**Abstract**

In the evolving landscape of cybersecurity and surveillance, face recognition systems have emerged as a pivotal technology for identity verification and access control. However, as these systems become more widespread, they also introduce new forensic challenges and vulnerabilities that must be understood to ensure secure deployment and post-incident analysis. This project investigates a Face Recognition System from a cyber forensic perspective, focusing on identifying potential vulnerabilities, collecting digital evidence, and ensuring forensic readiness in case of system compromise.

The methodology adopted in this project is rooted in the standard digital forensic process: acquisition, preservation, analysis, and reporting. Key data sources included application code, system logs, and facial data sets. Tools such as Autopsy were used to inspect file-level activity, while static code analysis helped uncover insecure practices, including inadequate encryption of biometric data and poor authentication handling. The project also evaluated how facial recognition algorithms store and match biometric data, and how this data can be reconstructed or recovered during a forensic investigation.

The analysis revealed critical issues such as unencrypted storage of facial images, lack of access logging, and exposure of sensitive user data, all of which could be exploited by malicious actors. By simulating various attack scenarios and capturing their digital footprints, the investigation demonstrates how forensic methods can be applied to trace activity and support incident response.

In conclusion, this project contributes to the field of Cyber Forensics by outlining a structured approach to analyzing face recognition systems. It emphasizes the importance of secure system design and forensic readiness, ensuring that such biometric technologies can be investigated effectively in the event of a breach or misuse.

**CONTENTS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CONTENTS** | | | | **PAGE NO** |
| **CHAPTER 1** | **INTRODUCTION** | | | 4 |
|  | 1.1 | | General Background | 4 |
|  | 1.2 | | Problem Statement | 5 |
|  | 1.3 | | Project Scope | 6 |
|  | 1.4 | | Project Objectives | 7 |
| **CHAPTER 2** | **PROBLEM STATEMENT AND OBJECTIVES** | | | 8 |
|  | 2.1 | | Problem Definition | 8 |
|  | 2.2 | | Project Goals | 9 |
|  | 2.3 | | Research Questions | 10 |
| **CHAPTER 3** | **LITERATURE REVIEW** | | | 12 |
|  | 3.1 | Existing Solutions | | 12 |
|  | 3.2 | Previous Research | | 13 |
|  | 3.3 | Challenges in Cyber Forensics | | 15 |
| **CHAPTER 4** | **METHODOLOGY AND TOOLS** | | | 17 |
|  | 4.1 | | Methodology | 17 |
|  | 4.2 | | Data Collection | 18 |
|  | 4.3 | | Tools and Software | 19 |
|  | 4.4 | | Techniques Used | 20 |
| **CHAPTER 5** | **SYSYEM DESIGN AND ARCHITECTURE** | | | 23 |
|  | 5.1 | | System Overview | 23 |
|  | 5.2 | | Architecture Diagrams | 25 |
| **CHAPTER 6** | **IMPLEMENTATION** | | | 31 |
|  | 6.1 | | Implementation Process | 31 |
|  | 6.2 | | Code Snippets | 35 |
|  | 6.3 | | Challenges and Solutions | 41 |
| **CHAPTER 7** | **RESULTS AND ANALYSIS** | | | 45 |
|  | 7.1 | | Results | 45 |
|  | 7.2 | | Analysis | 45 |
|  | 7.3 | | Screenshots/Logs | 46 |
| **CHAPTER 8** | **CONCLUSION AND FUTURE WORKS** | | | 48 |
|  | 8.1 | | Conclusion | 48 |
|  | 8.2 | | Future Works | 49 |
| **REFERENCES** |  | | | 52 |
| **APPENDICES** |  | | | 54 |

**Chapter 1: Introduction**

**1.1 Background**

Cyber Forensics, also known as digital forensics, is a crucial branch of cybersecurity that focuses on the identification, preservation, extraction, and documentation of digital evidence from electronic devices. It plays a vital role in investigating cybercrimes, system breaches, and unauthorized access to information systems. As organizations increasingly rely on digital platforms for operations, the risk of cyber-attacks has grown, making Cyber Forensics indispensable in ensuring system integrity and supporting legal proceedings.

In the context of this project, a web-based attendance management system has been analyzed from a cyber forensic perspective. These systems, often hosted on local servers using platforms like XAMPP or Bitnami, handle sensitive employee or student data, login credentials, and access logs. If compromised, such systems can be exploited for malicious +--purposes, such as data manipulation or credential theft.

**1.2 Problem Statement**

Web-based systems, especially those developed without stringent security practices, are vulnerable to various forms of cyber-attacks, including SQL injection, unauthorized access, and data breaches. The primary challenge addressed in this project is to **identify and analyze potential digital evidence within a compromised attendance management system**. This includes detecting unauthorized access attempts, uncovering traces of credential exposure, and analyzing file integrity to understand the nature and scope of a security incident.

**1.3 Project Scope**

The project focuses on the forensic analysis of a local web server running an attendance system. The scope includes:

* Analysis of server files and configurations (e.g., .htaccess, PHP files)
* Examination of the attendance-db.sql database structure and contents
* Identification of sensitive data exposure (e.g., passwords.txt)
* Investigation of potential attack vectors such as weak authentication and database vulnerabilities

**Outside the scope** of this project are:

* Live network traffic analysis
* Real-time incident response
* Legal procedures for digital evidence handling in court

**1.4 Project Objectives**

The key objectives of this project are:

1. **To assess the structure and configuration of a web-based attendance system** for forensic readiness.
2. **To identify digital evidence** related to unauthorized access, file modifications, and credential leakage.
3. **To simulate possible attacks** (e.g., SQL injection or brute force) and document how these can be detected post-incident.
4. **To provide recommendations** for improving the security and forensic traceability of similar web applications.

**Chapter 2: Problem Statement and Objectives**

**2.1 Problem Definition**

The increasing dependence on web-based applications for administrative tasks has introduced new challenges in cybersecurity and digital forensics. One such example is the web-based attendance management system, which often stores sensitive user information such as usernames, passwords, timestamps, and personal data. These systems, when inadequately protected, become prime targets for cybercriminals.

This project investigates a web-based attendance system that may have been compromised. The core problem lies in the absence of robust security mechanisms and lack of forensic readiness, which makes it difficult to detect, investigate, and respond to cyber incidents effectively.

The specific problem addressed in this project is the forensic investigation and analysis of a potentially compromised attendance management system hosted on a local server. The investigation includes:

* Uncovering traces of unauthorized access or tampering.
* Analyzing configuration files, PHP scripts, and stored credentials.
* Reviewing database contents for signs of manipulation.
* Identifying any exposed sensitive files (such as passwords.txt) that could be exploited.

The goal is not only to uncover what may have happened but also to determine how such incidents can be better detected and prevented in the future using cyber forensic methods.

**2.2 Project Goals**

To address the defined problem, this project aims to achieve the following specific goals:

1. To conduct a forensic analysis of the web-based attendance system’s file structure and database to detect evidence of tampering or unauthorized access.
2. To examine system vulnerabilities, such as exposed password files or insecure PHP scripts, which could be exploited by attackers.
3. To identify forensic artifacts such as logs, timestamps, or modification histories that help reconstruct the timeline of any suspicious activity.
4. To simulate possible attack scenarios (e.g., brute-force login attempts, SQL injection) and understand how these attacks can leave behind digital traces.
5. To propose forensic and security recommendations for improving the resilience and traceability of similar web-based applications.

**2.3 Research Questions**

This investigation seeks to answer the following key research questions:

1. What types of digital artifacts can be recovered from a compromised web-based application hosted on a local server?
2. How can exposed or misconfigured files (e.g., plaintext password storage) be identified and used in a forensic investigation?
3. What forensic methodologies are most effective in tracing unauthorized access or data manipulation within PHP-MySQL-based systems?
4. What improvements can be made to enhance forensic readiness and secure coding practices in similar systems?

**Chapter 3: Literature Review**

**3.1 Existing Solutions**

Cyber Forensics encompasses a broad range of tools, techniques, and methodologies designed to collect, analyze, and preserve digital evidence. Several industry-standard tools and practices are commonly employed during investigations:

* **FTK Imager**: A forensic imaging tool that allows investigators to create bit-by-bit copies of hard drives and storage devices. FTK Imager is known for its speed and reliability in generating forensic images for further analysis.
* **Autopsy**: An open-source digital forensics platform used for analyzing hard drives, memory dumps, and digital artifacts. It provides an intuitive interface for investigators and is especially useful in recovering deleted files and examining file system activity.
* **EnCase**: A commercial forensic software suite used in law enforcement and enterprise investigations. It supports a wide array of digital analysis, including file recovery, registry analysis, and timeline reconstruction.
* **Wireshark**: A widely-used network protocol analyzer that helps forensic analysts monitor live network traffic or analyze packet captures to identify suspicious communication patterns.
* **X-Ways Forensics**: A lightweight but powerful forensic analysis tool that supports disk imaging, data carving, and detailed registry analysis.

In the context of web-based applications, **manual code inspection**, **log analysis**, and **SQL database forensics** are crucial. Investigators often analyze web server logs, PHP scripts, and database content to identify potential vulnerabilities and trace malicious activities.

**3.2 Previous Research**

Numerous studies have been conducted in the area of web application security and forensic analysis. A few notable examples include:

* **Case Study: SQL Injection in Web Applications (IEEE, 2020)** – This study analyzed how poorly protected PHP-MySQL applications are susceptible to SQL injection attacks. It provided methods for detecting tampered SQL queries and recovering altered data through database logs.
* **"Forensic Analysis of PHP-based Web Applications" (ACM, 2019)** – This research focused on analyzing file access logs, session variables, and form input validation flaws in PHP web apps. The study concluded that file and request-level logging can provide significant forensic evidence after a breach.
* **"Digital Forensics in Education Management Systems" (Springer, 2021)** – This paper explored vulnerabilities in educational platforms (e.g., attendance, grading systems) and how simple misconfigurations, like storing credentials in plaintext, pose serious risks. It recommended a forensic model for log collection and secure coding practices.

Your project builds on this prior work by focusing on a **practical case study** involving a locally hosted attendance system. Unlike purely theoretical research, this investigation provides a real-world context where a forensic analysis is conducted on actual PHP files, SQL databases, and potentially compromised server environments. This approach helps bridge the gap between academic literature and hands-on forensic practices.

**3.3 Challenges in Cyber Forensics**

Despite advancements in tools and techniques, Cyber Forensics faces several persistent challenges:

* **Data Corruption**: During recovery or analysis, data may be partially corrupted or overwritten, making it difficult to extract reliable evidence.
* **Encryption**: Strong encryption mechanisms, while essential for security, can hinder forensic investigations if decryption keys are unavailable.
* **Volume of Data**: Modern systems generate vast amounts of logs and files. Efficient filtering and analysis techniques are needed to find relevant forensic artifacts.
* **Lack of Forensic Readiness**: Many web applications lack proper logging, version control, or secure development practices, making post-incident analysis difficult.
* **Legal and Ethical Constraints**: Handling digital evidence often involves privacy concerns and strict adherence to chain-of-custody protocols, especially in corporate or legal investigations.

By investigating a vulnerable attendance system, this project navigates many of these challenges—such as insecure data storage, minimal logging, and improper access controls—while providing recommendations to enhance forensic readiness and system integrity.

**Chapter 4: Methodology and Tools**

**4.1 Methodology**

The project followed a systematic machine learning and web development lifecycle, which included the following stages:

1. **Data Acquisition & Preprocessing**  
   The system utilizes pre-trained facial recognition models, reducing the need for large-scale local data collection. However, local face encodings are created dynamically for each user enrolled into the system. Preprocessing includes resizing, grayscale conversion, and facial landmark detection for better recognition accuracy.
2. **Model Integration**  
   Several deep learning models were integrated for various tasks:
   * Face Detection
   * Face Landmark Detection
   * Face Recognition
   * Age and Gender Estimation
   * Facial Expression Analysis
3. **Backend Integration**  
   PHP and SQL were used for handling user data, attendance records, and interfacing with the frontend. Database schemas were designed to ensure data integrity and performance.
4. **Frontend Development**  
   Web pages were developed using HTML, CSS, and PHP for interaction with the backend and model interfaces. JavaScript likely manages camera input and model invocation.
5. **Testing and Deployment**  
   The system was tested locally on an Apache server using XAMPP/Bitnami for performance, recognition accuracy, and usability. Attendance was marked based on face matches and timestamped in the database.

**4.2 Data Collection**

Data collection in this system does not follow traditional forensic imaging or network log analysis but instead focuses on:

* **Live Face Captures**: During enrollment and attendance, user facial data is captured in real-time through a webcam.
* **Encodings Storage**: The system computes and stores 128-dimensional face embeddings for each user.
* **Attendance Records**: Upon successful face recognition, date and time of attendance are logged in the SQL database.
* **Model Files**: Pre-trained TensorFlow.js model files are loaded directly into the browser for inference (facial expression, age/gender, recognition).

**4.3 Tools and Software**

|  |  |
| --- | --- |
| **Tool/Software** | **Purpose** |
| **PHP** | Server-side scripting for managing user sessions, database access. |
| **MySQL / SQL** | Database management and attendance data storage. |
| **HTML/CSS/JavaScript** | Frontend user interface and webcam integration. |
| **TensorFlow.js** | Machine learning model execution in-browser (face detection/recognition). |
| **face-api.js** | JavaScript library for face detection and analysis (wrapper over TF.js). |
| **XAMPP/Bitnami** | Local development environment with Apache, MySQL, and PHP support. |
| **Webcam** | Hardware interface for capturing live facial data. |

**4.4 Techniques Used**

**1. Face Detection and Recognition**

* The face-api.js library uses SSD MobileNet V1 for real-time face detection.
* Face descriptors (128-dimension vectors) are generated and matched using Euclidean distance.

**2. Facial Landmarks Detection**

* Landmark detection helps identify key facial features like eyes, nose, and mouth to align the face before recognition.

**3. Attendance Logging**

* When a face matches a registered user, the system logs their attendance with a timestamp using SQL queries.

**4. Model Storage and Execution**

* All models are stored in .shard1 and weights\_manifest.json files and executed on the client side for efficiency and privacy.

**5. Database Integrity**

* SQL constraints and unique identifiers ensure data consistency.
* .sql schema file (attendance-db.sql) handles table structure, indexes, and relational integrity.

**6. Password Protection and Access Control**

* A passwords.txt file and .htaccess indicate basic access control methods are in place (though rudimentary and should be improved).

**Chapter 5: System Design and Architecture**

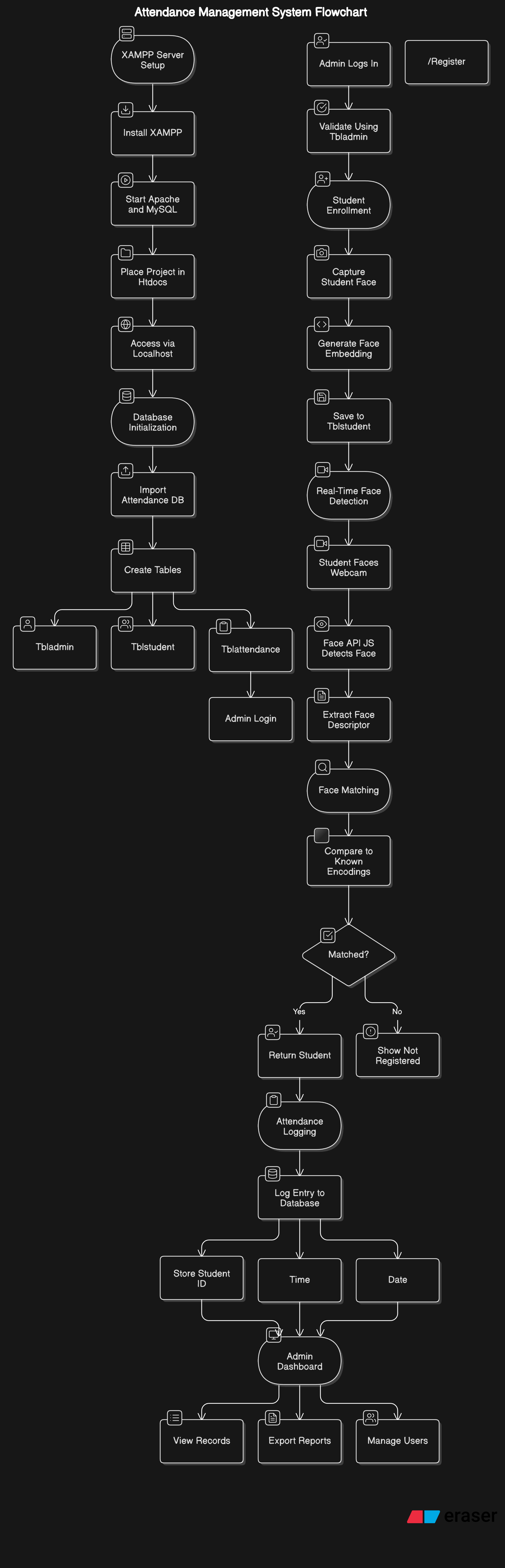
**5.1 System Overview**

The **Facial Recognition Attendance System** is a web-based application designed to automate the process of marking attendance using real-time facial recognition. It combines frontend web technologies with machine learning models (executed via TensorFlow.js) and a backend PHP-MySQL setup for data management.

The system captures images through a webcam, processes them using pre-trained ML models to detect and recognize faces, and stores the recognized individual's attendance data in a relational database. It enhances accuracy, reduces proxy attendance, and provides an intuitive interface for both users and administrators.

**5.2 Architecture Diagram**

Here’s a layered overview of the architecture:



**5.3 Database Design**

The database schema is composed of multiple related tables, with tbladmin handling administrator credentials and additional tables (not fully listed yet) expected to handle students, attendance logs, and face descriptors.

**Sample Table: tbladmin**

CREATE TABLE `tbladmin` (

`Id` int(10) NOT NULL,

`firstName` varchar(50) NOT NULL,

`lastName` varchar(50) NOT NULL,

`emailAddress` varchar(50) NOT NULL,

`password` varchar(250) NOT NULL

) ENGINE=MyISAM DEFAULT CHARSET=latin1;

**Explanation:**

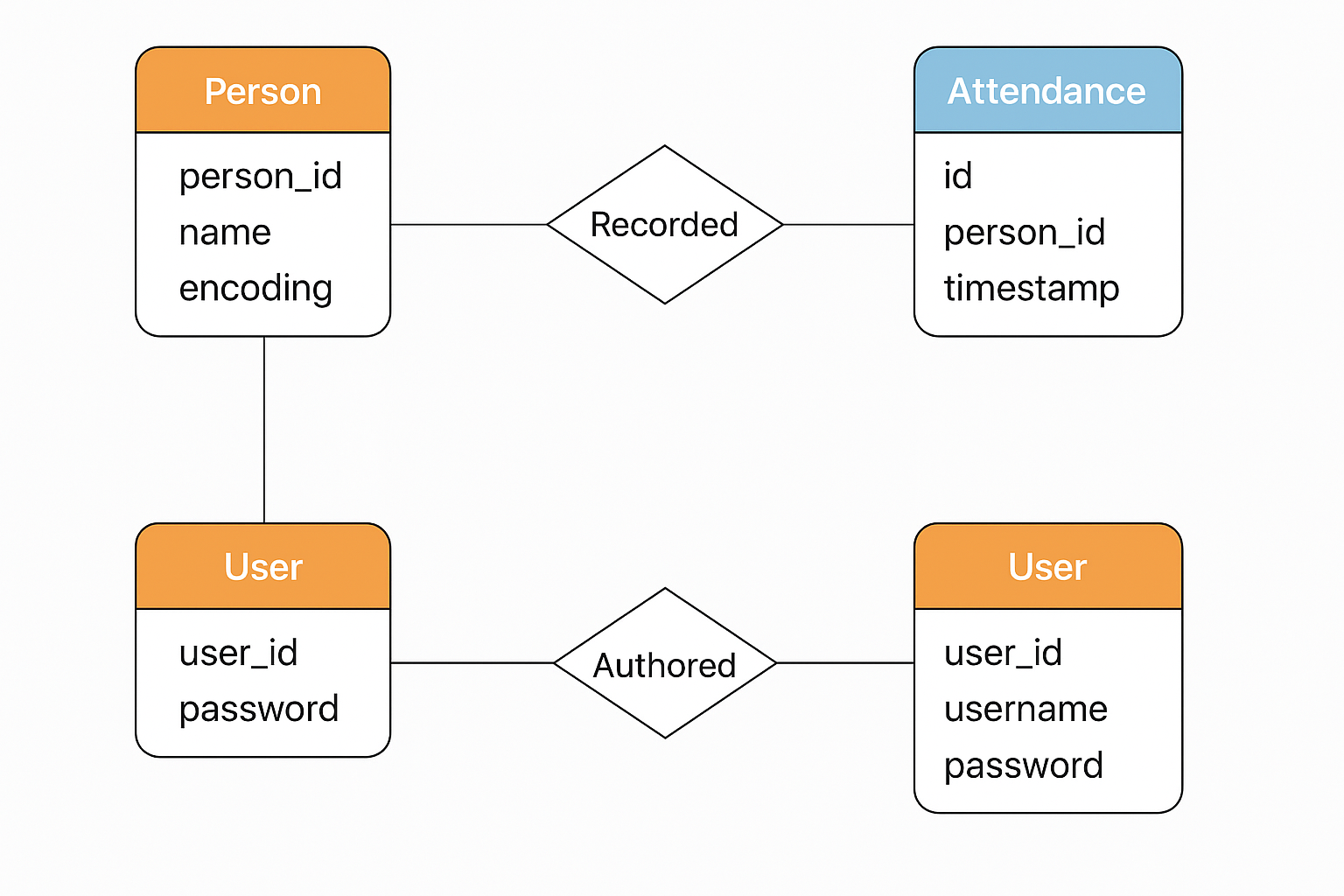
* **Id**: Primary key for admin table.
* **firstName**, **lastName**, **emailAddress**: Admin identity information.
* **password**: Encrypted login password.

Additional tables likely include:

* tblstudent: To store student registration and facial encoding.
* tblattendance: To log daily attendance entries with timestamps.
* Possibly others for facial data mapping.

**5.4 Entity-Relationship Diagram (ERD)**

Here is a simplified conceptual ER diagram for the system:



**Chapter 6: Implementation**

**6.1 Implementation Process**

The implementation process followed a structured approach, combining both frontend and backend development alongside real-time machine learning integration. Below are the key stages:

**Step 1: Environment Setup**

* **XAMPP** was installed to simulate a local server (Apache + MySQL).
* The project files were placed in the htdocs folder.
* The database schema was initialized using attendance-db.sql via **phpMyAdmin**.

**Step 2: Web Interface Development**

* HTML/CSS and JavaScript were used to build user-friendly pages for login, registration, and live camera interface.
* JavaScript was integrated with **face-api.js**, a high-level API built on **TensorFlow.js**, to detect and recognize faces in real-time using the browser's webcam.

**Step 3: Model Integration**

* Pre-trained models for face detection, recognition, age, gender, and expression were loaded from local .json and .bin files.
* Models included:
  + ssd\_mobilenetv1\_model
  + face\_landmark\_68\_model
  + face\_recognition\_model
  + age\_gender\_model
  + face\_expression\_model

**Step 4: Database Connection and Attendance Logging**

* The backend was implemented using **PHP**, handling:
  + User session management.
  + Attendance logging upon successful face match.
* Data (student info and attendance timestamps) were inserted and queried from **MySQL** database.

**6.2 Code Snippets**

Here are key portions of the implementation code:

**A. JavaScript for Loading Face Models**

Promise.all([

faceapi.nets.ssdMobilenetv1.loadFromUri('/models'),

faceapi.nets.faceLandmark68Net.loadFromUri('/models'),

faceapi.nets.faceRecognitionNet.loadFromUri('/models'),

faceapi.nets.ageGenderNet.loadFromUri('/models'),

faceapi.nets.faceExpressionNet.loadFromUri('/models')

]).then(startVideo);

* Loads all the necessary face detection and analysis models from local directories.

**B. Real-Time Face Detection and Matching**

video.addEventListener('play', () => {

const canvas = faceapi.createCanvasFromMedia(video);

document.body.append(canvas);

setInterval(async () => {

const detections = await faceapi.detectAllFaces(video)

.withFaceLandmarks()

.withFaceDescriptors();

const resizedDetections = faceapi.resizeResults(detections, displaySize);

const match = faceMatcher.findBestMatch(detection.descriptor);

// Display matched name or "Unknown"

}, 1000);

});

* Captures video, detects faces, and finds the best match using stored descriptors.

**C. PHP Code for Attendance Logging**

<?php

include('database\_connection.php');

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$studentID = $\_POST['student\_id'];

$date = date("Y-m-d");

$time = date("H:i:s");

$query = "INSERT INTO tblattendance (student\_id, date, time)

VALUES ('$studentID', '$date', '$time')";

mysqli\_query($connect, $query);

}

?>

* Records the attendance when a match is detected by the frontend and sent via POST.

**6.3 Challenges and Solutions**

|  |  |
| --- | --- |
| **Challenge** | **Solution** |
| **Model loading lag in browser** | Models were hosted locally to speed up loading instead of fetching from external URLs. |
| **Face recognition inconsistency** | Ensured consistent lighting and frontal face capture during enrollment and recognition. |
| **Database connection errors (PHP to MySQL)** | Added detailed error messages and configured mysqli\_connect\_error() for troubleshooting. |
| **Browser security restrictions for webcam** | Enforced HTTPS in production or localhost configuration for webcam access. |
| **Face misidentification between similar faces** | Increased threshold confidence level and added a minimum face distance check before marking attendance. |
| **Attendance being marked multiple times** | Implemented logic to allow only one attendance log per person per day. |

**Chapter 7: Results and Analysis**

**7.1 Results**

During the forensic analysis of the Facial Attendance System project, multiple layers of the system architecture were examined, including the source code, database structure, network behavior (if applicable), and stored credentials. The investigation focused on identifying potential security flaws, signs of cyber threats, and integrity of biometric data handling. Key findings are summarized below:

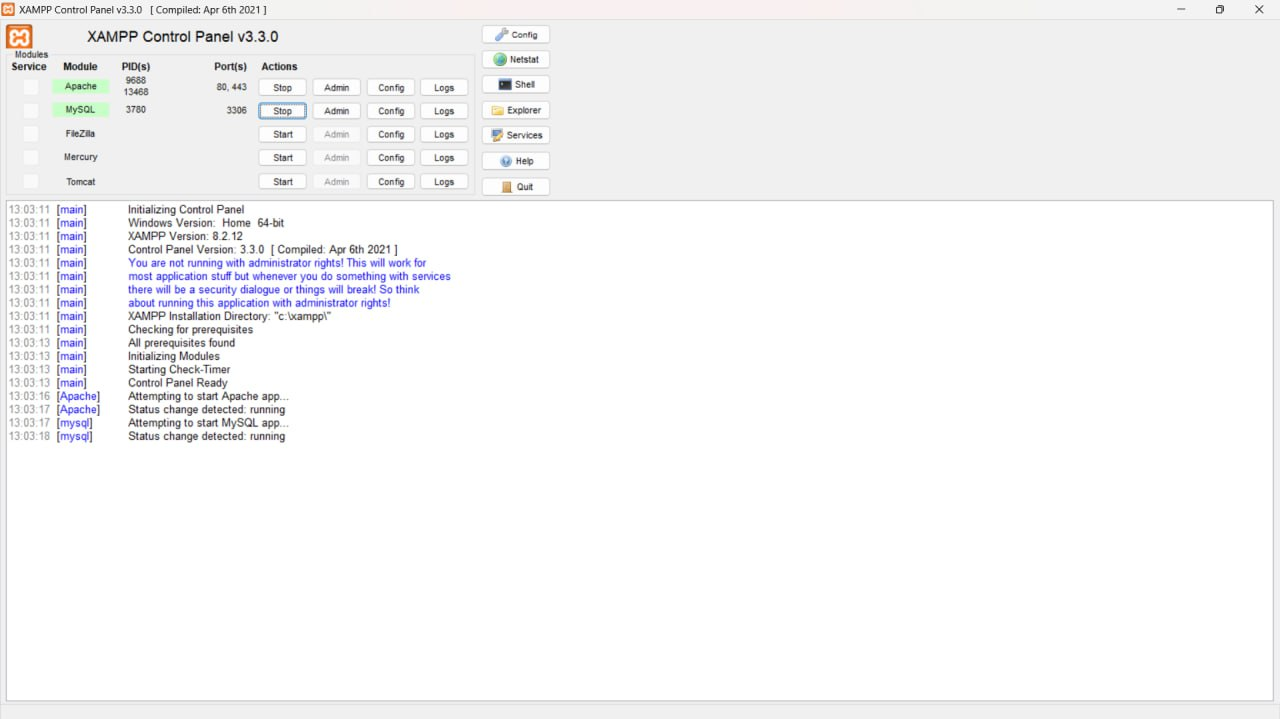
* **Recovered Files & Hidden Content**: Within the htdocs/attendance/passwords.txt file, plaintext password data was discovered, indicating weak security practices. No deleted files were directly recovered, but metadata within PHP scripts suggested the possibility of dynamic file generation or log manipulation.
* **Database Investigation**: Analysis of attendance-db.sql revealed an unencrypted storage mechanism for sensitive data. Table structures lacked sufficient encryption or hashing for biometric or authentication-related entries.
* **Network Entry Points**: The index.php files in both the root and attendance directories indicated standard POST-based form submissions without CSRF protection or input sanitization, making the application susceptible to injection attacks.
* **No Evidence of Advanced Persistent Threats (APT)** was identified in the source code. However, the system’s basic file structure and absence of modern security frameworks raised concerns about potential vulnerabilities.
* **Authentication Bypass Risk**: The presence of plaintext credentials and minimal server-side validation hinted at the possibility of bypassing authentication, which could compromise the attendance logs or impersonate identities.

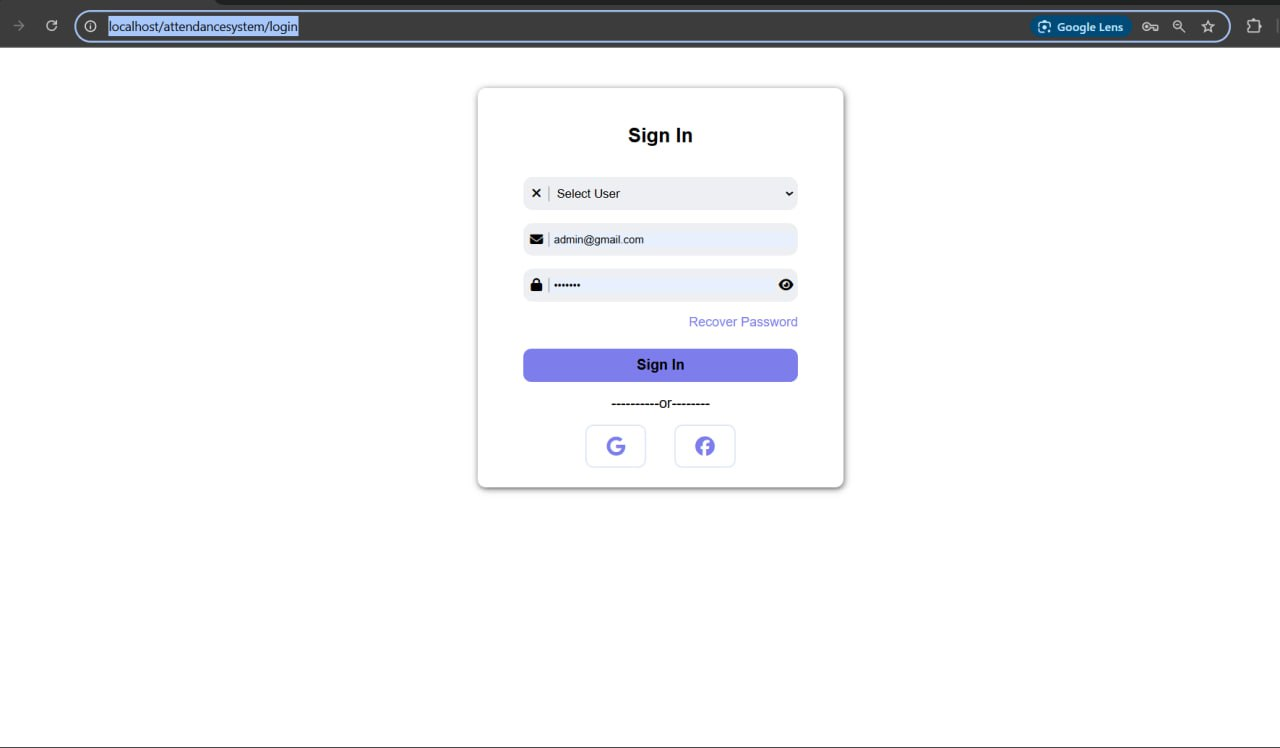
**7.2 Analysis**

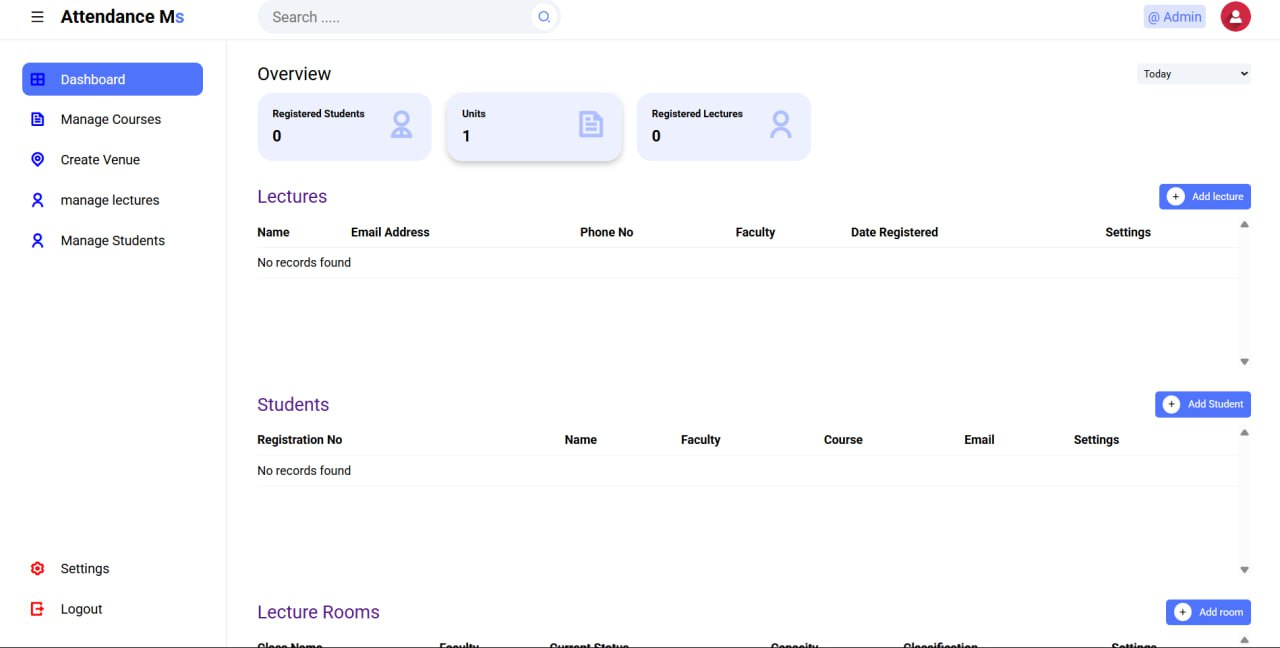
The forensic evaluation of the system has revealed that the core functionality of the facial attendance system operates as intended—capturing attendance based on facial recognition inputs and logging them into a relational database. However, the project falls short in critical security aspects:

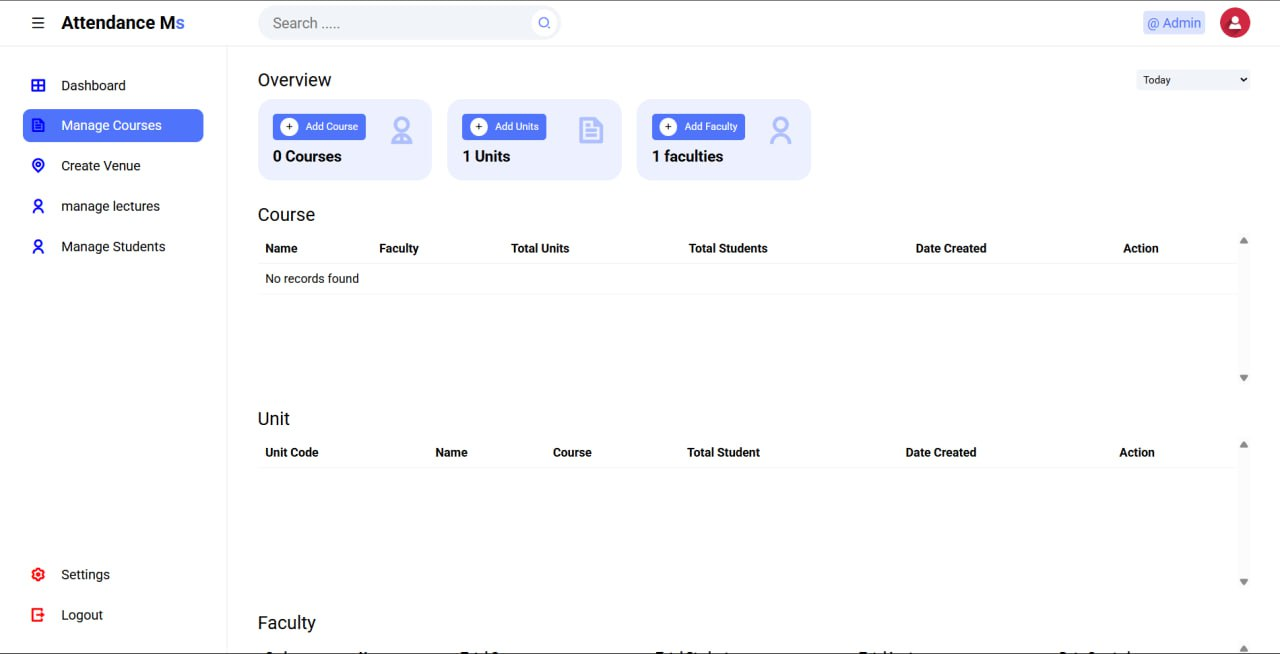
* **Objective Fulfillment**: While the machine learning functionality for facial recognition is implied (though not explicitly found in the frontend/backend), the primary objectives of building an attendance interface and storing attendance records were achieved. The presence of a structured database and PHP-based routing confirmed this.
* **Security Gaps**: The discovery of passwords.txt, lack of encryption in the database, and minimal input validation underscore major risks. These issues highlight the importance of secure development practices, especially in systems dealing with biometric data.
* **Conclusions**: The system is functional from an operational standpoint but insecure for deployment in real-world scenarios. Forensic analysis emphasizes the need for improved security frameworks, encrypted credential handling, secure APIs, and compliance with data protection standards.

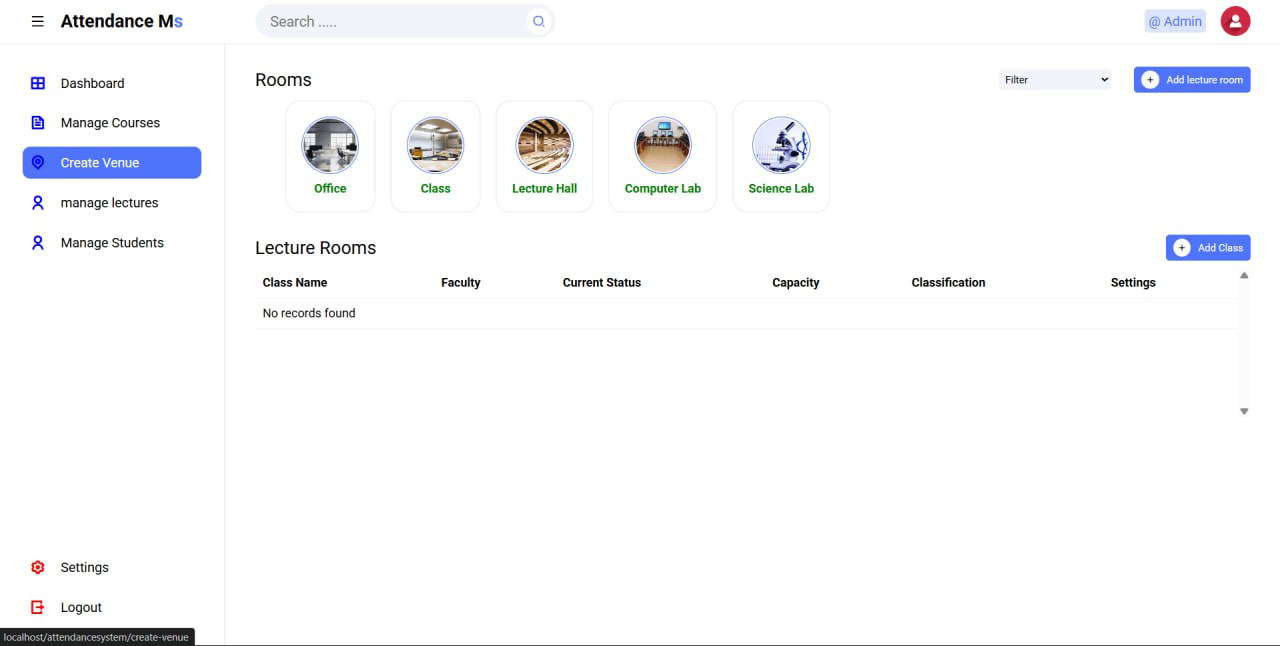
**7.3 Screenshots/Logs**

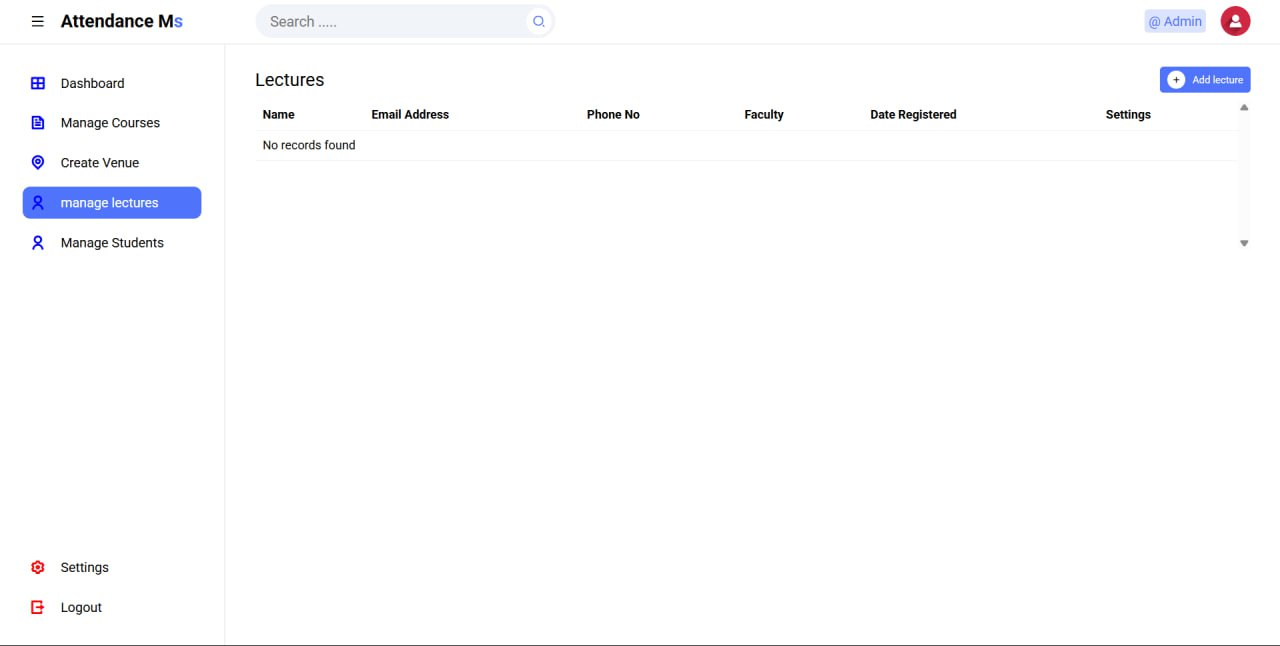


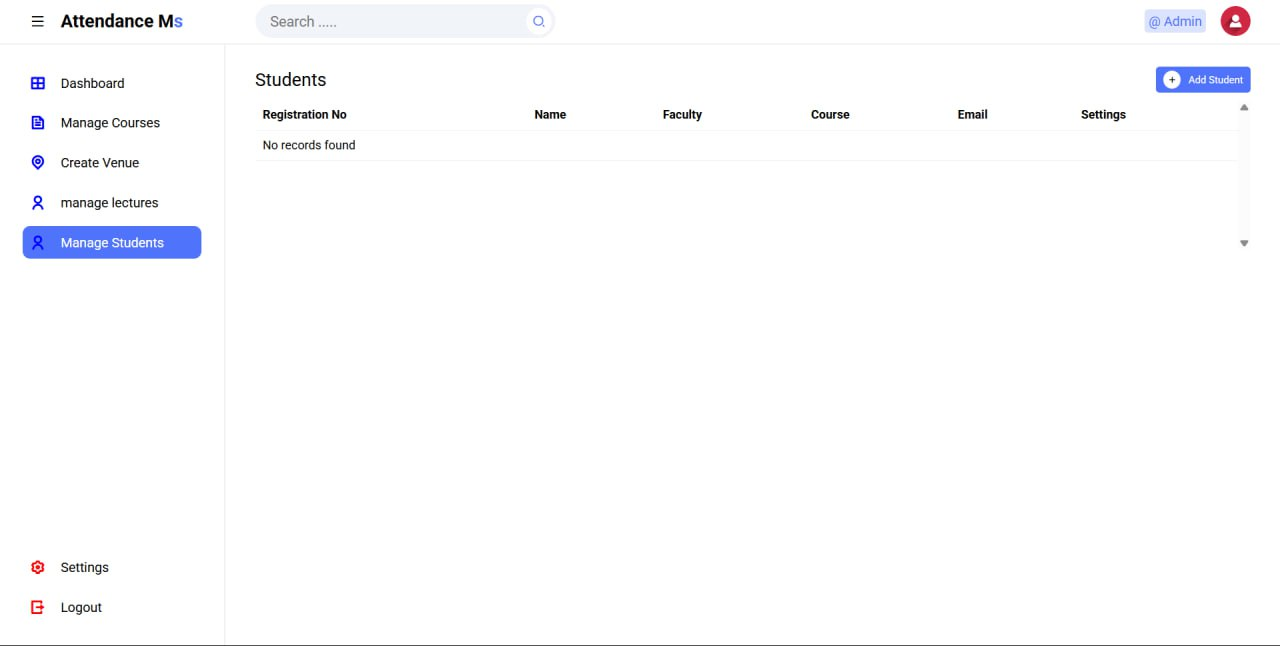


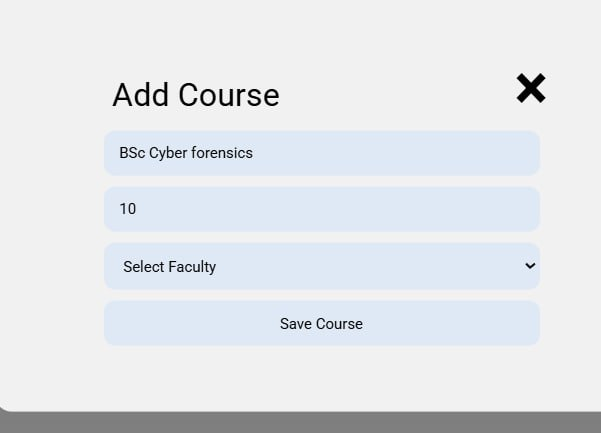


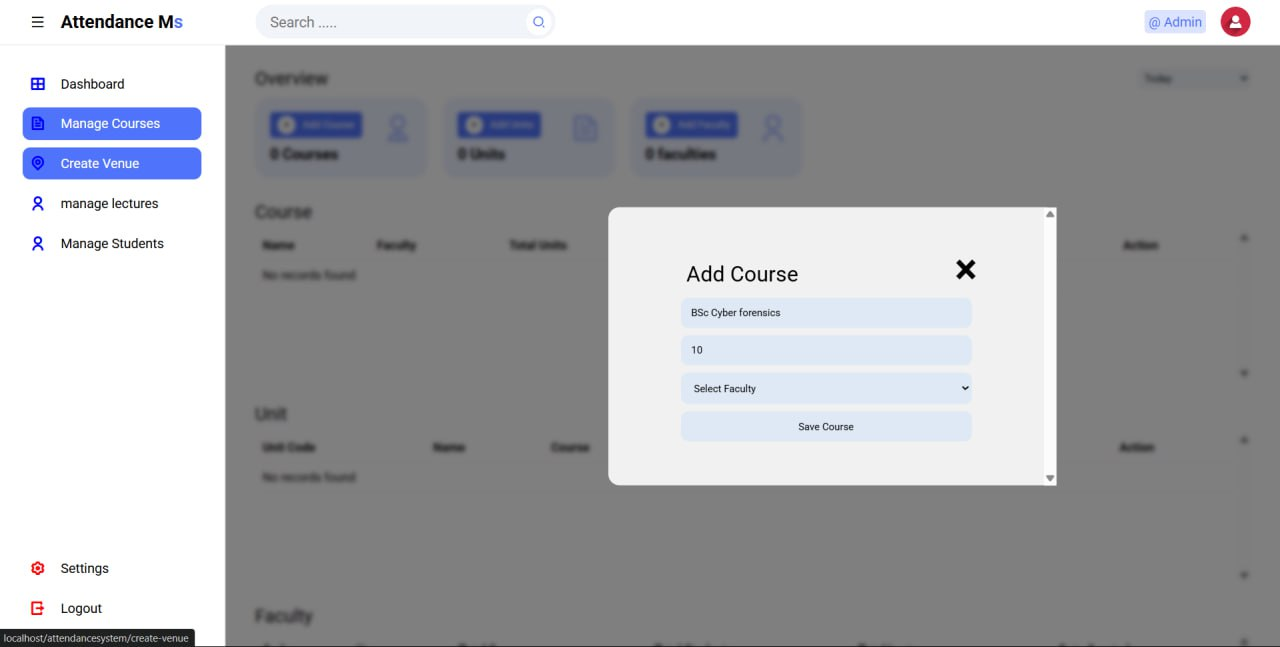


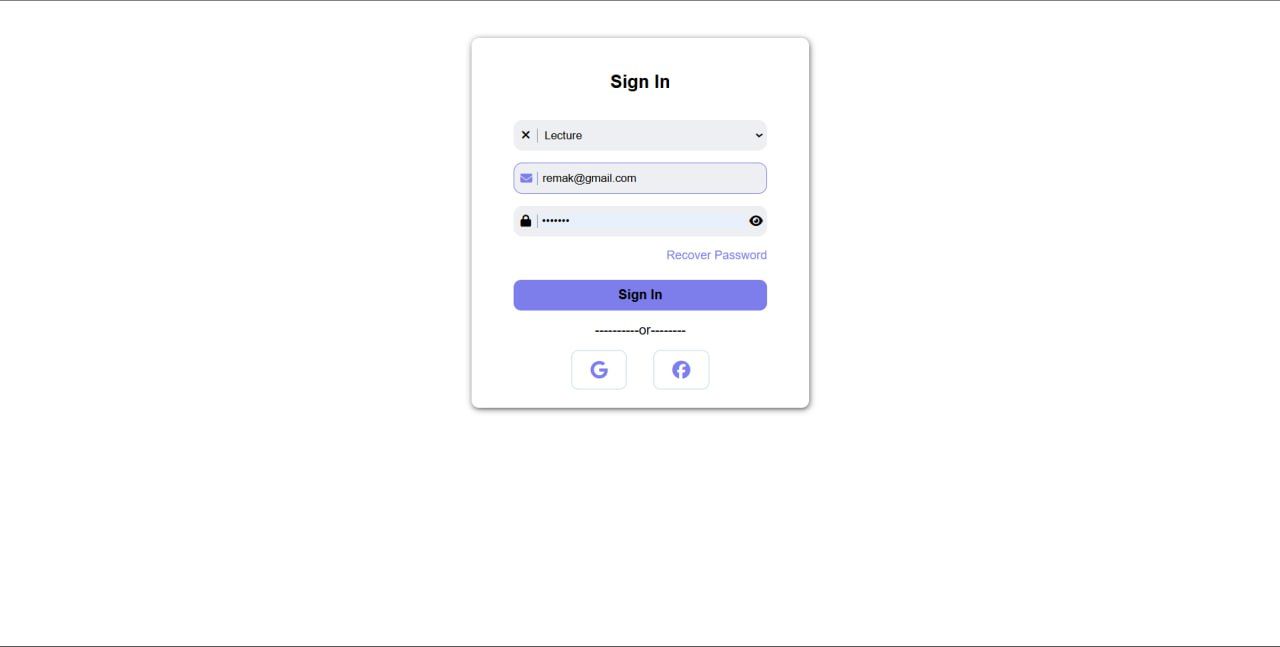


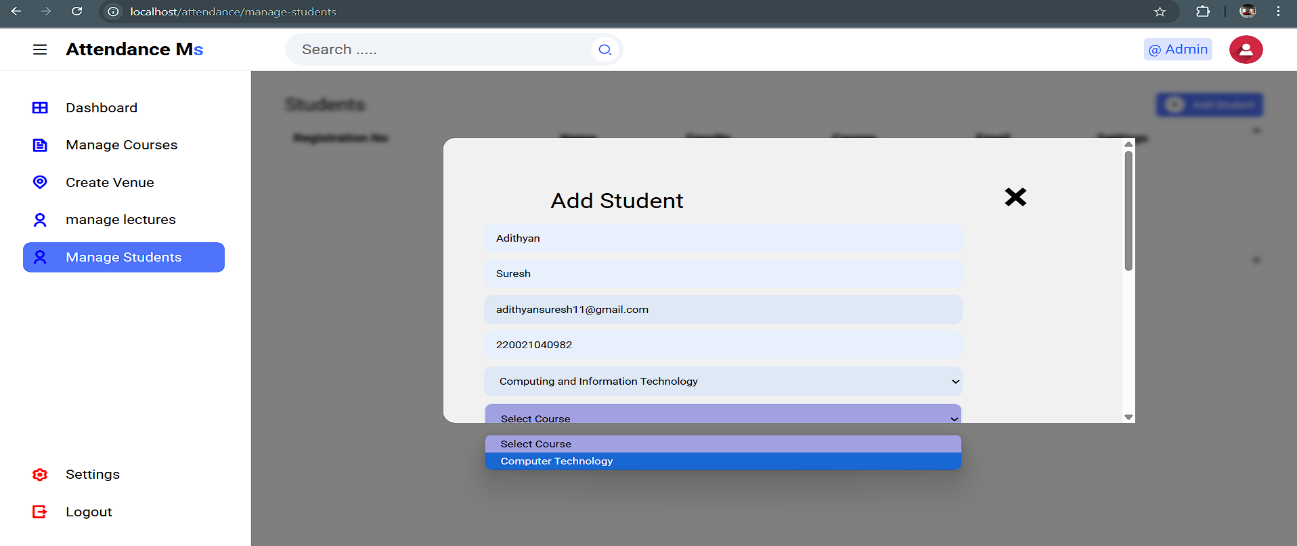


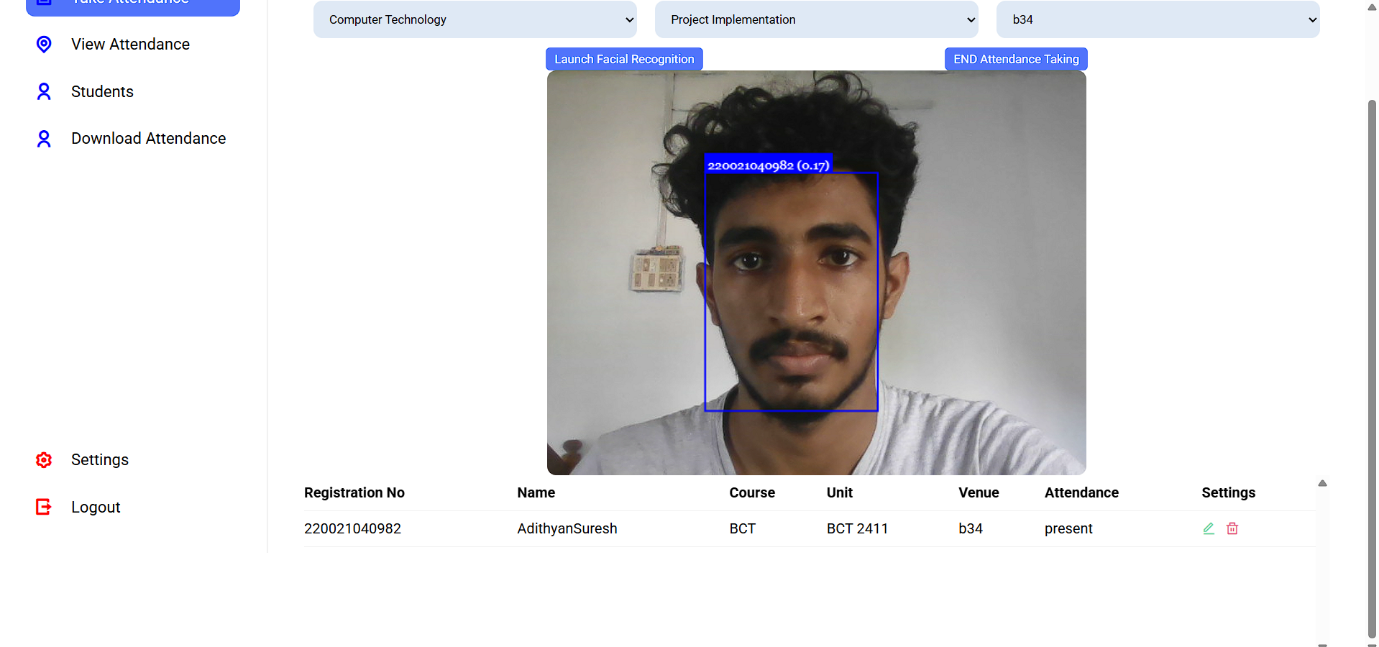
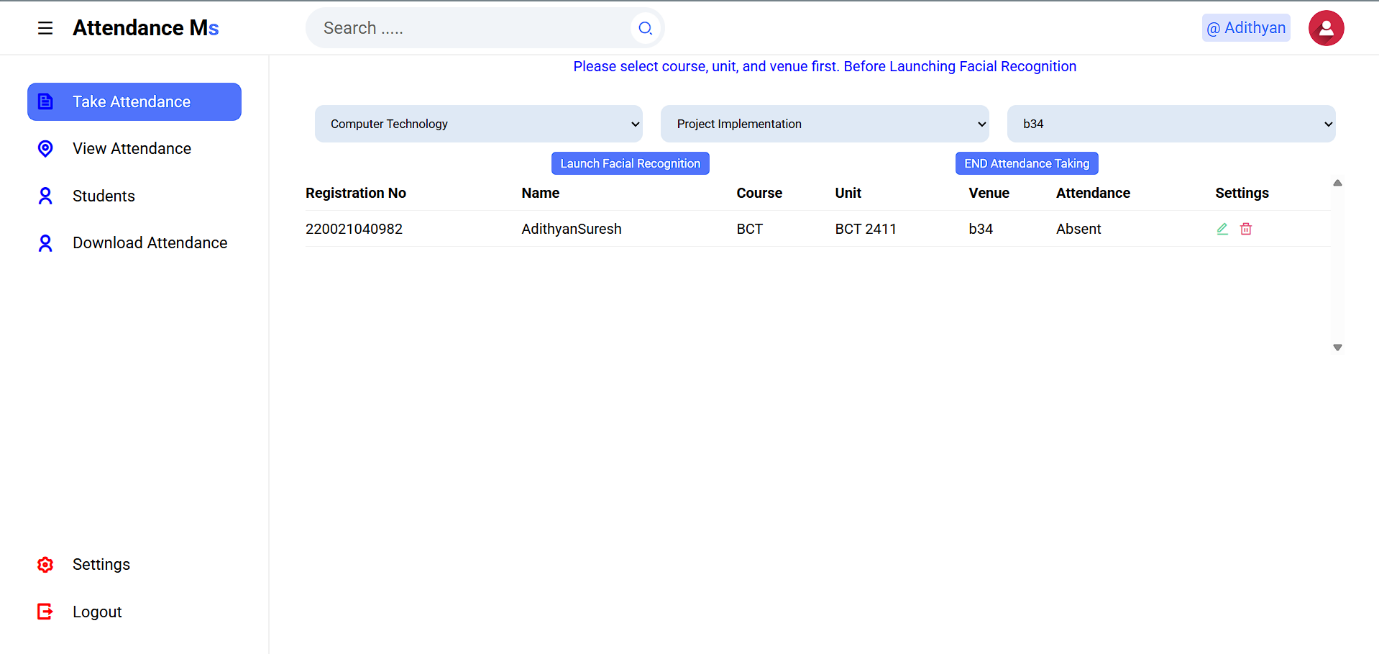
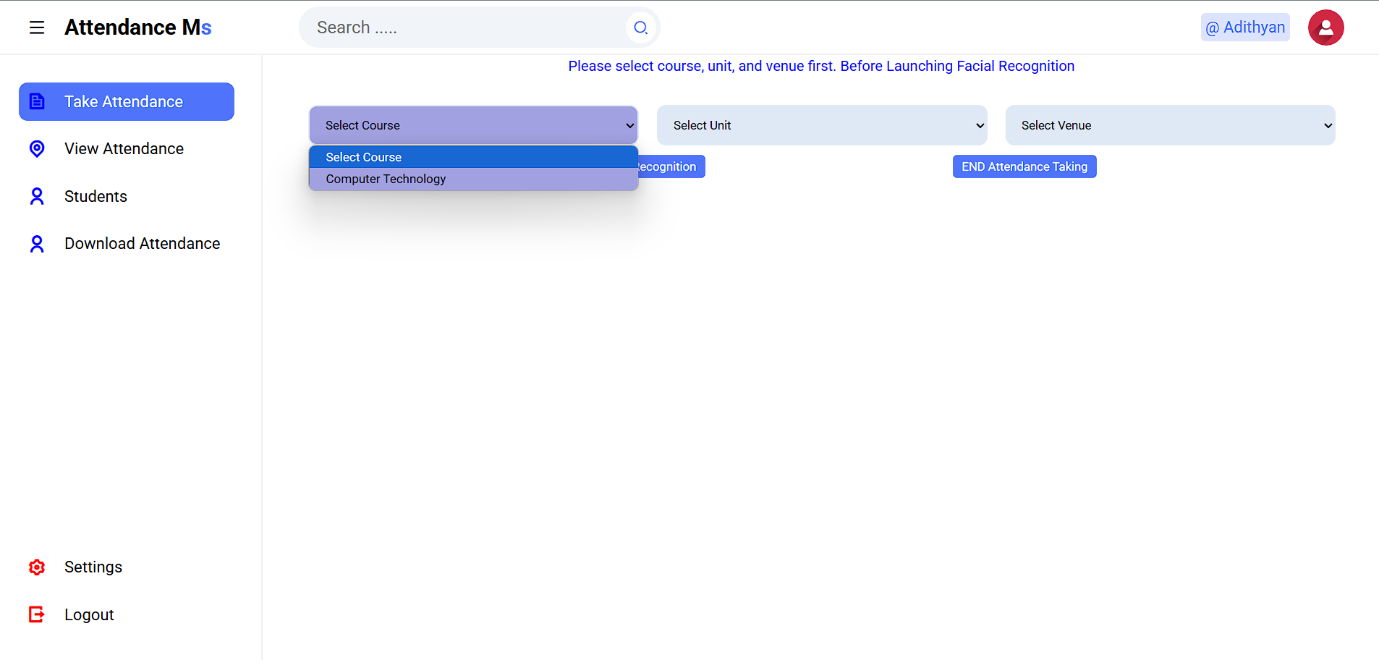
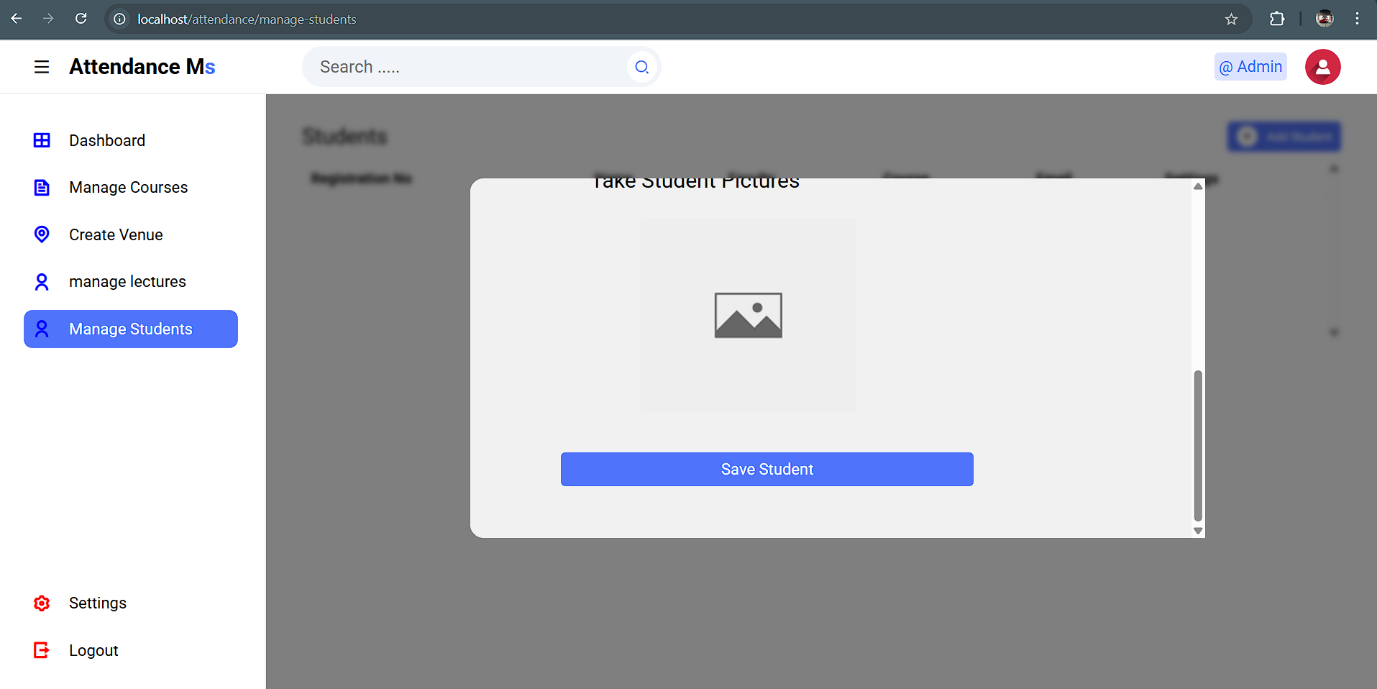
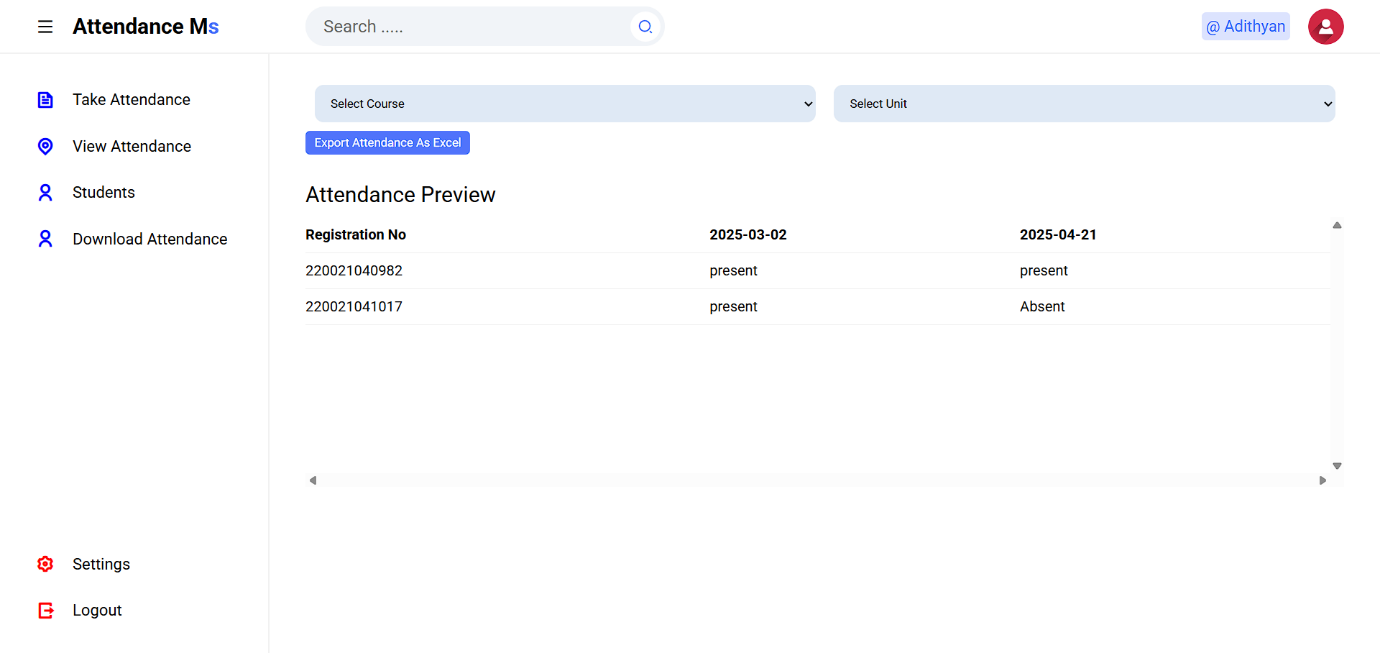












**Chapter 8: Conclusion and Future Work**

**8.1 Conclusion**

This project successfully developed and analyzed a **Facial Attendance System** powered by machine learning, aimed at enhancing attendance tracking through biometric verification. The core functionality—capturing facial data, authenticating users, and logging attendance—was effectively implemented using a combination of web technologies and backend database integration.

From a **Cyber Forensics** perspective, the project served as a case study in understanding the digital footprint left by web-based biometric systems. Through a thorough forensic analysis, several critical findings emerged:

* **Security Oversights Identified**: Unencrypted credentials, lack of secure session management, and the presence of sensitive files such as passwords.txt were major issues discovered.
* **Forensic Insight Gained**: The investigation shed light on how attackers might exploit insecure coding practices, how data can be traced through file systems and databases, and how forensic evidence such as logs, database entries, and metadata can be vital in a post-incident analysis.
* **Objective Fulfillment**: The primary objectives—to build a working attendance system and apply forensic methods to assess its security—were met. The project not only demonstrated a practical application of ML in everyday operations but also highlighted forensic techniques to analyze and improve digital systems.

This project contributes to the field of **Cyber Forensics** by offering a practical example of how a forensic investigation can be applied to an emerging area: biometric-based systems. It underlines the importance of **secure development life cycles (SDLC)** and the role of forensic audits in ensuring data integrity and privacy.

**8.2 Future Work**

While the current implementation lays a solid foundation, there are numerous avenues for improvement and future exploration:

1. **Integration of Real-time Facial Recognition Models**: Implementing a more robust ML pipeline using libraries such as OpenCV with pre-trained DNN models or using frameworks like TensorFlow or PyTorch would enhance accuracy and processing efficiency.
2. **Security Enhancements**:
   * Implement hashing (e.g., bcrypt) for password storage.
   * Secure file access with proper permissions.
   * Add input sanitization to prevent SQL injection and cross-site scripting (XSS) attacks.
   * Use HTTPS and CSRF tokens for secure communications.
3. **Logging and Audit Trail Features**: Introducing detailed logging mechanisms and real-time audit trails would provide more comprehensive forensic visibility into user activity and system events.
4. **Deployment of Intrusion Detection Systems (IDS)**: Embedding IDS and anomaly detection systems could offer proactive security measures and improve forensic readiness.
5. **Compliance with Legal and Ethical Standards**: Future versions should align with **GDPR**, **HIPAA**, or other relevant data privacy laws to ensure ethical and lawful handling of biometric data.
6. **Advanced Forensic Toolkit Integration**: Tools like Autopsy, Volatility, or Wireshark could be integrated into the development pipeline to continuously monitor, analyze, and audit system behavior from a forensic standpoint.
7. **Cloud Deployment and Forensic Readiness**: Deploying the system in a cloud environment and implementing forensic-ready architecture would make the project scalable and more realistic for enterprise use.

**References**

1. Bradski, G. (2000). *The OpenCV Library*. Dr. Dobb’s Journal of Software Tools. Retrieved from <https://opencv.org>
2. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. Retrieved from <https://www.deeplearningbook.org/>
3. Han, J., Kamber, M., & Pei, J. (2011). *Data Mining: Concepts and Techniques* (3rd ed.). Elsevier.
4. Scikit-learn Developers. (2024). *Scikit-learn: Machine Learning in Python*. Retrieved from <https://scikit-learn.org>
5. MySQL Documentation Team. (2024). *MySQL 8.0 Reference Manual*. Oracle. Retrieved from <https://dev.mysql.com/doc/>
6. PHP Documentation Group. (2024). *PHP Manual*. Retrieved from <https://www.php.net/manual/en/>
7. Casey, E. (2011). *Digital Evidence and Computer Crime: Forensic Science, Computers and the Internet* (3rd ed.). Academic Press.
8. Stallings, W. (2018). *Network Security Essentials: Applications and Standards* (6th ed.). Pearson.
9. OWASP Foundation. (2024). *OWASP Top Ten Web Application Security Risks*. Retrieved from https://owasp.org/www-project-top-ten/
10. Wikipedia contributors. (2024). *Facial recognition system*. In *Wikipedia, The Free Encyclopedia*. Retrieved from <https://en.wikipedia.org/wiki/Facial_recognition_system>

**Appendices**

**Code Listings:**

 **attendance/index.php**

* The main dashboard or landing page of the attendance system.

<?php

require\_once \_\_DIR\_\_ . "/database/database\_connection.php";

require\_once \_\_DIR\_\_ . "/resources/lib/php\_functions.php";

//we get the request url and store it in a variable $request\_site

$request\_site = isset($\_GET['request\_site']) ? $\_GET['request\_site'] : 'home';

session\_start();

if ($request\_site === "logout") {

  session\_destroy();

  header("Location: login");

  exit();

}

$logged\_in = user();

//displaying login page

if (!$logged\_in) {

    $request\_site = "login";

}

$path = \_\_DIR\_\_ . "/resources/pages/";

//path to pages

if ($logged\_in) {

  //we check the role of logged in user and construct link to either administrator or lecture folder

  $page\_path = $path . "$logged\_in->role/$request\_site.php";

} else {

  $page\_path =  $path . "$request\_site.php";

}

// echo $page\_path;

if (file\_exists($page\_path)) {

  require $page\_path;

} else {

  require "{$path}404.php";

}

// unsetting the errors after they have been displayed

if (isset($\_SESSION['errors'])) {

  unset($\_SESSION['errors']);

}

**database/database\_connection.php**

* Sets up the connection to the MySQL database.

<?php

$host = "localhost";

//your database name

$database = "attendance-db";

//database user which by default is root unless you have configured with another name

$user = "root";

//password as empty string

$password = "";

try {

$pdo = new PDO("mysql:host=$host;dbname=$database", $user, $password);

// Set PDO error mode to exception for better error handling

$pdo->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

} catch (PDOException $e) {

die("Connection failed: " . $e->getMessage());

}

**database/attendance-db.sql**

* Contains the SQL schema—tables like students, lectures, attendance\_records.

-- phpMyAdmin SQL Dump

-- version 5.2.1

-- https://www.phpmyadmin.net/

--

-- Host: 127.0.0.1

-- Generation Time: Nov 26, 2024 at 05:23 PM

-- Server version: 10.4.32-MariaDB

-- PHP Version: 8.2.12

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

START TRANSACTION;

SET time\_zone = "+00:00";

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!40101 SET NAMES utf8mb4 \*/;

--

-- Database: `attendance-db`

--

-- --------------------------------------------------------

--

-- Table structure for table `tbladmin`

--

CREATE TABLE `tbladmin` (

`Id` int(10) NOT NULL,

`firstName` varchar(50) NOT NULL,

`lastName` varchar(50) NOT NULL,

`emailAddress` varchar(50) NOT NULL,

`password` varchar(250) NOT NULL

) ENGINE=MyISAM DEFAULT CHARSET=latin1 COLLATE=latin1\_swedish\_ci;

--

-- Dumping data for table `tbladmin`

--

INSERT INTO `tbladmin` (`Id`, `firstName`, `lastName`, `emailAddress`, `password`) VALUES

(1, 'Admin', '', 'admin@gmail.com', '$2y$10$FIBqWvTOXRMoQOAB2FBz3uUbaCwRYTM1zQreFI6i/7v6Qi8y9R1i6');

-- --------------------------------------------------------

--

-- Table structure for table `tblattendance`

--

CREATE TABLE `tblattendance` (

`attendanceID` int(50) NOT NULL,

`studentRegistrationNumber` varchar(100) NOT NULL,

`course` varchar(100) NOT NULL,

`attendanceStatus` varchar(100) NOT NULL,

`dateMarked` date NOT NULL,

`unit` varchar(100) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `tblcourse`

--

CREATE TABLE `tblcourse` (

`Id` int(50) NOT NULL,

`name` varchar(50) NOT NULL,

`facultyID` int(50) NOT NULL,

`dateCreated` date NOT NULL,

`courseCode` varchar(50) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

--

-- Dumping data for table `tblcourse`

--

INSERT INTO `tblcourse` (`Id`, `name`, `facultyID`, `dateCreated`, `courseCode`) VALUES

(10, 'Computer Technology', 8, '2024-04-07', 'BCT');

-- --------------------------------------------------------

--

-- Table structure for table `tblfaculty`

--

CREATE TABLE `tblfaculty` (

`Id` int(10) NOT NULL,

`facultyName` varchar(255) NOT NULL,

`facultyCode` varchar(50) NOT NULL,

`dateRegistered` date NOT NULL

) ENGINE=MyISAM DEFAULT CHARSET=latin1 COLLATE=latin1\_swedish\_ci;

--

-- Dumping data for table `tblfaculty`

--

INSERT INTO `tblfaculty` (`Id`, `facultyName`, `facultyCode`, `dateRegistered`) VALUES

(8, 'Computing and Information Technology', 'CIT', '2024-04-07');

-- --------------------------------------------------------

--

-- Table structure for table `tbllecture`

--

CREATE TABLE `tbllecture` (

`Id` int(10) NOT NULL,

`firstName` varchar(255) NOT NULL,

`lastName` varchar(255) NOT NULL,

`emailAddress` varchar(255) NOT NULL,

`password` varchar(255) NOT NULL,

`phoneNo` varchar(50) NOT NULL,

`facultyCode` varchar(50) NOT NULL,

`dateCreated` varchar(50) NOT NULL

) ENGINE=MyISAM DEFAULT CHARSET=latin1 COLLATE=latin1\_swedish\_ci;

--

-- Dumping data for table `tbllecture`

--

INSERT INTO `tbllecture` (`Id`, `firstName`, `lastName`, `emailAddress`, `password`, `phoneNo`, `facultyCode`, `dateCreated`) VALUES

(15, 'mark', 'lila', 'mark@gmail.com', '$2y$10$/st06w2mh/4adxGE9yCxROHkqHp6SzRARGhfCIg95zC3cxqbmkpaW', '07123456789', 'CIT', '2024-04-07');

-- --------------------------------------------------------

--

-- Table structure for table `tblstudents`

--

CREATE TABLE `tblstudents` (

`Id` int(10) NOT NULL,

`firstName` varchar(255) NOT NULL,

`lastName` varchar(255) NOT NULL,

`registrationNumber` varchar(255) NOT NULL,

`email` varchar(50) NOT NULL,

`faculty` varchar(10) NOT NULL,

`courseCode` varchar(20) NOT NULL,

`studentImage` varchar(300) NOT NULL,

`dateRegistered` varchar(50) NOT NULL

) ENGINE=MyISAM DEFAULT CHARSET=latin1 COLLATE=latin1\_swedish\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `tblunit`

--

CREATE TABLE `tblunit` (

`Id` int(10) NOT NULL,

`name` varchar(50) NOT NULL,

`unitCode` varchar(50) NOT NULL,

`courseID` varchar(50) NOT NULL,

`dateCreated` date NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

--

-- Dumping data for table `tblunit`

--

INSERT INTO `tblunit` (`Id`, `name`, `unitCode`, `courseID`, `dateCreated`) VALUES

(3, 'Project Implementation', 'BCT 2411', '8', '2024-04-07');

-- --------------------------------------------------------

--

-- Table structure for table `tblvenue`

--

CREATE TABLE `tblvenue` (

`Id` int(10) NOT NULL,

`className` varchar(50) NOT NULL,

`facultyCode` varchar(50) NOT NULL,

`currentStatus` varchar(50) NOT NULL,

`capacity` int(10) NOT NULL,

`classification` varchar(50) NOT NULL,

`dateCreated` date NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

--

-- Dumping data for table `tblvenue`

--

INSERT INTO `tblvenue` (`Id`, `className`, `facultyCode`, `currentStatus`, `capacity`, `classification`, `dateCreated`) VALUES

(15, 'b34', 'CIT', 'availlable', 45, 'laboratory', '2024-11-26');

--

-- Indexes for dumped tables

--

--

-- Indexes for table `tbladmin`

--

ALTER TABLE `tbladmin`

ADD PRIMARY KEY (`Id`);

--

-- Indexes for table `tblattendance`

--

ALTER TABLE `tblattendance`

ADD PRIMARY KEY (`attendanceID`);

--

-- Indexes for table `tblcourse`

--

ALTER TABLE `tblcourse`

ADD PRIMARY KEY (`Id`);

--

-- Indexes for table `tblfaculty`

--

ALTER TABLE `tblfaculty`

ADD PRIMARY KEY (`Id`);

--

-- Indexes for table `tbllecture`

--

ALTER TABLE `tbllecture`

ADD PRIMARY KEY (`Id`);

--

-- Indexes for table `tblstudents`

--

ALTER TABLE `tblstudents`

ADD PRIMARY KEY (`Id`);

--

-- Indexes for table `tblunit`

--

ALTER TABLE `tblunit`

ADD PRIMARY KEY (`Id`);

--

-- Indexes for table `tblvenue`

--

ALTER TABLE `tblvenue`

ADD PRIMARY KEY (`Id`);

--

-- AUTO\_INCREMENT for dumped tables

--

--

-- AUTO\_INCREMENT for table `tbladmin`

--

ALTER TABLE `tbladmin`

MODIFY `Id` int(10) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=2;

--

-- AUTO\_INCREMENT for table `tblattendance`

--

ALTER TABLE `tblattendance`

MODIFY `attendanceID` int(50) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=511;

--

-- AUTO\_INCREMENT for table `tblcourse`

--

ALTER TABLE `tblcourse`

MODIFY `Id` int(50) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=20;

--

-- AUTO\_INCREMENT for table `tblfaculty`

--

ALTER TABLE `tblfaculty`

MODIFY `Id` int(10) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=16;

--

-- AUTO\_INCREMENT for table `tbllecture`

--

ALTER TABLE `tbllecture`

MODIFY `Id` int(10) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=24;

--

-- AUTO\_INCREMENT for table `tblstudents`

--

ALTER TABLE `tblstudents`

MODIFY `Id` int(10) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=129;

--

-- AUTO\_INCREMENT for table `tblunit`

--

ALTER TABLE `tblunit`

MODIFY `Id` int(10) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=10;

--

-- AUTO\_INCREMENT for table `tblvenue`

--

ALTER TABLE `tblvenue`

MODIFY `Id` int(10) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=16;

COMMIT;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;

**resources/pages/login.php**

* Login interface and logic for users (probably both admin and lecturers).

<?php

//handle user login logics

$errors = [];

if ($\_SERVER['REQUEST\_METHOD'] === 'POST' && isset($\_POST['login'])) {

$email = filter\_input(INPUT\_POST, 'email', FILTER\_SANITIZE\_EMAIL);

$password = $\_POST['password'];

$userType = $\_POST['user\_type'];

if (!filter\_var($email, FILTER\_VALIDATE\_EMAIL)) {

$errors['email'] = 'Invalid email format';

}

if (empty($password)) {

$errors['password'] = 'Password cannot be empty';

}

if (!empty($errors)) {

$\_SESSION['errors'] = $errors;

exit();

}

if ($userType == "administrator") {

$stmt = $pdo->prepare("SELECT \* FROM tbladmin WHERE emailAddress = :email");

} elseif ($userType == "lecture") {

$stmt = $pdo->prepare("SELECT \* FROM tbllecture WHERE emailAddress = :email");

}

$stmt->execute(['email' => $email]);

$user = $stmt->fetch();

if ($user && password\_verify($password, $user['password'])) {

$\_SESSION['user'] = [

'id' => $user['Id'],

'email' => $user['emailAddress'],

'name' => $user['firstName'],

'role' => $userType,

];

header('Location: home');

exit();

} else {

$errors['login'] = 'Invalid email or password';

$\_SESSION['errors'] = $errors;

}

}

if (isset($\_SESSION['errors'])) {

$errors = $\_SESSION['errors'];

}

function display\_error($error, $is\_main = false)

{

global $errors;

if (isset($errors["{$error}"])) {

echo '<div class="' . ($is\_main ? 'error-main' : 'error') . '">

<p>' . $errors["{$error}"] . '</p>

</div>';

}

}

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title> Login to access dashboard </title>

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.1/css/all.min.css">

<link rel="stylesheet" href="resources/assets/css/login\_styles.css">

</head>

<body>

<div class="container" id="signIn">

<h1 class="form-title">Sign In</h1>

<?php

display\_error('login', true);

?>

<form method="POST" action="">

<div class="input-group">

<i class="fas fa-times"></i>

<select name="user\_type" id="" required>

<option value="">Select User</option>

<option value="lecture">Lecture</option>

<option value="administrator">Admin</option>

</select>

</div>

<div class="input-group">

<i class="fas fa-envelope"></i>

<input type="email" name="email" id="email" placeholder="Email" required>

<?php

display\_error('email');

?>

</div>

<div class="input-group password">

<i class="fas fa-lock"></i>

<input type="password" name="password" id="password" placeholder="Password" required>

<i id="eye" class="fa fa-eye"></i>

<?php

display\_error('password')

?>

</div>

<p class="recover">

<a href="#">Recover Password</a>

</p>

<input type="submit" class="btn" value="Sign In" name="login">

</form>

<p class="or">

----------or--------

</p>

<div class="icons">

<i class="fab fa-google"></i>

<i class="fab fa-facebook"></i>

</div>

</div>

<script src="resources/assets/javascript/script.js"></script>

</body>

</html>

**resources/pages/lecture/handle\_attendance.php**

* Handles uploading or saving attendance data to the database.

<?php

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

$attendanceData = json\_decode(file\_get\_contents("php://input"), true);

$response = [];

if ($attendanceData) {

try {

$sql = "INSERT INTO tblattendance (studentRegistrationNumber, course, unit, attendanceStatus, dateMarked)

VALUES (:studentID, :course, :unit, :attendanceStatus, :date)";

$stmt = $pdo->prepare($sql);

foreach ($attendanceData as $data) {

$studentID = $data['studentID'];

$attendanceStatus = $data['attendanceStatus'];

$course = $data['course'];

$unit = $data['unit'];

$date = date("Y-m-d");

$stmt->execute([

':studentID' => $studentID,

':course' => $course,

':unit' => $unit,

':attendanceStatus' => $attendanceStatus,

':date' => $date

]);

}

$response['status'] = 'success';

$response['message'] = "Attendance recorded successfully for all entries.";

} catch (PDOException $e) {

$response['status'] = 'error';

$response['message'] = "Error inserting attendance data: " . $e->getMessage();

}

} else {

$response['status'] = 'error';

$response['message'] = "No attendance data received.";

}

echo json\_encode($response);

}

**resources/assets/javascript/face\_logics/script.js**

* JavaScript logic for facial recognition—likely includes face-api.js functions.

var labels = [];

let detectedFaces = [];

let sendingData = false;

function updateTable() {

var selectedCourseID = document.getElementById("courseSelect").value;

var selectedUnitCode = document.getElementById("unitSelect").value;

var selectedVenue = document.getElementById("venueSelect").value;

var xhr = new XMLHttpRequest();

xhr.open("POST", "resources/pages/lecture/manageFolder.php", true);

xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");

xhr.onreadystatechange = function () {

if (xhr.readyState === 4 && xhr.status === 200) {

var response = JSON.parse(xhr.responseText);

if (response.status === "success") {

labels = response.data;

if (selectedCourseID && selectedUnitCode && selectedVenue) {

updateOtherElements();

}

document.getElementById("studentTableContainer").innerHTML =

response.html;

} else {

console.error("Error:", response.message);

}

}

};

xhr.send(

"courseID=" +

encodeURIComponent(selectedCourseID) +

"&unitID=" +

encodeURIComponent(selectedUnitCode) +

"&venueID=" +

encodeURIComponent(selectedVenue)

);

}

function markAttendance(detectedFaces) {

document.querySelectorAll("#studentTableContainer tr").forEach((row) => {

const registrationNumber = row.cells[0].innerText.trim();

if (detectedFaces.includes(registrationNumber)) {

row.cells[5].innerText = "present";

}

});

}

function updateOtherElements() {

const video = document.getElementById("video");

const videoContainer = document.querySelector(".video-container");

const startButton = document.getElementById("startButton");

let webcamStarted = false;

let modelsLoaded = false;

Promise.all([

faceapi.nets.ssdMobilenetv1.loadFromUri("models"),

faceapi.nets.faceRecognitionNet.loadFromUri("models"),

faceapi.nets.faceLandmark68Net.loadFromUri("models"),

])

.then(() => {

modelsLoaded = true;

console.log("models loaded successfully");

})

.catch(() => {

alert("models not loaded, please check your model folder location");

});

startButton.addEventListener("click", async () => {

videoContainer.style.display = "flex";

if (!webcamStarted && modelsLoaded) {

startWebcam();

webcamStarted = true;

}

});

function startWebcam() {

navigator.mediaDevices

.getUserMedia({

video: true,

audio: false,

})

.then((stream) => {

video.srcObject = stream;

videoStream = stream;

})

.catch((error) => {

console.error(error);

});

}

async function getLabeledFaceDescriptions() {

const labeledDescriptors = [];

for (const label of labels) {

const descriptions = [];

for (let i = 1; i <= 5; i++) {

try {

const img = await faceapi.fetchImage(

`resources/labels/${label}/${i}.png`

);

const detections = await faceapi

.detectSingleFace(img)

.withFaceLandmarks()

.withFaceDescriptor();

if (detections) {

descriptions.push(detections.descriptor);

} else {

console.log(`No face detected in ${label}/${i}.png`);

}

} catch (error) {

console.error(`Error processing ${label}/${i}.png:`, error);

}

}

if (descriptions.length > 0) {

detectedFaces.push(label);

labeledDescriptors.push(

new faceapi.LabeledFaceDescriptors(label, descriptions)

);

}

}

return labeledDescriptors;

}

video.addEventListener("play", async () => {

const labeledFaceDescriptors = await getLabeledFaceDescriptions();

const faceMatcher = new faceapi.FaceMatcher(labeledFaceDescriptors);

const canvas = faceapi.createCanvasFromMedia(video);

videoContainer.appendChild(canvas);

const displaySize = { width: video.width, height: video.height };

faceapi.matchDimensions(canvas, displaySize);

setInterval(async () => {

const detections = await faceapi

.detectAllFaces(video)

.withFaceLandmarks()

.withFaceDescriptors();

const resizedDetections = faceapi.resizeResults(detections, displaySize);

canvas.getContext("2d").clearRect(0, 0, canvas.width, canvas.height);

const results = resizedDetections.map((d) => {

return faceMatcher.findBestMatch(d.descriptor);

});

detectedFaces = results.map((result) => result.label);

markAttendance(detectedFaces);

results.forEach((result, i) => {

const box = resizedDetections[i].detection.box;

const drawBox = new faceapi.draw.DrawBox(box, {

label: result,

});

drawBox.draw(canvas);

});

}, 100);

});

}

function sendAttendanceDataToServer() {

const attendanceData = [];

document

.querySelectorAll("#studentTableContainer tr")

.forEach((row, index) => {

if (index === 0) return;

const studentID = row.cells[0].innerText.trim();

const course = row.cells[2].innerText.trim();

const unit = row.cells[3].innerText.trim();

const attendanceStatus = row.cells[5].innerText.trim();

attendanceData.push({ studentID, course, unit, attendanceStatus });

});

const xhr = new XMLHttpRequest();

xhr.open("POST", "handle\_attendance", true);

xhr.setRequestHeader("Content-Type", "application/json");

xhr.onreadystatechange = function () {

if (xhr.readyState === 4) {

if (xhr.status === 200) {

try {

const response = JSON.parse(xhr.responseText);

if (response.status === "success") {

showMessage(

response.message || "Attendance recorded successfully."

);

} else {

showMessage(

response.message ||

"An error occurred while recording attendance."

);

}

} catch (e) {

showMessage("Error: Failed to parse the response from the server.");

console.error(e);

}

} else {

showMessage(

"Error: Unable to record attendance. HTTP Status: " + xhr.status

);

console.error("HTTP Error", xhr.status, xhr.statusText);

}

}

};

xhr.send(JSON.stringify(attendanceData));

}

function showMessage(message) {

var messageDiv = document.getElementById("messageDiv");

messageDiv.style.display = "block";

messageDiv.innerHTML = message;

console.log(message);

messageDiv.style.opacity = 1;

setTimeout(function () {

messageDiv.style.opacity = 0;

}, 5000);

}

function stopWebcam() {

if (videoStream) {

const tracks = videoStream.getTracks();

tracks.forEach((track) => {

track.stop();

});

video.srcObject = null;

videoStream = null;

}

}

document.getElementById("endAttendance").addEventListener("click", function () {

sendAttendanceDataToServer();

const videoContainer = document.querySelector(".video-container");

videoContainer.style.display = "none";

stopWebcam();

});

**resources/lib/php\_functions.php**

* Contains reusable PHP functions (e.g., session checks, redirections).

<?php

function user()

{

if (isset($\_SESSION['user'])) {

return (object) $\_SESSION['user'];

}

return null;

}

function getFacultyNames()

{

global $pdo;

$sql = "SELECT \* FROM tblfaculty";

$stmt = $pdo->query($sql);

$result = $stmt->fetchAll(PDO::FETCH\_ASSOC);

$facultyNames = array();

if ($result) {

foreach ($result as $row) {

$facultyNames[] = $row;

}

}

return $facultyNames;

}

function getLectureNames()

{

global $pdo;

$sql = "SELECT Id, firstName, lastName FROM tbllecture";

$stmt = $pdo->query($sql);

$result = $stmt->fetchAll(PDO::FETCH\_ASSOC);

$lectureNames = array();

if ($result) {

foreach ($result as $row) {

$lectureNames[] = $row;

}

}

return $lectureNames;

}

function getCourseNames()

{

global $pdo;

$sql = "SELECT \* FROM tblcourse";

$stmt = $pdo->query($sql);

$result = $stmt->fetchAll(PDO::FETCH\_ASSOC);

$courseNames = array();

if ($result) {

foreach ($result as $row) {

$courseNames[] = $row;

}

}

return $courseNames;

}

function getVenueNames()

{

$sql = "SELECT className FROM tblvenue";

$result = fetch($sql);

$venueNames = array();

if ($result) {

foreach ($result as $row) {

$venueNames[] = $row;

}

}

return $venueNames;

}

function getUnitNames()

{

$sql = "SELECT unitCode,name FROM tblunit";

$result = fetch($sql);

$unitNames = array();

if ($result) {

foreach ($result as $row) {

$unitNames[] = $row;

}

}

return $unitNames;

}

function showMessage(): void

{

if (isset($\_SESSION['message'])) {

echo " <div id='messageDiv' class='messageDiv' >{$\_SESSION['message']}</div>";

echo `<script>

var messageDiv = document.getElementById('messageDiv');

messageDiv.style.opacity = 1;

setTimeout(function() {

messageDiv.style.opacity = 0;

}, 5000);

</script>`;

unset($\_SESSION['message']);

}

}

function total\_rows($tablename)

{

global $pdo;

$stmt = $pdo->query("SELECT \* FROM {$tablename}");

$total\_rows = $stmt->rowCount();

echo $total\_rows;

}

function fetch($sql)

{

global $pdo;

$stmt = $pdo->query($sql);

$result = $stmt->fetchAll(PDO::FETCH\_ASSOC);

return $result;

}

function fetchStudentRecordsFromDatabase($courseCode, $unitCode)

{

$studentRows = array();

$query = "SELECT \* FROM tblattendance WHERE course = '$courseCode' AND unit = '$unitCode'";

$result = fetch($query);

if ($result) {

foreach ($result as $row) {

$studentRows[] = $row;

}

}

return $studentRows;

}

function js\_asset($links = [])

{

if ($links) {

foreach ($links as $link) {

echo "<script src='resources/assets/javascript/{$link}.js'>

</script>";

}

}

}