PARSHWANATH CHARITABLE TRUST'S
A.P. SHAH INSTITUTE OF TECHNOLOGY
Department of Computer Science and Engineering
Data Science

C^3 (Code. Collab. Commit): A Collaborative Code Editor using repository Level LLM.

Group No. 02

Rohan Waghode **21107008**
Meet Jamsutkar **22207004**
Arya Patil **21107009**
Urvi Padelkar **21107054**

Project Guide
Prof. Anagha Aher

Contents

Abstract

The challenges of modern software development include scattered collaboration, poor management of code, and difficulty in maintaining large-scale codebases. Developers often struggle with real-time collaboration, handling code complexity, and ensuring code quality, especially when working in distributed teams.

This project therefore aims to develop an advanced code editor that can be used collaboratively by developers to boost their productivity when using large language models. This will include functionalities that perform intelligent completion, real-time collaboration, and easy integration with version control systems. Frontend technologies like Electron and Monaco Editor will provide a smooth user experience, while backend technologies such as FastAPI will ensure robust functionality. This platform will streamline workflows and make team collaboration more efficient.

Introduction

As projects grow increasingly complex, efficient collaboration, effective code management, and accurate code completion have become vital in today's fast-paced software development landscape, particularly in the fast-paced technological environment. One of the major problems observed is the difficulty developers face in coordinating and collaborating in real-time, especially in distributed teams. The existing code editors and version control tools offer basic collaboration functionalities, but they often lack intelligent, context-aware code suggestions and seamless real-time collaboration capabilities. This results in inefficiencies, errors, and time-consuming code reviews.

Introduction

Motivation:

- **Challenges in Collaboration and Code Management:** Developers face difficulties in real-time collaboration, especially in distributed teams. Existing tools offer basic functionality but lack seamless, context-aware integration.
- **Inefficiencies in Current Tools:** Existing code editors and version control tools fail to provide intelligent code suggestions and streamlined collaboration, resulting in errors, inefficiencies, and lengthy code reviews.
- **Motivation for a New Solution:** The increasing complexity of projects and global collaboration demands an integrated environment that enhances workflows, reduces errors, and provides repository-level intelligent code completion.
- **Focus on Productivity and Efficiency:** By addressing the limitations in current development tools, the goal is to improve both individual productivity and team efficiency with a smart, context-aware assistant.

Introduction

Proposed Solution:

- **Integrating Real-time Collaboration:** We aim to build a solution that offers seamless collaboration, even for distributed teams, overcoming the limitations of current tools.
- **Intelligent Code Completion:** By incorporating repository-level intelligent code suggestions, we seek to reduce errors and improve code accuracy and efficiency.
- **Automated Code Analysis:** The solution will offer automated code analysis to manage complex codebases, saving time and improving workflow efficiency.
- **Focus on Smart, Contextual Assistance:** The goal is to simplify workflows by creating a smart assistant that enhances individual and team productivity in a global, remote work environment.

Objectives

1. Develop a collaborative platform for real-time simultaneous code editing using Apache Kafka or a similar socket-based streaming framework.
2. Implement code generation, querying, and analysis based on iterative retrieval augmented generation with repository-level context.
3. Implement smart codebase management with Apache Nutch-based semantic search and context-based smart commits, alongside automated documentation generation using Latent Semantic Scaling.
4. Develop a dashboard that visualizes individual coding behavioral patterns and team performance metrics, leveraging data analytics to enhance productivity and collaboration.

Literature Review

Sr.no	Title	Author(s)	Year	Methodology	Drawback
1	Automatically Generating Commit Messages from Diffs using Neural Machine Translation	Siyuan Jiang, Ameer Armaly, and Collin McMillan	2017	The methodology includes data collection and preparation, gathering over 2 million commit messages and diffs from GitHub projects, focusing on verb-direct object patterns. A Recurrent Neural Network (RNN) with attention was then trained using Nematus to translate diffs into commit messages. Finally, the model was evaluated through automatic metrics like BLEU scores and human assessments, comparing generated and human-written messages for semantic similarity.	The quality of generated messages depends on the quality of the training data; poor data leads to poor results. Training requires significant GPU resources and large datasets, which may not be available for all projects. BLEU scores focus on n-gram overlap and may miss semantic nuances that human evaluators could detect.
2	Collaborative Code Editors - Enabling Real-Time Multi-User Coding and Knowledge Sharing	Khushwant Virdi, Anup Lal Yadav, Azhar Ashraf Gadoo, Navjot Singh Talwandi	2023	The methodology involves the integration of WebSocket communication for real-time data transfer and Operational Transformation (OT) algorithms for conflict detection and resolution in collaborative code editing. A prototype was developed to facilitate seamless multi-user coding experiences, and performance metrics were evaluated to assess the effectiveness of the implemented technologies.	Operational transformation algorithms become more complex with user count and edit frequency, while WebSocket communication can cause delays in high concurrency environments. Conflict resolution techniques have limitations in managing concurrent edits, and scalability challenges arise with increasing user numbers. Integrating WebSocket communication with OT algorithms and real-time synchronization demands a robust, resource-intensive backend.

Literature Review

Sr.no	Title	Author(s)	Year	Methodology	Drawback
3	An evaluation on Automated Technical Documentation Generator Tools	Mahdi Jaberzadeh Ansari	2022	The process involves setting up automated tools like Mintlify, Document! X, Doxygen, Imagix, and JDocEditor for generating Java documentation. Evaluation criteria such as accuracy and completeness are defined and used to test these tools on various Java projects. Data on quality and performance is collected and analyzed both qualitatively and quantitatively. The process concludes with recommendations based on a comparative analysis of the tools' strengths and weaknesses.	Quality issues arise when documentation tools lack sophistication, potentially leading to inaccurate interpretations of complex code; for example, Mintlify may struggle with simple declarations and assignments. There is also a risk of incomplete documentation if the tools are not properly configured or miss critical details. This can result in gaps that developers need to manually address, despite the use of automated tools.
4	RepoCoder: Repository-Level Code Completion Through Iterative Retrieval and Generation	Fengji Zhang, Bei Chen, Yue Zhang, Jacky Keung, Jin Liu, Daoguang Zan, Yi Mao, Jian-Guang Lou, Weizhu Chen	2023	RepoCoder uses an iterative pipeline combining a similarity-based retriever with a pre-trained code language model, enabling continuous retrieval and generation of code based on repository-level context. It leverages information across multiple files within a repository, improving code completion accuracy over traditional in-file tools. RepoCoder's performance is validated with RepoBench, a benchmark using real-world repositories to test code completion scenarios. Experimental results demonstrate significant improvements compared to traditional methods.	1. High Complexity: Requires significant computational resources and setup. 2. Hyper-parameter Sensitivity: Needs careful tuning for different contexts. 3. Evaluation Overhead: Unit test evaluation can be resource-intensive and limiting in less documented projects.

Problem Definition

- **Challenges in Collaboration and Code Management:** Developers face difficulties in real-time collaboration, especially in distributed teams. Existing tools offer basic functionality but lack seamless, context-aware integration.
- **Inefficiencies in Current Tools:** Existing code editors and version control tools fail to provide intelligent code suggestions and streamlined collaboration, resulting in errors, inefficiencies, and lengthy code reviews.
- **Motivation for a New Solution:** The increasing complexity of projects and global collaboration demands an integrated environment that enhances workflows, reduces errors, and provides repository-level intelligent code completion.
- **Focus on Productivity and Efficiency:** By addressing the limitations in current development tools, the goal is to improve both individual productivity and team efficiency with a smart, context-aware assistant.

Research Gap(Limitations of existing systems)

- **Resource Demands**: Systems often need significant computational power and large datasets, making them unsuitable for smaller projects.
- **Scalability Issues:** Managing low latency and synchronization becomes challenging as the number of users increases.
- **Conflict Resolution Challenges:** Timestamp-based and semantic conflict detection methods struggle with handling multiple concurrent edits.
- **Evaluation Limitations:** Metrics like BLEU scores may not accurately reflect the semantic quality of generated content, leading to poor assessments.
- **Documentation Gaps:** Automated tools may produce incomplete or inaccurate documentation, especially with complex code.
- **High System Complexity:** Real-time collaboration tools, using WebSocket communication and OT algorithms, demand a robust, resource-heavy architecture.

Problem Definition

- **Real-time, Multi-user Collaboration:** Enable multiple developers to work simultaneously on the same codebase without delays or conflicts.
- **Repository-Level Contextual Code Completion:** Implement LLM-based intelligent code suggestions that understand the entire repository, offering precise code completions and avoiding redundant manual search efforts.
- **Advanced Codebase Management:** Integrate smart codebase management features, including semantic search, automated documentation, and smart commits to enhance project organization.
- **Team Performance Metrics:** Provide visual dashboards that offer insights into individual and team performance, enabling better management of development workflows and productivity enhancement.

Scope

1. Develop a web-based collaborative code editor utilizing Electron and Monaco Editor for real-time simultaneous code editing, targeting software development teams, remote developers, coding bootcamps, and educational institutions.
2. Implement a repository-level LLM-based code completion system with iterative refinement and hybrid search approaches for intelligent code suggestions,benefiting developers working with large codebases.
3. Integrate advanced code analysis, querying, and documentation features using Lint, Babel, and AST to ensure code quality and streamline development processes for development teams, QA engineers, and technical writers.
4. Build a robust and scalable backend infrastructure using FastAPI, language servers, and cloud platforms like AWS,Azure, or GCP to support high availability and performance, ideal for organizations seeking efficient coding solutions.

Technological Stack

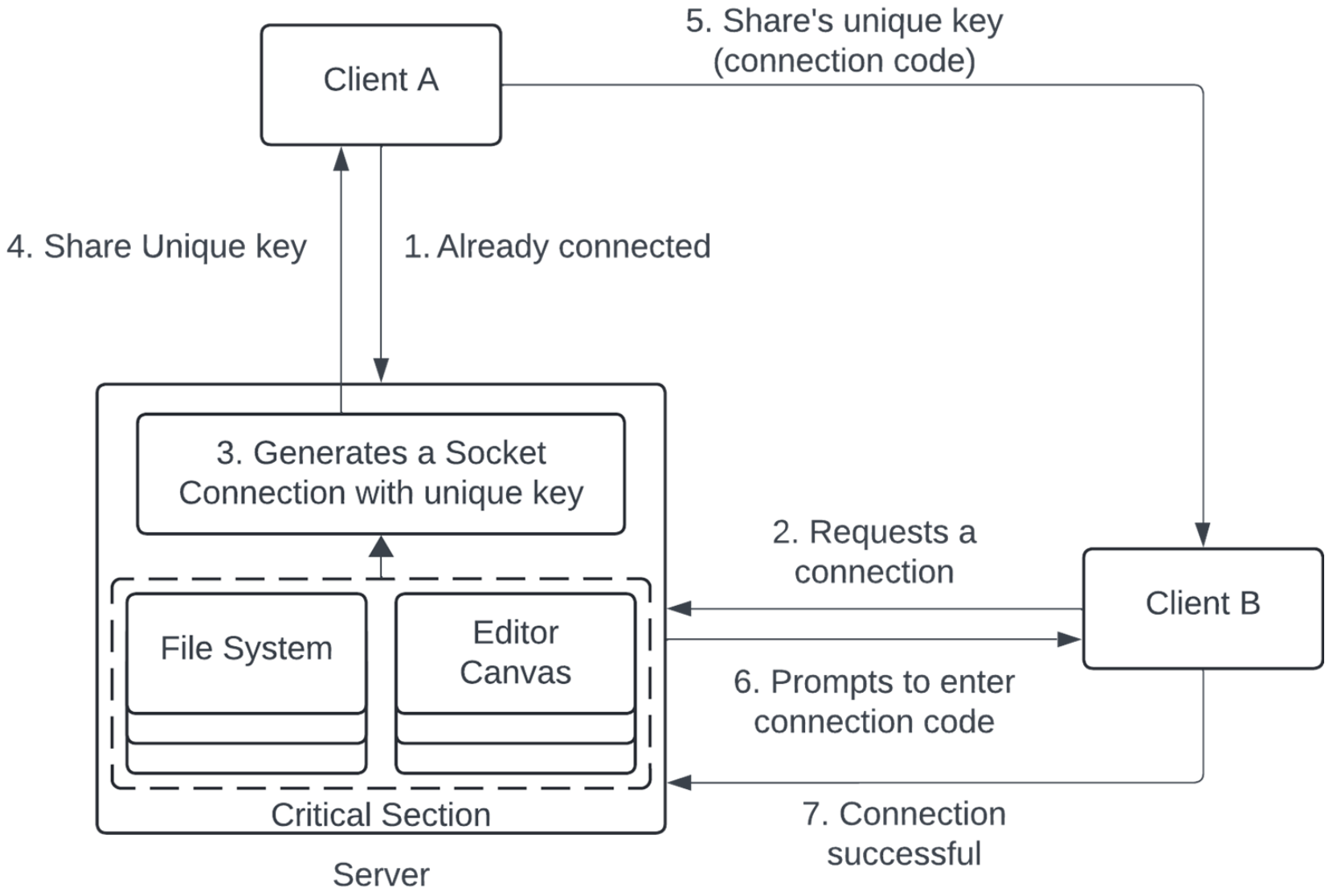
- **Language:** Python, Javascript
- **Frameworks:** Electron JS, FastAPI, Tensorflow Keras
- **Databases:** Postgres, Redis
- **Editor: :**
    - **CodeMirror:** An Open Source base code editor platform that allows for editing of files, history management, parenthesis checking and easily extendable to integrate new features
    - **HDFS:** Open Source Big data based file management system able to handle SQL, Document objects and other frequently used data
- **Collaborative:**
    - **Websockets:** To maintain shared connections and allow multiple user's to work on single repository/files
    - **OT/CRDT Algorithms:** To manage conflicts, race conditions and critical sections, to ensure provisioning and error free updation of data
    - **Kafka:** Open Source Streaming framework, part of the apache ecosystem allowing for seamless data streaming.

Technological Stack

- **Code Generation:**
  - **OpenAI Codex Iterative Process:** To integrate seamless code completion.
  - **JSONL Parsing:** Python libraries like json, ujson for handling .jsonl files.
  - **CodeBERT:** Vectorization
  - **FAISS:** Similarity Search
- **Automated Documentation:**
  - **Pygments:** To detect the programming language in which a particular piece of code is written
  - **Abstract Syntax Trees (AST/ESPrisma/JavaParser):** To Analyse code and parse data.
  - **pipdeptree/node-remote-ls/maven-dependency-plugin:** To analyse and maintain dependencies
  - **Pinecone:** To store vector embeddings data used for RAG based LLMs
- **Automated Smart Commits:**
  - **Celery/Cron Jobs:** To Automate regularized branch commits for log based security.
  - **SLMs:** To integrate fast short context based content generation for commit messages
  - **Git:** A base open source requirement to build the system on

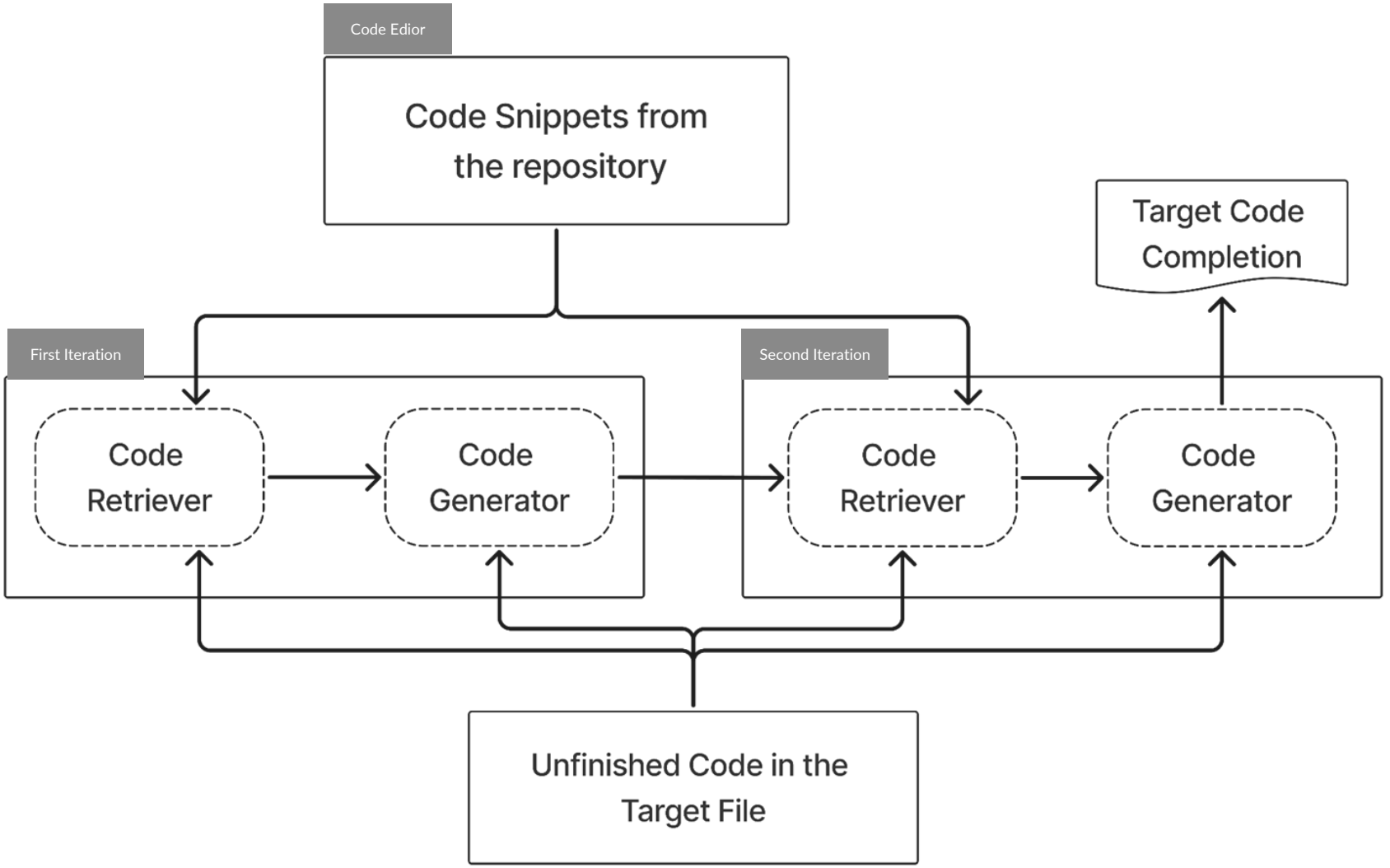
Proposed system architecture/Working

1. Collborative Editor system Architecture



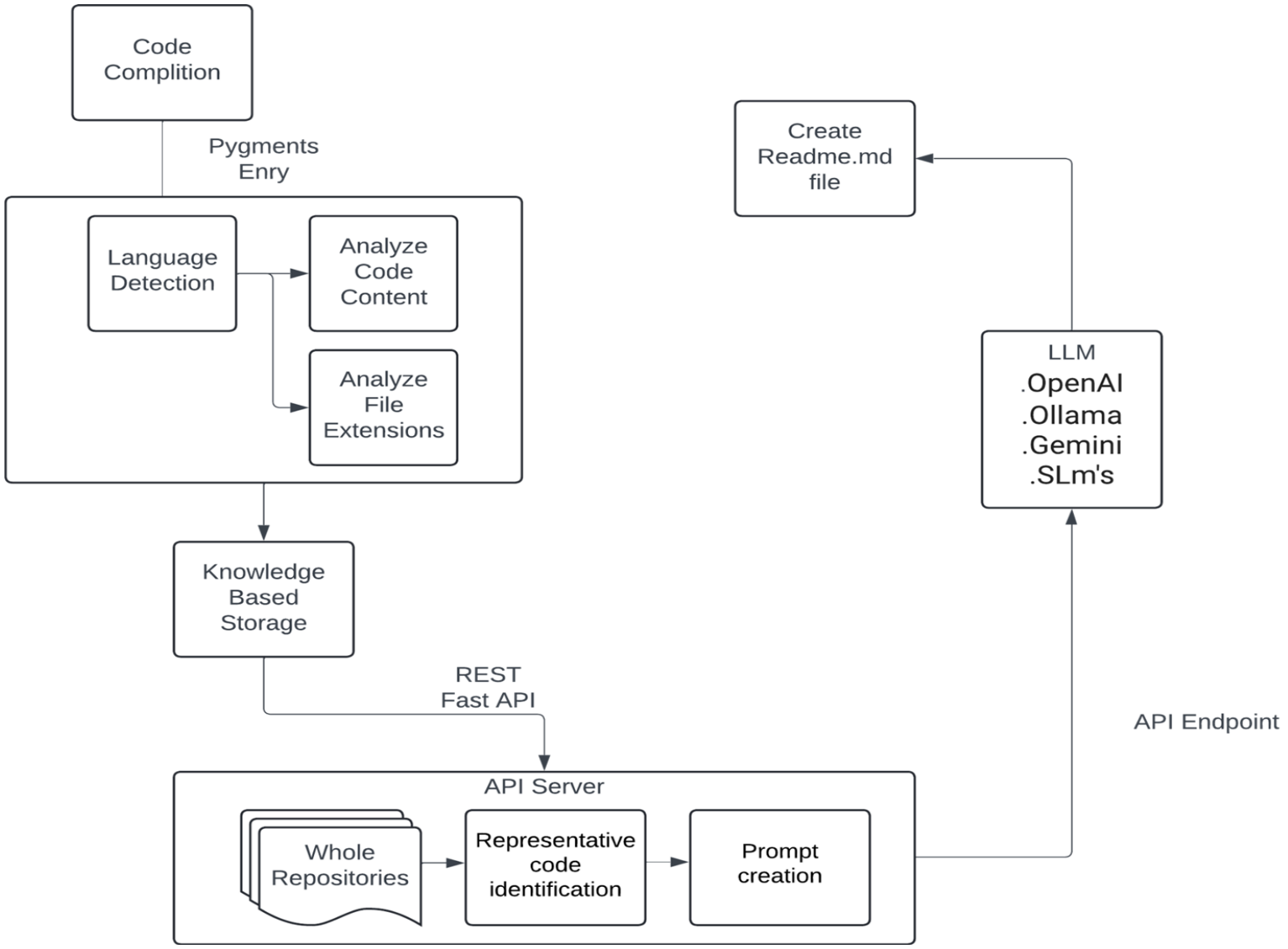
Proposed system architecture/Working

2. Repository Level Code generation:



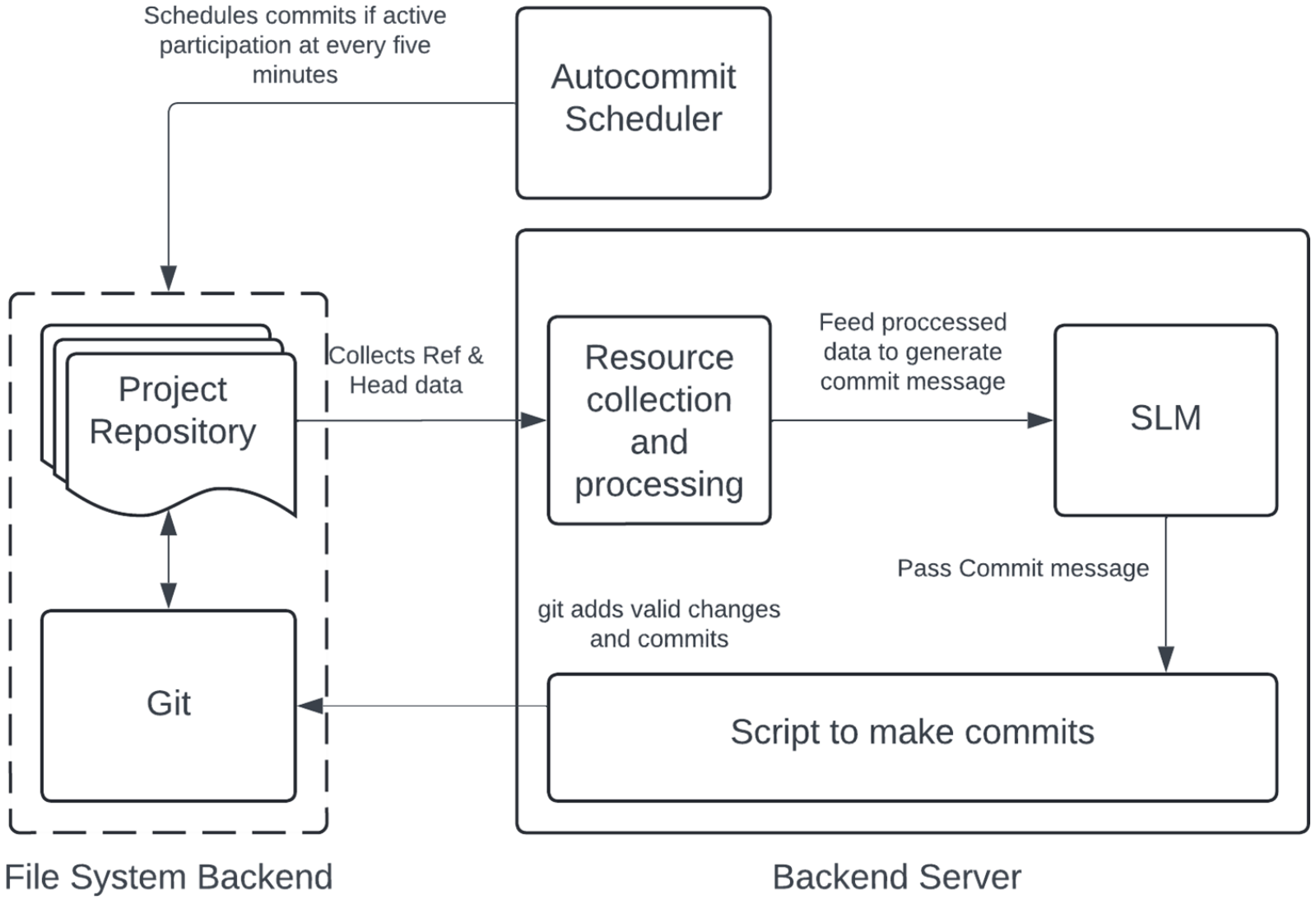
Proposed system architecture/Working

3. Document generation



Proposed system architecture/Working

4. Automated smart commits



References

- K. Virdi, A. L. Yadav, A. A. Gadoo and N. S. Talwandi, "**Collaborative Code Editors - Enabling Real-Time Multi-User Coding and Knowledge Sharing," 2023** 3rd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)
- Zhang, Fengji, et al. **"Repocoder: Repository-level code completion through iterative retrieval and generation." (2023)**.
- Koreeda, Yuta, et al. **"LARCH: Large Language Model-based Automatic Readme Creation with Heuristics." 2023** *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*.
- Ansari, Mahdi Jaberzadeh. **"An evaluation on Automated Technical Documentation Generator Tools."** (2022)*The University of Calgary* .
- H. Zhou, Y. Ma, W. Xu, M. Wang, B. Du and H. Fan, **"Context-based Operation Merging in Real-Time Collaborative Programming Environments," 2022** IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD)

References

- F. Calefato, G. Castellano and V. Rossano, **"A Revision Control System for Image Editing in Collaborative Multimedia Design," 2018** 22nd International Conference Information Visualisation (IV),
- S. Jiang, A. Armaly and C. McMillan, "**Automatically generating commit messages from diffs using neural machine translation," 2017** 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)
- H. Fan, H. Zhu, Q. Liu, Y. Shi and C. Sun, **"Shared-locking for semantic conflict prevention in real-time collaborative programming," 2017** IEEE 21st International Conference on Computer Supported Cooperative Work in Design (CSCWD)

Prototype Design Demonstration

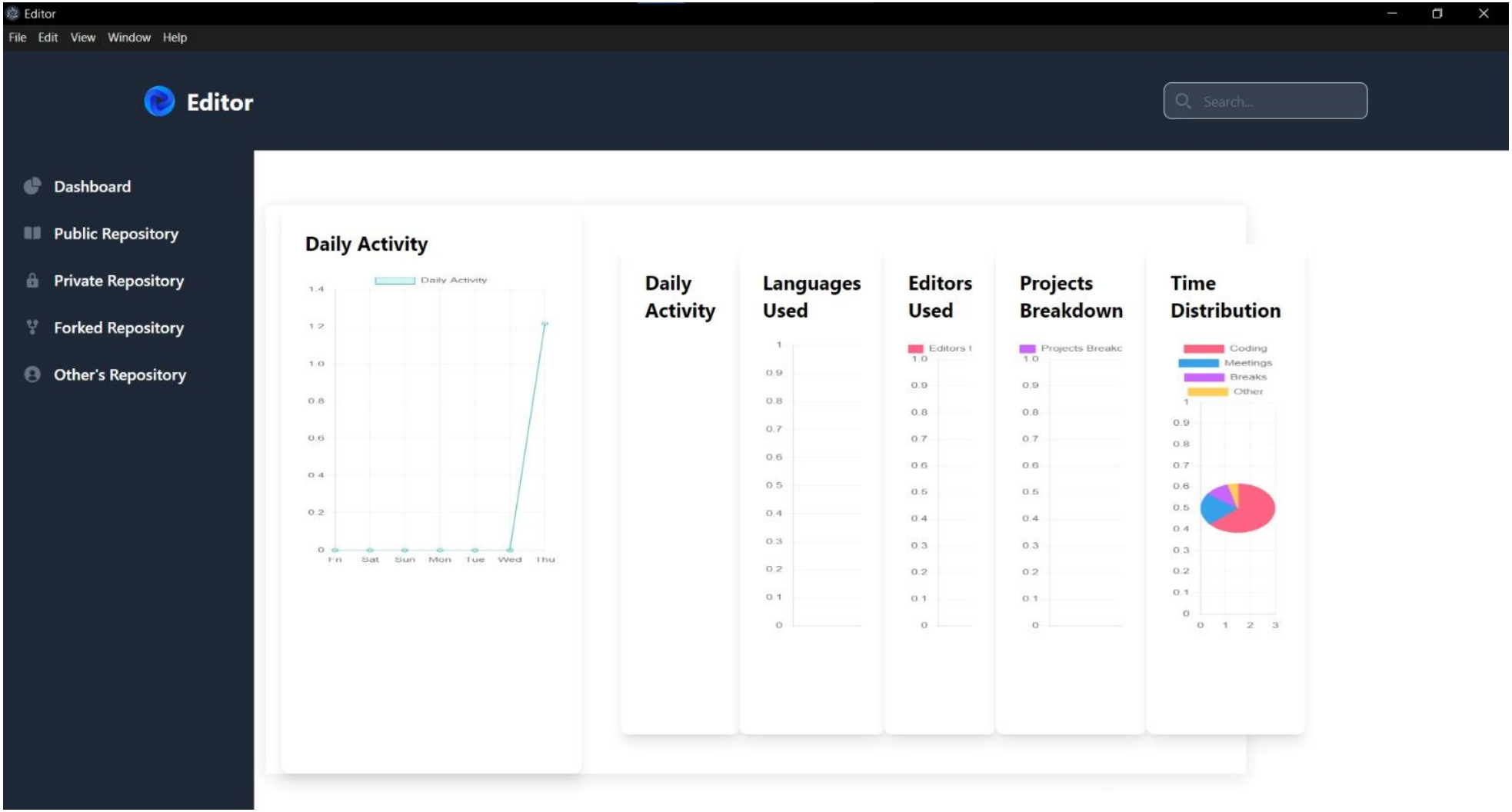


Figure 1: Programming Analytics Dashboard

Prototype Design Demonstration

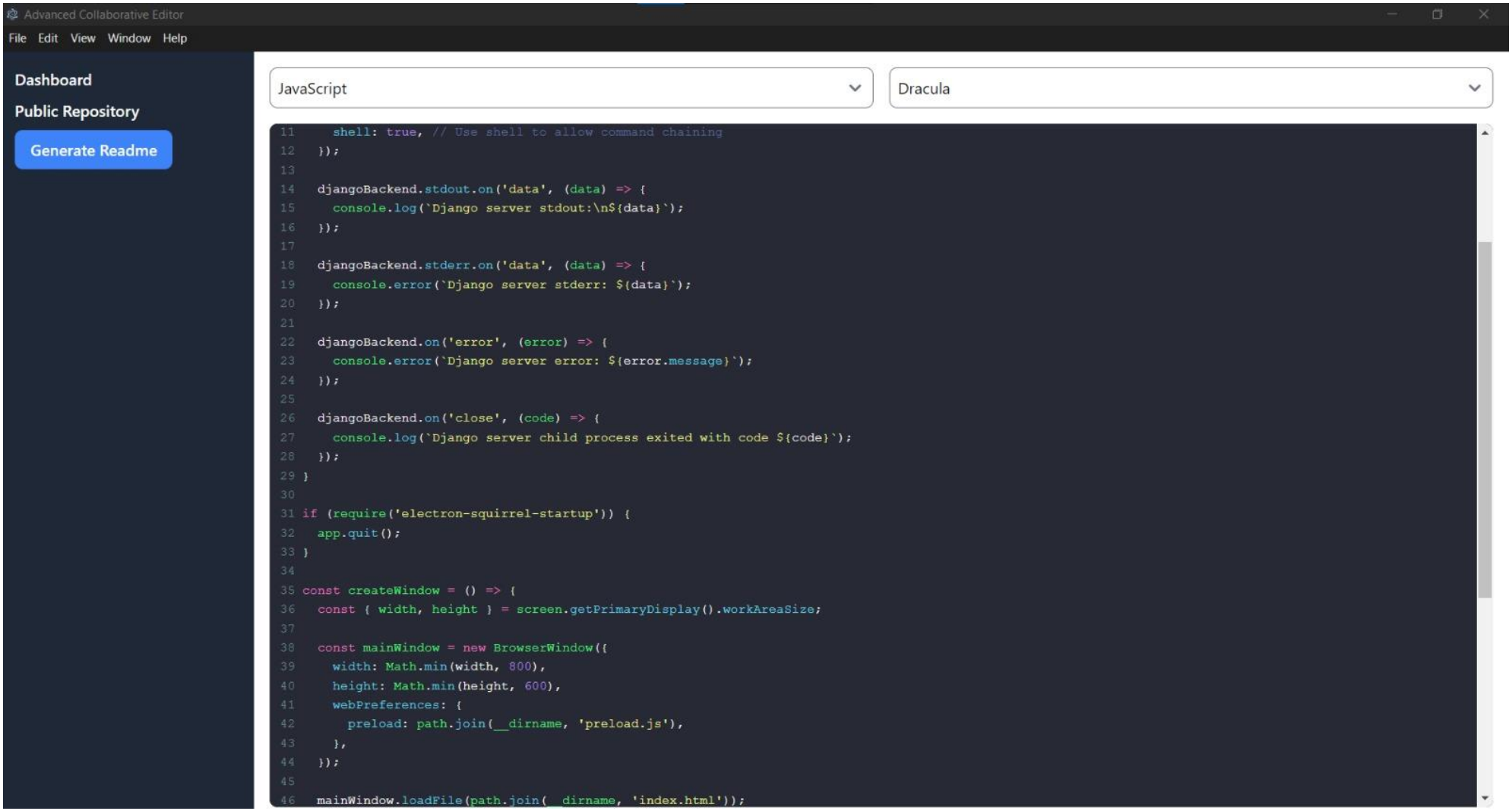


Figure 2: Collaborative code editor with Read me Generation

Prototype Design Demonstration

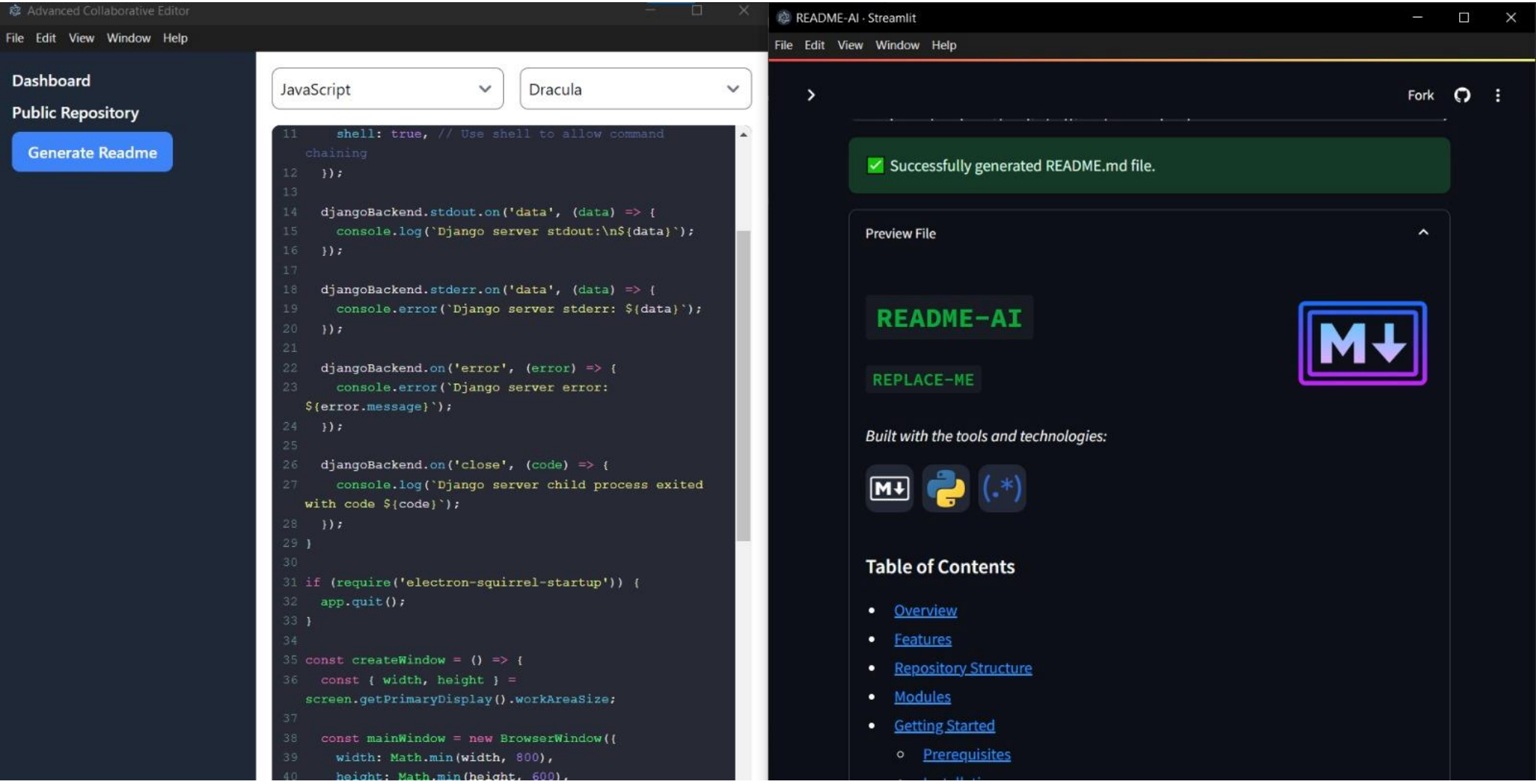


Figure 3: Readme Generation

Implementation Status

1. Development of a collaborative platform for real-time simultaneous code editing utilizing Apache Kafka or a comparable socket-based streaming framework: **Successfully implemented with a functional basic code editor that supports real-time collaborative editing through WebSockets.**
2. Implementation of code generation, querying, and analysis leveraging iterative retrieval-augmented generation within a repository-level context**: Currently requires data training to proceed with further implementation.**
3. Smart codebase management incorporating Apache Nutch-based semantic search, context-aware smart commits, and automated documentation generation utilizing Latent Semantic Scaling: **Automated documentation generation has been completed; however, Git control for the remaining functionalities is yet to be integrated**.
4. Development of a dashboard that visualizes individual coding behavioral patterns and team performance metrics using data analytics: **Successfully operational, with all real-time graphs functioning as expected.**

Thank You...!!