Requirements

CS 3704 : Intermediate Software Design

Prepared by
Caroline Joseph
Mason Gelletly
Quinn Anderson
Tatiana Monteiro

Epic Fellows

09/13/ 2023

Requirements Workshop:

1. Provide an example of five hypothetical non-functional requirements for this system. Be sure to include the specific type of requirement discussed in class, with each requirement coming from a unique category.

1. *Usability:* Text on the UI must be visible from 1 meter, with a consistent style and color scheme throughout the screens.
2. *Reliability:* The matching algorithm should be extremely reliable, with a failure to match due to error instead of lack of matches rate of 0.01%.
3. *Performance:* The server should not take no more than 2 seconds to respond to the user in some capacity when being directly interacted with.
4. *Supportability:* The codebase should be thoroughly documented, well organized, and by no means hard-coded to promote scalability and future updates.
5. *Implementation/Constraints:* The system should be capable of running on any Windows or UNIX based operating system.

2. Provide an example of five hypothetical functional requirements for this system.

1. The application must allow users to create unique profiles with general information that will aid in the matching algorithm.
2. The application must have a notification system that will promptly and accurately notify the user in the case of potential relevant matches.
3. The application must have a matching algorithm that leverages the information in the submitted user form, the type of form that was submitted, as well as the user profile to create an optimal pairing.
4. The application must maintain several different form templates that the user can choose from, each denoting a different reason for matching and each profoundly affecting the matching algorithm.
5. The application must implement a reporting system to flag users that misuse or otherwise abuse the application.

3. Think of a specific task required to complete each of the functional requirements and non-functional requirements mentioned above (10 total). Estimate the amount of effort needed to complete this task using function points (i.e., using the values here). Briefly explain your answer.

Functional Requirements

1. Developing a user registration and profile module (20 points)
   Creating a user profile involves setting up a user database, UI for registration, and setting up security
2. Implementing the notification engine (5 points)
   Involves backend processing, email/SMS services, and handling real-time notifications
3. Design and develop matching algorithm(40 points)
   The main tool for our system, this requires logic, handling of different criterias, and machine learning
4. Creating form templates (2 points)
   Requires UI design for each backend processing for, and integration with the matching algorithm
5. Setting up user reporting and flagging system (8 points)
   Requires UI for reporting, backend process for flags, and maybe action on users

Non-functional requirements

1. Usability: Design a consistent UI/Ux theme and implement visibility checks)(5 points)
   Ensuring visibility and consistency is a basic aspect of UI/UX design
2. Run extensive testing (8 points)
   We need to test rigorously so we can ensure a 0.01% failure rate.
3. Perform server optimization processing and response times (13 points)
   Performing performance enhancements to optimize the server.
4. Keep a well kep document for entire codebase and ensure proper organization (13)
   This is time consuming and may require restructuring, refactoring, and organizing code.
5. Ensure cross-platform compatibility (13 points)
   Might involve setting up virtual environments for Windows and UNIX and testing on both platforms.

4. Write three user stories from the perspective of at least two different actors. Provide the acceptance criteria for these stories.

1. As a beginner programmer, I want to match with a more experienced programmer, so that I can improve my coding skills.

Acceptance criteria:
Given beginner programmer has pair-programming matcher tool
When beginner programmer submits a profile
And there is a match in the pool
Then tool offers the user possible pairings

2. As a programmer on a project, I want to match with a similar programmer, so that we can work together cohesively on the same project.
   Acceptance criteria:
   Given both programmers has pair-programming matcher tool
   When programmer 1 submits a profile
   And programmer 2 submits a similar profile
   Then tool will alert both of them that there is a possible match
3. As a front-end programmer working on a full-stack project, I want to match with a back-end programmer, so that we can work collaboratively to develop this project.
   Acceptance criteria:
   Given both programmers has matcher tool
   When both programmers submit a profile
   Then tool will alert both of them that a match has been made

5. Provide two examples of risk that could potentially impact this project. Explain how you would mitigate these risks if you were implementing your project as a software system.

1. One risk would be if the database with the programmer's preferences was not secure or if it was susceptible to an outside attack. We would mitigate this risk by implementing a more secure database with encryptions in order to protect sensitive data.
2. Another risk would be if the system were to crash suddenly and all of the data was lost. We would mitigate this by storing the preferences and user data in a backup in case something were to happen to the main system.

6. Describe which process your team would use for requirements elicitation from clients or customers, and explain why.

We would use the individual interview process because we can go more in-depth on what the client or customer wants, as well as ask follow-up questions related specifically to that individual or who they represent. Furthermore, we can prevent a group herd mentality and be able to figure out what each individual truly thinks. Although we will get a large amount of data, we can then sort and figure out what is important and what we can discard.

Requirements Analysis:

Use Case 1

New programmer wants to match with a more experienced programmer for mentorship.

Precondition:

User must create a profile that includes their preferences for availability, expertise, and more.

Main Flow:

User will request a match [S1]. The matching tool will generate complimentative possible matches for the user to pick from based on their profile [S2].

Subflows:

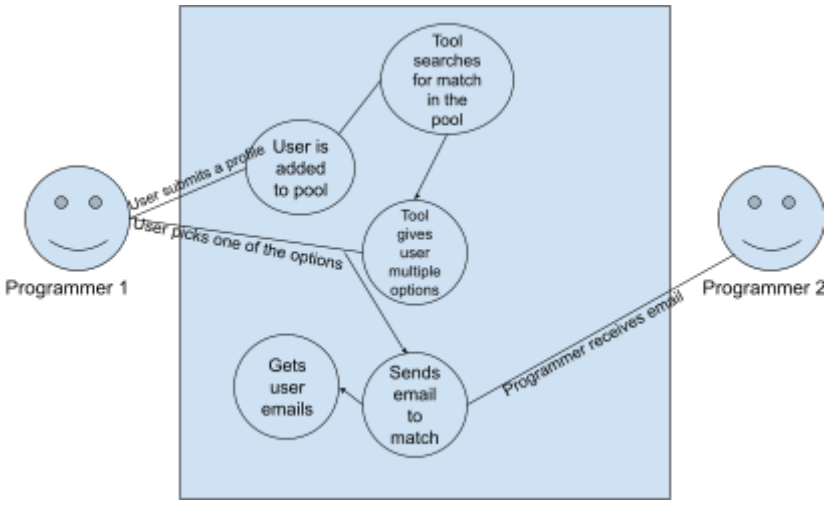
[S1] User requests a pairing.

[S2] Matching tool will look through current options and give the user possible pairings that mostly match their preferences.

[S3] User can select one from the proposed pairings and can directly contact them to start pair programming.

Alternative Flows:

[E1] There are no possible pairings, in the case there are no good pairings or no one in the matching pool.



Use Case 2:

Programmer wants to join the pool to assist coworkers and does not need a match right now.

Precondition:

User has preferences in general, like availability, expertise, language, etc.

Main Flow:

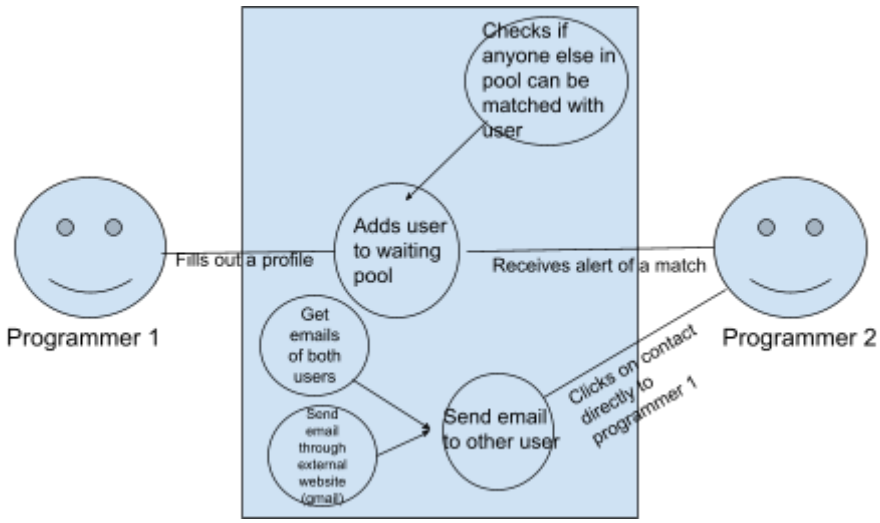
User will open an application and select a button to make a profile [S1]. Once the form is submitted, the tool will update the user's preferences in the system [S2]. The user will be in the pool until another programmer who fits the requirements requests a match through the matching tool [S3]. If a match is made, the user is alerted and can directly contact the match to begin pair programming [S4].

Subflows:

[S1] User begins to make a profile.
[S2] Tool updates user's preferences in the system.
[S3] User waits in the pool until a match is made.
[S4] If a match is made, the user is alerted and can contact the other programmer.

Alternative Flows:

[E1] No possible pairings can be made
[E2] User provides unallowed preferences, like no availability or like red as the coding language



Use Case 3:

Engineer is hard-blocked, needs urgent help to be productive.

Precondition:

Users have an account with their basic information: work expertise, language, etc...

Main Flow:

Users will open the application and request a match, they will fill out the form they are presented with, complete with information on their blocker [S1]. The user will be in the pool until another programmer is available [S2]. If a match is made, the user is alerted and can directly contact the match to start pair programming [S3].

Subflows:

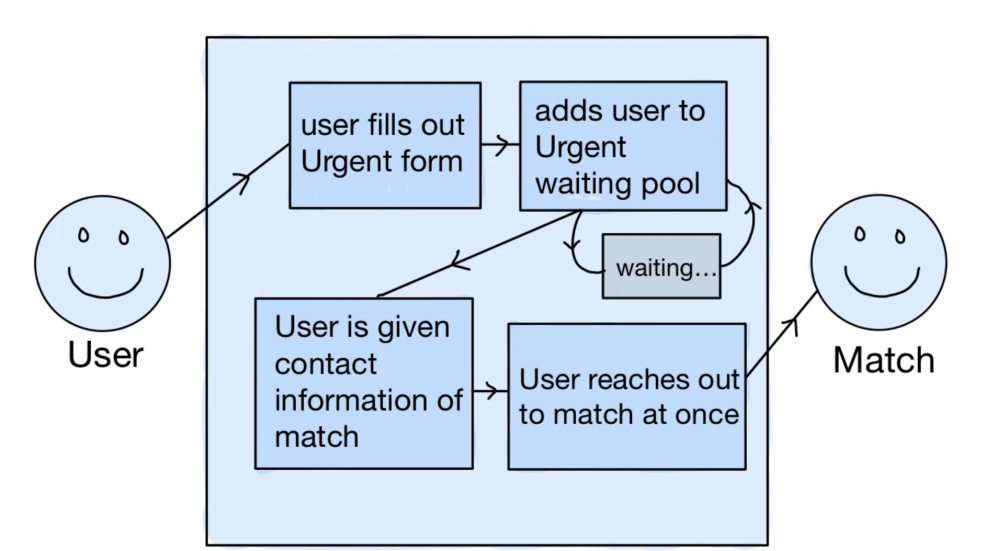
[S1] Within the form, users will input information about their blocker, as well as marking the match-need as ‘Urgent’.

[S2] The matching process will be far less restrictive since the form is marked as ‘Urgent’. Instead of vetting for a strictly perfect match, any coworker who is willing to help and has minimal experience is considered.

[S3] If a match is made, the user is shown the profile of the potential match and is given contact details such that they can set up a meeting. Due to the urgency, this meeting should take place very soon after matching.

Alternative Flows:

[E1] No possible pairings can be made. In this case, the user will sit in the pool until they decide to exit.



Use Case 4:

Engineer has an idea that they would like to see implemented, seeking a partner to work with and bounce ideas off of.

Precondition:

User has an account with their basic information: work expertise, language, etc...

Main Flow:

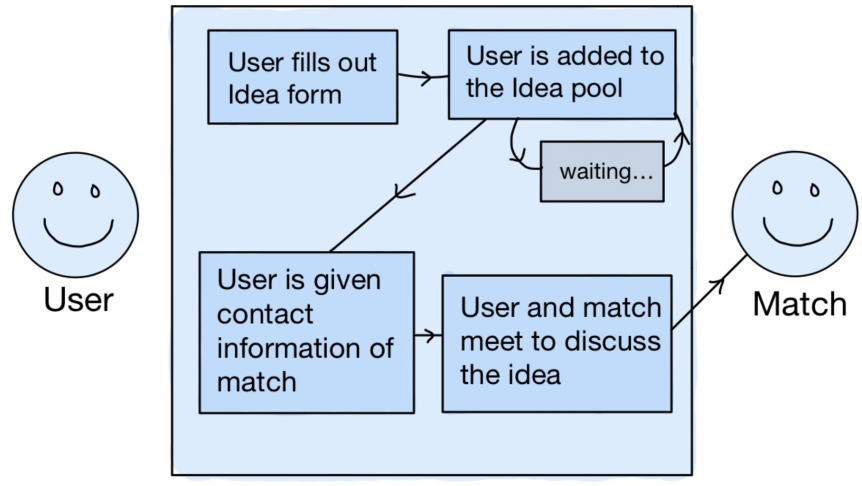
User will open the application and request a match, they will fill out the form they are presented with, complete with a flag denoting they have a project in mind and the basics of said project [S1]. The user will be in the pool until another programmer is available [S2]. If a match is made, the user is alerted and can directly contact the match to set up a call and begin to discuss the potential project [S3].

Subflows:

[S1] Within the form, user will input base level information about their idea/project/thing they would like to implement. The system will mark this match query as a potential project, which will affect the matching algorithm appropriately.
[S2] The matching process will be far less prompt, and is meant in this case to likely span over the course of days/weeks. This sort of matching is more akin to an open job posting, where a match would be truly interested in getting involved with the user.
[S3] If a match is made, the user is shown the profile of the potential match and is given contact details such that they can set up a meeting. This will likely be a general call, but if it goes well could blossom into a functioning team working towards the idea's implementation.

Alternative Flows:

[E1] No possible pairings can be made. In this case, the user will sit in the pool until they decide to exit. In this case, deciding to exit could be after many months or even years.



Use Case 5:

Diversity Match (promote diversity with a match from a differing team/demographic; to provide more diverse ideas)

Precondition:

User has an account with their basic information: work expertise, language, etc..

Main Flow:

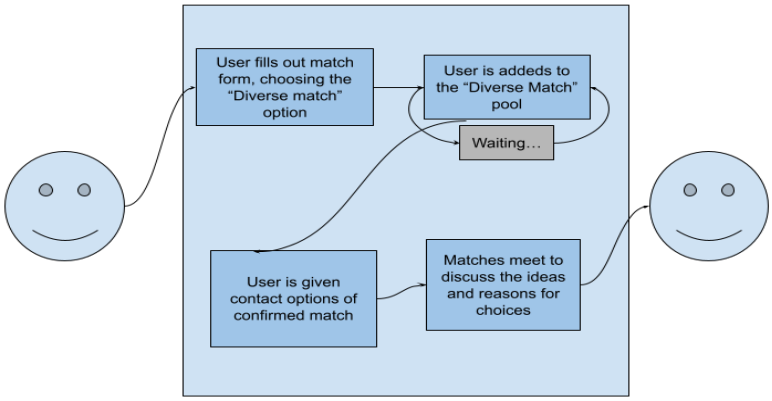
[S1] User logs into the applications and requests a match checking the “Diversity Match” options
[S2] System checks for potential matches from different demographics/team.
[S3] System suggests matches found
[S4] User reviews suggestion and confirms matches
[S5] Notification is sent to both users
[S6] If a match is made, the user is shown the profile of the potential match and is given contact details, such that they can set up a meeting. The user can also choose to contact them via the app.

Subflows: After matching

[S1] Users decide to consult with another diversity match for a broader perspective. After finishing users are asked to rate the matching and provide feedback about matches, meetings, etc. User the requests additional matches, and the system follows the main flow to provide matches.
[S2] User decides to withdraw from the initial confirmed meeting. The withdrawing used updates match status, and the system sends notification to other users and suggests rescheduling or finding a new match.

Alternative Flows: No match found:

[E1] System notifies the user that no diverse match is currently available
[E2] Users can opt for regular match or wait for diverse until a match is found



Process Deliverable:

Waterfall: submit supplementary planning documentation

We are using the waterfall method which understands that all of our requirements can be understood at the start, and then we have the while we work on our project. Our process deliverable is then based on some of these requirements we have come up with along with the planning we have done for it. We are focusing specifically on the cost, assumptions, risks, and the success metrics.

Overall, our cost is very dynamic, as starting off we do not really need any investments or large sums of money. Our application will be a pairer, and we can use minimal cost to get the algorithm working. We most likely will use machine learning models and there are lots of ones that can be used or trained for free or at a small cost. As our team grows we would have to hire more people which would be more costly. Furthermore, when we want our program to work across the nation we would need to put money into it to advertise and actually get companies to use it. The good thing is that we can do a lot at a low cost for the time being until we expand.

The next part of the plan deals with the assumptions we make of our stakeholders. Our biggest assumption is the companies have a need for pairing. We are basing this assumption off of our time in school, as we have noticed that there are cases of imposter syndrome and bad matches in group work. We are thinking that companies also have this need and problem, and that our project will address this.

We must also address the risks in our plans. There is a risk linked with the assumptions that either companies and stakeholders do not have a need for this. We also acknowledge the fact that by the time we complete our project similar products may exist or may be developed. It is important that we do our work with stakeholders and really make sure there is a need for our exact solution.

We also must measure our success by success metrics. First we need to make sure we are on a solid and good plan, along with getting our research done with stakeholders done fast. We must also research what products we will use in our project first as well. Doing so will allow us to estimate our costs, revenue, and profits, while making sure our product is on track to succeed. We also must measure the marginal utility of each of our employees, and as our product grows, hire more people to work on, and divide up the tasks into teams.

Overall, these four main key areas highlight our supplementary planning documentation for our product.