06 Integrate Signup Flow with Amazon or Azure Liveness Detection libraries

Integrate the eSignet signup service with available liveness detection/face matching libraries from Amazon or Azure to enhance identity verification.

Complexity Level: Medium

Overview:

The current implementation of eSignet's Signup service lacks real liveness detection. While it accesses the device's webcam, it only captures and sends dummy frames without performing any actual verification to confirm the presence of a live user. This placeholder logic weakens the security of the identity verification process and leaves the system vulnerable to spoofing attempts.

To address this, proper liveness detection must be implemented using existing libraries that support such functionality (e.g., those offered by Amazon, Azure, etc.).

Recommended Tech Stack:

Cloud Service Integration
Liveness detection and face comparison can be implemented using available cloud libraries, such as provided by Amazon or Azure.

Exact Tasks:

Mandatory Tasks:

1. Choose and Configure Liveness Detection API

a. **Select a Cloud Provider:**
   Choose a liveness detection service such as Amazon Rekognition or Azure Face API with Liveness Detection, based on feasibility, documentation, ease of integration, and available features.
b. **Set Up Cloud Credentials and Access:**
   Create a cloud account, enable liveness APIs, and configure secure access using IAM roles or API keys, with credentials managed via environment variables or secrets managers.
c. **Understand API Capabilities and Limits:**
   Review rate limits, pricing, latency, supported media formats, and response structure to plan for efficient integration.

2. Implement Backend Service for Liveness Detection

a. **Extend Signup Service:**
   Enhance the existing identity verification plugin in the eSignet's signup service to support liveness detection.
b. **Capture and Send Input Data:**
   i. Capture a short video or sequence of frames via webcam during signup.
   ii. Format and send the data as per the selected API's requirements.
c. **Handle and Parse API Responses:**
   i. Receive the liveness score or result from the API.
   ii. Parse response payloads to extract status (e.g., isLive, confidenceScore, faceMatchResult).

3. Connect Liveness Result to Signup Flow

a. **Return Liveness Status and Implement UI Responses:**
   Securely and promptly communicate the liveness detection result to the frontend. Based on the outcome, trigger appropriate UI behavior:
   i. **Success:** Allow the user to proceed.
   ii. **Failure:** Display relevant error messages and provide a retry option.
   iii. **System Error:** Notify the user with clear instructions or guide them to an alternate verification flow.
b. **Control Signup Progression:**
   Ensure that user registration only proceeds if the liveness check is successful and meets a defined confidence threshold.

4. Integration and Demo

a. **Complete Workflow Integration:**
   i. Ensure seamless integration of liveness detection into the full signup workflow.
   ii. Validate edge cases (e.g., no face detected, poor lighting, low confidence score).

Good-to-Have Tasks

1. **Liveness Verification Timer:**
   Introduce a visible timer on the frontend during the each liveness detection step to guide users and prevent timeouts. This improves user experience and ensures timely completion of the verification process.
2. **Timeouts:**
   A default timer will be set for each step. If the user exceeds the allotted time, they must restart the process from that step.
3. **Real-Time Feedback on Camera Input:**
   Provide real-time feedback while the user is being captured (e.g., “Face not detected,” “Too far,” “Too dark”) to help users adjust their camera positioning or lighting conditions.

Bonus Tasks:

1. **Implement Face Matching**
   As an added layer of security, implement face matching to verify that the live user matches the individual on a presented government-issued ID document. This step complements liveness detection and helps prevent identity spoofing.
2. **Prompt for ID Document Upload or Capture:**
   After successful liveness verification, prompt the user to either upload or capture an image of their official ID (e.g., driver's license, passport).
3. **Extract Face from ID Document:**
   Utilize existing cloud libraries to detect and extract the face image from the uploaded ID document. Ensure proper preprocessing (e.g., orientation correction, cropping) to enhance accuracy.
4. **Perform Face Comparison:**
   Use the selected provider’s face comparison API to match the extracted face from the ID with the live face captured during liveness detection.
5. **Configure Matching Threshold and Accuracy Controls:**
   Set an appropriate confidence threshold (e.g., 90%) to determine a successful match. Implement logic to handle borderline cases, false positives, and false negatives gracefully.
6. **Handle Outcome in Signup Flow:**
   Allow the user to proceed with registration only if the face match is successful. In case of a mismatch, provide an informative message and options to retry or contact support.

Deliverables

Participants are required to submit the following comprehensive deliverables to successfully complete the challenge:

1. Signup portal with liveness detection support:
A complete and thoroughly tested signup application that includes:

a. Integration with any available liveness detection library, such as Amazon Rekognition or Azure Face API, for real liveness verification.
b. A working signup flow that only allows registration upon successful liveness verification.
c. Frontend webcam capture functionality, with support for real-time camera feedback and user guidance.

2. Demonstration & Code

a. **Demonstration Video** – Share a video demonstration that showcases the working solution end-to-end.
b. **Source Code Repository** – Provide access to the full source code through a public repository (GitHub/GitLab).
c. **Presentation Deck** – Include a PPT summarizing the approach, results, and potential impact of the solution.

3. Documentation

a. **Setup and Installation Guide** – Step-by-step instructions to build, deploy, and run the application locally or on supported devices.
b. **Architecture Overview** – Description of the overall system architecture, data flow, and storage strategy.
c. **Technology Stack** – Explanation of all technologies, frameworks, and tools used in the solution.
d. **Features Summary** – Concise summary of all implemented features, both required and additional.
e. **Assumptions** – Provide a detailed list of assumptions made during development.
f. **API Documentation** – Detailed description of the new API endpoints developed as part of the solution.

Resources:

1. Signup Service - GitHub Link
2. Self Registration Portal for generating UIN: Link
3. Signup integration guides: Link
4. eSignet Collab user guide: Link