**CS3354 Software Engineering**

**Final Project Deliverable 1**

DatingGo

Brandon Maweu, Yukyeong Ko, Daniel Bae, Elias Khan, Kylie Quinney, Ayush Sharma, Anthony Vu, Alan Nguyen

1. **[5 POINTS]** Well described delegation of tasks, i.e. who did what in the project. Now that your project is complete, you are required to submit the delegation of tasks from beginning of the project until the end. Please make sure to fairly distribute tasks in the team and remember that in the end of the semester, each member of a team will receive the same grade. See grading policy below for more detail. If no/poor contribution by a member, please specify clearly so that we can grade each student fairly.

## Delegated Tasks by Member:

**Deliverable 1:**

Kylie Quinney: Assist in application of appropriate architectural design and creation of diagrams.

Yukyeong ko: Assist in creating architectural design

Ayush Sharma: Make the Class Diagram and help in other diagrams.

Daniel Bae: I will be implementing functional requirements.

Elias Khan: Creating a Use Case Diagram based on our project.

Anthony Vu: I researched and established all the non-functional requirements that the app entails

Brandon Maweu: Assisting in the Architectural Design and setting up the Github Repository.

**Deliverable 2:**

Ayush Sharma: Cost, effort and pricing estimations

Elias Khan: Estimating the effort and cost required for our software or additional software needed.

Daniel Bae: I will help analyze the estimated cost for required software and potential hardware products.

Kylie Quinney: Create estimation of project scheduling.

Brandon Maweu: Creating the Software Testing plan and creating unit tests for functions in the application.

Anthony Vu: I will be planning and establishing the software test plans to test the functionality of the app. I will also compare our system to similar apps and design user interfaces for our app.

Alan Nguyen: I will be assisting in Software Test Planning.

Yukyeong ko: I will assist in building a test plan for testing our unit and results.

## Total Task:

Deliverable 1:

* Feedback from proposal.
* Diagrams (use case + sequence + class) (3 people: Ayush, Kylie, Elias)
* software requirements (functional and non functional) (2 people: Daniel, Anthony)
* 1 architectural design (2-3 people: Kylie, Brandon, Yukyeong)
* Github Stuff - individual (except tasks 1.3-1.5) (Brandon for set up)?

Deliverable 2:

* Project scheduling (1 person: Kylie)
* Cost, effort and pricing estimations (software + hardware + personnel) (3 people: Daniel, Elias, Ayush)
* Software test Plan - (4 people: Brandon, Anthony, Alan, Yukyeong)
* Comparison of our work to similar designs (Anthony)
* Conclusion (Yukyeong)
* References (everyone)
* Presentation slides (conclusion/evaluation + comparison + references) (everyone)
* Implemented code??

## Scholarly Paper: No

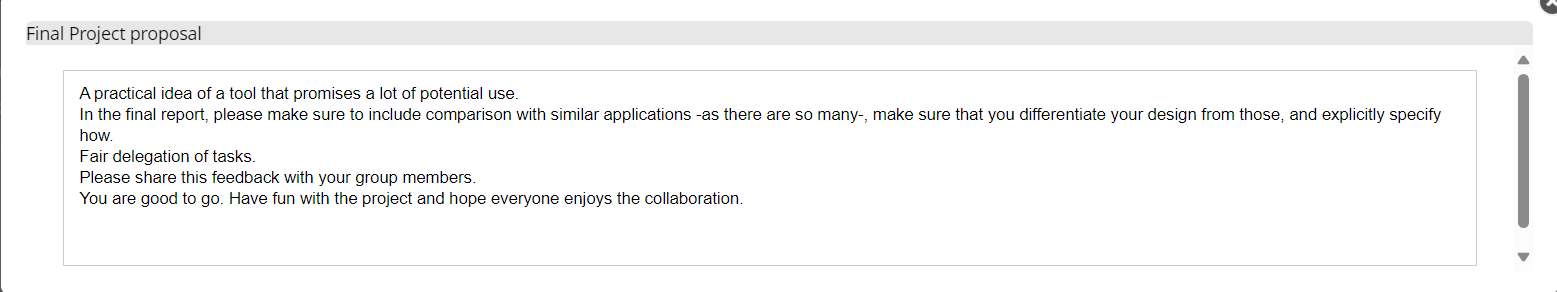
**2. [10 POINTS] Everything required and already submitted in Final Project Deliverable 1. Please specify this part as “Project Deliverable 1 content”.**

## Implementation:

We are designing a potential dating app that incorporates unique social elements and innovative technology. It is a mobile app that encompasses GPS tracking and real time user interaction where they can send requests, view, and meet up with each other. It will have similar geographic and concurrent interaction as the popular mobile game ‘Pokemon Go’. The purpose of the app is for social interaction/dating which is why it will share some similarities as ‘Tinder’. Users will be able to create and view other profiles. However unlike those apps, ‘DatingGo’ aims to combine elements from both and encourage in person mingling in public areas where interested individuals in the same area are given the ability to connect.

## Motivation:

We decided on a dating app because it presents a unique challenge. We want to add our own twists on the basic dating app design in a way that is relevant to our age demographic. We recognized that there was a strong demand for matchmaking apps, and our unique app would combat the common issues of miscommunication and awkwardness prevalent in present-day dating apps. We expect that this app will be used by single individuals throughout the nation as a new and fun way to meet possible romantic partners.



To address the feedback, we documented how our app was similar and different to the apps Tinder and Pokemon Go in the implementation section. The fair delegation of tasks was addressed by having team members help out other team members with their tasks when they were stuck or needed more assistance.

1. **[10 POINTS]** Setting up a Github repository. Please use your utdallas email accounts only for each group member.

<https://github.com/BrandonMDallas/3354-DatingGo>

* 1. Each team member should create a GitHub account if you don’t already have one.
  2. Create a GitHub repository named 3354-teamName. (whatever your team name will be).
  3. Add all team members, and the TA as collaborators. Our TA has already posted her GitHub info in EL:

TA GitHub id:

TA email:

* 1. Make the first commit to the repository (i.e., a README file with [team name] as its content).
  2. Make another commit including a pdf/txt/doc file named “project\_scope”. If you choose a predefined topic (one of the 4 topics described in the “Project Topic Ideas” section of this document), the contents of the file should be identical to the corresponding project in this section. If you choose other topics, the contents should follow a similar structure.
  3. Keep all your project related files in your repository as we will check them. Include the URL of your team project repository into your project deliverable 1 report.

**Important Note:**

* + - Tasks 1.3 - 1.5 should be performed by different team members. We will check the commit history for these activities.
    - Do not include credentials (e.g., UTD ID) in the repository.
    - Only commits performed before the deadline will be considered. Do not forget to push your changes after you have done the work!

1. **[5 POINTS]** Delegation of tasks: Who is doing what. If no contribution, please specify as it will help us grade each group member fairly.

## Delegated Tasks by Member:

**Deliverable 1:**

Kylie Quinney: Assist in application of appropriate architectural design and creation of diagrams.

Yukyeong ko: Assist in creating architectural design

Ayush Sharma: Make the Class Diagram and help in other diagrams.

Daniel Bae: I will be implementing functional requirements.

Elias Khan: Creating a Use Case Diagram based on our project.

Anthony Vu: I will be researching and establishing all the non-functional requirements that the app entails. I am also defining the project scope for our app and describing the main services as well as the software process model our app follows.

Brandon Maweu: Assisting in the Architectural Design and setting up the Github Repository.

**Deliverable 2:**

Ayush Sharma: Cost, effort and pricing estimations

Elias Khan: Estimating the effort and cost required for our software or additional software needed.

Daniel Bae: I will help analyze the estimated cost for required software and potential hardware products.

Kylie Quinney: Create estimation of project scheduling.

Brandon Maweu: Creating the Software Testing plan and creating unit tests for functions in the application.

Anthony Vu: I will be writing a unit test for one method of our software. I will also be comparing our system design to similar apps and designing the user interfaces to demo in the presentation.

Alan Nguyen: I will be writing the conclusion.

Yukyeong ko: I will assist in building a test plan for testing our unit and results.

1. **[5 POINTS]** Which software process model is employed in the project and why. (Ch 2)

The software process model employed is the prototyping software model because it is great for mobile apps that depend on its user base. We will create a prototype of the app to test and rework based on user feedback. This will allow us to quickly develop the basic functionality and vision of the product and continually make changes to achieve the desired complete app for our users.

1. **[15 POINTS]** Software Requirements including
   1. **[5 POINTS]** Functional requirements. To simplify your design, please keep your functional requirements in the range minimum 5 (five) to maximum 7 (seven). (Ch 4)

Creating Profiles

1. The app must allow users to customize and create their own profiles. This would include creating user bio, and changing settings to suit their individual preferences. This serves as the user’s online identity within the application.

Setting Mingling Status

1. To manage their visibility on the applications map, users may select their mingling status. Users in the area may see where they are when they have the mingling status set to on, as shown by a pin on the map. By setting the status to “off”, one may regulate either visibility and engagement availability by hiding their location.

Viewing Potential Matches

1. Within their location, users can display perspective matches as pins on a map. They may view the profile of another user by clicking on a pin. Through interactive exploration, this technology encourages user involvement by displaying potential matches based on their parameters and individual preferences.

Compatibility Score

1. Using a thorough examination of user behaviors and preferences, the app determines and presents a compatibility score for every possible match. This score adds subtlety to the decision making process by assisting users in determining the probability of a good match.

Sending Mingling Requests

1. Users can start a conversation with possible matches by sending them a “mingling request”. Requests are open for users to approve or reject. When accepted, they proceed to the new level, “mingle”, which allows them to have a more in-depth interaction/engagement.
   1. **[10 POINTS]** Non-functional requirements (use **all** non-functional requirement types listed in Figure 4.3 - Ch 4. This means provide one nonfunctional requirement for each of the leaves of Figure 4.3. You can certainly make assumptions, even make up government/country based rules, requirements to be able to provide one for each. Please explicitly specify if you are considering such assumptions.)

Non-Functional Requirements

Usability

1. The app will run on both IOS and Android devices.

Performance

1. The app should load all users in the area on the map within 1.5 seconds

Space

1. The app should be able to use up to .025 GB per hour for the real time maps and locations.

Dependability

1. The servers should have an uptime of 98%.

Security

1. All network traffic should be sent over HTTPS so that it is encrypted.

Environmental

1. The app should be usable in all areas covered by GPS satellites.

Operational

1. There should be different geographical servers to balance all the proximal concurrent users and their requests.

Development

1. The system should support the deployment of updates to the app in case of bugs or additional features.

Regulatory

1. The app will follow data privacy and protection laws that require users to be informed how the app collects and utilizes their data as soon as they download the app.

Ethical

1. User data will not be sold to 3rd party services.

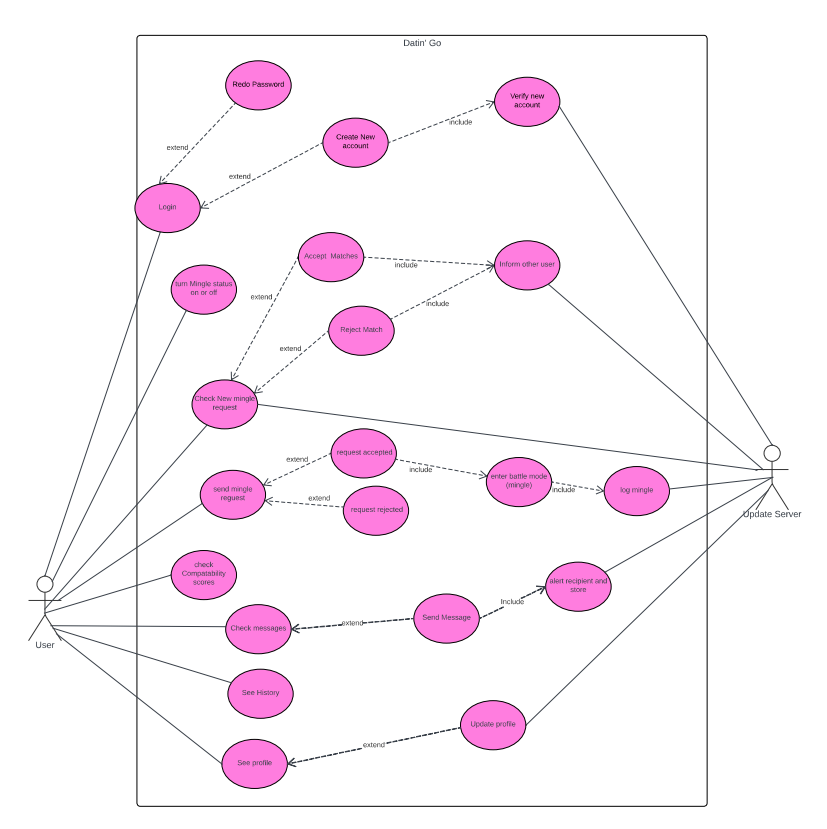
Accounting

1. An advertising model will be used that allows in-app ads to be displayed to generate revenue. The documenting of revenue from total ads displayed will follow the standards for financial reporting as defined in GAAP which most US companies follow.

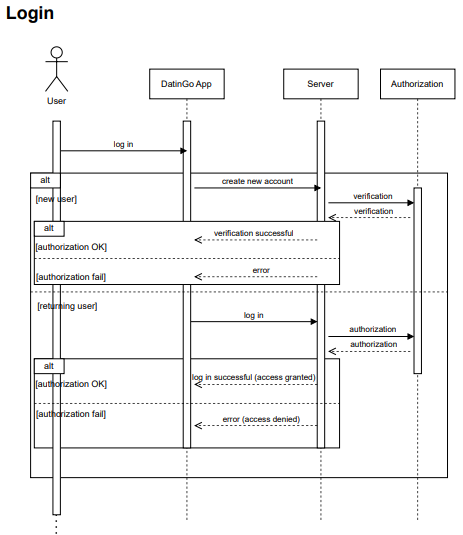
Safety

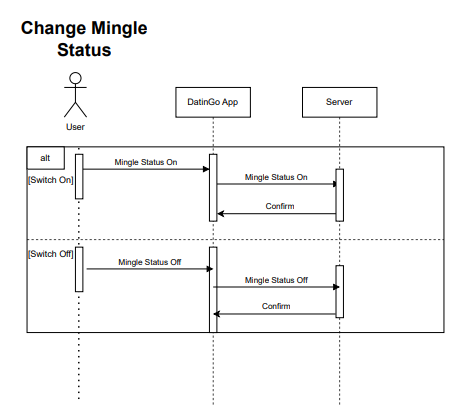
1. All users must be 18 years or older to use the app.

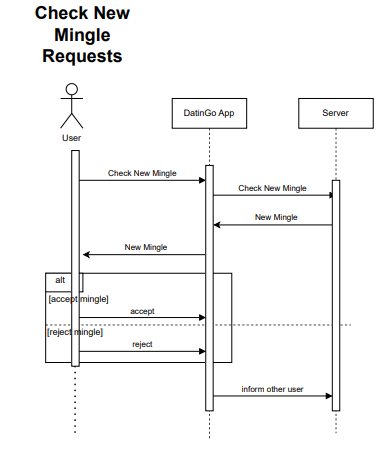
1. **[15 POINTS]** Use case diagram – Provide a use case diagram (similar to Figure 5.5) for your project. Please note that there can be more than one use case diagrams as your project might be very comprehensive. (Ch 5 and Ch 7)

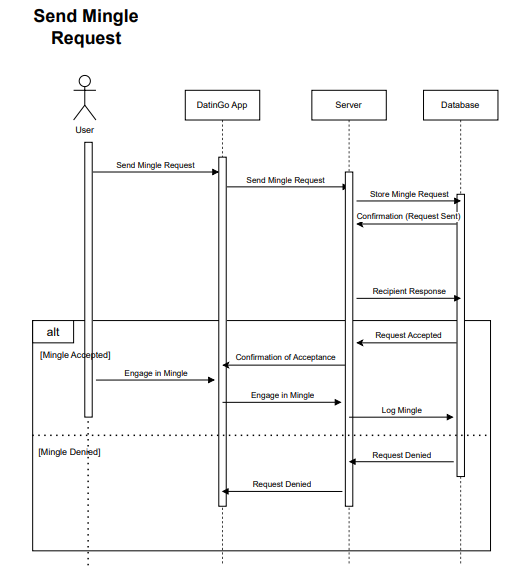


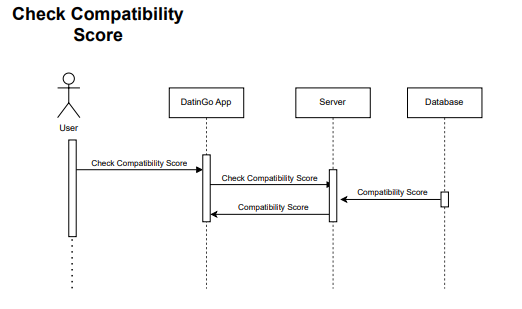
1. **[15 POINTS]** Sequence diagram – Provide sequence diagrams (similar to Figure 5.6 and Figure 5.7) for each use case of your project. Please note that there **should** be an individual sequence diagram for each use case of your project. (Ch 5 and Ch 7)

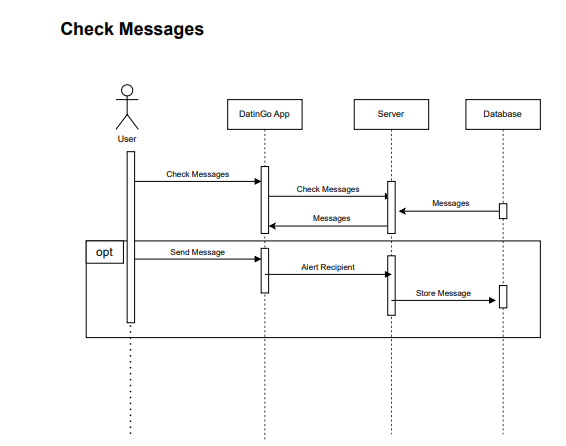


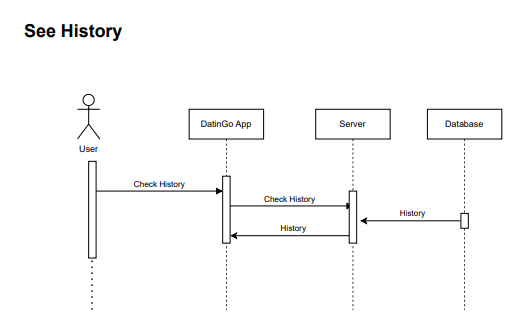


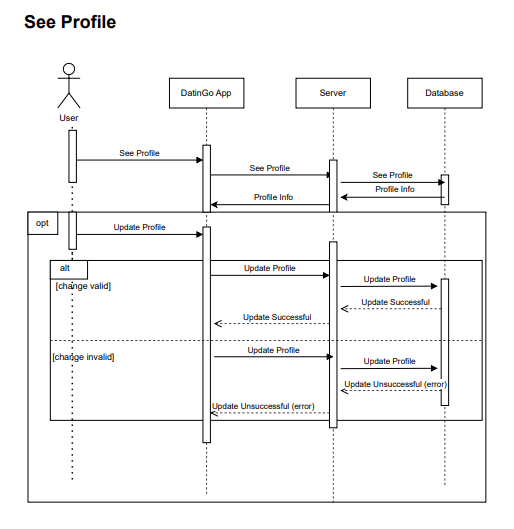




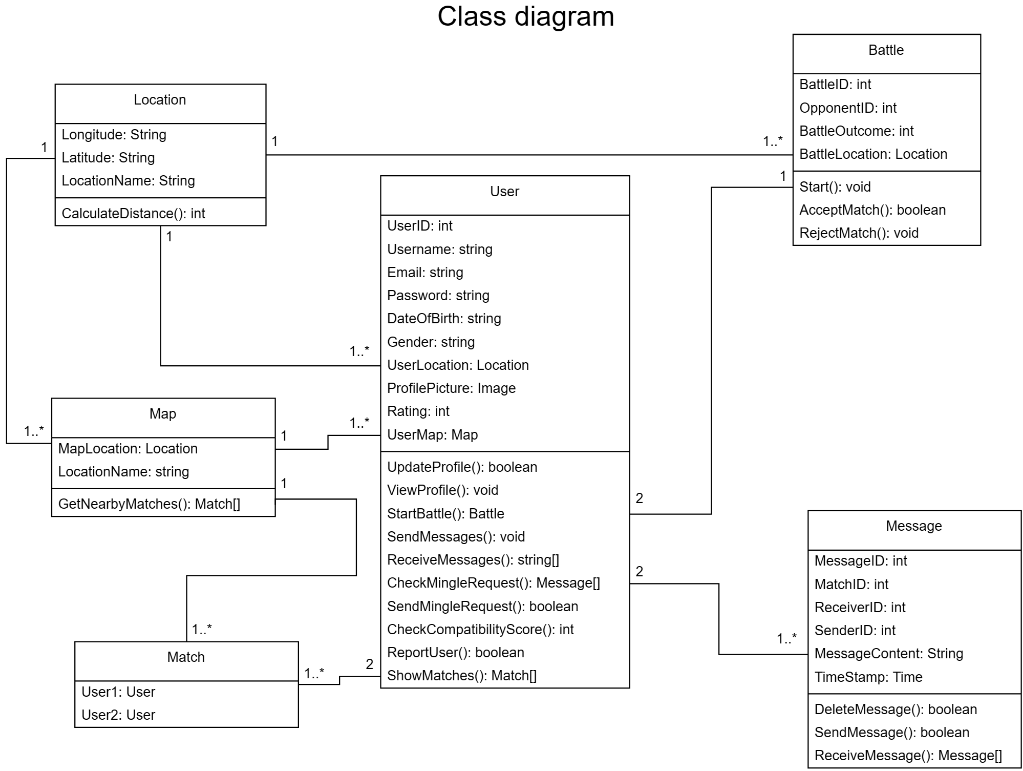




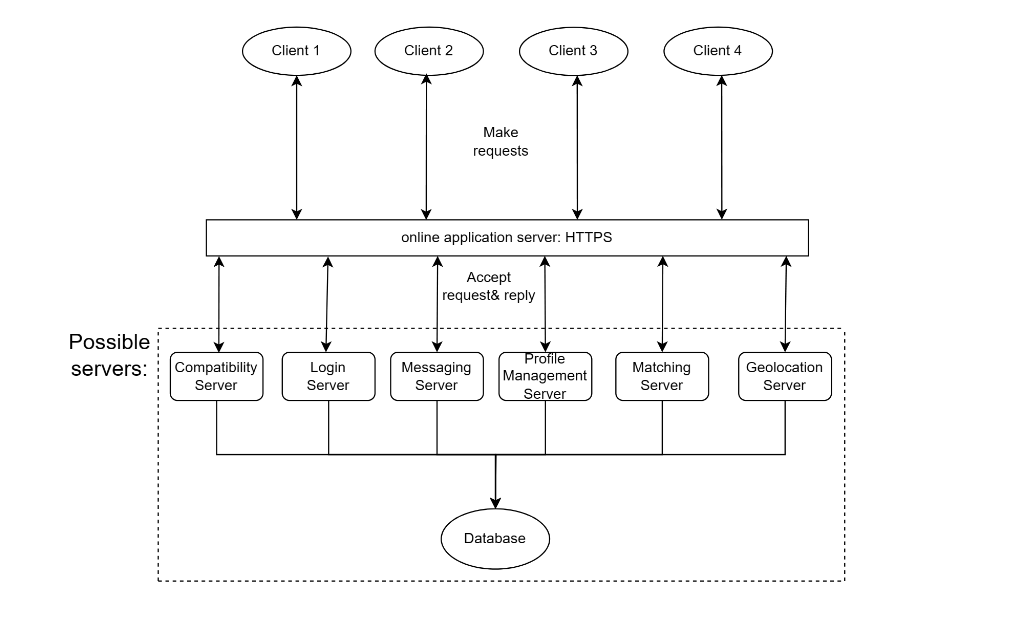




1. **[15 POINTS]** Class diagram – Provide a class diagram (similar to Figure 5.9) of your project. The class diagram should be unique (only one) and should include all classes of your project. Please make sure to include cardinalities, and relationship types (such as generalization and aggregation) between classes in your class diagram. Also make sure that each class has class name, attributes, and methods named (Ch 5).



1. **[15 POINTS]** Architectural design – Provide an architectural design of your project. Based on the characteristics of your project, choose and apply **only one** appropriate architectural pattern from the following list: (Ch 6 section 6.3) 9.1. Model-View-Controller (MVC) pattern (similar to Figure 6.6)
   1. Layered architecture pattern (similar to Figure 6.9)
   2. Repository architecture pattern (similar to Figure 6.11)
   3. Client-server architecture pattern (similar to Figure 6.13)
   4. Pipe and filter architecture pattern (similar to Figure 6.15)

****

We decided to take the Client-Server approach for our dating application because it allows for the functionality of our system to be organized in services. The users of our applications are all clients that use these services which splits the workload between entities. In addition, security and authentication were one of our major concerns and the client-server architecture approach excels in data security and allows for scalability into the future. Furthermore, deploying new fixes to our application can be deployed on the server which would not require clients to update their version of the application since it would be a fix for the service that the server is providing. Lastly, having a centralized storage for data is important since profiles, messages, and geolocation data are being handled. Things such as pictures, private messages, and location should be handled with importance, and having a centralized secured server is important to our application.

Project Scope:

Mobile Dating Application

1.1 Profile Software

1.1.1 Create, View, Delete Profile

1.1.2 Support adding photos for your profile.

1.1.3 Support adding a bio for your profile

1.1.4 Support adding hobbies for your profile

1.1.5 Support adding preferences for your profile

1.2 Mingling

1.2.1 Allow a user to initiate and send a mingle request to

another user

1.2.2 Allow a user to accept or reject a mingling request

1.2.3 Send and receive messages to accepted matches

1.2.4 Search messages via text query

1.2.5 Support blocking profiles & messages

1.2.6 Allow termination of current mingle

1.2.7 Tracks how long two users mingle with each other which is

used for the calculation of the compatibility score with

future possible matches

1.3 Matching

1.3.1 Display all other users open to mingle in the same area

1.3.2 Allow user to view the profiles of other users in the area

1.3.3 Display compatibility scores between two users based on

the calculation of preferences and matching history

1.3.4 Allow a user to turn off mingling status and have

geographic location not be displayed to other users

1.3.5 Support reporting profiles

1.4 Geographical Tracking and Displaying

1.4.1 Tracks the geographic location of all concurrent users

1.4.2 Support a graphical geographic map that displays users as

clickable pins and mingling hot spots

1.4.3 Calculates and displays mingling hot spots on map based

on trending and frequent mingling of users

3. [35 POINTS] Project Scheduling, Cost, Effort and Pricing Estimation, Project duration

and staffing: Include a detailed study of project scheduling, cost and pricing estimation

for your project. Please include the following for scheduling and estimation studies:

3.1. [5 POINTS] Project Scheduling.

Through function point analysis, we are able to determine that our project will take about 1.5 (working) weeks to complete using a team of 3. The details of the schedule are outlined below.

Project Start Date: Monday, May 13th 2024

Project End Date: Friday, May 24nd 2024

Our project is scheduled to commence on Monday, May 13th, 2023, and conclude on Friday, May 24nd, 2023. The estimated duration for the project is 1.5 weeks, calculated based on a 3-person team working 8-hour days from Monday through Friday. Weekends will be excluded from the schedule, allowing the team to have Saturday and Sunday off. This results in a total of 10 working days, providing ample time for the team to complete the project's tasks. This timeline is made under the assumption that there will be breaks in work throughout the day, but it is essential to note that real-world projects may encounter unforeseen challenges or require additional time for meetings, reviews, or adjustments.

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. Please choose one of the two alternative cost modeling techniques and

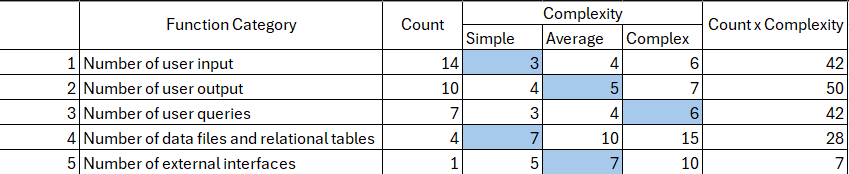
apply that only:

- Function Point (FP)

- Application composition

**Functional Point analysis:**

We have chosen the functional point analysis for cost/effort estimation as it provides the most flexibility in changing requirements hence changing the overall plan. In the FP estimation method we can easily change the complexity weight as well as the count of any function and other metrics can be easily and automatically recalculated to reflect the changes.   
  
The complexity selected for each function is marked in blue.

  
 User input: Inputs include profile section such as account information, creation, and mingling, for potential matches multiple inputs in the form of location, response, requesting, accepting, and messaging.

User output: Includes profile information, mingling status, locations, matches, incoming requests, incoming chats, and compatibility score.

Queries: Includes account creation, compatibility, matches, location search, mingle request, account updating, searching users.

Data Storage: Profile data, Messages, Map data and Mingling data.

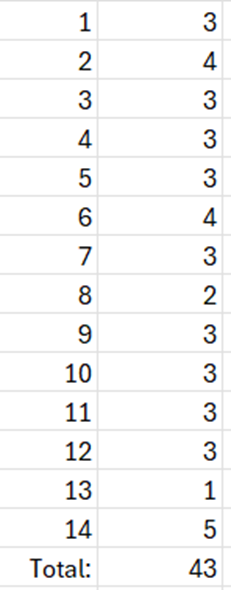
External: Maps

Count x Complexity is the product of a function count and the selected complexity for it.

The GFP is then the sum of each of the Count x Complexity.

GFP = 42 + 50 + 42 + 28 + 7

= 169

Next step is to calculate the PCA  


PCA = 0.65 + 0.01(43)

= 1.08

Now using the PCA we can calculate the FP.

FP = GFP x PCA

= 169\* 1.08

~ 183

The estimated effort can be calculated as,

(Productivity is assumed as 40 per week)

E = FP / Productivity

= 183/40

= 4.6

Team Size is assumed to be 3.

Duration is estimated as,

D = E / Team Size

= 4.6 / 3

= 1.5

~ 1.5 weeks

Answer: An estimated 1.5 weeks of effort will be required to build the software when a team size of 3 is chosen.

Total estimate cost to develop our software = 4.5 person-months (Effort) \* $6,000 per person-month \* 0.375 months (Duration) + ≈ $10,125 USD

$6,000 is the salary given to our employees, and this is based on the average CS salary. This may change depending on how much profit is brought in by our application. Furthermore, relying on a kickstarter or sponsorship for the initial payment to our employees will be ideal. Half the salary will be disbursed within the first couple of days depending on the progress. The rest will be disbursed after launch.

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

The two servers will consist of a physical server and a cloud server.

This would be considered a hybrid client-server setup utilizing an on-premise and cloud server. This hybrid approach would ensure maintainability and reliability between managing user interactions and data.

**On-premise Server** - **Dell PowerEdge R450**

Benefits:

* Easy to maintain and manage
* Lower operational demands
* Beneficial for smaller teams / businesses, accounting in for our initial application development and dating app requirements/necessities.

Specifications:

* **Processor:** Intel Xeon Silver 4310 (2.1GHz)
* **Memory:** 16 GB DDR4
* **Storage:** 1x 480GB SSD SATA
* **Estimated Cost:** $3,089

**Cloud Server - AWS EC2 c5.2xlarge**

Benefits:

* Could manage high user interactions without disrupting performance.
* Best suited for our growing application that may need to manage increasing loads, supporting scalability.
* Sufficient for handling loads without overspending.

Specifications:

* **vCPU:** 16 virtual CPUs
* **Memory:** 32 GiB (Gigabytes, 1GiB ~ 1.074 GB)
* **Instance Storage:** EBS - Only
* **Network Bandwidth:** Up to 10Gbps
* **EBS Bandwidth:** Up to 4,750 Mbps
* **Estimated Cost:** Rate for On-Demand Pricing: $0.68 per hour
  + $0.68/hour x 730 hours = $496.40 / month

On-premise Set Up Fee: $3,089

On-Demand Cloud Server: $496.40 / month

Note that only the first month will be included in our total costs, as future payments will be taken out of the profit made by our software.

3.4. [5 POINTS] Estimated cost of software products (such as licensed software, etc.)  
  
Software that will be required to develop this software would include a collaborative IDE, a design tool, and a Version Control System.

* For a Collaborative design tool Intellij IDEA can be used which can cost nearly $600 per developer per year so $600 \* 3 = $1800.
* For a Design tool would be $10 per month so for 3 developers for less than a month will be = $30.
* The free version of GIT and hosting it on Github will be more than sufficient for the purposes

Total Software Cost: $1800 + $30 = $1830.

3.5. [5 POINTS] Estimated cost of personnel (number of people to code the end

product, training cost after installation)

The estimated team size is 3. Training costs after installation will be associated with the technical support team and ongoing developer training. The training technical team supports and handles queries and troubleshoots issues that users may face, and for maintenance, the development team will take updates on the latest in app development, security measures, and other relevant technologies. Training includes certifications on specific programming languages (JAVA/Python/JS) used as well as security or cloud certifications (AWS) and on average may cost $50. Average laptop such as a Dell laptop can cost $1500. As such, our total estimated personnel cost is 3 \* [50 (training cost) + 1500 (laptop cost) ] = 4650.

Total cost of our project = cost of software development + cost of hardware + cost software + cost of personnel + cost of maintenance.

Typically, software maintenance costs about 15-20% of the original development costs per year, so total cost for personnel will be 0.2(10,125) = $2025.

Thus our total costs = $10,125 + $1830 + ($3,089 + $496.40) + $4650 + $2025 = $22215

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.

We decided to create a test plan for two modules in our dating application. These modules are our MingleRequest system and our Profile System. We decided to use Blackbox testing techniques to unit test our functionalities. We decided to use an equivalence partition to determine our test cases for our different functions.

For our mingle request system we wanted to test valid and invalid inputs and their appropriate responses. The function mingleResponse(int responsecode) is a function that takes an integer and determines whether a user has accepted, denied, or is pending your mingle request.The MIN & MAX values for our partitions are 100, and 500 respectively. The valid inputs for mingleResponse are two response codes which are {200,202} and invalid inputs are [MIN,200) & [201] & (202,MAX]. For our valid inputs a responsecode of 200 should return 1 meaning that the user has accepted your mingle and a response code of 202 should return a value of 2 which means your mingle request is still currently pending. It returns a value of 0 otherwise meaning the request wasn’t accepted.

For our profile system we wanted to test valid and invalid inputs for our user’s age. The function validAge(int age) takes in an integer and determines whether that age is valid. Our requirements for the application is that users under the age of 18 can not access our application. The MIN and MAX values for our age test cases are 0 and 121 respectively. The valid inputs for age are [18,120], the invalid inputs for age are [MIN,18) & (120, MAX]. Our test cases for validAge are the numbers 17, 21, 122 which covers all the possible results from our function. Age 17 should return false as it's an invalid age since the user is younger than 18. Age 21 should return true as it’s a valid age in our range from [18-120]. Lastly, Age 122 should return false as it's an age beyond the ranges that we take for our application.

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.

Our system has some similarities to Tinder with profile management and matchmaking algorithms. We offer the capability to allow users to customize their own profiles with pictures, bios, preferences as well as view other profiles. Our system will then store all data related to profiles like Tinder. Another similarity is the use of matchmaking algorithms. Tinder assigns a score to each user based on app behavior, and then groups users into similar buckets (1). Our app works on a similar premise by assessing the user behavior and past mingling outcome history to group users. Then when a user chooses to view another user, a compatibility score between the two users is computed based on the likelihood of a successful interaction based on similarities. Our system will also have similarities to Tinder and Pokemon Go in terms of GPS capabilities and geosharding. To distribute load and cluster user data geographically, we will have many servers distributed regionally. These other apps use a similar technique to deal with high density areas as well as cache relevant data to a specific region for increased performance (1). Our app will be able to geographically track users and display them and other users in the same vicinity on a viewable map. Our geosharding will help ensure user queries and data are clustered to the relevant region where a user is located.

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.

In conclusion, our team has planned to design a potential dating app that accompanies a solid foundation for socializing with various individuals with GPS tracking technologies. Unlike other common popular dating apps, it focuses on encouraging people to mingle with one another within a specific nearby area and allows them to connect with each other. The goal was to create a unique dating experience that encourages users to explore new places, meet new people, and form playful and engaging connections. Key feature of this dating app is combining the property of a general dating app and the augmented reality mobile game Pokemon Go, which encourages interaction users to cross over the real and virtual world actively. Concerning developing dating apps, effective, user-friendly, and secure properties should be ensured for the app. Since ‘DatingGo’ like the one mentioned above is based on combining properties of general dating apps like Tinder and ’Pokemon Go’ game, usability is intuitive and easy to navigate. It has robust profile management including photos, bios, hobbies, and preferences, which allows user to connect their information through social media. Mingling functionalities of ‘DatingGo’ enable users to initiate or respond to mingling requests, and exchange messages, and effectively ensure the credibility of the application system by reporting or blocking the users. Especially, the platform pays attention to increasing access of users by providing providing potential matching system depending on geographical proximity and personal preference. The app's geographical tracking capability enhances user experience by pinpointing mingling hotspots on an interactive map, encouraging exploration and real-world engagement. The platform makes users involved in the connection and matches within their nearby location through the Battle mode function and system providing Map. An innovative aspect of the platform is its ability to track and analyze the duration and quality of interactions, which contributes to a compatibility score that aids in future match suggestions.

The most cared part is that security and privacy will be considered by providing compatibility score and hiding individual-related statues like mingling, and location while robust mechanisms would be in place to verify user identity in authentication. Also, according to data management, data transmission and storage for user data will be protected through encryption in databases and files.

Despite some deviations from the initial expansive feature set due to budget and technical limitations, these adjustments were necessary to focus on core functionalities and ensure user privacy and safety, aligning with user feedback and emphasizing a balanced approach between engaging game elements and practical dating features.

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.

[1 ] K. Dissanayake, “Tinder — Fully explained System Design and Architecture,”

*Medium*, Nov. 15, 2023.

https://kasunprageethdissanayake.medium.com/tinder-fully-explained-system-design-and-architecture-1225ecdfe64e

[2] Boris Shiklo, Andy Lipnitski, “Software Maintenance costs”  
 ScienceSoft  
https://www.scnsoft.com/software-development/maintenance-and-support/costs#:~:text=Depending%20on%20the%20software%20type,two%20different%20cloud%2Dbased%20solutions.

Also include:

8. [10 POINTS] Presentation slides. No min/max number of slides enforced. Please make

sure that you can complete presentation within 20 (twenty) minutes.

Following template could be a good start to prepare your presentations. As each project

topic is different, a variety in presentation style is expected and welcome.

- Title of your project together with participants

- Objective of the project designed

- Cost estimation

- Project timeline (timeline of the project designed, NOT the time you’ve spent

on it)

- Functional and non-functional requirements. If too long, select representative

items.

- Use case diagram

- Sequence diagram for a selected representative operation of the project.

- Class diagram

- Architectural design

- Model-View-Controller (MVC) pattern (similar to Figure 6.6)

- Layered architecture pattern (similar to Figure 6.9)

- Repository architecture pattern (similar to Figure 6.11)

- Client-server architecture pattern (similar to Figure 6.13)

- Pipe and filter architecture pattern (similar to Figure 6.15)

- Preferably a demo of user interface design that shows screen to screen

transitions though no full functionality is required.

- OPTIONAL: IF implemented the project, a demo of your implementation.

9. OPTIONAL PART [POSSIBLE EXTRA CREDIT UP TO 10 POINTS]. Your

program code (if fully implemented the project, not required otherwise). Please

note that implementation is not required for the final project. Groups are welcome

to implement their work, if they choose to do so.

[This part may qualify for extra credit, if you implement and submit the

implementation code together with your project. The extra credit will be

determined based on the quality of your implementation.

Furthermore, any fully implemented project qualifies for scholar publication

afterwards. This most probably will involve further commitment to work more

an write a scholar paper to send to a Conference for publication.]

10. [5 POINTS] GitHub requirement:

Make sure at least one member of your group commits everything for project deliverable

2 to your GitHub repository, i.e.

- Your final project deliverable 2 report

- Unit test code for a sample unit of your project

- Implementation code (if you have implemented your project)

- Presentation slides

Still, one member of your team should also submit the required project deliverable 2

materials to eLearning.

Please note: This is just a suggested outline. You are welcome to add more content if

you feel necessary.