Team Sokka

Integrated Development Tools Project

Alex Udodik, Jacob Huber, Malachi Potts, Shawn Poole

Mentor: Brandon Busby

Instructor: Bolong Zeng

CptS 421 - Software Design Project I

October 26, 2020

Project Description

This project will provide software engineers with a set of development tools including a Visual Studio Code, Version-Control extension, a notification system, and a code-search application. The VS Code extension allows a user to avoid having to stage every file change to be committed, as it stages them automatically. The notification system will enable a user to receive visual alerts for any version control changes made. The code search tool allows the user to filter different commits based on the month, day, year, time, email, and author. The GUI sends a query to the SQL database and will display the query results. This allows the user to view useful insight about the commit history of the repo. The goal of the project is to be able to construct these tools using the database locally first, then transition the database over to AWS / Microsoft Azure so every user's information is always up to date because it only exists in one instance on the server. With these tools, software engineers will be able to increase their productivity in developing software products. The health of applications run by users will be monitored via telemetry, gathering data about the performance, usage, and health of the software, facilitating data-driven decision making.

Stakeholder / **Client Requirements & Needs**:

- Visual Studio Code extension allows stakeholders to have file changes automatically staged.
- User interface allows the client to view useful queries on different authors commits.
- The user interface allows the client to search through the code base to find different git commits up to the most recent commit, viewing insight about who is creating the commits, over what time period commits are being made, and the number of commits made by a person over a certain time period.
- The database that stores the git metadata gets updated to a certain commit level
- The user will have access to real time version control data, so they can make decisions based on real time data.
- The types of queries displayed by the GUI will be diverse enough to suit a variety of user's needs.
- The communication between the GUI and the server will be efficient and quick regardless of the size of the repo, only updating when necessary.
- Ensure reliability through rigorous testing, ensuring the information is correct across multiple systems when the server goes live, and making sure the information is correct across multiple systems that connect to the repo.
- User's privacy will remain intact for collection of health and usage data facilitated by the API.

Performance Requirements

- **Plug-In:** When a client uses the visual studio plug-in, any time a file gets changed, created, or deleted, the plug-in should have a response time of less than 40ms.

- **Code Search Tool:** The performance metric of the code search tool is its query response time; it should take less than 1 second to retrieve the results of a database query.
- **Health & Usage API:** The health and usage API should constantly monitor the health and usage data with updates to the server occurring as soon as possible.
- **Database Server:** The query response time should execute queries in at most 500ms. The response time will depend on how much data is being stored. This should allow stakeholders to run through different queries through the database quickly and more effectively. accessibility of the database server should give clients 4 nines availability (99.99%). After using a local database, it will utilize AWS / Microsoft Azure for its server. Giving access everywhere with an internet connection.

Mock-Up GUI:

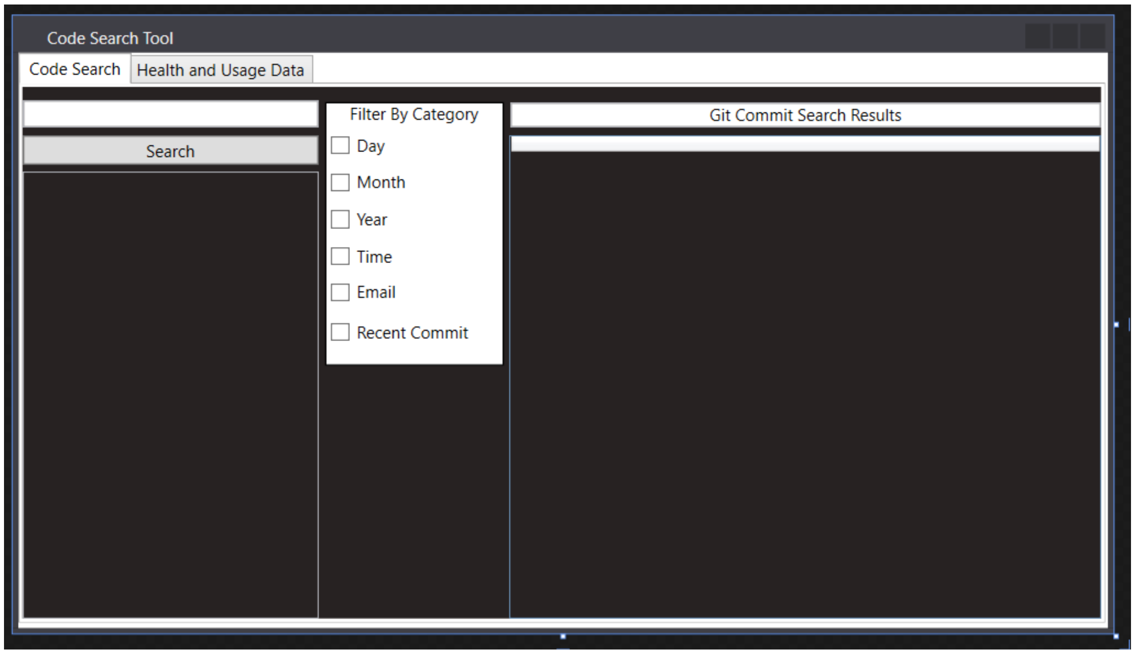

Figure 1: Overview of a draft of a potential GUI for the code search tool.

Storyboard:

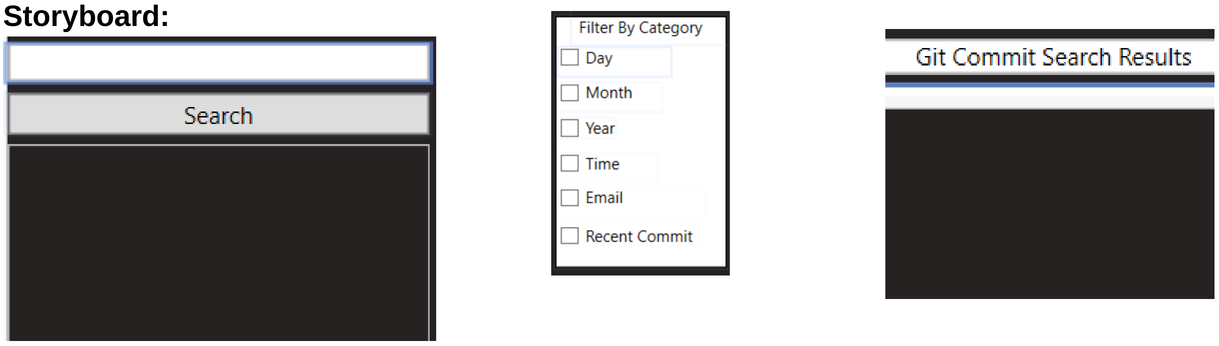

Figure 2: Close up look on the GUI
components. (Left to right: Query search, filter by category and search results)

- **Search Text Box:** The search box will allow stakeholders to search through the names of all users that staged commits, or have already committed to git, filtering out users by matching their name.
- **Filtering:** The filter by category will allow the client to filter the commits based on the day, month, time, year, email, specific users, as well as enabling a user to filter by a time range, or aggregating different filters to meet flexible search criteria.
- **Git Commit Search Results Box:** When the client selects a user in the search text box, it will display all of the selected user's commits. When a checkbox is checked, it will filter the commits in the search results box based on that checkbox.

Mapping of Requirements to Technical Specifications

Number	Need	Metric
1	Usability of Visual Studio Plug-In	The plug-in is plug-and-play, increasing the user's productivity immediately
2	Fast git executables	The executables should run instantly when a file is modified
3	Local SQL Server Availability	The SQL server should have 4 nines availability (99.99%).
4	Code Search Tool GUI Ease-of-Use	The GUI should take less than 10 minutes to figure out how to use it
5	Code Search Tool Query Response Time	The queries sent by the GUI should return the results in less than 1 second.
6	Notification System Response	Notification system should notify the user within a minute
7	Health & Usage Data API Availability Time	The API should be able to monitor the health and usage data as soon as possible.
8	System Hardware Requirements	Windows 10's minimum requirements (1 Ghz Processor and 1 GB of system memory)
9	Space requirements for software download	The Visual Studio Extension, Code Search Tool should be less than 10MB download

10	Code Search Tool SQL Queries	Able to filter by any attribute of a commit and periodically update every hour.

Table 1: Tabular format showing user needs and their metric

Preliminary Implementation Plan

- **Hardware:** The user must be running a computer device that is able to run Microsoft Visual Studio Code along with an active SQL server at the same time.

- **Software:** The required software is Visual Studio Code, the GUI application, the plugin, the API.

- **Network:** A network connection will be used in order for parsed git log information to be inserted into the SQL server. However, if a network connection is unavailable, data will be saved locally to disk and will be submitted to the SQL server once an internet connection is available.

- **Data Components:** The important part of the code-search application is the ability to show filtered commit information. Before a commit's information is stored on the SQL server, a database schema is constructed in a way that allows a user to filter by many attributes of a commit. For example: The number of times a user has committed within a certain date/time range, what type of files were modified by this user, the name of the files and file extension type that were created/modified by a particular user. After a schema has been created, commit information is retrieved using the "git log" command from within a repository, parsed and inserted into the SQL server database immediately if an internet connection is available. Figure 3 shows multiple users running the code-search tool application and data is inserted immediately. When any user commits a change to a file, it updates the SQL server (Depending on internet connection). If they do not have an internet connection, it does not update the SQL server and saves locally to disk. Constructing SQL queries will be performed within the code-search tool application and the application will connect to the SQL server and display the requested data within the code-search application. The code-search application essentially acts as a GUI wrapper to display commit information that is filtered by the user.

- **Relationships:** When running Visual Studio Code with the version-control extension enabled, any file activity such as creation, deletion, or modification will automatically stage those file changes to be committed. The application will update the database by parsing the output of git log for a given repository. The computer when running the application will retrieve commit data from the repository, check it against the database stored on the server and update it if necessary, then the application will retrieve data from the database on the server by sending it queries to display it in a useful way. The API will be added into code in catch statements and at other useful spots to then communicate back to the server when the code utilizing the API calls it. The server will store useful information about the health and usage data of software products developed with the API.

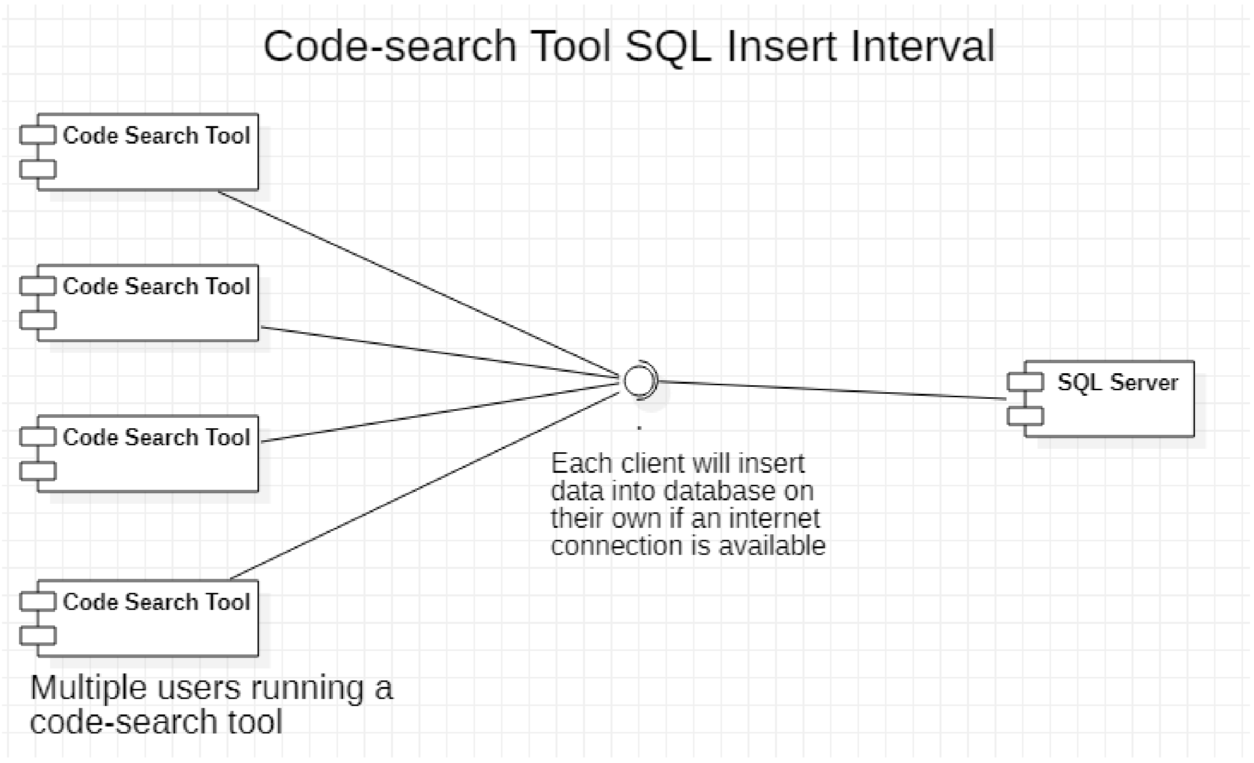
Code-search Tool SQL Insert Interval
Code Search Tool
Code Search Tool
Code Search Tool
Code Search Tool
Multiple users running a code-search tool
Each client will insert data into database on their own if an internet connection is available
SQL Server

Figure 3: Displays multiple code search clients that are run by multiple users and insertions into the SQL server are instant if an internet connection is available.