

CS 3354 Software Engineering
Final Project Deliverable 2

100%

Susana Lainez, Amy Abraham, Nguyen Le, Orlando Garcia, Soumaya Hajji, Daniel Arrant,
James Ostlind

Delegation of Tasks:

Soumaya Hajji: Created the GitHub Repository, added collaborators and created project_proposal and uploaded files, created a Use Case diagram. Wrote on similar designs paragraph and worked on powerpoint slides.

Nguyen Le: Helped with the project_scope and created the sequence diagrams. Used MS Project to create the Task Table, Activity Bar Diagram, Resource Table, and Project Timeline.

Amy Abraham: Created the README commit, wrote about the software process model and architectural model we plan to use, created class diagram, Function point, created slides, created user interface design.

Susana Lainez: Created project_scope and committed the changes to the repository. Created Class diagram, implemented feedback in project proposal. Worked on project scheduling, specifically creating the task table and resource table. As well, doing the cost modeling technique (function point) and worked on the powerpoint.

Orlando Garcia: Created functional and non-functional requirements under software requirements section. Calculated estimated personnel costs, software costs, and hardware costs.

Daniel Arrant: Worked on MVC architecture diagram and partly on non-functional requirements. Worked on JUnit testing.

James Ostlind: Developed View User Page and Message User sequence diagrams.

Deliverable 1 Content

~~~~~

### **Feedback Provided:** (from Proposal)

- Dear All,

Thank you so much for the original ideas you proposed. Essentially, each team can go ahead and start working on their proposed project. In the meantime, please make sure that you include the following in your **final project deliverable 2 (final deliverable)**:

- A comprehensive research to find similar project implementations. Cite your findings properly using IEEE citation format.
- Make sure that you are adding extra feature(s) to uniquely differentiate your design from already existing similar implementations. Clearly explain what these feature(s) are.
- A comparison of your design with similar implementations in the field. This could be in any format of your choice, such as a table, paragraphs, charts, etc.

Best of luck with your projects and hope everyone enjoys working on them.

- To comply with the feedback, we researched other dating apps and explained how our app is different from those out in the market.

---Proposal---

- **The title of your project:** 100%
- the group members (firstname and lastname)
  - Susana Lainez
  - Amy Abraham
  - Nguyen Le
  - Orlando Garcia
  - Soumaya Hajji
  - Daniel Arrant
  - James Ostlind
- **What you will be doing:**
  - Creating a dating application that pairs users using common interests. Based on how many common interests a person has with another individual, a person is assigned a compatibility ratio from 0% to 100%. Questions involving common books, movies, tv shows, music, hobbies, etc. will be asked when the user starts a profile. As users start swiping through potential matches, they will see how compatible (percentage-wise) they are with other individuals on the application. Along with that filters such as distance, religion, occupation, etc. will also be available for users.
- **A detailed description of your motivation (why you chose to do this particular project), where you expect your design to be used in real life.**
  - Individuals are often looking for new people to meet and interact with. However, with how busy they are with classes and work, it can become hard to get out there and introduce yourself to new individuals, especially for those that tend to be more introverted. As a result, we have decided to create a dating app that allows

individuals to interact with other people that possess similar interests in categories such as movies, TV shows, books, and music.

Based on these criteria, the 100% algorithm will provide the user with a number between 1 and 100 based on how compatible two individuals are. This is different than current dating apps on the market for it provides the user with a rating on how compatible they are with others, something that is not included in applications like Tinder and Bumble. However, similar to other dating apps, 100% displays the pictures of other users, as well as some of their information like name and age, as well as the incorporation of direct message so that users can interact with one another. In addition, current dating apps like tinder also have similar features where you can swipe one way or the other depending on whether you like a person or not, allowing for more user interaction and feedback [2].

**- The list of tasks delegated to each member**

**Team leader:** Ensure that each assignment is submitted on time, help team coordinator(s) arrange meetings and remind team of special due dates (when they are not available).

- Member(s): Amy Abraham

**Team Coordinator:** Arrange meetings and remind team of special due dates.

- Member(s): Susana Lainez, Nguyen Le

**Lead Designer:** Ensure that the diagrams are drawn out nicely and clearly with sufficient details.

- Member(s): Daniel Arrant

**Lead Programmer:** Make sure that the code is well implemented.

- Member(s): James Ostlind

**Team Member:** Help with project and update any changes that were made with the team on either Scrum or any other framework.

- Member(s): Soumaya Hajji, Orlando Garcia

---End of Proposal---

**Setting Up a GitHub Repository:**

<https://github.com/souhajji/3354-100percent>

### **Delegation of Tasks:**

Soumaya Hajji: Created the GitHub Repository, added collaborators and created project\_proposal and uploaded files, created a Use Case diagram.

Nguyen Le: Helped with the project\_scope and created the sequence diagrams.

Amy Abraham: Created the README commit, wrote about the software process model and architectural model we plan to use, created class diagram.

Susana Lainez: Created project\_scope and committed the changes to the repository. Created Class diagram, implemented feedback in project proposal.

Orlando Garcia: Created functional and non-functional requirements under software requirements section.

Daniel Arrant: Worked on MVC architecture diagram, non-functional requirements.

James Ostlind: Developed View User Page and Message User sequence diagrams.

### **Software Process Model:**

Spiral Model is the model we will be using in the development of our dating application. The Spiral model combines the iterative nature of prototyping and the systematic aspect of the waterfall model. This means that we will be able to continuously make changes as the software evolves. Changes according to customer preferences and new application updates will be able to be performed if the Spiral model is used as the Software process model for this project.

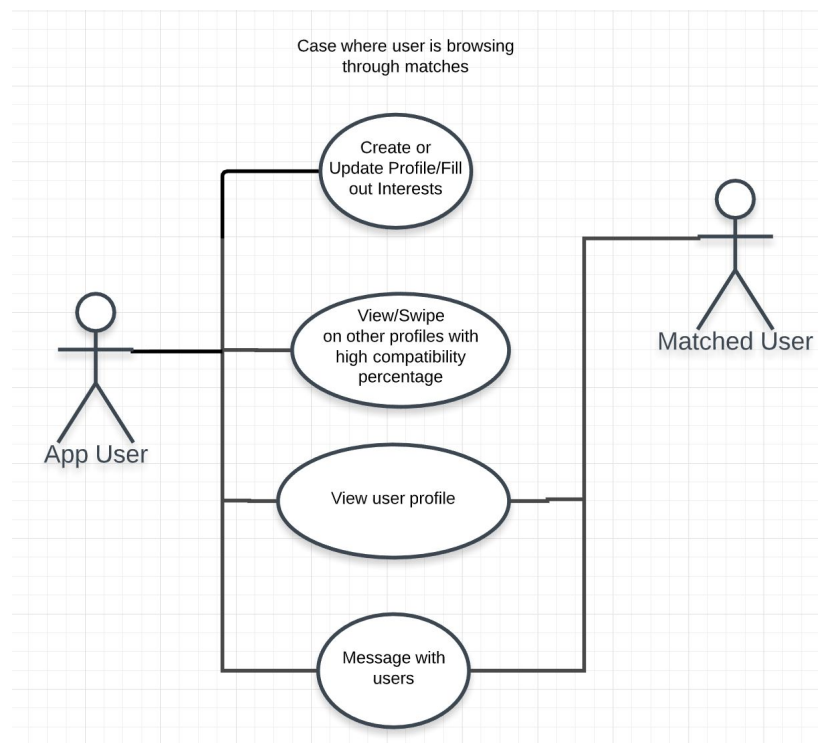
### **Software Requirements:**

- **Functional Requirements**
  - Matches users based on mutual interests.
  - It allows users to swipe through people with similar interests.
  - Gives a 1-100 compatibility percentage between two users.
  - Will allow interaction between matched users through messaging.
  - Users will swipe up if they're interested or down if they're not interested.
- **Non-Functional Requirements**
  - **Product Requirements**
    - **Efficiency Requirements**
      - **Performance Requirements**

- App must load matches within 10 seconds.
  - App must evaluate compatibility percentage in 3 seconds or less.
- **Space Requirements**
  - App shall not exceed 200MB of space on a user's phone.
- **Dependability Requirements**
  - App will always allow communication between two users.
  - Any downtime occurrences should be addressed and fixed within 45 minutes.
  - App will keep user information secure by providing password-protected accounts.
- **Usability Requirements**
  - App will provide users with accounts to showcase interests, pictures, and a description of themselves.
- **Organizational Requirements**
  - **Environmental Requirements**
    - App will run Apple's iOS and Google's Android mobile operating systems.
    - App will be available for download through Apple's App Store and Google's Play Store applications.
  - **Operational Requirements**
    - Users will choose a username and password to access their accounts.
    - User will be able to add data to their profile in the form of pictures, biographies, and interests.
    - App will be able to access files on the users device.
    - Users will be able to report malicious users on the platform.
    - Users will be able to view matched user accounts.
    - Users will be able to send messages to matched users through a messaging interface on the application.
  - **Development Requirements**
    - Android app will be built and maintained using Android Studio IDE, iOS app will be built and maintained using Xcode IDE.
    - Android app will be implemented using Java.
    - iOS app will be implemented using Swift.
- **External Requirements**
  - **Regulatory Requirements**
    - No users below the age of 18.
    - Follows Apple's app store guidelines.

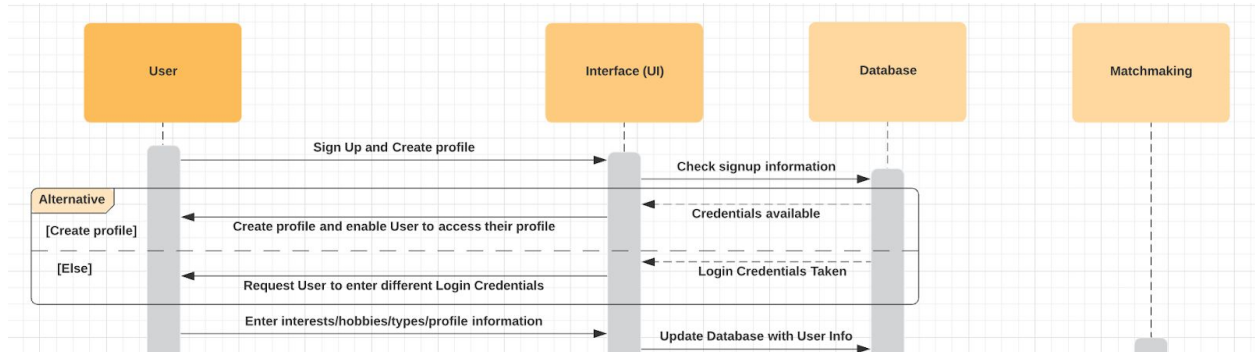
- Follows Google's developer policies.
- **Ethical Requirements**
  - Must discourage the exchange of inappropriate messages between users on the app.
- **Legislative Requirements**
  - **Accounting Requirements**
    - App must be free to download through Apple's App Store and Google's Play Store.
  - **Safety/Security Requirements**
    - App will filter out fake accounts to prevent cybercrime (online scams, phishing, etc).

**Use Case Diagram:**

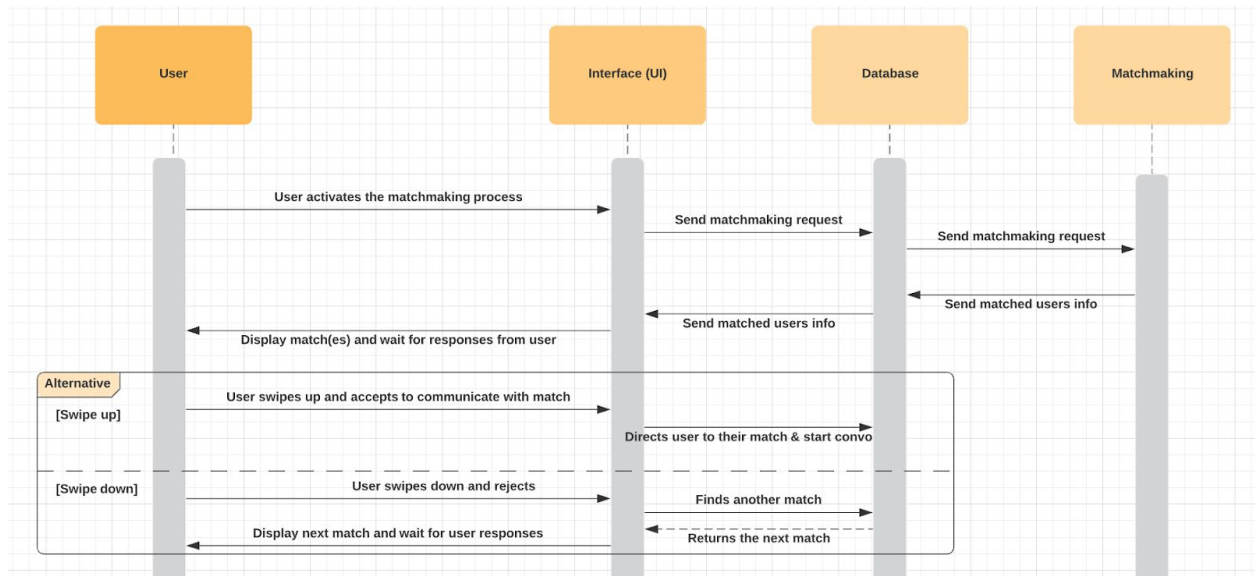


## Sequence Diagram:

### Create profile/Fill out interests

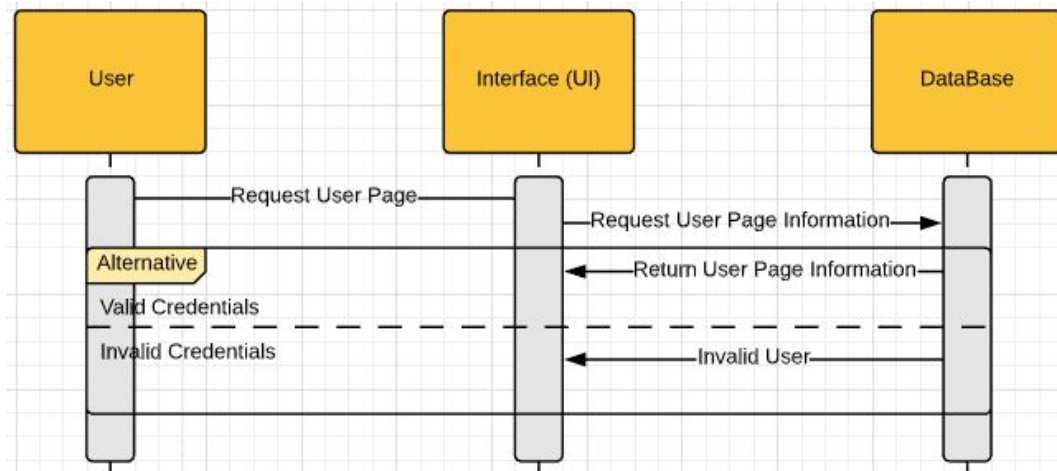


### Matchmaking

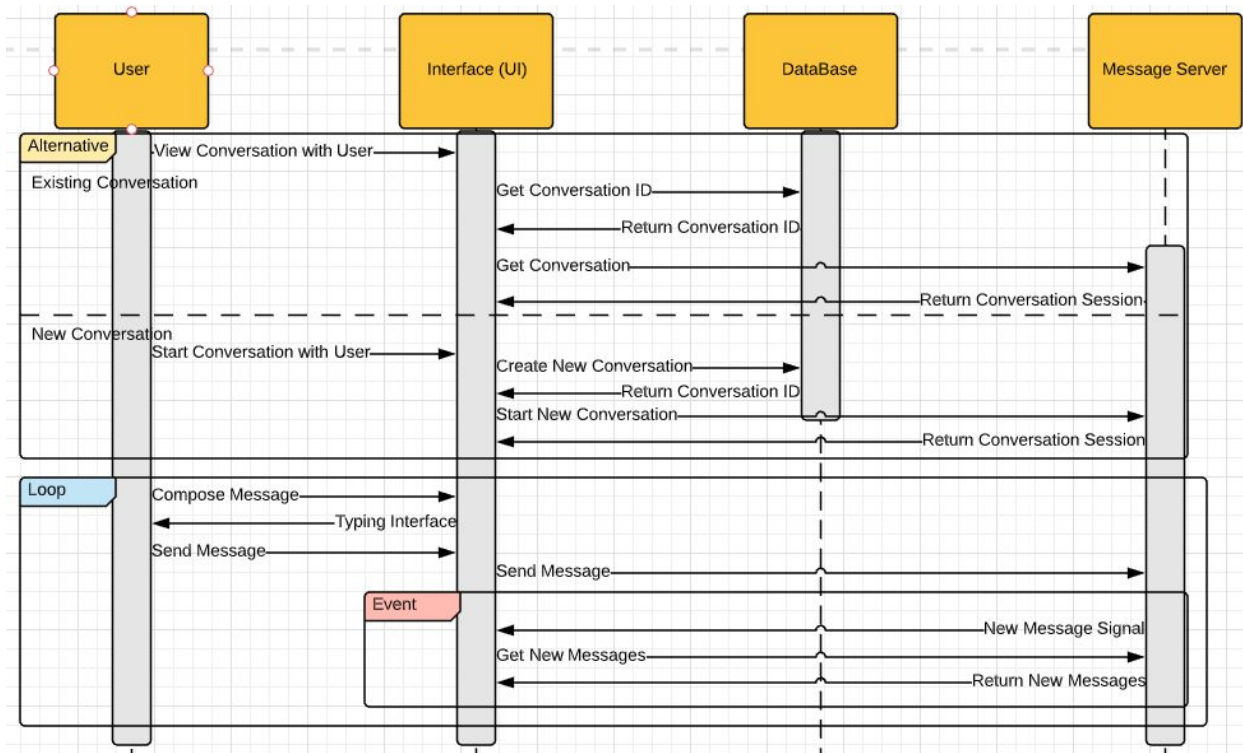




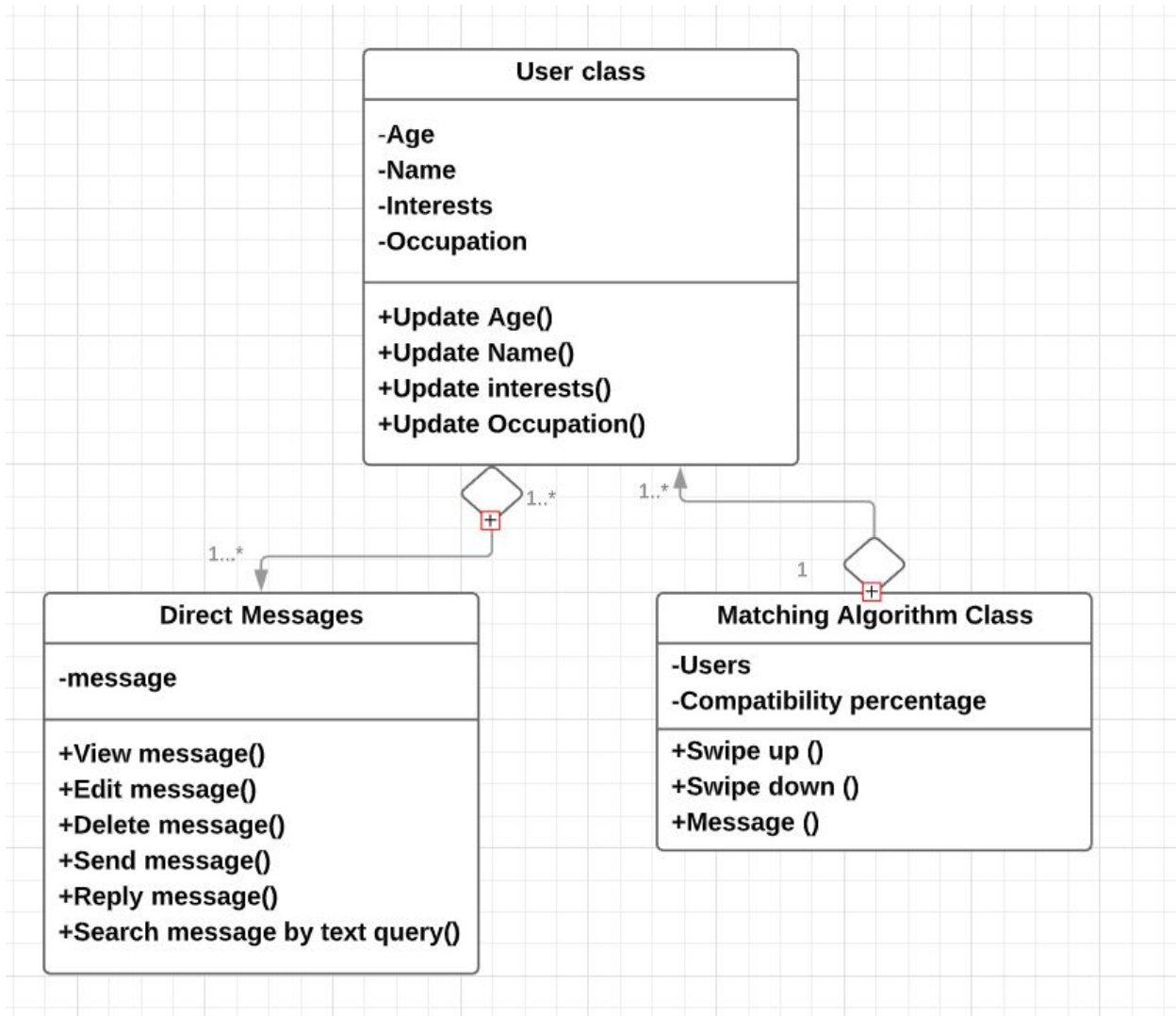
## View user profile:



## Message with users:

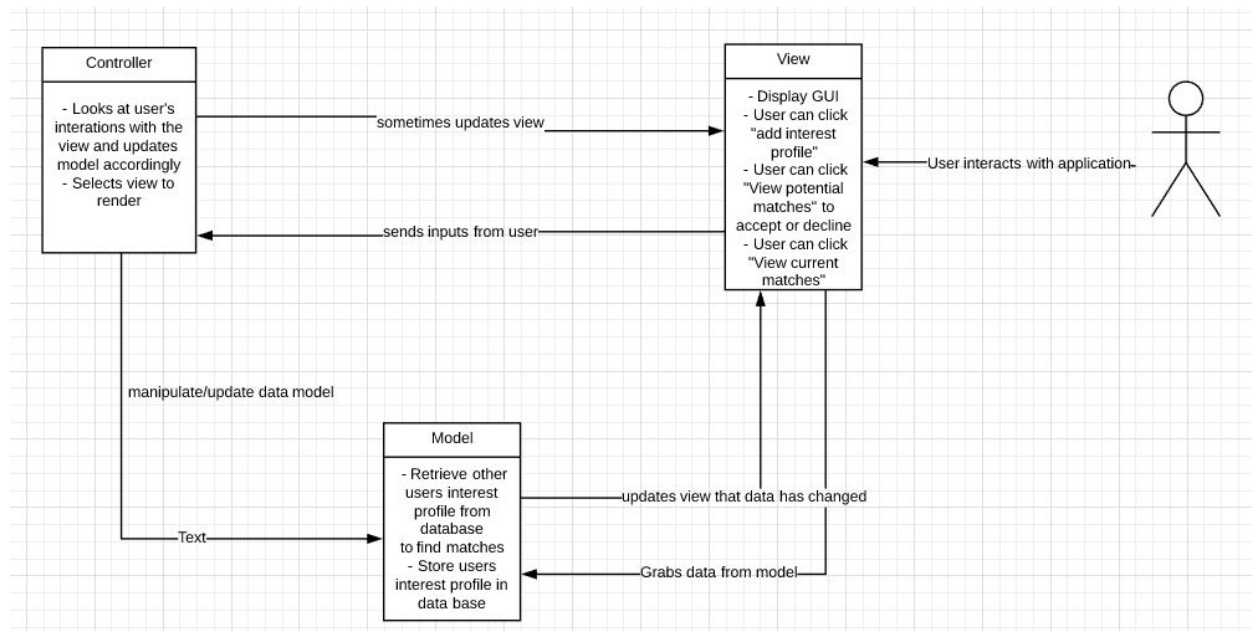


## Class Diagram:



## Architectural Design: MVC Pattern

The architectural pattern we will use is MVC because changes to system data, presentation of the data, and how the user interacts with the data all correlate and work together for a software application. For our dating app, user interaction is extremely important and MVC takes into account user interaction and adjusts data and the presentation of the data on the app. It is a simple pattern that is seen used in many software that allows for changes and easily shows the interactions between components in software.



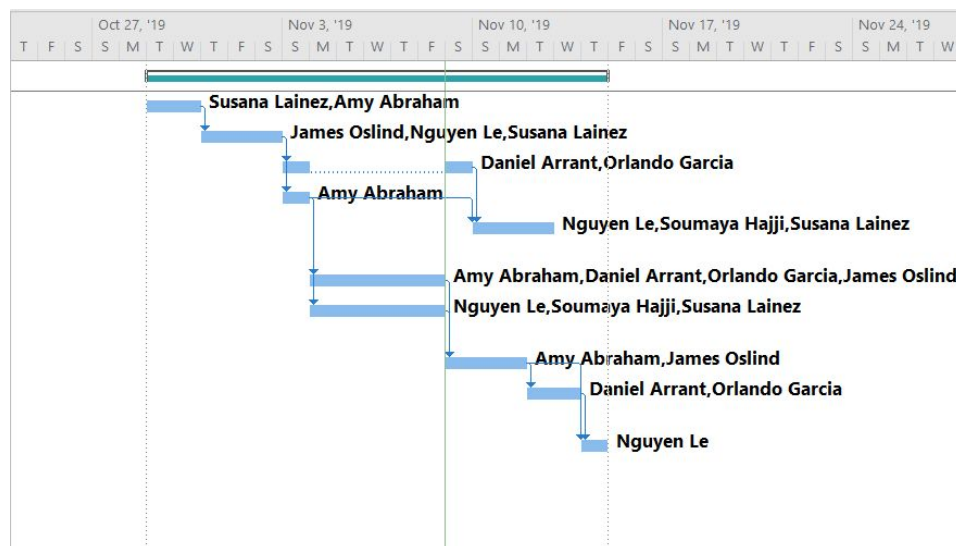
## Project Scheduling and Estimation Studies

**3.1 [20 POINTS] Project Scheduling.** Please note that what you present should be the timeline of the project designed, NOT the time you've spent on it. Use an automated tool (such as MS Project) to plan a schedule for your project. It should include **tasks, durations, and dependencies** for your project provided on a table (similar to **Figure 23.5**), as well as an **activity bar chart** (similar to **Figure 23.6**) drawn using an automated tool (such as MS Project).


### Task Table:

|    |  | Task Mode | Task ID | Explanation                 | Duration (days) | Start        | Finish       | Predecessors | Resource Names                            |
|----|--|-----------|---------|-----------------------------|-----------------|--------------|--------------|--------------|-------------------------------------------|
| 1  |  |           |         | GroupAll                    | 17 days         | Tue 10/29/19 | Thu 11/14/19 |              |                                           |
| 2  |  |           | A       | Requirements gathering      | 2 days          | Tue 10/29/19 | Wed 10/30/19 |              | Susana Lainez,Amy Abraham                 |
| 3  |  |           | B       | Analysis                    | 3 days          | Thu 10/31/19 | Sat 11/2/19  | 2            | James Oslind,Nguyen Le,Susana Lainez      |
| 4  |  |           | C       | Feasibility study           | 2 days          | Sun 11/3/19  | Sat 11/9/19  | 3            | Daniel Arrant,Orlando Garcia              |
| 5  |  |           | D       | Analyze Alternative plans   | 1 day           | Sun 11/3/19  | Sun 11/3/19  | 3            | Amy Abraham                               |
| 6  |  |           | E       | Draft Design Specifications | 3 days          | Sun 11/10/19 | Tue 11/12/19 | 5,4          | Nguyen Le,Soumaya Hajji,Susana Lainez     |
| 7  |  |           | F       | Implement Backend           | 5 days          | Mon 11/4/19  | Fri 11/8/19  | 5            | Amy Abraham,Daniel Arrant,Orlando Garcia, |
| 8  |  |           | G       | Implement User Interface    | 5 days          | Mon 11/4/19  | Fri 11/8/19  | 5            | Nguyen Le,Soumaya Hajji,Susana Lainez     |
| 9  |  |           | H       | Test Overall System         | 3 days          | Sat 11/9/19  | Mon 11/11/19 | 7,8          | Amy Abraham,James Oslind                  |
| 10 |  |           | I       | Make Necessary Changes      | 2 days          | Tue 11/12/19 | Wed 11/13/19 | 9            | Daniel Arrant,Orlando Garcia              |
| 11 |  |           | J       | Launch Application          | 1 day           | Thu 11/14/19 | Thu 11/14/19 | 9,10         | Nguyen Le                                 |

### Activity Bar Chart:



### (Optional) Resource Table

|   |  | Resource Name ▾ | Type ▾ | Initials ▾ | Group ▾ | Max. ▾ | Std. Rate ▾ |  |
|---|-----------------------------------------------------------------------------------|-----------------|--------|------------|---------|--------|-------------|--|
| 1 |                                                                                   | Susana Lainez   | Work   | SL         |         | 100%   | \$0.00/hr   |  |
| 2 |                                                                                   | Amy Abraham     | Work   | AA         |         | 100%   | \$0.00/hr   |  |
| 3 |                                                                                   | Nguyen Le       | Work   | NL         |         | 100%   | \$0.00/hr   |  |
| 4 |                                                                                   | Orlando Garcia  | Work   | OG         |         | 100%   | \$0.00/hr   |  |
| 5 |                                                                                   | Soumaya Hajji   | Work   | SH         |         | 100%   | \$0.00/hr   |  |
| 6 |                                                                                   | Daniel Arrant   | Work   | DA         |         | 100%   | \$0.00/hr   |  |
| 7 |                                                                                   | James Oslind    | Work   | JO         |         | 100%   | \$0.00/hr   |  |
|   |                                                                                   |                 |        |            |         |        |             |  |

**3.2 [15 POINTS] Cost, Effort and Pricing Estimation. Describe in detail which method you use to calculate the estimated cost and in turn the price for your project. Some cost modeling techniques you may use are listed as follows:**

- 1. Function Point Or any of the following COCOMO II estimation models**
- 2. Application composition**
- 3. Early design**
- 4. Post-architecture**

Based on our dating app 100%, we have estimated to have the following function category counts:

Number of user input: 10

Number of user output: 20

Number of user queries: 8

Number of data files and relational tables: 25

Number of external interfaces: 3

The complexity for each of these is simple.

In addition, procession value for the 14 questions are all average except for online data entry and internal processing which is significant, and user friendliness which is essential ( PC6 = 4, PC10 = 4, PC14 = 5). In addition, we estimate that the productivity of our team will be approximately 40 function points per person-week.

|     | Function category                          | Count | Complexity |         |         | Count X Complexity |
|-----|--------------------------------------------|-------|------------|---------|---------|--------------------|
|     |                                            |       | Simple     | Average | Complex |                    |
| 1   | Number of user input                       | 10    | 3          | 4       | 6       | 30                 |
| 2   | Number of user output                      | 20    | 4          | 5       | 7       | 80                 |
| 3   | Number of user queries                     | 8     | 3          | 4       | 6       | 24                 |
| 4   | Number of data files and relational tables | 25    | 7          | 10      | 15      | 175                |
| 5   | Number of external interface               | 3     | 5          | 7       | 10      | 15                 |
| GPF |                                            |       |            |         |         | 324                |

$$GFP = 10 \times 3 + 20 \times 4 + 8 \times 3 + 25 \times 7 + 3 \times 5 = 324$$

$$PCA = 0.65 + .01 \times (11 \times 3 + 2 \times 4 + 1 \times 5) = 0.65 + .01 \times 46 = 1.11$$

$$FP = GFP \times PCA = 324 \times 1.11 = 359.64 \text{ FP}$$

Therefore the estimated effort is obtained as:

$$E = FP / \text{productivity} = 359.64 / 40 = 8.991 \sim 9 \text{ person-weeks}$$

Team size is 7, then the project duration

$$D = E / \text{team size} = 9 / 7 = 1.29 \sim 2 \text{ weeks}$$

### **3.3 [5 POINTS] Estimated cost of hardware products (such as servers, etc.)**

Backend and Servers: AWS Cloud Services (\$300 - \$700 per month)

Development: 4 Apple MacBook Pros (\$12,100)

**Estimated First Year Cost: \$15700 - \$20500**

**Estimated Cost After First Year: \$3600 - \$8400**

### **3.4. [5 POINTS] Estimated cost of software products (such as licensed software, etc.)**

Development Tools: JetBrains All Products Pack (\$649 1st year, \$519 2nd year, \$389 afterwards), Fluid UI Team (\$499 per year)

Project Management Tools: Trello Business Class (\$70 per month)

Developer Memberships: Apple Developer Program (\$99 per year), Google Play Developer Program (\$25 one time)

**Estimated First Year Cost: \$2112**

**Estimated Cost After First Year: \$1977**

### **3.5. [5 POINTS] Estimated cost of personnel (number of people to code the end product, training cost after installation)**

Leadership Personnel: 1 Team Lead, 2 Team Coordinators

Coding Personnel: 2 Programmers, 1 Lead Programmer

Design Personnel: 1 Lead Designer

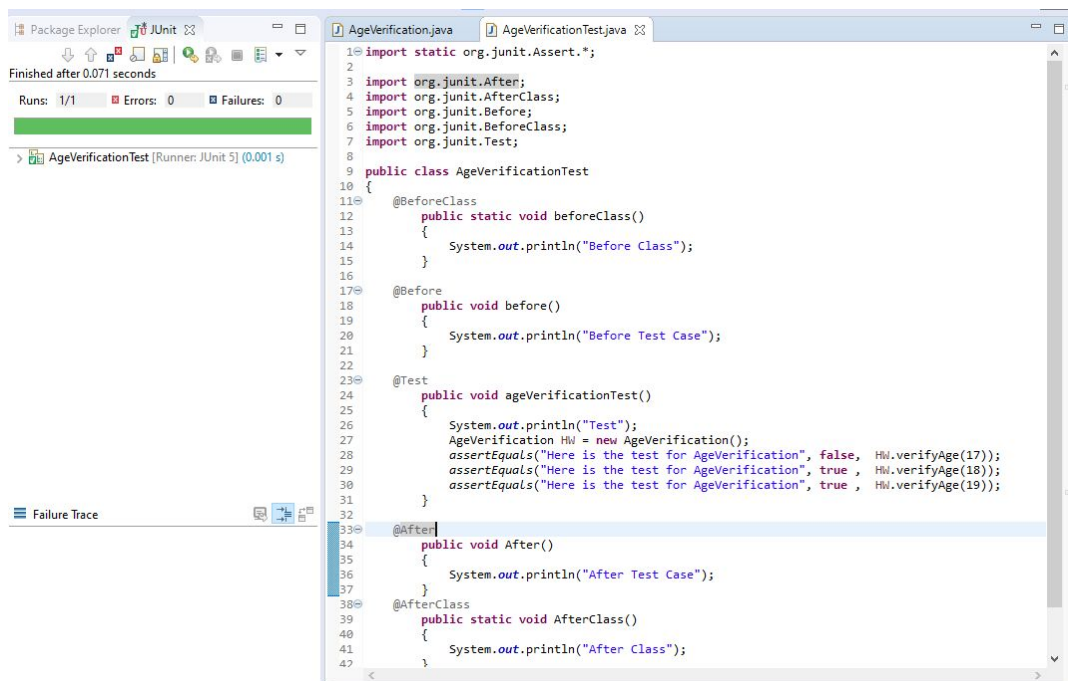
**Total Cost of Personnel: None**

**4. [10 POINTS] A test plan for your software: Describe the test plan for testing minimum one unit of your software. As an evidence, write a code for one unit (a method for example) of your software in a programming language of your choice, then use an automated testing tool (such as JUnit for a Java unit) to test your unit and present results. Clearly define what test case(s) are provided for testing purposes and what results are obtained (Ch 8). Include your test code as additional document in your zip file submitted.**

The software system needs to be able to verify that the user's age is above or equal to the minimum required age of 18. The code includes the method "verifyAge(int age)" which takes in the user's age and returns true if their age is 18 or above, and returns false otherwise.

```
1 public class AgeVerification
2 {
3     public boolean verifyAge(int age)
4     {
5         boolean isOlderThan18 = (age >= 18);
6         return isOlderThan18;
7     }
8 }
9
```

A Junit test was developed to test the functionality of this method with different age inputs. The test cases used were: 17, 18, and 19. The unit test provided the expected results. False was returned by the method when the input age was 17 and true when the input age was 18 or 19.



The screenshot displays an IDE with two main windows. The left window shows the JUnit test results for 'AgeVerificationTest'. It indicates that the test finished after 0.071 seconds, with 1/1 runs, 0 errors, and 0 failures. The right window shows the source code for 'AgeVerificationTest.java'. The code includes imports for JUnit, a 'beforeClass' method, a 'before' method, a 'test' method, an 'after' method, and an 'afterClass' method. The 'test' method contains three assertions: one for age 17 (expecting false), and two for ages 18 and 19 (expecting true).

```
1 import static org.junit.Assert.*;
2
3 import org.junit.After;
4 import org.junit.AfterClass;
5 import org.junit.Before;
6 import org.junit.BeforeClass;
7 import org.junit.Test;
8
9 public class AgeVerificationTest
10 {
11     @BeforeClass
12     public static void beforeClass()
13     {
14         System.out.println("Before Class");
15     }
16
17     @Before
18     public void before()
19     {
20         System.out.println("Before Test Case");
21     }
22
23     @Test
24     public void ageVerificationTest()
25     {
26         System.out.println("Test");
27         AgeVerification HW = new AgeVerification();
28         assertEquals("Here is the test for AgeVerification", false, HW.verifyAge(17));
29         assertEquals("Here is the test for AgeVerification", true, HW.verifyAge(18));
30         assertEquals("Here is the test for AgeVerification", true, HW.verifyAge(19));
31     }
32
33     @After
34     public void After()
35     {
36         System.out.println("After Test Case");
37     }
38
39     @AfterClass
40     public static void AfterClass()
41     {
42         System.out.println("After Class");
43     }
44 }
```

The source files and screenshots are included in the zip file.



**5. [10 POINTS] Comparison of your work with similar designs. This step requires a thorough search in the field of your project domain. Please cite any references you make.**

By doing some research, we found out that there are a multitude of different dating applications. The most popular ones are Tinder, Bumble, OkCupid, and Grindr. Tinder lets each user create a profile with up to 500 characters and 6 images. When looking for matches, Tinder shows a user another user's profile and allows them to either swipe right, meaning they like them, or swipe left, meaning they are not interested. A user can also use a super like to show the other user that they really like them. Tinder has a free version that only allows limited swipes however. For unlimited swipes, users will have to pay a monthly fee. Bumble is very similar to Tinder in the form of profile and selection set up. However, different to Tinder, if a man and woman match up with each other, the woman has to message first. OkCupid lets users create a large profile, rather than a concise one. Users can answer questions as well as what answer you would like your potential match to be. The app then creates a percentile score that shows compatibility between users. Grindr is a dating app specifically for the LGBTQ+ community. The app also allows users to specify what they are looking for, such as finding love, making friends, or just a casual hookup.

Our dating app, 100%, is very much a combination of Tinder and OkCupid. Like OkCupid, our app provides a percentage of compatibility between two users. Like Tinder, once users see another's account and percentage, they can swipe if they like them or not (up for yes, down for no).

**6. [10 POINTS] Conclusion - Please make an evaluation of your work, describe any changes that you needed to make (if any), if things have deviated from what you had originally planned for and try to give justification for such changes.**

The preliminary design of our application has addressed many of the relevant issues and proposes a design architecture that will lead to a well developed mobile app. We have examined and developed schemes for all of the major subsystems of the client side application. As well we took other, similar, applications into consideration and evaluated our design against their design choices. We then developed a rough production timeline and expense estimates required to develop our application. If implemented, our plan would have a high likelihood of reaching completion.

**7. [5 POINTS] References: Please include properly cited references in IEEE paper referencing format. Please review the IEEE referencing format document at the URL:**

<https://ieeedataport.org/sites/default/files/analysis/27/IEEE%20Citation%20Guidelines.pdf>). It means that your references should be numbered, and these numbers properly cited in your project report.

Work Cited

- [1] Apple Inc, "App Store Review Guidelines," *App Store Review Guidelines - Apple Developer*, 12-Sep-19AD. [Online]. Available: <https://developer.apple.com/app-store/review/guidelines/>. [Accessed: 18-Oct-2019].
- [2] M. Jansen, "Looking for Love or Just Some Fun? Cozy up with the Best Dating Apps of 2019." *Digital Trends*, 17 Oct. 2019. [Online] Available: [www.digitaltrends.com/mobile/best-dating-apps/](http://www.digitaltrends.com/mobile/best-dating-apps/). [Accessed on: 17 Oct. 2019]
- [3] "Date, Meet, Network Better," *Bumble*. [Online]. Available: <https://bumble.com/en/>. [Accessed: 17-Oct-2019].
- [4] "Developer Policy Center," *Google*. [Online]. Available: <https://play.google.com/about/developer-content-policy/>. [Accessed: 18-Oct-19]
- [5] "Find the people you've crossed paths with," *happn*. [Online]. Available: <https://www.happn.com/en/>. [Accessed: 17-Oct-2019].
- [6] "Match. Chat. Date.," *Tinder*. [Online]. Available: <https://tinder.com/?lang=en>. [Accessed: 17-Oct-2019].
- [7] OkCupid, "Free Online Dating," *OkCupid*. [Online]. Available: <https://www.okcupid.com/>. [Accessed: 17-Oct-2019].
- [8] M. Jansen, "Looking for love or just some fun? Cozy up with the best dating apps of 2019," *Digital Trends*, 06-Nov-2019. [Online]. Available: <https://www.digitaltrends.com/mobile/best-dating-apps/>. [Accessed: 09-Nov-2019].
- [9] "Enter Business Name (You Can Change It Later)." *App Maker | Free App Creator | Mobile App Builder Online*, <https://snappy.appypie.com/appbuilder/creator-software/build>.