Logo, company name

Description automatically generated

Princess Sumaya University for Technology

King Hussein School for Computing Sciences

**?**   
**Social Media Platform**

**Prepared By:**

Mutlaq Alsadeed   
Yazan Farrah

**Supervised By:**

Prof. Walid Salameh

Project Submitted in partial fulfillment for the degree of Bachelor of Science in Computer Science

Summer-2022

**Declaration of Originality**

This document has been written entirely by the undersigned team members of the project. The source of every quoted text is clearly cited and there is no ambiguity in where the quoted text begins and ends. The source of any illustration, image or table that is not the work of the team members is also clearly cited. We are aware that using non-original text or material or paraphrasing or modifying it without proper citation is a violation of the university’s regulations and is subject to legal actions.

Names and Signatures of team members:

Mutlaq Alsadeed

Yazan Farrah

**Acknowledgments**

We Would like to thank Prof. Walid Salameh for being supportive and providing us with the right guidance in our project, because of his guidance we didn’t need any help as his explanations were very comprehensive and detailed in every step of the way.

And I would like to thank my partner Yazan for his endurance and ability to learn fast and keep up with my fast pace as he was able to learn Figma, and SE Diagrams in an efficient way to go along with every plan we made.

**Summary**

For us this is not a simple graduation project, for us this is a real business opportunity that we are going to pursue.

As anime fans ourselves we know firsthand how this idea can benefit the fanbase community and the content creating community both finically and socially.

Our motivation is finical and belief in the idea because of the excitement it gives to all the anime fans, on the other hand one of the main reasons we are perusing this project is that we did extensive research on every anime related application and we found out that no one came close to making this idea.

Our mythology lies in finding the mistakes in popular applications and correcting them in our application, and providing the precis and needed features in such applications and using them to our advantage in our application.

With our practical experience we believe that we can provide a world class application that will be home to many content creators and regular fans that want to keep up with the latest updates and news for animes.

This documentation will be given to anyone who will eventually come on board with us whether the person is developer, business consultant, or anyone who will be adding value to our project.

This is why we tried to make this documentation as fun to read as possible… Enjoy.

**List of Abbreviations**

**UML**: Unified Modeling Language.

**AMV**: Anime Music Video.

**Mad House**: Anime making studio based in Japan-Tokyo.

**Ufotable**: Anime making studio based in Japan-Suginami City.

**Kyoto** Animation: Anime making studio based in Japan-Uji.

**API:** Application Programming Interface**.**

**MongoDB:** NoSQL Database.

**Node.js:** Server-Side Framework.

**Express.js:** Node.js Framework.

**Nest.js:** Node.js framework that encapsulate Node + Express.

**Mongoose:** MongoDB Driver.

**Redis:** caching database used as a data structure for high performance queries.

**Cloudinary:** Cloud database for storing any kind of media.

**Figma:** Design system for sharing UI designs.

Contents

[Chapter 1  Introduction 5](#_Toc111842218)

[1.1 Overview 6](#_Toc111842219)

[1.1.1 New Era 6](#_Toc111842220)

[1.1.2 The Community 6](#_Toc111842221)

[1.2 Problem Statement 6](#_Toc111842222)

[1.2.1 Fulfillment 6](#_Toc111842223)

[1.2.2 Money Making Machines 7](#_Toc111842224)

[1.3 Related Work 7](#_Toc111842225)

[1.3.1 Amino 7](#_Toc111842226)

[1.3.2 Instagram 7](#_Toc111842227)

[1.4 Document Outline 8](#_Toc111842228)

[Chapter 2  Project Plan 8](#_Toc111842229)

[2.2 Project Deliverables 8](#_Toc111842230)

[2.2 Project Tasks 8](#_Toc111842231)

[2.2.3 Analysis 8](#_Toc111842232)

[2.2.4 Design 9](#_Toc111842233)

[2.2.5 Implementation 10](#_Toc111842234)

[2.3 Roles and Responsibilities 11](#_Toc111842235)

[2.4 Risk Assessment 12](#_Toc111842236)

[2.5 Cost Estimation 12](#_Toc111842237)

[2.6 Project Management Tools 12](#_Toc111842238)

[Chapter 3  Requirements Specification 13](#_Toc111842239)

[3.1 Stakeholders 13](#_Toc111842240)

[3.2 Platform Requirements 13](#_Toc111842241)

[3.3 Functional Requirements 13](#_Toc111842242)

[3.4 Non-Functional Requirements 14](#_Toc111842243)

[3.5 Other Requirements 14](#_Toc111842244)

[Chapter 4 System Design 14](#_Toc111842245)

[4.1 Logical Model Design 14](#_Toc111842246)

[4.2 Physical Model Design 21](#_Toc111842247)

**Table of Figures**

[Figure 1. Class Diagram. 15](#_Toc111847353)

[Figure 2. Case Diagram. 17](#_Toc111847354)

[Figure 3. Object Diagram. 18](#_Toc111847355)

[Figure 4. Component Diagram. 19](#_Toc111847356)

[Figure 5. Deployment Diagram. 20](#_Toc111847357)

[Figure 6.Sequence Diagram. 21](#_Toc111847358)

[Figure 7. Admin State Diagram. 22](#_Toc111847359)

[Figure 8. User State Diagram 23](#_Toc111847360)

[Figure 9. Database Schema. 23](#_Toc111847361)

[Figure 10. Database Admin Code. 25](#_Toc111847362)

[Figure 11. Database Observer Code. 26](#_Toc111847363)

[Figure 12 Database user code. 27](#_Toc111847364)

[Figure 13. Database Post Code. 28](#_Toc111847365)

[Figure 14 Database Comment Code. 28](#_Toc111847366)

**Table of Tables**

[Table 1. Documentation outline. 8](#_Toc111846515)

[Table 2. Analysis. 9](#_Toc111846516)

[Table 3. Design. 10](#_Toc111846517)

[Table 4 Implementation. 10](#_Toc111846518)

[Table 5. Roles and responsibilities. 11](#_Toc111846519)

[Table 6. MongoDB Cost. 12](#_Toc111846520)

[Table 7. Email Service Cost. 12](#_Toc111846521)

[Table 8. Stakeholders. 13](#_Toc111846522)

# Chapter 1  Introduction

# 1.1 Overview

We’re going to explain the basic approach for this project in a list of reasons.

# 1.1.1 New Era

Especially for this project we took our skills further and exceeded our limitations and expectations, so we specifically used technologies that are more advanced than the common technologies used to be in university graduation projects, the whole point of this approach is to relief ourselves for the long run; especially because this graduation project will be turned into a real business in the future so we made sure our work will be easy to follow up on and get on board very quickly.

# 1.1.2 The Community

As we’ve already discussed this project will directly help introverted Anime fans and AMV editors both financially and socially, as well as nowadays people on social media tend to enjoy the type of content that we plan on embracing. Judging from the researches that we have read it shows that a platform such as the one we’re planning making will be very helpful and needed for such a large fanbase (200 million and growing exponentially), and said community has no exclusive app or platform to express the real fans, the community is literally scattered around various social media platforms,

# 1.2 Problem Statement

We listed the reasons as subsections ordered by importance.

# Fulfillment

Study shows that many introverts tend to relate to Japanese media (e. g. Manga, Anime),

And this study done in many foreign countries including the U.K. and the U.S.A and recently

in Arab states of the Persian Gulf (the Gulf countries).

around 100 million people watch anime. And it is increasing exponentially day by day,

especially in this period of time.

So, we aim fulfill these people by making an application that will help them make new

connections and communicate with people from all around the world sharing similar

interests.

# 1.2.2 Money Making Machines

In recent years the most popular anime studios (e. g. Mad House, Ufotable, Kyoto Animation) have created a new marketing scheme that is adaptable to new trends which is whenever that they plan to animate new Mangas, they work on a short exciting trailer and pay anime related pages on popular social medias to post a meme (Funny picture / video) or an AMV, now these pages usually relay on making the AMV on a computer with paid software for editing and spending around (1-5) hours on making the AMV even more exciting

# 1.3 Related Work

# 1.3.1 Amino

Amino has one of the worst UI designs that we have ever seen, which made the whole application very unappealing to any user, which clearly explains why the application has a limited following, however Amino’s vision was to create a community-based application mimicking reddit, although Amino was successful only in a handful of communities and to be precise it found success in Anime related communities

Now the question is **why would anyone leave Amino and come to our application?**

The answer is very simple they won’t…

From a user’s perspective no one cares if they have two applications similar to each other, however they definitely will have a favorite application, and this is our goal.

**What would make our application better?**

If anyone tries to use Amino, they will have a difficult time navigating the application, since it is a teasing task to learn how to use such an application (Amino) so we plan on fixing the problem.

Amino has a circular design, which means literally you can loop inside the application which makes it very confusing for the user.

It is very obvious that you will need to provide an easy and smooth chatting between the community members, which in Amino it’s a terrible experience.

**Fixing the problem**

Two university students cannot possibly outclass a whole company with many resources and tons of employees, however a bigger company can!

We plan on using Instagram’s methods for implementing overlapping features.

# 1.3.2 Instagram

With over 12 years of experience and hundreds of millions of dollars sunk into making the perfect UI/UX design, Instagram successfully changed the way people think when using applications, for example you cannot possibly have a hard time navigating through the application or difficulty using a feature, so we plan on mimicking the way Instagram implement features to make the best possible Anime fanbase application.

# Document Outline

Table 1. Documentation outline.

|  |  |
| --- | --- |
| Chapter title | Description |
| Chapter 1: Introduction | Giving a brief idea about our application in general. |
| Chapter 2: Project plan | Here we describe in details how we did the required tasks and how we divided the work between both of us (Mutlaq & Yazan). |
| Chapter 3: Requirements Specification | Talking about the stakeholders and discussing the functional and non-functional requirements, as well as hardware specifications. |
| Chapter 4: System Design | UML diagrams to describe the high-level and low-level design of this project002E |

# Chapter 2  Project Plan

# 2.2 Project Deliverables

List and describe the deliverables of the system. Examples of deliverables include source code, documentation files, executables, datasets, databases, etc.

# 2.2 Project Tasks

# 2.2.3 Analysis

Table 2. Analysis.

|  |  |  |
| --- | --- | --- |
| **Task name** | **Details** | **Assignation** |
| Scalability/  Modularity/  Clean Code | We went through the trouble of learning the most scalable and modular frameworks to ensure our code is always top-level, and we read a book about clean code just to ensure that our work will be top tier | Mutlaq & Yazan |
| Programming languages | Using Dart for mobile development with Flutter framework as well as JavaScript | Yazan & Mutlaq |
| Frameworks | Flutter is an open-source framework by Google for building beautiful, natively compiled, multi-platform applications from a single codebase. | Yazan |
| Platforms | Aws, Firebase | Mutlaq & Yazan |
| Research | We have been studying the Anime community for more than 5 years as well as both of us are Anime fans. | Mutlaq & Yazan |
| Security | We’re not security specialists, however no mid-tier attacker will be able to damage our systems HOPEFULLY.  We’ll block Mongo injection, CSRF attacks, DDoS attacks and Helmet attacks. | Mutlaq & Yazan |
| Performance | We’ll be making sure to make our code as clean and secure as possible with following the best standards, as well as best practices. | Mutlaq & Yazan |

# 2.2.4 Design

Table 3. Design.

|  |  |  |
| --- | --- | --- |
| **Task name** | **Details** | **Assignation** |
| Communication | We have been using Discord application for daily communicating and brainstorming together since we can share files/screen and many more! | Mutlaq & Yazan |
| Database design | Following the eight rules of thumb in MongoDB, we’ll be creating the most efficient and fit database for our systems. | Mutlaq |
| UI/UX | Using Figma as well as Adobe XD designing tools for making the best possible user interface/experience. | Yazan |
| Prototypes | Sharing UI prototypes together and ideas to make sure we’ll be developing and making the best possible experience for our users. | Yazan |

# 2.2.5 Implementation

Table 4 Implementation.

|  |  |  |
| --- | --- | --- |
| **Task name** | **Details** | **Assignation** |
| Mobile Application | Writing our mobile application code with dart language using Flutter framework since it offers us one code for both iOS and Android operating systems. | Yazan |
| Web application | Using Angular modularity and scalability to optimize an admin panel to fit for anytime and place. | Mutlaq |
| API endpoints | Node + express + Nest server fit to handle anytime of request with high efficiency. | Mutlaq |
| Cloud data/media database | We’ll be using Cloudinary a well MongoDB Atlas for it’s easy and high-performance dataflow. | Mutlaq |
| Version control configurations | We will be simulating a real-life company work flow to ensure that any type of developer will be having an easy time migrating to our systems. | Mutlaq |
| Stress testing | Pushing the server to its limits to know exactly the point at which the server crashes and optimizing the application to work with the limit. | Mutlaq |
| Unit testing | Knowing how every input/output data in the systems are behaving and bug finding. | Mutlaq |

# 2.3 Roles and Responsibilities

Clarification: each role/responsibility isn’t exclusive to a one member; meaning that we will be working hand-to-hand on everything, however we’ll be classifying roles/responsibilities based on the primary member

Table 5. Roles and responsibilities.

|  |  |
| --- | --- |
| Project Management | Mutlaq Alsadeed |
| API Planning and Building | Mutlaq Alsadeed |
| Technology Research | Mutlaq Alsadeed |
| Admin panel design and implementation | Mutlaq Alsadeed |
| Backend Design | Mutlaq Alsadeed |
| Database Design | Mutlaq Alsadeed |
| AWS Functionality | Mutlaq Alsadeed |
| Research | Yazan Farrah |
| UI/UX Research | Yazan Farrah |
| Mobile App Design and Implementation | Yazan Farrah |
| Firebase Functionality | Yazan Farrah |
| Feature Planning | Yazan Farrah |
| State Management | Yazan Farrah |

**Project Management:** Dissecting the development process into comprehensive stages and injecting real production knowledge into the application.

**API Planning and Building:** Deciding which architectural concept to apply on the API’s and deciding endpoint functionality.

**Technology Research:** Deciding which technologies/framework best suited for the application

**Admin panel design and implementation:** Designing the admin panel in terms of UI/UX and functionality and implementing it using Angular 14

**Backend Design:** Configuring the API endpoints to be used in the mobile and web application.

**Database Design:** Choosing which DBMS to use and designing the schema.

**AWS Functionality:** Depending on the performance of the whole project we might resort to using AWS for more speed and efficiency.

**Research:** We used various resources for our researches that we linked down in the references section and we put our focus on the statistics, studying what the anime community likes, understanding the shared interests between the Anime fans.

**UI/UX Research:** Knowing what colors to use, widgets, animation so we present them in a comfortable way for our users making sure they enjoy spending more and more time on our application.

**Mobile App Design and Implementation:** Here we used an open-source UI software development kit (Flutter) since we can develop cross platform applications for Android, iOS and the web from a single codebase.

**Firebase Functionality:** Using firebase functionality to fully to optimize some features that will require heavy lifting on the server side and make performance much higher, such as push notifications.

**Feature Planning:** Studying and searching for the best features to use in our application is a must! So here we’ll be making sure to add features helping the user to enjoy our application in an easy as well as entertaining ways.

**State Management:** The state management is one of the most popular and necessary processes in the lifecycle of an application, so it can help make the code easy to test and reusable by separating presentation from business logic, the importance of state management that at any given time you always where the data is flowing.

# 2.4 Risk Assessment

When it comes to risks, this project will have 0 apparent risks, if one was to look at any social media platform and assist their risks the first thing that will come to mind is that what if after all the trouble no one used our application?, well for us personally we choose development environments that have free tiers, to enable us of making a fully functioning systems, however that capacity will be very small obviously, no free tier will store larger amount of data but it is enough to make a small number of users with full media liberty.

The usual risk for these apps is that the development will const a lot.

If we were to request a company to make a system of this scale, the minimum number we would hear is 30K JD, the amount of time and resources that a company will allocate developers to is reasonable for this scale, however we already have the experience needed to make these systems ourselves so we eliminated the development risk.

The only thing that remains is what if users don’t participate in our app?

And there is definitely no guarantee that anyone will use it…

However as we already said this project didn’t come up as a graduation project idea, Mutlaq came up with the idea two years ago, and created an account on Instagram (@Dr.Senkuo) to test the possibilities of people being open to the idea, after reaching a certain amount of followers the conclusions were:-

1. Anime Studios pay content creators to promote new animes or seasons
2. Anime fans listen to content creators
3. Content pages ask for a small amount of money for promotion (5-50$ USD)
4. Interaction is more compared to other genres

So if we were to promote our application through these pages we will be sure at least a small amount of people will be interested enough to download the application, after that our hope lies in the quality of content and application and start a word of mouth movement to kickstart our project.

# 2.5 Cost Estimation

MongoDB:

Table 6. MongoDB Cost.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Cluster Tier** | **Storage** | **Ram** | **vCPUs** | **Base Price** |
| M50 | 160 GB | 32 | 8 | $2.00/hr |

Email Service:

Table 7. Email Service Cost.

|  |  |
| --- | --- |
| **Nodemailer** | **SendGrid** |
| $820/year | $30/10K emails |

Hosting:

Web /API Hosting: $10/month

Play Store (Android) Mobile Application Hosting: $25 (Lifetime)

App Store (iOS) Mobile Application Hosting: $100/year

Cloudinary Hosting: 90/month

# 2.6 Project Management Tools

Version control system: Git/GitHub, Communication: Discord.

# Chapter 3  Requirements Specification

# 3.1 Stakeholders

Table 8. Stakeholders.

|  |  |  |  |
| --- | --- | --- | --- |
| **Admin** | **AMV editors** | **Anime fans** | **Observer** |
| The admin will be able to delete/ suspend user’s actions etc... | Editors who would be able to cut anime videos, combine clips, custom video, etc... | Add or interact with posts, chat with users, etc... | A person who would be monitoring whether there are inappropriate content/ posts and can delete them or ban users |

# 3.2 Platform Requirements

* **Client-Side Mobile/Web Applications**

Table 9. Platform Requirements.

|  |  |  |
| --- | --- | --- |
| Requirement ID | Description | Priority |
| PRCS1 | Smart Phone / Personal Computer | Essential |
| PRCS2 | Internet Connection Access | Essential |

# 3.3 Functional Requirements

Table 10. Functional Requirements.

|  |  |
| --- | --- |
| ID | Requirement |
| FR1 | Authentications |
| FR2 | Authorization |
| FR3 | Profiles / with edit functionality |
| FR4 | CRUD Posts |
| FR5 | CRUD Comments |
| FR6 | Chatting |
| FR7 | Post Interaction |
| FR8 | Comment Interaction |
| FR9 | Account Search |
| FR10 | Notifications |
| FR11 | Observer Control |
| FR12 | Admin Control |

* **Authentications [FR1]:** Implement registration and logging in for Admin/User/Observer, with hashed password and ability to reset or change passwords.
* **Authorization [FR2]:** Build an access control system inside our server to give access to actions only allowed for rightful owners.
* **Profiles [FR3]:** Show a profile page for every user to edit any personal information and control whether other people can see it or not.
* **Post [FR4]:** Users have CRUD functionality over their posts
* **Comment [FR5]:** Users have CRUD functionality over their comments
* **Chatting [FR6]:** Realtime chat between users implemented with web sockets
* **Post Interaction [FR7]:** Ability to like / comment / share / send to friend on every post
* **Comment Interaction [FR8]:** Ability to like / comment on every comment
* **Account Search [FR9]:** Search for friends or popular accounts
* **Notifications [FR10]:** Push notifications for Post interaction, messages, and new updates
* **Observer Control [FR11]:** Observers have the ability to delete any post or comment that is inappropriate and block the user who posted said content
* **Admin Control [FR12]:** Admins add/delete Observers and Admins and block Users

# 3.4 Non-Functional Requirements

Table 11. Non-Functional Requirements.

|  |  |
| --- | --- |
| ID | requirement |
| NFR1 | Security |
| NFR2 | accessibility |
| NFR3 | Scalability |
| NFR4 | Modularity / Reusability |
| NFR5 | Reliability |

* Security [NFR1]: Steps we are going to do to make sure we achieve the best secure system we can

1. Avoid using vulnerable third-party packages during development to avoid threats from attackers exploiting said vulnerabilities
2. Protect our systems from: - MongoDB Injection, Cross Site Scripting, CSRF Attacks, DDoS Attacks, Token forgery, and Man in The Middle
3. Resetting a password requires access from the email you entered on registration, changing a password requires knowledge of old password

* Accessibility [NFR2]: YAZAN
* Scalability [NFR3]: As we said before we plan on taking this project beyond a graduation project so we are going to make sure we make this project as scalable as possible by only picking LTS technologies and trending programming techniques and making every codebase dynamic and fit for all possible scenarios
* Modularity and Reusability [NFR4]: Angular, Node Express Nest, Flutter, MongoDB, AWS, Firebase, and Redis are all technologies that fully support a modular way of design and implementation and we plan on maximizing their fullest potential, every codebase will be reusable and modular
* Reliability [NFR5]: We will be preforming Unit Testing and Stress Testing our servers and I/O streams to ensure at any given time our system will be up and running smoothly

# 3.5 Other Requirements

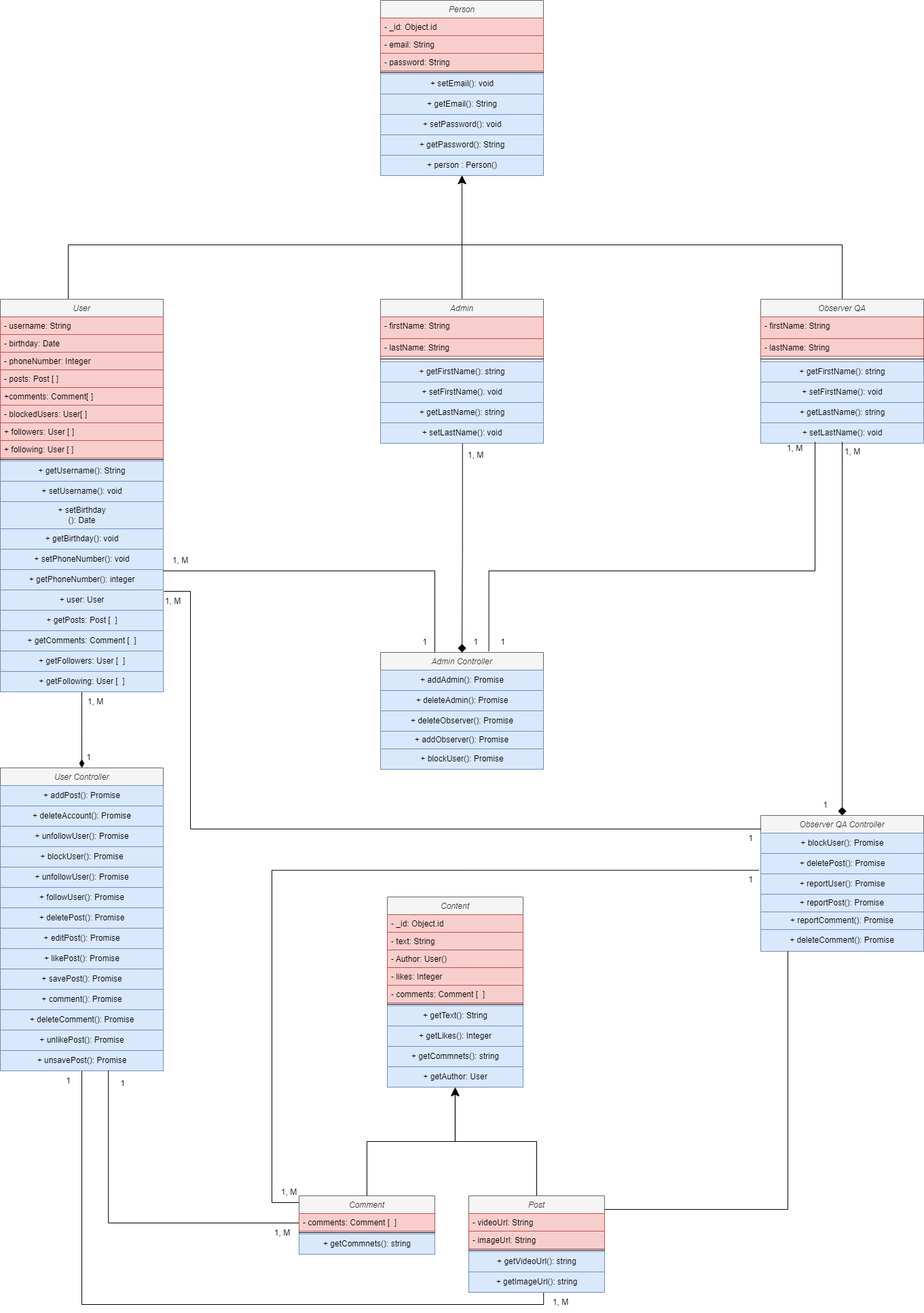
Include here any requirements that may not directly fall under any of the sections before. Examples include restrictions on which APIs can be used, data transmission protocols, data storage formats, etc.

# Chapter 4 System Design

# 4.1 Logical Model Design

* Class Diagrams

Figure 1. Class Diagram.



User, Admin, and Observer have email, password, and \_id in common and

they inherit these attributes from Person Class

Each Class's business logic is placed into Class Controller and the class is aggregated with it's Controller

Admin controller has Association relation with Observer because Admin Controller affect Observer class by removing or adding an Observer

Admin controller has Association relation with User because Admin Controller affect User class by removing or adding a User

Observer controller has Association relation with Post because Observer Controller affect Post class by removing or flagging a Post

Observer controller has Association relation with Comment because Observer Controller affect Comment class by removing or flagging a Comment

Observer controller has Association relation with User because Observer Controller affect User class by blocking a User

User controller has Association relation with Post because User Controller affect Post class by removing or adding a Post

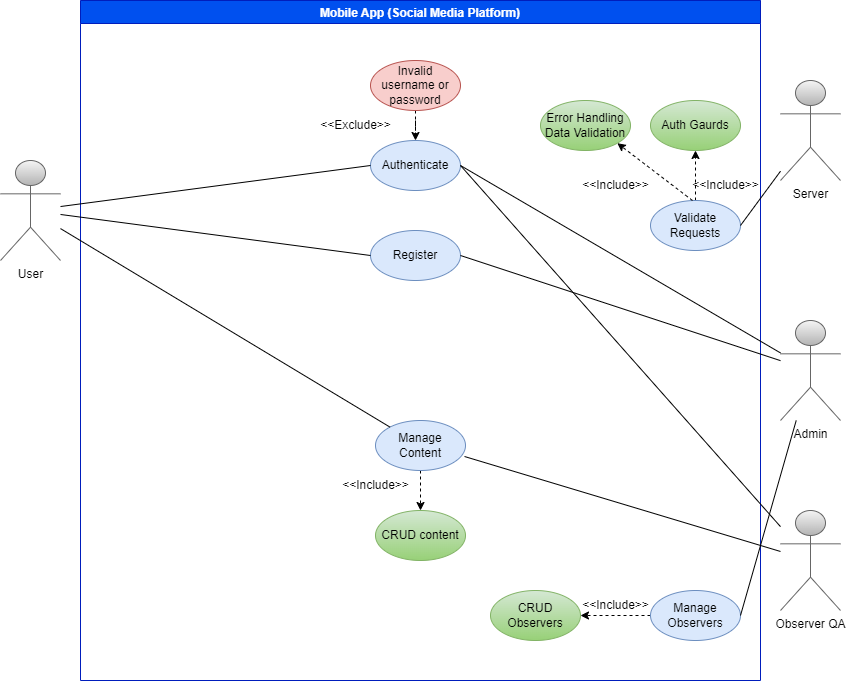
User controller has Association relation with Comment because User Controller affect Comment class by removing or adding a Comment

Comment, and Post have text, Author, likes, comments and \_id in common and

they inherit these attributes from Content Class

* Use Case Diagrams

Figure 2. Case Diagram.



User, Admin, Observer can authenticate themselves and the server verifies their identities and issues a JWT Token and stores it inside the client browser/device

User, Admin can register themselves, but Observer can't because an Observer must be created from an Admin

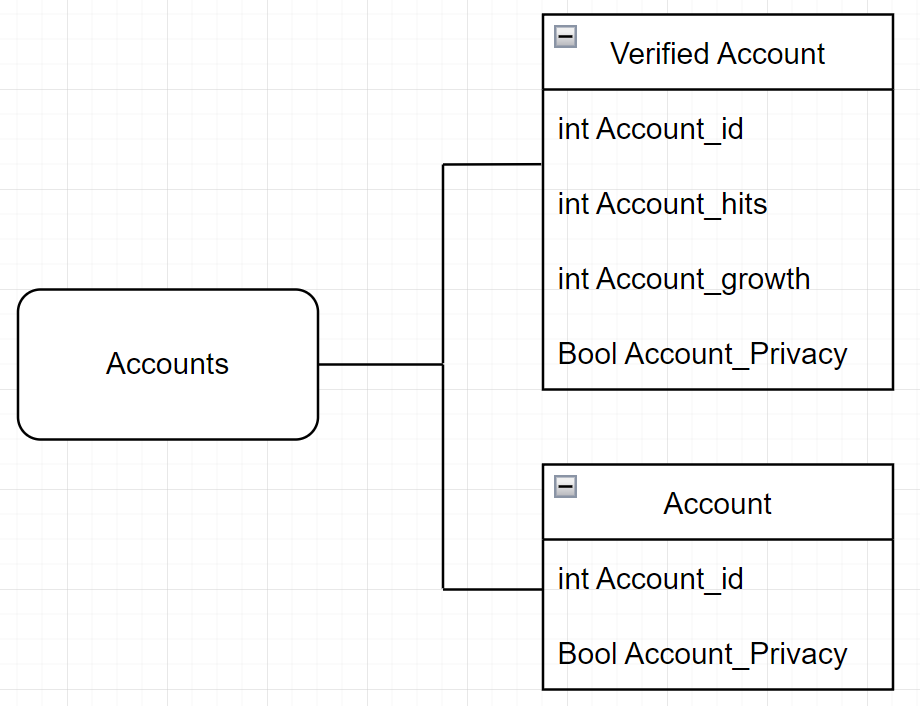
Server will validate requests by signing the Token issued on the request, also the sever can guard, validate data, and handle errors coming from each request

User, Observer manage any CRUD operation on content (post + comment)

Admin manages Observers with CRUD functionality

* Object Diagrams

Figure 3. Object Diagram.

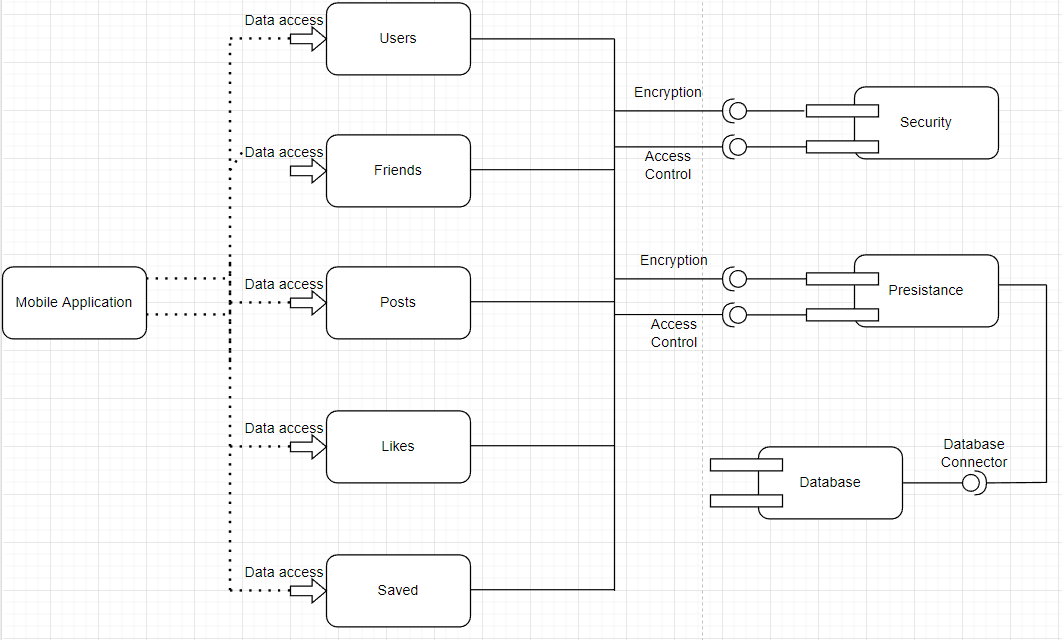


In the above figure we can notice that our application users are divided into two categories: First are the verified user accounts who usually most of them are content creators, the second type are the normal users.

We can see that the verified users can check their account hits which stands for how many people visit their page on daily basis without following them, the account growth that will be measured on a certain algorithm to help the user knowing whether their content is good or they might need to change a few things.

* Component Diagram

Figure 4. Component Diagram.



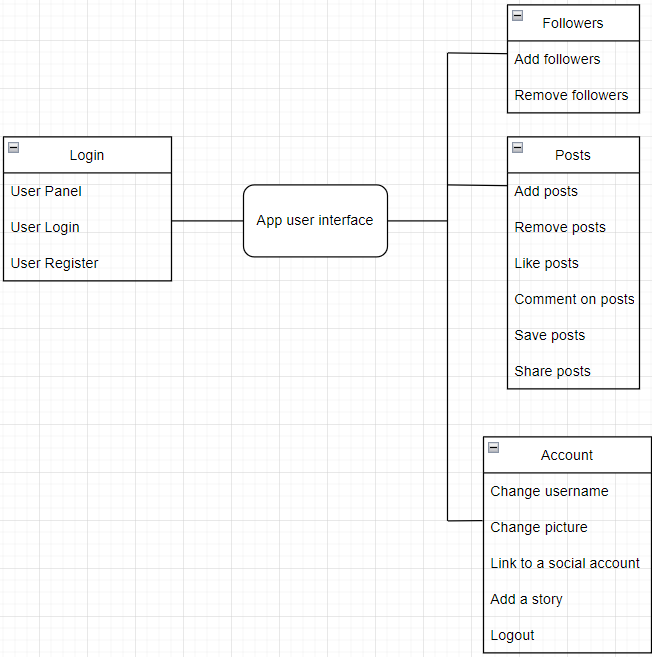
In the above figure, we describe what the user can do when opening our application:

The first page that appears to the user is the home page where they can access

their friends, posts, likes, as well as the saved page.

* Deployment Diagram

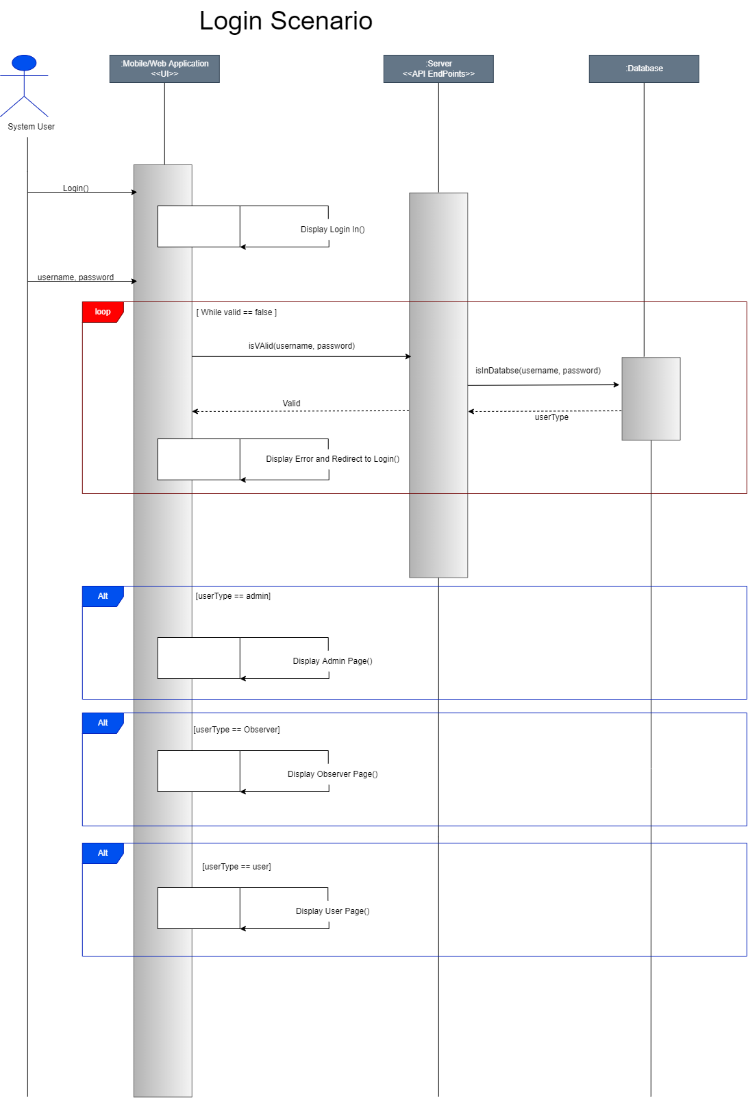
Figure 5. Deployment Diagram.



The above figure describes our user interface for the user and the features they can do after registering or signing up

* Sequence Diagram

Figure 6.Sequence Diagram.



System User: Could be an admin, Observer, or mobile app User

As could be seen from the diagram, this architecture is a microservice architecture, simply because the more abstract and modular your design is, the better performance you will have

User will only communicate with the UI that appears (mobile or web view).

When the User makes a request to login, the UI will respond with the appropriate login screen.

On entering the credentials, the request that will be sent will prompt the UI to make an HTTP request to the corresponding API.

The API will make a request to the database to see if the user exists, if the database returns the userType then we know the user was verified.

The API then will sign a Token and returns it to the UI, and the UI will display the correct User page

If the user was not verified, it will prompt the User to reenter the credentials

* State Transition Diagram

Figure 7. Admin State Diagram.

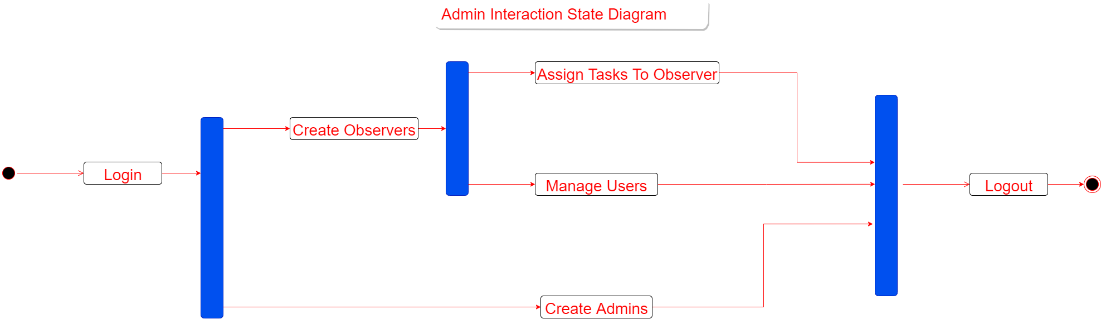
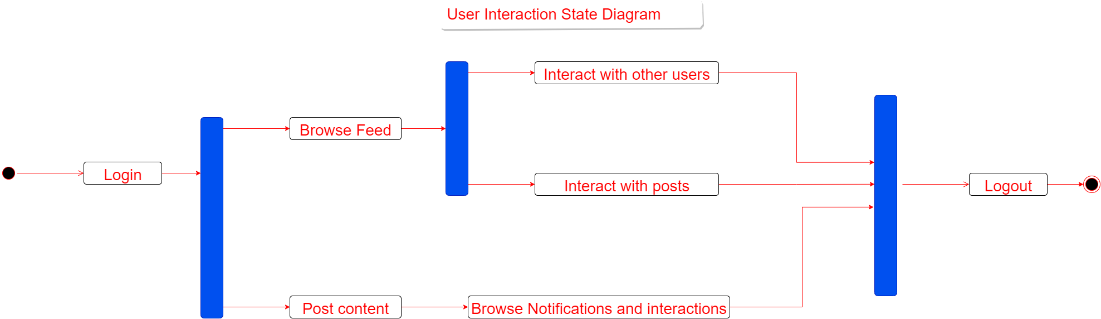


Figure 8. User State Diagram



In both approaches, you need to describe the design of the data in the system. Use Entity Relationship Diagrams (ERD) and provide the detailed database schema.

Describe also any design choices that are related to the user interface. Describe the different screens (or web-pages), and how the flow moves between them. Use appropriate diagrams like trees to describe the structure of web-pages (for example).

Make sure to organize this chapter into sections and subsections in a manner that is appropriate to the provided information. Make sure also to stick the formatting used in the other chapters.

# 4.2 Physical Model Design

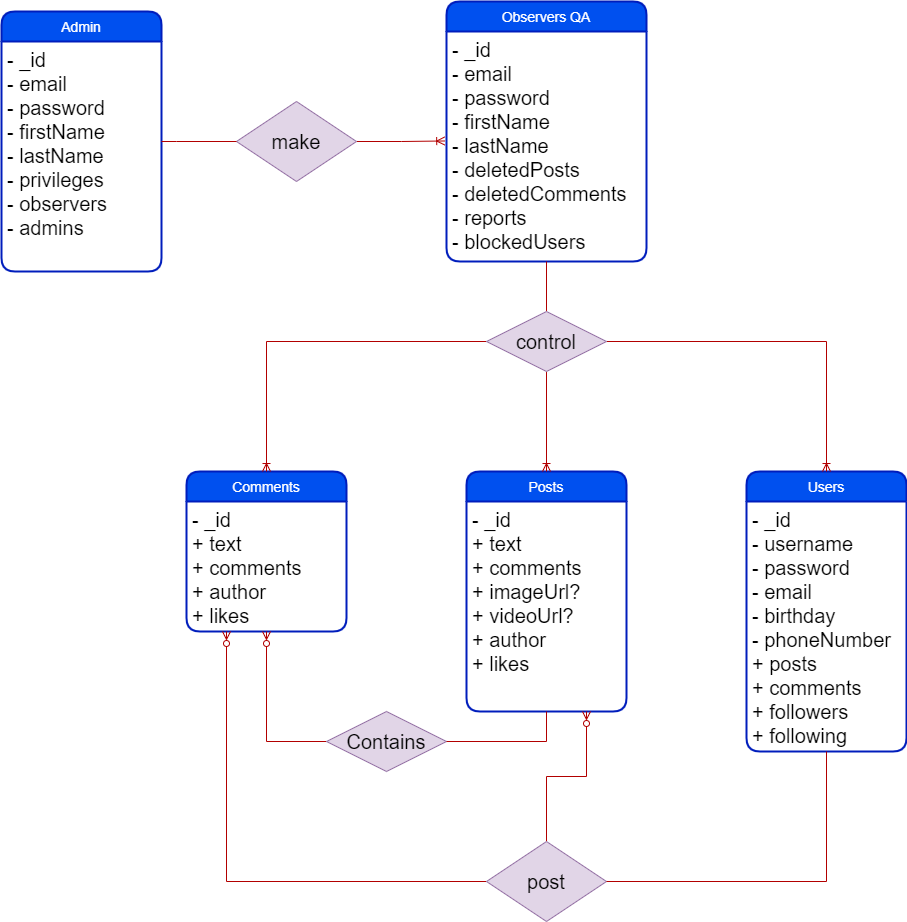
System Design must include the following:

- Reports Design

- User Interface Design

- Database Design: Database design must be presented in the form of normalized database (third normal form). This includes stored and transmitted data.

Figure 9. Database Schema.



Database Schema

Admin has

Email: for logging in or resetting a password

Password: for logging in

First Name + Last Name: Identification and keeping track of who added/removed Observers

Privileges: Only a top admin can add other admins

Observers: Observer the admin added

Admins: Admins this Admin added

Admin can make (1 to many) Observers

Observers

Email: for logging in or resetting a password

Password: for logging in

First Name + Last Name: Identification and keeping track of who added/removed Observers

Deleted Posts: Posts this Observer deleted

Deleted Comments: Comments this Observer deleted

Blocked Users: Users this Observer Blocked

Reports: Report for a user who violated terms

Observer Controls (1 to many) Posts + Comments + Users

Users

Email: for logging in or resetting a password

Password: for logging in

Birthday: To know if this user is of age to use our app

Phone Number: In case a user doesn't have access to his email he can use his phone Number for auth

Username: Displayed inside the app

Posts: Created Posts

Comments: Created Comments

Followers: Users who followed this User

Followers: Users who are followed by this User

User post (0 to many) Posts + Comments

Posts

Text: Textual information for the Posts

Likes: Number of users who performed "like" action

Comments: Comments made by Users on this post

Image Url: incase User wants to add a photo from his device the file will be stored on Cloudinary Database and take a url from there and display it inside the Posts

Video Url: incase User wants to add a vide from his device the file will be stored on Cloudinary Database and take a url from there and display it inside the Posts

Author: User who made the Post

Post contains (0 to many) Comments

Comments

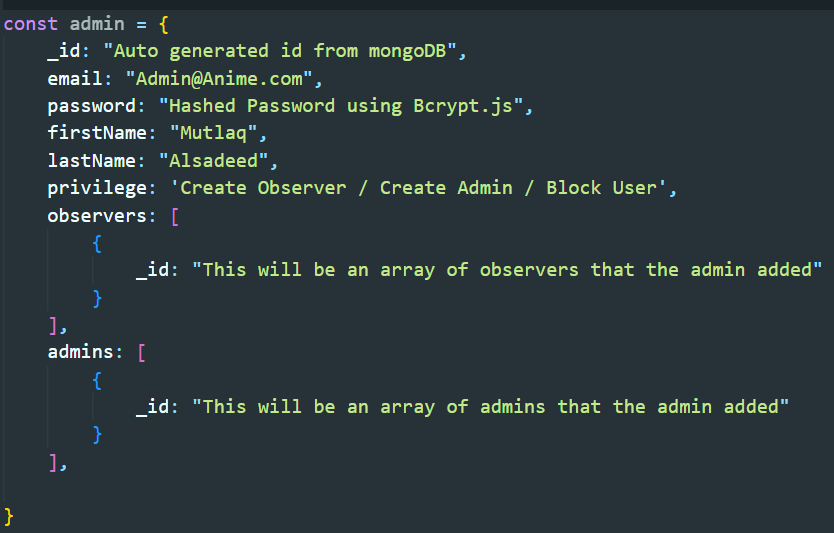
Text: Textual information for the Comment

Likes: Number of users who performed "like" action

Comments: Comments made by Users on this post

Author: User who made the Comment

Figure 10. Database Admin Code.



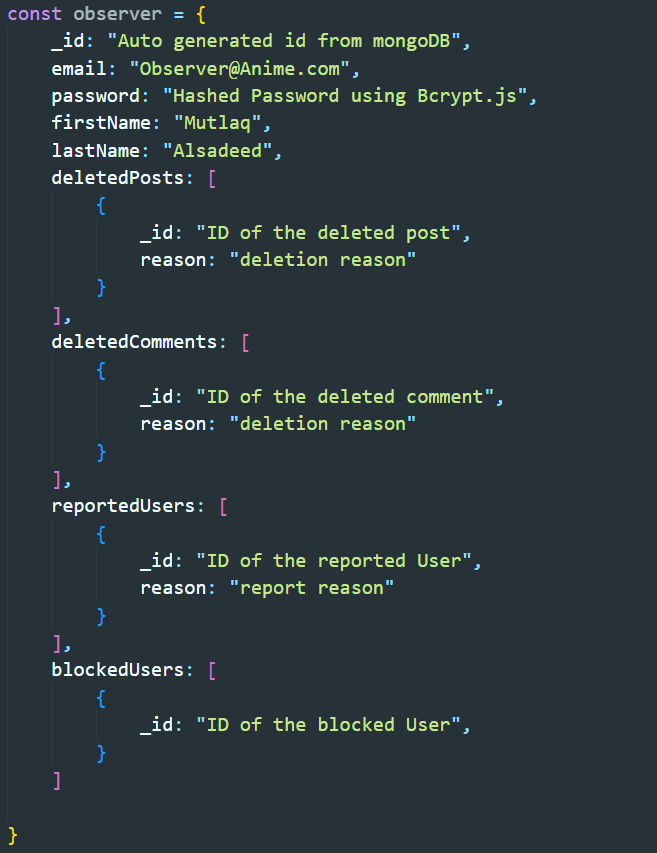
Figure 11. Database Observer Code.

Figure 12 Database user code.

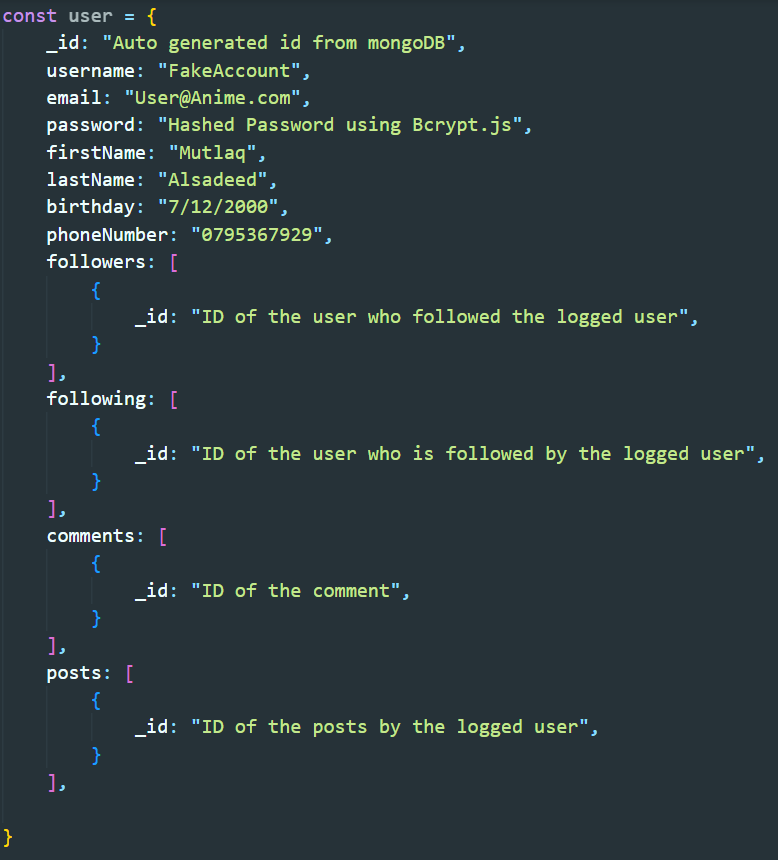


Figure 13. Database Post Code.

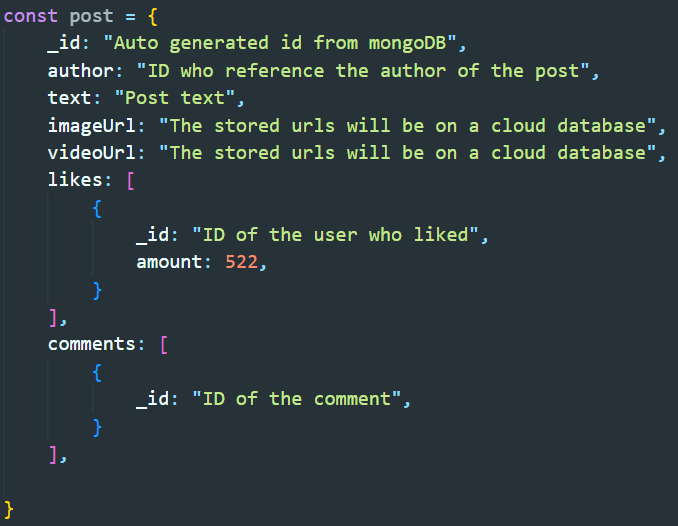
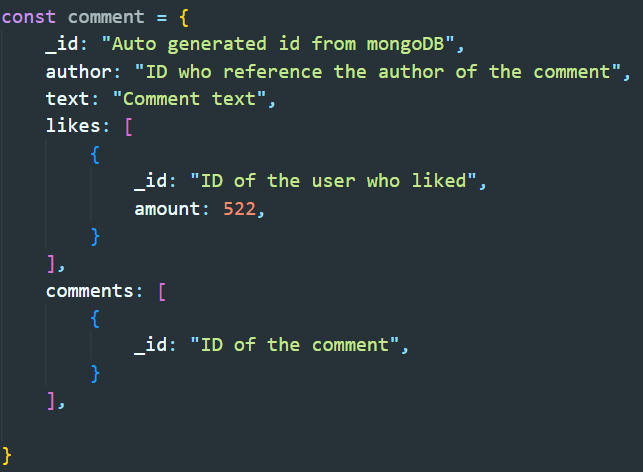


Figure 14 Database Comment Code.



**References**

Schwartz, H., Eichstaedt, J., Kern, M., Dziurzynski, L., Ramones, S., Agrawal, M., Shah, A., Kosinski, M., Stillwell, D., Seligman, M. and Ungar, L., 2022. *Personality, Gender, and Age in the Language of social media: The Open-Vocabulary Approach*. [online] Available at: <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0073791>.