Algo-Mate

A. Bochman, L. Dias, N. Lamberson, & D. Lamoreaux

University of Massachusetts Lowell — Software Engineering II Prof. James Daly

Motivation

Whenever programmers need to find code snippets, design patterns, or need help understanding how a design pattern is implemented, they generally find themselves on websites like Stack Overflow and CPP Reference to find solutions to their problems and outlines for commonly used patterns to insert into their code. Whether a programmer is just starting to experiment with these extremely important programming standards, or has to learn them a new language, many spend a large amount of their time that could be used productively searching the internet for the proper syntax for the language they are coding in.

One issue that we face every day as programmers is the constant threat of distraction. According to Gloria Mark of the Department of Informatics at the University of California[1], it takes the average person 23 minutes and 45 seconds to return to a task fully after a distraction, and there are no stronger distractions than the vast world of the internet. A single moment of weakness, or a simple mis-click might land someone at the front page of Reddit, for instance, and they may find themselves reading click-bait articles for hours.

There is a strong need for a solution within the IDE for references on how to complete these specific objectives, and while there are currently some code completion capabilities built into Visual Studio Code, they only cover basic snippets such as loops. Our goal is to add more functionality by including code insertion for several common design patterns with assisted completion and helpful comments to get users on the right track without the risk of distraction.

Objectives

The Algo-Mate team wants to streamline the insertion of common blocks of code to prevent programmers from losing focus of Visual Studio Code (VS Code) while increasing productivity. In other words, we're making tedious design patterns easier to implement across multiple programming languages. Ideally for users of VS Code, gone will be the days where you transition from C++ to Java to Python back to C++ and forget the exact syntax of writing a design pattern. With Algo-Mate, the buttons, controls, and auto-completed code snippets we're proposing will aim to increase the efficiency, productivity, and even enjoyability of writing code in supported languages.

A simple help menu will be placed on the status bar at the bottom of each window will provide the user with a list of the design patterns which are implemented in Algo-Mate in each

[1] https://www.ics.uci.edu/~gmark/chi08-mark.pdf

programming language. When a user selects a language and an algorithm they will be linked to a reputable website which provides thorough information on the use and implementation of the design pattern.

We've settled on implementing support for the C++ language first as the team has the most experience with it. Depending on how development pans out given our ~3-month deadline, support for Java, JavaScript, and Python will be considered, in this order. The language of the file in the editor is detected by the extension. Typing short codes or common phrases associated with a design pattern will allow the user to autocomplete a design pattern by hitting enter or clicking the desired option in the dropdown menu. Additionally, inline comments will accompany the auto-completed code to provide useful information on how the design pattern works, and what data is required to populate the remainder of the algorithm.

Finally, a web page will be developed to act as a hub for the project and will contain relevant information on all aspects of the project. This list includes: an **overview** with a synopsis and eventual demo video; **members** section with information and links to each members' LinkedIn, GitHub, personal website, and email; an **updates** section for release notes, documents, and resources; and finally, **external links** will be provided to private sections and the project's source code.

Milestones

The first major milestone which was completed during the mid-semester week had to be adjusted as our implementation plans changed. Initially by March 5th, 2021, we planned to have a functional webpage on GitHub with appropriate tabs for the project, four implementations of algorithms to be auto-completed and a barebones functionality of the extension is published to the VS Marketplace. Due to our shifting towards implementing completion for design patterns, we were only able to implement one completion: Singleton pattern, however we met all other goals that we set. Having a functional webpage operational early will allow the team to use it as a progress tracker. As we continue the project, new deliverables and release notes will be published regularly as the semester progresses. When the semester comes to pass, the webpage will be completed and will act as an accurate record for anyone interested in the Algo-Mate project.

The other major component of Algo-Mate is the UI. Buttons and options have also already been implemented as part of our second milestone to allow the user to control certain parts of Algo-Mate and use the help menu to find information on the design patterns they wish to insert. The UI was completed by March 26th, 2021 as planned. Auto-completion of four design patterns: Singleton, Builder, Adapter, and Observer will be implemented in C++, Java, JavaScript, and Python by April 19th, 2021.

As we approach the final major milestone, we will complete the final bug fixes, wrapping up the code by April 21st. The demonstration video and final presentation will be completed by April 24th. This will allow for any small edits to be made before the final week of classes, which takes place starting April 26th, 2021. Completing the project before this week is essential because

the demonstration video, class presentation, and final iteration of the project web page depend on it. The Algo-Mate team will spend the weekend of April 24th & 25th rehearsing the presentation before presentations begin the following week.

Approach

Algo-Mate will implement 4 total design patterns, aiming to reach the most common and useful patterns. These design patterns will be implemented in C++ from the start, and then implemented in Java, JavaScript, and Python as the project progresses. The 4 design patterns will fall into 3 categories, creational, structural, and behavioral. Creational design patterns focus on creating objects and initializing classes. Structural design patterns are focused on forming larger structures through the combination of classes and objects. Lastly, behavioral design patterns focus on identifying communication between objects, determining their responsibilities. Within these groups of design patterns, we plan on implementing:

1. Singleton (Creational)
2. Builder (Creational)
3. Adapter (Structural)
4. Observer (Behavioral)

These design patterns will autofill similar to how IntelliSense works in VS Code, retrieving the templates from a JSON file that stores the keywords and their corresponding design patterns. IntelliSense autofill works by analyzing what the user is typing and displays suggestions based on the input. For Algo-Mate, this will behave the same way, where words like “singleton” or “builder” can be typed into a file in VS Code and Algo-Mate will give the suggestion to autofill. This can be seen in Figure 1 below:

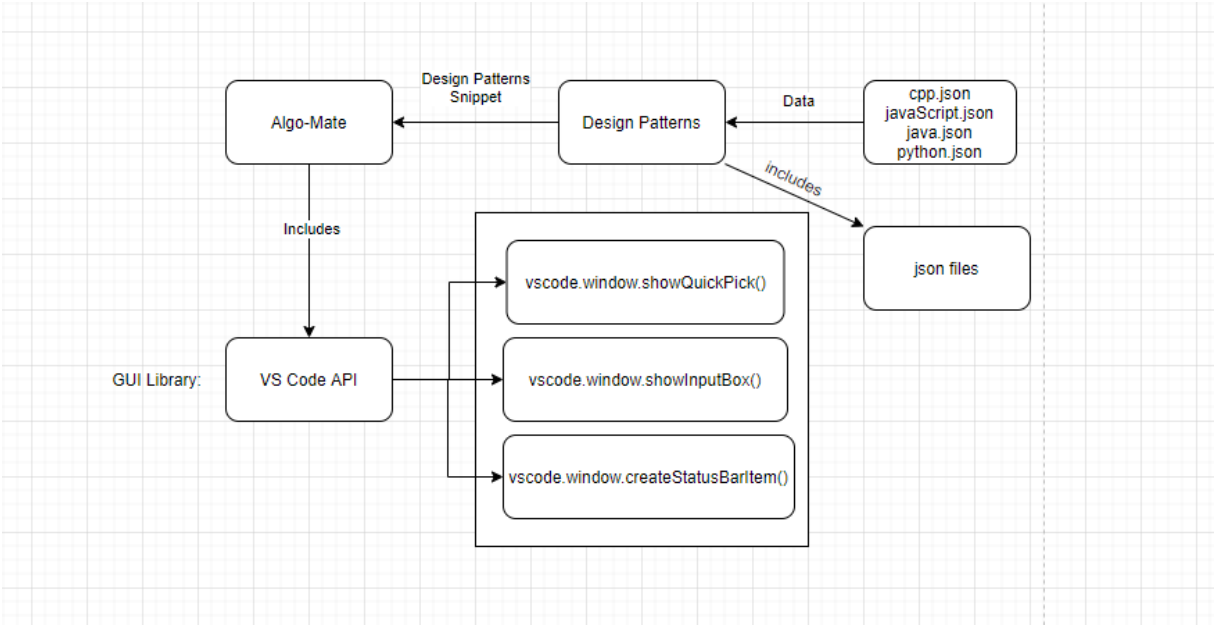

Figure 1. Use Case Diagram - Selecting Design Patterns with Algo-Mate

After determining the correct design pattern to autofill (see Figure 2), the respective design pattern will be placed into the file you are working on, containing comments on how the pattern works. These comments will detail the operation of the design pattern and point the user in the right direction when choosing what variables to make edits to in order for the design pattern to be effectively implemented into their project (see Figure 3). Algo-Mate will give you the option to step through the design pattern, changing key variables and values to update the template to your use case.

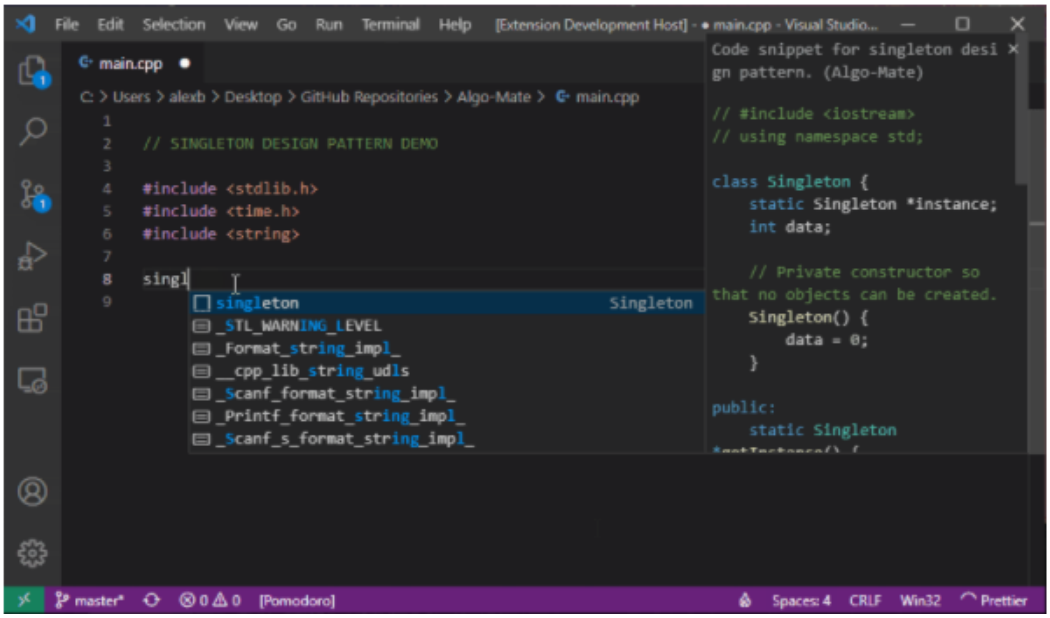

Figure 2. Design Pattern Auto-Completion: Singleton

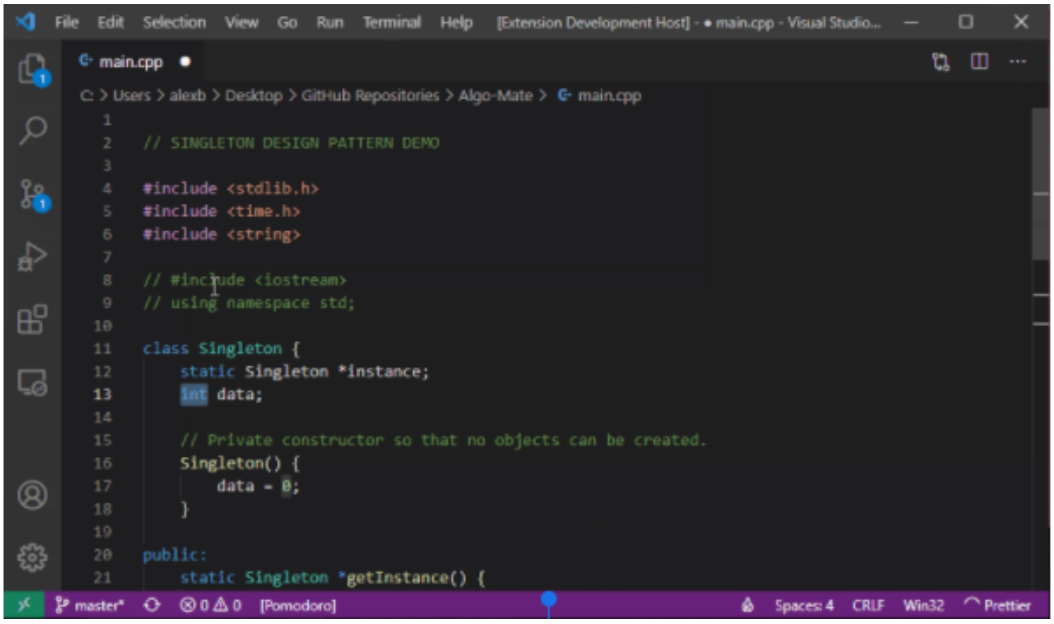

Figure 3. Singleton Design Pattern Template

Alternatively, this can be accomplished through the Algo-Mate UI located at the bottom of the status bar in VS Code (see Figure 4 & Figure 5 below). By clicking the Algo-Mate option in the status bar, we can select a design pattern and the language we wish for it to be written in, and insert it into the file we are currently working within.

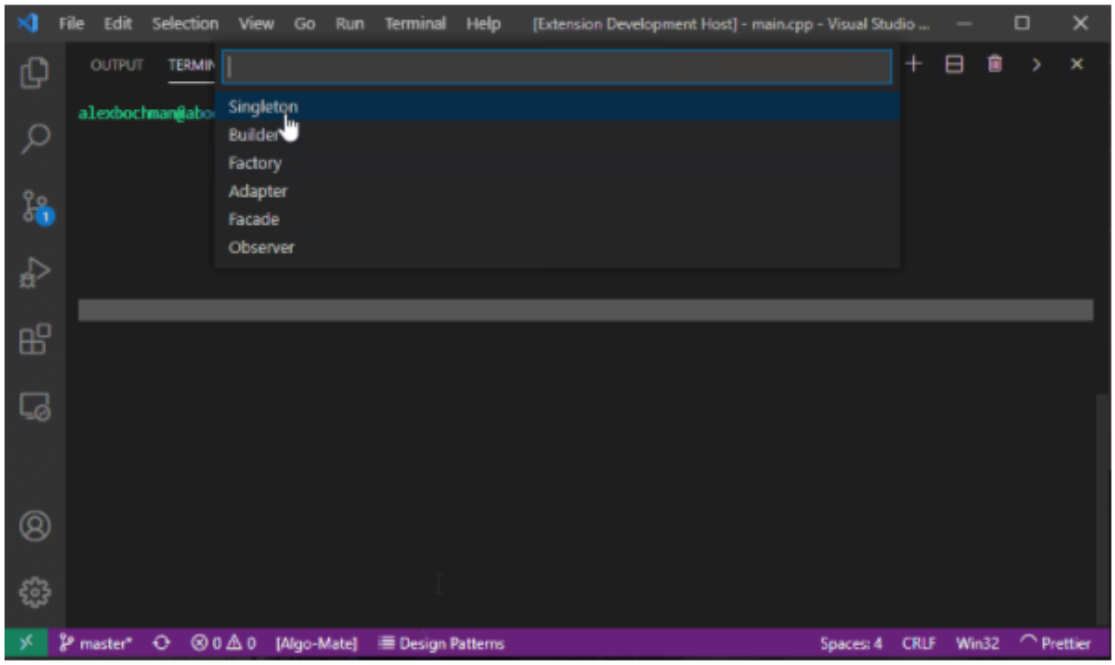

Figure 4. GUI Design Pattern Selection Options

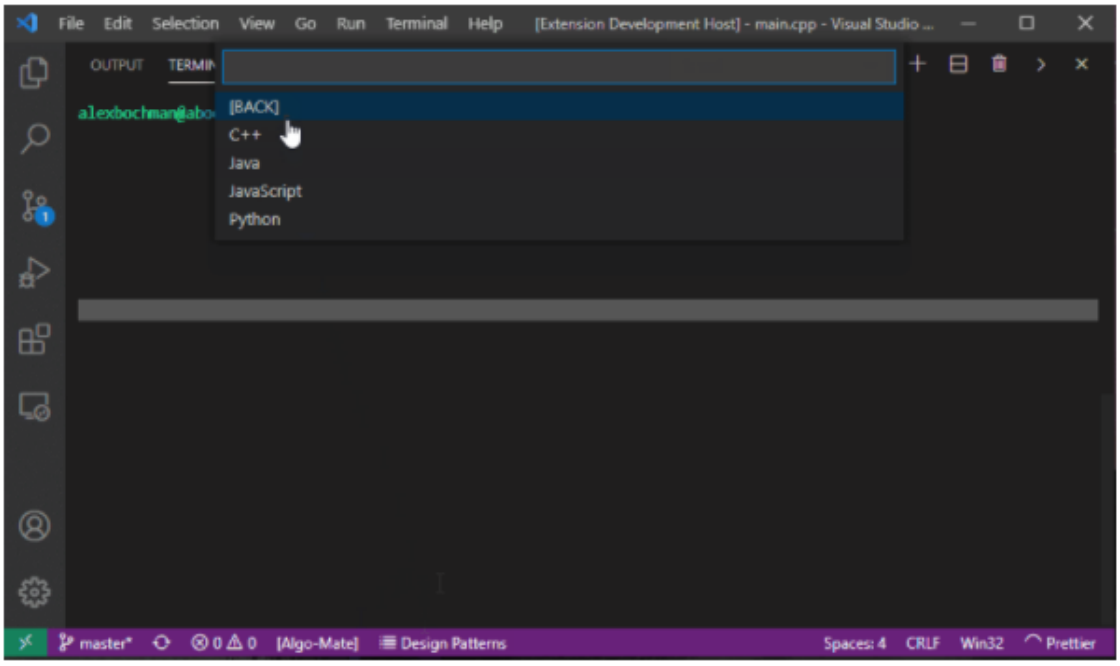

Figure 5. GUI Languages Selection Options

These interactions are depicted in Figure 6 below, where a user opens VS Code, and begins to type in a design pattern, letting Algo-Mate autocomplete the design pattern or by selecting the design pattern through the GUI. We saw this be completed in Figures 2 & 3 for typing the design pattern, or in Figures 4 & 5 for going through the GUI.

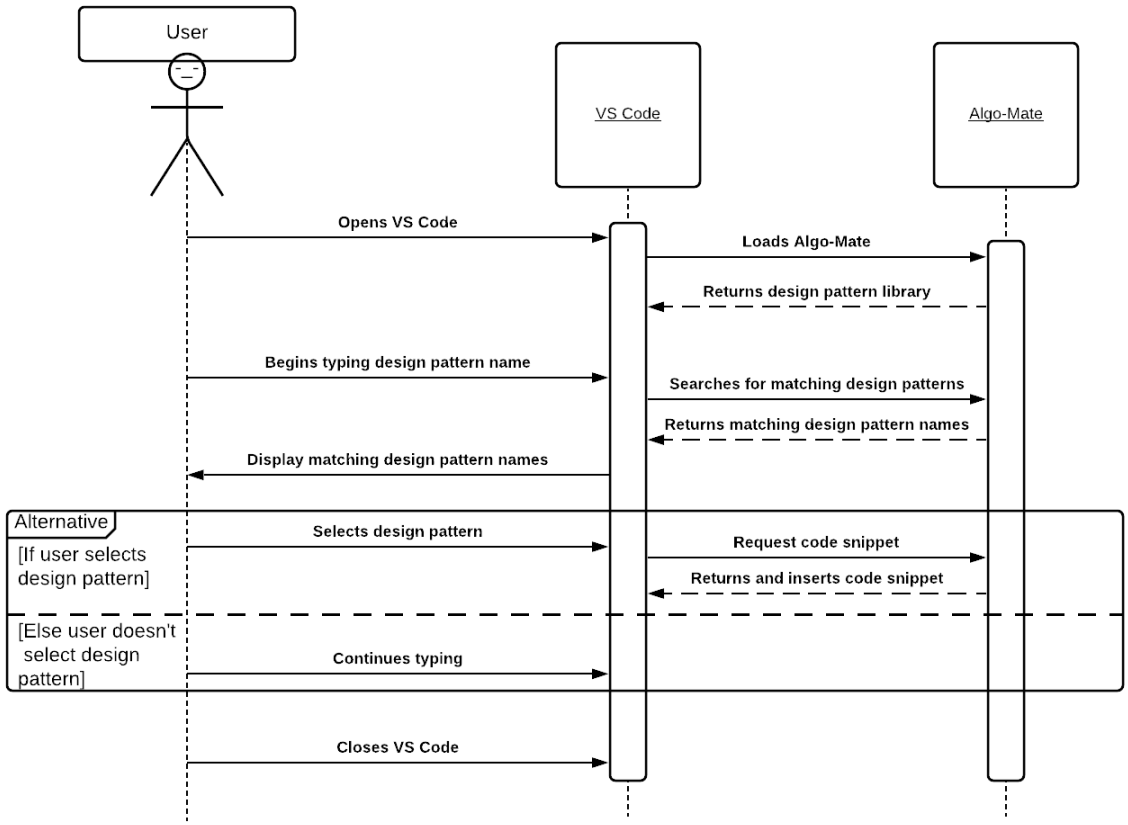

Figure 6. Design Pattern Sequence Diagram - Selecting Design Patterns

Currently, something like this does not exist within the VS Code marketplace, leaving a huge opening for Algo-Mate to fill. Algo-Mate will improve upon the foundations of IntelliSense, while adding robust functionality for design patterns in the process. With the addition of comments to the template, Algo-Mate aims to remove the friction of learning design pattern templates, and allows users to get right into implementing them within their projects. With a deeper understanding of the design patterns provided, we hope users will be able to broaden their techniques without spending hours of time researching exactly how to implement each design pattern from scratch.

Building and Testing

Building Algo-Mate can easily be accomplished by following the instructions[2] below:

Download and Use Extension

1. Make sure Visual Studio Code is installed. Download here: https://code.visualstudio.com/download
2. Download and install the extension
    a. Option 1: From Visual Studio Marketplace: https://marketplace.visualstudio.com/items?itemName=AlexBochman.algo-mate
    b. Option 2: From within VS Code, click on the extensions tab on the left-hand sidebar and search for "Algo-Mate"

Clone, Build, Develop, and Test Instructions

1. Make sure Visual Studio Code is installed. Download here: https://marketplace.visualstudio.com/items?itemName=AlexBochman.algo-mate
2. Clone the latest version of Algo-Mate: https://github.com/alexbochman/Algo-Mate.git
3. Open the Algo-Mate project in VS Code
    a. Make any changes to the files in the "src" folder, or the package.json
4. Click Run > Start Debugging to test Algo-Mate
5. In the VS Code development window that pops up
    a. You'll see [Algo-Mate] on the status bar. Clicking it will bring up UI options.
    b. Typing out any of the prefixes from any of the JSON files within a file that qualifies (.cpp, .java, .js, .py) will display design patterns that can be autocompleted.
    c. After autocompleting a design pattern, points of interest (POI) will be highlighted. They can be modified, and groups of POIs will be changed automatically while still highlighted.
    d. Hitting tab will navigate to the next POI.
    e. Hitting shift+tab will navigate back to the previous POI.

Currently, Algo-Mate has Continuous Integration (CI) being accomplished through Travis CI. It is linked to the GitHub repository and runs testing every time a commit is pushed to GitHub. These test cases are being built out as functionality is added. Currently, we have 2 tests being done through Jest, one that passes the build and one that fails the build, all being shown on the build history on Travis CI[3].

[2] https://github.com/alexbochman/Algo-Mate/blob/master/README.md
[3] https://travis-ci.com/github/alexbochman/Algo-Mate/builds

Results

_____By the time Algo-Mate was graded for its architecture design, we realized the novelty and usefulness of our project was holding it back. We pivoted and came up with a plan to develop an extension centering around design patterns instead of basic algorithms. See the extensive results list below:

- We've implemented autocompletion for the singleton design pattern for C++ within our cpp.json file. Additionally, this allows users to tab back and forth to different points of interest (POI).
- We've published the extension to the VS Code extension store using node package manager.
- We've tested autocompletion between different file types. For example, autocompleting the singleton design pattern will check what the user's current file type is, and will pull the design pattern from the correct JSON file. In short, you won't see JavaScript code in a C++ file. This is due configurations within our package.json file.
- We've published a web page to compile all of the extension's information and documentation.
- Building and testing:
    - Created a YAML file (.travis.yml) to automate building and testing.
    - Configured Travis CI to work with the GitHub repository.
    - Wrote basic tests that failed and succeeded when expected.
    - Wrote tests that implemented VS Code extension API. This is currently an issue as our testing frameworks don't recognize VS Code API, causing errors, and causing our builds to fail.

- GitHub repository: https://github.com/alexbochman/Algo-Mate
- Algo-Mate extension download for VS Code: https://marketplace.visualstudio.com/items?itemName=AlexBochman.algo-mate
- Project webpage: https://alexbochman.github.io/ClassesAndProjects/SW-Eng/Algo-MateProject/Algo-Mate.html#

Ensuring Algo-Mate is working consistently is key to a user's experience. For this reason, we want to ensure the time to insert a design pattern of a certain language is consistent, and does not take long. To ensure Algo-Mate is performing as intended we have created an automated script to run insertions of a design pattern and records the time it takes to complete this operation. This script is located within the "user-interface.js" file, where once the Algo-Mate application is activated, the function getBuildData will run automatically. It will run 100 insertions of a design pattern, recording the time (in milliseconds) it takes to complete the

operation. It then converts this data to a csv called “data.csv” and exports it to the users workspace.

This data allows us to see variations in the insertion times, identifying where issues may be arising. After this data is gathered we can generate a histogram for each design pattern and see the range of insertion times for each design pattern. An example can be seen in Figure 7 below, where we take 100 runs of the Singleton design pattern insertion and generate a histogram showing the times it took to complete this operation.

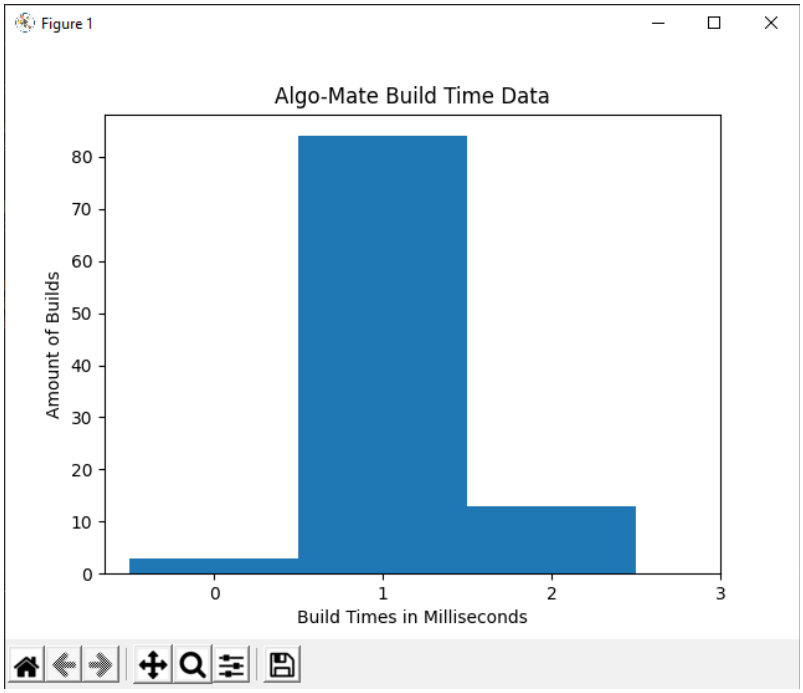

Figure 7. Average Insertion Time For Singleton Design Pattern

This figure can be generated by running the “Algo_Mate_Results.py” file using the following command “python3 Algo_Mate_Results.py data.csv” where the csv file we are using is provided as a command line argument. This csv is generated beforehand using the “user-interface.js” file, as mentioned above. More information on running this can be found within Algo-Mate’s README.

Deliverables

The proposed deliverable for this project is a Visual Studio Code extension which will be used to insert code fragments and design patterns with comments to help the user implement them easily into their code. This plugin will initially be built out for C++, and as the project develops we will implement design patterns for Java, JavaScript, and Python (in this order). The Algo-Mate extension extension itself will be built using JavaScript, and will be capable of

detecting what language the current file is written in, allowing for the correct design pattern to be placed in without the user selecting which language they are using.

For each language, we will implement four total design patterns in a JSON file which will be supported within Algo-Mate: Singleton, Builder, Adapter, and Observer. These will cover the 3 types of design patterns, that being creational, structural, and behavioral. Included in the extension will be a readme file to help users install and quickly begin using the extension. This readme will contain instructions on how to select design patterns, choose the right language, edit the templates, and more.

Risks

The potential risk factors for the Algo-Mate project involves certain conflicts with the structural, behavioral, and creational design patterns. Creational patterns such as the Singleton pattern, may be difficult to complete successful unit tests because most testing frameworks rely on inheritance when it mocks objects. This can be a problem because the singleton class is private and having to override static methods is usually impossible in most languages. The Adapter Structural Pattern can also be a problem due to its overall complexity. If the user were to implement the Adapter Pattern, they may need to create new interfaces and classes that match the rest of their code. Our solution to these issues is to provide comments and tips within the design patterns to help the user understand how it works so they can integrate it well enough with their code. It is important that we have all design patterns be as straightforward as possible in order to prevent future contingencies for the user.

The other issue we may face is the number of patterns in different languages we'll be able to create by the deadline. As mentioned before, we plan on creating algorithms in C++ since it's the language we're most familiar with and will try to implement as many as possible. However, creating algorithms in different languages such as Python, Java, and JavaScript may be more time consuming and it will depend on how much time we have left.

Fortunately, everyone in the group is familiar with the VS Code API[4] and is comfortable with creating this extension in JavaScript. The main payoff for this project is that users will have access to a list of common design patterns that can be used with their code rather than having to look up the algorithm's basic structure. By doing this we get to learn more about the different design patterns ourselves and we plan on spreading our knowledge to the user. Overall, the estimated time we plan on spending on this project is about 3-4 hours a week for each person and possibly more depending on the challenges we may face along the way.

Feedback

- Addressed concerns with Algo-Mate applying to too niche of an audience with just algorithms. Shifted to a focus on design patterns to allow for a broader, more defined use case.

[4] https://code.visualstudio.com/api/references/vscode-api

- Adjusted the potential risks the user may face now they're focusing our project towards different design patterns instead of generic algorithms.
- Fixed the use-case diagram in Figure 1 so it addresses its structural design and background process.
- Adjusted the sequence diagram in Figure 2 as a triggered event instead of a loop and showed how the user interacts with Algo-Mate.
- Addressed issues of explaining UI elements and not showing them in use.

Schedule

Date	**Goal**
February 18th, 2021	Proposal Completion – Compile each component of the proposal that was divided among the members
February 19th, 2021	Proposal Due Date – Proofread and submit proposal before class
February 26th, 2021	Work on auto-completion of design patterns in C++
March 5th, 2021 **Mid-semester week**	● Have a working webpage up this week with these sections: overview, members, updates, and a link to GitHub ● Complete implementation of auto-completion for four design patterns in C++. ● Publish this version on the Visual Studio Marketplace
March 26th, 2021	Finish UI
April 19th, 2021	Finish auto-completion of four design patterns in each of our four languages: C++, JavaScript, Java, and Python
April 21st, 2021	Finalize bug fixes, ensure code is in a polished and functional state
April 24th, 2021	Complete the final presentation and begin rehearsing
April 26th-April 30th, 2021 **Final week of classes**	● Completion of the webpage ● Publish completed extension on Visual Studio Marketplace ● Complete and submit final reports and demonstration video ● Present completed project to the class during this week