COLLEGE CODE	:9623
COLLEGE NAME	:Amrita College of Engineering and Technology
DEPARTMENT	:Computer Science Engineering
STUDENT NM-ID	: 72B12532CC0A7CCBFE14FB74FAB82BF3
ROLL NO	:962323104064
DATE	:11-09-2025

Completed the project named as Phase 1

TECHNOLOGY PROJECT NAME :
AngularJS with SQL Integration

SUBMITTED BY,

NAME: NANDHAJA R.V`

MOBILE NO:94899668508

AngularJS with SQL Integration

1. Problem Statement

In today's data-driven world, businesses heavily depend on real-time access to structured data stored in databases. AngularJS, although a legacy front-end framework, is still widely used in many enterprise applications. However, integrating AngularJS with SQL databases often poses challenges due to weak API structures, security vulnerabilities, and poor scalability.

Key issues identified:

- Lack of direct communication between AngularJS and SQL database without a secure backend.
- Difficulty in managing large data sets with minimal delay.
- Need for role-based access control to ensure data security.
- Challenges in user experience due to outdated UI patterns and manual workflows.
- Inefficient data management causing duplication and inconsistencies.

The proposed solution is to develop an AngularJS application integrated with SQL through secure APIs, focusing on user-friendly data management, secure operations, and scalability for future needs.

2. Users & Stakeholders

Primary Users

1. Admin Users
   - o Manage user accounts, roles, and permissions.
   - o Responsible for ensuring data integrity.
   - o Can generate system-wide reports.
2. End Users (Employees/Customers)
   - o Perform CRUD operations (Create, Read, Update, Delete) on business data.

o Access data based on permissions assigned by Admin.

o Generate filtered views or personal reports.

Stakeholders

1. Business Owners / Management
   - o Expect improved efficiency and decision-making through quick data access.
   - o Want reduced costs in system maintenance.
   - o Demand a scalable system for future business growth.
2. Developers / Technical Team
   - o Need clear API documentation for maintainability.
   - o Should be able to integrate third-party services if required.
   - o Expect modular architecture for easy upgrades.
3. IT/Support Teams
   - o Responsible for system uptime and troubleshooting.
   - o Require admin-level logs for diagnosing issues.
   - o Ensure compliance with company security policies.

3. User Stories

Admin Stories

1. As an Admin, I want to add, edit, and remove users so that I can control who has access to the system.
2. As an Admin, I want to assign roles and permissions to users so that sensitive data remains protected.
3. As an Admin, I want to monitor user activity logs so that I can identify unauthorized actions.

End User Stories

4. As a User, I want to log in securely so that my data remains private.

5. As a User, I want to view a list of records in a searchable table so that I can quickly find relevant information.

6. As a User, I want to edit and update my own records so that I can keep data accurate.

7. As a User, I want to download reports so that I can use the data offline.

Business Owner Stories

8. As a Business Owner, I want system-generated reports so that I can make informed business decisions.

9. As a Business Owner, I want to reduce operational inefficiencies so that productivity increases.

Technical Stories

10. As a Developer, I want well-structured API endpoints so that integration becomes easier.

11. As IT Support, I want detailed error logs so that I can troubleshoot faster.

4. MVP Features

User Management Module

- o Login/Logout (Authentication using JWT or session management).
- o Role-based Access Control (Admin vs End-user).

Data Management Module

- o CRUD operations for SQL database records.
- o Search and Filter functionality for quick data access.
- o Validation to avoid duplicate/incomplete data.

Reporting Module

- o Simple reporting dashboard for data visualization.
- o Export functionality (CSV, Excel, or PDF).

Error Handling & Security

- o Input validation at both front-end and back-end.
- o SQL injection protection.
- o Error messages displayed in a user-friendly way.

UI/UX

- o Responsive AngularJS UI (desktop & mobile).
- o Simple navigation with minimal learning curve.

5. Wireframes & API Endpoint List

Wireframes (Conceptual)

1. Login Page – Username/Password   Auth token generated.

2. Admin Dashboard – User list, role management, and system reports.
3. User Dashboard – Data table of records with options to Add/Edit/Delete.
4. Report Page – Data summary charts and export buttons.

API Endpoint List

Authentication

- o POST /api/login – Authenticate user.
- o POST /api/logout – End session.

User Management

- o GET /api/users – Fetch user list.
- o POST /api/users – Add new user.
- o PUT /api/users/:id – Update user details.
- o DELETE /api/users/:id – Remove user.

Data Management

- o GET /api/records – Fetch all records.
- o POST /api/records – Insert a new record.
- o PUT /api/records/:id – Update existing record.
- o DELETE /api/records/:id – Delete record.

Reporting

- o GET /api/reports – Generate reports from SQL queries.
- o GET /api/reports/export – Export reports in CSV/Excel/PDF.

6. Acceptance Criteria

1. Login & Authentication
   - o User must be able to log in with valid credentials.
   - o Invalid login attempts should return appropriate error messages.
2. User Management
   - o Admins can add/edit/remove users.
   - o End-users cannot access admin-only features.
3. Data Handling
   - o Users can create, read, update, and delete records successfully.
   - o Data validation ensures no blank or duplicate entries.
4. Reporting
   - o Reports can be generated and exported without errors.

- o Filters must work correctly in dashboards.

5. UI/UX
   - o The system must be responsive across devices.
   - o Navigation must be intuitive with minimal training.
6. Security
   - o SQL injection attacks are prevented.
   - o Unauthorized users cannot access protected APIs.
7. Performance
   - o System should handle at least 1000 concurrent records without lag.
   - o API response time < 2 seconds for standard queries.
8. Error Handling
   - o Errors are logged in a structured way.
   - o User-facing errors are clear and actionable.