

Process Deliverables - Agile

What Went Well:

1. **Selection of Match Criteria:** Decided on the optimal set of criteria to ask users at signup for an accurate matching experience and balance comprehensiveness with simplicity.
2. **User Familiarity Features:** Initial designs for staged information reveal and interactive prompts were completed. These features aim to gradually introduce users to their matches, fostering familiarity and trust.
3. **Profile Enhancements:** The addition of bio sections, skill tags, and portfolio links (GitHub, LinkedIn, etc.) received positive preliminary feedback, as they improve searchability and help users present themselves professionally.

Challenges:

1. **Algorithm Complexity:** Balancing multiple factors like skills, availability, and project compatibility remains complex. Fine-tuning the weight and priority of these criteria requires further iteration.
2. **User Onboarding:** Designing an onboarding process that captures comprehensive yet concise information for accurate matching is challenging.
3. **Privacy Considerations:** Developing features that balance user privacy with enough transparency to foster trust between matches is an ongoing challenge.

Action Items Completed:

1. **Selection of Match Criteria:**
 - a. Rank criteria based on their impact on match accuracy (e.g., skills, experience level, preferred project types, availability).
 - b. Gather user feedback on potential signup questions to determine their relevance and importance.
2. **User Familiarity Features:**
 - a. Developed prototypes for staged information reveals, enabling users to control how much information they share at various stages.
 - b. Designed "connection milestones" that unlock new profile details or prompts as users engage with each other.
3. **Profile Enhancements:**
 - a. Users can now link external profiles (GitHub, LinkedIn, portfolio sites).
 - b. Skill tags and an expanded bio section are in place for better user representation and discoverability.

Prioritized Tasks for the Next Sprint:

1. Finalize and Test Matching Algorithm:

- a. Conduct detailed testing to refine algorithm accuracy using feedback from beta testers.
- b. Adjust criteria weights to ensure relevance and improve overall user satisfaction.

2. Enhance Privacy and Trust Features:

- a. Implement interactive prompts that encourage users to share professional goals and interests.
- b. Refine staged information reveals better support user comfort during initial interactions.

3. Prepare for Beta Release:

- a. Conduct user testing on the beta version, focusing on the core matching and familiarity features.
- b. Collect and analyze user feedback to inform final adjustments.

4. User Interface Polish:

- a. Enhance UI/UX elements based on tester feedback, ensuring the swipe-style interface remains intuitive and efficient.

Goal for the Next Sprint:

Release a polished beta version of the programmer-matching app with fully integrated features, including:

- Refined matching algorithm.
- Gradual profile reveals.
- Enhanced user profiles with skill tags, bios, and external links.

This release will provide an opportunity for test users to experience the app's core functionality and offer valuable feedback for final adjustments before the full launch.

High Level Design

The *Client-Server Architecture* is the most appropriate choice for structuring the system, as it efficiently partitions responsibilities between clients and servers. In this model, the server

acts as a centralized authority for processing and managing core functionalities, such as user authentication, matchmaking algorithms, and data storage, while the client side, typically a web or mobile application, provides a user-friendly interface for interacting with the system. This architecture is particularly well-suited for the proposed system, as it relies on seamless communication between clients and servers over a network to enable engineers to search for and connect with compatible collaborators in real time.

The Client-Server Architecture enhances scalability and maintainability by allowing the server to handle multiple concurrent client requests, making it adaptable to increased user demand. Additionally, centralizing the business logic and data on the server simplifies updates and security management, ensuring that sensitive user data, such as profiles and project preferences, are securely managed. Meanwhile, the client side remains lightweight and flexible, allowing for improvements to the user interface without disrupting backend processes. This separation of concerns ensures that both the server and client can evolve independently, supporting iterative development and feature enhancements. By leveraging the strengths of Client-Server Architecture, the system can deliver reliable and efficient matchmaking functionality while maintaining a streamlined and responsive user experience.

Low Level Design

When it comes to implementing the matching algorithm, we believe that the Strategy Pattern is the most suitable approach/choice. This is simply because it allows us to define a family of algorithms, encapsulate each and every one of them, and make them interchangeable. That being said, this choice aligns well with the need for flexibility and experimentation with different matching criteria.

- Justification for using the strategy pattern
 - It enables us to add/modify matching algorithms such as having skill-based matching, or even schedule compatibility. This could be added without making any alterations to the existing codebase.
 - Each matching integration could possibly be implemented as an independent strategy, which makes the code more modular and maintainable.
 - The desired matching strategy can be selected at runtime based on user preferences, or system requirements in certain cases, which enhances adaptability.
- Sample Code Representation
 - The following implementation provides a clear approach when it comes to handling the matching algorithm, and aligning with project's requirements for scalability and flexibility.
 - We can use the structured code as something similar to this:

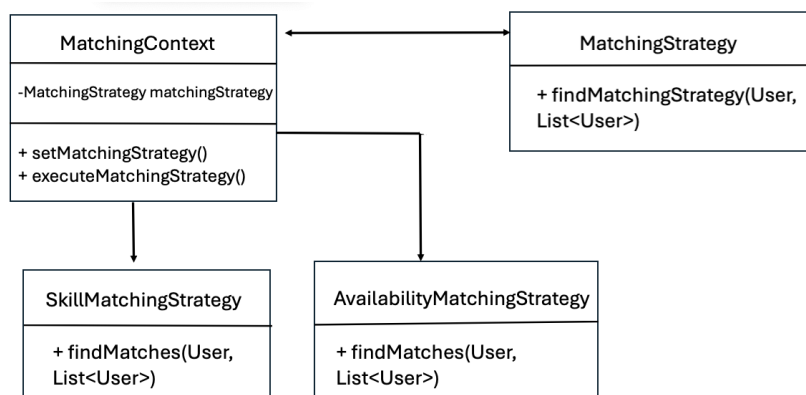
```
34 MatchingContext matchingContext = new MatchingContext();  
35 matchingContext.setMatchingStrategy(new SkillMatchingStrategy());  
36 List<User> skillMatches = matchingContext.executeMatchingStrategy(currentUser, allUsers);
```

```

1 interface MatchingStrategy {
2     List<User> findMatches(User currentUser, List<User> allMatchingUsers);
3 }
4
5 class SkillMatchingStrategy implements MatchingStrategy {
6     public List<User> findMatches(User currentUser, List<User> allMatchingUsers)
7     {
8         return allMatchingUsers.filter(user -> user.skills.containsAll
9             (currentUser.skills));
10    }
11 }
12
13 class AvailabilityMatchingStrategy implements MatchingStrategy {
14     public List<User> findMatches(User currentUser, List<User> allMatchingUsers)
15     {
16         return allUsers.filter(user -> user.availability.overlaps(currentUser
17             .availability));
18    }
19 }
20
21 class MatchingContext {
22     private MatchingStrategy matchingStrategy;
23
24     public void setMatchingStrategy(MatchingStrategy strategy) {
25         matchingStrategy = strategy;
26     }
27
28     public List<User> executeMatchingStrategy(User currentUser, List<User>
29         allMatchingUsers) {
30         return matchingStrategy.findMatches(currentUser, allMatchingUsers);
31     }
32 }

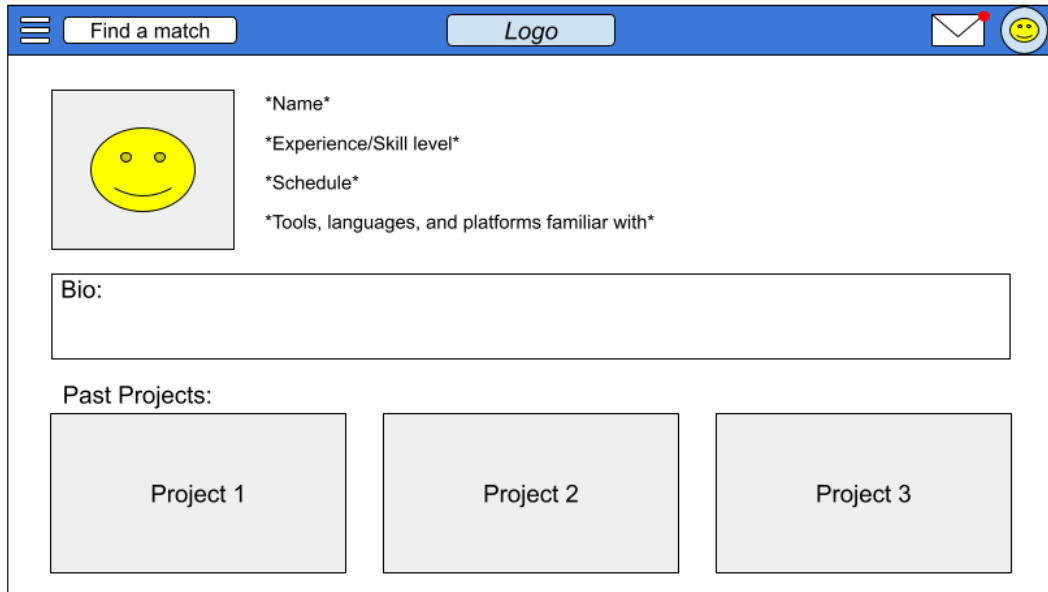
```

- UML Design



Design Sketch

Wireframe of profile view and editing page



The image shows a user profile form within a web browser window. The browser's address bar contains the text "Find a match" and "Logo". The profile form itself has a blue header bar with a hamburger menu icon, a "Find a match" button, a "Logo" button, an envelope icon, and a smiley face icon. The main content area of the form includes a square placeholder for a profile picture (containing a yellow smiley face), followed by four text input fields labeled with asterisks: "*Name*", "*Experience/Skill level*", "*Schedule*", and "*Tools, languages, and platforms familiar with*". Below these is a text area labeled "Bio:". At the bottom, a section titled "Past Projects:" contains three rectangular boxes labeled "Project 1", "Project 2", and "Project 3".

We wanted our site to be relatively simple and easy to use and navigate, so the profile only has information that would be relevant to a potential partner match, such as their name, profile picture, experience and skill level, schedule, known tools and languages, a brief bio, and a showcase of their past projects. We especially wanted to prioritize seeing past projects, as this is a good indicator and proof of the person's skill level and experience, so the past projects section is made large and easy to see. These choices help with the minimalistic design heuristic, allowing users to easily see what is most important. We also wanted it to be relatively easy to navigate to the profile page, inbox, home page, and match finding page, so there are buttons for these functions located in the header bar. This is similar to many other social media platforms, so users will likely be familiar with the layout. This helps with the heuristic of consistency and standards.