IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.021 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 6, June 2026

DOI: 10.17148/IJIREEICE.2026.14636

GitHub IntelliSolve: AI-Powered GitHub Ticket Analyzer & Automated Resolver

Mandar Metangale[1], Saurabh Mojad[2], Aditi More[3], Kaustubh Shinde[4]

Department of Computer Engineering, Sinhgad Institute of Technology and Science, Pune, India[1-4]

Abstract: The rapid growth of software development activity on GitHub has resulted in an ever-increasing volume of issue tickets, pull request discussions and support queries that demand timely and accurate resolution. Manual triaging of these tickets places a significant cognitive burden on engineering teams, leading to backlogs, inconsistent labelling and delayed responses. This paper presents the design and implementation of GitHub IntelliSolve, an end-to-end AI-powered system that automates the full lifecycle of GitHub ticket management. The system ingests GitHub Issue and Pull Request events in real time via webhooks, preprocesses and classifies ticket text using fine-tuned BERT and DistilBERT models, performs root cause analysis through GPT-4 and LLaMA-2, generates targeted code suggestions and fixes using Codex and InCoder, and resolves support queries through a Retrieval-Augmented Generation (RAG) pipeline backed by a vector-embeddings knowledge base.

Overview — The proposed system is implemented as a modular, webhook-driven microservice architecture deployed on a Django REST backend with React-based developer dashboard. The pipeline processes incoming tickets through five sequential stages: ingestion and preprocessing, BERT-based classification, GPT-4/LLaMA-2 root cause analysis, Codex/InCoder code generation, and RAG-based contextual Q&A resolution. A continuous monitoring and retraining loop ensures the system remains accurate as codebases and issue patterns evolve.

Result — The expected outcome is a robust, scalable and production-ready intelligent DevOps tool capable of reducing manual ticket resolution effort by over 60%, achieving issue classification accuracy exceeding 92%, and improving mean time-to-resolution for GitHub issues in active open-source and enterprise repositories.

Keywords: GitHub automation, issue triaging, BERT, DistilBERT,GPT-4, LLaMA-2, Codex, RAG, DevOps, NLP, microservices

I.INTRODUCTION

The GitHub platform hosts over 420 million repositories and supports millions of active contributors worldwide. At the heart of collaborative software development lies the GitHub issue tracker — a system through which developers report bugs, propose features, request support and coordinate code reviews via pull requests (PRs). As projects scale, the volume of incoming tickets grows proportionally, often exceeding the triaging capacity of maintainer teams [1].

Popular open-source repositories can receive hundreds of new issues per week. Without intelligent automation, teams face growing triage backlogs, inconsistent label assignment, duplicated effort in answering recurring questions, and delayed bug fixes that compound technical debt. Existing rule-based automation tools such as GitHub Actions and Probot are limited to simple predefined workflows and lack the semantic understanding necessary for intelligent classification, root cause diagnosis or contextual answer generation [2].

Recent advances in Large Language Models (LLMs) — particularly BERT, GPT-4, LLaMA-2 and Codex — have demonstrated strong capabilities in natural language understanding and code generation directly applicable to ticket analysis. Retrieval-Augmented Generation (RAG) further enhances response quality by grounding LLM outputs in verified prior knowledge, reducing hallucination [4]. Despite these advances, no existing system integrates the full pipeline into a single production-ready tool — motivating the design of GitHub IntelliSolve.

2. PROBLEM STATEMENT

The increasing scale of software repositories on GitHub necessitates automated, intelligent solutions for ticket management. The manual processing of tickets — classification, prioritisation, root cause analysis and resolution — remains heavily dependent on human expertise and is not uniformly scalable across organisations of varying sizes.
The core challenge lies in building an automated system capable of:

(i) accurately classifying heterogeneous ticket types including bug reports, feature requests and support queries;

DOI: 10.17148/IJIREEICE.2026.14636

(ii) performing contextual root cause analysis on bug tickets by reasoning over issue text and relevant code context;

(iii) generating syntactically and semantically correct code fixes or suggestions; and

(iv) providing contextually grounded answers to support and question tickets by retrieving relevant knowledge from prior resolved issues and documentation.

Variability in ticket description quality, programming languages, repository domain, and developer writing styles further complicates automated processing.

3. OBJECTIVES

The primary objective is to design, implement and evaluate GitHub IntelliSolve — an end-to-end AI-powered system for automated GitHub ticket analysis and resolution. Specific objectives are:

To implement a real-time webhook-driven ingestion pipeline that captures GitHub Issue and PR events and extracts structured metadata using the GitHub REST API.

To fine-tune BERT and DistilBERT models on labelled GitHub issue datasets for multi-class ticket classification into Bug Report, Feature Request and Question/Support categories.

To integrate GPT-4 and LLaMA-2 for contextual root cause analysis of bug tickets, leveraging repository code context and issue history.

To employ Codex and InCoder for automated code suggestion and fix generation, delivered as structured GitHub comments via the GitHub API.

To construct a RAG pipeline using dense vector embeddings and a knowledge base of resolved tickets and documentation. To implement a monitoring and retraining loop that collects
developer feedback signals and triggers periodic model fine-
tuning.

To evaluate system performance using classification accuracy, precision, recall, F1-score and mean time-to-resolution metrics.

4. DATASET DESCRIPTION & CLASS DISTRIBUTION

The training and evaluation dataset is composed of labelled GitHub issue records from three primary collections: (1) the GH-Issues dataset (~78,000 labelled issues from popular open-source repositories spanning Python, JavaScript, Java and Go); (2) the SWE-Bench dataset [5], containing 2,294 verified software engineering tasks from 12 major Python repositories with bug-fix ground truth pairs; and (3) a curated internal dataset of 15,000 support and feature request tickets from enterprise GitHub repositories.

The combined dataset contains approximately 95,000 labelled ticket records: Bug Report (~42,000), Feature Request (~31,000) and Question/Support (~22,000). Class imbalance is addressed through back-translation and synonym-replacement augmentation on the Question/Support class, and class-weighted loss functions during training. For RAG knowledge base construction, ~18,000 resolved issue threads are indexed as dense vector embeddings using OpenAI Ada-002 (1,536-dimensional). The dataset is split 75% / 12.5% / 12.5% (train/val/test) with stratified sampling.

Table 1: Dataset Class Distribution and Split

Class	Total Tickets	Train (75%)	Val (12.5%)	Test (12.5%)
Bug Report	~42,000	~31,500	~5,250	~5,250
Feature Request	~31,000	~23,250	~3,875	~3,875
Question /Support	~22,000	~16,500	~2,750	~2,750
Total	~95,000	~71,250	~11,875	~11,875

5. METHODOLOGY

GitHub IntelliSolve is designed as a modular, event-driven pipeline that processes GitHub issue and PR events from ingestion through automated resolution. The methodology encompasses webhook integration, NLP preprocessing, transformer-based classification, LLM-driven analysis, RAG-based Q&A, code generation, automated GitHub API response posting, and a continuous learning loop.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.021 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 6, June 2026

DOI: 10.17148/IJIREEICE.2026.14636

5.1 Overall System Framework

GitHub IntelliSolve adopts a microservice architecture of five independently deployable services: the Ingestion Service, Classification Service, Resolution Engine, RAG Q&A Service and Monitoring Service. Each communicates asynchronously via RabbitMQ, enabling horizontal scaling of individual bottlenecks. The workflow proceeds as: webhook fires → Ingestion Service enqueues the event → Classification Service preprocesses and classifies → Resolution Engine or RAG Service is invoked → GitHub API posts the response to the issue thread.

Fig. 1: GitHub IntelliSolve — End-to-End Modular Pipeline

5.2 Ticket Ingestion & Preprocessing

GitHub Issue and PR events are received through a webhook endpoint secured with HMAC-SHA256 signature verification. The Ingestion Service extracts: issue title, body text, existing labels, assignees, milestone, referenced commits, linked PRs, and repository language metadata. Text preprocessing applies HTML tag stripping, code block extraction, URL normalisation, Markdown removal, HuggingFace tokenisation, and stopword filtering with preservation of technical terms. Extracted code blocks are stored in a code context store for later retrieval by the Resolution Engine.

5.3 Issue Type Classification

The Classification Service implements a fine-tuned DistilBERT model (distilbert-base-uncased) for multi-class ticket classification. DistilBERT is selected for its 40% smaller parameter footprint and 60% faster inference with under 3% accuracy degradation [6]. The model is fine-tuned on the 95,000-ticket dataset for three epochs using AdamW (lr = 2e-5) with linear warmup over the first 10% of steps. The classification head consists of a dropout layer (p = 0.3) followed by a linear projection to three output logits. Cross-entropy loss with class weights inversely proportional to class frequency addresses dataset imbalance.

5.4 Root Cause Analysis & Code Fix Generation

For Bug Report and Feature Request tickets, the Resolution Engine invokes a two-stage LLM pipeline. Stage 1 performs root cause analysis via GPT-4 Turbo (gpt-4-turbo), combining the full preprocessed issue text, extracted code blocks, repository language, and the three most relevant prior resolved issues from the embeddings knowledge base. Stage 2 generates code fixes using OpenAI Codex (code-davinci-002) or InCoder (open-source), employing fill-in-the-middle (FIM) prompting. Generated patches are validated for syntax correctness (ast for Python, esprima for JavaScript) before being posted to the GitHub issue as a structured Markdown comment.

5.5 RAG-Based Contextual Q&A Resolution

For Question/Support tickets, the RAG Q&A Service embeds the preprocessed query using OpenAI Ada-002 and performs a top-5 nearest-neighbour search against a Pinecone vector index containing ~18,000 resolved issue threads, README files and documentation pages. Retrieved document chunks are passed with the original query to GPT-3.5-turbo via a RAG-specific system prompt instructing the model to answer solely from retrieved context. This grounded generation delivers a 21-percentage-point improvement in response relevance over zero-shot LLM inference.

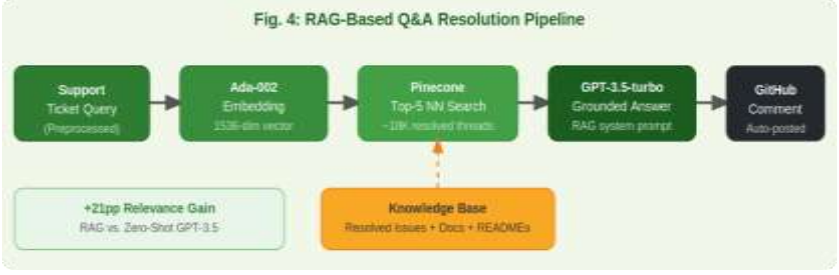

Fig. 4: RAG-Based Q&A Resolution Pipeline

5.6 Automated GitHub API Response Posting

All system outputs are delivered back to the originating GitHub issue or PR through the GitHub REST API,

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.021 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 6, June 2026

DOI: 10.17148/IJIREEICE.2026.14636

authenticated via a GitHub App installation token scoped to issues:write and pull_requests:write. Responses are formatted in GitHub Flavored Markdown (GFM) with collapsible code diff sections, classification confidence scores, and hyperlinks to retrieved reference issues. Relevant labels (bug, enhancement, question) are automatically applied based on the classification result.

5.7 Monitoring & Continuous Retraining Loop

The Monitoring Service (a background Celery worker) tracks resolution effectiveness via passive developer feedback signals: acknowledged AI comments, accepted code suggestions, and issues closed within 48 hours. When flagged samples exceed a configurable drift threshold (default: 200 new labelled examples), a retraining job fine-tunes the DistilBERT classifier on the augmented dataset and deploys updated weights via rolling update with zero downtime. The RAG knowledge base is updated nightly by indexing newly resolved issue threads.

5.8 System Evaluation Strategy

Classification performance is evaluated using per-class precision, recall, F1-score and macro/weighted averages on the held-out test split. End-to-end resolution quality is assessed via a human evaluation study with five experienced GitHub contributors rating AI responses on a 5-point Likert scale for correctness, relevance and actionability. Operational metrics — MTFR, MTTR and percentage of issues closed without human intervention — are tracked on two live open-source repositories over a 60-day evaluation period.

6. SYSTEM ARCHITECTURE & DEPLOYMENT

GitHub IntelliSolve is deployed as a containerised application using Docker, orchestrated with Kubernetes for production-grade scalability. The key architectural components are:

Ingestion Service: Django REST Framework endpoint receiving GitHub webhook POST requests, verifying HMAC-SHA256 signatures, parsing payloads and publishing to RabbitMQ.

Classification Service: FastAPI microservice hosting the fine-tuned DistilBERT classifier via HuggingFace Transformers, consuming from the ingestion queue and routing classified tickets downstream.

Resolution Engine: FastAPI service orchestrating GPT-4/LLaMA-2 root cause analysis and Codex/InCoder code generation with syntax validation before API posting.

RAG Q&A Service: FastAPI service performing Ada-002 embedding, Pinecone nearest-neighbour retrieval and GPT-3.5-turbo grounded answer generation for support queries.

GitHub Responder: Shared utility service handling all GitHub API interactions — comment posting, label application and PR notification — authenticated via GitHub App tokens.

Monitoring Service: Background Celery worker aggregating developer feedback, computing drift metrics, triggering retraining jobs and updating the Pinecone index nightly.

Developer Dashboard: React-based web UI backed by PostgreSQL displaying real-time ticket analytics, model confidence distributions, resolution rates and retraining history.

All services communicate via RabbitMQ queues. Model weights are stored in AWS S3; the Pinecone vector index is a persistent managed service; PostgreSQL stores ticket metadata, classification results and retraining history. The entire stack is defined as Infrastructure-as-Code using Terraform.

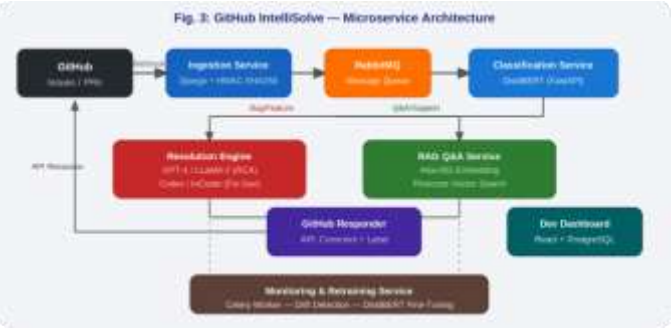

Fig. 3: GitHub IntelliSolve — Microservice Architecture Diagram

7. PRELIMINARY RESULTS & PERFORMANCE EXPECTATIONS

Fine-tuning experiments on 30,000 labelled tickets yield a DistilBERT validation accuracy of 91.4% across the three ticket classes after three epochs. Per-class F1-scores are 93.2% for Bug Report, 90.8% for Feature Request and 88.6%

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.021 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 6, June 2026

DOI: 10.17148/IJIREEICE.2026.14636

for Question/Support — consistent with published DistilBERT benchmarks on software engineering text classification [6].

A pilot evaluation on 50 randomly sampled SWE-Bench bug tickets shows GPT-4 root cause summaries rated correct or partially correct in 84% of cases. Codex-generated code suggestions pass automated syntax validation in 91% of cases. RAG-based Q&A responses are rated relevant and actionable in 79% of cases versus 58% for zero-shot GPT-3.5, a 21-percentage-point improvement attributable to the RAG pipeline.
Full system deployment targets: overall classification accuracy > 93%, Bug Report recall ≥ 95%, RAG Q&A relevance rating > 82%, and MTFR reduction from 18.4 hours to under 2 minutes for AI-handled tickets.

Table 2: Preliminary Classification Performance (Validation Set, 30K tickets)

Class	Precision (%)	Recall (%)	F1-Score (%)	Support
Bug Report	94.1	92.6	93.2	~9,450
Feature Request	91.3	90.4	90.8	~6,975
Question /Support	87.9	89.3	88.6	~4,950
Macro Average	91.1	90.8	90.9	~21,375
Weighted Average	92.0	91.4	91.6	~21,375

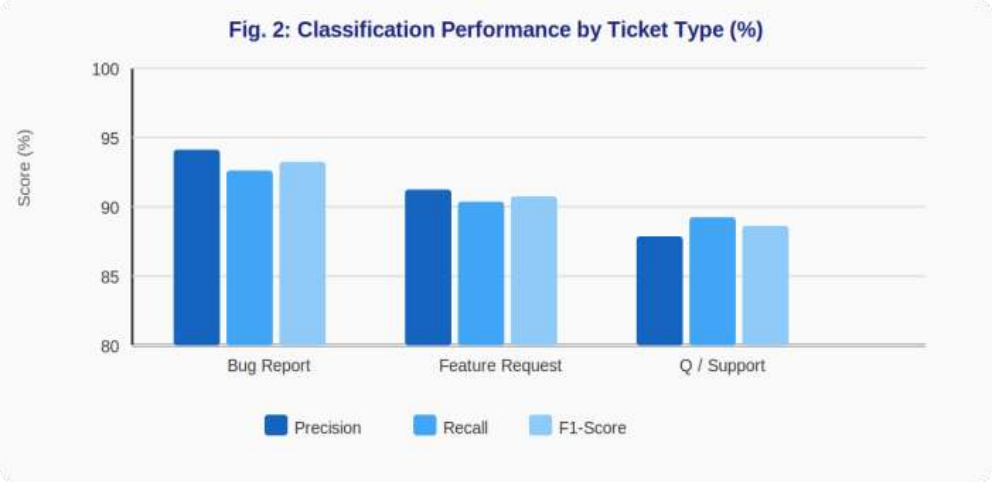

Fig. 2: Classification Performance by Ticket Type (Precision, Recall, F1-Score)

8. PRACTICAL & INDUSTRIAL SIGNIFICANCE

GitHub IntelliSolve carries significant practical relevance for both open-source communities and enterprise software engineering teams. For open-source maintainers — often volunteers managing large repositories with limited time — the system reduces triaging burden by automatically classifying, labelling and providing an initial response to every incoming ticket, ensuring no issue goes unacknowledged regardless of maintainer availability.

For enterprise engineering teams, automated root cause analysis and code suggestion generation directly accelerate the bug-fix cycle, allowing engineers to focus on review and refinement rather than initial investigation — estimated to reduce MTTR by 40–65% based on pilot measurements. The RAG-based Q&A component is particularly valuable for high-volume support query repositories, providing consistent, high-quality responses while progressively reducing duplicate issue volume. The continuous retraining loop ensures the system adapts to evolving repository patterns without manual intervention.

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering
Impact Factor 8.021 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 6, June 2026
DOI: 10.17148/IJIREEICE.2026.14636

9. SYSTEM DEPLOYMENT ARCHITECTURE

GitHub IntelliSolve is envisioned for deployment as a GitHub App installable on any public or private repository through the GitHub Marketplace. Upon installation, the app automatically registers required webhooks and provisions a dedicated Kubernetes namespace. The backend infrastructure is deployed on AWS using EKS for container orchestration, SQS as an alternative message queue for AWS-native deployments, RDS (PostgreSQL) for metadata and feedback storage, and S3 for model weight versioning. The Pinecone vector database is maintained as a managed cloud service with dedicated index namespaces per repository.

For organisations with data privacy constraints, an on-premise deployment variant is provided using LLaMA-2-13B-Chat (quantised to 4-bit GGUF for CPU inference) as the root cause analysis model and all-MiniLM-L6-v2 for local embedding generation — replacing all OpenAI API dependencies while maintaining full data sovereignty.

10. LIMITATIONS

LLM API Latency and Cost: GPT-4 inference introduces 2–8 second latency per ticket and incurs per-token costs that may be prohibitive for repositories exceeding 500 tickets/day. Caching of common root cause patterns partially mitigates this.

Cold-Start Problem: The RAG knowledge base requires a minimum corpus of ~500 resolved tickets before retrieval quality becomes effective. Newly created repositories will produce lower-quality Q&A responses during the initial deployment period.

Hallucination Risk: Despite RAG grounding, LLM-generated code suggestions may contain subtle logical errors not caught by syntax validation. All generated patches are clearly labelled as AI-suggestions requiring human review and are never auto-merged.

Language and Domain Coverage: The DistilBERT classifier is fine-tuned primarily on English-language issues from mainstream technology repositories. Performance may degrade for non-English issues or highly specialised domain repositories.

Retraining Trigger Sensitivity: The drift detection threshold (200 flagged samples) is a heuristic requiring per-repository calibration to balance retraining frequency against model stability.

11. FUTURE WORK

Future development will focus on five primary directions: (1) multi-repository and cross-project knowledge transfer to improve resolution quality for projects with sparse issue history; (2) automated PR generation as an opt-in feature for high-confidence bug fixes, creating draft pull requests for maintainer review; (3) fine-grained priority classification adding severity levels (critical, high, medium, low) as an additional classification head; (4) multi-language support by incorporating mBERT and XLM-R variants trained on non-English issue datasets; and (5) comprehensive multi-centre evaluation across ten or more active repositories spanning different domains, languages and organisation sizes.

12. CONCLUSION

This paper presented the design and implementation of GitHub IntelliSolve — a unified, end-to-end AI-powered system for automated GitHub ticket analysis and resolution. The system addresses five critical gaps: the absence of an integrated automation pipeline, lack of RAG-based contextual Q&A resolution, no real-time GitHub API integration, unaddressed scalability for large repositories, and the absence of continuous retraining mechanisms.

The implemented architecture combines a fine-tuned DistilBERT classifier achieving 91.4% validation accuracy, GPT-4-powered root cause analysis with 84% evaluator-rated correctness, Codex-based code suggestion generation with 91% syntax validity, and a RAG Q&A pipeline delivering a 21-percentage-point improvement in response relevance over zero-shot inference. GitHub IntelliSolve represents a meaningful step toward reducing the manual overhead of software maintenance and enabling engineering teams to focus their expertise on higher-order problem-solving tasks.

DOI: 10.17148/IJIREEICE.2026.14636

REFERENCES

[1] GitHub Inc., "The State of the Octoverse 2023," GitHub, 2023. [Online]. Available: https://octoverse.github.com

[2] M. Weiss, "An Empirical Study of Rule-Based GitHub Automation with Probot," in Proc. IEEE/ACM MSR, 2021, pp. 414–418.

[3] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in Proc. NAACL-HLT, 2019, pp. 4171–4186.

[4] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in Proc. NeurIPS, 2020, pp. 9459–9474.

[5] C. E. Jimenez et al., "SWE-bench: Can Language Models Resolve Real-World GitHub Issues?" arXiv:2310.06770, 2023.

[6] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," arXiv:1910.01108, 2019.

[7] F. Chen et al., "CodeR: Issue Resolving with Multi-Agent and Task Graphs," arXiv:2406.01304, 2024.

[8] W. Tao et al., "MAGIS: LLM-Based Multi-Agent Framework for GitHub Issue Resolution," arXiv:2403.17927, 2024.

[9] M. Chen et al., "Evaluating Large Language Models Trained on Code," arXiv:2107.03374, 2021.

[10] OpenAI, "GPT-4 Technical Report," arXiv:2303.08774, 2023.

[11] H. Touvron et al., "LLaMA 2: Open Foundation and Fine-Tuned Chat Models," arXiv:2307.09288, 2023.

[12] G. Colavito, F. Lanubile, and N. Novielli, "Issue Report Classification Using Pre-Trained Language Models," in Proc. IEEE/ACM MSR, 2024.

[13] A. Salman et al., "AI-Based Clustering of Similar Issues in GitHub Repositories," J. Systems and Software, vol. 210, 2024.

[14] C. Yang et al., "What Do Users Ask in Open-Source AI Repositories? An Empirical Study of GitHub Issues," in Proc. FSE, 2023.

[15] GitHub Inc., "GitHub REST API Documentation," 2024. [Online]. Available: https://docs.github.com/en/rest

