Milestone 3

Clayton Seals ; Bryan Smith ; Oscar Lopez ; Dakota Vickers ; Holly Northen

Human Computer Interaction - CSCI 4800/6800

Dr. Michael E Cotterell

November 11, 2020

User Stories

1. As a Student, I want to be able to access my Instructor's lecture via video chat, so that I can learn class material.
    a. The Instructor starts a video session, then sends the session link to the Student(s), finally, the Student(s) access their Instructor's lecture session via video chat using that link.
2. As a Student. I want to be able to notify my Instructor and other Students via chat function, so that I can give and receive assistance and feedback.
    a. The Student clicks the "Request Help" button on the default Student view, then the Instructor receives a notification in their Help Request Queue, finally, the Student waits until they are visited by the Instructor.
3. As a Student, I want to be able to have access to my own IDE, so that I can code without interference from other users.
4. As an Instructor, I want to be able to make a video session private, so that only Students with the session ID can join.
5. As an Instructor, I want to be able to choose which programming language my Students use, so that it is easier to assist all Students in one session.
6. As an Instructor, I want to be able to divide Students into their own IDE sessions, so that I am able to help Students on a one-on-one basis.
7. As a User, I want to be able to add multiple files to my project to increase the encapsulation of the project.
8. As a Student, I want to be able to view an instructor's text editor while using my own text editor.

9. As a User, I want to be able to run my program on the system, so that I can generate output and function from my program.
10. As a Student, I want to navigate an elementary interface, so that I do not get confused by the amount of options.
11. As an Instructor, I want to be able to allow my students to participate in coding exercises so that they can gain problem solving experience.
12. As a User, I want to work on separate tasks while being able to see what my partner is doing, so I need to be able to save my work easily and navigate the project quickly.
13. As a User, I want my IDE to show syntax errors that are made, so that my debugging process is easier.
14. As a User, I want to be able to compile my code individually so that I do not have to recompile all my files.
15. As a User, I want to be able to have different window themes to make my coding environment more enjoyable (user experience).

Our justification for choosing our main 4 User Stories is as follows. These User Stories are what separate our idea from a standard IDE or in class education. User Story 1 shows accessibility, User Story 2 shows the ability to request help from the professor, User Story 6 shows the ability to select between one-on-one sessions or group sessions, and User Story 10 demonstrates the idea of simplicity, as our focus for our service revolves around into-level programmers.

User Experience Requirements

Focusing on the idea of an Instructor-Student implementation, we organized our 4 main User Stories into an acronym. The User Experience for our service needs to be PEAR.

Private/Public : As an Instructor, I want to be able to divide Students into their own IDE sessions, so that I am able to help Students on a one-on-one basis.

Elementary : As a Student, I want to navigate an elementary interface, so that I do not get confused by the amount of options.

Accessible : As a Student, I want to be able to access my Instructor's lecture via video chat, so that I can learn class material.

Request Help: As a Student. I want to be able to notify my Instructor and other Students via chat function, so that I can give and receive assistance and feedback.

Using PEAR as our focus, we were better able to work on designing our wireframes and the eventual mockup.

Ideation and Preliminary Design

The designs we came up with for User Story 1 involved a large amount of ideas that took shape in multiple ways for the steps involved in accomplishing the user story.

As a Student, I want to be able to access my Instructor's lecture via video chat, so that I can learn class material. While we were designing our initial ideas for logging in and joining a session, we relied upon the work of Don Norman.

Specifically his idea of Discoverability, "Is it possible to even figure out what actions are possible and where and how to perform them?" This quote sums up our intentions when following through with the production of our wireframes. As a group, we decided upon making our program easy to use and understand. The most interesting part of the design process for this user story was that we all had various ideas that culminated in the default student window, join session window, and the initial login window. These different pages had various levels of detail, and as discussed in the video, they all contributed to mockups that are shown further along in the presentation.

For the user story related to requesting help, we felt that it was important to have a “request help” function that was independent from the chat function so that students could privately request assistance from the instructor. The students can do this by using the help request button in their IDE, signified with the text “Request Help”. This will cause the student to be added to the help request queue, which is visible only from the instructor’s interface. The instructor can then select the first student on the queue to be put into a one-on-one session with that student where she can privately ask the instructor for help or assistance. After helping the student, the student is removed from the queue (though the instructor can manually retain the student in the queue if she wants to follow-up again), and the instructor can move on to the next student in the queue or return to the normal session. We also decided that it was best to show the student their place in the queue as a means of feedback after they ask for help, and we also gave the student an option to cancel the request before the instructor gets to her in case she no longer needs help. As mentioned before, we focused heavily on the concept of discoverability when designing our interface, so it was important that students could easily figure out how this function works. We believe that we have accomplished this through good organization and clear signifiers.

When considering how to design a wireframe to address the user story for breakout sessions, we decided to base the functionality, if not the interface design itself, on Zoom. We felt that Zoom offered a fairly robust breakout system[1][2]. We wanted to give instructors a way to trigger breakouts easily, and to have options for customizing how those breakouts would be structured. We wanted to have clear signifiers for our users, and consistent, simple to understand feedback as well. Finally, we wanted to stay consistent with the mapping principles laid out by Norman [3].

Our first wireframe (labeled “First Pass” in the breakout presentation link) had a small cluster of menu buttons in the top righthand corner of the screen. We initially felt that an always visible set of controls would be most useful. First Pass also includes views for when either group of individual breakouts are selected. We show each of these from the instructor and student view. The student group view allows students to collaborate on a single coding space, similar to google docs.

Our second wireframe, labeled “Second Pass” expands on the features needed to allow for a robust instructor experience in the creation of breakout groups. We have included the ability to generate groups randomly by number of members, or by a set number of groups. We have also included the ability for instructors to launch pre-set groups based on his or her needs. We made the decision to include breakout options with a main toolbar on the left hand side of the screen, since once the breakouts are created, a user no longer needs to see options for their creation. Clicking on this tool will open a popup window with options. This wireframe includes artboards to display the instructor's view of creating groups and what an instructor might see upon successful creation of breakouts. We added a pane to the right of the screen to allow for navigation through the various breakout sessions.

Our third alternative, labeled "Third Pass" implements the breakout options again as a standalone button, but this time with a drop-down menu for randomized or predetermined groups. In this wireframe we dove more deeply into how an instructor might configure preset breakout groups, and moved the navigation pane from the right hand side of the screen to the bottom middle section. We felt that allowing instructors to have a variety of methods for creating predefined groups would enhance the user experience and provide flexibility for applications of our potential product. In this wireframe, we can see the instructor view as he or she selects the breakout dropdown, and can then choose to randomize groups, load a configuration, or create a configuration. If the user opts to build a configuration for later use, they can click on students (signified by a green outline on their square), enter student IDs, or enter student emails. Whether the user selects randomized or preconfigured breakouts, they will have the ability to navigate between groups quickly and easily through the panel of breakout room links.

[1] Zoom. Managing Breakout Rooms. Retrieved from

https://support.zoom.us/hc/en-us/articles/206476313

[2] Zoom. Pre-assigning participants to breakout rooms. Retrieved from

https://support.zoom.us/hc/en-us/articles/360032752671-Pre-assigning-participants-to-breakout-r
ooms

[3] 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.

Detailed Designs

Throughout our mockup process you will see how we took the best designs from each alternative designed wireframe and formulated a higher fidelity mockup. For the first user story regarding accessibility to a user's Instructor's lecture. The student or instructor is welcomed with a login screen. You can see we designed a simple but clear logo representing "pair programming". Like many other interfaces the user is able to type in their username and password. In the bottom right corner we can see a login button clearly indicating to a user they will press this in order to login. Once a user logs in they will be met with the screen that actually allows a user to access the instructors lecture. The student will join with their instructor's session ID and the instructor will host with a specified session. We found this to be the best design because the interface displays exactly what a user can do, and the user cannot stray from the given options. Again simplicity is key for our solution.

From here we are met with the default student screen. This screen and screens following will Display how our mock up designs are in parallel with our user story regarding an elementary interface. We can see for the student they are able to minimize or collapse their file directory allowing for more area for one's text editors. If a user desires even more room for their text editor they are also able to press the view button which will then change a students interface to displaying only their text editor. We believed that having this option to display both the student and instructor text editor or simply just the students was important for our design and something many of us found useful. For our button layout in the center we decided to list the clickable buttons on our interface in the order we found relevant. When coding a user will most likely type some code then will try to compile their code. If the code is unsuccessful they are greeted with a clear signifier stating compile error and the user attempts again. Again the user will compile their code and if the code is successfully compiled they will then run their code with the “run” button and the output is displayed in the console. They then will end their program with the “end” button. If there were issues they would be able to debug with the “debug” button and if they are unsuccessful at trying to debug their own code they may request help. We found the view button and theme button were not needed as much when it came to the hierarchy of importance, due to those being more of a user experience. We decided to put those buttons towards the bottom. For the chat we designed the chat to be at the top of the screen to allow more horizontal real estate for the text editors so that a user's code would not formulate on multiple lines as quickly, had been vertically put. We found that the collapsible main menu and other tools would also allow for more real estate for the chat function if wanted.

If we look at the instructor's view it is similar in the sense of simplicity. A lot of the same buttons are displayed except the view button. The instructor has the option to change how many students' video feeds are displayed. We wanted The instructor to have this option of a speaker view of the student who is speaking no student view to allow a more simplistic interface display or the typical default multiple student view. Again our goal was an elementary and simple interface.

Next on our mock ups of user stories we will look at the user story regarding students to receive and request help. Starting with the student we believed the easiest and most concise way for a student to request help was to have a help request button similar to one raising their hand in class. This is a clickable button to which when the student presses it the interface notifies them that their help request has been received, due to the help request hand then displaying “getting help “ . From there the student also sees what number in line they are to receive help. If a student is waiting and realizes that help is no longer needed the user is also given an option to cancel help. This allows the instructor to get to the people that truly need help rather wasting time figuring out who did and did not actually need assistance. We found that the aspect of confirmation that a button was pressed and options to undo an action were something that we specifically preferred. As Don Norman states in his book The Design of Everyday Things, “The human nervous system is equipped with numerous feedback mechanisms, including visual, auditory, and touch sensors, as well as … muscle and limb movements.” Our feedback to the Help request button appeals to human’s visual sensors.

When we make our way to the instructor’s view of help requests we see the instructor has a similar clickable button in the shape of a hand however stating “help requests”. From here the instructor is able to click on the help requests button and the screens found within the interface are resized to allow a new section within the instructors screen. This will allow the instructor to actively see all help requests and whether they are being currently helped or waiting. When looking at the text editor within the IDE, it clearly states what students code an instructor is looking at due to the clear signifier in the top left corner of the text editor. If you look at the help requests window there are green signifiers and a label “currently helping” displaying to the instructor who is currently being helped. The red signifier and label “waiting” displays to the instructor who is currently waiting in the queue. The clickable button “ next student” allows the instructor when finished helping the current student, move on to the next student waiting. When the instructor presses the button “next student”, the text editor displayed then changes to the next student the instructor is helping and the student who received assistance is now out of the queue. To end the instructors help session, we made a clickable button with the label “end help session” signifying that when clicked the help request window will close . We wanted the help request window to be something easy accessible for the instructor at all times, so that if a student needed help the instructor would be aware.

Last but not least we will look into the mockup for the user story regarding an instructor to divide students into their own IDE sessions aka breakouts. For this we will start by looking at the instructor view. We believed that the instructor should have a breakout button signified by a clickable button with a group of people resembling a meeting situation. When the instructor clicks on this button a popup window appears for the instructor. We felt that having a popup window rather than a window appearing on the interface was a better idea because once the breakout groups were made the instructor would not need that window again. The instructor is presented with the options for a breakout group along with an area to type the instructions or prompt for that specific breakout session. The instructor is able to click on what type of breakout groups they wanted to choose and there is a clear check signifier displaying to them their current choice. Once confirmed the instructor is now in a breakout session view. The instructor is able to see each student's code by pressing the buttons labeled “next student code “or “previous student code “. We thought this would be a beneficial aspect to include for an instructor to see each student's code to determine if one was stuck, or how far the students were with the designated task. Once the instructor finds that breakout groups can be over with the instructor will press the button labeled “ end breakout “ to close the breakout groups.

If we look at the students view for a breakout session. The students interface looks similar to the default one however instead of the instructors text editor there are the instructors instructions or prompt for the breakout groups. Again we wanted this window to be collapsible for more area for the text editor to have.

Overall we found our mockups to be designed in a way that is consistent with our target audience which is those new to programming. We have elementary ideals and designs found within our mockups. Through our collaborative design on the wireframes and each alternative design of wireframes we believe that we took the best design choices to be included in our mock-ups and overall solution to the proposed problem.