****

**a. Problem Statement**

Traditional password-based authentication is vulnerable to various attacks like phishing, brute-force, and credential stuffing. This poses a significant security risk, especially for applications handling sensitive data. Two-factor authentication (2FA) adds an extra layer of security by requiring a second factor beyond the password, making it significantly more challenging for attackers to gain access.

**b. Our solution to the problem**

This report proposes integrating 2FA into a Django application using the CodevoWeb API. CodevoWeb provides a robust and user-friendly API that simplifies the implementation of 2FA. This solution offers the following benefits:

Increased security: 2FA significantly reduces the risk of unauthorized access by requiring a second factor, such as a code sent to a user's phone or a fingerprint scan.

Improved usability: CodevoWeb's API offers various 2FA methods to cater to different user preferences, like SMS codes, authenticator apps, and security keys. We use authenticator app.

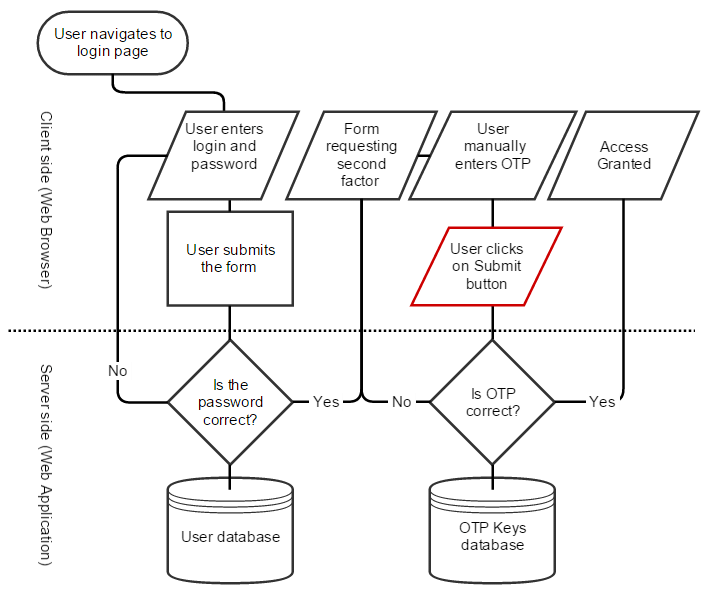
Easy integration: The CodevoWeb API is well-documented and provides Python libraries for smooth integration into Django applications.

**c. Technical details**

**Technologies used:**

* **Django:** Web framework
* **CodevoWeb API**: 2FA provider
* **Git and Github:** VCS

**Diagram:**

****

**Flow:**

* User enters username and password.
* Django sends username and password to CodevoWeb API for verification.
* If valid, CodevoWeb API sends a verification code to the user's chosen method like (SMS authentication, authenticator app etc).
* User enters the verification code.
* CodevoWeb API verifies the code and sends a success/failure response to Django.
* Django grants access if the code is valid, otherwise, denies access**.**

**Usage:**

* The application should clearly inform users about 2FA and its benefits.
* Users should be able to choose their preferred 2FA method.
* The application should provide clear instructions for using 2FA.
* The application should offer backup options for recovery in case of lost authentication tokens.

**Challenges Faced:**

* **Configuration**: Setting up the CodevoWeb API and integrating it with Django requires careful configuration and API calls.
* **Usability:** Ensuring a seamless user experience for different 2FA methods requires careful UI/UX design and error handling.
* **Security:** Implementing secure coding practices and handling potential vulnerabilities within the integration code.

**d- Future Directions or Challenges**

* Integrating more advanced 2FA methods, such as biometric authentication.
* Regularly updating the 2FA implementation to adapt to emerging security threats.
* Further refining the 2FA process for improved usability without compromising security.
* Optimizing the system for increased user base and maintaining high performance during peak loads.

**Additional Notes:**

* This report provides a high-level overview of 2FA integration in Django using CodevoWeb API.
* Specific implementation details may vary depending on the application's requirements.
* It is crucial to consult the CodevoWeb API documentation and best practices for secure integration

[Github Repo Link](https://github.com/Numeer/2-factor-authentication-while-login)