**Smart Feedback Collection and Analysis System**

|  |  |
| --- | --- |
| Industry Project Title | Smart Feedback Collection and Analysis System |
| Name of the Company | Tata Consultancy Services (TCS) |
| Name of the Institute | JNTUH University College of Engineering Manthani |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Start Date | End Date | Total Effort (hrs.) | Project Environment | Tools used |
| 10th October 2025 | 5th November 2025 | 90 hours | Node.js, HTML, CSS, JavaScript, SQLite3, Sentiment Analysis API, Chart.js | Visual Studio Code, GitHub, Browser Developer Tools |

**TABLE OF CONTENT**

* Acknowledgements
* Objective and Scope
* Problem Statement
* Existing Approaches
* Approach / Methodology - Tools and Technologies used
* Workflow
* Assumptions
* Implementation - Data collection, Processing Steps, Diagrams - Charts, Table
* Solution Design
* Challenges & Opportunities
* Reflections on the project
* Recommendations
* Outcome / Conclusion
* Enhancement Scope
* Link to code and executable file
* Research questions and responses
* References

**ACKNOWLEDGEMENTS**

I sincerely express my gratitude to **Tata Consultancy Services (TCS)** for providing this valuable opportunity to work on the Smart Feedback Collection and Analysis System project.  
I would also like to thank **JNTUH University College of Engineering Manthani** for their support and guidance throughout this project.  
This experience allowed me to gain hands-on exposure to full-stack web development, sentiment analysis, and real-time data visualization.

**OBJECTIVE AND SCOPE**

**OBJECTIVE**  
To design and implement a web-based Smart Feedback Collection and Analysis System that not only enables users to provide feedback online but also ensures robust data integrity, multi-role access, and seamless visualization of results for data-driven decision-making. The system is built with SQLite for strong data consistency and is architected for future analytical.

**SCOPE**

1. Collect real-time feedback through a user-friendly web interface.
2. Automatically classify feedback as **Positive**, **Negative**, or **Neutral** using NLP-based sentiment analysis.
3. Display analytics and trends in a **dashboard with charts**.
4. Support multiple user roles (Guest, Registered User, Admin).
5. Store and manage feedback efficiently using SQLite3.
6. Ensure scalability and extensibility for future enhancements.

**PROBLEM STATEMENT**

Organizations often face challenges in analyzing user feedback effectively. Manual methods are tedious, inconsistent, and lack real-time insights.  
 Existing feedback collection systems do not leverage machine learning or sentiment analysis to understand user emotions or satisfaction levels.  
 Hence, there is a need for an **intelligent and automated platform** that can collect, analyze, and visualize feedback efficiently.

**EXISTING APPROACHES**

**Traditional feedback collection systems often rely on manual reviews or basic online forms:**

* **No Automated Sentiment Analysis: Manual checks make them slow and error-prone.**
* **Lack of Real-Time Dashboards: Results are scattered and not visualized for actionable insights.**
* **Poor Backend Processing: Many systems have flat or rigid architectures, lacking proper data validation or integrity controls.**
* **Limited Role Management: Most tools support a single admin or basic user authentication without role-based access.**
* **While enterprise tools exist, they are often expensive and do not offer educational or event-focused customization.**

**APPROACH / METHODOLOGY - TOOLS AND TECHNOLOGIES USED**

**TOOLS AND TECHNOLOGIES**

* Frontend:HTML5, CSS3, JavaScript, Chart.js
* Backend: Node.js, Express.js
* Database: SQLite3
* Libraries: Sentiment Analysis API (NPM), CORS, JWT Authentication
* Development Tools: VS Code, Postman, GitHubb
* Testing Tools: Postman, MySQL Workbench
* Operating System: Windows 11

### METHODOLOGY

The project follows an **Agile iterative model**, involving:

* Database schema design.
* Backend API creation.
* Frontend integration with AJAX calls.
* Sentiment analysis integration.
* Dashboard and visualization development.
* Testing and deployment simulation.

**WORKFLOW**

[User <-> React Frontend]

|

(API)

|

[Node.js + Express Backend]

|

(SQLite3 Database <-> Sentiment Analysis API)

### ****Workflow Steps****

1. **User submits feedback** through the React frontend.
2. **Frontend** sends the feedback data to the backend API (Node.js/Express.js) via a POST request.
3. **Backend** receives the feedback on a designated endpoint and validates the input.
4. **Sentiment Analysis API** (NPM package, e.g. sentiment or similar) is called within the backend:
   * The feedback text is passed to the sentiment analysis function.
   * The API processes and returns a sentiment score or label (Positive, Negative, Neutral).
5. **Backend** attaches the sentiment result to the feedback entry.
6. **Feedback data** (including sentiment label/score) is saved into the SQLite3 database.
7. **Admin or analytics dashboard** fetches and visualizes sentiment-tagged feedback from the database, powering trends and sentiment distribution charts.

**ASSUMPTIONS**

* **Users submit clear, meaningful feedback in English.**
* **Database (SQLite3) is always connected for all transactions.**
* **APIs and network requests work reliably, with minimal errors.**
* **Only authorized users (via authentication/role) can access admin dashboards.**
* **Sentiment analysis handles English input accurately.**
* **The system is used on Windows 11 in a stable internet environment.**

**IMPLEMENTATION - DATA COLLECTION, PROCESSING STEPS, DIAGRAMS - CHARTS, TABLE**

* DATA COLLECTION
* Users (guests or registered) submit feedback through a responsive React form on the frontend. Form input is validated locally for required fields and basic constraints (e.g., minimum length).

### ****Processing Steps:****

* **Receive user input:**  
  User submits feedback; frontend sends the data in JSON format via an API request to the backend.
* **Clean and preprocess text:**  
  Backend sanitizes/cleans the feedback text (removes special symbols, trims whitespace, escapes dangerous characters).
* **Run text through Sentiment API:**  
  Backend processes the cleaned text using the Sentiment Analysis NPM package.
* **Assign polarity:**  
  Feedback is classified as Positive, Negative, or Neutral (polarity label and/or score).
* **Store in SQLite3:**  
  Classified feedback, sentiment result, timestamp, and user ID (if registered) are stored in the SQLite3 database.

**VISUALIZATION**

Admin dashboard built using **Chart.js** includes:

**Pie Chart** – Sentiment Breakdown

**Bar Chart** – Feedback count by Type

**Line chart** – Average sentiment score over time

**Table** – All Feedbacks

**SOLUTION DESIGN**

Folder Structure

feedback-app/

├── backend/

│ ├── server.js

│ ├── feedback.db

│ └── package.json

├── frontend/

│ ├── src/

│ │ ├── pages/

│ │ │ ├── GuestPage.js

│ │ │ ├── UserPage.js

│ │ │ └── AdminPage.js

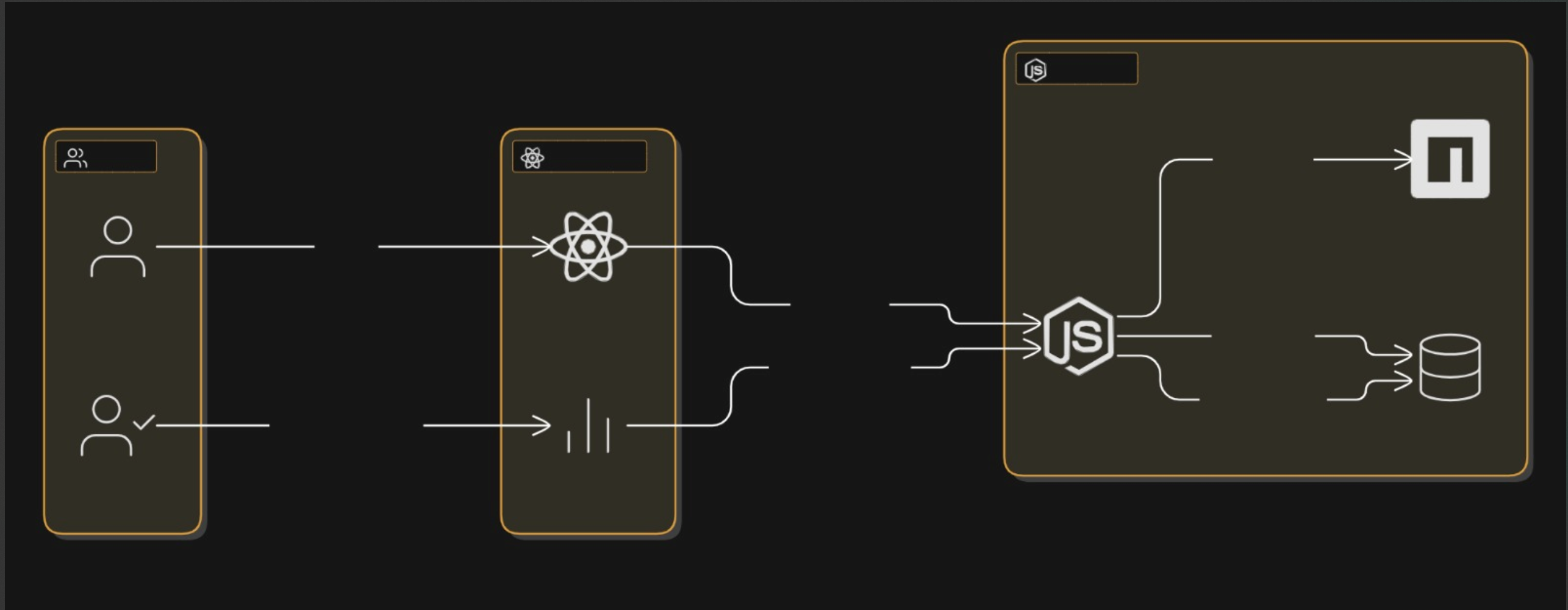
│ │ ├── App.js

│ ├── public/

│ └── package.json

└── README.md

**WEBAPP FLOW**

****

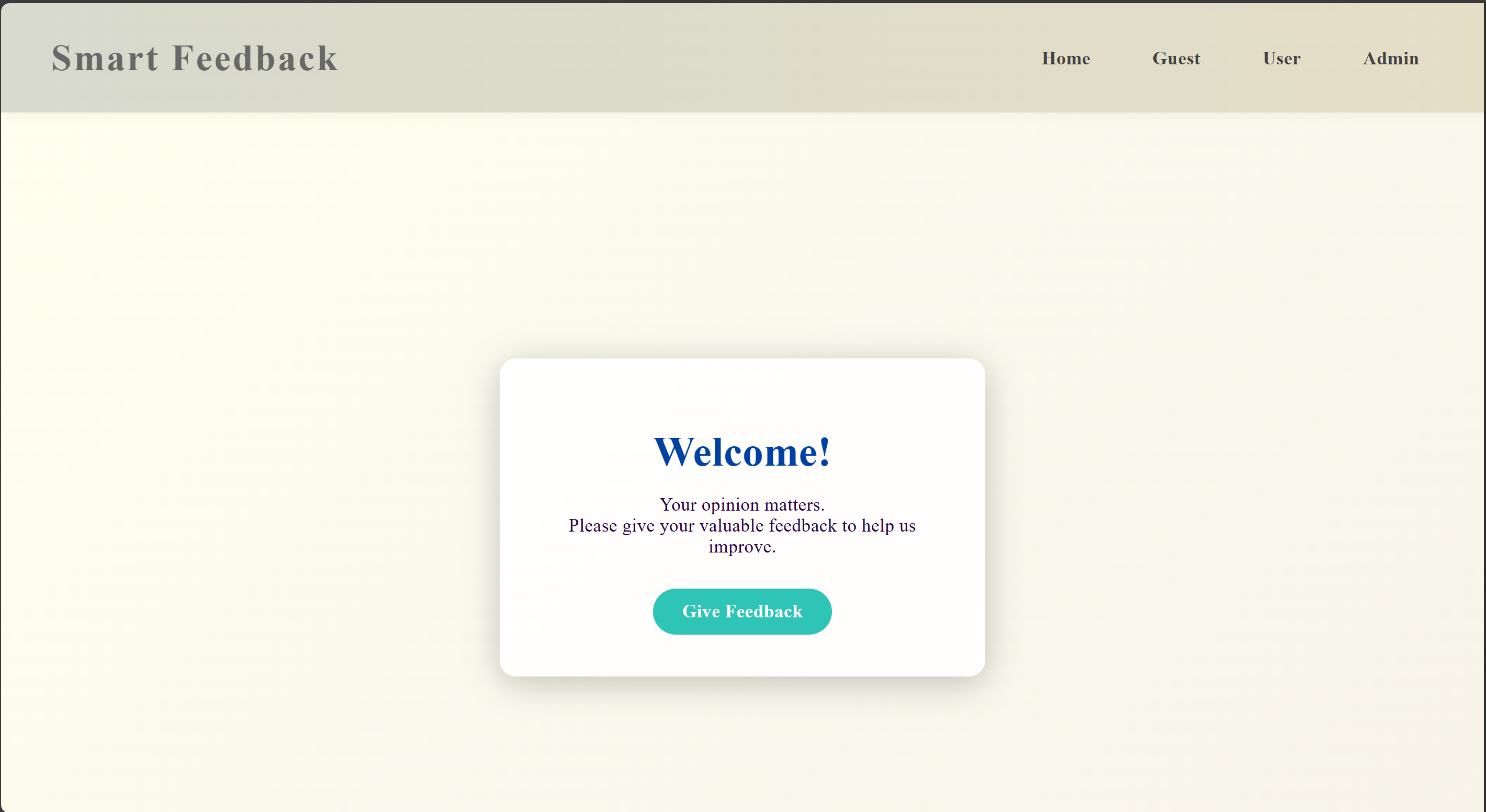
### ****Database Schema (MongoDB)****

**users:** { name, email, password, role }

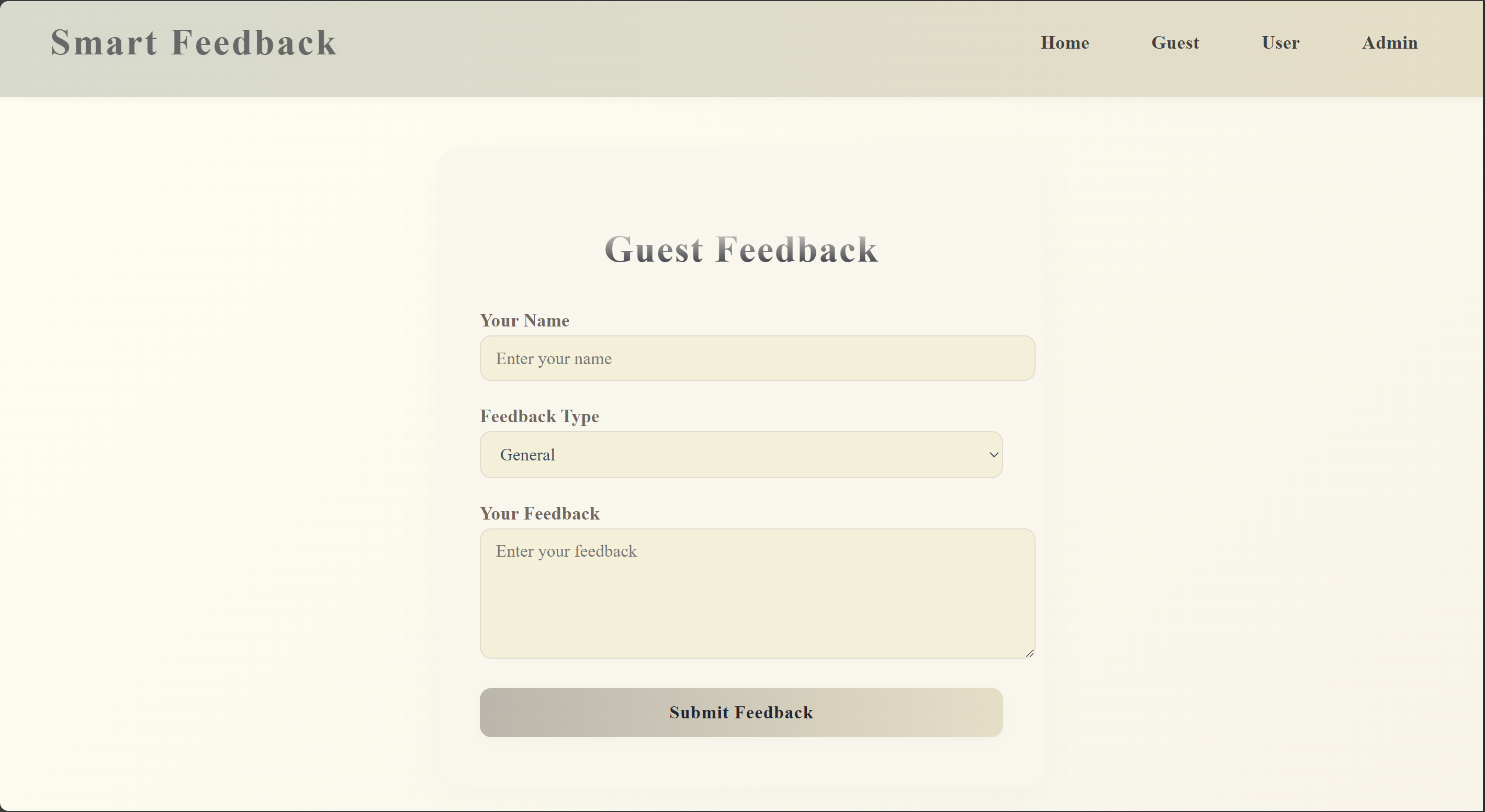
**feedback:** { feedbackText, sentimentLabel, sentimentScore, createdAt }

**WIREFRAMES (TEXT-BASED)**

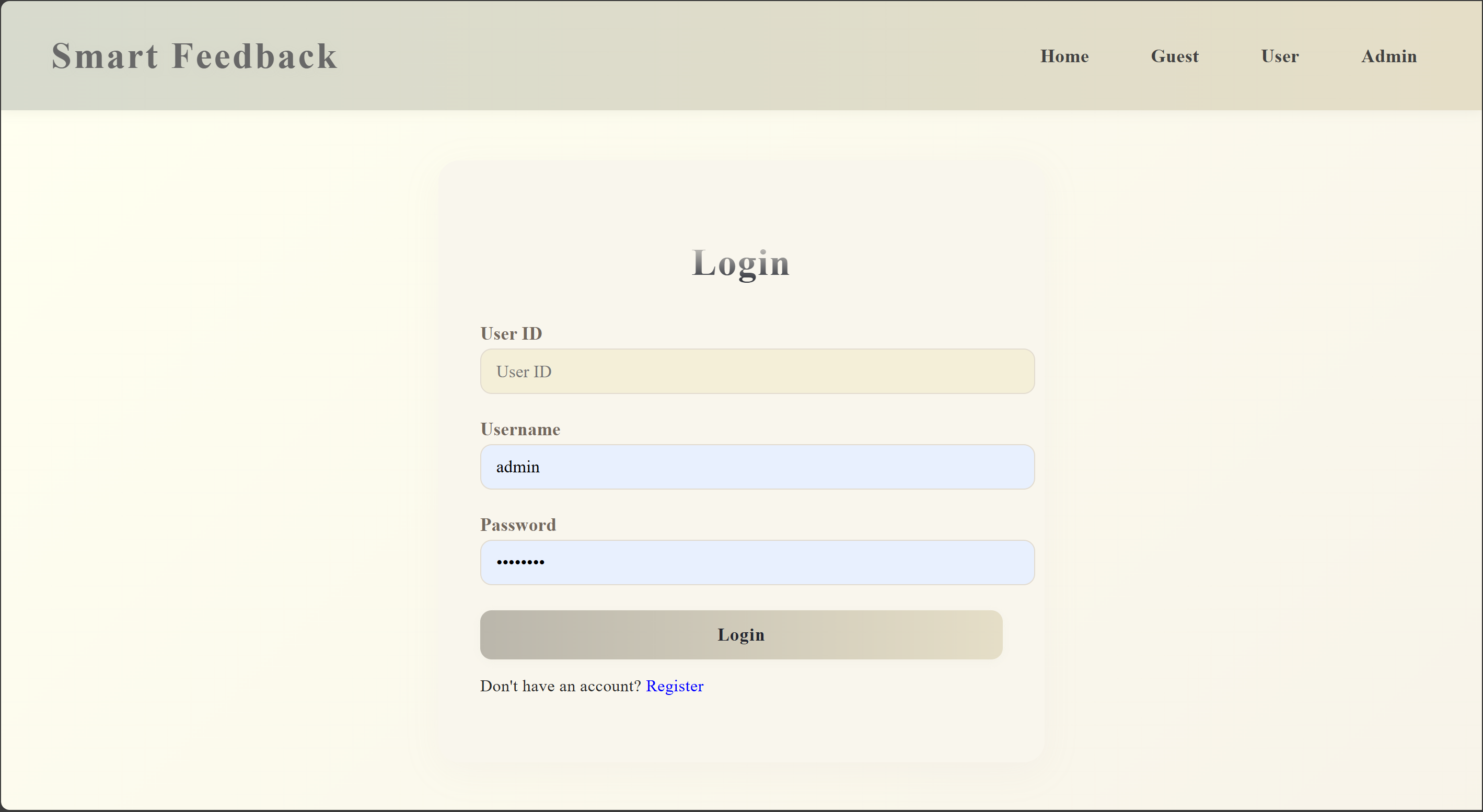
**Home Page**

****

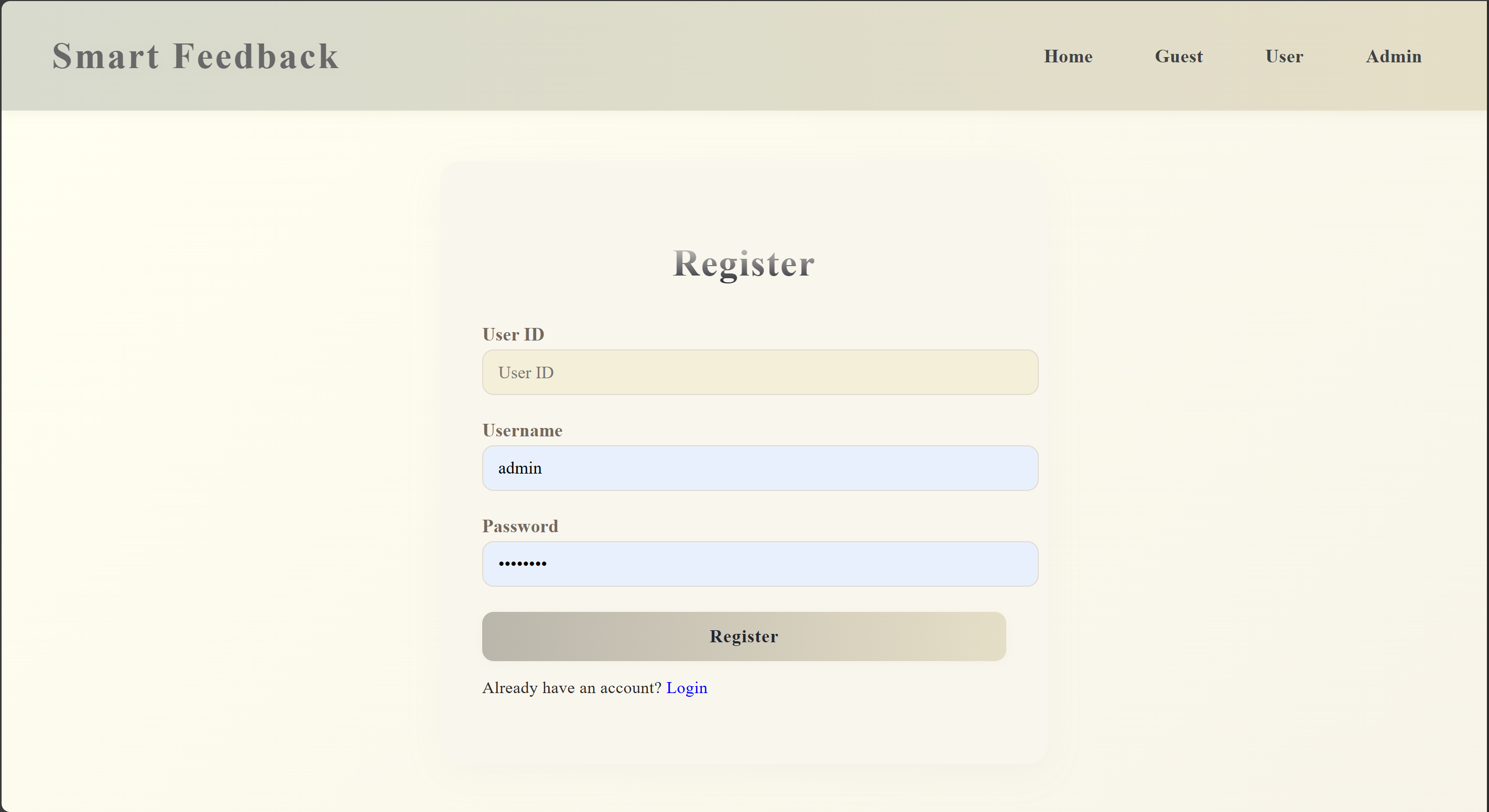
**Guest Login Page**

****

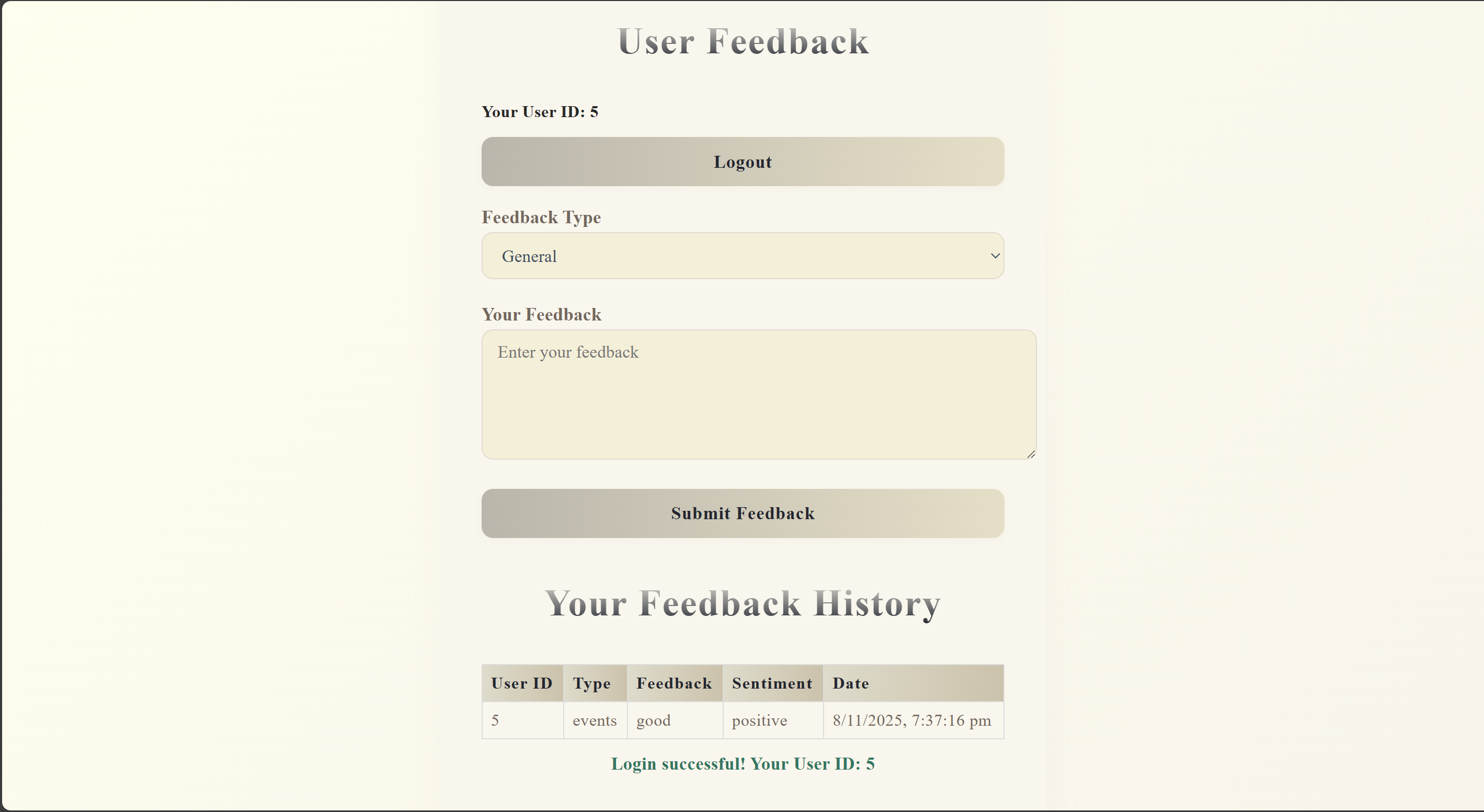
**User Login page**

****

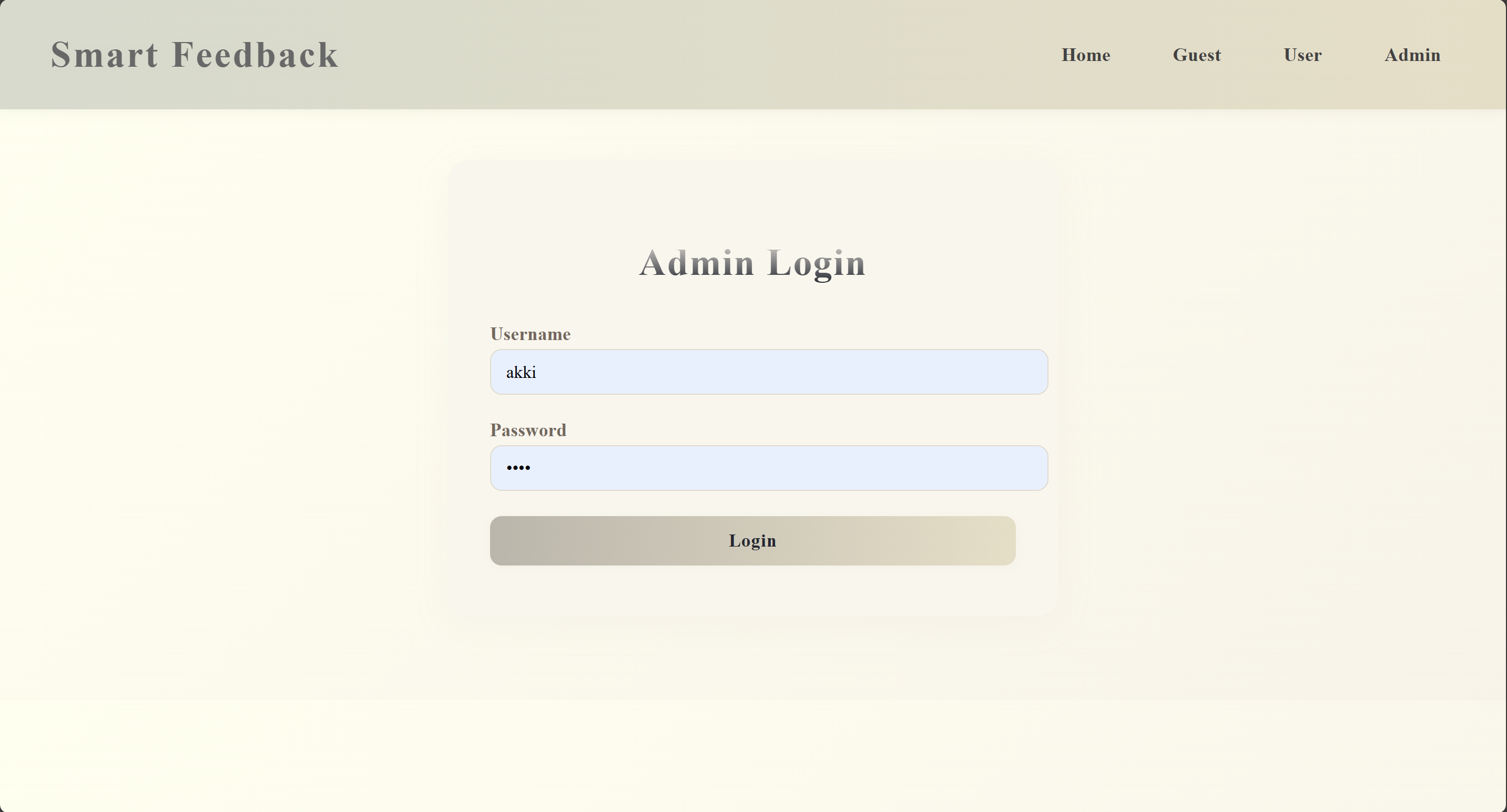
**User Registration page**

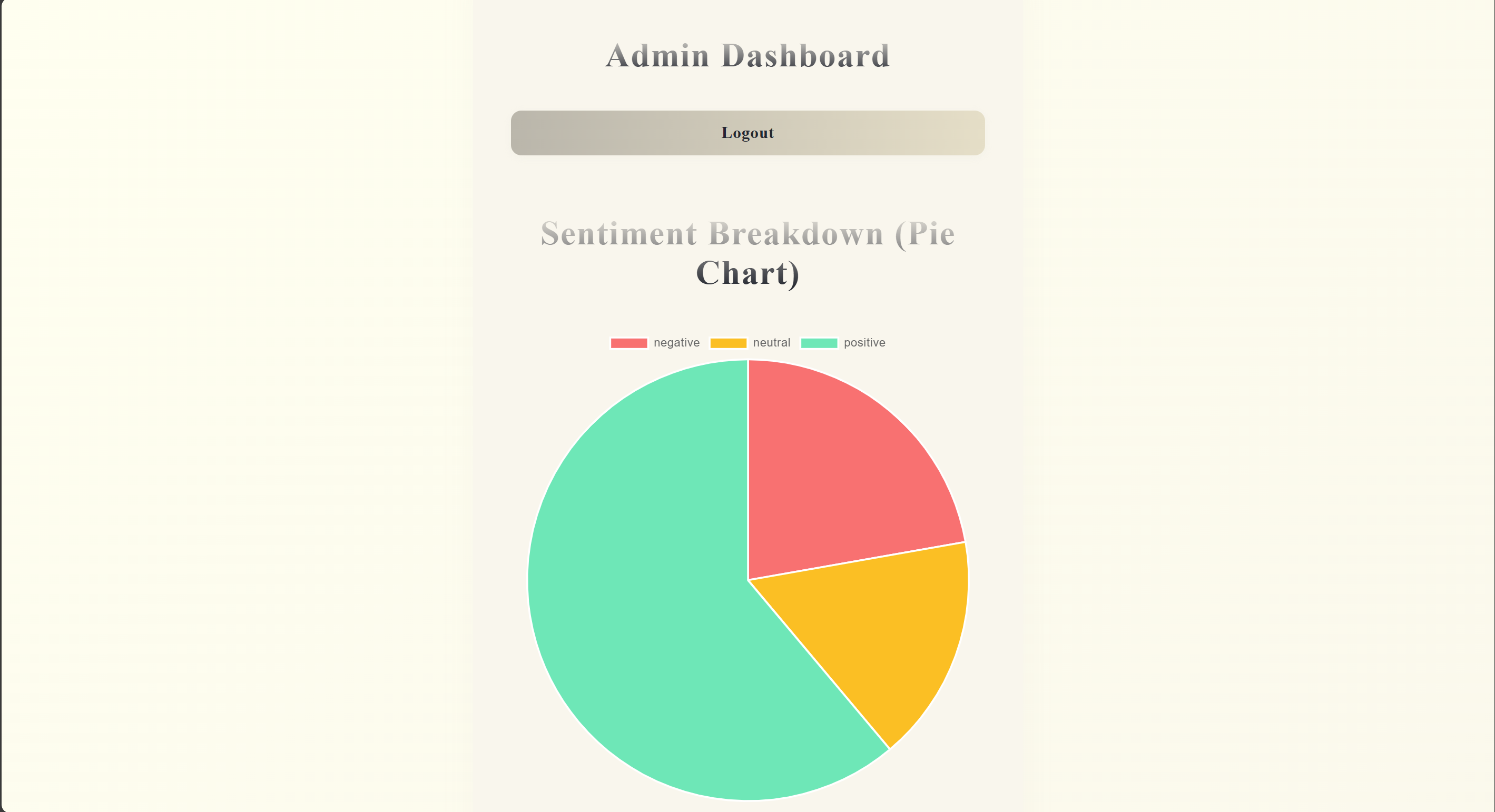
****

**User Feedback form**

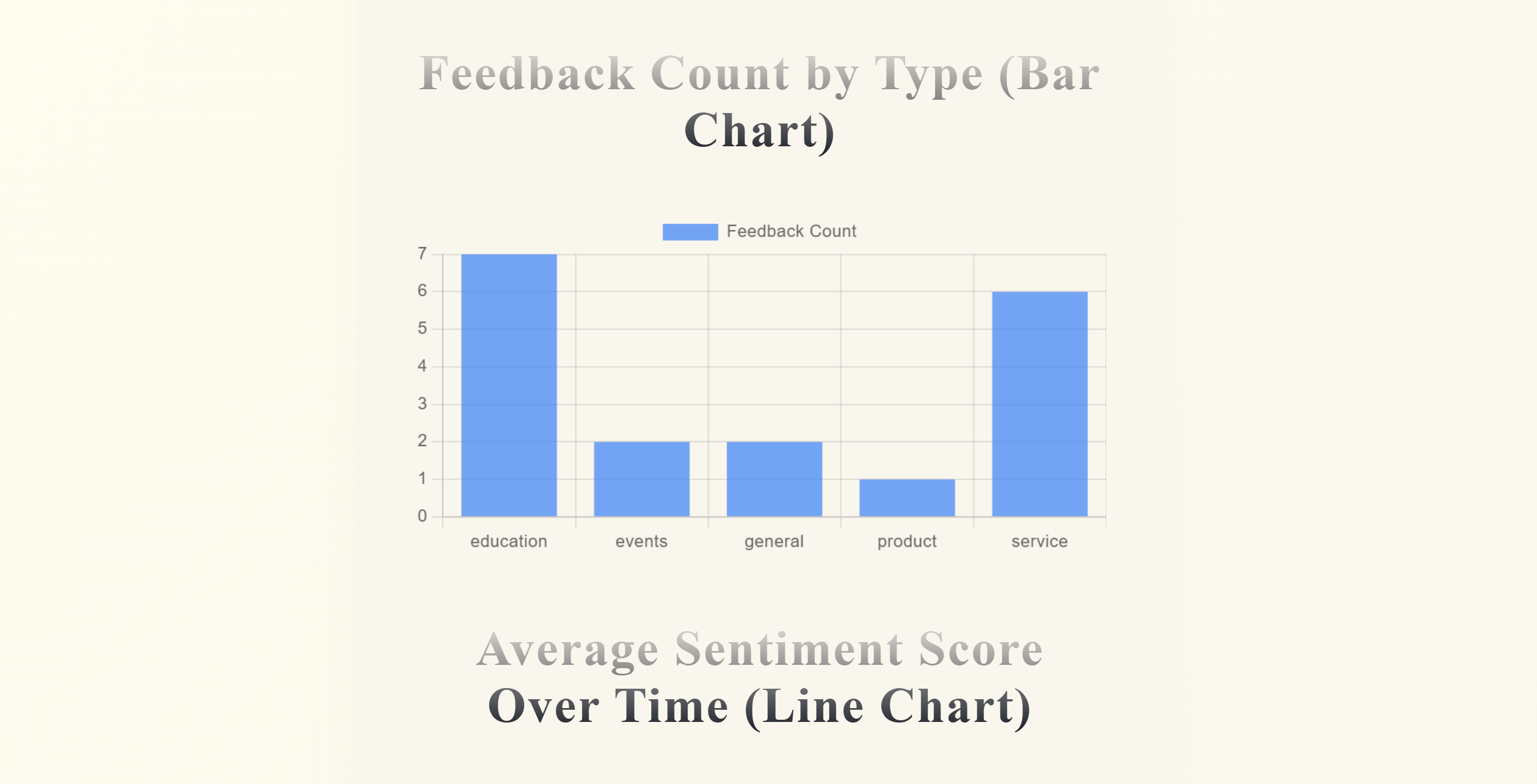


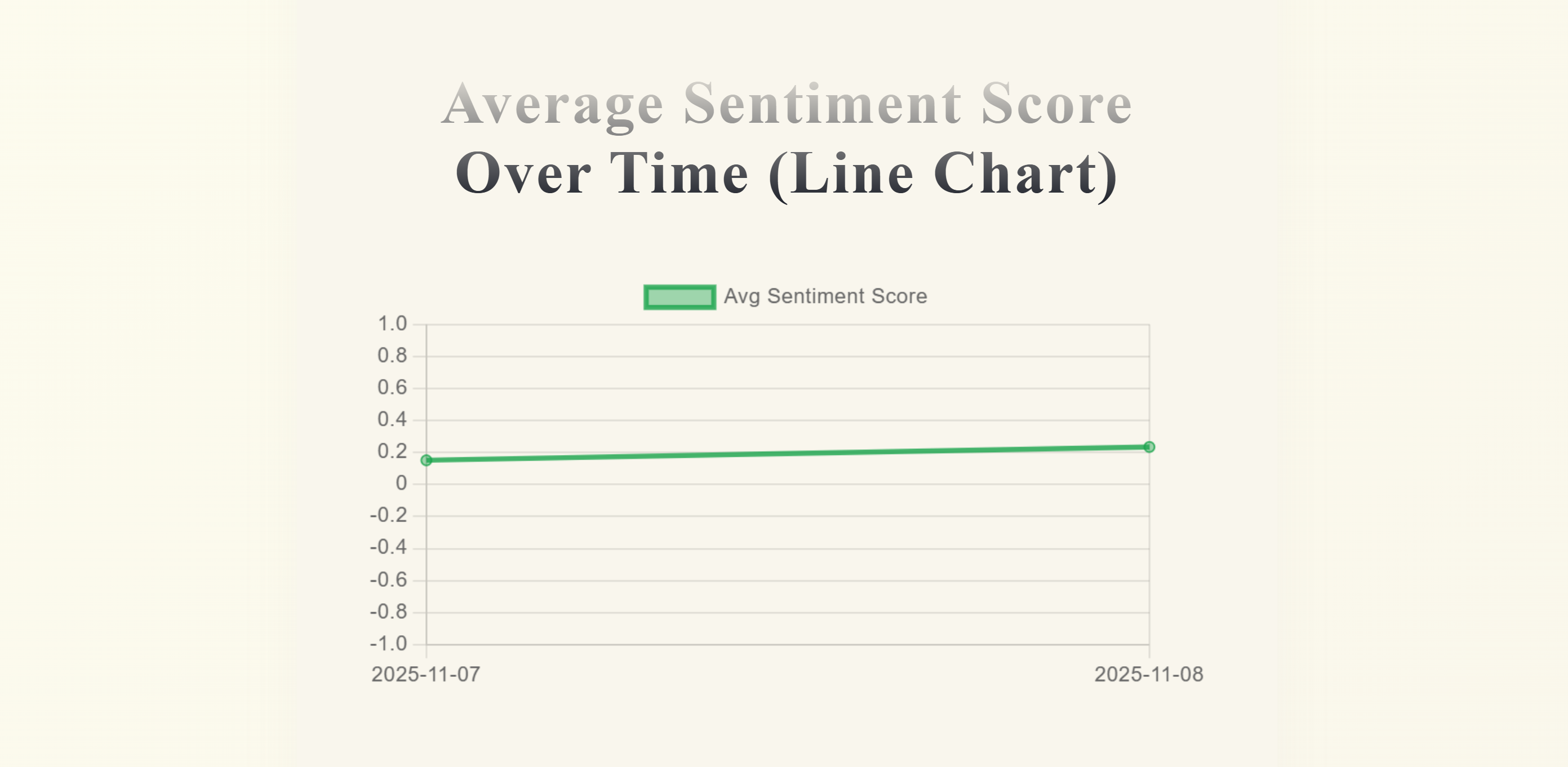
**Admin Login**



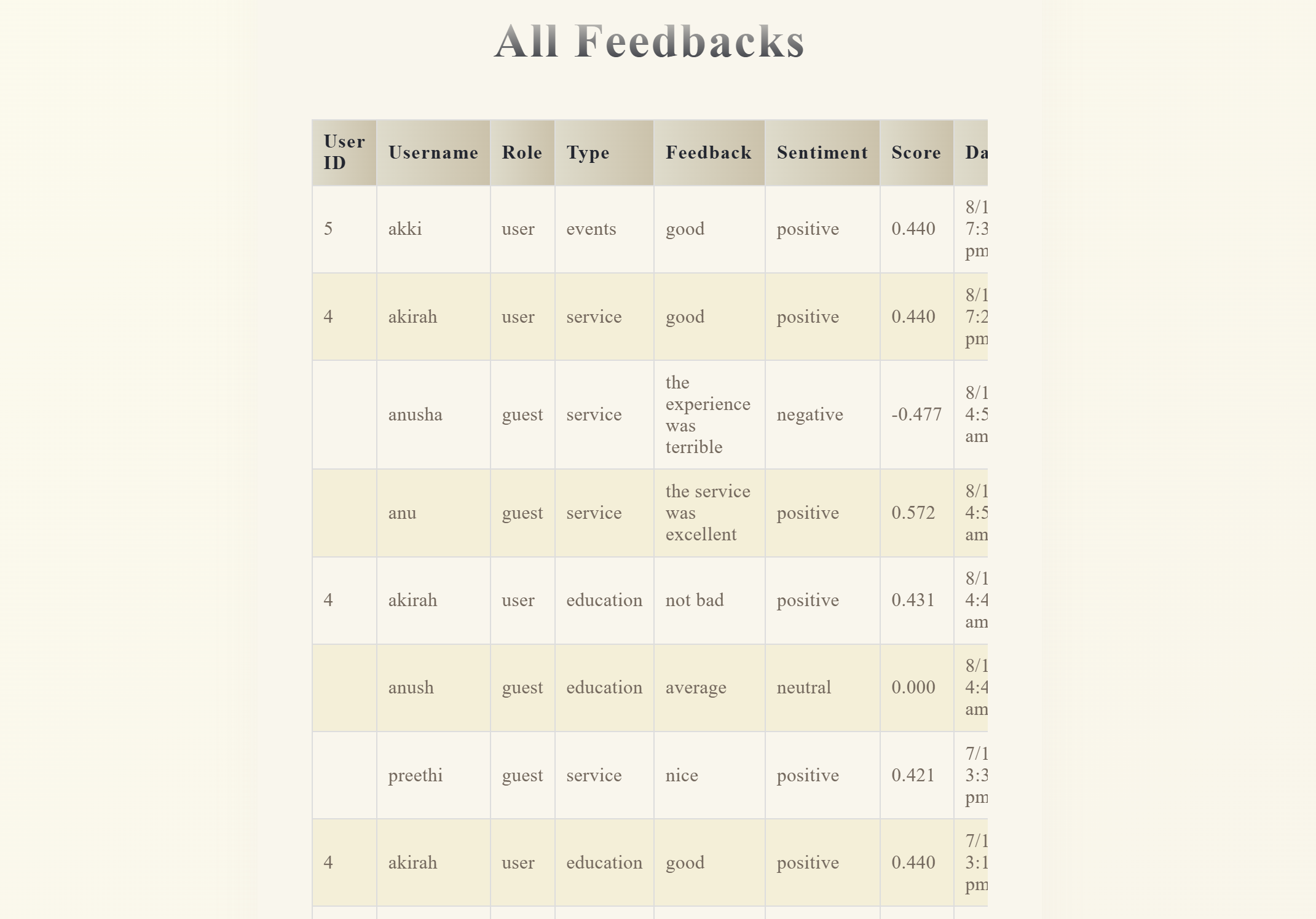
Admin Dashboard visualization–pie chart

**Visualization – Bar chart**

****

Visualization – Line charts

Visualization of feedback history

****

**CHALLENGES & OPPORTUNITIES**

**CHALLENGES**

* Ensuring API response speed under load.
* Managing asynchronous calls in JavaScript effectively.
* Maintaining high accuracy in sentiment classification.

**OPPORTUNITIES**

* Extendable to other industries for survey management.
* Can integrate AI-based models for deeper analysis.
* Deployable on cloud for scalable enterprise usage.

**REFLECTIONS ON THE PROJECT**

This project helped strengthen my understanding of full-stack development, backend APIs, and real-time data visualization.  
Working with Node.js, Express, and SQLite improved my ability to design scalable systems and understand data flow between components.

**RECOMMENDATIONS**

* Add multilingual feedback analysis using translation APIs.
* Include role-based dashboards and data export options.
* Integrate deep learning models for improved sentiment accuracy.
* Deploy the system on cloud platforms like AWS or Render.

**OUTCOME / CONCLUSION**

The **Smart Feedback Collection and Analysis System** automates user feedback management efficiently.  
It performs sentiment classification using NLP, visualizes data in real time, and improves decision-making for event and organizational analysis.

**ENHANCEMENT SCOPE**

* Integrate live chat and real-time sentiment monitoring.
* Add PDF/Excel report generation.
* Create a mobile version for quick feedback submissions.
* Implement email notifications for new feedback entries.

**LINK TO CODE AND EXECUTABLE FILE**

The complete source code and setup guide are available in the attached project archive:

<https://github.com/chadalavadaharika/smart_feedback.git>

**To run:**

node server.js # Start backend

npm start # Launch frontend

**RESEARCH QUESTIONS AND RESPONSES**

**Q1:** How effective is sentiment analysis for real-time feedback?  
**A1:** With basic NLP models, accuracy ranges from 85–90%, which is sufficient for general feedback analysis.

**Q2:** What architecture best supports scalability?  
**A2:** MVC-based Node.js + NoSQL structure supports modular and scalable design.

**Q3:** How does visualization improve decision-making?  
**A3:** Graphical dashboards allow quick identification of trends and satisfaction levels.

**REFERENCES**

1. Node.js Documentation – <https://nodejs.org>
2. Chart.js – <https://www.chartjs.org>
3. Sentiment API – <https://www.npmjs.com/package/sentiment>
4. TCS Project Guidelines – <https://www.tcs.com>