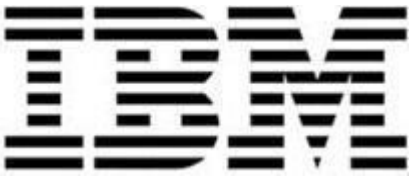

COLLEGE CODE : 8203

COLLEGE NAME : AVC COLLEGE OF ENGINEERING

DEPARTMENT : CSE

STUDENT NM ID : 4CF0B691E7230C98668406C5A3493455

ROLL NO : 820323104065

DATE : 08-09-25

COMPLETED THE PROJECT

NAMED AS : PHASE 1 TECHNOLOGY

PROJECT NAME : RESTFUL CONTACT

SUBMITTED BY,
MULLAI .P

MOBILE NO : 7548875472

IBM-NJ-RESTful Contact Management API

Phase 1 – Problem Understanding & Requirements

1. Problem Statement

Intoday’s digitalworld, organizations need an efficient way to manage their contacts — such as clients, vendors, partners, and team members. Traditional spreadsheets or manual methods are timeconsuming, prone to human error, and lack integration with modern applications.

The RESTful Contact Management API aims to provide a simple yet powerful interface to Create, Read, Update, and Delete (CRUD) contact information. It will serve as the backend service that can be integrated with various front-end platforms (web or mobile).
Key goals of the project include:

- Centralized storage of contactdetails (name, phone, email, address, etc.)
- Fast and secure access via RESTful endpoints.
- Scalability for future features (e.g., search, tagging, importing/exporting contacts).
- Clear documentation for developers to integrate the API.

2. Users & Stakeholders

Thesystemwillservedifferenttypes of users and stakeholders:

User/ Stakeholder	Role	Objective
End Users	Individuals or teams who will store and retrieve contacts using the API	Want easy contact management and integration with apps
Front-End Developers	Build UI (web/mobile) that consumes the RESTful API	Need reliable endpoints and clear documentation
Administrators	Oversee system usage, manage roles and permissions	Require secure authentication and monitoring tools
Project Manager	Supervises the project phases and timelines	Ensures requirements and deadlines are met
Testing Team	Validates the functionality and performance of the API	Ensures robustness and usability
Client / Organization	Owns the product	Needs scalable and maintainable solution

3. User Stories

Wecapturerequirements as user stories to reflect real-world needs:

1. Asan end user, I want to add a new contact with name, email, phone, and address so that I can store important information.
2. Asan end user, I want to update a contact's details whenever information changes.
3. Asan end user, I want to delete a contact I no longer need.
4. Asan end user, I want to search for contacts by name, phone, or email for quick access.

5. As a developer, I want to access contacts via a RESTful API so I can integrate it into my applications.
6. As an administrator, I want to manage users and permissions to ensure secure access.
7. As a client, I want reliable data storage and performance to support future growth.

4. MVP (Minimum Viable Product) Features

The first version of the API will focus on core features:

- User registration and authentication(basic or token-based).
- CRUD operations for contacts.
- Input validation and error handling.
- Simple search by name or email.
- RESTful API documentation (e.g.,Swagger/OpenAPI).
- Logging basic activities for debugging.

Future enhancements (not in MVP):

- Import/export contacts (CSV, Excel). Bulk
- contact management. Role-based access control.
- Advanced analytics (e.g., contact usage reports).
-

5. Wireframes / API Endpoint List

a) Wireframe (Concept)

- For a simple front-end(optional for API testing):
- Login / Signup screen.
- Dashboard showing listof contacts.
- Add Contact form.
- Contact detail view &edit form.

b) API Endpoint List (Tentative)

Endpoint	Method	Description
/api/auth/register	POST	Register a new user
/api/auth/login	POST	Login to receive authentication token
/api/contacts	GET	Retrieve all contacts for logged-in user
/api/contacts/:id	GET	Retrieve specific contact by ID
/api/contacts	POST	Create a new contact
/api/contacts/:id	PUT	Update an existing contact
/api/contacts/:id	DELETE	Delete a contact
/api/contacts/search	GET	Search contacts by query parameter

6. Acceptance Criteria

Theprojectwillbeconsideredacceptable when:

- The API performs all CRUD operations successfully with valid data.
- All endpoints return appropriate status codes and messages.
- Authentication and basic security measures are implemented.
- Data persists in a reliable database (e.g., MongoDB or PostgreSQL).
- API documentation is provided for developers.
- Unit and integration tests pass for core features.
- The API integrates smoothly with a sample front-end or API testing tool (e.g., Postman).