Table of Contents

SDK Integration

Github Link: https://github.com/rizkypascal/simple_onboarding_app

Architectural Analysis

Three key observations or implementation decisions a customer must make:

- Avoid consuming the iProov backend API directly from UI platforms such as Web, Android, or iOS. Doing so increases project complexity by adding extra steps like payload transformation, serialization and deserialization, error handling, rate limiting, and exponential backoff retries. This can degrade application responsiveness and introduce unnecessary issues. Instead, always use the SDK to ensure a platform-standardized and concise codebase, as the SDK is designed to fulfill the required functionalities efficiently
- You must develop an API layer aligned with MACH principles (Microservices, API-first, Cloud-native, and Headless) to effectively manage varying levels of user traffic and throughput. All traffic must be secured using HTTPS, SSL, or TLS to prevent man-in-the-middle attacks. This API layer should handle tasks such as request sanitation, logging, data provisioning for analytics, and other processes that support business decision-making and issue tracking. There are two key processes require interaction with the iProov backend API:
    - Generating a token to initialize the app SDK for video streaming
    - Verifying the validation result

  Both processes should be managed within your API layer. Additionally, ensure backward compatibility with the iProov backend API to accommodate future updates. Furthermore, if certain workflows can be handled asynchronously, consider adopting an event-driven architecture to optimize traffic flow and prevent quota or rate limit issues with the iProov backend API

- Always adhere to compliance, regulations, and industry standards, as IT and system audits are conducted annually. Your system's integration with third parties will be assessed by auditors, so implementations must align with security standards such as ISO/IEC 27xxx. These guidelines ensure that, at a minimum, data encryption is performed if data is stored and that personally identifiable information (PII) is not exposed. Additionally, since every API integration requires authentication, ensure that API authentication attributes, such as API secrets and keys, are stored in a highly secure environment. Utilize solutions like Secret Managers from AWS, GCP, or other Cloud Service Providers instead of embedding API credentials at the project or code level.

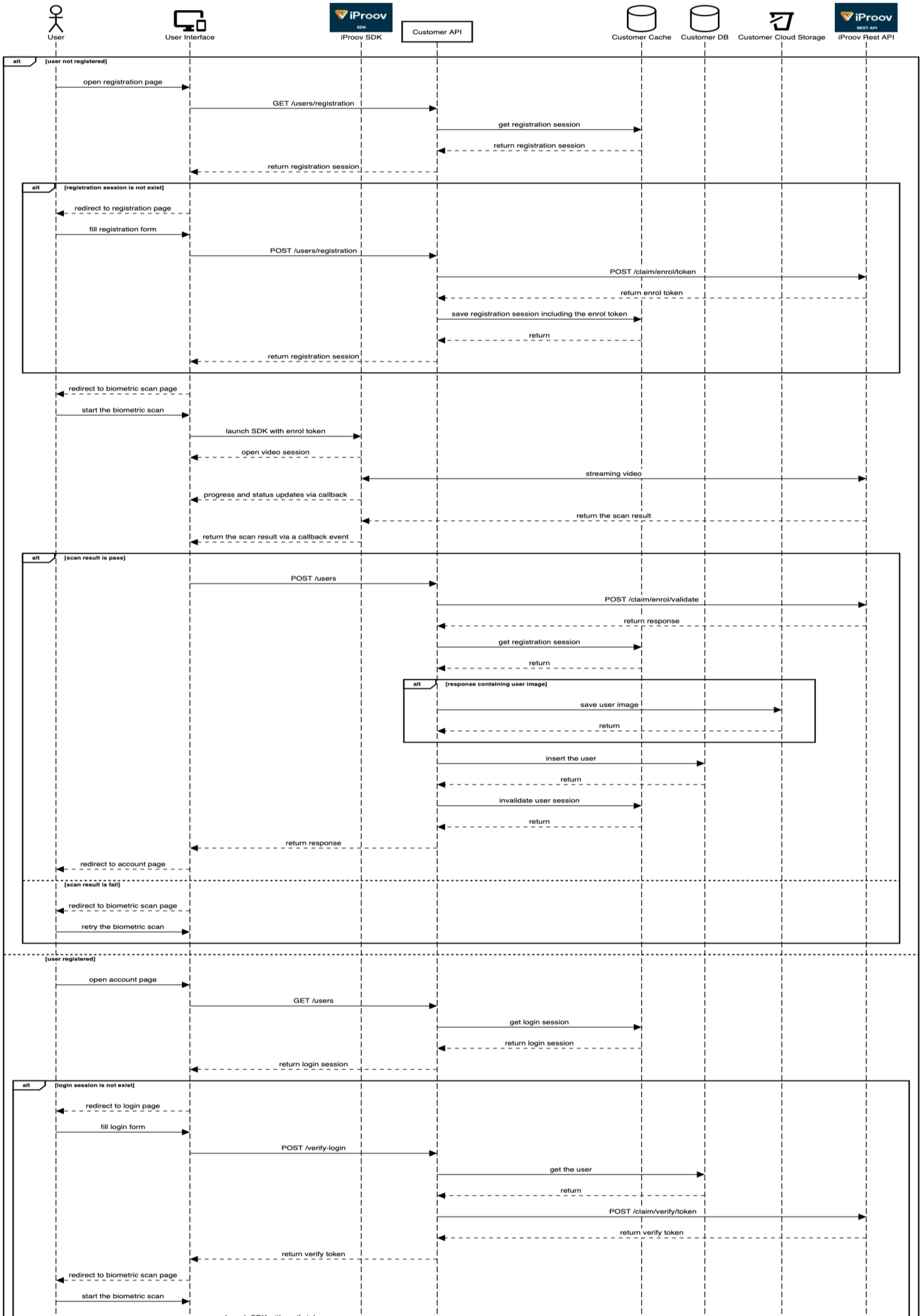
Follow-up questions for the customer regarding architectural decisions:

- Do you intend to use a reverse proxy during the verification session? If so, you may need to take additional steps to configure your own certificate list
- Are you familiar with security standards at the API level, particularly in mitigating the top 10 OWASP vulnerabilities? Implementing proper security measures is crucial, as DDoS attacks can lead to rate limit issues, increased costs, and disruptions in the verification and validation process for users
- Are you familiar with cloud deployment and event-driven architecture? Scalability is essential for handling simultaneous user requests efficiently and preventing rate limit issues
- Are you operating in multiple regions? If so, you may need to comply with each region’s cybersecurity laws and configure your SDK to support the respective languages. Additionally, iProov may not meet these requirements in certain regions, so it’s important to take this into account

Technical Diagrams

Proposed fundamental customer journey for registration and login with iProov verification support.

User Registration and Login Process with iProov Feature



Appendix

Sequence diagram script: https://github.com/rizkypascal/simple_onboarding_app?tab=readme-ov-file#documentation