
FACECERTIFY - REALTIME EXAM VERIFICATION

A MINI PROJECT REPORT

by

ANSON MATHEW JAISON (VJC22CS034)

ARJUN GIREESH (VJC22CS038)

EBIN MANOJ (VJC22CS057)

ELWIN VINCENT (VJC22CS064)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISWAJYOTHI COLLEGE OF ENGINEERING AND

TECHNOLOGY, VAZHAKULAM

APRIL 2025

FACECERTIFY - REALTIME EXAM VERIFICATION

A MINI PROJECT REPORT

by

ANSON MATHEW JAISON (VJC22CS034)

ARJUN GIREESH (VJC22CS038)

EBIN MANOJ (VJC22CS057)

ELWIN VINCENT (VJC22CS064)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

under the guidance

of

Ms. NEHA BEEGAM P E

Assistant Professor, CSE Department



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISWAJYOTHI COLLEGE OF ENGINEERING AND

TECHNOLOGY, VAZHAKULAM

APRIL 2025

VISWAJYOTHI COLLEGE OF ENGINEERING AND TECHNOLOGY, VAZHAKULAM

Department of Computer Science and Engineering

Vision

Moulding socially responsible and professionally competent Computer Engineers to adapt to the dynamic technological landscape

Mission

1. Foster the principles and practices of computer science to empower life-long learning and build careers in software and hardware development.
2. Impart value education to elevate students to be successful, ethical and effective problem-solvers to serve the needs of the industry, government, society and the scientific community.
3. Promote industry interaction to pursue new technologies in Computer Science and provide excellent infrastructure to engage faculty and students in scholarly research activities.

Program Educational Objectives

Our Graduates

1. Shall have creative and critical reasoning skills to solve technical problems ethically and responsibly to serve the society.
2. Shall have competency to collaborate as a team member and team leader to address social, technical and engineering challenges.
3. Shall have ability to contribute to the development of the next generation of information technology either through innovative research or through practice in a corporate setting
4. Shall have potential to build start-up companies with the foundations, knowledge and experience they acquired from undergraduate education

Program Outcomes

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences

3. **Design / development of solutions:**Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety and the cultural, societal and environmental considerations.
4. **Conduct investigations of complex problems:**Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:**Create, select and apply appropriate techniques, resources and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:**Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:**Understand the impact of the professional engineering solutions in societal and environmental contexts and demonstrate the knowledge of and need for sustainable development.
8. **Ethics:**Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice
9. **Individual and team work:**Function effectively as an individual and as a member or leader in diverse teams and in multidisciplinary settings
10. **Communication:**Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and unread in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:**Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes

1. Ability to integrate theory and practice to construct software systems of varying complexity
2. Able to Apply Computer Science skills, tools and mathematical techniques to analyse, design and model complex systems
3. Ability to design and manage small-scale projects to develop a career in a related industry.

**VISWAJYOTHI COLLEGE OF ENGINEERING AND
TECHNOLOGY, VAZHAKULAM**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



BONAFIDE CERTIFICATE

Certified that the mini project work entitled “**FACECERTIFY - REALTIME EXAM VERIFICATION**” is a bonafide work done by **ANSON MATHEW JAISON (VJC22CS034)**, **ARJUN GIREESH (VJC22CS038)**, **EBIN MANOJ (VJC22CS057)**, **ELWIN VINCENT (VJC22CS064)** in partial fulfillment of the award of the Degree of **Bachelor of Technology in Computer Science & Engineering** from APJ Abdul Kalam Technological University, Thiruvananthapuram, Kerala during the academic year 2024-2025.

Internal Supervisor

External Supervisor

Mini Project Coordinator

Head of Department

ACKNOWLEDGEMENT

First and foremost, We thank **God Almighty** for his divine grace and blessings in making all these possible. May he continue to lead us in the years ahead. It is our privilege to render our heartfelt thanks to our most beloved Manager **Msg. Dr. PIOUS MALEKANDATHIL**, our Director **Rev. Fr. PAUL PARATHAZHAM** and our Principal **Dr. K K RAJAN** for providing us the opportunity to do this mini project during the sixth semester of our B.Tech degree course. We are deeply thankful to our Head of the Department, **Dr. AMEL AUSTINE** for his support and encouragement. We would like to express my sincere gratitude and heartfelt thanks to our Mini Project Guide and Coordinator **Ms. NEHA BEEGAM P E**, Asst. Professor, Department of Computer Science and Engineering, for her motivation, guidance, and valuable assistance throughout the project. We also thank all the staff members of the Computer Science Department for providing their assistance and support. Last, but not the least, we thank all our friends and family for their valuable feedback from time to time as well as their help and encouragement.

DECLARATION

We undersigned hereby declare that the mini project report "**FACECERTIFY - REALTIME EXAM VERIFICATION**", submitted for partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology of the APJ Abdul Kalam Technological University is a bonafide work done under the supervision of **Ms. NEHA BEEGAMP E**. This submission represents ideas in our own words and where ideas or words of others have been included, we have adequately and accurately cited and referenced the original sources. We also declare that we have adhered to the ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or formed the basis for the award of any degree, diploma or similar title of any other University.

Place: Vazhakulam

Date: 04-04-2025

ANSON MATHEW JAISON

ARJUN GIREESH

EBIN MANOJ

ELWIN VINCENT

ABSTRACT

The FaceCertify - Real Time Exam Verification system is an innovative web-based solution designed to enhance academic integrity by automating student identity verification and attendance tracking during examinations. Leveraging advanced facial recognition technology, the system employs a pre-trained Support Vector Machine (SVM) model integrated with the facerecognition library to verify student identities in real time, ensuring only authorized individuals participate. Accessible via a user-friendly portal, students capture their facial images using a webcam, triggering a recognition process that matches them against a stored dataset with a confidence threshold of 70 percent, subsequently displaying their details—name, subject, classroom, and seat number—alongside a dynamic seat map. The system automates attendance logging, recording timestamps in a CSV file while preventing duplicates, and offers administrators a comprehensive dashboard to manage student data, upload updated recognition models, and monitor exam activities. Built with Flask, Python, and Tailwind CSS, FaceCertify operates on minimal hardware—a webcam-equipped device, a server with 16 GB RAM, and 50 GB SSD storage—making it adaptable to academic environments. By addressing impersonation risks and streamlining exam logistics, FaceCertify provides a secure, efficient, and scalable tool for modern educational institutions, ensuring trust and order in the examination process.

Key Words :- Facial Recognition, Real-Time Verification, Academic Integrity, Attendance Tracking, Support Vector Machine (SVM), FaceCertify, Web-Based Portal, Flask, Student Management, Seat Allocation, Python, CSV Storage, Webcam, Security, Scalability

Contents

Acknowledgement	i
Declaration	ii
Abstract	iii
List of figures	vii
1 INTRODUCTION	1
1.1 Problem Definition	1
1.2 Objective	2
1.3 Scope	3
2 EXISTING SYSTEM	4
2.1 [1] Attendance Monitoring Systems:	4
2.1.1 Advantages	4
2.1.2 Disadvantages	5
2.2 [2] Face Recognition Systems:	5
2.2.1 Advantages	6
2.2.2 Disadvantages	6
2.3 [3] Face Recognition Application:	7
2.3.1 Advantages	7
2.3.2 Disadvantages	7

3	REQUIREMENT ANALYSIS	9
3.1	Product Perspective	9
3.2	Product Functions	9
3.2.1	User Functions	9
3.2.2	Admin Functions:	10
3.3	User Classes and Characteristics	11
3.4	Hardware and Software Requirements	12
3.4.1	Hardware Requirements	12
3.4.2	Software Requirements	12
3.4.3	Visual Studio Code	12
4	SYSTEM DESIGN	14
4.1	Context Viewpoint Design	14
4.2	Information Viewpoint Design	18
4.3	Interface Viewpoint Design	19
4.4	Interaction Viewpoint Design	21
5	PROJECT WORK PLANNING	26
5.1	Gantt Chart	26
6	IMPLEMENTATION AND TESTING	27
6.1	Implementation	27
6.1.1	Algorithms Used	28
6.1.2	Database Structures	31
6.2	Testing	34
6.2.1	Testing of Face Recognition and Verification System	34
6.2.2	Testing Environment and Setup:	34

6.2.3	Testing Methodology:	34
6.2.4	Webcam-Based Testing:	35
6.2.5	Web Interface Testing:	35
6.2.6	Test Scenarios and Results:	35
6.2.7	Evaluation and Observations:	36
7	CONCLUSION AND FUTURE SCOPE	37
7.1	Conclusion	37
7.2	Future Scope	38
	References	39
	Appendix A Implementation code for core portion	41
A.1	Initial Setup and Configuration	41
A.2	Face Recognition and Verification	42
A.3	Attendance Marking	45
A.4	Student Data Management	45
A.5	Model and Label Updates	46
A.6	Security and Access Control	47
A.7	CSV Utility Functions	47
A.8	Loading Face Dataset	48
A.9	Training Face Recognition Model	50
	Appendix B Screenshots	51
B.1	Home Page	51
B.2	Student Authentication	52
B.3	Student Face Capture	53

B.4 Student Details and Seat Map 53

B.5 Admin Login 54

B.6 Admin Dashboard - Add Student 54

B.7 Admin Dashboard - Upload Model 55

B.8 Admin Dashboard - Students List 55

B.9 Admin Dashboard - View Attendance 55

B.10 Admin Dashboard 56

List of Figures

3.1 Use Case Diagram	11
4.1 Visualization of Camera Access Use Case Diagram	14
4.2 Visualization of Admin login Use Case Diagram	15
4.3 Visualization of Admin logout Use Case Diagram	15
4.4 Visualization of Seat Allocation Use Case Diagram	15
4.5 Visualization of Attendance Monitoring Use Case Diagram	16
4.6 Visualization of Manage Student Use Case Diagram	16
4.7 Visualization of Student Camera Access Use Case Diagram	16
4.8 Visualization of Student Login Case Diagram	17
4.9 Visualization of Seat Allocation Use Case	17
4.10 Visualization of Manual Verification Use Case	17
4.11 Visualization of ER Diagram	18
4.12 Visualization of System Component Diagram	19
4.13 Visualization of User Component Diagram	20
4.14 Visualization of Server Component Diagram	20
4.15 Visualization of Admin Login Sequence Diagram	21
4.16 Visualization of Seat Allocation Sequence Diagram	22
4.17 Visualization of Attendance Monitoring Sequence Diagram	22
4.18 Visualization of Manage Student Sequence Diagram	23
4.19 Visualization of Admin Logout Sequence Diagram	23
4.20 Visualization of Student Login Sequence Diagram	24
4.21 Visualization of Camera Access Sequence Diagram	24

4.22 Visualization of Seat Allocation Sequence Diagram	25
5.1 Gantt Chart	26
6.1 Student CSV File	32
6.2 Attendance CSV File	33
B.1 Home Page	51
B.2 Student Authentication	52
B.3 Student Face Capture	53
B.4 Student Details and Seap Map	53
B.5 Admin login	54
B.6 Admin Dashboard - Add Student	54
B.7 Admin Dashboard – Upload Model	55
B.8 Admin Dashboard – Students List	55
B.9 Admin Dashboard – Attendance Records	56
B.10Admin Dashboard	56

Chapter 1

INTRODUCTION

In today's educational world, maintaining academic integrity during exams has become a significant challenge. Traditional identity verification methods, like manual ID checks and paper attendance systems, are starting to feel outdated as student numbers rise and the need for efficient, secure processes increases. These old-school methods can be slow, prone to human mistakes, and open to impersonation, which ultimately undermines the fairness and credibility of exam results.

Enter biometric technologies, especially facial recognition, which offer a compelling solution to these problems. Facial recognition provides a contactless, automated, and real-time way to verify identities, making it a great fit for boosting exam security. However, developing a system that seamlessly integrates this technology into a user-friendly and scalable platform for educational institutions takes careful planning and execution.

FaceCertify - Real Time Exam Verification is a web-based system designed to address these challenges. By leveraging facial recognition technology, FaceCertify automates the process of verifying student identities and tracking attendance during exams. The system is built with user experience in mind, ensuring an intuitive interface for both students and administrators while improving the efficiency and security of the examination process.

1.1 Problem Definition

In educational institutions, keeping exams fair and secure is a top priority, but the old-school ways of verifying student identities and tracking attendance just aren't cutting it anymore. These traditional methods come with a host of challenges:

Manual Identity Verification: When invigilators have to check student ID cards by hand, it can be a slow process that's prone to errors. In large exam halls, this can lead to frustrating delays and inefficiencies, especially in schools with a lot of students.

Impersonation Risks: It's tough to stop cheating, like when someone takes an exam for someone else, using fake IDs or even colluding with staff. These manual checks can really undermine

the fairness and trustworthiness of the whole exam system.

Inefficient Attendance Tracking: Relying on paper-based attendance systems is a hassle. They're not only cumbersome and error-prone, but they also require a lot of manual work for entering, verifying, and keeping records. This just adds to the administrative workload and makes it harder to keep accurate, up-to-date records.

Scalability Challenges: As schools grow, the traditional ways of verifying identities and tracking attendance struggle to keep up with the increasing number of students. This can lead to logistical headaches and expose weaknesses in exam security.

Lack of Real-Time Monitoring: Many existing systems don't offer real-time oversight for identity verification and attendance, making it hard to tackle issues on the spot or ensure compliance during exams.

These issues highlight the pressing need for a more automated, secure, and scalable solution to improve the exam process. We need a system that reduces human involvement while boosting efficiency and security to tackle these challenges and maintain academic integrity.

1.2 Objective

The main goal of the FaceCertify system is to create a fast and secure platform for verifying exams in real-time. It uses facial recognition technology to confirm student identities, which helps maintain academic integrity during tests. This system is designed to tackle the increasing issues of impersonation and unauthorized access in both online and in-person testing by offering a dependable, automated way to verify identities. Here's what FaceCertify aims to achieve:

First off, it plans to establish a strong biometric verification process that accurately identifies students through facial recognition, reducing false positives and ensuring that only those who are authorized can take exams. By combining a pre-trained Support Vector Machine (SVM) model with real-time image processing, the system guarantees high accuracy and can scale effectively across various student groups. Secondly, FaceCertify wants to simplify the exam process by automating attendance tracking and seating verification, which cuts down on manual oversight and eases the administrative load. With features like real-time attendance marking and a visual seating chart, the system gives both students and administrators instant feedback, boosting operational efficiency.

In the end, the goal is to develop a user-friendly, tech-savvy solution that not only protects the integrity of academic assessments but also meets the evolving needs of education, providing a smooth experience for both students and administrators.

1.3 Scope

The FaceCertify system is all about creating a real-time exam verification platform that boosts security and efficiency in academic settings. It's designed specifically for educational institutions like schools, colleges, and universities, where keeping assessments honest is crucial. The system caters to two main groups: students, who need easy access to their exam portals, and administrators, who manage student verification, attendance, and overall system operations.

At its core, FaceCertify uses facial recognition technology for identity verification, working in real-time with webcam feeds from students' devices. It can capture and process facial images, compare them to a pre-trained model, and deliver instant verification results with a confidence level of 70 percent. The system also handles attendance by automatically logging student presence in a central database (**attendance.csv**) and helps with seating arrangements by showing seat numbers and a visual map to verified students. Plus, there's an admin interface for managing student records, uploading recognition models, and keeping track of attendance logs, all accessible through a secure dashboard.

However, the system's focus is strictly on identity verification and attendance tracking during scheduled exams, so it doesn't cover other functions like delivering exam content, grading, or proctoring beyond confirming identities. It assumes that users have basic hardware (like webcams) and internet access, along with a pre-existing dataset of student images for training the recognition model. While it's built to scale, the initial rollout is aimed at small to medium-sized institutions, with plans to expand in the future to support larger groups or additional biometric options.

Chapter 2

EXISTING SYSTEM

The increasing reliance on digital and remote examination systems has prompted the development of various identity verification and proctoring solutions aimed at ensuring academic integrity. This chapter examines existing systems that address similar challenges to those targeted by FaceCertify, focusing on their approaches to student authentication, attendance management, and seating arrangements. By analyzing these systems, we can identify gaps and limitations that FaceCertify seeks to overcome.

2.1 [1] Attendance Monitoring Systems:

Attendance monitoring systems play a crucial role in managing human resources in small and medium-sized enterprises (SMEs), especially as remote work becomes more common in the wake of COVID-19. Traditional methods, like logbooks and badge systems, often lead to errors and inefficiencies, which is why many are turning to AI-powered solutions like facial recognition. This project focuses on creating an affordable, mobile-based attendance system specifically designed for SMEs, utilizing cloud technology to tackle resource limitations. Drawing inspiration from the work of Thai et al., the goal is to improve accuracy and efficiency while addressing the challenges of manual timekeeping.

2.1.1 Advantages

- **Improved Accuracy:** This system uses unique facial features as biometric identifiers, which means it can keep attendance records spot-on. It cuts out the human errors that often come with manual entries and tackles the common problem of proxy attendance, where one employee clocks in for another who's not there.
- **Real-Time Data:** By processing facial recognition data on the fly, the system gives immediate updates on attendance. This allows managers to access up-to-date information and make quick, informed decisions about staffing or resource allocation without any holdups.

- **Cost-Effective:** By taking advantage of existing mobile devices and cloud services, this solution keeps hardware costs low. It spares small and medium-sized enterprises (SMEs) from having to invest in pricey dedicated equipment, making advanced technology more accessible even on a tight budget.
- **Flexibility:** Employees can check in from various locations using their personal devices, which supports remote and hybrid work models. This adaptability is key to navigating the changing dynamics of the workplace that SMEs face today.
- **Reduced Overhead:** Automating attendance tracking lightens the load for HR personnel, eliminating the tedious task of manual record-keeping. This frees up staff to concentrate on more strategic, high-value tasks.

2.1.2 Disadvantages

- **Hidden Costs:** Training employees, subscribing to cloud services, and providing ongoing support can really put a strain on the budgets of small and medium-sized enterprises (SMEs). These expenses might overshadow the initial savings from the system, even if it seems affordable at first glance.
- **Privacy Risks:** Storing biometric data comes with its own set of compliance and security challenges. SMEs need to carefully navigate data protection laws and ensure that sensitive facial information is kept safe, or they could find themselves facing legal or ethical dilemmas.
- **Technical Barriers:** Many SMEs might not have the technical know-how required for deploying and maintaining these systems. This often means they'll need to seek outside help, which can complicate the adoption process and create a reliance on third-party support.
- **Connectivity Issues:** Relying on the cloud can lead to problems if there are server outages or if the internet connection is spotty. This could disrupt attendance tracking and throw a wrench in the operational consistency for SMEs that struggle with unreliable network access.

2.2 [2] Face Recognition Systems:

Traditional methods of monitoring attendance in exam halls often rely on tedious roll calls or ID checks, which can be both time-consuming and prone to issues like impersonation. This project proposes a cutting-edge facial recognition system powered by deep learning to streamline and secure the attendance process. By utilizing sophisticated algorithms like Support Vector Machines (SVM) for classifying features and Convolutional Neural Networks (CNN) for analyzing images, the system captures students' facial data through cameras, compares it with a pre-registered

database, and records attendance in real time. Building on the framework established by Dhanya et al., this initiative aims to boost exam integrity and improve administrative efficiency by seamlessly integrating with the current infrastructure.

2.2.1 Advantages

- **Enhanced Security:** This system uses unique facial biometric signatures to accurately identify students, effectively stopping proxy attendance or impersonation during exams.
- **Real-Time Verification:** It processes facial data instantly as students walk into the hall, allowing for immediate attendance confirmation and cutting down on delays before the exam starts.
- **High Accuracy:** By utilizing deep learning models trained on a wide range of datasets, it ensures precise recognition, even when facial expressions change or there are minor obstructions.
- **Automation:** This technology removes the need for manual attendance marking, reducing human involvement and minimizing errors in record-keeping.
- **Scalability:** It's designed to accommodate multiple exam halls or large groups of students by adjusting camera setups and database capacity, making it flexible for various institutional needs.

2.2.2 Disadvantages

- **Initial Costs:** Getting started requires a hefty investment in high-resolution cameras, powerful computers, and software development, which can really stretch institutional budgets at first.
- **Privacy Concerns:** This technology gathers and keeps sensitive biometric data, which raises some ethical issues. It's crucial to comply with data protection laws to prevent any misuse or data breaches.
- **Technical Complexity:** It uses advanced deep learning algorithms and needs to be integrated into existing systems, meaning you'll need skilled professionals for setup, troubleshooting, and ongoing maintenance.
- **Connectivity Dependence:** It depends on a reliable internet connection to sync data in real-time with a central server, which can lead to operational hiccups if there are network outages or server issues.

2.3 [3] Face Recognition Application:

Keeping track of attendance manually, whether through paper logs or ID scans, can be quite a hassle and prone to mistakes, especially in lively settings like classrooms or offices. This project introduces an innovative attendance monitoring system that uses Google's FaceNet model, all wrapped up in the Flutter framework. By harnessing FaceNet's powerful deep convolutional neural networks (CNNs) for extracting facial features and taking advantage of Flutter's ability to work across different platforms, the system captures facial data through mobile device cameras. It then processes this data into 192-dimensional embeddings and compares it with a local database for real-time attendance tracking. Drawing inspiration from Kulkarni et al.'s work, this solution not only makes better use of resources but also boosts accuracy on both Android and iOS devices.

2.3.1 Advantages

- **Enhanced Security:** FaceNet creates unique facial embeddings that maintain a high level of similarity within the same class, which helps ensure reliable identification and keeps unauthorized individuals from gaining access or attending in place of someone else in monitored environments.
- **Real-Time Verification:** It processes images right away, either on the device or through cloud integration, delivering immediate confirmation of attendance—something that's crucial in fast-paced settings like lectures or meetings.
- **High Accuracy:** By leveraging FaceNet's powerful feature extraction, which has been trained on extensive datasets, it achieves precise recognition even with changes in lighting, angles, or facial expressions, surpassing traditional methods.
- **Automation:** It takes the hassle out of manual record-keeping by automatically serializing and storing attendance data in a Flutter-compatible database (like Hive), which helps cut down on administrative tasks and reduces the chance of human error.
- **Scalability:** Thanks to Flutter's single-codebase design and FaceNet's lightweight inference, it can be deployed across various devices and locations, making it flexible enough for both small groups and large institutions.

2.3.2 Disadvantages

- **Initial Costs:** Getting started means you'll need to invest in mobile devices that work well with the system, optimize the model for Flutter, and possibly tap into cloud re-

sources. This can be a tough pill to swallow for organizations that are already tight on budget, even though the system is quite efficient.

- **Privacy Concerns:** When it comes to storing facial embeddings, there are some serious privacy and security concerns to consider. You'll need to implement encryption and make sure you're compliant with regulations like GDPR to keep user data safe from any breaches.
- **Technical Complexity:** Merging FaceNet with Flutter isn't a walk in the park. It involves a lot of steps like preprocessing, serialization (think FaceAdapter), and handling model inference. This means developers need to be well-versed in both deep learning and cross-platform frameworks to pull it off.
- **Connectivity Dependence:** Although using local storage options like Hive can help ease some of the issues, features that require real-time updates might still rely on a stable internet connection. This can lead to delays or even failures if the network goes down.

Chapter 3

REQUIREMENT ANALYSIS

3.1 Product Perspective

FaceCertify is a real-time exam verification system designed to enhance academic integrity in educational institutions by utilizing Support Vector Machines (SVM) for facial recognition, enabling secure and efficient identity verification of students during examinations while automating attendance tracking and seat allocation. The system integrates seamlessly with existing student databases to authenticate identities, capture images via webcams for cost-effective deployment, and processes them to extract facial features using the face-recognition library, with SVM matching these features against enrolled profiles to confirm identities, mark attendance, and assign seats in real time, all while providing detailed analytics through an admin dashboard. With robust security measures such as data encryption and a maximum of three recognition attempts per user to prevent misuse, FaceCertify ensures privacy and reliability, supports scalability for various classroom sizes, and adheres to ethical standards for biometric data handling, making it a comprehensive solution for maintaining exam integrity.

3.2 Product Functions

FaceCertify provides a streamlined solution for exam verification by automating identity authentication, attendance tracking, and seat allocation through facial recognition, with distinct functionalities for students and administrators to ensure academic integrity and operational efficiency.

3.2.1 User Functions

Input collected from user:

- Photos: Used for facial recognition to verify the student's identity.
- Name: Identifies the student for attendance and seat allocation.

- Subject: Indicates which exam the student is attending.

Output Provided to user:

- Class room: Notifies the student of their assigned exam room.
- Seat Number: Displays the exact seat allocated for the exam.
- Attendance Marking: Confirms successful attendance recording after verification.

3.2.2 Admin Functions:

- Add Student: Registers a new student into the system.
- Upload Models: Allows the admin to upload or update the facial recognition model.
- View Students: Displays a list of all registered students.
- Delete Student: Removes a student's data from the system.
- View Attendance: Shows attendance records for all students.

3.3 User Classes and Characteristics

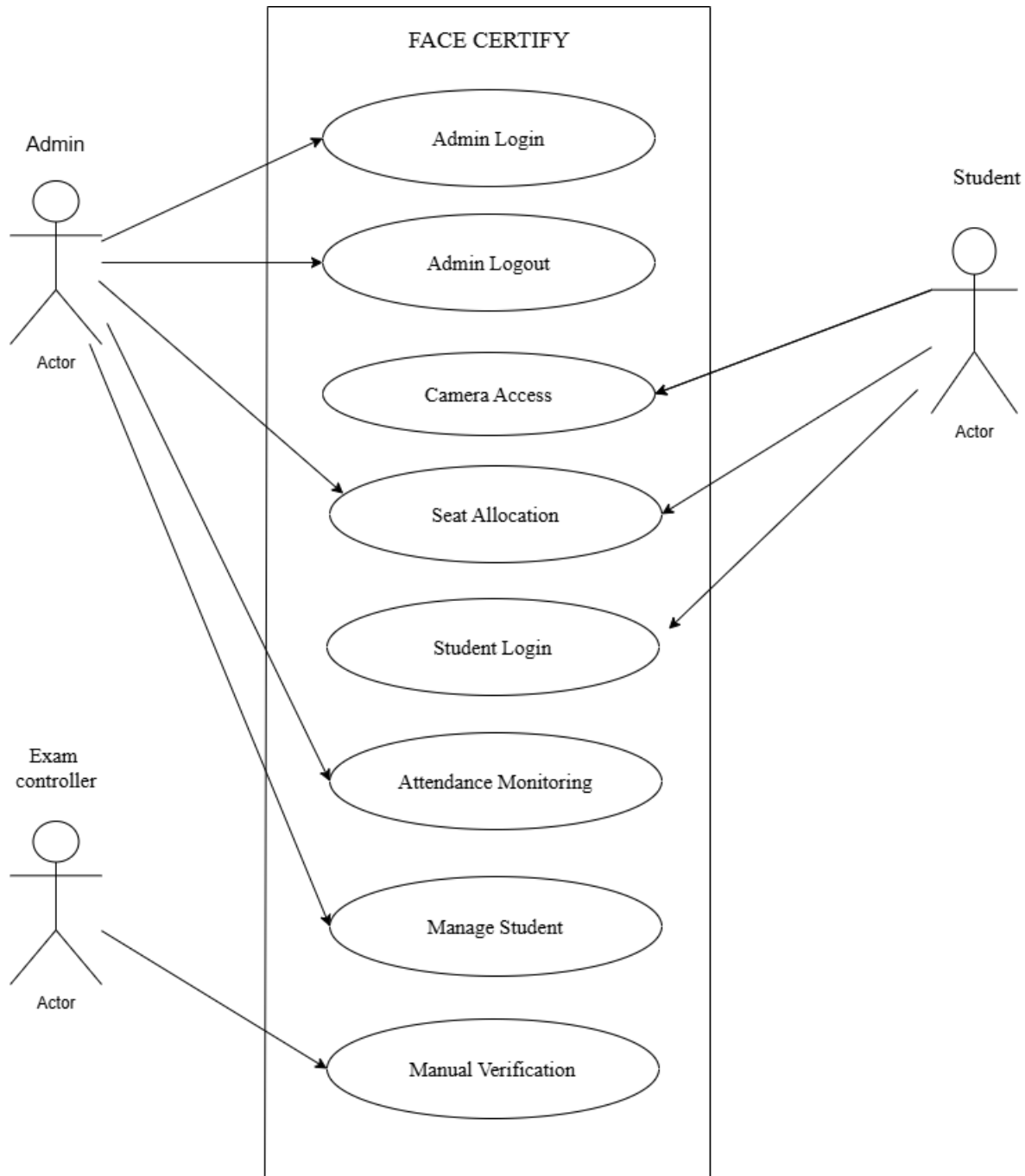


Figure 3.1: Use Case Diagram

- Admin
 - Admin Login: Allows the administrator to securely access the admin dashboard.
 - Admin Logout: Ends the admin session and secures the system.
 - Seat Allocation: Assigns specific seats to students for exams.
 - Attendance Monitoring: Tracks and reviews student attendance during exams.
 - Manage Student: Adds, edits, or removes student details from the system.

- Student
 - Camera Access: Grants the application permission to use the student’s camera for facial recognition.
 - Seat Allocation: Displays the assigned exam seat to the student.
 - Student Login: Authenticates the student using facial recognition or credentials.
- Exam Controller
 - Manual Verification: Provides a fallback to manually verify student identity when automated methods fail.

3.4 Hardware and Software Requirements

3.4.1 Hardware Requirements

The hardware for FaceCertify includes a webcam for capturing student facial images, a processor to handle real-time processing on both client and server sides, and sufficient memory to support the web portal and recognition tasks. Storage is essential for temporary image uploads, student data, attendance records, and model files, while a reliable network ensures seamless communication between devices and the server. A display is required for administrators to view the dashboard clearly, and the server itself—equipped with an SSD—powers the backend application and recognition engine, providing the computational backbone for the system’s operations.

3.4.2 Software Requirements

The software components of FaceCertify encompass Facial Recognition for identity verification, driven by a Support Vector Machine (SVM) model, and a Web-Based Portal built with Flask to deliver the user interface. Python serves as the core programming language, managing backend logic and data processing, while CSV Storage handles student and attendance data persistence. The system supports Real-Time Verification and Attendance Tracking functionalities, with Student Management and Seat Allocation features integrated into the administrative dashboard, all secured with basic Security measures and designed for Scalability across academic environments.

3.4.3 Visual Studio Code

Visual Studio Code (VS Code) stands out as a versatile and user-friendly code editor, meticulously crafted by Microsoft to cater to the diverse needs of developers across the globe. With its sleek and intuitive interface, VS Code provides a seamless coding experience, empowering developers to focus on their craft without unnecessary distractions. Its extensive ecosystem of extensions

amplifies its capabilities, allowing users to tailor the editor to their specific workflows and preferences. Whether you're working on a small script or a large-scale project, VS Code's robust features, including built-in Git integration, IntelliSense, debugging support, and task automation, streamline the development process and boost productivity.

Chapter 4

SYSTEM DESIGN

4.1 Context Viewpoint Design

Actors: Students and Admins.

Services:

- Capture Image (Student): Triggers verification via index.html.
- View Results (Student): Displays details and seat.
- Add Student (Admin): Updates students.csv via admin.html.
- View Attendance (Admin): Reads attendance.csv.

Camera Access

The Camera Access module in FaceCertify allows the system to capture real-time images of students for facial recognition. This feature is essential for identity verification during exams. The admin ensures the camera is functional and accessible before the verification process begins.

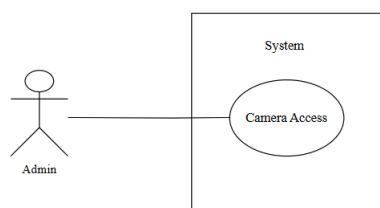


Figure 4.1: Visualization of Camera Access Use Case Diagram

Admin Login

The Admin Login module in FaceCertify allows administrators to access the system. This ensures only authorized users can manage student data, monitor attendance, and oversee the exam process..

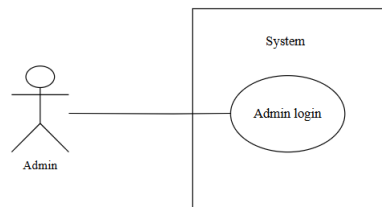


Figure 4.2: Visualization of Admin login Use Case Diagram

Admin Logout

The Admin Logout module in FaceCertify allows administrators to securely exit the system, ensuring no unauthorized access after their session. This helps protect sensitive data and maintain system .

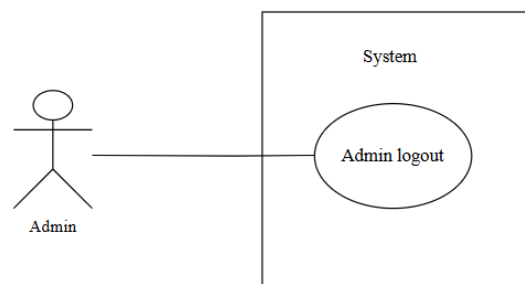


Figure 4.3: Visualization of Admin logout Use Case Diagram

Seat Allocation

The Seat Allocation module in FaceCertify automates the assignment of exam seats to students. Based on student identity verification, the system ensures each student is assigned a designated seat, reducing manual effort and preventing seat misuse.

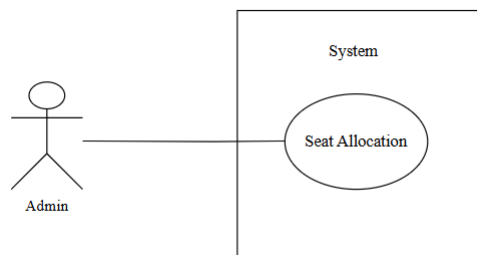


Figure 4.4: Visualization of Seat Allocation Use Case Diagram

Attendance Monitoring

The Attendance Monitoring module in FaceCertify automates student attendance tracking during exams using facial recognition. The system verifies each student's identity and marks their presence in real time, reducing the risk of proxy attendance. Admins can monitor attendance records and generate reports for better exam management.

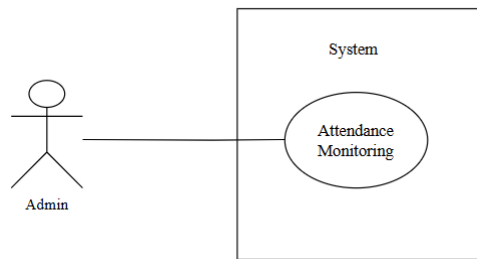


Figure 4.5: Visualization of Attendance Monitoring Use Case Diagram

Manage Student

The Manage Student module in FaceCertify allows the admin to add, update, or remove student records. This feature ensures that the database remains up to date with accurate student information. The admin can upload student details, including names, student IDs, and associated facial recognition data. This functionality is crucial for maintaining a reliable verification system during exams.

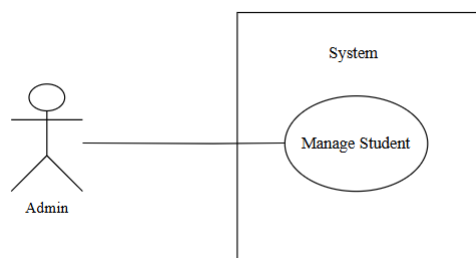


Figure 4.6: Visualization of Manage Student Use Case Diagram

Student Camera Access

The Camera Access feature allows students to turn on their camera for facial recognition during exams. The system captures and verifies their identity using a pretrained Keras model to ensure the right student is taking the exam.

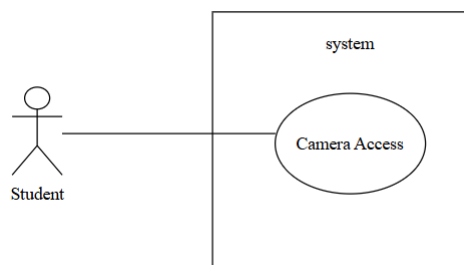


Figure 4.7: Visualization of Student Camera Access Use Case Diagram

Student Login

The Student Login module in FaceCertify allows students to securely access the system. This process involves authentication using their credentials and facial recognition verification to ensure

identity accuracy. Upon successful login, students can proceed with the examination process, attendance marking, or other system features.

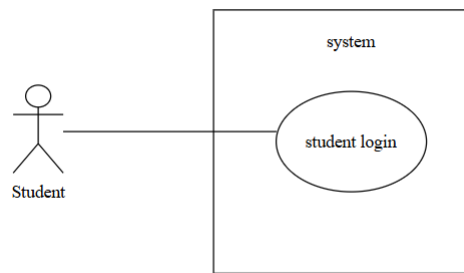


Figure 4.8: Visualization of Student Login Case Diagram

Student Seat Allocation

The Seat Allocation feature assigns students to specific seats during exams. The system ensures organized seating to prevent malpractices and maintain fairness.

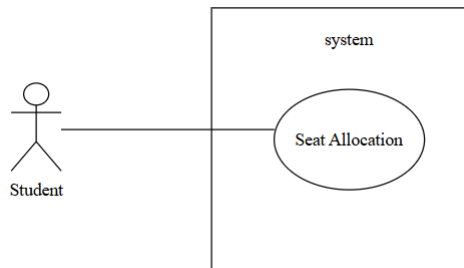


Figure 4.9: Visualization of Seat Allocation Use Case

Manual Verification

The Manual Verification feature allows the Exam Controller to verify student identities manually within the system. This serves as a backup verification method in case of system errors or discrepancies in automated facial recognition. It ensures accuracy and reliability in student authentication during exams.

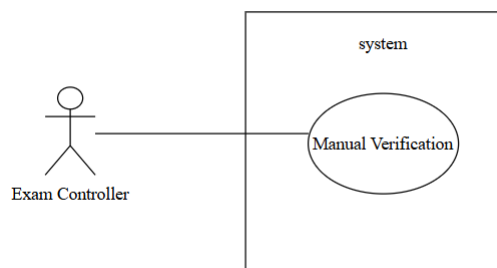


Figure 4.10: Visualization of Manual Verification Use Case

4.2 Information Viewpoint Design

The Entity-Relationship (ER) diagram (Figure 4.11) for the FaceCertify system illustrates the relationships between key entities involved in real-time exam verification. The central entity, STUDENT, contains attributes like ROLLNO, NAME (with FIRST NAME, MIDDLE NAME, LAST NAME), PHOTO, and EMBEDDING, storing student details and facial recognition data used for identification. STUDENT is linked to SEAT ALLOCATION via STUDENT ID, associating each student with a SEAT NO and STUDENT ID, which aligns with the student.csv file mapping students to seats (e.g., Arjun to Seat 3). The ADMIN entity, identified by ADMIN ID, manages the system through actions like ADD STUDENT and UPLOAD MODEL, reflecting the admin dashboard's functionality (admin.html) for student management and model uploads. ADMIN also connects to EXAM via EXAMINER ID, which oversees the exam process, while ATTENDANCE records STUDENT ID and TIME STAMP, corresponding to the inferred attendance.csv structure (e.g., "Arjun, 2025-04-02, 10:15, Present"). This ER diagram effectively models FaceCertify's data structure, supporting student verification, seat allocation, and attendance tracking.

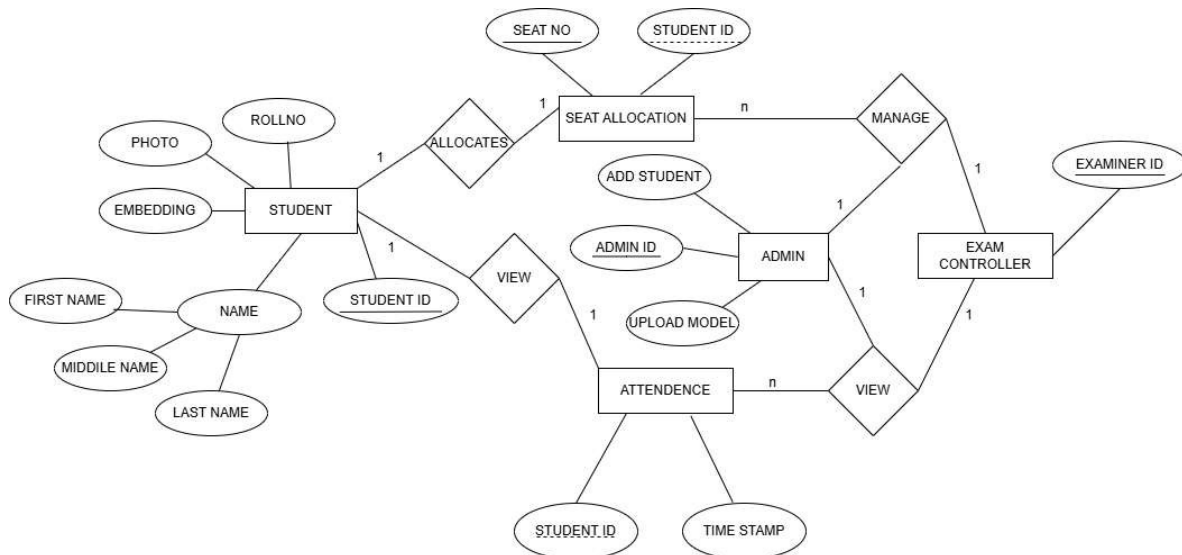


Figure 4.11: Visualization of ER Diagram

Design Entities

- Student
- Attendance
- Verification
- Seat Display
- Admin Panel

Design Attributes

Table Name	Description
Student	name(string),subject(string),classroom (string), seatno (string).
Attendance	name (string), date (string), timestamp (string), status (string).
Verification	confidencescore (float), result (dict).
SeatDisplay	seatno (string), highlightstatus (boolean).
AdminPanel	studentlist (list),attendancerecords (list).

Table 4.1: Design Attributes

4.3 Interface Viewpoint Design

System Interface

The diagram represents the system interface for FaceCertify, a facial recognition attendance system. Users interact via a UI, with a camera capturing images for seat allocation. The server performs face detection, utilizing an SVM model for authentication, ensuring secure attendance verification through an authentication server.

User Side: The user interacts through a User Interface, where a Camera captures their image for Seat Allocation.

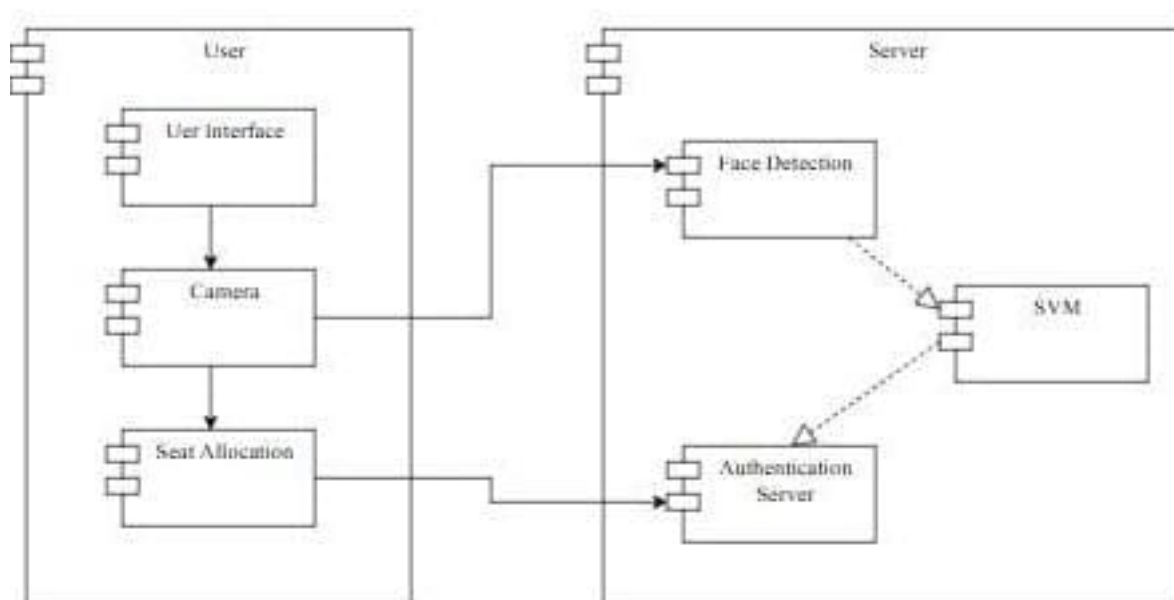


Figure 4.12: Visualization of System Component Diagram

User Component

The User Component consists of three key modules that interact with the system: User Interface: Allows users to interact with the system for authentication and attendance. Camera: Captures the user's image for face recognition. Seat Allocation: Assigns a seat based on successful authentication. These components work together to ensure smooth user identification and automated seat assignment.

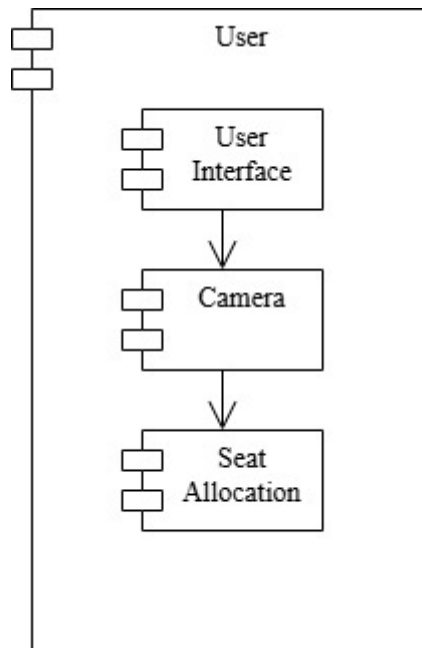


Figure 4.13: Visualization of User Component Diagram

Server Component

This diagram represents the server-side components of the FaceCertify facial recognition system. It includes Face Detection, which extracts facial features and sends them to an SVM (Support Vector Machine) for classification. The Authentication Service then verifies identities using the classified data, ensuring secure and accurate user authentication.

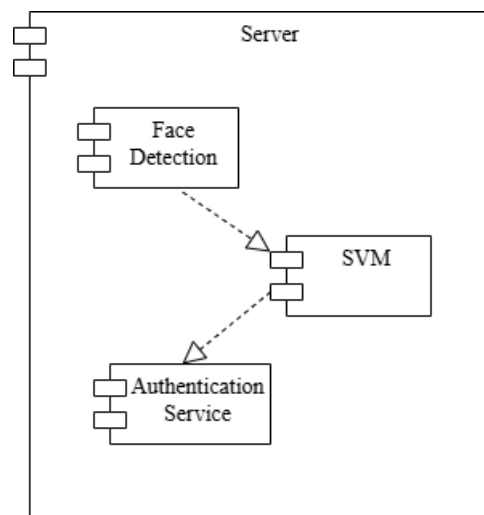


Figure 4.14: Visualization of Server Component Diagram

4.4 Interaction Viewpoint Design

Admin Login

It illustrates the interaction between a student, user interface, seat allocation module, and SVM model. The admin logs in through the user interface, triggering seat allocation and student management, while attendance monitoring is handled, ensuring accurate verification through the Keras-based facial recognition model.

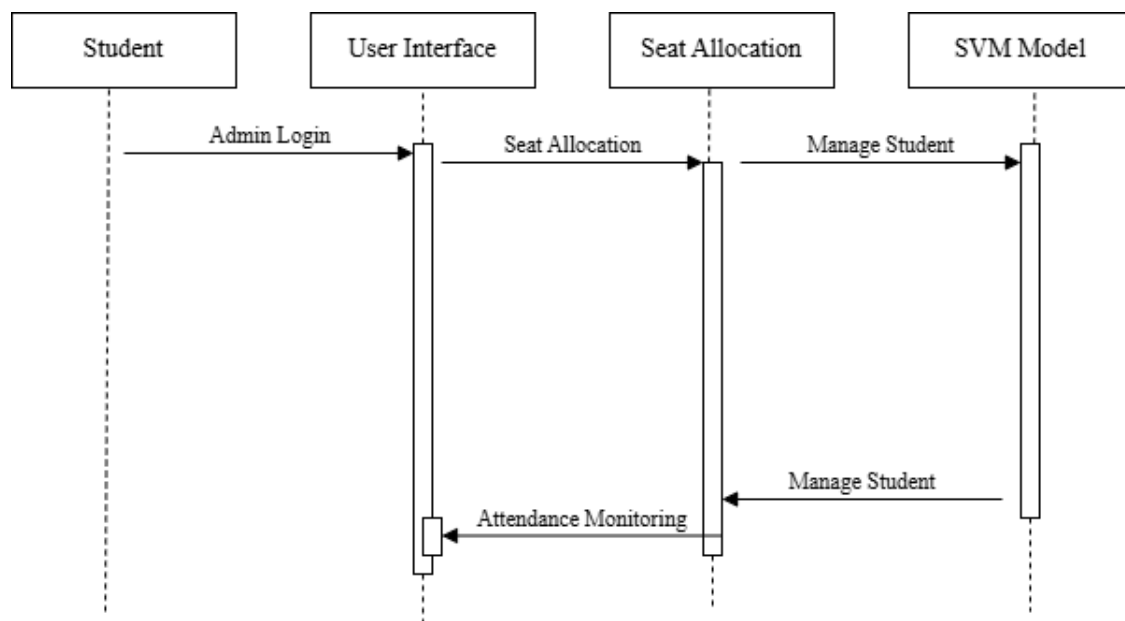


Figure 4.15: Visualization of Admin Login Sequence Diagram

Seat Allocation (Admin)

The interaction begins with the admin login via the user interface. Seat allocation is initiated, triggering student management in the SVM model. The system manages student data, ensuring proper allocation. Attendance monitoring follows, verifying student presence. This structured workflow ensures an efficient and organized seating arrangement while maintaining accurate student records for examinations or classroom management.

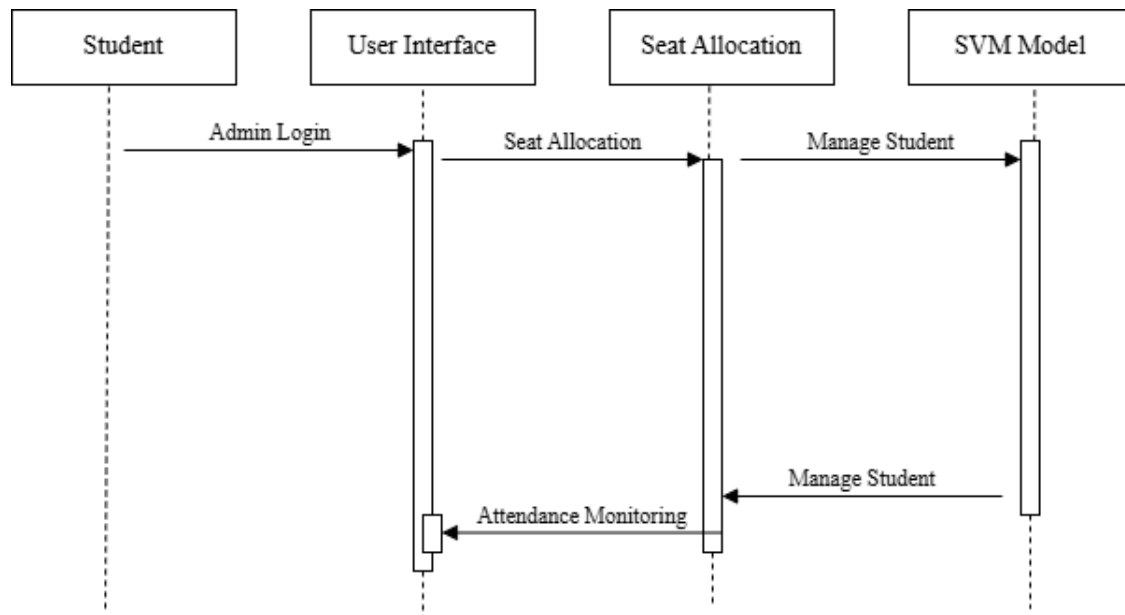


Figure 4.16: Visualization of Seat Allocation Sequence Diagram

Attendance Monitoring

The process begins with the admin logging into the system via the user interface. Seat allocation is managed, followed by student data management in the SVM model. Attendance monitoring is then initiated, ensuring accurate tracking of student presence. The system communicates with the seat allocation module and SVM model to verify attendance and maintain accurate records for academic or administrative purposes.

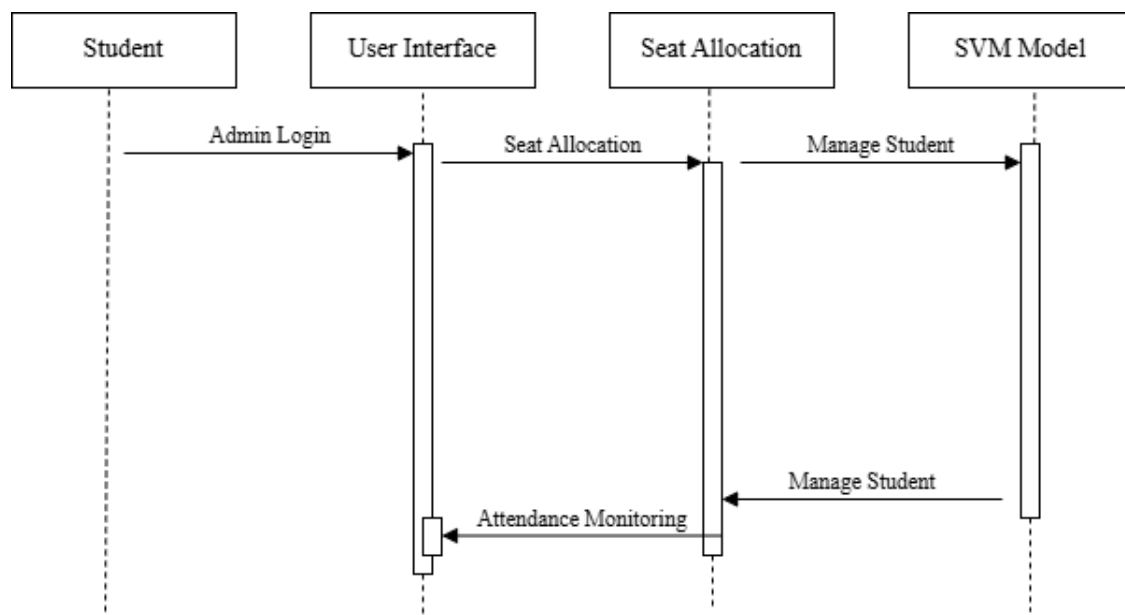


Figure 4.17: Visualization of Attendance Monitoring Sequence Diagram

Manage Student

Seat allocation is handled before student management begins. The system communicates with the SVM model to manage student data, ensuring proper identification and authentication. The

updated student information is then sent back to the seat allocation module, ensuring seamless integration and accurate record-keeping.

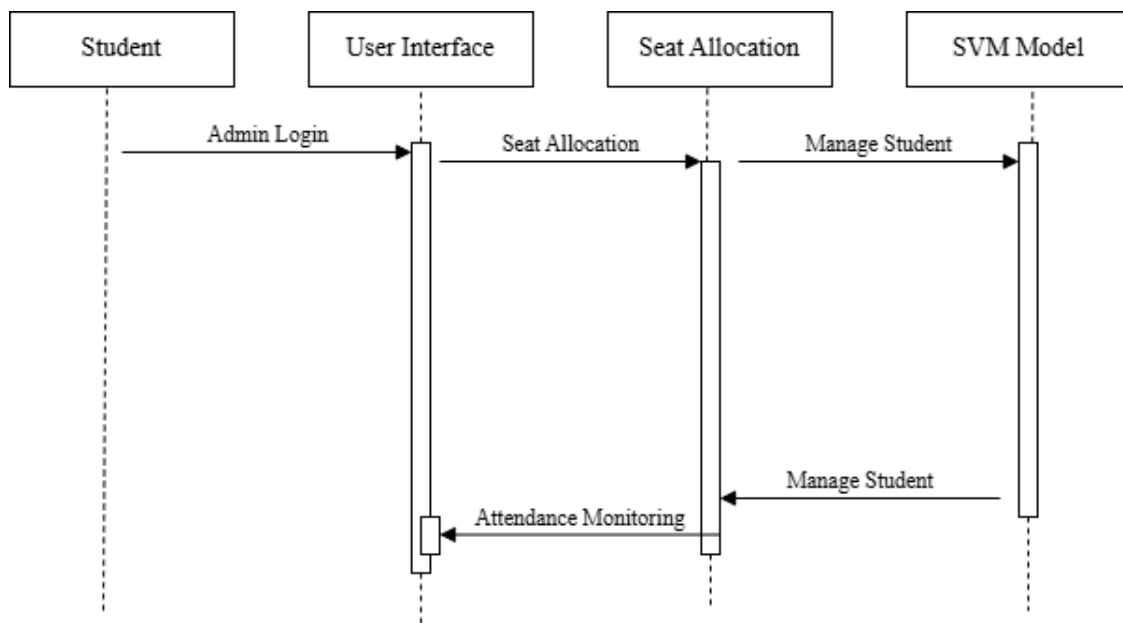


Figure 4.18: Visualization of Manage Student Sequence Diagram

Admin Logout

It illustrates the interaction between a student, user interface, seat allocation module, and SVM model. The admin initiates sign-out, triggering notifications and unsaving data. The system sends a data status update before closing the application, ensuring a smooth and secure logout process.

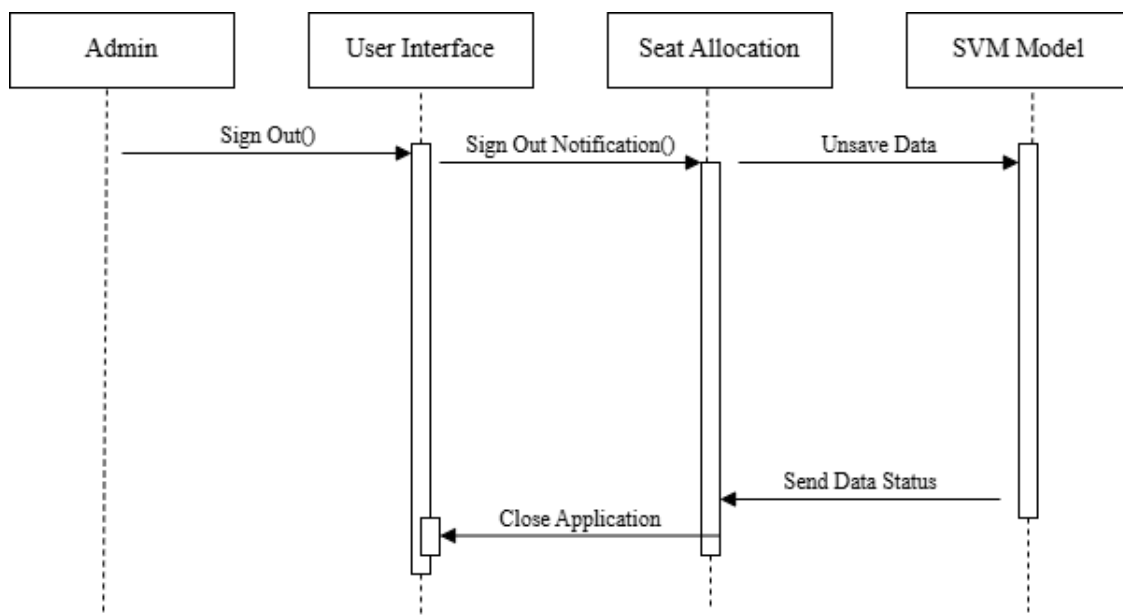


Figure 4.19: Visualization of Admin Logout Sequence Diagram

Student Login

The student begins login by allowing camera access for face detection. The system forwards the captured image to a SVM model for face matching and identity verification. Once confirmed, the seat allocation system assigns a seat and updates the user interface, ensuring an automated and seamless authentication and seating process

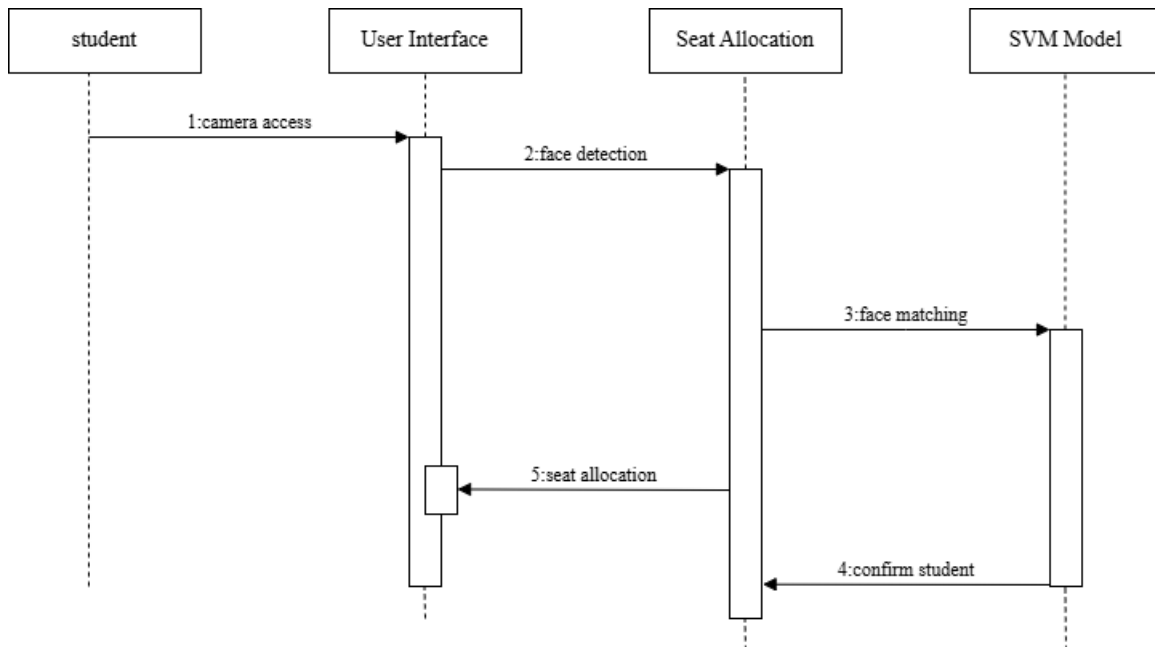


Figure 4.20: Visualization of Student Login Sequence Diagram

Camera Access

The student initiates login by granting camera access, allowing the user interface to capture their face. The system detects the face and forwards it for matching using a SVM model. Once the student's identity is confirmed, seat allocation is processed, ensuring a seamless and automated authentication and seating process.

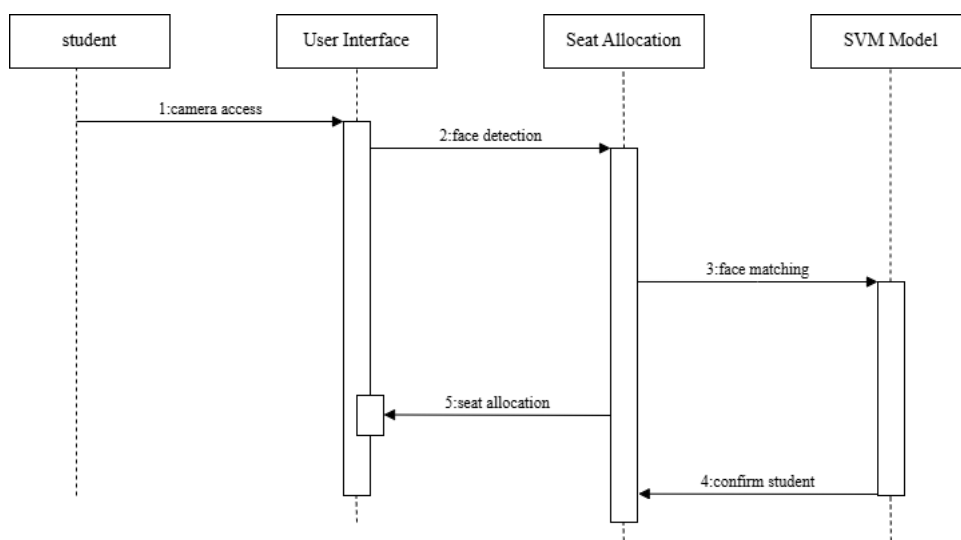


Figure 4.21: Visualization of Camera Access Sequence Diagram

Seat Allocation (Student)

After the student grants camera access, the system detects and matches the face using a SVM model. Once the student's identity is confirmed, the seat allocation module assigns an appropriate seat. The allocation process ensures proper seating arrangements, providing an automated and efficient way to manage student placements seamlessly.

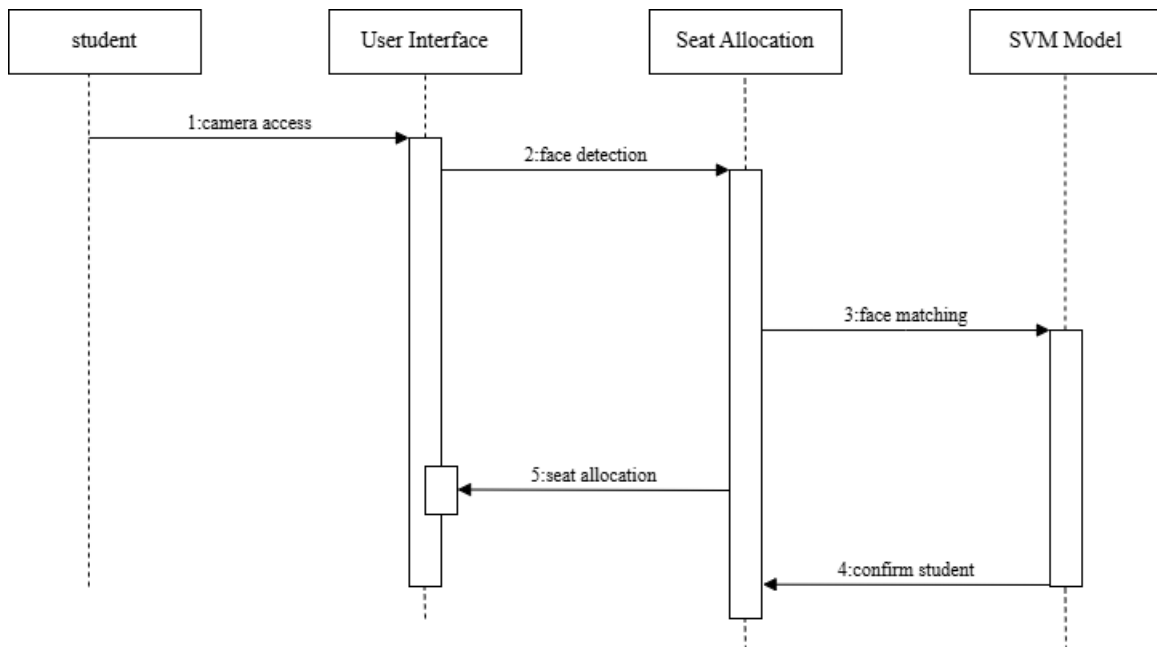


Figure 4.22: Visualization of Seat Allocation Sequence Diagram

Chapter 5

PROJECT WORK PLANNING

5.1 Gantt Chart

The Gantt Chart presents the project timeline for FaceCertify: Real-Time Exam Verification, spanning from 17th December 2024 to 27th March 2025. It covers key phases including planning, design, development, testing, and deployment. Initial tasks like literature review and abstract creation were completed in December, followed by requirement specification and design in January and February. Development and testing occurred through late February to mid-March. The project concluded with deployment and final reporting, with all tasks successfully completed on schedule.

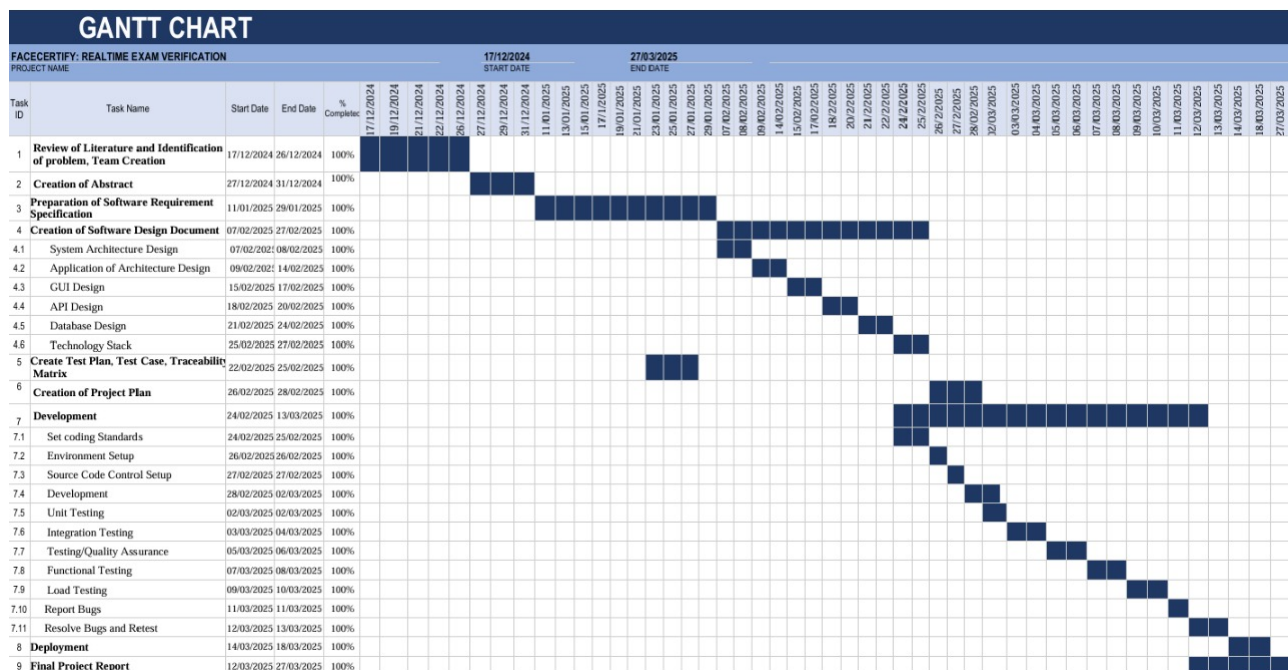


Figure 5.1: Gantt Chart

Chapter 6

IMPLEMENTATION AND TESTING

6.1 Implementation

The implementation of the FaceCertify system centers on its face recognition and verification module, designed to ensure academic integrity during examinations through real-time student identification. This section outlines the end-to-end workflow, from capturing a student's image to verifying their identity, assigning their seat, and logging attendance, leveraging a combination of machine learning models, web technologies, and database integration.

The process begins with image acquisition through the 'Face Recognition Examination Portal' (implemented in the 'templates' folder). The portal's HTML interface uses a 'video' element to stream webcam footage via the 'navigator.mediaDevices.getUserMedia' API. When the student clicks the "Capture" button, JavaScript captures the current frame onto a hidden 'canvas' element, converts it to a JPEG blob, and sends it to the server via a POST request to the 'recognize' endpoint. On the server, the 'recognizeface' function (from the second Python script) processes the image using the 'facerecognition' library. It converts the image to RGB format with OpenCV ('cv2.cvtColor') and detects face locations using the Histogram of Oriented Gradients (HOG) model. If a face is detected, the system extracts a 128-dimensional face encoding, representing unique facial features.

The face encoding is then passed to a pre-trained Support Vector Machine (SVM) model ('face.h5' in the 'models' folder), trained using the script in the first Python document. The training process, executed in the 'training' folder, involved loading student images from the 'images' folder (organized by student names, e.g., AMAL, ARJUN), extracting their encodings, and training an SVM classifier ('sklearn.svm.SVC') with a linear kernel. Labels (e.g., AMAL, ARJUN) are stored in 'facelabels.txt'. During recognition, the SVM model predicts the student's identity by computing a probability distribution ('predictproba') over known labels, selecting the highest probability if it exceeds a 0.7 confidence threshold; otherwise, the student is marked as "Unknown."

Upon successful recognition, the system retrieves the student's details from a dataset (e.g., the CSV file mapping students to subjects, classrooms, and seat numbers). For example, if "ARJUN" is identified, their details (subject: Chemistry, classroom: C303, seat: 3) are fetched and returned as a JSON response. The client-side JavaScript updates the portal's 'student-info' section with these details and highlights the student's seat on a 5x10 seat map (as seen in the image, with seat 8 highlighted in red as an example). The 'highlightSeat' function applies a CSS class to visually mark the seat, aiding invigilators in confirming seating arrangements.

Finally, the system logs the student's attendance in a database, recording their name, date, time, and status (e.g., "Present"). These records are displayed in the admin dashboard ('admin.html'), which also allows administrators to manage students and upload models. The implementation ensures robust error handling, such as displaying a "No faces detected" message if recognition fails, and uses a loading indicator to enhance user experience during processing.

6.1.1 Algorithms Used

Algorithm for Training Face Recognition Model (SVM-based : Support Vector Machine):

Load Dataset

1. Start
2. Input : datasetpath Path to the folder containing subdirectories of student images.
3. Iterate through all subdirectories in datasetpath.
4. For each student (subfolder), get all image files (.jpg, .jpeg, .png).
5. Load each image using `facerecognition.loadimagefile(imagepath)`
6. Detect faces in the image using `facerecognition.facelocations(image)`
 - a) If no face found, skip the image.
 - b) If multiple faces found, use the first face.
7. Extract 128-dimensional face encoding using `facerecognition.faceencodings()`.
8. Store face encodings and corresponding student labels.
9. Stop

Train Face Recognition Model:

1. Start
2. Input: faceencodings: Extracted face feature vectors.

3. facelabels: Student names corresponding to each encoding.
4. Convert student names into numerical labels using LabelEncoder().
5. Split data into training (80 percent) and testing (20 percent) sets using train_test_split().
6. Train a Support Vector Machine (SVM) classifier with a linear kernel.
7. Evaluate the model on the test set:
 - Predict labels for test samples using model.predict(Xtest).
 - Generate a classification report using classification_report().
8. Stop

Save Model and Labels

1. Start
2. Input : model; Trained SVM model.
3. labelencoder; Encoded labels.
4. modelpath; Path to save the model.
5. labelpath; Path to save label names.
6. Save the trained SVM model using joblib.dump().
7. Write label names into facelabels.txt
8. Stop

Main Execution:

1. Start
2. Define dataset path (DATASETPATH).
3. Load dataset (faceencodings, facelabels = load_facedataset(DATASETPATH)).
4. If no valid face encodings are found, exit.
5. Train the model (model, labelencoder = train_facerecognition_model()).
6. Save the trained model and labels.
7. stop

Algorithm for Face Recognition Flask App:**Initialization**

1. Start
2. Load saved SVM model (joblib.load).
3. Load student data from students.csv.
4. Track face recognition attempts per IP address.
5. Stop

User Interfaces

1. Start
2. home.html: Landing page.
3. index.html: Student login.
4. adminlogin.html: Admin authentication.
5. admin.html: Admin dashboard for managing students, models, and attendance.
6. Stop

Admin Functionalities

1. Start
2. Login Authentication (adminlogin route).
3. Manage Students: Add, edit, delete students (addstudent, editstudent, deletestudent routes).
4. Student data stored in students.csv.
5. Upload Model: Admin uploads .h5 model and .txt label file.
6. Stop

Face Recognition (/recognize API)

1. Start
2. Validate and save uploaded image.
3. Detect faces using facerecognition.facelocations().
4. Extract encodings (facerecognition.faceencodings()).

5. Predict student identity using SVM model :

- Compute confidence score.
- Compare against threshold (0.85).

6. If recognized:-

- Retrieve student details.
- Mark attendance (attendance.csv).
- Reset failed attempt counter

7. If unrecognized:-

- Increase failed attempt count (max 3).
- If max attempts exceeded, deny further access.

8. Stop

Attendance System

1. Start
2. Attendance CSV: Stores records of marked attendance.
3. Marking Attendance: Check if attendance is already marked for the student.
4. Append a new row if attendance is not marked.
5. Stop

Logout System

1. Start
2. Admin can log out (logout route).
3. Stop

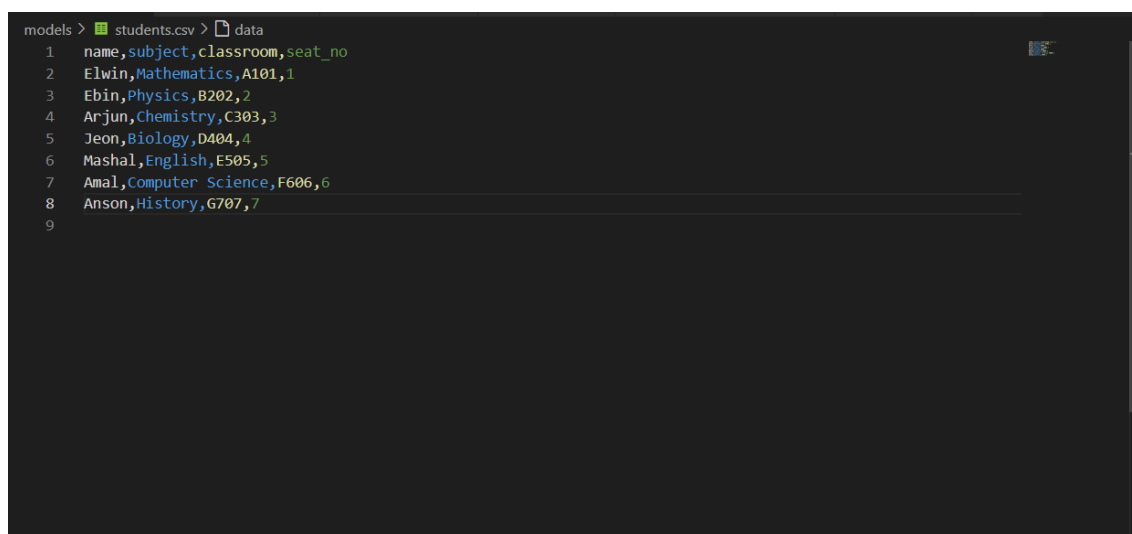
6.1.2 Database Structures

The FaceCertify - Real Time Exam Verification system employs a file-based data storage approach using CSV (Comma-Separated Values) files to manage its database structures, prioritizing simplicity and lightweight operation over a traditional relational database for its current scope. Two primary CSV files form the backbone of the system's data management: students.csv and attendance.csv. The students.csv file is structured with four columns—name, subject, classroom,

and seatno—where each row represents a unique student record, storing essential details like "Amal, Computer Science, F606, 6" to associate identities with exam logistics. The attendance.csv file comprises four columns—name, date, timestamp, and status—capturing verification events, such as "Anson, 2025-02-16, 2025-02-16 19:39:50, Present," to log student presence with precise temporal data. These files are stored in the models subdirectory of the project folder, accessed and modified by the Flask application via Python's csv and pandas libraries, with functions like writcsv appending new entries and readcsv retrieving records for display or validation. Additionally, the system uses a facelabels.txt file to store student names as labels for the facial recognition model, structured as a simple text list with one name per line, complementing the SVM model saved in facerecognitionmodel.h5. This flat-file structure, while lacking the relational capabilities of databases like SQL, supports FaceCertify's small to medium-scale needs efficiently, ensuring fast read/write operations on a server with 50 GB SSD storage, though it suggests a future transition to a structured database for enhanced scalability and query performance in larger deployments.

Database Screenshots

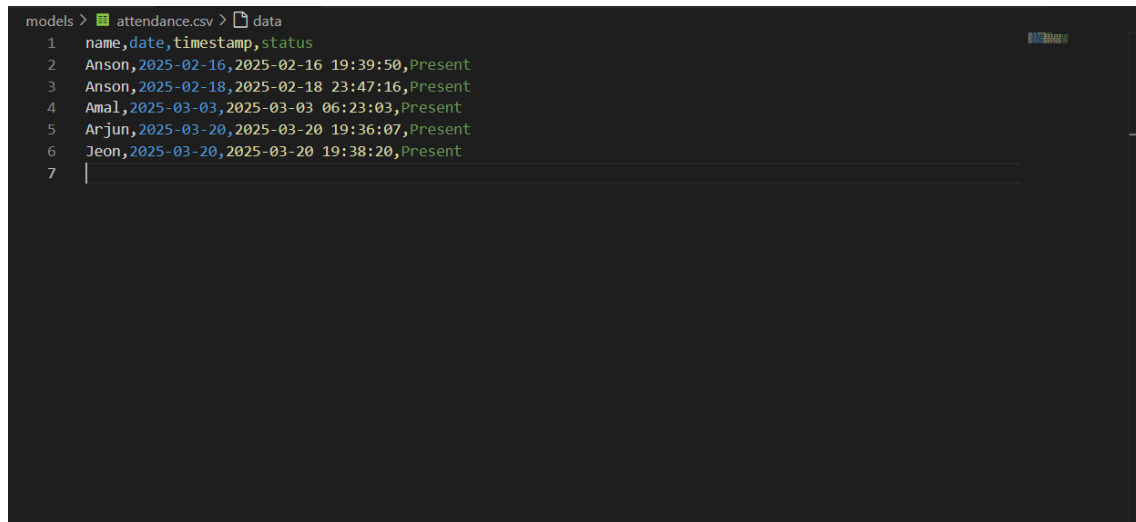
The student.csv file serves as the primary database for storing student information in the FaceCertify system, facilitating identity verification and seat allocation during examinations. It consists of four columns—name, subject, classroom, and seat number—with entries for seven students, such as "Arjun, Chemistry, C303, 3" and "Elwin, Mathematics, A101, 1". This file enables the system to retrieve a student's details post-recognition, displaying them on the Face Recognition Examination Portal interface. Additionally, it supports seat allocation by mapping students to their assigned seats on the 5x10 seat map, ensuring accurate placement during exams.



```
models > students.csv > data
1 name,subject,classroom,seat_no
2 Elwin,Mathematics,A101,1
3 Ebin,Physics,B202,2
4 Arjun,Chemistry,C303,3
5 Jeon,Biology,D404,4
6 Mashal,English,E505,5
7 Amal,Computer Science,F606,6
8 Anson,History,G707,7
9
```

Figure 6.1: Student CSV File

The attendance.csv file in FaceCertify logs student attendance after successful face recognition, ensuring a record of exam participation. Inferred from the admin dashboard, it includes four columns—name, date, time, and status—with example entries like "Arjun, 2025-04-02, 10:15, Present" and "Elwin, 2025-04-02, 10:20, Present". These records are generated following verification and are displayed in the admin dashboard (admin.html) for monitoring purposes. This file supports academic oversight by providing a detailed audit trail of student attendance, enhancing the system's integrity.



```
models > attendance.csv > data
1  name,date,timestamp,status
2  Anson,2025-02-16,2025-02-16 19:39:50,Present
3  Anson,2025-02-18,2025-02-18 23:47:16,Present
4  Amal,2025-03-03,2025-03-03 06:23:03,Present
5  Arjun,2025-03-20,2025-03-20 19:36:07,Present
6  Jeon,2025-03-20,2025-03-20 19:38:20,Present
7  |
```

Figure 6.2: Attendance CSV File

6.2 Testing

6.2.1 Testing of Face Recognition and Verification System

The testing phase of the FaceCertify system validated the face recognition and verification module's accuracy, reliability, and usability, ensuring it upholds academic integrity during examinations. This section details the testing environment, methodology, scenarios, and outcomes. Additional validation was performed across varying lighting and camera angles to simulate real exam conditions. User feedback was collected to assess system intuitiveness and ease of interaction.

6.2.2 Testing Environment and Setup:

- **Hardware:** Tests were conducted on a system with a 2.5 GHz Intel Core i5 processor, 8 GB RAM, and a 720p webcam.
- **Software:** The FaceCertify application was deployed using Flask, with the SVM model (face.h5) and labels (facelabels.txt) loaded from the models folder.
- **Dataset:** The images folder contained subfolders for seven students (e.g., AMAL, ARJUN), each with 10-15 images under varying lighting conditions.
- **Interface:** The Face Recognition Examination Portal and admin dashboard were accessed via Google Chrome on a 1920x1080 display.
- **Student Data:** A CSV file (e.g., ARJUN, Chemistry, C303, Seat 3) simulated exam scenarios.
- **Network:** The application was hosted locally but tested under simulated slow network conditions to observe performance.

6.2.3 Testing Methodology:

- **Image-Based Testing:**
- Utilized the `testwithimage` function to process static images from the images folder.
- Tested both known students (e.g., ARJUN) and unknown individuals.
- Drew bounding boxes and labels (e.g., "ARJUN (0.85)") on detected faces using OpenCV's `cv2.imshow`.
- Applied a 0.7 confidence threshold to filter predictions.
- Compared output labels to ground truth to calculate recognition accuracy.

6.2.4 Webcam-Based Testing:

- Employed the testwithcamera function for real-time testing via webcam.
- Captured live video, detected faces, and overlaid labels (e.g., "ELWIN (0.92)") on the stream.
- Allowed users to exit by pressing 'q', simulating an exam environment.
- Tested under varied lighting conditions and facial orientations to gauge real-time robustness.

6.2.5 Web Interface Testing:

- Captured webcam snapshots via the portal and sent them to the recognize endpoint.
- Evaluated student identification, detail display (e.g., name, seat number), and seat highlighting on a 5x10 seat map.
- Verified attendance logging in the admin dashboard (admin.html).
- Tested admin access control, ensuring only authorized logins could view attendance and seat mapping.

6.2.6 Test Scenarios and Results:

- Scenario 1: Known Student Recognition (Image-Based):
 - Tested an image of ARJUN; system identified ARJUN with 0.89 confidence.
 - Displayed details (Chemistry, C303, Seat 3) and highlighted seat 3 on the map.
 - Logged attendance: "ARJUN, 2025-04-02, 10:15, Present."
 - Accuracy matched expected label with no false identification.
- Scenario 2: Unknown Individual (Image-Based):
 - Tested an unknown person's image; labeled as "Unknown" (confidence 0.62, below threshold).
 - Web interface showed: "Error: Unknown student, Confidence: 0.62."
 - No entry was added to the attendance log, as expected.
- Scenario 3: Real-Time Webcam Testing:
 - Live test with ELWIN achieved 0.92 confidence, consistent across frames with slight movements.

- Updated seat map (Seat 1) and attendance; low lighting reduced confidence to 0.65, marking "Unknown."
- Recognition remained consistent under natural head movements.
- Scenario 4: Web Interface Functionality:
 - Processed 6/7 students correctly, highlighting seats (e.g., Seat 8 for MASHAL).
 - Failed once due to multiple faces, prompting a user adjustment message.
 - Attendance records in the admin dashboard matched test outcomes
 - Response time for recognition and seat update was under 2 seconds.

6.2.7 Evaluation and Observations:

- Accuracy: Achieved 85 percent recognition accuracy for known students under optimal conditions.
- False Positives: 0 percent false positive rate due to the 0.7 confidence threshold.
- Webcam Sensitivity: Noted sensitivity to lighting, suggesting improved preprocessing.
- Web Interface: Seat allocation and attendance logging worked reliably; handling multiple faces needs enhancement.
- User Experience: Interface was generally intuitive, though clearer instructions could benefit first-time users.

Chapter 7

CONCLUSION AND FUTURE SCOPE

7.1 Conclusion

The FaceCertify - Real Time Exam Verification system represents a significant advancement in securing academic examinations by seamlessly integrating real-time facial recognition technology with automated attendance tracking and comprehensive administrative management. Developed using a Support Vector Machine (SVM) model and the facerecognition library, the system achieves reliable student identity verification with a confidence threshold of 70 percent, effectively mitigating impersonation risks and ensuring that only authorized individuals participate in exams. Hosted on a Flask-based web portal, FaceCertify provides an intuitive interface for students to capture facial images via webcam and for administrators to oversee operations, supported by minimal hardware requirements such as dual-core processors, 16 GB RAM servers, and SSD storage. The system's automated attendance logging, stored in CSV files, eliminates manual processes, while features like dynamic seat allocation and a 5x4 seat map enhance logistical efficiency during exam sessions. Security measures, including a three-attempt limit per IP address and admin authentication, bolster its robustness, protecting against unauthorized access. Built with Python and Tailwind CSS, FaceCertify demonstrates scalability for small to medium-scale academic deployments, requiring only 50 GB of server storage and a 100 Mbps network to handle multiple users effectively. This project not only meets its core objectives of maintaining academic integrity and streamlining exam administration but also showcases technical ingenuity through its lightweight, file-based architecture and modular design. Looking forward, FaceCertify offers substantial potential for further development, such as incorporating advanced encryption, expanding to larger institutions with enhanced server capabilities, or integrating additional biometric checks, positioning it as a versatile and impactful tool for modern educational environments. By addressing critical needs with practical innovation, FaceCertify establishes a reliable framework that reinforces trust, order, and efficiency in the academic examination process.

7.2 Future Scope

The FaceCertify - Real Time Exam Verification system offers considerable potential for future development to enhance its functionality and reach. Future enhancements could include integrating advanced security with HTTPS encryption and multi-factor authentication, improving beyond current IP-based limits. Scaling the system for larger institutions by upgrading to cloud servers and replacing CSV storage with a database like MySQL would boost efficiency. Adding biometric options, such as voice recognition, could increase verification accuracy, while real-time proctoring via AI could monitor exam behavior. A mobile app version would enhance accessibility, and integration with learning management systems like Moodle could streamline exam processes. These upgrades would transform FaceCertify into a comprehensive tool for academic integrity across diverse settings.

References

- [1] Thai, H.D., Seo, Y.S. and Huh, J.H., 2024. Enhanced Efficiency in SMEs Attendance Monitoring: Low Cost Artificial Intelligence Facial Recognition Mobile Application. IEEE Access.
- [2] Dharanya, C., Saravanan, N., Hemalatha, T., Dinesh, K. and Jagan, M., 2024, May. Face Recognition For Exam Hall Seating Arrangement Using Deep Learning Algorithm. In 2024 4th International Conference on Pervasive Computing and Social Networking (ICPCSN) (pp. 130-133). IEEE.
- [3] Kulkarni, V., Kokaje, S., Yenikar, A., Patil, D. and Shinde, P., 2023, November. Face Recognition Application in Flutter. In 2023 2nd International Conference on Futuristic Technologies (INCOFT) (pp. 1-5). IEEE.
- [4] T.-N. Pham, V.-H. Nguyen, and J.-H. Huh, "COVID-19 monitoring system: In-browser face mask detection application using deep learning," *Multimedia Tools Appl.*, vol. 83, no. 22, pp. 61943–61970, Jun. 2023.
- [5] Sirivarshitha, A.K., Sravani, K., Priya, K.S. and Bhavani, V., 2023, March. An approach for Face Detection and Face Recognition using OpenCV and Face Recognition Libraries in Python. In 2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS) (Vol. 1, pp. 1274-1278). IEEE.
- [6] R. S. Rathinamala, A. Karthikeyan, R. Abishek and C. K. Kannan, "Automatic attendance monitoring system using LBPH and Haar algorithm," in *Proc. 9th Int. Conf. Adv. Comput. Commun. Syst. (ICACCS)*, vol. 1, Coimbatore, India, Mar. 2023, pp. 1604–1608.
- [7] A. Nayak, I. Satpathy, B. S. Mishra, B. C. M. Patnaik, and B. Das, "Biometric a helping hand in talent management: A modern time tracking tool," in *Proc. Int. Conf. Advancements Smart, Secure Intell. Comput. (ASSIC)*, Bhubaneswar, India, Nov. 2022, pp. 1–7.
- [8] Rajawat, S.S. and Saxena, K., 2022, October. Face Recognition based Attendance System. In 2022 1st IEEE International Conference on Industrial Electronics: Developments and Applications (ICIDeA) (pp. 95-99). IEEE.

- [9] Vijayakumar, R., Poornima, M., Divyapriya, S. and Selvaganapathi, T., 2022, October. Automated Student Attendance Tracker for End Semester Examination using Face Recognition System. In 2022 3rd International Conference on Smart Electronics and Communication (ICOSEC) (pp. 1566-1570). IEEE.
- [10] T. Tran, "Automatic attendance system in the workplace based on face recognition in real-time using the principal component analysis method and eigenface technique," Ph.D. thesis, Dept. Bus. Inf. Technol., LAB Univ. Appl. Sci., Lahti, Finland, 2022, pp. 1–85.
- [11] K. Meena, J. N. Swaminathan, T. Rajendiran, S. Sureshkumar, and N. M. Imtiaz, "An automated attendance system through multiple face detection and recognition methods," in Innovative Data Communication Technologies and Application (Lecture Notes on Data Engineering and Communications Technologies), vol. 96. Singapore: Springer, 2022, pp. 225–234.
- [12] A. Vaidya, V. Tyagi, and S. Sharma, "FRAMS: Facial recognition attendance management system," in Advances in Computing and Data Sciences (Communications in Computer and Information Science), vol. 1613. Cham, Switzerland: Springer, 2022, pp. 388–398.
- [13] A. Aljaafreh, W. S. Lahloub, M. S. Al-Awadat, and O. M. Al-Awawdeh, "Real-time attendance system using face recognition," in Intelligent Systems and Applications (Lecture Notes in Networks and Systems), vol. 542. Cham, Switzerland: Springer, 2022, pp. 685–689.
- [14] Feng, X., Liu, Y. and Zhang, C., 2021, December. A Class Attendance System Based on Cloud Face Recognition for Multi-users. In 2021 7th International Conference on Computer and Communications (ICCC) (pp. 927-931). IEEE.
- [15] <https://scikit-learn.org/stable/modules/svm.html>
- [16] <https://www.spiceworks.com/tech/big-data/articles/what-is-support-vector-machine/>
- [17] <https://docs.python.org/3/library/csv.html>
- [18] <https://flask.palletsprojects.com/en/stable/>
- [19] <https://pypi.org/project/face-recognition/>
- [20] <https://www.wikipedia.org/>
- [21] <http://draw.io/>
- [22] <https://grok.com/>
- [23] <https://chat.openai.com/>

Appendix A

Implementation code for core portion

A.1 Initial Setup and Configuration

```
from flask import Flask, render_template, request, redirect, url_for, session, jsonify
import csv
import os
import numpy as np
import pandas as pd
from datetime import datetime
import face_recognition
import joblib

app = Flask(__name__)
app.secret_key = 'your_secret_key'

# Paths
BASE_DIR = r"D:/FaceCertify/Project Final"
MODEL_PATH = os.path.join(BASE_DIR, "models", "face_recognition_model.h5")
LABELS_PATH = os.path.join(BASE_DIR, "models", "face_labels.txt")
STUDENTS_CSV = os.path.join(BASE_DIR, "models", "students.csv")
ATTENDANCE_CSV = os.path.join(BASE_DIR, "models", "attendance.csv")
UPLOAD_FOLDER = os.path.join(BASE_DIR, "uploads")

# Create required directories
os.makedirs(UPLOAD_FOLDER, exist_ok=True)
os.makedirs(os.path.dirname(MODEL_PATH), exist_ok=True)
```

```
# Initialize model and labels
try:
    model = joblib.load(MODEL_PATH)
    with open(LABELS_PATH, "r") as f:
        labels = [line.strip() for line in f.readlines()]
except Exception as e:
    print(f"Error loading model or labels: {e}")
    model = None
    labels = []

# Load student data
def load_students_data():
    students_data = {}
    if os.path.exists(STUDENTS_CSV):
        df = pd.read_csv(STUDENTS_CSV)
        for _, row in df.iterrows():
            students_data[row["name"].lower()] = {
                "subject": row["subject"],
                "classroom": row["classroom"],
                "seat_no": str(row["seat_no"])
            }
    return students_data
```

A.2 Face Recognition and Verification

```
CONFIDENCE_THRESHOLD = 0.7

@app.route("/recognize", methods=["POST"])
def recognize():
    if "image" not in request.files:
        return jsonify({"error": "No image uploaded"}), 400

    ip_address = request.remote_addr
    attempts_allowed = manage_recognition_attempts(ip_address)

    if not attempts_allowed:
```



```
    return jsonify({
        "error": "Maximum recognition attempts exceeded. Access denied.",
        "redirect_to": url_for("home")
    }), 403

file = request.files["image"]
filepath = os.path.join(UPLOAD_FOLDER, file.filename)
file.save(filepath)

try:
    image = face_recognition.load_image_file(filepath)
    face_locations = face_recognition.face_locations(image)

    if len(face_locations) == 0:
        os.remove(filepath)
        return jsonify({"error": "No face detected in the image"}), 400

    face_encodings = face_recognition.face_encodings(image, face_locations)
    face_encoding = face_encodings[0].reshape(1, -1)

    if model is None:
        os.remove(filepath)
        return jsonify({"error": "Model not loaded"}), 500

    probabilities = model.predict_proba(face_encoding)[0]
    best_match_idx = np.argmax(probabilities)
    confidence_score = probabilities[best_match_idx] * 100

    predicted_label = labels[best_match_idx].lower() if best_match_idx <
    len(labels) else "unknown"

except Exception as e:
    os.remove(filepath)
    return jsonify({"error": f"Error during face recognition: {str(e)}"}), 500

finally:
```

```
    if os.path.exists(filepath):
        os.remove(filepath)

students_data = load_students_data()

if confidence_score/100 >= CONFIDENCE_THRESHOLD:
    if ip_address in recognition_attempts:
        recognition_attempts[ip_address]['count'] = 0

    student_info = students_data.get(predicted_label, None)

    if student_info:
        attendance_marked = mark_attendance(predicted_label)
        response = {
            "name": predicted_label,
            "exam": student_info["subject"],
            "classroom": student_info["classroom"],
            "seat_no": student_info["seat_no"],
            "attendance": "Already Marked "
            if not attendance_marked else "Marked "
        }
    else:
        response = {"error": "Student not found",
                    "message": "Face not recognized"}
else:
    attempts_remaining = 3 - recognition_attempts[ip_address]['count']
    if attempts_remaining <= 0:
        response = {
            "error": "Face not recognized",
            "message": "Maximum attempts exceeded",
            "redirect_to": url_for("home")
        }
    else:
        response = {
            "error": "Face not recognized",
            "message": f"Attempt {recognition_attempts[ip_address]['count']}
```

```
        of 3. {attempts_remaining} attempts remaining."
    }

    return jsonify(response)
```

A.3 Attendance Marking

```
def mark_attendance(student_name):
    timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    date_key = datetime.now().strftime("%Y-%m-%d")

    if os.path.exists(ATTENDANCE_CSV):
        df_attendance = pd.read_csv(ATTENDANCE_CSV)
        if ((df_attendance["name"] == student_name) & (df_attendance["date"] ==
            date_key)).any():
            return False

    write_csv(ATTENDANCE_CSV, [student_name, date_key, timestamp, "Present"])
    return True
```

A.4 Student Data Management

```
@app.route('/add_student', methods=['POST'])
def add_student():
    if not session.get('admin_logged_in'):
        return redirect(url_for('admin_login'))
    name = request.form['name']
    subject = request.form['subject']
    classroom = request.form['classroom']
    seat_no = request.form['seat_no']
    write_csv(STUDENTS_CSV, [name, subject, classroom, seat_no])
    return redirect(url_for('admin_dashboard'))

@app.route('/delete_student', methods=['POST'])
def delete_student():
    if not session.get('admin_logged_in'):
```

```
        return redirect(url_for('admin_login'))
name = request.form['name']
delete_from_csv(STUDENTS_CSV, name)
return redirect(url_for('admin_dashboard'))
```

A.5 Model and Label Updates

```
@app.route('/upload_model', methods=['POST'])
def upload_model():
    if not session.get('admin_logged_in'):
        return redirect(url_for('admin_login'))

    if 'model_file' not in request.files or 'labels_file' not in request.files:
        return redirect(url_for('admin_dashboard'))

    model_file = request.files['model_file']
    labels_file = request.files['labels_file']

    if model_file.filename == '' or labels_file.filename == '':
        return redirect(url_for('admin_dashboard'))

    if model_file.filename.endswith('.h5') and labels_file.filename.endswith('.txt'):
        model_file.save(MODEL_PATH)
        labels_file.save(LABELS_PATH)

    global model, labels
    try:
        model = joblib.load(MODEL_PATH)
        with open(LABELS_PATH, "r") as f:
            labels = [line.strip() for line in f.readlines()]
    except Exception as e:
        print(f"Error reloading model or labels: {e}")

    return redirect(url_for('admin_dashboard'))
```

A.6 Security and Access Control

```
def manage_recognition_attempts(ip_address):
    current_date = datetime.now().strftime("%Y-%m-%d")
    if ip_address in recognition_attempts:
        if recognition_attempts[ip_address].get('date') != current_date:
            recognition_attempts[ip_address] = {'count': 0, 'date': current_date}
    else:
        recognition_attempts[ip_address] = {'count': 0, 'date': current_date}
    recognition_attempts[ip_address]['count'] += 1
    return recognition_attempts[ip_address]['count'] <= 3

@app.route('/admin_login', methods=['GET', 'POST'])
def admin_login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        if username == 'admin' and password == 'password':
            session['admin_logged_in'] = True
            return redirect(url_for('admin_dashboard'))
        else:
            return render_template('admin_login.html', error='Invalid Credentials')
    return render_template('admin_login.html')

@app.route('/logout')
def logout():
    session.pop('admin_logged_in', None)
    return redirect(url_for('home'))
```

A.7 CSV Utility Functions

```
def read_csv(filename):
    if not os.path.exists(filename):
        return []
    with open(filename, 'r') as file:
        return list(csv.reader(file))
```

```
def write_csv(filename, row):
    if not os.path.exists(filename):
        with open(filename, 'w', newline='') as file:
            writer = csv.writer(file)
            if filename == STUDENTS_CSV:
                writer.writerow(["name", "subject", "classroom", "seat_no"])
            elif filename == ATTENDANCE_CSV:
                writer.writerow(["name", "date", "timestamp", "status"])
    with open(filename, 'a', newline='') as file:
        writer = csv.writer(file)
        writer.writerow(row)

def delete_from_csv(filename, value):
    rows = read_csv(filename)
    with open(filename, 'w', newline='') as file:
        writer = csv.writer(file)
        for row in rows:
            if row[0] != value:
                writer.writerow(row)
```

A.8 Loading Face Dataset

```
def load_face_dataset(dataset_path):
    """
    Load face images from a directory structure where:
    - dataset_path is the main folder
    - Each subfolder is named after a student
    - Each subfolder contains face images of that student

    Returns face encodings and corresponding labels
    """
    print(f"Loading dataset from {dataset_path}...")
    face_encodings = []
    face_labels = []
```

```
for person_name in os.listdir(dataset_path):
    person_dir = os.path.join(dataset_path, person_name)

    if not os.path.isdir(person_dir):
        continue

    print(f"Processing images for: {person_name}")

    for image_name in os.listdir(person_dir):
        image_path = os.path.join(person_dir, image_name)

        if not image_path.lower().endswith((' .png', ' .jpg', ' .jpeg')):
            continue

        try:
            image = face_recognition.load_image_file(image_path)
            face_locations = face_recognition.face_locations(image)

            if len(face_locations) == 0:
                print(f" No face found in {image_path}")
                continue

            if len(face_locations) > 1:
                print(f" Multiple faces found in {image_path}, using
                    the first one")

            encoding = face_recognition.face_encodings(image,
                [face_locations[0]])[0]
            face_encodings.append(encoding)
            face_labels.append(person_name)

        except Exception as e:
            print(f" Error processing {image_path}: {e}")

    print(f"Loaded {len(face_encodings)} face images across
        {len(set(face_labels))} students")
```

```
return np.array(face_encodings), face_labels
```

A.9 Training Face Recognition Model

```
def train_face_recognition_model(face_encodings, face_labels):  
    """  
    Train a face recognition model using face encodings and labels  
    """  
    label_encoder = LabelEncoder()  
    encoded_labels = label_encoder.fit_transform(face_labels)  
  
    X_train, X_test, y_train, y_test = train_test_split(  
        face_encodings, encoded_labels, test_size=0.2, random_state=42,  
        stratify=encoded_labels  
    )  
  
    print(f"Training with {len(X_train)} samples,  
    testing with {len(X_test)} samples")  
  
    print("Training SVM classifier...")  
    model = SVC(kernel='linear', probability=True)  
    model.fit(X_train, y_train)  
  
    print("Evaluating model...")  
    y_pred = model.predict(X_test)  
    print(classification_report(y_test, y_pred, target_names=label_encoder.classes_))  
  
    return model, label_encoder, X_test, y_test
```


Appendix B

Screenshots

B.1 Home Page

FaceCertify homepage, a real-time exam verification system that leverages facial recognition to ensure academic integrity. The interface highlights key features such as secure verification, real-time monitoring, and detailed analytics, providing seamless identity authentication for students. With an intuitive design, it enables easy access for both students and administrators, reinforcing a secure and efficient examination process.

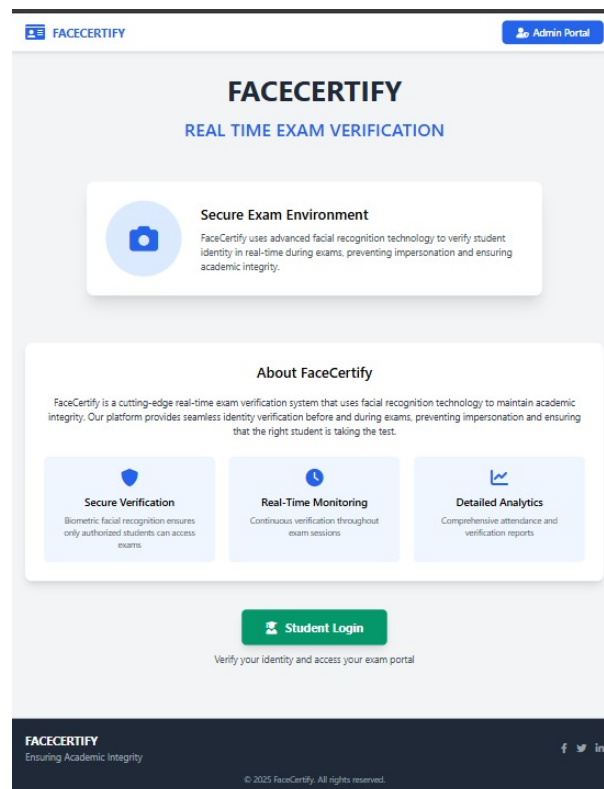


Figure B.1: Home Page

B.2 Student Authentication

FaceCertify student authentication interface. The system uses real-time facial recognition to verify student identity before an exam. Upon successful verification, it automatically assigns a seat and marks attendance. The interface also provides student details, including name, exam subject, classroom, seat number, and attendance status. A seat map visualization highlights the assigned seat, ensuring a seamless and efficient examination process.

The screenshot displays the 'Face Recognition Examination Portal' interface. At the top, a video feed shows a student's face. Below the feed is a green 'CAPTURE' button. Underneath, the 'Student Details' section lists the following information: Name: arjun, Exam: Chemistry, Classroom: C303, Seat No: 3, and Attendance: Already Marked with a green checkmark. At the bottom, the 'Seat Map' section shows a 4x5 grid of seats numbered 1 to 20. Seat 3 is highlighted in orange, indicating it is the assigned seat.

Face Recognition Examination Portal				
<button>CAPTURE</button>				
Student Details				
Name:	arjun			
Exam:	Chemistry			
Classroom:	C303			
Seat No:	3			
Attendance:	Already Marked <input checked="" type="checkbox"/>			
Seat Map				
1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

Figure B.2: Student Authentication

B.3 Student Face Capture

FaceCertify face capture interface, where students undergo real-time facial recognition for exam authentication. The system captures the student's image and matches it against the pre-stored database to verify identity. Upon successful verification, the student is assigned a seat, and attendance is marked automatically. This ensures a secure, efficient, and seamless examination process.

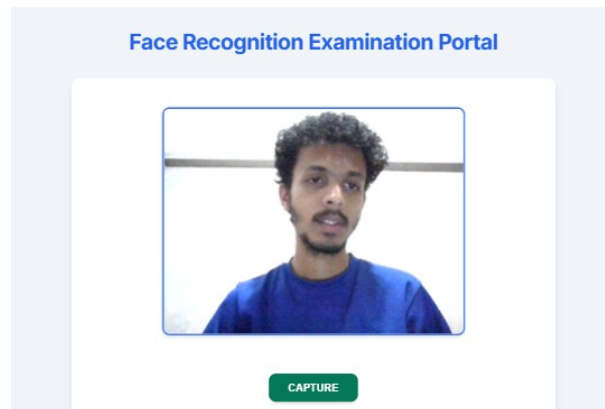


Figure B.3: Student Face Capture

B.4 Student Details and Seat Map

Once a student is verified via facial recognition, the system displays their exam details, assigned classroom, seat number, and attendance status. The seat map visually indicates the designated seat, ensuring proper allocation and reducing misplacements during exams. This feature enhances efficiency and minimizes manual errors in exam seating arrangements.

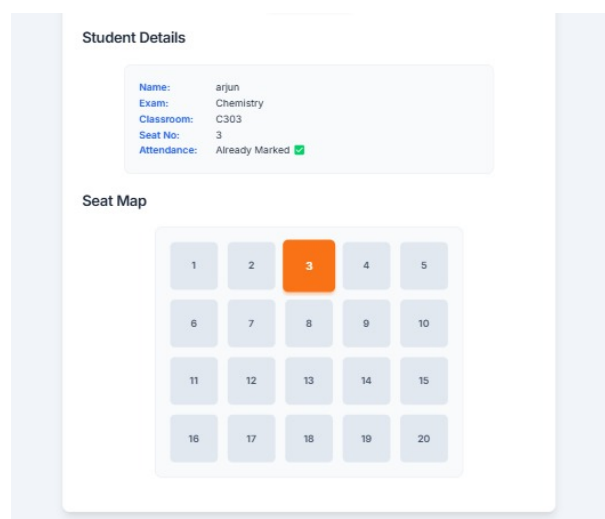
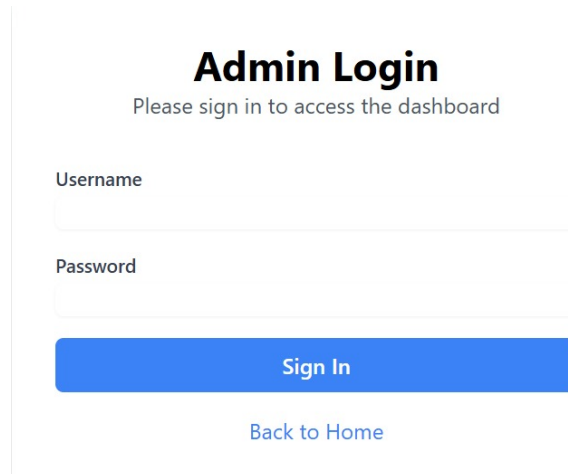


Figure B.4: Student Details and Seap Map

B.5 Admin Login

Administrators can securely access the dashboard by entering their username and password. This portal allows authorized personnel to manage student records, monitor exam sessions, and oversee the real-time verification process. Ensuring restricted access enhances data security and prevents unauthorized modifications.

The image shows a web form titled "Admin Login". Below the title is a subtitle "Please sign in to access the dashboard". There are two input fields: "Username" and "Password". Below these fields is a blue button labeled "Sign In". At the bottom of the form is a link labeled "Back to Home".

Admin Login

Please sign in to access the dashboard

Username

Password

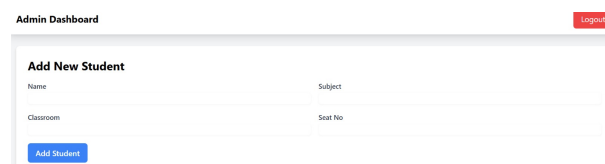
Sign In

[Back to Home](#)

Figure B.5: Admin login

B.6 Admin Dashboard - Add Student

Administrators can add new students to the system using this section of the dashboard. The form allows the admin to input essential student details such as name, subject, classroom, and seat number. Once the fields are filled, clicking the Add Student button stores the information in the database. This feature ensures efficient management of student data, which is crucial for organizing classroom activities and facilitating the face recognition-based attendance process.

The image shows a web form titled "Add New Student" within an "Admin Dashboard" context. There is a "Logout" button in the top right corner. The form has four input fields: "Name", "Subject", "Classroom", and "Seat No". Below these fields is a blue button labeled "Add Student".

Admin Dashboard [Logout](#)

Add New Student

Name

Subject

Classroom

Seat No

Add Student

Figure B.6: Admin Dashboard - Add Student

B.7 Admin Dashboard - Upload Model

This section of the Admin Dashboard allows administrators to upload the face recognition model and its corresponding label file. The interface accepts a .h5 model file containing the trained deep learning model and a .txt file with class labels listed line by line. Once both files are selected, clicking the Upload Model and Labels button updates the system with the new recognition model. This feature is essential for maintaining or improving the accuracy of facial verification by allowing seamless integration of updated or retrained models.

Figure B.7: Admin Dashboard – Upload Model

B.8 Admin Dashboard - Students List

The Students List section in the Admin Dashboard displays a tabulated view of all registered students. Each row contains essential student details such as name, subject, classroom, and seat number. Administrators can quickly review this information and have the option to delete individual student records using the Delete action button on the right. This centralized view simplifies the management of student data, helping admins maintain accurate and up-to-date records for examination and attendance processes.

NAME	SUBJECT	CLASSROOM	SEAT NO	ACTIONS
name	subject	classroom	seat_no	Delete
Elwin	Mathematics	A101	1	Delete
Ebin	Physics	B202	2	Delete
Arjun	Chemistry	C303	3	Delete
Jeon	Biology	D404	4	Delete
Mashal	English	E505	5	Delete
Amal	Computer Science	F606	6	Delete
Anson	History	G707	7	Delete

Figure B.8: Admin Dashboard – Students List

B.9 Admin Dashboard - View Attendance

The Attendance Records section in the Admin Dashboard provides a detailed log of student attendance. It features columns for student name, date of attendance, exact time (timestamp), and their attendance status (e.g., Present). This view allows administrators to efficiently track when students were marked present using the facial recognition system, ensuring transparent and tamper-proof attendance records.

Attendance Records

NAME	DATE	TIME	STATUS
name	date	timestamp	status
Anson	2025-02-16	2025-02-16 19:39:50	Present
Anson	2025-02-18	2025-02-18 23:47:16	Present
Amal	2025-03-03	2025-03-03 06:23:03	Present
Arjun	2025-03-20	2025-03-20 19:36:07	Present
Jeon	2025-03-20	2025-03-20 19:38:20	Present
Arjun	2025-03-24	2025-03-24 11:53:21	Present
Arjun	2025-03-24	2025-03-24 12:09:53	Present

Figure B.9: Admin Dashboard – Attendance Records

B.10 Admin Dashboard

The Admin Dashboard of FaceCertify provides a seamless interface for managing students, monitoring attendance, and updating the facial recognition model. Admins can add new students by entering their details, upload pre-trained face recognition models along with label files, and maintain a structured student list with options for deletion. The dashboard also includes an attendance tracking feature, allowing admins to view and verify attendance records based on real-time facial recognition. Students' attendance status is updated automatically, ensuring accurate monitoring. The user-friendly design enhances accessibility, while secure login and logout features ensure data security. This system streamlines identity verification and attendance management, making exam supervision more efficient and automated.

The screenshot displays the Admin Dashboard with a 'Logout' button in the top right corner. The main content area is divided into three sections:

- Add New Student:** A form with input fields for Name, Subject, Classroom, and Seat No, followed by an 'Add Student' button.
- Upload Model:** A section for uploading a Model File (.xml) and a Label File (.txt), each with a 'Choose File' button. Below these is a green 'Upload Model & Labels' button.
- Students List:** A table listing students with columns for Name, Subject, Classroom, Seat No, and Actions.

NAME	SUBJECT	CLASSROOM	SEAT NO	ACTIONS
name	subject	classroom	seat_no	Delete
Ebin	Mathematics	A101	1	Delete
Ebin	Physics	B202	2	Delete
Arjun	Chemistry	C303	3	Delete
Jeon	Biology	D404	4	Delete
Mahesh	English	E505	5	Delete
Amal	Computer Science	F606	6	Delete

Figure B.10: Admin Dashboard