Brainium

AI Development Process for VIRIDI SaaS Invoice Processing Platform with GitLab

Introduction

This document outlines the standardized development process for integrating 13 AI-driven components into the VIRIDI platform, utilizing one Python developer and one React developer. The components, detailed in the *AI Integration for VIRIDI Platform* document , include AI-Driven Auto Transformer, Invoice Fraud Detection, Smart Translation, Chatbot, Smart Statistics, Invoice Discrepancy Warning Flags, Automatic Product Classification, Smart Product Information Management (PIM), Economical/Environmental Purchases, Waste Reduction, ESG Vendor Assessment, Sustainability Reporting, and AI in UX and Smart Feedback Detection. The process leverages an AWS-exclusive technology stack, a GitLab Private Instance for code versioning and CI/CD, and aligns with VIRIDI's sustainability goals (e.g., GHG emissions tracking, ESG compliance). It defines stages, developer responsibilities, and benefits, with a Mermaid diagram for clarity, omitting component-specific workflows as they are covered in the referenced document.

Technology Stack

CATEGORY	TECHNOLOGY	PURPOSE
Programming Languages	Python 3.13	AI model development, data processing, backend logic (Python developer).
	JavaScript (React)	Front-end development for Admin Panel (React developer).
AI/ML Frameworks	TensorFlow/PyTorch	Training fraud, discrepancy, classification, and ESG models (Python).
	Hugging Face Transformers	NLP for Translation, Chatbot, PIM, ESG, and UX feedback (Python).
	Pandas/NumPy	Data manipulation for Statistics, Discrepancy, Waste, and Purchases (Python).
	Matplotlib/Seaborn	Visualization for Smart Statistics and Sustainability Reporting (Python).
Front-End	React, Tailwind CSS	Admin Panel UI for Product Detail Cards, visualizations, feedback (React).
API Integration	FastAPI	RESTful APIs for all AI components (Python).
	OpenAPI/Swagger	Documenting invoice and vendor APIs (Python).

CATEGORY	TECHNOLOGY	PURPOSE
Databases	Amazon RDS (PostgreSQL)	Storing invoice, order, classification, ESG, and feedback data (Python).
	Amazon ElastiCache (Redis)	Caching API responses, chatbot, discrepancy, and UX data (Python).
	Amazon OpenSearch Service (Vector DB)	Storing embeddings for Translation, Chatbot, PIM, ESG, and UX (Python).
Cloud Infrastructure	AWS EC2/AWS Lambda	Hosting AI models and scaling compute (Python).
	Amazon S3	Storing synthetic data, reports, visualizations, and archives (Python).
Other Tools	Docker	Containerizing AI services on AWS ECS (Python).
	AWS CloudWatch	Monitoring performance and logging errors (Python).
	GitLab Private Instance	Code versioning and CI/CD integration (Python, React).

Development Process

The development process is iterative, following Agile principles, and consists of five stages: Design, Development, Testing, Deployment, and Monitoring/Refinement. Each stage assigns specific responsibilities to the Python and React developers, ensuring efficient collaboration and integration with AWS services and the GitLab Private Instance for CI/CD.

1. Design

- **Objective**: Define requirements and architecture for each AI component, aligning with sustainability goals (e.g., CO2e tracking for PIM, ESG metrics for Vendor Assessment).
- **Python Developer**:
    - Define backend requirements (e.g., model specs for Fraud Detection, API endpoints for Auto Transformer).
    - Design data schemas for Amazon RDS and embeddings for Amazon OpenSearch Service.
    - Specify FastAPI endpoints for integration with front-end and external APIs.
- **React Developer**:
    - Design UI components for the VIRIDI Admin Panel (e.g., Product Detail Cards for PIM, visualization dashboards for Smart Statistics).
    - Define API consumption requirements for FastAPI endpoints.
- **Outputs**: Requirement documents, API specifications (OpenAPI/Swagger), UI wireframes.

2. Development

- **Objective**: Implement backend AI models, APIs, and front-end interfaces, committing code to the GitLab Private Instance.

- **Python Developer**:
  - Develop AI models using TensorFlow/PyTorch (e.g., fraud detection, ESG scoring) and Hugging Face Transformers (e.g., Translation, Chatbot).
  - Implement FastAPI endpoints for data processing (e.g., Pandas for Waste Reduction) and visualization (Matplotlib/Seaborn for Sustainability Reporting).
  - Containerize services using Docker for AWS ECS deployment.
  - Commit backend code to GitLab, triggering CI/CD pipelines.
- **React Developer**:
  - Build React components with Tailwind CSS for Admin Panel features (e.g., interactive charts for Smart Statistics, feedback forms for UX Detection).
  - Integrate with FastAPI endpoints for data retrieval and display.
  - Commit front-end code to GitLab, ensuring CI/CD compatibility.
- **Outputs**: Python AI models, FastAPI services, React components, Docker images.

3. Testing

- **Objective**: Validate functionality, performance, and sustainability metrics in a controlled environment.
- **Python Developer**:
  - Write unit and integration tests for AI models and APIs using pytest, validating against synthetic (Phase 1) or real data (Phases 2-3).
  - Test model accuracy (e.g., fraud detection precision, ESG scoring) and API response times, logging results in Amazon RDS.
  - Run tests in Docker sandbox on AWS ECS via GitLab CI/CD pipelines.
- **React Developer**:
  - Write unit tests for React components using Jest, ensuring UI responsiveness and data rendering (e.g., Product Detail Cards).
  - Test API integration with FastAPI endpoints, verifying data display and user interactions.
  - Execute tests via GitLab CI/CD pipelines.
- **Outputs**: Test reports, validated models, and UI components.

4. Deployment

- **Objective**: Deploy components to production on AWS ECS, integrating with AWS services.
- **Python Developer**:
  - Deploy Dockerized FastAPI services and AI models to AWS ECS, configuring Amazon RDS, OpenSearch, S3, and ElastiCache.
  - Set up AWS CloudWatch for monitoring API performance and model errors.
  - Ensure data storage (e.g., invoices, ESG scores) and embeddings (e.g., for Chatbot, PIM) are correctly integrated.
- **React Developer**:
  - Deploy React Admin Panel to AWS ECS or AWS Amplify, ensuring seamless integration with FastAPI endpoints.
  - Configure caching with Amazon ElastiCache for UI performance (e.g., cached visualizations).
- **Outputs**: Deployed services, production-ready Admin Panel.

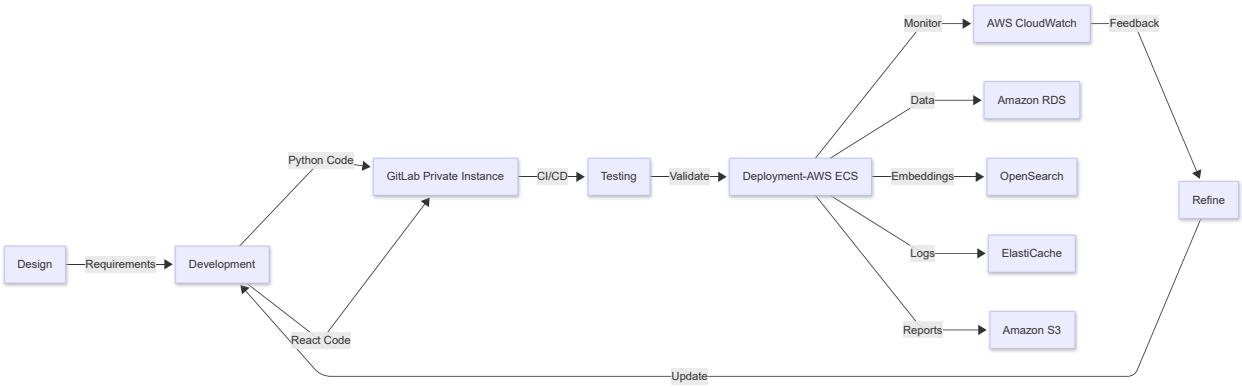
5. Monitoring and Refinement

- **Objective**: Monitor performance, collect feedback, and refine models/UI to improve accuracy and usability.
- **Python Developer**:
  - Monitor API and model performance using AWS CloudWatch, logging errors in Amazon ElastiCache.
  - Collect user feedback (e.g., model corrections for Translation, ESG flags) in Amazon RDS.
  - Retrain AI models (e.g., TensorFlow for Fraud Detection, Transformers for UX Feedback) based on feedback, redeploying via GitLab CI/CD.
- **React Developer**:
  - Monitor UI performance and user interactions via AWS CloudWatch and Amazon ElastiCache.
  - Update React components based on user feedback (e.g., improved feedback forms for UX Detection).
  - Redeploy UI updates via GitLab CI/CD.
- **Outputs**: Performance logs, refined models, updated UI components.

Benefits

- **Efficiency**: Two developers streamline backend and front-end tasks, leveraging GitLab CI/CD for automation.
- **Scalability**: AWS-exclusive stack (ECS, RDS, OpenSearch) supports growing data and user demands.
- **Sustainability**: Feedback loops refine ESG-focused components (e.g., Sustainability Reporting, Economical Purchases).
- **Usability**: React Admin Panel enhances user interaction with AI-driven features (e.g., Product Detail Cards, visualizations).
- **Consistency**: Standardized process ensures uniform development across all 13 components.

Flow Diagram



Conclusion

The development process for the 13 AI components of the VIRIDI platform leverages one Python developer for backend AI models and APIs and one React developer for front-end Admin Panel features. The five-stage process—Design, Development, Testing, Deployment, and Monitoring/Refinement—ensures efficient collaboration, with GitLab Private Instance CI/CD automating testing and deployment to AWS ECS. The process supports VIRIDI's goals of automation, sustainability, and ESG compliance, with feedback loops refining models and UI. The Mermaid diagram provides a clear visualization of the development lifecycle, complementing the phased integration plan.