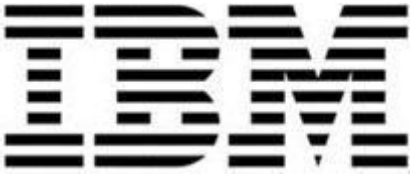

COLLEGE CODE:0004

COLLEGE NAME : MADRAS INSTITUTE OF TECHNOLOGY, ANNA UNIVERSITY.

DEPARTMENT : INFORMATION TECHNOLOGY.

Completed the project named as

Phase : 3 – FE

NAME : FEEDBACK COLLECTION SYSTEM

SUBMITTED BY:

NAME :KALIMUTHU K

Feedback Collection System-project

Phase 3: MVP IMPLEMENTATION

Project Setup

The Implementation involved setting up and developing the Feedback Collection System using the MERN Stack (MongoDB, Express.js, React.js, Node.js).
This phase focused on implementing all essential modules such as feedback form submission, admin authentication, and analytics visualization.

Frontend (React.js)

The frontend was built using React.js to provide a fast, responsive, and modular Single Page Application (SPA). Material UI was used for styling, and Axios handled data exchange with the backend.

Implementation :

- Created dynamic pages for user feedback, admin login, and analytics.
- Managed navigation using React Router.
- Integrated Chart.js for visual analytics in the admin dashboard.
- Designed all pages with a modern, clean interface using Material UI.

Backend (Node.js + Express)

The backend was developed using Node.js and Express.js to manage APIs, user authentication, and database operations. A MongoDB Atlas database was configured for persistent cloud storage.

Database Connection :

```
mongoose.connect(process.env.MONGO_URI)
.then(() => console.log("MongoDB Connected Successfully"))
```

- Implemented RESTful APIs for all major system functionalities.
- Used bcrypt.js for password encryption.
- Integrated JWT (JSON Web Token) for secure admin authentication.
- Established CORS and dotenv configurations for safe environment management.

Core Features Implementation

The system consists of four key modules developed and integrated during this phase: Landing Page, Feedback Form Page, Creator Login/Registration, and Creator Dashboard. Each module communicates with the MongoDB database via the backend APIs**.**

Landing Page

The landing page acts as the main entry point for all users. It provides access to available feedback forms and a login option for administrators.

Functions:

- Display all available feedback forms fetched through the /api/forms endpoint.
- Include a Search Bar for filtering forms by title.
- Provide an Admin Login Button redirecting to the creator login interface.
- Ensure real-time synchronization of form availability with the database.

Feedback Form Page

The feedback page collects structured user feedback dynamically based on the selected form. Each feedback form retrieves its structure from the database through the form ID.

Implemented Data Fields:

- Name *(optional)*
- Email *(optional)*
- Rating *(1–5 scale)*
- Comments *(text area)*
- Custom Questions *(configured by creator)*

API Endpoint Used:

Method	Endpoint	Description
POST	/api/admin/login	Authenticate admin
POST	/api/forms	Create new feedback form
GET	/api/forms	Retrieve all feedback forms
POST	/api/forms/:id/submit	Submit feedback
GET	/api/forms/:id/results	View form results (admin only)

Functionality:

- Form data is validated and submitted via Axios.
- Feedback is stored in the responses collection in MongoDB.
- Confirmation message displayed upon successful submission.

Creator Login / Registration

The Creator (Admin) login and registration system enables secure authentication and access control.

Endpoints:

- POST /api/creator/register
- POST /api/creator/login

Workflow:

1. New creator registers with name, email, and password.
2. Password is encrypted using bcrypt before saving to MongoDB.
3. On successful login, a JWT Token is generated and stored in the session.
4. Authenticated users gain access to the Creator Dashboard.

Security Implementation:

- JWT-based authentication to prevent unauthorized access.
- Encrypted password storage for enhanced privacy.
- Token validation through middleware in each API request.

Creator Dashboard

The dashboard serves as the management and analytics hub for each registered admin. It displays feedback form insights, allows form creation, and enables data visualization.

Dashboard Functionalities:

- **Create New Form:**
  - Endpoint: POST /api/forms/create
  - Allows defining title, description, and question type**.**
- **View All Forms:**
  - Endpoint: GET /api/forms/:adminId
  - Displays all forms created by the current admin.
- **Edit Form Details:**

- Endpoint: PUT /api/forms/:id
- Enables modifying questions or titles.

- **View Analytics:**
  - Endpoint: GET /api/responses/:formId
  - Displays total feedback responses and average ratings.

Visualization:

- Used react-chartjs-2 to generate bar and pie charts.
- Graphs show user response distribution, average rating, and comment summaries.

Data Storage

- The system uses **MongoDB Atlas** for secure and flexible storage of all data entities. Each data component is structured as a collection, ensuring efficient retrieval and scalability.

Collection Name	Purpose	Example Fields
admins	Stores admin credentials and session data	{ adminName, email, passwordHash }
forms	Holds all form metadata and questions	{ title, description, questions[] }
responses	Stores user feedback linked to formId	{ formId, name, rating, comments }

Testing Core Features

All system modules were tested thoroughly using Postman, MongoDB Atlas, and manual UI validation. Both frontend and backend tests ensured consistent data flow and accuracy**.**

Feature	Test Type	Testing Method	Tool Used
API Endpoints	Backend	CRUD operation verification	Postman
Form Submission	Frontend	Manual data entry & DB validation	Browser + MongoDB Atlas
Admin Authentication	Integration	Token-based login validation	Postman

Dashboard UI	Frontend	Interface responsiveness & data display	Browser
Analytics Visualization	Frontend	Chart rendering & metric accuracy	Browser

Version Control (GitHub)

Git and GitHub were used for maintaining version control and ensuring collaborative workflow throughout development.

Commands Executed:

*git init

*git add .

*git commit -m "Initial MVP setup"

*git push -u origin main

Branching Workflow:

- main → Stable branch for production-ready code.
- dev → Active development branch.
- feature/ → Separate feature-specific branches (e.g., feature/dashboard-ui).

Commit Message Format

Type	Example Message
feat	feat: added admin dashboard analytics module
fix	fix: resolved feedback API error
ui	ui: enhanced landing page design

The use of GitHub ensured continuous tracking of code changes and smooth integration between frontend and backend teams.