



# Smart Attendance System

Smile to be counted, no proxies allowed.

## ABSTRACT

The Smart Attendance System is a Python program that automates attendance using face recognition, voice commands, and a themed Tkinter interface. It captures faces via webcam, logs attendance to CSV, and provides an admin vault for viewing records, blocking users, and managing registrations.

<b>Report By</b>	Shehar Bano
<b>Student I'D</b>	70151528
<b>Program</b>	BS-IET Section-I
<b>Course</b>	Artificial Intelligence
<b>Submitted To</b>	Dr. Syed M Hamedoon
<b>Date of Submission</b>	January 6 <sup>th</sup> , 2026

## Contents

1.	Project Description .....	2
1.1.	Core Stack .....	2
1.2.	Data Files .....	2
2.	Project flow diagram .....	3
3.	Project code.....	4
4.	Output .....	6
5.	Scope of my work.....	8
6.	Github repository link.....	8
7.	Quick setup and run.....	8

## Project Description

The **Smart Attendance System** is a Python program designed to modernize and automate attendance marking. It combines computer vision, voice interaction, and a playful Tkinter interface to create a system that is both functional and engaging. By leveraging face recognition technology, it ensures accurate identification of individuals, while voice commands and themed dialogs make the experience more user-friendly.

At its core, the system uses a webcam to capture live video streams. Faces are processed using the LBPH (Local Binary Patterns Histograms) algorithm in OpenCV, compared against a trained dataset of registered users. When a match is found, attendance is logged in a CSV file with name, ID, date, time, and day. Duplicate entries are prevented by checking if a user has already been marked present.

A secure **Admin Vault** provides advanced functionality: viewing attendance in a stylized table, blocking/unblocking users, and marking admin attendance. The workflow includes login credentials, admin face capture, and fullscreen controls to ensure authority.

The system also integrates **voice control** for hands-free operation. Using the wake word “hello,” users can issue commands such as starting attendance, opening the admin panel, viewing logs, blocking/unblocking users, or exiting. Voice feedback via Windows SAPI makes the experience conversational and immersive.

To maintain usability, camera feeds and dialogs remain topmost. Unknown users are guided through auto-registration, where face samples are collected and linked to a name and numeric ID. This data is added to the dataset, retraining the recognition model for future use.

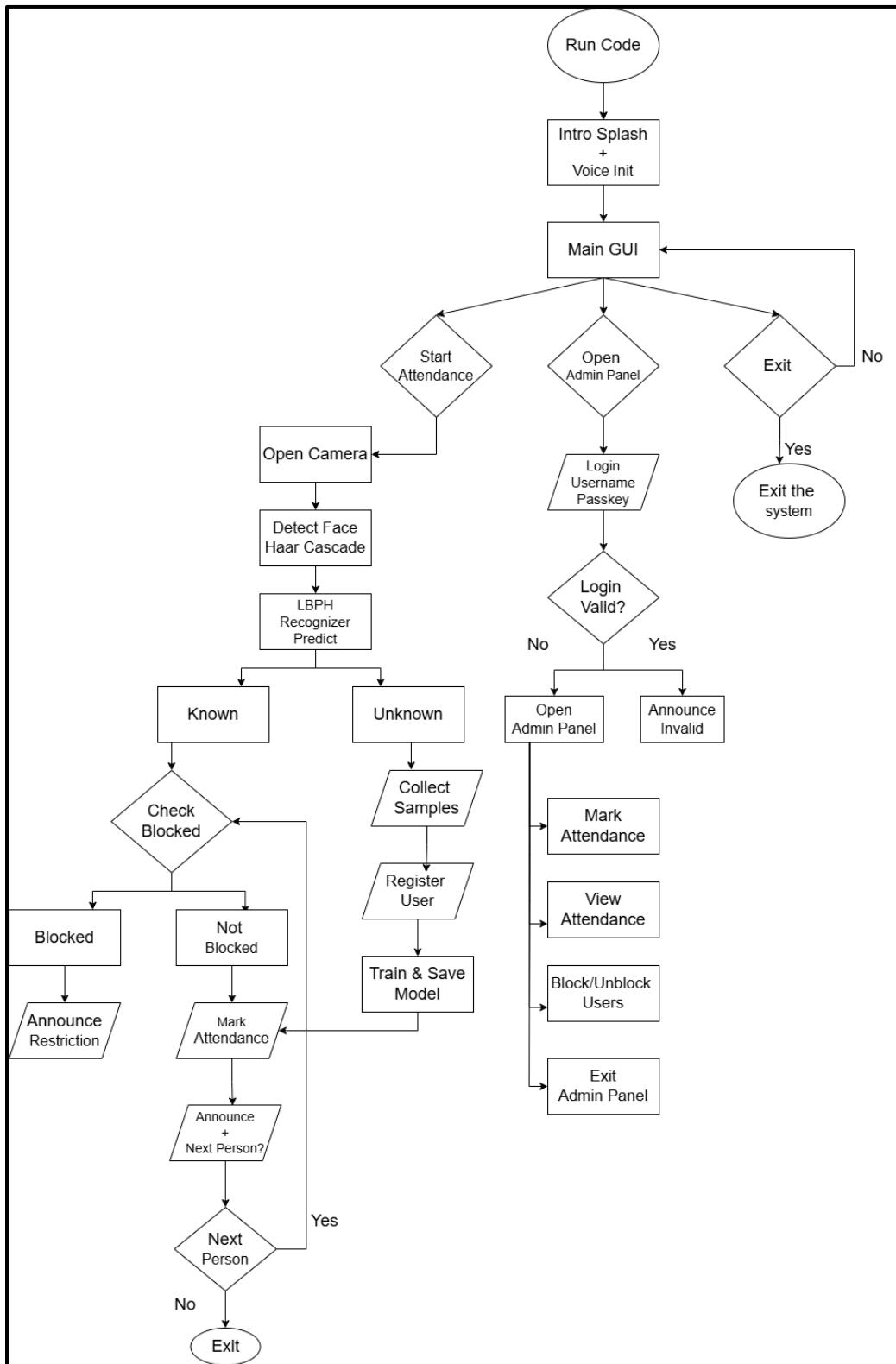
## Core Stack

- **OpenCV (LBPH)**: Face detection and recognition.
- **OpenCV (Haar Cascade)**: Face detection using pre-trained classifiers.
- **Tkinter (GUI)**: User interface with themed dialogs and panels.
- **PIL (ImageTk)**: Displaying camera frames in Tkinter windows.
- **SAPI (Windows Speech)**: Voice feedback and announcements.
- **speech\_recognition (Google)**: Voice command input.
- **CSV/JSON**: Storage for attendance logs, user maps, and blocked lists.

## Data Files

- **Faces directory:** faces/<Name>/\*.jpg — stores user face samples.
- **Admin face:** faces/Admin/admin.jpg — admin's registered face image.
- **Model:** trainer.yml — trained LBPH model for recognition.
- **Maps:** label\_map.json, id\_map.json — maps labels to names and IDs.
- **Attendance log:** attendance.csv — records attendance entries.
- **Blocked users:** blocked.json — list of users restricted from attendance.

## Project flow diagram



## Project code

- **Main GUI:** “Attendance Ship Ahoy!” screen with buttons (Start Attendance, Admin Panel, Exit).

```

def main_gui():
    ensure_attendance_file()
    root = tk.Tk()
    root.title("Attendance System - Captain Sheharbano's Pirate Ship")
    root.attributes('-fullscreen', True)
    root.attributes('-topmost', True)
    root.configure(bg='#ffe4e1')
    root.bind("<Escape>", lambda event: root.destroy())

    font = ("Comic Sans MS", 20, "bold")
    main_frame = tk.Frame(root, bg='#ffe4e1')
    main_frame.place(relx=0.5, rely=0.5, anchor="center")

    tk.Label(main_frame, text="Ahoy, Code-Pirates!\nAttendance Ship Ahoy!",
            bg='#ffe4e1', fg='#4b0082', font=("Comic Sans MS", 30, "bold")).pack(pady=40)

    def create_main_btn(text, cmd, bgc, fgc):
        return tk.Button(main_frame, text=text, command=cmd,
                         bg=bgc, fg=fgc, font=font, relief='raised', bd=5,
                         width=25, height=2, activebackground='#ffd700',
                         activeforeground='#4b0082')

    create_main_btn(
        "Start Attendance",
        lambda: [speak_async("Booting roll-call rocket boosters!"),
                 attendance_workflow()],
        '#ffb6c1', '#4b0082'
    ).pack(pady=15)

    create_main_btn(
        "Admin Panel",
        lambda: [speak_async("Cracking open secret admin vault..."),
                 admin_panel()],
        '#87cefa', '#4b0082'
    ).pack(pady=15)

    create_main_btn(
        "Exit",
        lambda: [speak_sync("Shutting down the matrix... Code-bye!"),
                 root.destroy()],
        '#90ee90', '#4b0082'
    ).pack(pady=15)

```

- **Admin vault:** Full screen panel with buttons (View Attendance, Block/Unblock, Mark Admin Attendance, Close Panel).

```

tk.Label(admin_frame, text="Ye Admin Vault", bg='#ffff5ee',
         fg="#4b0082", font=("Comic Sans MS", 30, "bold")).pack(pady=30)

def create_btn(text, cmd, bgc, fgc):
    return tk.Button(admin_frame, text=text, command=cmd,
                     bg=bgc, fg=fgc, font=font, relief='raised', bd=5,
                     width=25, height=2, activebackground="#ffd700",
                     activeforeground="#4b0082")

create_btn(
    "View Attendance",
    lambda: [speak_async("Summoning the sacred scroll of attendance..."),
              view_attendance_gui()],
    "#ffb347", "#4b0082"
).pack(pady=10)

create_btn(
    "Block User",
    lambda: [speak_async("Casting a ban-spell..."), block_user()],
    "#87cefa", "#4b0082"
).pack(pady=10)

create_btn(
    "Unblock User",
    lambda: [speak_async("Breaking the curse..."), unblock_user()],
    "#90ee90", "#4b0082"
).pack(pady=10)

create_btn(
    "Mark Admin Attendance",
    lambda: [speak_async("Admin roll-call rocket launching!"),
              camera_for_one_user(is_admin=True)],
    "#ffb6c1", "#4b0082"
).pack(pady=10)

create_btn(
    "Close Panel",
    close_admin_panel,
    "#ff7f50", "#4b0082"
).pack(pady=15)

```

- **Attendance Workflow:** The workflow launches a “roll-call rocket,” opens the camera, marks attendance, and loops for next users.

```

def attendance_workflow(skip_initial_prompt=False):
    if not skip_initial_prompt:
        speak_async("Do you wish to ignite the roll-call rocket?")
        ans = stylish_messagebox(
            "Start Attendance",
            "Open camera to mark attendance?\n(Admins use Admin Panel)",
            type='yesno'
        )
        if not ans:
            speak_async("Abort mission! Roll-call rocket stays grounded.")
            return
        speak_async("Booting up the roll-call rocket boosters!")
    else:
        speak_async("Starting attendance directly on your command!")

    while True:
        camera_for_one_user(is_admin=False)
        speak_async("Is another brave soul ready for roll-call?")
        cont = stylish_messagebox(
            "Next Person",
            "Does anyone else want to mark attendance?",
            type='yesno'
        )
        if not cont:
            speak_async("Roll-call rocket has landed. Mission complete!")
            break

```

## Output

- **Voice control:** Terminal logs for “hello”, “open admin panel”, etc.

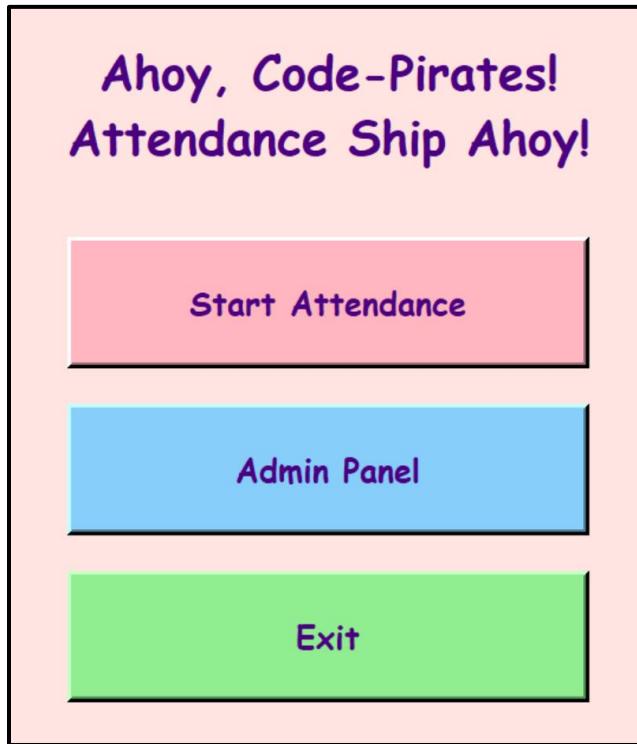
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

● Speaking (sync): My name is Jerry... summoned into existence by Sherry. Let's wander through my little world together.
Jerry says: Ahoy, code-pirates! Attendance ship is sailing!
Calibrating microphone for ambient noise...
Listening...
Jerry says: Voice control ready. Say 'Hello' to wake me, then give your command. Say 'Over' to stop listening.
Listening...
Listening...
Heard: hello
Jerry says: erry here! I'm listening. Say 'Over' when you're done.
Listening...
Listening...
Heard: admin panel
Processing voice command: open_admin_panel
Jerry says: Admin login spell activated! Whisper your username...
Listening...
Listening...
Listening...
Jerry says: Now, chant your secret passkey...
Listening...
Jerry says: Hail to the code-chief! Admin powers unlocked!
Listening...
Listening...
Jerry says: Look at the magic eye. Press S to capture your legendary admin mugshot!
Listening...
Jerry says: Gotcha! Admin mugshot stored in the digital vault!
Listening...

```

- **Main GUI:** A welcome screen with pirate-themed buttons for starting attendance, accessing the admin panel, or exiting.



- **Attendance Viewer:** A themed attendance table showing stamped entries with name, ID, date, time, and day.

A screenshot of a Windows-style application window titled "Sacred Scroll of Attendance" with a sub-titile "The Sacred Attendance Scroll". The window contains a table with five columns: Name, ID, Date, Time, and Day. There are two rows of data:

Name	ID	Date	Time	Day
sheharbano404	ADMINID	2026-01-02	07:37:33 PM	Friday
Toufeeque Ahmad	2130	2026-01-02	07:39:35 PM	Friday

A "Close Scroll" button is located at the bottom center of the window.

## Scope of my work

- **Implemented:**
  - **Face detection & recognition:** Haar cascade + LBPH with dataset building and model persistence.
  - **Auto-registration:** Buffered unknown faces, guided name/ID capture, retraining.
  - **Attendance logging:** CSV with date, time, day; duplicate prevention per day.
  - **Admin features:** View attendance, block/unblock users, mark admin attendance, admin face capture.
  - **Voice control:** Wake word, command routing via queue, async speech synthesis.
  - **UI polish:** Topmost windows, fullscreen panels, custom dialogs, playful copy.
- **Assumptions & constraints:**
  - **Windows-only speech (SAPI):** Uses `win32com.client` for TTS.
  - **Internet required for speech recognition:** Google API via `speech_recognition`.
  - **Single camera index:** Uses `cv2.VideoCapture(0)`.
  - **LBPH threshold:** `UNKNOWN_THRESHOLD = 70.0` tuned for this dataset; may need calibration.
- **Future scope:**
  - **Cross-platform TTS & offline ASR:** Replace SAPI/Google with platform-agnostic or offline engines.
  - **Model improvements:** Add data augmentation, per-user sample management, and confidence calibration UI.
  - **Security:** Hash admin passkey, role-based access, audit logs.
  - **Database backend:** Migrate CSV/JSON to SQLite/MySQL with triggers and audit tables.
  - **UI/UX:** Theming, animations, and brand-consistent components (your specialty).

## Github repository link

- **Repository:**

## Quick setup and run

- **Requirements:**
  - Python 3.9+
  - `opencv-contrib-python`, `numpy`, `Pillow`, `pywin32`, `speechrecognition`, `pyaudio` (or alternative), `tkinter` (bundled with Python on Windows)
- **Install:**

```
pip install opencv-contrib-python numpy Pillow pywin32 SpeechRecognition
pyaudio
```

- **Run:**

```
python smart_attendance.py
```

- **First-time steps:**

1. **Run the application** Launch the main Python file to initialize GUI, voice engine, and queues.
2. **Admin login**
  - Click **Admin Panel** on the main GUI.
  - Enter username: sheharbano404 and passkey: Snape (can be changed in code).
  - Unlocks admin powers and opens the fullscreen vault.
3. **Capture admin face**
  - On first login, press **S** in the camera window to save your face as faces/Admin/admin.jpg.
  - Triggers training with admin image.
4. **Train model**
  - Automatically triggered after admin face capture or new user registration.
  - Uses build\_dataset\_and\_label\_map() and train\_and\_save\_recognizer() to save trainer.yml.
5. **Start attendance**
  - From main GUI, click **Start Attendance** or say “hello... start attendance”.
  - Opens camera, detects faces, and predicts identity using LBPH.
6. **Register unknown user**
  - If face is unknown, collects 5 samples.
  - Prompts for name and numeric ID via stylish\_askstring().
  - Saves images, updates id\_map.json, retrains model, and marks attendance.
7. **Mark attendance**
  - Logs name, ID, date, time, and day to attendance.csv.
  - Prevents duplicate marking per day using has\_marked\_today().
8. **View attendance**
  - Admins can click **View Attendance** in the vault.
  - Opens a themed GUI table from view\_attendance\_gui().
9. **Block/unblock users**
  - Admins can block users via block\_user() or unblock via unblock\_user().
  - Updates blocked.json and restricts attendance for blocked names.
10. **Voice commands (optional)**
  - Say “hello” to activate.
  - Commands include: “start attendance”, “open admin panel”, “view attendance”, “block user”, “unblock user”, “exit system”.