Bilkent University
Department of Computer Engineering

Alkahest

An auto code reviewing extension for VSCode

CS 453 Application Lifecycle Management
Term Project Proposal Report

Alper Göçmen 22002948

Sarper Arda Bakır 21902781

Deniz Tuna Onguner 22001788

Section 1

Instructor: Eray Tüzün

Teaching Assistant(s): Yahya Elnouby

Project Proposal - Second Draft
March 1, 2024, Friday

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Application Lifecycle Management course CS453.

1. Application Name: Alkahest[1]

2. Application Definition

The proposed project aims to develop a comprehensive Visual Studio Code (VSCode) plugin tailored for software practitioners. This plugin will serve as a multifunctional toolset, consolidating essential features such as code duplication finding, bug identification, and resolution assistance within the VSCode environment. The primary goal is to streamline the software development process by offering a unified solution that integrates diverse functionalities into a single, user-friendly interface.

3. Brief Description of the Problem

In software development, practitioners encounter the challenge of navigating through multiple websites, tools, and applications to perform essential tasks such as code duplication, bug identification, and resolution. The fragmented nature of existing solutions often leads to inefficiencies, as developers must switch between disparate platforms, sacrificing productivity and workflow continuity. Spending an average of 20 minutes per day switching between various tools and websites for tasks leads to a 15% loss in productivity [1]. Additionally, developers typically juggle between 5 different platforms, further hindering workflow continuity and collaboration. To address these challenges effectively, software practitioners require a cohesive and integrated toolset consolidating diverse functionalities within a single environment. Currently, such a comprehensive solution is necessary for workflow optimization, collaboration, and code quality enhancement efforts within the software development community. The proposed VSCode plugin aims to mitigate the challenges associated with fragmented toolsets and elevate the software development experience for practitioners worldwide. This approach will potentially improve developer productivity by 20% and enhance code quality by %10, benefitting millions of developers worldwide [2].

4. Motivation

Why should you solve this problem (time benefits, ease of use, etc.)?

1. **Time Benefits:** By consolidating essential development functionalities into a single VSCode plugin, practitioners can significantly reduce the time spent navigating disparate tools and resources.

[1] A hypothetical universal solvent or substance that could dissolve any other substance, including gold.

2. **Ease of Use:** The plugin offers an intuitive and user-friendly interface, simplifying complex tasks such as code duplication finding, bug identification, and resolution. Practitioners can seamlessly access and utilize diverse functionalities within the familiar environment of Visual Studio Code, enhancing workflow efficiency and ease of adoption.

Why would practitioners need your tool?

1. **Comprehensive Toolkit:** The VSCode plugin provides a comprehensive toolkit tailored to the specific needs of software practitioners. By consolidating multiple features—including code review, summarization, bug identification, and resolution—into a single platform, the plugin offers a one-stop solution for diverse development tasks.

2. **Elevated Code Quality:** With built-in code analysis, error detection, and resolution assistance, the plugin empowers practitioners to uphold high standards of code quality and best practices adherence. The plugin promotes cleaner, more maintainable codebases by providing automated suggestions and guidance for resolving common coding errors and inefficiencies.

5. Initial High-Level Requirements

These functionalities and features allow users to review their code in their local codebase. The input will be all the files currently open in the VSCode. The extension will review the code and find the duplicated code lines and bugs or mistakes if there are any, automatically. The output will be all duplicated lines and their locations, the bugs or mistakes, and related solutions. It is the auto-review process before the pull request for individuals to check their codes for improvement.

Code Duplication Functionality:

- Allow users to initiate and conduct code reviews within the VSCode environment.
- Enable reviewers to provide feedback, comments, and suggestions on code segments or files.
- Support collaborative review processes among team members, including the ability to track review progress and status.

Bug Identification and Resolution:

- Implement robust bug and mistake detection mechanisms integrated with intelligent analysis algorithms.
- Identify common coding errors, bugs, or inefficiencies within codebases.
- Provide automated or guided solutions for resolving identified issues, drawing from internal resources and external knowledge bases like Stack Overflow.

We plan to conduct code scanning at 1 MB per second, although this may vary depending on internet speed. After scanning, the resolution part will be faster. After scanning the code, we anticipate a faster subsequent process as we plan to submit the relevant bugs and issues to Gemini, expecting quick solutions.

6. Competitor Applications

- OpenAI / ChatGPT

  ChatGPT is very good at understanding natural language and can be fine-tuned for specific programming languages and coding styles. Also, it has access to a massive dataset that allows it to generate a wider range of potential solutions. However, it may struggle to understand the broader context of code, leading to suggestions that might not be entirely relevant or accurate. It has been designed as a general-purpose language model, not explicitly optimized for code analysis and bug detection. Also, it may sometimes generate incorrect or nonsensical suggestions for code fixes or mistakenly flag code as duplicates.

- Github Copilot

  Copilot is built into popular code editors, streamlining use as part of a developer's workflow. It is trained on a massive volume of code, allowing it to recognize coding patterns, suggest potential improvements, or identify duplication. Also, it has continuous learning, meaning it improves its suggestions based on user feedback and real-world code examples. However, It may sometimes propose inappropriate or inaccurate code completions, requiring careful review from the developer. It can struggle with code that relies on complex logic or specific domain knowledge. Lastly, it may confidently generate incorrect or even harmful code snippets.

- Review Board

  Review Board is designed for peer review and code quality evaluation, enhancing the ability to catch bugs and potential issues through collaboration. It efficiently highlights changes in code for detailed examination, increasing the accuracy and efficiency of bug finding. Also, it supports configurable workflows and integration with other versions of control systems. However, it relies heavily on the expertise and vigilance of reviewers, introducing variability in the process. It is less focused on automatic bug detection compared to AI-driven tools. Also, the manual review can be time-consuming, particularly in large projects, leading to overhead.

- Crucible

  Crucible provides structured code review processes, ensuring systematic bug and quality checks. It helps catch issues before they're integrated and allows reviewing the quality of commits. Also, it connects with popular issue trackers and development tools. However, it primarily relies on human input for code quality assessment. It is less advanced in utilizing AI-driven detection of bugs and potential code duplication. Lastly, it can entail some overhead regarding configuration and integration with existing workflows.

In conclusion, the platforms offer a range of features, including code editing, debugging, and version control within a single interface. Strengths include comprehensive toolsets and deep integration with development workflows. However, they may need more flexibility or customization options desired by some users, and their learning curve can be steep for newcomers.

The tools specialize in facilitating code review processes, providing features for commenting, tracking changes, and managing review cycles. Strengths include dedicated functionality for code review, but they may require users to switch between multiple tools for other development tasks, leading to workflow interruptions and inefficiencies.

7. Possible Technical Solutions to Use in the Application

In this project, the detection of bugs and duplicated lines will be facilitated by leveraging SonarQube/SonarCloud, ensuring a comprehensive analysis of code quality. To enhance problem resolution, Google Generative AI will be employed, providing advanced solutions for rectifying identified issues and optimizing code efficiency within the VSCode extension.

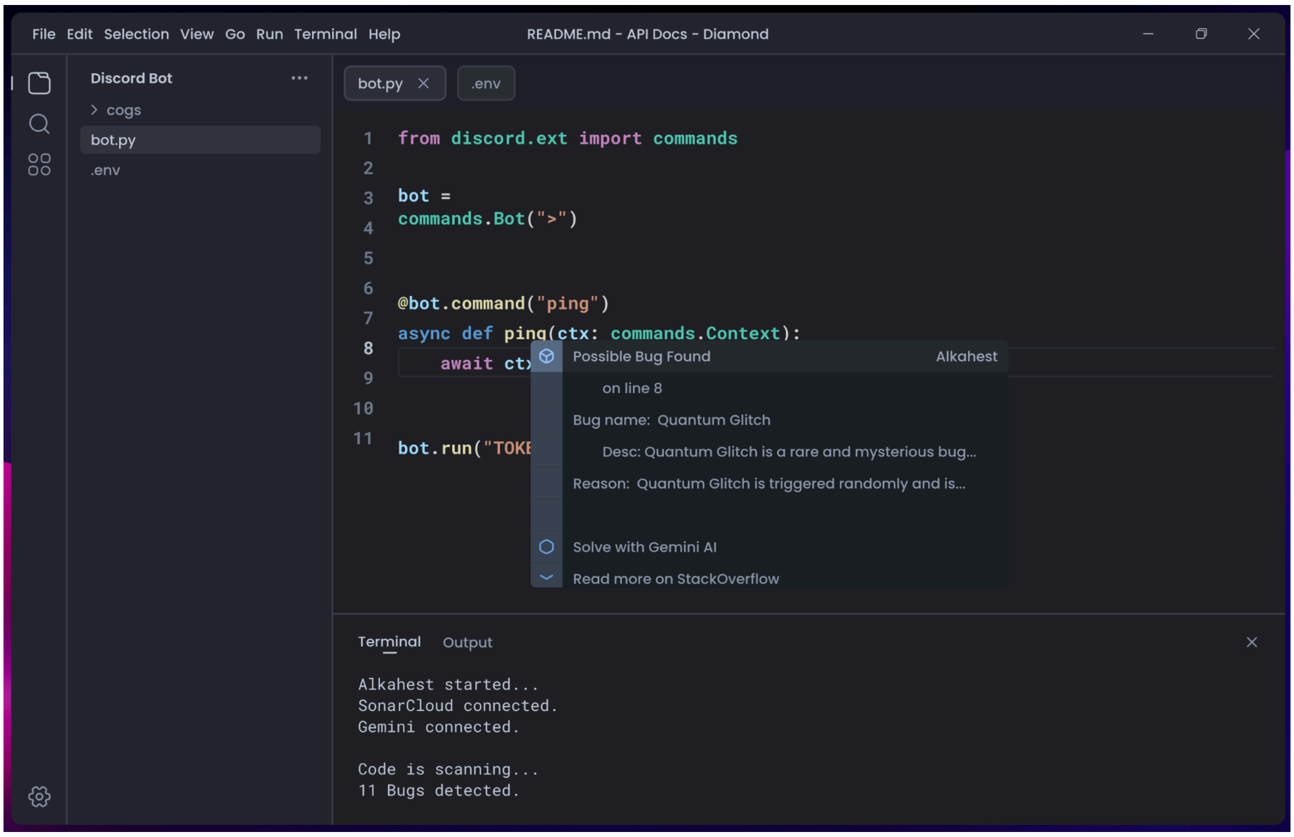
The utilization of Google Generative AI in this project aims to address code duplication by intelligently reducing redundancy. Additionally, it will provide targeted solutions for bugs identified by SonarCloud, enhancing the overall code quality and effectiveness of the VSCode extension.

This project will harness the capabilities of the SonarCloud API to conduct thorough bug and duplicated lines detection within the codebase. Concurrently, the Google Generative AI API will be invoked through respective endpoints, facilitating the generation of tailored solutions to address the identified issues, thus optimizing code quality within the VSCode extension.

Techs, Languages and Frameworks that are planned to be utilized

- Yeoman
- TypeScript (^5.3.3)
- Node.js (18.x)
- google/generative-ai (^0.2.0)
- eslint (^8.56.0)
- SonarCloud/SonarQube with sonar-scanner-npm package for Node.js

8. Provide one mockup image that briefly describes what your solution does.


File Edit Selection View Go Run Terminal Help
README.md - API Docs - Diamond
Discord Bot
cogs
bot.py
.env
bot.py
.env
Terminal
Output

```
from discord.ext import commands

bot =
commands.Bot(">")


@bot.command("ping")
async def ping(ctx: commands.Context):
    await ctx

bot.run("TOKE
```

Possible Bug Found Alkahest

on line 8

Bug name: Quantum Glitch

Desc: Quantum Glitch is a rare and mysterious bug...

Reason: Quantum Glitch is triggered randomly and is...

Solve with Gemini AI

Read more on StackOverflow

```
Alkahest started...
SonarCloud connected.
Gemini connected.

Code is scanning...
11 Bugs detected.
```

References:

[1]
https://www.synapsestudios.com/learn/software-development-partners-should-avoid-context-switching

[2]
https://www.xcubelabs.com/blog/a-comprehensive-guide-to-integrated-development-environments-ides/