Voter Authentication System Architecture

A Microservice-Based Approach

Rares Nedelcu

March 25, 2025

1 Architecture Diagram

```
flowchart LR
    A[User] -->|Interacts with| B["Frontend (Tailwind)"]

    B -->|Sends requests to| C["API Gateway (Optional)"]

    C -->|Routes to| D["Voter Authentication Service"]
    C -->|Routes to| E["ID Card Processing Microservice"]
    C -->|Routes to| F["Selfie Verification Microservice"]
    C -->|Routes to| G["Election & Campaign Management Service"]

    D -->|Issues JWT| H["Token Store / JWT"]
    D -->|Reads/Writes| I[(Database)]

    E -->|OCR| J["Tesseract or Google Vision API"]
    F -->|Face Compare| K["AWS Rekognition / Azure Face"]
    G -->|Manages posts, elections| I

    I -->|Stores voter/user data| I
```

The above Mermaid code outlines the system's major components:

- **User (A)** interacts with the Tailwind-based frontend (B).
- The optional API Gateway (C) routes requests to the respective microservices:

  - **Voter Authentication Service (D)** for login, registration, and JWT issuance.
  - **ID Card Processing (E)** uses OCR to read uploaded ID documents.
  - **Selfie Verification (F)** compares user selfies against ID card photos.
  - **Election & Campaign Management (G)** handles election creation, campaign content, and voting records.

- A **Database (I)** stores user, election, and vote data.
- External services (J, K) are used for OCR and face recognition.

2 Detailed Explanation of Components

2.1 Frontend (Tailwind)

This component offers a responsive UI design using Tailwind CSS, possibly combined with a JavaScript framework (e.g., React or Vue). It sends user actions (e.g., registration forms, vote submissions) to the back-end microservices.

2.2 Microservices and External APIs

Voter Authentication Service handles credential checks and session tokens. **ID Card Processing Microservice** runs OCR on uploaded ID images. **Selfie Verification Microservice** compares face data using AWS Rekognition or Azure Face. **Election & Campaign Management** organizes election details, candidate campaigns, and stores user votes.

2.3 Security and Token Issuance

Authenticated sessions are maintained via JSON Web Tokens (JWTs), issued upon user login or registration. These tokens require periodic renewal to uphold security.

3 References

1. Fowler, M., & Lewis, J. (2014). *Microservices*. Retrieved from `https://martinfowler.com/articles/microservices.html`
   Summary: Explains the principles of microservice architecture, including service autonomy, decentralized governance, and continuous delivery practices.

2. Richardson, C. (2018). *Microservices Patterns*. Addison-Wesley.
   Summary: Provides design strategies and patterns for building scalable, maintainable microservice applications using Java and related frameworks.

3. Google Cloud. (n.d.). *Cloud Vision API Documentation*. Retrieved from `https://cloud.google.com/vision`
   Summary: Official documentation on using Google's OCR and image analysis services for text extraction, label detection, and more.

4. Smith, R. (2007). An overview of the Tesseract OCR engine. *Ninth International Conference on Document Analysis and Recognition* (Vol. 2, pp. 629-633). IEEE.
   Summary: A technical paper detailing the architecture, training, and performance considerations of Tesseract OCR.

5. Amazon. (n.d.). *Amazon Rekognition Developer Guide*. Retrieved from `https://docs.aws.amazon.com/rekognition/`
   Summary: Explains how to implement face recognition, object detection, and image analysis with AWS Rekognition, including best practices and code examples.

6. Microsoft. (n.d.). *Azure Face API Documentation*. Retrieved from `https://learn.microsoft.com/azure/cognitive-services/face/`
   Summary: Provides guidelines for detecting, identifying, and comparing faces using Azure Cognitive Services with REST APIs or SDKs.

7. Jones, M., Bradley, J., & Sakimura, N. (2015). *JSON Web Token (JWT)*. IETF, RFC 7519. Retrieved from `https://datatracker.ietf.org/doc/html/rfc7519`
   Summary: Defines the standard for creating secure, claim-based authentication tokens for web applications.

8. Tailwind Labs. (n.d.). *Tailwind CSS Documentation*. Retrieved from `https://tailwindcss.com/`
Summary: Official reference for using Tailwind CSS, offering utility-first styling, configuration options, and best practices for responsive design.

9. Spring Security. (n.d.). *Reference Documentation for Spring Security*. Retrieved from `https://spring.io/projects/spring-security`
Summary: Explains how to integrate Spring Security with Java-based microservices, including JWT, OAuth2, and multi-factor authentication.

10. NIST. (2019). *Digital Identity Guidelines* (Special Publication 800-63-3). Retrieved from `https://csrc.nist.gov/publications/detail/sp/800-63/3/final`
Summary: Provides guidelines on digital identity services, assurance levels, and best practices for secure authentication and proofing of identity.