Final Report

Product Owner:

Trevor Bolton

Scrum Master:

Albin Kyle

Software Engineers:

Mahmood Shilleh
Qusai Amer
Peiman Mohseni
Gowtham Batchala

Date: 11/30/2020

I. Two Paragraph Summary

The main customer for our project, Nayef Yassin who is representing Microsoft (stakeholders) required that we develop a service that allows coders to be interviewed in a group-like manner as opposed to an individual manner. We give them a question that they all work on together in their own IDE's, in which each person's work is dependent on the other. This requires that many users can code at the same time and be able to see each other's IDE's via tab selection.

The Concord application meets this by satisfying the main customer need of allowing users to code at the same time through multiple IDE's while maintaining all of the other features that are important for this to work such as the (calendar, email, sign in, login). This is all done while adhering to the security principles our customer stressed, being from the Cybersecurity division at Microsoft.

II. User Stories:

a. Settings Tab:

This User Story was given a high priority. The current Settings tab on our website is shown in (Figure 1), in comparison to the mockup we generated in Iteration 0 (Figure 2). This feature was successfully implemented and will be deployed on Heroku in this iteration.

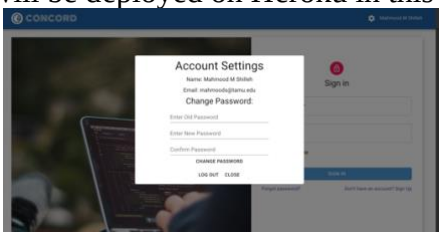

Figure 1: Settings Tab Concord

Figure 2: Settings Tab Mockup

b. Login/Signup Page:

This User Story was given a high priority. Most of this user story was implemented in iteration 2, however the forgot password feature as shown in the mockup and Concord website has been pushed and will be implemented this final iteration. Overall, this feature was also successfully implemented and allows the user to create accounts and sign in; this feature is slightly different in that the login page is now on the home page of concord (Figure 3), in contrast to the original mockup (Figure 5).

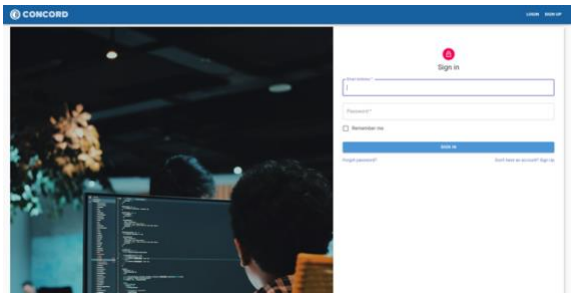

Figure 3: Login Page Concord

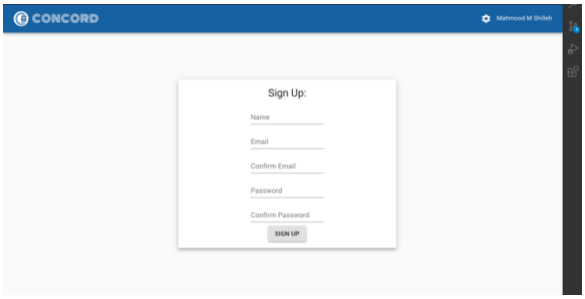

Figure 4: Sign Up Page Concord

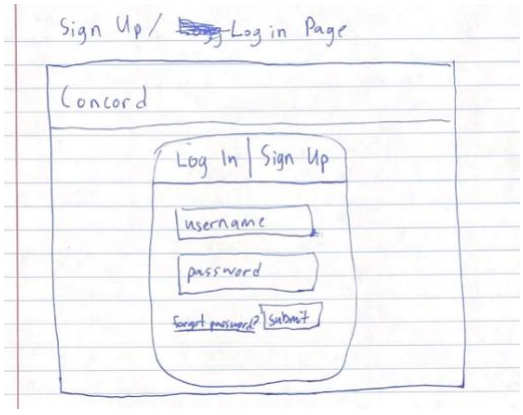

Figure 5: Login/Signup Mockup

c. Past meetings:

Based on the purpose of Concord, this feature held high priority since users can access established meetings. Each meeting holds a unique room identifier where programming sessions occur among

multiple users and can be monitored. Each meeting event holds information concerning session details (e.g. post-session information such as notes from the interviewer).

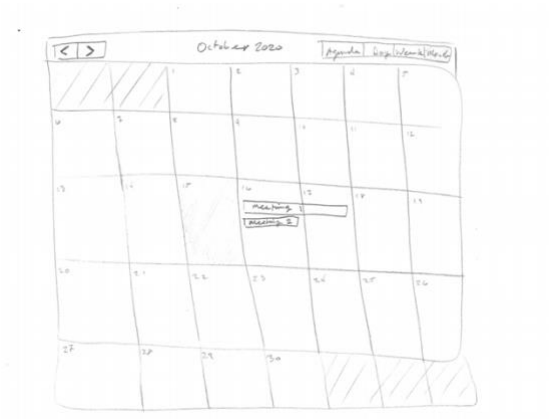

Figure 6: Calendar Mockup

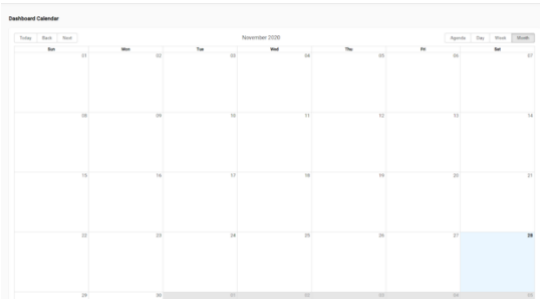

Figure 7: Calendar Concord

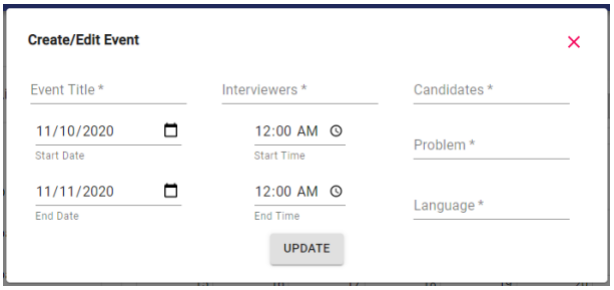

Figure 8: Calendar Concord Create/Edit Events

d. Future meetings:

Much like the past meetings feature, future meetings can be planned on a schedule where events can be generated and shared among several users. Future events can be defined with a title, interviewers, candidates, problem, language, start date, and end date (Figure 8).

e. Coding Tabs:

The coding environment page consists of three parts: Problem statement, coding editor, and terminal. The problem statement contains the question that the user should try to solve. Each user will be assigned to his/her specific tab. Though different users are allowed to communicate with each other (using services like Microsoft Teams), they cannot manipulate the content of other users' tabs.

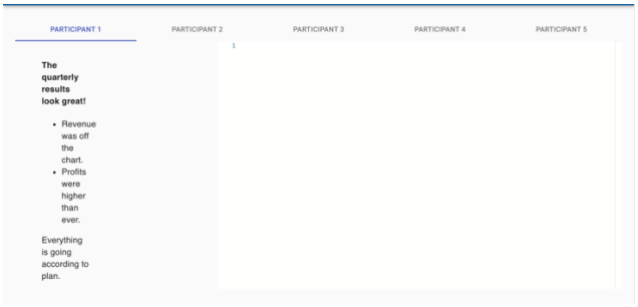

Figure 9: Coding IDE Concord

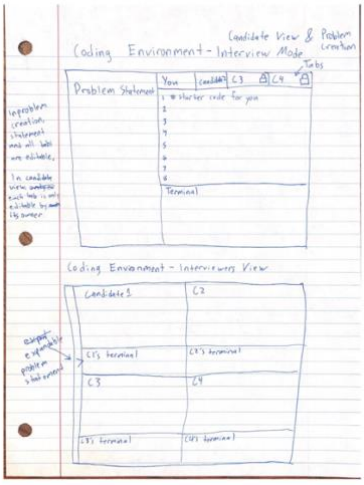

Figure 10: Coding IDE Mockup

f. Contact Us:

This user story was low priority but Nayef suggested it would be a nice and quick thing to add to our website. It was implemented in this final iteration.

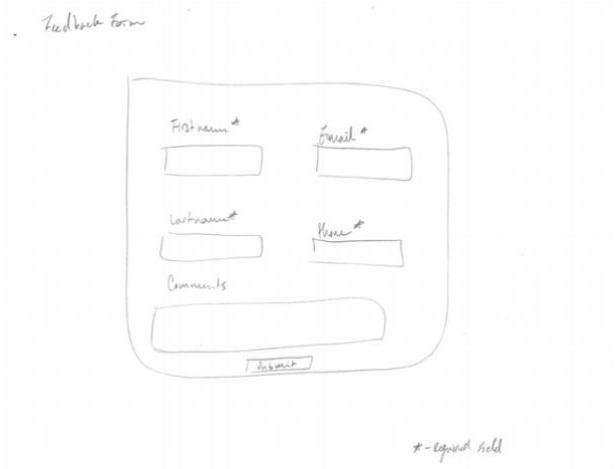

Figure 11: Contact Us Mockup

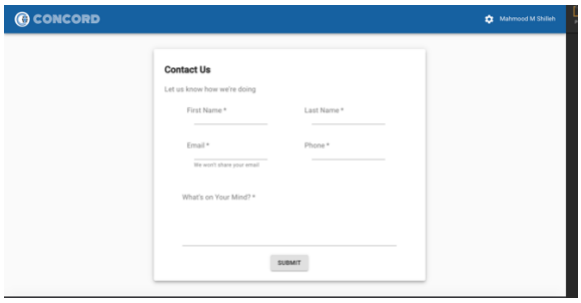

Figure 12: Contact Us Concord

g. Homepage:

This user story has high priority and was implemented in the final iteration. It gives a sleek look to our website and allows the user to access things like forgot password, signup, and contact.

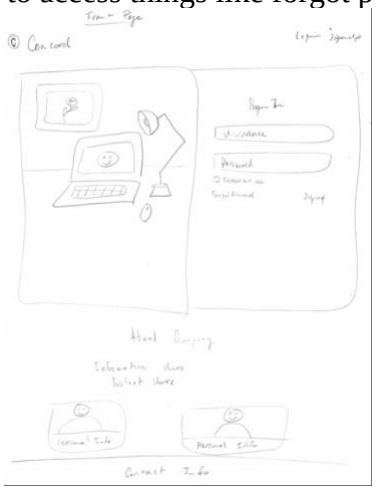

Figure 13: Home Page Mockup

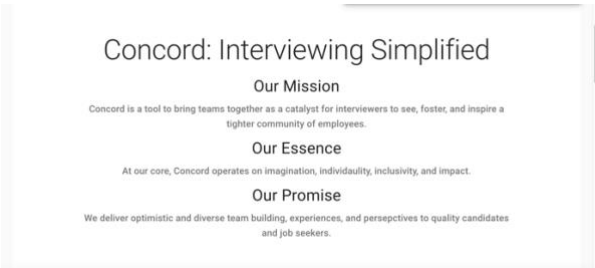

Figure 13: Home Page (Piece) Concord

h. User Stories not Implemented:
    a. Create Join Room for both Users

The interviewer needs to create a room so interviewees can code. This was high priority.

    b. Split Screen Option

So that everyone can see each other's code at the same time, this user story was given high priority. However, the team found difficulty in creating such an IDE. More time is needed.

    c. Chat Box

So coders can chat during the interview, this was given low priority and thus was not focused on.

    d. Allow Others

The interviewer would want to allow others into the coding room for security, but because the lack of time this has not been added.

    e. 3rd Party Video Chat

In order for the interviewer and candidates to attend an event they would need to connect with video and voice. This was high priority but has not been implemented in the front end yet.

    f. Invite Others

(High priority) Everything for this is setup, email, calendar and etc., however the front end and mutations for this have not been written and will not be included in this final iteration.

III. Team Roles:

No roles have changed.

Product Owner: Trevor Bolton
Scrum Master: Albin Kyle Myscich

IV. Iterations:

a. Iteration 1:

This iteration allowed the members to create a conceptual UML of the design diagram concerning the system's architecture. Implemented user stories involved an event calendar, page header, login form, and sign-up form. Each of which were demoed to our customer where feedback inspired

refined and practical user stories. Aside from general implementations, the team dedicated adding to the existing set of user stories involving expectations in the existing implemented features.

b. Iteration 2:

This iteration was spent refining the previous iteration's progress. User stories involving the UI and sign in process were tailored to simplify the process where movements towards implementing the database into the backend. Upon further review, the product customer provided supplemental feedback involving an emphasis on improvements in user communication, interviewer interactions, and login fail-safes. Furthermore, modifications to the codebase were discussed with the product owner and how recommended systems can be integrated.

c. Iteration 3:

In this iteration, the team began improving the previous set of implemented features concerning the UI interaction with the backend and database. The team also introduced Puppeteer testing The product customer provided supplemental feedback concerning general software development advice; recommendations for practical project staging allows for variance in user access (for instance, Heroku pipelining) and role based access control (RBAC) for rights and privileges allowed for a set of users. From the product customer's advice, we decided to continue development with their recommendations in mind for implementation.

d. Iteration 4:

In the final iteration, the team implemented settings options, user identification/privileges, relational events, homepage UI and WebSocket interactions/integration. Furthermore, the general WebSocket server-client interactions were planned and implemented as an extension of the site's general functions, while user emailing now allows users to recover passwords and invite other members. Moreover, database management was improved upon resolving internal conflicts, where test evaluations expanded to WebSocket and database interactions. Form this. the product customer provided remarks concerning improvements in WebSocket security. Although not completely done, full WebSocket integration, and relational backend interactions, and front-page UI are close to completion.

V. Customer Meetings:

a. Oct 2, 2020 12:00 PM:

In this meeting Nayef simply met the team. We all introduced our roles for the project and we explained what Concord would be and what the project was for. We went over the user stories and he helped assess the priorities.

b. Oct 21st, 2020 12:00 PM

In this meeting we discussed some of the features that were implemented for Concord, mostly the frontend for login and signup were finished and Nayef stressed the importance of the forgot password feature. He also mentioned incorporating a “like” or “star” feature in the calendar of Concord so that the user can favor an event to keep their eye on, which we have yet to implement since it was very low priority. He mentioned that the user should be able to sign in with email, candidates having a shared notes feature, note taking feature for the interviewer, and also stressed the importance of communication features during the interviewing process

c. Nov 11th, 2020 8:00 PM

In this meeting we discussed suggestions to integrate hierarchical promotion of testing as per common practices supported by Microsoft Azure products like private/public preview and general availability. The customer also recommended Role-based access control.

d. Nov 23, 2020 12:00 PM

This meeting we addressed integrating entity design principles when adding datasets, this included encapsulation and a single public contract. Another thing talked about in regard to design principles was simplicity. Regarding simplicity, one suggestion was that the consumer of an entity should be able to interact with the entity based on the accepted industry or domain definitions of the entity. The behavior details of the entity should be kept hidden and should be prevented from distorting the interaction. Our customer also suggested that we apply WebSocket safety and security measures such as WSS protocols, tunneling avoidance, validating client input, validating server data, authorization, and origin header.

e. Nov 25, 2020 12:00 PM:

Each of the members reviewed their contributions to Concord and asked for final thoughts and advice. General advice from the product customer entailed UI placement convenience (i.e., moving the login and sign up to the front page) and methods to ensure a quality experience from the user’s perspective, integrating navigable layouts and intuitive design patterns. On another note, code maintenance was also stressed as an important aspect of constructing long-lasting code bases.

VI. BDD/TDD – Benefits:

The BDD and TDD process for this project was mostly driven by Puppeteer, which is a Node library testing framework that provides a high-level API to control Chrome or Chromium. A big issue with Puppeteer is that some of the API control takes a long time to implement, and tests were not passing unless long timeouts were imposed in the code. This makes running the test quite long. However, since the team is new to Puppeteer this is something that can be improved in the future by increasing our understanding of the Puppeteer framework. Additionally, backend testing needs to be implemented in the future using Jest, which is a JavaScript framework for creating, running, and structuring tests. Jest would be primarily used to test both proper creation of entities in the

database and GraphQL code (which was used significantly in the project to handle backend processes).

VII. Configuration Management:

For our development process, we used feature branches in order to implement new features into the code base. Because of this, we ended up having many branches opening and closing all the time. This is a great way to ensure that the codebase stays clean because we can review every single line of code that comes into through Pull Requests. We tried to keep our number of releases to match the number of iterations, so we would make a new release at the end of each sprint.

VIII. Production Release Process:

Some issues that occurred when releasing to production on Heroku were in the initial creation of the project, we needed to handle extra configuration since our project was utilizing Webpack, React, and Node.js. There were also some issues with setting up the Heroku database and ensuring that the credentials matched, but once all of these small processes were set up. Deploying new releases to Heroku was as simple as pushing to a remote repository using git. We also had to create a second Heroku app in order to host one of our separate microservices in order to run the python code that is being written in the interviews. This process was made extremely simple by Heroku.

IX. GitHub, AWS:

No issues were experienced with AWS Cloud 9 since all of the contributors to this project worked on their local machines. GitHub was used very frequently in this project, however. The main issue with GitHub was that some of the contributors were a little unfamiliar with it and there was a learning curve, but that was solved through the course of the project. Additionally, there were some cumbersome merge conflicts that were handled, but for the most part GitHub was extremely beneficial.

X. Code Climate, SimpleCov, other tools:

Tools such as CodeClimate and SimpleCov were not utilized for this project. One of the major tools our team used for this project was the GraphQL playground that allows you to interact with data in our PostgreSQL database through the queries and mutations we generated for our entities. This proved very useful because it allowed us to test our GraphQL code before we connected it to the frontend.

XI. Links:

https://www.pivotaltracker.com/n/projects/2467158
https://github.com/TBolton2000/concord/projects/
http://concord-app.herokuapp.com/
Demo Video Link at:
https://github.com/TBolton2000/concord/blob/master/documentation/Fall2020/FinalVideo.txt