**Software Requirements and Design Document**

**For**

**Group 22**

Version 1.0

**Authors**:

Olivia J.

Nicolas C.

Felipe L.

# Overview (5 points)

Our system is using Django, which takes care of role based access, authentication, and our database. We are using python for the backend to implement our list logic, posting, and other user interactions. We will be using html/css and bootstrap for frontend development.

# Functional Requirements (10 points)

1. User Posts
   1. System shall allow users to post photos (high)
      1. System shall allow users to caption their photo (low)
   2. System shall allow users to remove photos (high)
2. Account Management
   1. System shall allow users to create an account (high)
      1. System shall provide a page for users to view and edit their profile (high)
      2. System shall allow users to set a profile name (high)
         1. System shall allow users to change their profile name (low)
      3. System shall allow users to log in to their account (high)
      4. System shall allow users to log out of their account (high)
   2. System shall require users to set a secure password during registration (medium)
      1. System shall require users to confirm the password during registration (low)
   3. System shall allow users to reset or recover their password via email (low)
3. Interactions with Other Posts
   1. System shall allow users to like posts (medium)
   2. System shall allow users to comment on other posts (low)
   3. System shall allow users to follow other users (low)
4. Admin Features
   1. System shall allow an admin account (high)
      1. Admin shall be able to remove any user’s post (high)
      2. Admin shall have the ability to suspend or delete user accounts (low)
5. Bucket List Generation and Management
   1. System shall provide user with a initial randomized list of 25 bucket list items upon account creation (high)
   2. System shall allow the user to reroll a bucket list item up to two times (low)
   3. System shall track the completion status of bucket list items for each user (high)
6. Search and Navigation
   1. System shall have the ability to search other users’ profiles and view their progress (medium)
   2. System shall provide a button to navigate to the home page at all times (high)
   3. System shall provide a button to navigate to the user’s profile from the home page (high)
7. Security and Authentication
   1. System shall enforce a 30-minute lockout after 3 failed login attempts (low)

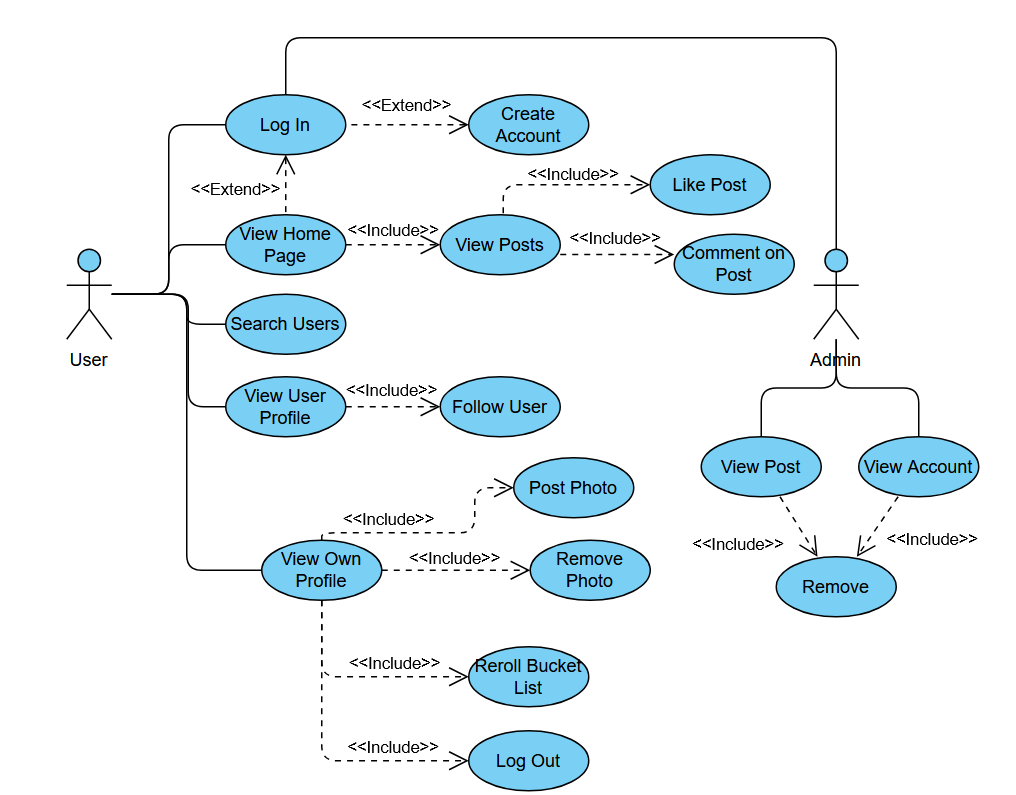
# Non-functional Requirements (10 points)

1. Security
   1. System shall store user passwords in database
   2. System shall store user passwords using secure cryptographic hashing
   3. System shall resist brute force attacks with mechanisms like account lockouts and rate-limiting
2. Performance
   1. System shall load any page or feature in no more than 2 seconds, under normal load conditions
   2. System shall handle up to 100 concurrent users without significant performance degradation.
3. Reliability and Usability
   1. System shall execute the desired behavior upon the first click of any button
   2. Code shall be well-documented, readable, and maintainable
   3. System shall follow user-centered design principles for ease of use
4. Scalability
   1. System shall be designed to accommodate future growth, supporting new features or user influx without requiring major architectural changes
5. Data Integrity
   1. System shall ensure that data related to user progress, posts, and interactions is stored reliably and consistently.

# Use Case Diagram (10 points)

*This section presents the* ***use case diagram*** *and the* ***textual descriptions*** *of the use cases for the system under development. The use case diagram should contain all the use cases and relationships between them needed to describe the functionality to be developed. If you discover new use cases between two increments, update the diagram for your future increments.*

***Textual descriptions of use cases****: For the first increment, the textual descriptions for the use cases are not required. However, the textual descriptions for all use cases discovered for your system are required for the second and third iterations.*



# Class Diagram and/or Sequence Diagrams (15 points)

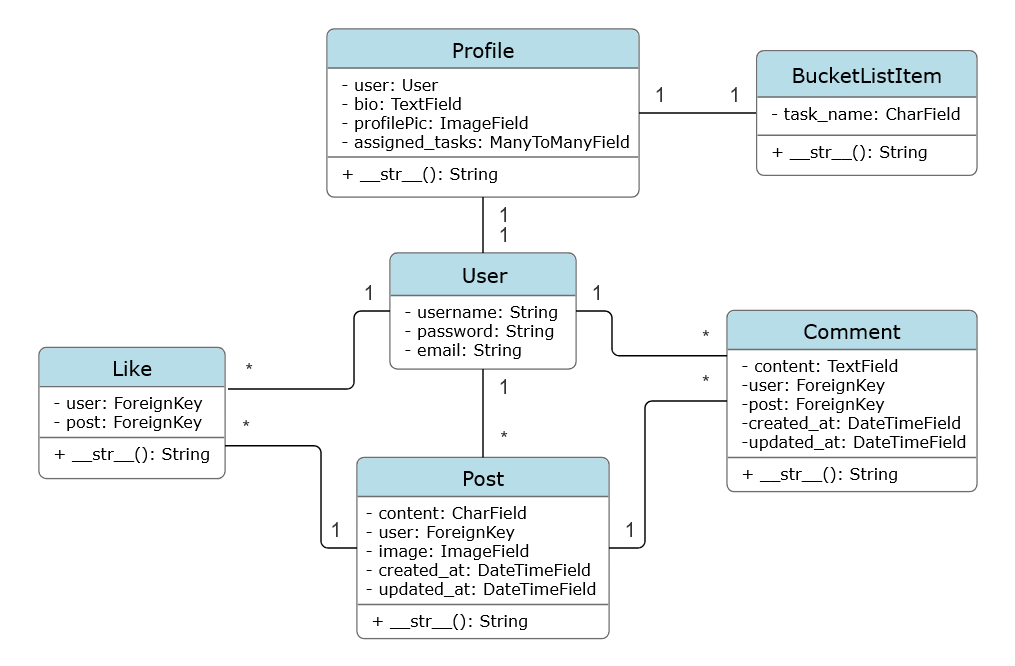
*This section presents a high-level overview of the anticipated system architecture using a* ***class******diagram*** *and/or* ***sequence diagrams****.*

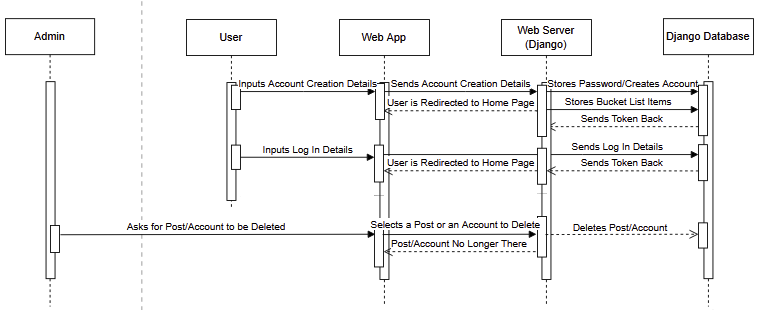
*If the main* ***paradigm*** *used in your project is* ***Object Oriented*** *(i.e., you have classes or something that acts similar to classes in your system), then draw the* ***Class Diagram******of the entire system and Sequence Diagrams for the three (3) most important use cases in your system.***

*If the main* ***paradigm*** *in your system is* ***not Object Oriented*** *(i.e., you* ***do not*** *have classes**or anything similar to classes in your system) then only draw* ***Sequence Diagrams****,* ***but for all the use cases of your system.*** *In this case, we will use a modified version of Sequence Diagrams, where instead of objects, the lifelines will represent the functions in the system involved in the action sequence.*

***Class Diagrams*** *show the* ***fundamental objects/classes*** *that must be modeled with the system to satisfy its requirements and* ***the relationships*** *between them. Each class rectangle on the diagram* ***must also include the attributes and the methods of the class*** *(they can be refined between increments). All the* ***relationships between classes and their multiplicity*** *must be shown on the class diagram.*

*A* ***Sequence Diagram*** *simply depicts* ***interaction******between objects*** *(or* ***functions -*** *in our case - for non-OOP systems) in a sequential order, i.e. the order in which these interactions take place. Sequence diagrams describe how and in what order the objects in a system function.*



**

# Operating Environment (5 points)

The program uses version 5.1.2 of Django and version 3.10.0 of Python. It is able to run on Linux, Windows, or Mac as long as it can access a browser.

# Assumptions and Dependencies (5 points)

One assumption is that Django will function properly. Specifically, version 5.1.2. From this, it also assumes that a lot of Django’s features will be running properly, like its authentication system or its database. It also assumes that it will be ran from a computer that it is compatible with and that it will be able to upload photos. It assumes a reasonable sized and number of files uploaded at a time.