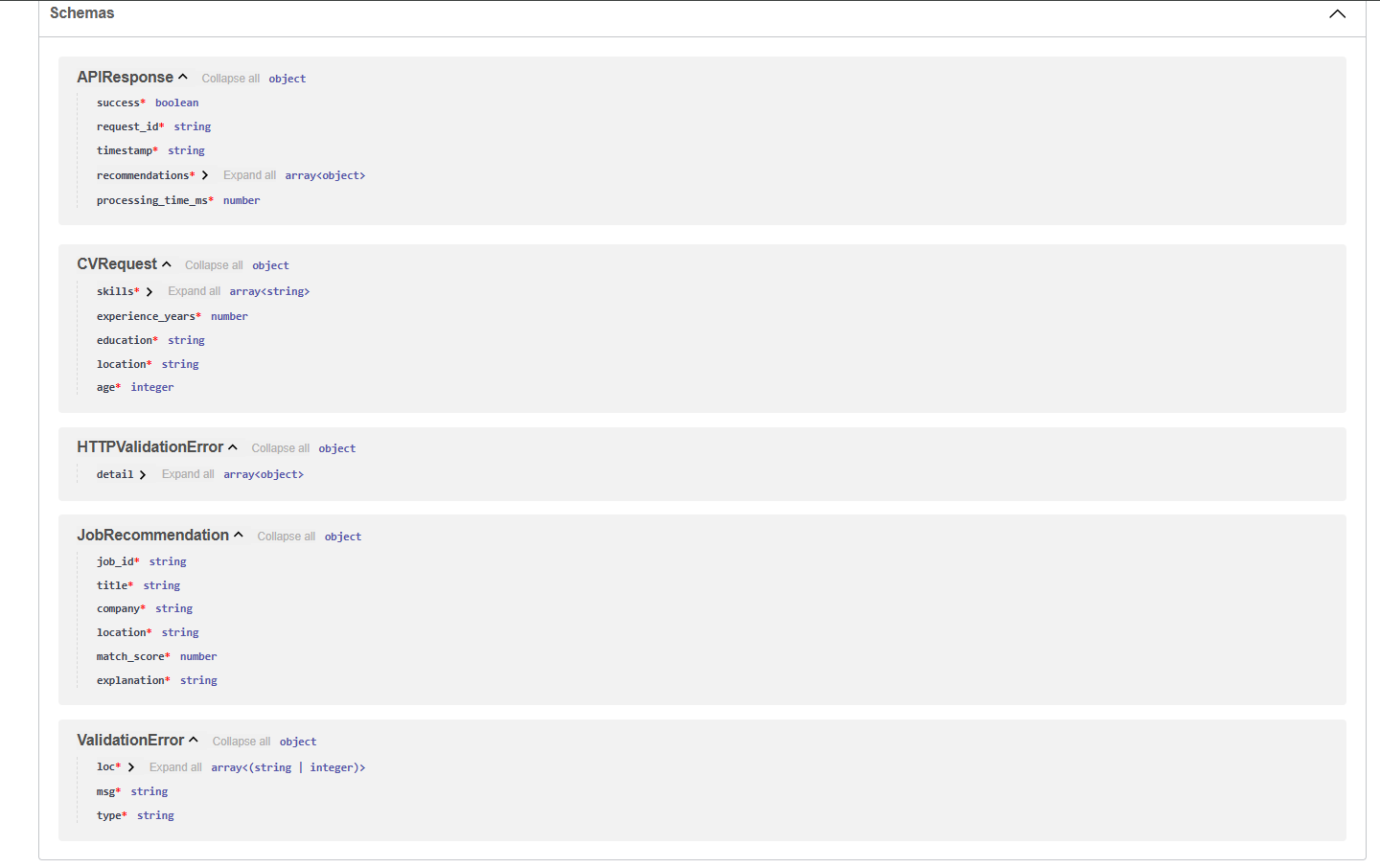
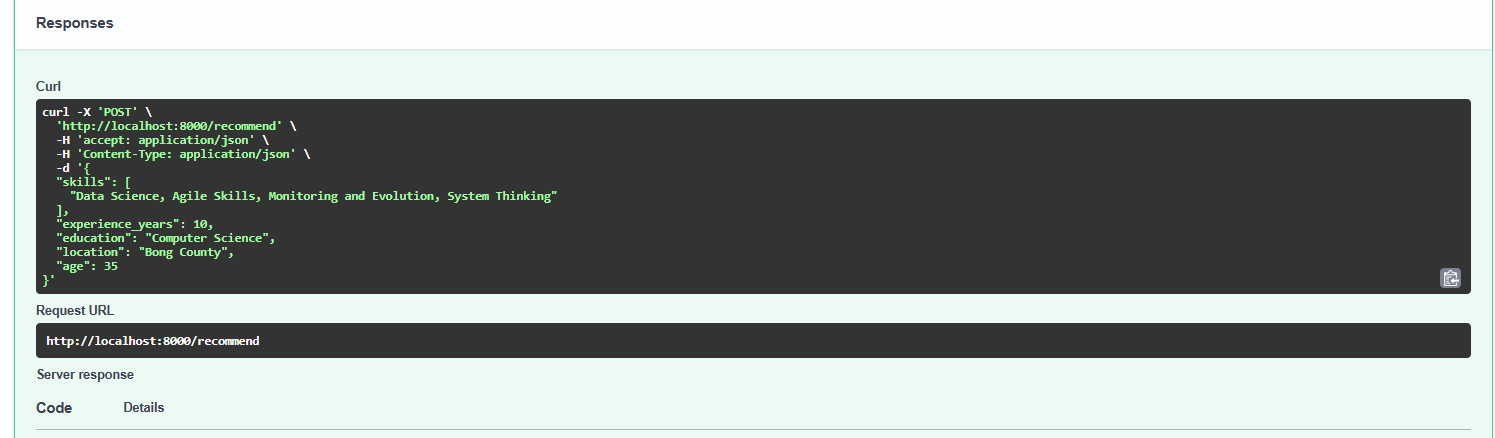
## WorkWise Liberia API



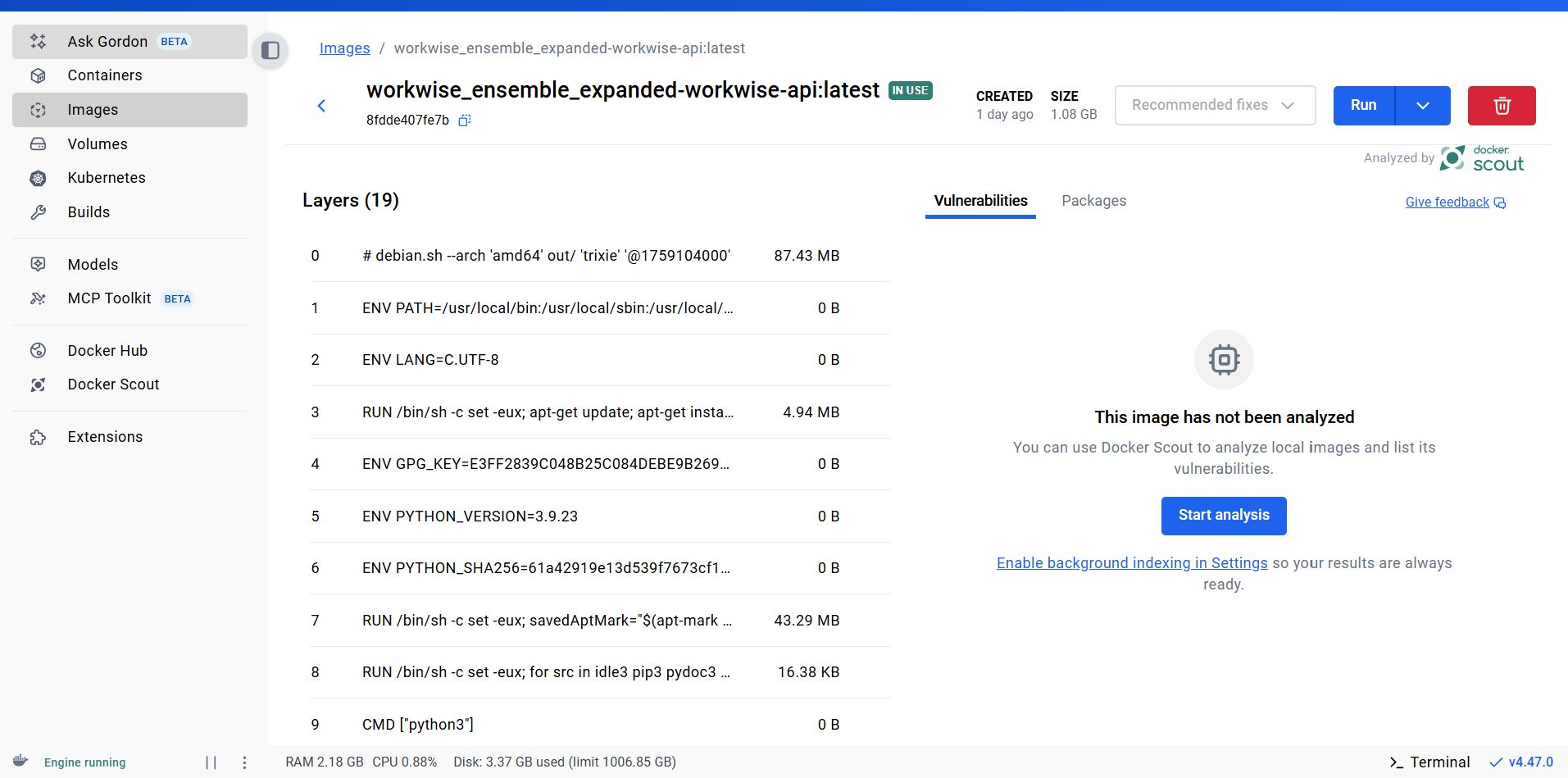


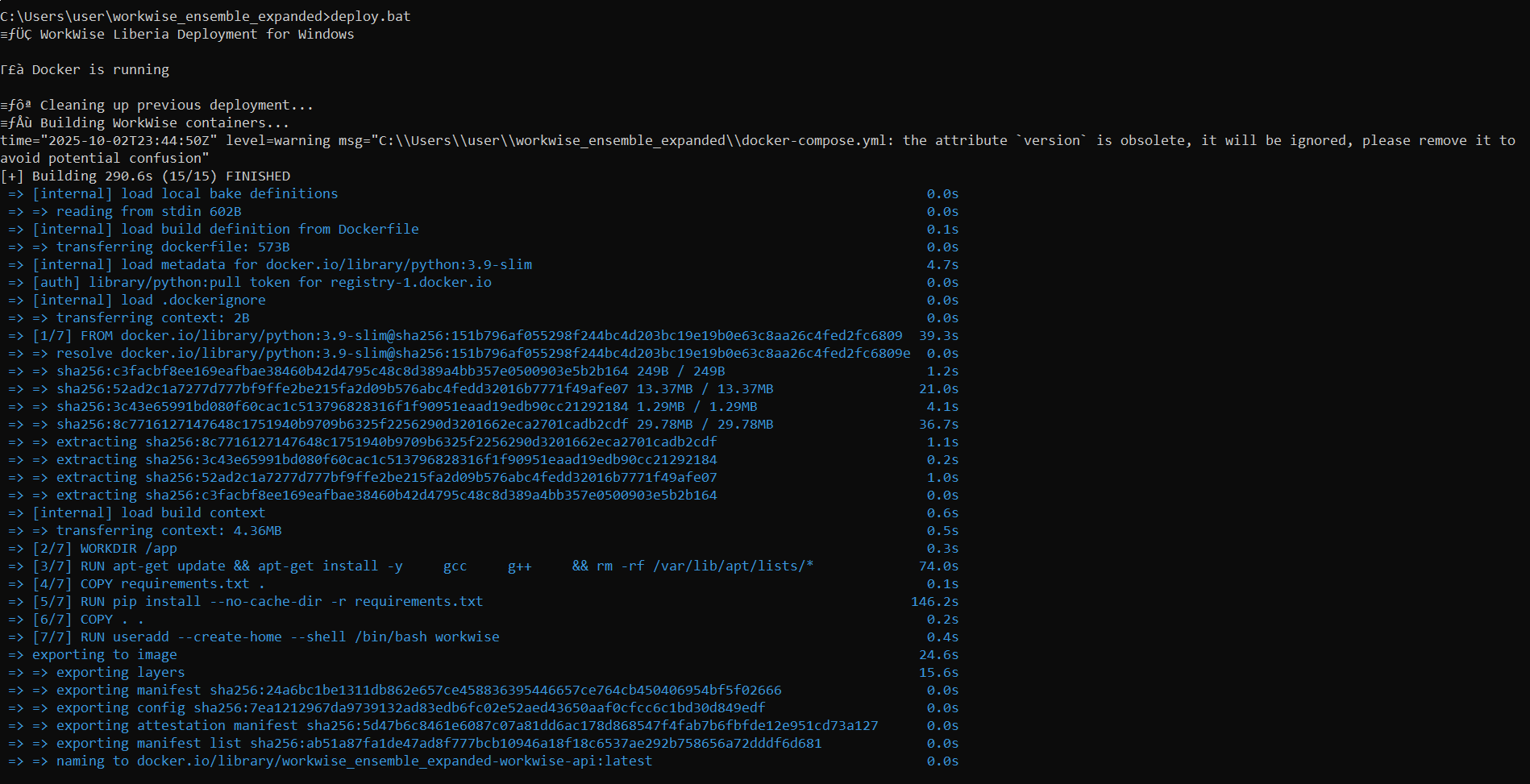






**WorkWise Docker**

****

**WorkWise CommendLine**

