SWE 3313

Software Requirements Specification Document

Flock

Group 11

Ben Pallotti, Emmanuel Okafor, Victoria Miller, Alex Nguyen, Armando Ortiz

Team Members: for each member indicate

All members participated equally

a: did his/her share
b: did more than his/her share
c: did less than his/her share (explain if necessary)
d: did nothing

Emmanuel Okafor:**a** Armando Ortiz:**a** Alex Nguyen: **a** Victoria Miller: **a**

SWE 3313

Version: (1)
Date: (10/9/2022)

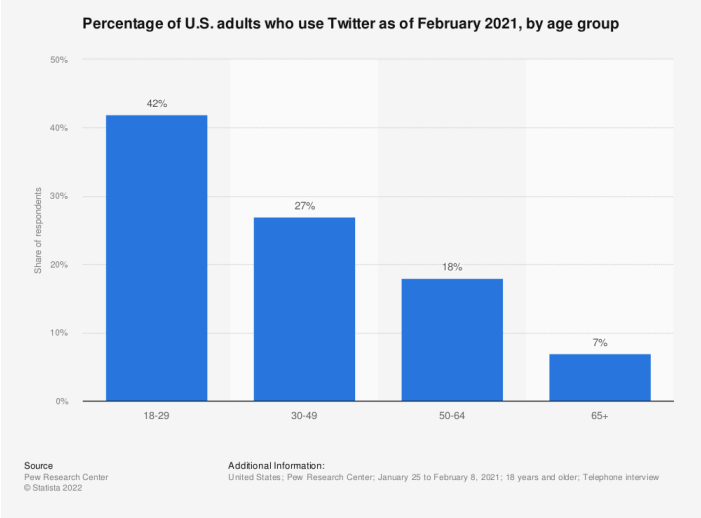
TABLE OF CONTENTS:

1. Introduction

1.1 Purpose

This software requirement specification document is intended to give an overview of an application that uses an algorithm to find the interest of twitter users. The app will then use these common interests to be able to connect with other users. The user will be able to choose if they seek to build a friendship, date, or network with other users. This app will be called Flock. The main programming language that we will be using is Java.

The intended audience for Flock is people between the ages of 18-35 years old. We are targeting this age range because it contains the highest number of twitter users. People who are in college or at work can find it hard to have time to make relationships, however flock would make it easier since Flock will help you find other people that have the same interests as the user.



1.2 Scope

The software product to be produced will be called **Flock**. It is a mobile application that, at minimum, should be able to run properly on the latest Samsung mobile devices as well as its previous two generations. The overall theme of Flock is, “Birds of a Feather, Flock together”. Thus, its objective and main goal is to surround the user with people of their own kind. The application’s purpose is that of a social networking service that helps find communities, friends, or even a significant other based on who or what you follow, like, or retweet on Twitter. For example, if a customer were to select that they were looking for a friend on Flock, then it would look for active Twitter users that follow the same accounts the customer follows, look for Twitter posts that the customer also liked with (prioritizing the customer’s more recently liked posts rather than older ones), and finally look for accounts that retweeted what the customer has also retweeted, or liked the original post of what the customer retweeted. After analysis, the application recommends Twitter Accounts based on how similar it is to the follows, likes, and retweets of the customer’s Twitter Account. The more similar a user’s likes, follows, and retweets are to the customer’s, the more likely they will be presented as a recommended friend.

1.3 Definitions, Acronyms, and Abbreviations

Retweet: A feature of Twitter that allows you to essentially copy a post made by another Twitter user and post it yourself, allowing you to share the Tweet with your own followers. This feature also credits the original post and user.

Like: A feature present in Twitter, but found in many social media platforms. A button, denoted by a heart shape on Twitter which found underneath a Tweet, that when clicked allows you to express to Twitter and anybody else viewing or following you that you "liked", were interested in, or enjoyed a post.

Follow: A feature present in Twitter and many social media platforms. If you click the "follow" button, located on another user's account/profile, then you are able to be notified about said user's various activities. These activities include what they post, what they comment on another post, what they like, who they follow, and what they retweet.

Social Networking Service (SNS): An online/web-based platform with the purpose of facilitating relationships with other people through same interests, background, communication, or communication. Popular examples include Twitter, Facebook, Instagram, MySpace, and TikTok

Match: A match in the case of Flock is when a user of the application finds a friend, date, or community on Twitter thanks to the use of the software

1.4 References

Twitter demographic bar chart on 1.1 purpose, Oct 9 2022, https://www.statista.com/statistics/265647/share-of-us-internet-users-who-use-twitter-by-age-group/

2. Specific Requirements

1.1 Flock must be able to run on at least an Android mobile device

1.2 Flock must be able to operate properly on the latest mobile device and its two previous generations (if IPhone 14 is latest, Flock must be able to run on IPhone 13 and IPhone 12)

2.1 Flock must be able to track and document what the customer has liked, followed, and retweeted on Twitter

2.2 Flock must be able to find other Twitter users with the same likes, follows, retweets, and like or retweets (if customer likes a post that the user retweeted, or if the user likes a post the customer retweeted) as the customer's on Twitter

3.1 Flock must have a search feature where you can search for other Twitter users based on the keywords you use on your tweet (if a customer were to say "love dogs" in their Tweet, then Flock must be able to find another user who also tweets "love dogs")

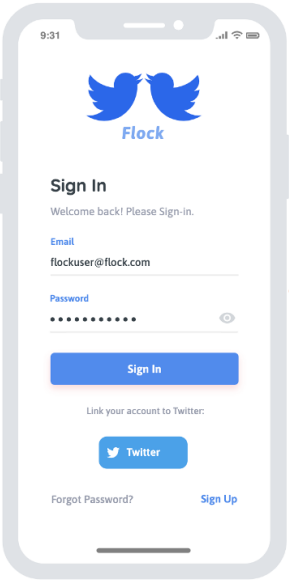
3.2 Flock must allow customers to be able to customize their searches for Twitter users, such as a search for only one similarity (in likes, follows, retweets, or keywords), two similarities, or all four

4.1 Flock must prioritize matches between other Flock users (if a customer is searching for a friend and a Flock user is searching as a friend, but does not share as many similarities in likes, retweets, or follows as a Twitter user who is not on Flock, the Flock user will still be presented as one of the recommendations

5.1 After a customer makes a match, Flock will either create a chatroom between the users if the match also uses Flock, or will send them to Twitter to talk to their match

2.1 External Interfaces

Our main external interfaces are user input and access to Flock user's Twitter account. When the user opens the Flock app it will ask them to either login or to link their account to twitter. We will need the user to give us permission to access their twitter. This will allow the app's algorithm to find similarities with other users, and will allow them to connect with other users. Another area where we will require user input is in the main menu the user will have the option to choose what they would like to do on the app. The user can choose what type of relationship they are looking for (dating, friendship, or networking).



2.2 Functional Requirements

- Data Storage
  - The system shall track and store the users likes, follows, and retweets on Twitter
  - The system shall utilize a key word feature. In which a user will use specific keywords in their tweets to expand the user's data alongside the likes, follows, and retweets.
  - The system shall have users select if they are looking to be connected with the intention

of networking, dating, or building friendship.

- User Matching
    - The system shall utilize this data to compare users' data
    - The system shall connect users based on their common data
    - The system shall have a search function that operates through the user searching for keywords and finding other users manually by searching the database for users that have utilized that keyword or contain it in their data.
- Post Match
    - The system shall connect these users by notifying them of a match
    - The system will then connect them through a live chat room.

2.3 Non-Functional Requirements

The below are all Non-Functional Requirements that we have for our project.

2.3.1 Performance

We anticipate supporting 100,000 simultaneous users. This application must be lightweight and send high numbers of messages immediately for normal workload conditions. For peak workload conditions, while users are sending extremely high volumes, there may be a short wait to send messages depending on conectability and availability. In both cases, the operator will not have to wait for the message to send before closing the app. Messages featuring text only will be prioritized over messages featuring images during peak hours.

2.3.2 Reliability

In order to have the required reliability we must make sure that there won't be a user input that makes the app crash. If the user does put in a wrong input then there should be a try catch block in the code and it should guide the user to put in the correct input. For example when a user is logging in to the app and puts in the wrong email or password the app should not just stop running or crash. The app should show the user where they made the mistake and have them re-enter their email or password. Also we need to make sure that our code is clean and adjust our algorithm to make it understandable. This will allow the runtime to be faster, because if the app is too slow then users will most likely get off the app or become frustrated and delete the app. Another way we can see if the app has good reliability is we can run tests on the app to check if the app is running smoothly.

2.3.3 Availability

Flock will be a system that runs 24/7. Like most modern dating sites, your matches and data will be held on a server which will not affect your experience in case your system fails. Restarting the app will simply restart the process of retrieving data you had previously retrieved. In the instant chat messaging, for example, your unsent message will not be saved because it is not data that reached our servers. On the other hand, messages that were sent and received by the server will be displayed and saved. Therefore, the system shall allow users to restart the application after failure at the loss of any data that was not sent to the host servers.

2.3.4 Security

First of all for security we will have users login and they can decide to make a separate account or just link their twitter account with the Flock app. The user will have to remember their password and not tell others their password. Also if they forgot their password they will be able to retrieve it using their email and they will be able to make a new password. Also in order to prevent people from hacking accounts we plan on hashing user passwords. This allows users to insert their passwords without it being shown as plain text in the software, and once the software receives the password it will scramble the characters that the user put in, and have them shown in a different manner. We also plan on using symmetric or asymmetric encryption; the only downside to encryption is that it uses up more cpu.

2.3.5 Maintainability

Proper maintainability procedures will be followed when coding the software. In order to have good maintainability, one must also have good readability. As such, the names of classes, methods, variables, and other entities in the code will be understandable and show their intent without the need of a comment in order to make it easier to maintain in the future. Files will also have the same considerations to properly organize and document everything

Code will also have proper indentation/spacing to better show structure. Being able to determine easily where the code begins and ends allows for easier maintenance and less confusion when working on it in the future

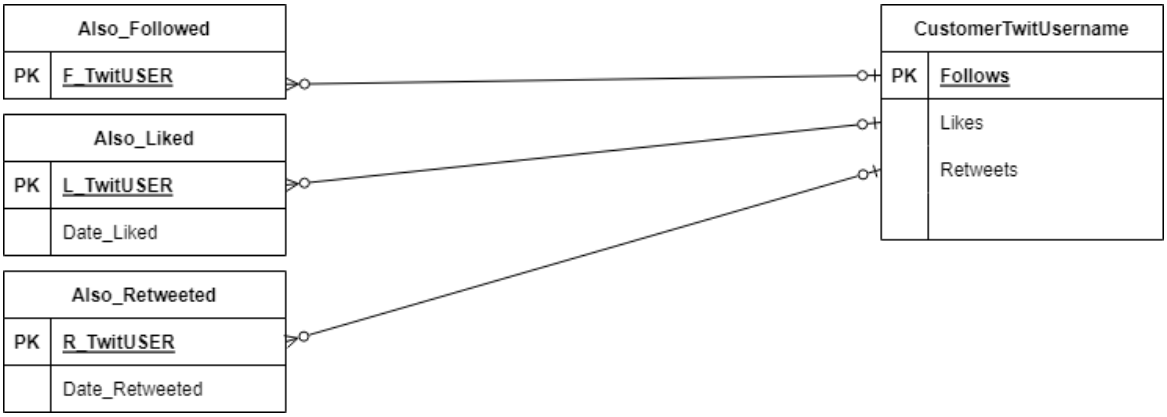
Methods will be kept to a maximum size of 50 lines. Methods should fill one purpose and nothing more to help maintainability. Having too large of a method may make it more error prone, as only so many strings can fit on a page and having too many lines in a method will make it more difficult to review.

The application will also receive periodic updates as needed, either from a phone update or in order to fix any errors found in the reviews of the mobile app store or emailed directly to the team on a support page.

2.3.6 Portability

The software will be written in Java which is extremely portable. As long as you can install the version of JDMK required on your Operating System the code will run on your device. The code itself will be written using the IDE IntelliJ thus not a language subset. The code is also not specific to any operating system but will be coded using Windows despite its ability to be used on different Operating Systems.

2.4 Database Requirements

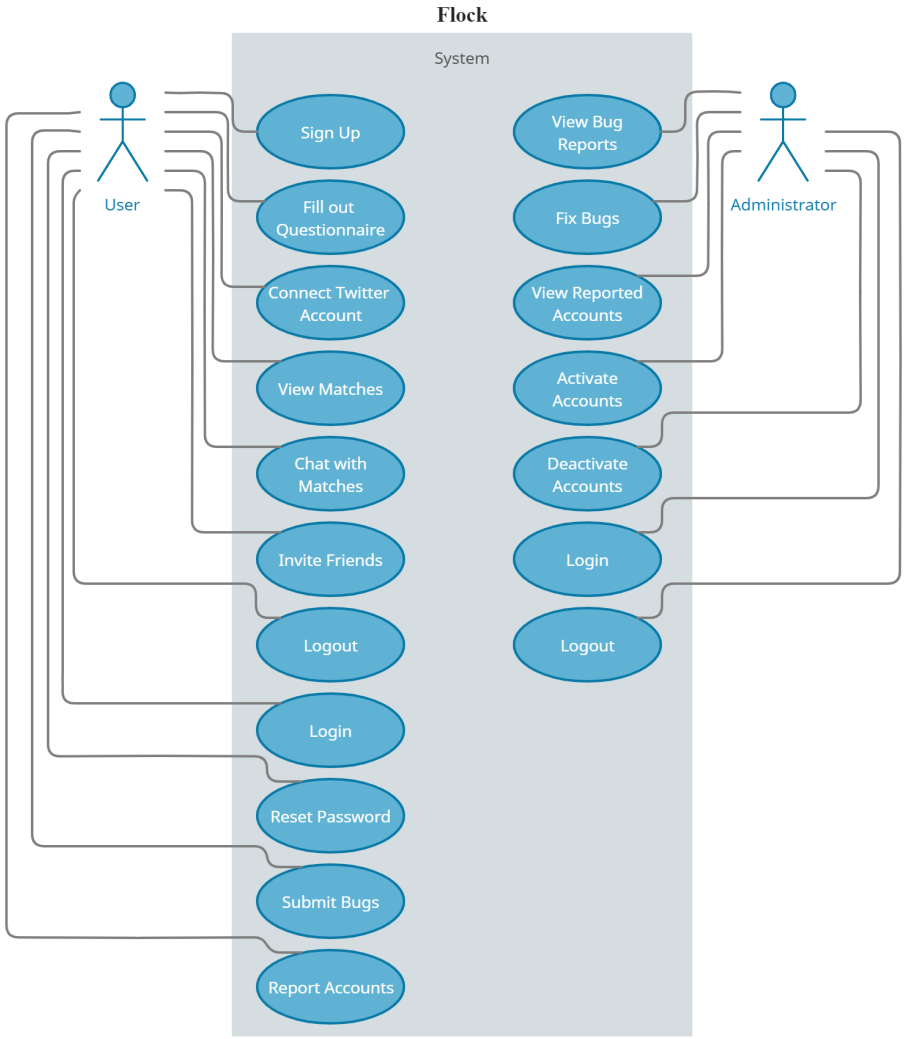


2.5 Design Constraints

- Bare minimum requirements state Android Platform Support
- Programming language restricted to (bare minimum) those that support Android, such as Java
- Will use a mobile user interface (mobileUI)
- Database will not be as prominent as Twitter's

3. Use Case Models

3.1 Use Case Diagrams



3.2 Use Case Descriptions

User - Use Cases

Sign Up Use Case

Use case must allow user to register as a new user with a new email address.

Pre-conditions for this use case are as follows: First and last name, birthdate to verify age, a valid email address which has not been previously registered with Flock, and a secure password which contains a minimum of 8 characters and one uppercase letter.

Post-conditions for this use case are as follows: after a user has registered, they may have access to the

Fill Out Questionnaire and Connect Twitter Account Use Cases.

Scenario:

1. User downloads the Flock App
2. User opens app, which takes them to the login screen
3. User selects the signup button in the bottom right corner
4. User is prompted to fill in all required fields (first and last name, birthdate, email address, and password)
5. User selects submit, and is registered for Flock

Fill Out Questionnaire Use Case

Use case must allow user to complete the questionnaire provided.

Pre-conditions for this use case are: User must be registered with Flock.

Post-conditions for this use case are: User will gain access to all other User Use Cases.

Connect Twitter Use Case

Use case must allow user to connect their Twitter account.

Pre-conditions for this use case are: User must have an existing Twitter account and an existing Flock account.

Post-conditions for this use case are: User will then have their twitter account analyzed by Flock, and once matches are made, user will be able to view and chat with any Flock matches.

Scenario:

1. User logs in to their Flock account
2. User selects the connect to Twitter button
3. User is brought to a Twitter login screen
4. User signs in to their Twitter account
5. User gives Flock permission to access their account information
6. User is redirected back to Flock
7. User is presented an account connection success screen
8. User is allowed to continue on the app

View Matches Use Case

Use case must allow user to view all of their Flock matches.

Pre-conditions for this use case are: User must have their Twitter account connected to Flock.

Post-conditions for this use case are: If user accepts a match, they will be able to then chat with them. If user denys a match, they will no longer see that profile in their matches.

Chat with Matches Use Case

Use case must allow user to chat with all of their Flock matches.

Pre-conditions for this use case are: user must have accepted at minimum one (1) match to chat with.

Post-conditions for this use case are: user will have access to the Report Account Use Case

Invite Friends Use Case

Use case must allow user to send a link to friends via SMS message to invite them to join Flock.

Pre-conditions for this use case are: user must have an active Flock account, and have completed the

provided Questionnaire.

There are no post-conditions for this use case.

Logout Use Case

Use case must allow user to logout of their Flock account.

Pre-conditions for this use case are: user must be logged in to their Flock account.

Post-conditions for this use case are: user will be logged out of their Flock account.

Login Use Case

Use case must allow user to login to their Flock account.

Pre-conditions for this use case are: user must be logged out of their Flock account and have a valid email address and password combination.

Post-conditions for this use case are: user will be logged in to their Flock account and have access to their account and information.

Reset Password Use Case

Use case must allow user to reset their Flock password.

Pre-conditions for this use case are: user must select the reset password option on the login screen, and have a valid email address registered with Flock.

Post-conditions for this use case are: a Reset Your Password email will be sent to the user's provided email and the user will have the option to go to a Reset Your Password screen.

Submit Bugs Use Case

Use case must allow user to send a Bug report to our Administration team.

Pre-conditions for this use case are: user must have selected the Bug Report button, filled out all required fields, and submitted the report.

Post-conditions for this use case are: Bug report is sent to our Administration team for review.

Report Account Use Case

Use case must allow user to send an Account report to our Administration team.

Pre-conditions for this use case are: user must have selected the Report Account button, filled out all required fields, and submitted the report.

Post-conditions for this use case are: Account report is sent to our Administration team for review.

Scenario:

1. User matches with a spam or inappropriate account
2. User selects the Report Account button in the top right corner
3. User fills out the Report fields (Reason for reporting, would you like to block this account, and additional details you would like to add)
4. User clicks the submit report button
5. Report is sent to our Administration team

Administrator - Use Cases

View Bug Reports Use Case

Use case must allow administrator to view all submitted bug reports.

Pre-conditions for this use case are: administrator must have logged in and selected the bug reports page.

Post-conditions for this use case are: the bug reports page will be visible, and selected bug reports will be able to be read.

Scenario:

1. Administrator logs in
2. Administrator selects the Bug Reports page
3. Administrator is presented with a list of open Bug Reports
4. Administrator is able to select individual reports to view and mark as complete
5. Administrator returns to Administrator Home Page

Fix Bugs Use Case

Use case must allow administrator to access accounts and upload new code to correct any bugs found.
Pre-conditions for this use case are: administrator must be logged in to the administration account.
Post-conditions for this use case are: account corrections or code corrections will be processed.

View Reported Accounts Use Case

Use case must allow administrator to view all submitted reported accounts.

Pre-conditions for this use case are: administrator must have logged in and selected the reported accounts page.

Post-conditions for this use case are: the reported accounts page will be visible, and selected reported accounts will be able to be read.

Activate Accounts Use Case

Use case must allow administrator to activate any currently deactivated accounts.
Pre-conditions for this use case are: selected account must be currently deactivated and administrator must verify that the account selected is the correct account.

Post-conditions for this use case are: selected account must be made active, and the user notified that the account is again available for use.

Deactivate Accounts Use Case

Use case must allow administrator to deactivate any currently active accounts.
Pre-conditions for this use case are: selected account must be currently active and administrator must verify that the account selected is the correct account.

Post-conditions for this use case are: selected account must be deactivated, and the user notified that the account is no longer available for use.

Logout Use Case

Use case must allow administrator to logout of their Flock account.

Pre-conditions for this use case are: administrator must be logged in to their Flock account.

Post-conditions for this use case are: administrator will be logged out of their Flock account.

Login Use Case

Use case must allow administrator to login to their Flock account.

Pre-conditions for this use case are: administrator must be logged out of their Flock account and have a valid email address and password combination.

Post-conditions for this use case are: administrator will be logged in to their Flock account and have access to the administrator account functionalities.