Sprint One Iteration Plan

June 16, 2023

Contents

1 Process

During our initial sprint planning meeting, we engaged in a range of diverse agendas and assigned tasks according to a well-defined development schedule. Ultimately, the team came together to thoroughly review the product backlog to discuss a figurative team goal. This collaborative effort enabled us to estimate the planning of our sprint 1 leading into the roles of team members, designated events, and artifact production.

1.1 Roles responsibilities

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

1. For the dashboard, Vikram and Ansh form a development pair responsible for creating a user interface that provides users with an overview of all job postings. Users can view additional details by either clicking on the postings or utilizing an API call. As the project progresses, this page will undergo continuous enhancements, making it a crucial foundation for future sprints.

2. The job posting creation team is responsible for developing a user interface that allows recruiters to create job postings. This interface will be seamlessly connected to the dashboard and supported by an API, enabling recruiters to efficiently create and manage job postings.

3. The login and sign-up team focuses on designing and implementing a user interface that facilitates user authentication and account creation. Alongside the frontend components, the team will develop a backend API to handle user login and sign-up processes. Additionally, they will incorporate a JSON Web Token (JWT) for authentication purposes, ensuring secure access to different pages.

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.

1.2 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. Big Coding Session: On June 10th, the team will hold a full-day coding session from 10:00 am to 7:00 pm Eastern Time. This in-person meeting, located on campus, will involve the entire team working together. The session aims to foster open discussions, encourage idea-sharing, facilitate collaboration to overcome obstacles, and maximize productivity as a cohesive group.

3. Quick Weekly Sync Meetings: Each team responsible for a specific area will have weekly sync meetings. These meetings will be conducted either online via Discord or in person, depending on the pair program's schedule. The purpose of these sync meetings is to enable developers to stay connected and collaborate on the development progress of their respective features.

1.3 Artifacts

As this section is optional, we would like feedback on how to incorporate it for the next sprint.

2 Product: Goals and tasks

2.1 Sign Up

One of our top priorities is to allow users to easily sign up and create an account on CoBuild. Users should be able to register with a username and password, as well as provide additional information such as email, resume, and school details.

- The team will need to create a backend API with a user sign-up schema that enables users to sign up and create an account.

- Users should have the ability to enter their email, username, password, and other relevant information.

- During the account creation process, it is essential to validate the provided arguments and ensure that the user does not already exist in the system.

- The backend will have an associated frontend that will showcase a user interface for entering signing up parameters.
- After entering the required parameters, users should be able to simply click a button to proceed and create their account.

2.2 Login

Upon successfully creating an account, users should be able to log in and gain access to the website. After signing in, users will receive a JSON Web Token (JWT) that will be securely stored in the website's cookies. This token will enable authentication throughout the website, ensuring a seamless and secure user experience.

- The team will be responsible for developing a backend API with a user sign-in schema that enables users to sign in and access their accounts. This API will provide the necessary functionalities to facilitate the sign-in process and ensure a secure login experience for users.
- Users will be required to enter their username and password, and the API will query the database to verify if the user exists.
- The backend will be accompanied by a corresponding frontend that will feature a user interface allowing users to enter and sign in with their credentials.
- If the user exists, we will provide a JSON Web Token (JWT) for authentication purposes, enabling the user to proceed to the dashboard.
- The JWT must be saved as a cookie.

2.3 Dashboard

After signing in users should be able to view all job postings on a dashboard from multiple companies and be able to view specific job postings to see more details.

- The team will need to develop a backend that retrieves a job posting from the database using a GET API request.
- The backend will feature a dedicated user interface for the dashboard, showcasing all job postings from the database along with their corresponding pictures and job titles.
- When users click on a job posting, they will be redirected to a screen displaying the detailed information of that particular job posting.

2.4 View Job Posting

After viewing the job postings on the dashboard, users should be able to view individual job postings to get more details.

- The team will require a backend that can retrieve a job posting created in the database using a GET API request and a job post schema. This backend functionality will enable fetching the relevant information of a job posting from the database.
- The frontend of the job posting will provide additional details about the job, including the application deadline, location, and job description. This information will be presented to users, allowing them to gain a comprehensive understanding of the job opportunity.

2.5 Create Job Posting

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

- The team will need to develop a backend that can create a job posting in the database using a POST API request and a job post schema. This backend functionality will enable the storage and management of job postings, allowing recruiters or authorized users to add new job listings to the database.
- The job post will require essential information such as a jobId, title, deadline, postdate, required skills, and other relevant details. These fields are necessary for accurately describing the job posting and providing comprehensive information to potential applicants.
- The dedicated backend will be connected to a frontend that enables recruiters to create job postings using a user interface. Through this interface, recruiters will be able to upload the necessary parameters and details for the job postings.
- Additionally, recruiters will have the ability to view related job postings from the same company below their newly created job posting.

3 Product: Artifacts

3.1 Mock-ups for user interface (UI) design

Use draw.io or Lucidchart 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.

3.2 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.

3.3 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.

3.4 Contribution Instructions:

The goal is to clearly lay out instructions for contributors on how to participate in the CoBuild initiative. Here are guidelines for setting up a development environment, choosing branch names, tracking issues, and using the pull request system. These rules will make it easier to organise a cooperative contribution workflow.