University of Arkansas – CSCE Department
Capstone II– Project Proposal – Fall 2019

Hézuò

Michael Tran, Jacob Fung, Markus Sadowski, Daniel Bowden, Hunter Nauman, Derek Taylor

Abstract

There aren't many good programs that allow users to code together. The goal of our project is to make an application that provides a good all-around solution to collaborative programming. It will utilize a peer-to-peer structure where whoever starts the session will act as the server. We plan on basing our application in Python, and connecting the peers to each other through sockets.When our project is finished, it will allow users to work in real time with each other easily and effectively.

1.0 Problem

Many current solutions don't offer synchronized compilation, and it can be hard to manage a project that multiple people are working on. When more than one person is coding a project, communication is key. Hours upon hours can be saved if errors due to miscommunication are reduced. It can be very frustrating to have your code not compile due to miscommunication errors.

Some services, such as Github, provide solutions for collaborative programming with merging and branching. We hope to create an application that removes the need for this, and makes coding together much simpler. Github can be intimidating to find your way around, especially to those new to coding. Even though it allows users to work together, we want a better program that allows updates to occur in real time, attempts to avoid errors in miscommunication and creates a smoother workflow overall.

2.0 Objective

The objective of this project is to create an application that allows users to work together in real time on coding projects.

3.0 Background

3.1 Key Concepts

Collaborative programming-

Collaborative programming entails working together on a programming project. Our project attempts to step this up one notch by allowing users to work together in real time. Collaborative programming can lead to many errors caused by miscommunication, since it is challenging to keep track of who's doing what in a coding project with multiple people.

3.2 Related Work

There have actually been multiple iterations of collaborative coding tools created for developers. Some very popular options include the Live Share extension to VSCode [1], Teletype for Atom [2], and Remote Collab for Sublime Text [3], however, there are multiple other collaborative coding environments that have gained traction in the last couple of years [4]. All of these collaborative coding environments are quite impressive, however, they have their own bugs and restrictions that typically revolve around connection issues among users and restrictions around the directories that users can access. From testing and former usage with Microsoft's Live Share extension to VSCode [1], we came to the realization that only one directory could be specified at the creation of the session (based on the host's local directories) and that in order to change directories the session would have to be closed and a new session would have to be created. Because these issues seem very prevalent in most environments and can be a nuisance when working on large projects or collaborating with a large number of peole, our team would like to work towards improving connection problems and also widen the abilities offered to users such as the ability to share multiple directories and not only those located on the host machine. We're also interested in the possibility of creating a suite of tools catered to class environments so teachers can more efficiently teach elementary coding classes to their students.

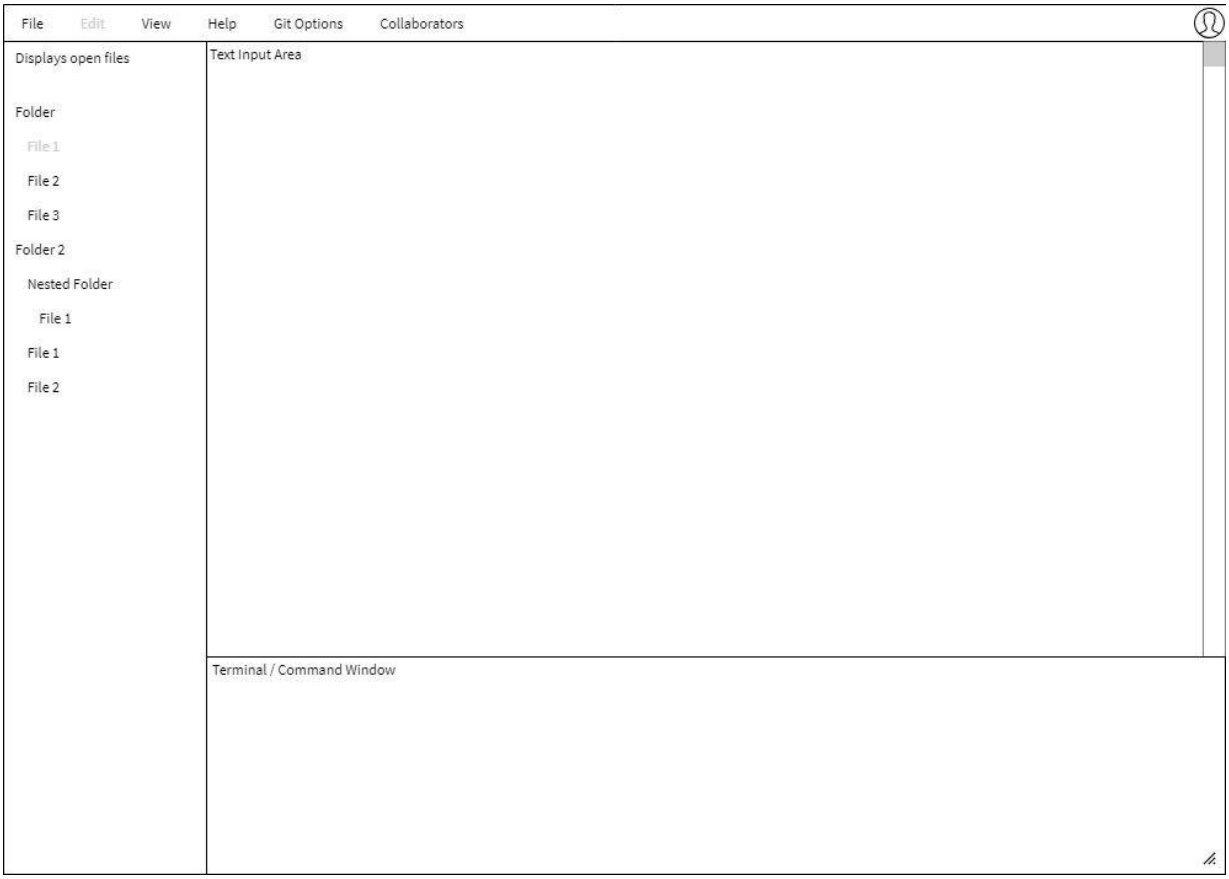
4.0 Design

4.1 Requirements and/or Use Cases and/or Design Goals

There are multiple requirements that come with a collaborative coding environment. The environment must have an integrated text editor with support for multiple languages. The software would also need to have the ability to allow multiple users to connect to a host machine or server and share directories/files that each member/selective members will have access to and will be able to see and make changes to, in real time. Another requirement is that each member should have the ability to build and compile code while others are coding. In order to complete this we think access to multiple terminals for each user would be a necessity as well as collaborative terminals so each user can see the current output/or progress being made without having to run the code themselves.

4.2 [High Level / Detailed] Architecture

Our current thoughts on the high level design are to primarily write the application in Python and use a peer-to-peer architecture to accomplish synchronization between clients. Currently, a high level sample of the final UI design would be something like:



Overall, this would be the same type of layout as a standard text editor but with the added collaborative features with other integrations to assist with group code editing. Since this project is software only and peer-to-peer, the only and main component is the text editor itself. As a peer-to-peer application, it will handle its own connections, groups, and security over the connection established between those in the collaborative group. Originally, the only design detail was that it was to be a collaborative text editor, but the design as refined into a more standard looking text editor along with some additional points. These additional points include a fully shared and interactive terminal at the bottom of the window, a collaborators button to manage collaborators and the permissions they have over the files in the workspace. Hopefully, some sort of configuration file can be used to store the way a workspace is set up and the permissions per file of the collaborators. This way, it is easy to open and resume a workspace with the involved collaborators. To better integrate these collaboration features, there will be a display next to each folder/file to show your available permissions, and settings to control the permissions surrounding the use, execution, and viewing of the terminal. To make this happen through Python, we will be using the built in socket libraries and such from the Python standard library as well as some sort of graphics library such as PyQt to draw and maintain the UI itself. The current main idea for the client synchronization revolves around writing a custom command protocol to queue up and resolve edits.

4.3 Risks

Risk	Risk Reduction
Unauthorized Users	Will have permissions that can be given by the host
Loss of Data	Have either a host that stores the files locally or store the files on a remote server that is accessed
Accessing Data Re otely	Either have access to a terminal that remotes into the host and can access their local storage or on a remote server

4.4 Tasks

1. Understand the scope and resources needed for the project
    a. We start looking into what exists and what resources are available to be used
2. Designing the project and components
    a. Design a plan for how the UI will look, how front/back end will be set up, what programming language, etc.
    b. Assign front and back end tasks
3. Implementation the project
    a. Running the program with the pieces
4. Integrating the project
    a. Using the github client, keep track of changes and bringing in each of our functions into one cohesive program
5. Testing the project
    a. Have some test cases such as helloworld between 3 users to see if it works
6. Revising the project
    a. See the results from testing and fixing bugs
7. 2nd Round of Testing
    a. A 2nd round of testing to see if there are any more bugs and see if program is performing the way we want it to
8. Debug/Clean up Phase
    a. Go back through the program and clean up messy functions and comments

4.5 Schedule

Task	Description	Dates
Understand scope . . .	Get realistic parameters about going forward about the project	11/18 - 12/13
Design	Begin constructing how the project will look and delegating tasks	1/13 - 1/25

Implementation	Begin working on assigned tasks	1/26 - 2/22
Integrating	Bring the pieces together for one functional program	2/23 - 3/7
Testing	Test the limits of the program, looking for errors or bugs	3/8 - 3/21
Revisions	Fix the bugs	3/29 - 4/11
More Testing	Continue testing for special cases or add features	4/12 - 4/25
Final Polish	Clean the code	4/26 - 5/2

4.6 Deliverables

- Schedule: A timeline detailing the start and end of each phase in the development of the project.
- Design Document: Contains a high level plan for the design of the project which will outline the goals for the program in its complete state. It will also detail the milestones we plan to reach over the course of the project (no dates). Lastly, a basic step by step guide to the creation of the project will also be included to help those that might attempt to recreate the project.
- Project Mockup: Visual presentation detailing the features and the look of the GUI we intend to achieve by release.
- Milestone Plan: Planned out milestone goals for the project with specific dates.
- Python Source Code: Source code for the main program as well as any additional relevant code.
- GitHub Repository: Link to the team's GitHub repository for the project.
- Bug Report: document containing a list of known bugs as well as how to reproduce them if known.
- Readme: A simple text file explaining how to install and use the program.

- Final Report: This will contain information such as the project's features, and the overall design ideas behind the source code.
- Zip file containing final report and source code

5.0 Key Personnel

Derek Taylor - Taylor is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed relevant courses such as Software Engineering, Programming Paradigms, and Operating Systems. Most relevant experience stems from the relevant courses completed and also participation in Dr. Luu's CIVU lab by writing primarily Python to accomplish computer vision tasks. Tasks have not fully been finalized and assigned, but will most likely be working on backend functionality with synchronization.

Hunter Nauman - Nauman is a senior Computer Engineering major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed multiple programming courses within the department including Software Engineering and Programming Paradigms. He currently does research in the field of Hardware Security and intends to seek out a Ph.D in the field of Hardware Security as well. At the moment we have yet to fully disperse tasks, however he will most likely be responsible for backend functionality as well as development of the GUI and additional functionalities missing in existing implementations of collaborative coding tools.

Jacob Fung - Fung is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed courses relevant to the project such as Programming Paradigms, Software Engineering, Database Management, and Big Data Analytics. He will be assisting in the development of backend architecture and project management.

Michael Tran - Tran is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Honors Paradigms, Software Engineering, and Database Management. He is responsible for keeping the group on task and working on front end capability.

Daniel Bowden - Bowden is a senior Computer Engineering major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed multiple programming classes including Software Engineering, and is currently taking Computer Networks which will be beneficial for this project. He has had a mainframe systems programming internship at Ensono in Conway, Arkansas. He will be working on the GUI and assisting with the functionality.

Markus Sadowski - Sadowski is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Information Security and Software Engineering, and is currently in Networks. These courses will help to successfully complete this project. He will be responsible for working on the frontend architecture.

6.0 Facilities and Equipment

Possibly need access to a server that can be used to host the session/website that multiple clients can connect to if we decide to make the server the host instead of a local user. This server will store the data on the clients and the data applicable to the session. Access to the Acxiom lab will be required this is where most of the project will be made and designed.

7.0 References

[1] Microsoft. “Visual Studio Live Share: Visual Studio.” *Visual Studio* , 4 Nov. 2019, visualstudio.microsoft.com/services/live-share/.

[2] “Code Together in Real Time in Atom.” *Teletype.atom.io* , teletype.atom.io/.

[3] TeamRemote. “Remote Collab for SublimeText.” *Remote Collab for SublimeText by TeamRemote*, teamremote.github.io/remote-sublime/.

[4] Dennis Gaebel. “9 Real-Time Code Collaboration Tools for Developers”, https://webdesign.tutsplus.com/articles/real-time-code-collaboration-tools-fordevelopers--cms-30494