COLLEGE CODE: 8203

COLLEGE: AVC COLLEGE OF ENGINEERING

DEPARTMENT: INFORMATION TECHNOLOGY

STUDENT NM-ID: A8497BAAFDD3B8D816E4BACCB59D2039

ROLL NO: 23IT119

DATE:22/09/2025

Completed the project named as Phase III

TECHNOLOGY PROJECT NAME: Feedback Collection System

SUBMITTED BY,

NAME: YASARDEEN.P

MOBILE NO: 6369675068

PROJECT SETUP :

Step	Description	Tools Involved
1. Project Initialization & Dependencies	Create backend and frontend folders. Initialize Node.js project. Install essential packages (Express, Mongoose, Cors, Nodemailer, Sentiment).	npm init -y, npm install express mongoose cors nodemailer sentiment body-parser
2. Environment Configuration	Create a .env file in backend directory to securely store critical keys (MongoDB connection string, email credentials).	dotenv, MongoDB Atlas, SMTP credentials
3. Server Initialization	Create backend/server.js to set up Express.js application, define port (5000), configure middleware, and connect to MongoDB.	Node.js, Express.js, Mongoose
4. Database Schema Setup	Define Mongoose schema for feedback (name, email, feedback, rating, sentiment, timestamp).	Mongoose, MongoDB
5. Frontend File Structure	Create core UI files in frontend folder: index.html (form), style.css (design), app.js (API calls to backend).	HTML, CSS, JavaScript

CORE FEATURES IMPLEMENTATION:

This phase implements the core functionality of the Feedback Collection System:

1. REST API ENDPOINTS (EXPRESS.JS):

- POST /feedback → Receives user input, validates, performs sentiment analysis, and stores feedback in MongoDB.
- GET /feedback → Retrieves all stored feedback.
- GET /feedback?rating=5 → Retrieves filtered feedback by rating.

2. INPUT VALIDATION:

- Ensures all required fields are present.
- Rating must be between 1–5.

3. SENTIMENT ANALYSIS INTEGRATION:

- Uses the npm sentiment package to analyze text feedback.
- Stores result as Positive or Negative.

4. EMAIL CONFIRMATION:

- Uses Nodemailer to send confirmation email after submission.

5. FRONTEND DISPLAY:

- User submits feedback through form.
- Confirmation or error message is shown based on backend response.

DATA STORAGE (LOCALSTATE/DATABASE):

The Feedback Collection System uses MongoDB as the primary database:

1. PRIMARY DATA SOURCE:

- MongoDB stores structured feedback documents.
- Each document contains: name, email, feedback, rating, sentiment, createdAt.

2. SCHEMA DESIGN (MONGOOSE):

- name: String (required)
- email: String (required)
- feedback: String (required)
- rating: Number (1–5, required)
- sentiment: String (auto-calculated)
- createdAt: Date (default: now)

3. DATA FLOW:

- User submits form → Backend validates & analyzes → Data saved in MongoDB → Confirmation sent to user.

TESTING CORE FEATURES:

Testing ensures that the system works as expected across all modules:

1. API FUNCTIONALITY TEST (POSTMAN):

- Test Case 1: Submit valid feedback → 200 OK, stored in DB.
- Test Case 2: Submit invalid rating → 400 Bad Request.
- Test Case 3: Missing required fields → Error message returned.

2. DATABASE TEST:

- Ensure data is correctly inserted into MongoDB.
- Verify schema validation rejects incorrect entries.

3. EMAIL TEST:

- Confirm that Nodemailer successfully sends confirmation emails.
- Handle failures gracefully with error messages.

4. FRONTEND TEST:

- Verify form validation (empty fields, invalid email).
- Check success and error messages display correctly.

VERSION CONTROL (GITHUB):

GitHub is used for version control and collaboration:

- A dedicated repository is created for the Feedback Collection System.(ex: https://github.com/YASARDEEN/NAAN-MUDHALVAN-IBM-AVCCE-YASARDEEN_P.git)
- All changes are tracked using commits.
- Branching is used for developing features separately before merging.
- Issues and pull requests are used to track bugs and improvements.