AuthX: A Java-Based Framework for Authentication and Authorization Using Design Patterns

Krisha Patel
Student At Gujarat Vidyapith, Ahemdabad
Dept. of Computer Science
240160510033.cs@gujaratvidyapith.org

Abstract

*Secure user authentication and authorization are critical components of modern software systems. This paper presents **AuthX**, a Java-based framework designed to simplify and standardize the implementation of authentication and authorization mechanisms using object-oriented design principles. AuthX incorporates multiple design patterns—**Strategy**, **Factory**, **Singleton**, and **Decorator**—to promote modularity, reusability, and scalability. The framework supports essential features such as user login, multi-factor authentication (MFA) via OTP, session management, and role-based access control (RBAC). By applying design patterns, AuthX ensures loose coupling between components, making it easier to extend or modify functionality without affecting the overall system. The implementation demonstrates how established design principles can be effectively applied to build a robust and maintainable security framework for Java applications.*

Keywords: Authentication, Authorization, Java, Design Patterns, Strategy Pattern, Factory Pattern, Singleton Pattern, Decorator Pattern, Multi-Factor Authentication (MFA), Role-Based Access Control (RBAC), Object-Oriented Design.

1. Introduction

The increasing demand for secure systems has elevated the importance of robust authentication and authorization mechanisms. AuthX was developed as a framework to implement these security features using object-oriented design principles and software design patterns. The framework allows developers to integrate secure login systems, session handling, role-based access, and additional layers such as OTP.

2. Framework Modules

2.1 Authentication Mechanisms

AuthX supports multiple login types: basic username-password login and OTP-based MFA. The framework makes it easy to switch or combine methods using the Strategy pattern.

2.2 Authorization & RBAC

Role-Based Access Control (RBAC) is implemented, allowing fine-grained access to different modules and features based on user roles (e.g., Admin, User).

2.3 Session Management

The Singleton pattern is used to manage user sessions across the application, ensuring consistent behavior and resource access.

2.4 UI Styling with Decorator

AuthX allows dynamic customization of the login UI using the Decorator pattern. Features like light/dark modes and can be plugged in at runtime.

3. Implementation

Core Interfaces and Classes

`IAuthStrategy` – Interface for all authentication strategies.

`BasicAuth`, `MFAAuth` – Concrete strategy classes.

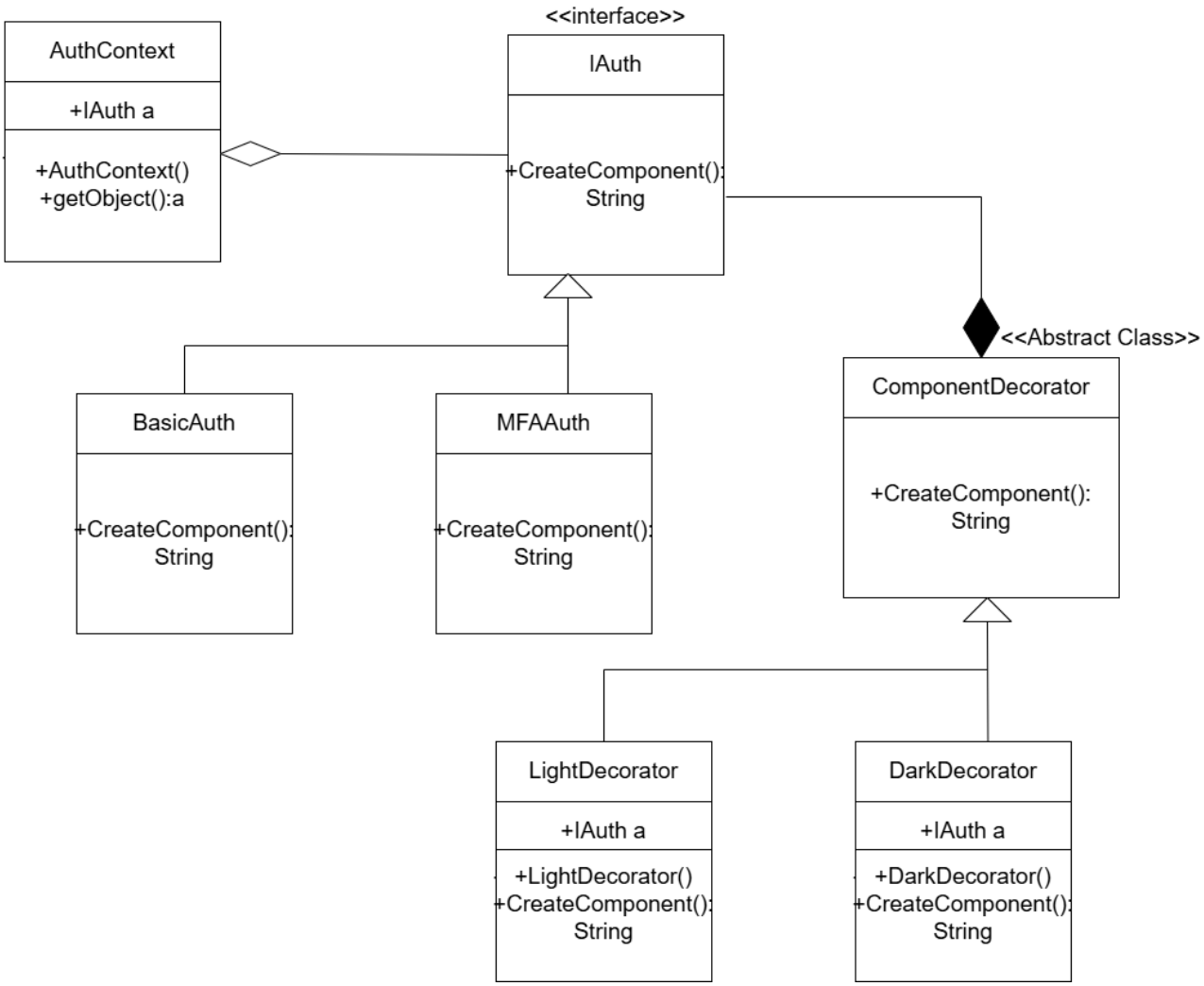
`AuthFactory` – Factory class to return the appropriate strategy.

`SessionManager` – Singleton managing login sessions.

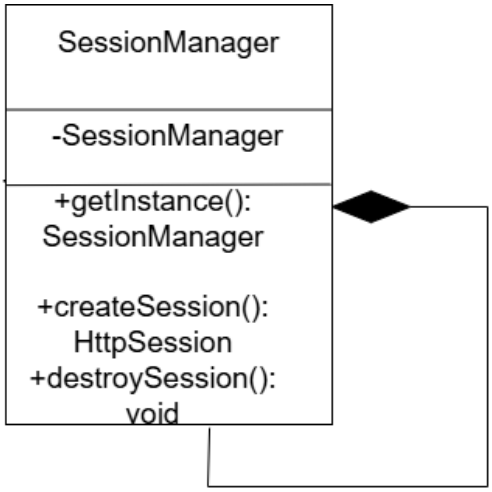
`ComponentDecorator` – For UI security enhancement.

Class Diagram

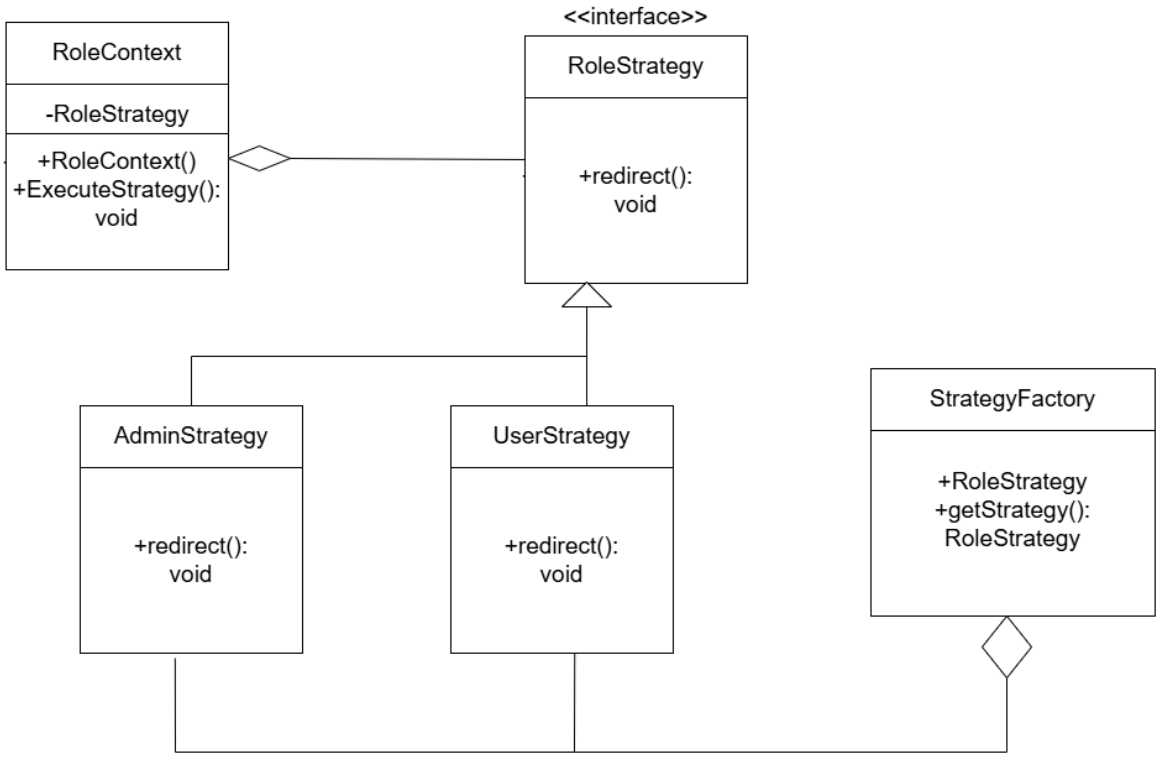
i)Authentication & Decoration class Diagram



ii)Session Management



iii) Authorization



Code Snippet Example

i)Authentication

```
//Strategy Interface
public interface IAuth {
   public String createComponent();
}

//Concerete Strategies
public class BasicAuth implements IAuth{
   @Override
   public String createComponent(){
        return “ComponentString”;
    }
}

public class MFAAuth implements IAuth{
   @Override
   public String createComponent(){
        return “ComponentStringWithOTP”;
    }
}
```

```
//Context
public class AuthContext {
   IAuth a;
   public AuthContext(IAuth a) {
      this.a = a;
   }
   public IAuth getComponentObject() {
      return a;
   }
}
```

4. Use Case Scenarios

User registers and logs in with a password.Admin and User can access their respective dashboards according to th role. A user is prompted for OTP after password login.

5. Conclusion

This paper demonstrates how AuthX leverages well-known design patterns to build a maintainable and extensible security framework. While developed for academic purposes, the structure and modularity of AuthX can serve as a foundation for real-world secure Java applications.

7. References

Parikh, A. D., & Buddhdev, B. V. (2008). *E-Governance solution based on observer design pattern*. National Conference on Architecturing Future IT Systems (NCAFIS'08), Devi Ahilya University, Indore. Retrieved from https://www.researchgate.net/publication/249313456

Gamma, Erich, Richard Helm, Ralph Johnson, and John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994*