

# IssueOps

*Continuous AI for GitHub Issues.*

MID-TERM REVIEW

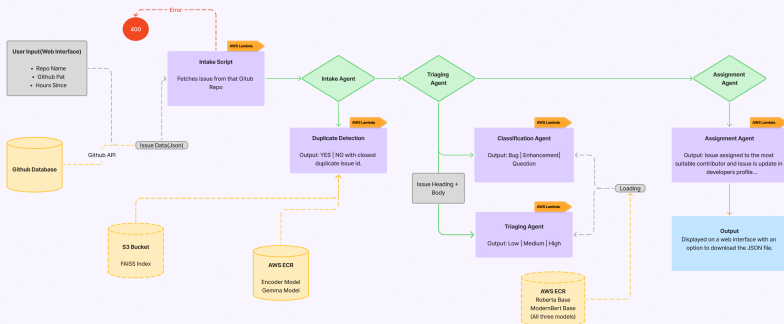
Team: VaaS



DS 252 : Introduction to Cloud Computing

October 10, 2025

## Workflow Diagram



## Intake Script

### 1. Initialization

- Receives repo name & GitHub PAT
- Configures time window (default: 1hr)

### 2. Issue Collection

- Queries GitHub API with time filter
- Fetches recent issues and PRs
- Distinguishes Issues vs Pull Requests

### 3. Classification Pipeline

- Heavy Type Classifier
- Priority Classifier (P0-P3)
- Assignee Recommender
- Aggregates results with scores

## Duplicate Detection Agent

### 1. Preprocessing Pipeline

- Extracts title and body from JSON
- Cleans whitespace and special chars
- Normalizes text formatting

### 2. Embedding Generation

- HuggingFace Google embeddings
- Converts text to dense vectors

### 3. FAISS Index Search

- S3-backed FAISS index storage
- Cosine similarity search
- Retrieves top-K candidates

### 4. Cross-Encoder Validation

- Threshold-based filtering
- Reranks using Cross-Encoder
- Assigns similarity scores (0-1)
- Returns YES/NO decision

## Classification Agent

**RoBERTa-base** (125M params)  
12 layers | 768 hidden | 12 heads  
Context: 512 tokens

**Fine-tuning:** 55K GitHub issues  
**Labels:** Bug | Enhancement | Question

### Performance Metrics:

Category	Precision	Recall	F1	H-Loss
Bug	81%	81%	81%	0.14
Enhancement	78%	72%	74%	0.15
Question	79%	81%	80%	0.15
<b>Macro-Avg</b>	<b>79%</b>	<b>78%</b>	<b>78%</b>	<b>0.15</b>

Training: 48 min (Google Colab T4 GPU )

## Priority Agent

**ModernBERT-base** (149M params)  
22 layers | 768 hidden | 12 heads  
Context: 8192 tokens

**Architecture:** Ensemble of 3 Models  
(Low vs. Not-Low | Medium vs. Not-Medium  
| High vs. Not-High)

**Fine-tuning:** GitHub & Jira

### Performance Metrics:

Priority	AUC	Precision	Acc	F1
Low	0.90	0.81	81%	0.81
Medium	0.92	0.83	83%	0.83
High	0.91	0.82	82%	0.82
<b>Macro-Avg</b>	<b>0.91</b>	<b>0.82</b>	<b>82%</b>	<b>0.82</b>

Binary classification outperforms multiclass

<sup>1</sup>"Automatic Issue Classifier: A Transfer Learning Framework for Classifying Issue Reports" arXiv 2022

<sup>2</sup>"Enhancing Task Prioritization in Software Development Issues Tracking System" google scholar 2025

# Assignment Agent

- ① **Developer Profiles**  
Contains developer names, expertise keywords, and an initially empty list for issues.
- ② **New Issue Created**  
GitHub forwards the new issue's data (title, body, labels) to a server using a webhook.
- ③ **Text-to-Numbers**  
The issue's is converted into a numerical vector so the model can understand it. This is done using Sklearn's TfidfVectorizer.
- ④ **Prediction**  
A pre-trained MultinomialNB classifier calculates the probability for each developer and selects the one with the highest score.
- ⑤ **Assign and Update**  
The issue is officially assigned to the predicted developer on GitHub, and the local developer profile is updated.

# AI Tools & Assistants Used by Team

## Vinay

### Tools Used:

- Cursor IDE
- Perplexity
- Google Gemini
- Warp Agentic Terminal

## Ashwin K M

### Tools Used:

- Claude Sonnet
- Perplexity AI
- Gemini 2.5 Pro
- ChatGPT-5

## Sai Harsh

### Tools Used:

- Perplexity AI
- Gemini 2.5 Pro
- ChatGPT-5
- Github Copilot

## Anmol Gill

### Tools Used:

- Gemini 2.5 Pro
- ChatGPT-5
- Github Copilot
- Perplexity AI

## Challenges Faced

- **Context Limitations:** AI struggled with understanding complex project architecture and proposes complex fixes to bugs
- **Code Quality Issues:** Generated code required significant review and refactoring (30% error rate)
- **Integration Challenges:** Difficulty adapting AI outputs to existing codebase standards and team conventions.

# Challenges Faced

## 1. Data Quality & Model Performance

- Fetching results from top repositories of popular programming languages introduced popularity bias in our dataset
- Data drift in issue patterns requiring continuous model retraining
- Duplicate detection accuracy challenges with semantic similarity
- Balancing precision vs. recall for spam and duplicate filtering

## 2. Infrastructure & Deployment

- ML models were very large requiring lengthy downloads from HuggingFace and ECR pushes
- Docker Desktop compatibility issues on Windows, requiring WSL-based builds
- AI tools persistently suggested CloudBuild instead of Docker workflows
- Free-tier cloud deployment imposed significant resource constraints

# Plan for Completion

## Agent Development

- Build remaining Intake, Triage, and Assignment Agents
- Start building Notification Agent
- Fine-tune embeddings for improved accuracy
- Conduct load testing and performance optimization

## Infrastructure & Deployment

- Deploy application on GitHub Marketplace
- Use Terraform for infrastructure management
- Implement failover mechanisms and feature engineering

## Final Deliverables

- Record comprehensive demo video
- Finalize project report with performance metrics
- Prepare and deliver final presentation
- Submit complete codebase and documentation

## Success Criteria

- Reduce triage time from 2+ days to **<4 hours**
- Achieve **60%** reduction in notification noise
- Handle **100+ issues/hour**
- Deploy production-ready system on AWS