University of Arkansas – CSCE Department
Capstone II– Final Report – Spring 2020

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 utilizes a peer-to-peer structure where whoever starts the session will act as the server/host. Our application is in Python and connects the peers to each other through websockets. We were able to complete the basic foundation of our project, however, we were unable to complete everything that we wanted to with the time we had. 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 and Visual Studio Live, provide solutions for collaborative programming with merging and branching. Our application aims to remove 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 wanted 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 provide a way for 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. However, this has not yet been fully implemented. 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 people, 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. This would be implemented after the successful addition of the collaborative portion of the code.

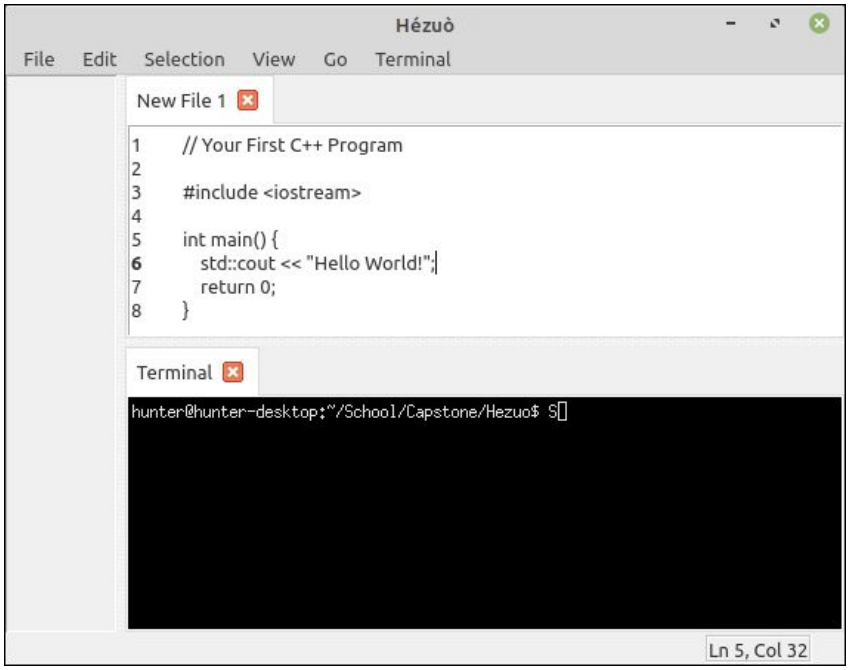
4.0 Design

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. Our program does such, however does not work in real time yet. Another requirement is that each member should have the ability to build and compile code while others are coding. We hope to have this added in the future.

4.2 [High Level / Detailed] Architecture

The following image displays the high level architecture of Hezuo:



The name of our application is listed at the top. Below is the menu bar, consisting of File, Edit, Selection, View, Go, and Terminal. Each of these has their own drop down menu. Files can be created, opened, and saved like any other text editor. The file name is displayed as “New File 1”. Below is the corresponding code. Each file has a line and column number, and the values are displayed in the bottom right corner based on where the cursor is. The terminal can be pulled up successfully on Linux and Mac, but not Windows.

4.3 Risks

Risk	Risk Reduction
Unauthorized Users	In the future we will have permissions that can be given by the host
Loss of Data	Have either a host that stores the files locally
Accessing Data Remotely	Have access to a terminal that remotes into the host and receives the data from the host

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. Front End: Daniel, Michael, Markus, Jake
   c. Back End: Hunter, Derek
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

- GitHub Repository: https://github.com/hjnauman/Hezuo
- Website Containing All of our documents: https://wordpressua.uark.edu/capstone/fall-spring-2019-2020/teams-6-10/team-6-hezuo/
- Project Proposal and Proposal Slides: Details the initial information and goals that we set when proposing the idea of the project
- Final Report: A report of the project and it's status at the end of the semester
- Final Powerpoint: Updated information and pictures regarding the current state of the project as of the end of the school year
- Tasks List: Breakdown of the task assignment for the project
- Schedule: A schedule that describes the overall guideline of the project
- GitHub Repository: Contains all of the source code of the project as well as a readme describing how to install and run the project.
- Poster: Contains a poster design that is a brief overview of the project and what it offers

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. He worked on backend functionality with synchronization as well as the GUI.

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. He was responsible for research and back end functionality which primarily entailed our server client connection code and defining protocols.

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 assisted in the development of the GUI.

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 was 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 Embedded Systems, and has taken Programming Paradigms which was beneficial for this project. He has had a mainframe systems programming internship at Ensono in Conway, Arkansas. He worked on the GUI and assisted 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, Software Engineering, Computer Networks, and Computer Graphics. These courses helped to successfully complete this project. He was partially responsible for working on the frontend architecture.

6.0 Facilities and Equipment

We met weekly at the Axciom lab prior to COVID-19. We also met bi-weekly online in smaller groups assigned with certain tasks. We updated and shared our code through Github.

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-for-developers--cms-30494