DevOps Project

INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)
Dundigal, Hyderabad – 500 043

Certificate

This is to certify that it is a bonafied record of practical work done by Mr. / Ms.

P. SRIMAN _____ bearing the roll no. **23951A66L5** _____

of **IV Sem** _____class **CSE(AI&ML)** _____ branch in

the **DevOps** _____laboratory during the academic

year **2024-2025** _____ under our supervision.

Head of the department **Lecturer – in charge**

Signature of External Examiner **Signature of Internal Examiner**

DevOps Project on:

FEED BACK COLLECTIONS

INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal, Hyderabad - 500 043, Telangana

Bachelor of Technology

in

Computer Science and Engineering(AI&ML)

by

P. SRIMAN

23951A66L5

DECLARATION

I certify that

a. The work contained in this report is original and has been done by me under the guidance of my supervisor (s).
b. The work has not been submitted to any other Institute for any degree or diploma.
c. I have followed the guidelines provided by the Institute for preparing the report.
d. I have conformed to the norms and guidelines given in the Code of Conduct of the Institute.
e. Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the report and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

P. sriman

Place: Hyderabad **Signature of the Student**

Date: 03-07-2025 **Roll No: 23951A66L5**

ABSTRACT

The Feedback Collection Web Application is a dynamic, user-focused system designed to gather, manage, and analyse user feedback for continuous improvement of products, services, and digital experiences. Developed as part of a DevOps Engineering initiative, the application emphasizes real-time user engagement and agile response mechanisms. The system aims to bridge the gap between end-users and developers by providing a structured platform where users can express their opinions, report bugs, suggest features, and share overall satisfaction with the services or products they interact with.

Built using core web technologies such as **HTML** for layout and structure and **JavaScript** for interactivity, the application offers a responsive and accessible user interface. The development lifecycle was enhanced through the use of **Git** for version control and **GitHub** for remote collaboration and repository management. These tools facilitated effective team collaboration, iterative development, and efficient issue tracking. **Visual Studio Code** served as the primary development environment, allowing for streamlined code editing, debugging, and performance optimization.

From a DevOps perspective, this project showcases the application of continuous integration and automation to ensure that changes are efficiently tested and deployed. This approach supports iterative feature releases, prompt bug fixes, and active user-driven enhancements. The feedback collected through the system enables developers and organizations to make informed decisions based on actual user needs and behaviour patterns, enhancing overall software quality and user satisfaction.

The project highlights the synergy between user center design and DevOps principles in modern software engineering. It reflects how feedback systems play a vital role in increasing usability, improving customer loyalty, and guiding continuous improvement in digital applications.

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 About the Project

This project centers around building a responsive and user-friendly **Feedback Collection Web Application**, aimed at enabling users to easily submit feedback regarding products, services, or web experiences. The application helps organizations collect valuable insights from users, identify areas of improvement, and enhance the overall user experience. Users can register/login and submit their opinions, suggestions, or issues through a structured feedback form.

The frontend is developed using **HTML, CSS, and JavaScript**, ensuring a smooth, intuitive user interface that adapts well across devices. The design emphasizes clean UI/UX principles, offering a seamless navigation experience for users submitting feedback.

Currently, the application functions as a **frontend-only project**, but it is fully structured for future backend integration using technologies such as **Node.js, Express, MongoDB**, or **Firebase**. **Git** is used for version control and the project is hosted on **GitHub**, supporting collaborative development and modular code management.

This project also establishes a strong foundation for integrating **DevOps practices** in future iterations, including CI/CD pipelines, backend containerization via Docker, and automated deployment. The system encourages reusable components, scalable architecture, and frontend-readiness for full-stack deployment.

1.2 Requirements

To build and design the Feedback Collection platform, the following tools, technologies, and resources were utilized:

GitHub Repository

- The complete source code is stored in a GitHub repository.
- The repository includes:
    - HTML pages (e.g., index.html, feedback.html)
    - CSS files for styling and layout responsiveness
    - JavaScript files handling UI behavior and validation
    - Optional assets like icons and banners

Frontend Development

- Built using:
    - **HTML**: For defining the content structure
    - **CSS**: For applying styling, layouts, and transitions

    - **JavaScript**: For enabling interactive elements (e.g., form submission, modal pop-ups)
- Key UI Components:
    - Feedback form with fields like name, rating, comments
    - User login and registration modals
    - Confirmation popups upon feedback submission
    - Responsive layout with animated sections and transitions

Feedback Data Structure

- Static form submission is used to simulate real user feedback
- Each feedback includes:
    - User name, email (optional), feedback message
    - Rating score or selection dropdown
    - Submission timestamp (manually simulated in frontend)
    - Tags like “Bug”, “Feature Request”, “General Opinion”

Design Elements

- Icons and SVGs used to represent feedback categories (e.g., thumbs up, alert, suggestion bulb)
- Consistent color palette and typography for modern appearance
- Responsive card layouts and modal windows

1.3 Prerequisites

Technical Knowledge

- Basic understanding of **HTML/CSS/JavaScript** for frontend development
- Familiarity with **Git & GitHub** for version control
- Awareness of **UI/UX design** principles, especially for form design and readability

Tooling Setup

- Git installed locally with access to GitHub
- Code editor like **Visual Studio Code**
- Browser (Chrome, Firefox, Edge) for testing the UI
- (Optional) UI design tools like **Figma** or **Canva** for planning interface layout

Application Readiness

- Completed frontend pages: home, feedback form, login, and thank-you screen
- Clean and semantic HTML markup
- CSS structured with media queries for full responsiveness
- JavaScript integrated for handling modals, form submission, and alerts
- Uniform visual design maintained throughout for professional appeal

CHAPTER 2

METHODOLOGY

This project follows a structured and modular web development approach to build the **Feedback Collection Web Application**, a responsive and interactive platform focused on gathering user feedback efficiently. While this version focuses solely on frontend implementation, it is fully designed with future backend and DevOps integrations in mind. The development emphasizes clean UI design, organized code structure, version control, and scalability for full-stack transformation.

Phases of Implementation

1. Code Development and Version Control

- The project is built using **HTML, CSS, and JavaScript**, with modular code separation for clarity and maintainability.
- Web pages include a feedback form, confirmation messages, and login/register modals for user interaction.
- Core JavaScript is used for handling events such as modal display, input validation, and UI transitions.
- **Git** is used for version control to track and manage code changes across development stages.
- A **GitHub repository** is maintained to support team collaboration, code sharing, and future expansion with backend services.

2. Frontend Architecture and Component Design

- The frontend is divided into distinct components and UI sections, including:
    - Feedback Form Section (with input fields, dropdowns, and text areas)
    - Hero Banner (with introductory message and call to action)
    - Success Message Popup (on feedback submission)
- CSS Flexbox and media queries are applied to ensure **full responsiveness** across devices.
- JavaScript handles form submission logic, input field validations, and modal toggling using DOM manipulation.
- Feedback data is currently static and displayed on the page to simulate dynamic behavior. The layout is designed to support future data integration from backend APIs.

3. Backend Integration Readiness

- Though backend functionality is not implemented in this version, the system is architected for future integration with REST APIs.
- All input forms are structured with:

    - Field-level validation
    - UI-ready handlers for POST methods
    - Preparation for success/error state management
- Buttons, modals, and form actions are built with appropriate IDs and event listeners to easily connect with future endpoints like `/api/submit-feedback` or `/api/auth/login`.
- The directory structure and frontend logic are arranged to support technologies like **Node.js**, **Express**, or **Firebase Functions** in subsequent versions.

4. Deployment (Frontend Hosting)

- The application is static and ready to be hosted on platforms like:
    - **GitHub Pages**
    - **Netlify**
    - **Vercel**
- The GitHub repository contains all necessary assets (HTML, CSS, JS, icons) for deployment without additional dependencies.
- CI/CD integration can be applied in future phases to enable automated build and deployment pipelines.
- The lightweight and portable structure ensures quick testing and public hosting for user trials and demonstrations.

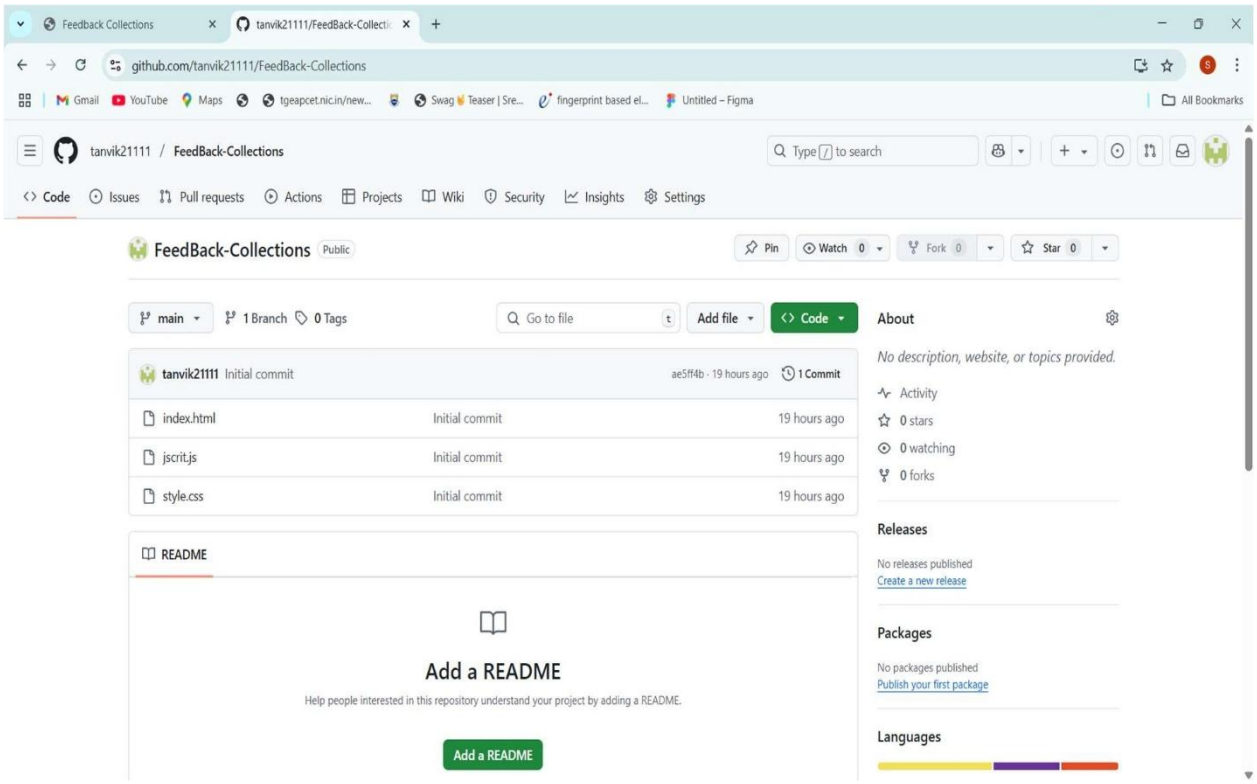
This structured methodology ensures that the Feedback Collection Web Application is not only functional and visually polished in its current state but also future-proof and ready for backend integration, continuous deployment, and DevOps-based scaling. By emphasizing clean architecture, modularity, and maintainability, the platform is well-positioned for real-world use and progressive enhancements.

CHAPTER 3

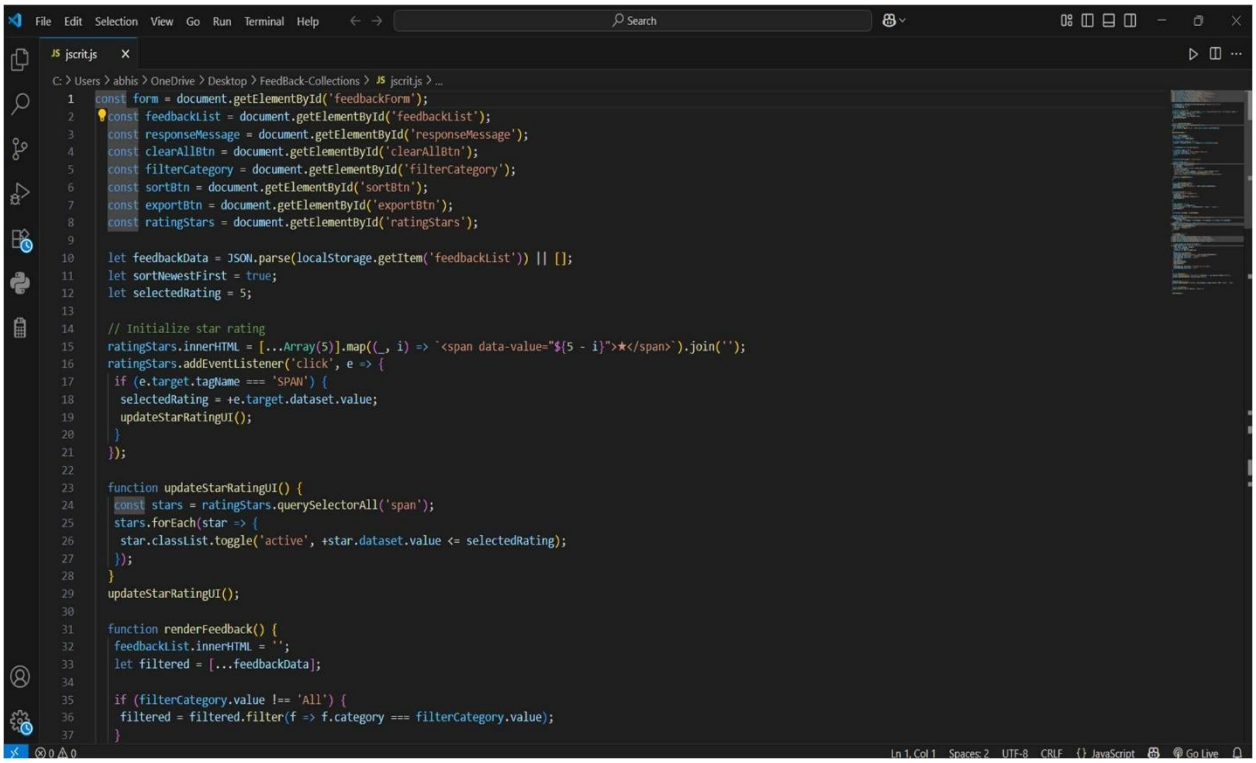
RESULTS AND DISCUSSION

EXECUTION:

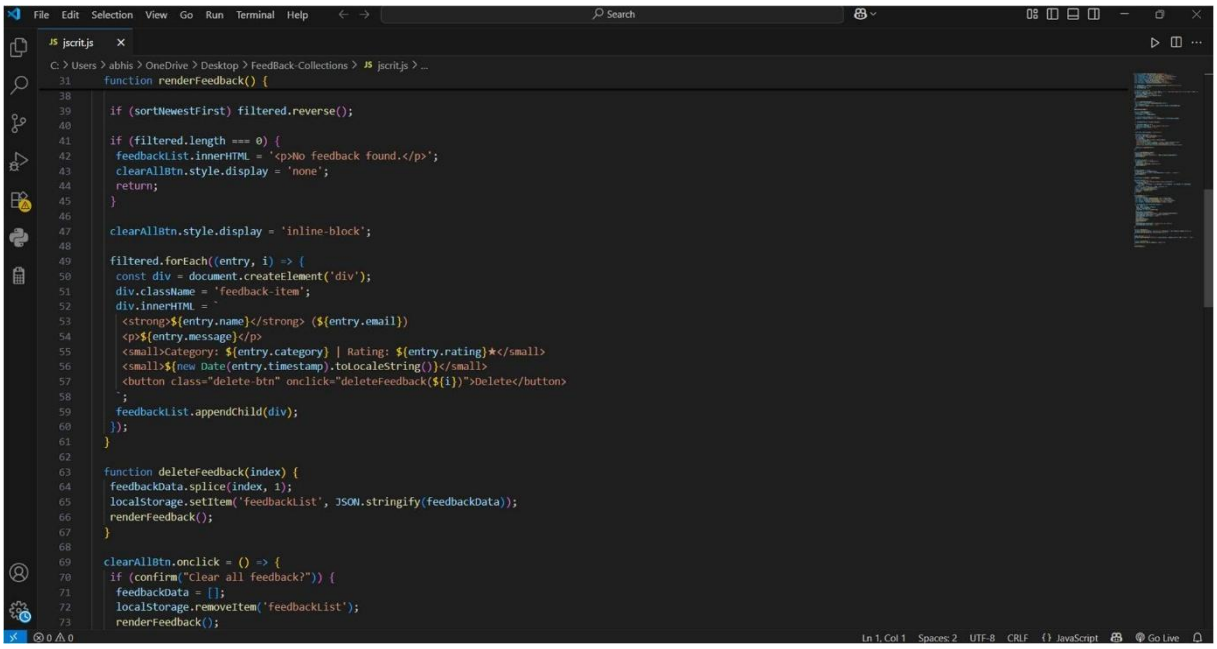
GitHub:



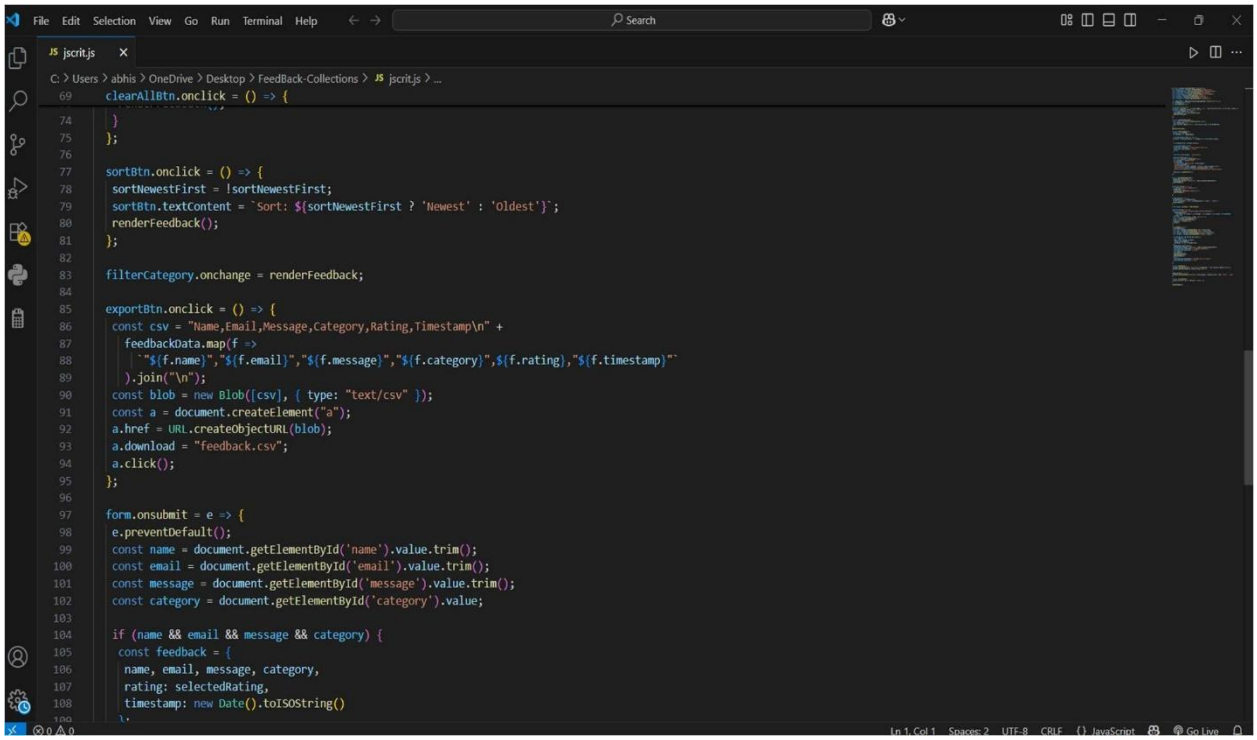
Code:



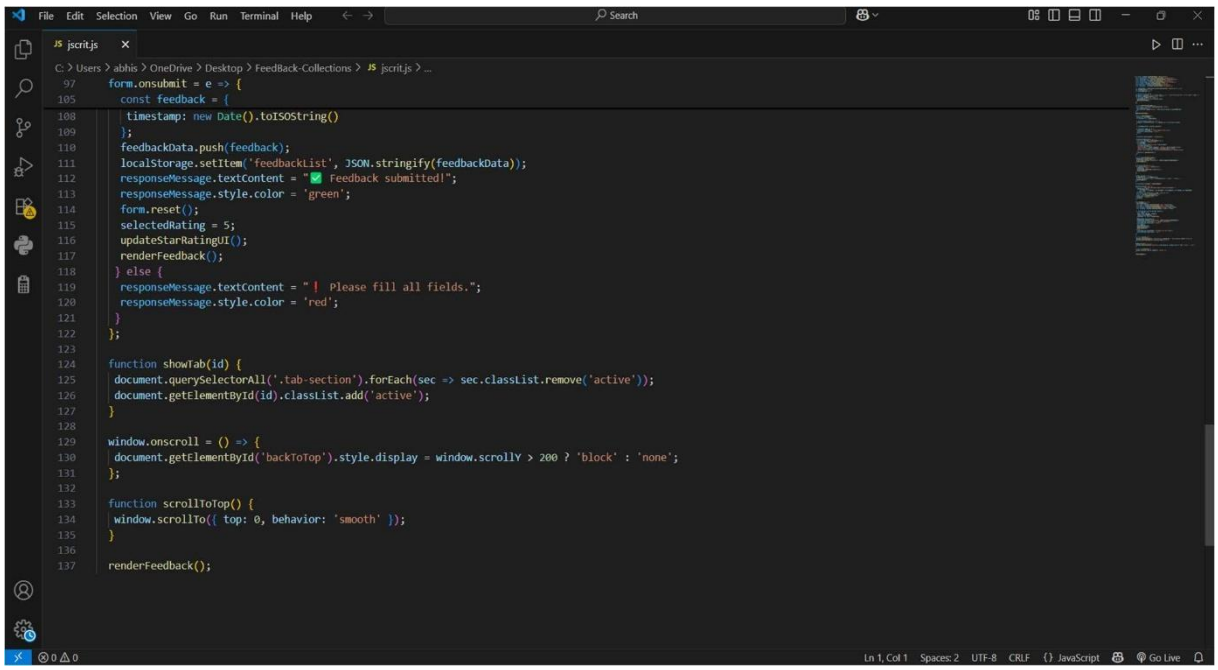
```
const form = document.getElementById('feedbackForm');
const feedbackList = document.getElementById('feedbackList');
const responseMessage = document.getElementById('responseMessage');
const clearAllBtn = document.getElementById('clearAllBtn');
const filterCategory = document.getElementById('filterCategory');
const sortBtn = document.getElementById('sortBtn');
const exportBtn = document.getElementById('exportBtn');
const ratingStars = document.getElementById('ratingStars');

let feedbackData = JSON.parse(localStorage.getItem('feedbackList')) || [];
let sortNewestFirst = true;
let selectedRating = 5;

// Initialize star rating
ratingStars.innerHTML = [...Array(5)].map((_, i) => `<span data-value="${5 - i}">★</span>`).join('');
ratingStars.addEventListener('click', e => {
 if (e.target.tagName === 'SPAN') {
  selectedRating = +e.target.dataset.value;
  updateStarRatingUI();
 }
});

function updateStarRatingUI() {
 const stars = ratingStars.querySelectorAll('span');
 stars.forEach(star => {
  star.classList.toggle('active', +star.dataset.value <= selectedRating);
 });
}
updateStarRatingUI();

function renderFeedback() {
 feedbackList.innerHTML = '';
 let filtered = [...feedbackData];

 if (filterCategory.value !== 'All') {
  filtered = filtered.filter(f => f.category === filterCategory.value);
 }
```



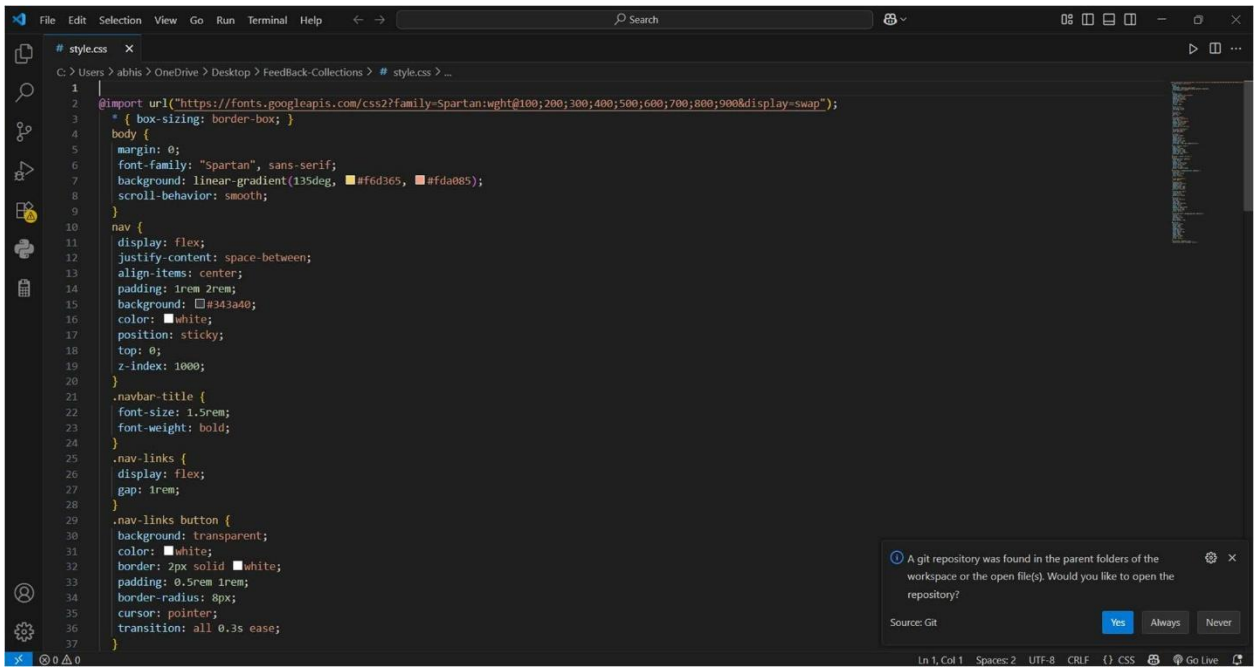
```
function renderFeedback() {

 if (sortNewestFirst) filtered.reverse();

 if (filtered.length === 0) {
  feedbackList.innerHTML = '<p>No feedback found.</p>';
  clearAllBtn.style.display = 'none';
  return;
 }

 clearAllBtn.style.display = 'inline-block';

 filtered.forEach((entry, i) => {
  const div = document.createElement('div');
  div.className = 'feedback-item';
  div.innerHTML = `
   <strong>${entry.name}</strong> (${entry.email})
   <p>${entry.message}</p>
   <small>Category: ${entry.category} | Rating: ${entry.rating}★</small>
   <small>${new Date(entry.timestamp).toLocaleString()}</small>
   <button class="delete-btn" onclick="deleteFeedback(${i})">Delete</button>
  `;
  feedbackList.appendChild(div);
 });
}

function deleteFeedback(index) {
 feedbackData.splice(index, 1);
 localStorage.setItem('feedbackList', JSON.stringify(feedbackData));
 renderFeedback();
}

clearAllBtn.onclick = () => {
 if (confirm("Clear all feedback?")) {
  feedbackData = [];
  localStorage.removeItem('feedbackList');
  renderFeedback();
```



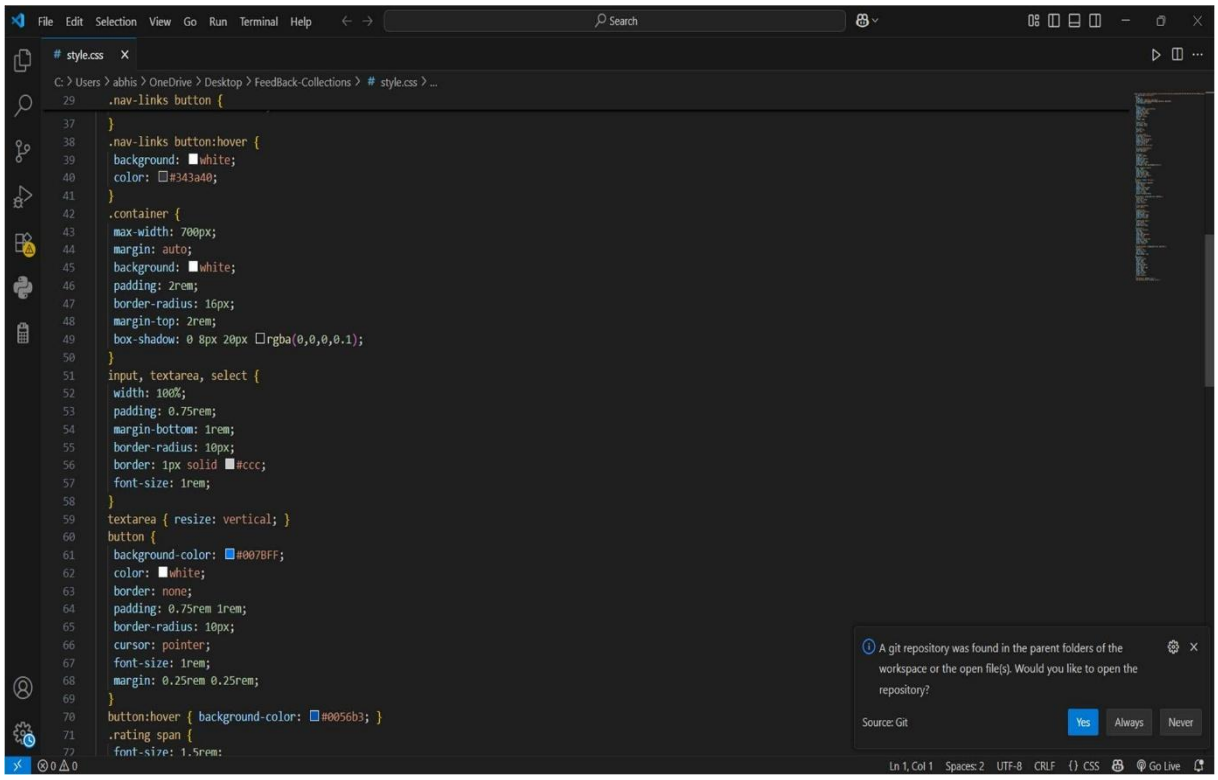
```
clearAllBtn.onclick = () => {
  }
};

sortBtn.onclick = () => {
 sortNewestFirst = !sortNewestFirst;
 sortBtn.textContent = `Sort: ${sortNewestFirst ? 'Newest' : 'Oldest'}`;
 renderFeedback();
};

filterCategory.onchange = renderFeedback;

exportBtn.onclick = () => {
 const csv = "Name,Email,Message,Category,Rating,Timestamp\n" +
   feedbackData.map(f =>
     `"${f.name}","${f.email}","${f.message}","${f.category}",${f.rating},"${f.timestamp}"`
   ).join("\n");
 const blob = new Blob([csv], { type: "text/csv" });
 const a = document.createElement("a");
 a.href = URL.createObjectURL(blob);
 a.download = "feedback.csv";
 a.click();
};

form.onsubmit = e => {
 e.preventDefault();
 const name = document.getElementById('name').value.trim();
 const email = document.getElementById('email').value.trim();
 const message = document.getElementById('message').value.trim();
 const category = document.getElementById('category').value;

 if (name && email && message && category) {
  const feedback = {
   name, email, message, category,
   rating: selectedRating,
   timestamp: new Date().toISOString()
  };
```



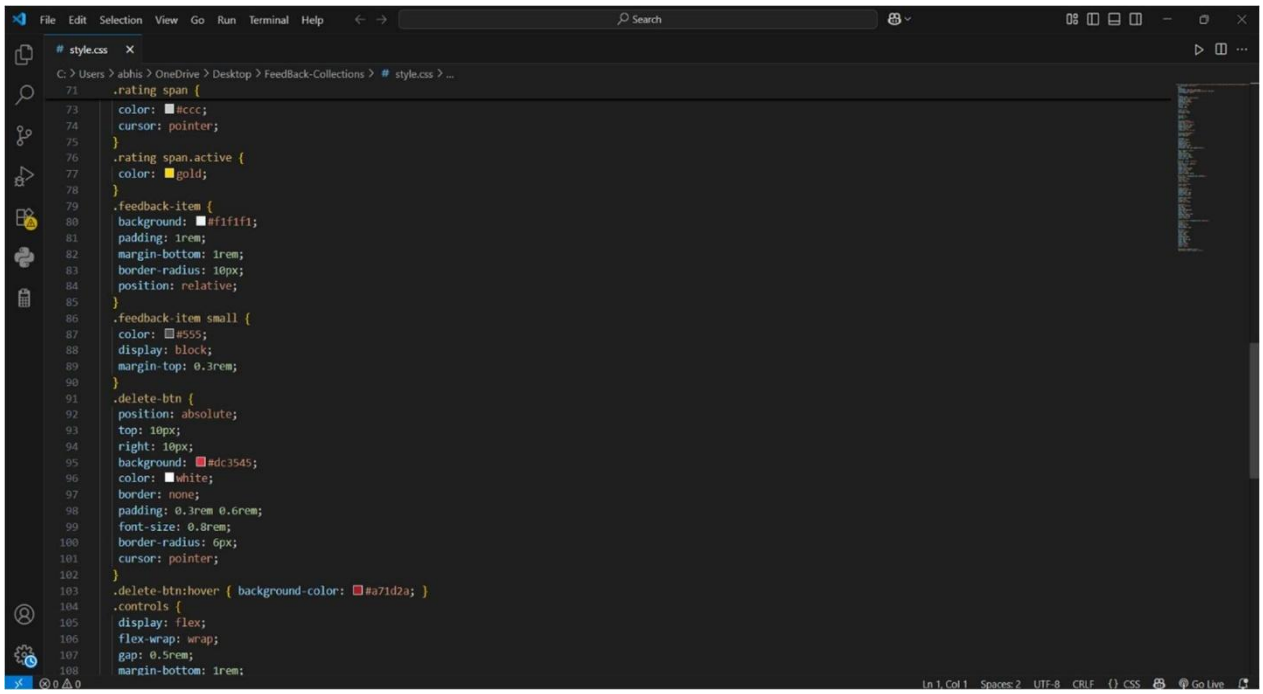
```
form.onsubmit = e => {
  const feedback = {
   timestamp: new Date().toISOString()
  };
  feedbackData.push(feedback);
  localStorage.setItem('feedbackList', JSON.stringify(feedbackData));
  responseMessage.textContent = "✅ Feedback submitted!";
  responseMessage.style.color = 'green';
  form.reset();
  selectedRating = 5;
  updateStarRatingUI();
  renderFeedback();
 } else {
  responseMessage.textContent = "❗ Please fill all fields.";
  responseMessage.style.color = 'red';
 }
};

function showTab(id) {
 document.querySelectorAll('.tab-section').forEach(sec => sec.classList.remove('active'));
 document.getElementById(id).classList.add('active');
}

window.onscroll = () => {
 document.getElementById('backToTop').style.display = window.scrollY > 200 ? 'block' : 'none';
};

function scrollToTop() {
 window.scrollTo({ top: 0, behavior: 'smooth' });
}

renderFeedback();
```



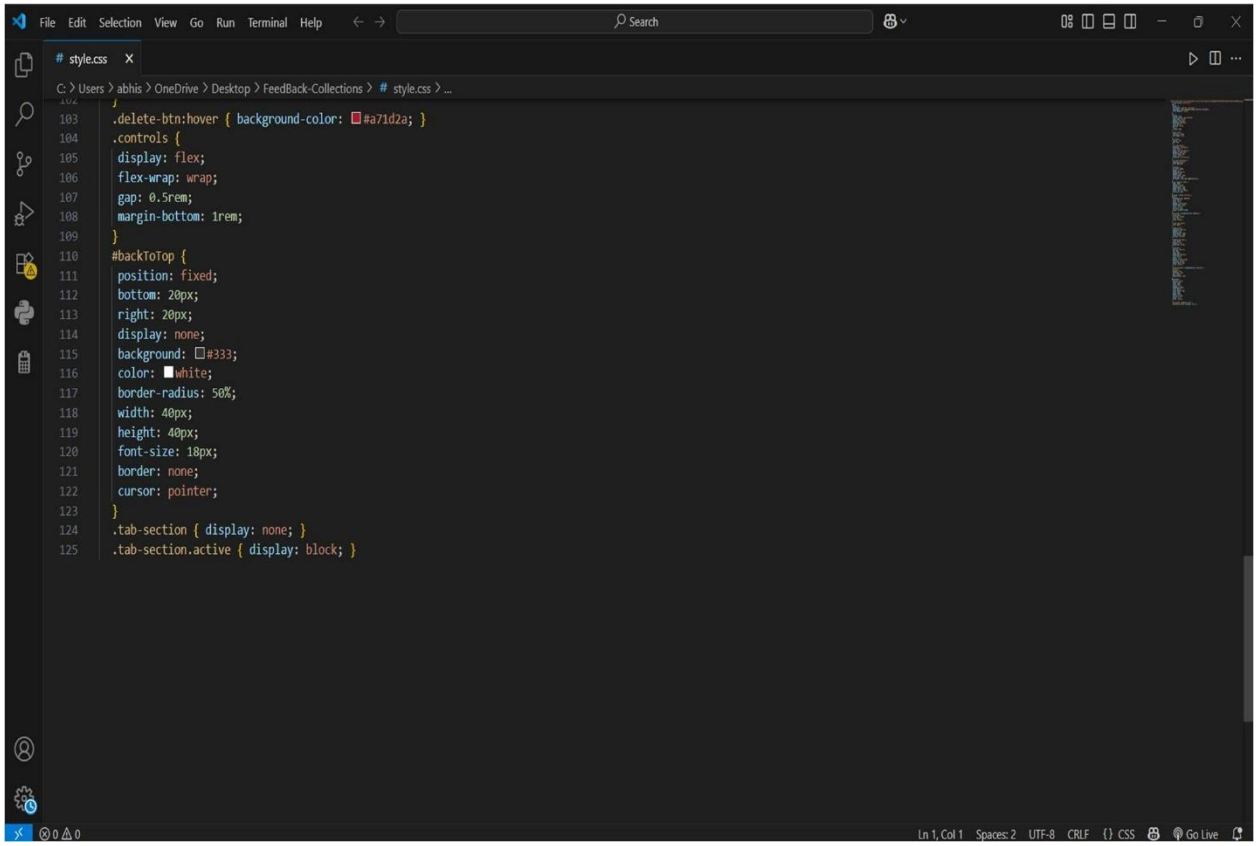
```
@import url("https://fonts.googleapis.com/css2?family=Spartan:wght@100;200;300;400;500;600;700;800;900&display=swap");
* { box-sizing: border-box; }
body {
 margin: 0;
 font-family: "Spartan", sans-serif;
 background: linear-gradient(135deg, #f6d365, #fda085);
 scroll-behavior: smooth;
}
nav {
 display: flex;
 justify-content: space-between;
 align-items: center;
 padding: 1rem 2rem;
 background: #343a40;
 color: white;
 position: sticky;
 top: 0;
 z-index: 1000;
}
.navbar-title {
 font-size: 1.5rem;
 font-weight: bold;
}
.nav-links {
 display: flex;
 gap: 1rem;
}
.nav-links button {
 background: transparent;
 color: white;
 border: 2px solid white;
 padding: 0.5rem 1rem;
 border-radius: 8px;
 cursor: pointer;
 transition: all 0.3s ease;
}
```



```
.nav-links button {
}
.nav-links button:hover {
 background: white;
 color: #343a40;
}
.container {
 max-width: 700px;
 margin: auto;
 background: white;
 padding: 2rem;
 border-radius: 16px;
 margin-top: 2rem;
 box-shadow: 0 8px 20px rgba(0,0,0,0.1);
}
input, textarea, select {
 width: 100%;
 padding: 0.75rem;
 margin-bottom: 1rem;
 border-radius: 10px;
 border: 1px solid #ccc;
 font-size: 1rem;
}
textarea { resize: vertical; }
button {
 background-color: #007BFF;
 color: white;
 border: none;
 padding: 0.75rem 1rem;
 border-radius: 10px;
 cursor: pointer;
 font-size: 1rem;
 margin: 0.25rem 0.25rem;
}
button:hover { background-color: #0056b3; }
.rating span {
 font-size: 1.5rem;
```

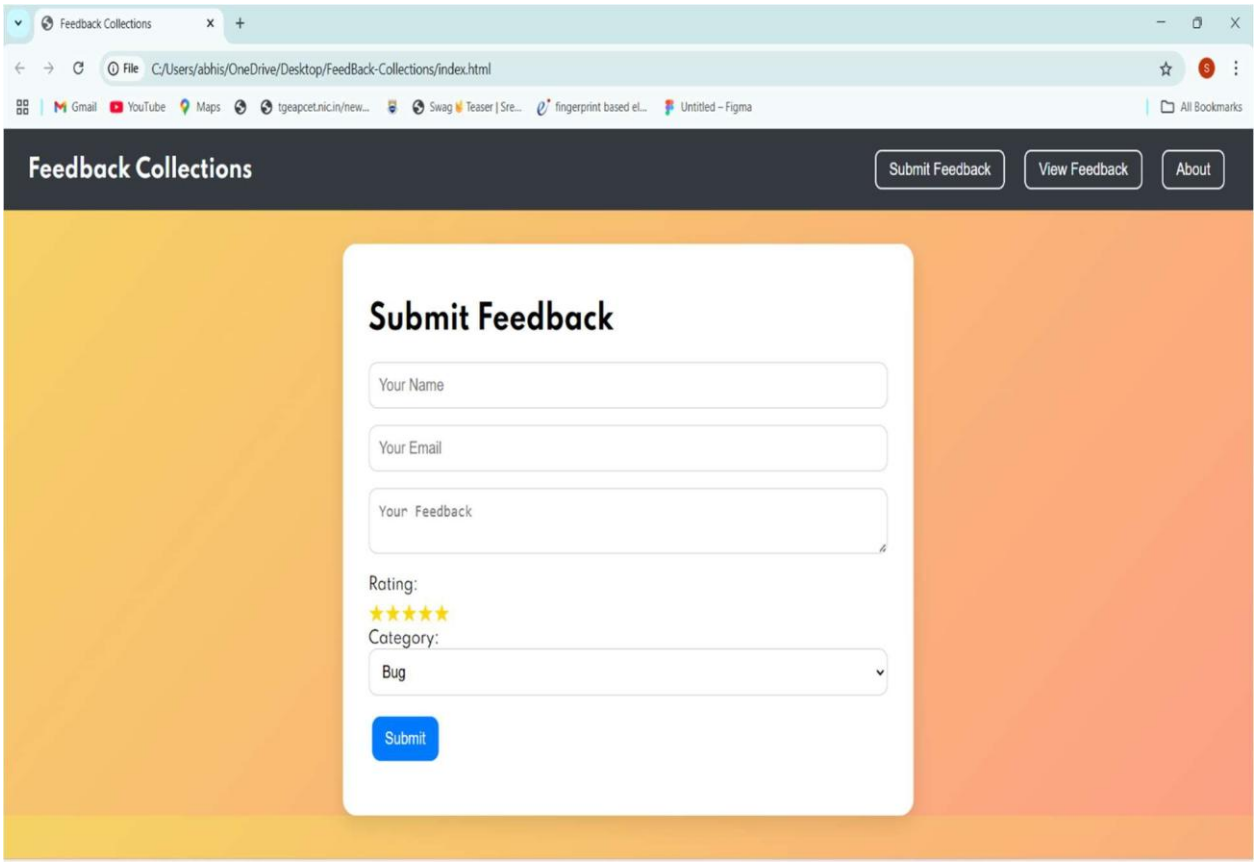


```
.rating span {
 color: #ccc;
 cursor: pointer;
}
.rating span.active {
 color: gold;
}
.feedback-item {
 background: #f1f1f1;
 padding: 1rem;
 margin-bottom: 1rem;
 border-radius: 10px;
 position: relative;
}
.feedback-item small {
 color: #555;
 display: block;
 margin-top: 0.3rem;
}
.delete-btn {
 position: absolute;
 top: 10px;
 right: 10px;
 background: #dc3545;
 color: white;
 border: none;
 padding: 0.3rem 0.6rem;
 font-size: 0.8rem;
 border-radius: 6px;
 cursor: pointer;
}
.delete-btn:hover { background-color: #a71d2a; }
.controls {
 display: flex;
 flex-wrap: wrap;
 gap: 0.5rem;
 margin-bottom: 1rem;
```



```
.delete-btn:hover { background-color: #a71d2a; }
.controls {
 display: flex;
 flex-wrap: wrap;
 gap: 0.5rem;
 margin-bottom: 1rem;
}
#backToTop {
 position: fixed;
 bottom: 20px;
 right: 20px;
 display: none;
 background: #333;
 color: white;
 border-radius: 50%;
 width: 40px;
 height: 40px;
 font-size: 18px;
 border: none;
 cursor: pointer;
}
.tab-section { display: none; }
.tab-section.active { display: block; }
```

Output:


Feedback Collections
File C:/Users/abhis/OneDrive/Desktop/FeedBack-Collections/index.html
Gmail YouTube Maps tgeapcet.nic.in/new... Swag Teaser | Sre... fingerprint based el... Untitled – Figma All Bookmarks
Feedback Collections
Submit Feedback
View Feedback
About
Submit Feedback
Your Name
Your Email
Your Feedback
Rating:
Category:
Bug
Submit

CHAPTER 4

CONCLUSION

This project demonstrates how a well-structured and responsive frontend can form a solid foundation for a modern **Feedback Collection Web Application**. Through clean user interface design, modular frontend coding, and a focus on scalability, the application effectively addresses the primary goal of collecting user feedback in a seamless and user-friendly manner.

Built using **HTML, CSS, and JavaScript**, the application ensures a responsive design and intuitive user experience across devices. Version control is managed using **Git**, with a remote repository hosted on **GitHub**, enabling maintainability, code traceability, and future team collaboration. The use of modular code components and clear separation of concerns provides a flexible structure that is easy to extend and maintain.

Although backend connectivity and data persistence have not been implemented in the current version, the application is fully prepared for future full-stack transformation. The feedback form and authentication modals are designed with structured input handling and event-ready form components, allowing for straightforward integration with RESTful APIs and backend technologies such as **Node.js**, **Express**, and **MongoDB** or **Firebase**.

The project is also aligned with **DevOps principles**, with a focus on modularity, portability, and deployment readiness. The frontend can be hosted on platforms such as **GitHub Pages**, **Netlify**, or **Vercel**, with minimal setup. Additionally, the codebase is structured to support future enhancements like **CI/CD pipelines**, **Docker-based containerization**, and **cloud-based deployment workflows**.

In terms of development principles, this implementation highlights scalability, clarity, and extensibility. The modular and semantic structure ensures that new features—such as feedback analytics dashboards, user authentication systems, and real-time feedback syncing—can be incorporated with minimal restructuring.