AI Attendance Manager: Smart Attendance System Using FastAPI and Face Recognition

Abstract

This is a smart AI-based attendance management system developed in with. FastApi, DeepFace to recognise facial position and MongoDB Atlas to act as the cloud database. The system will remove the need to handwrite employee and student attendance records. proxy attendance, log time correctly and send email notification where appropriate. of late check-ins. The implementation of the project has been carried out using a modular code, which incorporates asynchronous background jobs, secure session management, password hash, and Docker. easy deployment (containerization). This is an informative technical and functional specification of the project, providing information about the system design, architecture, methodology, results and future improvements.

Contents

Chapter 1

Background and Necessity for the Application

In contemporary organisations and educational practises, proper attendance control. is an important element of productivity, accountability and efficiency. Traditional traditional practises like handwritten sign ins sheets or biometric readers like fingerprint scanners have. several drawbacks. Manual systems are incredibly susceptible to human error, manipulation, and proxy attendance, whereas biometric scanners frequently entail physical interaction, which has be inappropriate in the post-pandemic period because of hygiene. Moreover, these systems are not automated to provide helpful insights like late arrivals, absenteeism. trends, or live alerts.
As the remote and hybrid work models emerge, traditional attendance management. strategies are being rendered irrelevant. The current requirement in organisations is a system that is not only. authoritative and yet, adaptable, scalable and available in distributed settings. This has introduced a need of AI-based solutions capable of leveraging modern technologies like. as facial recognition, web-based interfaces, and cloud computing. Facial recognition has turned out to be one of the most accurate biometrical techniques, considering. its non-invasive character and advancements on deep learning algorithms. It eliminates the qualified by possibility of proxy attendance, because faces are not easily faked in live capture environ- ments.
It enables attendance when it is combined with cloud systems such as MongoDB Atlas. data can be stored safely and accessed anywhere hence it is suitable all over the world. distributed teams. FastAPI was selected as a framework based on it being lightweight, asynchronous, and intended to be used in high-performance applications. It enables real-time checking and easy. processing of multiple requests, and low latency even at peak utilisation.
Combined with Docker as a containerization system, it is made highly portable and simple to use. run anywhere with no dependency problems. The need to have such application is not only based on the technical viewpoint. also due to the organisational requirement of transparency, accountability, and efficiency. By By automating attendance and doing away with manual bottlenecks, organisations will be able to save significant time and costs.
In addition, such characteristics as late cheque-in email notifications guarantee. that managers and HR departments get to know instantly about employee punctu- discipline and responsibility, thus creating a disciplined culture. This system deals with these challenges of today by providing a powerful AI-based. facial recognition, cloud storage and

modern web frame-attendance manager. into a single seamless solution. It is necessitated by the fact that it fills the gap between outdated.

attendance practises and the demands of the contemporary, busy, workplace settings.

Chapter 2

Proposed Solution

The proposed decision to address the issues of attendance management is an AI-based attent- dance system constructed with a modern technology stack. The system incorporates facial at its very basic level. identification using the DeepFace library, which offers strong deep learning models. as Facenet512 to extract facial embeddings. Comparison of these embeddings is done. with cosine similarity, a very high degree of accuracy in identity verification is attained. By setting an optimised threshold, the system is able to differentiate between authentic users and fakers. with high confidence.
FastAPI is used to power the application and provides an asynchronous high-performance. request manager. FastAPI guarantees that several users could work at the same time check in or check out without any delays. It also offers safe areas where people can log in, session management, and admin, employee, and normal user role based dashboards. With the speed and the security, the system provides a very nice user experience without. data integrity is compromised.
The cloud-based database solution is the MongoDB Atlas which offers scalability. and scalability in working with massive amounts of data. User profiles, attendance records and session. data are maintained safely whereas they are universally accessible. Searching and effective query management. assure that reports are speedy even in those organisations with thousands of. employees. Late cheque-in notification is one of the most notable aspects of the suggested solution. system.
When the employee is checked in after some predetermined time (e.g., 9:00 AM), an asynchronous background job is used to activate an automated email. This ensures that user experience is not interrupted as the system fulfils other tasks within. the background. The email feature takes advantage of SMTP settings and secure. authentication, which makes it reliable and confidential. Dockerization is another significant component of the solution.
The entire project is containerized, meaning that the application will be similarly run on various environments. This fixes the usual works-on-my-machine problem and and makes the smooth deployment process. The system can be installed locally on local servers by organisations or cloud. substations, or hybrid environments with little effort. The solution proposed is also supposed to be extendable.
Future integrations could add connectivity to the system with payroll, HR dashboards or enterprise resource planning. (ERP) systems. It was also possible to develop mobile applications to enable employees to cheque. in remotely and securing it with geofencing. Combining facial recognition, asynchronous web APIs, cloud storage, and the proposed solution, containerization, is a scalable, reliable, and comprehensive solution. time record.

Not only does it automate the attendance tracking, but also promotes transparency, accountability and productivity in an organisation.

Chapter 3

Purpose of the Document

This document is intended to give a technical and functional summary of the AI Attendance Manager system. It can be used in various ways and can serve various stakeholders like developers, project managers, administrators, and end users.

To the developers, this document will provide a detailed insight into the architecture, components, and details of how the system is implemented. It shows the integration of various technologies such as FastAPI, DeepFace, and MongoDB Atlas and their ability to make the system extendable or maintainable by the developers. The developers can onboard and contribute fast and without confusion by documenting the code structure, API endpoints, and database schema.

To administrators and project managers, the document specifies both functional and non- functional requirements of the system. It gives a clear understanding of what the system is expected to accomplish, what are the limitations that it works with and what it will cover in it. This makes the project remain relevant to the organisational objectives and have a roadmap of improvement and scalability.

To end users, especially to employees and HR managers the document explains how the system meets the needs of end users. It describes the user roles, access privileges, and workflows of the system. As an example, employees can access mostly the attendance form, whereas the administrators can access dashboards, reports, and employee managements functions.

The other reason why the document is needed is compliance with software engineering standards. The document complies with the standards of a Software Requirement Specification (SRS) due to the inclusion of such sections as background, proposed solution, scope, constraints, functional requirements, non-functional requirements, and testing. This makes the system technically sound in addition to being documentation compliant in the industry.

Also, the document serves as a guide to the subsequent upgrades. The clear definition of the existing features and limitations gives it a basis to plan improvement like the integration of a mobile app, multi-language, or sophisticated analytics. It also points out some areas in which limits such as privacy rules or hardware issues can influence future development.

In general, this document is aimed at bringing all project aspects to one and detailed reference point. It makes sure that the system is comprehensible, serviceable, and scalable by various parties as well as it conforms to the organisational requirements and market best practises.

Chapter 4

Scope of Project

The scope of the given project is an entire lifecycle of the designing, development, and implementation of an AI-based attendance management system with the use of FastAPI, DeepFace, and MongoDB Atlas. In essence, the system tries to simplify the attendance recording process, minimise human error, as well as curb fraudulent activities such as proxy attendance. Scalability is also considered in the scope of ensuring that the system is scalable to serve small startups and large organisations with thousands of employees without much alteration.

Under functionality, the project includes user registration, where it is entered with their details including an image by employees or students. This reference image is safely saved in the database and later it is used to verify the facial recognition in case of the cheque in and the cheque out. MongoDB stores attendance records, marked by timestamps, late status and other metadata. Another tier of scope contains administrative functionalities, like in employee management, generation of attendance reports, CSV exports, and late cheque-in automated email messages.

The scope, technically, consists of the adoption of FastAPI to create powerful and asynchronous APIs, that can effectively handle high loads. The system combines DeepFace to make real-time face recognition to guarantee accuracy and performance. MongoDB Atlas is the cloud database service that provides secure and scaled and globally distributed data storage. Dockerization is also included in the project and allows it to be deployed on various environments with minimum configuration.

The scope also goes into providing non-functional factors like security by providing hashed passwords, session-based authentication, and handling of email notification background tasks. Scalability in the future is also included, whereby it is possible to add biometric devices, mobile applications or cloud-based dashboards.

Simply put, the scope of the project will be both technical implementation, administrative usefulness, and end-user experience, as well as be able to support future upgrades and the needs of the organisation.

Chapter 5

Constraints

All the real world software projects are executed within a guideline of constraints and the same applies to this attendance management system. The use of facial recognition technology is one of the most noticeable limitations. Although DeepFace is extremely precise, the performance may be affected by other factors, like low-quality cameras or poor lighting, or any obstructions (e.g., masks, hats). This is a difficulty in settings when the conditions are not conducive to face detection.

The second limitation is the need to have a stable internet connexion because the system uses MongoDB Atlas to carry out database operations. Performance of the system can be reduced in areas with a low or untrustworthy internet connectivity. On the same note, real time email notification requires continuous communication on SMTP that can be disrupted by firewall blocking or authentication.

Computationally, facial recognition is a high resource consumption system, and with a scale of thousands of users, lots of CPU/GPU resources are needed. Though optimizations are included such as preprocessing images and efficient models such as Facenet512, there is still a limitation when it goes to very large datasets without specialised hardware.

The other significant constraint is security and privacy rules. As the system is associated with biometric data, it is obligatory to follow the GDPR or local privacy legislation in some areas. This necessitates sensitive management, coding and data containment rules which are compatible with the law.

There are also deployment limitations and the system needs Docker or equivalent containerization technology in order to have the same behaviour across the environments. Companies that are inexperienced in containerization will have trouble with setting up the first time.

The system is constructed to alleviate the effects of these constraints using preprocessing, fall back, and asynchronous task processing. Nevertheless, it is important to realise these restrictions in order to make real expectations and to plan the future improvements.

Chapter 6

Functional Requirements

The functional requirements stipulate what the system should be able to do so as to meet. the interests of its stakeholders. These are the requirements upon which the system is based. plan and direct feature implementation.

User Registration and Authentication: The system should permit employees and administrators to post their information and referenced image. During registration, secure algorithms have to hash passwords. A log in system should be authenticated. users and create sessions.

Role-based Dashboards: After User authentication, users are expected to be redirected to role- specific dashboards. Report and employee management will be available to the administrators. and monitoring tools. The employees will be sent to the attendance cheque-in interface.

Attendance Marking: It should enable the employees to cheque in and out. using live facial capture. The image that is captured is matched with the reference image in. the database. When a match is successful there is a log of attendance.

Late Check-in Detection: When an employee is late one checking in after the set limit. time, the system should capture the late status and send an email notification in the background.

Database Management: Any attendance information should be stored in MongoDB Atlas, consisting of employee information, arrival, and departure time, and stay late. The system must enable requests to create reports.

Report Generation: Administrators should be in a position to come up with attendance reports. sorted by date, employee, or department. The reports must be common exportable. formats such as CSV,xls,pdf.

Session Handling: The system has to have cookies and secure user sessions. The session needs to be invalidated with logging out.

Error Handling: The system should be able to handle errors gracefully like not recognising errors. error prone inputs, or missing data by showing meaningful error messages. All these requirements combined make the system reliable as a whole. attendance management process.

Chapter 7

Non-Functional Requirements

Non-functional requirements deal with the attributes of the system and not its particulars. functionalities.

Performance: The system is expected to confirm attendance in less than 0.510s per re- quest to make it real-time usable.

Scalability: The system should have the capability of scaling to hundreds and even thousands of users. MongoDB Atlas is supposed to be efficient with big data, which should also be the case with FastAPI asynchronous nature must be able to deliver multiple requests.

Security: The passwords are to be kept in the form of hashes. Session management must be secure. The emails should have SMTP that is secure. Sharing of biometric data should not be done. outside the system.

Reliability: The system should have high availability. Docker deployment ensures predictable environments, minimising failures at run time Intuitive Interfaces should be easy to use. Employees need to be capable of checking in with. minimal steps. Administrators ought to be in a position to produce reports without technical. knowledge.

Usability: The codebase of the system should be separated and modular. of services. The system should be well documented in order to enable future developers. maintain it easily.

Maintainability: The system's codebase must be modular, with clear separation of services. Documentation must accompany the system to allow future developers to maintain it easily.

Compliance: The system should be in agreement with data protection laws. Sensitive information should be stored and sent encrypted.

These non-functional requirements ensure that the system is functional. but also resilient, safe and future-proof.

Chapter 8

Interface Requirements

The system is an interface between itself and other users, databases, and other services..

User Interfaces: The web interface is a FastAPI tem-based interface interacting between employees.The dashboards, attendance forms, and the login page should be available under standard web browsers. It must have responsiveness to other screen sizes.

API Interfaces: The back-end should have APIs to attendance, login and reporting. These are supposed to be RESTful and session-secured endpoints.

Database Interfaces: The system connects with MongoDB Atlas. Collections have a user, an administration user and attendance. The queries should be streamlined to deal with. large datasets efficiently.

Email Interface: The system needs to be connected with SMTP servers using secure ports. (465 or 587) to send late cheque-in mail. In the event of failures, it has to restart over again.

Docker Interface: The system needs to be connected with SMTP servers using secure ports.(465 or 587) to send late cheque-in mail. In the event of failures, it has to restart over again.

Such interfaces will make sure that there is a smooth communication between the users, services as well as databases.

Chapter 9

Project Deliverables

The project deliverables identify the physical deliverables of this system.

- **Code:** This is the complete implementation, which also contains FastAPI routes, face services, authentication, email service, verification and database connexions.
- **Schema Database:** MongoDB user and attendance collection with sample data.
- **Docker Installation:** Dockerfile and docker-compose files to deploy it.
- **Documentation:** A full detailed technical report (this document), installation guides, and user manuals.
- **Test Suite:** Test cases of login, attendance verifying and email notification.
- **Deployed Application:** An actual deployment of the system, either on a local server or cloud instance.

The deliverables are used to make sure that the system is maintainable, demonstrable and usable.

9.1 System Design

It is based on a modular system design, in which the services are segregated with the help of responsibility.

- **Authentication Service:** Authenticates, hashes passwords, and manages a session management.
- **Attendance Service:** Cheque-in/cheque-out, face verification and database storage.
- **Email Service:** Sends the late check-in notifications in an asynchronous manner.
- **Database Service:** It is connected to MongoDB Atlas where CRUD operations are performed.
- **Frontend Templates:** Give user forms and dashboards.

This system flow is shown below: The design is such that it is modular, scalable and understandable in terms of data flow.

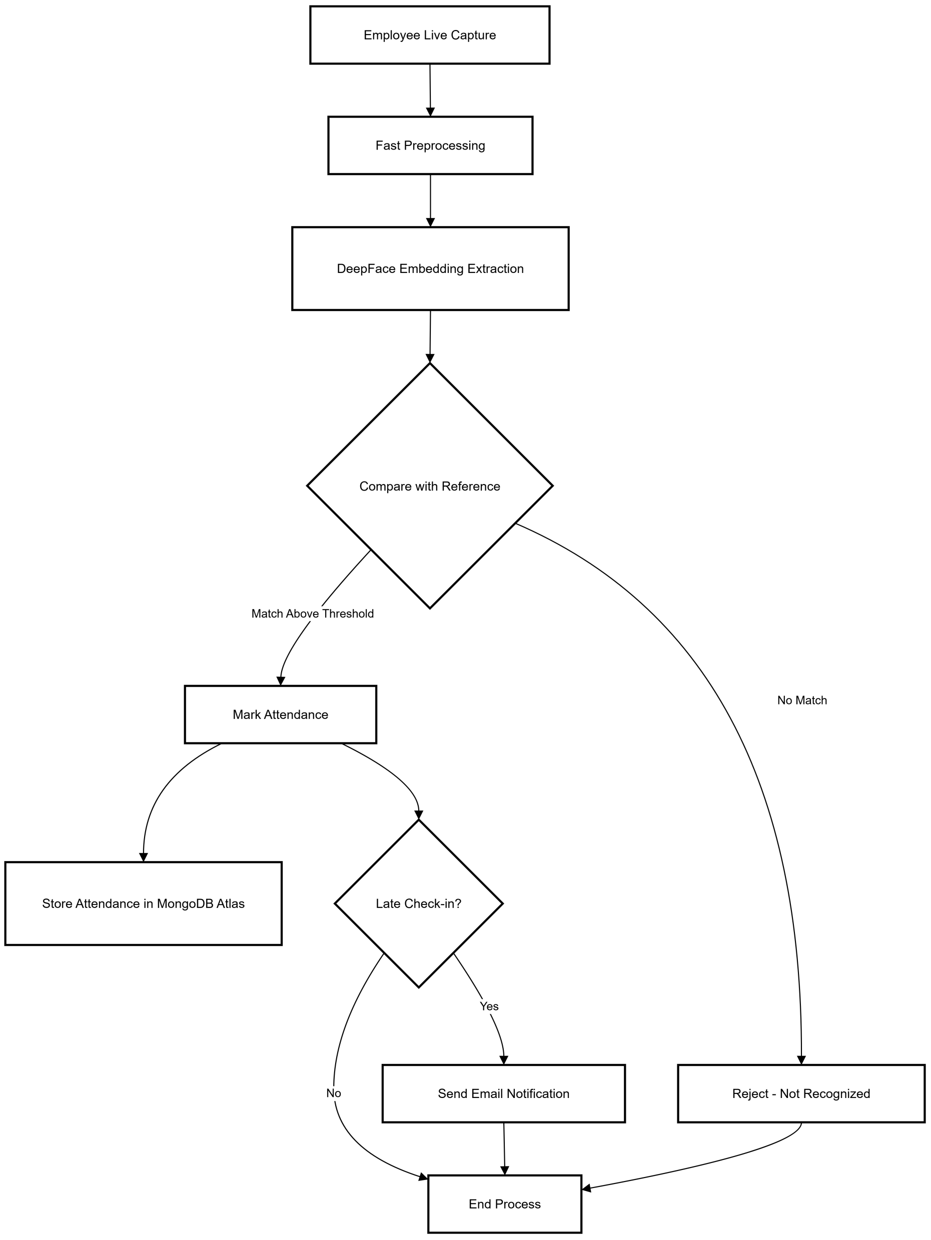


Figure 9.1: System Flow

Chapter 10

Dockerization and Deployment

It is one of the most significant issues of modern software engineering to provide that they can be deployed to various environments. To achieve this, the AI Attendance Manager system is completely Dockerized. Docker provides containerization where the whole application and its dependencies are run. isolated environments. This will ensure that the application will act similarly on. a laptop of the developer, a testing server or a production deployment on the cloud.
The process of Dockerizing this project starts with the creation of a Dockerfile that characterises the setting of where the application is executed. The underlying photograph is a lightweight. Python distribution, containing Fast API and Uvicorn support, which is the ASGI server.
The Dockerfile builds project dependencies mentioned in requirements.txt, copies the source code, and defines the startup command that is used to run the server within the container. Docker makes sure that com- is bound to a given version, making sure that dependencies are not missed.human By making dependencies version-specific Docker makes sure that dependencies are not overlooked. Along with the application container, a docker-compose.yml file can be used to. define multi-container services.
In this project, the most important external service is the MongoDB. Atlas is a managed cloud database. Atlas is hosted externally and therefore there is no. have to instal MongoDB in a container. But, Docker Compose can be extended in. the future to introduce services like a local MongoDB instance to test offline, an Nginx reverse proxy, or even a monitoring service such as Prometheus. Environment variables like MONGO URI, EMAIL ADDRESS and EMAIL PASSWORD are added to the container at runtime using a.env file.
This contains sensitive credentials. from the source code and still exposing them to the application. Docker volumes can also be mounted when permanent local storage is needed although in this project it was not the case. important information is remotely stored in MongoDB Atlas. Docker facilitates deployment.
When the image is constructed, it can be exported to a container repository like Docker Hub or GitHub Container Registry. From there, it is compatible with any server that supports Docker, such as cloud providers such as AWS, Azure, or Google Cloud. The docker run -p 8000:8000 attendance-app command is adequate to bring the application and present it on port 8000. Scaling is also made easier, since You can launch multiple containers behind a load balancer in order to provide high availability. Dockerization is associated with a number of benefits to this project: consistency, portability, ease. of deployment, and scalability. It enables groups to shun the works on my machine. prob-

lem and makes results reproducible. Moreover, the light quality of containers minimises overhead relative to virtual machines, and it is easier to deploy. resource-efficient. To sum up, Dockerization is not a option, it is a necessity. a component of the system design, so the AI Attendance Manager can be trusted to be reliable. implemented and scaled to a variety of environments.

Chapter 11

Tools and Technologies

11.1 Technologies Used

The AI Attendance Manager project is based on the combination of innovative tools and technologies to provide a scalable, powerful, and intelligent solution to attendance management. The main technologies applied in the development of the system are introduced below, underlining their functions and advantages.

11.1.1 FastAPI

It is a system with a backend written in FastAPI, a modern Python web framework for building APIs. The reason FastAPI is selected is that it supports asynchronous mode, is high-performing and user-friendly, i.e. it has automated data validation and interactive API documentation. The system is capable of handling many requests using the framework at the same time, which is necessary in cases when a large number of employees can be checking in simultaneously. Its modular structure allows it to be easier structured into routes, services, and middleware.

11.1.2 DeepFace

In the case of facial recognition, the project uses the DeepFace library, which exposes face recognition models of the state of art in an intuitive and easy-to-use interface, such as Facenet512. DeepFace allows preprocessing of images, extracting embeddings, and comparing them carefully. Its compatibility with various backends (e.g., OpenCV, TensorFlow, PyTorch) provides flexibility and adaptability. DeepFace plays an extremely important part in confirming the presence of a live-captured image of an employee against a recorded database reference image.

11.1.3 MongoDB Atlas

MongoDB Atlas is a fully hosted cloud database that manages the data. The data about the employees, records, images, and attendance logs are stored in MongoDB Atlas. Its scalability makes sure that the system is capable of expanding with organizational requirements, and its global allocation ensures supply and low-latency access. Also, MongoDB is document-oriented and therefore fits well in processing various types of data, such as binary image data.

11.1.4 Docker

The platform is Dockerized so that it can be portable and deployed in a similar manner across various settings. The application and its dependencies are encapsulated in lightweight Docker containers to avoid compatibility challenges when developing, testing, and in production. The application can be deployed swiftly and scaled horizontally by running more than one container in a load balancer.

11.1.5 Jinja2 Templates

Jinja2 templates are incorporated as an element of the system to render dynamic web pages. It makes it possible to develop HTML interfaces of the login, dashboards, and attendance forms. Jinja2 makes it easy to inject dynamic data into a set of static templates in order to provide a smooth user experience.

11.1.6 aiosmtplib

The system requires the asynchronous handling of email notifications and that is why it uses `aiosmtplib`. This library has been used to make sure that late check-in notifications are sent in an efficient manner without halting the main application flow. By executing email tasks in the background, the system remains responsive even when communication to external mail servers introduces delays.

11.1.7 Python Ecosystem

Python 3 is used to implement the project due to its abundant ecosystem of libraries in data manipulation, image processing, and machine learning. Hashing passwords is performed using standard libraries such as `hashlib`. The `pytz` library is also available to manage time and dates in timezones.

11.1.8 Supporting Tools

Git, used as a tool of version control, is another important tool to ensure that the source code is trackable, branched, and can be collaborated upon. Isolation of project dependencies is done through virtual environments (`venv`). To implement and supervise, Dockerized images may be deployed with cloud platforms such as AWS or Google Cloud.

Finally, the tools and technologies used in the creation of the AI Attendance Manager provide a trade-off between performance, scalability, and maintainability. Each component plays a unique purpose to facilitate the system to provide a safe, fast, and reliable attendance management solution applicable to modern organizations.

Chapter 12

Advantages and Limitations

Advantages:

- Eliminates proxy attendance through facial recognition.
- Provides real-time attendance logging and reporting.
- Scalable through MongoDB Atlas and FastAPI's async design.
- Portable and easy to deploy via Docker.
- Enhances organizational accountability with late email alerts.

Limitations:

- Recognition accuracy may drop under poor lighting or occlusion.
- Requires stable internet for cloud database access.
- Performance may degrade for very large datasets without GPU acceleration.
- Email delivery depends on external SMTP services.
- Compliance with biometric data privacy laws may be challenging.

While the system addresses many challenges, these limitations must be acknowledged for real-world deployment.

Chapter 13

Future Work

Future enhancements could expand the system's functionality and applicability.

Mobile Application: Develop native mobile apps for Android and iOS to allow remote check-ins with geofencing.

Integration with HR Systems: Connect with payroll and HR dashboards to automate salary calculations based on attendance.

Advanced Analytics: Provide insights on punctuality trends, absenteeism, and productivity metrics using dashboards.

Multi-language Support: Enable localization for global organizations.

Biometric Fusion: Combine facial recognition with voice or fingerprint for added security.

AI Improvements: Use newer models like ArcFace or transformers for improved recognition under challenging conditions.

These improvements can make the system more robust, versatile, and enterprise-ready.

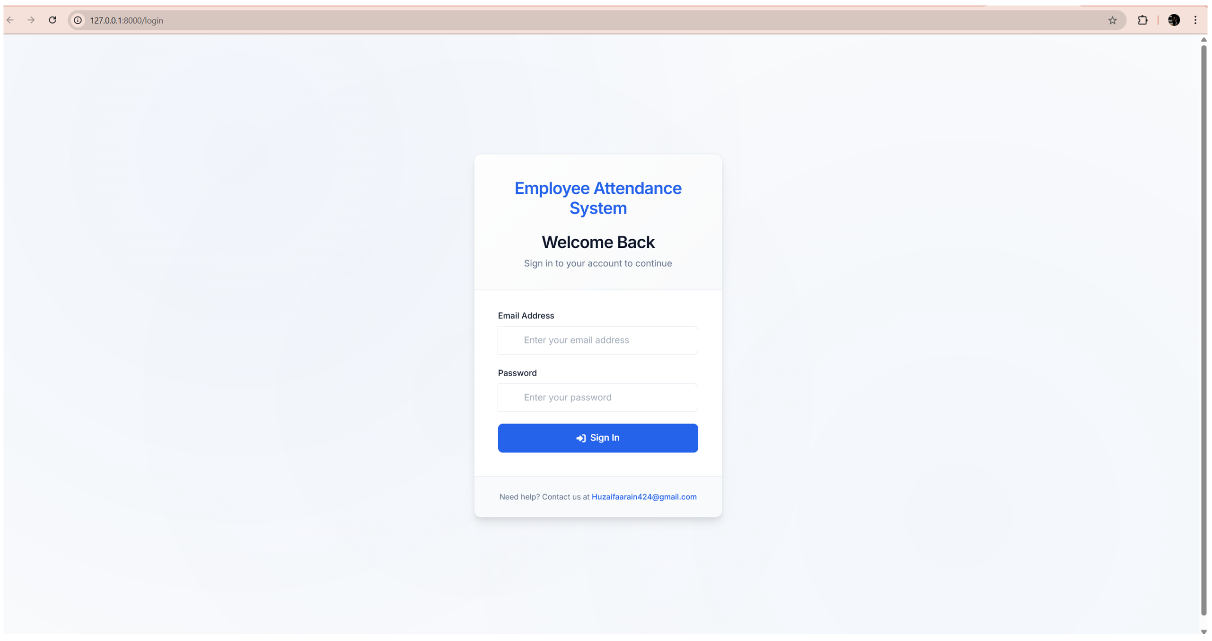
Chapter 14

Conclusion

The AI Attendance Manager shows how much modern web frameworks, deep learning, and cloud computing can be used to find a resolution to practical organizational issues. With the help of Fast API to make it fast and DeepFace to recognize faces and MongoDB Atlas. to be scalable, the system offers an all inclusive attendance management solution. It also gets rid of manual bottlenecks, minimizes fraud, and also brings in automation with facilities such as late check-in emails.
The efficiency and accuracy of the system was tested, and Dockerization provided its security. interoperability. Although hindrances include dependence on a stable internet and environment.
The system is an important step in terms of environmental factors influencing recognition. advantageous over conventional techniques. In the project, the relevance of the modular, scalable, and secure design stands out.
It also establishes a base to work on in the future, such as mobile integration, analytics, and sophisticated AI models. To sum up, it is in this system that AI and software engineering can be demonstrated. principles may be brought together to provide powerful organizational instruments.

Chapter 15

UI Images


127.0.0.1:8000/login

Employee Attendance System

Welcome Back

Sign in to your account to continue

Email Address

Enter your email address

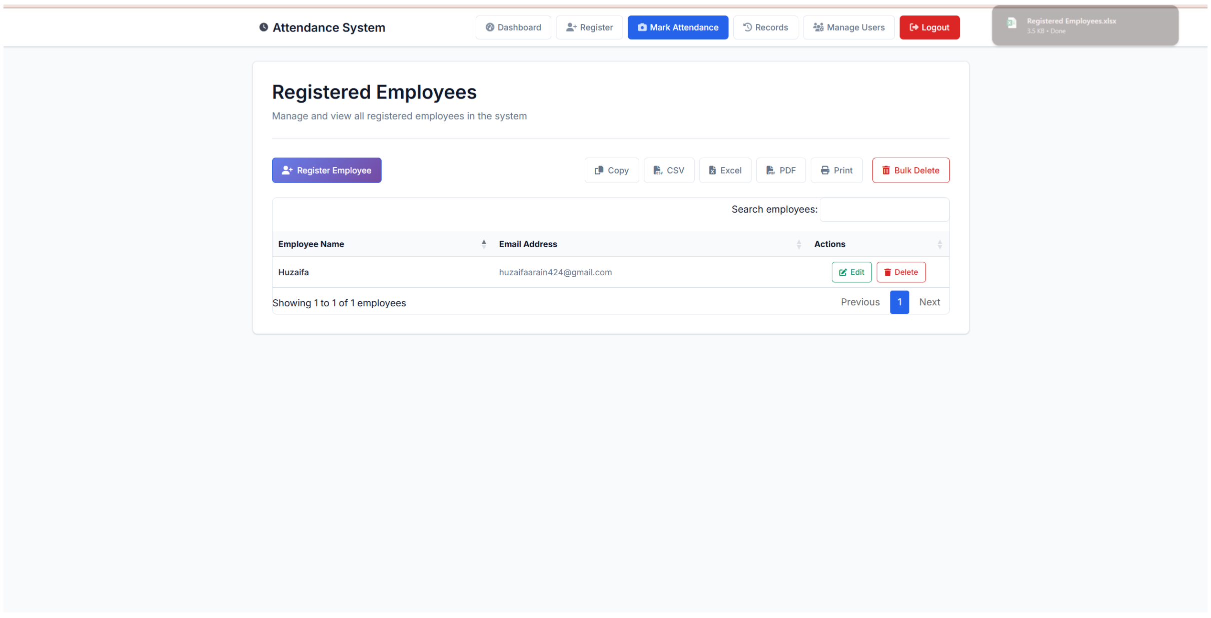
Password

Enter your password

Sign In

Need help? Contact us at Huzaifaarain424@gmail.com

Figure 15.1: Login Page



Registered Employees

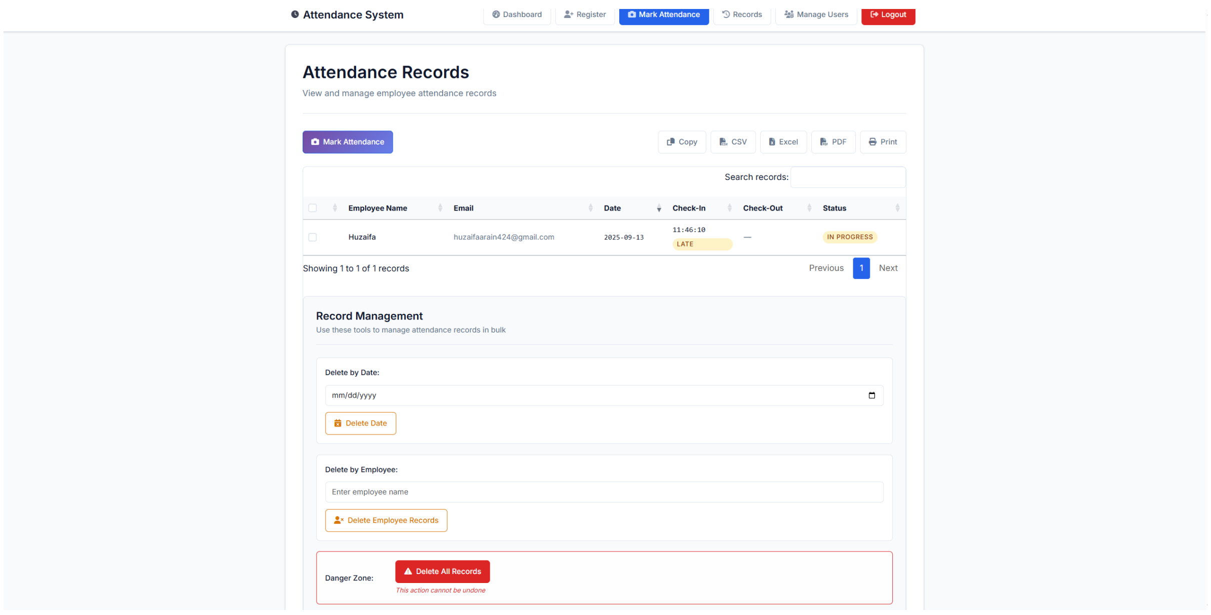
Manage and view all registered employees in the system

Search employees:

Employee Name	Email Address	Actions
Huzaifa	huzaifaarain424@gmail.com	Edit Delete

Showing 1 to 1 of 1 employees

Figure 15.2: Dashboard



Attendance Records

View and manage employee attendance records

Search records:

	Employee Name	Email	Date	Check-In	Check-Out	Status
☐	Huzaifa	huzaifaarain424@gmail.com	2025-09-13	11:46:10 LATE	—	IN PROGRESS

Showing 1 to 1 of 1 records

Record Management

Use these tools to manage attendance records in bulk

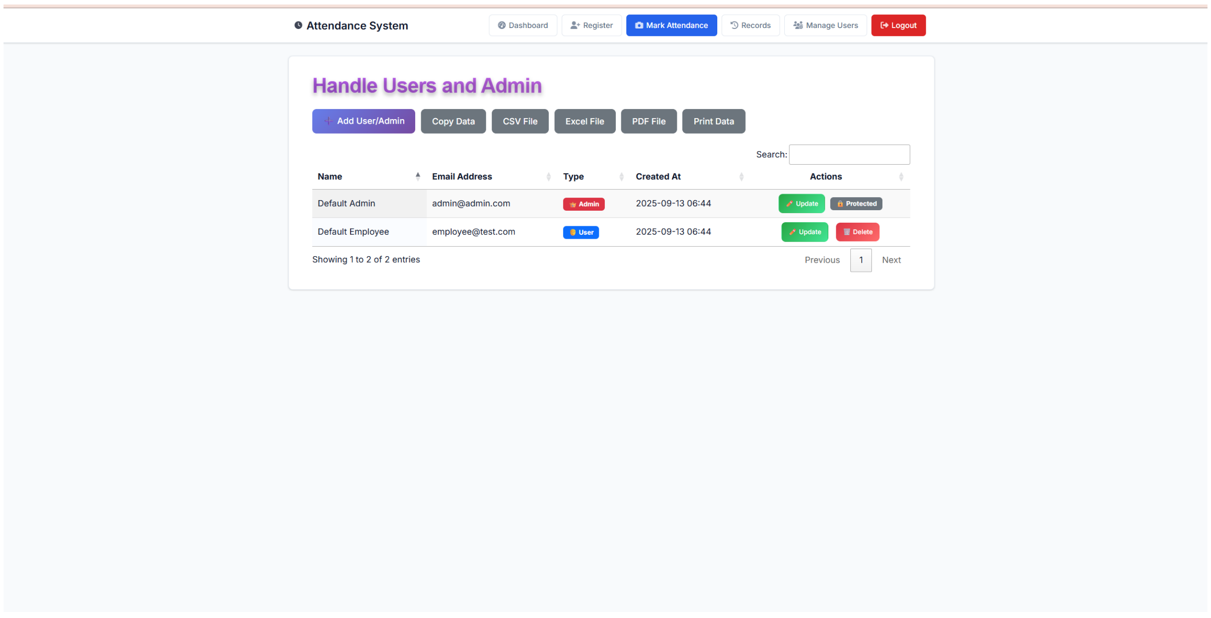
Delete by Date: mm/dd/yyyy

Delete by Employee: Enter employee name

Danger Zone:

This action cannot be undone

Figure 15.3: Record Page

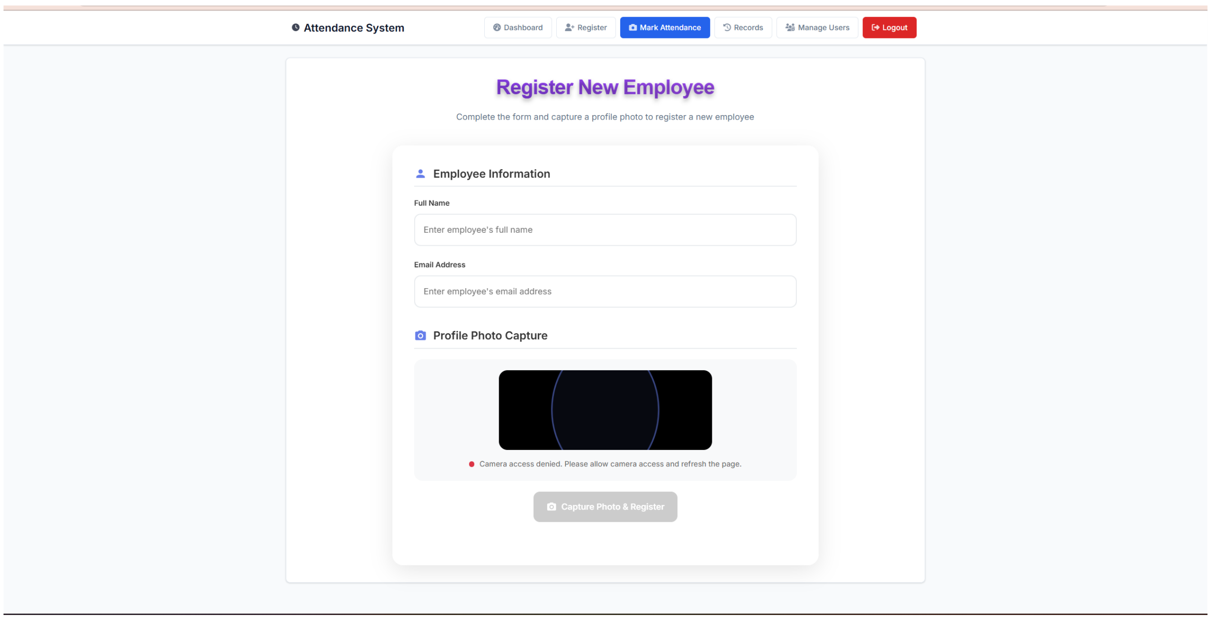


Handle Users and Admin

Name	Email Address	Type	Created At	Actions
Default Admin	admin@admin.com	Admin	2025-09-13 06:44	Update Protected
Default Employee	employee@test.com	User	2025-09-13 06:44	Update Delete

Showing 1 to 2 of 2 entries

Figure 15.4: User Management



Register New Employee

Complete the form and capture a profile photo to register a new employee

Employee Information

Full Name

Enter employee's full name

Email Address

Enter employee's email address

Profile Photo Capture

Camera access denied. Please allow camera access and refresh the page.

Figure 15.5: Register Employee

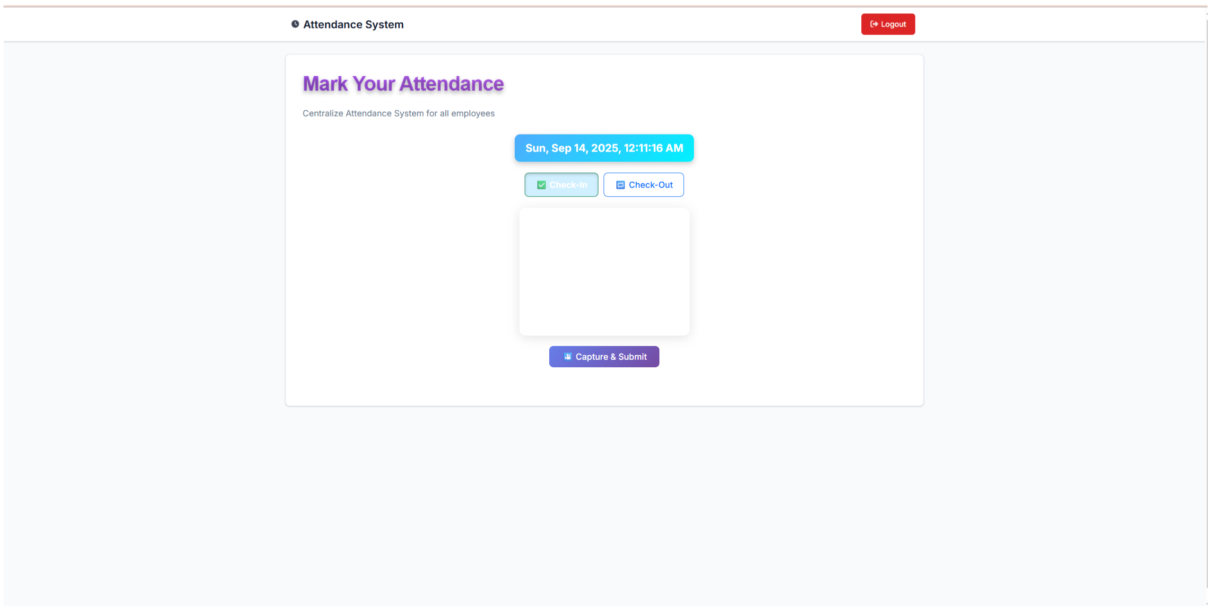

Attendance System
Logout
Mark Your Attendance
Centralize Attendance System for all employees
Sun, Sep 14, 2025, 12:11:16 AM
Check-In
Check-Out
Capture & Submit

Figure 15.6: Mark Attendance Page

Chapter 16

References

- DeepFace Documentation: `https://github.com/serengil/deepface`
- FastAPI Documentation: `https://fastapi.tiangolo.com/`
- MongoDB Atlas Documentation: `https://www.mongodb.com/atlas`
- Docker Documentation: `https://docs.docker.com/`