# ASSESSMENT

# DECAGON SSO SYSTEM

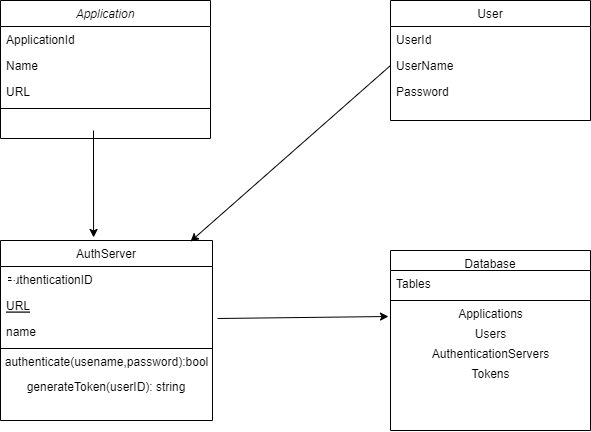
# **NWANU MARTINS**

1. A diagram showing the different systems for Decagon SSO system

A black rectangle with a black background

Description automatically generated with low confidence

1. UML Diagrams and Database Diagram



3) Approach to OOP and how to design loosely couped systems.

A loosely coupled SSO system using an Object-Oriented Programming (OOP) approach could be designed by following these steps:

* Create an abstract class for the SSO system. This class will contain the core components of the SSO system such as authentication methods, authorization methods, and authentication tokens.
* Create concrete classes for each authentication and authorization provider. These classes should have a common interface so that they can be interchangeable in the SSO system.
* Create a Service class that will be responsible for managing the authentication and authorization providers. This class will handle all requests to the SSO system, such as logging in and out, as well as providing access to the authentication tokens.
* Create a client class that will be used by the applications wanting to use the SSO system. This class will handle all communication with the SSO system and will provide the authentication tokens when needed.
* Create an authentication token class. This class will be responsible for generating, storing, and validating authentication tokens.
* Create a Data Access Layer that will be responsible for storing user information, authentication tokens, and other data related to the SSO system.
* Finally, create a User

4) A deployment plans

* Establish Requirements: Identify the system requirements, such as user authentication and authorization, identity management, single sign-on, and integration with existing systems.
* Design the System: Develop a system design that meets the identified requirements.
* Identify Resources: Identify the hardware, software, and personnel necessary to implement the system.
* Develop a Project Plan: Create a project plan that outlines the timeline and tasks required to successfully deploy the system.
* Test and Debug the System: Test and debug the system before deployment to ensure that it meets the required specifications.
* Train Personnel: Train personnel on the system’s use and maintenance.
* Deploy the System: Deploy the system to the intended users.
* Monitor and Maintain System: Monitor the system for any issues that arise and provide maintenance as necessary.

5) Your design should be based on a specific software architecture.

The Software Architecture design for an SSO system typically consists of the following components:

* Authentication Service: This is responsible for authenticating the user and providing a secure authentication token.
* Authorization Service: This is responsible for authorizing the user to access the application or service.
* Single Sign-on Service: This is responsible for providing a single sign-on experience for the user across all applications and services.
* Identity Management Service: This is responsible for managing user identities and providing single sign-on capabilities.
* Security Token Service: This is responsible for issuing and managing security tokens used to authenticate and access protected resources.
* Application Integration Service: This is responsible for integrating the SSO system with the applications and services.
* User Interface Service: This is responsible for providing the user interface that allows the user to access and manage their information.

6) Write about an approach to security

An approach to securing an SSO system

* Implement Multi-Factor Authentication: Multi-factor authentication (MFA) is a security system that requires more than one method of authentication from independent categories of credentials to verify the user’s identity for a login or other transaction. Examples of MFA include using a combination of something the user knows (e.g., a password or PIN), something the user has (e.g., a hardware token or cell phone), or something the user is (e.g., biometric data).
* Use Encryption: Encrypting data as it moves between the user and the application helps safeguard it from unauthorized access. Encryption can also be used to secure communications between the application and the identity provider.
* Use Access Control Lists: Access control lists (ACLs) are used to define who has access to which resources and what actions they can perform. These lists can be used to specify which users have access to certain applications and services, as well as what type of access they can perform.
* Monitor Logs and Activity: Logs and other activity should be monitored for suspicious activity, such as failed login attempts or unusual usage patterns. This can help detect and prevent unauthorized access.
* Regularly Update

7) How can this system scale.

How can SSO System be Scaled

SSO systems can be scaled by using load balancing and clustering solutions. Load balancing solutions can be used to distribute client requests across multiple servers, while clustering solutions can be used to replicate data across multiple servers. Additionally, SSO systems can be scaled by adding more servers and increasing the capacity of the existing servers. Security measures should also be taken to ensure that the SSO system is protected from potential vulnerabilities.

8) How can we take advantage of concurrency and multi-core processors?

How can we take advantage of concurrency and multi-core processors for SSO System

* Utilize thread-level parallelism to process requests in parallel. This can be done by splitting requests up into multiple threads so that they can be handled simultaneously.
* Use a message-oriented middleware system such as ActiveMQ or RabbitMQ to queue requests and process them asynchronously.
* Use multiprocessing to spread the work over multiple cores. This can be done by running multiple processes on the same machine, each one with its own thread pool.
* Implement caching techniques to reduce the amount of database access. This can help reduce the amount of I/O operations that need to be performed and thus improve performance.
* Utilize a database cluster for scalability and availability. This can help ensure that the system can handle more requests and remain available in the event of a single database node failure.

9) Write about how you set up an agile team to build this product

how you set up an agile team to build an SSO product

* Identify the Product Vision: It’s important to clearly define the purpose of the SSO product, what it should do, and how it should be used. This will help guide the team's decisions and ensure everyone is on the same page.
* Assemble the Team: The team should include a mix of professionals with different backgrounds, including software engineers, designers, and product managers.
* Establish Collaborative Practices: Agile teams should use collaborative practices such as continuous integration, test-driven development, and version control.
* Create a Backlog: Establish a backlog of tasks and prioritize the items that need to be done first. This will help the team focus on what is most important and allow them to quickly adapt to changes.
* Plan Iterations: Break the project down into smaller iterations and decide on a timeline for each iteration. This will help the team closely track progress and maintain momentum.
* Monitor Progress: Monitor progress regularly and adjust the team’s approach as needed. This will ensure that the team is on track to complete the project on time.