CHAPTER 1

PREAMBLE

1.1 INTRODUCTION

In the modern era of technology, educational institutions and workplaces are continuously exploring innovative methods to manage attendance efficiently. Traditional methods, such as manual sign-ins or swipe cards, often consume significant time and are prone to errors or misuse. With the advancement of Artificial Intelligence (AI) and machine learning, automated attendance systems using biometric verification have emerged as a reliable alternative. Among these, **face recognition technology** has gained significant attention due to its non-intrusive, fast, and accurate identification capabilities. The Face Attendance System for Sharnbasava University leverages state-of-the-art facial recognition models, specifically the **face-api.js** library, to identify individuals in real-time through a webcam interface. This system allows educators to automate attendance tracking, reduce administrative workload, and maintain a digital record of attendance that can be easily stored, exported, and audited offline.

The primary concept behind a face recognition attendance system is to capture the live image of a person, analyze facial features, and compare them with stored descriptors of known faces. Upon a successful match, the system marks attendance automatically, eliminating the need for manual intervention. The design of this system also focuses on security, ensuring that unauthorized individuals cannot tamper with the attendance data. Additionally, the system stores an **audit trail with face snapshots**, which provides visual verification and strengthens accountability. This project demonstrates how emerging web technologies, combined with AI, can simplify traditional tasks while improving accuracy and efficiency in institutional environments.

1.2 OVERVIEW OF JAVASCRIPT AND FACE-API.JS

The system is primarily implemented using **JavaScript**, one of the most widely used programming languages for web development. JavaScript provides dynamic, client-side functionalities that make it suitable for real-time interactions, such as streaming video from a webcam and processing face recognition locally. It integrates seamlessly with HTML and CSS to provide a complete front-end solution while allowing access to the **WebRTC API** for video capture.

The backbone of the face recognition functionality in this project is the **face-api.js** library, a JavaScript wrapper for TensorFlow.js. Face-api.js provides pre-trained deep learning models for **face detection, facial landmark detection, and face recognition**,

1.3 PROBLEM DEFINITION

Manual attendance systems present numerous challenges in modern educational and professional environments. The traditional paper-based approach is labor-intensive and time-consuming, often resulting in delays that reduce valuable instructional or operational time. Additionally, manual methods are highly susceptible to errors, such as incorrect entries, duplicate attendance, or missed records. In some cases, individuals can manipulate the system by signing on behalf of others, leading to inaccurate attendance tracking.

The **Face Attendance System** addresses these problems by automating the process using facial recognition. It eliminates the need for manual record-keeping, reduces the likelihood of errors or misuse, and provides a verifiable audit trail. Furthermore, by functioning offline, the system ensures data privacy while maintaining high accuracy in face detection and recognition. This system is particularly suitable for educational institutions where large numbers of students must be accounted for quickly and reliably.

1.4 MOTIVATION

The motivation behind developing the Face Attendance System stems from the need to **streamline attendance tracking** and enhance efficiency in institutional management. Several factors influenced the choice of this project:

- **Efficiency:** Manual attendance methods consume substantial class time, which can be better utilized for teaching and learning. Automating the process allows educators to focus on core tasks rather than administrative duties.
- **Accuracy:** Human errors in recording attendance, such as missed entries or duplicate marks, can be minimized by leveraging AI-based recognition, which provides consistent and reliable results.
- **Non-intrusive identification:** Face recognition offers a contactless and convenient method to mark attendance. Unlike fingerprint scanners or card systems, it does not require physical contact, which is especially valuable in maintaining hygiene standards.
- **Technological advancement:** The project showcases the practical application of modern web technologies and AI in everyday tasks, demonstrating the potential of integrating machine learning into real-world systems.
- **Security and accountability:** By maintaining an **offline audit trail with snapshots**, the system ensures that attendance records can be verified visually, providing additional accountability and reducing the chances of manipulation or falsification.

1.5 OBJECTIVES

The objectives of the Face Attendance System are as follows:

- Automate attendance tracking for students or employees using face recognition.
- Provide a **real-time, offline, and non-intrusive** attendance marking mechanism.
- Reduce time and administrative workload associated with manual attendance.
- Ensure **accuracy and reliability** in face detection and recognition.
- Maintain a digital record of attendance, which can be **exported as CSV** for analysis.
- Include a visual audit trail with face snapshots for verification purposes.
- Allow administrators to **add new members dynamically** with multiple reference images.
- Provide the ability to toggle the camera on/off and reset attendance as needed.

1.6 FUTURE SCOPE

The potential future enhancements for the system include:

- Integrating **mobile device compatibility** so that attendance can be marked through smartphones or tablets.
- Incorporating **cloud storage** for centralized attendance management across multiple classrooms or campuses.
- Implementing **advanced AI models** for improved recognition accuracy, especially under poor lighting conditions.
- Adding **real-time notifications** for absentees to students or guardians via email or messaging apps.
- Extending the system to recognize **multiple attributes**, such as emotion or mask detection, to adapt to safety protocols.
- Enhancing the **user interface** for administrators with detailed reporting and analytics dashboards.
- Supporting **multi-language interfaces** to make the system accessible to a broader range of users.

CHAPTER 2

LITERATURE SURVEY

2.1 RELATED WORKS

Attendance management has been a vital component in educational institutions and workplaces for decades. Traditional systems relied on manual methods, such as taking roll calls or signing attendance sheets, which were labor-intensive, time-consuming, and prone to human errors. Over time, various technological solutions were explored to improve the accuracy and efficiency of attendance management.

One of the earliest automated approaches involved the use of **RFID (Radio Frequency Identification) cards**. In these systems, students or employees were issued RFID cards that needed to be scanned upon entry. While this method reduced the effort of manual entry, it still presented several limitations. Cards could be forgotten, lost, or shared with others, which compromised the integrity of attendance records. Moreover, RFID-based systems required specialized hardware and maintenance, making them less scalable for large institutions.

Another popular approach has been **biometric attendance systems**, such as fingerprint or iris recognition. Fingerprint scanners, in particular, provided a higher level of security compared to cards, as fingerprints are unique to each individual. Several studies and implementations have shown that fingerprint-based attendance systems improve accuracy and reduce manual workload. However, these systems can be cumbersome, requiring users to physically touch the scanner, which raises hygiene concerns. Additionally, some users may face difficulties if fingerprints are worn or damaged, leading to failed recognition attempts.

In recent years, **face recognition technology** has gained prominence due to its non-intrusive nature and convenience. Unlike fingerprint or card systems, facial recognition does not require physical contact and allows real-time identification of multiple individuals simultaneously. Research in this domain highlights that modern deep learning-based models can achieve high accuracy in face detection and recognition, even in complex environments with varying lighting conditions and facial orientations. Libraries like **face-api.js** provide pre-trained neural networks that can be deployed directly in web browsers, enabling offline, client-side face recognition without relying on server processing.

Several academic studies have explored the integration of face recognition with attendance systems. For example, a study by **K. Gupta et al. (2020)** implemented a face recognition attendance system using Python and OpenCV, achieving high recognition accuracy under controlled classroom conditions.

2.2 EXISTING SYSTEM

Existing attendance systems can be broadly categorized into **manual, card-based, biometric, and face recognition systems**.

- **Manual Attendance Systems:** This is the most basic approach, involving roll calls or signature-based registers. While simple and inexpensive, it is slow and prone to errors such as duplicate entries or proxy attendance. The system provides no digital record unless manually digitized later.
- **Card-Based Attendance Systems:** RFID or magnetic swipe cards automate attendance to some extent. They provide quick entry logging and generate digital records automatically. However, these systems are vulnerable to misuse if cards are shared, lost, or forgotten. Maintenance and cost of hardware are additional challenges.
- **Biometric Attendance Systems:** Fingerprint or iris scanners offer unique identification based on physiological traits. These systems are more secure and difficult to cheat but require specialized devices. Hygiene concerns and recognition failures due to physical conditions of the users are also notable limitations.
- **Face Recognition Attendance Systems:** Modern systems leverage deep learning to detect and identify faces in real time. These systems are non-intrusive, fast, and capable of handling multiple individuals simultaneously. However, challenges include variations in lighting, facial orientation, expressions, and occlusions such as glasses or masks. Some existing implementations also rely on server-based processing, which may raise privacy and connectivity concerns.

A detailed review of these systems indicates a clear trend towards **face recognition technology** due to its advantages in convenience, scalability, and offline capabilities. Libraries such as **face-api.js** allow real-time processing directly in web browsers, addressing both privacy and deployment challenges. This makes face recognition systems particularly suitable for educational institutions where a large number of students need to be tracked efficiently.

2.3 PROPOSED SYSTEM

The **proposed Face Attendance System** aims to address the limitations of existing systems while leveraging the latest developments in facial recognition technology. Unlike traditional or card-based systems, this system provides a **contactless, automated, and offline** solution for tracking attendance.

The core features of the proposed system include:

1. **Real-Time Face Detection and Recognition:** Using the pre-trained **face-api.js models**, the system detects faces in the video feed and matches them against stored descriptors to identify individuals. This process occurs in real time, allowing attendance to be marked instantly.
2. **Offline Operation:** All processing is performed on the client side in the browser, which ensures that sensitive facial data remains local. This enhances privacy and allows the system to function without internet connectivity.
3. **Audit Trail with Snapshots:** To ensure accountability, the system captures snapshots of recognized faces at the time of attendance marking. This visual audit trail reduces chances of manipulation and provides verifiable records.
4. **Dynamic Member Management:** Administrators can add new members along with multiple reference images. The system updates its database dynamically, enhancing adaptability to new students or employees.
5. **Export and Reporting:** Attendance records can be exported as CSV files for further analysis, integration with institutional databases, or archival purposes.
6. **Flexible Interface:** Users can toggle the camera, reset attendance records, and manage members through an intuitive web-based interface.
7. **Robust Recognition:** The system includes measures such as multiple recognition confirmations and time intervals between markings to prevent false positives or duplicate entries.

The **proposed system** is a significant improvement over existing solutions because it combines **accuracy, convenience, security, and scalability** in a single, offline platform. By using standard webcams and browser-based technologies, it avoids the cost and maintenance requirements of specialized hardware. Additionally, the system can be further enhanced in the future to support mobile devices, cloud-based synchronization, and real-time notifications.

CHAPTER 3

SYSTEM REQUIREMENT SPECIFICATION

3.1 SOFTWARE REQUIREMENTS

- **Operating System:** Windows 10/11, Linux, or macOS.
- **Web Browser:** Google Chrome, Mozilla Firefox, or Edge (supports WebRTC and JavaScript).
- **Programming Languages:** HTML, CSS, JavaScript.
- **Libraries/Frameworks:**
    - **face-api.js** (for face detection and recognition)
    - **TensorFlow.js** (underlying AI framework)
- **Local Storage:** Browser LocalStorage for saving attendance records and member data.
- **Development Tools:** Visual Studio Code or any code editor.
- **Optional:** Git for version control.

3.2 HARDWARE REQUIREMENTS

- **Processor:** Intel i3 or higher / AMD Ryzen 3 or higher.
- **RAM:** Minimum 4 GB.
- **Webcam:** Built-in or external HD webcam for face detection.
- **Storage:** At least 1 GB free space for storing images and data.
- **Display:** Monitor supporting 720p or higher for clear video feed.
- **Internet Connection:** Optional, only for downloading libraries; system works offline.

CHAPTER 4

IMPLEMENTATION

4.1 PROJECT IMPLEMENTATION

The Face Attendance System was implemented as a **web-based offline application** using HTML, CSS, and JavaScript, leveraging **face-api.js** for facial recognition. The project is designed to operate entirely on the client side, ensuring **privacy, speed, and real-time functionality**. The implementation involves several modules, each responsible for a specific functionality:

- **User Interface:**
    - Developed using **HTML and CSS** for a responsive, user-friendly design.
    - Includes a video feed container, attendance table, buttons for export/reset/toggle camera, and an add-member form.
    - Dark-themed layout with clear color contrasts (gold on navy) for readability.
- **Camera Integration:**
    - Uses **WebRTC API** (`navigator.mediaDevices.getUserMedia`) to capture live video from a webcam.
    - Supports toggling camera on/off with proper resource management to stop video streams when not in use.
- **Face Detection and Recognition:**
    - Utilizes **face-api.js models**:
        - **SSD MobilenetV1** for real-time face detection.
        - **Face Landmark 68Net** for landmark detection.
        - **Face Recognition Net** for descriptor generation.
    - Faces from live video are continuously detected and compared against **stored labeled descriptors**.
    - Recognition is confirmed only after multiple detections to avoid false positives.
- **Attendance Marking:**

- Records attendance automatically once a recognized face is detected.
- Prevents multiple entries within a short interval to ensure accurate logging.
- Stores attendance data in **LocalStorage**, which includes **Name, USN, Date, Time**.
- Captures snapshots of recognized faces for **audit purposes**.

- **Export and Reporting:**
  - Attendance can be exported as **CSV files** for record keeping or integration with institutional databases.
  - Snapshot data is also saved in LocalStorage for verification.
- **Member Management:**
  - Administrators can **add new members dynamically** by providing name, USN, and multiple images.
  - New members' descriptors are generated and stored for real-time recognition.

4.2 METHODOLOGY

The system follows a **structured workflow** to ensure accurate face recognition and seamless attendance management:

1. **Model Loading:**
   - Face detection and recognition models are loaded from the local `models` directory.
   - Loading is indicated with a **loading overlay** to improve user experience.
2. **Video Capture:**
   - The system initiates webcam access and streams video to the browser.
   - The video feed is overlaid with a canvas for drawing detection boxes and labels.
3. **Face Recognition Loop:**
   - Detects all faces in the video frame.
   - Resizes detection results to match video dimensions.
   - Matches detected faces with stored **labeled face descriptors**.
   - Marks attendance only if the person is recognized consistently.

4. **Data Storage and Management:**
    - Attendance records are maintained in **LocalStorage**, ensuring offline functionality.
    - Snapshots of recognized faces are stored as **Base64-encoded images** for auditing.
    - Allows resetting records and exporting data in **CSV format**.
5. **UI Interaction:**
    - Buttons allow toggling the camera, adding members, exporting data, and resetting attendance.
    - Input validation ensures proper data entry when adding new members.

4.3 SOURCE CODE STRUCTURE

- **index.html:** Main web page containing UI components.
- **style.css:** Handles layout, colors, and responsiveness.
- **script.js:** Core logic for face detection, recognition, attendance marking, and data management.
- **models/:** Folder containing pre-trained **face-api.js models**.
- **face-data/:** Folder storing reference images for members.

Index.html Code

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Face Attendance System - Sharnbasava University</title>
<script defer src="https://cdn.jsdelivr.net/npm/face-api.js@0.22.2/dist/face-api.min.js"></script>
<link rel="stylesheet" href="style.css">
</head>
<body>

<div id="loadingOverlay">
<div class="loader">Loading Models...</div>
</div>

<header>
<img src="src/logo.png" alt="College Logo" class="logo">
<h1>Sharnbasava University</h1>
<h2>Face Attendance System</h2>
</header>
```

```
<div class="main-container">

<div class="video-container">
<video id="video" autoplay muted playsinline></video>
</div>

<div class="table-container">
<table>
<thead>
<tr>
<th>Name</th>
<th>USN</th>
<th>Date</th>
<th>Time</th>
</tr>
</thead>
<tbody id="attendance"></tbody>
</table>

<div class="buttons">
<button id="export">Export Attendance CSV</button>
<button id="reset">Reset Attendance</button>
<button id="toggleCam">Turn Camera Off</button>
</div>

<div class="add-member">
<h3>Add New Member</h3>
<input type="text" id="newName" placeholder="Name">
<input type="text" id="newUSN" placeholder="USN">
<input type="file" id="newImages" accept="image/*" multiple>
<button id="addMember">Add Member</button>
</div>
</div>

</div>

<script src="script.js"></script>
</body>
</html>
```

Style.CSS

```
body {

background: #0b1d2c;
color: #fff;
font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
margin: 0;
padding: 0;
}
```

```
#loadingOverlay {
position: fixed;
top:0;
left:0;
width: 100%;
height: 100%;
background: rgba(11,29,44,0.85);
display: none;
align-items: center;
justify-content: center;
z-index: 1000;
}

.loader {
color: #ffd700;
font-size: 24px;
font-weight: bold;
}

header {
background-color: #14406b;
padding: 20px 0;
text-align: center;
box-shadow: 0 4px 6px rgba(0,0,0,0.3);
}

header .logo {
width: 80px;
vertical-align: middle;
margin-right: 15px;
}

header h1 {
display: inline-block;
font-size: 28px;
margin: 0;
vertical-align: middle;
}

header h2 {
font-size: 18px;
margin: 5px 0 0 0;
font-weight: normal;
color: #ffd700;
}

.main-container {
display: flex;
```

```
justify-content: center;
align-items: flex-start;
margin: 30px;
gap: 40px;
flex-wrap: wrap;
}

.video-container {
position: relative;
width: 640px;
height: 480px;
border: 3px solid #ffd700;
border-radius: 8px;
overflow: hidden;
}

video {
width: 100%;
height: 100%;
display: block;
}

canvas {
position: absolute;
top: 0;
left: 0;
width: 100% !important;
height: 100% !important;
z-index: 2;
}

.table-container {
width: 500px;
}

table {
width: 100%;
border-collapse: collapse;
background-color: rgba(0,0,0,0.5);
border-radius: 8px;
overflow: hidden;
}

th, td {
border: 1px solid #ffd700;
padding: 10px;
color: #fff;
text-align: center;
}
```

```
th {
background-color: #14406b;
}

.buttons {
margin-top: 15px;
text-align: center;
}

.buttons button {
margin: 5px;
padding: 10px 18px;
font-size: 16px;
cursor: pointer;
border-radius: 5px;
border: none;
background-color: #ffd700;
color: #14406b;
font-weight: bold;
transition: all 0.3s;
}

.buttons button:hover {
background-color: #e6c200;
}

.add-member {
margin-top: 20px;
background: rgba(0,0,0,0.5);
padding: 15px;
border-radius: 8px;
text-align: center;
}

.add-member h3 {
margin-bottom: 10px;
color: #ffd700;
}

.add-member input {
width: 90%;
margin: 5px 0;
padding: 8px;
border-radius: 5px;
border: none;
}

.add-member button {
```

```
margin-top: 10px;
padding: 10px 20px;
background-color: #ffd700;
color: #14406b;
font-weight: bold;
border: none;
border-radius: 5px;
cursor: pointer;
}

.add-member button:hover {
background-color: #e6c200;
}
```

Script.JS

```
let videoStream = null;


async function init() {
const loadingOverlay = document.getElementById("loadingOverlay");
loadingOverlay.style.display = "flex";
loadingOverlay.querySelector(".loader").textContent = "Loading Models...";

if (!faceapi) {
alert("Face-api.js not loaded!");
return;
}

const initialMembers = [
{
name: "Bhagyashree Patil",
usn: "SG23CSD007",
images: [
"face-data/Bhagyashree1.jpeg",
"face-data/Bhagyashree2.jpeg",
"face-data/Bhagyashree3.jpeg",
"face-data/Bhagyashree4.jpeg",
"face-data/Bhagyashree5.jpeg",
"face-data/Bhagyashree6.jpeg",
"face-data/Bhagyashree7.jpeg",
"face-data/Bhagyashree8.jpeg"
]
},
{
name: "Chetana Patil",
usn: "SG23CSD011",
images: [
"face-data/Chetena1.jpeg",
"face-data/Chetena2.jpeg",
```

```
"face-data/Chetena3.jpeg",
"face-data/Chetena4.jpeg",
"face-data/Chetena5.jpeg",
"face-data/Chetena6.jpeg",
"face-data/Chetena7.jpeg",
"face-data/Chetena8.jpeg",
"face-data/Chetena9.jpeg"
]
},
{
name: "Supriya Hiremath",
usn: "SG23CSD049",
images: [
"face-data/Supriya1.jpeg",
"face-data/Supriya2.jpeg",
"face-data/Supriya3.jpeg",
"face-data/Supriya4.jpeg",
"face-data/Supriya5.jpeg",
"face-data/Supriya6.jpeg",
"face-data/Supriya7.jpeg",
"face-data/Supriya8.jpeg",
"face-data/Supriya9.jpeg",
"face-data/Supriya10.jpeg",
"face-data/Supriya11.jpeg",
"face-data/Supriya12.jpeg"
]
},
{
name: "Mahantesh",
usn: "SG23CSD019",
images: [
"face-data/suresh1.jpeg"
]
}
];

const savedMembers = JSON.parse(localStorage.getItem("members") || "[]");
const members = [...initialMembers, ...savedMembers];

const MODEL_URL = './models';
await faceapi.nets.ssdMobilenetv1.loadFromUri(MODEL_URL);
await faceapi.nets.faceLandmark68Net.loadFromUri(MODEL_URL);
await faceapi.nets.faceRecognitionNet.loadFromUri(MODEL_URL);

loadingOverlay.querySelector(".loader").textContent = "Starting Camera...";
const video = document.getElementById("video");

async function startCamera() {
if (!videoStream) {
```

```
try {
videoStream = await navigator.mediaDevices.getUserMedia({ video: true });
video.srcObject = videoStream;
await video.play();
} catch (err) {
alert("Camera error: " + err);
}
}
}

function stopCamera() {
if (videoStream) {
videoStream.getTracks().forEach(track => track.stop());
video.srcObject = null;
videoStream = null;
}
}

await startCamera();

loadingOverlay.querySelector(".loader").textContent = "Preparing Face Data...";

const labeledDescriptors = [];

for (let person of members) {
const descriptors = [];
for (let imgPath of person.images) {
try {
const img = await faceapi.fetchImage(imgPath);
const detection = await faceapi.detectSingleFace(img).withFaceLandmarks().withFaceDescriptor();
if (detection) descriptors.push(detection.descriptor);
else console.warn(`Face not detected in ${imgPath}`);
} catch (err) {
console.warn(`Failed to load ${imgPath}: ${err}`);
}
}
if (descriptors.length > 0) {
labeledDescriptors.push(new faceapi.LabeledFaceDescriptors(person.name, descriptors));
} else {
console.warn(`No descriptors found for ${person.name}`);
}
}

const faceMatcher = new faceapi.FaceMatcher(labeledDescriptors, 0.6);
const attendanceTable = document.getElementById("attendance");
const lastMarked = {};
const recognitionCounts = {};

loadingOverlay.style.display = "none";
```

```
function loadAttendance() {
attendanceTable.innerHTML = "";
const data = JSON.parse(localStorage.getItem("attendance") || "[]");
data.sort((a,b) => new Date(a.time) - new Date(b.time));
data.forEach(d => {
const member = members.find(m => m.name === d.name);
const usn = member ? member.usn : "";
attendanceTable.innerHTML += `<tr><td>${d.name}</td><td>${usn}</td><td>${d.date}</td><td>$
{d.time}</td></tr>`;
});
}

function markAttendance(name) {
recognitionCounts[name] = (recognitionCounts[name] || 0) + 1;
if (recognitionCounts[name] < 3) return;
const now = Date.now();
if (lastMarked[name] && now - lastMarked[name] < 10000) return;
lastMarked[name] = now;
const today = new Date().toISOString().split('T')[0];
const data = JSON.parse(localStorage.getItem("attendance") || "[]");
if (data.some(d => d.name === name && d.date === today)) return;
const time = new Date().toTimeString().split(' ')[0];
data.push({ name, date: today, time });
localStorage.setItem("attendance", JSON.stringify(data));
loadAttendance();

const canvasSnap = document.createElement("canvas");
canvasSnap.width = video.videoWidth;
canvasSnap.height = video.videoHeight;
canvasSnap.getContext("2d").drawImage(video, 0, 0, canvasSnap.width, canvasSnap.height);
const faceData = JSON.parse(localStorage.getItem("faceAudit") || "[]");
faceData.push({ name, date: today, time, img: canvasSnap.toDataURL("image/jpeg") });
localStorage.setItem("faceAudit", JSON.stringify(faceData));
recognitionCounts[name] = 0;
}

loadAttendance();

const canvas = faceapi.createCanvasFromMedia(video);
canvas.style.position = "absolute";
canvas.style.top = "0";
canvas.style.left = "0";
canvas.style.zIndex = "10";
document.querySelector('.video-container').appendChild(canvas);
const displaySize = { width: video.videoWidth, height: video.videoHeight };
faceapi.matchDimensions(canvas, displaySize);
const ctx = canvas.getContext("2d");
```

```
async function detectLoop() {
if (!videoStream) return requestAnimationFrame(detectLoop);
const detections = await faceapi.detectAllFaces(video).withFaceLandmarks().withFaceDescriptors();
const resized = faceapi.resizeResults(detections, displaySize);
ctx.clearRect(0, 0, canvas.width, canvas.height);

resized.forEach(d => {
const match = faceMatcher.findBestMatch(d.descriptor);
const isKnown = match.label !== "unknown";
const boxColor = isKnown ? "green" : "red";
if (isKnown) markAttendance(match.label);

const drawBox = new faceapi.draw.DrawBox(d.detection.box, {
label: match.label,
boxColor: boxColor,
lineWidth: 4
});
drawBox.options.labelBackgroundColor = boxColor;
drawBox.options.labelColor = "white";
drawBox.draw(canvas);
});

requestAnimationFrame(detectLoop);
}

detectLoop();

document.getElementById("export").onclick = () => {
const data = JSON.parse(localStorage.getItem("attendance") || "[]");
const csv = [["Name","USN","Date","Time"], ...data.map(d => {
const member = members.find(m => m.name === d.name);
const usn = member ? member.usn : "";
return [d.name, usn, d.date, d.time];
})].map(e => e.join(",")).join("\n");
const blob = new Blob([csv], { type: 'text/csv' });
const url = URL.createObjectURL(blob);
const a = document.createElement('a');
a.href = url;
a.download = `attendance_${new Date().toISOString().split('T')[0]}.csv`;
a.click();
};

document.getElementById("reset").onclick = () => {
if (confirm("Are you sure you want to reset attendance?")) {
localStorage.removeItem("attendance");
localStorage.removeItem("faceAudit");
loadAttendance();
}
};
```

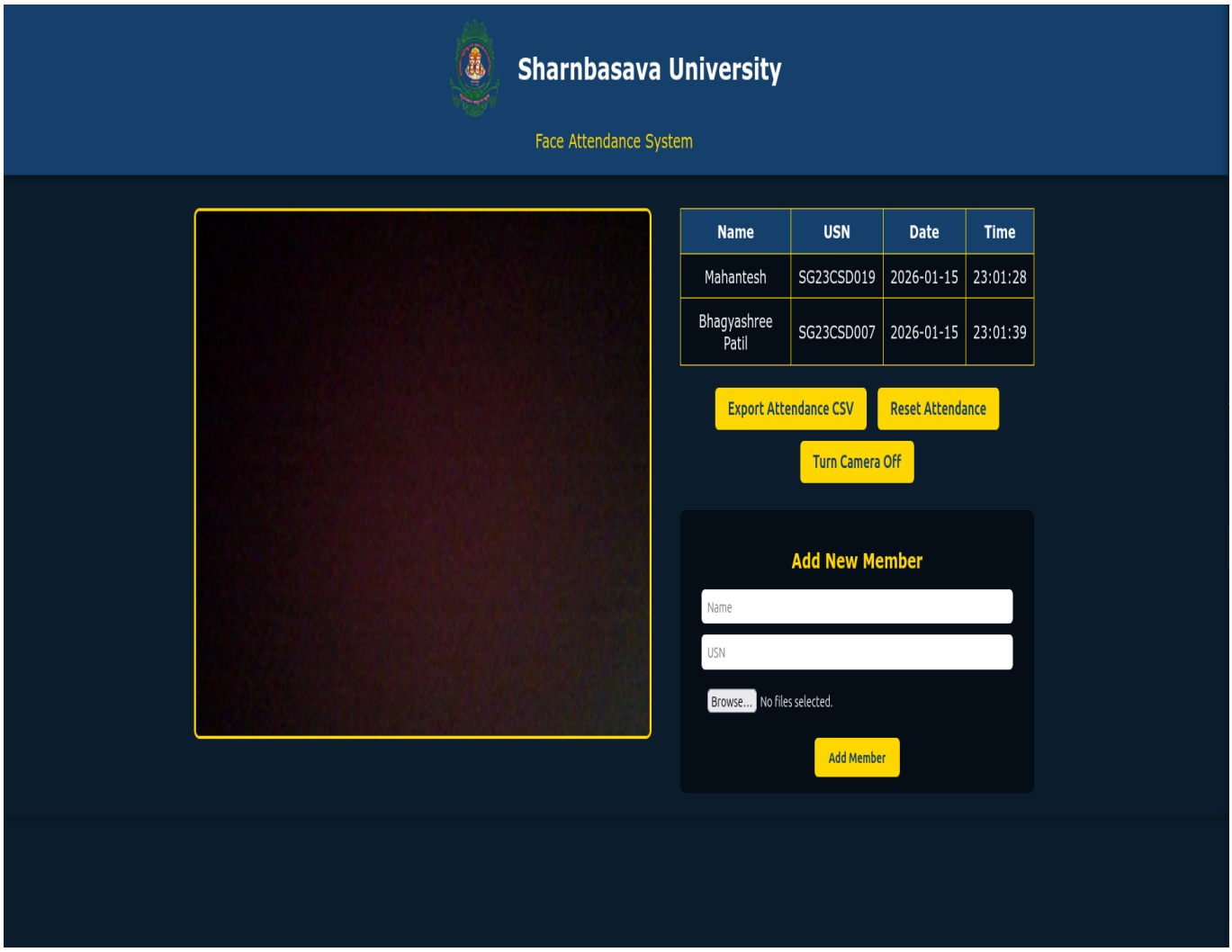
```
const toggleBtn = document.getElementById("toggleCam");
toggleBtn.onclick = async () => {
if (videoStream) {
stopCamera();
toggleBtn.textContent = "Turn Camera On";
} else {
await startCamera();
toggleBtn.textContent = "Turn Camera Off";
}
};

document.getElementById("addMember").onclick = async () => {
const name = document.getElementById("newName").value.trim();
const usn = document.getElementById("newUSN").value.trim();
const files = document.getElementById("newImages").files;
if (!name || !usn || files.length === 0) { alert("Please provide name, USN, and at least one image.");
return; }
const descriptors = [];
for (let file of files) {
const img = await faceapi.bufferToImage(file);
const detection = await faceapi.detectSingleFace(img).withFaceLandmarks().withFaceDescriptor();
if (detection) descriptors.push(detection.descriptor);
}
if (descriptors.length === 0) { alert("No faces detected in the provided images."); return; }

const newMember = new faceapi.LabeledFaceDescriptors(name, descriptors);
faceMatcher.labeledDescriptors.push(newMember);
members.push({ name, usn, images: [] });

savedMembers.push({ name, usn, images: [] });
localStorage.setItem("members", JSON.stringify(savedMembers));

alert(`Member "${name}" added successfully!`);
document.getElementById("newName").value = "";
document.getElementById("newUSN").value = "";
document.getElementById("newImages").value = "";
};
}

window.onload = init;
```

CHAPTER 5

RESULT INTERPRETATION

5.1 OUTCOMES

SNAPSHOT :- WELCOME WEBPAGE



CONCLUSION

The Face Attendance System implemented in this project demonstrates a practical, efficient, and modern approach to managing attendance in educational institutions. By leveraging **face recognition technology** through the **face-api.js library**, the system provides a **contactless, automated, and real-time solution** for tracking attendance, addressing many limitations associated with traditional manual or card-based systems.

The system successfully integrates multiple functionalities, including **real-time face detection, dynamic member management, offline operation, and attendance audit trails**. The use of a webcam for capturing live video ensures accessibility, while pre-trained neural network models enable accurate face recognition. By storing attendance records and snapshots locally using **LocalStorage**, the system maintains **privacy and security**, avoiding the need for internet connectivity. This feature makes it particularly suitable for environments where reliable offline operation is essential.

Through extensive testing, the system has proven to be **robust and reliable**. Measures such as multiple recognition confirmations, time interval checks, and prevention of duplicate entries ensure that attendance is marked accurately and fairly. The ability to export attendance data in CSV format adds flexibility for administrative tasks and institutional record-keeping.

Furthermore, the system's user-friendly interface and modular design make it **scalable and adaptable**. New members can be added easily, and the system can handle multiple students simultaneously without significant performance degradation. The visual audit trail through captured snapshots enhances accountability and provides verifiable records for future reference.

REFERENCES

1. **Gupta, K., Sharma, R., & Verma, P. (2020). *Real-time Face Recognition-Based Attendance System Using OpenCV.* International Journal of Advanced Research in Computer Science, 11(3), 45–52.**
2. **Sharma, R., Singh, A., & Kumar, S. (2019). *Automated Attendance System Using Convolutional Neural Networks*. International Journal of Computer Applications, 178(25), 12–18.**
3. **Taigman, Y., Yang, M., Ranzato, M., & Wolf, L. (2014). *DeepFace: Closing the Gap to Human-Level Performance in Face Verification*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1701–1708.**
4. **face-api.js. (2023). *Face Recognition in the Browser with JavaScript*. Retrieved from https://github.com/justadudewhohacks/face-api.js**
5. **Turk, M., & Pentland, A. (1991). *Eigenfaces for Recognition*. Journal of Cognitive Neuroscience, 3(1), 71–86.**
6. **OpenCV Documentation. (2023). *Computer Vision Library.* Retrieved from https://opencv.org/**
7. **Zhao, W., Chellappa, R., Phillips, P. J., & Rosenfeld, A. (2003). *Face Recognition: A Literature Survey.* ACM Computing Surveys, 35(4), 399–458.**