IssueOps

Continuous AI for GitHub Issues.

“We already burn countless unpaid hours triaging issues and dealing with duplicates instead of writing code.”

— Mike McQuaid, Lead Maintainer of Homebrew

Team: VaaS

DS 252 : Introduction to Cloud Computing

September 17, 2025

The Problem

Developer productivity is being drained by GitHub Issues.

7.3 hours lost per week
per developer to triage, duplicates, and notifications

3+ day delays
before issues are even categorized or routed

59% of maintainers
consider quitting due to burnout and overhead

Market Opportunity

- **150M+** developers on GitHub
- **420M+** repositories hosted
- **$37.45B** automation market by 2030
- **$21.6M** TAM (GitHub-specific)
- **9.52%** CAGR growth rate

Revenue Potential

- Team Subscriptions: **$14.4M/year**
- Individual Freemium: **$6.0M/year**
- Enterprise Plans: **$1.2M/year**
- **Total TAM:** $21.6M annually

Our Solution

IssueOps Agents automate the GitHub issue lifecycle end-to-end:
Intake → *Triage* → *Assignment* → *Notification*.

Our Core Ideas

- **Connected Workflow:** track each issue as it moves across agents; preserve context and state.
- **Smart Triage:** combine AI understanding with lightweight rules for consistent decisions.
- **Elastic Scaling:** containerized services that scale with repo activity; no manual ops burden.

Key Value Proposition

- Cut triage delays from days to hours; surface priorities fast.
- Reduce notification noise with batching, rate limits, and targeted alerts.
- Clear ownership and next actions through labels, assignees, and linked duplicates.

How it Works

End-to-End Flow

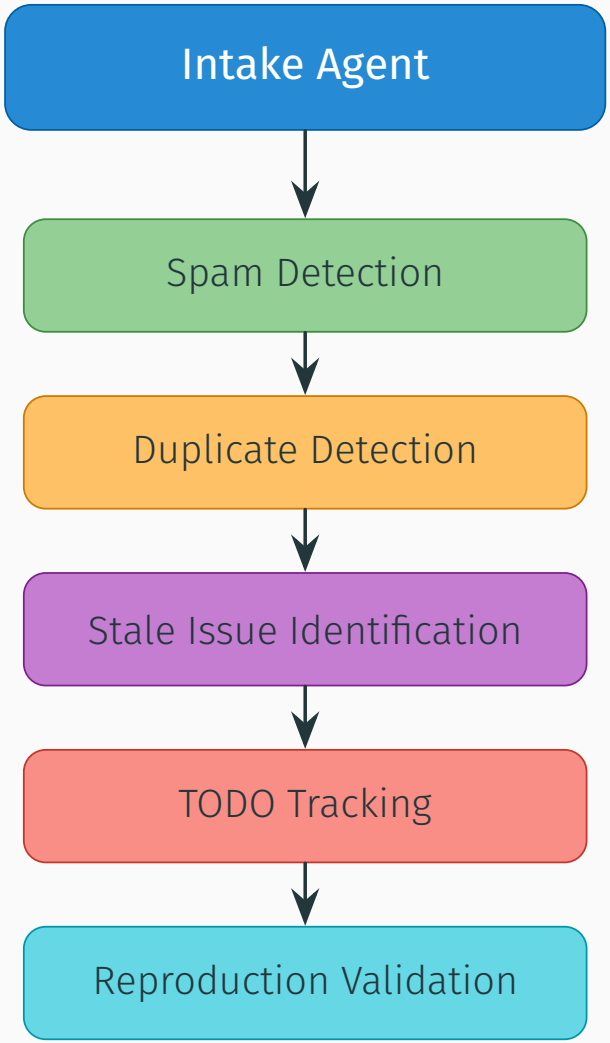
- When a user installs the app we do a one time scan of the repo and send the file to our server, which is stores as context for the future.
- When a new issue is created on GitHub, a webhook sends the issue details to our service.
- Our service receives the data, processes it, and determines the correct label using AI.
- The label is applied back to the issue on GitHub through the API.

Data Movement

- **From:** GitHub (issue title, description, metadata).
- **To:** Our server for analysis and decision-making (might send data as lambda functions to reduce load).
- **Back to:** GitHub with updated labels and status.

$$\text{GitHub Issue} \Rightarrow \text{Webhook} \Rightarrow \text{Our Service} \Rightarrow \text{GitHub (Updated)}$$

Intake Agent

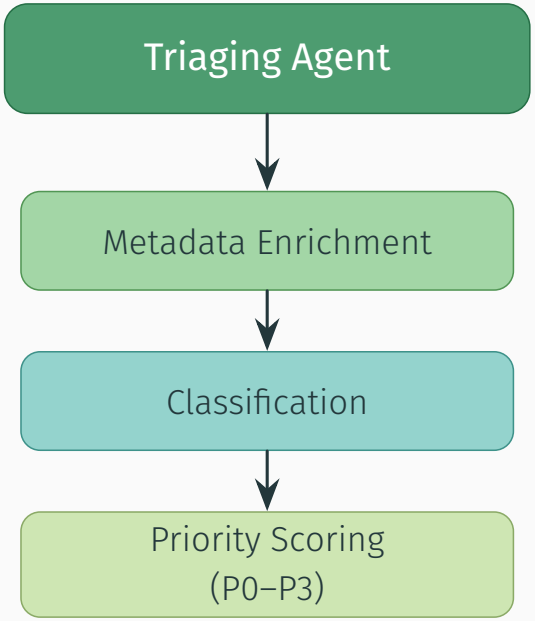


The Intake Agent is the first gatekeeper. It cleans, organizes, and enriches every incoming issue so only actionable work reaches developers.

Key functions

1. **Spam Shield:** Blocks bots, junk, and irrelevant noise before it clutters the repo.
2. **Duplicate Finder:** Detects similar issues and links them into one clear thread.
3. **Stale Detector:** Flags outdated or abandoned issues, keeping the backlog lean.
4. **TODO Tracker:** Scans code for TODO notes and turns them into GitHub issues automatically.
5. **Repo Validator:** Ensures bug reports include clear steps for reliable reproduction.

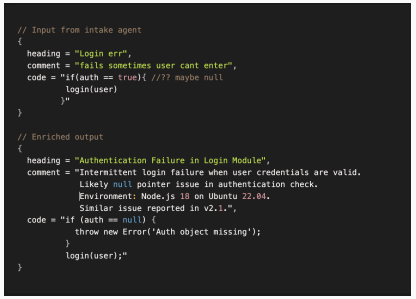
Triaging Agent



The Triage Agent first enriches incoming issues with relevant metadata, to build a complete picture. This enriched context allows the agent to accurately classify the issue and set a precise priority

Key functions

1. **Metadata Enrichment:** Adds context like environment, dependencies, severity, and history to speed up fixes.

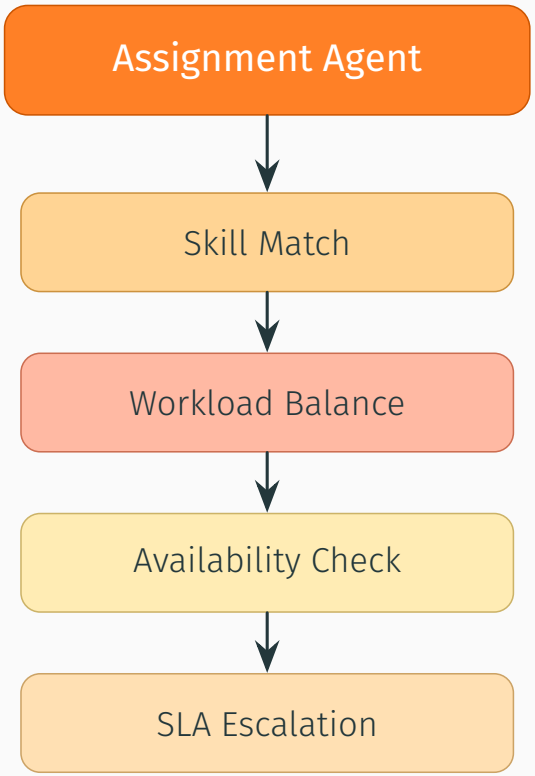


```
// Input from intake agent
{
  heading = "Login err",
  comment = "fails sometimes user cant enter",
  code = "if(auth == true){ //?? maybe null
          login(user)
         }"
}

// Enriched output
{
  heading = "Authentication Failure in Login Module",
  comment = "Intermittent login failure when user credentials are valid.
             Likely null pointer issue in authentication check.
             Environment: Node.js 18 on Ubuntu 22.04.
             Similar issue reported in v2.1.",
  code = "if (auth == null) {
            throw new Error('Auth object missing');
          }
          login(user);"
}
```

2. **Classification:** AI + rules map issues to components and labels instantly.
3. **Priority Scoring:** Assigns P0–P3 based on impact and urgency.

Assignment Agent

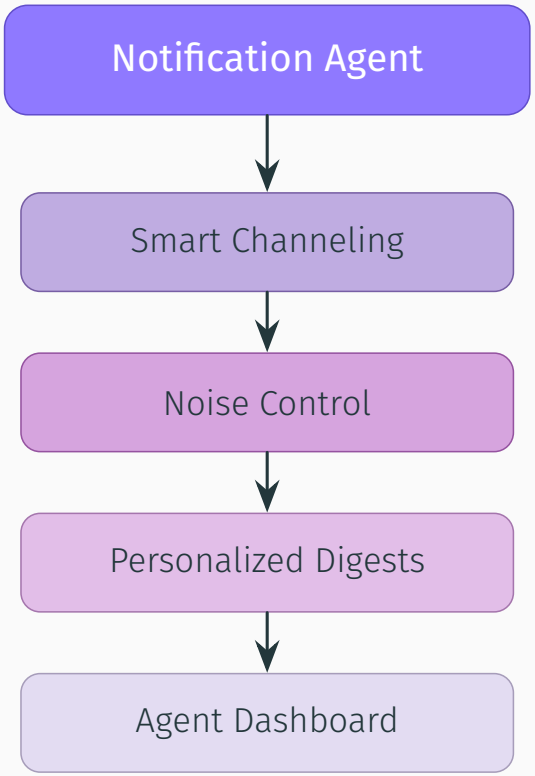


The Assignment Agent maps triaged issues to the best person or team, respecting skill, availability, and SLA constraints.

Key functions

1. **Skill Match:** Maps labels and components to contributors' expertise so the right person gets the job.
2. **Workload Balance:** Distributes assignments to avoid overload and keep throughput steady.
3. **Availability Check:** Respects calendars, on-call status, and recent activity when choosing assignees.
4. **SLA Escalation:** Automatically escalates or reassigns when timeouts or priority SLAs are breached.

Notification Agent



The Notification Agent delivers alerts with precision and gives teams a live dashboard to track who got which issue, when, and how it's progressing.

Key functions

1. **Smart Channeling:** Routes updates through Slack, Email, or GitHub based on context.
2. **Noise Control:** Batches and rate-limits alerts to prevent fatigue.
3. **Personalized Digests:** Curates updates and escalates urgent items instantly.
4. **Agent Dashboard:** Shows issue ownership, notification history, and response times in real time.

Workflow

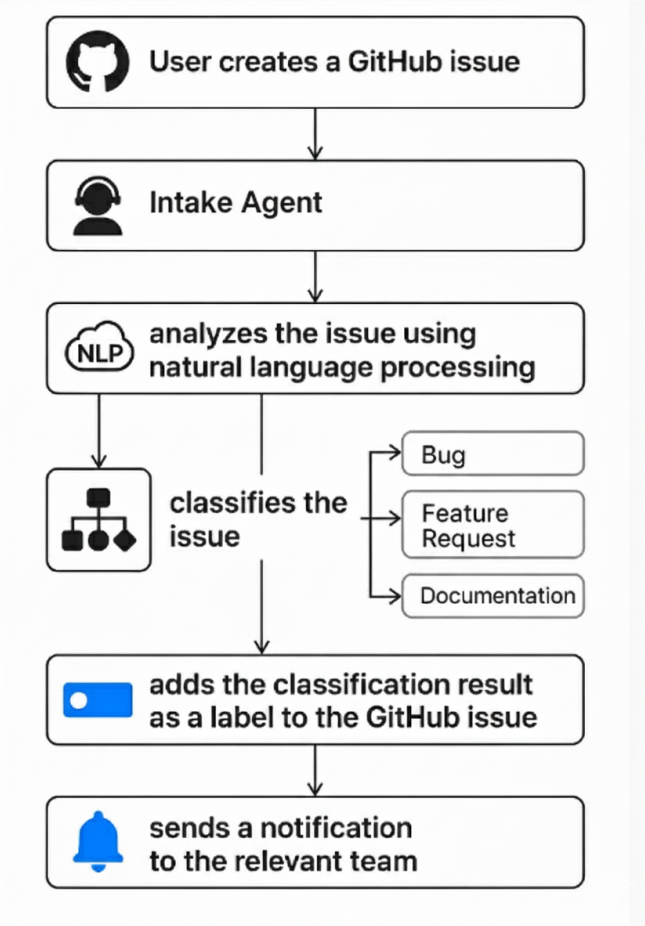


Figure 1: Workflow

Tech Stack

Core Platform

- GitHub App (repo-scoped permissions)
- Python + Flask for webhook handling
- Runs in a Docker container (local or cloud)
- Optional: simple cron for retries

Data & Config

- Config in YAML/JSON (labels, owners)
- Optional SQLite/Postgres for logs
- Secrets via .env or GitHub Actions

AI & Logic

- Zero-shot classification via hosted API
- Lightweight rules for mapping predictions
- Confidence threshold for safe automation

Integrations & DevOps

- GitHub REST API for labels/updates
- Optional Slack/Email notifications
- ngrok for local webhook testing
- CI/CD with GitHub Actions
- Basic logs + health check endpoint

Goal: Simple, modular, and easy to deploy on free-tier or university VM.

2-Week MVP: What It Does & What We Are Doing

What the MVP Does

- Takes new issues from GitHub and sends them to our server automatically.
- Runs a simple **4-step classification** on each issue: *duplicate?* $\rightarrow$ *issue type (bug/feature/...)* $\rightarrow$ *assign to the right engineer type* $\rightarrow$ *notify based on priority/role*.
- Sends the results back to GitHub: labels applied, owners notified; duplicates handled (per repo policy).

What We Are Doing in These 2 Weeks

- **Initial setup:** perform a one-time full scan of the repo and save key data **encrypted** on our server (or a simple free SQL database) for faster, consistent decisions later.
- **Week 1:** connect GitHub to our service (webhook) and receive issue data reliably.
- **Week 2:** implement classification and apply updates back to GitHub (labels, basic duplicate handling, owner assignment + notifications in a minimal form).

GitHub $\Rightarrow$ Our Server $\Rightarrow$ 4-Step Classification $\Rightarrow$ Labels/Assign/Notify $\Rightarrow$ GitHub (Updated)

Project Timeline and Assignments

Weeks 1-2: Foundational Agent Development

Ashwin: Build Intake (Core). | **Vinay:** Build Triage and Intake (TODO). | **Anmol:** Design Assignment. | **Sai Harsh:** Build Intake (Repro Validation).

↓

Weeks 3-4: Full Agent Suite Development

Ashwin: Refine Intake. | **Vinay:** Refine Triage. | **Anmol:** Build Assignment Agent. | **Sai Harsh:** Build Notification Agent.

↓

Weeks 5-6: Full Pipeline Integration

Ashwin: Integrate Intake. | **Vinay:** Integrate Triage. | **Anmol:** Integrate Assignment. | **Sai Harsh:** Integrate Notifications.

↓

Weeks 7-8: Verification and Documentation

Ashwin: Document Intake | **Vinay:** Document Triage | **Anmol:** Document Assignment | **Sai Harsh:** Verification & Document Notifications