COLLEGE CODE	**:**9623
COLLEGE NAME	**:**Amrita College of Engineering and Technology
DEPARTMENT	**:**Computer Science Engineering
STUDENT NM-ID	**:**3D79BCC0253A11381F341B7934308AF4
ROLL NO	**:**962323104070
DATE	**:**11-09-2025

Completed the project named as Phase 2

TECHNOLOGY PROJECT NAME :
AngularJS with SQL Integration

SUBMITTED BY,

NAME: M.PADMA SHREE

MOBILE NO:9487108477

1. Tech Stack Selection

Frontend: AngularJS (MVC pattern, modular, reusable components, two-way data binding)

Backend: Node.js with Express.js (to handle API requests and act as middleware between AngularJS and SQL DB)

Database: MySQL or PostgreSQL (relational DB, structured queries, secure transactions)

ORM (Optional): Sequelize/TypeORM (to simplify DB operations with SQL)

Authentication: JWT (JSON Web Tokens) or OAuth2 for secure login sessions

Hosting:

Frontend	Vercel / Netlify / AWS Amplify
Backend + DB	AWS RDS + EC2, or DigitalOcean droplet, or Firebase alternative

2. UI Structure / API Schema Design

UI Structure

Login / Signup Screen – Authentication

Dashboard – Summary of user data

Main Functional Page(s):

Forms (input)

Tables (data view)

Charts (data visualization)

Admin Panel – Manage users, settings, logs

Error & Notification Component – Feedback to user

API Schema Design

APIs (RESTful, JSON-based):

Auth Routes:

POST /api/auth/login	Authenticate user
POST /api/auth/signup	Register user

User Routes:

GET /api/users/:id	Fetch user details
PUT /api/users/:id	Update user details

Data Routes (example for records):

GET /api/data	Fetch records
POST /api/data	Insert record
PUT /api/data/:id	Update record
DELETE /api/data/:id	Delete record

3. Data Handling Approach

Frontend (AngularJS):

Use Services for API calls (separation of concerns)

Use Controllers & Models for state/data binding

Use ngStorage / LocalStorage for caching session tokens

Backend (Node.js/Express):

Use API validation middleware (Joi/Express-validator)

Implement error handling (try/catch, custom error messages)

Maintain logging for DB queries

Database (SQL):

Normalize schema (3NF) to reduce redundancy

Index frequently queried fields for performance

Secure with parameterized queries (to prevent SQL injection)

4. Component / Module Diagram

Modules in AngularJS:

1. Auth Module – Login, signup, token handling
2. Dashboard Module – Summary view with charts & data
3. Data Management Module – CRUD (Create, Read, Update, Delete) operations
4. Admin Module – Manage users, roles, permissions
5. Shared Module – Common services, utilities, error handling, UI components

Backend Modules:

Auth Service

User Service

Data Service (CRUD ops)

DB Connection Layer

5. Basic Flow Diagram

```
[User]
  !
[AngularJS UI]
  !   (HTTP Request)
[Express.js Backend]
  !   (SQL Query via Sequelize/Knex)
[SQL Database]
  !   (Response)
[Backend formats JSON]
  !
[AngularJS Controller updates View]
  !
[User sees updated data]
```