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

November 16, 2020

Concepts, Algorithms, or Other Formal Solutions to Build the Product

- **Git Metadata Retrieval:** A child process is created to invoke the command line window and will be passed in the command “git” with “log” as the parameter. After the command and argument has been executed successfully, git’s metadata will be returned to the Code Search Application in the form of a string.

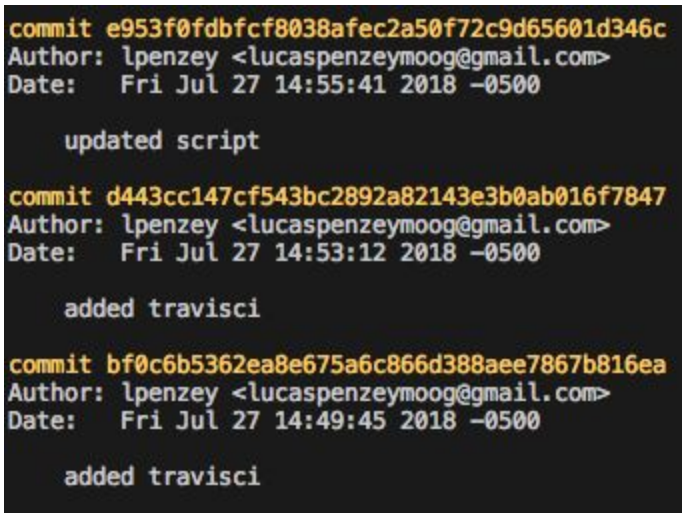

Figure 1: Sample ‘git log’ output.

- **Git Metadata Parsing:** In order to separate out and use the info retrieved in the git log output, we parse the info accordingly. Each line of the git log output is then checked if it starts with the word “commit”, and if it does, this means that the current line is a new commit and the previously parsed lines contain information about the previous commit (author information, message, date committed, commit id). Each line for a single commit is parsed and will be inserted into the SQL database.
- **Running Git Executables:** Running git executables are within the Visual Studio Code Extension. Each time a file event handler is triggered, it runs the git add command. After the git add command is executed it moves the files to the staging area for them to be committed then pushed to the git repository.
- **Builder Pattern:** To create queries for the database follows a similar pattern, with only marginal differences depending on the type of information created. To streamline this process, the builder pattern is implemented to reduce the repetitiveness of writing the many combinations of queries possible. The builder pattern allows for simple construction by storing all of the specific conditional data and removing the formatting and syntax. For a select query, the class has a setColumns method, a setTable method, a setConditional method, a setGroupBy method, and a setOrderBy method, to handle all the ways a select query could be constructed. Finally, a build method returns the completed query. Knowledge about the necessary components to structure a select query is still necessary.
    - Some examples of queries include:

    - SELECT commitID, author, date, message FROM commits
    - SELECT commitID, author, message, MAX(date) as mostRecent FROM commits, authors WHERE commits.author = authors.author ORDER BY date
    - SELECT author, COUNT(author) as numcommits FROM commits GROUP BY author ORDER BY numcommits DESC LIMIT 1
- **GUI Design:** The main design for the GUI is that it shows commit information, the commit itself, the code/changes made to the code, as well as contributors to those changes. Users can also enter date ranges, authors, time ranges to execute queries based on the filters they put in. Once filters are entered, it displays all the relevant commit information to the GUI.
- **Suite of Tools**: The general idea of how each of the components in the suite of tools interact is as follows. The VS Code extension's role is to provide automatic staging to git whenever files are saved/created/deleted. The Code Search tool's role is to parse git's metadata for the repository and insert it into the SQL database. After any type of data has been inserted into the SQL database, the commit information can be filtered by connecting to the SQL server and requesting the particular information that is requested. The Health and Usage API Tool is a pipeline that allows the developers to view errors and statistics from multiple clients of the VS Code extension and Code Search Tool. The data that is able to be viewed from within the Health and Usage API Tool will come from the VS Code extension, Code Search Tool, and Notification System. Below, Figure 1 provides a visual representation of how the components work together.

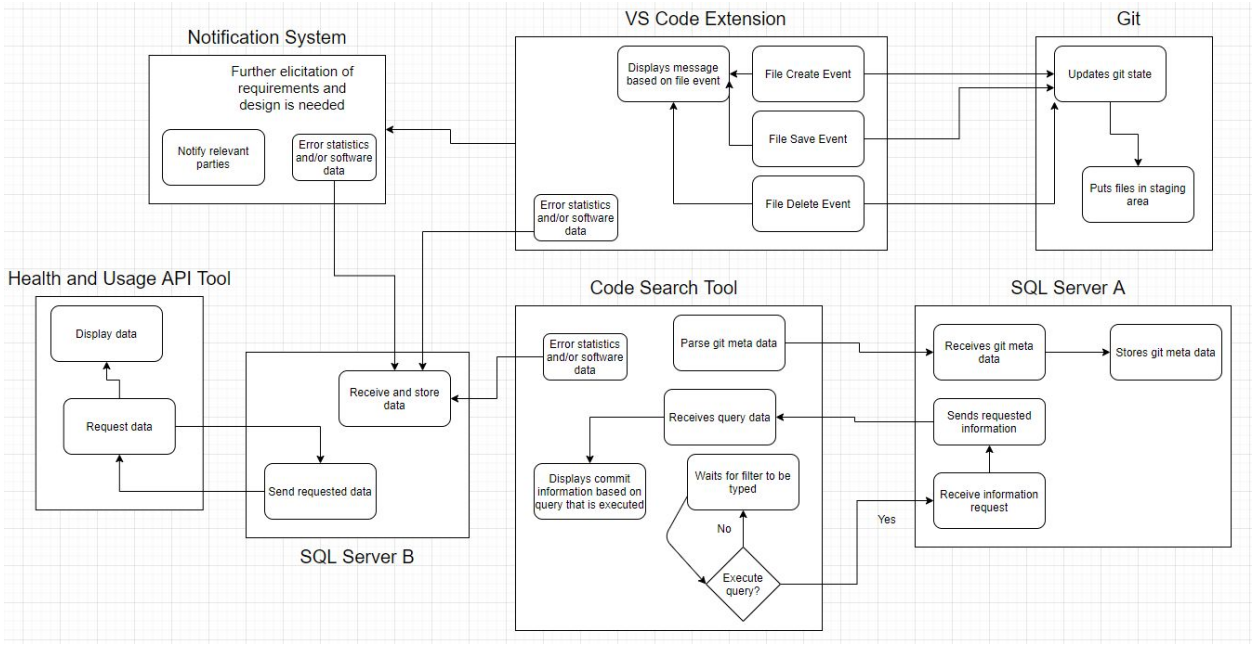

Figure 2: The interactions between the VS Code extension, git, Code Search Tool, Health and Usage API Tool, SQL Servers.

Solution Selection Process—Discussion of Design Choices

External Decision

- **GUI Design:** The user-interface was designed in such a way so that all relevant information can be displayed to the user at once. Unused components may be hidden.

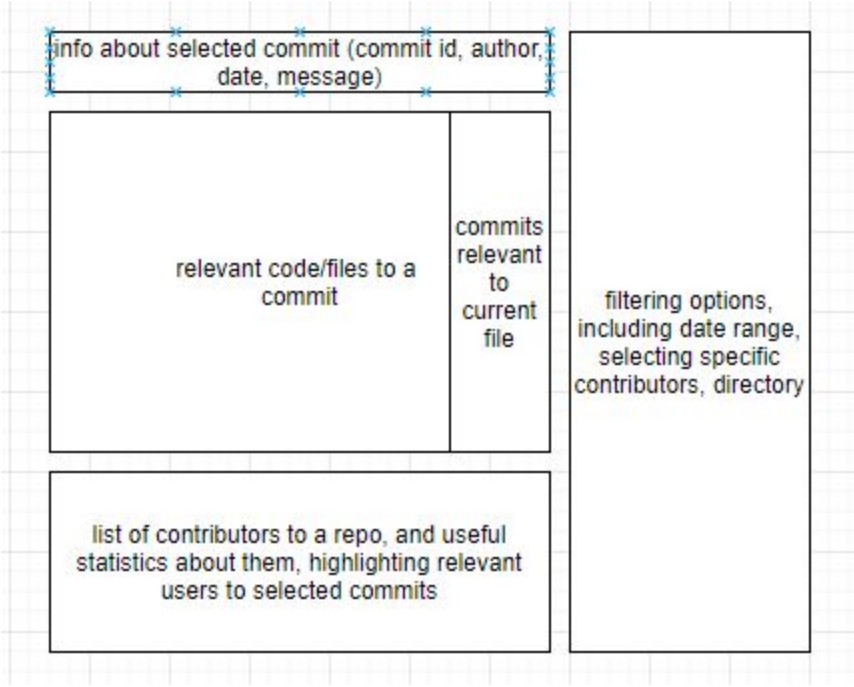

Figure 3: A mockup of the potential GUI design.

- **SQL Queries:** Users can search commits, files, dates and code on the Code Search Tool that utilizes different SQL queries to filter through according to various options. The design of the SQL queries was to make sure it was straightforward for users to filter out useful data, like from a specific directory or within a certain date range. Users can filter out different users' commits based on different dates, times, and code authors. The mockup design shown in Figure 3 and the queries selected were based on feedback from potential users on what information was desired the most.
- **Database:** The database schema is being designed in such a way that allows stakeholders to execute the most common queries decided above most efficiently.
- **VS Code Extension:** The extension design is straightforward due to the fact that it has only one job: to provide automatic git commands to stage a file to git when a file has been modified (file save, edit, and delete). Some design choices were also considered such as having the extension parse "git log" metadata, but that idea was not

implemented because that design choice would lead to coupling the extension with the SQL database. The impression is that the Code Search Tool will be involved with parsing the “git log” metadata along with inserting that metadata and retrieving commit information from the SQL database. It would not make sense to have both the extension and Code Search Tool interact with the SQL database.

Detailed Implementation Framework

GUI & Database: The GUI executes SQL queries on a database initially set up using PostgreSQL, to be migrated to cloud-based hosting in the future, and will be the main tool that is displaying the relevant commit information from the database (i.e. date, author, time). The database will be storing all of the information about all commits from all users. When a query is executed in the GUI, it pulls the data from the database which is displayed to the GUI. The GUI allows stakeholders to enter different information into textboxes which will execute pre-staged SQL queries that displays the information from the database based on what filter is typed in (date range, author, time range).

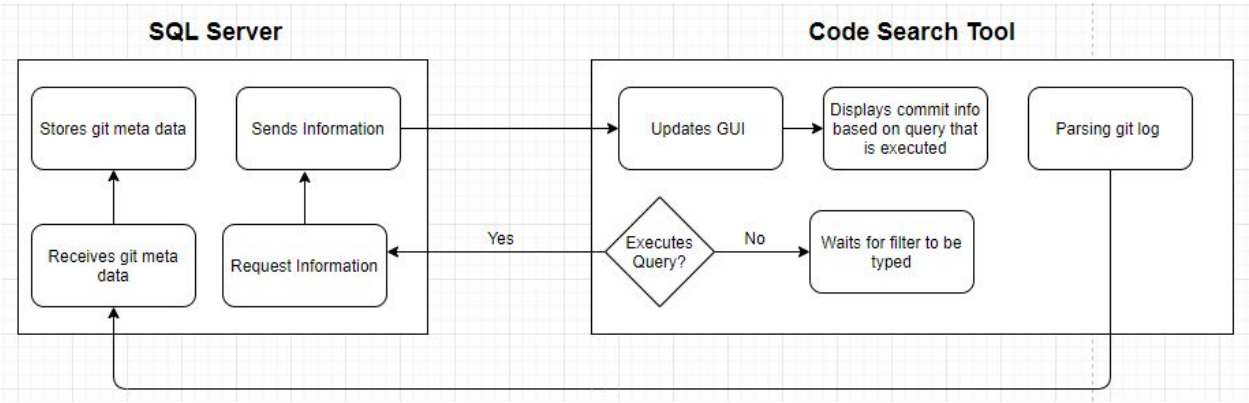

Figure 4: Shows the connectivity and interaction between the SQL Server and Code Search Tool.

- **Visual Studio Code Extension & Git:** The visual studio extension runs git executables within file event handlers. When a file is created, deleted or changed it displays in Visual Studio Code notifying the user what change was made to the file. Once a file is modified, created, or deleted it puts that file in the staging area for it to be committed to a git repository.

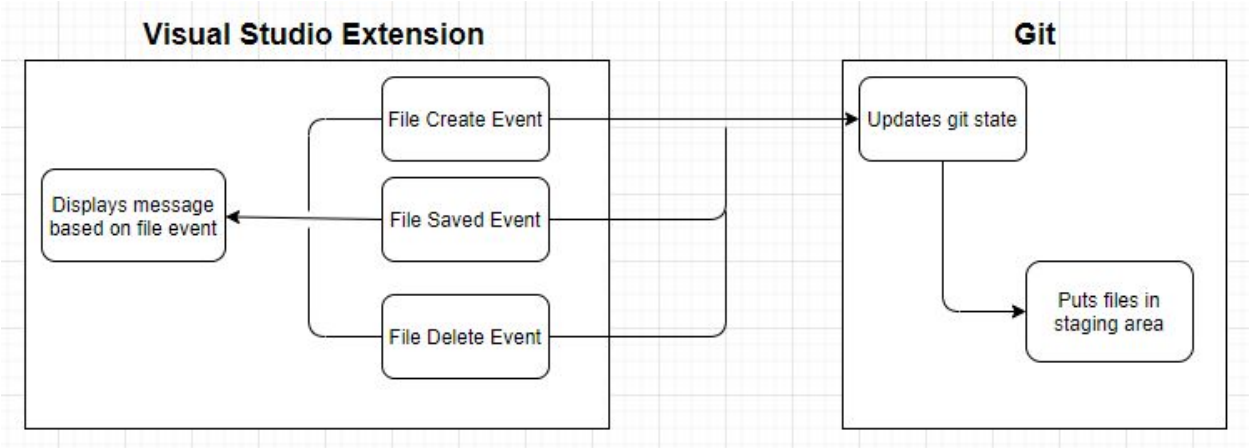

Figure 5: Shows the connectivity and interaction between the VS Code Extension and the git repository that is stored locally on the host's machine.

Preliminary Software Testing Plan

Tests:

- Testing the query tool to ensure queries are generated correctly by providing filtering options and checking if each SQL select statement returns the properly filtered information (date, author, id, message).
- Testing the output of specified queries on the database to ensure the database schema is set up correctly
- Testing the GUI with data that exceeds display boundaries to ensure text wrapping is handled
- Testing each individual filter and pairs of filters along with a few more complex filter sets, to make sure all conditions are set and met correctly
    - Options including, date range, author, specific days of the week or time of day, frequency of contributions, directory
- Testing the application's ability to auto update when it is launched to ensure most up to date data access upon launch. This does not ensure data won't become stale if the application is not closed and relaunched. Evaluation for whether this use case needs to be covered will be conducted.

Features for Alpha Version:

- The code search tool will allow stakeholders to view specific repository information within the view.
- The code search tool will also allow stakeholders to view code aligned with the commit that the user chooses. It shows which parts of the code have been changed.
- Users will be able to filter commits by date ranges, time ranges, directory, file, or author

- Below the code and the commits that are listed, the code search tool will show the contributors, highlighting the specific author who made the changes.

Summary of the State of This Project

The state of the project is as follows:

- The extension has been designed and completed. An extra feature that is optional and can potentially be added at a later time is to include a button to push the staged files to the remote repository, a button to unstage files that have already been staged and another for reverting files.
- Code Search Tool: The parser for the “git log” metadata has been designed and finished. A GUI prototype has been created and the most common queries are a work in progress. A schema for the associated database is being designed in parallel with the GUI, the schema and GUI layout depends on the queries.