Sprint One Iteration Plan

July 7, 2023

Contents

1 Process

During our second sprint iteration, we conducted a comprehensive sprint planning meeting in person. The meeting covered a variety of different topics and objectives. We allocated tasks according to a clearly defined development schedule. As a result, the team collaborated to thoroughly analyze our product backlog and the outcomes of our initial sprint. This collaborative effort enabled us to effectively estimate the planning for our second sprint, clearly outlining the roles of team members, events, and artifact production

1.1 Changes from previous sprint

Improved Communication: The team collectively decided to improve our communication channels in order to facilitate constant communication within the team

- Why: We are making this change to address existing communication gaps and promote better information sharing. The main issue this change aims to resolve is the potential occurrence of individual implementation that deviates from the team's collective decisions, leading to off-track or non-detailed functionality.
- Success Metric: The success of this change can be measured by evaluating the frequency and quality of communication among team members, as well as assessing the reported levels of satisfaction and engagement. Additionally, the activity level of communication channels such as Discord can serve as an additional indicator to track the effectiveness of this change.

Refinement of User Stories: The team collectively decided to establish a process for refining user stories throughout the iteration.

- Why: We are making this change to address the issue of generalized user stories that require further breakdown into separate tasks and stories. By doing so, we aim to improve clarity and accuracy for team members.
- Success Metric: The success of this change can be measured by tracking the number of user stories that undergo refinement iterations and assessing the overall satisfaction of the team.

Weekly Retrospectives: The team collectively decided to establish a process for conducting weekly retrospectives, with the goal of having a measurable metric to track the progress of the sprint.

- Why: We are making this change to address both communication and feedback. By conducting weekly retrospective meetings, we can analyze areas that require more attention and provide valuable feedback to improve our processes.
- Success Metric: The success of this change can be measured by tracking the implementation of action items identified in retrospectives and observing improved collaboration within the group.

1.2 Roles responsibilities

To ensure effective collaboration within our team, we have assigned specific roles and responsibilities to team members. During our planning meeting, we decided to adopt a variant of pair programming, where two programmers work together on most user stories. This collaborative approach extends to both frontend and backend development, testing, and software implementation.

1. **COB-3 As a user, I should be able to select my university and the respective coursework I've completed so that employers can see my educational background:** Minjun, with the assistance of Vikram, is responsible for implementing a user interface that enables users to select their school, programming skills, and course worker during the signup process.
2. **COB-2 As a user, I should be able to upload PDF documents of my transcript, resume, and cover letters so that I can provide them to potential employers:** Andrew is responsible for implementing Amazon S3 Cloud storage for various documents, including PDFs like transcripts, resumes, and cover letters, as well as user profiles. Additionally, he will assist other developers in implementing the storage on their respective pages.
3. **COB-7 As a user, I should be able to apply to job postings so that I can express my interest in the positions:** Vedat and Ashwin are responsible for managing both the frontend and backend aspects of the functionality that allows users to apply to job postings by submitting a resume and cover letter.
4. **COB-8 As a user, I should be able to attempt the respective coding assessments for job postings so that employers can evaluate my technical skills:** Ansh, with the assistance of Vikram, is responsible for developing a coding assessment page that enables users to attempt a coding assessment using a public API.
5. **COB-14 As a recruiter, I should be able to create a job posting with a title, location, description, and deadline so that I can attract potential candidates and encourage them to apply:** Vikram will be responsible for refining the recruiter job posting page developed during sprint 1. His tasks include making necessary adjustments to ensure seamless frontend-backend connectivity.

6. **COB-24 As a user, I should be able to access a leaderboard so that I can view myself and my user profile:** Ashwin will be responsible for creating a leaderboard frontend with a dedicated backend that connects both the user application page and assessment page.
7. **COB-28 As a user, I should be able to configure my profile to show my educational background and applied jobs:** Minjun will be responsible for the custom user profile which will showcase the users bio and education background.
8. **COB-23 As a recruiter, I should be able to sign up so I can save my data and keep track of my job postings:** Dhruvin will be responsible for creating the recruiter side sign up page which will allow recruiters to sign up and create their own account through a dedicated user interface.

In addition to their specific roles, all team members have a collective responsibility to support and assist one another whenever needed. This collaborative spirit plays a crucial role in fostering a cohesive and successful team dynamic.

1.3 Events

1. **Daily Scrum Meetings:** These meetings will take place online via Discord or Slack, and occasionally in-person at the campus instructional center building. Scheduled at 8:00 pm Eastern Time, each daily scrum meeting will provide an opportunity for team members to share their accomplishments for the day, discuss their plans for the following day, and address any obstacles they may be facing.
2. **Weekly Retrospective Meetings:** The team will be responsible for conducting weekly retrospective meetings. These meetings will be held either online via Discord or in person, depending on the pair program's schedule. The primary objective of these meetings is to analyze areas that require further attention and provide valuable feedback to improve our features.

1.4 Artifacts

1. **Jira Sprint Backlog:** The sprint backlog comprises all the user stories that have been collectively decided for the current sprint. It serves as a tool for tracking the main developer assigned to each story and assessing the overall comprehensiveness of the stories. The sprint backlog facilitates detailed tracking of user stories, including dependencies, linked issues, and progress updates. As tasks are completed or progress is made, team members update the Jira Sprint Backlog to reflect the current status.
2. **Discord Task Assignment Channel:** The Discord task assignment channel is facilitated by the Scrum Master. Before the sprint begins, the team holds a call to discuss task assignments, which are ultimately led by the Scrum Master. This channel enables real-time discussions, clarifications, and negotiations regarding task assignments. The Scrum Master ensures a fair distribution of tasks by considering team members' comfortability and availability, taking into account individual strengths and capacity.
3. **Google Calendar:** Utilizing Google Calendar provides the team with a visual representation of the task schedule, aiding in effective work management and prioritization. The calendar allows for systematic organization of subtasks associated with each task, facilitating a granular breakdown of work within the calendar events.

1.5 Git / GitHub workflow

The team maintains two main branches: "main," which serves as the production branch, and "Developmental," which functions as the development branch. Each team member creates their own branch from the "Developmental" branch to work on their respective user story. The naming convention for these branches follows the format of ¡type¿/COB-xxx-name-of-ticket, enabling easier integration and organization with Jira.

Once a developer completes their work on their current branch, they create a pull request to merge their changes into a corresponding branch that relates to the user story. This approach allows for further collaboration on the merged branch to consolidate the user stories. The pull request undergoes review by the developer who owns the branch being merged into. The title of each pull request includes the name and Jira ticket numbers for better traceability.

After several iterations of merging branches and resolving any conflicts, the changes from the single branch are eventually merged into the "Developmental" branch. This merge is reviewed by the entire team as a whole.

2 Product: Goals and tasks

2.1 User/Recruiter Document Upload

Various pages on the website require users to upload PDF documents such as resumes and cover letters.

- To handle document storage, the team will utilize a cloud-based system like Amazon S3.

- The backend will be responsible for uploading the documents inputted from the frontend to the cloud and storing the corresponding document links in the database.

2.2 Coding Assessment

A user should have the ability to attempt a coding assessment relevant to a specific job posting. This functionality will pave the way for future user stories where users can attempt coding assessments for job postings they are interested in.

- The team will need to conduct research to find a dedicated API that enables code writing on a web browser. Additionally, they will need to explore and find a test case API capable of handling inputted code and assessing its success against predefined test cases.
- To achieve this, a frontend similar to LeetCode will be necessary. Once the user completes the coding assessment, the score will be sent to the backend and updated on the user's application.

2.3 Job Applications

While browsing through job listings, users should have the ability to apply to the specific job posting they are interested in.

- The team will be responsible for developing a dedicated pop-up box that appears when a user clicks the apply button. This dialog box should provide information on how to apply and specify the required documents that need to be uploaded.
- Once the user submits the application, the backend will create an application through an application schema, storing the user's application details. Additionally, the application will include links to the cloud storage where the respective required documents will be stored.

2.4 Recruiter Sign up

One of our top priorities is to allow recruiters to easily sign up and create an account on CoBuild. Recruiters should be able to register with a username and password, as well as provide additional information such as email and pdf documents.

- The team will be responsible for developing a backend API that includes a recruiter sign-up feature. This feature will allow recruiters to create an account by providing their email, username, password, and other relevant information.
- During the account creation process, it is crucial to validate the provided arguments and ensure that the user does not already exist in the system.
- To complement the backend functionality, there will also be an associated frontend. This frontend will provide a user interface for recruiters to enter the required sign-up parameters.

2.5 Leaderboard

Users and recruiters should be able to see a dedicated leaderboard with each posting to see how users have done on the coding assessment.

- The team will be responsible for developing a backend system that fetches the applications for a specific job posting and sorts the job postings based on user scores. This sorting will determine the ranking of the job postings.
- In addition, a dedicated frontend will be created to display the leaderboard, showcasing the users' rankings based on their scores.
- Furthermore, the team will need to incorporate functionality that allows sorting the leaderboard by the submission day, providing the option to view rankings based on the date of submission.

2.6 Profile configuration

Users and recruiters should be able to edit their profile page.

- The team will need to create an edit feature on the frontend, which is then sent to the backend where it updates the database with the new edits.

2.7 Create Job Post

Recruiters should be able to create individual job postings so applicants are able to apply and view the post.

- The team will need to polish the backend of this story so that when the post is created it is then displayed on the /jobs route.

3 Product: Artifacts

3.1 Diagram of a database schema:

Use figma to update the visual representation of the database schema for CoBuild. With the help of this diagram, the structure of the MongoDB database will be made clear, along with the connections between various collections (such as user profiles, job listings, and assessment data). As the database becomes more complicated with the use of different collections, database schema is essential for understanding and organizing the data.

3.2 Documentation for API:

Use tools like Insomnia or Postman to provide thorough documentation for the CoBuild backend API. The request/response formats, authentication methods, and API endpoints should all be covered in depth in this documentation. It aids in making sure that the frontend and backend development teams are properly integrated and communicated with. Futher, API documentation should be provided for developers who need to freshen up on the API's used to develop the coding assessment portion of the website. This will allow for further integration for future endeavours for the product.