

# System Design Document: Social Media Privacy

Jaime Wong, Jose Ladipo, Kayla Johnson,  
Rohit Koul, Shreya Rawal, Simran Kaukab Shaik  
Department of Information Systems  
University of Maryland, Baltimore County

December 9, 2019

## Contents

<b>1 Introduction</b>	<b>2</b>
<b>2 Data Flow Diagrams</b>	<b>4</b>
<b>3 Entity Relationship Diagram</b>	<b>9</b>
<b>4 Class Diagram</b>	<b>14</b>
<b>5 State Diagram</b>	<b>17</b>
<b>6 Discussion of Coupling and Cohesion</b>	<b>19</b>
<b>7 Interface Structure Diagram</b>	<b>20</b>
<b>8 Updated Project Plan</b>	<b>23</b>

## 1 Introduction

The system design phase is the follow up to the analysis phase in the Software Development Life-cycle process (SDLC). We presented a monitoring system solution to protect social media privacy that Generic Corporation has shown interest in pursuing and approved for the next phase of development. The problem analysis and requirements document helped to identify the business problem and feasible requirements for a solution. The process of gathering requirements and analysis helped to identify the features for developing this system. After gathering all of the relevant information from the customer, we can understand and document what exactly is needed in the list of requirements

so developers have a clear understanding of what exactly to build. The next phase of the project is the design phase. The design phase is used as the general input to the system architecture, and reference point for the system's implementation.

The purpose of the System Design Document (SDD) is to define the interactions between the system components for the software development team to have the correct direction to develop the system per the requirements described in the requirements document. The system design document will convey in thorough detail how the software should be built. Previously, we identified that there is no effective monitoring system in place to protect and ethically monitor social media privacy. Our product intends to provide a 24/7 surveillance system that will protect the user's social media privacy that follows security and compliance regulations. The structure of this paper will include several graphical design diagrams with a narrative discussion to follow that supports the requirements gathered from the analysis document. The system design document will contain six diagrams to describe the base level design of the system we intend to create. The diagrams included in this paper are data flow diagrams, entity-relationship diagrams, class, and state diagrams, and interface structure diagrams, in that particular order. The data flow diagram will show how the data flows through the system, how data is transformed, and how it is stored. The entity diagram shows the overall framework and explains the relationships between the different entities in the overall system. The remaining diagrams provide a more detailed view of the data such as the attributes, methods, and life-cycle of data involved in the system. Each diagram will have an accompanying justification for the creation of each diagram and its importance to the design of the system as a whole.

Going through the process of design helps to ensure that all aspects of the system are covered. The process of modeling the system through graphical representations is useful for making sure all of the requirements are satisfied. This is an important lesson to learn when transitioning to the implementation phase. The developers will not be able to ensure that the system satisfies all of the requirements unless the design is thoroughly documented. This will also help to minimize errors made during the implementation phase. The system design document also identifies all of the data, attributes, methods, and relationships so that the system works as one cohesive unit and not several different parts. The goal of the system design document is to have a point of reference that will be used as a "proof of concept" blueprint for the base level system. Another goal of the design phase is to transform all of the requirements from the analysis phase into detailed specifications that cover the entire system. The System

## Entity Relationships for Privacy and Social Media

Design Document will be circulated back to the major stakeholders so that there is a common understanding and documentation throughout each phase of the SDLC.

It is imperative to mention the importance of this document compared to the requirements document and the next phase after the design document. The requirements document describes a problem and the justification for the project. Our team came up with what we determined to be reasonable requirements that would resolve a business problem with satisfactory results for the benefit of Generic Corporation. The design document is in place to direct and track the information needed to define the actual architecture of the system. The development team will not be able to write code solely based on requirements. The development team will reference this document for the low-level design specifications and overall design goals. Specifications are needed for each system component including the hardware (from the requirements document), external interfaces, and internal communications. The SDD defines both the high and low-level specifications and is intended for several audiences. The development team will be looking for low-level design specifications to write the program code. The project management and project team may also use this document to communicate with the stakeholders regarding the overall user-oriented functional design. The SDD is not concrete, as the document should be expanded to fit the overall project needs. The SDD is the last roadblock before the creation of the tangible system.

With all of the data that is being passed and accessed through the system, the development team needs specifications on how to store it. The SDD will define all the major data objects and structures used to store and process the data. The diagrams modeled in this document are useful for how to design the database management system. After defining how data is stored in the system, the implementation and testing phase is the next phase in the SDLC process. The implementation and testing phase, often combined in Agile methodology, which we have selected, is where the actual coding is written by the developers. They need to have a good understanding of the design of the application in order to generate and write the actual code for the program to know how the application will function. It is at this stage in the SDLC that project management will know whether or not the project is getting close to the problem solution or if it is still lacking.

## 2 Data Flow Diagrams

**Data Privacy DFD:** The parent diagram below is the Level 0 data flow diagram exploded from the Context Diagram presented in the previous document. The data flow diagram shown below is the child of the IoT User represented in the Context Diagram from the Problem Analysis and Design Document. This diagram intends to show how the monitoring system will securely grant user access to their social media platform through multi-factor authentication. The data flow diagrams are included at the beginning of this document to give a more localized view of several aspects of the system and how data is stored and will flow through the system. A more detailed view of how the data flows, entities, and classes are discussed later in this document. Visualizing how the information flows through the system will help to ensure that the requirements listed during the analysis phase are satisfied.

In the Data Privacy DFD below, the process of logging into a social media account via the data monitoring system is explored. In process 1.0, the user will first supply account information for their social media account to which they wish to connect to our system. The account must then be validated and verified. Multi-factor Authentication is one of the requirements that were identified during the analysis phase and will be used for account verification. The user is then connected via a secure encrypted connection after their account has been verified and can now edit profile, make posts and upload photos. Per the requirements, these options will have the full capability so that the user has full rights to remove a tagged photo or a post that they do not wish to have affiliated with them. Data regarding the user's security preferences, GPS location preferences, likes, dislikes, and other content will be stored into the user account database. This is managed by the system administrator in process 4.0. The system information will be continuously updated with user data as they decide to connect more social media accounts for protection. The process of online registration is captured in the second child diagram below.

**Online Registration DFD:** The first child diagram presented is an explosion of the first process in the parent diagram. This data flow diagram illustrates how a user will register with our data privacy system. Upon initial use of our system, the user will be presented with an online registration form to fill out data to register an account with our system. This data will be stored into the User account database. Once this information is stored, the user's account will be registered and reviewed by the system administrator. Once the administrator acknowledges the account, the user will have successfully created account

## Entity Relationships for Privacy and Social Media

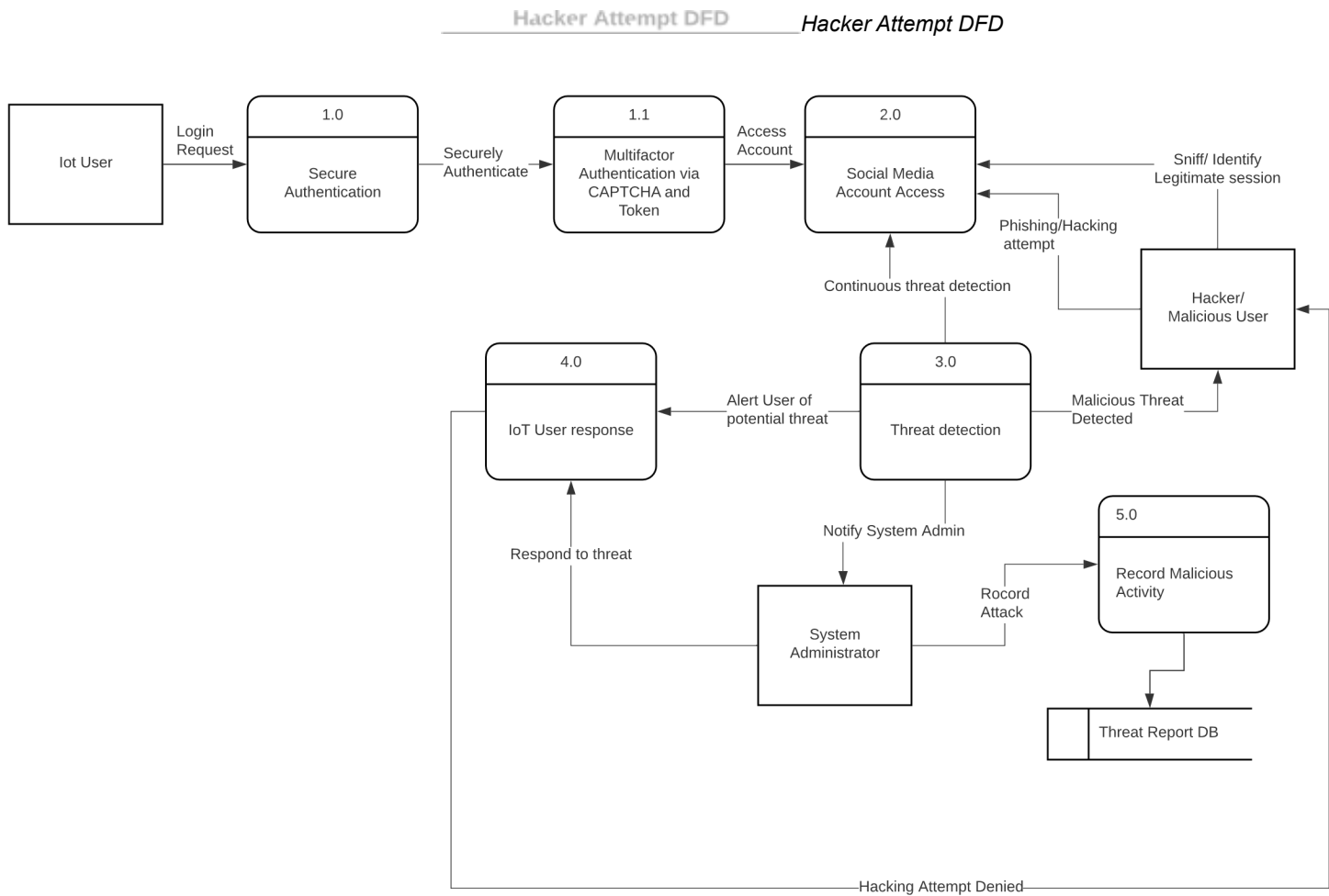
credentials and will be able to connect social media accounts to their discretion for data privacy protection.

**Social Media User Account Access DFD:** In the third child diagram, the process of user account verification via multi-factor authentication is explored. The IoT user makes a login attempt to connect their social media account that they have chosen to connect to our data privacy application. The user is then connected via secure encryption via HTTPS. The login request will be authenticated once the system verifies a secure internet connection. The user will then have to log in to their account via multi-factor authentication via two methods. The first will be a CAPTCHA test and entering a security pin. We have decided to satisfy this requirement by using CAPTCHA and a security token to access a social media account. CAPTCHA is a program that protects websites and applications against bots and computer programs by generating and grading tests only detectable by humans (captcha.net). This is discussed later in the class diagram. Users have to read and identify distorted text or images to securely login to their account. This is critical to our system as it provides the first layer of defense against users attempting to gain unauthorized access. Multi-Factor authentication protects against spammers searching for email addresses to send potentially harmful data to users in an attempt to access their data.

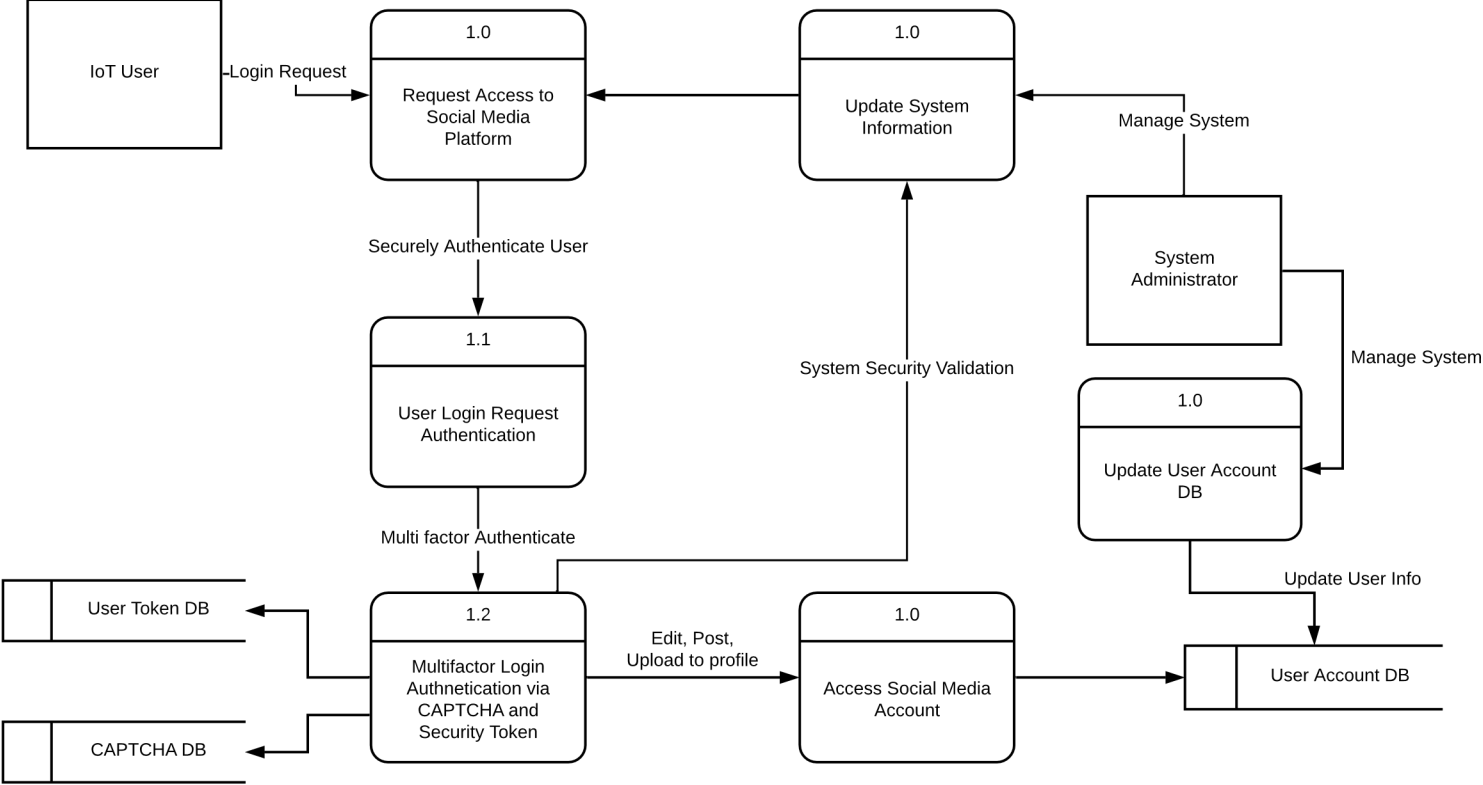
The Update System Information process relates to the functional requirements of the system that listed user authentication be mandatory for every login at different systems. The system information is managed and updated by a system administrator who will ensure that the user goes through the multi-factor authentication process when the system recognizes a new login attempt. The system will be updated through a system administrator to ensure that user credentials are properly managed. The system administrator is responsible for recording threats detected into the "Threat Report" database. This is after the system recognizes a malicious threat, identified by the "Threat Detection" process. The IoT user and system administrator are both notified via an alert or warning message. The user then response to the threat consciously deciding whether the malicious activity detected was intended. If the user identifies a malicious threat such as an advertisement with a broken or bad link, for example, upon the user's response, the system will deny the threat via automatic redirect or some other action. The system administrator will also receive a notification that the advertisement was malignant and the threat will be recorded in the "Threat Report" database.

**Hacker Attempt DFD:** The final DFD presented in this document explores the process of denial of malicious hacker threat attempts. In this DFD, there are three external entities in role: IoT user, System Administrator, and Hacker. This diagram shows the process of how the system protects against malicious users sending phishing emails and malware to users. The user sends a login request and the system grants access after verifying a secure and encrypted internet connection. The user must then log in via multi-factor authentication via CAPTCHA and token as explored in the previous DFD. Once they have logged into their account securely, they can access their accounts. The process of threat detection occurs in 3.0. The system is regularly scanning all social media content such as tweets, Facebook posts, ads, and photos. To illustrate the course of action when a malicious user attempts to gain access into the user's account, we have added a "Malicious User" entity. A malicious can attempt to gain access in several different ways. This can mean "sniffing" for a legitimate session or user accessing their account. A hacker may "sniff", which means to monitor packets of data that is being transferred over a network and intercept them for suspicious reasons. Alternatively, they can send phishing messages such as a private facebook message from a fake profile trying to convince a user to click on a bad URL. When this is detected, an alert will be sent to the IoT user and the System Administrator Entity. The IoT user will then respond to the threat warning and will have to authorize the content that the system has detected. The System Administrator will also receive a notification and then reports the activity to the Threat Report database. All of these entities work together to provide continuous protection. The system logs the threat and will block similar threats in the future.

Entity Relationships for Privacy and Social Media



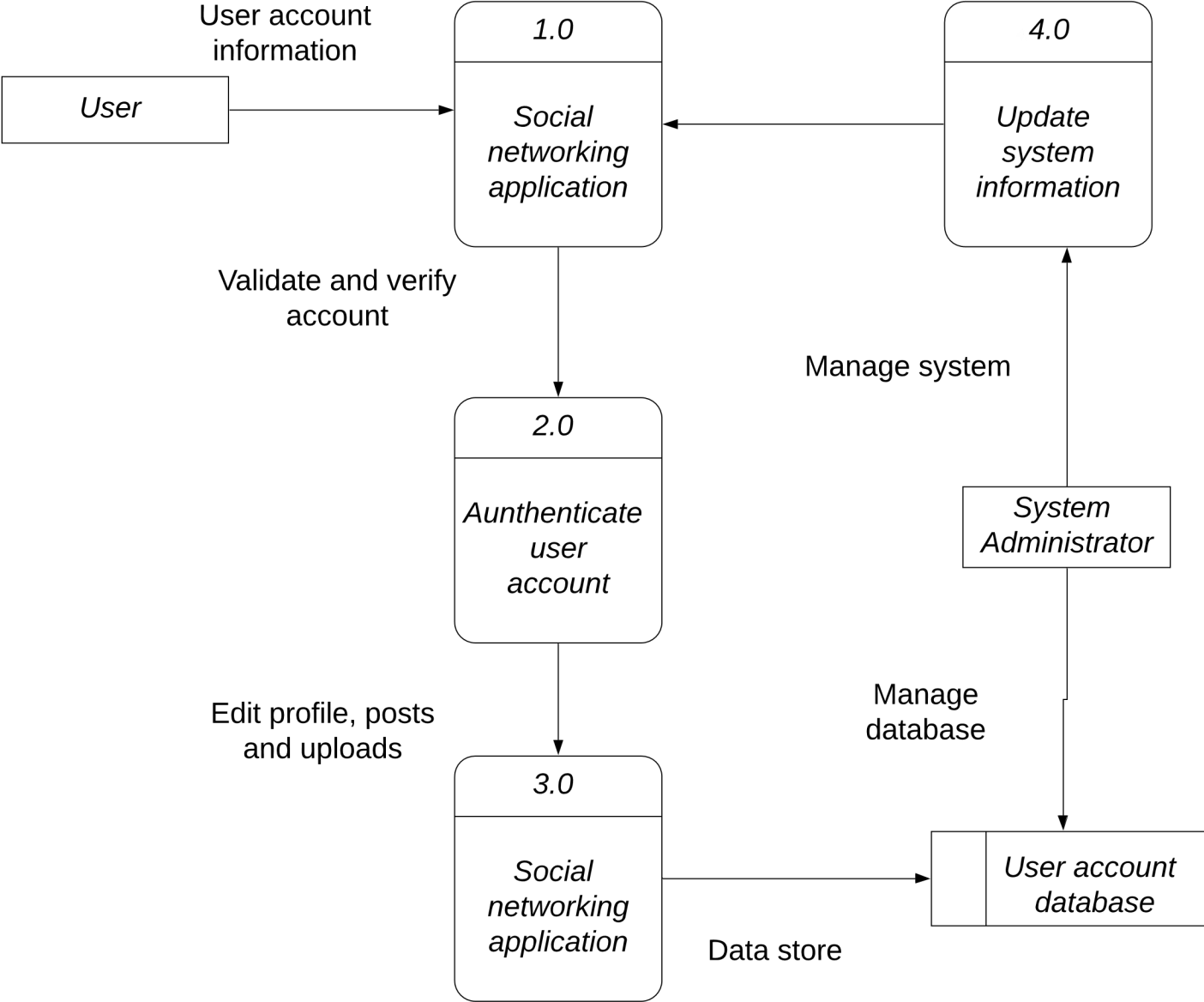
Social Media User Account  
Access



**Data Flow Diagram parent social networking application** *Data Flow*  
*Diagram parent social networking application*



Entity Relationships for Privacy and Social Media



### 3 Entity Relationship Diagram

An Entity Relationship Diagram (ER) is a data modeling diagram that projects information about systems entities and the relationships between those entities. The three basic elements of ER Diagram are Entities, Relationships, and Attributes. In creating an ER Diagram, we can identify and define entities, determine interactions between entities, identify cardinality between relationships, and map these relationships graphically. We have defined the key components of an ER diagram below:

**Entity** The structure of the data that can be people, places, or things.

**Cardinality** The uniqueness of the data and how often the data appears in other entities. Entities can have one-to-one, many-to-many, or many to-many relationship.

**Attribute** Description or identifiers of different entities.

**Relationship** An association that describes relationships between entities.

In an Entity Relationship diagram, we can identify the primary key and foreign keys that would be used when creating a table in a database for storing the data. With the help of entity-relationship diagrams, we will be able to associate the relationship between all the data structures as necessary. With the help of previous DFD's, we have identified the following entities:

1. IoT User (Internet of Things)
2. Media Applications
3. Privacy Systems
4. Malicious User/Threat/Hacker
5. Malicious Activity Recorder
6. System Administrator
7. Web Server Or Database **Entities for**

#### Privacy and Social Media

## Entity Relationships for Privacy and Social Media

**IoT User:** The IoT User plays the key role for the system's core functionality. The user connect their personal social media applications for access requests. It has the following attributes: IoT\_User\_ID, First\_Name, Last\_Name, Phone\_Number, Location, Account\_ID where IoT\_User\_ID is the primary key here to identify different users and Account\_ID is Foreign key for references with table Media Applications.

**Media Applications:** Media Applications represent different social media platforms such as Facebook or Twitter where people share their likes, dislikes, and comments. This is where general content will be stored. Media Applications has the following entity attributes: Account\_ID, Account\_Name, URL\_Name where Account\_ID is the primary key. The primary key is the unique constraint to avoid redundancy. System\_ID is a foreign key that relates to the table privacy system.

**Privacy Systems** Privacy systems has the following attributes: System\_ID, Authentication\_ID, Login\_Date, Logoff\_Date where System\_ID is the primary key. The privacy system safeguards the security and privacy of an application using multi-level authentication. Account\_Id is a foreign key that ensures referential integrity for the media applications table. System\_ID states the system specifications such as system number and name in the description below it. Authentication\_ID is the login details of a user where a unique id is generated in the system. Login\_Date will record the date when it started the application in the system. Logoff\_Date will record when the application in the system is turned off.

**Malicious User/Threat/Hacker** The Malicious user entity has the following attributes: Malicious\_User\_ID, IP\_Address, IPV\_Address, DNS\_Address, and Account\_Id where Malicious\_User\_ID is the primary key and Account\_Id is the foreign key referencing to Media Applications. Malicious\_User\_ID is a person, an ill-intentioned organization, contractor or intern that has misused or broke the rules of the company. IP\_Address represents a numerical based indication attached to each device connected to a computer network that uses Internet protocol for communication. IPV\_Address may be of two types IPV 4 and IPV 6 where each represents 32 bit, version 4, 4 numbers in the range of 0 to 255 separated by a dot and 64 bit, version 6, consisting of 8 groups of hexadecimal digits and where zeroes are there it is separated by a colon

respectively. DNS\_Address is a directory of domain names like a phone book which are used to translate into Internet Protocol Addresses. relationships between entities.

**Malicious Activity Recorder** Malicious Activity Recorder is a set of security tools that can track and detect malicious activity. The attributes Record\_ID, Record\_Name, Token\_ID, Malicious\_Activity\_type, Malicious\_User\_Id are attributes of the ER Diagram. Record\_Id is the identified unique identifier that increases sequentially for each record in the table and data is a primary key here. Record\_Name is a unique description of respective record\_id. Token\_ID is a token-based authentication to cache user profile information and to provide it to whoever needed. Malicious\_activity\_Type may be malware, bugs, bots, rootkits, spyware, trojan horses, viruses, or worms. Malicious\_activity\_Type is needed to indicate from the above list selection. Malicious\_User\_Id is a foreign key to this table that references to malicious\_user table.

**System Administrator** The System Administrator is responsible for system management in depth can be the person responsible for the upkeep, configuration and reliable operation of computer systems like servers. It has attributes like System\_Admin\_ID, System\_Update\_ID, System\_patch\_id, System\_Update\_Name, System\_Patch\_name, Configuration\_Patch\_id, database\_ID, Account\_Id. System\_Admin\_Id is the primary key whereas Database\_ID and Account\_Id are foreign keys relating to Web Server, Database and the Media Applications entities.

**Web Server Or Database** Web server is a server software or hardware dedicated to running software that can satisfy worldwide web requests and database is a collection of data that is organized for rapid search and retrieval by a computer. Has attributes like Database\_ID, Database\_name, database\_hostname, database\_servicename, database\_service\_job\_details. Where database\_ID is a unique id that distinguishes separate rows. The DB different types like Token DB or Captcha DB can be grabbed and information about it can be kept here.

---

There are several possible entity relationships between all of the attributes such as IoT user and media applications respectively. The relationship between these two is many to many because there can be multiple users who are using the system and

## Entity Relationships for Privacy and Social Media

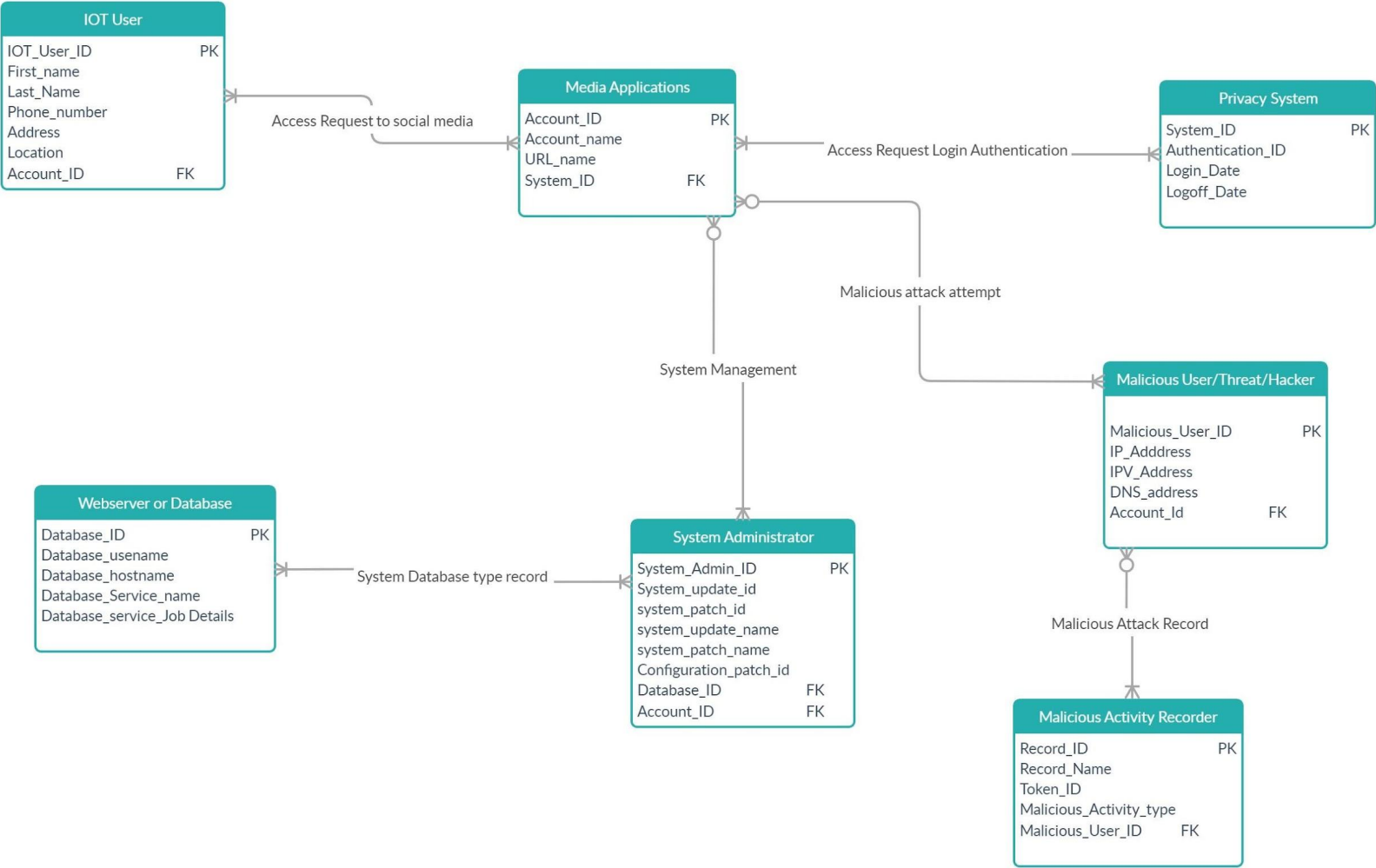
each user can have many social media accounts. This is the case for several other entity relationships.

In the case of the **System Administrator entity and the web server**, this relationship would be one-to-one. Typically there would be one system administrator who manages the database and web server for the system.

Additionally, there could also be a scenario where there could be multiple threats or hackers who would want to access the confidential information of a particular user from all over the world. For Instance, there could be a person who belongs to politics or a film celebrity whose personal information could be leaked easily, and hence the hackers find a suitable chance to find their personal information or information related to their bank accounts which are usually kept highly confidential. In other words, it is a kind of many to one relationship where there are multiple hackers who are looking forward to a particular person's information on the social media platform he or she could use.

**Media Applications - Privacy Systems:** Similarly, there is a case where each social media platforms such as Instagram, Facebook, Snapchat or Whatsapp has a unique code or **Authentication ID** which helps to identify a particular user and their level of accessibility on a particular platform. For example Database administrator of Facebook could have the maximum access rights to access the information of a person who has an account on Facebook, but at the same time the Database Administrator could also have an account which is for personal use only. So in this type of situation, there is a restriction of access rights as into the administrator or as a common user. In the case of cross verifying a particular user is authenticated to access how much of the information that depends on the privacy system. this is one to one relationship in the entity-relationship diagram.

**Malicious User/Hacker - Malicious activity recorder:** In this case, although there are multiple users who are active on several social media platforms, but for every individual user there's an activity tracker or recorder, which is responsible for keeping track of a particular user and the activities done by the user is recorded in the database. For instance, a particular user has posted a photo on a particular platform and has posted a video on the second platform. For a particular user, there is a unique id generated as soon as the user logs in the account for a particular day. For instance the session is not ended the id is retained in the activity recorder database and the activities for all the users are tracked.



### 4 Class Diagram

Class diagrams show the hierarchy of classes and are used as the building blocks of object-oriented software systems. Class diagrams are used to show classes, attributes, associations, and operations within object-oriented programming. Modeling classes in this particular diagram is beneficial for simplifying complex software design that will be presented to major stakeholders. Reducing long and verbose explanations in graphical diagrams can help create clear and concise communication between the software developers, the stakeholders, and management. Class diagrams also help to illustrate the entire system in a more holistic view. Class diagrams use specialization and generalization to describe various objects and the relationships between them whereas Entity Relationship Diagrams focus on the relationship between the entities themselves. The class diagram consists of the parent class, child classes, attributes, methods, and the association between them. An association is a relationship between two classes represented by an arrow. Associations have different multiplicities including one to one, one to many or many to many. There are class-level relationships in a class diagram. The inheritance relationships indicate that one of the two related subclasses is considered to be a specialized form of the parent (superclass) and the superclass is considered a generalization of the subclass.

We have chosen to create a class and state diagram to focus on specification models since we're proposing a software protection service for social media applications. A class diagram is best for this project because there will be multiple subclasses that the system interacts with. The class diagram will assist stakeholders to understand the methods and actions each object performs and identify the relationships between classes. The class diagram depicts how the client interacts with the application, along with all of the internal and external operations.

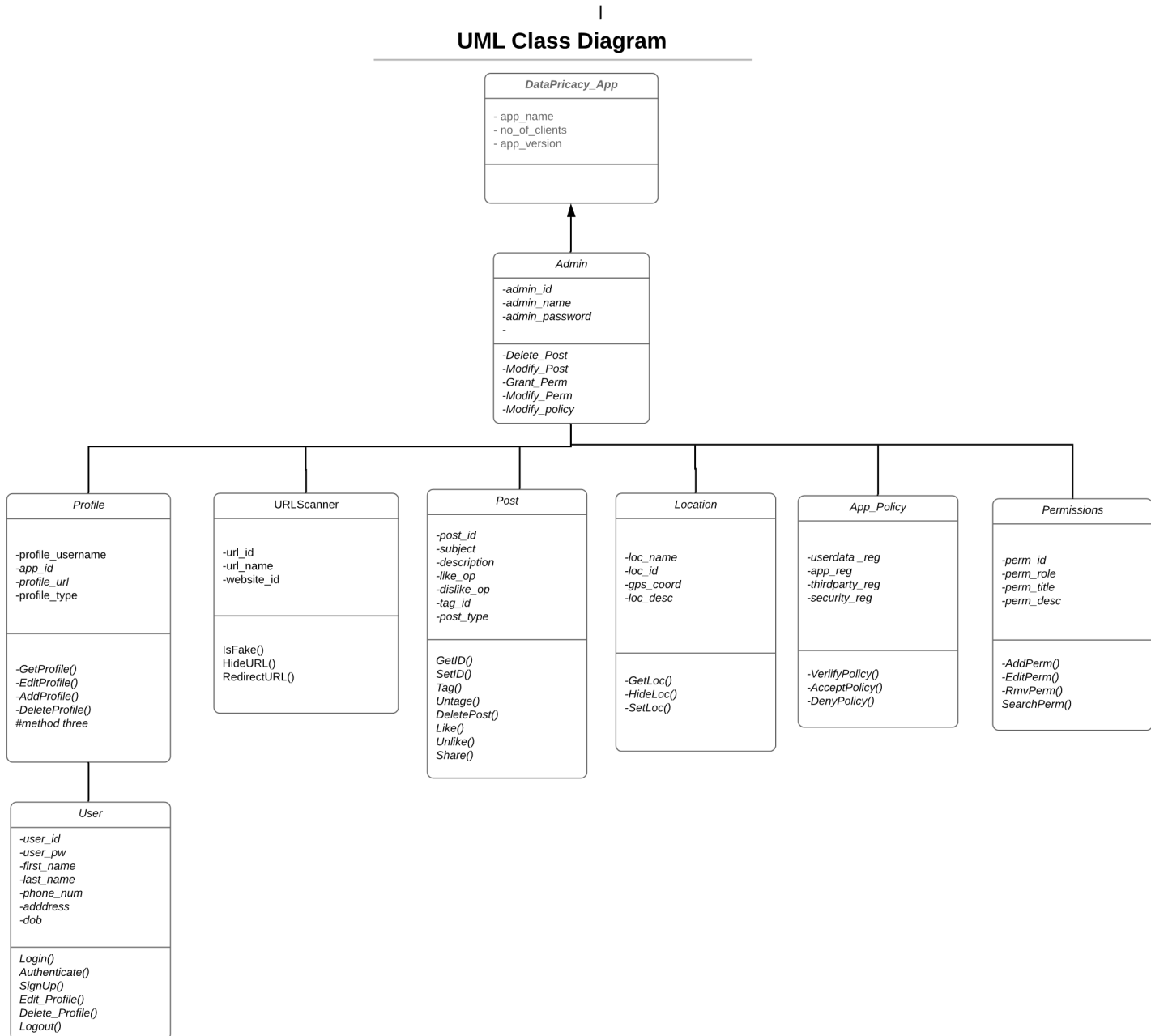
The administrator is the intermediary between the data privacy application, social media applications, and users. The administrator manages a variety of responsibilities including ensuring all social media applications comply with the new regulations and confirming that the user has accepted the terms. The administrator will be allowed to modify posts and permissions associated with the apps and remove any contracts that do not comply with the new standards. Each user signed in to our privacy system will have a profile that contains a record of each social media account. That way, an encrypted database with each person's login credentials is regularly monitored and secured by the system administrator. Once the user provides login credentials and the system verifies compliance with the user agreement pol-

icy for a specific social media account, access is granted. The URL scanner class checks the application to detect any malicious content and verifies a secure HTTPS connection before granting access. The location class allows the admin to control what applications have access to user locations and block any third-party apps from accessing it without proper permissions. The application policies class will detail the functional requirements such as the user agreement policy and server-level encryption that the databases must acquire to interact with user data. The permissions class allows the admin to grant and revoke certain permissions to the social media platform such as location or GPS tracking information. Each class has a clear set of tasks independent of each other to make development, maintaining and adding new features to each class simple and consistent. Each method within a class has a focused purpose allowing our object-oriented design more highly cohesive and loosely coupled. Coupling and cohesion will be discussed further in later sections.



## Entity Relationships for Privacy and Social Media

### UML Class Diagram



## 5 State Diagram

The state diagram allows us to understand the state of our data privacy application throughout its execution and some of the decisions that are to be evaluated during the different stages of the application while the system is running. The state diagram is useful for showing how the behavior or states of a

class changes throughout its life-cycle. The transitions are finite, which means that each class must have an initial, transition and a final state. Since our system is a monitoring system, it is beneficial to show how the behavior object changes in response to either an internal or external event.

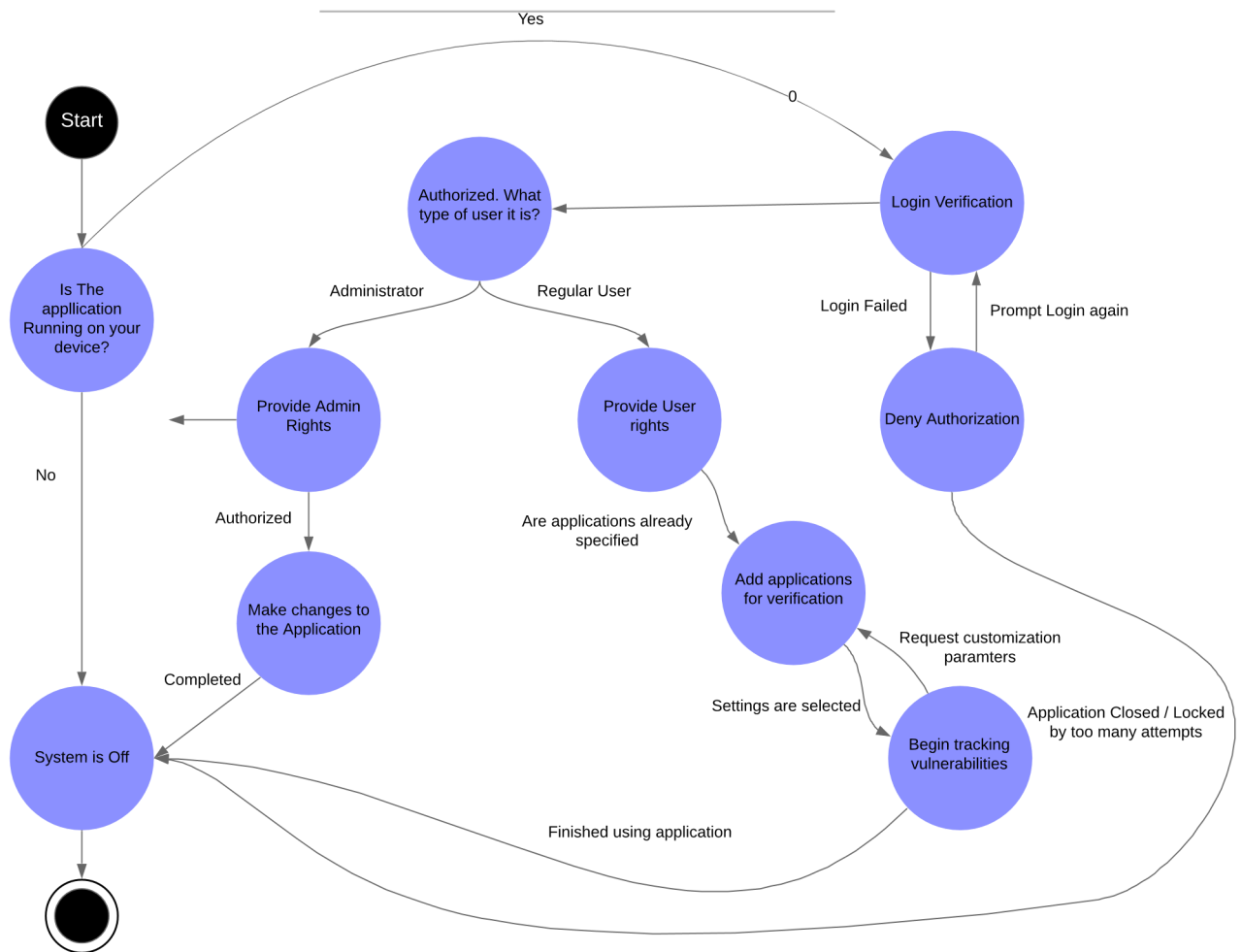
The State Diagram below is an overview of the state transition in our privacy application. As previously noted, objects must have an initial, transition, or final state. The initial state is indicated by The state diagram below shows us the flow of execution in response to a specific event. We confirm the status of whether the application is being used or not, and after verifying the user's login information we can obtain the proper rights to either be a developer which needs to maintain or make modifications to the application, or its a regular user beginning to use the application, it would bring us to a prompt to customize the application to the user needs before it can perform its tracking functionality and secure the information. In the event that the login information is incorrect it would prompt again to login. If the user has incorrect credentials, the account will lock automatically after 3 attempts and require a password reset. A new password will need to be created in compliance with non-functional requirements 11 and 7. Functional and Non-functional requirements can help a project achieve standards create, and formality. The state diagram can be derived from the requirements in order to describe the user's navigational perspective when it comes to the user interface design. In the "Begin tracking vulnerability state", this is where the majority of the time will be spent by all the key role players (IoT users and the System Administrator). In this state, a lot of the functional requirements will be satisfied including monitoring of fake profiles, allowing location access to third parties, and posting secure messages.

The initial status is the verification of whether or not the application is under operation by the user or administrator, before we can establish login verification, based on such results we'll realize whether the state is that the application is running or is off, and depending on the result it would continue to the next state or it will end and close. After checking the usage status of the application, we would verify security login, by checking the state of the privileges, like a regular user or an administrator. If the check fails the state would change to rejected and will close, and depending on the credentials inserted the state would shift for a full developer, or a customer user with its respective privileges, which would allow us to either hugely customize the application or to be granted access for usage of our application and make the necessary changes to begin tracking our accounts as intended on this application. After the proper settings are selected would grant access to our software to do the proper tests and will turn to an active state, and once completed the usage it would turn back to off once it is closed and exited.

## Entity Relationships for Privacy and Social Media

Determining the access by level of permission, allows us to control the data access across the life cycle, however having a proper interaction between our application and the social media pages the customer is registered, presents a challenge for the development team as it needs to go over such information, with at the same time safekeeping the customer's information, ensuring that its data is secured through the process, and at the same time, coordinating with third party authorizing us with the approval of the customer, to access all the information and handle it appropriately and at the same time, make sure that we are allowed to manage it, which is totally separate from the actual process of analysis as our application is intended to work.

State Diagram - Privacy Application



## 6 Discussion of Coupling and Cohesion

Systems are designed with the expectation of maximizing cohesion of the elements on their modules and minimize coupling. To better understand these concepts we are briefly explaining each concept and then we'll showcase of coupling and cohesion found in our system. According to our definitions from Dr. Massey's Chapter 7 slides for IS-636, cohesion, and coupling are defined as follows:

**Cohesion** Degree to which one module serves one and only one purpose.

**Coupling** Degree to which module of a system is connected to other modules of a system." Maintaining high cohesion is essential for systems to help prevent crashing, time outs, and low performance. The best way to improve cohesion in our system and minimize coupling is to prepare a breakdown data flow diagram, to begin with.

To protect our system from threats and identifying the user that is accessing our system we are using cohesive mechanisms like joining the efforts of CAPTCHA, login system, and multi-factor authentication to make sure that we are not being accessed by bots and the person that is trying to log in is who he claims. Similarly, there are some difficulties with coupling our authentication with ensuring that the user does not confuse the access to our application from the access to other social media accounts logins.

In our Privacy and Social Media application, cohesion and coupling are both essential. In the data privacy application, linking the application name, clients and app version also plays a vital role as the present version holds new data of clients if by any case previous versions are used then present trending data would be not applicable. Every application has an admin who is responsible for managing a database or web servers. This layer is attributed to the first level. Coupling between data privacy app and admin is essential in this case. Without these two modules, connectivity of data related to present clients and version control of the application is not possible because the admin handles version control. In case the admin needs to, they can access, grant or revoke permissions; access basic profile details like need to add, edit or delete profiles; post or tag an update; gain access to the geographical location of users or third party apps for privacy or security policies. An admin needs to access data from different modules where coupling is essential between modules where it is high and efforts must be taken to reduce this coupling else system failures or crashing chances would be pretty much higher. This layer can be attributed to the second

## Entity Relationships for Privacy and Social Media

level. Efforts to be taken are to delete the fake user accounts to reduce the load of the system and temporarily archive the user profiles in the database. Doing this ensures information that is not useful is deleted from the database to maintain the efficiency of the database.

High cohesion means keeping parts of a codebase that are related to each other in a single place. Low coupling is about separating unrelated parts of the codebase wherever possible. Make sure that the module does not do many things. A single module specifies the behavior. Coupling is needed for system connectivity to some extent. However, we must ensure that the cohesion between modules is functional and not coincidental and coupling remains minimal. This can be best explained using a simple example of a class in our class diagram. To discuss this further, we will use the URL Scanner Class for reference. There are three elements and three methods in this class. To make cohesion functional, the attributes and methods in this class are grouped exclusively because they are in relation to the overall behavior of the URL Scanner task as a whole. In this case, the URL Scanner class is cohesive but can be improved. To improve cohesion, the attributes `url_id`, `url_name`, and `website_id` should be private elements. Each method in the class: `IsFake()`, `HideURL()`, and `RedirectURL()` use each private element of the class. `IsFake()` would reference the `url_id` and `url_name` and `website_id` to determine whether or not a website URL that a user is attempting to access is fake. This should also be the case for the remaining two methods in the class. Making the elements private ensures high cohesiveness because only the methods declared inside of that class can access the private elements. This will reinforce that only the module itself specifies the behavior and it is not dependent on any other class.

The goal of outlining cohesion and coupling in our data privacy application is modularization. Each data module should work independently. As developers work to write the code for the system, they will want to ensure that every module performs the intended task individually. However, the modules come together in order for the system to come together and work together as a whole. Designing the program code in this way helps with maintainability. Creating independent modules means that they can be reused per each system requirement. Having to write them repeatedly would be superfluous. As the SDD is a “working” document, the modules are not set in stone. They can be modified in future revisions when the development team identifies cases of high coupling. The development team can use this document to reference the diagrams where cohesion and coupling may not be easily identifiable.

## 7 Interface Structure Diagram

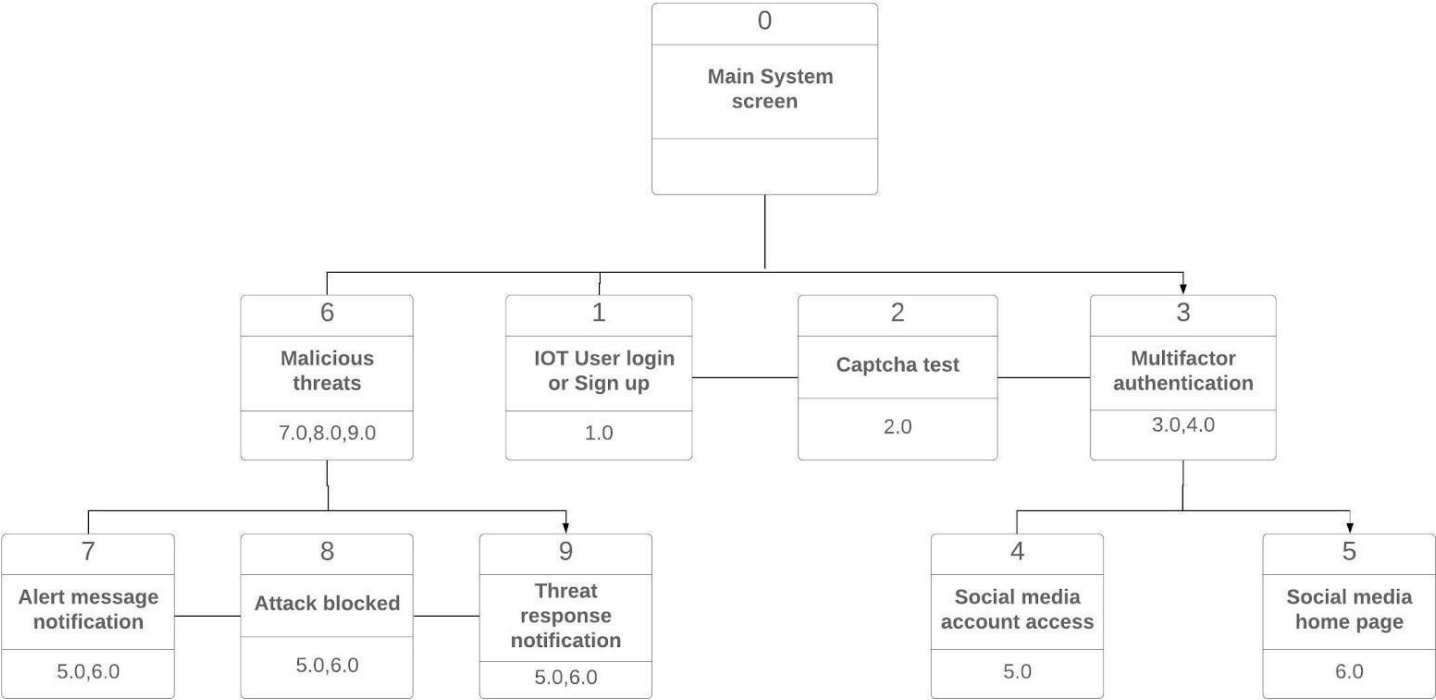
The Interface Structure Diagram (ISD) defines the basic components of the interface the user interacts with on display and how each component works to provide functionality to the user. It shows the actions the user inputs into the system and the output messages the system responds with. It shows all screens, notification messages, forms, and reports the system displays and how the user navigates each of them. Interface Structure Diagrams are based on Data Flow Diagrams. The basic structure of the ISD is very similar to the Data Flow Diagram but there are no standards for the ISD. There are also relationships to how the interfaces are linked with the overall system and external entities. Most systems have several ISD's, and there is typically one for each part of the system.

In our Data Privacy system, we chose to implement a monitoring system through Multifactor Authentication via CAPTCHA and security token that grants the user access safely and in a timely manner after verifying a secure connection. Our Interface structure diagram, which is based on the DFD, starts with the user accessing the social media account's main home page. The user is then presented the option to log in or sign up to create an account. Upon entering their login details and logging in, the system checks to verify if they are an authorized user that is logging in using a secure connection. It does through Multi-Factor Authentication. This is a process in which the user is prompted to provide two or more pieces of evidence to an authentication mechanism. Some factors of a multi-authentication scheme could be a password, a PIN or a physical object such as an ATM card. In this case, the user must enter their username and password to be granted access. This will protect the user against scammers from accessing their personal information.

CAPTCHA is an acronym for Completely Automated Public Turing test to tell Computers and Humans Apart. CAPTCHA is a challenge-response test that deters bots and computer programs from accessing a human's account illegally (Wikipedia, 2019). It is a computing algorithm to determine if the user is human or not. Through this system, the user is redirected to a screen that instructs them to correctly evaluate and enter a sequence of numbers or letters in a distorted image in order to securely login to their account. The user must pass this first layer of defense before it can proceed. The user has the option to choose a different sequence until it correctly answers. The user is allowed multiple attempts at this test before it blocks access. After these two authentication methods, the user is granted access to access the social media page and is redirected to their social media home page. This next phase occurs if the user encounters a cyber threat or is subject to scams like identity theft or fake accounts trying to gain personal information. The system identifies the attempt and notifies both the system administrator and the user. The system administrator documents the attack and what type it is. The user is given an alert

## **Entity Relationships for Privacy and Social Media**

to warn them against the potential threat while it focuses on resolving the situation. Once the threat is nullified, it will notify the user the threat has been blocked and provide the user steps to avoid that type of interaction in the future and deal with it. This process informs the user each step of the way and their options after the situation is resolved.











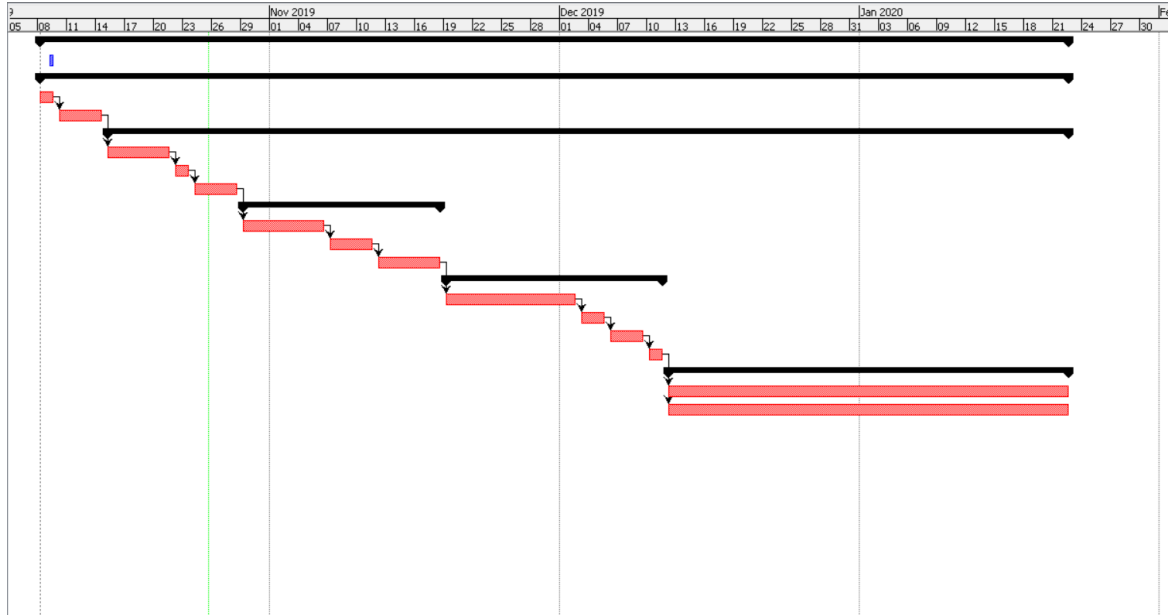


## Entity Relationships for Privacy and Social Media

### 8 Updated Project Plan

We are developing an application that will track and protect our customer's information for proper use on social media applications. In this development, we will need a team of designers to layout our requirements for this application, as well as developers to code, test and implement this application. The application would be performed according to the schedule presented in the document, and we are using a list of tasks to show the preliminary project plan with a Gantt chart to describe how the tasks are being linked.

		Name	Duration	Start	Finish	Predecessors
1		<b>Social Media Privacy App</b>	<b>77 days</b>	<b>10/8/19 8:00 AM</b>	<b>1/22/20 5:00 PM</b>	
2		Team meeting and information sharing	1 day	10/9/19 8:00 AM	10/9/19 5:00 PM	
3		<b>Planning Phase</b>	<b>77 days</b>	<b>10/8/19 8:00 AM</b>	<b>1/22/20 5:00 PM</b>	
4		Defining the problem statement	2 days	10/8/19 8:00 AM	10/9/19 5:00 PM	
5		Determining the scope of the project	3 days	10/10/19 8:00 AM	10/14/19 5:00 PM	4
6		<b>Analysis Phase</b>	<b>72 days</b>	<b>10/15/19 8:00 AM</b>	<b>1/22/20 5:00 PM</b>	
7		Feasibility Analysis	5 days	10/15/19 8:00 AM	10/21/19 5:00 PM	5
8		Selecting the SDLC Methodology	2 days	10/22/19 8:00 AM	10/23/19 5:00 PM	7
9		Requirements Summary	3 days	10/24/19 8:00 AM	10/28/19 5:00 PM	8
10		<b>Design Phase</b>	<b>15 days</b>	<b>10/29/19 8:00 AM</b>	<b>11/18/19 5:00 PM</b>	
11		Organize Development Team	7 days	10/29/19 8:00 AM	11/6/19 5:00 PM	9
12		Review Documentation	3 days	11/7/19 8:00 AM	11/11/19 5:00 PM	11
13		Design Application	5 days	11/12/19 8:00 AM	11/18/19 5:00 PM	12
14		<b>Implementation Phase</b>	<b>17 days</b>	<b>11/19/19 8:00 AM</b>	<b>12/11/19 5:00 PM</b>	
15		Write code for the application	10 days	11/19/19 8:00 AM	12/2/19 5:00 PM	13
16		Test the application	3 days	12/3/19 8:00 AM	12/5/19 5:00 PM	15
17		Getting approval and feedback	2 days	12/6/19 8:00 AM	12/9/19 5:00 PM	16
18		Deploy Application	2 days	12/10/19 8:00 AM	12/11/19 5:00 PM	17
19		<b>Maintenance Phase</b>	<b>30 days</b>	<b>12/12/19 8:00 AM</b>	<b>1/22/20 5:00 PM</b>	
20		Verify System stability	30 days	12/12/19 8:00 AM	1/22/20 5:00 PM	18
21		Make adjustments based on feedback	30 days	12/12/19 8:00 AM	1/22/20 5:00 PM	18



According to soltech's website <https://soltech.net/timeline-building-custom-software/> software projects are expected to last on average 4-9 months. In our case we are developing a real smaller scale of a project, which only lasts over a month and a half, and the plan is that we will begin with a meeting to discuss all of our original ideas to develop the application, share our insights, so we can kick off our project into a planning state where we can begin defining more specifically what the project is going to be about.

Defining a problem statement is a difficult task, therefore we initially thought that 2 days would be needed to discuss several aspects of it, however it does seem like very little time, but this can be reviewed throughout the project without affecting the final dates of the project. After the project has been defined we will go over the scope of what we are trying to accomplish, and this will be done during another meeting where we have gathered the problem statement and the thoughts from the original kick off meeting to make up a plan of what needs to be done, determining the scope and the requirements of the project.

Then we will begin the Analysis phase which is more complex and requires a little bit more of mathematical calculations as well as logical and risks interpretations and analysis, which is why we decided to add more days on our analysis because this would represent an important challenge for any development group to get together and work on resolving the potential risks and difficulties that they may face. This task requires a lot of subject matter experts and skillful members to arrive at a prompt conclusion with a very logical development.

For the design and implementation phase we needed to allocate enough time to ensure that our project would move forward to satisfy the requirements presented in the requirements

## Entity Relationships for Privacy and Social Media

document. In order to bring together all of our plans into action, there is a huge amount of time separated for this section as it is important that we get our development right when the project transitions to the implementation and testing phase. We need this time to ensure that the results are satisfactory and in compliance with the initial parameters. After our plan was put into action we also needed to ensure its functionality and longevity, so we decided to include as part of our initial plan of action and additional 30 days to maintain our system and support the customer. The additional time will assist the customer to make sure they understand our software and resolve any further inquiries. During this 30 day period, we will only consider the customer's emergency inquiries after the system is initially put into production. This time does not include general maintenance for future and regular operation of the system.

## References

- [1] Wikipedia contributors. (2019, November 30). CAPTCHA. In Wikipedia, The Free Encyclopedia. Retrieved 20:06, December 2, 2019, from <https://en.wikipedia.org/w/index.php?title=CAPTCHA&oldid=928612725>
- [2] Concepts of Cohesion and Coupling taken from IS-636 Class Slides from chapter 07 Software Architecture by Dr. Aaron Massey.
- [3] Wikipedia contributors. (2019, November 27). Class diagram. In Wikipedia, The Free Encyclopedia. Retrieved 21:02, December 3, 2019, from [https://en.wikipedia.org/w/index.php?title=Class\\_diagram&oldid=928178070](https://en.wikipedia.org/w/index.php?title=Class_diagram&oldid=928178070)
- [4] Soltech. (2019, December 9). *A Realistic Timeline For Building Custom Software*. Retrieved 18:46, December 9, 2019, from <https://soltech.net/timeline-building-custom-software/>