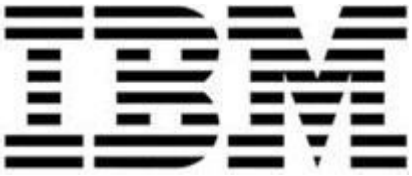

COLLEGE CODE : 8203

COLLEGE NAME : AVC COLLEGE OF ENGINEERING

DEPARTMENT : CSE

STUDENT NM ID : 574cdbfbdf129babef54e95edad1d5e9

ROLL NO : 820323104062

DATE : 08-09-25

COMPLETED THE PROJECT

NAMED AS : PHASE 3 TECHNOLOGY

PROJECT NAME : RESTFUL CONTACT MANAGEMENT API

SUBMITTED BY,

MIRUTHULALAKSHMI D.K

MOBILE NO : 9585920790

Phase 3 – MVP IMPLEMENTATION

1. Project Setup

Theproject setup stage is criticaltolay the foundation for smooth development and deployment. Thisincludes:

- Environment Configuration:

Setting up the local developmentenvironment with required tools such as Node.js/Express (or anotherframework), Postman/Insomnia for API testing, and a preferred codeeditor (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 requestparsing, Mongoose/Sequelize/Prisma for database interaction,and dotenv for environment variables.

- API Base URL:

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

- Security and Configurations:

Setting up environment variablesfor 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

Thecoreofthe MVPisthe contactmanagement 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 & Filtering (Optional):
Implement query parameters 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 ontheproject requirements. For MVP, MongoDB is often preferreddue 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 accesslogic in models or repository layers, and ensuring secure queriestoavoid injection attacks.

- Testing with Seed Data:
Populating the database with sample contacts for development and testing.

4. Testing Core Features

Testing ensures the API worksasexpected andmeetsquality standards.During Phase 3,thefocus is on unit testing, integration testing, and manual testing.

- Unit Testing:
UsingframeworkslikeJest,Mocha, or Chai to test controllers, routes, and database logicindividually.

- Integration Testing:
Testinghowdifferentcomponents (routes, middleware, database) work together.

- Manual Testing with Postman/Insomnia:
Verifying endpoints,checkingHTTPresponsecodes, headers, and payloads.

Test Cases Examples:

a. Add a new contactwith valid/invalid data.

b. Fetch contacts listwith pagination.

c. Update a contactandcheck if changes persist.

d. Delete a contactandconfirm removal from the database.

- Error Scenarios:
Testinginvalidinputs,missing fields, duplicate emails, and unauthorized access (ifauthentication added).

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

5. Version Control (GitHub)

Version control is vital tomaintain code history, manage collaboration, and supportdeployment pipelines. In this phase:

- Git Repository Setup:
Initializing a Git repositoryand pushing it to GitHub.
- Branching Strategy:
Following GitFloworfeature-branch workflows (e.g., main, develop, feature/add-contact).
- Commit Practices:
Writing clear, meaningful commit messages to document changes.
- Pull Requests and Code Reviews:
Using GitHub pull requests for peer reviews before merging into the main branch.
- Issue Tracking:
Using GitHub Issuesor Projects to manage tasks, bugs, and progress.

This ensures the project remains organized, changes are tracked, and collaboration between developers is efficient.

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.
- A Postman collection for endpoint testing.

7. Diagram – Phase 3 Overview

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