A

Project Report on

## DESIGNING SECURE AND EFFICIENT BIOMETRIC BASED ACCESS MECHANISM FOR CLOUD SERVICES

Submitted in partial fulfilment of the requirement for the award degree of

# Master of Science (Computer Science)

in

# Department of Computer Science

****

Submitted by

GALIBU AKHILA (24000I1023)

AISHWARYA (24000I1021) THUNGANI KESHAVULU (24000I1042)

**UNDER THE GUIDANCE OF SALONI FATHIMA**

**DEPARTMENT OF COMPUTER SCIENCE, KAKATIYA UNIVERSITY VIDYARANYAPURI, HANAMKONDA, WARANGAL-506009**

**2023-2025**

**DEPARTMENT OF COMPUTER SCIENCE KAKATIYA UNIVERSITY**

### VIDYARANYAPURI, HANAMKONDA, WARANGAL-506009

****

CERTIFICATE

This is to certify that GALIBU AKHILA (24000I1023) , AISHWARYA (24000I1021),

THUNGANI KESHAVULU (24000I1042) students of M.Sc(computer science) has satisfactorily completed the dissertation work entitled **“DESIGNING SECURE AND EFFICIENT BIOMETRIC-BASED ACCESS MECHANISM FOR CLOUD**

**SERVICES”** in the partial fulfilment of the requirements of the MSC during this academic year 2023-20245 under the guidance of **SALONI FATHIMA**, Assistant professor, Department of Computer Science, K.U.

**SUPERVISOR EXTERNAL HEAD**

**DECLARATION BY THE CANDIDATES**

We GALIBU AKHILA (24000I1023), AISHWARYA (24000I1021), THUNGANI

KESHAVULU (24000I1042) to hear by certify that the project report entitled “**DESIGNING SECURE AND EFFICIENT BIOMETRIC-BASED ACCESS MECHANISM FOR**

**CLOUD SERVICES** ”, under the guidance **SALONI FATHIMA ,** Department Of Computer Science, Kakatiya University, Warangal, submitted in partial fulfilment of the requirements for the Award of the Degree of MSc. Computer Science.

This is a record of Bonafide work carried out by us and the results embodied in this Project Report has not been produced / copied from any source. The results embodied in this Project Report have not been submitted to any other University or Institute for the Award of any other Degree or Diploma.

##### GALIBU AKHILA (24000I1023) AISHWARYA (24000I1021)

**THUNGANI KESHAVULU (24000I1042)**

**ACKNOWLEDGEMENT**

We thank the almighty for giving us the courage and perseverance in completing the project. This project itself is an acknowledgement for all those people who have given us their heartfelt co-operation in making this project a success.

We take this opportunity to express our deep and sincere gratitude to the project guide SALONI FATHIMA, Assistant Professor, for her/his valuable advices at every stage of this work. We feel that without his/her supervision and many hours of devoted guidance, stimulating and constructive criticism, and this would never have Crone out in this form.

We greatly indebted to Dr. B. RAMA, HEAD, Department of Computer Science, Kakatiya University.

We are thankful to Dr. B. RAMA, Dr. B. Manjula, Dr. D. Ramesh, Ch.Neelima,Saloni Fathima and Dr. T Rajini Devi, for their valuable support and providing the excellent facilities, motivation and valuable guidance throughout the project work. With their co- operation and encouragement, we have completed the project work in time.

Last but not least we would like to express our deep sense of gratitude and earnest thanks giving to our dear parents for their moral support and heartfelt co-operation in doing the project. We would also like to thank all the non-teaching staff and our friends, whose direct or indirect help enabled us to complete this work successfully.

**GALIBU AKHILA (24000I1023)**

**AISHWARYA (24000I1021) THUNGANI KESHAVULU (24000I1042)**

## ABSTRACT

The demand for remote data storage and computation services is increasing exponentially in our data-driven society; thus, the need for secure access to such data and services. In this paper, we design a new biometric-based authentication protocol to provide secure access to a remote (cloud) server. In the proposed approach, we consider biometric data of a user as a secret credential. We then derive a unique identity from the user’s biometric data, which is further used to generate the user’s private key. In addition, we propose an efficient approach to generate a session key between two communicating parties using two biometric templates for a secure message transmission. In other words, there is no need to store the user’s private key anywhere and the session key is generated without sharing any prior information. A detailed Real-Or- Random (ROR) model based formal security analysis, informal (non- mathematical) security analysis and also formal security verification using the broadly- accepted Automated Validation of Internet Security Protocols and Applications (AVISPA) tool reveal that the proposed approach can resist several known attacks against (passive/active) adversary. Finally, extensive experiments and a comparative study demonstrate the efficiency and utility of the proposed approach.

Index Terms Authentication, biometric-based security, cloud service access, session

key.

**Algorithms used:** Password Hashing, Biometric Matching(Image Hashing),Account lockout mechanism, Key Generation & Encryption(AES,RSA,HMAC),Session Handling

#### CONTENTS Page No

1. **INTRODUCTION** **01-05**
   1. Existing System 03
   2. Proposed System 04
   3. System Specification 04
      1. Hardware Requirements 05
      2. Software Requirements 05

##### LITERATURE SURVEY 06-09

##### SYSTEM ANALYSIS 10-20

* 1. Modules
  2. Preliminary Investigation
     1. Request classification
     2. Feasibility Analysis 3.2.3Request Approval

3.3 Feasibility Study

##### SYSTEM DESIGN & DEVELOPMENT 21-30

* 1. Input design
  2. Output design
  3. Use Case diagram
  4. Sequence diagram
  5. System Architecture

##### METHODOLOGY 31-33

1. **SYSTEM IMPLEMENTATION** **34-45**
   1. HTML
   2. My SQL
   3. Java Script
   4. Python

##### TESTING 46-48

* 1. Type of Testing

1. **PROGRAMMING CODE** **49-82**
   1. Output Screenshots
2. **CONCLUSION** **83-84**
3. **REFERENCE** **85**

## CHAPTER 1 INTRODUCTION

### INTRODUCTION

Cloud services store and process vast amounts of sensitive data, requiring strong authentication. Traditional methods (passwords, PINs) are vulnerable to hacking, phishing, and brute-force attacks. In this project we designing a fingerprint image of user as a Secret Credential. we capture a biometric fingerprint image of the user, and subsequently generate the private key and encrypt the biometric data as a query. This queried biometric data is then transmitted to the authentication server for matching with the stored data.Once the user is authenticated successfully. he/she ready to access his/her Services from the desired server.

To obtain secure access to the service server, mutual authentication between the user and authentication server, and also between the user and service server have been proposed using a short-term session key. Using two fingerprint data, we present a fast and robust approach to generate the session key. In addition, a biometric based message authenticator is also generated for message authenticity purpose.

We summarize the key contributions/benefits related to the proposed approach as below.

1. An effective way to transmit the user’s biometric data through the unsecured network channels to an authentication server is presented.
2. We propose an approach to generate a revocable private key directly from an irrevocable fingerprint image. There is no need to store the private key or a direct form of the user’s biometric data anywhere.
3. We mitigate the limitation in traditional mechanisms that require the user’s credentials to be stored in the authentication server.
4. We introduce a novel way to generate session keys.
5. In traditional authentication protocol, each entity requires some preloaded information; thus, incurring some overhead. We introduce a new mechanism to avoid the need for secret pre-loaded information.
6. A message authentication mechanism, as an alternative to the existing message authentication protocols (i.e., Message Authentication Code (MAC)), is introduced.

### EXISTING SYSTEM

A number of authentication mechanisms have been proposed in the literature, such as those based on Kerberos, OAuth and OpenID. Generally, these protocols seek to establish a secure delegated access mechanism among two communicating entities connected in a distributed system. These protocols are based on the underlying assumption that the remote server responsible for authentication is a trusted entity in the network. Specifically, a user first registers with a remote server. This is needed to ensure the authorization of the owner. When a user wishes to access a server, the remote server authenticates the user and the user also authenticates the server. Once both verification are successfully carried out, the user obtains access to the services from some remote server.

One key limitation in existing authentication mechanisms is that the user’s credentials are stored in the authentication server, which can be stolen and (mis)used to gain unauthorized access to various services. Also, to ensure secure and fast communication, existing mechanisms generally use symmetric key cryptography, which requires a number of cryptographic keys to be shared during the authentication process. This strategy results in an overhead to the authentication protocols. Designing secure and efficient authentication protocols is challenging, as evidenced by the weaknesses revealed in the published protocols of Jiang et al. , Althobaiti et al. , Xue et al. , Turkanovic et al. , Park et al. , Dhillon and Kalra , Kaul and Awasthi and Kang et al. – see also Section II. Therefore, in this paper we seek to design a secure and efficient authentication protocol. Specifically, we will first provide an alternative to conventional password-based authentication mechanism. Then, we demonstrate how one can build a secure communication between communicating parties involved in the authentication protocol, without having any secret pre-loaded (i.e., shared) information.

#### Drawbacks of Existing System

* The conventional method of calling the names of students manually is a time consuming event and takes up time of the period.
* Biometric attendance using fingerprint and iris may pose a security threat as the information of biometric can be hacked and leaked. Also, this system slow and expensive.
* RFID based attendance has its own problems. There is chance of losing the card or unauthorized person may misuse the card for proxy attendance.

### PROPOSED SYSTEM

In the proposed approach, we consider a fingerprint image of a user as a secret credential. From the fingerprint image, we generate a private key that is used to enroll the user’s credential secretly in the database of an authentication server. In the authentication phase, we capture a new biometric fingerprint image of the user, and subsequently generate the private key and encrypt the biometric data as a query. This queried biometric data is then transmitted to the authentication server for matching with the stored data. Once the user is authenticated successfully, he/she is ready to access his/her service from the desired server. To obtain secure access to the service server, mutual authentication between the user and authentication server, and also between the user and service server have been proposed using a short-term session key. Using two fingerprint data, we present a fast and robust approach to generate the session key. In addition, a biometric based message authenticator is also generated for message authenticity purpose.

We summarize the key contributions/benefits related to the proposed approach as below.

1. An effective way to transmit the user’s biometric data through the unsecured network channels to an authentication server is presented.
2. We propose an approach to generate a revocable private key directly from an irrevocable fingerprint image. There is no need to store the private key or a direct form of the user’s biometric data anywhere.
3. We mitigate the limitation in traditional mechanisms that require the user’s credentials to be stored in the authentication server.
4. We introduce a novel way to generate session keys.
5. In traditional authentication protocol, each entity requires some preloaded information; thus, incurring some overhead. We introduce a new mechanism to avoid the need for secret pre-loaded information.
6. A message authentication mechanism, as an alternative to the existing message authentication protocols (i.e., Message Authentication Code (MAC)), is introduced.

### ADVANTAGES

1. An efficient approach to generate a session key between two communicating parties using two biometric templates for a secure message transmission.
2. There is no need to store the user’s private key anywhere and the session key is generated without sharing any prior information**.**

### SYSTEM REQUIREMENTS

This section elaborates on the functional requirements of the application. The SRS itself can be divided into module, each module having specifications. In order to carry out the project, the following hardware and software is required.

### HARDWARE REQUIREMENTS

The following are the hardware requirements which we have used in our project.

* + - * Processor Needed : i3 or above.
      * RAM : 1 GB or more.
      * Hard disk : 256 GB or more.
      * Monitor : Any Monitor.
      * Keyboard : Standard Keyboard.
      * Mouse : Two or Three Button Mouse.

### SOFTWARE REQUIREMENTS

The following are software requirements:

* + - * Operating System : Microsoft Windows 7 or Higher windows OS, Linux or Mac any version.
      * Scripting Language : Python.
      * Front– End : HTML, CSS, and JavaScript.
      * Back– End : MYSQL, Django.

## CHAPTER 2 LITERATURE SURVEY

* 1. **LITERATURE SURVVEY**

#### The kerberos network authentication service (v5)

Since its initial development Kerberos has evolved to become the widely accepted system for implementing centralized authentication services. This paper discusses a strategy for using the symmetric key management facilities of Kerberos to implement directory based authorization.

The Kerberos authentication service, a part of MIT's Project Athena, is based on the Needham and Schroeder protocol. Timestamps depending on reliable synchronized clocks are used to guarantee the freshness of messages. As an improvement, we present a nonce-based protocol offering the same features as Kerberos. We generate a ticket in an initial message exchange which includes a generalized timestamp. Checking this generalized timestamp is left to the principal who created it. Consequently we do not need synchronized clocks. Our protocol has the property of using a minimal number of messages to establish an authenticated session key.

#### Biometric-Based Privacy-Preserving User Authentication Scheme for Cloud-Based Industrial Internet of Things Deployment

###### AUTHORS: A. K. Das, M. Wazid, N. Kumar, A. V. Vasilakos, and J. J. P. C. Rodrigues, ABSTRACT:

Due to the widespread popularity of Internet-enabled devices, Industrial Internet of Things (IIoT) becomes popular in recent years. However, as the smart devices share the information with each other using an open channel, i.e., Internet, so security and privacy of the shared information remains a paramount concern. There exist some solutions in the literature for preserving security and privacy in IIoT environment. However, due to their heavy computation and communication overheads, these solutions may not be applicable to wide category of applications in IIoT environment. Hence, in this paper, we propose a new biometric-based privacy preserving user authentication (BP2UA) scheme for cloud-based IIoT deployment. BP2UA consists of strong authentication between users and smart devices using pre established key agreement between smart devices and the gateway node.

The formal security analysis of BP2UA using the well-known real-or-random model is provided to prove its session key security. Moreover, an informal security analysis of BP2UA is also given to show its robustness against various types of known attacks. The computation and communication costs of BP2UA in comparison to the other existing schemes of its category demonstrate its effectiveness in the IIoT environment. Finally, the practical demonstration of BP2UA is also done using the NS2 simulation.

#### Security and Accuracy of Fingerprint-Based Biometrics: A Review

###### AUTHORS: W. Yang, S. Wang, J. Hu, G. Zheng, and C. Valli ABSTRACT:

Biometric systems are increasingly replacing traditional password- and token-based authentication systems. Security and recognition accuracy are the two most important aspects to consider in designing a biometric system. In this paper, a comprehensive review is presented to shed light on the latest developments in the study of fingerprint-based biometrics covering these two aspects with a view to improving system security and recognition accuracy. Based on a thorough analysis and discussion, limitations of existing research work are outlined and suggestions for future work are provided.

It is shown in the paper that researchers continue to face challenges in tackling the two most critical attacks to biometric systems, namely, attacks to the user interface and template databases. How to design proper countermeasures to thwart these attacks, thereby providing strong security and yet at the same time maintaining high recognition accuracy, is a hot research topic currently, as well as in the foreseeable future. Moreover, recognition accuracy under non-ideal conditions is more likely to be unsatisfactory and thus needs particular attention in biometric system design. Related challenges and current research trends are also outlined in this paper.

#### Difference co-occurrence matrix using BP neural network for fingerprint liveness detection

###### AUTHORS: C. Yuan, X. Sun, and Q. M. J. Wu ABSTRACT:

With the growing use of fingerprint identification systems in recent years, preventing fingerprint identification systems from being spoofed by artificial fake fingerprints has become a critical problem. In this paper, we put forward a novel method to detect fingerprint liveness based on BP neural network, which is used for the first time in the fingerprint liveness detection. Moreover, different from traditional detection methods, we propose a scheme to construct the input data and corresponding category labels. More effective and efficient texture features of fingerprints, which are used as the input data of the BP neural network, are computed to improve classification performance and obtain a better pre-trained network model. After a variety of preprocessing operations and image compression operations, gradient values in the horizontal and vertical directions are computed by using Laplacian operator, and difference co-occurrence matrices are constructed from the obtained gradient values. Then, the input data of neural network model are built based on two DCMs.

The pre-trained neural network models with diverse neuron nodes are learnt. Different experiments based on different parameters for the BP neural network have been conducted. Finally, classification accuracy of testing fingerprints is predicted based on the pre-trained networks. Experimental results on the LivDet 2013 show that the classification performance of our proposed method is effective and meanwhile provides a better detection accuracy compared with the majority of previously published results.

## CHAPTER 3 SYSTEM ANALYSIS

### MODULES

1. CLIENT
2. AUTHENTICATION SERVER
3. ADMIN
4. RESOURCE SERVER

##### CLIENT

Client has to register into application with basic details and he can able to login with username ,password and with fingerprint. Client can able sent request to the resource server. After sending the request he can get the response from the resource server. After getting the response from the server he can able view the file in the cloud. He can able to see all permission of files.

##### AUTHENTICATION SERVER.

Authentication Server need to login with username and password. After login he can able to view client details and authorize . Authentication server can able to view synthetic finger print images. Server can able to user client images.

##### ADMIN

Admin need to login with basic username and password. After login he can able to upload files those are useful to the user. He can able to view all uploaded files. Admin can able to add synthetic fingerprint images. Admin can able to view the data in the repository.

##### RESOURCE SERVER

Resource server need to login into the application using username and password. After login resource server he can able to view all client requests as well as he can able view all users access rights of files.

### MODULE DESIGN

The system is divided into several interconnected modules, each responsible for a specific function. Together, they provide a secure, efficient, and user-friendly biometric authentication system for cloud service access.

#### Fingerprint Acquisition Module

###### Purpose:

To capture high-quality fingerprint images from users.

###### Functionality:

Interface with a fingerprint scanner or mobile device sensor. Capture fingerprint images in real-time.

Perform initial quality checks (e.g., check clarity, completeness).

###### Tools/Technologies:

Hardware: Fingerprint sensor (e.g., optical, capacitive scanners)

Software: Device SDK, OpenCV for image capture and preliminary processing

#### Preprocessing Module

###### Purpose:

To enhance the fingerprint image and prepare it for feature extraction.

###### Functionality:

Noise reduction and smoothing. Contrast enhancement.

Image normalization (standardizing image size and orientation). Fingerprint segmentation (separating fingerprint region from background). **Techniques Used:**

Histogram Equalization Gaussian Filter Adaptive Thresholding

#### Feature Extraction Module

###### Purpose:

To extract unique features (fingerprint minutiae) from the processed fingerprint image.

###### Functionality:

Detect minutiae points (ridge endings, bifurcations). Extract orientation and frequency features.

Generate a feature vector or fingerprint template.

###### Tools:

NBIS (NIST Biometric Image Software)

OpenCV with custom fingerprint feature extractors

#### Template Protection and Storage Module

###### Purpose:

To securely store extracted fingerprint features.

###### Functionality:

Encrypt the fingerprint template using cryptographic algorithms (e.g., AES-256). Save the encrypted templates into a secure database.

Hash templates if necessary to prevent direct retrieval.

###### Technologies:

PyCryptodome (Python Encryption)

Database: MySQL with encryption enabled or MongoDB with encrypted fields

#### Authentication and Matching Module

###### Purpose:

To authenticate users based on their fingerprint scan.

###### Functionality:

Capture new fingerprint input. Preprocess and extract features.

Compare extracted features with stored templates using a matching algorithm. Calculate similarity score; approve or deny access based on a threshold.

###### Matching Methods:

Minutiae-based matching Ridge pattern matching

#### Cloud Access Management Module

###### Purpose:

To manage authenticated users' access to cloud services.

###### Functionality:

Grant or deny access based on authentication result.

Assign roles and permissions (Role-Based Access Control - RBAC). Log user activities for auditing.

###### Technologies:

AWS IAM / Azure Active Directory REST API for access control

## Preliminary investigation

#### Introduction

The rapid expansion of cloud computing has revolutionized data storage, access, and management. However, it has also introduced significant security challenges. Traditional password-based authentication systems are increasingly vulnerable to cyber threats such as phishing, credential theft, and brute-force attacks. To strengthen cloud security, this project investigates the design of a secure, biometric-based access mechanism, utilizing fingerprint recognition as a means of user authentication.

#### Problem Statement

Current authentication methods for cloud services are inadequate against evolving security threats. Passwords and token-based systems can be forgotten, stolen, or hacked. There is a growing need for a more secure, reliable, and user-friendly access mechanism. Biometric authentication, particularly fingerprint-based systems, offers a promising solution due to the uniqueness and non-replicability of biometric traits.

#### Objectives

* To analyse existing security models for cloud service access.
* To design a secure access mechanism incorporating fingerprint biometrics.
* To ensure the confidentiality, integrity, and availability of biometric data during storage and transmission.
* To evaluate the system’s resistance to common cyber-attacks and biometric spoofing.

#### Methodology

* Literature Survey: Reviewed research papers, security reports, and case studies related to cloud security and biometric authentication.
* Technology Assessment: Explored various fingerprint acquisition technologies and biometric cryptosystems.
* System Requirements Gathering: Identified functional and security requirements for integrating biometric authentication into cloud platforms.
* Feasibility Study: Assessed technical, operational, and legal feasibility, including compliance with data protection regulations like GDPR.

#### Key Findings

* Fingerprint authentication enhances access security by binding user identity to physiological characteristics.
* Cloud-based biometric systems face challenges such as biometric template protection, latency, and data privacy concerns.
* Techniques like biometric encryption, secure multiparty computation, and homomorphic encryption can mitigate some security risks.
* User acceptance of biometric authentication depends on perceived security, ease of use, and trust in the system.

#### Conclusion

The preliminary investigation confirms that designing a secure, fingerprint-based access mechanism for cloud services is both relevant and feasible. The solution must prioritize secure biometric template storage, encrypted data transmission, and strong user authentication protocols to ensure widespread adoption and trust.

#### Request Classification

* + - 1. **Introduction**

In a secure, biometric-based cloud access system, classifying user requests accurately is critical to maintaining security, ensuring smooth access, and detecting malicious activities. Request classification helps in deciding the appropriate authentication, verification, or denial responses based on the nature of the request.

#### Types of Requests

User interactions with the cloud service can be broadly classified into the following categories:

###### Authentication Requests:

Requests initiated when a user attempts to log in or access cloud resources using biometric credentials (fingerprint image).

###### Registration Requests:

Requests during user enrollment, where a new fingerprint is captured, processed, and securely stored in the system database.

###### Update Requests:

Requests for updating existing biometric data (e.g., re-enrollment if the fingerprint image quality is low or compromised).

###### Access Control Requests:

Requests involving permission checks for accessing specific cloud services, after successful biometric authentication.

###### Anomaly Detection Requests:

Requests that trigger alerts or additional verification steps, such as multiple failed authentication attempts or suspicious behaviour.

#### Classification Workflow

**Step 1:** Capture the incoming request metadata (e.g., IP address, device information, request type).

**Step 2:** Identify request category based on the operation (Authentication, Registration, Update, Access Control, or Anomaly).

**Step 3:** Route the request to the appropriate processing module.

**Step 4:** Apply security checks such as biometric matching, liveness detection, and device verification.

**Step 5:** Generate a system response (grant access, deny access, request additional verification, or alert admin).

#### Security Considerations

Biometric data must never be transmitted or stored in raw form; it should always be encrypted.

Frequent failed authentication attempts should be flagged and classified as potential security threats.

Request classification must ensure minimal latency to preserve user experience while maintaining high security standards.

## Feasibility Analysis

###### Introduction

Feasibility analysis evaluates whether the proposed biometric-based access mechanism for cloud services is practical, efficient, and sustainable. It examines various aspects, including technical, operational, economic, legal, and schedule feasibility, to ensure successful implementation.

###### Technical Feasibility

**Biometric Technology Availability:**

Fingerprint scanners are widely available, reliable, and have mature technology for accurate biometric capture and matching.

###### Cloud Integration:

Cloud platforms support integration with external authentication services through APIs and protocols like OAuth 2.0, SAML, and custom authentication layers.

###### Security Measures:

Advanced encryption techniques (such as AES, RSA) and secure biometric template storage (e.g., cancellable biometrics, biometric cryptosystems) can be employed to protect data during transmission and storage.

###### Scalability:

The system can be scaled to support a large number of users by utilizing cloud-based microservices and load balancing.

###### Conclusion:

Technically feasible with the proper choice of biometric devices, secure algorithms, and scalable cloud architecture.

###### Operational Feasibility User Acceptance:

Users are increasingly familiar with biometric authentication (seen in smartphones, banking apps), which improves acceptance rates.

###### Ease of Use:

Fingerprint scanning provides a fast and user-friendly authentication experience without the need for remembering complex passwords.

###### Training and Support:

Minimal training is required for end-users; administrators might require brief training on managing biometric data securely.

###### Conclusion:

Operationally feasible as it enhances both user experience and security.

###### Economic Feasibility

**Cost of Biometric Devices:**

Fingerprint scanners are relatively inexpensive and already embedded in many smartphones and laptops.

###### Development and Integration Costs:

Moderate investment needed for software development, system integration, and cloud service subscriptions.

###### Maintenance Costs:

Costs are manageable and largely limited to occasional updates, security audits, and device replacements.

###### Conclusion:

Economically feasible with a reasonable return on investment, especially when considering the improved security.

###### Legal and Ethical Feasibility Data Privacy Regulations:

The system must comply with regulations like GDPR, HIPAA, or other regional privacy laws ensuring biometric data protection.

###### Consent and Transparency:

Users must be informed about how their biometric data will be used, stored, and protected.

###### Ethical Concerns:

Adequate measures must be taken to prevent misuse or unauthorized access to biometric information.

###### Conclusion:

Legally and ethically feasible if designed with strict adherence to data protection laws and best practices.

###### Schedule Feasibility Development Timeline:

A typical project timeline (design, development, testing, and deployment) would fit within 6– 9 months, depending on complexity.

###### Availability of Resources:

Skilled developers, cloud specialists, and biometric system experts are readily available.

###### Conclusion:

The project is feasible within a reasonable timeframe with proper planning and resource allocation.

###### Overall Conclusion

The feasibility analysis shows that designing a secure, fingerprint-based access mechanism for cloud services is technically, operationally, economically, legally, and ethically viable. With careful attention to security, privacy, and user experience, the project promises a robust solution to enhance cloud service authentication.

#### Request Approval

###### Introduction

The request for the development and implementation of a secure, biometric-based access mechanism for cloud services is submitted for review and approval. This section outlines the rationale behind the project, the benefits it offers, and seeks formal authorization to proceed with full-scale development.

###### Purpose of the Request

The primary purpose of this request is to enhance the security of cloud service access by replacing or augmenting traditional authentication methods with fingerprint biometric authentication. This approach aims to reduce risks associated with password theft, unauthorized access, and data breaches.

###### Justification

**Security Enhancement**: Biometric authentication provides a more secure and reliable way to verify user identity.

**User Convenience**: Fingerprint authentication is fast, intuitive, and requires no memorization of passwords.

**Compliance**: The proposed system can help meet compliance requirements for secure access and data protection.

**Scalability**: The system can be scaled to support enterprise-level cloud services with large user bases.

###### Project Overview

**Title:** Designing Secure and Biometric-based Access Mechanism for Cloud Services

**Technology Used**: Biometric fingerprint recognition, cloud computing, encryption algorithms

**Expected Outcomes**: Enhanced access control, improved security posture, and a seamless user experience.

## FEASIBILITY STUDY

The feasibility of the project is analysed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

* **ECONOMICAL FEASIBILITY**
* **TECHNICAL FEASIBILITY**
* **SOCIAL FEASIBILITY**

**ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

##### TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

##### SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

## CHAPTER 4

**SYSTEM DESIGN & DEVELOPMENT**

### INPUT DESIGN

considering the requirements, procedures to collect the necessary input data in most efficiently designed. The input design has been done keeping in view that, the interaction of the user with the system being the most effective and simplified way.

#### Types of Inputs

The system will receive several types of input data during its operation, which are crucial for the authentication process

#### Biometric Data (Fingerprint)

**Input Type:** Digital fingerprint image.

**Format:** High-resolution grayscale or RGB fingerprint image (usually in .bmp or .png format) captured by a biometric sensor.

**Resolution:** Minimum of 500 DPI (dots per inch) to ensure high-quality fingerprint data.

**Input Method:** Fingerprint scanner integrated with a device (e.g., mobile phone, fingerprint reader).

**Data Processing:** The fingerprint image will be processed to extract key features (e.g., minutiae points) and encrypted into a template for comparison and storage.

## User Credentials

**Input Type:** Username and password (if required as a secondary authentication factor).

**Format:** Alphanumeric text.

**Input Method:** User inputs through a secure form (web interface, app interface, or authentication device).

**Validation:** Encrypted storage of passwords using hashing algorithms (e.g., bcrypt, Argon2).

###### Also the measures are taken for the following :

* + Controlling the amount of input
  + Avoid unauthorized access to the classroom.
  + Eliminating extra steps
  + Keeping the process simple
  + At this stage the input forms and screens are designed.

### OUTPUT DESIGN

All the screens of the system are designed with a view to provide the user with easy operations in simpler and efficient way, minimum key strokes possible. Instructions and important information is emphasized on the screen. Almost every screen is provided with no error and important messages and option selection facilitates. Emphasis is given for speedy processing and speedy transaction between the screens. Each screen assigned to make it as much user friendly as possible by using interactive procedures. So to say user can operate the system without much help from the operating manual.

#### Types of Outputs

The system will produce several types of outputs based on the authentication process, including feedback for users, system alerts, logs, and reports.

###### User Authentication Feedback

**Output Type:** Access granted or denied message. **Format:** Text-based message or graphical notification. **Content:**

**Access Granted:** "Authentication Successful. Welcome, [User Name]."

**Access Denied:** "Authentication Failed. Please try again or contact support."

**Secondary Authentication Required:** "Please enter your password for verification."

Purpose: To inform the user whether they were successfully authenticated and granted access to the cloud service.

###### Security Alerts

**Output Type:** Security alert message. **Format:** Text-based alert or notification. **Content:**

**Multiple Failed Attempts:** "Warning: Multiple failed authentication attempts detected for [User Name]."

**Suspicious Activity**: "Suspicious login attempt detected from [Location/Device]. Please verify your identity.

**Purpose:** To inform system administrators or users of potential security threats, such as brute-force attempts or login from unrecognized devices.

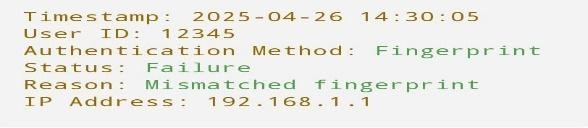
###### Authentication Logs

**Output Type:** Authentication logs for auditing. **Format:** Tabular or log file format (e.g., CSV, JSON). **Content:**

###### Log Entry Example:



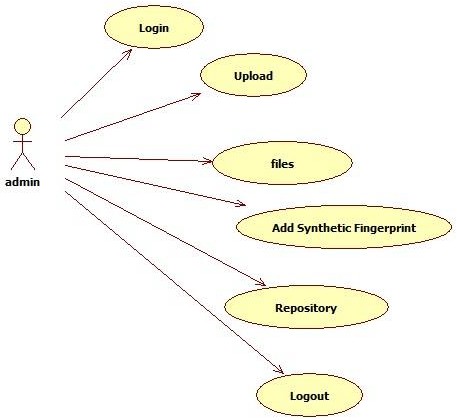
**Failure Log Example:**

****

**Purpose:** To store detailed records of every authentication attempt, whether successful or unsuccessful, for security analysis, auditing, and troubleshooting purposes.

### USE CASE DIAGRAM:

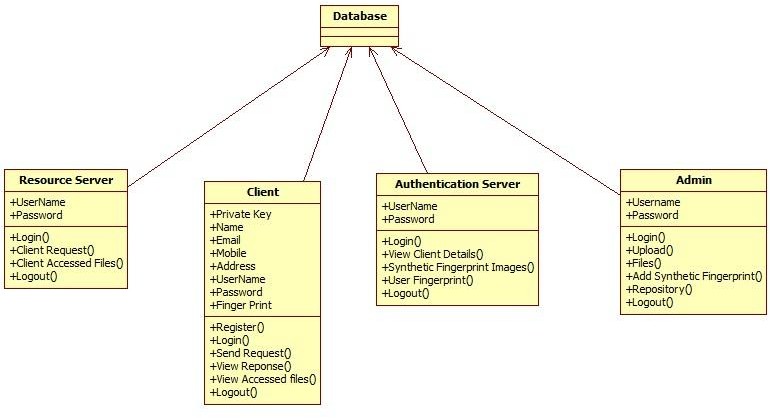
A use case diagram in the Unified Modelling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



* 1. Use Case Diagram

##### CLASS DIAGRAM:

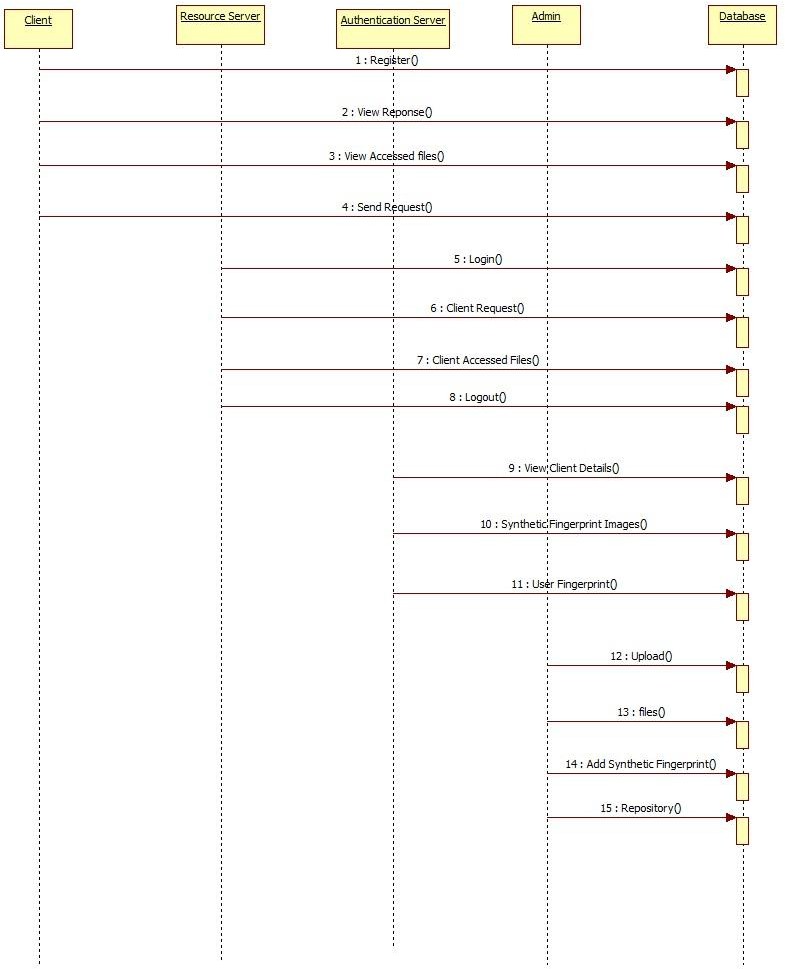
In software engineering, a class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



4.3.1 Class Diagram

### SEQUENCE DIAGRAM:

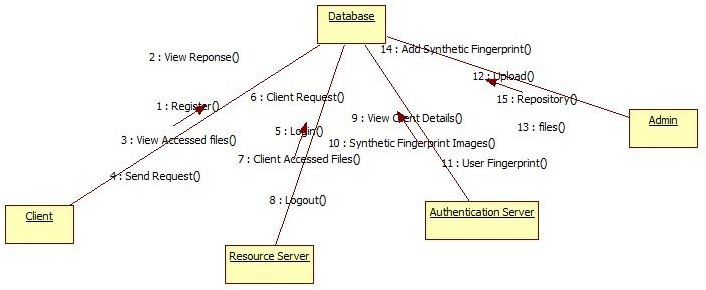
A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



* 1. Sequence Diagram

#### Collaboration Diagram:

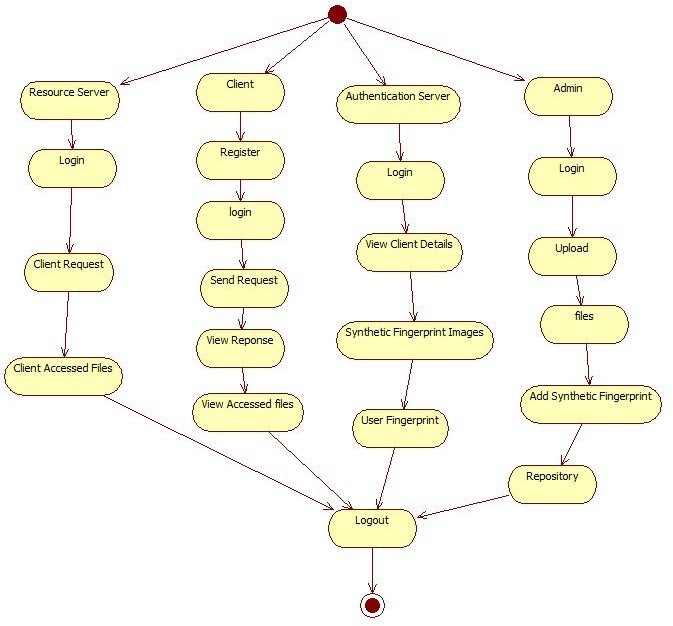
A collaboration diagram resembles a flowchart that portrays the roles, functionality and behaviour of individual objects as well as the overall operation of the system in real time. Objects are shown as rectangles with naming labels inside. These labels are preceded by colons and may be underlined. The relationships between the objects are shown as lines connecting the rectangles. The messages between objects are shown as arrows connecting the relevant rectangles along with labels that define the message sequencing.



4.4.1 Collaboration Diagram

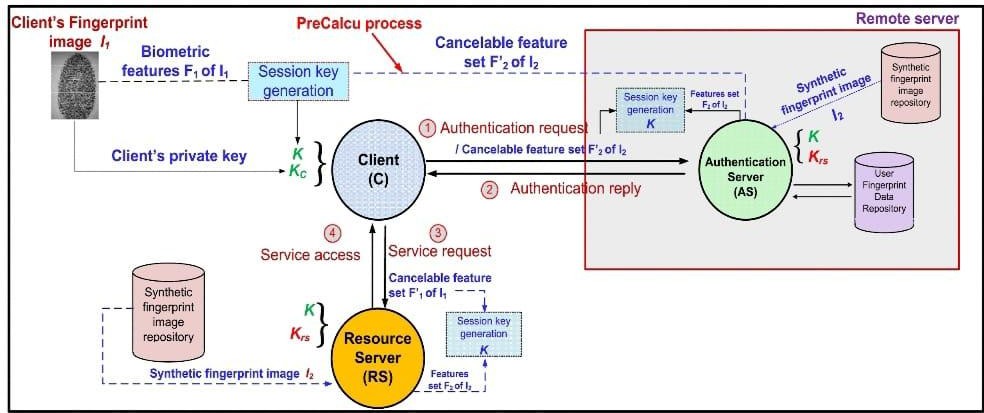
### ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



4.2.2 Activity Diagram

### SYSTEM ARCHITECTURE

****

* 1. System Architecture

## CHAPTER 5 METHODOLOGY

### METHODOLOGY

* + - **Data Collection**: A face detection system collects data by capturing facial images or videos through cameras. Advanced algorithms analyse these data, identifying key

facial features and patterns. The system uses this information to recognize and categorize faces, enabling applications such as security, authentication, and emotion analysis.

* + - **Data Preprocessing**: Perform data cleaning and preprocessing tasks, such as handling missing values, encoding categorical variables, and normalizing numerical features.

Split the dataset into training and testing sets for model evaluation.

* + - **Data Augmentation:** Data augmentation is a technique used in machine learning and deep learning to artificially increase the diversity of a training dataset by applying various transformations to the existing data. The goal is to improve the model's generalization performance and robustness by exposing it to a wider range of variations in the input data. It includes tasks like image rotation, image flipping, image scaling, image translation, etc.
    - **Feature Extraction/Selection**: Relevant features are extracted or selected from the pre-processed data. This step aims to identify the most informative attributes that can contribute to accurate face recognition. Feature engineering techniques may be applied to derive new features or transform existing ones.
    - **Model Architecture Selection**: Successful architectures include Region-based Convolutional Neural Networks (R-CNN), Fast R-CNN, and Faster R-CNN. Alternatives like Single Shot Multi-Box Detector (SSD), You Only Look Once (YOLO) and Multi-Task Cascaded Convolutional Neural Networks (MTCNN).
    - **Model Training**: The data is divided into training and testing datasets. The training dateset is used to build and train the machine learning models, while the testing

dateset is used to evaluate their performance.

* + - **Model Evaluation**: Evaluate the performance of each model using appropriate evaluation metrics, such as accuracy, precision, recall, or Mean Absolute Percentage Error (MAPE). Compare the performance of the models and select the one with the best predictive capability.
    - **Model Selection/Tuning**: Based on the evaluation results, the best-performing model is selected. Hyper parameter tuning may be performed to optimize the model's parameters and improve its predictive capabilities.
    - **Face Recognition**: The selected model is used to recognize the faces and mark the attendance accordingly.
    - **Model Deployment and Monitoring**: The trained and validated model is deployed in

a production environment, where it can be used for real-time application. The model's performance is monitored regularly to ensure its accuracy and reliability.

## CHAPTER 6 SYSTEM IMPLEMENTATION

### HTML

HTML is a language which is used to create web pages with html marking up a page to indicate its format, telling the web browser where you want a new line to begin or how you want text or images aligned and more are possible.

We used the following tags in our project.

###### Table:

Tables are so popular with web page authors is that they let you arrange the elements of a web page in such a way that the browser won’t rearrange them web page authors frequently use tables to structure web pages.

##### TR:

TR is used to create a row in a table encloses <TH> and <TD> elements. <TR> contain many attributes. Some of them are,

* ALIGN: specifies the horizontal alignment of the text in the table row.
* BGCOLOR: Specifies the background color for the row.
* BORDERCOLOR: Sets the external border color for the row.
* VALIGN: Sets the vertical alignment of the data in this row.

##### TH:

TH is used to create table heading.

* ALIGN: Sets the horizontal alignment of the content in the table cell. Sets LEFT, RIGHT, CENTER.
* BACKGROUND: Species the back ground image for the table cell.
* BGCOLOR: Specifies the background color of the table cell
* VALIGN: Sets the vertical alignment of the data. Sets to TOP, MIDDLE, BOTTOM or BASELINE.
* WIDTH: Specifies the width of the cell. Set to a pixel width or a percentage of the display area.

##### TD:

TD is used to create table data that appears in the cells of a table.

* ALIGN: Species the horizontal alignment of content in the table cell. Sets to LEFT, CENTER, RIGHT.
* BGCOLOR: Specifies the background image for the table cell.
* BGCOLOR: sets the background color of the table cells.
* WIDTH: Species the width of the cell

#### Frames:

Frames are used for either run off the page or display only small slices of what are supposed to be shown and to configure the frame we can use <FRAMESET>There are two important points to consider when working with <FRAMESET>.

* <FRAMESET> element actually takes the place of the <BODY> element in a document.
* Specifying actual pixel dimensions for frames .

<FRAME> Elements are used to create actual frames.

From the frameset point of view dividing the browser into tow vertical frames means creating two columns using the <FRAMESET> elements COLS attribute.

The syntax for vertical fragmentation is,

<FRAMESET COLS =”50%, 50%”>

</FRAMESET>

Similarly if we replace COLS with ROWS then we get horizontal fragmentation. The syntax for horizontal fragmentation is,

<FRAMESET ROWS=”50%, 50%”>

</FRAMESET>

#### Form:

The purpose of FORM is to create an HTML form; used to enclose HTML controls, like buttons and text fields.

#### Attribute:

* ACTION: Gives the URL that will handle the form data.
* NAME: Gives the name to the form so you can reference it in code set to an alphanumeric string.
* METHOD: method or protocol is used to sending data to the target action URL. The GET method is the default, it is used to send all form name/value pair information in an URL. Using the POST method, the content of the form are encoded as with the GET method, but are sent in environment variables.

###### Controls in HTML:

<INPUT TYPE =BUTTON>:

Creates an html button in a form.

##### ATTRIBUTES:

* NAME: gives the element a name. Set to alphanumeric characters.
* SIZE: sets the size.
* VALUE: sets the caption of the element.

<INPUT TYPE = PASSWORD>:

Creates a password text field, which makes typed input.

##### ATTRIBUTES:

* NAME: gives the element a name, set to alphanumeric characters.
* VALUE: sets the default content of the element.

<INPUT TYPE=RADIO>:

Creates a radio button in a form.

##### ATTRIBUTE:

* NAME: Gives the element a name. Set to alphanumeric character.
* VALUE: Sets the default content of the element.

<INPUT TYPE=SUBMIT>:

Creates a submit button that the user can click to send data in the form back to the web server

### JAVA SCRIPT:

Java script originally supported by Netscape navigator is the most popular web scripting language today. Java script lets you embedded programs right in your web pages and run these programs using the web browser. You place these programs in a <SCRIPT> element, usually within the <HEAD> element. If you want the script to write directly to the web page, place it in the <BODY> element.

###### Java script Methods: Writeln:

Document.writeln () is a method, which is used to write some text to the current web

page.

###### onClick:

Occurs when an element is clicked.

###### onLoad:

Occurs when the page loads.

###### onMouseDown:

Occurs when a mouse button goes down.

###### onMouseMove:

Occurs when the mouse moves.

###### OnUnload:

Occurs when a page is unloaded.

#### MySQL:

MySQL is an open source relational database management system (RDBMS).This is the most popular database system used with PHP. MySQL is distributed and supported by Oracle Corporation.MySQL runs on almost all platforms including Linux,

Unix and Windows. Although it can be used in a wide range of applications, MySQL is often associated with web applications and online publishing.

MySQL is an essential constituent of an open source enterprise stack called LAMP. LAMP is a web development platform that uses Linux as an operating system, in the form of Apache web server, MySQL relational database management system and PHP object-oriented scripting language.

###### Advantages of MySQL:

**Data Security:** MySQL is globally renowned for being the most secure and reliable database management system used in popular web applications including WordPress, Drupal, Joomla, Facebook and Twitter.

**High Performance:** MySQL features a distinct storage-engine framework that facilitates system administrators to configure the MySQL database server for a flawless performance.

**Round-the-Clock Up-time:** MySQL comes with the assurance of 24×7 up-time and offers a wide range of high-availability solutions, including specialized cluster servers and master/slave replication configurations.

**The Flexibility of Open Source:** All the fears and worries that arise in an open- source solution can be brought to an end with MySQL’s round-the-clock support and enterprise indemnification. The secure processing and trusted software of MySQL combine to provide effective transactions for large-volume projects. It makes maintenance, debugging and upgrades fast and easy while enhancing the end-user experience.

### PYTHON

Python programming language is used for building the machine learning model.

###### Introduction

Python is an object-oriented, high-level language, interpreted, dynamic and multipurpose programming language. Python is easy to learn yet powerful and versatile scripting language which makes it attractive for Application Development.Python's syntax and dynamic typing with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas.Python supports multiple programming pattern, including object-

oriented programming, imperative and functional programming or procedural styles. Python is not intended to work on special area such as web programming. That is why it is known as multipurpose because it can be used with web, enterprise, 3D CAD etc.We don't need to use data types to declare variable because it is dynamically typed so we can write a=l0 to declare an integer value in a variable. Python makes the development and debugging fast because there is no compilation step included in python development and edit-test-debug cycle is very fast.

###### Python Features

* **Easy to Use:** Python is easy to very easy to use and high-level language. Thus, it is a programmer-friendly language.
* **Interpreted Language:** Python is an interpreted language i.e., interpreter executes the code line by line at a time. This makes debugging easy and thus suitable for beginners.
* **Cross-platform language:** Python can run equally on different platforms such as Windows, Linux, Unix, Macintosh etc. Thus, Python is a portable language.
* **Free and Open Source:** Python language is freely available.
* **Object-Oriented language:** Python supports object-oriented language. The concept of classes and objects comes into existence.
* **Extensible:** It implies that other languages such as C/C+ can be used to compile the code and thus it can be used further in your Python code.
* **Large Standard Library:** Python has a large and broad library.
* **GUI Programming:** Graphical user interfaces can be developed using Python.

###### Python History

Python laid its foundation in the late 1980s. The implementation of Python was started in the December 1989 by Guido Van Rossum at CWI in Netherland.ABC programming language is said to be the predecessor of Python language which was capable of Exception Handling and interfacing with Amoeba Operating System.

###### Advantages of Python

Let’s see how Python dominates over other languages.

###### Extensive Libraries

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don’t have to write the complete code for that manually.

###### Extensible

As we have seen earlier, Python can be extended to other languages. You can

write some of your code in languages like C++ or C. This comes in handy,especially in projects.

###### Advantages of Python Over Other Languages:

* + - 1. **Less Coding**

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don’t have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

###### Affordable

Python is free therefore individuals, small companies or big organizations can

leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

###### Python is for Everyone

Python code can run on any machine whether it is Linux, Mac or Windows.

###### Disadvantages of Python

So far, we’ve seen why Python is a great choice for your project. But if you

choose it, you should be aware of its consequences as well. Let’s now see the downsides of choosing Python over another language.

###### Speed Limitations

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn’t a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

###### Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle. The reason it is not so famous despite the existence of Brython is that it isn’t that secure.

#### Django

###### Introduction:

Django is a powerful, high-level web framework written in Python that promotes rapid development and clean, pragmatic design. It is designed to help developers build secure, scalable, and maintainable web applications efficiently.

###### Key Characteristics of Django:

Open Source: Freely available with a strong community support.

###### MVT Architecture:

Django follows the Model-View-Template (MVT) architectural pattern, separating data, user interface, and business logic.

###### Security:

Django provides built-in security features to protect against threats like SQL Injection, Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), and Clickjacking.

###### Scalability

Suitable for both small and large-scale applications.Batteries Included: Comes with a wide range of features like authentication, URL routing, database management, form handling, and an admin panel.

###### MVT (Model-View-Template) Components:

**Model:** Handles database structure, creation, and access. Defines the data format (e.g., fingerprint records, user profiles).

**View:** Manages the logic that processes user requests and returns appropriate responses (e.g., verifying biometrics, authenticating users).

**Template:** Manages the front-end layout (HTML files) shown to users.

###### Advantages of Django:

**Rapid Development:** Reduces the time needed to build applications.

**Re-usability:** Apps and modules can be reused across different projects.

**Automatic Admin Interface:** Provides a ready-to-use interface for database management.

**ORM (Object-Relational Mapping):** Facilitates easy interaction with databases without writing raw SQL.

#### Use of Django in This Project:

In our project, Secure Cloud Services using Biometric Fingerprint Image, Django is used to:

* Handle user authentication through fingerprint verification.
* Manage fingerprint image uploads and storage securely.
* Implement encryption and decryption operations for secure file transfer.
* Organize user data and cloud services access.
* Provide a clean and user-friendly interface for interaction.

### XAMPP:

XAMPP:

XAMPP stands for Cross-Platform (X), Apache (A), MariaDB (M), PHP (P) and Perl (P). It is simply a web server if we want to make a website or designing and make a working website then XAMPP is useful .it gives an environment of his server works.

1. It contains apache, mysql, filezilla servers by which we can use them and helps us in login and logout sessions, cookies we gives a good help in websites
2. Also, it has wordpress feature by which it contains many themes of websites which are popular and we can use them to make a website without using so much php coding ,HTML, CSS etc.
3. How to use it: 1. if we are working on mysql then we just on the server of mysql and go to php admin page.
4. To work on php based web pages we just on the server and then, code on a notepad by using php pages.

##### ALGORITHMS USED

###### Password Hashing

**Explanation:**

* + Converts password into a secure one-way hash.
  + Protect passwords even if the database is hacked

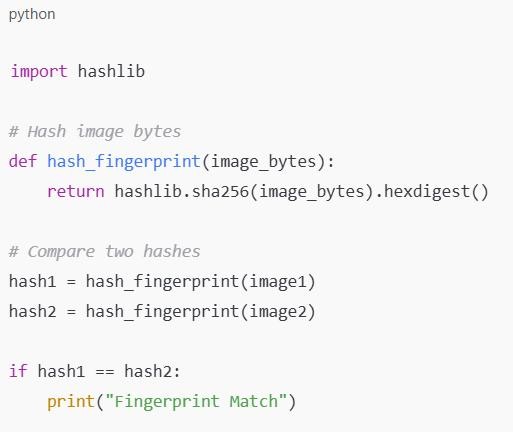
Syntax(Django Style):



###### Biometric Matching(Image Hashing)

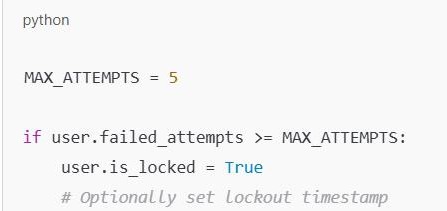
**Explanation:**

* + Hashes fingerprint image or its extracted features.
  + Used to compare new fingerprint data

securily

###### Account lockout MechanismExplanation Explanation:

* + Temporarily locks the user account after too many failed login attempts.



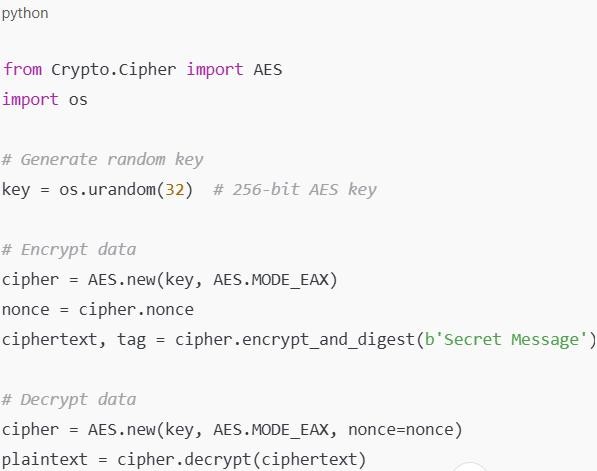
* + You check failed\_attempts during each login attempt.

###### Key Generation& Encryption

**Explanation:**

* + Generate encryption keys and use them to encrypt/decrypt sensitive data.

###### Syntax (AES Encryption using PyCryptodome):

****

1. **Session Handling(Django Session:**

* Django tracks logged-in users using server-side sessions.
* Session data is saved securely and linked to users via cookies.

## CHAPTER 7 TESTING

### 7.1 TESTING

Testing involves operation of a system or application under controlled conditions and evaluating the results. The controlled conditions should include both normal and abnormal conditions. Testing should intentionally attempt to make things go wrong to determine if things happen when they shouldn't or things don't happen when they should. It is oriented to 'detection'.

TYPES OF TESTINGS

###### Unit Testing:

Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. Unit testing is often automated but it can also be done manually. This testing mode is a component of Extreme Programming (XP), a pragmatic method of software development that takes a meticulous approach to building a product by means of continual testing and revision.

Unit tests are written from a programmer's perspective. They ensure that a particular method of a class successfully performs a set of specific tasks. Each test confirms that a method produces the expected output when given a known input.

###### Performance Testing:

Performance testing is the process of determining the speed or effectiveness of a computer, network, software program or device. This process can involve quantitative tests done in a lab, such as measuring the response time or the number of MIPS (millions of instructions per second) at which a system functions. Qualitative attributes such as Reliability, scalability and interoperability may also be evaluated. Performance testing is often done in conjunction with stress testing.

Performance testing can verify that a system meets the specifications claimed by its manufacturer or vendor. The process can compare two or more devices or programs in terms of parameters such as speed, data transfer rate, bandwidth, throughput, efficiency or reliability.

Performance testing can also be used as a diagnostic aid in locating communications bottlenecks. Often a system will work much better if a problem is resolved at a single point or in a single component. For example, even the fastest computer will function poorly on today's Web if the connection occurs at only 40 to 50 Kbps (kilobits per second).

###### Integration Testing:

Integration testing, also known as integration and testing (I&T), is a software development process which program units are combined and tested as groups in multiple ways. In this context, a unit is defined as the smallest testable part of an application. Integration testing can expose problems with the interfaces among program components before trouble occurs in real-world program execution. Integration testing is a component of Extreme Programming (XP), a pragmatic method of software development that takes a meticulous approach to building a product by means of continual testing and revision.

* 1. **Test cases:**

**Test case for Login form:**

**Test case for User Registration form:**

**Test case3:**

**Test case for Change Password:**

When the old password does not match with the new password ,then this results in displaying an error message as “ OLD PASSWORD DOES NOT MATCH WITH THE NEW PASSWORD”.

###### Test case 4:

**Test case for Forget Password:**

When a user forgets his password he is asked to enter Login name, ZIP code, Mobile number. If these are matched with the already stored ones then user will get his Original password.

## CHAPTER 8 PROGRAMMING CODE

1.INDEX

body { margin: 0;

font-family: 'Poppins', sans-serif; background-color: #f5f7fa; color: #333;

}

header {

background: linear-gradient(to right, #1a3a5f, #294a7d); color: white;

padding: 20px; text-align: center;

box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);

}

.title h1 {

font-size: 2.5rem; margin: 0;

font-weight: 700;

}

.title p {

margin: 5px 0 0; font-size: 1.2rem; font-weight: 300;

}

nav {

background-color: #243c60;

box-shadow: 0 4px 8px rgba(0,0,0,0.1);

}

nav ul {

list-style: none; display: flex;

justify-content: center; padding: 0;

margin: 0;

}

nav ul li { margin: 0;

}

nav ul li a {

padding: 16px 24px; display: block; color: white;

text-decoration: none; font-weight: 500;

transition: background 0.3s ease;

}

nav ul li a:hover {

background-color: #1c2e4a;

}

main {

padding: 40px 20px; background-color: #ffffff;

max-width: 1000px; margin: 30px auto; border-radius: 12px;

box-shadow: 0 10px 25px rgba(0, 0, 0, 0.05);

}

.image-container { text-align: center;

margin-bottom: 30px;

}

.image-container img { max-width: 100%; height: auto;

border-radius: 10px;

box-shadow: 0 5px 15px rgba(0,0,0,0.1);

}

.about-project h2 { font-size: 1.8rem; color: #1a3a5f;

margin-bottom: 10px;

}

.about-project p { font-size: 1rem; line-height: 1.8; text-align: justify;

}

footer {

text-align: right; padding: 15px 20px; font-size: 0.9rem; color: #888;

background-color: #f2f2f2; border-top: 1px solid #ddd;

}

/\* Dropdown styling \*/

.dropdown { position: relative;

}

.dropdown-content { display: none; position: absolute;

background-color: #1a3a5f; min-width: 160px;

box-shadow: 0 8px 16px rgba(0,0,0,0.2); z-index: 1;

border-radius: 0 0 8px 8px; overflow: hidden;

}

.dropdown-content li {

border-top: 1px solid rgba(255,255,255,0.1);

}

.dropdown-content li a { padding: 12px 16px;

display: block; color: white; text-align: left;

text-decoration: none; transition: background 0.3s;

}

.dropdown-content li a:hover { background-color: #0d213d;

}

/\* Show dropdown on hover \*/

.dropdown:hover .dropdown-content { display: block;

} 2.LOGIN

body {

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif; background: #f5f7fa;

margin: 0;

padding: 0;

}

header {

background-color: #2c3e50; color: white;

padding: 20px 0; text-align: center;

box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

}

.title h1 { margin: 0;

font-size: 2rem;

}

.title p {

margin: 5px 0 0; font-size: 1rem; font-weight: 300;

}

main {

display: flex;

justify-content: center; align-items: center; min-height: 70vh;

}

.form-container { background: white; padding: 30px 40px; border-radius: 10px;

box-shadow: 0 10px 25px rgba(0, 0, 0, 0.1);

width: 100%;

max-width: 400px;

}

form label { display: block;

margin-top: 15px; font-weight: bold;

}

form input[type="text"],

form input[type="password"] { width: 100%;

padding: 10px; margin-top: 5px; border-radius: 5px;

border: 1px solid #ccc; box-sizing: border-box;

}

form button { margin-top: 20px; padding: 10px;

background-color: #2980b9; color: white;

border: none; width: 100%; border-radius: 5px; cursor: pointer; font-size: 1rem;

}

form button:hover { background-color: #3498db;

}

footer {

text-align: center; padding: 15px;

background-color: #ecf0f1; color: #333;

position: fixed; bottom: 0;

width: 100%;

}

# BACKEND

import os

import mysql.connector import binascii

import logging

from PIL import Image import imagehash

from django.conf import settings

from django.shortcuts import render, redirect from django.http import HttpResponse, Http404

from django.core.files.storage import FileSystemStorage from django.utils.timezone import now

from django.contrib import messages

from django.contrib.auth.hashers import make\_password, check\_password from datetime import datetime

from utils.encryption\_utils import encrypt\_file, decrypt\_file

from utils.key\_utils import generate\_private\_key, generate\_session\_key from .decorators import login\_required\_custom, admin\_login\_required

logger = logging.getLogger(\_name\_)

# INDEX

def index(request): logger.info("Visited index page") return render(request, 'index.html')

# REGISTER USER

def register\_user(request):

if request.method == 'POST':

username = request.POST['username'] email = request.POST['email'] raw\_password = request.POST['password'] biometric\_file = request.FILES['biometric'] password = make\_password(raw\_password)

fs = FileSystemStorage(location=os.path.join(settings.MEDIA\_ROOT, 'biometric')) filename = fs.save(biometric\_file.name, biometric\_file)

biometric\_path = 'biometric/' + filename private\_key = generate\_private\_key()

conn = mysql.connector.connect(host="localhost", user="root", password="", database="BBAM")

cursor = conn.cursor() cursor.execute(

"INSERT INTO user (username, email, password, biometric\_path, private\_key) VALUES (%s, %s, %s, %s, %s)",

(username, email, password, biometric\_path, private\_key)

)

conn.commit() cursor.close() conn.close()

logger.info(f"New user registered: {username}, email: {email}") return redirect('login')

return render(request, 'register\_user.html')

# LOGIN

def login(request):

if request.method == 'POST':

username = request.POST['username'] input\_password = request.POST['password'] biometric\_file = request.FILES['biometric']

conn = mysql.connector.connect(host="localhost", user="root", password="", database="BBAM")

cursor = conn.cursor(dictionary=True)

cursor.execute("SELECT \* FROM user WHERE username = %s", (username,)) user = cursor.fetchone()

if user:

now = datetime.now() # Lockout check

if user['lockout\_until'] and now < user['lockout\_until']: messages.error(request, "Account is locked. Try again later.") cursor.close()

conn.close()

return render(request, 'login.html')

if check\_password(input\_password, user['password']): # Save uploaded biometric temporarily

fs = FileSystemStorage(location=os.path.join(settings.MEDIA\_ROOT, 'temp\_biometric'))

temp\_filename = fs.save(biometric\_file.name, biometric\_file)

uploaded\_biometric\_path = os.path.join(settings.MEDIA\_ROOT, 'temp\_biometric', temp\_filename)

stored\_biometric\_path = os.path.join(settings.MEDIA\_ROOT, user['biometric\_path'])

try:

uploaded\_hash =

imagehash.average\_hash(Image.open(uploaded\_biometric\_path))

stored\_hash = imagehash.average\_hash(Image.open(stored\_biometric\_path))

diff = uploaded\_hash - stored\_hash

if diff <= 5:

# Successful login

session\_key = generate\_session\_key(str(uploaded\_hash), user['private\_key'])

# Reset failed attempts

cursor.execute("UPDATE user SET failed\_attempts = 0, lockout\_until = NULL WHERE username = %s", (username,))

conn.commit()

# Set session

request.session['username'] = user['username'] request.session['email'] = user['email'] request.session['session\_key'] = session\_key.hex()

messages.success(request, "Login successful!") logger.info(f"Login SUCCESS for user: {username}") os.remove(uploaded\_biometric\_path)

cursor.close() conn.close()

return redirect('dashboard') else:

messages.error(request, "Biometric mismatch.") logger.warning(f"Biometric mismatch for user: {username}") os.remove(uploaded\_biometric\_path)

except Exception as e:

logger.error(f"Error processing biometric: {str(e)}") messages.error(request, "Error processing biometric.") if os.path.exists(uploaded\_biometric\_path):

os.remove(uploaded\_biometric\_path)

# If password is incorrect or biometric failed new\_attempts = user['failed\_attempts'] + 1 lockout\_time = None

if new\_attempts >= 5:

lockout\_time = now + timedelta(minutes=10)

messages.error(request, "Account locked due to too many failed attempts. Try again

later.")

logger.warning(f"User {username} locked out until {lockout\_time}") else:

messages.error(request, "Invalid username or password or biometric mismatch.") logger.warning(f"Failed login attempt {new\_attempts} for user {username}")

cursor.execute("UPDATE user SET failed\_attempts = %s, lockout\_until = %s WHERE username = %s",

(new\_attempts, lockout\_time, username)) conn.commit()

else:

messages.error(request, "Invalid username or password.")

cursor.close() conn.close()

return render(request, 'login.html')

# In your views.py

@login\_required\_custom def dashboard(request):

username = request.session['username']

conn = mysql.connector.connect(host="localhost", user="root", password="", database="BBAM")

cursor = conn.cursor(dictionary=True)

if request.method == 'POST': if 'upload' in request.POST:

uploaded\_file = request.FILES['file'] file\_data = uploaded\_file.read()

session\_key\_hex = request.session.get('session\_key')

if session\_key\_hex: try:

key = binascii.unhexlify(session\_key\_hex) encrypted\_data = encrypt\_file(file\_data, key)

upload\_path = os.path.join(settings.MEDIA\_ROOT, 'uploads') os.makedirs(upload\_path, exist\_ok=True)

file\_path = os.path.join(upload\_path, uploaded\_file.name)

with open(file\_path, 'wb') as f: f.write(encrypted\_data)

cursor.execute(

"INSERT INTO user\_files (username, filename) VALUES (%s, %s)", (username, 'uploads/' + uploaded\_file.name)

)

conn.commit()

logger.info(f"File uploaded by {username}: {uploaded\_file.name}") messages.success(request, "File uploaded successfully.")

except Exception as e:

logger.error(f"File encryption/upload failed: {str(e)}") messages.error(request, "Upload failed.")

else:

messages.error(request, "Session expired.") return redirect('login')

elif 'update\_biometric' in request.POST: new\_biometric\_file = request.FILES['new\_biometric']

fs = FileSystemStorage(location=os.path.join(settings.MEDIA\_ROOT, 'biometric')) filename = fs.save(new\_biometric\_file.name, new\_biometric\_file)

new\_biometric\_path = 'biometric/' + filename

cursor.execute("UPDATE user SET biometric\_path = %s WHERE username = %s", (new\_biometric\_path, username))

conn.commit()

messages.success(request, "Biometric data updated successfully.") logger.info(f"{username} updated biometric data.")

elif 'delete\_file' in request.POST:

file\_to\_delete = request.POST.get('filename')

file\_path = os.path.join(settings.MEDIA\_ROOT, file\_to\_delete)

try:

os.remove(file\_path)

cursor.execute("DELETE FROM user\_files WHERE username = %s AND filename = %s",

(username, file\_to\_delete)) conn.commit()

messages.success(request, "File deleted successfully.") logger.info(f"File deleted by {username}: {file\_to\_delete}")

except Exception as e:

logger.error(f"File deletion failed: {str(e)}") messages.error(request, "File deletion failed.")

cursor.execute("SELECT \* FROM user\_files WHERE username = %s", (username,))

files = cursor.fetchall() cursor.close() conn.close()

return render(request, 'dashboard.html', { 'username': username,

'email': request.session.get('email'), 'files': files

})

# DOWNLOAD FILE

@login\_required\_custom

def download\_file(request, filename): username = request.session.get('username')

session\_key\_hex = request.session.get('session\_key')

if not username or not session\_key\_hex: return redirect('login')

file\_path = os.path.join(settings.MEDIA\_ROOT, 'uploads', filename) if not os.path.exists(file\_path):

raise Http404("File not found")

try:

key = binascii.unhexlify(session\_key\_hex) with open(file\_path, 'rb') as f:

encrypted\_data = f.read()

decrypted\_data = decrypt\_file(encrypted\_data, key)

logger.info(f"File download by {username}: {filename}")

response = HttpResponse(decrypted\_data, content\_type='application/octet-stream') response['Content-Disposition'] = f'attachment; filename="{filename}"'

return response except Exception as e:

logger.error(f"Decryption failed: {str(e)}")

return HttpResponse("Decryption failed", status=403) # LOGOUT

def logout\_user(request): request.session.flush()

messages.success(request, "You have been logged out.") return redirect('login')

# ADMIN LOGIN

def admin\_login(request):

if request.method == 'POST': email = request.POST['email']

password\_input = request.POST['password'] logger.info(f"Admin login attempt: {email}")

conn = mysql.connector.connect(host="localhost", user="root", password="", database="BBAM")

cursor = conn.cursor(dictionary=True)

cursor.execute("SELECT \* FROM user WHERE email = %s AND role = 'admin'", (email,))

admin = cursor.fetchone() cursor.close()

conn.close()

if admin and check\_password(password\_input, admin['password']): request.session['admin\_email'] = admin['email'] request.session['admin\_username'] = admin['username'] messages.success(request, "Admin login successful!")

return redirect('admin\_dashboard') else:

messages.error(request, "Invalid credentials or not an admin.") logger.warning(f"Admin login FAILED: {email}")

return render(request, 'admin\_login.html') # ADMIN DASHBOARD

@admin\_login\_required

def admin\_dashboard(request):

admin\_email = request.session.get('admin\_email')

conn = mysql.connector.connect(host="localhost", user="root", password="", database="BBAM")

cursor = conn.cursor(dictionary=True) cursor.execute("SELECT username FROM user") users = cursor.fetchall()

cursor.close() conn.close()

return render(request, 'admin\_dashboard.html', { 'admin\_username': request.session.get('admin\_username'), 'admin\_email': admin\_email,

'users': users,

})

@admin\_login\_required def admin\_logout(request):

request.session.flush()

messages.success(request, "Logged out successfully.") return redirect('admin\_login')

@admin\_login\_required

def view\_user\_activities(request): username\_filter = request.GET.get('username') date\_filter = request.GET.get('date')

conn = mysql.connector.connect(host="localhost", user="root", password="", database="BBAM")

cursor = conn.cursor(dictionary=True)

base\_query = "SELECT \* FROM user\_files WHERE 1=1" params = []

if username\_filter:

base\_query += " AND username = %s" params.append(username\_filter)

if date\_filter:

base\_query += " AND DATE(upload\_time) = %s" params.append(date\_filter)

cursor.execute(base\_query, tuple(params)) files = cursor.fetchall()

cursor.execute("SELECT DISTINCT username FROM user\_files") usernames = cursor.fetchall()

cursor.close()

conn.close()

return render(request, 'admin\_user\_activities.html', { 'files': files,

'usernames': usernames, 'selected\_user': username\_filter, 'selected\_date': date\_filter,

})

@admin\_login\_required def log\_dashboard(request):

conn = mysql.connector.connect(host="localhost", user="root", password="", database="BBAM")

cursor = conn.cursor(dictionary=True)

cursor.execute("SELECT \* FROM log\_entry ORDER BY created\_at DESC LIMIT 100") logs = cursor.fetchall()

cursor.close() conn.close()

return render(request, 'log\_dashboard.html', {'logs': logs}) @admin\_login\_required

def view\_user(request, username):

conn = mysql.connector.connect(host="localhost", user="root", password="", database="BBAM")

cursor = conn.cursor(dictionary=True)

cursor.execute("SELECT username, email, private\_key FROM user WHERE username

= %s", (username,))

user = cursor.fetchone() if not user:

cursor.close()

conn.close()

raise Http404("User not found")

cursor.execute("SELECT filename, upload\_time FROM user\_files WHERE username = %s", (username,))

user\_files = cursor.fetchall() cursor.close()

conn.close()

return render(request, 'admin\_view\_user.html', { 'user': user,

'user\_files': user\_files,

'session\_key': request.session.get('session\_key')

})

#Auth\_Server\_login

logger = logging.getLogger(\_name\_) def auth\_serv\_login(request):

if request.method == 'POST': email = request.POST['email']

password\_input = request.POST['password'] logger.info(f"Auth Server login attempt: {email}")

conn = mysql.connector.connect(host="localhost", user="root", password="", database="BBAM")

cursor = conn.cursor(dictionary=True)

cursor.execute("SELECT \* FROM user WHERE email = %s AND role = 'Authentication Server'", (email,))

user = cursor.fetchone() cursor.close() conn.close()

if user and check\_password(password\_input, user['password']): request.session['user\_email'] = user['email'] request.session['user\_username'] = user['username'] messages.success(request, "Login successful!")

return redirect('auth\_serv\_dashboard') else:

messages.error(request, "Invalid credentials or unauthorized access.") logger.warning(f"Auth Server login FAILED: {email}")

return render(request, 'auth\_serv\_login.html')

from django.views.decorators.http import require\_POST def auth\_serv\_dashboard(request):

if not request.session.get('user\_email') or request.session.get('user\_username') is None: messages.error(request, "You must log in as an Authentication Server.")

return redirect('auth\_serv\_login')

conn = mysql.connector.connect(host="localhost", user="root", password="", database="BBAM")

cursor = conn.cursor(dictionary=True) if request.method == 'POST':

user\_to\_auth = request.POST.get('username\_to\_auth')

cursor.execute("UPDATE user SET is\_authenticated = 1 WHERE username = %s", (user\_to\_auth,))

conn.commit()

messages.success(request, f"User '{user\_to\_auth}' authenticated successfully.") logger.info(f"Authentication Server authenticated user: {user\_to\_auth}")

cursor.execute("SELECT username, email, is\_authenticated FROM user WHERE role = 'user'")

users = cursor.fetchall() cursor.close() conn.close()

return render(request, 'auth\_serv\_dashboard.html', { 'auth\_username': request.session.get('user\_username'), 'users': users,

})

def auth\_serv\_logout(request): request.session.flush()

messages.success(request, "You have been logged out.") return redirect('auth\_serv\_login')

import logging

import mysql.connector

from django.shortcuts import render, redirect from django.contrib import messages

from django.contrib.auth.hashers import check\_password

from django.views.decorators.http import require\_http\_methods logger = logging.getLogger(\_name\_) @require\_http\_methods(["GET", "POST"])

def resource\_serv\_login(request): if request.method == 'POST':

email = request.POST['email'] password\_input = request.POST['password']

logger.info(f"Resource Server login attempt: {email}")

conn = mysql.connector.connect(host="localhost", user="root", password="", database="BBAM")

cursor = conn.cursor(dictionary=True)

cursor.execute("SELECT \* FROM user WHERE email = %s AND role = 'Resource Server'", (email,))

user = cursor.fetchone() cursor.close() conn.close()

if user and check\_password(password\_input, user['password']): request.session['user\_email'] = user['email'] request.session['user\_username'] = user['username'] messages.success(request, "Login successful!")

return redirect('resource\_serv\_dashboard') else:

messages.error(request, "Invalid credentials or unauthorized access.") logger.warning(f"Resource Server login FAILED: {email}")

return render(request, 'resource\_server.html') def resource\_serv\_dashboard(request):

username\_filter = request.GET.get('username') date\_filter = request.GET.get('date')

conn = mysql.connector.connect(host="localhost", user="root", password="", database="BBAM")

cursor = conn.cursor(dictionary=True)

base\_query = "SELECT \* FROM user\_files WHERE 1=1" params = []

if username\_filter:

base\_query += " AND username = %s" params.append(username\_filter)

if date\_filter:

base\_query += " AND DATE(upload\_time) = %s" params.append(date\_filter)

cursor.execute(base\_query, tuple(params)) files = cursor.fetchall()

cursor.execute("SELECT DISTINCT username FROM user\_files") usernames = cursor.fetchall()

cursor.close() conn.close()

return render(request, 'resource\_serv\_dashboard.html', { 'files': files,

'usernames': usernames, 'selected\_user': username\_filter, 'selected\_date': date\_filter,

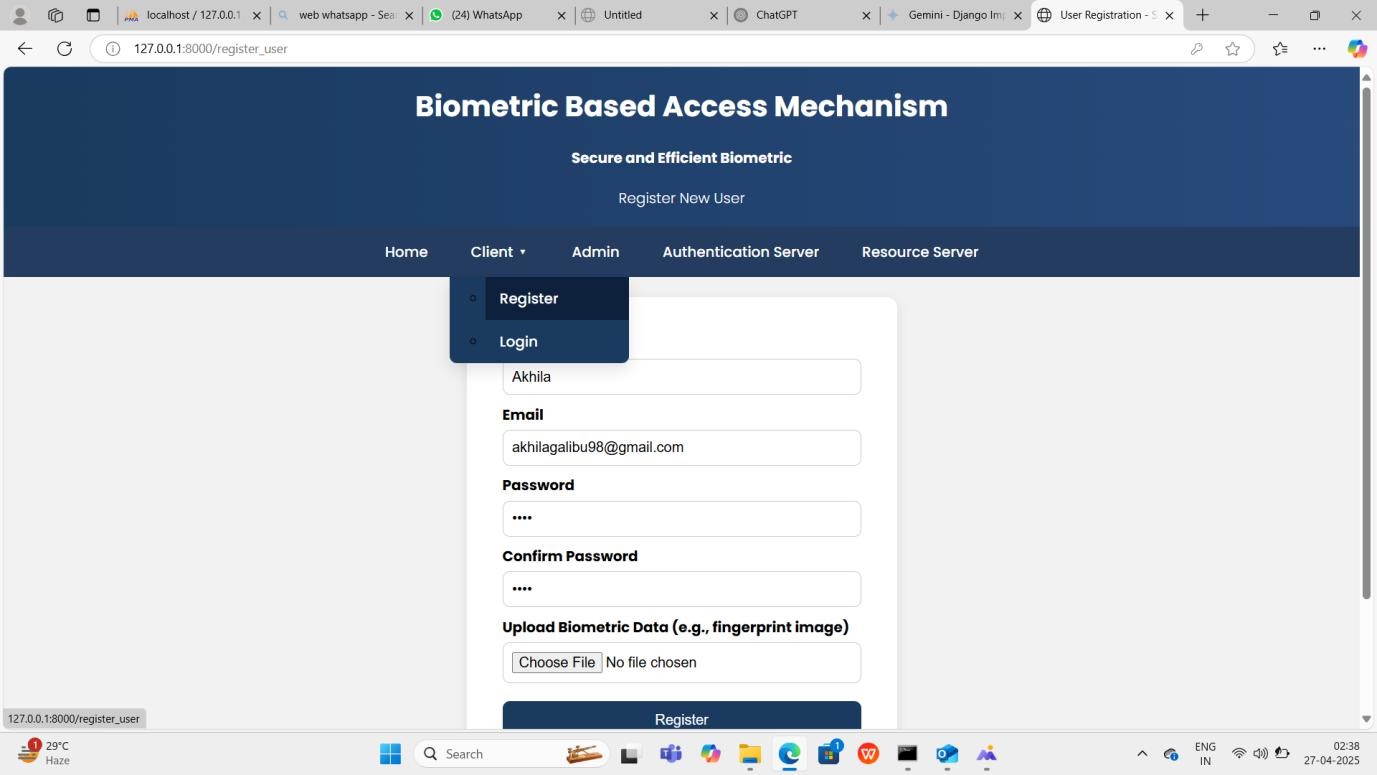
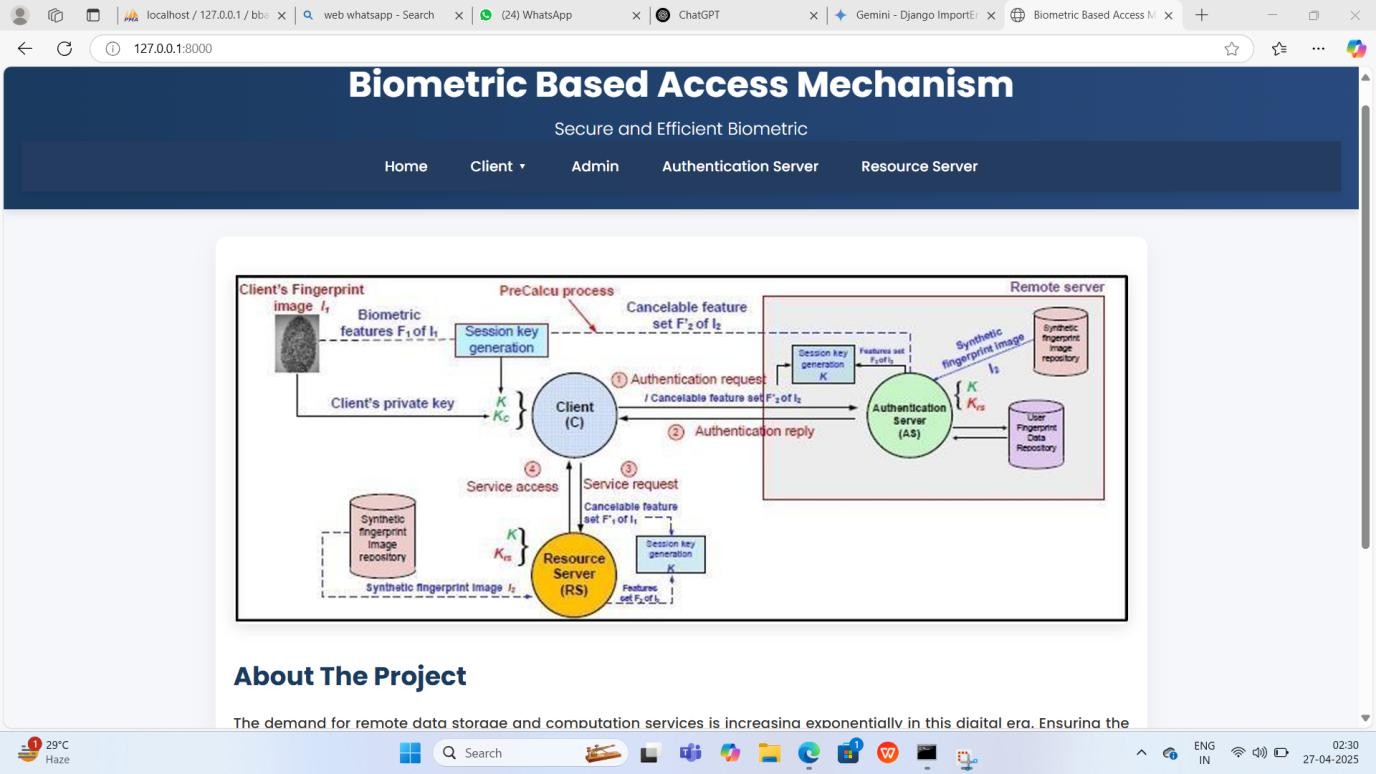
})

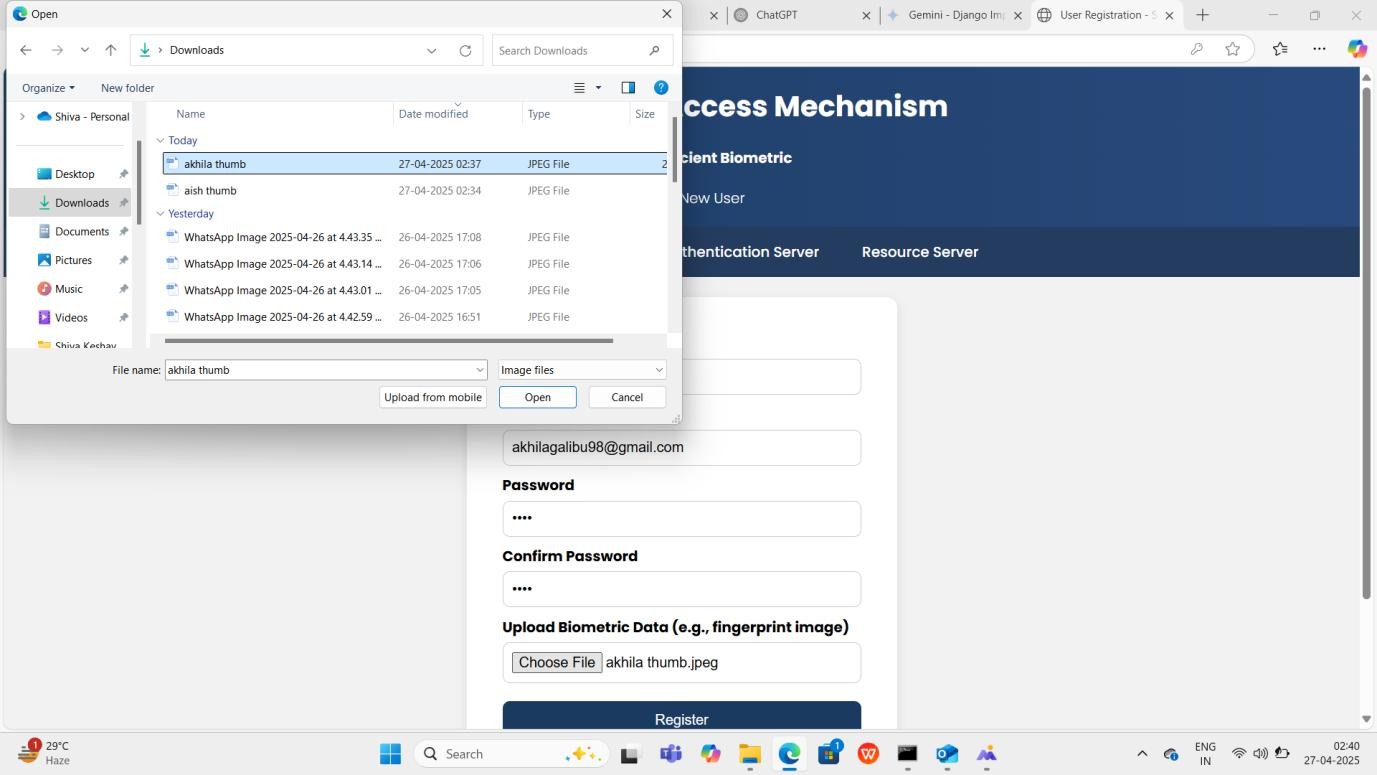
from django.shortcuts import redirect from django.contrib import messages

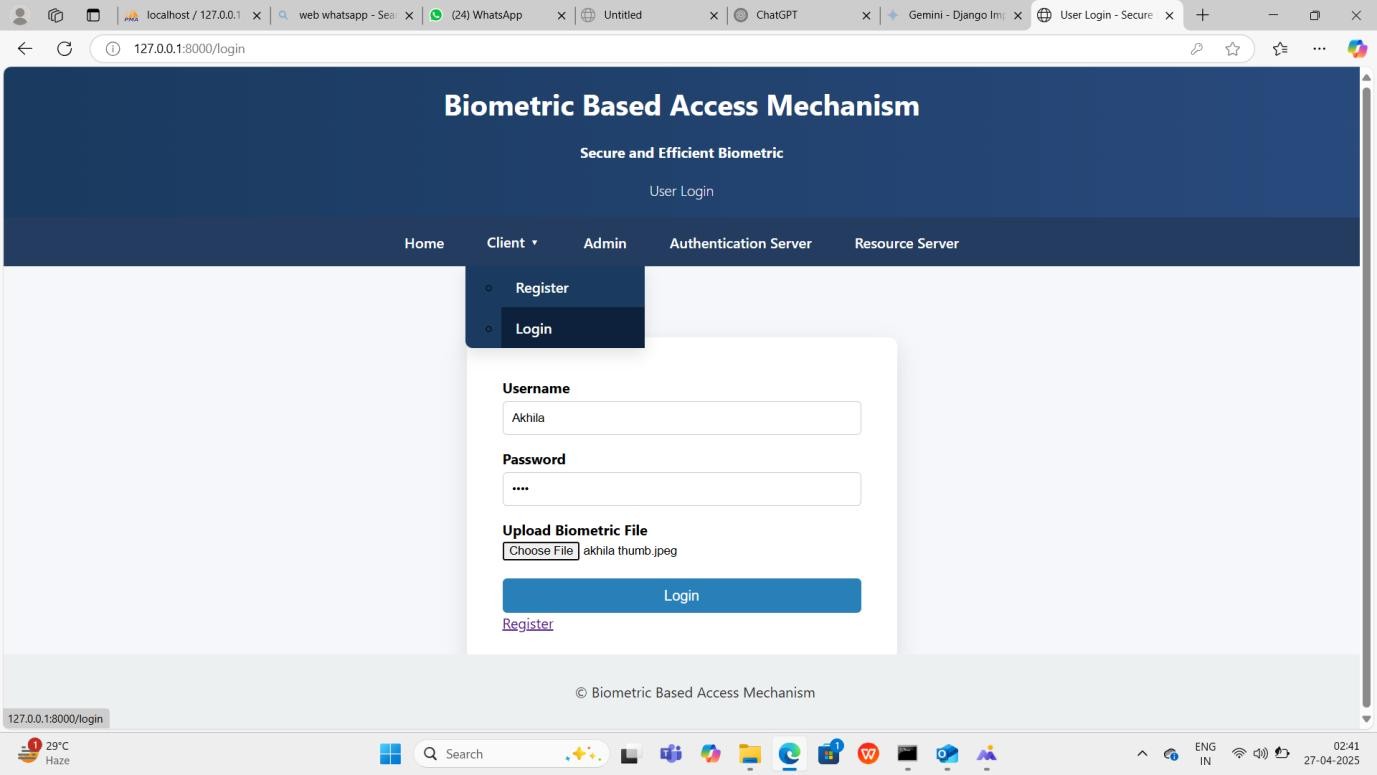
def logout\_resource\_serv(request): request.session.flush()

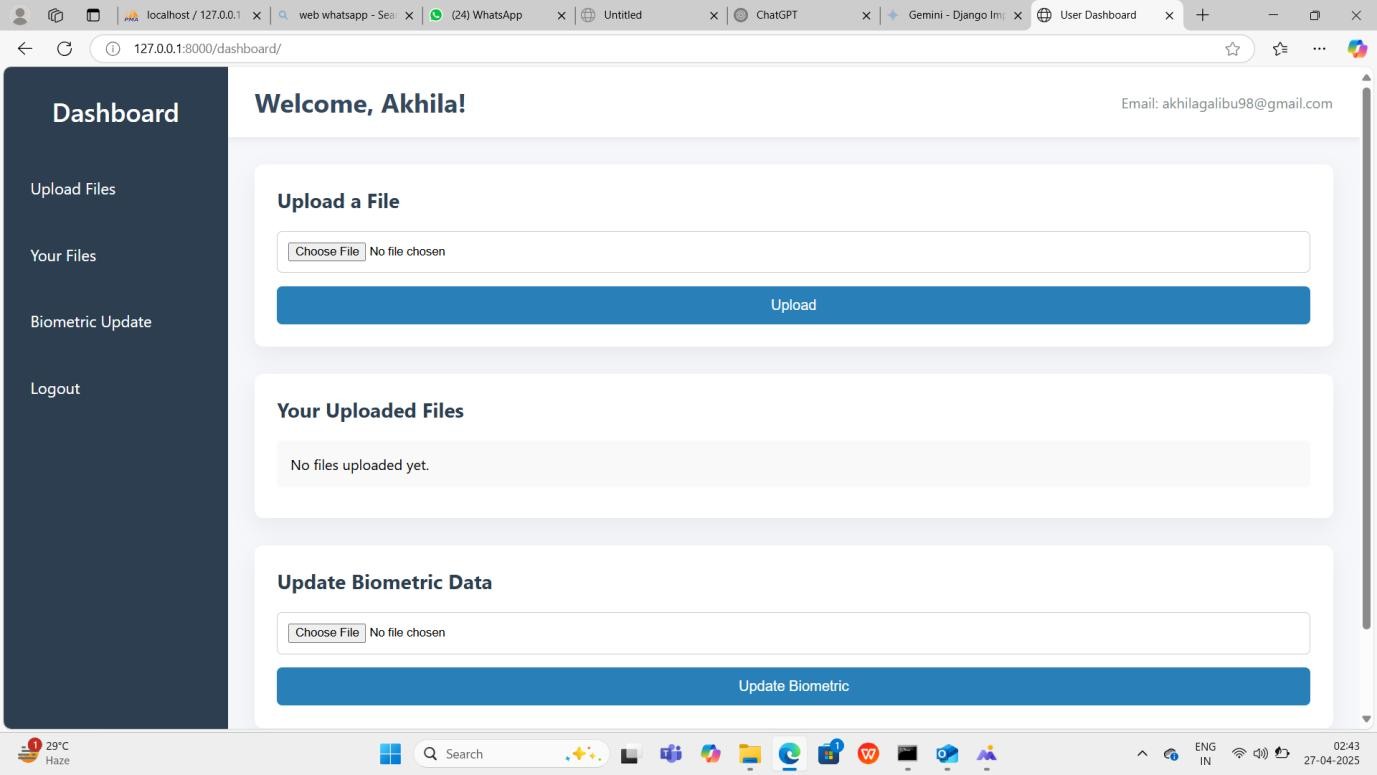
messages.success(request, "Logged out successfully.") return redirect('resource\_server')

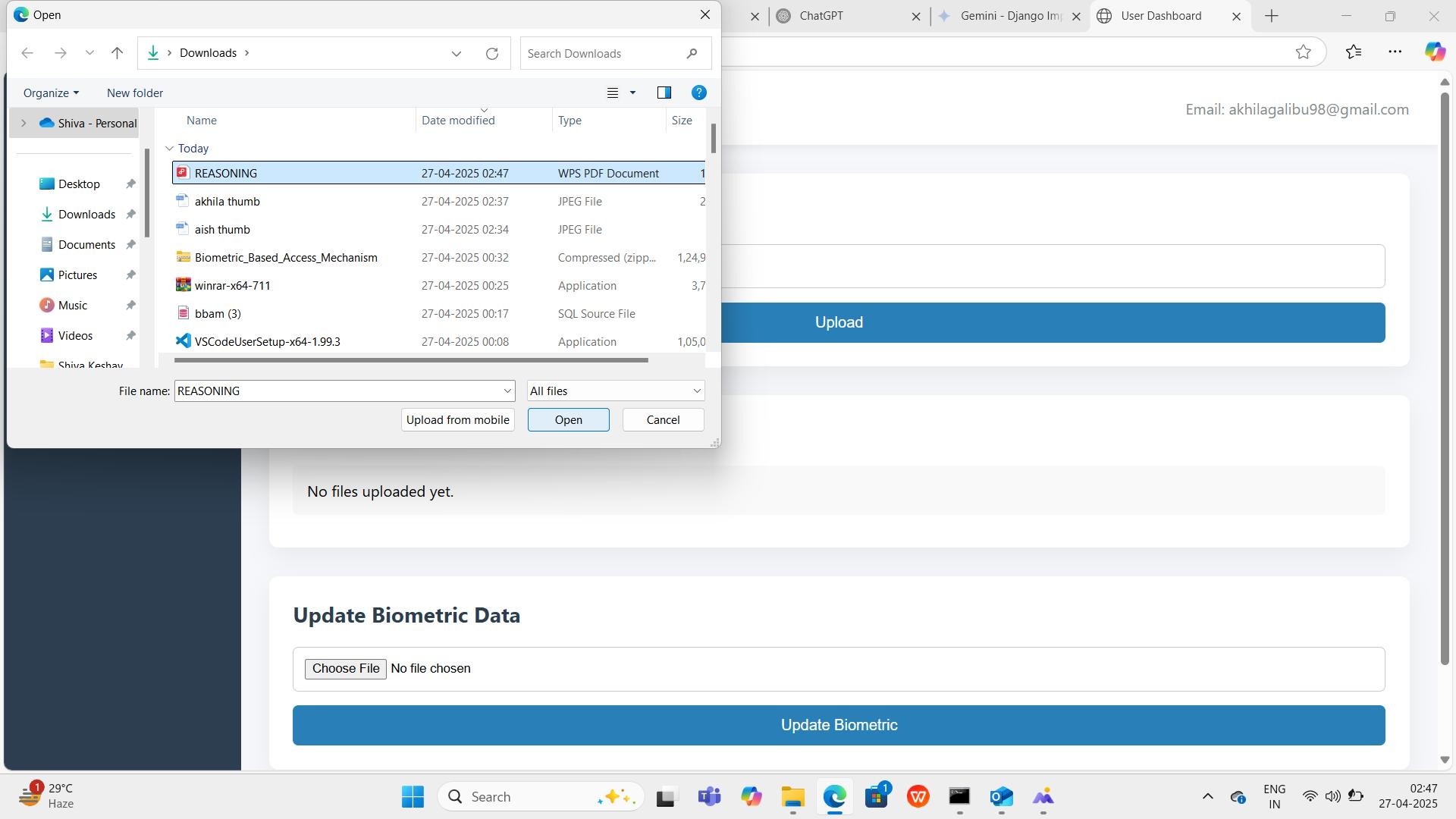
**8.1 OUTPUT SCREENSHOTS:** HOME PAGE

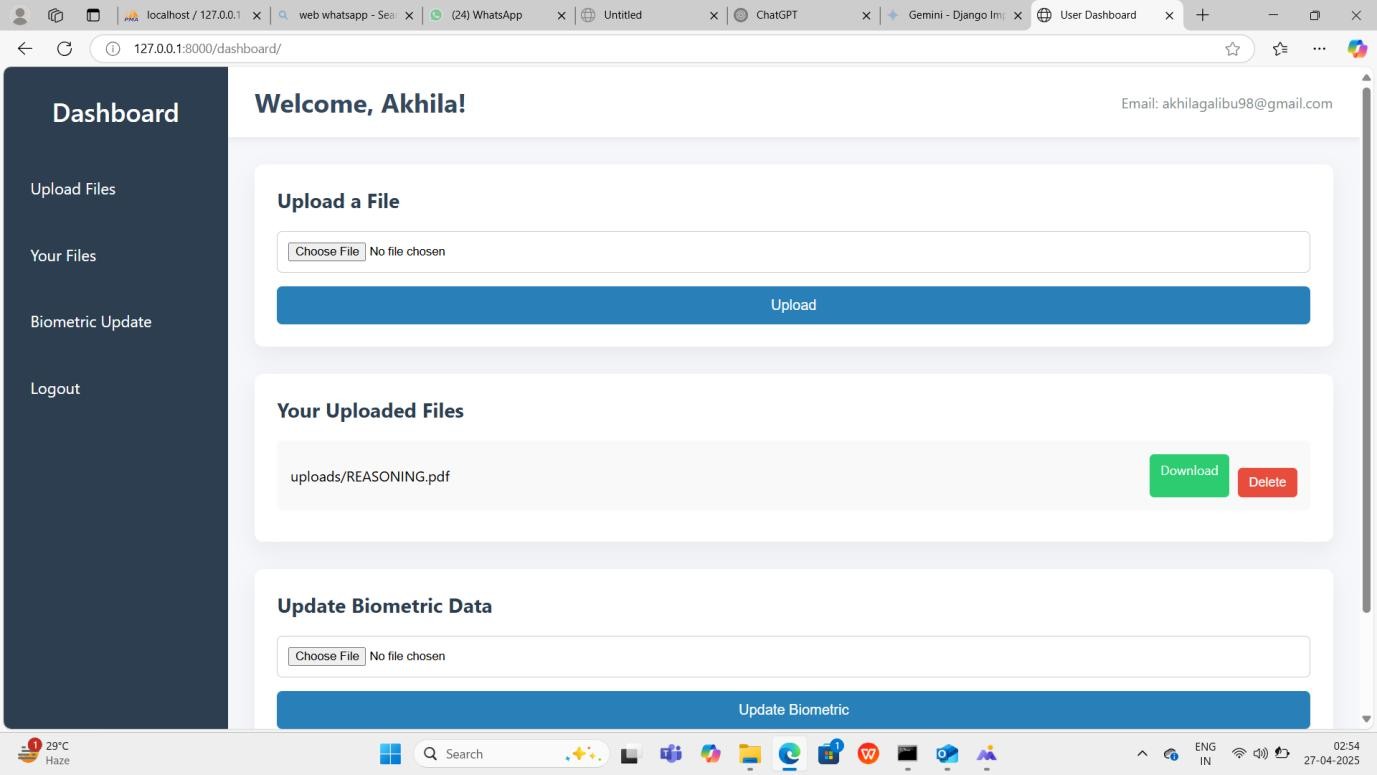


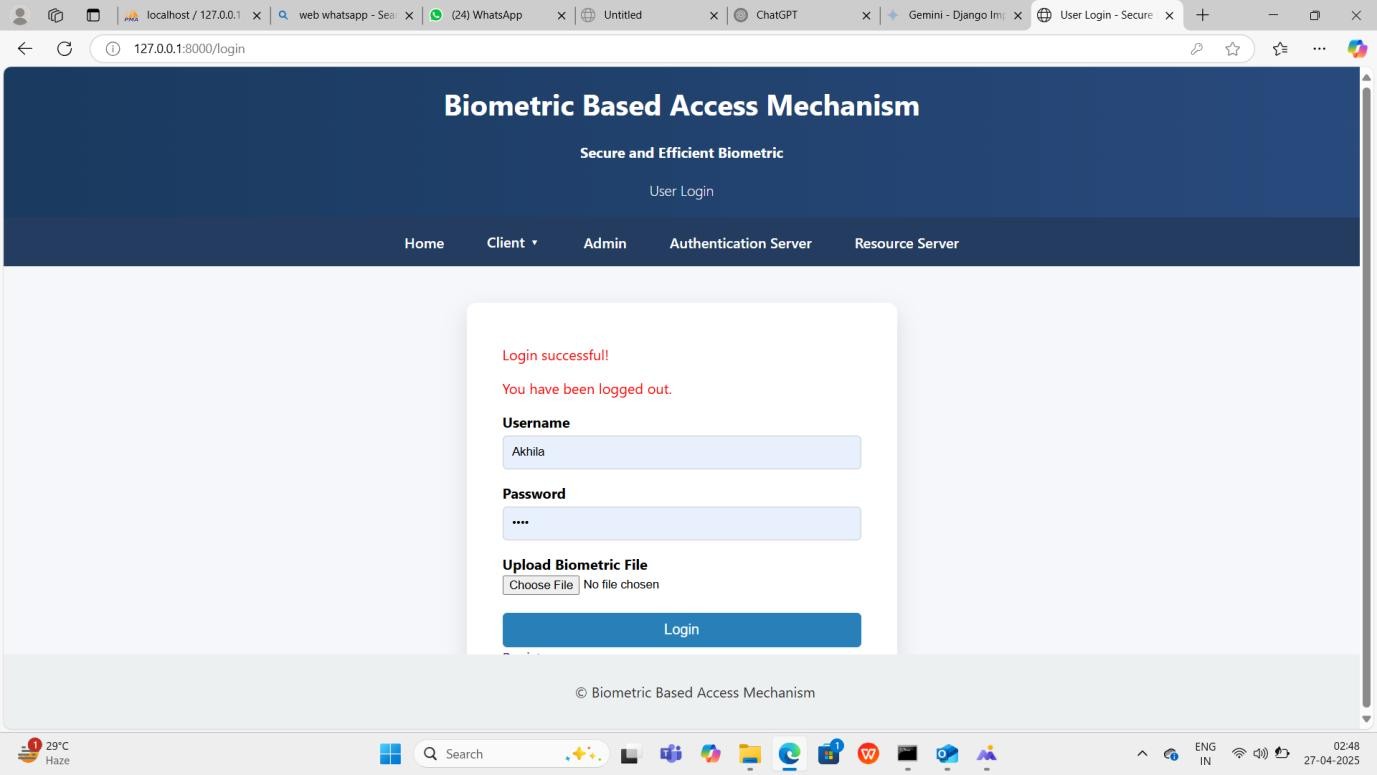


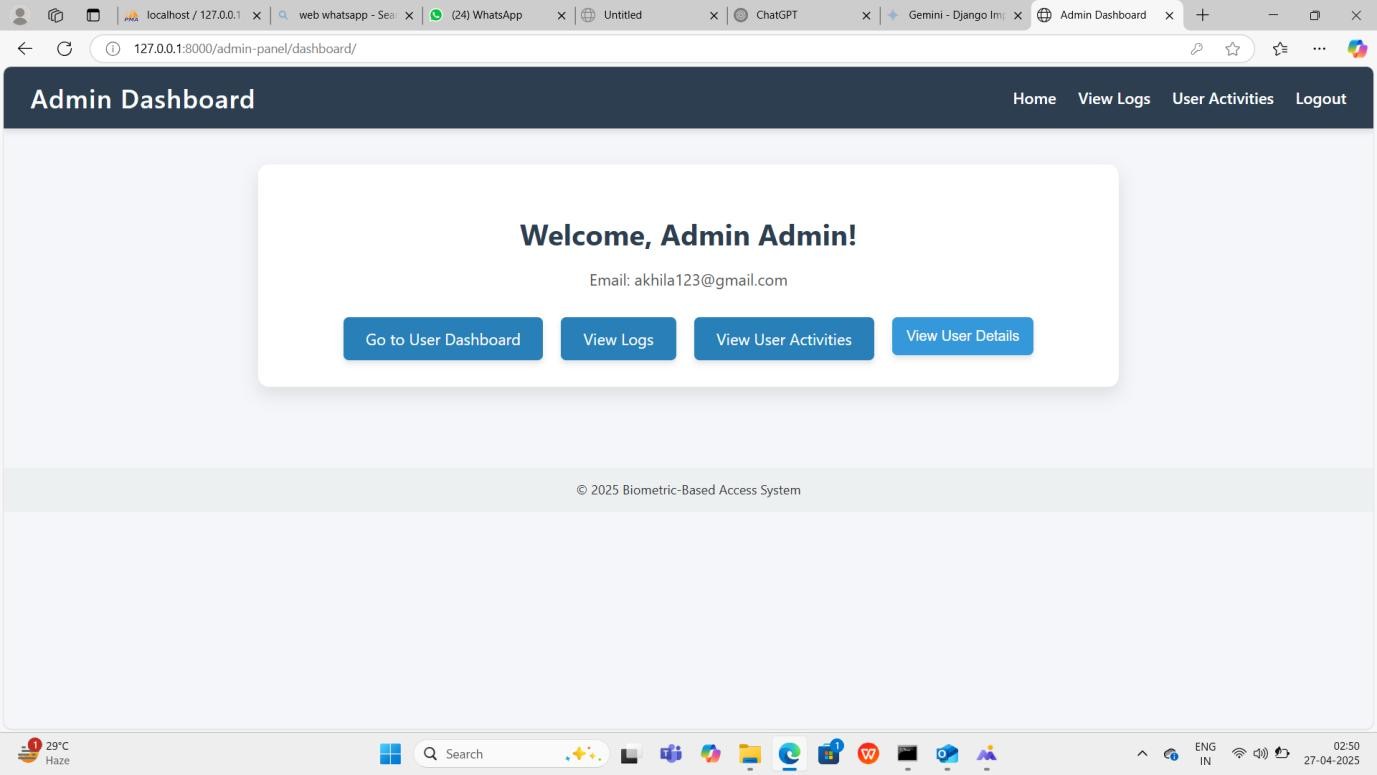


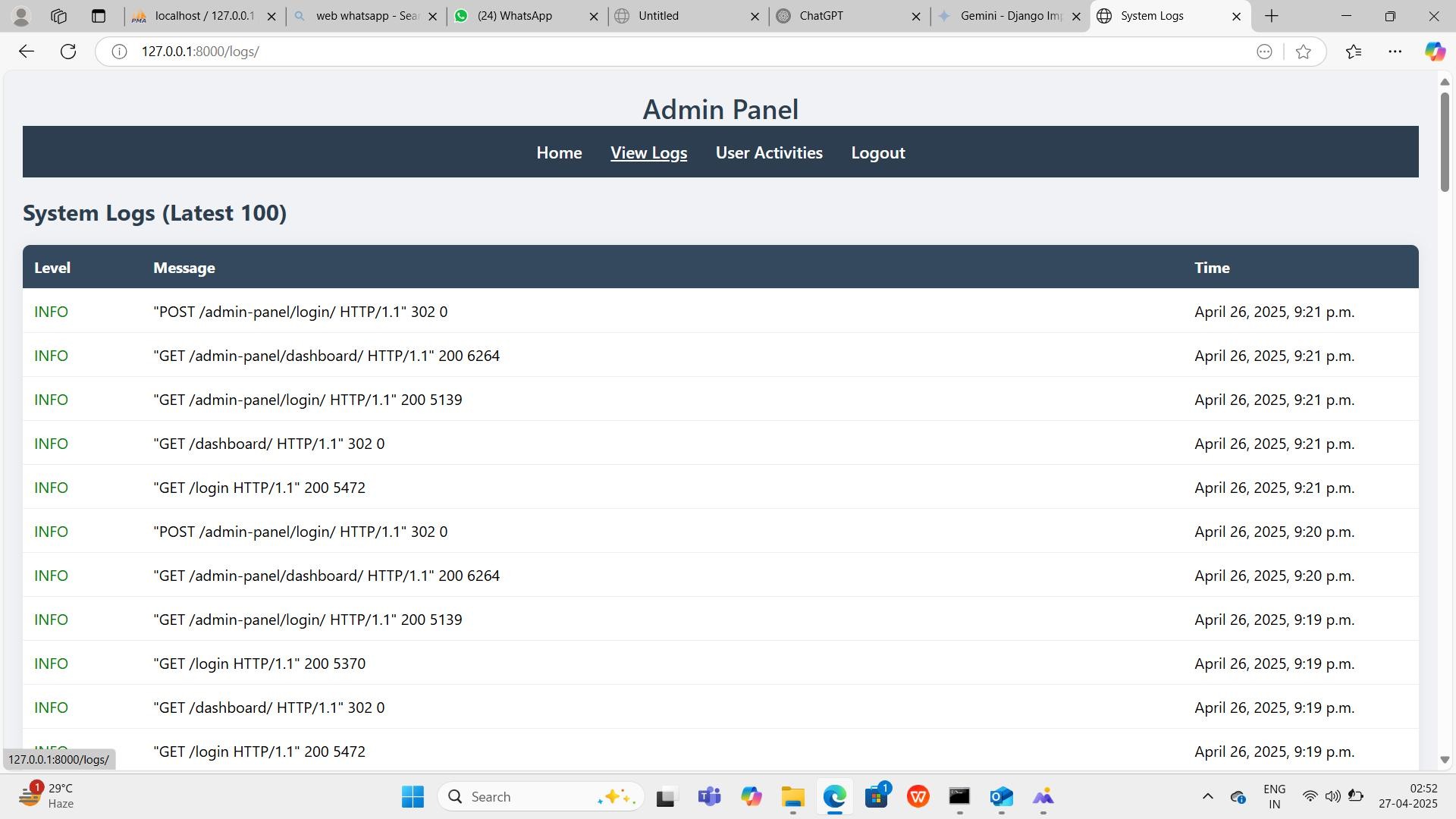


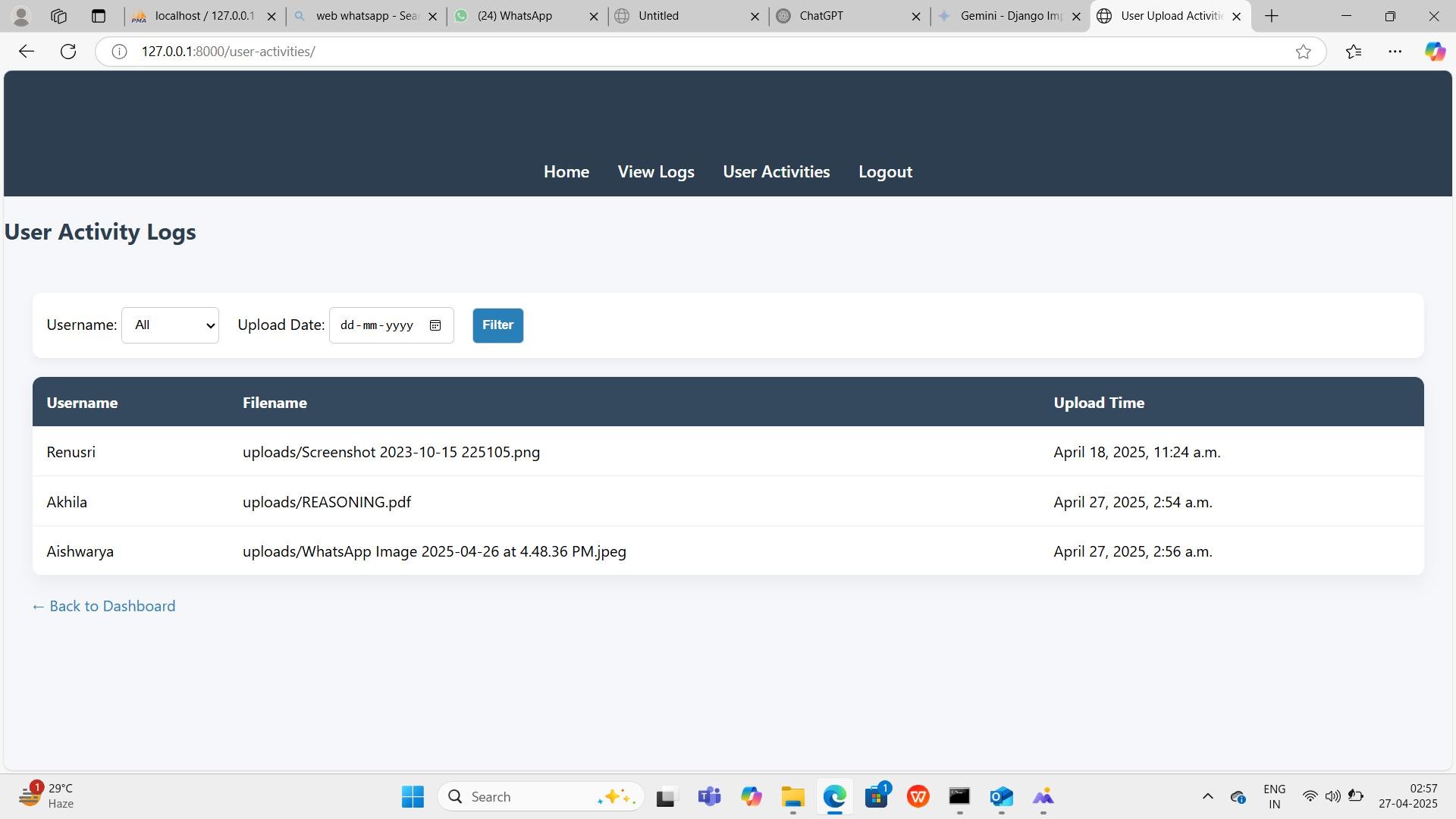


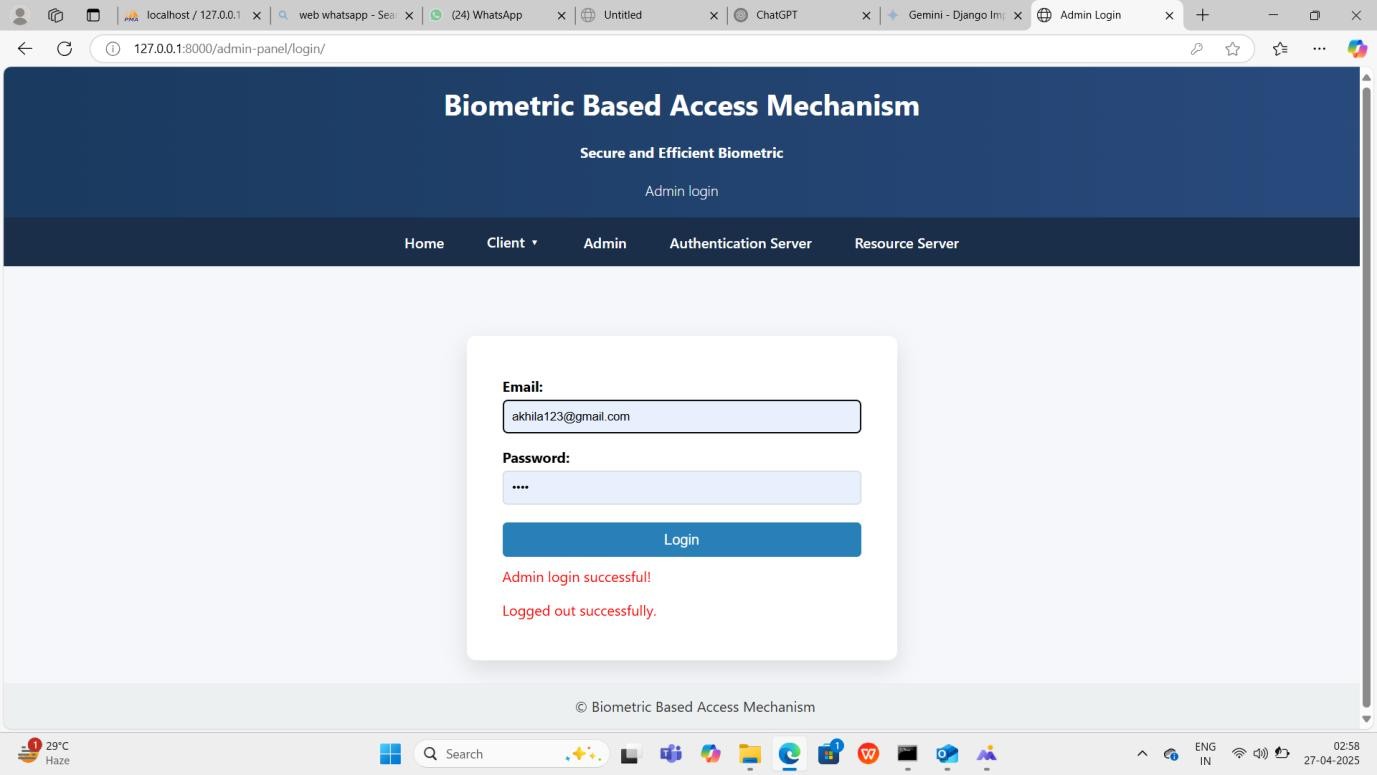


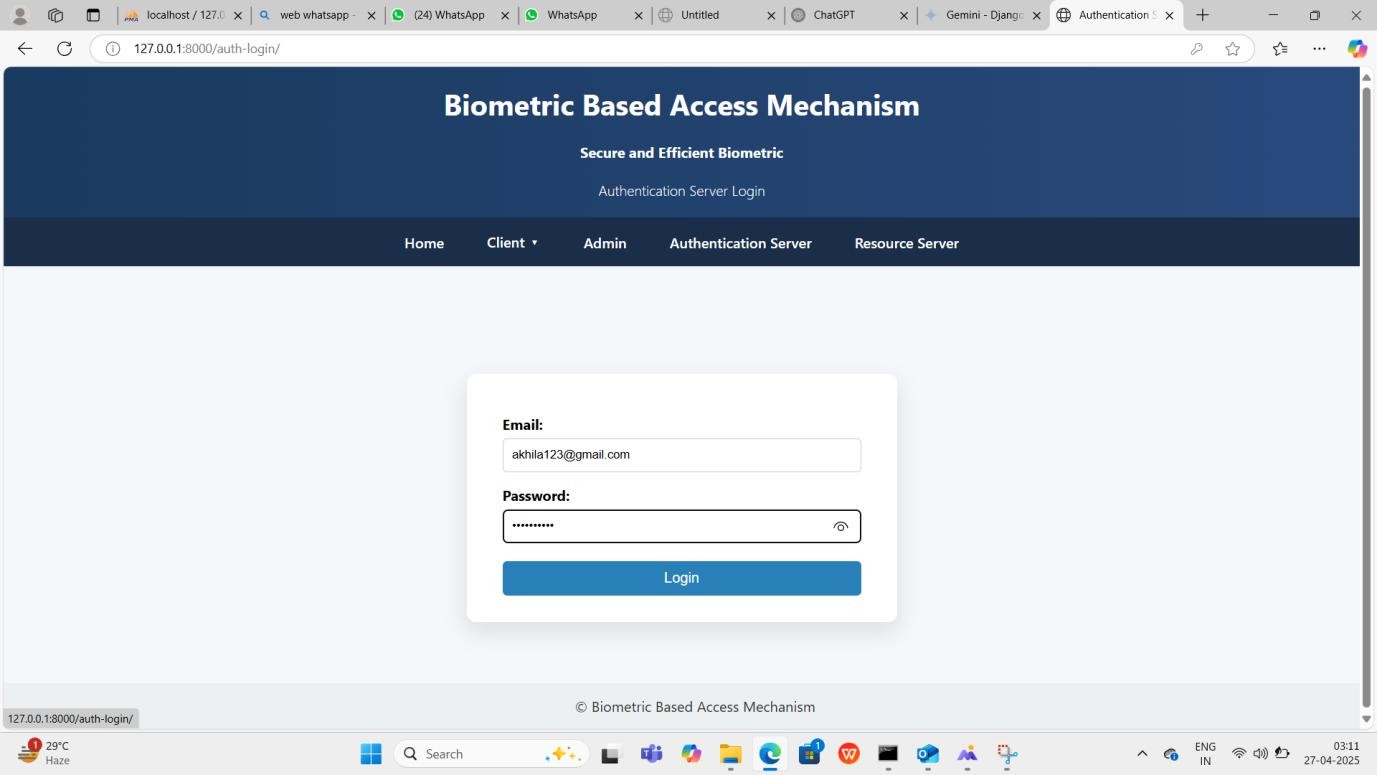


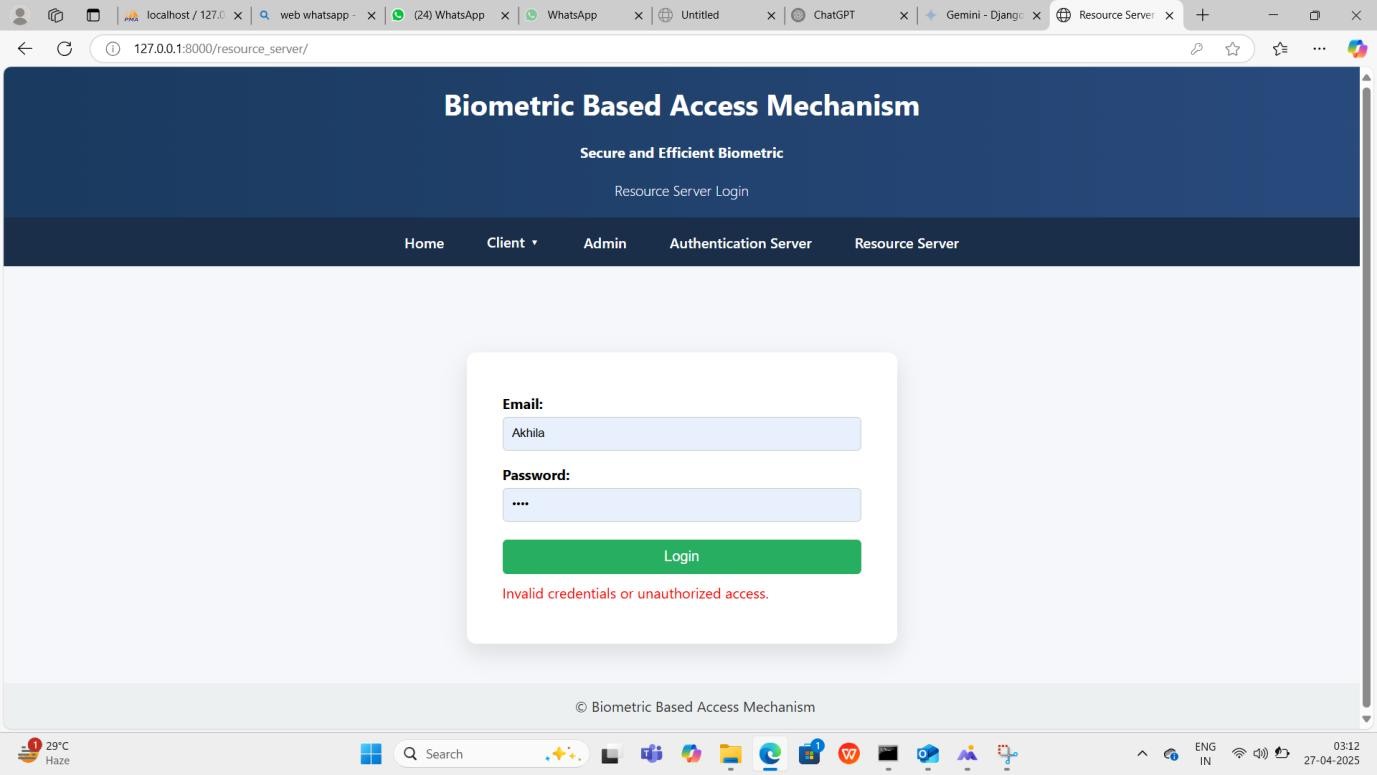












## CHAPTER 9 CONCLUSION

### CONCLUSION

Biometric has its unique advantages over conventional password and token-based security system, as evidenced by its increased adoption (e.g., on Android and iOS devices). In this paper, we introduced a biometric-based mechanism to authenticate a user seeking to access services and computational resources from a remote location. Our proposed approach allows one to generate a private key from a fingerprint biometric reveals, as it is possible to generate the same key from a fingerprint of a user with 95.12% accuracy. Our proposed session key generation approach using two biometric data does not require any prior information to be shared. A comparison of our approach with other similar authentication protocols reveals that our protocol is more resilient to several known attacks. Future research includes exploring other biometric traits and also multi-modal biometrics for other sensitive applications (e.g., in national security matters).

# 10. REFERENCE

1. C. Neuman, S. Hartman, K. Raeburn, “The kerberos network authentication service (v5),” RFC 4120, 2005.
2. “OAuth Protocol.” [Online]. Available: <http://www.oauth.net/>
3. “OpenID Protocol.” [Online]. Available: <http://openid.net/>
4. G. Wettstein, J. Grosen, and E. Rodriguez, “IDFusion: An open architecture for Kerberos based authorization,” Proc. AFS and Kerberos Best Practices Workshop, June 2006.
5. A. Kehne, J. Schonwalder, and H. Langendorfer, “A nonce-based protocol for multiple authentications,” ACM SIGOPS Operating System Review, vol. 26, no. 4, pp. 84–89, 1992.
6. D. Kang, J. Jung, H. Kim, Y. Lee, and D. Won, “Efficient and Secure Biometric-Based User Authenticated Key Agreement Scheme with Anonymity,” Security and Communication Networks, vol. 2018, pp. 1–14, 2018, Article ID 9046064, https://doi.org/10.1155/2018/9046064.