

Module 2 – System Design and Architecture

1. Introduction :

After completing the ideation phase, the next step in the development of the QR-Based Attendance System is the System Design and Architecture stage.

This module focuses on translating the conceptual framework established in Module 1 into a technical blueprint that defines how various components — such as the frontend, backend, database, and QR processing engine — interact to create a cohesive and efficient system.

The design emphasizes modularity, security, and scalability, ensuring that the system can handle large numbers of users and attendance sessions while maintaining data accuracy and integrity.

2. Objectives of System Design :

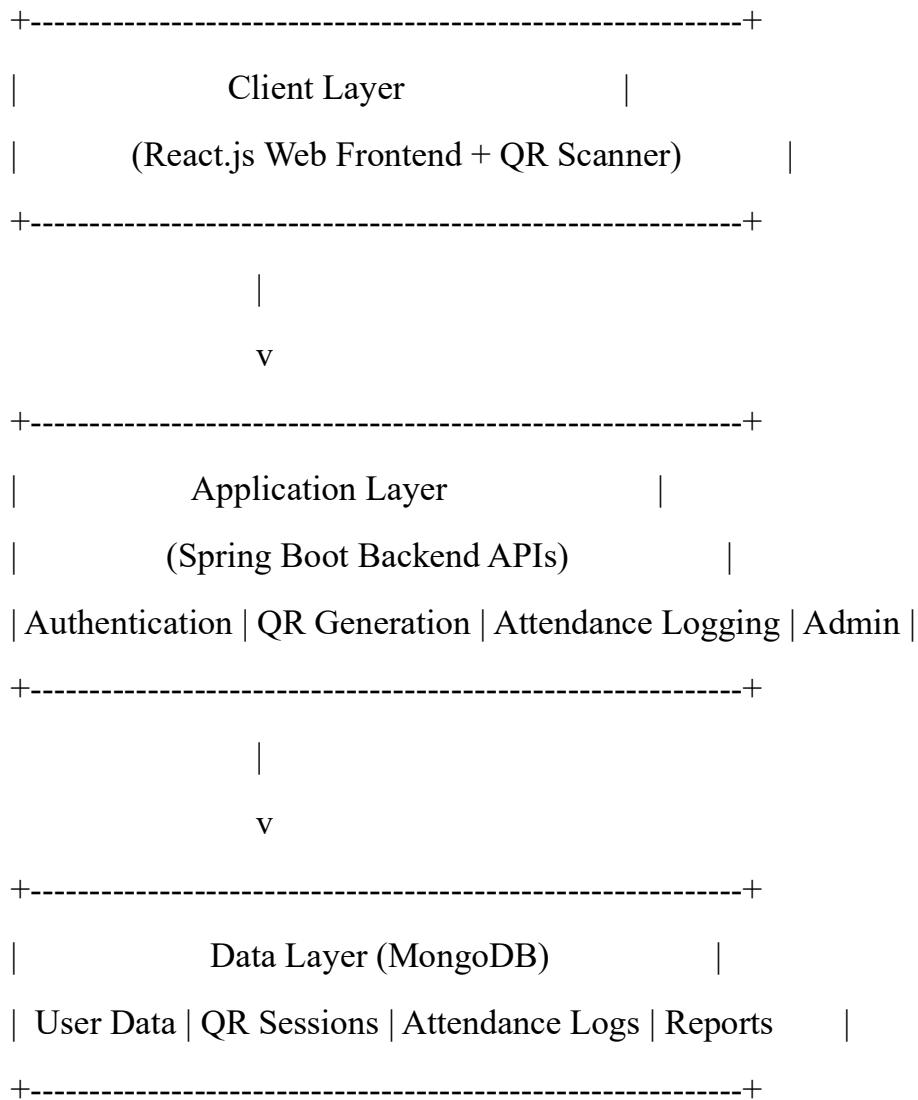
The primary goals of this module are to:

1. Design the overall architecture (high-level and low-level).
 2. Define the system components and their interactions.
 3. Create UML diagrams to visualize data flow and user interactions.
 4. Establish the database schema and its relationships.
 5. Outline the API structure and integration flow between modules.
 6. Ensure that security, maintainability, and scalability are considered in the system design.
-

3. High-Level Architecture :

The **QR-Based Attendance System** follows a **three-tier architecture**, consisting of the **frontend (presentation layer)**, **backend (application layer)**, and **database (data layer)**.

3.1 Architecture Overview :



3.2 Component Roles :

| Component | Technology | Purpose |
|-----------------|--------------------|--|
| Frontend | React.js | Provides web interface for students and admins, including QR scanning. |
| Backend | Spring Boot (Java) | Handles business logic, authentication, and data management via REST APIs. |
| Database | MongoDB | Stores user records, session data, and attendance logs. |

| Component | Technology | Purpose |
|-----------------------|-----------------------|--|
| Authentication | Spring Security + JWT | Manages secure login and role-based access. |
| QR Library | ZXing / QRGen | Generates and validates unique session QR codes. |

4. System Components :

4.1 Frontend (React.js)

- Implements the user interface for both students and admins.
- Integrates QR scanning using web camera access (react-qr-reader or html5-qrcode).
- Provides features like login, attendance confirmation, and dashboard view.
- Communicates with backend through RESTful API calls.

4.2 Backend (Spring Boot)

- Acts as the core logic controller of the application.
- Exposes APIs for authentication, QR generation, validation, and attendance management.
- Implements JWT-based authentication for secure access.
- Enforces validation to prevent duplicate or expired QR usage.

4.3 Database (MongoDB)

- Stores structured and unstructured data using flexible JSON-like documents.
- Three main collections:
 - User: Stores user information and roles.
 - QRSession: Stores session-specific QR data.

- AttendanceLog: Stores attendance records linked to users and sessions.

4.4 QR Generator and Validator

- Generates a unique QR code for each session with metadata (session ID, class ID, validity period).
- When scanned, the backend verifies:
 - Session validity (not expired).
 - User authenticity.
 - Duplicate entry prevention.

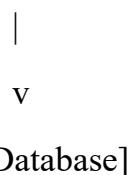
4.5 Admin Dashboard

- **Allows admins to:**
 - View and filter attendance logs.
 - Generate and export reports.
 - Manage user details (add, edit, delete).
 - Initiate new QR sessions.

6. Data Flow Diagram (DFD)

6.1 Level 0: Context Diagram

[Admin] ----> [QR-Based Attendance System] <---- [Student]



6.2 Level 1: Detailed Flow

1. Admin logs in and generates a QR code for the class/session.
2. The system stores the QR code details in the database with a validity timestamp.

3. Students scan the QR code through the frontend.
 4. The backend validates the QR code and the user's identity.
 5. Upon successful validation, attendance is logged.
 6. Admin retrieves and reviews attendance reports from the dashboard.
-

7. Database Design :

7.1 Collections and Schema Overview

1. User Collection

```
{  
  "id": "ObjectId",  
  "name": "String",  
  "email": "String",  
  "password": "String",  
  "role": "ADMIN/STUDENT"  
}
```

2. QRSession Collection

```
{  
  "id": "ObjectId",  
  "class_id": "String",  
  "date": "Date",  
  "valid_until": "DateTime"  
}
```

3. AttendanceLog Collection

```
{  
  "id": "ObjectId",  
  "user_id": "ObjectId",  
  "attendance": {  
    "status": "Present/Absent",  
    "date": "Date",  
    "time": "Time"  
  }  
}
```

```

    "qr_session_id": "ObjectId",
    "date": "Date",
    "time": "Time"
}

```

7.2 Relationships

- One **QRSession** can have multiple **AttendanceLogs**.
 - One **User** can be linked to multiple **AttendanceLogs**.
-

8. API Design Overview :

| HTTP Method | Endpoint | Description | Access |
|-------------|----------------------|---------------------------------------|---------|
| POST | /api/auth/login | Authenticate user and issue JWT token | Public |
| POST | /api/qr/create | Create a new QR session | Admin |
| POST | /api/attendance/mark | Record attendance after QR scan | Student |
| GET | /api/attendance/logs | Retrieve attendance logs | Admin |
| GET | /api/users | Fetch all users | Admin |

9. Security and Privacy Design :

- **JWT Authentication:** Ensures secure login and communication between frontend and backend.
- **Role-Based Access Control (RBAC):** Restricts admin and student permissions appropriately.
- **Data Encryption:** Passwords stored using BCrypt hashing.
- **Duplicate Prevention:** Each QR code session allows one entry per user.
- **Session Expiry:** Each QR code expires automatically after its scheduled duration.

10. Scalability and Performance Considerations :

- **Load Balancing:** NGINX or AWS Elastic Load Balancer for handling multiple requests.
 - **Containerization:** Dockerized services for deployment consistency.
 - **Database Indexing:** Optimize MongoDB queries for faster filtering.
 - **Caching:** Use Redis for frequently accessed data.
 - **Stateless Backend:** JWT allows scalable authentication without session persistence.
-

11. Expected Deliverables of Module 2 :

1. Completed **system architecture diagrams** (high- and low-level).
 2. Detailed **UML diagrams** (use case, class, DFD).
 3. Finalized **database schema** and relationships.
 4. Defined **API endpoints** for each core functionality.
 5. Documented **security and scalability** strategy.
-

13. Conclusion :

This module provides the complete architectural blueprint for the **QR-Based Attendance System**.

It defines how the system's components — frontend, backend, and database — interact cohesively to achieve automation, security, and efficiency.

The **UML and data flow diagrams** ensure clarity in development, while the defined **API and database structures** lay the groundwork for implementation in subsequent modules.

With a strong design foundation, the system is now ready for the next stage — **Database Implementation and Integration (Module 3)**.

