//TODO Milestone 2 Deliverable report

I. Task/Problem Definition

a. Remote work is increasingly important in the modern COVID-afflicted society. There exist many tools to facilitate tele-collaboration on all sorts of projects, but few mainstream options for remote tandem software development. This is a problem both educationally and professionally. Having a more effective platform for collaborating on software development can facilitate better planning, and therefor better performance and evaluation of software both by colleagues and instructors. As we see from Hattie and Timperley, better evaluation and feedback can help to improve student outcomes[1]. Our research and resulting solution will focus on how we can provide an effective learning tool to help improve student code quality and speed of completion for group assignments.

A typical interaction related to our research problem would be one that many UGA Computer Science students are familiar with: That of attempting to organize and complete group work remotely. Framed within the Seven Stages of Action[2] we can describe the interaction as follows:

i. Goal: The goal of our participants is to solve a computer programming problem without face-to-face interaction.

ii. Planning: Our participants would need to complete this task through some sort of remote communications application (maybe something we could consider integrating into our solution in the future), and agree on when, for how long, and on what they will be working.

iii. Specify: Participants can currently specify tasks needing complete both verbally and in writing.

iv. Perform: If all goes according to plan, students will perform the specified actions within the specified time frame.

v. Perceive: Participants will perceive the status of their work based on feedback from peers, and effectiveness of their solution based on testing.

vi. Feedback: Results can be evaluated by group members by their own pre-determined metrics, done after each participant has submitted his or her portion of the work.

vii. Compare: Results are compared to the original goals laid out in the planning and goal phases, and actions are taken (or not) based on the level of success.

b. Our goal is to make our interface useful to student coders. Tandem coding and group projects are important practices in both the academic and professional arenas, and while solutions to this problem exist, none are without flaws. Our solution would allow instructors to see how their students are attacking a given problem in real time and offer valuable feedback. It would also allow a coding partner or senior developer to either catch mistakes on the fly or work on a separate section of the implementation, making for more effective debugging and efficient project completion. We will focus on flexibility, autonomy, and feedback tools to facilitate a good learning environment for our users.

II. Analysis of Existing Solutions

a. Our problem of study is remote learning and the ways we can improve it to better suit the needs of students who are just beginning their journey of programming. There are a few existing solutions for our problem of study including the following: Git Duck, Code Together, and Code Sandbox. All this software has specific pros. Git Duck is a recent solution from 2019 that focuses on a collaborative coding experience with built in video chat options. Code Together is a pair programming add-on where one hosts a screen share of their ide for another programmer to join in. Code Sandbox is the most interesting option out of the three because it focuses on web development. Furthermore, CodeSandbox was created with a classroom setting in mind with its Classroom Mode feature, which allows the host to allow participants to edit, as if in a classroom whiteboard scenario. The key is that all of these technologies focus on the pair programming experience of developers. However, that is where the issue lies, these solutions are made for the experienced, not the novice. There is a need for a pair programming experience designed for students, by students, who understand the

struggles of embarking on the journey into programming.

b. There are not any formal, accepted guidelines in place for how any kind of IDE should be developed. However, most IDEs are developed with at least these three core principles in mind: unique features, high performance, and interface simplicity. Which of these principles a particular IDE's development team focuses on is often subject to the primary use case of that IDE. For IDEs primarily used for a specific language or framework, like IntelliJ or Pycharm, having features unique to that programming language to help expedite the development process is essential to focus on. For other IDEs like Android Studio and Xcode, which have emulators running inside of what are already very feature-heavy IDEs, optimizing performance is a primary focus. These IDEs often require more CPU resources than the average IDE. Interface simplicity is something that all IDE development teams must focus on, but it becomes difficult to pack in a lot of features while also keeping the interface from getting too cluttered. So, while this is a core principle of most IDE development teams, it often isn't the top priority. Something similar to IDEs that focuses on a simple interface over a lot of features are plain text editors like Sublime Text and Atom that allow users to start with a very simple interface and add plugins for different functionalities as they continue to use the software.

III. Proposed Solution

a. For our solution have proposed to develop a two-sided IDE, to allow users to engage in pair/cooperative programming. To ensure pair/cooperative programming is successful we wanted to establish an Integrated Development Environment that can be simultaneously edited run by two or more people with results displayed in real time. Because we are focused on those new to programming, we felt it was necessary to have clear signifiers displaying exactly what a user can and cannot do. In order to make our two-sided IDE successful our interface will afford the ability to be seen on multiple user ends. To fortify productive projects within our interface we will have what many interfaces have,

which is distinct signifiers indicating mistakes in code to afford an efficient debugging process.

In comparison to existing solutions, we have found that our solution will be similar in some ways. We want the novice user to be introduced into an environment that is not something to be fearful of. We found that many of the current existing solutions focus on the experienced user, thus the actual interface of current project being worked on look intimidating. We want for our users to have a simple yet efficient IDE for programming. Some of the existing solutions such as Code Together, is software needing to be installed on a user's computer. However, for our approach we found that we wanted our interface to be more web based allowing for data to be centralized and accessible at all times. Allowing users to work from anywhere at any time. We found that the existing solutions are continuously growing and developing to become more efficient and user friendly, however as previously mentioned these technologies focus on those with more experience. We want our solution to be the change that helps those with no to little experience to evolve into successful programmers.

IV. Measuring Success

a. For our design, we will be measuring success three different ways:
    i. Time to learn
        1. We will measure users' time to learn, specifically for completing introductory tasks relevant to our product because measuring how long users take to complete these tasks will help our team understand how simple our design can be, while still maintaining core programming functions. Introductory tasks include creating and removing files, creating and running a simple "Hello World" program, and using the chat and request help features to communicate with the professor.
    ii. Retention Over Time
        1. We will measure users' retention over time over a fixed period because our team wants users to be able to independently use the product, so they require as little assistance from the professor as

possible during lectures. We will form tasks and assign them to the user repeatedly throughout the week, monitoring how little they need assistance in the tasks.

iii. Subjective satisfaction

1. We will measure the users' subjective satisfaction because ensuring that our users feel satisfied with using the product will help guide future design and allow our User base to grow. Interviewing users, administering written surveys about their experience, and rating their liking of different features.

V. Works Cited

[1] John Hattie and Helen Temperley. 2007. The Power of Feedback *Review of Educational Research* 77, 1 (March 2007) DOI: 10.3102/003465430298487

[2] Donald A. Norman. *The Design of Everyday Things*. Basic Books, New York, New York, Revised and Expanded Edition edition, 2013. ISBN 978-0-465-05065-9.