A Biometric Recognition-Based Authentication System

Development of a Secure Prototype Focusing on Facial Recognition

Bachelor's thesis in Computer Science and Engineering

Reza Amani
Johanna Avlund
Alexander Gelin
Kevin Han
Farkas Sutthiburt
William Wigemo

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2025

Bachelor's thesis 2025

Reza Amani
Johanna Avlund
Alexander Gelin
Kevin Han
Farkas Sutthiburt
William Wigemo

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2025

A Biometric Recognition-Based Authentication System
Development of a Secure Prototype Focusing on Facial Recognition

Reza Amani
Johanna Avlund
Alexander Gelin
Kevin Han
Farkas Sutthiburt
William Wigemo

Supervisor: Rhouma Rhouma, Department of Computer Science and Engineering
Examiners: Patrik Jansson and Arne Linde, Department of Computer Science and Engineering
Graded by: Magnus Almgren, Department of Computer Science and Engineering

Bachelor's Thesis 2025
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone: +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2025

Development of a Secure Prototype Focusing on Facial Recognition

Reza Amani, Johanna Avlund, Alexander Gelin, Kevin Han, Farkas Sutthiburt, William Wigemo

Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Traditional password-based authentication is increasingly inadequate to protect online services, as it remains vulnerable to brute-force attacks, credential theft, and social engineering tactics. These threats are further exacerbated by common user behaviors, namely password reuse across different platforms and weak password selection. In response, facial recognition has emerged as a promising alternative, offering an intuitive and user-friendly form of authentication. However, existing facial recognition systems present their own challenges, including vulnerability to spoofing, privacy concerns, and inconsistent performance in different environments. This thesis presents the design and implementation of an authentication system, with a primary focus on a facial recognition system developed to address these limitations. A comparative analysis of available models identified YuNet as the most effective model for face detection, and FaceNet for face recognition. The system architecture incorporates a Flask-based backend with a responsive and user-friendly frontend interface, enabling secure user registration and login. To enhance accessibility, multiple alternative login methods have been incorporated, allowing users without camera access to authenticate securely. Additionally, to strengthen data security and privacy, all sensitive user information is securely encrypted before being stored. The system's performance was evaluated using standard classification metrics and carefully optimized to achieve an effective balance between security and usability. The result is an authentication framework that addresses both user convenience, modern security demands, and ethical concerns.

Keywords: Facial recognition, biometric authentication, deep learning, and security.

Sammandrag

Traditionell lösenordsbaserad autentisering är i allt högre grad otillräcklig för att skydda onlinetjänster, då den förblir sårbar för brute-force attacker, identitetsstöld och social manipulation. Dessa hot förvärras ytterligare av vanligt förekommande användarbeteenden, såsom återanvändning av lösenord över olika plattformar och valet av svaga lösenord. Som ett svar på dessa utmaningar har ansiktsigenkänning vuxit fram som ett lovande alternativ och erbjuder en intuitiv och användarvänlig form av autentisering. Befintliga system för ansiktsigenkänning medför dock egna begränsningar, däribland sårbarhet för spoofing, integritetsrelaterade risker samt ojämn prestanda i olika miljöer. Detta arbete presenterar design och implementation av ett autentiseringssystem baserat på ansiktsigenkänning, utvecklat för att hantera dessa begränsningar. En jämförande analys av tillgängliga modeller identifierade YuNet som den mest effektiva modellen för ansiktsdetektering och FaceNet för ansiktsigenkänning. Systemarkitekturen består av ett Flask-baserat backend tillsammans med ett responsivt och användarvänligt frontendgränssnitt, vilket möjliggör säker registrering och inloggning för användare. För att öka tillgänglighet har flera alternativa inloggningsmetoder integrerats, vilket möjliggör säker autentisering även för användare utan tillgång till kamera. Därtill krypteras all känslig användardata på ett säkert sätt innan den lagras, i syfte att stärka dataskydd och integritet. Systemets prestanda har utvärderats med hjälp av etablerade klassificeringsmått och har optimerats noggrant för att uppnå en effektiv balans mellan säkerhet och användbarhet. Resultatet är en autentiseringslösning som tar hänsyn till både användarvänlighet, moderna säkerhetskrav och etiska överväganden.

Nyckelord: Ansiktsigenkänning, biometrisk autentisering, djupinlärning, och säkerhet.

Acknowledgements

We would like to express our profound appreciation to our supervisor, Rhouma Rhouma, whose exceptional mentorship, guidance, and encouragement guided us throughout this project. We are immensely grateful for the countless hours you dedicated to brainstorming ideas with us, sharing your insightful feedback, and encouraging us at every stage. Thank you for not only being an advisor, but also a mentor who made this academic journey exceptionally educational and rewarding.

Reza Amani, Johanna Avlund, Alexander Gelin, Kevin Han,
Farkas Sutthiburt, William Wigemo,
Gothenburg, 2025

Contents

List of Figures

List of Tables

List of Acronyms

2FA Two-Factor Authentication. xiii, 31, 32, 34

AES Advanced Encryption Standard. 31, 45, 49
AUC Area Under The Curve. 14, 18, 19, 37, 44

CNN Convolutional Neural Network. 8, 9, 12, 26, 30
CVSS Common Vulnerability Scoring System. xv, 23, 26, 38–40, 43, 44

DAR Detection-Alignment-Recognition. 7, 49

FN False Negative. 12, 13, 45
FP False Positive. 12, 13, 37, 45
FPR False Positive Rate. xiii, 14, 23, 38

HTML Hypertext Markup Language. 20, 26, 27
HTTP Hypertext Transfer Protocol. 26, 27

LFW Labeled Faces in the Wild. xiii–xv, 17–19, 23, 26, 37, 38, 43, 46, I

MFA Multi-Factor Authentication. 1, 4, 5, 25, 32, 34, 45, 49
MitM Man-in-the-Middle. 15, 40, 43

OTP One-Time Password. 1, 4, 15, 32, 34, 40, 41

ROC Receiver Operating Characteristic. xiii, xiv, 14, 18, 19, 37, 44, I
RSA Rivest–Shamir–Adleman. 6, 32, 40, 43, 49

SSO Single Sign-On. 25, 32–34, 43, 49

TN True Negative. 12, 13
TOTP Time-based One-Time Password. 2, 34, 40, 49
TP True Positive. 12, 13, 37
TPR True Positive Rate. xiii, 14, 23, 29

UI User Interface. 26, 41

1 Introduction

Authentication systems are critical for securing digital services, ensuring that access to sensitive resources is granted exclusively to authorized users. Typically, users must present valid credentials at the beginning of a session to authenticate themselves and gain access. Modern systems implement a variety of authentication methods, all of which rely on the presentation and verification of user credentials.

The most common authentication method is password-based authentication. Although it has long been the standard, it suffers from low security due to poor user practices [59]. For example, weak passwords and reuse across platforms are increasingly vulnerable to phishing and brute-force attacks, which have led to widespread compromise of such systems. In contrast, biometric methods such as fingerprint, iris, and facial recognition remove the need for memorized secrets, allowing users to act as their own credentials [5].

Facial recognition, in particular, offers a contactless and user-friendly approach by leveraging built-in cameras and delivering fast response times, making it well-suited for consumer applications [5]. In practice, technologies such as Face ID and Windows Hello demonstrate the scalability and practicality of facial recognition.

Given its widespread adoption, facial recognition has emerged as an active field of research. Efforts have been made to improve not only security and robustness but also fairness. One example is the Multi-Factor Authentication (MFA) prototype developed by Siddharth and Khankan, which combines facial recognition with FIDO2 (Fast IDentity Online) and One-Time Password (OTP) [52]. While effective against phishing and brute-force attacks, the system lacks spoofing resistance and does not support modular evaluation of different models.

In parallel, other studies [6], [10] have raised concerns about demographic bias in facial recognition systems. This can result in reduced accuracy for underrepresented groups. These findings underscore the ethical responsibility of developers to ensure fairness and transparency in biometric authentication.

1.1 Purpose

In response to these challenges, this project aims to develop a prototype application for user authentication with facial recognition as the primary method. In addition, the system will also support alternative login options to offer greater flexibility and accessibility. The overarching goal of the project is to better understand the design and implementation of authentication systems in real-world scenarios, especially focusing on biometric methods and their associated challenges. Specifically, the project explores best practices for secure implementation, evaluates the performance and trade-offs of facial recognition models, and addresses societal and ethical considerations such as bias and privacy.

1.2 Scope

This project does not include the development or training of new facial recognition algorithms. Instead, it focuses on building a custom authentication module that integrates and evaluates existing facial recognition libraries. Developing a new algorithm would require extensive computational resources, large labeled datasets, and fine-tuning, which are beyond the scope of this project. Furthermore, the evaluation of the facial recognition models is limited by the size and diversity of the datasets available for testing, which may affect the generalizability of the results to wider populations.

Additionally, mobile platform integration and support for third-party Time-based One-Time Password (TOTP) apps (e.g., Microsoft Authenticator) are not included. These features require significant additional development time and testing to ensure reliable, cross-platform performance and secure synchronization. Given the project's time constraints and focus on core facial recognition authentication, these functionalities were deferred for future work.

1.3 Thesis Outline

The remainder of this thesis is organized as follows:

- **Chapter 2:** Overview of relevant authentication methods and related work.
- **Chapter 3:** Technical background on facial recognition.
- **Chapter 4:** Methodology, including model evaluation, tool selection, and prototype development.
- **Chapter 5:** Technical implementation.
- **Chapter 6:** Evaluation results.
- **Chapter 7:** Discussion of findings, limitations, and implications.
- **Chapter 8:** Conclusion.

Additional materials such as supplementary data and extended results are included in the appendices.

2
Authentication Methods

Authentication is the process of verifying a user's identity to prevent unauthorized access and to ensure system privacy and integrity [5]. Robust authentication is crucial for maintaining user security and trust. This chapter outlines various types of authentication methods, compares them across key factors, and explains the rationale for selecting facial recognition for this project.

2.1 Types of Authentication Methods

Authentication methods can be grouped into three distinct categories: knowledge-based, possession-based, and inherence-based [5]. Although the common goal is to verify a user by their provided credentials, the main differentiator is how the credential is verified and tied to the user.

2.1.1 Knowledge-Based Authentication

In many applications, one of the most widely used methods to confirm a user's identity is knowledge-based authentication [5]. This approach relies on the information that an individual retains, such as a password, a PIN, or responses to security questions. Although convenient and sometimes easy to recall, these memory-based credentials often fall short in terms of security. Such methods are often vulnerable to unauthorized access, guessing, theft, and user forgetfulness [5].

The most common form of knowledge-based authentication is password authentication [59]. With this method, users provide a password to verify their identity. However, these systems are often vulnerable to phishing attacks and other security risks, especially when users choose weak passwords or reuse them across different platforms. In addition, password-based systems are vulnerable to keylogging malware, which records keystrokes to steal passwords [44].

2.1.2 Possession-Based Authentication

Possession-based authentication verifies the identity of a user through a physical device or object they own [5]. This method has become widely adopted due to the portability of devices such as smartphones, USB keys, security tokens, and smart cards. It strengthens security by requiring users to present a tangible item in their possession, ensuring that even if one authentication factor is compromised, unauthorized access remains unlikely without the associated physical device [5].

A common implementation of possession-based authentication is the OTP, which generates short-lived numeric codes based on a shared secret and the current time [38]. Typically delivered via a smartphone app, these codes remain valid for only a short duration (e.g., 30 seconds), making them resistant to replay attacks. Among delivery methods, app-based OTPs are considered more secure than SMS or email due to lower interception risks [33].

2.1.3 Inherence-Based Authentication

Inherence-based authentication verifies identity based on physical or behavioral characteristics unique to an individual, such as fingerprints, iris patterns, or facial features. This category of authentication is commonly referred to as biometric authentication, since it relies on biological traits [5]. These traits are closely tied to the user and are difficult to replicate or share, making them valuable for secure identity verification. As a result, it eliminates the need for users to remember credentials or carry security tokens. This enhances the convenience for the user and eliminates the risk of device loss [5]. For instance, Face ID on Apple devices and Windows Hello are examples of modern systems that rely on facial recognition, simplifying the authentication process without compromising security.

While biometric authentication provides strong security and convenience, it also presents certain limitations. Biometric data is generally difficult to steal or guess, but systems remain vulnerable to spoofing attacks using deepfakes or high-resolution photos which can bypass facial recognition systems [5]. Furthermore, once compromised, biometric data cannot be reset or changed in the same way as passwords, making such breaches especially damaging. Besides security issues, there are also practical challenges. High-quality sensors such as fingerprint or iris scanners can be expensive, and their performance may vary in different conditions, including low lighting or dirty sensors [5], [43].

2.2 Multi-Factor Authentication

MFA is an additional security measure that enhances protection by requiring two or more independent authentication factors before granting access [35]. These factors are drawn from the three categories outlined in Section 2.1: something the user knows (e.g., a password or PIN), something the user has (e.g., a smartphone or OTP), and something the user is (e.g., a biometric trait).

By combining factors from different categories, MFA significantly reduces the risk of unauthorized access. Even if one factor is compromised, an attacker would still need to overcome the others to gain entry [39]. This layered approach provides strong protection and is widely used in real-world applications. For example, Google accounts can be configured to require users to log in with a password and then confirm their identity using a prompt on a trusted device, an authentication app, or biometric verification.

2.3 Comparison of Authentication Methods

Different authentication methods can be compared based on key factors such as security, usability, cost, and scalability. To give a clearer understanding of their strengths and weaknesses, Table 2.1 presents a comparison between knowledge-based, possession-based, and inherence-based approaches.

This comparison draws on findings from multiple studies [4], [26], [2], [31], with the table itself created to synthesize and present the information in a clear and accessible format.

Table 2.1: Comparison of Authentication Methods
(Note: ↑ = high is desirable, ↓ = low is desirable)

Criterion	**Knowledge-Based**	**Possession-Based**	**Inherence-Based**
Security (↑)	Low: Susceptible to phishing and weak passwords [4]	Medium: Depends on device integrity [26]	High: Difficult to forge biometric traits [2]
Usability (↑)	Medium: Requires memory, can cause password fatigue [4]	High: Easy to use, but devices can be lost or forgotten [31]	High: Always present, but performance may degrade in specific conditions (e.g., cold) [13]
Cost (↓)	Low: Minimal infrastructure required [4]	Medium: Additional hardware needed [26]	High: Requires biometric sensors [2]
Scalability (↑)	High: Easy to implement across many users [4]	Medium: Dependent on device distribution [26]	Low: Scalability hindered by hardware needs and user variability [2]

2.4 Related Work

Authentication methods, especially biometrics, have attracted significant research interest due to the potential for improved security and usability. For example, in the study by Alrawili et al., they emphasize the need for robust authentication systems that address usability, spoofing resistance, and sensitivity to biometric traits [5].

Siddharth and Khankan proposed a MFA system using facial recognition, OTP, and FIDO2 [52]. Their approach demonstrated improved security against phishing and brute-force attacks by leveraging multiple authentication factors. However, the system lacked modularity and robust spoofing defenses, highlighting ongoing challenges in creating flexible and secure biometric systems.

Another study evaluated facial recognition pipelines using the DeepFace framework, highlighting the importance of interoperability between detection and recognition components [49]. Their findings support a modular architecture that allows experimentation with alternative models to optimize performance and maintainability.

Security is a central concern in biometric systems, particularly against spoofing and data theft. Akhtar et al. provide a foundational review of biometric liveness detection, identifying major challenges in detecting presentation attacks such as photos and deepfakes [3]. Padhiar complements this with a comparison of symmetric and asymmetric encryption techniques, highlighting the strengths of public-key cryptography such as Rivest–Shamir–Adleman (RSA) [40]. Together, these works highlight the importance of the integration of anti-spoofing and cryptographic safeguards in secure authentication pipelines.

In addition, studies have brought attention to demographic bias in facial recognition technologies, revealing that many systems exhibit reduced accuracy for underrepresented groups [6], [10]. These findings indicate the need for fairness, transparency, and ethical consideration in the development of biometric authentication systems.

Taken together, these contributions provide a well-rounded foundation for advancing facial recognition-based authentication. These insights guide the project's focus on modularity and security in the design of authentication systems, while also informing the discussion of ethical considerations and the measures taken to address them.

2.5 Motivation for Facial Recognition

The reason behind exploring facial recognition in this project is its balance of security, usability, and accessibility [5], [43], [28], making it a promising method for modern authentication systems. Unlike fingerprint recognition, facial recognition is contactless, making it less intrusive and more hygienic, particularly for consumer devices [5]. Moreover, the widespread availability of built-in cameras eliminates the need for specialized biometric hardware, simplifying deployment.

As a passwordless solution, facial recognition overcomes several weaknesses of traditional methods. It is more difficult to replicate and directly ties authentication to the individual, providing enhanced security and improved user experience [27]. Despite these advantages, facial recognition systems face challenges such as spoofing attacks using photos or videos. However, these risks can be mitigated through robust anti-spoofing mechanisms [5].

Overall, when comparing authentication categories, inherence-based methods such as facial recognition stand out for their high security and usability, as summarized in Table 2.1. These factors position facial recognition as a strong candidate for secure and scalable identity verification in modern applications.

3 Facial Recognition

The main mechanism of facial recognition involves the analysis and comparison of an individual's facial features with stored biometric data [53], [61]. This chapter describes the typical pipeline for facial recognition tasks and different methods for facial feature extraction from facial images. Evaluation metrics are subsequently introduced to assess the performance of such pipelines. Lastly, societal and ethical concerns regarding facial recognition are introduced.

3.1 Pipeline

In modern facial recognition systems, the task is often divided into four key stages: detection, alignment, representation, and verification [49], [50], [64], as illustrated in Figure 3.1. The reason for these four distinct stages is that real-world facial recognition applications need to detect, align, and recognize faces from varying scenes in videos and images [24]. As a consequence, facial recognition is commonly described as part of a Detection-Alignment-Recognition (DAR) pipeline. All four of these stages are independent of each other [64] and implement various algorithms to properly receive images from, or prepare images for, the next stage of the DAR pipeline [24].

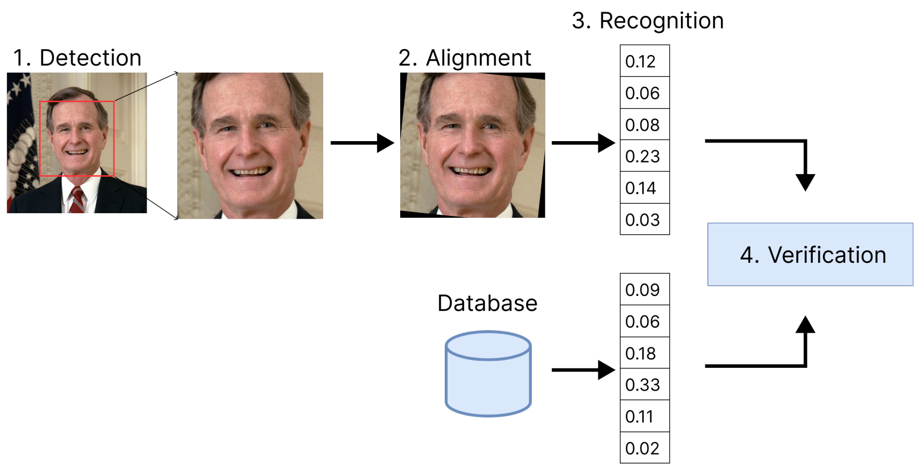


Figure 3.1: The facial recognition pipeline. The original image of George H. W. Bush is in the public domain [56] and has been modified.

3.1.1 Detection

The first step of the facial recognition pipeline is facial detection [50], [64]. In general, face detection is a subtask of object detection, where the goal is to locate and classify instances of faces within a given image. This is commonly achieved by generating bounding boxes around facial regions and assigning confidence scores indicating the likelihood of a face being present in each box.

Since captured images often include unnecessary information, the facial region must be isolated before further processing [51]. This localization makes the system more robust to external factors such as background clutter, occlusion, and variations in face position or size. In addition to identifying the bounding box of the facial region [51], the detection stage also plays a role in locating facial landmarks, such as eyes [49], which are critical for downstream tasks such as alignment and normalization.

One of the earliest and most influential approaches to real-time face detection is the Haar cascade classifier proposed by Viola and Jones [57]. It uses simple rectangular features that represent contrast patterns commonly found in facial regions, such as around the eyes and cheeks. These features are evaluated in a cascading structure that quickly filters out non-face regions, allowing for fast and efficient detection.

Haar cascade classifiers gained popularity due to their speed and are still available in widely used libraries such as OpenCV. However, they have significant limitations in handling variations in pose, lighting, and facial expressions [49]. To address these challenges, modern face detection models have shifted toward deep learning–based approaches, particularly Convolutional Neural Networks (CNNs) [54].

3.1.2 Alignment

After facial detection, an alignment step is performed [49]. Face alignment refers to the process of adjusting the orientation of a detected face, positioning the key facial landmarks in a standardized way. This step ensures that faces are presented in a consistent format, which improves the accuracy and reliability of subsequent recognition processes. The alignment step significantly enhances the performance of a facial recognition pipeline [49].

A technique for aligning a face involves calculating the angle between the eyes. By using the position of the eyes, the angle in between can be computed using Equation 3.1. The eye positions are provided from the detection stage.

$$\alpha = \arctan(\frac{a}{b}) \tag{3.1}$$

In the equation above, a and b are the vertical and horizontal distances between the eyes, respectively, and α is the angle of facial rotation. α is used to rotate the image so that the eyes align horizontally. Figure 3.2 visualizes the result of the alignment process.

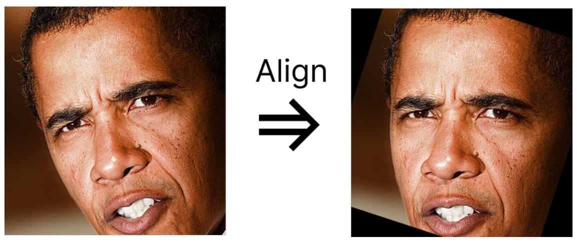


Figure 3.2: Facial alignment output given an input image. Original image of Barack Obama [32] has been modified, CC BY SA.

3.1.3 Representation

The result of the detection and alignment steps of the pipeline serves as inputs for the representation step [49]. This step is important because it extracts facial features from the image and converts them into vector embeddings, as illustrated in Figure 3.3.

A face embedding is a compressed numerical vector that represents a person's facial features for recognition tasks [23]. It is generated by processing a high-dimensional image to extract key characteristics such as texture, shape, and distances between facial landmarks while discarding irrelevant details. In modern systems, CNNs are commonly used to extract these embeddings.

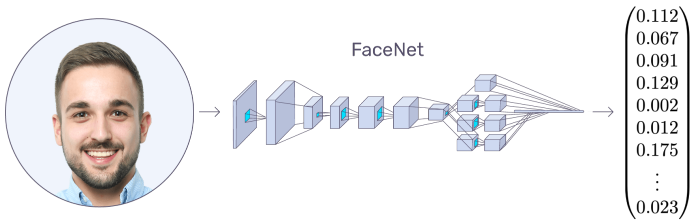


Figure 3.3: Facial recognition pipeline using FaceNet and MTCNN. [14]. Rendered with permission.

3.1.4 Verification

The verification step of the facial recognition pipeline aims to determine whether two face embeddings represent the same identity by comparing their vector representations [49]. This is done by calculating the distance or similarity score between the two embeddings and then applying a threshold. For example, if the score falls within a certain range, the two faces are considered a match. Equation 3.2 describes the classification in mathematical terms. *distance* is a function returning the distance between two face embeddings ($\vec{f_1}$ and $\vec{f}2$ in this case). θ is the distance threshold.

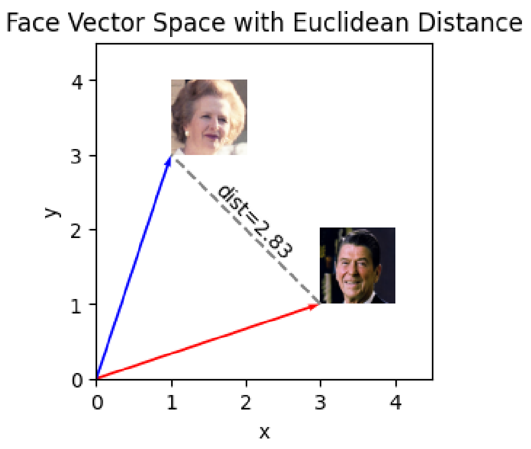
$$match = \begin{cases} 1 & \text{if } distance(\vec{f_1}, \vec{f_2}) \leq \theta \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

Several distance metrics [49] are commonly used to measure the similarity between face embeddings:

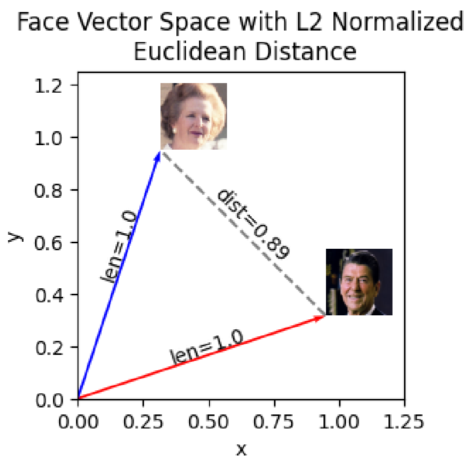
Euclidean Distance This metric is calculated as the straight-line distance between two points in the embedding space. It is intuitive and works well when the embeddings are distributed in a uniform space. The formula for Euclidean distance is presented in Equation 3.3.

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{3.3}$$

To illustrate Euclidean distance, Figure 3.4a shows two face embeddings in a 2-dimensional space. Although simplified, the same concept applies to higher dimensions: the closer the embeddings, the more similar the faces.



(a) Euclidean



(b) L2-normalized Euclidean

Figure 3.4: Example of Euclidean and L2-normalized Euclidean distance. The original images of Margaret Thatcher [55] and Ronald Reagan [15] is in the public domain and has been modified.

L2-normalized Euclidean Distance Building on the standard Euclidean distance metric, this approach normalizes the embeddings to unit length before computing the distance. This reduces sensitivity to scale differences and helps with stability when embeddings are produced by neural networks. The formula is presented in Equation 3.4 and 3.5. A geometric visualization of L2-normalized Euclidean distance is illustrated in Figure 3.4b. In this example, the two face embeddings are normalized to unit length, and the dashed line represents the Euclidean distance.

$$\hat{x} = \frac{x}{\|x\|_2}, \quad \hat{y} = \frac{y}{\|y\|_2} \tag{3.4}$$

$$d(\hat{x}, \hat{y}) = \sqrt{\sum_{i=1}^{n} (\hat{x}_i - \hat{y}_i)^2} \tag{3.5}$$

Cosine Similarity A measure of similarity that compares the direction of the vectors, rather than comparing absolute position. This is more meaningful in high-dimensional feature spaces like face embeddings [12]. Cosine similarity is computed according to Equation 3.6 and a visual example is presented in Figure 3.5. A score close to 1 indicates a strong similarity.

$$\cos(\theta) = \frac{x \cdot y}{\|x\|_2 \cdot \|y\|_2} \tag{3.6}$$

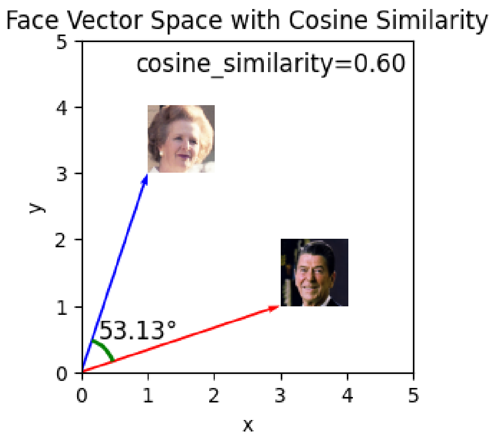


Figure 3.5: Visual example of cosine similarity. The original images of Margaret Thatcher [55] and Ronald Reagan [15] is in the public domain and has been modified.

In practice, cosine similarity and L2-normalized Euclidean distance are commonly used metrics for comparing face embeddings due to their robustness to scale variations [47], [60]. Choosing the right metric often depends on the architecture of the model and how the embeddings are distributed in the vector space. For example, some systems perform better with cosine similarity, especially when the model is trained to produce directionally consistent embeddings.

3.2 Convolutional Neural Networks

CNNs are deep learning-based feature extraction techniques that have become the most widely used approach in modern facial recognition systems [54]. They automatically learn patterns in images through multiple layers of convolution, pooling, and activation functions [34]. In facial recognition, the layers first extract basic facial features, such as edges and textures. As the network progresses through deeper layers, it identifies more complex, identity-specific patterns. When trained on large and diverse datasets, the network learns to represent face images as high-dimensional vectors that capture the unique characteristics of a person's face [53].

Prominent CNN-based models include DeepFace and FaceNet, each offering architectural strategies and training techniques [34]. These vector embeddings can be compared using the previously mentioned distance metrics to verify or identify faces [49]. Moreover, many of these models have demonstrated performance levels that surpass human accuracy in facial recognition tasks [49], [54].

3.3 Evaluation Metrics

Evaluating the performance of a facial recognition system involves assessing its classification accuracy and identifying both correct classifications and errors. This is typically achieved using a set of standardized evaluation metrics, many of which are derived from the confusion matrix.

3.3.1 Confusion Matrix

The confusion matrix is a fundamental tool used to evaluate the performance of a facial recognition system by comparing predicted outcomes to actual ground truth labels. In the case of binary classification, it categorizes predictions into four outcomes: True Positive (TP), False Positive (FP), False Negative (FN), and True Negative (TN). TP represents correctly identified faces, FP indicates faces falsely matched to an incorrect identity, FN refers to rejections of two facial images belonging to the same individual, and TN represents correctly rejected non-matching faces.

To help interpret these terms, consider the following example: a facial recognition system is tested on 10 different faces, some known to the system, others unknown. This small-scale scenario helps show how correct and incorrect predictions translate into values in the confusion matrix:

- 6 faces are correctly identified (TP = 6)
- 1 face is falsely accepted (FP = 1)
- 2 known faces are not recognized (FN = 2)
- 1 unknown face is correctly rejected (TN = 1)

This example then yields the confusion matrix presented in Table 3.1. The values from the confusion matrix serve as the basis for calculating key evaluation metrics such as accuracy, precision, recall, and F1-score. These metrics and formulas are applied in this project to evaluate and compare the performance of different facial recognition models.

Table 3.1: Confusion matrix example

	Predicted Positive	**Predicted Negative**
Actual Positive	6 (TP)	2 (FN)
Actual Negative	1 (FP)	1 (TN)

3.3.2 Performance Metrics

From the confusion matrix, several key performance metrics such as accuracy, precision, recall, and F1-score can be derived. Each metric captures a different aspect of the system's behavior, offering insights into its ability to correctly identify faces, avoid misclassifications, and maintain a balanced overall performance. The outcome of the following metrics will always range between 0 and 1, where values closer to 0 indicate worse performance, and values closer to 1 indicate better performance.

Accuracy Measures the proportion of correct predictions (TP + TN) out of all predictions [58]. Equation 3.7 presents the formula with example values, illustrating that the model correctly predicted 70% of all test cases.

$$ACC = \frac{TP + TN}{TP + TN + FN + FP} = \frac{6 + 1}{6 + 1 + 1 + 2} = \frac{7}{10} = 0.70 \tag{3.7}$$

Precision The number of true positive (TP) predictions divided by the total number of predicted positives [58]. Equation 3.8 shows the formula with example values. The result of the formula indicates that when the model predicted a positive match, it was correct about 86% of the time. A higher precision score reflects fewer false positives (FPs), which is critical in authentication, as a false positive would allow unauthorized access to the system.

$$PREC = \frac{TP}{TP + FP} = \frac{6}{6 + 1} = \frac{6}{7} \approx 0.86 \tag{3.8}$$

Recall The metric is defined as the proportion of actual positives that were correctly identified. It shows how well the model detects positive cases [58]. Equation 3.9 is presented below with example values given, and the result indicates that the model was able to correctly identify 75% of the actual positive faces.

$$REC = \frac{TP}{TP + FN} = \frac{6}{6 + 2} = \frac{6}{8} = 0.75 \tag{3.9}$$

F1-score The harmonic mean of precision and recall, offering a balance between the two when they are both important [58]. The formula for calculating the F1-score is seen in equation 3.10 with example values given. The result shows that the model achieves a good balance between precision and recall, with an overall F1 performance of 80%.

$$F1 = \frac{2 \cdot PREC \cdot REC}{PREC + REC} = \frac{2 \cdot 0.86 \cdot 0.75}{0.86 + 0.75} \approx 0.80 \tag{3.10}$$

Receiver Operating Characteristic (ROC) curve A graph that visualizes the trade-off between True Positive Rate (TPR) and False Positive Rate (FPR) [58]. Each point on the graph represents the TPR and FPR for a certain threshold in the verification step. A good classifier has a ROC curve that approaches the top-left corner, since the FPR can be minimized while the TPR is kept high. To illustrate how to analyze ROC curves, Figure 3.6 displays multiple curves. The dashed line represents a classifier that assigns class labels randomly, while the three other curves move progressively closer to the point (0, 1), indicating improving classifier performance.

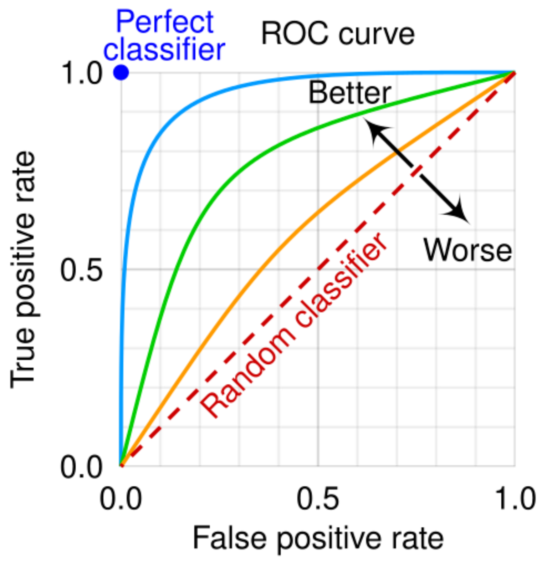


Figure 3.6: Example of a ROC curve with FPR and TPR. [11], CC BY SA.

Area Under The Curve (AUC) The area under the ROC curve [58]. An AUC closer to 1 indicates a better model. In Figure 3.6, the random chance classifier has an AUC of 0.5, while the left-most blue curve has an AUC close to 1, indicating significantly better performance.

3.4 Security Threats

Facial recognition technology is vulnerable to common cyberattacks targeting biometric traits. One such threat is the replay attack, where an adversary captures a legitimate user's facial image or video during a previous interaction or from external sources such as hidden recordings or social media. The attacker then reuses this captured data to deceive the facial recognition system into granting access [5], [46].

Another frequently observed threat is the spoofing attack, in which an attacker uses a printed photograph, a high-resolution screen that displays the user's face, or a deepfake video to deceive the facial recognition system [3]. These attacks exploit a system's inability to distinguish between a live user and a static or synthetic one.

More advanced threats include model inversion attacks, where attackers with access to stored facial embeddings in a database may attempt to reconstruct approximate facial images of users [22]. This is possible by leveraging the mathematical characteristics of the embeddings. Even with data storage safeguards, such as hashing and encryption, advanced attackers may still try to extrapolate sensitive biometric information.

In the event of a database breach, the consequences can be severe. Facial recognition systems that store biometric data risk having attackers gain access to sensitive information such as face embeddings and user email addresses. Even if raw images are not stored, leaking said embeddings and user information poses privacy risks.

Lastly, a Man-in-the-Middle (MitM) attack involves an attacker intercepting the communication between two parties, potentially capturing facial images, OTPs, or authentication responses. If encryption protocols are not enforced properly, an attacker could intercept, modify, or reuse transmitted data to gain unauthorized access [29].

3.5 Societal and Ethical Considerations

The increasing adoption of facial recognition technology introduces complex challenges in balancing security benefits with individual rights and freedom [30]. While the technology enables fast and contactless authentication, its deployment has raised wider societal and ethical questions. Understanding these concerns is crucial for contextualizing the role of biometric technology in modern applications.

3.5.1 Privacy and Data Sensitivity

Biometric data is both sensitive and immutable. Unlike passwords, it cannot be changed if compromised. Such data must therefore be handled with care. Once collected and stored, it is difficult to detect whether it has been accessed, copied, or used for other purposes without proper authorization or consent, especially if the system lacks robust auditing and monitoring mechanisms. This raises the risk of misuse and use beyond its intended purpose [30]. These concerns have prompted ongoing debates about the ethical boundaries of biometric data use.

3.5.2 Algorithmic Bias

A well-documented concern with facial recognition systems is the presence of algorithmic bias. Studies have shown that facial recognition systems often perform less accurately on individuals with darker skin tones and on women, leading to higher false rejection and false acceptance rates for these groups [6], [10]. For example, black individuals are disproportionately represented in law enforcement databases, increasing the risk of being incorrectly identified or unfairly targeted by automated identification technologies [7]. These disparities are generally attributed to imbalanced training data or unrepresentative model development, which can lead to inequitable outcomes in real-world deployments.

3.5.3 Surveillance and Social Impact

The use of facial recognition in surveillance, particularly in public or semi-public spaces, has prompted concerns about its potential impact on society [21]. Large-scale tracking of individuals in these settings can pose risks to democratic freedoms by undermining personal privacy and autonomy. Reports have documented instances where law enforcement requested footage from neighborhood surveillance systems, raising questions about potential misuse and informed consent in the use of such technologies [21], [9]. These examples illustrate how facial recognition can affect social norms and expectations of privacy when deployed at scale.

4 Methodology

A clear methodology was established to ensure the system met the project's objectives of high accuracy, real-time performance, and practical usability. The primary goal of the project is to explore the implementation of facial recognition as a secure and accessible authentication method in a real-world application. This includes evaluating how well different models perform in terms of speed, accuracy, and reliability, as well as identifying suitable tools for system development. Additionally, the project aims to balance technical effectiveness with user experience and ethical considerations such as bias and privacy.

This chapter outlines the structured approach used to design, implement, and evaluate the system. It covers model selection and benchmarking, system architecture, prototype development, and testing strategies.

4.1 Model Evaluation and Selection

Selecting the most suitable models was a necessary part of developing a reliable and efficient face recognition system. To ensure objective comparisons and reproducible results, the benchmark dataset Labeled Faces in the Wild (LFW) was used, offering standardized and widely recognized test cases. In addition, performance evaluation focused on speed and recognition reliability to balance efficiency with authentication accuracy.

Testing was performed by running models on fixed image sets drawn from the LFW dataset. These measurements formed the basis for decisions regarding the choice of models and tools, guiding the implementation toward an optimal balance between security, usability, and scalability. Specifically, models needed to deliver accurate and consistent recognition to support security, perform with minimal latency to ensure a smooth user experience, and remain lightweight and adaptable to enable deployment across various environments and support scalability to multiple users.

Candidate detection models were evaluated based on speed, accuracy, and their ability to localize facial landmarks [49]. YuNet was selected for its efficiency and suitability for real-time applications [62]. RetinaFace was included for its high accuracy in complex conditions, while MTCNN was chosen for its robust multi-task cascaded design. OpenCV and SSD were also tested, offering lightweight alternatives despite lower detection performance.

For facial recognition, FaceNet was selected as the primary model due to its high performance in generating discriminative embeddings, as discussed in Chapter 3. Other models such as ArcFace, VGG-Face, OpenFace, DeepID, and SFace were also considered for comparative evaluation.

4.1.1 ROC Curve Comparisons

To assess recognition performance, ROC curves were generated using 1,000 face pairs from the LFW dataset. Among the tested configurations, the combination of FaceNet for recognition and YuNet for detection yielded the highest AUC, indicating superior classification capability. As illustrated in Figure 4.1, this pairing demonstrated consistent accuracy across varying conditions. These results directly informed the model selection, with FaceNet and YuNet chosen as the system's primary recognition and detection components, respectively, due to their combined accuracy and speed. Additional ROC comparisons for other model combinations are provided in Appendix A.

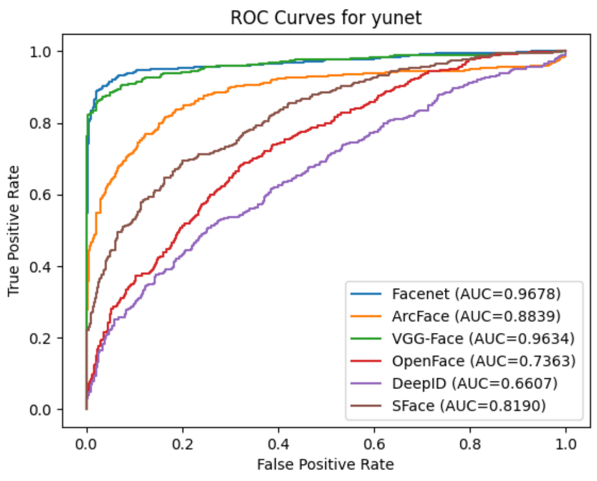


Figure 4.1: YuNet for detection and recognition evaluated on LFW.

4.1.2 Speed Comparisons

Since real-time performance is essential for usability, the end-to-end processing time, including detection, alignment, and recognition, was evaluated for various model combinations on fixed images. YuNet proved to be the fastest detection model, followed by RetinaFace, with OpenCV and MTCNN being less suitable for real-time systems. The choice of recognition model had little impact on timing, which informed the decision to prioritize YuNet for detection to optimize system responsiveness. The comparison is presented in Figure 4.2.

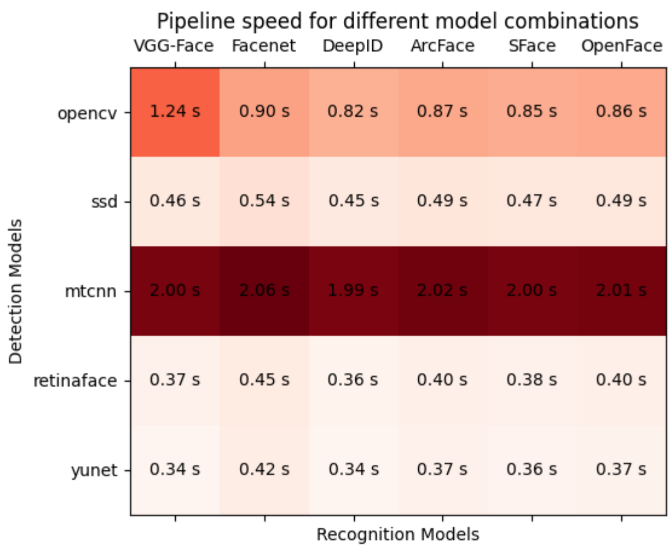


Figure 4.2: End-to-end pipeline times for different model combinations.

4.1.3 Distance Metric Comparisons

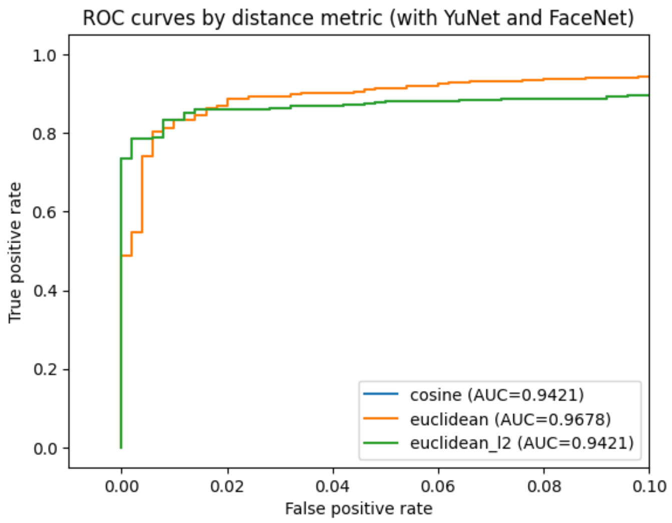


Figure 4.3: ROC curves for YuNet and FaceNet evaluated on LFW by distance metric.

Recognition accuracy also depends on the distance metric used to compare facial embeddings. Three metrics were compared: Euclidean distance, L2-normalized Euclidean distance, and cosine similarity. Evaluating on the LFW dataset revealed that the standard Euclidean distance yielded the highest AUC, leading to its adoption as the similarity measure in the system. The comparison is presented in Figure 4.3.

4.2 Prototype

The prototype consisted of two key components: the client application and the web server. Together, these components enabled a seamless and secure facial authentication process.

The client application handled user interaction, captured facial data during registration and login, and communicated with the server. User inputs and facial images were securely transmitted to the server for further processing.

The web server was built using Flask, and the backend was implemented following the Model-View-Controller (MVC) architecture. It supports both Hypertext Markup Language (HTML) and RESTful JSON responses, handles requests, processes facial recognition tasks, and manages database operations.

The client-server communication was responsible for handling the exchange of data between the client application and the server during both the login and registration processes. To illustrate how the system would operate in practice, the registration and login flows were designed to follow a structured sequence of steps, enabling clear and secure interaction between the client and server.

The registration flow is shown in Figure 4.4 and included the following steps:

1. The user was prompted to enter personal information

2. The application activated the webcam to capture the user's facial data

3. The captured image, along with the personal details, was sent to the web server

4. The server processed the authentication request by passing the image to the face recognition pipeline

5. The facial image was converted into a face embedding

6. The face embedding and user details were stored in the database

7. The server confirmed successful account registration

8. The application displayed a successful account registration

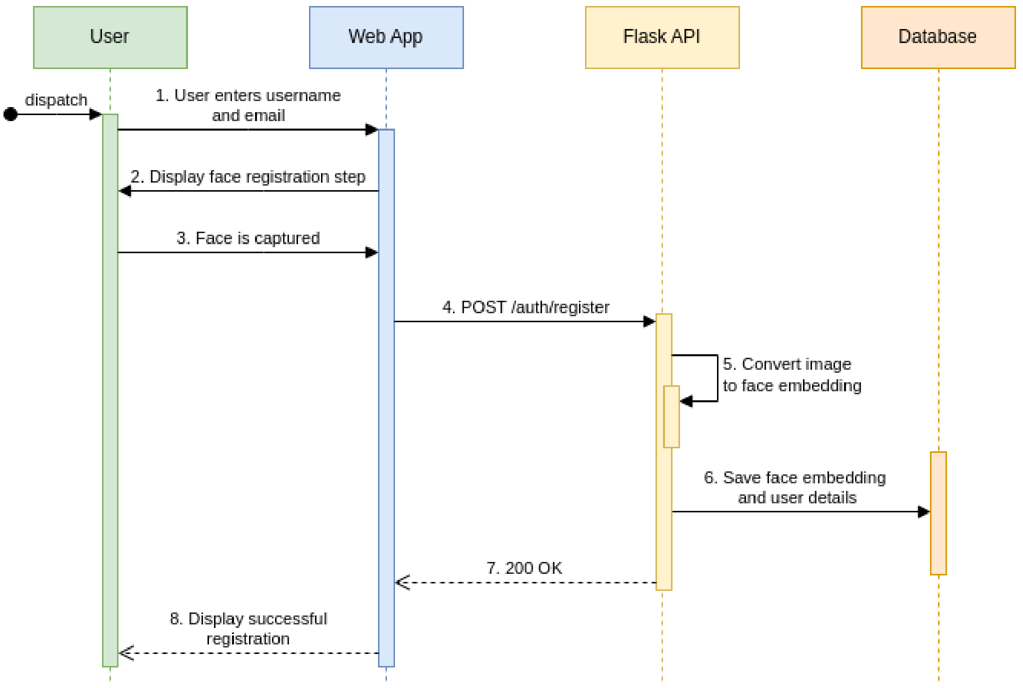

User
Web App
Flask API
Database
dispatch
1. User enters username and email
2. Display face registration step
3. Face is captured
4. POST /auth/register
5. Convert image to face embedding
6. Save face embedding and user details
7. 200 OK
8. Display successful registration

Figure 4.4: Overview of registration flow.

The login flow is visualized in Figure 4.5 and included the following steps:

1. The user was prompted to enter their username
2. The application activated the webcam to capture the user's facial data
3. The captured image and username were sent to the web server
4. The server processed the authentication request by passing the image to the face recognition pipeline
5. The captured image was converted to a face embedding
6. The database got a request to fetch the current user using username
7. The current user was found and sent to FlaskAPI
8. The converted face embedding vector was compared with the stored embedding vector
9. The server confirmed successful account login based on the similarity comparison between the embedding vectors
10. The user is redirected to the dashboard

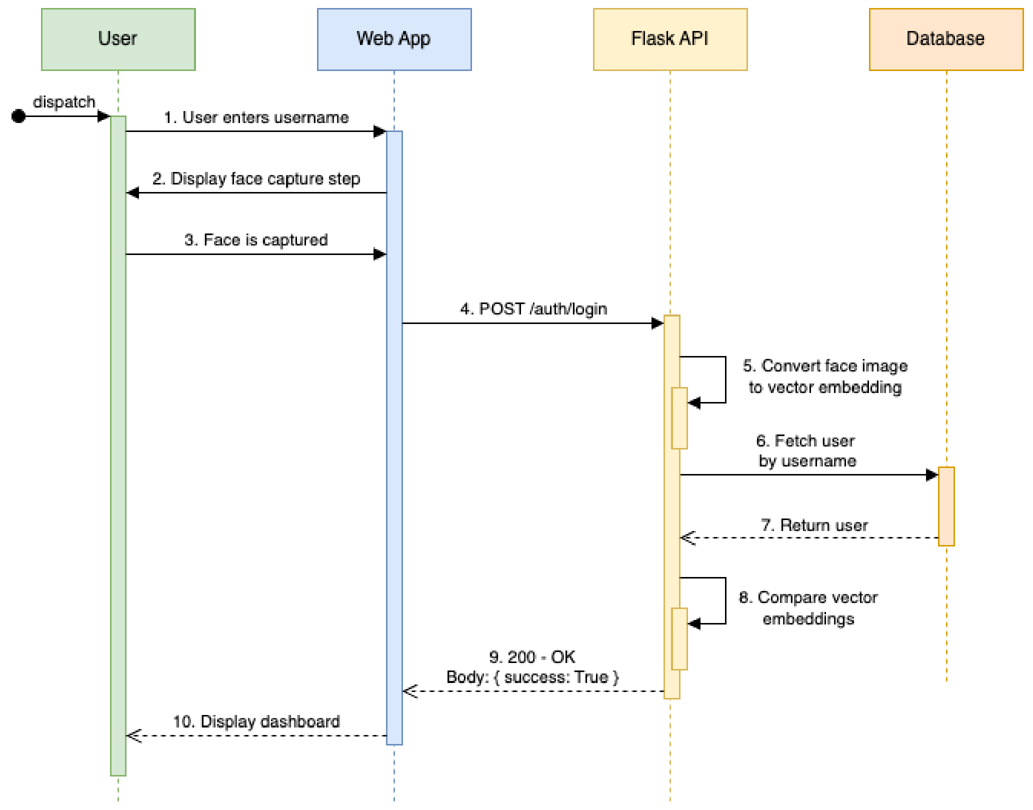

User
Web App
Flask API
Database
dispatch
1. User enters username
2. Display face capture step
3. Face is captured
4. POST /auth/login
5. Convert face image to vector embedding
6. Fetch user by username
7. Return user
8. Compare vector embeddings
9. 200 - OK Body: { success: True }
10. Display dashboard

Figure 4.5: Overview of a successful login flow.

4.3 Application Testing

Testing of the application was structured around four key areas: functional usability, system performance, security assessment, and software reliability. Each area targeted specific aspects of the system to ensure comprehensive validation of both functionality and real-world performance.

The practical testing of the facial recognition system was evaluated under various conditions, including lighting, facial expressions, and backgrounds. A small group of external users also provided feedback on the interface and usability, helping to identify and resolve design or functionality issues early in development.

To ensure a responsive login experience, performance testing was conducted by benchmarking the facial recognition pipeline under realistic usage scenarios. This involved timing 20 consecutive login attempts using two implementations: one based on the DeepFace framework and another custom-built for modularity and efficiency. Metrics such as average execution time and standard deviation were recorded to evaluate speed and consistency. All measurements were performed on a controlled test machine, with automation tools used to ensure repeatable conditions across test runs.

The system's robustness to potential threats was evaluated using threat modeling using the CVSS v4.0 calculator. This tool allowed for the evaluation of vulnerability severity within the application's scope, supporting the development of prioritized mitigation strategies based on the results.

Software reliability was supported through automated testing practices. Unit tests were implemented to verify the correctness and behavior of individual components. Mocking techniques, using libraries such as Mockito, enabled the simulation of dependencies and external services, allowing for isolated and reliable test cases.

To further optimize system performance and security, threshold tuning in the verification stage was also carried out. An optimal threshold should minimize the FPR to enhance security, while maintaining a high TPR for usability. To identify such a threshold, three different selection methods were employed. The pipeline was then evaluated using the LFW dataset, and classification metrics were computed for each approach. The threshold selection methods included:

- Youden's J Statistic – maximizes sensitivity and specificity [36]
- Minimal Distance to (0, 1) in ROC space – identifies the point closest to perfect classification [42]
- Constraint-based – selects the highest threshold that satisfies a predefined maximum FPR.

5

Technical Implementation

This chapter outlines the key components of the authentication system and the rationale behind the selected tools and technologies. It covers the client-server interaction, backend architecture, and the design of the facial recognition pipeline, including image validation and spoofing detection. The chapter also describes the database schema, encryption mechanisms, and the implementation of alternative login methods such as passwords, RSA key signing, Google SSO, and MFA.

5.1 Tools and Implementation Frameworks

The following tools and frameworks are used to build and support the system:

- **DeepFace** - A flexible Python framework providing streamlined access to state-of-the-art facial recognition and detection models such as VGG-Face, FaceNet, ArcFace, RetinaFace, and Dlib [48]. It supports both face recognition and facial attribute analysis with minimal configuration and customizable backends.

- **Yunet** - A lightweight and efficient deep learning model for face detection [62], selected for its speed and suitability for real-time applications.

- **FaceNet** - A deep learning model for extracting features and generating facial embeddings [47], used for high-performance face recognition and comparison tasks.

- **SQLite** - A lightweight relational database engine, ideal for small-scale projects where quick setup and low overhead are essential

- **Flask** - A web-application framework known for its simplicity and flexibility, used to host the system's user-facing interface

- **Scikit-learn** - A versatile Python machine learning library employed to evaluate the recognition system's performance and generate metrics, such as ROC curves [41].

- **GitHub** - Used for collaborative development, including GitHub Projects for task management, continuous integration pipelines for automated testing and style checking, and pull requests for code reviews.

- **LFW** - Labeled Faces in the Wild is a dataset containing 13,233 images of 5,749 individuals, designed to support facial recognition tasks in unconstrained settings [24]. It includes real-world variations in pose, lighting, and background, with non-overlapping training (2,200 pairs) and test (1,000 pairs) sets to ensure evaluation on unseen identities.
- **CVSS Calculator** - Common Vulnerability Scoring System (CVSS) v4.0 calculator, developed by the Forum of Incident Response and Security Teams (FIRST), was used to assess the severity of security threats in the system.

For continuous integration pipelines and environment setup, we refer the reader to Appendix B.

5.2 System Architecture

The system is divided into two distinct components: the client application and the web server. The client application is rendered in the browser and allows users to interact with the web server through an intuitive User Interface (UI). Between the client application and the web server, communication is established for the client application to modify and fetch resources on the server.

5.2.1 Client Application

The client application is implemented using HTML, Cascading Style Sheets (CSS), and JavaScript, ensuring a responsive and interactive user interface. HTML defines the structure of the application, including elements such as forms, buttons, and content layout. CSS is used to style the interface, ensuring visual consistency and responsiveness. JavaScript powers the dynamic behavior of the application, such as handling validation of forms, listening to the button clicks, performing the designated actions, facial image capture, API communication, and user interaction feedback.

To optimize performance and preserve server resources, the application performs client-side face detection using the CNN-based model TinyFaceDetector model [37], before initiating any server communication related to facial recognition. This ensures that only images containing a detectable face are sent to the server, reducing the number of unnecessary network requests and improving the overall efficiency of the system. Furthermore, the images are cropped around the visible face outline, with an added safety margin, before being sent to the server. This reduces the number of unnecessary pixels the backend needs to analyze.

The application communicates with the server through RESTful API calls to handle user registration, authentication, and session management. When a user submits the registration form, their captured facial image and inputted personal details are securely sent to the server for storage and future verification on login requests.

In addition to these HTTP requests, the client also establishes a persistent Web-

Socket connection via FlaskSocketIO specifically for the registration and login processes with facial recognition. Using this channel, the client emits named events (`'register'`, `'login'`) with the user's credentials and image data, and listens for server acknowledgments. This event-driven, bidirectional socket reduces the latency and per-message overhead of repeated HTTP requests, resulting in a faster and more seamless authentication experience.

While the client is responsible for capturing and sending the facial image, the authentication process is fully handled on the server. The system employs facial recognition to compare the newly captured image with previously stored facial features. If a match is confirmed, the server validates the user and grants access to the dashboard.

To maintain user sessions, the application uses cookie-based session management. Session cookies store user data locally in the browser, allowing users to remain logged in between sessions without the need to authenticate on subsequent visits.

Finally, the application also includes integrated error handling throughout to support a smooth user experience. If facial recognition fails or invalid data is submitted, clear and informative error messages guide the user to resolve the issue, minimizing frustration and interruptions.

5.2.2 Web Server

The backend is implemented in Python using Flask, providing endpoints that facilitate client-server communication. Some endpoints return HTML content that is displayed in the browser, while others follow REST principles to allow clients to access or update data. Additionally, certain endpoints are restricted and require user authentication before access. For example, the login endpoint enables clients to provide credentials in exchange for authentication. Table 5.1 presents the most frequently used facial recognition endpoints for registration and login in the system.

Table 5.1: Main facial recognition endpoints with their parameters and purpose

Path	**Parameters**	**Description**
`/auth/register`	username: String email: String image: JPEG ...	Register a new user. Email and username must be unique.
`/auth/login`	username: String image: JPEG	Login using facial recognition. In case of invalid credentials, an error response is returned.

The backend follows a layered architecture, as illustrated in Figure 5.1. The controller layer is composed of multiple controller classes, each responsible for handling incoming requests. Each request is routed to the appropriate controller method according to the configured Flask routing. The task of each controller is to handle the request by applying validation to the request body and then passing it to the service layer for further processing.

The service layer contains classes for handling business logic. For example, one service class handles authentication, registration, and fetch operations for users. The service layer communicates in turn with the data access layer, which contains repository classes. Each repository class extends a common superclass for Create-Read-Update-Delete (CRUD) functionality for each defined entity in the database. However, each repository class is tailored to the specific needs of the domain. It is the repository classes that interact directly with the database.

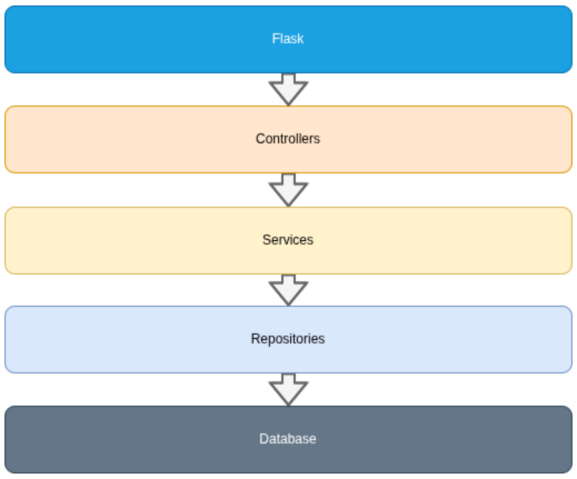


Figure 5.1: Layered architecture of the components in the backend.

5.3 Facial Recognition Pipeline

The facial recognition pipeline was implemented as an abstract class to allow flexibility in swapping between different implementations. To evaluate different architectural strategies, two concrete implementations were developed. One prioritizes integration simplicity, while the other was designed for modularity and maintainability. Both implementations inherit from the same abstract superclass, which defines a common interface: one method for extracting features from a face image and another for comparing two feature vectors to determine whether they represent the same person.

The first implementation of the pipeline superclass is called `FaceRecognition`. It utilizes DeepFace for each step of the pipeline process. The models used by DeepFace for detection and recognition are specified in the constructor arguments. This means that the models are specified at instantiation.

The second implementation, `ModularFaceRecognition`, follows the strategy design pattern by moving each algorithm in the facial recognition pipeline to separate classes. For example, for the detection part of the pipeline, the model used is interchangeable by providing an instance that extends `FaceDetectionStrategy`. The pipeline instance can then call upon each strategy which will solve each part of the facial recognition pipeline. With this design pattern, each step of the process (detection, alignment, recognition, and verification) is delegated to separate instances, contributing to modular, maintainable, and scalable code.

5.3.1 Face Image Validation

Before the usual four steps of the facial recognition pipeline, an initial image verification step was added. This step helps to ensure that the image quality is sufficient despite challenges such as lighting and camera quality. For the facial image to be accepted, it must meet the following requirements:

1. Image size must exceed 128x128. This is to enforce a certain image resolution to make sure the captured image has high enough quality to facilitate optimal facial feature extraction.
2. The face must occupy a certain percentage of the image.
3. The eyes must be aligned within 15 degrees of the horizontal line.
4. The face must be facing the camera.

To estimate whether a face is oriented toward the camera, a geometric approximation method was employed. This approach calculates yaw and pitch angles using basic trigonometric relationships between the eyes and the nose. These angles were approximated using Equations 5.1 and 5.2, respectively. The facial landmarks used in this method were extracted using YuNet.

$$yaw = \arctan\left(\frac{nose_x - eye_{center_x}}{2 \cdot eye_{dist}}\right) \tag{5.1}$$

$$pitch = \arctan\left(\frac{nose_y - eye_{center_y}}{eye_{dist}}\right) \tag{5.2}$$

5.3.2 Spoofing Detection

An anti-spoofing mechanism was implemented to secure the system against spoofing attacks. After the detection step, spoofing detection is applied to the facial region. The process is executed entirely on the server side, ensuring that users cannot bypass, manipulate, or disable this security layer. In case of a spoofing attempt, the pipeline raises an error and an appropriate error response will be returned to the client. Under the hood, the spoofing detector object utilizes an anti-spoofing deep learning model called FASNet which claims an TPR of 99.7% [1]. Papers on similar spoofing detection mechanisms have achieved similar results [25].

The FASNet model combines both analysis in the frequency domain and the spatial domain [1]. Analysis of texture patterns, color distribution, and motion within facial images is a part of spatial feature analysis [25]. Frequency domain analysis, on the other hand, detects irregularities in the frequency representation of a facial image. Examples of such irregularities include moiré effects on digital screens and the absence of natural skin texture.

To take advantage of analysis in both the spatial and frequency domain, FASNet is divided into two branches [1]. The first branch applies Fourier transform on the facial image. The output is the image in the frequency domain and is used as a ground truth label during training. The second branch is a CNN for face spoofing classification given a facial image. During training, feature maps generated from the convolutional layers are inputted into another CNN with three convolutional layers. The output of this smaller network is compared to the ground truth from the first branch, and a loss function is computed. Losses from both branches are combined to form the optimization function. It is this function that is minimized during training. As a result, the FASNet learns to spoof characteristics in both the spatial and frequency domain, thus enhancing detection accuracy [25]. An overview of the architecture of FASNet is illustrated in Figure 5.2.

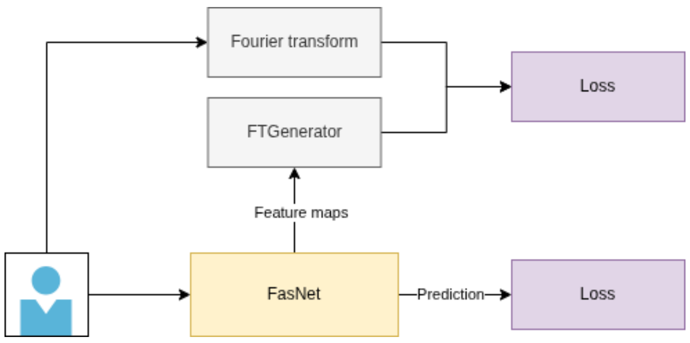


Figure 5.2: Overview of the architecture of FASNet.

5.4 Database

The SQLite database is structured with multiple tables to support persistent storage. Each table is designed to store distinct types of information, enabling logical separation and efficient data management. For example, the users table contains personal information provided during registration and is presented in Table 5.3. The data types used in the database are explained in Table 5.2.

Table 5.2: Explanation of common data types used in the database

Data Type	**Description**
INTEGER	Stores whole numbers (positive or negative) without decimal points.
VARCHAR(#)	Variable-length character string that can store up to # characters.
BLOB	Binary Large Object used to store large binary data such as images, audio files, or encrypted information like facial embeddings.

Table 5.3: Overview of the users table schema.

Column Name	Data Type	Description
`id`	INTEGER	Unique user identifier
`user_name`	VARCHAR(30)	User's username
`email`	BLOB	The encrypted email of the user
`first_name`	BLOB	First name of the user (encrypted)
`last_name`	BLOB	Last name of the user (encrypted)
`password`	BLOB	Password (encrypted)
`face_embedding`	BLOB	Face embedding extracted at registration (encrypted)
`twoFA`	BOOLEAN	Whether the user has enabled 2FA
`created_at`	DATETIME	Time of user registration

The users table has three one-to-many relationships with other tables used in the system such as `login_log` table, `log_book_entries` table, and `rsa_public_keys` table. The relationships between the tables are illustrated in Figure 5.3.

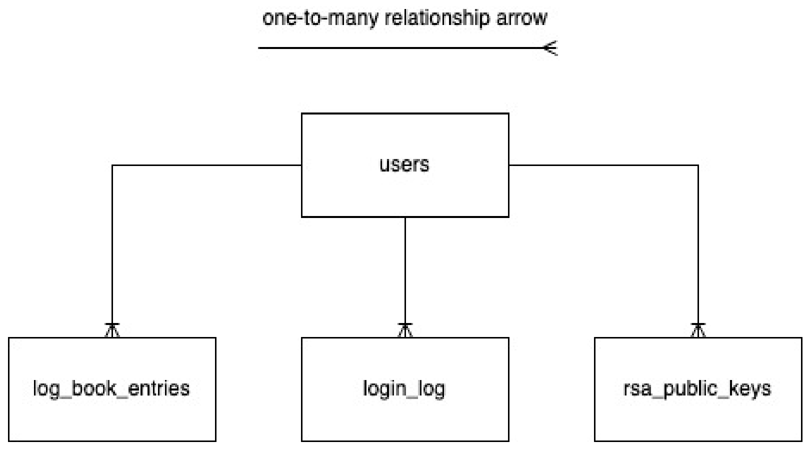


Figure 5.3: The entity relationships between different tables in the database.

Cryptography plays a key role in providing protection against data intrusions. Cryptography protects data from unauthorized access, ensuring integrity, confidentiality, and authenticity [40]. In simple terms, cryptography is the process of converting readable data (plaintext) to an unreadable format (ciphertext) through encryption, and reversing the process during decryption.

This implementation employs symmetric encryption, a method in which the same single key is used for both encryption and decryption operations [8]. Sensitive user data includes credentials and personal identifiers, which are encrypted before being stored in the database as Table 5.3 describes.

In the system, encryption is implemented using `Fernet` module from the Python `cryptography` library. Fernet combines Advanced Encryption Standard (AES) with a 128-bit key and Hash-based Message Authentication Code (HMAC)-SHA256 for securing encryption and integrity verification. If no key is found during system initialization, a new key is securely generated and stored locally for future use. Figure 5.4 illustrates a high-level overview of the encryption and decryption process.

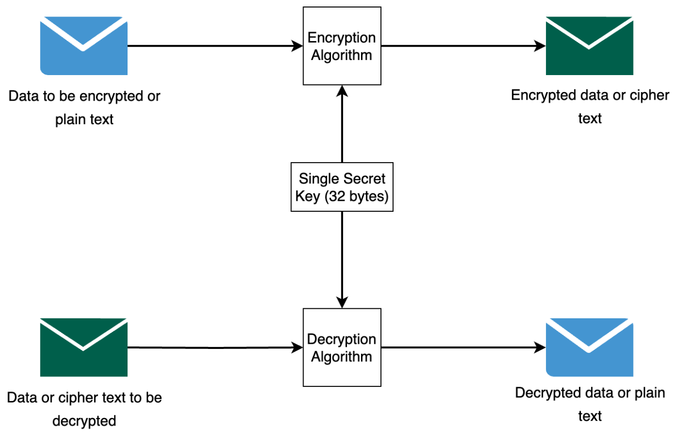

Data to be encrypted or plain text
Encryption Algorithm
Encrypted data or cipher text
Single Secret Key (32 bytes)
Data or cipher text to be decrypted
Decryption Algorithm
Decrypted data or plain text

Figure 5.4: An overview of encryption/decryption process using a single key.

5.5 Alternative Login Methods

In addition to facial recognition, the prototype supports MFA using OTP, as well as fallback authentication methods to sign in. These include password-based login, key-based challenge signing, and Single Sign-On (SSO) with Google. These alternatives enhance accessibility and system robustness in scenarios where facial recognition is impractical, such as low-light environments or the absence of a functioning webcam.

5.5.1 Password Authentication

After registration of a new account, users have the opportunity to enable password authentication. The system enforces standard password security requirements, including a minimum length of 8 characters, at least one uppercase letter, and one special character. Upon enabling password authentication, the user is required to use Two-Factor Authentication (2FA).

5.5.2 Key-based Challenge Signing

RSA key signing allows users to authenticate through a cryptographic challenge-response protocol. Before this can be done, the user must generate a RSA public-private key pair and authorize the public key in the settings in the web application. The two keys can be generated through a tool such as OpenSSL. The private key will never leave the user's computer.

Once a public key is authorized, the user can log in using key-based challenge signing. Upon entering their username and selecting their private and public key, a challenge is requested using the fingerprint of the selected public key. The fingerprint is a short sequence of bytes that uniquely identifies a longer public key. The server instantiates a session for the challenge signing and stores the fingerprint for verification in later steps. Then, the client signs it with the corresponding private key. The server verifies the signature using the public key to confirm the user's identity. This ensures that only the holder of the private key can authenticate. The full authentication flow is shown in Figure 5.5.

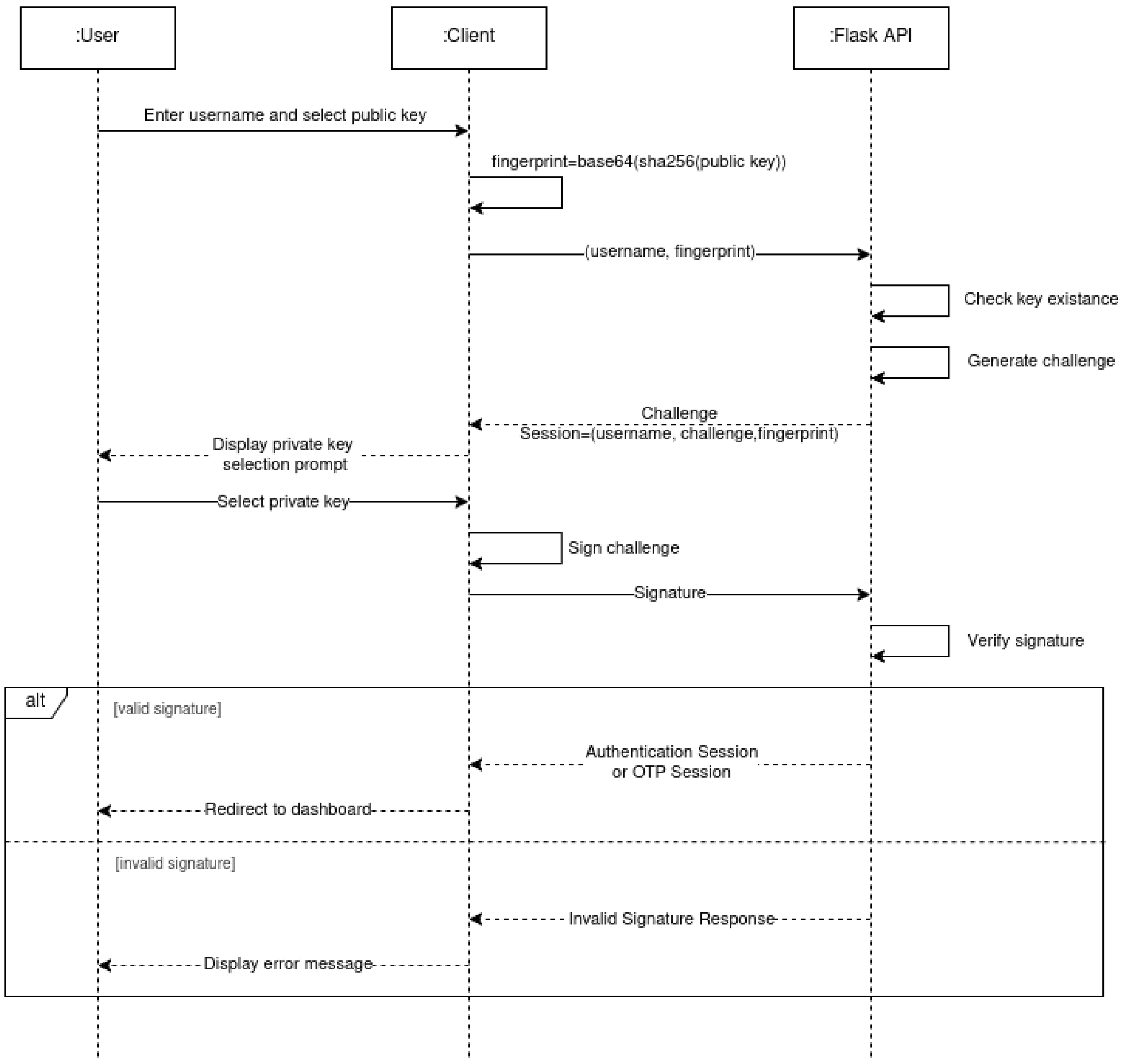


Figure 5.5: Log in flow for RSA key signing.

5.5.3 Single Sign-On

SSO is an authentication method that allows users to access multiple applications with a single set of credentials. By integrating with an identity provider such as Google, the prototype acts as a service provider since it delegates authentication to a trusted third party.

Currently, only Google is available as an SSO alternative on the platform. When a user selects the option to sign in with Google, they are redirected to Google's authentication interface, where they choose an account and authenticate using their Google credentials. Upon successful authentication, Google displays a consent screen outlining the data the application is requesting access to (username, full name, e-mail address), which the user must agree to in order to proceed.

Once consent is provided, Google generates a secure token set that is returned to the application via a secure redirect URI. This includes an ID token (used to identify the user) and optionally an access token (if further API access is requested). The platform then verifies the token's validity and retrieves the aforementioned user information. If the token is valid and the user is signing in for the first time, the user is also registered in the database before being logged in.

This implementation uses OpenID Connect (OIDC), an identity layer built on top of the OAuth 2.0 protocol. While OAuth 2.0 provides delegated authorization, OIDC extends it by allowing the prototype to authenticate users through a secure ID token.

5.5.4 Multi-Factor Authentication

The system utilizes MFA by combining the login methods with a OTP to enhance security. Figure 5.6 presents an example involving facial recognition. Classified as an inherence-based factor, facial recognition serves as the primary authentication method, while the TOTP acts as the secondary factor. After signing in using the first method, a TOTP is generated via the pyotp library and sent to the registered email address using Flask-Mail. The TOTP is time-sensitive and remains valid for 60 seconds before expiring. During this period, the user must enter the six-digit code to complete authentication. To implement this functionality, session-based tracking is used to temporarily store the TOTP secret and user identification until verification is achieved. Upon successful TOTP entry, the session data is cleared and access is granted. This dual-layer approach ensures that even if one factor is compromised, an intruder must still bypass the second to gain access.

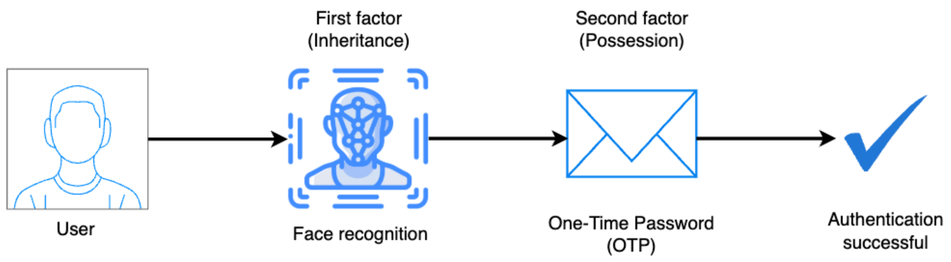


Figure 5.6: Sign-in process using 2FA.

6 Result

The outcome of the prototype is a functional authentication application that enables users to register an account and securely log in using facial recognition as the primary method. In addition to biometric authentication, the system supports alternative login methods to enhance accessibility and accommodate diverse user needs.

This chapter presents a detailed evaluation of the prototype's performance, focusing on key aspects such as login speed, recognition accuracy, and system responsiveness. It compares different implementation approaches to highlight differences in efficiency and reliability. The chapter also describes the threat model used to identify security risks and covers the user interface.

6.1 Performance

In a user application, providing quick feedback is a priority. Therefore, it is crucial for the login process to execute quickly on the system. The most computationally intensive part of the login process is facial recognition and verification, which is handled by the facial recognition pipeline.

6.1.1 Login Speed and Responsiveness

As discussed earlier in Section 5.3, two different pipeline implementations were created. The first, `FaceRecognition`, uses the DeepFace library, while the second implementation, `ModularFaceRecognition`, is customized to fit the needs of the project. Although the DeepFace implementation required less engineering effort, it became clear that its performance in terms of speed was suboptimal. To address this limitation, the alternative implementation was developed.

The speed differences between the two pipelines for consecutive login attempts are presented in Figure 6.1. The initial login attempt using `FaceRecognition` takes around 5 seconds to complete, whereas the modular pipeline requires only about 0.35 seconds. For subsequent attempts, the DeepFace implementation consistently takes around 1.5 seconds, while `ModularFaceRecognition` remains at approximately 0.35 seconds. The spike at attempt 16 for the DeepFace implementation is likely due to a temporary increase in CPU usage on the system at that time.

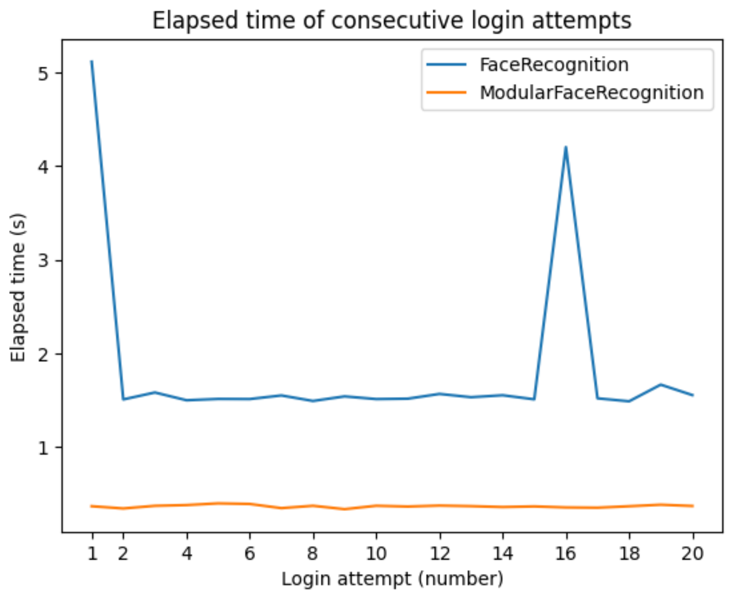


Figure 6.1: Login time for 20 attempts using `FaceRecognition` and `ModularFaceRecognition`.

`ModularFaceRecognition` is not only faster but also more consistent in its login times, as demonstrated by the results from 20 consecutive login attempts shown in Figure 6.2. The figure illustrates that `FaceRecognition` has a significantly higher standard deviation, meaning its login times are more spread out, while `ModularFaceRecognition` shows lower variation and greater stability. The black bars indicate the standard deviation for each set of measurements, and the text above each bar displays the average time.

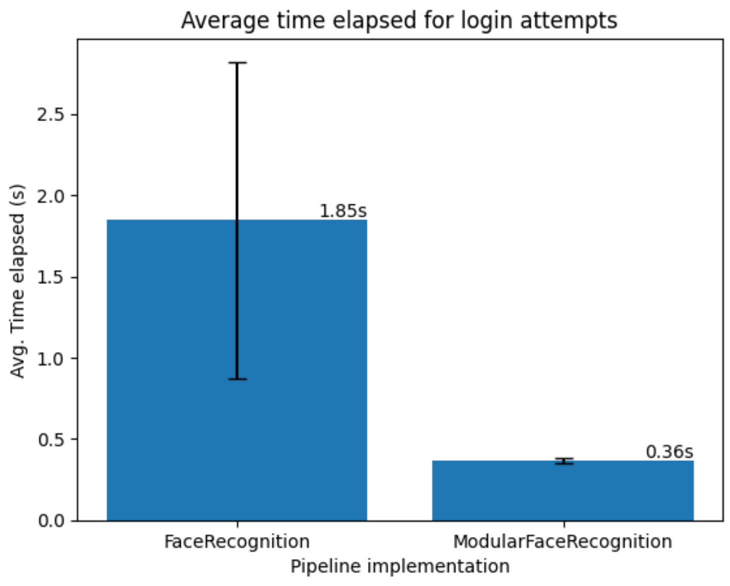


Figure 6.2: Comparisons of two pipelines and their average time for login attempts

6.1.2 Accuracy and Trade-Offs

Although speed is important, having an accurate facial recognition pipeline is crucial for security. To compare the two implementations in terms of accuracy, both were evaluated using the LFW dataset. Figure 6.3 shows the ROC curves for each pipeline. `FaceRecognition` performs slightly better, with a higher AUC score than `ModularFaceRecognition`. However, since the difference is relatively small and the latter is over 5 times quicker, `ModularFaceRecognition` is considered the overall best option.

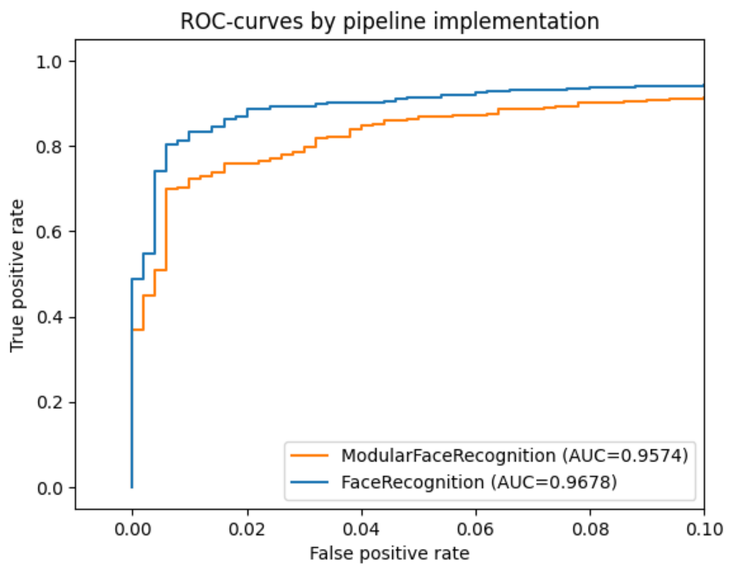


Figure 6.3: The ROC curves for the two pipelines implementations evaluated on LFW.

6.1.3 Threshold Evaluation

The `ModularFaceRecognition` class was evaluated on LFW by computing the distance between each face pair in the dataset. The resulting distances were used in conjunction with three threshold selection methods to find an appropriate threshold for the developed facial authentication system. Optimally, the threshold minimizes FPs for security and maximizes TPs for usability.

The resulting thresholds are presented in Table 6.1, together with their corresponding classification metrics. Both Youden's J statistic and the method of minimizing the distance to the upper left corner of the ROC space produced identical thresholds and performance scores. Among the evaluated metrics, precision is of particular importance. For both methods, the precision was calculated to be 92%, indicating that 8% of the positive predictions were false positives. In practical terms, this means that in 8% of the cases, an unauthorized user was incorrectly granted access to another individual's account.

Applying the constrained based approach on the other hand, yielded a precision score of 99.2%. This implies that in only 0.8% of the cases, an unauthorized user gained access to another individual's account. However, the increased precision score comes at a cost of a decreased recall score. For this threshold, the recall score was 51%, which means that in half of the login attempts, a user did not gain access to their account.

Table 6.1: Performance metrics for the pipeline on LFW with threshold yielded from 3 different methods

Method	Threshold	Accuracy	**Precision**	Recall	F1-Score
Youden's J Statistic	14.231	0.911	**0.920**	0.900	0.910
Minimize Distance to (0, 1)	14.231	0.911	**0.920**	0.900	0.910
Constraint-based (FPR $\leq$ 0.5%)	11.255	0.753	**0.992**	0.510	0.674

6.2 Threat Model

To assess the security of the system, a structured threat model was developed. Security threats were evaluated using CVSS v4.0, incorporating both base and environmental metrics to capture technical severity and context-specific mitigations. This method provided a quantitative and reproducible assessment of each identified threat relevant to the system with regard to potential real-world deployment. Figure 6.4 visualizes the identified threat landscape and potential attack vectors targeting the system, while Table 6.3 presents the corresponding CVSS-BE scores and metric breakdowns.

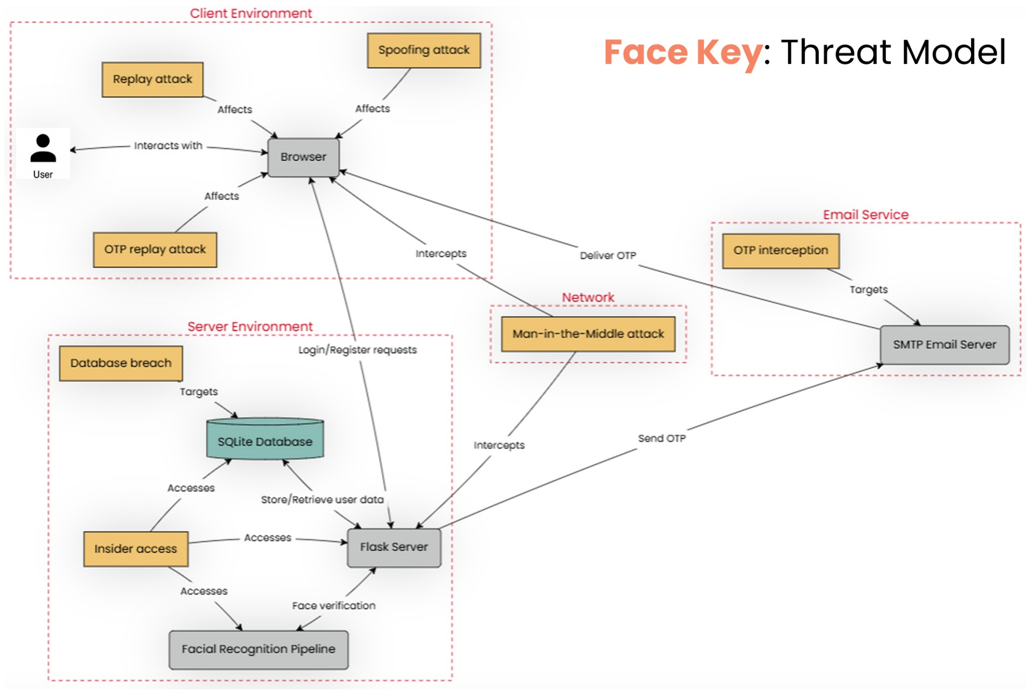


Figure 6.4: Threat model diagram illustrating the potential attack vectors targeting main components of the authentication system.

The system assumes the following potential adversaries:

- External attackers attempting to impersonate users or intercept sensitive data during communication.
- Local attackers who may have physical access to a user's device.
- Insiders with partial system access who may attempt to bypass restrictions or extract stored data.

The results presented in Table 6.3 summarize the severity of each identified threat targeting the developed authentication system, as evaluated using CVSS v4.0. The description of the metrics' impact is presented in Table 6.2.

Table 6.2: CVSS Metrics and Their Descriptions

Metric	Value	Description
Attack Vector	Network	The attack is exploitable remotely over a network.
	Adjacent	The attack requires access to the same shared physical or logical network.
	Local	The attacker must have local access to the device or system.
	Physical	The attacker must physically interact with the target.
Attack Complexity	Low	The attack can be executed reliably and repeatedly.
	High	Complex to exploit; security mechanisms must be bypassed.
Attack Requirements	None	No special environmental conditions are needed.
	Present	Some environmental conditions must exist.
Privileges Required	None	Attacker is unauthorized prior to the attack.
	Low	Low privileges are sufficient to perform the attack.
	High	Significant control or privileges over the system are required.
User Interaction	None	No user interaction is needed to exploit the system.
	Passive	Limited, involuntary user interaction is required.
	Active	Specific, conscious interaction from a targeted user is required.
Confidentiality Integrity Availability	None	No impact on the respective attribute.
	Low	Limited or partial impact.
	High	Significant impact.

To specifically tailor to the potential real-world deployment of the system, CVSS-BE scores have been used. Base metrics such as Attack Vector, Attack Requirements, Privileges Required, User Interaction, and the impacts on Confidentiality, Integrity, and Availability were used to establish initial severity. Environmental metrics, including the modified version of each base metric, were then applied to reflect the system's security measures, such as the use of HTTPS, encryption and hashing of sensitive data, and anti-spoofing mechanisms. The base metrics were manually filled with the technical characteristics of the attack itself independent of the specific system context, while the environmental metrics were filled in to reflect the developed system's architecture, threat landscape, and implemented security measures.

Table 6.3: CVSS-BE scores and metric breakdowns for each identified security threat.

Metric	Replay	Spoofing	Model Inversion	Database Breach	MitM
Attack Vector	Network	Network	Network	Network	Network
Attack Complexity	Low	Low	High	High	Low
Attack Requirements	Present	Present	None	None	None
Privileges Required	None	None	Low	High	None
User Interaction	None	None	None	None	None
Confidentiality	Low	Low	High	High	High
Integrity	Low	Low	High	High	High
Availability	Low	None	None	High	Low
Modified Attack Vector	Network	Network	Network	Network	Network
Modified Attack Complexity	Low	High	High	High	Low
Modified Attack Requirements	Present	Present	None	None	None
Modified Privileges Required	None	None	Low	Low	None
Modified User Interaction	None	None	None	None	None
Modified Confidentiality	Low	Low	High	Low	Low
Modified Integrity	None	None	High	Low	Low
Modified Availability	None	None	None	Low	Low
CVSS-BE Score	6.9 [19]	6.3 [20]	7.6 [18]	2.3 [16]	6.9 [17]

6.3 User Interface and Core Functionality

The client application is designed to support secure and efficient login with a primary focus on facial recognition. The user interface facilitates multiple authentication methods, with face recognition as the default, while offering users the ability to add an additional security layer through OTP verification.

Upon login, users are presented with a clean and straightforward screen illustrated in Figure 6.5. The interface clearly guides users through the authentication process, whether they are logging in with facial recognition, password, RSA key, or verifying a TOTP. This modular approach to login makes it possible to balance user convenience with security requirements, adapting to different environments or user preferences. The registration interface is straightforward as well, where at the end of the process, the user is required to register their identity using facial recognition.

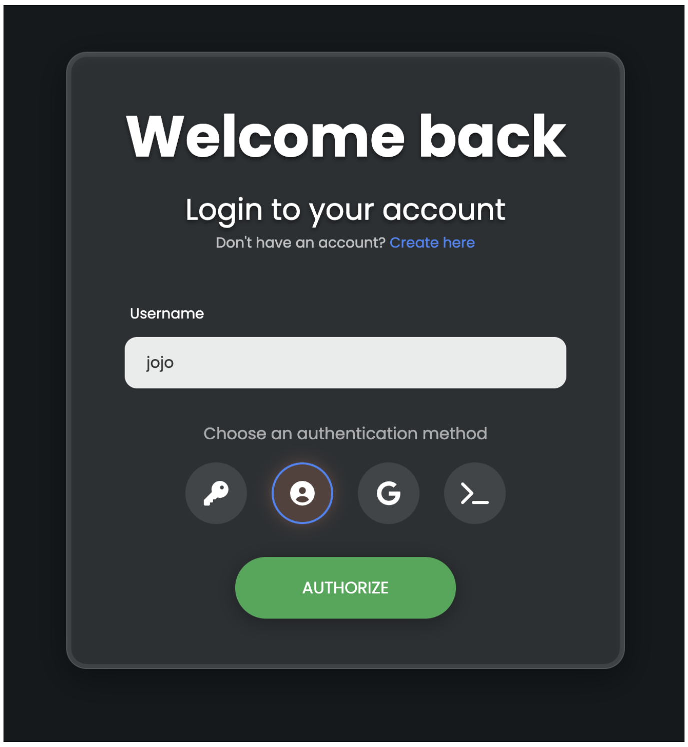

Welcome back
Login to your account
Don't have an account? Create here
Username
jojo
Choose an authentication method
AUTHORIZE

Figure 6.5: Login page

When choosing facial authentication, users are instructed to face the camera in a well-lit environment and ensure only one face is visible during capture. If multiple faces are detected, the system automatically selects the one most likely to match a registered identity or prompts the user to reposition. Feedback is displayed in real-time, informing the user whether the authentication was successful, failed, or if conditions (e.g., face angle) need adjustment. In addition, if the system detects a potential spoofing attempt, a corresponding alert is displayed on the screen as Figure 6.6 shows.

In cases where facial recognition fails or the user opts for an alternative method, the UI allows fallback to other authentication methods. As described in Section 5.5.4, when OTP is enabled, the user receives a one-time code via their registered email to verify their identity, adding an additional layer of protection against unauthorized access. This process occurs when the selected authentication method is granted access. The email the user receives is shown in Figure 6.7.

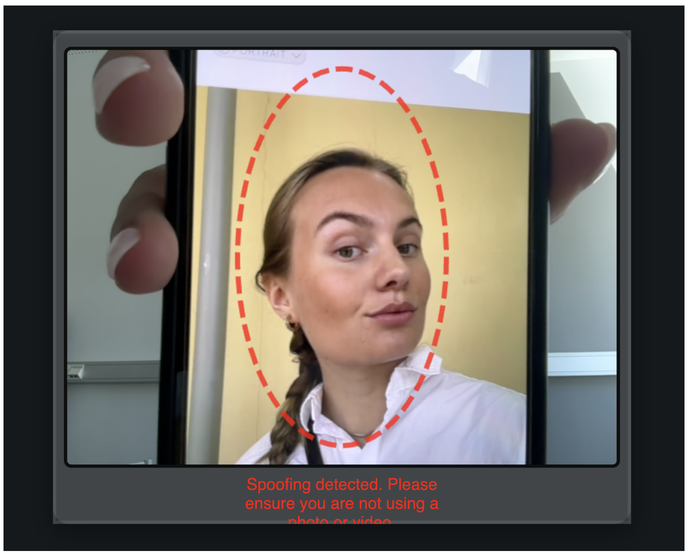
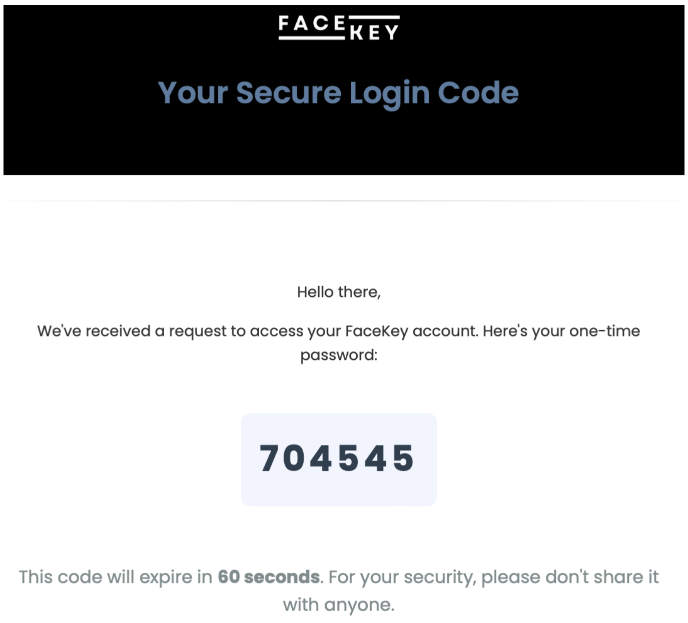


Figure 6.6: Spoofing detected



FACE KEY

Your Secure Login Code

Hello there,

We've received a request to access your FaceKey account. Here's your one-time password:

704545

This code will expire in **60 seconds**. For your security, please don't share it with anyone.

Figure 6.7: The email the user receives six-digit code.

While the main focus of the project was not on UI/UX design, visual and interactive elements were implemented with usability in mind. A dark color scheme, clear typography (Open Sans), and consistent visual cues (hover effects and color-coded messages) were used to ensure readability and intuitive navigation throughout the application. For visual examples and the final implementation of each page, please refer to the screenshots provided in Appendix C.

7 Discussion

This chapter discusses key aspects of the proposed prototype, building on the results presented in the previous chapter. It outlines the system's limitations, as well as its societal and ethical implications. Finally, potential future improvements are considered.

7.1 Major Findings

The project successfully developed a facial recognition-based authentication system with multiple fallback options, balancing usability, performance, and security. One of the most notable outcomes was the superior responsiveness of the custom `ModularFaceRecognition` pipeline. It outperformed the DeepFace-based implementation by achieving more than five times faster login times and lower latency variability. This result is highly relevant for user-facing authentication systems, where speed and predictability are essential for user satisfaction.

Despite a small decrease in accuracy compared to DeepFace, ModularFaceRecognition maintained 91% accuracy on the LFW dataset and reached a precision of 0.992 when using a stricter threshold. This trade-off prioritizes security by minimizing false acceptances — a critical factor in preventing unauthorized access.

In addition to biometric verification, the system integrates multiple authentication options including password login, RSA key signing, SSO with Google, and a basic anti-spoofing mechanism based on FASNet. These features collectively create a layered security architecture, demonstrating how different methods can complement each other to enhance overall robustness.

The CVSS-based threat model from Section 6.2 revealed several vulnerabilities, with MitM attack representing one of the higher security risks. HTTPS was implemented to secure communication channels, significantly reducing the risk of data interception and tampering by ensuring encrypted transmission of data. Other relatively high-scoring risks, including model inversion attacks and replay attacks, remain partially addressed through encryption, limited data exposure, and anti-spoofing measures.

The integration of an anti-spoofing mechanism significantly improves the system's defense against static image replays, deepfakes, and similar forms of impersonation. Although this reduces the feasibility of spoofing attacks, the actual defense performance depends on the model's robustness under diverse real-world conditions. Configuring the anti-spoofing threshold proved challenging. If set too high, the system falsely flagged real users as spoofing attempts. If set too low, it occasionally accepted photos displayed on mobile devices. These observations highlight the sensitivity of the mechanism and the need for further calibration and real-world testing.

The CVSS-BE scores supported these findings. Spoofing scored lower than most other threats, mainly due to the countermeasures of the system. Replay attacks and model inversion pose moderate threats, with the latter stemming from the fact that even limited access to the model could, under certain conditions, allow attackers to reconstruct facial features. This is reflected in the elevated confidentiality and integrity impact metrics. Although biometric templates are hashed and encrypted in the local database and privileges are restricted, model inversion remains a relevant concern, especially as inversion techniques continue to grow more advanced [63].

The combination of biometric and token-based verification methods creates a layered defense strategy. To further strengthen the system, hardened database access controls and more advanced anti-spoofing strategies, such as blink detection, head movement analysis, or randomized challenge-response protocols, could be implemented. Furthermore, while encryption protects stored data, exploring privacy-preserving biometric embedding methods, such as obfuscating the biometric data before storage, could further mitigate the risks associated with database breaches and reconstruction-based attacks [45] such as model inversion. These improvements would enhance the system's readiness for deployment in security-critical settings while addressing the residual risks identified through CVSS analysis.

7.2 Comparisons to Related Work

The system builds upon previous studies such as Siddharth and Khankan [52], which proposed an MFA solution but lacked modularity and spoof resistance. In contrast, this project emphasized flexibility, model interchangeability, and real-time performance, addressing gaps identified in prior literature. Moreover, the ROC and AUC evaluations aligned with best practices outlined in studies by Serengil and Özpinar [49], reinforcing the use of modular pipelines and empirical threshold tuning.

7.3 Societal and Ethical Concerns

While the developed facial recognition-based authentication system demonstrates promising performance in a controlled environment, its broader adoption requires a critical evaluation of the societal and ethical implications. As mentioned in Section 3.5, facial recognition technology poses significant concerns about privacy, bias, and potential misuse in surveillance contexts.

7.3.1 Privacy and Data Integrity

Given the immutable and sensitive nature of biometric data, the system was designed with security safeguards such as AES encryption and hashing. The use of HTTPS ensures secure transmission between the client and the server. These measures protect facial embeddings and prevent unauthorized access or tampering. Additionally, the current system only accepts user-submitted data for registering and login use with explicit consent, with no continuous monitoring or background tracking. Strong encryption alone is not sufficient, as without strict consent, biometric data collection can still potentially diminish public trust.

7.3.2 Bias and Discrimination

Although FaceNet was chosen based on strong empirical performance, the model inherits the limitations of its training data. As noted in Section 3.5, facial recognition systems often demonstrate decreased accuracy for underrepresented demographic groups. In this project, efforts were made to standardize inputs. This was done by instructing users to submit biometric data under consistent conditions, specifically by ensuring that they are close enough to the camera, front-facing, and centered. These distance and alignment constraints were implemented to help minimize performance variance. However, such controls do not solve the underlying algorithmic bias. To improve fairness, future iterations should include model retraining or fine-tuning using demographically balanced datasets, along with extensive performance evaluation against diverse user groups.

7.3.3 Surveillance and Public Trust

The use of facial recognition, particularly when deployed beyond voluntary contexts, raises concerns about public surveillance. Even when implemented for legitimate purposes, such as crime prevention, these systems risk being repurposed for intrusive monitoring. Recent cases involving facial recognition at protests or via home surveillance devices [21] illustrate the potential for abuse. Although our system avoids these practices, it operates within a broader technological and societal landscape that must be considered. Ethical design demands more than technical correctness and also requires accountability and safeguards to protect the freedom of individuals.

Furthermore, the accuracy of facial recognition technology can vary and is not 100% accurate, which means that there are both FPs and FNs. FPs, where an illegitimate individual is granted access, can undermine the security of the system, while FNs, where legitimate users are denied access, can lead to frustration and decreased user satisfaction. These inaccuracies highlight the potential risks in relying solely on facial recognition for authentication, particularly in high-security applications where both types of errors could have significant consequences. To mitigate the risk of illegitimate individuals being incorrectly authorized, the authentication system employs optional MFA, meaning that even in the case of FPs there is another factor of authentication.

7.4 Limitations

Although the system has been successfully implemented, certain limitations remain. These limitations became evident throughout the development process and persist even in the final version of the system. The primary constraints concern the selection of models and the limited extent of user testing, both of which have a direct impact on the system's performance.

7.4.1 Model Constraints

One of the key limitations in this work concerns the selection and evaluation of facial detection and recognition models. The models were systematically evaluated and selected as described in Section 4.1. However, although the decision was based on objective data, the resulting metrics are entirely based on the application on the LFW dataset. The dataset contains faces from varying angles, lighting, and pose, which makes it ideal for evaluating models for detecting and recognizing faces in unconstrained conditions.

The application of this authentication system differs slightly due to the fact that the system exerts partial control over captured images. Since there is interaction between the user and the system, the system can provide feedback such as that the user must look straight-forward. As a result, most images that are used in the facial recognition pipeline tend to share a similar angle and pose. Evaluating the pipeline on a dataset containing strictly frontal and centered face images would provide metrics more precise to the application of the system. In turn, other models might out-perform YuNet and FaceNet in this particular application.

Another limitation in model selection is that the evaluation was constrained to using Euclidean distance. It is possible that one model combination with cosine similarity, for example, would perform better than the selected models. This is because the distance metrics perform differently depending on the models. However, due to time and hardware constraints, comparing all different model combinations with all distance metrics would not be feasible.

7.4.2 User Testing

Another notable limitation is the minimal amount of user testing conducted during development. While the system was designed with usability and responsiveness in mind, its performance and user experience have not been extensively validated through real-world interaction. This limits the ability to assess how intuitive, accessible, and reliable the system is across a diverse user base. Without broader testing, potential issues related to camera quality, lighting conditions, or user behavior may have gone unnoticed. More comprehensive user testing would be necessary to evaluate the system's effectiveness and robustness in varied, practical scenarios.

7.5 Future Work

While the current system provides a solid foundation, several areas can be explored and expanded upon in future iterations. Enhancing its functionality and security features, as well as extending its reach across different platforms, will contribute to its long-term effectiveness and adaptability in real-world applications.

7.5.1 Anti-Spoofing Improvement

The current system could be enhanced by adding a more robust anti-spoofing feature to address vulnerabilities such as deepfakes or photo-based impersonation. Future development could include implementing blink recognition and challenge-response mechanisms, strengthening the system's defense against these attacks. Incorporating these measures would elevate the system's security, making it more robust and trustworthy for high-stakes applications.

7.5.2 Scalability and Deployment

In the future, the application could be expanded to support both mobile and desktop platforms, allowing users to authenticate across a wide range of devices. Additionally, it could integrate with other applications, enabling users to log in seamlessly via the system as part of a broader identity management system. Continuing to develop this project could lead to it becoming a key authentication solution for various services, offering flexibility and scalability as it adapts to the needs of both individual users and businesses.

7.5.3 Model Fine-tuning

To improve fairness and accuracy in diverse user groups, future development should include fine-tuning of the facial recognition model using additional datasets that reflect a wider range of demographic characteristics, such as skin tones, facial structures, and lighting conditions. By adapting the model using more representative training data, the system can better generalize to real-world users and reduce potential algorithmic bias. This would enhance the reliability of the system and help ensure consistent performance between various demographic groups.

8 Conclusion

In this thesis, an implementation of a facial recognition-based authentication system is presented. It successfully combines the DAR pipeline and the verification stage in order to efficiently handle facial data input. The system offers MFA in the form of a TOTP to strengthen security against spoofing. To ensure confidentiality and integrity of stored user data, the AES algorithm is used at registration for encrypting sensitive user data at rest. To ensure flexibility and ease of access, the platform also offers alternative sign-in methods, including password authentication, RSA key signing, and SSO with Google.

Based on a comprehensive literature review and a series of group-designed tests, YuNet and FaceNet were identified as the most effective detection and recognition models, respectively. The facial recognition pipeline was initially implemented using the DeepFace library at every stage. However, peer feedback from prototype testing revealed that the registration and login process lacked the responsiveness expected by end users. This prompted the development of a second implementation that allows full control over each stage of the pipeline. Even though the accuracy was slightly lower for the second implementation, the considerable speed improvement deems it a worthwhile trade-off. The consistency of the registration and login times also makes the second implementation more desirable.

Several areas of the system face limitations due to time and resource constraints. A more comprehensive model selection process would involve using a dataset better aligned with the system's requirements, such as facial angle, pose, and a broader range of demographic characteristics. Evaluating alternative distance metrics, such as cosine similarity, on the previously discarded recognition models could also yield improved performance. Additionally, integrating advanced techniques such as blink detection could enhance the system's already existing anti-spoofing capabilities. Gathering user feedback would further support this effort by identifying missing features and guiding improvements to the user interface.

In conclusion, the developed authentication system demonstrated the practical viability of facial recognition as a secure and accessible authentication method. It enabled meaningful evaluation of detection and recognition models within a modular framework and offered insight into the complexities of real-world system deployment. Lastly, discussions surrounding privacy, bias, surveillance, and related ethical concerns highlighted broader implications of deploying this technology.

Bibliography

[1] Minivision AI. Silent face anti-spoofing. `https://github.com/minivision-ai/Silent-Face-Anti-Spoofing`, 2020. Accessed: 2025-05-03.

[2] Zahid Akhtar, Christian Micheloni, and Gian Luca Foresti. Biometric liveness detection: Challenges and research opportunities. *IEEE Security & Privacy*, 13(5):63–72, 2015.

[3] Zahid Akhtar, Christian Micheloni, and Gian Luca Foresti. Biometric liveness detection: Challenges and research opportunities. *IEEE Security & Privacy*, 13(5):63–72, 2015.

[4] R. Alrawili, A. A. S. AlQahtani, and M. K. Khan. Comprehensive survey: Biometric user authentication application, evaluation, and discussion. *Computers and Electrical Engineering*, 119:109485, 2024.

[5] Reem Alrawili, Ali Abdullah S. AlQahtani, and Muhammad Khurram Khan. Comprehensive survey: Biometric user authentication application, evaluation, and discussion. *Computers and Electrical Engineering*, 119:109485, 2024.

[6] Ankita B, ChienChen H, Lauren P, and Lydia O. Impact of explainable ai on reduction of algorithm bias in facial recognition technologies, 2024.

[7] Fabio Bacchini and Ludovica Lorusso. Race, again: how face recognition technology reinforces racial discrimination. *Journal of Information, Communication and Ethics in Society*, 17(3):321 – 335, 2019.

[8] Mohammad Ubaidullah Bokhari and Qahtan Makki Shallal. A review on symmetric key encryption techniques in cryptography. *International journal of computer applications*, 147(10), 2016.

[9] Ben Bradford, Julia A Yesberg, Jonathan Jackson, and Paul Dawson. Live facial recognition: Trust and legitimacy as predictors of public support for police use of new technology. *The British Journal of Criminology*, 60(6):1502–1522, 05 2020.

[10] Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. volume 81, page 77 – 91, 2018.

[11] MartinThoma cmglee. Roc curve, 2018. Wikimedia Commons, Licensed under CC BY-SA 4.0.

[12] S. Dhanalakshmi and S. Chitra Devi. Tracking down high coverage configuration using clustering and fault detection. In *2015 Online International Conference on Green Engineering and Technologies (IC-GET)*, pages 1–5, 2015.

[13] Martin Drahansky. Liveness detection in biometrics. In Girija Chetty and Jucheng Yang, editors, *Advanced Biometric Technologies*, chapter 9. IntechOpen, Rijeka, 2011.

[14] Luka Dulčić. Face recognition with facenet and mtcnn, 2019. Accessed: 2025-05-02.

[15] Michael Evans. Official portrait of president reagan 1981, 1981. Wikimedia Commons, Public Domain.

[16] FIRST.ORG. Common Vulnerability Scoring System Version 4.0 Calculator: Database breach Attack, 2023. Accessed: 2025-05-15.

[17] FIRST.ORG. Common Vulnerability Scoring System Version 4.0 Calculator: Man-in-the-Middle (MitM) Attack, 2023. Accessed: 2025-05-15.

[18] FIRST.ORG. Common Vulnerability Scoring System Version 4.0 Calculator: Mondel Inversion Attack, 2023. Accessed: 2025-05-15.

[19] FIRST.ORG. Common Vulnerability Scoring System Version 4.0 Calculator: Replay Attack, 2023. Accessed: 2025-05-15.

[20] FIRST.ORG. Common Vulnerability Scoring System Version 4.0 Calculator: Spoofing Attack, 2023. Accessed: 2025-05-15.

[21] Asante Fola-Rose, Enoch Solomon, Keshawn Bryant, and Abraham Woubie. A systematic review of facial recognition methods: Advancements, applications, and ethical dilemmas. In *2024 IEEE International Conference on Information Reuse and Integration for Data Science (IRI)*, pages 314–319, 2024.

[22] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, page 1322–1333, New York, NY, USA, 2015. Association for Computing Machinery.

[23] Guosheng Hu, Li Zhang, and Li Zhang. When face recognition meets deep learning: An evaluation of convolutional neural networks for face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(10):2114–2127, 2016.

[24] Gary B Huang, Marwan Mattar, Tamara Berg, and Erik Learned-Miller. Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments.

[25] Zhanseri Ikram. Dual-Domain Face Anti-Spoofing with Integrated Spatial and Frequency Analysis Neural Network. In *2024 IEEE 4th International Conference on Smart Information Systems and Technologies (SIST)*, pages 228–232, May 2024.

[26] Internet Engineering Task Force (IETF). Internet security glossary, version 2. RFC 4949, 2007.

[27] Anil K. Jain, Arun Ross, and Salil Prabhakar. An introduction to biometric recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(1):4–20, 2004.

[28] Prashant Johri and Ms. Sonia Arora. Review of the issues and a thorough investigation of biometric authentication systems. In *2022 International Interdisciplinary Humanitarian Conference for Sustainability (IIHC)*, pages 892–897, 2022.

[29] Vyron Kampourakis, Georgios Kambourakis, Efstratios Chatzoglou, and Christos Zaroliagis. Revisiting man-in-the-middle attacks against https. *Network Security*, 2022(3), 2022.

[30] Pramod Ambadas Karole, Lasya Vyakaranam, Sangita Arun Mandlik, Hrushikesh Joshi, Shamim Alam, and Uday Chandrakant Patkar. Biometric technology in national security: Legal boundaries and ethical issues. *Biometric Technology Today*, 2024(9), 2024. Accessed: 2025-02-07.

[31] Agata Kruzikova, Lenka Knapova, David Smahel, Lenka Dedkova, and Vashek Matyas. Usable and secure? user perception of four authentication methods for mobile banking. *Computers & Security*, 115:102603, 2022.

[32] Ari Levinson. Obama portrait 2006, 2006. Wikimedia Commons, Licensed under CC BY-SA 3.0.

[33] Logto. What is time-based one-time password (totp)? In *Auth Wiki*, Apr 2025. Available: https://auth-wiki.logto.io/totp.

[34] Guangxin Lou and Hongzhen Shi. Face image recognition based on convolutional neural network. *China Communications*, 17(2):117–124, Feb 2020.

[35] Tamara Mohamed. Security of multifactor authentication model to improve authentication systems. 10 2019.

[36] Elisa M. Molanes-López and Emilio Letón. Inference of the youden index and associated threshold using empirical likelihood for quantiles. *Statistics in Medicine*, 30(19):2467 – 2480, 2011.

[37] Vincent Mühler. face-api.js: Tinyfacedetector module. `https://github.com/justadudewhohacks/face-api.js/tree/master/src/tinyFaceDetector`, 2019. Accessed: 2025-05-14.

[38] D. M'Raihi, S. Machani, M. Pei, and J. Rydell. Totp: Time-based one-time password algorithm. `https://datatracker.ietf.org/doc/html/rfc6238`, 2011. RFC 6238, Internet Engineering Task Force (IETF).

[39] Aleksandr Ometov, Sergey Bezzateev, Niko Mäkitalo, Sergey Andreev, Tommi Mikkonen, and Yevgeni Koucheryavy. Multi-factor authentication: A survey. *Cryptography*, 2(1), 2018.

[40] Sneha Padhiar. *A Comparative Study on Symmetric and Asymmetric Key Encryption Techniques*. 09 2021.

[41] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[42] Neil J. Perkins and Enrique F. Schisterman. The inconsistency of "optimal" cutpoints obtained using two criteria based on the receiver operating characteristic curve. *American Journal of Epidemiology*, 163(7):670 – 675, 2006.

[43] Midhuna Jyothi R and N. Jeyanthi. A review of modern authentication methods in digital systems. In *2023 Annual International Conference on Emerging Research Areas: International Conference on Intelligent Systems (AICERA/ICIS)*, pages 1–6, 2023.

[44] Simone Raponi and Roberto Di Pietro. A longitudinal study on web-sites password management (in)security: Evidence and remedies. *IEEE Access*, 8:52075–52090, 2020.

[45] N. K. Ratha, J. H. Connell, and R. M. Bolle. Enhancing security and privacy in biometrics-based authentication systems. *IBM Systems Journal*, 40(3):614–634, 2001.

[46] Christian Rathgeb, Andreas Uhl, and Peter Wild. *Potential Attacks*, pages 233–244. Springer New York, New York, NY, 2013.

[47] Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A Unified Embedding for Face Recognition and Clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823.

[48] Ş. İ. Serengil. deepface. GitHub repository, 2024. Available: `https://github.com/serengil/deepface`. Accessed: 2025-03-07.

[49] Ş. İ. Serengil and A. Özpinar. A benchmark of facial recognition pipelines and co-usability performances of modules. *Bilişim Teknolojileri Dergisi*, 17(2):95–104, April 2024.

[50] Şefik İlkin Serengil and Alper Ozpinar. Lightface: A hybrid deep face recognition framework. In *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)*, pages 1–5, 2020.

[51] Iwan Setyawan, Yohan Wibowo, Ivanna K. Timotius, and Andreas A. Febrianto. A human face detection system based on schneiderman-kanade method. In *2011 6th International Conference on Telecommunication Systems, Services, and Applications (TSSA)*, pages 1–4, 2011.

[52] A. Siddharth and D. Khankan. Secure password-less authentication: A multi-factor approach using facial recognition, fido2 standard, and dynamic otp. Master's thesis, Dept. of Computer Science and Engineering, Chalmers Univ. of Technology and Univ. of Gothenburg, Gothenburg, Sweden, 2025.

[53] Abhishek Kumar Singh, Sumit Kumar, Bhavesh Kumar, and Alok Singh Chauhan. Face recognition using machine learning. In *2023 International Conference on Sustainable Emerging Innovations in Engineering and Technology (ICSEIET)*, pages 361–364, 2023.

[54] Hassan Ugail. Chapter 6 - deep face recognition using full and partial face images. In E.R. Davies and Matthew A. Turk, editors, *Advanced Methods and Deep Learning in Computer Vision*, Computer Vision and Pattern Recognition, pages 221–241. Academic Press, 2022.

[55] Unknown photographer. Margaret thatcher (1987), 1987. Wikimedia Commons, Public Domain.

[56] David Valdez. George h. w. bush presidential portrait (cropped), 1989. Wikimedia Commons, Public Domain.

[57] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I, 2001.

[58] Zeljko Vujovic. Classification model evaluation metrics. *International Journal of Advanced Computer Science and Applications*, Volume 12:599–606, 07 2021.

[59] Nikola Vukobrat, Nemanja Maček, Saša Adamović, Muzafer Saračević, and Milan Gnjatović. Chapter 4 - implementation of two factor authentication using face and iris biometrics. In Bharat Bhushan, Sudhir Kumar Sharma, Muzafer Saračević, and Azedine Boulmakoul, editors, *Blockchain Technology Solutions*

for the Security of IoT-Based Healthcare Systems, Cognitive Data Science in Sustainable Computing, pages 77–96. Academic Press, 2023.

[60] Feng Wang, Jian Cheng, Weiyang Liu, and Haijun Liu. Additive margin softmax for face verification. *IEEE Signal Processing Letters*, 25(7):926–930, July 2018.

[61] Xuezhong Wang. Research on key technologies of face recognition data storage security. In *2023 Fourth International Conference on Frontiers of Computers and Communication Engineering (FCCE)*, pages 31–34, 2023.

[62] W. Wu, H. Peng, and S. Yu. YuNet: A Tiny Millisecond-level Face Detector. 20(5):656–665.

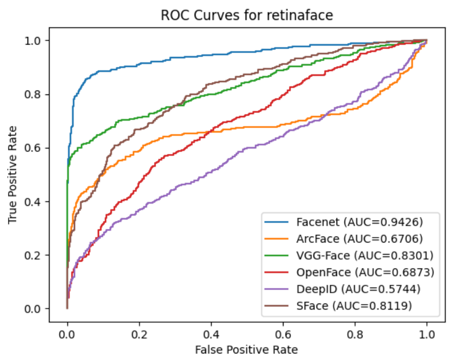
[63] Wencheng Yang, Song Wang, Di Wu, Taotao Cai, Yanming Zhu, Shicheng Wei, Yiying Zhang, Xu Yang, Zhaohui Tang, and Yan Li. Deep learning model inversion attacks and defenses: a comprehensive survey. *Artificial Intelligence Review*, 58(8):242, 2025.

[64] Hongxin Zhang and Liying Chi. End-to-end spatial transform face detection and recognition. *Virtual Reality & Intelligent Hardware*, 2(2):119–131, 2020. Special issue on Visual interaction and its application.

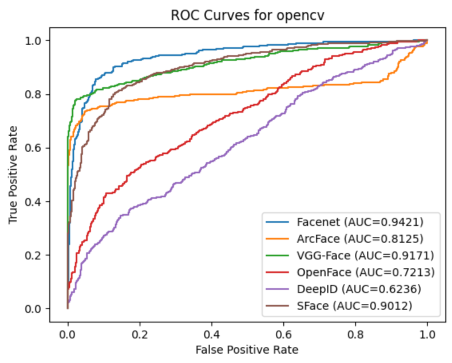
A

Additional ROC Curves

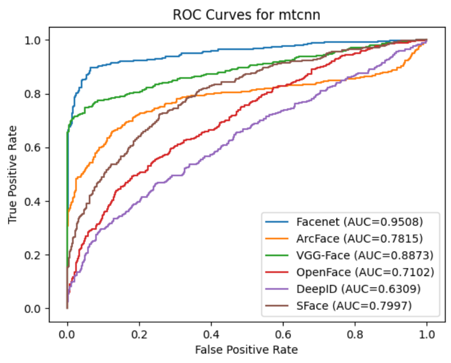
Each plot represents the ROC curves for one detection model in combination with all chosen recognition models.



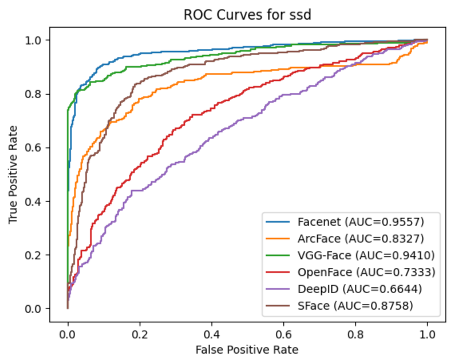
(a) RetinaFace



(b) OpenCV



(c) MTCNN



(d) SSD

Figure A.1: Additional ROC curves for detection and recognition model combinations evaluated on LFW.

B
Development Workflow

To ensure consistency and minimize environment-related issues, a development container was set up using Visual Studio Code's Dev Containers feature. This creates a standardized Docker-based environment with the same dependencies and Operating System (OS) across all contributors, avoiding common issues such as missing drivers, OS-specific Python packages, or version conflicts.
Project management was handled through GitHub Projects, using kanban boards to organize tasks into "To Do", "In Progress", "Review", and "Done". Each issue corresponded to a feature or fix and was addressed in a separate branch. Once completed, changes were submitted via pull requests. Before the changes could be merged, it had to pass linting and automated tests in a GitHub Actions workflow, see Figure B.1.

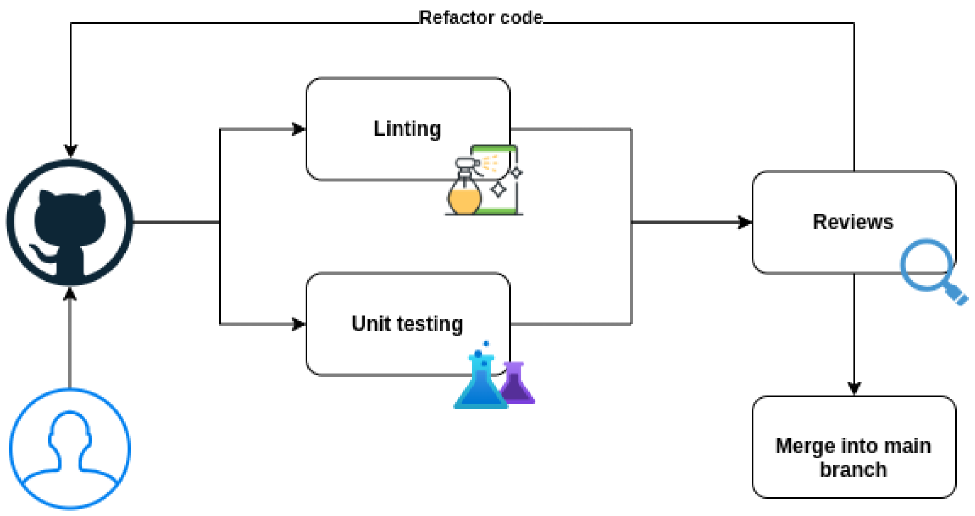


Figure B.1: The GitHub workflow for pull requests before code changes could be merged into the main branch

The GitHub Actions pipeline enforces code quality and correctness. Before a pull request can be merged, the code must pass the linting and unit testing steps. The linting ensures that the code complies with the Python Enhancement Proposal 8 (PEP8) coding standard. Any failing tests or linting errors will block the pull request from being merged until the issues are resolved.

C
Additional Pages of Application

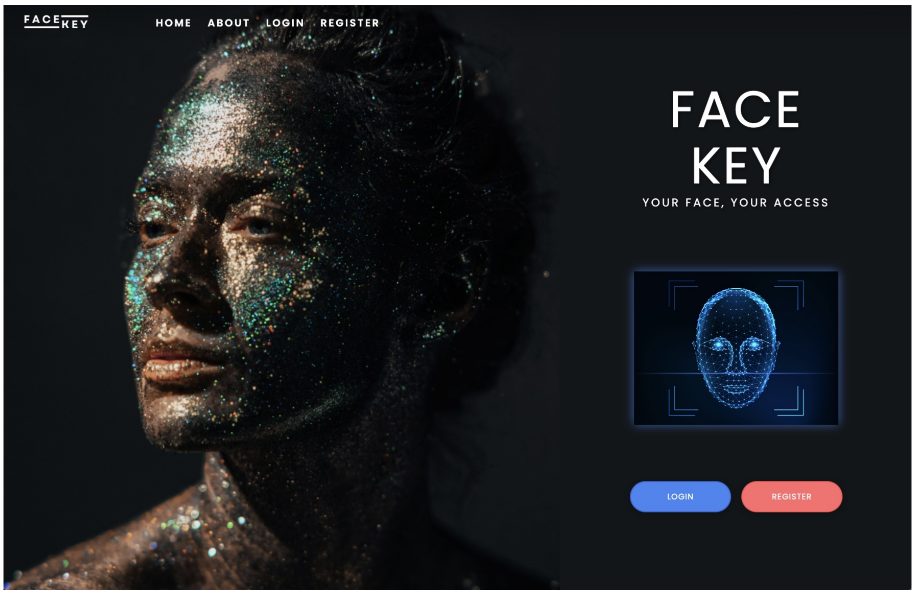
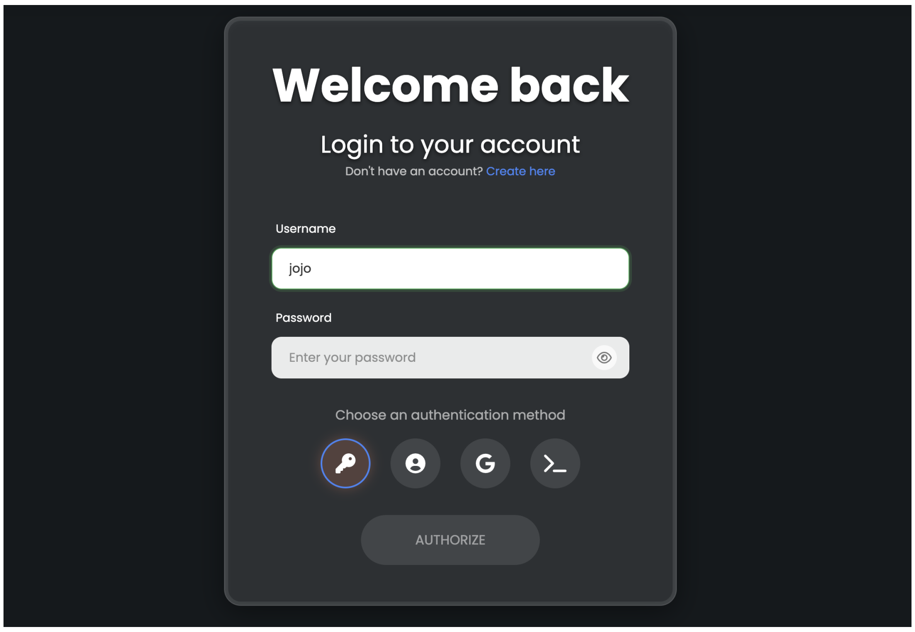


Figure C.1: The home page, main entry point to the application



Welcome back

Login to your account

Don't have an account? Create here

Username

jojo

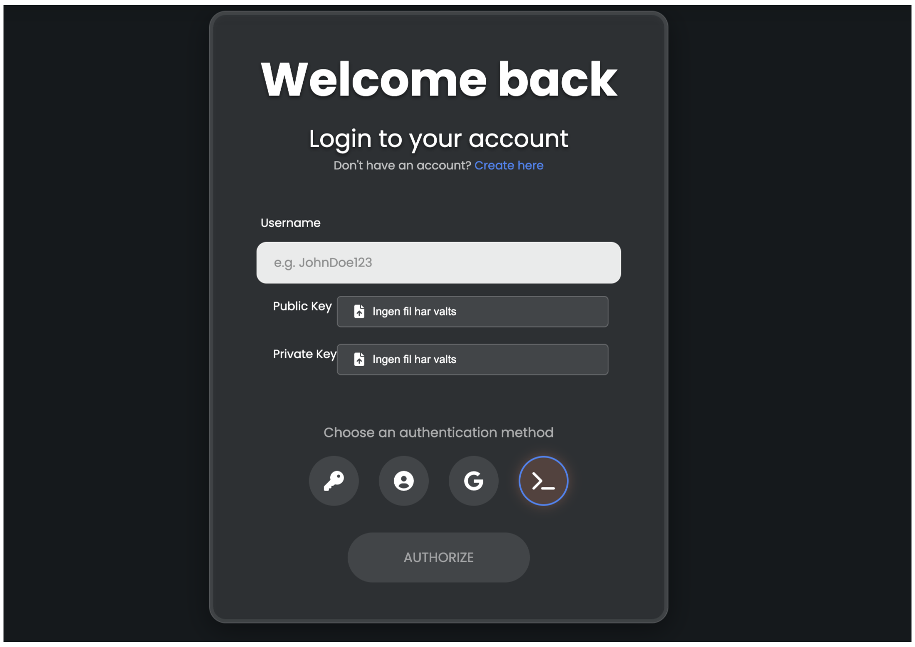
Password

Enter your password

Choose an authentication method

AUTHORIZE

Figure C.2: Password Login



Welcome back

Login to your account

Don't have an account? Create here

Username

e.g. JohnDoe123

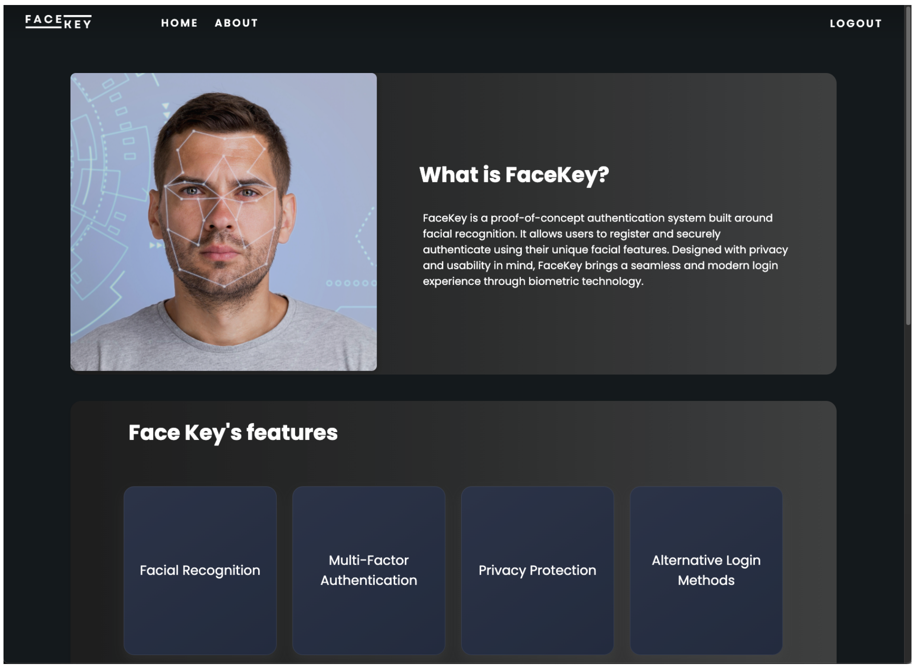
Public Key: Ingen fil har valts

Private Key: Ingen fil har valts

Choose an authentication method

AUTHORIZE

Figure C.3: RSA key singin



What is FaceKey?

FaceKey is a proof-of-concept authentication system built around facial recognition. It allows users to register and securely authenticate using their unique facial features. Designed with privacy and usability in mind, FaceKey brings a seamless and modern login experience through biometric technology.

Face Key's features

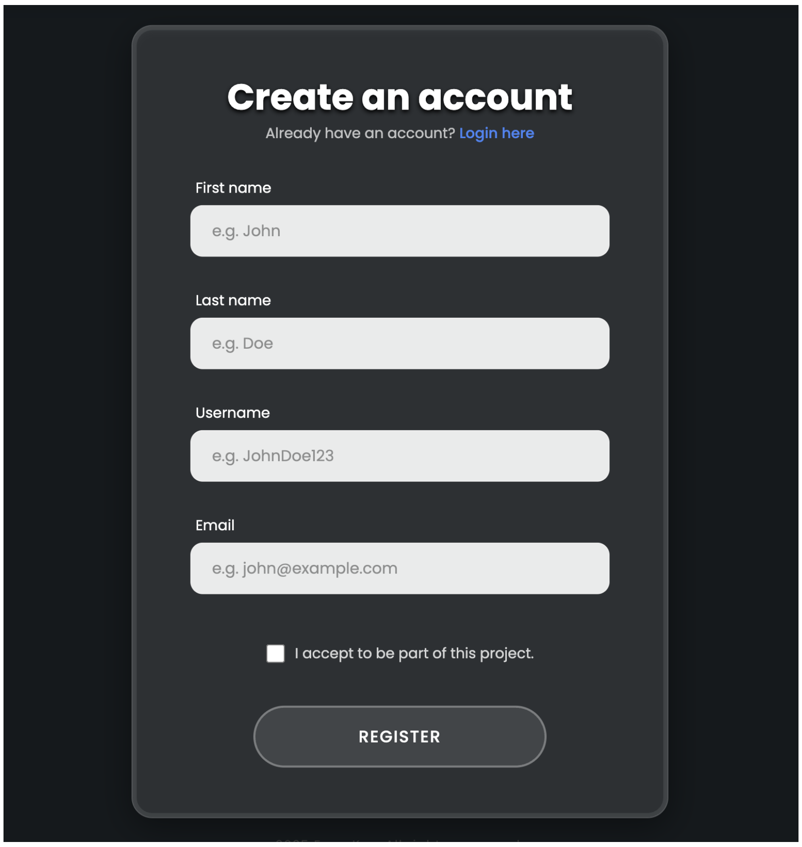
Facial Recognition

Multi-Factor Authentication

Privacy Protection

Alternative Login Methods

Figure C.4: The About page, detailed information about the application



Create an account

Already have an account? Login here

First name: e.g. John

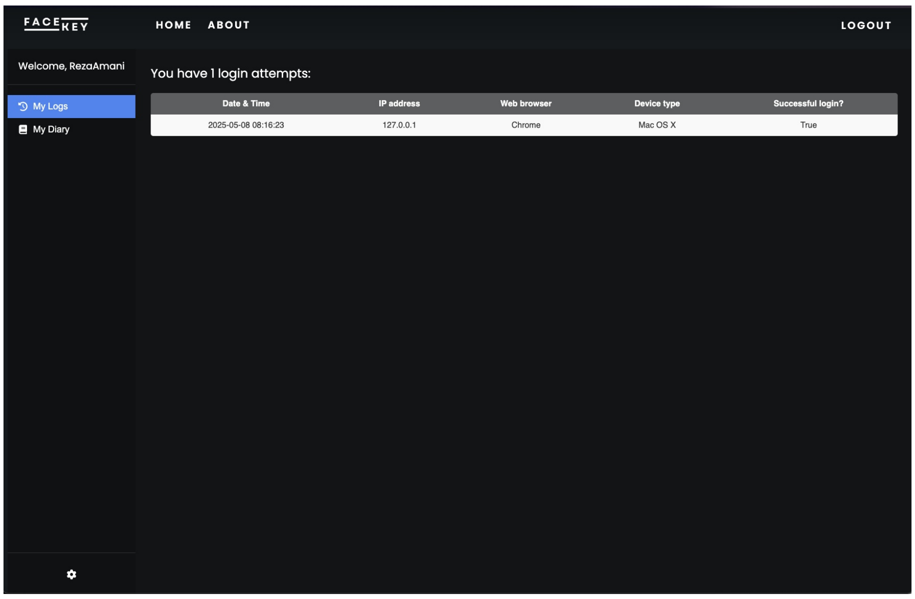
Last name: e.g. Doe

Username: e.g. JohnDoe123

Email: e.g. john@example.com

☐ I accept to be part of this project.

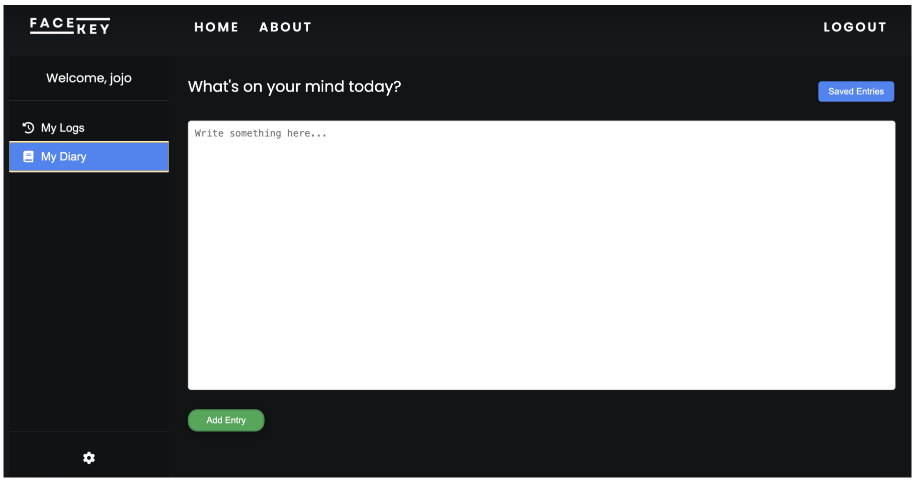
Figure C.5: The Registration page



You have 1 login attempts:

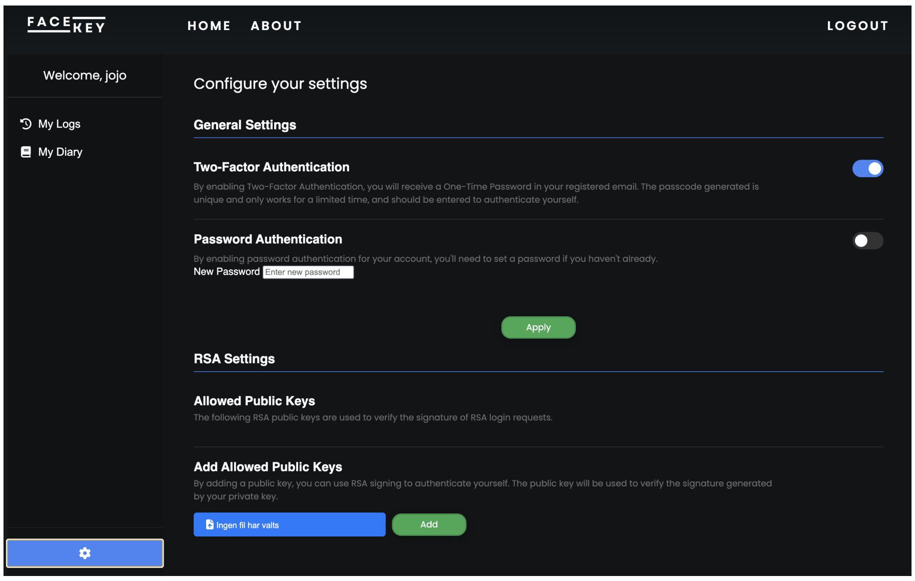
Date & Time	IP address	Web browser	Device type	Successful login?
2025-05-08 08:16:23	127.0.0.1	Chrome	Mac OS X	True

Figure C.6: The Dashboard



What's on your mind today?

Figure C.7: A diary feature for collecting notes



Configure your settings

General Settings

Two-Factor Authentication

By enabling Two-Factor Authentication, you will receive a One-Time Password in your registered email. The passcode generated is unique and only works for a limited time, and should be entered to authenticate yourself.

Password Authentication

By enabling password authentication for your account, you'll need to set a password if you haven't already.

New Password Enter new password

Apply

RSA Settings

Allowed Public Keys

The following RSA public keys are used to verify the signature of RSA login requests.

Add Allowed Public Keys

By adding a public key, you can use RSA signing to authenticate yourself. The public key will be used to verify the signature generated by your private key.

Ingen fil har valts

Add

Figure C.8: A Setting page where a user can enable MFA and activate other login methods