Module 3 – Database Implementation and Integration

1. Introduction :

In this module, we move from the design phase to the **implementation of the database** — the backbone of the **QR-Based Attendance System**.

The database ensures that all key entities such as users, QR sessions, and attendance logs are stored securely and can be accessed efficiently.

This module covers:

- The creation of the **MongoDB database schema**,
- Implementation of **Spring Boot entity models and repositories**,
- Establishing the **backend–database connection**, and
- Performing **CRUD operations** (Create, Read, Update, Delete) for each entity.

The objective is to build a solid data management layer that ensures **data integrity, security, and performance** throughout the system.

---

2. Objectives :

The key objectives of this module are:

1. To design and implement a robust **MongoDB database**.
2. To establish **data models** and **relationships** using Spring Boot.
3. To implement **repository layers** for CRUD operations.
4. To ensure **data validation and security** at the database level.
5. To integrate the **backend (Spring Boot)** with **MongoDB** for seamless communication.

---

3. Database Overview :

3.1 Database Type:

- **MongoDB (NoSQL Document Database)**
  Chosen for its:
- Schema flexibility (ideal for web-based applications).
- Scalability and performance in handling large attendance logs.
- JSON-based data structure compatible with REST APIs.

3.2 Database Name:

qr_attendance_db

3.3 Collections:

1. users
2. qr_sessions
3. attendance_logs

Each collection stores documents representing real-world entities.

---

4. Database Schema Design :

4.1 User Collection

Stores user credentials and role information.

Field	Type	Description
_id	ObjectId	Unique user ID
name	String	Full name of the user
email	String	Login email (unique)
password	String	Hashed password using BCrypt
role	String (Enum: "ADMIN", "STUDENT")	User type
createdAt	Date	Timestamp of account creation

Example Document:


```
{
  "_id": "64a9f0329a3f2d0012345678",
  "name": "John Doe",
  "email": "john@example.com",
  "password": "$2a$10$hashedPassword",
  "role": "STUDENT",
  "createdAt": "2025-10-12T10:00:00Z"
}
```

4.2 QRSession Collection

Stores details of each class/session for which a QR code is generated.

Field	Type	Description
_id	ObjectId	Session ID
classId	String	Identifier of the class
date	Date	Session date
validUntil	DateTime	Expiry time of the QR code
qrCode	String	Base64-encoded or URL of generated QR image

Example Document:


```
{
  "_id": "64a9f0553a3f2d0012345689",
  "classId": "CSE2025-DBMS",
  "date": "2025-11-12",
  "validUntil": "2025-11-12T11:15:00Z",
  "qrCode": "..."
```

}

4.3 AttendanceLog Collection :

Stores attendance records when users scan QR codes.

Field	Type	Description
_id	ObjectId	Attendance record ID
userId	ObjectId	Linked user
qrSessionId	ObjectId	Linked QR session
date	Date	Date of check-in
time	Time	Time of check-in
status	String (Enum: "PRESENT", "ABSENT")	Attendance status

Example Document:


```
{
  "_id": "64a9f0883a3f2d0012345700",
  "userId": "64a9f0329a3f2d0012345678",
  "qrSessionId": "64a9f0553a3f2d0012345689",
  "date": "2025-11-12",
  "time": "10:05:12",
  "status": "PRESENT"
}
```


-

---

6. Backend Integration with MongoDB

6.1 Configuration (application.properties)

```
spring.data.mongodb.uri=mongodb://localhost:27017/qr_attendance_db
spring.data.mongodb.database=qr_attendance_db
```

6.2 Entity Classes (Spring Boot)

User.java

```
@Document(collection = "users")
public class User {
    @Id
    private String id;
    private String name;
    private String email;
    private String password;
    private String role;
    private Date createdAt = new Date();
}
```

QRSession.java

```
@Document(collection = "qr_sessions")
public class QRSession {
    @Id
    private String id;
    private String classId;
    private Date date;
    private Date validUntil;
    private String qrCode;
}
```

AttendanceLog.java

```
@Document(collection = "attendance_logs")
```

```
public class AttendanceLog {
    @Id
    private String id;
    private String userId;
    private String qrSessionId;
    private Date date;
    private String time;
    private String status;
}
```

6.3 Repository Interfaces

UserRepository.java

```
public interface UserRepository extends MongoRepository<User, String> {
    Optional<User> findByEmail(String email);
}
```

QRSessionRepository.java

```
public interface QRSessionRepository extends MongoRepository<QRSession, String> {
    List<QRSession> findByDate(Date date);
}
```

AttendanceRepository.java

```
public interface AttendanceRepository extends MongoRepository<AttendanceLog, String> {
    List<AttendanceLog> findByUserId(String userId);
    List<AttendanceLog> findByQrSessionId(String qrSessionId);
}
```

---

7. CRUD Operations :

7.1 Create User

API Endpoint: POST /api/users/register


```
{
  "name": "Alice Sharma",
  "email": "alice@example.com",
  "password": "123456",
  "role": "STUDENT"
}
```


7.2 Create QR Session

API Endpoint: POST /api/qr/create


```
{
  "classId": "CSE2025-DBMS",
  "date": "2025-11-12",
  "validUntil": "2025-11-12T11:10:00"
}
```


7.3 Record Attendance

API Endpoint: POST /api/attendance/mark


```
{
  "userId": "64a9f0329a3f2d0012345678",
  "qrSessionId": "64a9f0553a3f2d0012345689"
}
```


---

8. Data Validation and Constraints :

- **Email Uniqueness:** Prevent duplicate user accounts.
- **QR Validity:** Attendance can only be marked before validUntil.

- **Duplicate Check:** One user per session → one attendance record.
- **Auto Timestamping:** Automatic recording of date/time on attendance.

---

9. Security Considerations :

- **BCrypt Password Hashing:** For all stored passwords.
- **JWT Authentication:** For API access.
- **Database Access Control:**
    - Separate user accounts for dev/test/production environments.
- **Backup Policy:**
    - Automated daily backup using mongodump.

---

10. Testing Database Integration :

Test Cases

Test Case	Input	Expected Output	Status
User Creation	JSON Payload	User inserted successfully	✅
Duplicate Email	Existing email	Error: “Email already exists”	✅
QR Session Creation	Class data	QR generated and stored	✅
Invalid QR Scan	Expired session	Error: “QR expired”	✅
Attendance Logging	Valid QR and User	“Attendance marked”	✅

---

11. Expected Deliverables of Module 3 :

1. Fully implemented MongoDB collections.
2. Backend models, repositories, and configurations integrated.
3. CRUD API endpoints tested and verified.
4. Data validation, encryption, and error handling implemented.

5. Working database connection between backend and application server.

---

13. Conclusion :

The **Database Implementation and Integration module** establishes a secure, scalable, and efficient data foundation for the **QR-Based Attendance System**.

With MongoDB successfully integrated and all collections properly modeled, the system is now capable of handling real-time user and attendance data.

This robust data layer ensures reliability and consistency in future operations such as QR validation, attendance reporting, and user management — forming the backbone of all subsequent modules.