**AuthFlow**

**Authenticator**

**Software Requirements Document**

**1. Introduction**

**1.1 Purpose**

The purpose of this document is to outline the software requirements for the development of an Authenticator Application. This application aims to enhance security through time-based one-time passwords (TOTP) for multi-factor user authentication.

**1.2 Scope**

The Authenticator App will enable users to associate their accounts with the app, generate time-based codes, and use these codes for secure login. The document provides an overview of high-level requirements, project glossary, and domain model.

**2. High-Level Requirements**

1. **User Registration and Setup**
   * Users can register their accounts on the web app.
   * During the setup, users associate their accounts by scanning a QR code or manually entering a secret key.
2. **Shared Secret Management**
   * The app generates a unique shared secret for each user during setup.
   * The shared secret is securely stored on the user's device and exchanged securely between the app and the server.
3. **Code Generation**
   * The app continuously generates time-based codes using the TOTP algorithm and the shared secret.
   * Codes are six or eight digits long and updated at regular intervals (30 to 90 seconds).
4. **Code Presentation**
   * Users can view the current time-based code associated with their account in the app.
5. **User Authentication**
   * During login attempts, users are prompted to enter the current code displayed in their app.
   * The server independently generates a code based on the shared secret and compares it to the user-entered code for authentication.
6. **Code Expiration and Renewal**
   * The generated code has a limited validity period (e.g., 30 seconds).
   * After expiration, users need to generate and enter a new code for subsequent login attempts.
7. **Security Measures**
   * Implement secure storage of shared secrets on both the server and user devices.
   * Utilize HTTPS for encrypted data transmission between client and server.
8. **User Recovery**
   * Implement a secure recovery mechanism for users who lose access to their authenticator app.
9. **User Interface**
   * Design a user-friendly interface for account registration, setup, and code presentation.
   * Ensure mobile responsiveness for a seamless user experience.
10. **Error Handling and Feedback**
    * Implement robust error handling for invalid codes, expired codes, and other potential issues.
    * Provide clear and informative feedback to users during the authentication process.

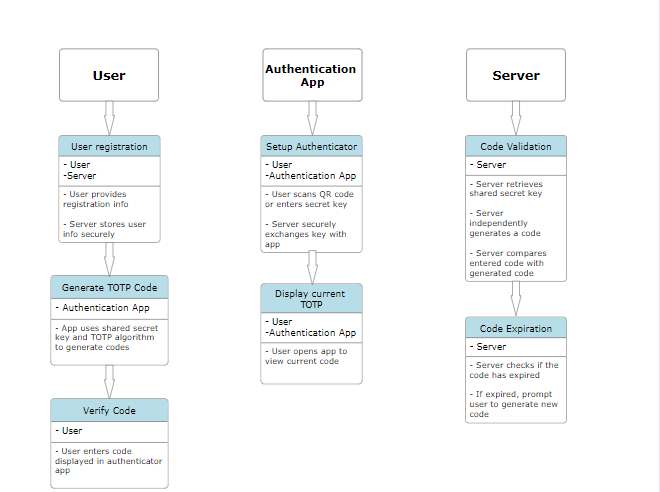
**3. Project Glossary**

* **TOTP:** Time-based One-Time Password.
* **QR Code:** A two-dimensional barcode used for efficient data encoding.
* **Shared Secret:** A unique code synchronized between the app and the server.
* **Authentication:** The process of verifying the identity of a user.
* **HTTPS:** Hypertext Transfer Protocol Secure, ensuring encrypted communication.
* **User Recovery:** The process of regaining access to an account after losing access to the authenticator app.

**4. Domain Model**

* **User:** Represents an individual with an account on the Authenticator App.
* **Authenticator App:** The application is responsible for generating and managing time-based codes.
* **Server:** Manages user authentication, stores shared secrets, and verifies codes.
* **Shared Secret:** A unique secret key shared between the app and the server for code generation.

**5. Use Case Diagram**

****

Actors:

User

Authenticator App

Server

Use Cases:

User Registration

Setup Authenticator (Scan QR Code/Enter Secret Key)

Generate TOTP Code

Display Current TOTP Code

User Login

Verify Code

Code Validation

Code Expiration

Associations:

User associates with "Setup Authenticator"

Authenticator App associates with "Generate TOTP Code" and "Display Current TOTP Code"

Server associates with "User Registration," "User Login," "Verify Code," "Code Validation," and "Code Expiration"

Use Cases:

User Registration:

Description: Users register their accounts on the platform.

Actors: User, Server

Preconditions: User is not registered.

Flow:

User provides necessary registration information.

Server stores user information securely.

Setup Authenticator:

Description: Users associate their accounts with the authenticator app.

Actors: User, Authenticator App, Server

Preconditions: User is registered.

Flow:

User scans QR code or manually enters a secret key.

Server securely exchanges the secret key with the app.

Generate TOTP Code:

Description: Authenticator continuously generates time-based codes.

Actors: Authenticator App

Preconditions: Authenticator is set up.

Flow:

App uses shared secret and TOTP algorithm to generate codes.

Display Current TOTP Code:

Description: User views the current code associated with their account.

Actors: User, Authenticator App

Preconditions: Authenticator is set up.

Flow:

User opens the app to view the current code.

User Login:

Description: User attempts to log in.

Actors: User

Preconditions: User is registered and authenticator is set up.

Flow:

User enters login credentials.

Verify Code:

Description: User is prompted to enter the current code.

Actors: User

Preconditions: User is logging in.

Flow:

User enters the code displayed in the authenticator app.

Code Validation:

Description: Server validates the entered code.

Actors: Server

Preconditions: User entered a code.

Flow:

Server retrieves shared secret.

Server independently generates a code.

Server compares the entered code with the generated code.

Code Expiration:

Description: Codes have a limited validity period.

Actors: Server

Preconditions: User attempted login.

Flow:

Server checks if the code has expired.

If expired, prompt the user to generate a new code.

GUI Prototype:

A screenshot of a login screen

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

STORYBOARD  
  
  
A screenshot of a computer

Description automatically generated