Algo-Mate Proposal

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 an algorithm works, they generally find themselves on websites like Stack Overflow and CPP Reference to find solutions to their problems and outlines for commonly used algorithms to insert into their code. Whether a programmer is just starting their career, or has to learn 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 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 random number generation algorithms, data structures, and many other 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 algorithms 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 data structure or an effective random number generator. 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 user interface will provide options to toggle the desired programming language. The user interface will provide buttons on the taskbar and drop-down lists from the command

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

palette. One stretch goal, if time permits, includes the implementation of JavaScript code to read the current working file's extension to determine which language the user is in.

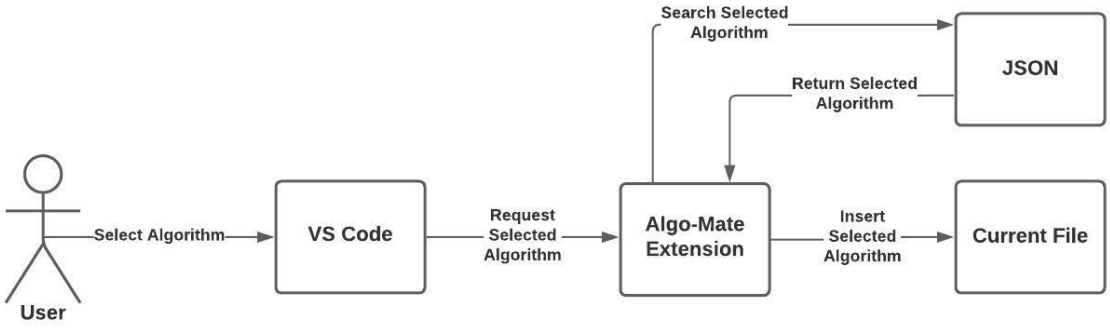

Figure 1: The user interacts with Algo-Mate through the built in VS Code auto-completion

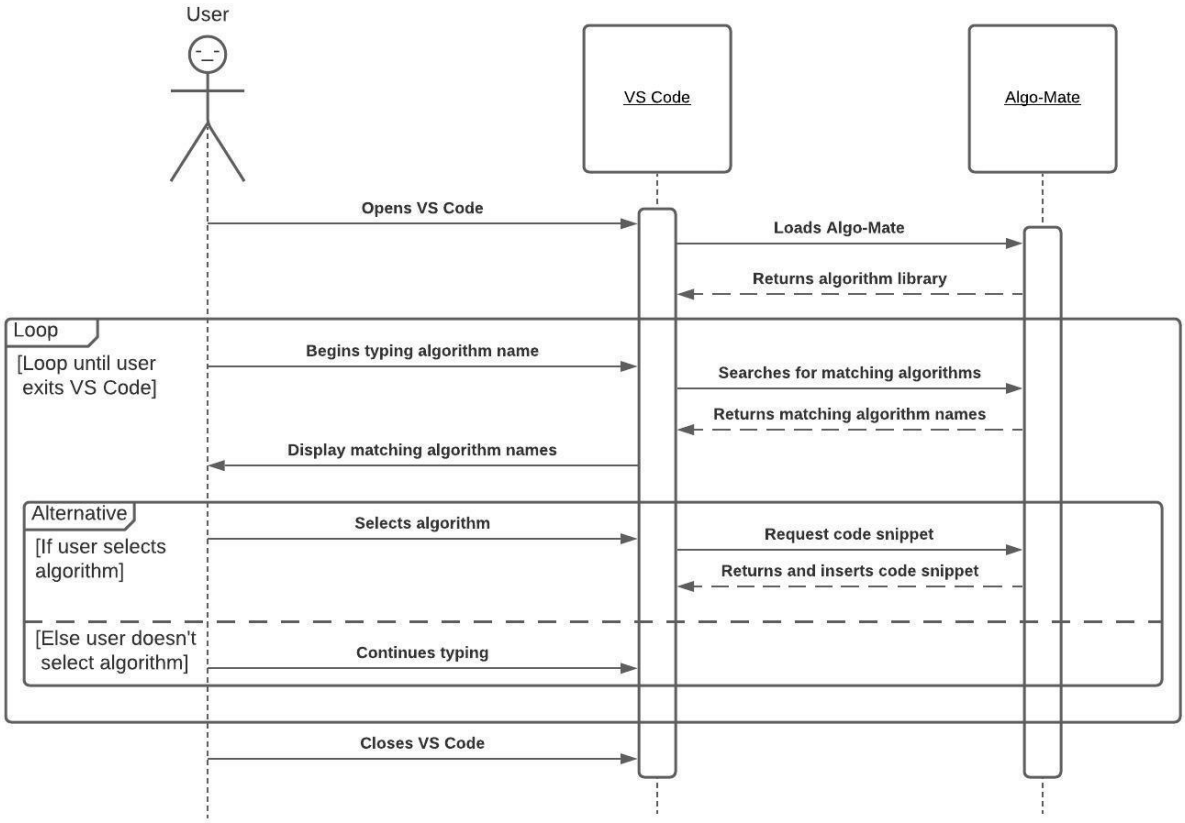

Figure 2: Once VS Code has been opened, Algo-Mate will constantly be monitoring user input for algorithm names in it's database which will be offered to the user through VS Code auto-completion.

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. While the user interface controls, options, or filenames dictate which language the user is working in, short codes or common phrases associated with an algorithm will allow the user to autocomplete an algorithm 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 algorithm 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 will be during the mid-semester week. By March 5th, 2021, the Algo-Mate team plans 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. Having a functional webpage operational early will allow the team to use it as a progress tracker. 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 will be implemented to allow the user to control certain parts of Algo-Mate, such as selecting the type of file the user is working in. The UI will be completed by March 26th, 2021. Auto-completion of four algorithms in Java, JavaScript, and Python will be completed by April 12th, 2021. By April 19th, 2021, we will implement auto-completion of 12 algorithms in each of our four languages: C++, Java, JavaScript, and Python.

The final major milestone will be during the final week of classes which takes place starting April 30th, 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. Additionally, and ideally before this week, the Algo-Mate team is expected to have rehearsed and started recording their presentations.

Approach

Visual Studio Code already includes a way to autocomplete some basic elements like loops, functions, and more with the proper fields filled out with variables to get you started. As the user begins typing, the built in code completion in VS Code will display suggestions based on the input. Other than this, something along the lines of Algo-Mate is not implemented into the VS Code marketplace. This leaves a huge opening for Algo-Mate to fill, while improving upon the foundations set in place by the team at Microsoft. Currently, we are seeing things like this being done with simple calls to a JSON file. It works like how auto-completion does when typing on a keyboard, giving recommendations for what algorithm you are trying to type. After you select it to complete while typing, it will put in a generic version of the algorithm for the language you are using into the code file you are working on, allowing you to edit as you see fit.

We plan to build off VS Code's current implementation, adding functionality for more complex algorithms instead of simple looping or functions. The first of these we plan on implementing are different forms of random number generation and searching algorithms. Algo-Mate will generate these algorithms, combining data structures and functions from different libraries to make developing complex algorithms easier for the user. The user will be able to autocomplete like how VS Code's "IntelliSense" implementation works; However we will also be implementing a GUI to interact with, allowing the user to choose from options available, instead of needing to know the exact name of the algorithm. This allows for more discoverability of techniques, and lets people get a full understanding of the algorithms at their disposal before making their code. With a deeper understanding of the algorithms at hand, we hope users will be able to broaden their techniques without spending hours of time researching exactly how to implement them.

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 only support C++, and as time allows, we will implement code insertion for other languages including Python, Java, and JavaScript. The extension will be written in JavaScript, and will detect what language the current file is written in.

For each language, we will implement twelve total algorithms and design patterns in a JSON file which will support the plugin: efficient random number generation; stack; queue; singly linked lists; doubly linked lists; quick sort; merge sort; insertion sort; bucket sort; depth first search in linear data structures; breadth first binary search in linear data structures and reverse array. Included in the extension will be a readme file to help users quickly begin using the extension.

Risks

The potential risk factors for the Algo-Mate project involves compatibility issues and time management. The biggest issue we may face is having the predetermined algorithms we create be compatible with the user's code. This problem could lead to runtime errors depending on the programming language used and how the user structures their code. For instance, if the user chooses to use one of our search algorithms, the number of parameters may get changed based on what the user is doing. Our solution to this issue is to provide comments and tips within the algorithms 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 algorithms be as basic as possible in order to prevent future contingencies for the user.

The other issue we may face is the number of algorithms 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[2] 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 algorithms 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 algorithms 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

We have addressed all feedback.

Schedule

Date	**Goal**
February 18, 2021	Proposal Completion – Compile each component of the proposal that was divided among the members
February 19, 2021	Proposal Due Date – Proofread and submit proposal before class
February 26, 2021	Work on auto-completion of algorithms in C++
March 5, 2021 **Mid-semester week**	● Have a working webpage up this week with these sections: overview, members, updates, and a link to GitHub

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

	● Complete implementation of auto-completion for four algorithms in C++. ● Publish this version on the Visual Studio Marketplace
March 26, 2021	Finish UI
April 12, 2021	Finish auto-completion of four algorithms in additional languages: JavaScript, Java, and Python
April 19th, 2021	Finish auto-completion of twelve algorithms in each of our four languages: C++, JavaScript, Java, and Python
April 26, 2021	Completion of the project and the webpage
April 30, 2021 **Final week of classes**	● Publish completed extension on Visual Studio Marketplace ● Complete and submit final reports and demonstration video. ● Present completed project to the class during this week

Mockup

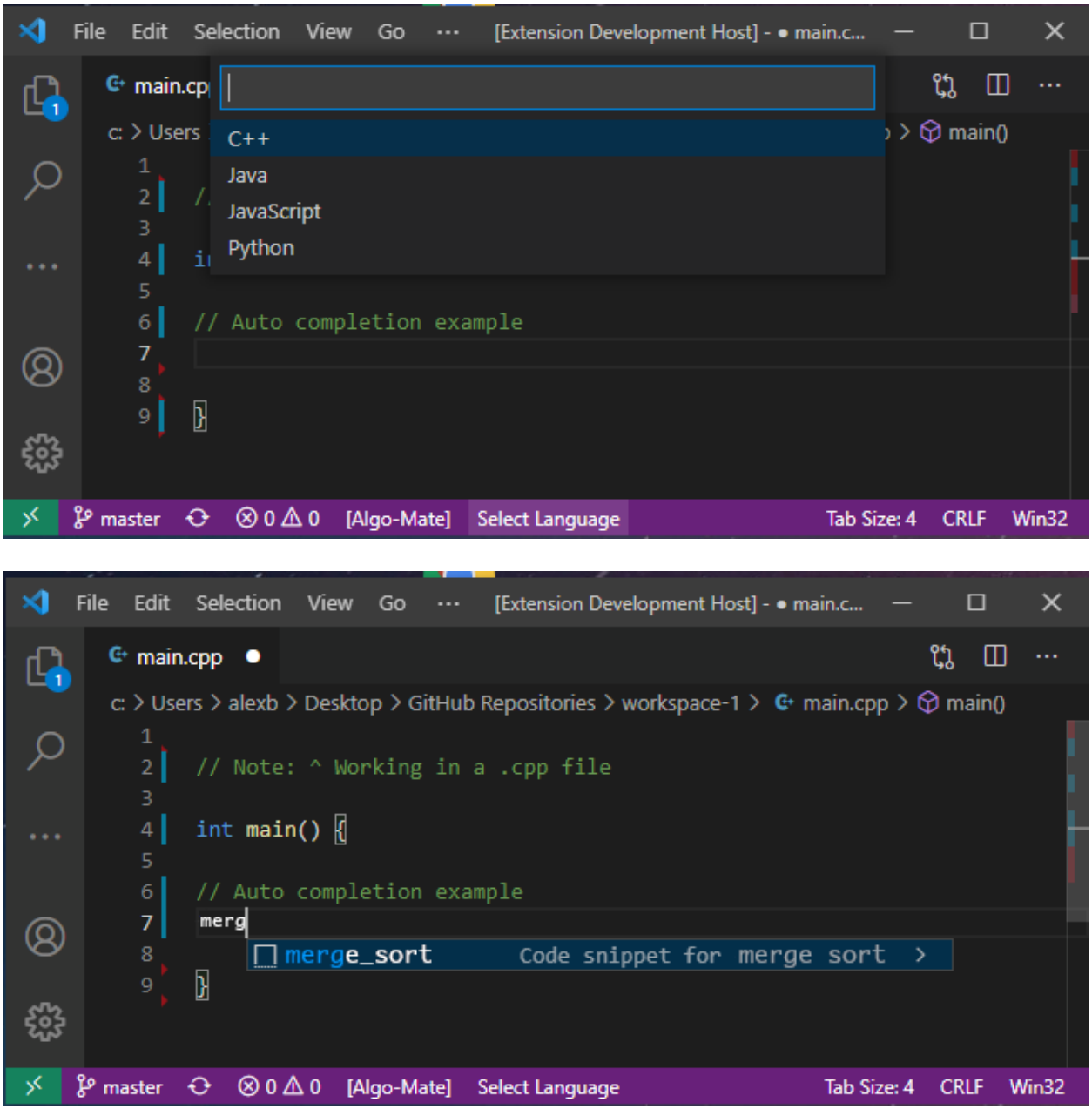

Figure 3: Algo-Mate UI Mockup

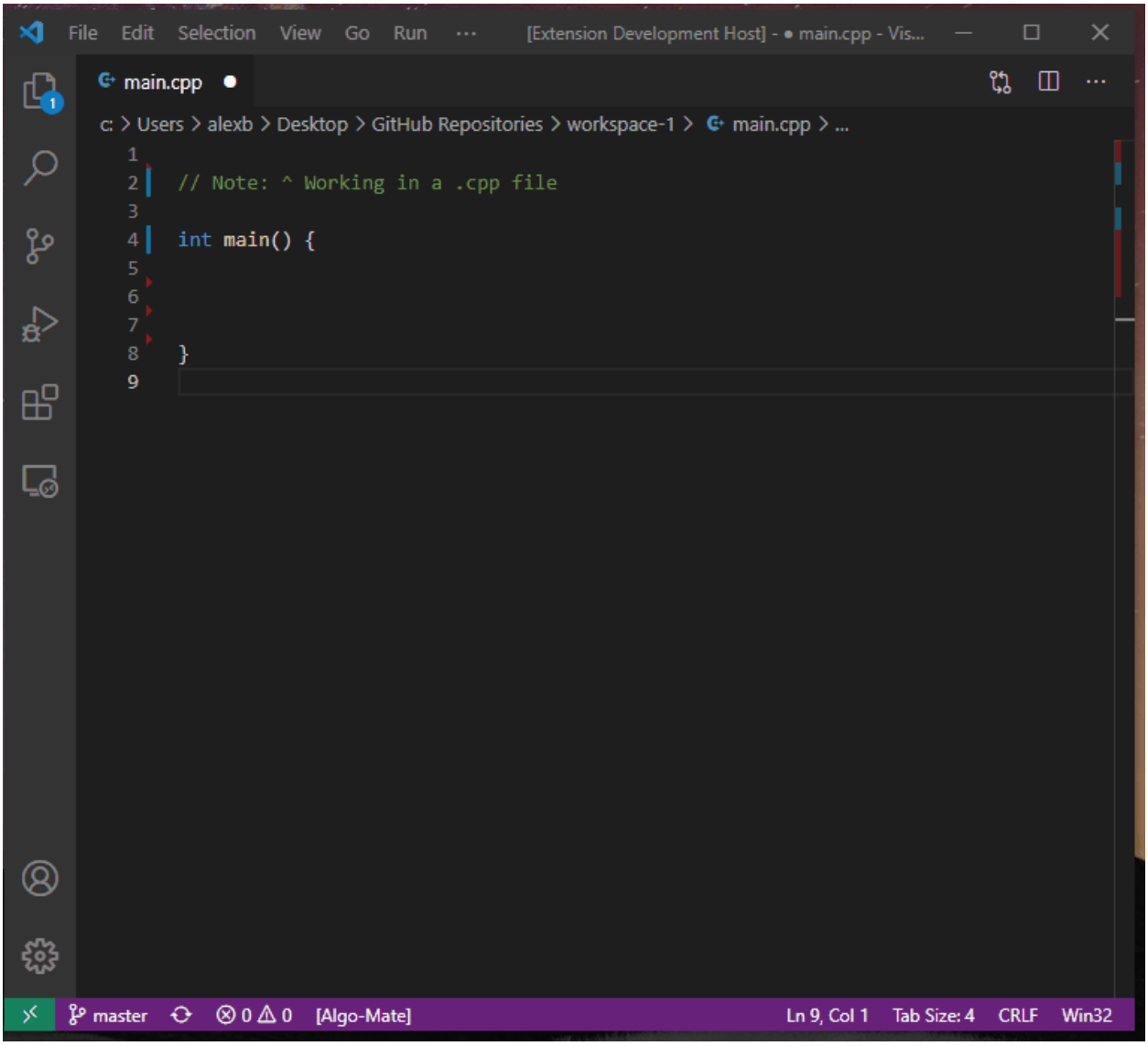

Figure 4: Algo-Mate UI Mockup GIF