

# Secura

*"Reinforcing digital identity security"*

In a digital-first world, data breaches are rapidly increasing and the continuous evolution of attacking and identity fraud tools is witnessed especially with the revolution of AI. Therefore, the password is becoming the weakest link, the current authenticator apps are phishable, SIMs are interceptable and static biometrics can be spoofed with a photo all leading to catastrophic data leaks and financial losses in healthcare, finance, defence and other sectors that deal with sensitive data.

Introducing our solution "Secura", an intelligent multi-layered MFA backend system that goes beyond traditional security. We combine polymorphic hashed splitted passwords, liveness-checked encrypted facial embeddings and cognitive OTP all within a clear user friendly experience. Through which we transform default hashed password storage into computational intensive puzzles for attackers. With the cognitive OTP we replace static and vulnerable codes to be stolen, with a challenge that requires human presence and engagement and through the liveness-checked biometric scanner we ensure that a real person is behind the screen not a photo or a video. This isn't multi-factor, it is a multi-dimensional security approach designed to obstruct attacks like SIM swapping, phishing, biometric spoofing and the various password attacks.

**Our vision** is to become the standard authentication layer for any entity that deals with sensitive data, setting a new benchmark for security and redefining identity security across various interactions in digital spaces.

## Core Key Value Propositions

For Technical Architects: Going beyond basic hashing, the system is engineered to make stolen data useless. This is achieved by sharding the passwords and applying polymorphic hashing through independently hashing password halves with different algorithms and recommended PBKDF2 iteration counts, thus making password cracking computationally intensive. Extending the principle of turning intercepted/hijacked data useless to the cognitive MFA layer where a hijacked one-time code is useless without the session-specific challenge asking the user to enter specific random digits from the OTP, thus neutralizing phishing and interception attacks. The same principle is applied to the biometric MFA layer where images are never stored raw instead the embeddings are encrypted with Fernet encryption before storage achieving confidentiality & integrity, in addition to the liveness-checks applied to defeat spoofing attacks.

For Business leaders & Enterprises: Directly targeting business's biggest digital risks which are financial losses resulted by accounts takeovers and reputational damages from data breaches. By eliminating the primary vectors of these threats; credential stuffing, phishing, etc and ensuring that even if data is stolen, it cannot be used to compromise user accounts, we protect your revenue and brand's trust. Our architecture is built for compliance, providing clear evidence for auditors that customer data is protected by the strongest available authentication measures simplifying the challenging regulation processes.

For End Users: Log in with confidence not complexity, your choice is between solving a small interactive puzzle or using your face which is a key that can't be stolen or forgotten. By following clear, guided steps; you're putting a stopper to attackers, utilizing high measures of security that work for you, not against your experience. All the complexity happens behind the scenes, giving you a simpler, safer way to access your accounts with peace of mind.

# MVP Technical Documentation

The MVP's code is a procedural code intended to resemble (mock) the back-end system service "Secura" flow. The service gets integrated to the existing main register/login front-end portals of the entity who wants to implement the system, thus preventing front-end overhauls or major redesigns therefore a front-end page is not an integral part of the MVP yet UI/UX design specifications are planned and coded to guide front-end implementation as well as to visualize user's interaction flow with the system (shared later as pictures) emphasizing again that the user's main interaction with the system in real production is through front-end portals, CLI is for MVP source code purposes only. The current console interface (IDE code terminal) is considered the main channel of interaction with the program MVP, designed to show guidance & output in user-friendly manner using functions such as the `print_pause()` which implements timed delays between outputs to prevent information overload and enhance readability.

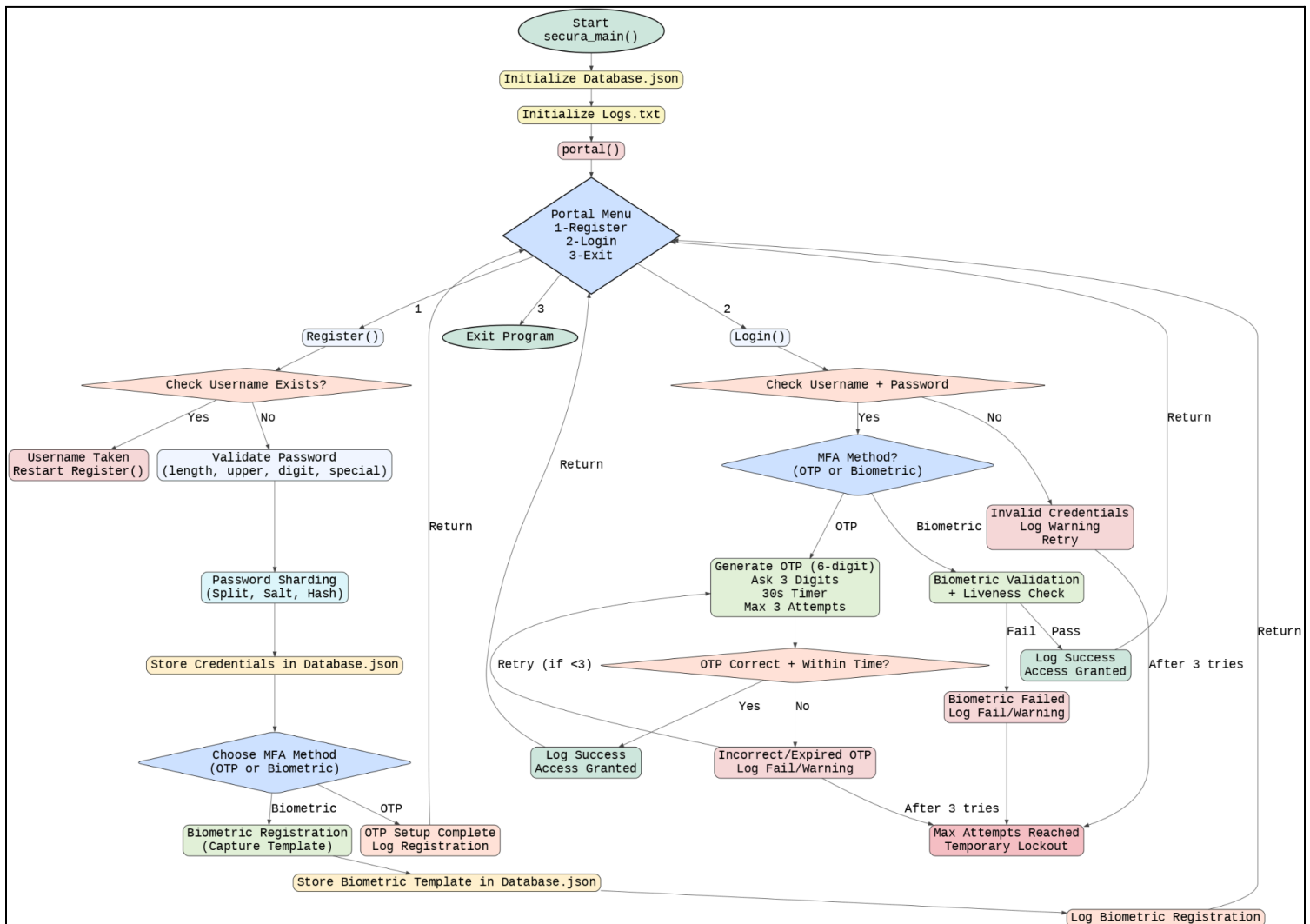
- A JSON file is auto created to mimic the database, storing user's credentials and a ".txt" file is auto created to mimic logging systems with timestamped operation logs for each event with format:  
YYYY-MM-DD HH:MM:SS [LEVEL] Message

2025-09-16	14:17:16	[FAIL] OTP expired for user 'rokaia'.
2025-09-16	14:17:26	[WARNING] Login attempt 3 with invalid credentials for 'rokaia'.
2025-09-16	14:17:26	[ERROR] Maximum login attempts reached. Access blocked.
2025-09-16	15:27:05	[INFO] New user 'tester' registered.
2025-09-16	15:27:29	[WARNING] Login attempt 1 with invalid credentials for 'rokaia'.
2025-09-16	15:27:37	[WARNING] Login attempt 2 with invalid credentials for 'tester'.
2025-09-16	15:27:44	[WARNING] Login attempt 3 with invalid credentials for 'rokaia'.
2025-09-16	15:27:44	[ERROR] Maximum login attempts reached. Access blocked.
2025-09-16	15:28:16	[SUCCESS] User 'rokaia' entered correct credentials.

```
{
  "username": [
    "hashed_shard1_hex", # PBKDF2-SHA256
    "hashed_shard2_hex", # PBKDF2-SHA512
    "salt1_hex",          # 16-byte salt
    "salt2_hex"           # 16-byte salt
  ]
}
```

### JSON file (mock database) schema

# Secura Process Flowchart



- The main entry point `secura_main()` initializes the database and logging system, orchestrates the registration and authentication flow and manages the process lifecycle

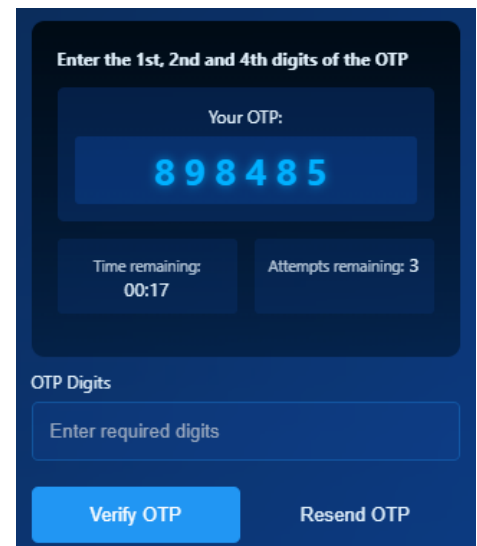
- **Password Mechanisms:** 3 functions are responsible for handling password validation, sharding & polymorphic hashing and verification

1. **password\_validation (called during registration):** takes password input by newly registered user, transforms it into string and checks its strength by validating its length, presence of digits, uppercase characters and special characters with infinite retry loop.
2. **password\_sharding (called during registration):** responsible for splitting the validated password into 2 halves (shards), adding randomly generated 16 bytes salt for each shard using the operating system's cryptographically secure random number generator "os.urandom(16)" then hashes each shard and its salt using a different hashing algorithm (SHA-256 or SHA-512) using hashlib & applying PBKDF2 according to OWASP's recommended iteration counts to each hashed shard and finally storing them as hexadecimal values in database
3. **password\_verify (called during login):** takes the password entered by the user during the login phase, rehashes it with the same approach in `password_sharding` and compares it with the stored password (shards & salts) in the database. hmac library is imported so that HMAC is only used for verification by comparison in a constant-time manner.

```
# Password sharding & polymorphic hashing function (using PBKDF2) "registration phase"
def password_sharding(password):
    half = len(password)//2
    shard1 = password[:half]
    shard2 = password[half:]
    salt1 = os.urandom(16)
    salt2 = os.urandom(16)
    # Polymorphic hashing through 2 different algorithms and iterations
    hash_shard1 = hashlib.pbkdf2_hmac("sha256", shard1.encode(), salt1, 600_000)
    hash_shard2 = hashlib.pbkdf2_hmac("sha512", shard2.encode(), salt2, 210_000)
    return hash_shard1.hex(), hash_shard2.hex(), salt1.hex(), salt2.hex()

# Password verification function (stored hashed shards) "login phase comparison"
def password_verify(password, stored_shards):
    stored_shard1, stored_shard2, hex_salt1, hex_salt2 = stored_shards
    half = len(password)//2
    shard1 = password[:half]
    shard2 = password[half:]
    salt1 = bytes.fromhex(hex_salt1)
    salt2 = bytes.fromhex(hex_salt2)
    shard1_digest = hashlib.pbkdf2_hmac("sha256", shard1.encode(), salt1, 600_000).hex()
    shard2_digest = hashlib.pbkdf2_hmac("sha512", shard2.encode(), salt2, 210_000).hex()
    return hmac.compare_digest(stored_shard1, shard1_digest) and hmac.compare_digest(stored_shard2, shard2_digest)
```

- **Cognitive OTP (MFA layer option 1):** If the user chose this as their MFA layer during registration phase, then upon logging in, a 6 random-digit number is generated with time-based expiry of 30 sec and max number of 3 attempts to verify the sent OTP. The OTP verification process is done through a puzzle appearing on the user's screen asking them to enter the digits present at specific positions chosen randomly by the program (1st, 3rd, 5th, etc.) and the log file gets updated with each event. In real scenarios the OTP won't be pushed to the user screen (this is just the MVP), instead it'll be sent via SMS or authenticator token app



- **User Interaction:** The function `portal()` is the main application loop for the program's navigation between registration, login and exit options. Asking the user to enter the number corresponding to the option they want.

```
-----
Welcome!
Enter the number of the operation you want:
1- Register a new account
2- Already have an account? Login to your account
3- Exit the program
Register(1) , Login(2) , Exit(3)
```

The function `valid_input()` validates user choices over the allowed options with retry loop for input sanitization

#### - Registration & Login:

- `register()` : Creates new user account, checks username availability, functions `password_validation` and `password_sharding` are called within it, shows options for users to choose their MFA layer, logs registration events in log text file and redirects to the main portal after successful registration.
- `login()` : verifies user credentials, limits login attempts to 3 attempts, implements the chosen MFA method (cognitive OTP or biometric scanner), logs all authentication processes, executes a temporary system lockout after 3 failed attempts.

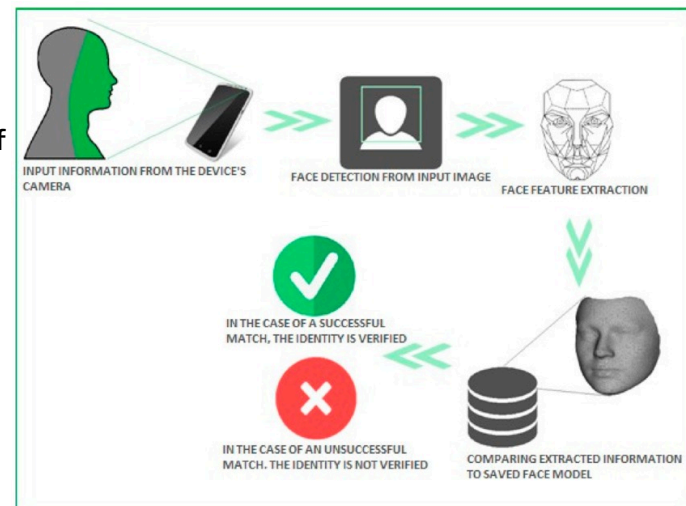
```
# Menu input specificity validation
def remove(text):
    return text.replace(" ", "")
def valid_input(string, valids):
    while True:
        valid_input = remove(input(string).lower())
        if valid_input in valids:
            return valid_input
        else:
            print_pause("Invalid input, please try again")
```

## Biometric Scanner (Face ID) (MFA layer option 2)

Encryption/Database preparation functions: `"load_db()"`, `"saved_db()"` and `"get_cipher()"`

- Creates an encrypted JSON file that stores the face embedding encrypted with a symmetric Fernet key (AES-128-CBC) and HMAC-SHA256 authentication. Decrypt it in the `load_db` function to use the data inside it and then encrypt the dictionary of the database into an encrypted file again.
- The cipher key is in 32-byte safe URL base64 encoded key format. It is stored in an environment variable (`BIOMETRIC_DB_KEY`) and supports both string and bytes key formats.
- Handling errors by showing a runtime error when finding a missing or invalid key

*Note:* The diagram represents an overview visual flowchart of the biometric scanner [for reference credits rights](#)



#### Capturing photo: `"take_photo()"` function

Using OpenCV VideoCapture for webcam access. It resets the camera adjustments by setting the resolution of the captured image to 640x480 pixels (VGA standard) and the camera brightness to 0.5 (50% of maximum)

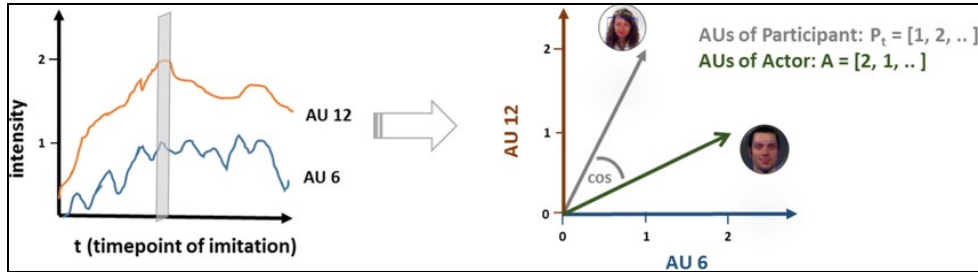
#### Face processing functions: `"detecting_main_face()"` and `"liveness_check()"` functions

- Getting the main face in the photo if two or more faces are detected and to output a single face. Uses the ArcFace model (ArcFace 512D embedding generation) to extract facial features and creates embeddings, returning none if an empty face embedding (no faces detected) by calculating the area of each face using the Detection Backend: SSD (DeepFace implementation) and getting the face with the largest area as the main face (the closest face to the camera)
- Liveness check in the verification function to ensure that the user is a real person not a static photo or video to prevent spoofing. It's done by comparing the facial positions using a threshold of relative movement of >10% relative movement required between captures and computing the movement as a percentage of the face width. The code is resilient to errors by using (try, except exception as e) with printing the error type to show the error reason clearly for troubleshooting & debugging purposes.

### Face registration and verification: "facial\_register()" and "facial\_verification()"

Registers the new user, stores their facial embedding in a 512-dimensional vector which is an ArcFace default and then normalises it in L2 norm. It ensures that these values are saved in uint8 format throughout the process and then uses the cosine similarity (cosine distance) to match the live face with the stored one in the database based on a threshold of 0.45 which we decided based on multiple previous code tests

*Note:* The graphs purpose only to demonstrate cosine distance applications for facial comparison & verification [for reference credits rights](#)



## UI/UX Front-End Design Snapshots (for visualization purposes only)

Registration screen showing the SECURA logo, a title bar, and input fields for Username and Password. Below the fields are buttons for Register and Login.

Login screen showing the SECURA logo, a title bar, and input fields for Username and Password. Below the fields is a Login button and a Back button.

Choose Your MFA screen showing the SECURA logo, a title bar, and a selection screen for authentication methods. The screen displays the text "Please select your preferred authentication method." and two buttons: Cognitive OTP and Biometric Scanner. A Back button is also present.

MFA Menu


MFA Verification screen showing the SECURA logo, a title bar, and a verification screen. The screen displays the text "Enter the 3rd, 4th and 6th digits of the OTP" and a large display showing the OTP "914470". Below the display are buttons for Time remaining: 00:06 and Attempts remaining: 3. At the bottom, there is an input field for OTP Digits and buttons for Verify OTP and Resend OTP.

Cognitive Otp screen

Biometric Scan screen showing the SECURA logo, a title bar, and a biometric scan screen. The screen displays the text "Please tilt your head to the left." and "Follow the instructions below to verify your identity." Below the text are three buttons: Looking for your face..., Tilt your head LEFT, and Tilt your head RIGHT. At the bottom, there is a button for Face captured. Verifying...

Facial Scanner Instructions

# Examples of pop up messages



## Login

Biometric scan canceled. Please try again or use OTP.

Username


rokaia

Password

\*\*\*\*\*

Login

[← Back](#)



## MFA Verification

Access Denied. Expired OTP.

Enter the 3rd, 4th and 6th digits of the OTP

Your OTP:

9 1 4 4 7 0

Time remaining: 00:00


Attempts remaining: 3

OTP Digits

Enter required digits

Verify OTP


Resend OTP





## Biometric Scan


No camera detected. Access Denied.

Follow the instructions below to verify your identity.

 Looking for your face...


 Tilt your head LEFT

 Tilt your head RIGHT

 Face captured. Verifying...

Cancel

[← Back](#)



## Login

Credentials are successfully found! Proceeding to MFA...

Username

rokaia

Password

\*\*\*\*\*

Login

[← Back](#)

# Business Model

## 1- Customer Segments (Primarily Business to Business "B2B")

- Financial institutions: Banks, insurance companies, fintech companies & services
- Healthcare providers: Hospitals and health record platforms
- Government, defense & military entities
- Enterprise companies in various sectors which require IAM
- SaaS companies: B2B or B2C software providers offering secure login experiences

## 2- Value Propositions

- Robust password security and identity verification
- Strong defense against different password attacks, OTP hijacking and biometric spoofing
- Seamless integration of the solution as a backend service aimed at enhancing security connected to the entity's original front-end service without requiring a front-end overhaul thus reducing time-to-value
- Enhanced user-friendly experience
- Security insights achieved by continuous logging of all authentication events
- Regulatory compliance & auditability

## 3- Channels

- **Direct sales** through targeting large entities accounts in finance, healthcare, governance, etc
- **Industry conferences**
- **Partnerships** with cloud service providers, IAM providers and cybersecurity consulting agencies

## 4- Customer Relationships

- Technical support e.g. chatbots for client enterprises as well as developer support teams & comprehensive documentation for developers relations
- Long term partnership making Secura a trusted security layer partner that evolves with the entity's needs
- Hands-on support and consultations during setup so Secura plugs seamlessly into the customer's infrastructure without front-end overhauls
- Continuous proactive security updates & compliance reassurance

## 5- Revenue Streams

- Implementation of the back-end service and onboarding fees
- Subscription-based licensing by offering tiered subscription plans for different customer segments

## 6- Cost Structure

- Cloud & infrastructure costs      - Sales & marketing costs      - Technology and R&D sector costs and investments

## 7- Key Resources

- Human: Cybersecurity specialists, AI developers, Backend developers, customer support teams
- Infrastructural: cloud hosting services and storage, security tools and frameworks

## 8- Key Partners

- Cloud service providers      - Cybersecurity consultancies      - IAM vendors      - Research institutions

## 9- Key Activities

- Developing, updating and maintaining password storage security algorithms and the MFA framework
- Providing support within integrating the back-end system to the organizations' login portals
- Regular pentesting to assess the system's security & auditing security
- Operating and monitoring storage infrastructure

## Social Impact

- Simplifying secure access for non-technical users empowering individuals to protect their digital identities
- Building user trust in digital services across different domains
- Protection of sensitive information and critical sectors: healthcare, finance, defense or even the government online services where Secura can be the backbone of a secure digital citizen ID system all reflecting on national economic stability and security.

## Economic Impact

- Saving costs of data breaches as average cost of a data breach has globally reached \$4.88M in 2024 with healthcare breaches averaging \$10.93M
- Regulatory compliance with frameworks like HIPAA or PCI DSS as non-compliance result in multi-million dollars fines
- Enterprises' brand trust, revenue and reputation protection
- Secura comes in hand with market opportunity as the Identity & Access Management (IAM) market is projected to surpass \$25B by 2030

## Future Work (Aspirations for Secura V2)

- Implementing more features regarding user session management & system's performance speed optimization
- Integrating Key Management Service (KMS) within securing biometric data encryption key storage and managing the fernet key without intercepting user seamless flow asking for the environment key in real deployment
- Robustifying facial ID scanner features; lightness, quality, user distance from the camera, thus increasing adaptability for a more flexible user experience in real deployment
- Developing the liveness check into other approaches such as blink detection
- Logs get identified by user's IP instead of username in upscaling from a MVP (further log masking) in real deployment
- OTP delivery via SMS in real deployment as well as implementing Argon2 in hashing of real production
- Storing the password shards across separate database tables with different access controls for distributed authentication
- Aspirations for integrating AI-based solutions in intelligence & detections for suspicious authentication patterns to learn more from behavioural analysis & detection