Secure authentication system

By members:

Aruzhan Narikbaeva

Nuray Kenzhegulova

Marzhan Abdibaeva

Introduction to

Secure authentication system

Search

Project Overview: description of the Secure Authentication System.

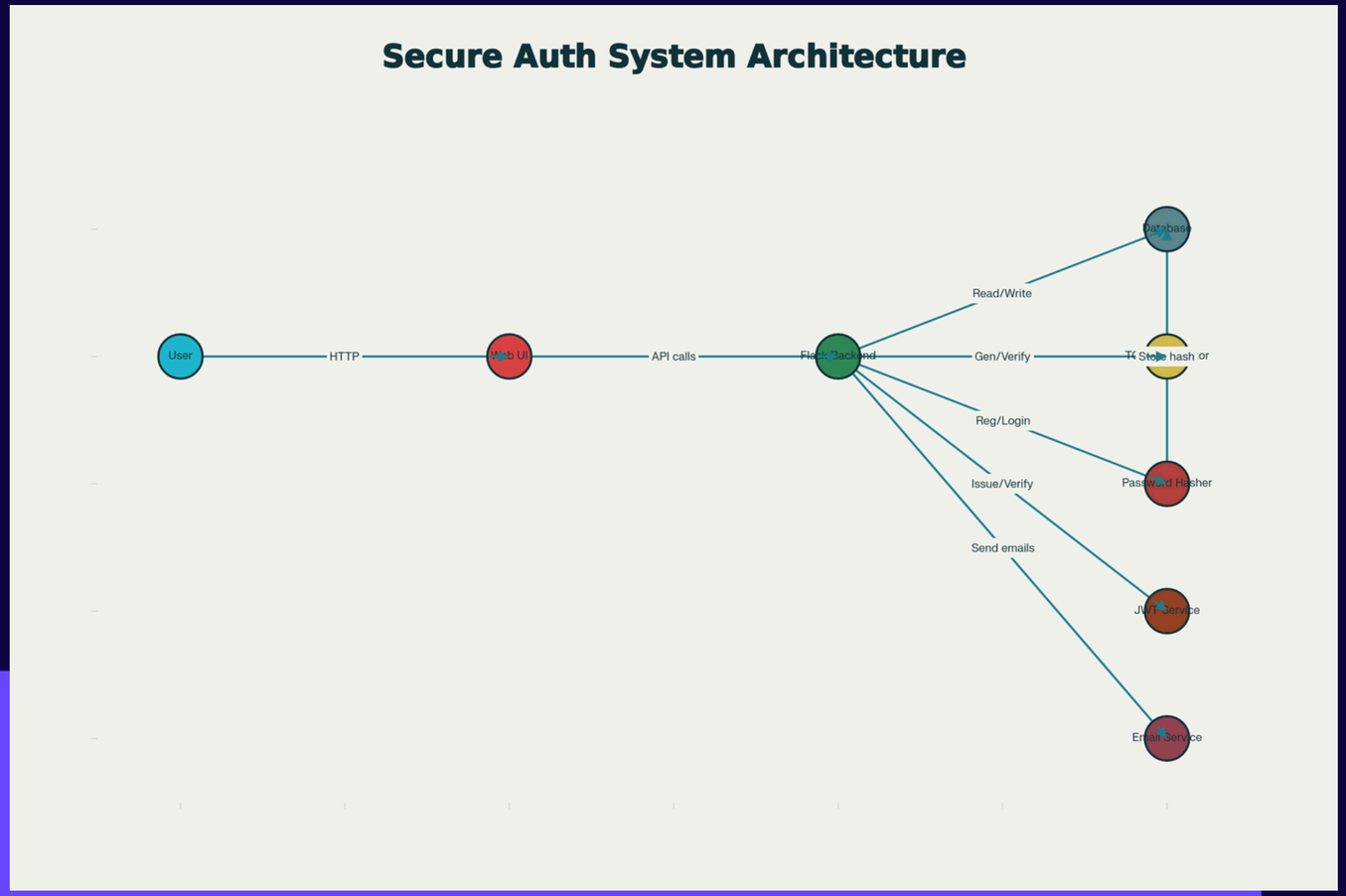
- Our project implements a Secure Authentication System that integrates multi-factor authentication (MFA), password hashing, JWT token management, and account recovery features.

Problem Statement:

- We address the issue of weak authentication and password breaches by combining password security with time-based one-time passwords (TOTP), as well as providing account recovery and password reset features.

Architecture

Search



Cryptocurrency transactions are recorded on a blockchain, a distributed ledger maintained by a network of computers. This ensures transparency, as every transaction is visible and immutable.

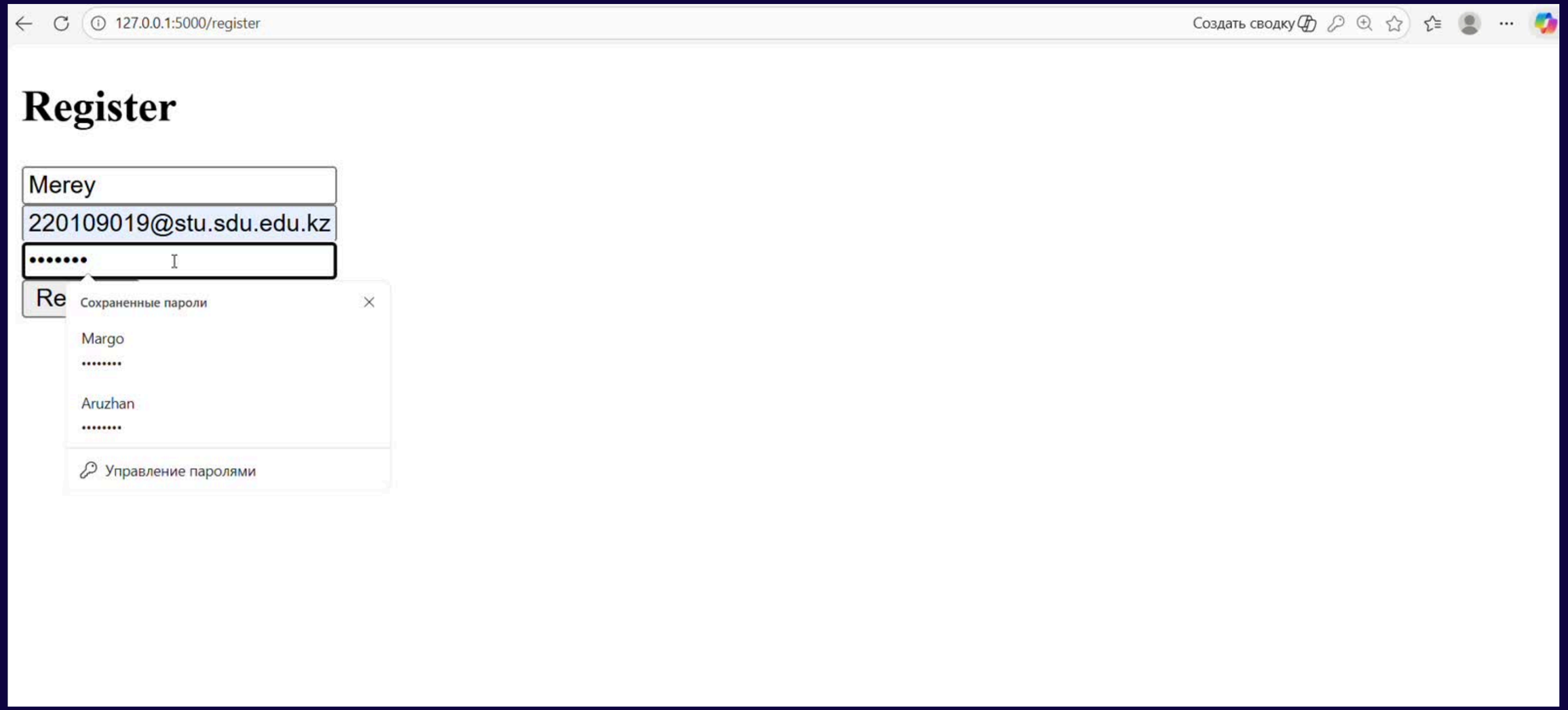
Cryptographic techniques secure these transactions, ensuring data integrity and user anonymity.

Cryptographic Components

Search

Cryptographic Components

1. Password Hashing (bcrypt)
   - Function: Passwords are securely hashed using bcrypt, a strong and widely-used hashing algorithm. This ensures that user passwords are stored in an irreversible format, enhancing security.
   - Implementation: We use the bcrypt library to hash passwords before storing them and verify them during login.
2. Time-based One-Time Password (TOTP)
   - Function: Adds an extra layer of security through two-factor authentication (2FA). A time-based one-time password is generated and verified during login.
   - Implementation: We use the pyotp library to generate and verify 6-digit TOTP codes, which are time-sensitive and expire every 30 seconds.
3. JSON Web Tokens (JWT)
   - Function: JWT is used for session management and secure communication between client and server. It allows the server to authenticate users without storing session data in the server.
   - Implementation: We use HMAC-SHA256 to sign the JWT token, ensuring its integrity. The token is used for user authentication during login and authorization for restricted routes.
4. Session Management (Flask-Session)
   - Function: Flask-Session is used to securely store session data on the server. Session information, such as username, email, and hashed passwords, is stored securely across requests.
   - Implementation: Flask-Session helps ensure that session data is encrypted and stored persistently across user interactions.
5. Encryption (RSA or ECDSA)
   - Function: Public-key encryption is used for secure key exchange and token signing, ensuring that sensitive data (like passwords or tokens) is not exposed during transmission.
   - Implementation: RSA or ECDSA algorithms are used for encrypting and signing JWT tokens to ensure integrity and authenticity.


127.0.0.1:5000/register
Создать сводку
Register
Merey
220109019@stu.sdu.edu.kz
Re
Сохраненные пароли
Margo
Aruzhan
Управление паролями

Conclusion

We have implemented a secure authentication system with key features such as multi-factor authentication (MFA), password hashing (bcrypt), JWT for session management, and password reset with secure tokens. The system also supports account recovery mechanisms.
The cryptographic components, including TOTP, password hashing, and JWT, ensure strong security for user data. Key challenges included integrating MFA and secure token generation, while lessons learned highlight the importance of robust security measures.