

INTRUDER DETECTION AND EMAIL ALERTING SYSTEM

A PROJECT REPORT

Submitted

In partial fulfillment of the requirements for

the award of the degree of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

by

- 1. 20B01A1226 - CH. SRAVYA**
- 2. 20B01A1225 - CH. BHAVANA SAI**
- 3. 20B01A1231 - CH. BALA PADMAJA PRAVEENA**
- 4. 20B01A1251 - G. SRI PADMA PRANEETHA**

Under the guidance of

Dr. D. Venkata Naga Raju, Ph.D

Professor & Head of the Department

(IT)



**DEPARTMENT OF INFORMATION TECHNOLOGY
SHRI VISHNU ENGINEERING COLLEGE FOR WOMEN
(AUTONOMOUS)**

BHIMAVARAM – 534 202, INDIA

2023-24

SHRI VISHNU ENGINEERING COLLEGE FOR WOMEN
(AUTONOMOUS)
BHIMAVARAM - 534 202
DEPARTMENT OF INFORMATION TECHNOLOGY



DECLARATION

I therefore pronounce that the work depicted in this project report entitled **“Intruder detection and Email Alerting System”** which is being presented by us in partial fulfillment for the award of Degree of **Bachelor of Technology** in the **Department of Information Technology** to Shri Vishnu Engineering College for Women, Bhimavaram, is the consequence of investigations completed by us under the supervision of **Dr.D.Venkata Naga Raju**, Professor and HOD, Department of IT, SVECW.

The work is unique and has not been submitted to a limited extent or full for the honor of any Degree in any other University or Institute.

NAME	REGD NO	SIGNATURE
Ch.Sravya	20B01A1226	
Ch.B.P.Praveena	20B01A1231	
G.S.P.Praneetha	20B01A1251	
Ch.Bhavana Sai	20B01A1225	

Place: Bhimavaram

Date:

**SHRI VISHNU ENGINEERING COLLEGE FOR WOMEN
(AUTONOMOUS)**

BHIMAVARAM – 534 202

DEPARTMENT OF INFORMATION TECHNOLOGY



BONAFIDE CERTIFICATE

This is to certify that the Project Report entitled “**INTRUDER DETECTION AND EMAIL ALERTING SYSTEM**” that is being submitted by “**Ch.Sravya [20B01A1226], Ch.B.P.Praveena [20B01A1231], G.S.P.Praneetha [20B01A1251], Ch.Bhavana Sai [20B01A1225]**” in partial fulfillment for the award of **Bachelor of Technology in Information Technology** to the Shri Vishnu Engineering College for Women, Bhimavaram is a record of bonafide work did by them under my supervision during the academic year 2023 - 24.

To the best of my insight, the work consolidated in this postulation has not been submitted somewhere else for the honor of any degree.

Internal Guide
Dr. D. Venkata Naga Raju
Head of the Department,
Department of IT

Dr. D. Venkata Naga Raju
Head of the Department,
Department of IT

External Examiner

ACKNOWLEDGEMENTS

Behind every achievement lies an unfathomable sea of gratitude to those who activated it, without whom it would never ever come into existence. To them we lay a words of gratitude imprinted within us.

We wish to place our deep sense of gratitude to **Sri. K. V. Vishnu Raju**, Chairman of SVECW for his constant support on our each and every progressive work.

We are thankful to **Dr. G. Srinivasa Rao**, Principal of SVECW, for being a source of inspiration and constant encouragement.

We wish to express our sincere thanks to **Dr. P. Srinivasa Raju**, Vice-Principal of SVECW, for being a source of inspiration and constant encouragement.

We wish to express our sincere thanks to our Head of the Department, IT and our guide **Dr. D. Venkata Naga Raju** for being a source of inspiration and constant encouragement. We wish to thank him for his unflinching devotion and valuable suggestions to complete our main project successfully in time.

Project Associates

Ch.Sravya	[20B01A1226]
Ch.B.P.Praveena	[20B01A1231]
G.S.P.Praneetha	[20B01A1251]
Ch.Bhavana Sai	[20B01A1225]

TABLE OF CONTENTS

ABSTRACT	i
LIST OF FIGURES	ii
LIST OF TABLES	iv
Chapter 1. INTRODUCTION	1
Chapter 2. SYSTEM ANALYSIS.....	4
2.1 Existing System.....	4
2.2 Proposed System	5
2.3 Feasibility Study.....	7
Chapter 3. SYSTEM REQUIREMENT SPECIFICATION	8
3.1 System Specification.....	8
3.2 Requirement Specification.....	11
3.3 Technology	12
3.3.1 Streamlit.....	12
3.3.2 Open CV.....	14
3.3.3 Twilio.....	16
3.3.4 SMTP.....	18
3.3.5 MySQL	20
Chapter 4. SYSTEM DESIGN	25
4.1 Database Design.....	28
4.2 UML Diagrams	29
Chapter 5. SYSTEM TESTING	33
5.1 Introduction.....	33
5.2 Types of Testing.....	34
5.3 Test Cases	36
Chapter 6. LAYOUT DESIGNS	38
Chapter 7. CONCLUSION AND FUTURE WORK.....	47
Chapter 8. REFERENCES	52
Chapter 9. APPENDIX	54
9.1 Sample Code	54

ABSTRACT

With the rising concerns about unauthorized access and intrusions, there is a growing demand for robust security systems. This project aims to enhance security measures by leveraging advanced technologies to detect intruders promptly. The "Intruder Detection and Email Alerting System" is an innovative security solution designed to enhance the protection. The primary objective of this project is to develop a reliable and efficient system capable of detecting unauthorized access and promptly notifying relevant stakeholders through email and SMS alerts.

The system integrates various sensors, including motion detectors and video cameras, to continuously monitor the surroundings in real-time. Using advanced computer vision algorithms, it accurately identifies potential intruders and activates an alert mechanism when a security breach is detected. The integration of Streamlit enables users to visualize and interact with the system's status, configurations, and alerts through a web-based interface.

Through facial recognition algorithms, user identities are verified during login, significantly reducing vulnerabilities associated with conventional password-based methods. Upon successful authentication, users gain access to system functionalities, including image capture and user registration. Intrusion attempts trigger immediate email notifications to system administrators, facilitating prompt response and heightened security vigilance. The paper presents the system's architecture, implementation details, and evaluation metrics, demonstrating its efficacy in safeguarding against unauthorized access.

In conclusion, the "Streamlit-based Intruder Detection and Email Alerting System" represents a sophisticated and accessible approach to enhancing security in various environments. By leveraging Streamlit capabilities for creating interactive web applications, the system provides an intuitive interface for users to monitor, configure, and receive timely alerts about potential security breaches, thereby contributing to a more responsive and user-centric intruder detection solution.

LIST OF FIGURES

Figure No. & Description	Page No
2.1 System Architecture	6
3.1 Depicts how Twilio delivers SMS	18
3.2 Database	21
3.3 Authorized Details	22
4.1 Use case Diagram for User, Admin Interaction with database	29
4.2 Class Diagram of Application	30
4.3 Sequence Diagram of Application	32
6.1 If the user is unauthorized	38
6.2 If the user is authorized, the application allows user for further process.	39
6.3 Unauthorized alert message	40
6.4 Authorized User allowed to the system	41
6.5 No person detected unauthorized	41
6.6 Recognize with or without spectacles	42
6.7 Detects even in the poor lighting conditions.	43
6.8 Add new user	44
6.9 Updated Database	45

6.10 Email alert	46
6.11 SMS alert	46

LIST OF TABLES

Table No & Description	Page No
5.3.1 Test cases for Login Page	36
5.3.2 Test cases for image capturing	37

1.INTRODUCTION

In recent years, the integration of advanced technologies has become imperative for enhancing security measures in various domains. One critical aspect of security is the protection of physical spaces, where the need for efficient intruder detection systems has grown significantly. This research paper presents a comprehensive study and implementation of an "Intruder Detection and Email Alerting System" employing the powerful combination of Streamlit and OpenCV technologies. Intrusion detection systems are pivotal in safeguarding premises against unauthorized access and potential security threats. The conventional methods often fall short in providing real-time monitoring and alerting capabilities, necessitating the exploration of innovative solutions. Leveraging the capabilities of computer vision and user-friendly web application frameworks, this project aims to deliver a robust and accessible system for intruder detection. The utilization of OpenCV, an open-source computer vision library, facilitates the development of sophisticated image and video processing algorithms. This enables the system to analyze live video feeds or recorded footage, identifying potential intruders based on predefined criteria such as motion patterns or object recognition. Simultaneously, the integration of Streamlit, a Python library for creating interactive web applications, ensures a seamless and user-friendly interface for system configuration and monitoring.

System Modules:

- User Interface
- Authentication
- Video Capturing
- Alert messages
- Add User

User Interface:

Upon accessing the login page, the user inputs their username and password into the designated fields. After entering the credentials, the user clicks on the login button to initiate the authentication process. Upon submission, the system verifies the entered credentials against the stored user data. If the credentials match, the user gains access to the system and can proceed to retrieve or interact with the desired information.

Authentication:

Authentication is the process of verifying the validity of the credentials provided by a user. This is done by comparing the provided credentials against the information stored in a database of authorized users on an authentication server. If the credentials match, the process is considered successful, and the user is granted access.

Video Capturing:

Following successful authentication, the system activates video capturing functionality using OpenCV to monitor for potential intruders. OpenCV is utilized to capture real-time video footage from connected cameras or devices. The captured video stream is analyzed in real-time to detect any unauthorized access attempts or suspicious activities. If an intruder is detected, the system can trigger predefined security protocols or alerts for further action. This feature enhances the system's security by providing visual surveillance capabilities alongside authentication mechanisms.

Alert Messages:

The procedure for confirming the legitimacy of an image that a user has submitted. This is accomplished by matching the data in a database of authorized users with the photographs that have been provided. The procedure is deemed successful and access is given to the user if the credentials match. An alert message containing the intruder's image and a Twilio SMS alert will be issued to the authorized account if the credentials are not received.

Add User :

The system provides an "Add User" functionality where authorized users can input the credentials of individuals to be added. Upon submission, the system verifies the provided credentials through authentication. If successful, the individual is considered approved and added to the system. This process ensures that only authorized users can add new individuals to the system. Unauthorized individuals are denied access unless their credentials pass authentication. This functionality enhances system security by controlling access to approved users only.

2. SYSTEM ANALYSIS

System analysis is an important activity that's takes place when we are building a new system or changing the existing one. Analysis helps to understand the existing system and requirements necessary for building the new system. If there is no existing system then analysis defines only the requirements.

One of the most important factors in the system analysis is to understand the system and its problems. A good understanding of the system enables designer to identify and correct problems. Based on the drawbacks of existing system is being planned. So the total definition of the given problem has been analyzed.

2.1 Existing System

- The previous Intrusions detection systems were mostly developed using Snort framework.
- If in case any intruder is detected, the owner could only get information about the intruder being spotted and he has to manually find who it is.

Drawbacks

- The system is unable to distinguish between human beings and another moving objects.
- The owner would be able to get alerts only about intruder getting spotted, but they cannot get the details about the intruder.
- The system is inefficient to identify the user if the surrounding conditions are not ideal.

2.2 Proposed System

- This application we have developed is a secure and robust Intruder detection and email alerting system.
- The current applications has friendly user interface and performs two layers of filtration to detect and eradicate the intruder.
- Email alerts will be sent to the owner of the application along with the image of the intruder, thereby aiding the owner to find who tries to sneak into the system.
- An SMS alert will also be sent to the owner along with the Email alerts to aid in case of internet failure.
- The access to use the application will be provided to the user only when he is proved to be an authorized person.

Advantages

- The application could be able to detect human beings and performs face comparisons to detect intruders.
- Saves a lot of efforts of the owner by sending the image of the intruder through email alert.
- Aids in the case of internet failure by sending SMS alert.
- Can work very efficiently even in the case of non-ideal surroundings.

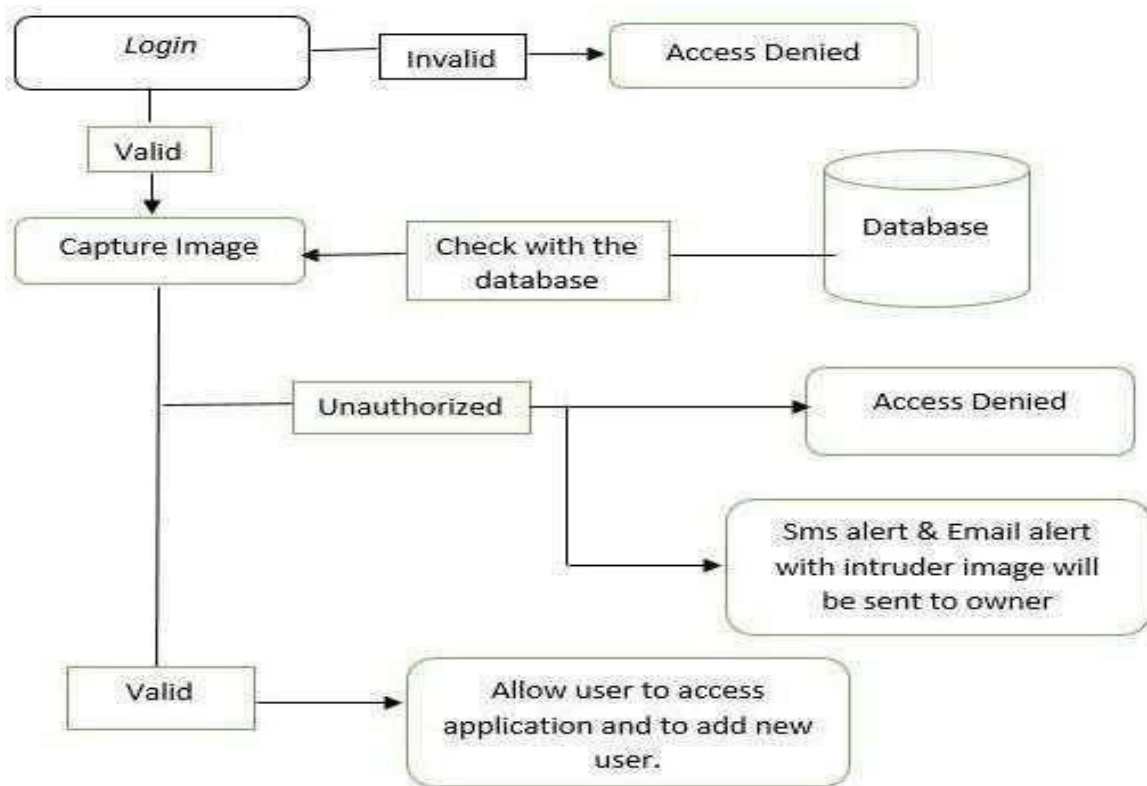


Fig 2.1 System Architecture

The System Architecture is represented in fig 2.1. Initially user tries to login if the credentials are given wrong the access will be denied, if the credentials are valid it captures the image the image captured will be verified with the available data in Database if the person image is valid they can access through the account and can add new users if the person image is invalid an email and SMS alert will be sent along with the intruder image.

2.3 Feasibility Study :

It defines whether an application is technically, operationally and economically feasible.

Types of Feasibilities

1. Technical Feasibility
2. Operational Feasibility
3. Economical Feasibility

Technical Feasibility:

The system is technically feasible as we have implemented this application using technologies like Streamlit, Open CV and twilio which have many inbuilt features for developing an application.

Operational Feasibility:

The system is operationally feasible as we can access this application from any location irrespective of place and time.

Economical Feasibility:

The system is economically feasible as it is developed using open sources and free tools like Streamlit, Open CV and twilio which doesn't require any maintenance cost.

3.SYSTEM REQUIREMENT SPECIFICATION

3.1 System Specification:

We wanted to develop an application that eradicates intruders and provide safe, secure and robust environment for the user. With the zeal to implement the same, we made out a plan. Then, we discussed with the team and total system is analyzed and detailed report of the requirements is finalized.

Modules

1. User Login
2. Live Capturing
3. Image Comparison
4. Sending Alerts
5. Add New User

1. User Interface

- In order to make the application user friendly, we have integrated our python application with web. It helps user to easily interact with the application.

2. User Login

- Only authorized user has access
 - Username
 - Password

Functionality:

The authorized person determined by the management can only be able to login using his credentials. User login is the first layer of intruder filtration. If the user credentials are invalid, the user cannot be allowed into the system.

3. Live Capturing

- The live camera captures the user image.

Functionality:

Once the user enters the valid login credentials, then the live camera turns on and captures the image of the user. This step is done to aid intruder detection.

4. Image Comparison

- The captured image is compared with the ones of the authorized people.
- Can be able to work efficient when the surroundings are not ideal.

Functionality:

The application compares the captured image with the images already stored in the database and determine whether the person is an intruder or an authorized person. The application can adjust the surroundings of the user even if they are not ideal and can derive perfect results.

5. Sending Alerts

- Email alert with intruder image will be sent
- SMS alert will also be sent.
- Further access denied.

Functionality:

If the user is recognized as an intruder, then an email alert with the image of the intruder will be sent to the owner. Apart from that, an SMS alert will also be sent to the owner, as an additional aid in the case of internet failure. Further access will be denied for the intruder.

6. Add New User

- Only authorized user has access
 - Username
 - Password
 - E-mail
 - Image

Functionality:

If the user is recognized as authorized, then he can access the application and can also be able to add the new user, whom he considers as safe option.

3.2 Requirement Specification

Purpose:

The main purpose for preparing this document is to give a general insight into the analysis and requirements of the existing system or situation for determining the operational characteristics of the system.

Scope:

This document plays a vital role in the software development life cycle (SDLC) as it describes the complete requirements of the system. It is meant for use by the developers and will be the basis for testing the system. Any changes made to the requirements in the future will have to go through formal change approval process.

Software and Hardware Specifications

➤ **Software Requirements**

- Frontend : Streamlit
- Database : MySQL
- IDE : Visual Studio Code
- Programming Language : Python
- Operating System : Windows

➤ **Hardware Requirements**

- 8GB RAM
- Camera

3.3 Technologies used

The technologies we used in our application are Streamlit, Open CV, Twilio, SMTP, MySQL

1. Streamlit:

Streamlit is an open-source Python library that allows you to create web applications for machine learning and data science projects quickly and easily. It simplifies the process of building interactive web apps by providing a simple and intuitive way to create user interfaces directly from Python scripts.

Key features and aspects of Streamlit include:

- **Easy-to-use:** Streamlit provides a clean and simple API that allows developers to create interactive web apps using familiar Python scripting. It follows a "Python-first" philosophy, meaning that you can build applications entirely using Python code without needing to write HTML, CSS, or JavaScript.
- **Rapid development:** With Streamlit, you can build and iterate on web applications quickly, making it ideal for prototyping, experimentation, and showcasing data science projects.
- **Built-in widgets:** Streamlit offers a variety of built-in widgets such as sliders, buttons, checkboxes, and text inputs, which enable users to

interact with data and models in real-time. These widgets can be easily integrated into your Python scripts with minimal code.

- **Data visualization:** Streamlit supports seamless integration with popular Python libraries for data visualization such as Matplotlib, Plotly, and Altair, allowing you to create interactive charts, graphs, and plots within your web applications.
- **Custom components:** While Streamlit provides a wide range of built-in components, it also allows you to create custom components and extensions to tailor your applications to specific use cases.
- **Sharing and deployment:** Streamlit applications can be easily shared and deployed on various platforms including Streamlit Sharing, Heroku, AWS, and others. This makes it straightforward to share your data science projects with colleagues, stakeholders, or the broader community.
- **Community and ecosystem:** Streamlit has a vibrant and active community of users and contributors who share tips, examples, and extensions through forums, GitHub, and other channels. This community-driven approach fosters collaboration and innovation within the Streamlit ecosystem.

Overall, Streamlit is a powerful tool for building interactive web applications for data science and machine learning projects, offering simplicity, flexibility, and rapid development capabilities.

2. Open CV:

OpenCV, short for Open Source Computer Vision Library, is an open-source computer vision and machine learning software library. Originally developed by Intel, it is now maintained by a community of developers and is widely used in various fields such as robotics, augmented reality, medical image analysis, and more. OpenCV is written in C++ and has interfaces for Python, Java, and MATLAB/Octave, making it accessible to a broad audience of developers.

Key features and capabilities of OpenCV include:

- **Image Processing:** OpenCV provides a wide range of functions for basic and advanced image processing tasks such as filtering, edge detection, image enhancement, color manipulation, and morphological operations.
- **Computer Vision Algorithms:** OpenCV includes implementations of many computer vision algorithms such as feature detection (e.g., SIFT, SURF), object detection (e.g., Haar cascades, HOG), object tracking, optical flow, and camera calibration.
- **Machine Learning:** OpenCV integrates with machine learning libraries like TensorFlow and PyTorch, enabling the development and deployment of machine learning models for tasks such as image classification, object detection, and semantic segmentation.

- **Camera Calibration and Geometry:** OpenCV provides tools for camera calibration, stereo vision, and geometric transformations, which are essential for tasks like 3D reconstruction, depth estimation, and camera pose estimation.
- **Video Processing:** OpenCV supports video I/O operations, allowing developers to capture video streams from cameras, process video frames in real-time, and save processed videos.
- **Cross-platform Compatibility:** OpenCV is compatible with various operating systems including Windows, Linux, macOS, Android, and iOS, making it suitable for developing applications across different platforms.
- **Community and Documentation:** OpenCV has a large and active community of developers who contribute to its development and provide support through forums, mailing lists, and other channels. Additionally, OpenCV provides extensive documentation, tutorials, and examples to help developers get started with using the library.

Overall, OpenCV is a powerful and versatile library for computer vision and image processing tasks, offering a comprehensive set of features and algorithms for both beginners and experienced developers. It is widely used in academia, industry, and research for a wide range of applications in computer vision and beyond.

3. Twilio:

Twilio is a cloud communications platform that provides APIs (Application Programming Interfaces) for developers to programmatically integrate voice, messaging, and video capabilities into their applications. Founded in 2008, Twilio has become a popular choice for building communication features into web and mobile applications due to its ease of use, scalability, and flexibility.

Here are some key components and features of Twilio:

- **Voice API:** Twilio's Voice API allows developers to make and receive phone calls programmatically. Developers can use the API to initiate calls, play audio files, record conversations, and implement interactive voice response (IVR) systems.
- **Messaging API:** Twilio's Messaging API enables developers to send and receive SMS (Short Message Service) and MMS (Multimedia Messaging Service) messages. This API can be used to send notifications, alerts, reminders, and two-factor authentication codes, among other use cases.
- **Video API:** Twilio's Video API allows developers to embed real-time video communication capabilities into their applications. Developers can create video chat applications, video conferencing solutions, and live streaming platforms using this API.

- **WhatsApp API:** Twilio offers an API for integrating with WhatsApp, allowing developers to send and receive messages on the popular messaging platform. This enables businesses to engage with customers on WhatsApp using automated messages and chatbots.
- **Authentication and Authorization:** Twilio provides authentication mechanisms, such as API keys and access tokens, to secure communication between applications and the Twilio platform. Developers can control access to Twilio resources and monitor usage through the Twilio Console.
- **Webhooks and Events:** Twilio supports webhooks, which are HTTP callbacks triggered by events such as incoming calls, messages, or status updates. Developers can use webhooks to handle incoming communication and respond dynamically based on user interactions.
- **Integration with other services:** Twilio integrates with various third-party services and platforms, including popular cloud providers, CRM (Customer Relationship Management) systems, and marketing automation tools. This allows developers to leverage Twilio's communication capabilities within their existing workflows and applications.

Overall, Twilio empowers developers to build powerful communication features into their applications without having to manage complex infrastructure or telecommunications infrastructure. It has a comprehensive set

of APIs, robust documentation, and developer-friendly tools, making it a preferred choice for businesses and developers looking to enhance their applications with voice, messaging, and video capabilities.



Fig 3.1 depicts how twilio delivers SMS.

4. SMTP:

SMTP stands for Simple Mail Transfer Protocol. It is a communication protocol used for sending email messages between servers. SMTP is a crucial component of the email delivery process, facilitating the transmission of messages from the sender's email client or server to the recipient's email server.

Here's how SMTP works:

- **Client-Server Communication:** SMTP operates on a client-server model, where an email client (such as Microsoft Outlook, Gmail, or Thunderbird) acts as the client, and an email server (SMTP server) acts as the server.
- **Message Composition:** When a user composes an email using their email client, the client software communicates with the SMTP server to

send the message. The user specifies the recipient's email address, subject, message body, and any attachments.

- **SMTP Handshake:** The email client establishes a connection with the SMTP server using TCP (Transmission Control Protocol) on port 25 (default for SMTP). This connection is known as an SMTP session.
- **Message Transfer:** Once the connection is established, the email client sends the email message to the SMTP server. The SMTP server processes the message and forwards it to the recipient's email server.
- **Relaying:** If the recipient's email server is different from the sender's SMTP server (as is often the case), the sender's SMTP server relays the message to the recipient's SMTP server. This involves querying DNS (Domain Name System) servers to determine the recipient's mail exchanger (MX) record, which specifies the server responsible for receiving emails for the recipient's domain.
- **Delivery to Recipient:** The recipient's SMTP server receives the email message and stores it in the recipient's mailbox. The recipient can then access the message using their email client or webmail interface.
- **Error Handling and Notifications:** During the SMTP transaction, error messages may be generated if there are issues with the delivery process (e.g., invalid recipient address, server unavailable). These error messages are communicated between SMTP servers, allowing senders to be notified of delivery failures or delays.

SMTP is a standardized protocol defined in RFC 5321 and subsequent RFCs. While port 25 is the default port for SMTP communication, secure variants such as SMTPS (SMTP Secure) and SMTP over TLS (Transport Layer Security) are also commonly used to encrypt email transmissions and enhance security.

5. MySQL:

MySQL is an open-source relational database management system (RDBMS) that is widely used for managing structured data. Developed by MySQL AB, which was later acquired by Oracle Corporation, MySQL is known for its reliability, performance, and ease of use. It is a popular choice for web applications and is often used in conjunction with programming languages like PHP, Python, and Java.

Key features of MySQL include:

- **Relational Database:** MySQL follows the relational model and organizes data into tables with rows and columns. It supports SQL (Structured Query Language) for querying and manipulating data.

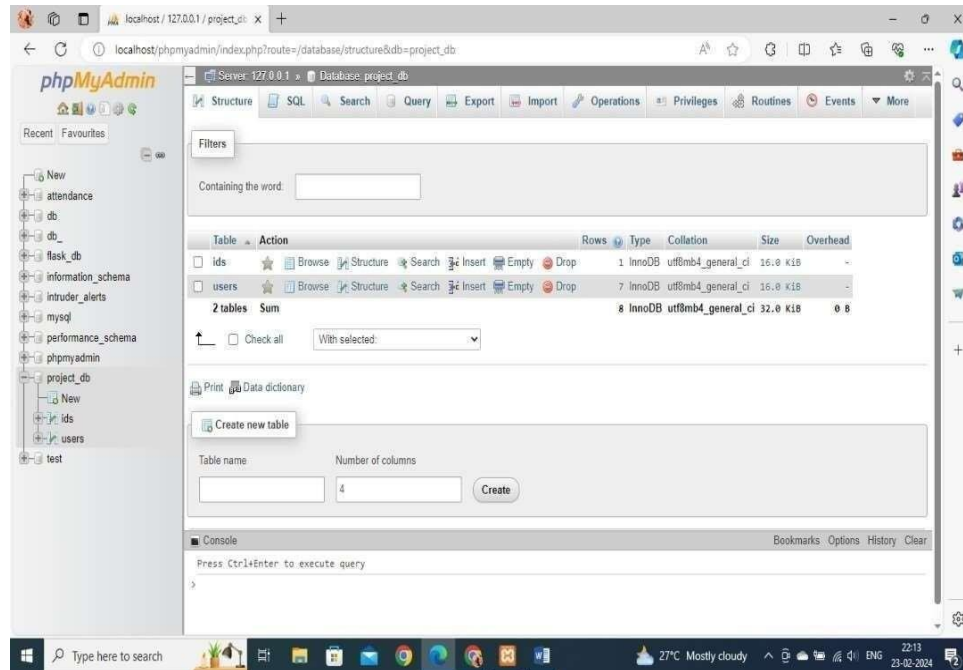


Fig 3.2 Database.

This is the structure of MySQL database we have implemented in our project. The MySQL database stores user credentials for authentication. It manages user accounts and associated images. The application connects to MySQL for data retrieval and storage. User authentication and management are facilitated through MySQL queries and interactions.

Intruder Detection and Email Alerting System

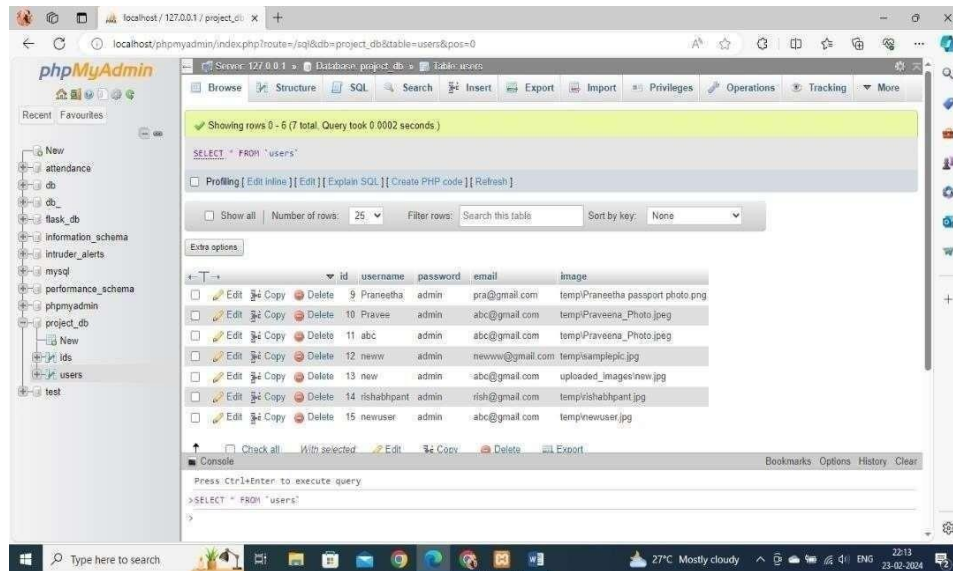


Fig 3.3 Authorized Details

User data is stored in a MySQL table named users. Each row represents a user with columns like id, username, password, email, and image. When a new user registers, their details are inserted into this table. During login, the application retrieves user information from the table to authenticate the user.

- **Scalability:** MySQL is designed to handle both small-scale and large-scale databases. It supports various storage engines, such as InnoDB, MyISAM, and Memory, each optimized for different use cases.
- **High Performance:** MySQL is known for its high performance, with features like indexing, caching, and query optimization to ensure efficient data retrieval and processing.
- **Replication and Clustering:** MySQL supports replication, allowing data to be copied and synchronized across multiple servers for redundancy and scalability. It also supports clustering for high availability and load balancing.
- **Security:** MySQL provides robust security features, including user authentication, access control, and data encryption, to protect sensitive information stored in the database.
- **Cross-Platform Compatibility:** MySQL is compatible with various operating systems, including Linux, Windows, macOS, and Unix-like systems, making it suitable for deployment in diverse environments.
- **Community and Support:** MySQL has a large and active community of users and developers who contribute to its development, provide support, and share resources like tutorials, forums, and documentation.

Overall, MySQL is a powerful and versatile database management system that is widely used in the industry for building scalable and reliable web applications, e-commerce platforms, content management systems, and more. Its combination of performance, scalability, and ease of use makes it a popular choice for developers and organizations of all sizes.

4.SYSTEM DESIGN

Introduction:

Software design is a process of problem solving and planning for software solutions after the purpose and specification of software are determined, software developers will design or employ designers to develop a plan for a solution. It includes low level component and algorithm implementation issues as well as the architectural view.

Software Design Consideration

There are many aspects to consider in the design of software.

Compatibility:

Ensure platform-agnostic libraries and standardized APIs. Utilize web technologies like HTML, CSS, and JavaScript for cross-platform compatibility. Employ virtualization and containerization for abstracting platform dependencies. Adopt cloud-native principles for scalability and flexibility. Thoroughly test across different platforms to identify compatibility issues early. Embrace industry standards and widely accepted protocols for interoperability. Design modular architectures to accommodate diverse environments. Utilize automated testing tools and continuous integration pipelines. Leverage Docker and Kubernetes for container orchestration. Prioritize compatibility in each stage of software development.

Extensibility:

Design modular components with clear interfaces to facilitate easy integration of new features. Implement a plugin system allowing seamless addition or removal of functionalities. Use design patterns like the decorator pattern to dynamically extend functionality. Prioritize abstraction and loose coupling to prevent ripple effects on existing architecture when introducing new features.

Fault-Tolerance:

Implement robust error handling mechanisms to gracefully manage failures. Utilize redundancy and replication strategies to ensure continuous operation. Employ techniques like circuit breakers and retries to mitigate transient failures. Monitor system health in real-time and automatically initiate recovery procedures when anomalies are detected.

Maintainability:

The application is designed with processes and tools to facilitate swift restoration to predefined states. Robust backup and recovery mechanisms ensure rapid recovery from unexpected incidents or errors. Maintenance procedures are streamlined to minimize downtime and swiftly address any deviations from the specified condition. Through proactive monitoring and efficient workflows, the application sustains a high level of maintainability by swiftly rectifying deviations within specified timeframes.

Modularity:

The application is structured into distinct, self-contained modules, each with a clearly defined purpose. These modular components enable easier management, updates, and troubleshooting, enhancing maintainability. By isolating functionalities into separate modules, the application promotes code reuse and scalability. The modular design facilitates efficient collaboration among development teams and reduces the risk of system-wide failures.

Reliability:

The application consistently delivers the intended functionality as per defined criteria and conditions. It maintains operational stability and performance over a predetermined duration without deviation. Users can rely on the application to fulfill its designated tasks without unexpected interruptions. Through rigorous testing and monitoring, the application ensures sustained functionality within defined timeframes.

Security:

The application employs robust security measures to resist hostile actions and external influences. It implements stringent protocols to safeguard against unauthorized access and malicious attacks. Continuous monitoring and updates ensure the application's resilience to evolving threats. With a focus on proactive risk mitigation, the application maintains a high level of security posture.

4.1 Database Design:

The design starts with the end user view of the organization called conceptual requirements and user in a decision-making, which uses information obtained by accessing the database. The end users provide data to be stored in the database.

Table Structure

➤ Users

- Id
- Username
- Email
- password
- image

➤ Constraints

Primary Key: Id

4.2 UML Diagrams:

The unified Modeling Language(UML)I is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software intensive system. The UML gives us a standard way to write a systems blueprints, covering conceptual things, such as business processes and system functions, as well as classes written in a specific programming language, database schemas, and reusable software components.

1. Use Case Diagram:

Use Case diagrams are one of the five diagrams in UML for modeling the dynamic aspects of systems. Use Case diagrams are central to modeling the behavior of a system, a subsystem, or a class. Actors are external entities that interact with the system. Examples of actors include users like client, administrator etc, or another system like central database.

The overview of the whole project through use case diagram is as follows:

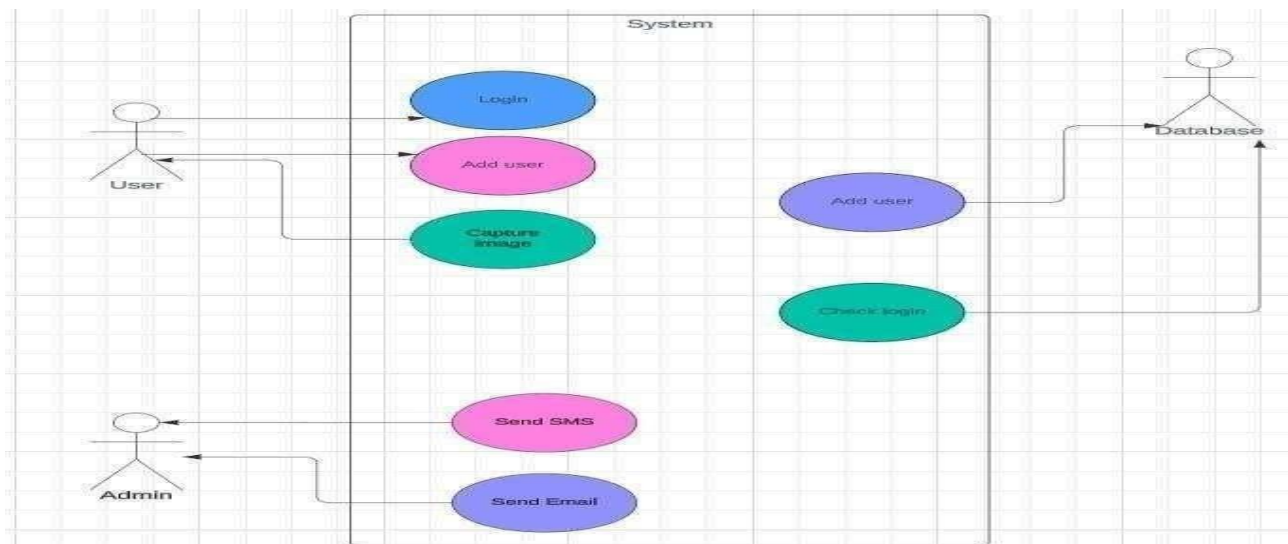


Fig 4.1 Use case Diagram for User, Admin interaction with Database

This system allows users to send SMS messages and capture images. After logging in, users can utilize these functionalities. Captured images are stored securely within the system's database. Additionally, an admin has the ability to manage user accounts, including adding new users and reviewing login information.

2. Class Diagram:

A class diagram shows a set of classes, interfaces, and collaborations and their relationships. Class diagrams are used to illustrate the static design view of a system.

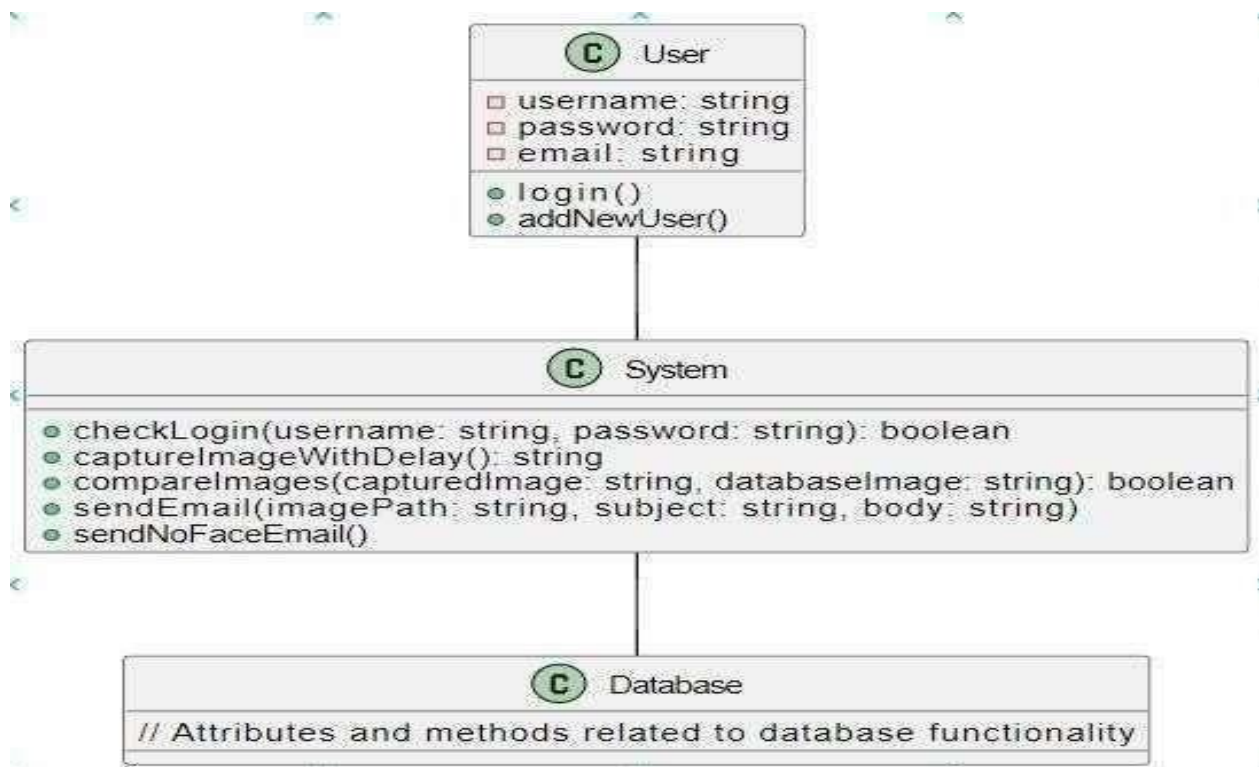


Fig 4.2 Class Diagram of Application

Users are required to input their username and password to initiate access to the system. This data undergoes validation against an authorized user database. Upon successful verification, access is granted for the user to proceed. Conversely, incorrect credentials lead to access denial accompanied by an error message. In certain scenarios, additional verification steps, such as providing a security code, may be necessary to ensure heightened security prior to access authorization.

3.Sequence Diagram:

Sequence diagram is an interaction diagram that emphasizes the time ordering of messages. Graphically, a sequence diagram is a table that shows objects arranged along X axis and messages, ordered in increasing time along the Y axis. It contains the object life line that represents the existence of an object over a period of time. There is a focus of control that shows the period of time during which an object is performing an action, either directly or through a subordinate procedure.

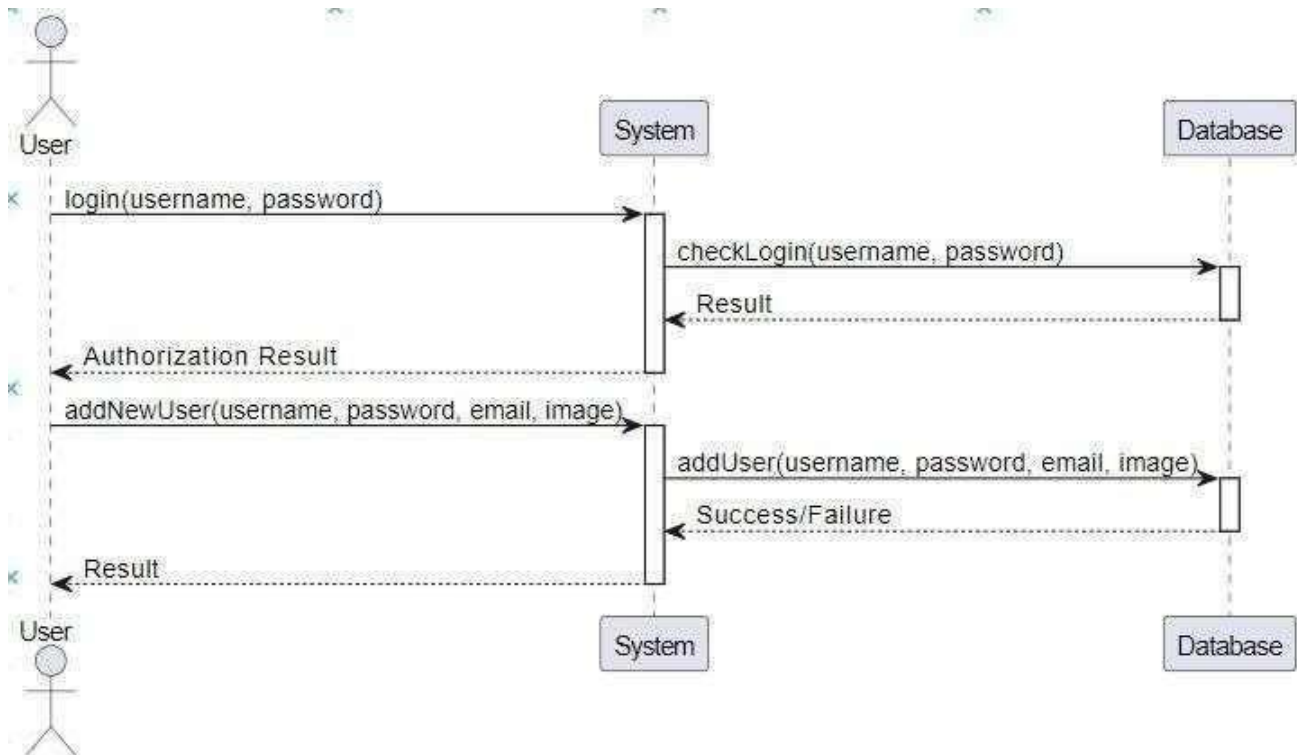


Fig 4.3 Sequence Diagram of Application

To access this system, users first provide their username and password. This information is then verified against a database of authorized users. If everything matches, access is granted, allowing the user to proceed. However, incorrect credentials result in denied access and an error message. Additionally, some situations might require further verification steps, like entering a security code, for enhanced security before granting access.

5. SYSTEM TESTING

5.1 Introduction:

Testing is the process of detecting errors. Testing performs a very critical role for quality assurance and for ensuring the reliability of software. The results of testing are used later on during maintenance also.

Psychology of Testing:

The aim of testing is often to demonstrate that a program works by showing that it has no errors. The basic purpose of testing phase is to detect the errors that may be present in the program. Hence one should not start testing with the intent of showing that a program works, but the intent should be to show that a program doesn't work. Testing is the process of executing a program with the intent of finding the errors.

Testing Objectives:

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Stating formally, we can say,

- Testing is a process of executing a program with the intent of finding an error.
- A successful test is one that uncovers an as yet undiscovered error.
- A good test case is one that has a high probability of finding error, if it exists.

- The tests are inadequate to detect possibly present errors.
- The software more or less confirms to the quality and reliable standards.

5.2 Types of Testing

- Unit Testing
- Link Testing
- System Testing
- Acceptance Testing

Unit Testing:

Unit testing focuses verification effort on the smallest unit of software i.e. the module. Using the detailed design and the process specifications testing is done to uncover errors within the boundary of the module. All modules must be successful in the unit test before the start of the integration testing begins.

Link Testing:

Link testing does not test software but rather the integration of each module in system. The primary concern is the compatibility of each module. The programmer tests where modules are designed with different parameters, length, type etc.

Integration Testing:

After the unit testing we have to perform integration testing. The goal here is to see if modules can be integrated properly, the emphasis being on testing interfaces between modules. This testing activity can be considered as testing the design and hence the emphasis on testing module interactions.

System Testing:

Here the entire software system is tested. The reference document for this process is the requirements document, and the goal is to see if software meets its requirements. Here entire project has been tested against requirements of project and it is checked whether all requirements of project have been satisfied or not.

Acceptance Testing:

Acceptance Test is performed with realistic data of the client to demonstrate that the software is working satisfactorily. Testing here is focused on external behavior of the system; the internal logic of program is not emphasized.

5.3 Test Cases

Login Page: Status : a.Checked

b.Verified

c.Passed

Test_Scenario_id	Test_Case_Description	Test_Case_id	Test_steps	Test_data	Expected Result	Actual Result	Status
TS_001	When authorized person tries to login with valid credentials	TC_Login_001	1. Enter valid username 2. Enter valid password 3. click on loginbutton	username: 1231 password: svecw123	Login Successful	Welcome message	a b c
TS_001	When authorized person tries to login with invalid credentials	TC_Login_002	1. Enter valid username 2. Enter invalid password 3. click on login button	username: 1231 password: svecw12	A message will be displayed as invalid.	A message will be displayed as invalid.	a b c
TS_001	When unauthorized person tries to login with valid credentials	TC_Login_003	Enter invalid username Enter valid password click on login button	username: 1231 password: svecw123	Login Successful	Welcome message	a b c
TS_001	When unauthorized person tries to login with invalid credentials	TC_Login_004	1. Enter invalid username 2. Enter invalid password 3. click on login button	username: 1231 password: svecw12	A message will be displayed as invalid.	A message will be displayed as invalid.	a b c

Table 5.3.1 Test Cases for Registration Page

Image Capturing: Status :

a.Checked

b.Verified

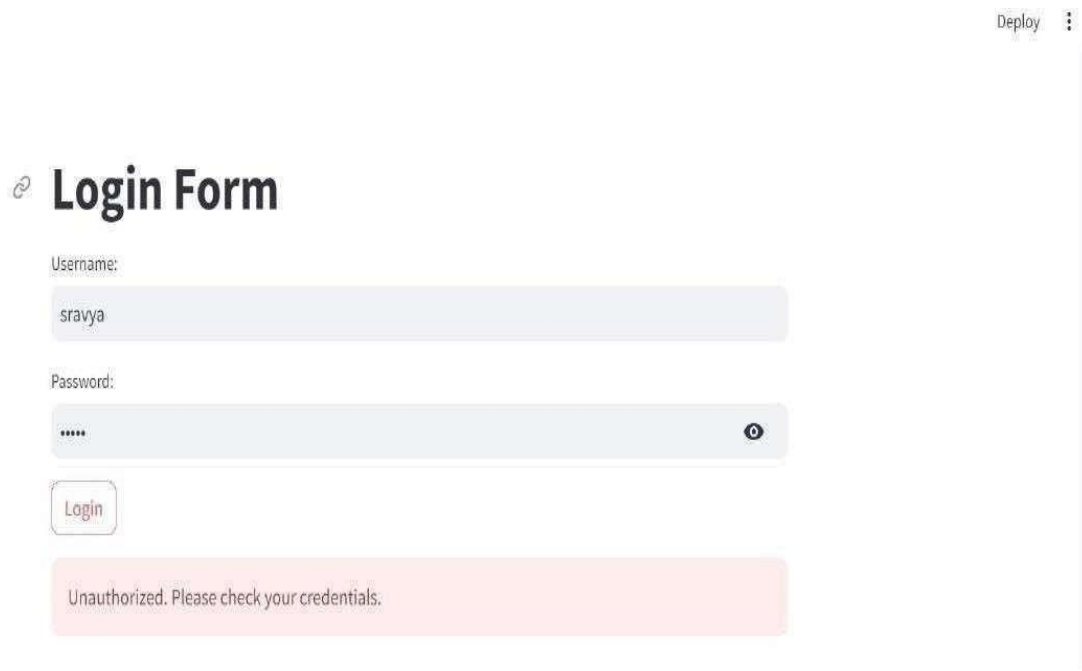
c.Passed

Test_Scenario_id	Test_Case_Description	Test_Case_id	Test_steps	Test_data	Expected Result	Actual Result	Status
TS_002	When no person is found.	TC_IC_001	A live image is automatically captured.	Image comparison will be done.	A message will be displayed as invalid. Email and sms alert will be sent.	A message will be displayed as invalid. Email and sms alert will be sent.	a b c
TS_002	Person is found. But the images does not match.	TC_IC_002	A live image is automatically captured.	Image comparison will be done.	A message will be displayed as unauthorized. Email and sms alert will be sent.	A message will be displayed as unauthorized. Email and sms alert will be sent.	a b c
TS_002	Person is authorized, but the lighting conditions are poor.	TC_IC_003	A live image is automatically captured.	Image comparison will be done.	Identifies the person as authorized and gives him access to use the application.	Identifies the person as authorized and gives him access to use the application.	a b c
TS_002	Person is unauthorized, but the lighting conditions are poor.	TC_IC_004	A live image is automatically captured.	Image comparison will be done.	A message will be displayed as unauthorized. Email and sms alert will be sent.	A message will be displayed as unauthorized. Email and sms alert will be sent.	a b c

Table 5.3.2 Test Cases for Login Page

6. LAYOUT DESIGNS

Login page: The user needs to provide his/her username and password in the Login page and after submitting, the system will check whether the user is authorized or not with the help of the database.



The screenshot displays a web application interface. At the top right, there is a 'Deploy' button and a vertical ellipsis menu. The main content area features a 'Login Form' with a title icon. Below the title, there are two input fields: 'Username:' containing the text 'sravya' and 'Password:' containing masked characters '*****'. A 'Login' button is positioned below the password field. At the bottom of the form, a red error message box states 'Unauthorized. Please check your credentials.'.

Fig 6.1 If the user is unauthorized, the application will display the same and would not allow user for further procedure.

The user is considered as unauthorized in two scenarios. They are as follows:

- 1) If the user enters wrong username. Then the person is considered as unauthorized.
- 2) If the user enters wrong Password. Then the person is considered as unauthorized.

Login Form

Username:

Password:

Welcome!

Fig 6.2 If the user is authorized, the application will would allow user for further procedure.

If the user enters correct username and password then he is considered as authorized. Then a “Welcome” message is displayed on screen and the user can access the website as he is considered as authorized.

Image Capture : Once the user successfully logs into the application, then live camera turns on and captures the image of the user. Then, checks it with the image of the person already present in the database. The system could also handle some non-ideal environmental conditions.



Fig 6.3 If the image is not matched, the user is considered as unauthorized, the application will display the same and would not allow user for further procedure.



Fig 6.4 If the image matches, the application welcomes the user and provides him/her the access for further processing.



Fig 6.5 If no face is identified in the image, the system displays the same and restricts further processing.

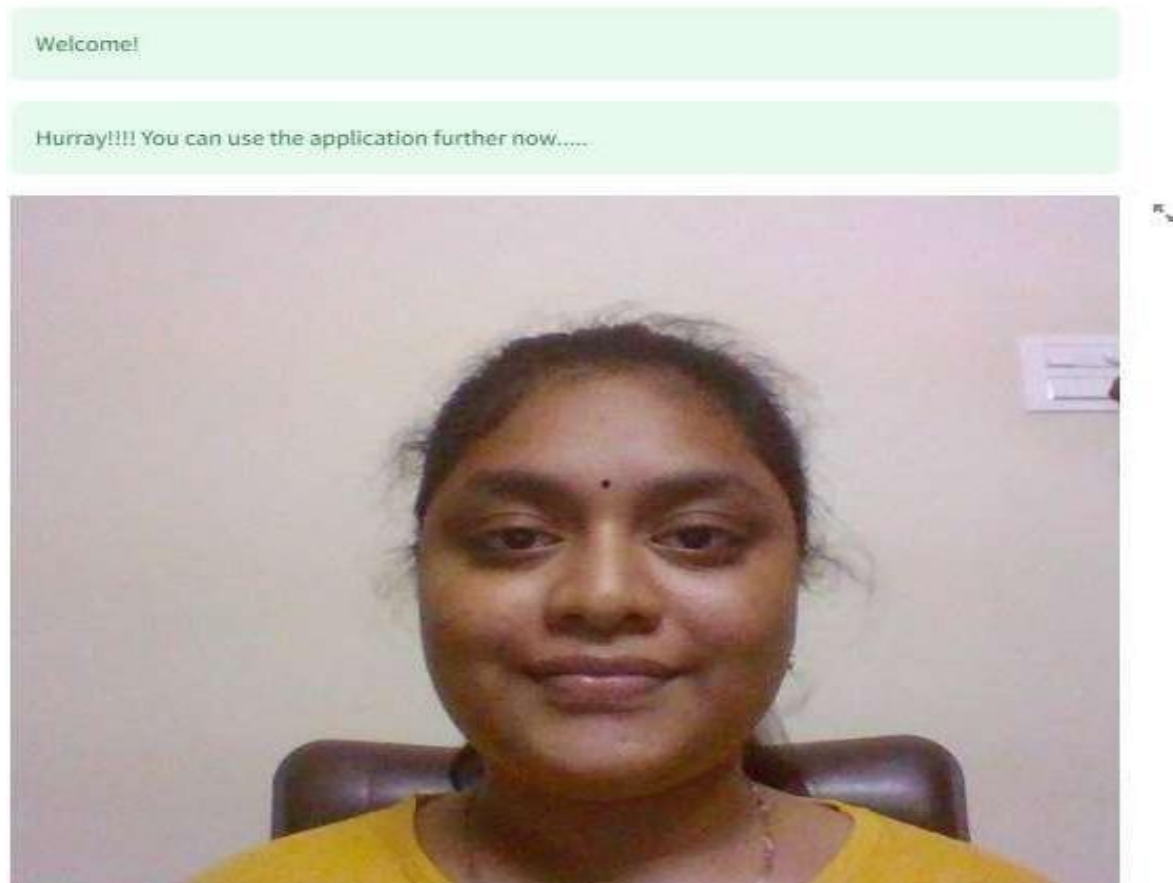


Fig 6.6 If the image matches, the application welcomes the user and provides him/her the access for further processing.

The application is reliable enough to recognize the persons irrespective of whether the person is wearing objects such as spectacles.

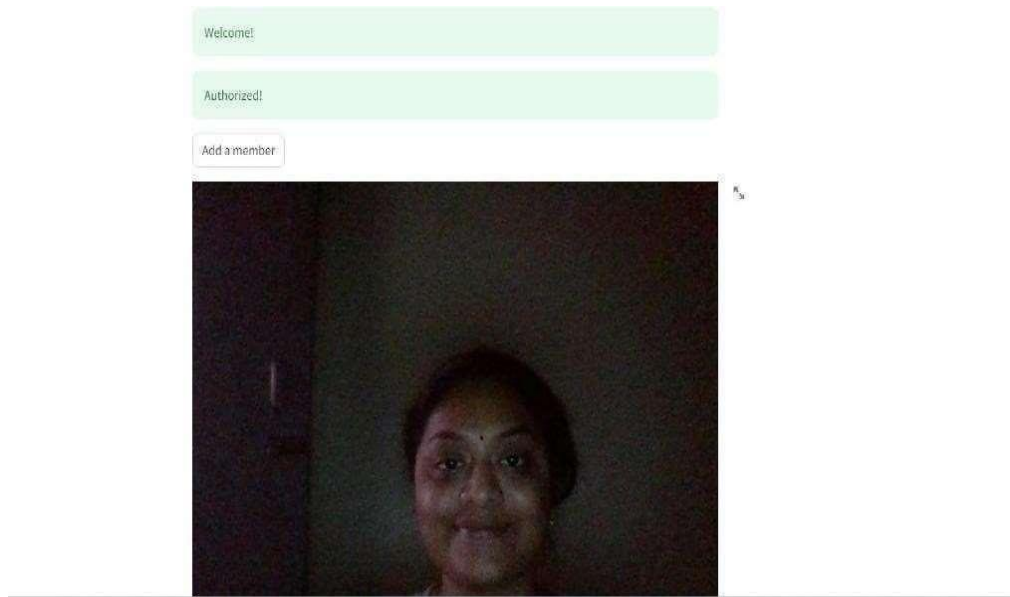


Fig 6.7 Even if the lighting conditions are not so ideal, the application can still enhance the image and provide suitable output.

For example, in the above picture even though the lighting conditions are very poor the system still managed to recognize the person.

Add new user: If the image matches and the user is recognized as authorized, he/she has the permission to access the application. They can easily add a new user, whom they consider are needed.

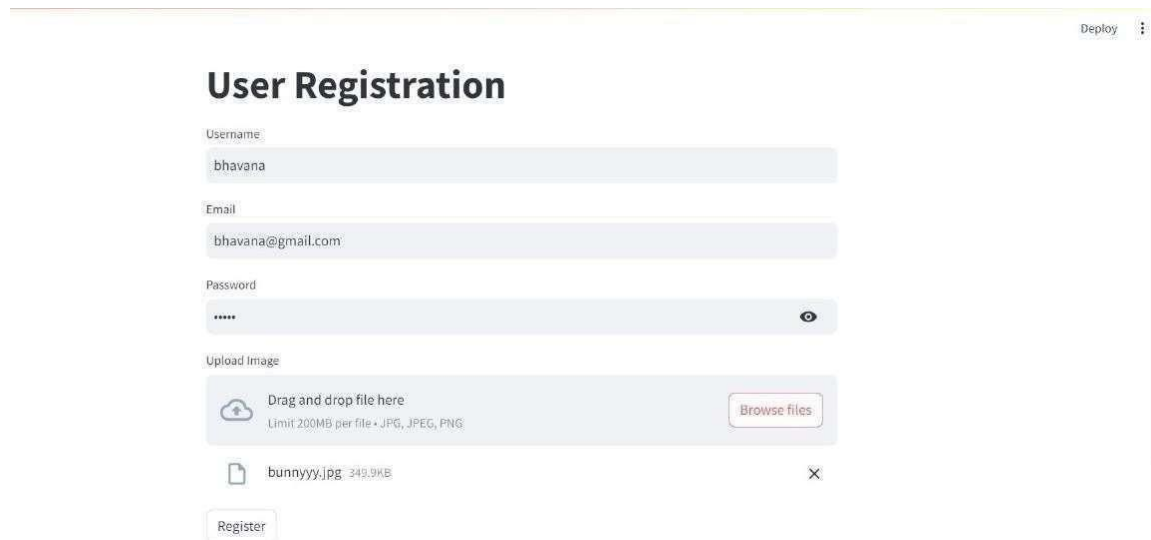
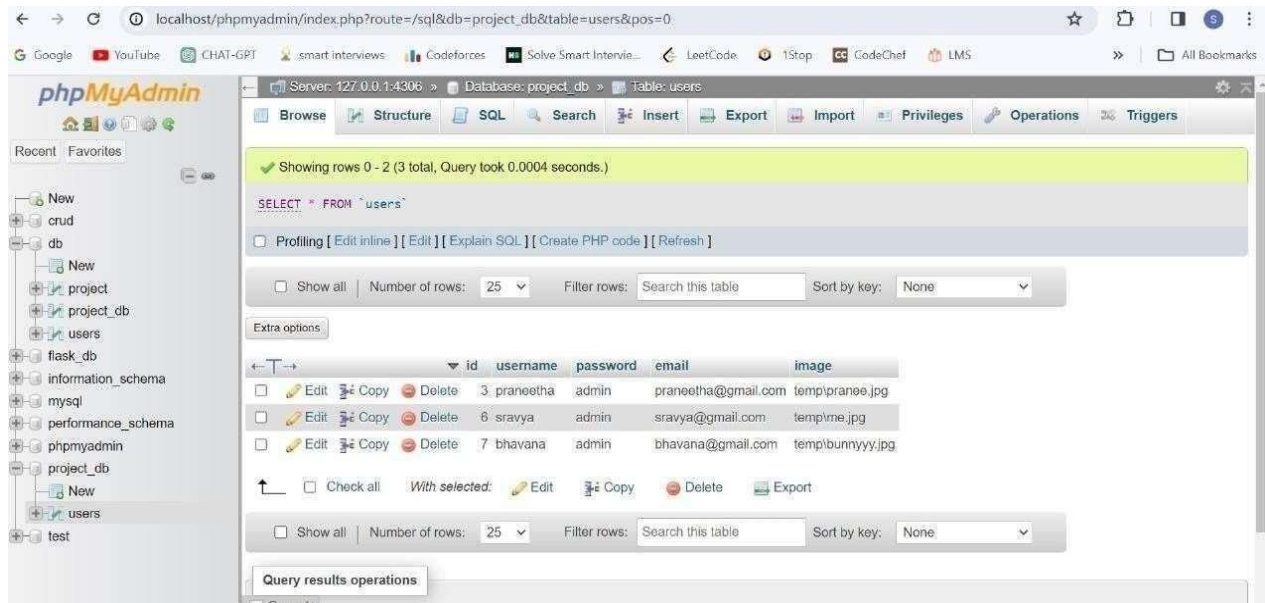
The image shows a web application interface for user registration. At the top right, there is a 'Deploy' button with a dropdown arrow. The main heading is 'User Registration'. Below it, there are four input fields: 'Username' with the value 'bhavana', 'Email' with the value 'bhavana@gmail.com', 'Password' with masked characters '*****' and an eye icon for toggling visibility, and 'Upload Image'. The 'Upload Image' section includes a drag-and-drop area with the text 'Drag and drop file here' and 'Limit 200MB per file • JPG, JPEG, PNG', a 'Browse files' button, and a file preview for 'bunnyyy.jpg' (349.9KB) with a close icon. At the bottom left of the form is a 'Register' button.

Fig 6.8 Add a new user.

A user is required to fill Username, Password and Email columns. Along with that user needs to upload the image of the concerned person. Then if he clicks on submit button the details of the concerned person will be stored in the database.



Showing rows 0 - 2 (3 total, Query took 0.0004 seconds.)

```
SELECT * FROM `users`
```

Number of rows: 25 Filter rows: Search this table Sort by key: None

	id	username	password	email	image
<input type="checkbox"/>	3	praneetha	admin	praneetha@gmail.com	temp/pranee.jpg
<input type="checkbox"/>	6	sravya	admin	sravya@gmail.com	temp/me.jpg
<input type="checkbox"/>	7	bhavana	admin	bhavana@gmail.com	temp/bunnyyy.jpg

Number of rows: 25 Filter rows: Search this table Sort by key: None

Fig 6.9 This is how the details are stored in the database.

After successfully entering the person's details in the registration form and clicking the submit button the details like their username, password, email and image along with the auto-incremented value of id will be stored in the database as shown in the figure 6.8.

Alerting: If the images does not match, the user is considered as an intruder and the system will automatically send the email to the owner with the image of the intruder and also an sms alert.

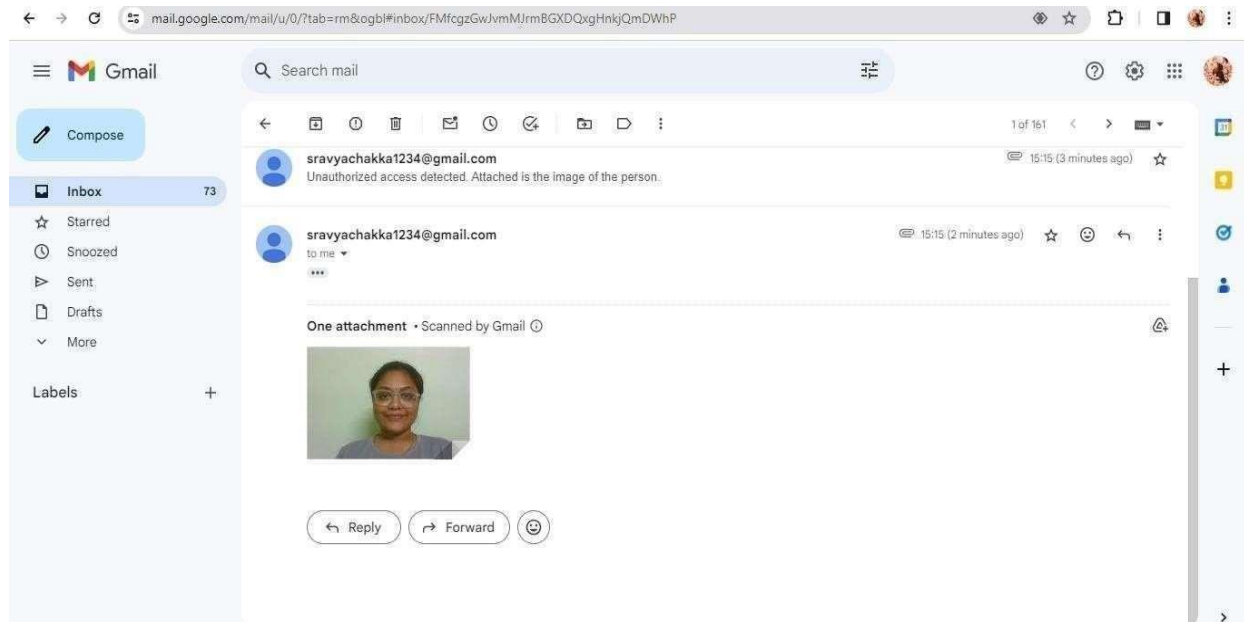


Fig 6.10 Email Alert

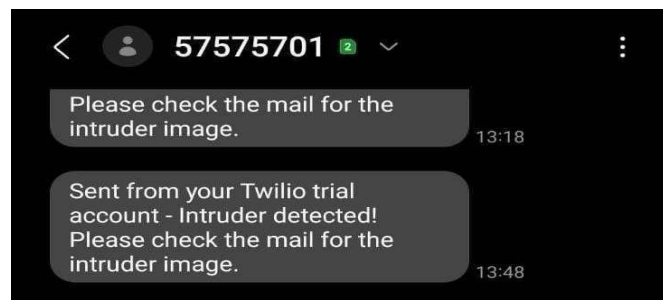


Fig 6.11 SMS alert.

7. CONCLUSION AND FUTURE WORK

In conclusion, the project aimed to develop a secure login system with facial recognition capabilities using Python and various libraries such as Streamlit, OpenCV, face recognition, and Twilio. The system integrated with a MySQL database to store user credentials and images for authentication.

Throughout the development process, several key objectives were achieved:

User Authentication:

1. The user authentication feature was meticulously designed to ensure robust verification of user credentials stored securely in the MySQL database.
2. Advanced encryption techniques were implemented to protect sensitive user data, including passwords, from unauthorized access or tampering.
3. The system employed salting and hashing algorithms to enhance the security of stored passwords, minimizing the risk of password-related vulnerabilities.
4. User authentication mechanisms underwent rigorous testing to validate their effectiveness in preventing unauthorized access and ensuring the integrity of user accounts.
5. Additionally, the system incorporated measures to prevent common security threats such as brute-force attacks and SQL injection, bolstering its overall resilience.

Facial Recognition:

1. The facial recognition feature leveraged state-of-the-art computer vision algorithms and machine learning techniques to accurately identify and authenticate users based on facial features.

2. Advanced image processing capabilities were integrated to detect and extract facial landmarks, enabling precise facial recognition even under varying lighting conditions or facial expressions.
3. The system employed a robust facial recognition model trained on a diverse dataset to ensure high accuracy and reliability in user authentication.
4. Continuous model refinement and optimization were conducted to improve the system's ability to recognize faces accurately and efficiently over time.
5. Facial recognition algorithms were augmented with anti-spoofing measures to detect and prevent fraudulent attempts to bypass authentication using fake or manipulated images.

Email Notifications:

1. Configurable Alerting System: The email notification feature allowed administrators to configure alerts based on specific criteria, such as the number of failed login attempts or access from unrecognized IP addresses.
2. Real-Time Monitoring: The system continuously monitored user login activities, instantly triggering email notifications upon detecting unauthorized access or suspicious login attempts.
3. Customizable Templates: Administrators could customize email notification templates to include relevant details such as user account information, timestamps, and IP addresses for forensic analysis.
4. Integration with Authentication Events: Email alerts were seamlessly integrated with authentication events, providing context-rich notifications to administrators for immediate action.
5. Audit Trail: The system maintained an audit trail of all email notifications sent, ensuring accountability and compliance with security policies and regulatory requirements.

SMS Alerts:

1. Immediate Notification: Twilio's API integration facilitated real-time SMS alerts to notify administrators promptly of any security breaches or unauthorized access attempts.
2. Redundancy and Reliability: The SMS alerting system was designed with redundancy and failover mechanisms to ensure uninterrupted communication with administrators, even during network disruptions or service outages.
3. Two-Factor Authentication (2FA): SMS alerts could be leveraged as part of a two-factor authentication (2FA) process, providing an additional layer of security for sensitive operations or privileged accounts.
4. Escalation Policies: Administrators could define escalation policies for SMS alerts, enabling notifications to be sent to multiple stakeholders or on-call personnel in case of critical incidents.
5. Compliance Logging: The system logged all SMS alert activities, including delivery status and response actions, to maintain an auditable trail of communication for compliance and incident response purposes.

User Management :

1. Streamlined Onboarding Process: The user management module provided a user-friendly interface for administrators to add new users effortlessly by inputting necessary details such as username, password, and email.
2. Image Upload for Facial Recognition: Administrators could upload user images directly through the user management interface, simplifying the process of enrolling users for facial recognition authentication.
3. Role-Based Access Control (RBAC): The system supported RBAC, allowing administrators to assign roles and permissions to users based on their

responsibilities and access requirements.

4. Account Deactivation: Administrators had the capability to deactivate or suspend user accounts temporarily, providing granular control over user access and security.

5. Audit Trail and Logging: User management actions, such as account creation, modification, or deletion, were logged systematically to maintain a comprehensive audit trail for compliance and accountability purposes.

Robustness:

1. Graceful Error Handling: The system implemented robust error-handling mechanisms to gracefully manage unexpected errors and exceptions, preventing system crashes or data corruption.

2. Fail-Safe Mechanisms: Built-in fail-safe mechanisms were employed to ensure data integrity and system stability, including transaction rollback and database consistency checks.

3. Automated Recovery Procedures: The system featured automated recovery procedures to restore system functionality in the event of a failure, minimizing downtime and disruption to users.

4. Scalability and Performance Optimization: Continuous performance monitoring and optimization efforts were undertaken to ensure the system's scalability and responsiveness under varying loads and usage patterns.

5. Proactive Monitoring and Alerting: Real-time monitoring tools were integrated to proactively identify potential issues or anomalies, triggering alerts for timely intervention and resolution by system administrators.

In summary, the developed system offers a robust and secure login mechanism with facial recognition capabilities. It addresses the need for enhanced security in user authentication processes and provides real-time alerts to administrators in case of security breaches. Further enhancements could include scalability improvements and integration with additional security features for comprehensive protection against unauthorized access.

8. REFERENCES

- M. Andriansyah, M. Subali, I. Purwanto, S. A. Irianto and R. A. Pramono, "e-KTP as the basis of home security system using arduino UNO," 2017 4th International Conference on Computer Application And Information Processing Technology (CAIPT), Kuta Bali, 2017, pp.1-5 .
- A. A. Dibazar, A. Yousefi, H. O. Park, B. Lu, S. George and T. W. Berger, "Intelligent acoustic and vibration recognition/alert systems for security breaching detection, close proximity danger identification, and perimeter protection," 2010 IEEE International Conference on Technologies for Homeland Security (HST), Waltham, MA, 2010, pp. 351-356, doi: 10.1109/THS.2010.5654931.
- G. Vakulya and G. Simon, "Low power accelerometer based intrusion and tamper detector," 2014 IEEE 11th International Multi-Conference on Systems, Signals Devices (SSD14), Barcelona, Spain, 2014, pp. 1-6, doi: 10.1109/SSD.2014.6808878.
- Iyer Saikumar, Gaonkar Pranjali, Wadekar Shweta, Kohmaria Nayan and Upadhyay Prashant, IoT based Intruder Detection System Using GSM (April 8, 2020). Proceedings of the 3rd International Conference on Advances in Science Technology (ICAST) 2020, Available at SSRN: <https://ssrn.com/abstract=3572326> or <http://dx.doi.org/10.2139/ssrn.3572326>.
- Zhengbing Hu, V. Nimko and P. Bykovyy, "Fuzzy logic based method to estimate the risk of alarm system false detection," Proceedings of International Conference on Modern Problem of Radio Engineering, Telecommunications and Computer Science, Lviv, UKraine, 2012, pp. 452-453

- H. Yan, G. Shi, Q. Wang and S. Hao, "Identification of Damaging Activities for Perimeter Security," 2009 International Conference on Signal Processing Systems, Singapore, 2009, pp. 162-166, doi: 10.1109/ICSPS.2009.17 [7] A Anitha, "Home security system using internet of things" 2017 IOP Conf. Ser.: Mater. Sci. Eng.263 042026

9. APPENDIX

9.1 Sample Code:

```
import streamlit as st
import mysql.connector
import cv2
from PIL import Image
import face_recognition
import time
import smtplib
import numpy as np
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from email.mime.image import MIMEImage
from twilio.rest import Client
import os

def check_login(username, password):
    try:
        conn = mysql.connector.connect(
            host="localhost",
            port=4306,
            user="root",
            password="",
            database="project_db"
        )
```

```
    cursor = conn.cursor()
    query = f"SELECT * FROM users WHERE username = '{username}' AND
password = '{password}'"
    cursor.execute(query)
    result = cursor.fetchone()
    conn.close()

    return result is not None, result

except Exception as e:
    st.error(f"An error occurred: {e}")
    return False, None

def capture_image_with_delay():
    cap = cv2.VideoCapture(0)

    time.sleep(3)
    ret, frame = cap.read()
    image_path = "captured_image.jpg"
    cv2.imwrite(image_path, frame)

    cap.release()

    return image_path

tolerance = 0.6
def compare_images(captured_image_path, database_image_path):
```



```
captured_image = face_recognition.load_image_file(captured_image_path)

captured_image_rgb = cv2.cvtColor(captured_image, cv2.COLOR_BGR2RGB)

captured_face_locations = face_recognition.face_locations(captured_image_rgb)

if len(captured_face_locations) == 0:

    return False

captured_encodings = face_recognition.face_encodings(captured_image_rgb,
captured_face_locations)

database_image = face_recognition.load_image_file(database_image_path)
database_encoding = face_recognition.face_encodings(database_image)[0]
if captured_encodings:
    for captured_encoding in captured_encodings:
        face_distance = face_recognition.face_distance([database_encoding],
captured_encoding)
        if face_distance <= tolerance:
            return True
    return False

def send_email(image_path, subject, body):
    sender_email = "sravyachakka1234@gmail.com"
    sender_password = "jtfguevtkjvxqdie"
```

```
receiver_email = "praveenachinthalapudi1012@gmail.com"

msg = MIMEMultipart()
msg['From'] = sender_email
msg['To'] = receiver_email
msg['Subject'] = subject

msg.attach(MIMEText(body, 'plain'))

with open(image_path, 'rb') as image_file:
    attachment = MIMEImage(image_file.read())
    attachment.add_header('Content-Disposition', 'attachment',
filename="captured_image.jpg")
    msg.attach(attachment)

with smtplib.SMTP('smtp.gmail.com', 587) as server:
    server.starttls()
    server.login(sender_email, sender_password)
    server.sendmail(sender_email, receiver_email, msg.as_string())
    st.success('Mail sent successfully!')

def send_sms_alert():
    account_sid = 'ACe7449f99771dce63bc30f95798e3498c'
    auth_token = 'e532d3a53923e2f4947facd1d0a26edb'
    twilio_phone_number = '+16592187247'
    recipient_number = '+919121949113'
    client = Client(account_sid, auth_token)
```

```
message = "Intruder detected! Please check the mail for the intruder image."
try:
    client.messages.create(
        to=recipient_number,
        from_=twilio_phone_number,
        body=message
    )
    st.success('SMS sent successfully!')
except Exception as e:
    st.error(f'Error: {str(e)}')

def main():
    st.title("Login Form")
    username = st.text_input("Username:")
    password = st.text_input("Password:", type="password")

    if 'image_authorized' not in st.session_state:
        st.session_state.image_authorized = False

    if st.button("Login"):
        if username and password:
            authorized, user_data = check_login(username, password)

            if authorized:
                st.success("Welcome!")
                captured_image_path = capture_image_with_delay()
                database_image_path = user_data[4]
```

```
if compare_images(captured_image_path, database_image_path):
    st.success("Hurray!!!! You can use the application further now.....")
    st.write("Now you can add a user")

    st.session_state.image_authorized = True

else:
    st.image(Image.open(captured_image_path), caption="Captured Image",
use_column_width=True)

    st.error("Person using the application is found to be unauthorised!!!!")
    send_email(captured_image_path, "Unauthorized Access Detected",
"Unauthorized access detected. Attached is the image clicked.")
    send_sms_alert()

else:
    st.error("Unauthorized. Please check your credentials.")

if st.session_state.image_authorized:
    upload_dir = "temp"
    if not os.path.exists(upload_dir):
        os.makedirs(upload_dir)
    st.write("### Add New User")
    new_username = st.text_input("New Username:")
    new_password = st.text_input("New Password:", type="password")
    new_email = st.text_input("Email:")
    uploaded_file = st.file_uploader("Upload Image")
```

```
if st.button("Submit"):
    if new_username and new_password and new_email and uploaded_file:
        image_path = os.path.join(upload_dir, f"{new_username}.jpg")
        with open(image_path, "wb") as f:
            f.write(uploaded_file.getvalue())
        try:
            conn = mysql.connector.connect(
                host="localhost",
                port=4306,
                user="root",
                password="",
                database="project_db"
            )
            cursor = conn.cursor()
            cursor.execute("INSERT INTO users (username, password, email,
image) VALUES (%s, %s, %s, %s)",
                (new_username, new_password, new_email, image_path))
            conn.commit()

            st.success("User added successfully!")

        except Exception as e:
            st.error(f"An error occurred while adding the user: {e}")
        finally:
            conn.close()
    else:
```

```
st.error("Please fill in all fields.")
```

```
if __name__ == "__main__":
```

```
    main()
```