Process Deliverables - Agile

Retrospective Summary

In our last sprint, we focused on the initial design and theoretical development of our programmer-matching app. Here are the key takeaways:

- **What Went Well:**
  - **User Persona Research:** Our research into user personas was thorough, giving us a clear understanding of user expectations for a professional matching app. These insights guided our decisions on core functionality and design.
  - **UI/UX Design:** Our swipe-style interface received positive feedback, with testers finding it intuitive and user-friendly.
- **Challenges:**
  - **Refinement of Matching Algorithm:** Designing an effective algorithm to match users based on skills, experience, and project preferences has been challenging. Determining which factors are most impactful and how they should be weighed requires more analysis and testing.
  - **Selection of Match Criteria:** Deciding on the optimal set of criteria to ask users at signup for an accurate matching experience has proven to be complex, as we balance comprehensiveness with simplicity.
  - **User Familiarity and Safety:** Integrating features that foster a sense of comfort and familiarity between potential matches is crucial. We need to strike a balance between providing enough information for users to feel connected and ensuring privacy until they feel ready to meet.
- **Action Items:**
  - **Refine Matching Algorithm:** Continue developing the matching algorithm with an emphasis on skill-based compatibility and common interests.
  - **Finalize Matching Criteria:** Finalize the list of user-provided factors that will be incorporated into the matching algorithm to ensure relevancy and ease of use.
  - **Develop Familiarity and Privacy Features:** Design and implement features that allow users to gradually learn about one another over time, such as staged profile reveals, connection milestones, or interactive prompts. This will promote a sense of familiarity and build trust without compromising privacy.

Prioritized Tasks

For our upcoming sprint, the focus is on refining matching and familiarity features, improving and finalizing the matching algorithm and moving closer to successful development. Here's the plan:

- **Matching Algorithm Refinement:**
    a. Prioritize the development of an algorithm that accurately matches programmers based on skill levels, languages, project preferences, and availability.
    b. Implement machine learning to improve match accuracy based on user interactions and feedback.
- **User Familiarity and Privacy Features:**
    a. Design staged information reveals or interactive prompts to allow users to gradually learn about their matches.
    b. Develop "connection milestones" that trigger more profile information or conversation starters as users engage, helping build familiarity and trust without compromising privacy.
- **User Profile Enhancements:**
    a. Allow users to add links to GitHub, LinkedIn, and portfolio sites.
    b. Introduce a bio section for users to describe their interests and goals.
    c. Implement skill tags for easier matching and searchability.

Goal for Next Sprint: Release a beta version that includes refined matching, gradual profile reveals to enable test users to experience the core features and provide valuable feedback for further development.

Requirements Analysis

Five hypothetical non-functional requirements for our system

1. Usability requirement: The website must show the profiles in an easy-to-read manner.

The website's purpose is to help programmers find partners of similar skillsets and schedules. Thus, the profiles must be easy to read for users to easily achieve this goal.

2. Reliability requirement: The website should be able to recover from crashes quickly

Websites crash often, so the ability to recover from them quickly to allow users to continue using it.

3. Performance requirement: The server must be able to support around 5,000 people at once.

The website will be used by many people, so it should be able to be used by many people at once.

4. Supportability requirement: Must be able to run on Windows, Linux, and Mac, as well as the major web browsers (Chrome, Edge, Firefox, etc.)

There are many different operating systems and browsers that the users may be using, so the website should be able to support the major operating systems and browsers.

5. Implementation requirement: Must use Javascript/HTML5/CSS

These are some the most common languages for web development, so it will be easy for new programmers to be able to understand and work on the code.

Five hypothetical functional requirements for our system

1. User Profile Creation:

Users (engineers) can create a profile that includes their skills, experience level, preferred work styles, availability, and goals.

2. Skill-Based Matching Algorithm:

The system will use a matching algorithm to identify and suggest compatible partners based on users' skills, work style, availability, and project goals.

3. Real-Time Availability Display:

Users can update and view the availability of potential partners in real-time, allowing them to find partners whose schedules align.

4. Communication Platform Integration:

The system provides an integrated communication platform (e.g., chat or messaging) to allow matched partners to communicate directly through the website.

5. Feedback and Rating System:

After each collaboration, users can leave feedback and ratings for each other, helping future users to make informed decisions.

Use Cases

Here are five formal use cases for the Pair Matcher system, along with descriptions.

Use Case 1: User Profile Creation

Description: A user creates a profile by entering personal information such as skills, experience level, work style, availability, and goals.

Main Flow:

The user navigates to the profile creation page.

The system prompts the user to enter information about their skills, experience, work style, availability, and goals.

The user submits the information.

The system saves the profile and confirms profile creation.

Sequence Diagram:

Actors: User, System

Key Interactions:

User inputs profile information.

System saves information and confirms profile creation.

Use Case 2: Matching Partners Based on Skills

Description: The system matches users based on their profiles, particularly skills, experience, and work styles.

Main Flow:

The user requests a match for a project.

The system analyzes the user's profile and searches for compatible partners.

The system displays a list of potential matches based on compatibility scores.

The user selects a potential partner from the list.

Sequence Diagram:

Actors: User, System

Key Interactions:

User requests a match.

System performs matching and returns compatible profiles.

Use Case 3: Viewing Partner Availability

Description: Users can view the availability of potential partners in real-time to ensure their schedules align.

Main Flow:

The user views potential matches.

The user selects a partner’s profile.

The system displays the selected partner’s availability.

The user confirms compatibility in terms of availability.

Sequence Diagram:

Actors: User, System

Key Interactions:

User requests availability information.

System retrieves and displays availability data.

Use Case 4: Messaging Matched Partner

Description: Once matched, the user can initiate a chat or messaging session with their partner.

Main Flow:

The user selects a matched partner.

The user initiates a chat session.

The system opens a messaging window.

The user and partner exchange messages within the platform.

Sequence Diagram:

Actors: User, Matched Partner, System

Key Interactions:

User initiates chat.

System opens messaging window and relays messages between users.

Use Case 5: Providing Feedback and Ratings

Description: After a collaboration, users can rate and provide feedback on their experience with the partner.

Main Flow:

The user navigates to the feedback section of their profile.

The user selects the partner they collaborated with.

The user submits a rating and feedback about the partner.

The system saves the feedback and updates the partner's profile rating.

Sequence Diagram:

Actors: User, System

Key Interactions:

User inputs feedback.

System saves feedback and updates the partner's rating.

Requirements Specification

Actor One: New User

- User Story 1: Profile Creation
    - As a new user, I want to create a detailed profile that allows me to showcase my skills, experience, and project preferences. This way, I can be matched accurately with other programmers.
    - Acceptance Criteria:
        - The profile page must have input fields so that the user could fill information about their skills, years of experience, project types, and even availability.
        - Users should be able to upload a personal profile picture.
        - The system of the application must validate mandatory fields before submission.

- User Story 2: Initial Match View
    - As a new user, it would be nice to view initial match recommendations after completing my profile. This way, I have the ability to see potential collaborators more quickly.

- Acceptance Criteria:

    - The matching algorithm must generate a list of at least 10 potential matches upon profile completion. This algorithm will be fed through profile information that are filled by the user once they complete their profile.
    - Each match preview should display some valuable information about the person, such as primary skills, level of experience, number of projects...etc
    - A user must have the ability to refresh the page that displays potential matches, and get a new list of suggested people/matches.

Actor Two: Returning User

- User Story 3: Connection Milestones

    - As a returning user, I want to see more profile details of my matches as I engage in conversations with them. This way, I can build trust over time.

    - Acceptance Criteria

        - The system should reveal more profile information at specific milestones, such as 10 messages being exchanged between the users.
        - Users must be notified when additional information about their profile has become available to other people they're in touch with.
        - Privacy settings should allow users to control which information, and which details, could become publicly available to others.
- User Story 4: Interactive Prompts for Familiarity

    - As a returning user, I was to receive interactive prompts and conversation starters in order to keep discussions engaging so that I can get to know my matches better.

    - Acceptance Criteria

        - Prompts should be based on interests or mutual skills, and not just random.
        - Users should have the option to dismiss, or ask a prompt to show up later if they were busy at the moment is shows up.
        - The system could potentially log prompt usage by users for continuous learning and future improvements for the application.

Effort Estimation Using Function Points

- User Story 1: Profile Creation
    - Subtasks and Effort Estimation
        - Designing input fields and validation logic: 5
        - Developing image uploading functionality: 7
        - Integration with the database: 25
    - Explanation
        - Creating a profile involves multiple input fields, data validation, and even an image upload feature. Each of these require easy-medium effort levels. Integrating the information with the database could get a little complicated at first, which is why it has higher function points, but as soon as the base setup/integration is functioning properly, then it would be a matter of adding columns to the database later on for any updates.

- User Story 2: Initial Match View
    - Subtasks and Effort Estimation
        - Developing algorithm integration for initial matches: 20
        - Designing a user interface for match previews: 5
        - Implementing a refresh list functionality: 2
    - Explanation
        - This task includes building an algorithm, which could get a little complicated depending on the approach that we'll end up using. More complex algorithms would result in better results. The UI and refreshing functionality should not take much time/effort to implement.

- User Story 3: Connection Milestones
    - Subtasks and Effort Estimation

        - Define and develop milestone logic: 10
        - Implementing a notification system: 2
        - Adjusting user privacy settings: 2

    - Explanation

        - Defining the milestone logic is a bit harder than it looks. The system would have to keep track of whatever data is being passed between the users, and then determine whether more profile details should be revealed or not, and how the user has their privacy settings set up (allowing certain information to be shared while others not to). Notifications and updating privacy settings are not that complicated to implement.

- User Story 4: Interactive Prompts for Familiarity
    - Subtasks and Effort Estimation

        - Developing prompt generation logic: 5
        - Designing a user interface for the prompt display: 3
        - Implementing user action logging: 10

    - Explanation

        - Creating the prompts and displaying them is the easy part. The complex part is how to track user logging , where to store it, and how to utilize it later on.