Sprint 2 documentation

Backlog: https://outdora-project.atlassian.net/jira/software/projects/SCRUM/boards/1/backlog

Forecast

Matching Algorithm

User Story: As a user, I want a matching algorithm that considers my preferences so that I can receive relevant and compatible match suggestions.

Details:

- Requirements:
  - Algorithm to consider adventure, skills, and behaviors within a selected mile range
  - Displays match feedback if users match each other
- Dependencies:
  - Design DB table for preferences
  - Design matching UI
- Acceptance Criteria:
  - Algorithm returns accurate matches
  - Successfully displays match feedback when the users match each other
- Estimated in Story Points: 8

Messaging

User Story: As a user, I want a messaging UI that allows me to communicate easily and effectively with my matches, ensuring smooth and intuitive conversations.

Details:

- Requirements:
  - Design and develop a user-friendly messaging UI that integrates seamlessly with the matching algorithm.
  - Implement components for sending and receiving messages, including text input, notifications, and chat history.
  - Ensure real-time message updates and notifications for ongoing conversations.
- Dependencies:
  - Integrate with the matching algorithm to retrieve and display relevant user matches.
- Acceptance Criteria:
  - Messaging UI is intuitive and easy to use, allowing users to send and receive messages without issues.
  - Components function correctly for real-time updates, notifications, and conversation management.
- Estimated in Story Points: 26

Continuous Integration

User Story: As a user, I want an application with continuous integration that constantly updates and minimizes issues to ensure that the matching algorithm for preferences (adventure, skills, and behaviors within a selected mile range) works seamlessly. This includes the algorithm accurately returning relevant match suggestions and displaying match feedback when users match each other.

Details:

- Requirements:
    - Implement continuous integration to support regular updates and issue minimization.
    - Ensure the algorithm considers user preferences and operates within a selected mile range.
    - Maintain functionality to display match feedback when users match each other.
- Dependencies:
    - Design DB table for preferences.
    - Design matching UI.
- Acceptance Criteria:
    - The algorithm returns accurate matches based on user preferences.
    - Match feedback is successfully displayed when users match each other.
    - Continuous integration is implemented with minimal issues.
- Estimated in Story Points: 5

Testing

User Story: As a user, I want an application that incorporates comprehensive checks and balances to ensure that it operates flawlessly and delivers a seamless experience.

Details:

- Requirements:
    - Ensure the integration pipeline includes comprehensive testing to detect and address issues early, minimizing disruptions and maintaining application stability.
    - Test the algorithm to ensure it returns relevant and accurate match suggestions based on user inputs.
    - System enforces password strength requirements and provides feedback on password complexity.
- Dependencies:
    - Application has a reliable connection to the database, with proper configuration for accessing and interacting with the data.

Sprint 2 Rationale

Matching Algorithm

Creating an engaging and intuitive matching interface is essential for fostering user engagement and facilitating meaningful connections. This user story entails implementing features such as swipe interactions, profile displays, and feedback mechanisms to ensure a seamless user experience. The matching algorithm is designed to provide accurate and relevant match suggestions based on user-specified preferences, including interests, skills, and behaviors, within a specified distance range. This ensures that the algorithm effectively connects users who share mutual interests and align well with each other.

Messaging UI and Components

This task involves designing a responsive messaging UI and components. Key elements include creating intuitive messaging inputs and message bubbles, ensuring clear instructions, and implementing validation messages. This effort is crucial as it complements foundational tasks, such as login functionalities, and enhances the platform by enabling users to message people they are matched with. The goal is to provide a seamless user experience and robust customization capabilities within the platform.

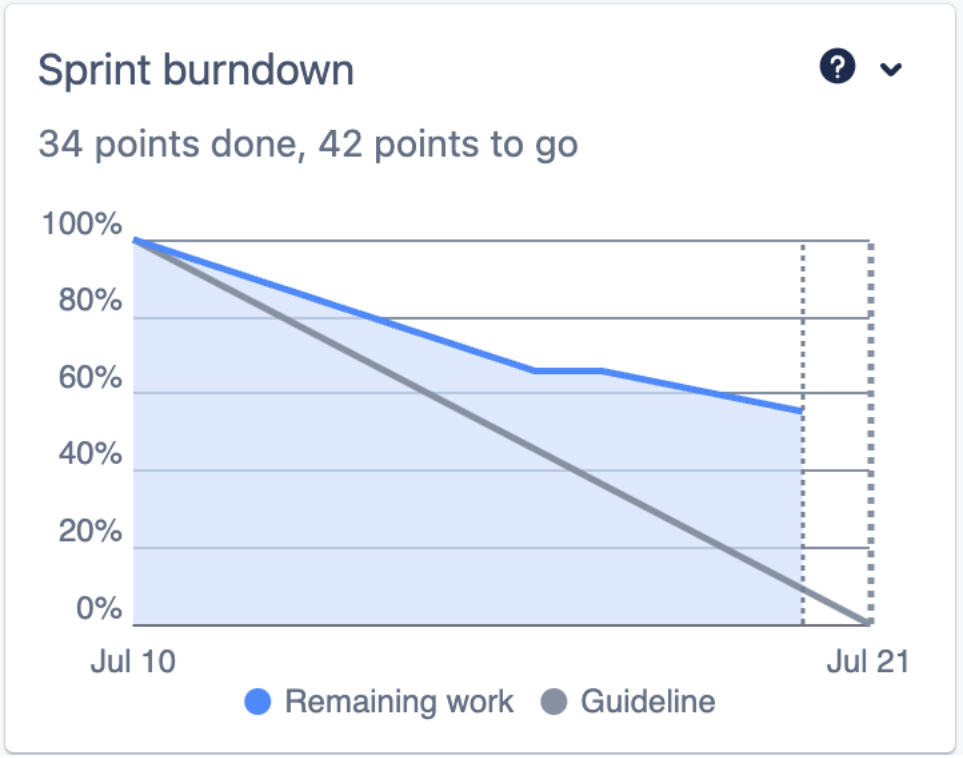
Continuous Integration

Implementing Continuous Integration (CI) enhances our ability to maintain a consistently functional and up-to-date application. This practice involves integrating and testing code changes frequently to catch issues early and ensure seamless operation. The estimated story points reflect the complexity of setting up robust automated testing and deployment pipelines, as well as the ongoing effort to monitor and address any issues promptly. By implementing CI, we aim to streamline our development process, ensuring a reliable and high-quality user experience while maintaining the highest standards of security and performance.

Test Cases

This task involves both developing a user-friendly interface for photo uploading and establishing secure data handling using Firebase. Given the complexity of ensuring seamless photo upload functionality, including error handling for various file formats and sizes, comprehensive test cases are necessary to guarantee the system's reliability and security. These test cases will cover scenarios such as different file formats, size limitations, upload interruptions, and data integrity. By thoroughly testing these aspects, we ensure the robustness and functionality of the photo upload feature. The effort required for this comprehensive testing aligns with the allocated story points, reflecting the importance of a seamless and secure photo upload process within the application.

Burndown Chart



Daily Scrum Questions

What did you do in the last 24 hours that helped the Development Team meet the Sprint Goal?
Alaina: I worked on the messaging component functionality and worked with Parker and Aliyah to help troubleshoot. The messaging component successfully sends messages back and forth now.
Parker: I worked on fixing bugs that were prohibiting the messaging from working. This involved fixing the fetch method as well as adding the chats to both users on either side of the match. Matching now works correctly and both users get a match when one user gets a match.
Aliyah: I assisted with Aliana on the messaging components.
Nishik: I added the profile upload functionality to the app. The profile upload feature takes a photo from the user and adds it to their profile.

What will you do in the next 24 hours to help the Development Team meet the Sprint Goal?
Alaina: I will host a sprint review and retrospective to close out our sprint and I will help anyone with additional tasks they need to complete to finish the sprint.
Parker: Finalize match making as well as add the swipe functionality.

Aliyah: I will continue to work on the messaging screen and I will help with teammates with any additional task.
Cesar: Will continue to work on generating more test cases for the new components that were added.
Nishik: I will clean up the code for the profile upload functionality and then have my team members review the code before merging with the main branch.

Do you see any impediment that prevents you or the Development Team from meeting the Sprint Goal? What are the impediments? What is your impediment removal plan?
Alaina: I believe we are on track this time and there aren't many issues, I feel the requirements for the sprint were a bit better organized. The main functionality of the app is complete with only some finishing touches and presentational materials left to complete. If there are any impediments our plan is to have the most important features complete earlier than the final presentation and sprint 3 closure so we have time to adjust and cut unnecessary work.
Parker: The team has worked well together. We just have some permission issues we need to overcome. The overall component structure is going well.
Cesar: So far the team has done a fantastic job with keeping up with their individual tasks. I don't see any impediments as we wrap up sprint 3.
Aliyah: I feel that the team is working well together and delegating tasks to complete the project in a timely manner. I don't believe the team has any impediments.
Nishik: After dedicating time to work on the project and my schedule being more open, I'm able to work on the project more comfortably to meet the spring goals. I don't believe our team has any impediments going forward with sprint 3.

Pair programming

Alaina: I worked with Aliyah to resolve an error in the messaging component so I could display her UI work and start my work on the functionality. We worked through Microsoft teams to collaborate on the code, and she shared screens with me so I could see her dilemma. Then I opened her branch on my end and found the error relating to a misnamed variable and was able to get the component running.

Parker also worked with me on the messaging and matchmaking code to diagnose an issue affecting both of our work. The messages were being stored as two separate chats, causing them to not register for the other match and I suggested combining the chats into one. Parker did some troubleshooting and worked out a solution using a recipient Id to track the chats.

Continuous Integration Rationale

I chose AWS CodeBuild for our CI setup because it integrates smoothly with GitHub, making our deployment process more efficient. AWS CodeBuild was straightforward to set up and flexible enough to meet our project's needs. Creating the buildspec file, which defines the build instructions, was also simple and intuitive. Another reason for choosing AWS CodeBuild is its ability to grow with our project. As our

project expands, the build service can handle increased workloads without any issues. AWS CodeBuild also has strong security features that ensure our build processes are safe and follow best practices.

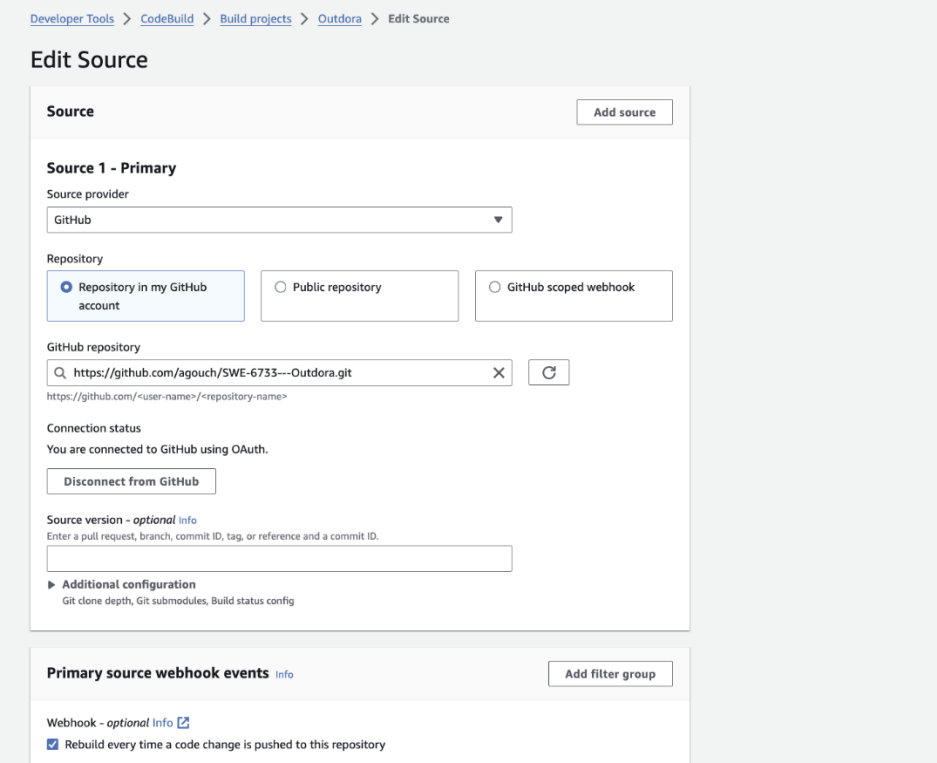
CI Workflow

We have set up a CI workflow that automatically builds our code and runs tests every time we merge to the main branch. This setup helps keep our codebase stable by checking changes before they are integrated.

Workflow Steps

1. Checkout Code: The workflow checks out the code from the repository.
2. Set Up Node.js: The workflow sets up Node.js version 18.
3. Install Dependencies: The workflow installs project dependencies using npm install.
4. Validate Expo Project: The workflow runs npx expo-doctor to validate the Expo project setup.
5. Run Tests: The workflow executes tests using npm test.

Evidence of CI Setup



Sprint Review and Retrospective Video

Sprint 2 Review and Retro.mp4

Demo Video

Sprint 2 App Demo.MOV