AtlantisGate: Technical Strategy

AtlantisGate Project Overview

The mission is to leverage cutting-edge technologies and methodologies to create a secure, user-friendly, and efficient platform. This platform will house the digital identities of over 100,000 citizens, acting as the cornerstone for employment and citizen verification processes.

Discovery *(Thanks ChatGPT for being AtlantisGate Client AI)*

Legacy Tech Stack

Our legacy system, which we refer to as the "FALCON" tech stack, is a critical part of our existing infrastructure. The FALCON tech stack comprises the following components:

Database: The legacy system uses a relational database management system, specifically PostgreSQL, to store and manage verification data. This database contains information related to citizen verification, employment records, and other relevant data for our 100,000 citizens.

Application Layer: The application layer of the FALCON stack is primarily built using Java-based technologies. It includes custom-built APIs and services that are responsible for handling requests related to verification and data management.

API Integration: Third-party integration with the legacy system is primarily achieved through RESTful APIs. These APIs provide secure endpoints for external parties to interact with our system. Secure API keys and authentication mechanisms are used to ensure secure data exchange.

Data Format: The system supports both JSON and XML data formats for communication. This flexibility allows third parties to choose the format that best suits their integration needs.

Security Measures: Security is a top priority in the legacy system. It employs industry-standard encryption protocols and follows OWASP guidelines for web application security. Access to the system is tightly controlled, and user authentication is mandatory for all interactions.

Data Privacy: We strictly adhere to data privacy regulations and ensure that sensitive information is handled with the utmost care. The legacy system complies with AtlantisGate's data protection policies.

As we transition to the new digital verification platform, we aim to maintain and enhance the security measures and data handling capabilities. The strategic roadmap will include a plan for migrating the existing data from the legacy system to the new platform while ensuring data integrity and security throughout the process.

Current data footprint & growth forecast

As of the current state of our legacy system, each citizen record in the database contains the following details:

Number of Columns: There are approximately 25 columns in the database for each citizen record. These columns include personal identification information, employment history, citizenship status, and other relevant data points.

Size: On average, the size of each citizen record can vary, but it typically falls within the range of 10 to 20 kilobytes (KB) of data. The size may vary depending on the completeness of the citizen's information and the number of employment records associated with them.

Please note that these figures are approximate and may vary slightly based on individual citizen records and the data they contain. The digital verification platform will need to accommodate and

manage this data efficiently while ensuring scalability for the growing number of citizens and records.

With this population growth rate, the target data size at the end of 3 years would be approximately 10.99 gigabytes in addition to the current data size.
Currently, with approximately 100,000 citizens and an average of 25 columns per citizen record, the total data size in the database would be roughly:
100,000 citizens * 25 columns per citizen = 2,500,000 columns
Assuming an average data size of 15 kilobytes (KB) per citizen record, the current data size would be approximately:
2,500,000 columns * 15 KB per column = 37,500,000 KB
To convert this into more commonly used units:
37,500,000 KB / 1,024 = 36,621.09 MB (approximately 36.6 gigabytes)
Regarding the target size at the end of 3 years of population growth, this would depend on various factors such as the rate of population growth and the average data size per citizen. If we assume a conservative estimate of an additional 10,000 citizens per year, we can estimate the data size as follows:
10,000 new citizens * 25 columns per citizen * 15 KB per column * 3 years = 11,250,000 KB
To convert this into more commonly used units:
11,250,000 KB / 1,024 = 10,986.33 MB (approximately 10.99 gigabytes)
So, with this population growth rate, the target data size at the end of 3 years would be approximately 10.99 gigabytes in addition to the current data size. This is a simplified estimate and may vary based on actual population growth and data management practices. The digital verification platform should be designed with scalability in mind to accommodate this growth.

Methods of Identify Verification

At the launch of the digital verification platform for AtlantisGate, we should support a variety of methods to ensure comprehensive and reliable citizen verification. These methods include:

Biometric Verification: Implement biometric authentication via facial recognition to uniquely identify citizens.
Document Verification: Support the verification of official documents, such as passports, national IDs, and driver's licenses, to confirm the identity of citizens.
Employment Verification: Verify citizens' employment status, including their job history and current employment, to support their eligibility for services and benefits.

System Requirements

1. This ecosystem aims to facilitate seamless employment verification and citizen services, ensuring every individual's smooth transition into the society of AtlantisGate.

2. Digital Verification System (Codename: CitVerify): The part of the platform that will house the digital identities of over 100,000 citizens, acting as the cornerstone for employment and citizen verification processes.
3. Citizen Connect (Codename: CitConnect): The part of the platform that will provide verification services for a particular citizen or or to a set of organizations to verify various approved aspects about a citizen.
4. Assuming +100,000 population growth over 3 years, there will be approximately 10GB over 3 years for a total of about ~50GB.

Digital Verification Requirements

Functional Requirements

1. Ability to verify a user based on a set of uploaded documents from a website or mobile app AND facial recognition.
2. Ability to verify a user based on a set of uploaded documents AND employment history.

Data Discovery

1. Identify the current citizen data lives
    a. Employment data
    b. Document data
    c. Biometric data, if any
2. How often is this data refreshed and/or updated?

Backend Requirements

1. Must complete API design with FE team before building
2. Both verification methods must be available through secured persistent connections such as Websockets, XMPP, WebRTC, etc
3. Implement facial detection with liveness detection
    a. Must be deployed to iOS and Android (most support versions within reason)
    b. Must be deployed to webcam through the web as well
    c. Must implement Liveness Detection (blink detection, face movement, etc) to protect against fraud.
        i. Must have a level of confidence setting that can be configured and changed at any time (ex 0-1, 1 being confident, 0.1 being not confident)
        ii. Metadata for the score must be persisted against each verification attempt for historical purposes

4. Implement text recognition of document uploads from web, mobile phones using standard file upload document formats (pdf, jpn, png, tiff, etc)
5. Implement Message Queue for scaling and persistence / re-try

Front End / App Requirements

1. Flows
    a. User Registration / Login
        i. Additional Authentication Methods where applicable 2FA, Face ID etc
        ii. Email or SMS confirmation
    b. Citizen Verification - The user to tap on “Verify Identity” to start the process
        i. Prompted for either a Doc Upload or Biometric Verification
        ii. Doc upload flow
            1. Upload one of: passport, driver's license or naturalization card
            2. Enter the last 20 years of employment verification
        iii. Biometric Verification flow
            1. Upload one of: passport, drivers license or naturalization card
            2. Open the camera of the device or laptop/desktop and proceed to live detection algorithms
        iv. At the end of each process, the platform must present user with a clear Success or Failure message with avenues to try again to contact help.
    c. Privacy, TOS, GDPR-related stuff

Machine Learning + Fraud API

ML Requirements

1. Identify/acquire a classified data set of fraudulent and non-fraudulent data
2. Sanitize data
3. Create Feature data and split sets (training, validation, test sets)
4. Select appropriate model such as logistic regression, random forests etc
5. Train model to be evaluated against:
    a. Accuracy
    b. Precision
    c. Recall
    d. F1-score
    e. AUC-ROC
6. Deploy model to API

Fraud API + Cron Requirements

1. Create cron jobs to check the health of the model
    a. Data Drift
    b. Automated Alerts / Cloudwatch for score performance
2. Create API to expose model to Application Layer

General Backend APIs for Platform

Requirements

1. User registration + confirmation
2. User login
3. Password Reset
4. Profile updates
5. Identify Verification API implementation

CI/CD Requirements

Requirements

1. Base AWS containerization to start development
    a. Environments
        i. Local
        ii. Test (QA)
        iii. Stage (QA)
        iv. Production
2. Setup Jenkins automation server
3. Setup Terraform as infra as code service
4. Ensure all Github repos are captured
5. Configure Jenkins to perform tests (unit + integration)
6. Install SonarQube for code quality and security
7. Integrate artifact factory (pip since this is python)
8. Env management (Terraform)
9. Deploy to prod (Code Deploy, Terraform)
10. Testing CI/CD
    a. Vagrant/Docker simulations
    b. Failure simulations (builds, tests, deploys, load, etc)
11. Disaster recovery

Database Engineering / Security Team

Requirements

1. Encrypt database in transit
2. Encrypt database at rest
3. Data Masking strategies w/ CICD teams
4. Logging with PII scrubbing
5. ACL controls throughout tech stack
6. Disaster recovery and Data Retention Policies
7. Leverage Generative AI to create data sets for pseudo PII creation

Database Migration

Based on the growth scale of what that client AI said, we don't need to do massive data migration for the PSQL database. We can scale it horizontally, at worst.

Design/UI/UX

Requirements

1. Construct wireframes and a high-fidelity for
    a. iOS App
    b. Android App
    c. Website
2. Mobile responsive
3. Generate a design specification detailing the design principles adhered to, user flows, and feedback incorporation.
4. Deliver specs to Front End team
5. Generate Style Guide for the platform
6. Leverage Generative AI internal tooling to generate UI wireframes based on user feedback

Mobile App Development - iOS and Android

Requirements

1. User registration + confirmation
2. User login
3. Password Reset
4. Profile updates
5. Identify Verification API implementation

QA

Requirements

1. Unit tests
2. Blackbox tests
3. Integration tests
4. Leverage Generative AI to :
    a. Create fresh test data for test hardnesses and seeding
    b. Create UI test cases based on a requirements documents
    c. Create unit tests based on code in a repo

Resource Allocation and Budget

Per Cilent GPT, here's my budget:
A rough estimate for a project of this nature, considering the need for advanced security measures, AI/ML integration, user interface design, and database development, could range from $1.5 million to $3 million. This budget would cover development, infrastructure costs, testing, and project management over 2 months.

Assumptions

1. $3M budget and 2 month timeline
2. Budget applies to only engineering team resources, not infrastructure, software, recurring costs etc.
3. Blended rate model after negotiations with vendors

ASSUMPTIONS	
Budget applies to resources only	
Budget	*$3,000,000*
Months to Complete	*2*
Resources	**Quantity**
Full Stack Python	4
Data Engineers	2
ML Engineers	2
iOS	2
Android	2
React	2
Design/UI/UX	3
Devops/IT	3
Project Manager	1
QA	4
# of Resources	25
Blended Rate/Hr	$175
$ Per Day	$35,000
$ Per Month	$758,333
Budget	$3,000,000
Development Resources	$ (1,516,667)
Remaining for padding	$ 1,483,333

Risks

Regulatory Changes

- Mitigation
    - Stay on top of news like “Biden Targets Artificial Intelligence in Broad Regulation Order” https://apple.news/AbQDU10sYQpGUJD-PvJXiiA released Oct 30 2023 and adjust accordingly if needed

Data Security and Privacy

- Mitigation:

1. Implement strong encryption for data at rest and in transit.
2. Establish strict access controls and role-based permissions.
3. Conduct regular security audits and penetration testing.
4. Monitor data access and changes using comprehensive logs.
5. PII data masking, controls, ACL, DRP

Technology Integration Challenges

- Mitigation:
    1. Rigorously test and validate technology integrations.
    2. Maintain open communication with technology partners.
    3. Establish fallback plans in case of integration failures.

Data Quality Issues

- Mitigation:
    1. Implement data quality controls and validation checks.
    2. Conduct data cleansing and validation before ingestion.
    3. Monitor and address data quality issues in real-time.

Resource Constraints

- Mitigation:
    1. Continuously monitor resource allocation and adjust as needed.
    2. Cross-train team members to fill resource gaps - especially the full-stack developers early on

Scope Creep

- Mitigation:
    1. Establish a change control process to manage scope changes IF major creep becomes an issue, otherwise must stay agile and flexible.
    2. Clearly define project requirements and deliverables upfront
    3. Communicate the importance of adhering to the defined scope to all project stakeholders.

Data Breaches and Cybersecurity Threats

- Mitigation:

1. Regularly update and patch software to protect against vulnerabilities as part of CI/CD
2. Educate all team members on cybersecurity best practices.
3. Develop and test an incident response plan to address data breaches - Red/Blue team

Third-Party Dependencies

- Mitigation:
    1. Carefully assess and select reliable third-party vendors.
    2. Establish Service Level Agreements (SLAs) with vendors.
    3. Maintain contingency plans in case third-party services fail.

Team Member Attrition

- Mitigation:
    1. Implement a knowledge sharing system to preserve institutional knowledge.
    2. Develop a succession plan for key team roles.
    3. Foster a positive work environment to retain team members.

Inadequate Testing and QA

- Mitigation:
    1. Implement a rigorous testing plan that covers all aspects of the project.
    2. Conduct user acceptance testing (UAT) to validate functionality.
    3. Perform load and stress testing frequently to ensure system scalability.

Delays in Agile Development

- Mitigation:
    1. Continuously review and adjust sprint plans and priorities. Daily standups.
    2. Maintain close communication between cross-functional teams.
    3. Kanban FTW