CoBuild

> _Note:_ This document is meant to be written during (or shortly after) your initial planning meeting.
> It does not really make sense for you to edit this document much (if at all) while working on the project - Instead, at the end of the planning phase, you can refer back to this document and decide which parts of your plan you are happy with and which parts you would like to change.

Iteration 3

* Start date: July 9 2023
* End date: August 4 2023

Process

During our second sprint iteration, we conducted a comprehensive sprint planning meeting that encompassed a variety of different topics and objectives in person. We allocated tasks based on a clearly defined development schedule. As a result, the team united to extensively analyze our product backlog and the results of our previous initial sprint. This collaborative effort allowed us to effectively estimate the planning for our second sprint, outlining the roles of team members, events, and artifact production.

Changes from previous iteration

List the most significant changes you made to your process (if any).

Splitting the team into Frontend, Backend, and Fullstack:
The team collectively decided to enhance our teamwork and specializations by splitting into key developer roles..

- Why: We are making this change to address productivity and knowledge of team members with their comfortability in handling certain code. The main issue this item solves would be any wasted time spent by developers who are uncomfortable with working with certain code bases.
- Success Metric: The success of this change can be measured by evaluating the amount of user stories completed in correspondence to previous sprints to compare productivity.

* At most 3 items
* Start with the most significant change
* For each change, explain why you are making it and what you are hoping to achieve from it
* Ideally, for each change, you will define a clear success metric (i.e. something you can measure at the end of the iteration to determine whether the change you made was successful)

> *Note:* If you are not making any changes to your process, it means that you are happy with all of the decisions you made in the previous iterations.
> In this case, list what you consider to be the most significant process decisions your team made. For each decision, explain why you consider it successful, and what success metric you are using (or could use) to assert that the decision is successful.

Roles & responsibilities

Describe the different roles on the team and the responsibilities associated with each role.

To ensure effective collaboration within our team, we have assigned different roles to team members, each carrying specific responsibilities. During our planning meeting, we decided to implement a variant of pair programming for each user story, involving two programmers working together for most user stories. This collaboration extends to both frontend and backend development, testing, and software implementation.

COB-4 As a user, I should be able to configure my profile to showcase my resume and cover letter so that employers can assess my skills and experience.

COB-17 As a recruiter, I should be able to track data about my job postings, such as the number of views, click-through rate to external links, etc., so that I can measure their effectiveness.

COB-18 As a recruiter, I should be able to view all my job postings so that I can monitor the positions I've posted.

COB-33 As a recruiter, I should be able to view the coding assessment submissions from applicants for a specific job posting so that I can evaluate their technical skills.

COB-38 As a user, I should be able to go through multiple different job pages so that I can view job postings in a organized matter

-

Events

Describe meetings (and other events) you are planning to have:

* When and where? In-person or online?
* What's the purpose of each meeting?
* Other events could be coding sessions, code reviews, quick weekly sync' meeting online, etc.

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.

Weekly Retrospective Meetings: The team will be responsible for having weekly retrospective meeting meetings. These meetings will be conducted either online via Discord or in person, depending on the pair program's schedule. The purpose of these meetings is to analyze areas that require more attention and provide valuable feedback. features.

Artifacts

List/describe the artifacts you will produce in order to organize your team.

* Artifacts can be To-do lists, Task boards, schedule(s), etc.
* We want to understand:
  * How do you keep track of what needs to get done?
  * How do you prioritize tasks?
  * How do tasks get assigned to team members?

Jira Sprint Backlog

- The sprint backlog contains all the necessary user stories that have been collectively decided for the current sprint. It helps keep track of who is the main developer on the story and the overall comprehensiveness of the story. It allows for 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.

Discord Task Assignment Channel

- The discord task assignment channel is led by the scrum master and before the sprint the team hops on call to discuss assignment of tasks which are led by the scrum master. It allows for real-time discussion, clarifications, and negotiation regarding task assignments. By considering team members' comfortability and availability, the Scrum Master ensures a fair distribution of tasks, taking into account individual strengths and capacity.

Google Calendar

- By utilizing Google Calendar, the team benefits from a visual representation of the task schedule, helping to manage and prioritize work effectively. The subtasks associated with each task can be systematically organized, allowing for a granular breakdown of work within the calendar events.

Git / GitHub workflow

Describe your Git / GitHub workflow.
Essentially, we want to understand how your team members share a codebase and avoid conflicts.

 * Be concise, yet precise.
For example, "we use pull-requests" is not a precise statement since it leaves too many open questions - Pull-requests from where to where? Who reviews the pull-requests? Who is responsible for merging them? Etc.

 * If applicable, specify any naming conventions or standards you decide to adopt.

 * Don't forget to **explain why** you chose this 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.

Product

This entire section is mandatory.

Goals and tasks

 * Describe your goals for this iteration and the tasks that you will have to complete in order to achieve these goals.
 * Order the items from most to least important.
 * Feel free (but not obligated) to specify some/all tasks as user stories.

Job Search and Sorting Functionality:

- Implement job postings search functionality and a UI for users to search based on position or name.
- Create a frontend UI to allow users to sort job postings based on relevancy, deadline, or all.
- Modify the backend to filter and sort job postings based on user preferences.
- Improve search performance and optimize database queries to ensure fast and efficient search results.
- Implement pagination for search results to enhance user experience.

Coding Assessment and Ranking:

- Create a frontend user interface for recruiters to input assessment details, test cases, and general description.
- Implement the backend for creating coding assessments with an appropriate schema and API.
- Implement backend APIs to allow users to write and test their code for coding assessments.
- Create both the frontend and backend to display users on the leaderboard and fetch user assessment details.
- Implement a ranking system based on time, score, and complexity for recruiters to analyze the best candidates.
- Implement secure API endpoints for coding assessments to prevent unauthorized access and ensure data privacy.

User Job Application and Bookmarking:

- Implement the frontend and backend portions for displaying user-applied jobs on their profile.
- Create the frontend on the job posting page to allow users to bookmark a job.
- Implement the backend API to allow users to save job postings.
- Provide an option for users to withdraw their job applications and remove bookmarked jobs.

Email Confirmation on Signup:

- Implement both the frontend and backend functionality to allow users to verify their account via email.
- Send confirmation emails with unique verification links to users upon successful signup.
- Handle email verification and mark user accounts as verified in the database.
- Implement error handling and user-friendly messages for email verification status.

User Profile Configuration:

- Develop the frontend to allow users to save and configure skills for their profiles.
- Implement validation for profile configuration to ensure data consistency and integrity.
- Add the ability for users to update and edit their profile information.
- Enhance the user interface with a user-friendly design for profile configuration.

User and Recruiter Document Upload:

- Enable users and recruiters to upload documents, such as resumes and cover letters, to their profiles or job postings.
- Implement backend APIs to handle document upload, storage, and retrieval using cloud-based solutions.
- Securely store document links in the database and associate them with user profiles and job postings

Artifacts

List/describe the artifacts you will produce in order to present your project idea.

* Artifacts can be text, code, images, videos, interactive mock-ups and/or any other useful artifact you can think of.
* Make sure to explain the purpose of each artifact (i.e. Why is it on your to-do list? Why is it useful for your team?)
* Be concise, yet precise.
  For example: "Build the website" is not precise at all, but "Build a static home page and upload it somewhere, so that it is publicly accessible" is much clearer.

Diagram of a database schema

Use figma to create a 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). It facilitates preparation and guarantees the correctness of the database design.

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.