**A Course Project Submitted to**

**Department of Computing**

**College of Engineering and Applied Sciences**

**American University of Kuwait**

**In Partial Fulfilment of the Requirements for the**

**Software Engineering Course – CSIS 330**

**<ITweet>**

**By**

**<Mustafa A. Karkour, S00049859>**

**Course Instructor**

**Dr. Aaron Rasheed Rababaah**

**Date**

**Spring, 2020**

**TABLE OF CONTENTS**

**LIST OF TABLES**

**LIST OF FIGURES**

**ABSTRACT**

**CHAPTER I**

1. **INTRODUCTION**
   1. Background

Social networking apps have shown a significant impact on society, especially on young users. Social apps allow users around the world to share ideas, stories, memories, events, and interests with their friends, family, and people who share the same interest. Apps such as Facebook, Instagram, and Snapchat have attracted millions of users, many of whom have integrated these apps into their daily life activities. And from there I came up with ITweet. ITweet gives users the opportunity to explore other's ideas and share their posts with others. Through ITweet, users can comment on posts and get feedback on their posts.

* 1. System Overview

ITweet is a social media app similar to Twitter. It is enriched with Firebase and uses Firebase features like authentication, storage, and database. The app computes Firebase quires to validate new users, verify accounts, and store user's information and posts. Moreover, ITweet is compiled using Java and tested on a virtual emulator from Android Studio. The app is a Minimum Viable Product that aims to build social features for Android users.

* 1. Document Conventions
* FDB: Firebase Database
* FS: Firebase Storage
* FA: Firebase Authentication
  1. Intended Readership

The Software Requirements Document is intended for:

* Client engineer who can improve and add more user requirements to have a better understanding of what the customer needs.
* Developers who can review and adjust the project's system requirements and implementation. Also, to direct their effort to improve or add more features to the project.
* UI Designers who can make the project's design as beautiful and user-friendly as possible.
* Testers can use this document as a base for their testing strategy because some bugs are easier to find using the user and system requirements document.
* Project Manager who can assess the project cost and set the necessary budget for the project, anticipate the lifetime of the project and arrange any contract for future maintenance.
* End-users of this application who wish to read about the project's features.
  1. Product Scope
* The app allows users to have an account.
* The app allows users to manage their profile page.
* The app allows users to add and delete posts.
* The app allows users to add and delete comments.
* The app allows users to delete their account.
* The app is user-friendly, such that it is easy to be used by any age group.
* The app is efficient since it uses minimal amount of memory.
* The app is scalable, it can be enlarged to handle huge number of users.
* The app is maintainable, such that it can be upgraded in the future to add new features or adjust the old ones.

**CHAPTER II**

1. **SOFTWARE REQUIREMENTS SPECIFICATION – SRS**
   1. User Requirements
2. The software shall be able to create accounts for new users.
3. The software shall be able to retrieve all the user's information when the user login.
4. Every user shall have a profile page containing only his information.
5. Every user shall have a settings page, where he can edit his username, but not his email and delete his account if he wishes to.
6. The software shall view all the posts from all users in the explore page.
7. Every user shall be able to add a post.
8. Every user shall be able to delete his posts.
9. The software shall have a comment section in the post page, where every user can comment, also we will keep track of the time when the user commented.
10. Every user shall be able to delete his comments.
11. The software should allow the user to search for a particular post from the explore page.
    1. System Requirements
    2. Every user must have a non-empty name, unique email, valid password, and a user profile photo.
    3. FA will notify the user by sending a message to the software if the user is trying to sign up using other's registered email.
    4. The software must validate the user's inputs before saving them in FA and FS.
    5. The software must add the user's information to FA and FS sections before signing him up.

2.1 User's personal info and posts must be retrieved to the software once he signed in successfully.

3.1 The profile page will only display user's name, email, profile image and his cover photo.

3.2 The user has the option to have a cover photo or not, if so, his cover photo will be saved in the FS section.

3.3 The profile page contains a post section, where it only displays the user's posts.

4.1 Once the user commits to edit his username, the new username name will be updated in the FA section then in his profile page.

4.2 The user has the option to delete his account, if so, his information will be deleted from FA, FDB and FS, and therefore, the explore page will adapt the changes by deleting his posts and comments if available.

5.1 The Explore page must retrieve all the posts with their comments from FDB including the post's title, description, and the post owner's photo.

6.1 The software will allow users to add a post from the explore page, where every post has a title, description and an image, then the post's information will be saved in FDB with a unique post identifier and the post's image will be saved in FS.

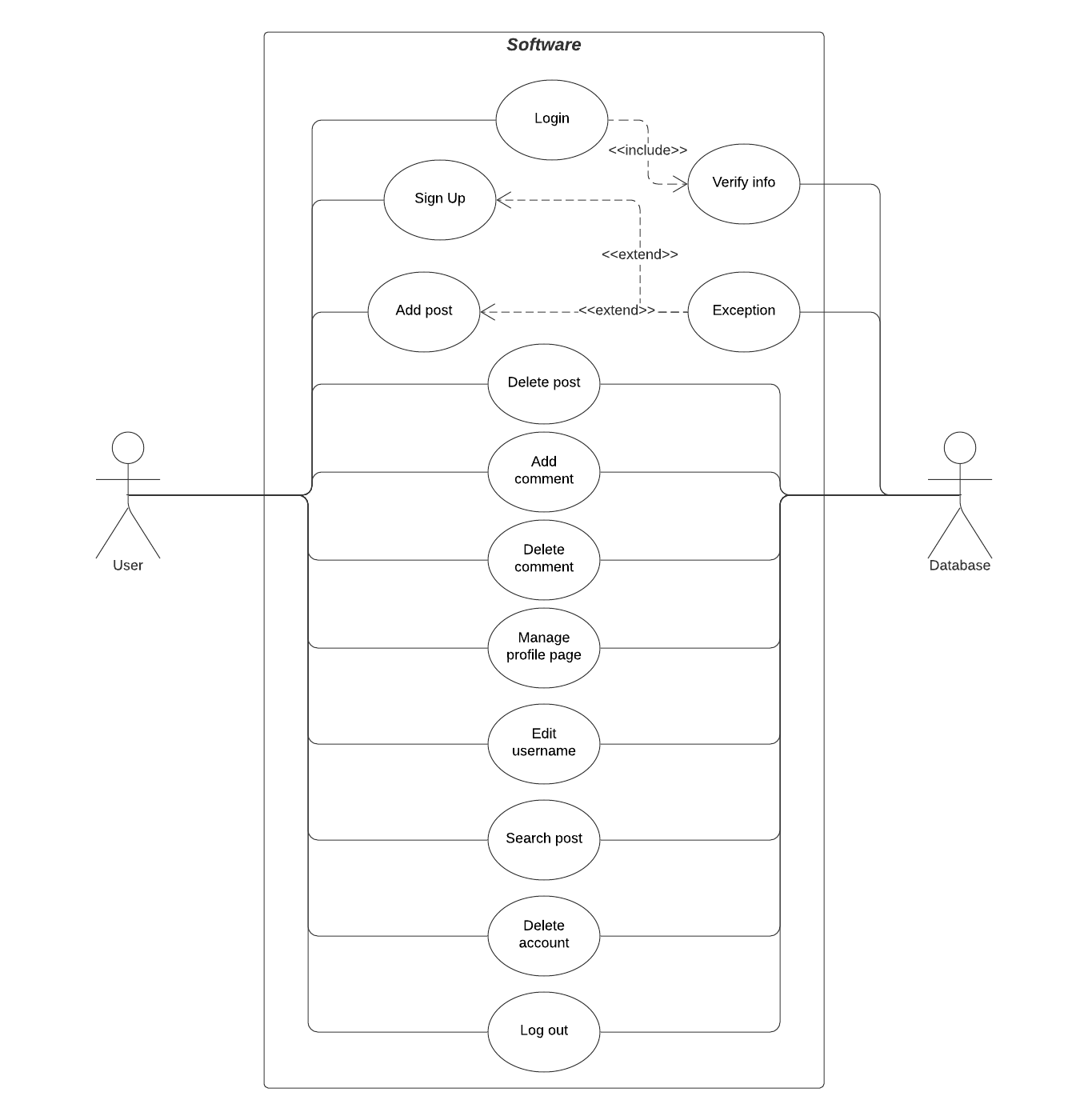
7.0 The software will check if the current user's email matches the post's owner's email, if so the user can delete his post then the post information (including its comments) will be delete from the FDB and FS otherwise, the feature is blocked.

8.1 The software will allow every user to comment on any post. Every comment will display the user's name, profile photo beside the context of the comment. Also, it will show the timestamp when the user added the comment. Every comment will be saved in the FDB with the user unique identifier.

9.1 The software will check if the current user's email matches the comment's owner's email, if so, the user can delete his comment and it will be deleted from the FDB, otherwise the feature will be blocked.

10.1 The software will allow the user to search for a particular post by inserting its title or description.

* 1. Feasibility Study
* Need: The software will help users explore other's ideas and topics with the opportunity to get feedback on their ideas.
* Urgency: The software is for entertainment use more than educational use. So, users can spend their spare time discovering what others are sharing.
* Value: The software can provide an additional income to the business through advertisement and partners.
* Integration: The system will be flexible such that it can fit and integrate with other systems to retrieve and transfer information to other systems easily.
* Availability: The software does not need any unfamiliar technology to be built or enhanced.
* Scope: The software will support some features that are in any social media app including but not limited to registering, adding comments and deleting posts. The software will not support chatting or adding friends features.
* Resources: The software doesn't need any budget to start with, but its time consuming.
  1. Non-Functional Requirements – NFR
* Security.
* Scalability.
* Usability: User-Friendly.
* Efficiency.
* Privacy.
* Maintainability.
  1. Functional Requirements Specifications – FRS
* Create User Account.
* Login.
* Add Post.
* Delete Post.
* Add Comment.
* Delete Comment.
* Manage Profile Page.
* Edit username.
* Search Post.
* Delete Account.
* Log out.
  1. Use Cases



* 1. Requirements Traceability Matrix - RTM

FUN001: Login

FUN002: Sign up

FUN003: Add post

FUN004: Delete post

FUN005: Add comment

FUN006: Delete comment

FUN007: Search post

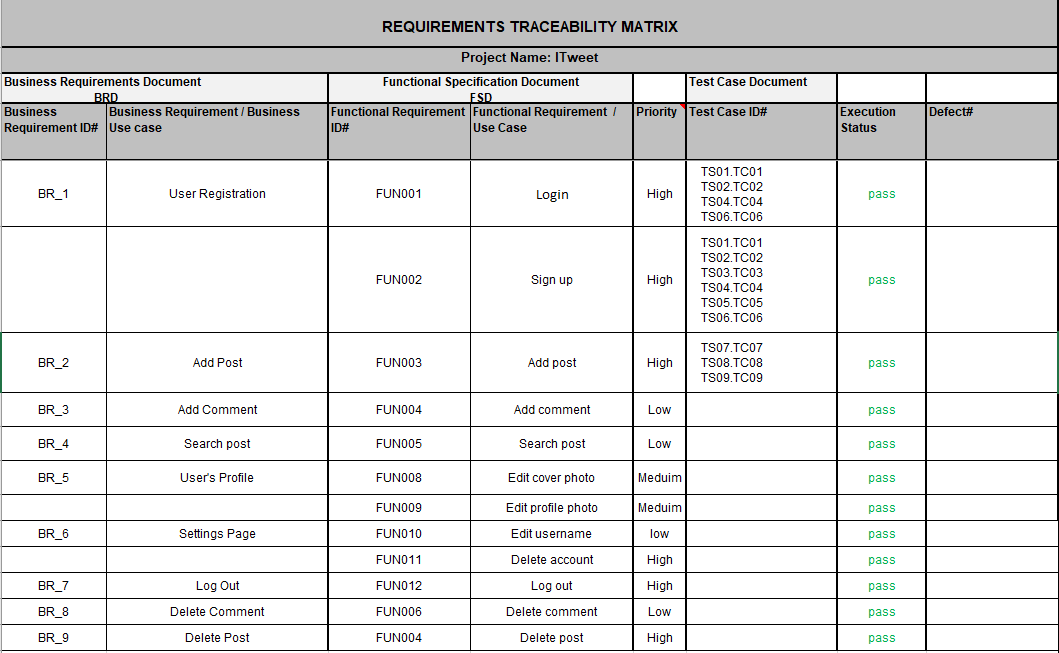
FUN008: Edit cover photo

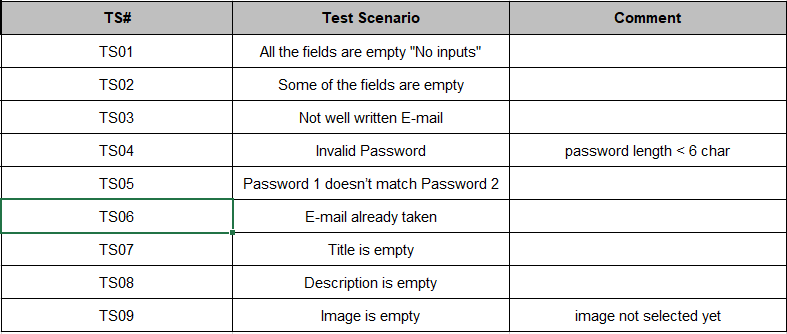
FUN009: Edit profile photo

FUN010: Edit username

FUN011: Delete account

FUN012: Log out

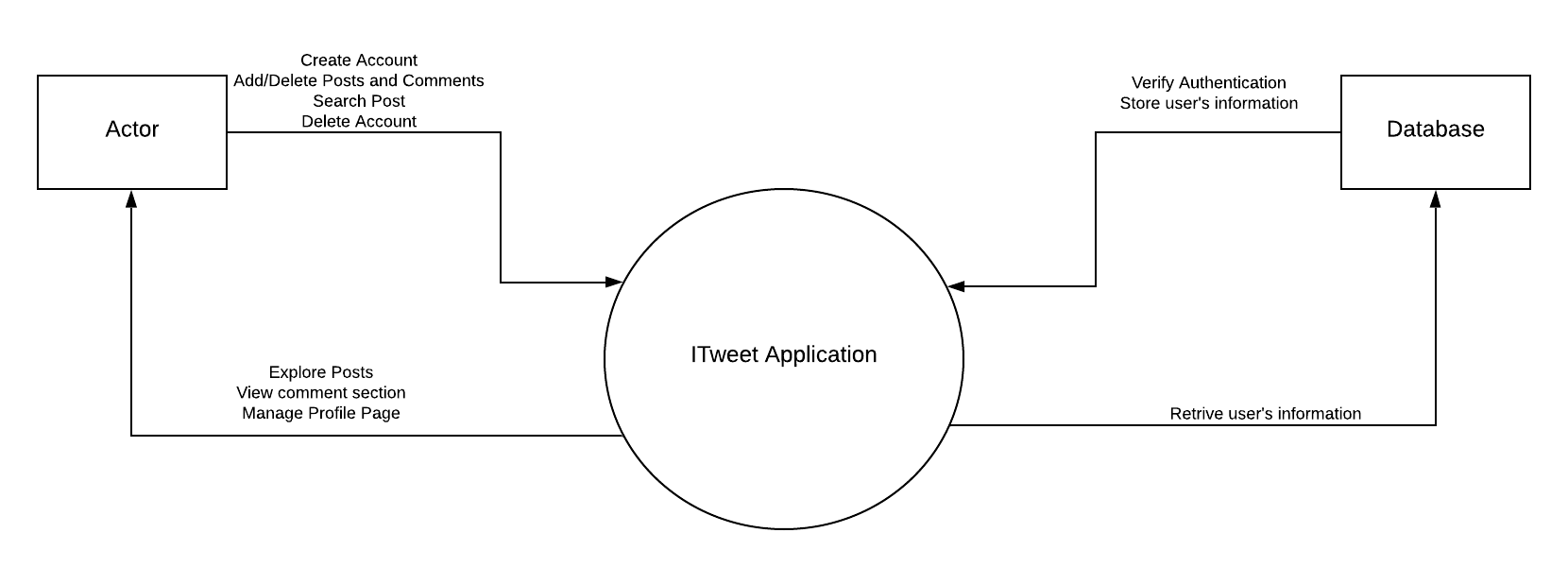




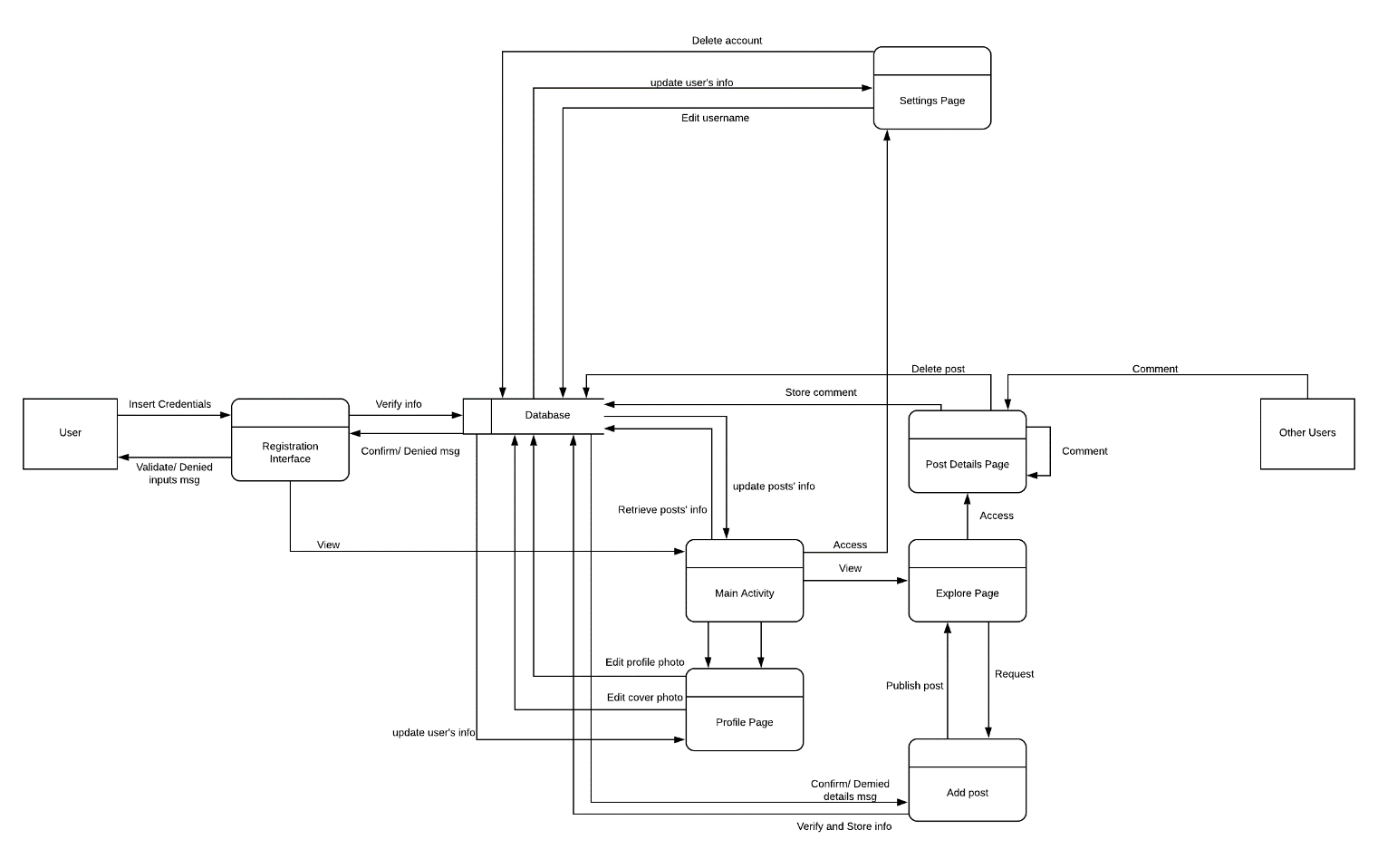
* 1. Preliminary System Architecture

**CHAPTER III**

1. **SYSTEM DESIGN**
   1. Context Architectural Model



* 1. Data Flow Model

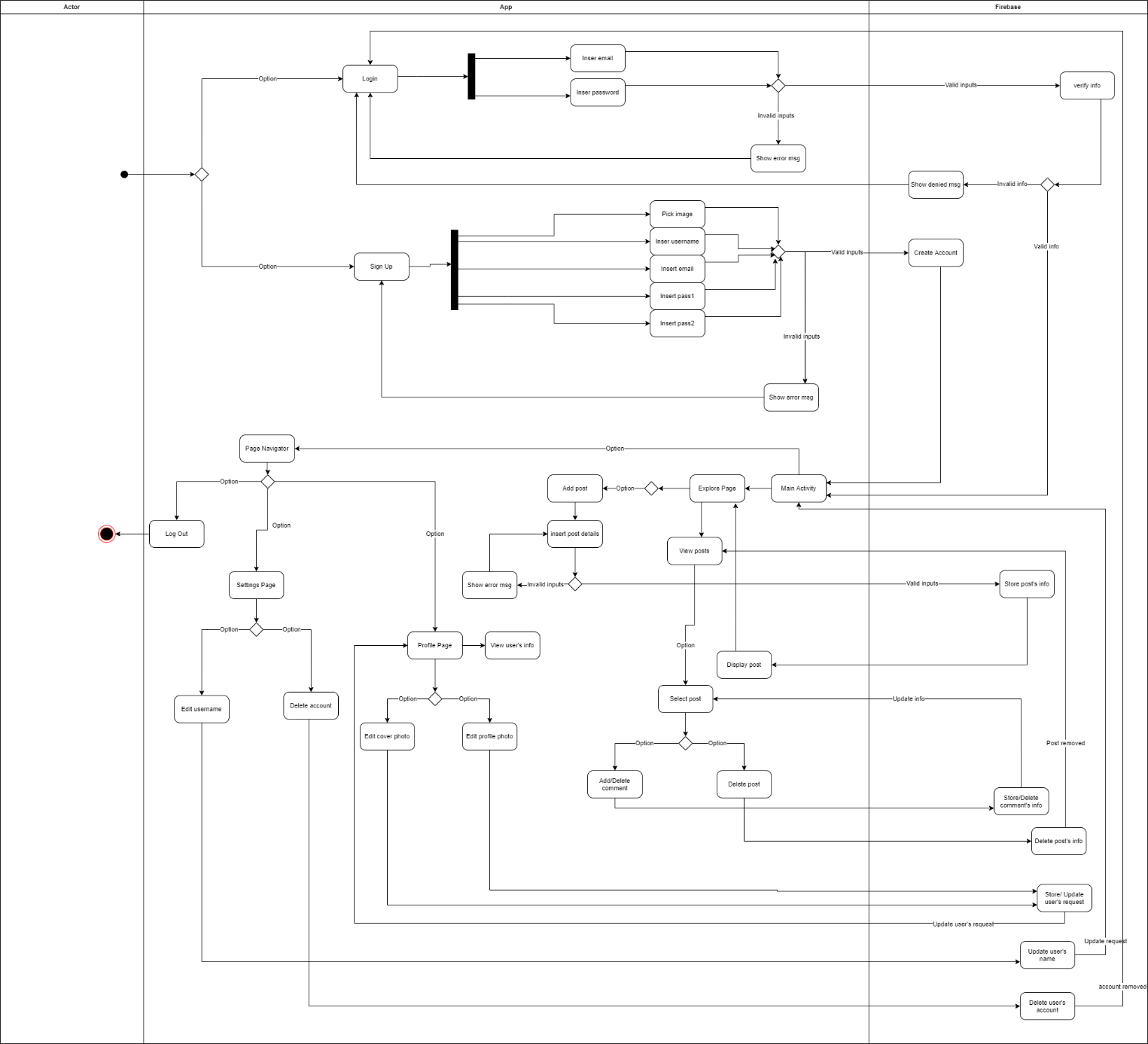


* 1. Architectural Pattern Selection

Since the user's account, post and comment data are independent and can be changed for every user, then it must be modeled in a specific way, such that the Adapter can manage these data and update them to the View. From the View, user can request changes to the Model, then the Model structures the request's data for the Adapter to reads it again. Therefore, Model-View-Controller pattern is the most stabile pattern for the app.

* 1. Detailed Design: System, Subsystem, Component and Unit Design
  2. Process Model





* 1. Use Cases Specification

|  |  |
| --- | --- |
| Use Case Name: Login |  |
| Actor(s): | User, FA |
| Description: | Allow the user to sign in if the account exist |
| Data: | Email and password |
| Stimulus/Trigger: | Splash Page or Sign up Page |
| Response: | Validate user's inputs credentials, if valid, then go to the Explore Page, else show error message |
| Comments: | The inputs must be valid Email and password |

|  |  |
| --- | --- |
| Use Case Name: Sign Up |  |
| Actor(s): | User, FA and FS |
| Description: | Allow the user to create an account |
| Data: | Name, Email, password and User photo |
| Stimulus/Trigger: | Login Page |
| Response: | User's personal info is stored in the FA section and his photo is uploaded to the FS section. Then an account created message would display if he signed up successfully, then go to the Explore Page, else show error message |
| Comments: | The account must have non-empty Name, valid Email and password, and user photo |

|  |  |
| --- | --- |
| Use Case Name: Add post |  |
| Actor(s): | User, FDB and FS |
| Description: | Allow the user to add a post |
| Data: | Title, Description, and Image |
| Stimulus/Trigger: | Explore Page |
| Response: | Post's info is added to the FDB with a unique post identifier and the post's image is uploaded to the FS section. Recent posts are displayed in the top of the Explore Page |
| Comments: | The post must have non-empty title, description and picked image. The date of the published post is saved in the FDB section and displayed in the Post Details Page |

|  |  |
| --- | --- |
| Use Case Name: Delete post |  |
| Actor(s): | User, FDB and FS |
| Description: | Allow the user to delete a post |
| Data: | Confirm request |
| Stimulus/Trigger: | Post Details Page |
| Response: | Deleting post's info from the FDB and removing the post's image from the FS section, then leaving the Post Details Page to Explore page where the post is removed |
| Comments: | The post's owner's email must match the current user's email to confirm the request |

|  |  |
| --- | --- |
| Use Case Name: Add comment |  |
| Actor(s): | User, FDB |
| Description: | Allow the user to add a comment |
| Data: | Comment's context |
| Stimulus/Trigger: | Post Details Page |
| Response: | Comment's info added to the Comment section in the FDB, then update the Comment section in the Post Details Page. Recent comments are displayed in the top of the Comment section |
| Comments: | The comment's timestamp is saved in the FDB (Comment section) and displayed beside the comment in the Post Details Page's Comment section |

|  |  |
| --- | --- |
| Use Case Name: Delete comment |  |
| Actor(s): | User, FDB |
| Description: | Allow the user to delete a comment |
| Data: | Confirm request |
| Stimulus/Trigger: | Post Details Page (Comment section) |
| Response: | Comment's info is deleted from the FDB (Comment section), then update the Comment section in the Post Details Page |
| Comments: | The comment's owner's email must match the current user's email to confirm the request |

|  |  |
| --- | --- |
| Use Case Name: Search post |  |
| Actor(s): | User |
| Description: | Allow the user to search for a particular post |
| Data: | Post's Title or Description |
| Stimulus/Trigger: | Explore Page |
| Response: | Only Display the post from the Explore Page if exist |
| Comments: |  |

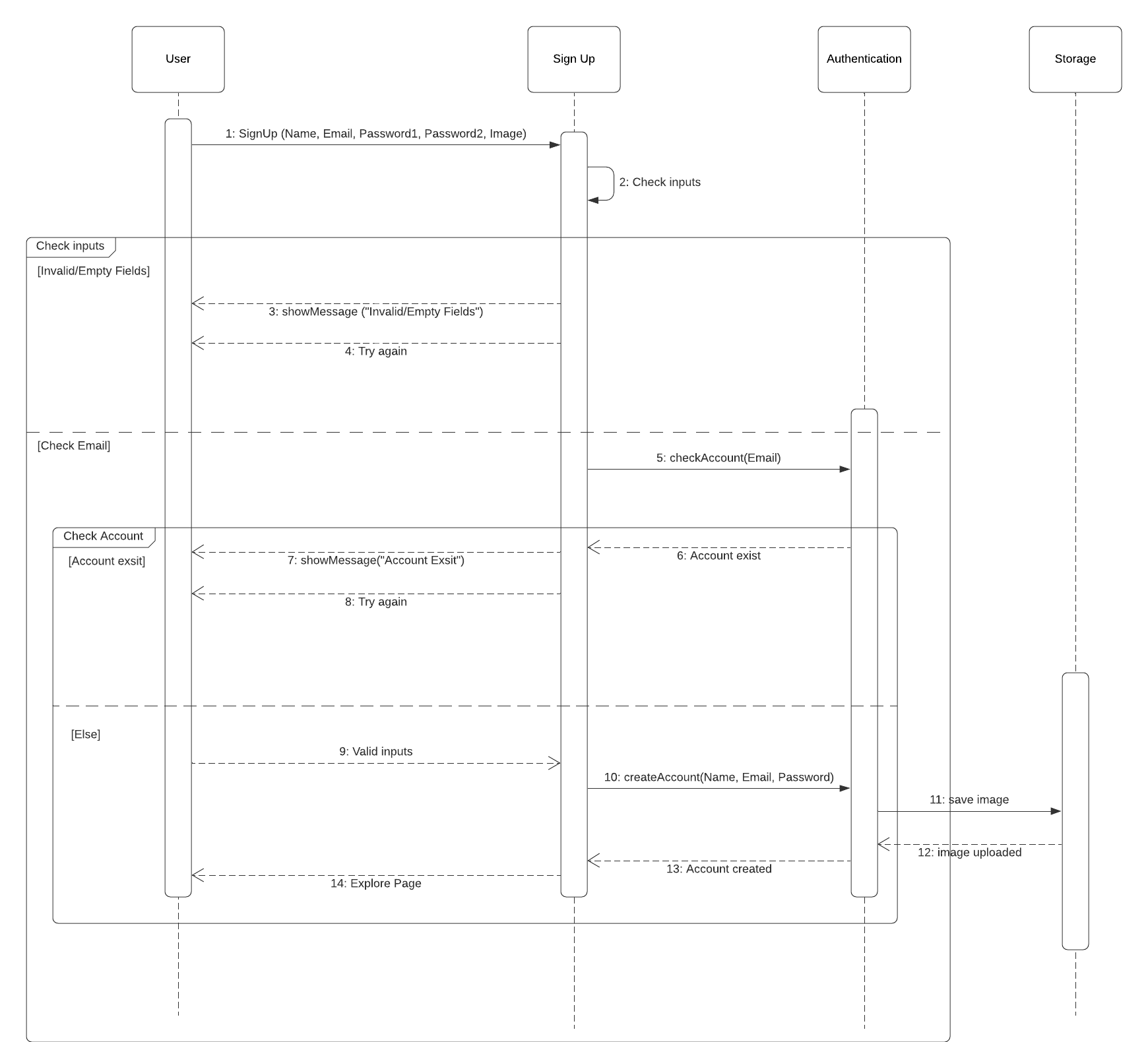
|  |  |
| --- | --- |
| Use Case Name: Edit Cover Photo |  |
| Actor(s): | User, FS |
| Description: | Allow the user to add/edit a cover photo |
| Data: | Image |
| Stimulus/Trigger: | Profile Page |
| Response: | Cover photo is uploaded to the FS section with the user's email as a unique identifier, then display the Cover photo in the Profile Page |
| Comments: | It's optional to have a cover photo |

|  |  |
| --- | --- |
| Use Case Name: Edit Profile Photo |  |
| Actor(s): | User, FDB and FS |
| Description: | Allow the user to edit his profile photo |
| Data: | Image |
| Stimulus/Trigger: | Profile Page |
| Response: | The user's photo is uploaded to the FS section with his email as a unique identifier, then his user's photo is updated in every post he published |
| Comments: |  |

* 1. Sequence Diagrams

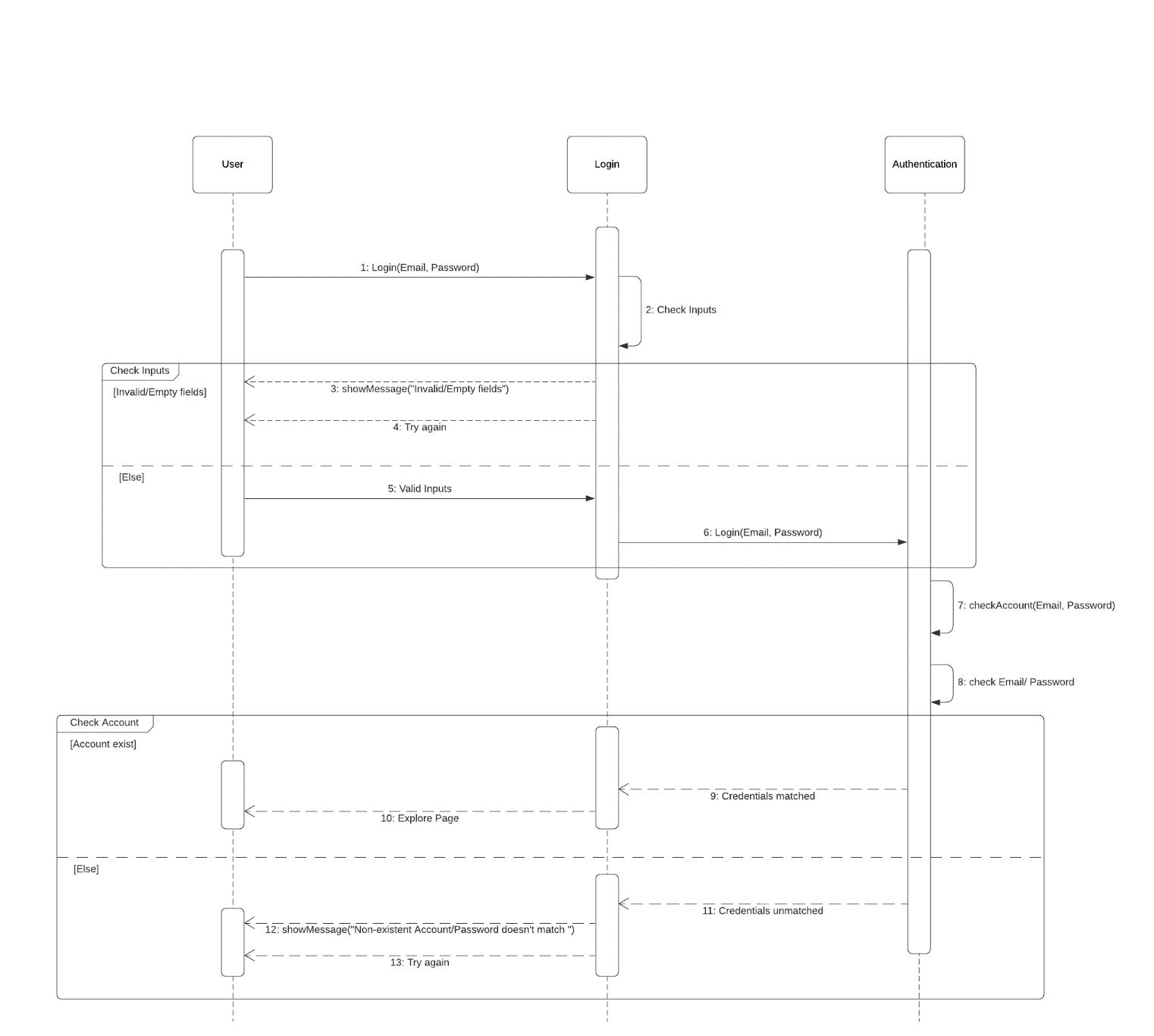
1. Sign up





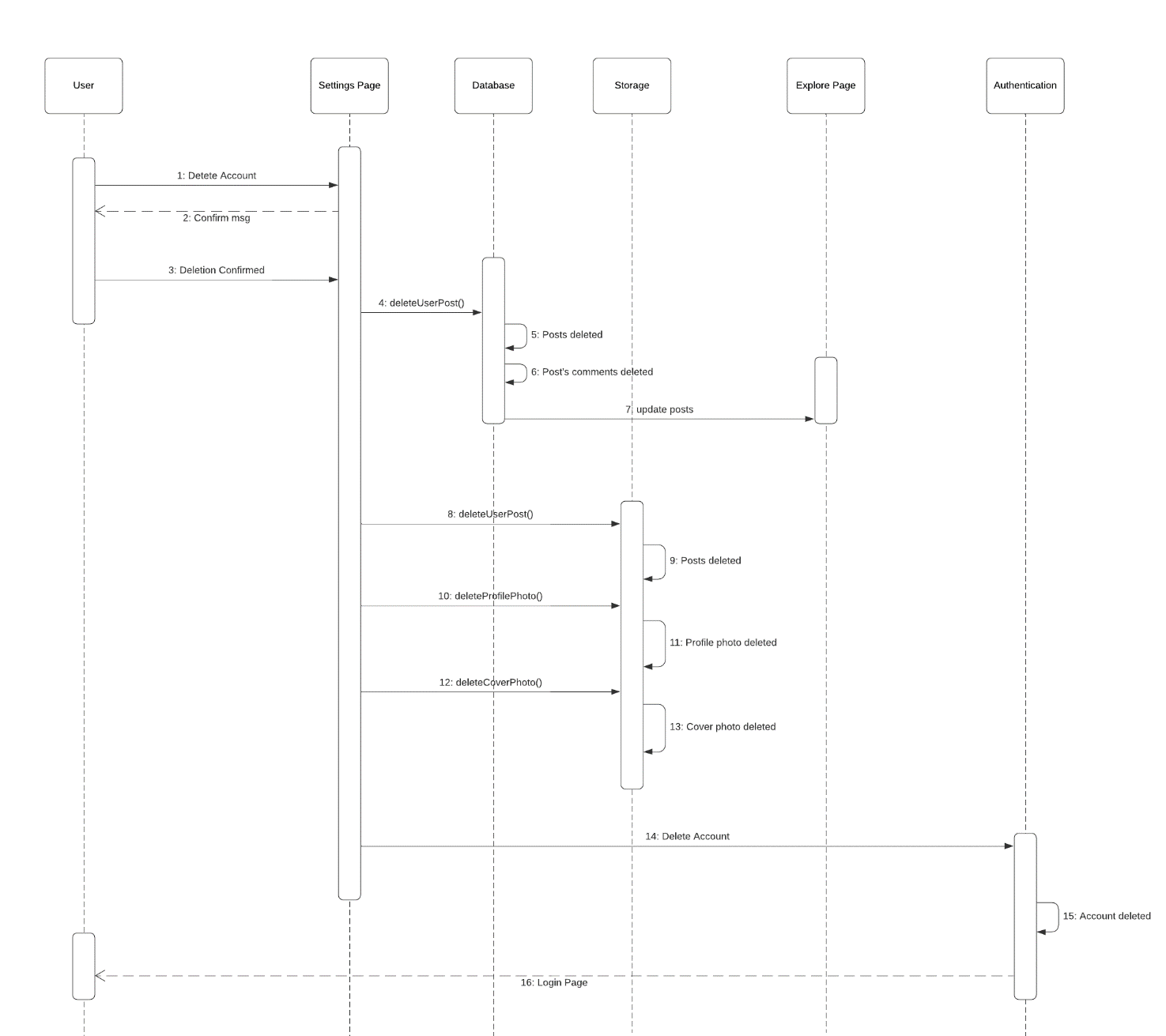
1. Log in





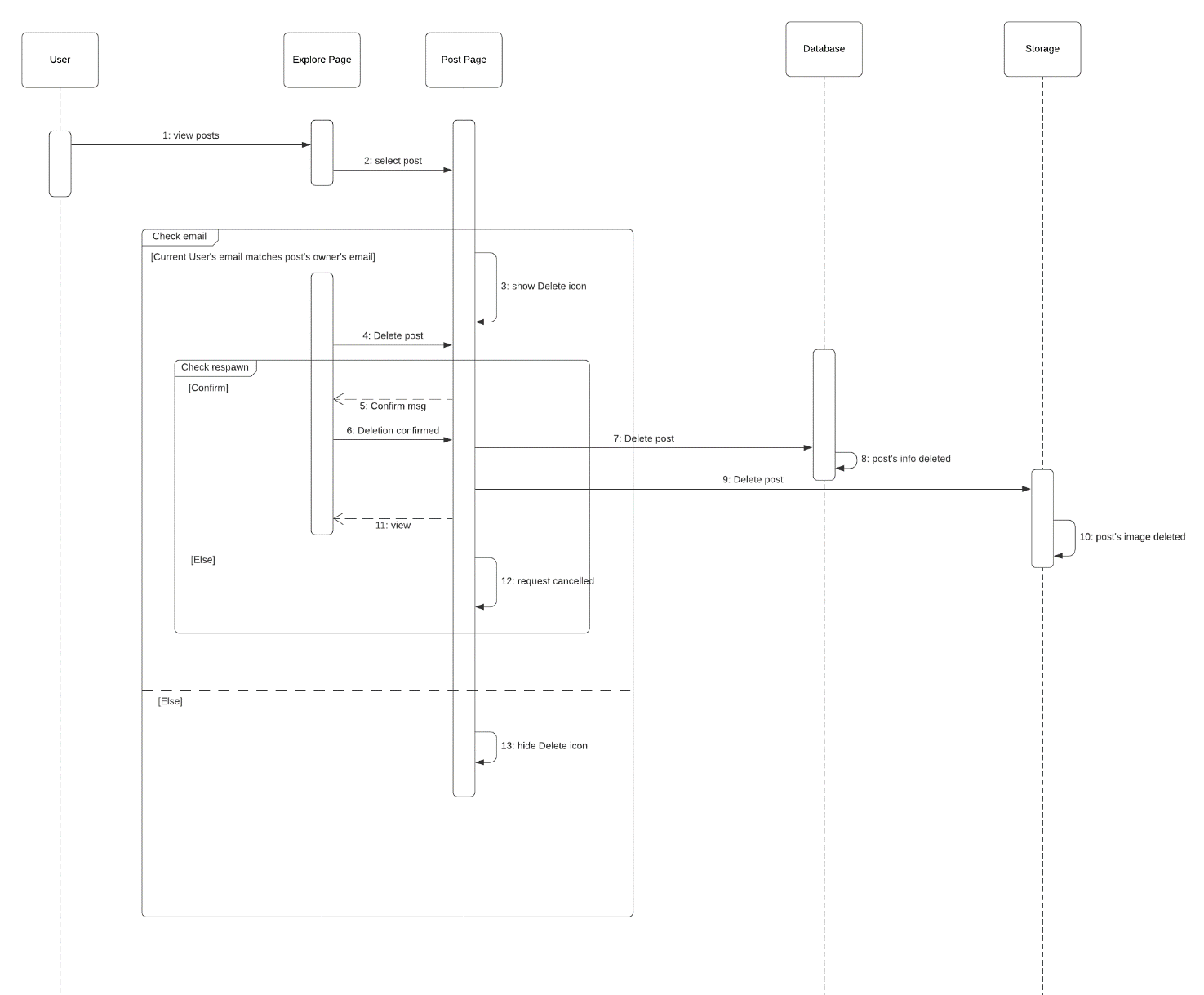
1. Delete account





1. Delete post

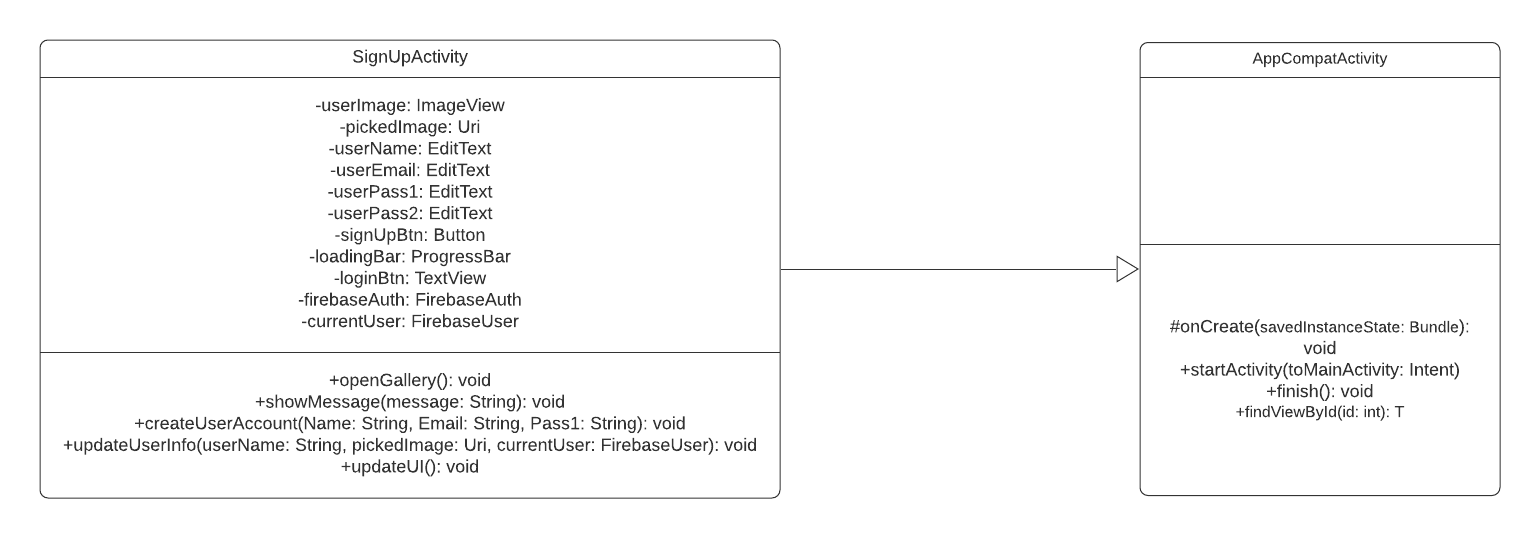




* 1. Class Structure Diagram

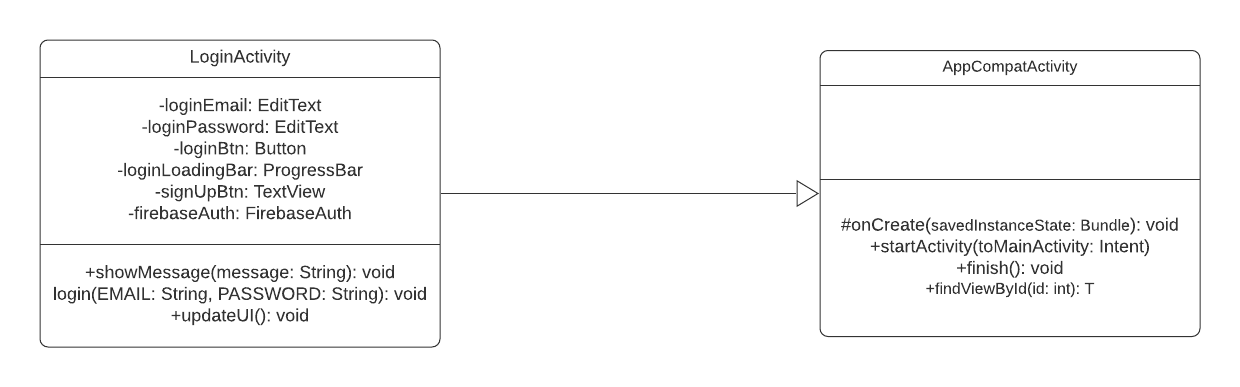
1. Sign up



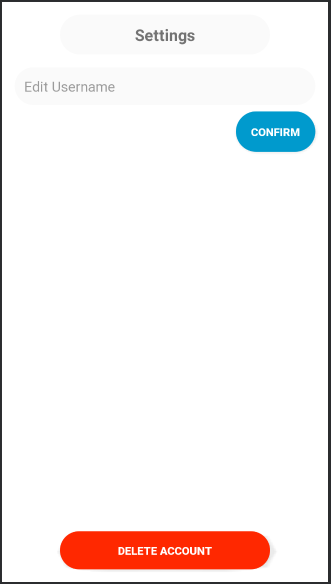
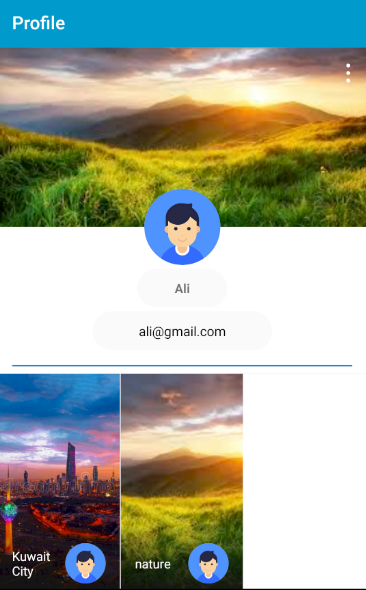
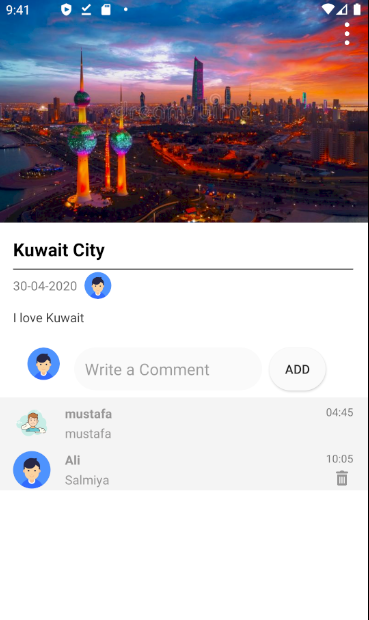
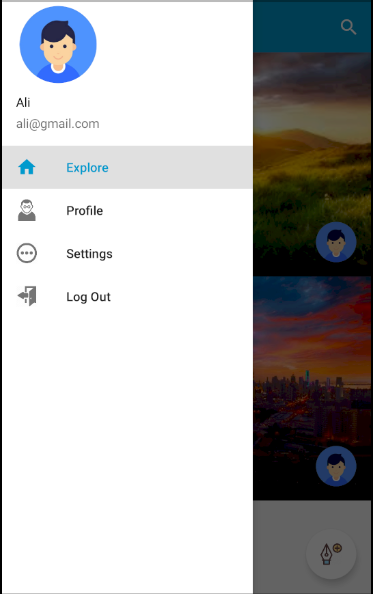
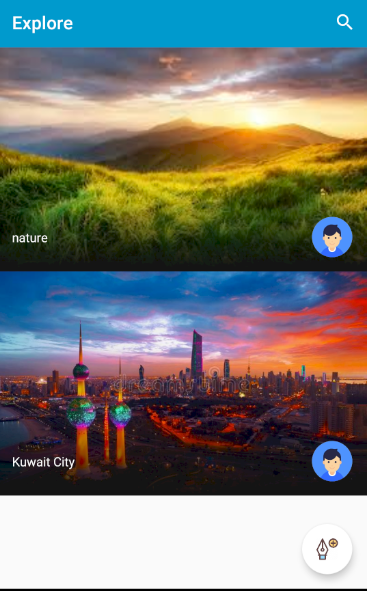
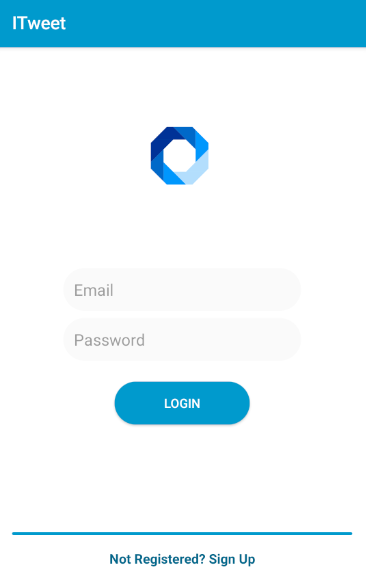
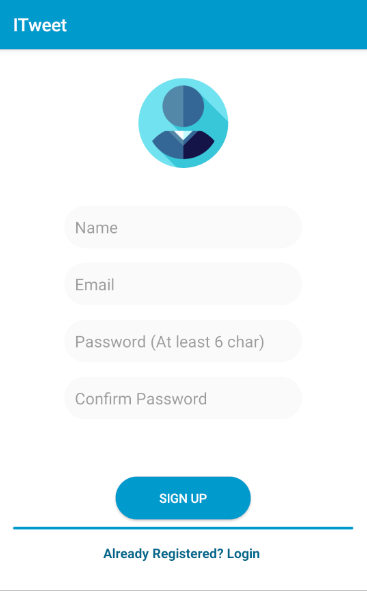


1. Log in





* 1. Finite State Machine (if Applicable)
  2. Prototype



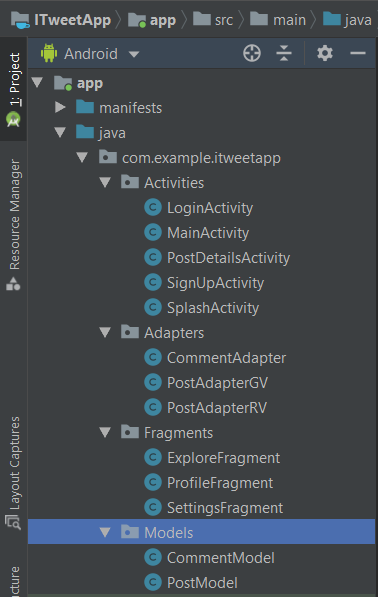
**CHAPTER IV**

1. **SYSTEM IMPLEMENTATION**
   1. Make, Buy or Lease Analysis

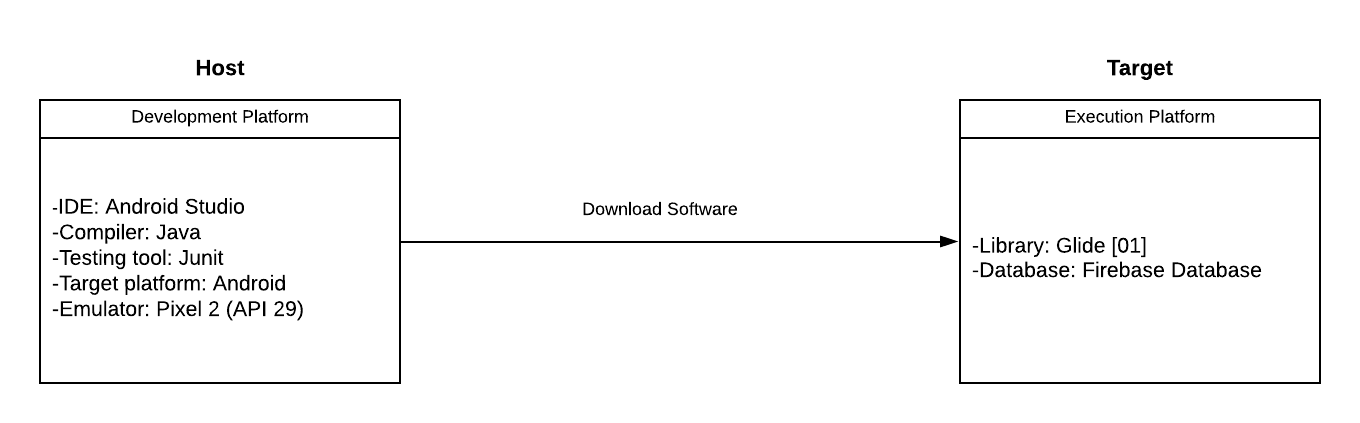
Buy: Glide Library [02] (Free)

Lease: Firebase (Free)

* 1. Coding



* 1. Configuration Management
* Sdk Version: 29
* Firebase-auth: 16.0.5
* Firebase-storage: 16.0.4
* Firebase-database: 16.0.4
* Glide: 4.11.0 [02]
* Junit: 4.12
* Navigation-fragment: 2.0.0
* Navigation-ui: 2.0.0
* Lifecycle-extensions: 2.0.0
  1. Host Target Specification
* Host
* IDE: Android Studio
* Compiler: Java
* Testing tool: Junit
* Target platform: Android
* Emulator: Pixel 2 (API 29)
* Target
  + - Library: Glide [02]
    - Database: Firebase Database



* 1. Deployment/Physical Diagram

**CHAPTER V**

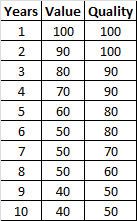
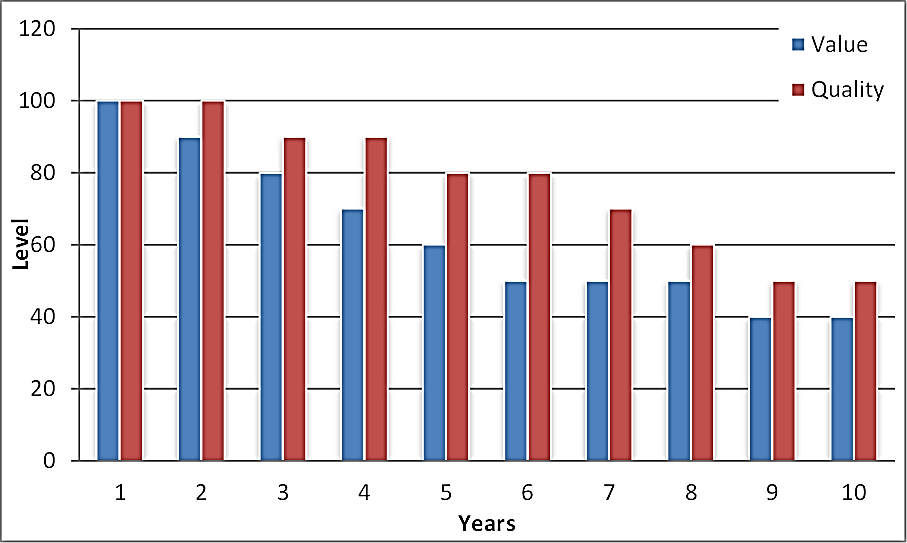
1. **SYSTEM TESTING**
   1. **System Inspection**
      1. SRS
      2. Architecture
      3. Design Models
      4. Code
   2. **System Testing**
      1. Development Testing
         1. Unit
         2. Component
         3. Subsystem
         4. System
      2. Release Testing
      3. Acceptance Testing

**CHAPTER VI**

1. **SYSTEM EVOLUTION**
   1. Anticipated System Evolution

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Change** | **Affected Comps** | **Proposed Solution** | **Cost Range** |
| 1 | GUI | Client app | Redesign GUI | KD500-2000 |
| 2 | Database | Server | Replace database with more secure one | KD2000-5000 |
| 3 | Add biometric authentication | Client app | Add a new class to handle the new hardware. All SE docs need to reflect the change. | KD3000-10000 |
| 4 | … |  |  |  |

* 1. Anticipated System Transitions till Phase-out
     1. Prediction of system quality and value for long term. Present data using a table and a column chart.

 ****

* + 1. Justify your predictions

**CHAPTER VII**

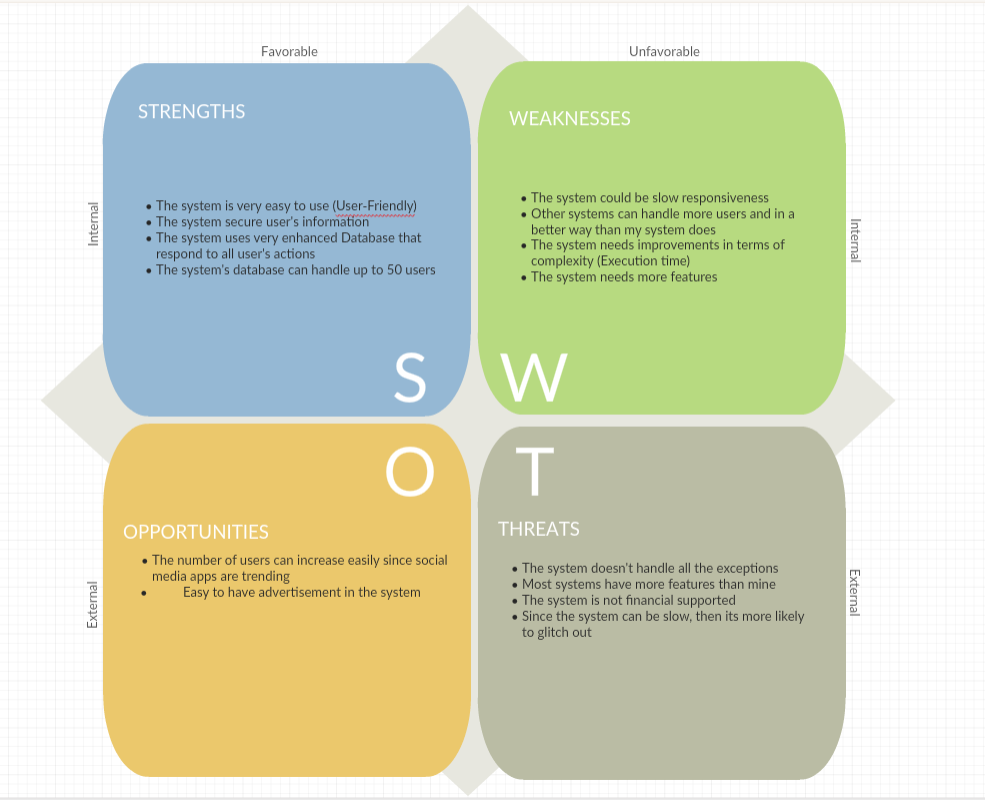
1. **SYSTEM PLANNING AND MANAGEMENT**
   1. Milestones and Deliverables Table
   2. Gantt Chart
   3. Staff Allocation Chart
   4. Activity Network
   5. CoCoMo based Costing
   6. Risks Identification
   7. Risks Assessment Matrix - RAM
   8. Root-Cause/Fishbone - Ishiwaka Diagram

**CHAPTER VIII**

1. **CONCLUSIONS**
   1. **Seven Design Smells**

How does your system stand with regard to:

* + 1. Rigidity
  + System's components are independent to a certain degree. Therefore, the system is flexible and easy to adjust.
    1. Fragility
  + The system is difficult to break.
    1. Immobility
  + The system doesn't contain parts that are reusable.
    1. Viscosity
  + Low viscosity.
    1. Needless Complexity
  + The system contains some nested queries and loops; therefore, the system has high complexity with respect to execution time.
    1. Need Repetition
  + Some functions are repeated as a source, but they could not be abstracted.
    1. Opacity
  + The code is written in a clear and expressive manner.
  1. **System Strengths and Weaknesses – SWOT**

****

* 1. **Future Work and Recommendations**

1. Implementing more features.
2. Improving the system's responsiveness.
3. Take more advantage of Firebase features.
4. Make the system runs on cross-platforms.

**CHAPTER IX**

1. **APPENDICES** 
   1. Appendix 1
   2. Appendix 2
   3. Appendix 3

**CHAPTER X**

1. **REFERENCES**

[01] Rababaah, Aaron R. (2020), Class Notes and Supplements, Software Engineering CSIS 330, Department of Computing, College of Engineering and Applied Sciences, American University of Kuwait, Spring-2020.

[02] Judd Sam, Glide, (2014), GitHub repository, <https://github.com/bumptech/glide>.