Module 5 – Frontend Development and User Interface

1. Introduction :

The **Frontend Development Module** focuses on designing and implementing the **user interface (UI)** and **user experience (UX)** of the **QR-Based Attendance System** using **React.js**.

The frontend acts as the **presentation layer**, allowing users (students and admins) to interact with the backend through an intuitive and responsive web interface.

This module ensures the system is **user-friendly**, **secure**, and **functional**, offering features such as:

- QR code scanning for students,
- Dashboard and attendance reporting for admins, and
- Seamless integration with backend APIs (developed in Module 4).

The design prioritizes **simplicity, clarity, and real-time interactivity**, ensuring efficient attendance management through an accessible web application.

---

2. Objectives :

The primary objectives of Module 5 are:

1. To design an intuitive and responsive **React.js frontend**.
2. To integrate the frontend with backend REST APIs using **Axios**.
3. To implement **QR scanning functionality** using camera input.
4. To build **admin dashboards** for attendance monitoring and reporting.
5. To ensure **secure and role-based navigation** using JWT authentication.
6. To provide an optimized and mobile-friendly user experience.

---

3. Frontend Technology Stack :

Component	Technology / Library
Frontend Framework	React.js (Vite or Create React App)
State Management	React Context API / Redux Toolkit
Routing	React Router DOM
HTTP Requests	Axios
QR Scanning	react-qr-reader / html5-qrcode
Styling	TailwindCSS / Material-UI
Authentication	JWT Token Validation
Build Tool	Vite / npm
Deployment	NGINX / AWS Amplify / Localhost

---

4. Frontend Architecture Overview :

4.1 Component-Based Architecture

```
src/
│
├── components/
│   ├── Navbar.jsx
│   ├── Footer.jsx
│   ├── QRScanner.jsx
│   ├── DashboardTable.jsx
│
├── pages/
│   ├── LoginPage.jsx
```

```
│   ├── RegisterPage.jsx
│   ├── StudentDashboard.jsx
│   ├── AdminDashboard.jsx
│
├── context/
│   ├── AuthContext.jsx
│
├── services/
│   ├── api.js
│
├── App.jsx
└── index.js
```

4.2 Frontend Data Flow

[User Interface] → [React Components] → [Axios API Call] → [Spring Boot Backend] → [MongoDB Database]

---

5. Key Frontend Modules :

5.1 Authentication and Role-Based Routing :

- **JWT Authentication:**
  After login, a JWT token is stored in the browser's localStorage.
- **Protected Routes:**
  Certain pages (like Admin Dashboard) are only accessible to authorized users.

Example Code:

```
// src/context/AuthContext.jsx

import { createContext, useState } from "react";
```

```
export const AuthContext = createContext();

export const AuthProvider = ({ children }) => {
  const [user, setUser] = useState(null);

  const login = (token, role) => {
    localStorage.setItem("jwt", token);
    localStorage.setItem("role", role);
    setUser({ token, role });
  };

  const logout = () => {
    localStorage.clear();
    setUser(null);
  };

  return (
    <AuthContext.Provider value={{ user, login, logout }}>
      {children}
    </AuthContext.Provider>
  );
};
```

Protected Route Example:

```
import { Navigate } from "react-router-dom";

const ProtectedRoute = ({ children }) => {
```

```
  const token = localStorage.getItem("jwt");
  return token ? children : <Navigate to="/login" />;
};
```

5.2 Login and Registration Pages :

LoginPage.jsx :

```
import React, { useState } from "react";
import axios from "axios";

function LoginPage() {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");

  const handleLogin = async () => {
    try {
      const response = await axios.post("/api/auth/login", { email, password });
      localStorage.setItem("jwt", response.data.token);
      alert("Login Successful!");
      window.location.href = "/dashboard";
    } catch (error) {
      alert("Invalid credentials");
    }
  };

  return (
    <div className="flex flex-col items-center p-10">
```

```
      <h1 className="text-2xl font-bold mb-5">Login</h1>
      <input type="email" placeholder="Email" onChange={(e) =>
setEmail(e.target.value)} className="border p-2 mb-3 w-64" />
      <input type="password" placeholder="Password" onChange={(e) =>
setPassword(e.target.value)} className="border p-2 mb-3 w-64" />
      <button onClick={handleLogin} className="bg-blue-600 text-white px-4
py-2 rounded">Login</button>
    </div>
  );
}

export default LoginPage;
```

5.3 Student Dashboard with QR Scanner :

Students can mark attendance by scanning the live QR code displayed by the admin.

QRScanner.jsx :

```
import React from "react";
import { QrReader } from "react-qr-reader";
import axios from "axios";

function QRScanner() {
  const handleScan = async (data) => {
    if (data) {
      try {
        const token = localStorage.getItem("jwt");
        await axios.post("/api/attendance/mark", { qrSessionId: data }, {
```

```
        headers: { Authorization: `Bearer ${token}` }
      });
      alert("Attendance marked successfully!");
    } catch (error) {
      alert("Error: " + error.response.data.message);
    }
  }
};

return (
  <div className="flex flex-col items-center mt-10">
    <h2 className="text-lg mb-4">Scan QR Code to Mark Attendance</h2>
    <div className="border rounded-lg p-4 shadow-lg">
      <QrReader
        onResult={(result, error) => {
          if (!!result) handleScan(result?.text);
        }}
        style={{ width: "300px" }}
      />
    </div>
  </div>
);
}

export default QRScanner;
```

5.4 Admin Dashboard :

The admin dashboard provides:

- Overview of attendance logs.
- Filters by date or user.
- Report export feature.

AdminDashboard.jsx :

```
import React, { useEffect, useState } from "react";
import axios from "axios";

function AdminDashboard() {
  const [logs, setLogs] = useState([]);

  useEffect(() => {
    const fetchLogs = async () => {
      const token = localStorage.getItem("jwt");
      const response = await axios.get("/api/admin/logs", {
        headers: { Authorization: `Bearer ${token}` }
      });
      setLogs(response.data);
    };
    fetchLogs();
  }, []);

  return (
    <div className="p-8">
      <h1 className="text-2xl font-semibold mb-5">Admin Dashboard</h1>
```

```
    <table className="table-auto w-full border-collapse border border-gray-
400">
      <thead className="bg-gray-200">
        <tr>
          <th className="border p-2">User ID</th>
          <th className="border p-2">QR Session</th>
          <th className="border p-2">Date</th>
          <th className="border p-2">Time</th>
          <th className="border p-2">Status</th>
        </tr>
      </thead>
      <tbody>
        {logs.map((log) => (
          <tr key={log.id}>
            <td className="border p-2">{log.userId}</td>
            <td className="border p-2">{log.qrSessionId}</td>
            <td className="border p-2">{log.date}</td>
            <td className="border p-2">{log.time}</td>
            <td className="border p-2 text-green-700">{log.status}</td>
          </tr>
        ))}
      </tbody>
    </table>
  </div>
 );
}
```

```
export default AdminDashboard;
```

5.5 API Integration

All backend endpoints are called using **Axios** with JWT authorization headers.

api.js

```
import axios from "axios";

const api = axios.create({
  baseURL: "http://localhost:8080/api",
});

api.interceptors.request.use((config) => {
  const token = localStorage.getItem("jwt");
  if (token) config.headers.Authorization = `Bearer ${token}`;
  return config;
});

export default api;
```

---

6. Styling and User Interface Design :

The interface follows a **minimalistic, responsive, and clean design** using **TailwindCSS** or **Material UI**.

6.1 UI Guidelines :

- Use a consistent color palette (blue, white, gray).
- Responsive grid layout for mobile and desktop.

- Rounded buttons and input fields.
- Interactive feedback (toasts, alerts).

6.2 Example Page Layout :

| Navbar: Logo | Dashboard | Logout |

| Sidebar (Admin): View Logs | Generate QR |

| Main Content: Reports / Scanner / Analytics |

| Footer: © 2025 QR Attendance System |

---

7. Testing and Validation :

Test Case	Expected Result	Status
Login with valid credentials	Redirect to dashboard	✅
Scan valid QR	Attendance recorded	✅
Scan expired QR	Error “QR expired”	✅
Access admin page as student	Access denied	✅
Logout clears session	Redirects to login	✅

All components were tested in browsers (Chrome, Edge, Firefox) and on mobile view using Chrome DevTools.

---

8. Expected Deliverables of Module 5 :

1. Complete **React.js frontend application**.
2. **QR scanner integration** for students.
3. **Admin dashboard** for viewing logs and reports.
4. **Role-based routing and JWT authentication**.
5. Fully responsive UI design tested across devices.

---

9. Future Path (Next Module Overview) :

Module 6 – Testing, Deployment, and Future Enhancements

- Conduct unit and system testing.
- Deploy using Docker and NGINX on AWS.
- Document all APIs and workflows.
- Plan enhancements such as facial recognition and mobile app support.

---

10. Conclusion :

The **Frontend Development and UI Module** provides the visual and interactive face of the QR-Based Attendance System.
Through a responsive React.js interface, it enables users to efficiently interact with the system while maintaining secure communication with the backend via JWT authentication.

This completes the user-facing layer of the project, setting the stage for **Module 6 – Testing, Deployment, and Future Enhancements**, which will finalize and optimize the system for real-world use.