Attendance Logging System Using Webcam

Submitted by: Ashik Muhammed S

Submitted to: ICT Academy of Kerala

Date: 03/11/2024

# Introduction

The Attendance Logging System Using Webcam streamlines the attendance process by leveraging facial recognition technology. Using a Convolutional Neural Network (CNN) model, the system performs real-time face recognition, enabling accurate and efficient attendance tracking. Built with Flask for the back end, HTML and CSS for the front end, and SQLite for data storage, the system provides a user-friendly platform for administrators to securely monitor attendance data, capture photos, and manage user information.

This project aims to create a reliable and intuitive attendance logging system that uses deep learning techniques to recognize faces and log attendance automatically. With a centralized web interface, administrators can easily monitor attendance, add new users, and access detailed attendance data, making the system both practical and effective for daily use.

Objectives:

1. 1. Develop a reliable and user-friendly attendance logging system.
2. 2. Employ deep learning techniques (CNN) for real-time face detection and recognition.
3. 3. Provide a centralized web interface for admins to monitor, add users, and access attendance data.

# System Architecture and Algorithm

**Algorithm Used: Convolutional Neural Network (CNN)**

CNNs are widely used for image recognition tasks due to their ability to identify spatial features within images. In this project, a CNN model is trained to recognize and match facial patterns, which allows the system to identify registered users based on webcam input.

**System Components:**

1. 1. Flask (Back-end Framework):  
   Handles server requests, routes, and interaction with the database. Provides endpoints for face detection, photo capture, and user management.
2. 2. Python (Core Functionalities):  
   Used for face detection and capturing user photos via the webcam. Integrates with the CNN model to perform face recognition in real-time. Manages user data, adding new users to the system, and updating attendance records.
3. 3. HTML & CSS (Front-end):  
   Designed to be responsive and user-friendly for various devices. Provides the structure and styling for the admin login, live stream view, and attendance reports.
4. 4. SQLite (Database):  
   Stores user data (name, user ID, photos) and attendance records. Supports data retrieval based on specified date ranges for attendance tracking.

# Front-End Implementation

1. User Interface Design:  
   Created with HTML and CSS, ensuring a responsive layout that adjusts to different screen sizes. The interface allows admins to log in, view live video streams, add new users, and access attendance records based on selected dates.
2. Flask Integration with Front-End:  
   Flask routes manage page rendering and form submissions, enabling seamless interactions. A live video stream from the webcam is displayed to the admin for real-time monitoring. Upon successful face recognition, Flask triggers updates to the attendance records in SQLite.
3. Admin Panel Functionalities:

* Admin Login: A secure login page for administrators to access system functions.
* Live Stream View: Real-time display of the webcam feed, enabling admins to monitor attendance.
* Use Management: An interface to add new users by capturing photos and storing them for recognition. Date-Based
* Attendance Reports: A filter to view attendance records based on specific dates

# Face Detection and Recognition Process

1. Face Detection:  
   The webcam feed is processed in real-time, capturing frames to identify faces. A face detection algorithm pre-processes frames, cropping and resizing images for optimal recognition by the CNN model.
2. Photo Capture and User Registration:  
   When adding a new user, the system captures multiple photos of the user’s face. These images are used to train or update the CNN model, improving accuracy in recognizing the individual.
3. Face Recognition:  
   When an individual appears in the frame, the CNN model compares the detected face with stored user images. Upon a successful match, the system automatically logs attendance data, storing the user’s ID and timestamp.

# Database Structure and Design

1. Database Schema:  
User Table: Stores information on registered users (user ID, name, photos). Attendance Table: Logs each user’s attendance, including the date and timestamp. Admin Credentials Table: Securely stores admin login credentials for system access.

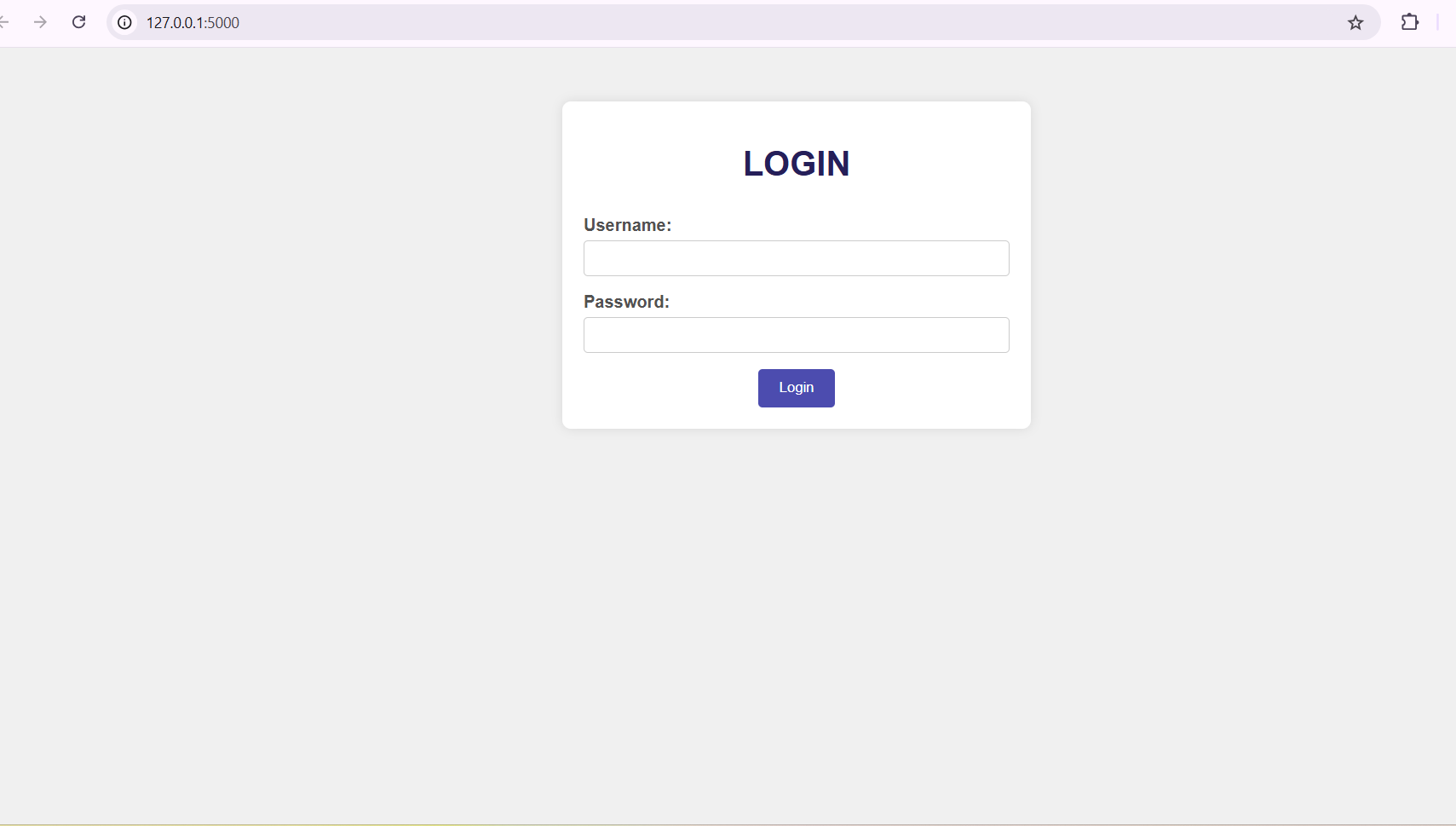
2. Data Storage and Retrieval:  
Attendance records are timestamped, allowing for date-based queries. The system retrieves attendance data upon request, filtering results by specific dates.

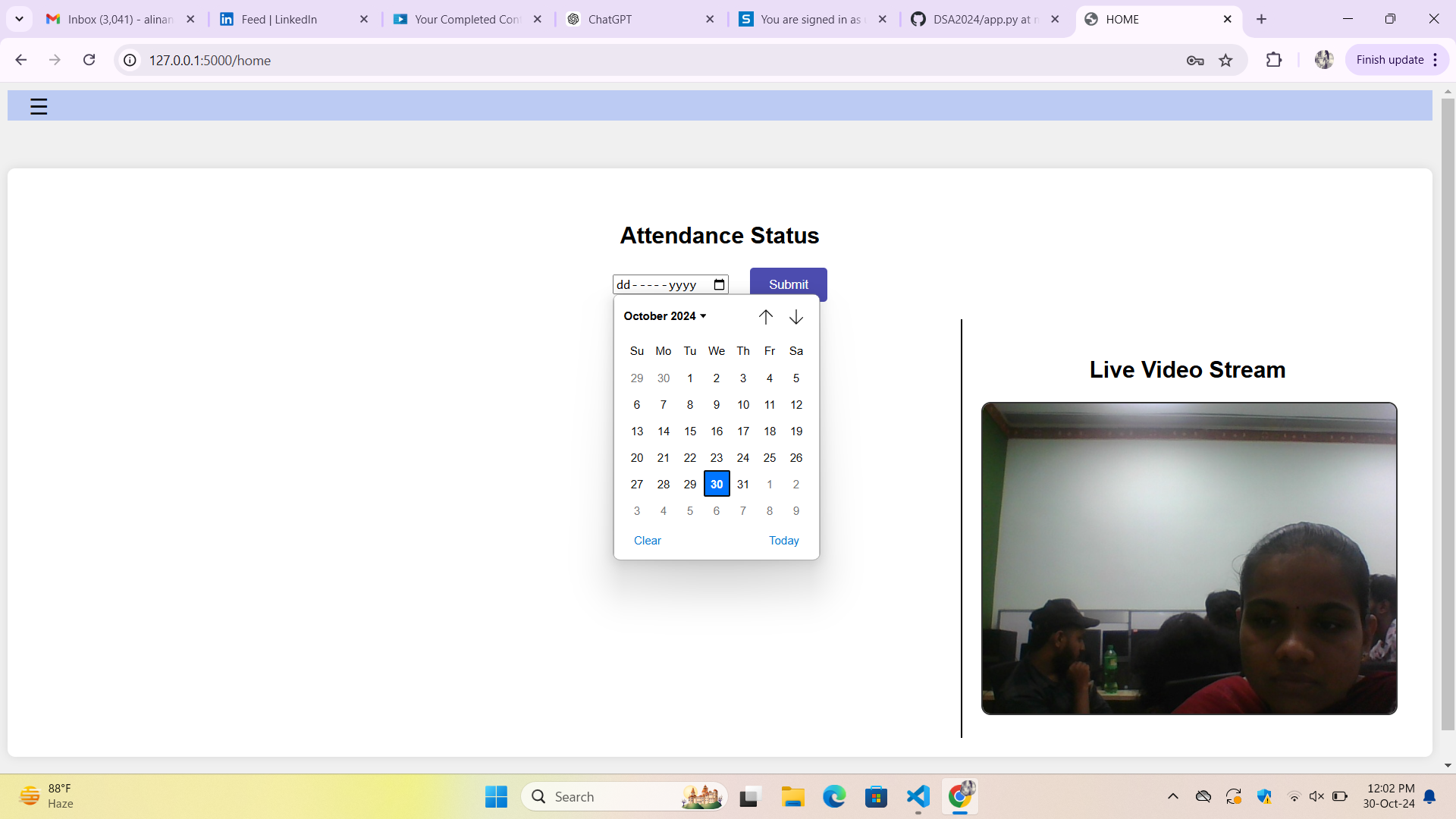
# Workflow and Use Cases

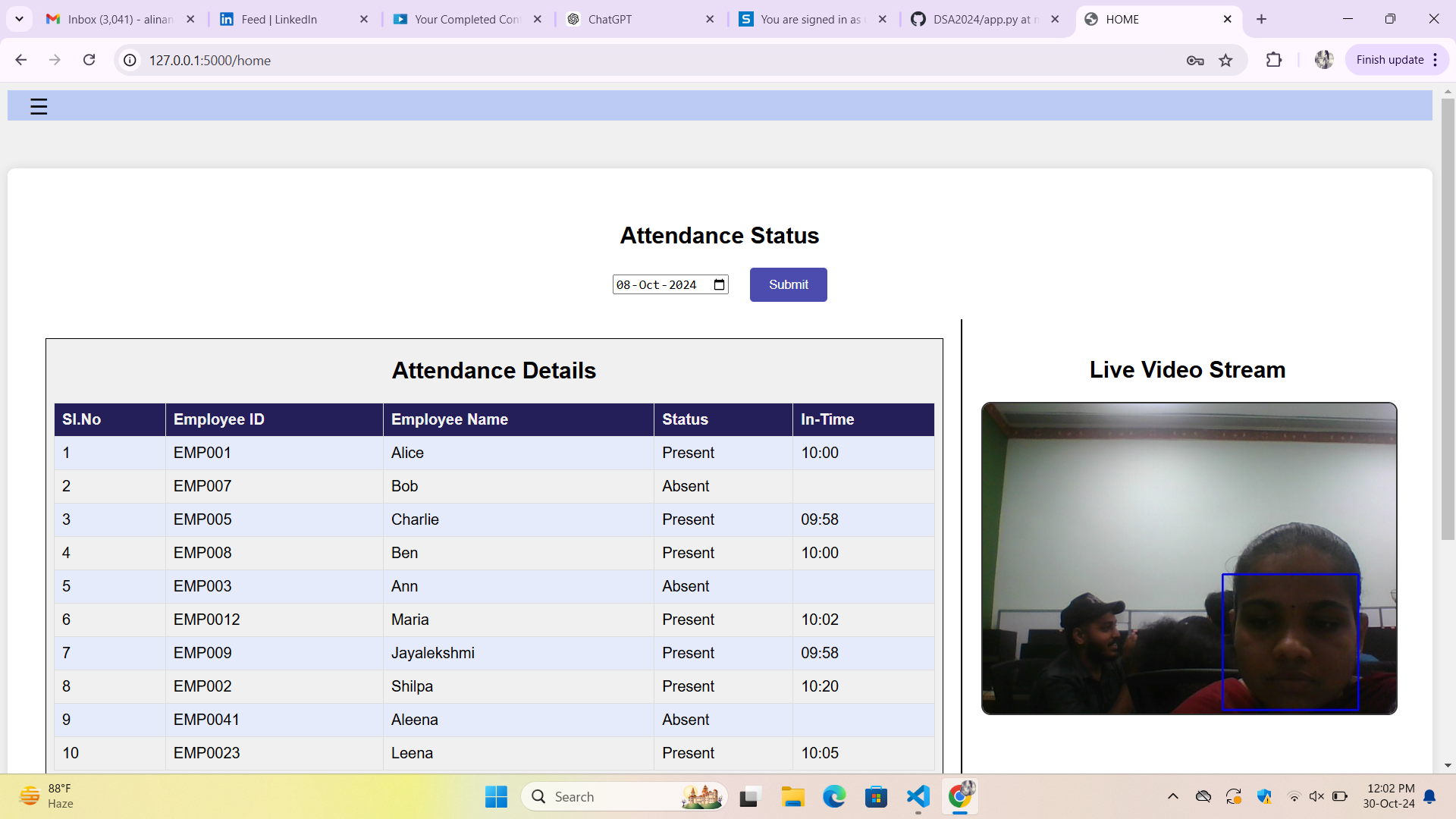
1. System Workflow:  
The admin logs into the system, gains access to the live video stream, and can monitor attendance. Registered users are automatically recognized, and their attendance is logged without manual input. Attendance records can be accessed by date, enabling quick and easy data retrieval.

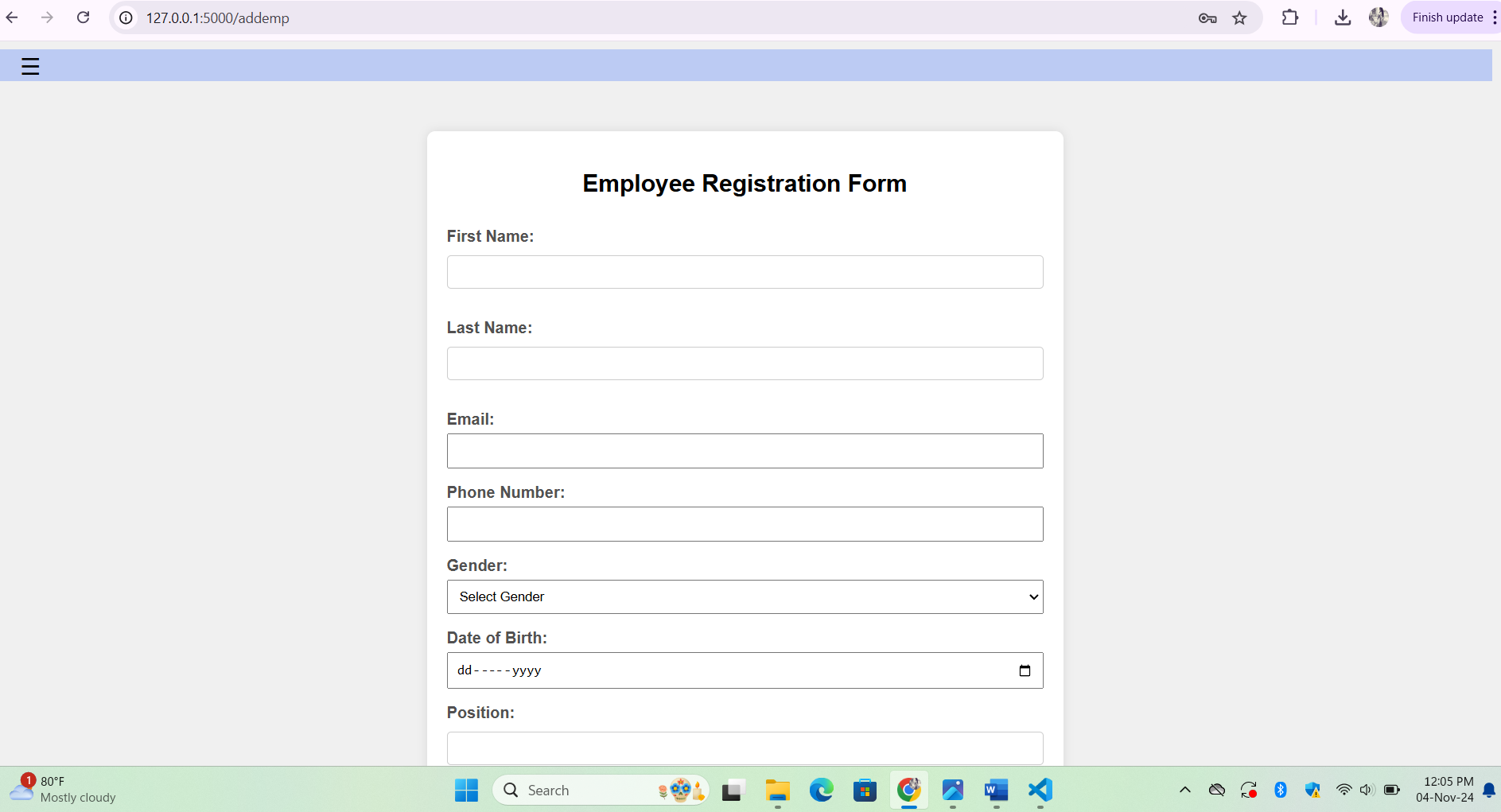
2. Use Cases:  
Adding New Users: Admin captures images and saves them in the database, updating the CNN model. Monitoring Attendance: Admin views the live feed to oversee attendance logging. Accessing Records: Admin views attendance logs for specific dates, useful for reporting and auditing.

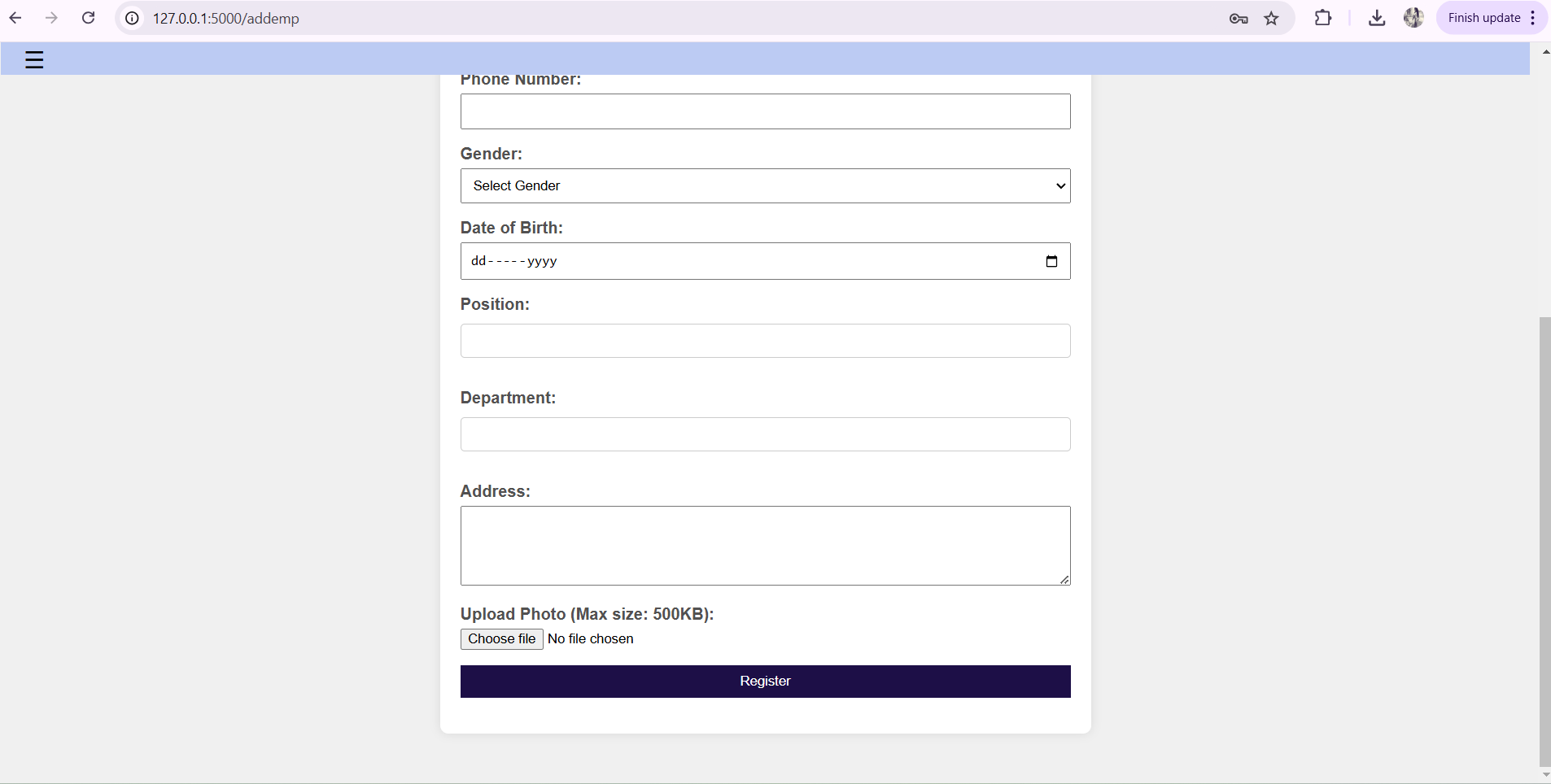
# Results

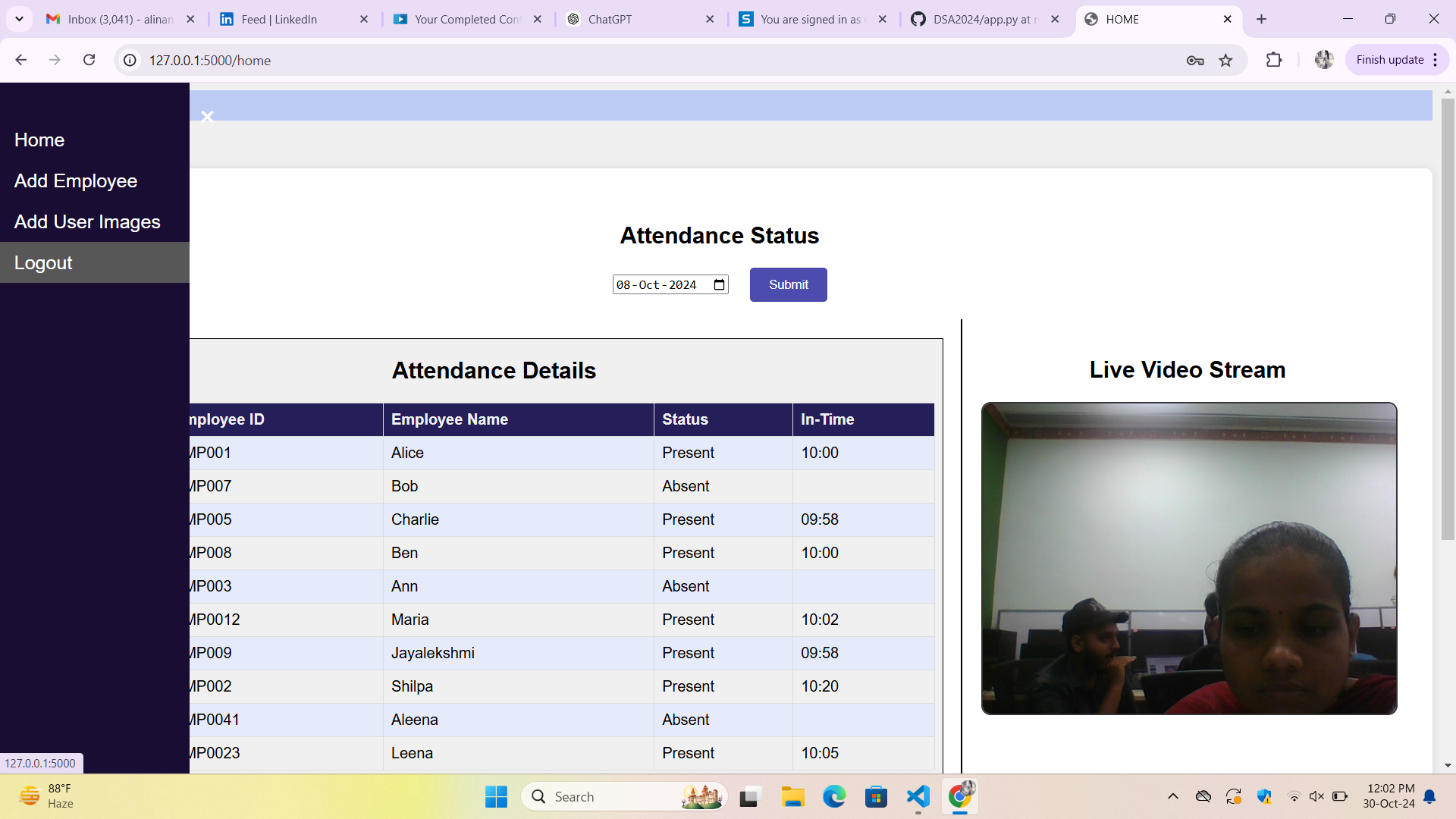












# Conclusion and Future Enhancements

This attendance logging system provides an effective and scalable solution for automating attendance management through facial recognition. By leveraging a Convolutional Neural Network (CNN) model, the system allows real-time face detection and recognition, minimizing manual processes and improving accuracy in tracking attendance. The combination of Flask for the backend, HTML and CSS for the frontend, and SQLite for data storage has created a reliable and user-friendly platform tailored to meet daily attendance needs efficiently.

Future enhancements to the system could significantly boost its capabilities. Cloud integration would allow for remote data storage, accommodating larger data volumes and facilitating easier access. Additionally, implementing advanced security features, such as encryption, would enhance the protection of user data and attendance logs. Continued refinement of the CNN model would also further improve recognition accuracy, making this system even more robust and adaptable to diverse environments.