



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 3 TECHNOLOGY

PROJECT NAME : RESTFULCONTACTMANAGEMENT API

SUBMITTED BY,  
MULLAI P

MOBILE NO : 7548875472

# Phase 3 – MVP IMPLEMENTATION

## 1. Project Setup

The project setup stage is critical to lay the foundation for smooth development and deployment. This includes:



### Environment Configuration:

Setting up the local development environment with required tools such as Node.js/Express (or another framework), Postman/Insomnia for API testing, and a preferred code editor (VS Code).



### Directory Structure:

Designing a clean, modular folder structure to ensure scalability and maintainability. For example:



### Dependencies Installation:

Installing core dependencies such as Express.js for routing, bodyparser for request parsing, Mongoose/Sequelize/Prisma for database interaction, and dotenv for environment variables.



### API Base URL:

Defining a consistent base URL for all endpoints (e.g., /api/v1/contacts).



### Security and Configurations:

Setting up environment variables for database connection strings, API keys, and JWT secrets if authentication is planned.

This systematic setup ensures that the core API features can be implemented and extended easily.

## 2. Core Features Implementation

The core of the MVP is the contact management functionality, implemented as a RESTful API. The major operations to be implemented:

- Create Contact (POST):

Allows adding a new contact with fields like name, email, phone number, and address.

- Read Contacts (GET):

Fetches all contacts or a single contact by ID. Supports pagination and filtering where necessary.

- Update Contact (PUT/PATCH):

Allows updating an existing contact's details such as email, phone, or address.

- Delete Contact (DELETE):

Removes a contact from the system by its ID.

- Search (GET /search): for searching

- by name or email,  
enhancing the API usability.

- Validation & Error Handling:

Applying input validation (e.g., using Joi or express-validator) and sending structured error responses for invalid requests.

Each feature is implemented following RESTful principles, ensuring stateless communication, proper use of HTTP verbs, and meaningful status codes.

### 3. Data Storage (Local State / Database)

A reliable data storage mechanism is essential for a contact management system. Phase 3 covers designing and implementing data persistence.

#### ● Database Selection:

Using a relational database(MySQL/PostgreSQL) or NoSQL (MongoDB) depending on the project requirements. For MVP, MongoDB is often preferred due to its flexibility with JSON-like documents.

#### ● Schema Design:

Creating a Contact schema/model with fields like:

- a. id (Primary Key/UUID)
- b. name (String)
- c. email (String, unique)
- d. phone (String)
- e. address (String)
- f. createdAt / updatedAt

#### ● Local Development Database:

Running a local instance of the database using Docker or a local server.

#### ● CRUD Integration:

Implementing data access logic in models or repository layers, and ensuring secure queries to avoid injection attacks.

## ● Testing with Seed Data:

Populating the database with sample contacts for development and testing.

## 4. Testing Core Features

Testing ensures the API works as expected and meets quality

standards. During Phase 3, the focus is on unit testing, integration testing, and manual testing.

## ● Unit Testing:

Using frameworks like Jest, Mocha, or Chai to test controllers, routes, and database logic individually.

## ● Integration Testing:

Testing how different components (routes, middleware, database) work together.

## ● Manual Testing with Postman/Insomnia:

Verifying endpoints, checking HTTP response codes, headers, and payloads.

Test Cases Examples:

- a. Add a new contact with valid/invalid data.
- b. Fetch contacts list with pagination.
- c. Update a contact and check if changes persist.
- d. Delete a contact and confirm removal from the database.

## ● Error Scenarios:

Testing invalid inputs, missing fields, duplicate emails, and unauthorized access (if authentication added).

This testing stage helps identify bugs early and ensures the MVP is stable for the next phase.

## 5. Version Control (GitHub)

[https://github.com/miruthulalakshmi/Nm\\_IBM\\_AVC.git](https://github.com/miruthulalakshmi/Nm_IBM_AVC.git)

## 6.Expected Deliverables of Phase 3

- A fully functional RESTful Contact Management API with CRUD endpoints.
- A working database integration with persistent storage.
- Test cases and test results demonstrating the core features work correctly.
- GitHub repository with clean commits and documented code.
- APostman collection for endpoint testing.

## 7. Diagram – Phase 3 Overview

Below is a simple diagram showing how the components interact in Phase 3:

