

**Software and Computer Security**

**SOFE 4840U**

**Identity and Access Management**

**Final Report**

**Date: March, 27 2023**

**Project Group: Group 8**

| **Name** | **Student Id** |
| --- | --- |
| **Preet Patel** | **100708239** |
| [Tiwaloluwa Ojo](mailto:tiwaloluwa.ojo@ontariotechu.net) | **100700622** |
| **Waleed El Alawi** | **100764573** |
| **Aaditya Rajput** | **100622434** |

# Introduction

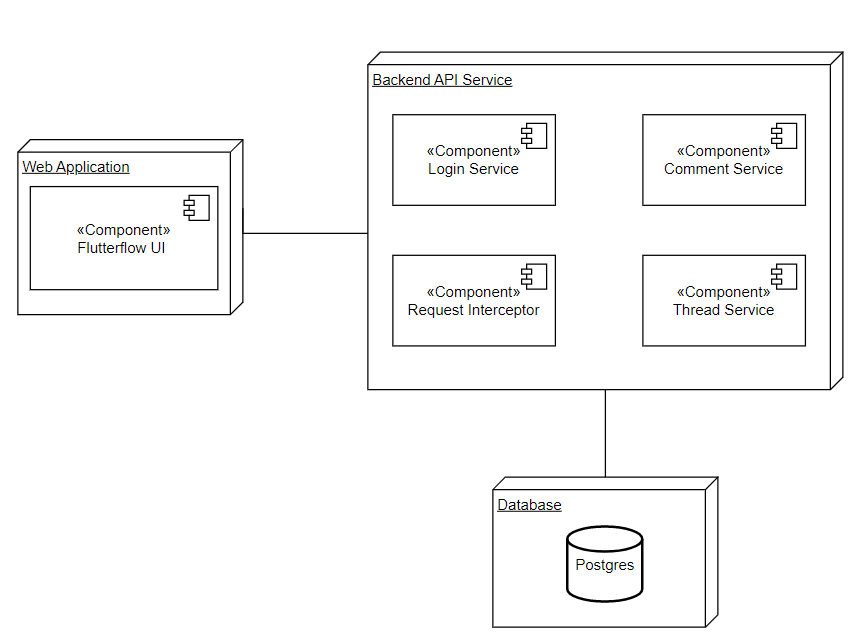
Identity and Access Management (IAM) is a framework of business processes, policies, and technologies that facilitates the management and access to digital intellectual property based on digital identities. The goal of this project is to build a IAM system that applies role based access on a forum (such as reddit) web application. The forum application will allow privileged users to create threads for discussions and let other users interact and participate in discussions by providing them the ability to post comments on the threads. The forum application is just used as an example for how the implemented IAM system would work with a web application. For the purpose of this project, the team will focus on implementing authorization using IAM and therefore will limit the functionality of the forum application to a basic implementation used only for demonstration purposes. IAM is a critical component of software security. This system will demonstrate the layers of protection against unauthorized access by authenticated users to various actions and pages of the web forum. This can be achieved by managing user identities, authentication, authorization and access control. The IAM system will shield critical functionalities and information that require higher level of security privileges from being accessed by the lower privileged users making the forum application resistant towards unauthorized access, internal attacks, data breaches, and cyber attacks. We will be using different tools and frameworks to accomplish the objective of this project.

The IAM framework for the forum platform we would like to develop would include key components such as:

1. **User Authentication:** Users are able to login and register using a username and password. The scope of this project does not include implementing 2-factor authentication or any advanced form of authentication. The project will also not focus on utilizing authorization tokens such as JWT tokens. We will be focusing on implementing counterattack measures using the authorization and role based access that is associated with IAM.
2. **Role-based Access Control:** Once a user is registered their access tier is determined. The roles and their privileges are listed below:
   * Viewer: Viewers can view threads, see comments under threads, and comment under threads themselves.
   * Creator: Creators can do everything GRs can as well as start their own thread.
   * Moderator: Moderators can delete threads and comments as well as everything Creator and Viewer can do
   * Administrator: Admins can edit threads/comments, as well as everything else Moderators, Creators, and Viewers can do.
3. **API access authorization:** The IAM system will ensure that users cannot perform unauthorized API calls which are not part of their role’s privileges.

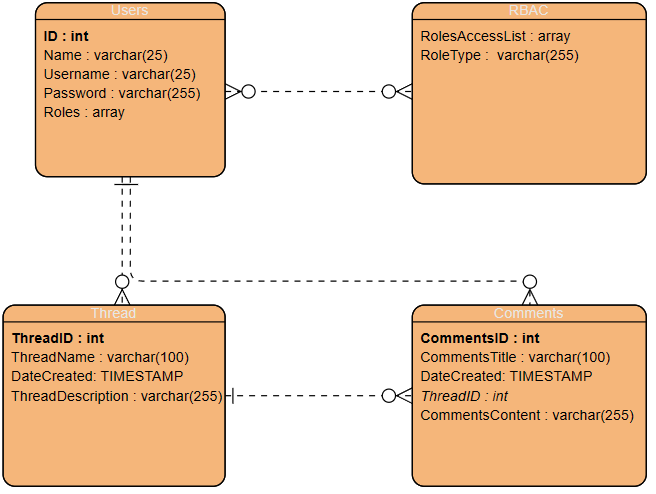
# Detailed Design

In order to implement the IAM system in this project, we will utilize a relational database to create an access matrix which will contain the data such as the defined roles that users can have in the forum application and also the API access permissions for each of the roles. This will create a role-based access control system that provides permissions to users based on their roles and rules stating what accesses are allowed to users in each defined role. Figure 1 below depicts a deployment diagram that describes the various components that will be required to implement the system is provided in the following image:



**Figure 1: Deployment Diagram**

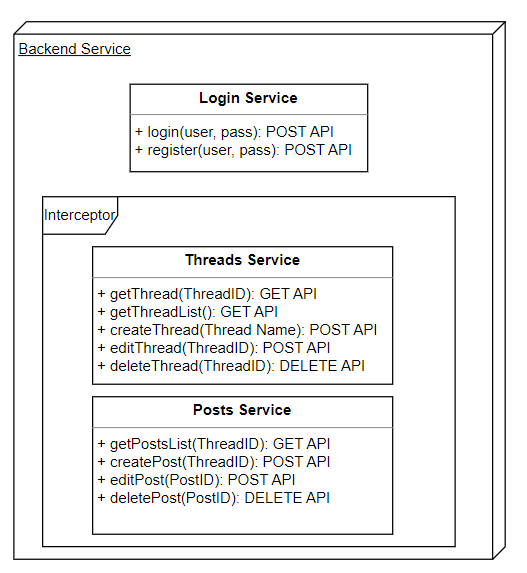
The first and most important component of the system that we will discuss is the database. The database will be used to implement and store the access matrix required to implement the IAM system. It will also be used to hold the application data such as the threads and comments which are essential to implement the forum application. Moreover, our database of choice is PostgreSQL. This DB was chosen for its unique array data type which will aid in the functionality of the access matrix. The entity relationship diagram below shows the structure of the data that will be stored in the database as well as their relationships to each other:



* **Users**: These represent the person(s) who have created an account with the application and are registered in the database. They are identified with a name, username, password, and an array of 1 or more roles. These roles include but are not limited to Admin, Moderator, Creator, and Viewer.
* **Thread**: This entity represents a conversation thread that multiple users have made comments on. Each thread is identified in the database by their thread name, the timestamp the thread was created, and a description of the threads. A single thread may have 1 to many comments.
* **Comments**: This entity represents the conversation item the user appends to the Thread. A comment can only have one thread.
* **RBAC**: Role based access control entity represents the type of roles as well as the resource they have access to.

The next component in the system that we will discuss is the web application that will be implemented using Flutterflow which will act as the user interface for the forum application. FlutterFlow is a visual development platform that allows developers to create mobile and web applications without writing code by enabling drag and drop components. The users will be able to interact with this interface to trigger various API requests that will be made to the backend service. Users that attempt to perform actions that they cannot perform due to their limited privileges determined by their role will be shown an error message that will be returned by the backend API service. An example of a “happy” interaction with the user interface involves the user attempting to view all the threads in the forum which every role of the system should have the permission to access. An example of an “unhappy” interaction would involve the user attempting to create a new thread without having the correct permission as this action requires a role of a creator or a role with higher level of security clearance.

Lastly, the final component will be the backend service which will include all the APIs that the system will require to implement the functionalities of the forum application involving login, threads, and comments. It also includes the request interceptor which will be used to implement the authorization performed by the IAM system. This implementation will utilize Java Spring Boot Framework for developing the various endpoints in the API. Figure 2 below shows component diagram of each of services and APIs that will be implemented in the backend service:

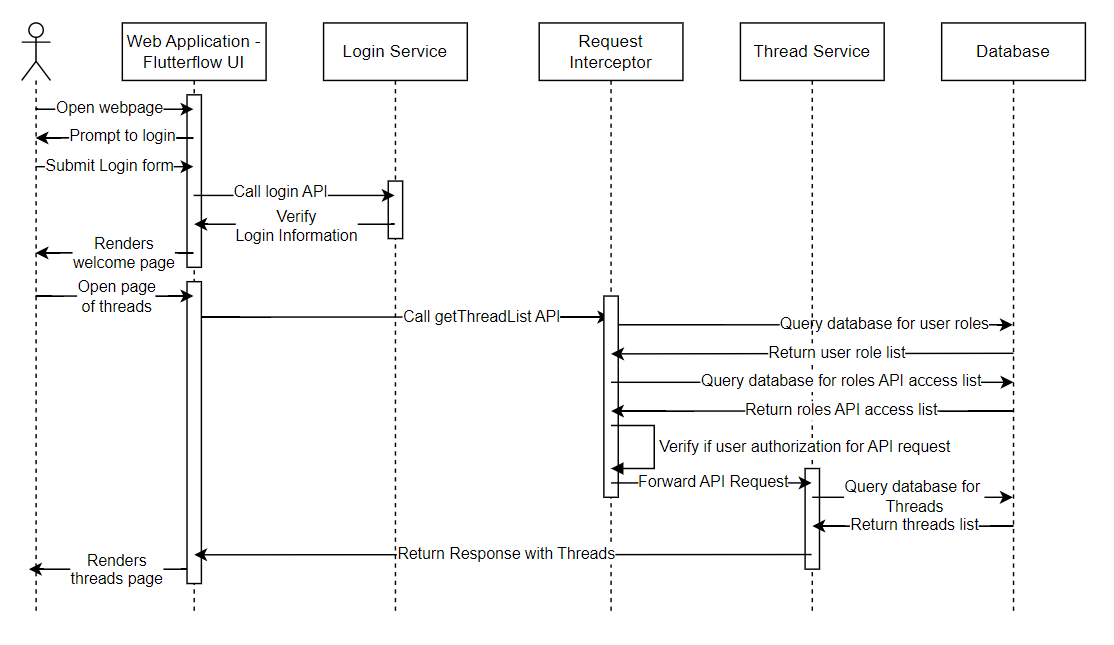


**Figure 2: Backend service and API Component Diagram**

The backend service will include the following three main service components:

1. Login Service - This service will include the login and register APIs which will call to register new users and allow existing users to login to the forum application.
2. Threads Service - This service includes all the APIs that are required in the forum application to create, view, edit, and delete the discussion threads.
3. Comments Service - This service includes all the APIs that are required in the application to create, view, edit, and delete comments on discussion threads.

The user will first have to login to the system to authenticate themselves to interact with the forum application. The user can then interact with the user interface to perform actions related to threads and comments which are the basic functionalities of the forum application. In order to implement the IAM system, we will utilize a request interceptor which will intercept any requests made to the thread or comments service to check if the user that is making the request has the authorization to perform that API request. The interceptor will query the database to obtain the user’s role and the corresponding permissions of the role from the access matrix to see if the user is authorized for the API call or not. If the user does not have the role with the permission to perform the API request, the interceptor will return an unauthorized error message and the request will not be performed. Whereas, if the user does have the correct permissions, then the request will be forwarded to the appropriate service to be handled correctly. The following sequence diagram illustrates a happy interaction between the user and the system when trying to obtain a list of all the threads on the forum:



**Figure 3: Sequence Diagram when user requests a list Threads in the Forum**

As laid out in this literature, our system will utilize various techniques and methods to achieve a robust system in which a user may only access what they have been granted permission to access.