COLLEGE CODE : 9222

COLLEGE NAME : THENI KAMMAVAR SANGAM COLLEGE OF TECHNOLOGY

DEPARTMENT : B.TECH(INFORMATION TECHNOLOGY)

STUDENT NM-ID : A857FF10960EB4C8CDB46821869

ROLL NO : 922223205012

DATE : 24.09.2025

Completed the project named as Phase_2_ TECHNOLOGY

PROJECT NAME : IBM-NJ- FEEDBACK COLLECTION SYSTEM

SUBMITTED BY,

NAME: GOVINDARAJAN R

MOBILE : 8825464206

FEEDBACK COLLECTION SYSTEM

SOLUTION DESIGN AND ARCHITECTURE

Tech Stack Selection :

The tech stack plays a crucial role in ensuring the system's scalability, performance, and maintainability.The following technologies were selected for this system:

Frontend:

- **Framework**: **React.js**React is a popular JavaScript library for building user interfaces. It provides a fast, responsive user experience and is highly modular, making it easier to maintain and scale as the system grows..

Backend:

- **Server**: **Node.js with Express**Node.js is a non-blocking, event-driven runtime environment. Express is a minimalistic framework that sits on top of Node.js, making it easy to build RESTful APIs. This combination ensures fast request handling and supports concurrent connections efficiently.

UI Structure / API Schema Design :

UI Structure:

The frontend UI is designed to be clean, simple, and intuitive for both users submitting feedback and admins analyzing the feedback. The main UI components are:

- **Feedback Submission Page**:
  - **Feedback Form**: Includes fields for selecting feedback category (e.g., bug, feature request), rating (1-5 stars), and additional comments.
  - **Submit Button**: Upon clicking, feedback data is sent to the backend via API call.
- **Admin Dashboard**:
  - **Feedback Overview**: Displays a summary of feedback with filters based on category, date, and rating.
  - **Feedback Details**: List of individual feedback submissions, with options to mark them as resolved or escalate them to relevant teams.

API Schema Design:

1. **POST /feedback**
   - **Request Body**: `{ "userId": "12345", "category": "bug", "rating": 4, "comme`
     `crashes on login" }`
   - **Response**: `{ "success": true, "message": "Feedback submitted successfully"`
2. **GET /feedback**
   - **Request Parameters**: `{ "page": 1, "limit": 10 }`
   - **Response**:
3. `{`

```
  "feedback": [
    { "id": "123", "userId": "12345", "category": "bug", "rating": 4, "comment": "App crashes on login", "date": "2025-09-23" },
    { "id": "124", "userId": "67890", "category": "feature", "rating": 5, "comment": "Great feature!" }
  ],
  "totalCount": 50
}
```

10. **POST /auth/login**
    - **Request Body**: `{ "username": "admin", "password": "password123" }`
    - **Response**: `{ "token": "jwt-token-here" }`
11. **GET /feedback/{id}**
    - **Request**: `{ "id": "123" }`
    - **Response**:

```
{
  "id": "123", "userId": "12345",
  "category": "bug",
  "rating": 4,
  "comment": "App crashes on login",
  "date": "2025-09-23"

}
```

Data Handling Approach :

Data Integrity:

- **Validation**: The backend ensures that the rating is between 1 and 5, and that the comment is a valid string. If any data is missing or invalid, an error message is returned.

Error Handling:

- Proper error handling is in place at every stage, ensuring users receive meaningful error messages if their feedback submission fails (e.g., missing fields, database connection failure).

Component / Module Diagram :

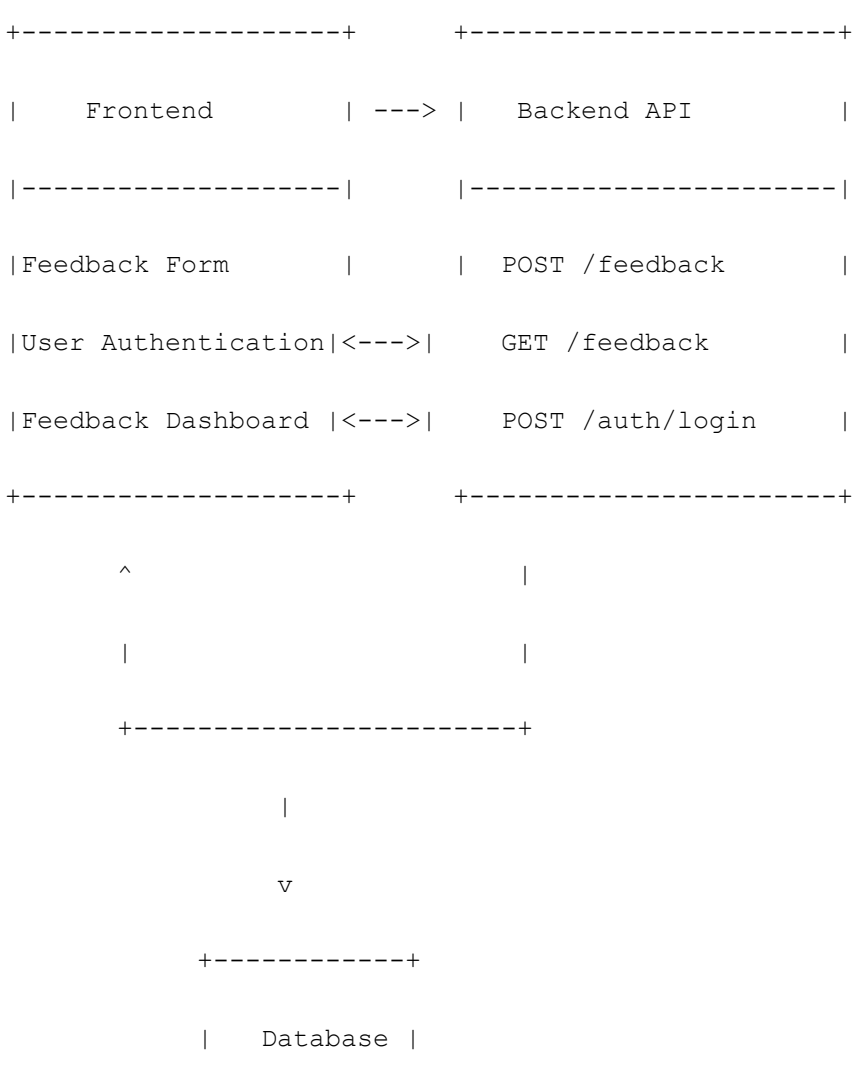
Frontend Components:

- **Feedback Form Component**: Handles user input and submits the data.
- **Dashboard Component**: Displays analytics and detailed feedback.

Backend Modules:

- **API Gateway**: Routes incoming requests to the correct service.
- **Feedback Service**: Manages CRUD operations for feedback.

Component Interaction Diagram:



```
| (MongoDB)  |

+------------+
```

Basic Flow Diagram :

1. **User Flow**:
   - **Start** → User visits feedback page → User fills out feedback form → Feedback is submitted via `POST` /feedback → System stores feedback in database → Success message displayed → **End**.
2. **Admin Flow**:
   - **Start** → Admin logs in → Admin views feedback dashboard → Admin analyzes feedback, filters by category/rating → Admin takes action (e.g., mark as resolved) → **End**.

Flow Diagram:

```
Start

|

v

User logs in or accesses feedback page

|

v

User submits feedback (rating, category, comment)

|

v

Feedback sent to server (POST /feedback)

|

v

Server stores feedback in database

|

v

User sees success message or failure error

|
```