

**CASE STUDY**  
**ON**  
**FEEDBACK MANAGEMENT**  
**SYSTEM**

# INTRODUCTION

1. The Feedback Management System is a **web-based application** developed to efficiently collect, manage, and analyse feedback from users in a structured and organized manner.
2. The system aims to **replace traditional paper-based and manual feedback collection methods**, thereby reducing human errors and improving data accuracy and reliability.
3. Administrators are provided with the ability to **create, manage, and publish feedback forms** for various events conducted by the organization.
4. Users can easily **select the published events and submit their feedback** through a simple and user-friendly interface.
5. The administrator can **view all submitted feedback, search feedback records, filter them based on ratings, and delete feedback when necessary**.
6. The project ensures **role-based access control, secure data handling, and ease of use**, making the system reliable and efficient for both administrators and users

## ABSTRACT

The **Feedback Management System (FMS)** is a web-based application designed to simplify and automate the process of collecting feedback for events conducted by an organization or institution. The system provides a structured platform where administrators can create and publish event-specific feedback forms, ensuring that feedback is collected in an organized and consistent manner. Users can securely submit their feedback for selected events through an easy-to-use interface, eliminating the need for traditional paper-based feedback methods. The primary objective of this project is to offer an efficient and reliable mechanism for feedback collection, management, and analysis. By automating the entire feedback process, the system reduces manual data handling, minimizes errors, and improves data accuracy. The application includes advanced features such as searching feedback records, filtering feedback based on ratings, and viewing feedback in an event-wise manner, enabling administrators to gain valuable insights and make informed decisions. The Feedback Management System is developed using modern technologies such as **Spring Boot** for backend development, **Spring Security** for authentication and role-based access control, **JPA (Hibernate)** for database interaction, and **MySQL** for data storage. The frontend is designed using **Thymeleaf, HTML, and CSS**, providing a responsive and user-friendly interface. The system follows a **layered architecture**, which enhances scalability, maintainability, and security, making it suitable for real-world applications.

# OBJECTIVES OF THE PROJECT

**1. To automate the feedback collection, process**

The system aims to replace manual and paper-based feedback collection methods with a fully automated digital solution. This helps in saving time, reducing human errors, and ensuring faster and more accurate feedback processing.

**2. To allow administrators to create and publish event-based feedback forms**

The application enables administrators to create feedback forms for different events and publish them as required. This ensures that feedback is collected separately for each event, making evaluation and analysis more structured and meaningful.

**3. To allow users to submit feedback for selected events**

Users can easily select an event from the available list and submit their feedback through a simple and user-friendly interface.

**4. To implement role-based authentication and authorization**

The system uses role-based access control to differentiate between administrators and users. This ensures that only authorized users can access specific features, improve system security and prevent unauthorized access to sensitive data.

**5. To enable administrators to search and filter feedback based on ratings**

The application provides powerful search and filter options that allow administrators to quickly locate feedback records and analyse them based on ratings.

# CLIENT REQUIREMENTS

The client requires a system with the following features:

- **Admin** should be able to:
  - Login securely
  - Create events
  - Publish/unpublish feedback forms
  - View all feedback
  - Filter feedback by rating
  - Search feedback by name or email
  - Delete feedback records
  
- **User** should be able to:
  - Register and login
  - Select an event
  - Submit feedback
  - Logout securely

## TECHNOLOGIES USED

**Framework:** Spring Boot Java Framework.

- Java 8+
- Maven
- STS
- Apache Tomcat
- Spring core, Spring security(jwt), Spring data JPA(Hibernate) etc
- MySQL Database.
- HTML and CSS

## SYSTEM REQUIREMENTS

- **Backend:** Spring Boot, Spring Security, JPA (Hibernate)
- **Database:** MySQL 8.0
- **IDE:** Spring Tool Suite (STS)
- **Web Server:** Apache Tomcat
- **Frontend:** HTML, CSS, Thymeleaf
- **Browser:** Google Chrome
- **Testing:** JUnit

# PROJECT MODULES

## 1.Admin Module

- Secure login
- Event creation
- Publish feedback forms
- View all feedback
- Search feedback
- Filter feedback by rating
- Delete feedback

## 2.User Module

- User registration
- Secure login
- Select event
- Submit feedback
- Logout

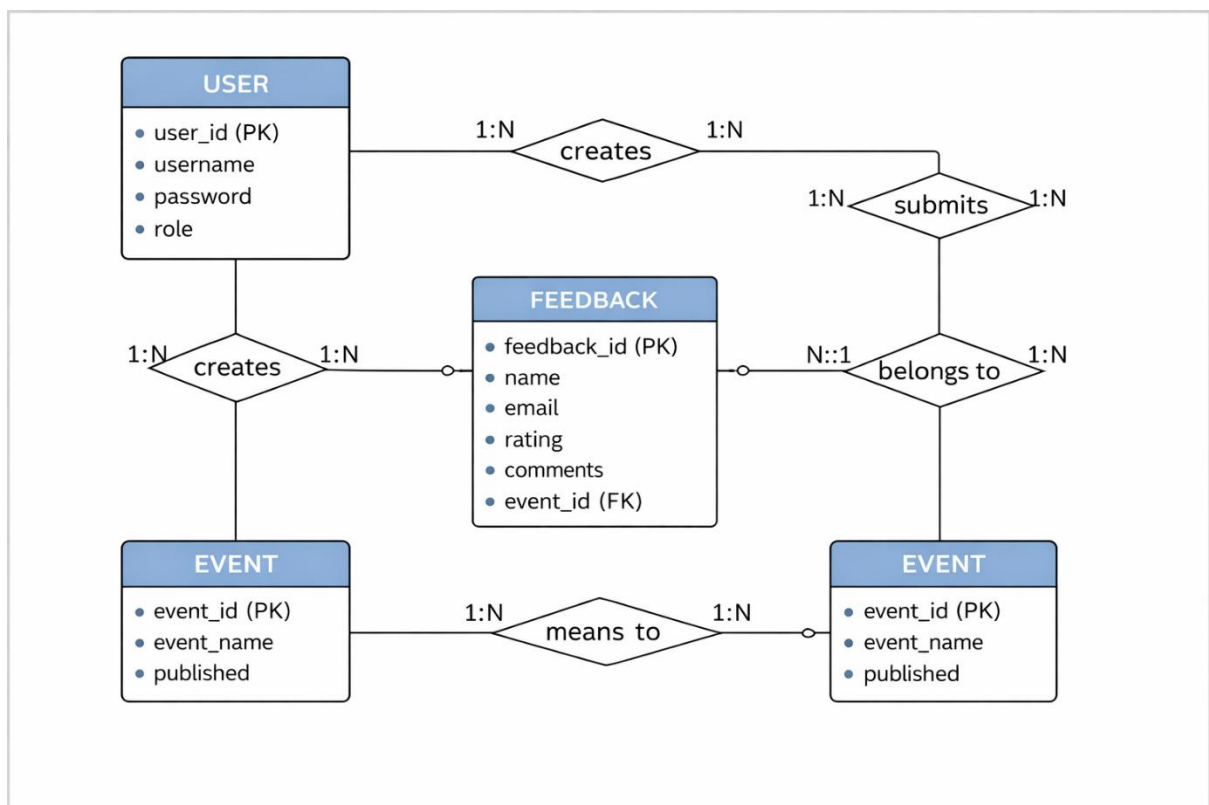
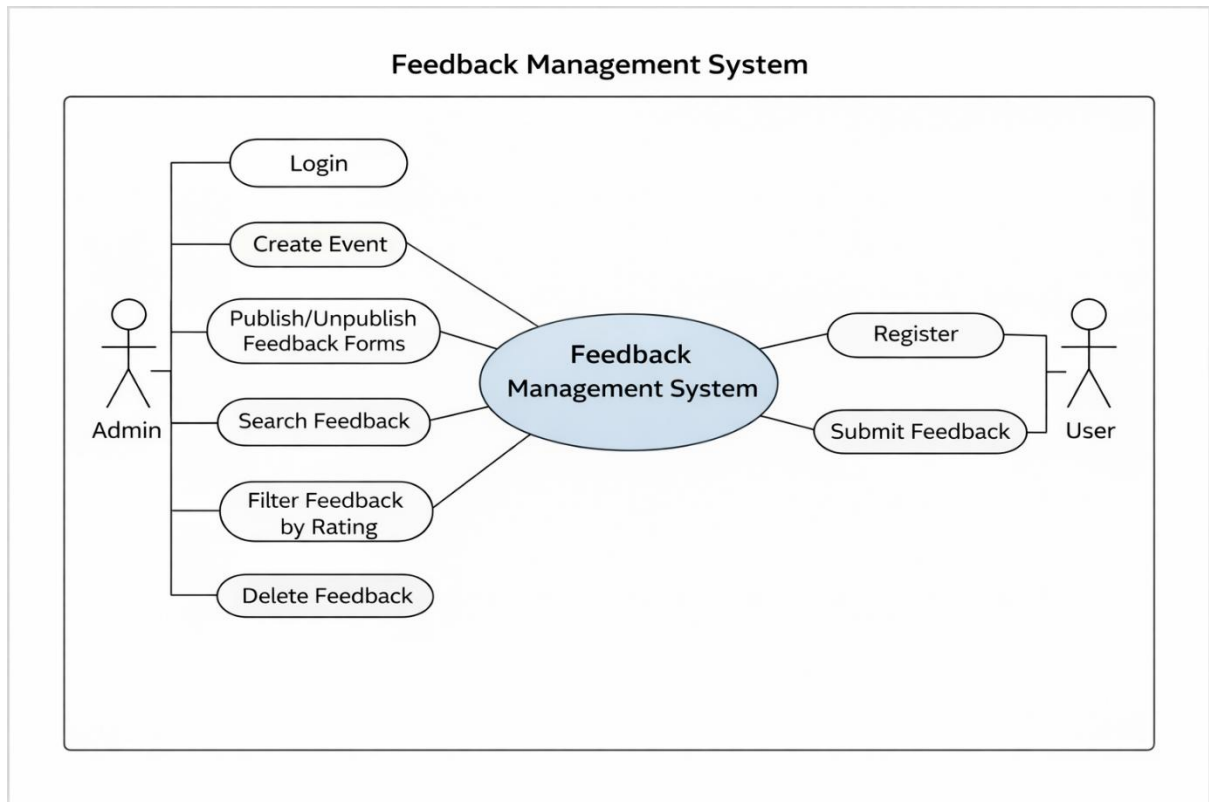
## 3.Event Module

- Event creation
- Event publishing
- Event listing

## 4.Feedback Module

- Feedback submission
- Feedback storage
- Feedback retrieval
- Feedback deletion

# ER DIAGRAM

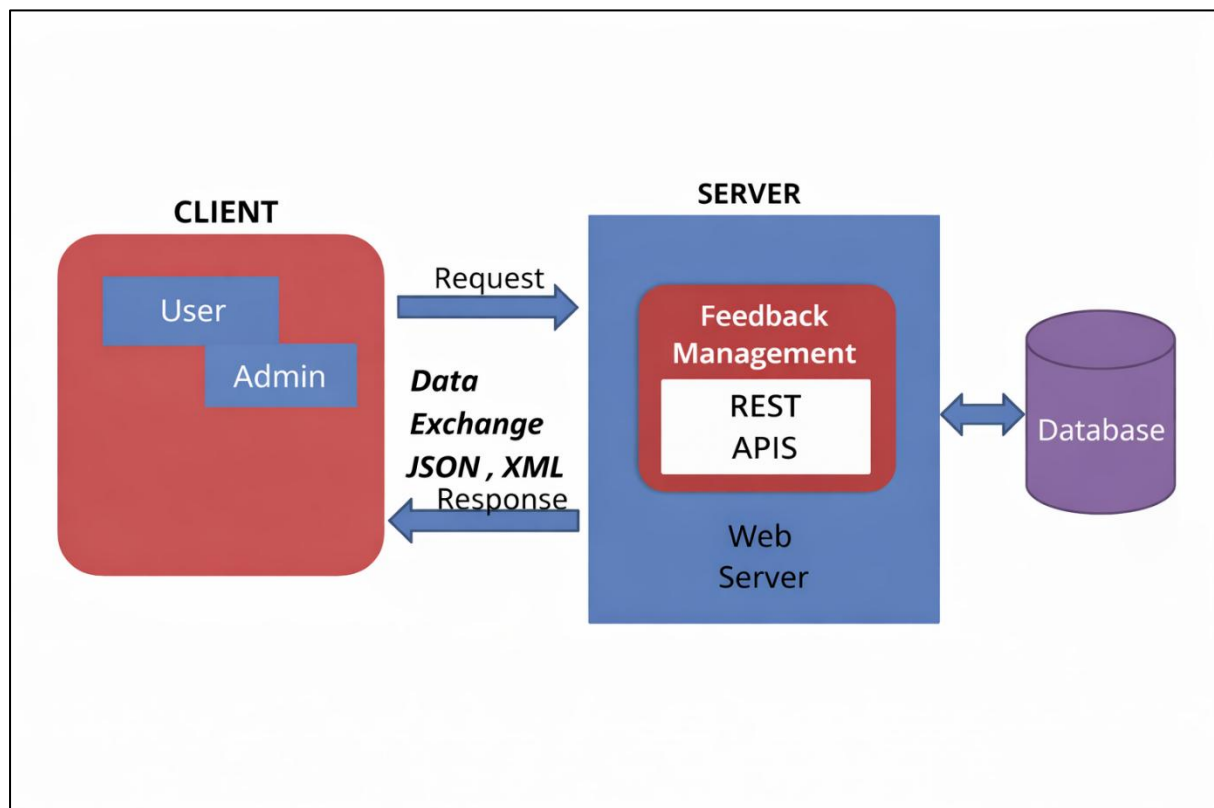




# CLIENT-SERVER ARCHITECTURE

The Feedback Management System is based on a **client–server architecture**, where the client and server have clearly defined roles. The client side consists of a web-based interface developed using **HTML, CSS, and Thymeleaf**, which is accessed through a web browser by users and administrators. All user actions such as login, event selection, feedback submission, searching, and filtering generate HTTP requests that are sent to the server.

The server side is implemented using **Spring Boot**, which handles request processing through controllers, business logic through services, and data access through repositories. The server communicates with a **MySQL database** to store and retrieve user, event, and feedback information. After processing the request, the server sends an appropriate response back to the client, ensuring secure, efficient, and reliable data exchange.



# **MODULES**

## **1. Admin Module**

1. Provides secure login access to administrators using role-based authentication.
2. Allows admins to create, manage, and publish feedback forms for different events.
3. Enables viewing, searching, filtering, and deleting of user feedback records.
4. Helps administrators analyse feedback effectively to improve event quality.

## **2. User Module**

1. Allows users to register and log in securely to the system.
2. Enables users to select published events and submit their feedback.
3. Provides a simple and user-friendly interface for feedback submission.
4. Ensures secure session handling and proper logout functionality.

## **3. Event Module**

1. Allows administrators to create events for which feedback can be collected.
2. Supports publishing and unpublishing of events based on requirements.
3. Displays only published events to users for feedback submission.
4. Helps organize feedback in an event-wise manner for better analysis.

## **4. Feedback Module**

1. Collects user feedback including name, email, rating, and comments.
2. Stores feedback securely in the database using JPA and MySQL.
3. Supports searching and filtering feedback based on ratings and keywords.
4. Allows administrators to manage feedback records efficiently.

## HTTP REQUEST METHODS

HTTP defines a set of request methods that indicate the action to be performed on feedback, events, and users in the system.

| HTTP Method | URL                                    | Description                                | Access Role    |
|-------------|--|--|----------------|
| GET         | http://localhost:8080/events           | Retrieves the list of all published events | User/<br>Admin |
| GET         | http://localhost:8080/feedback         | Retrieves all submitted feedback records   | Admin          |
| GET         | http://localhost:8080/feedback/event/5 | Retrieves feedback for the event with ID 5 | Admin          |
| POST        | http://localhost:8080/submit-feedback  | Submits feedback for a selected event      | User           |
| PUT         | http://localhost:8080/admin/event/10   | Updates details of the event with ID 10    | Admin          |
| DELETE      | http://localhost:8080/admin/feedback/7 | Deletes the feedback record with ID 7      | Admin          |

## URI-(UNIFORM RESOURCE IDENTIFIER)

A URI is the address used to identify a resource on the internet or a web application. It tells what resource you want and where it is located

| HTTP Method | URI   | Description                                    |
|-------------|---|--|
| GET         | <a href="http://localhost:8080/events/">http://localhost:8080/events/</a>                   | Returns the list of all published events       |
| GET         | <a href="http://localhost:8080/feedback/15">http://localhost:8080/feedback/15</a>           | Returns the feedback details of feedback ID 15 |
| POST        | <a href="http://localhost:8080/feedback/">http://localhost:8080/feedback/</a>               | Submits new feedback for an event              |
| PUT         | <a href="http://localhost:8080/admin/events/8">http://localhost:8080/admin/events/8</a>     | Updates the event details of event ID 8        |
| DELETE      | <a href="http://localhost:8080/admin/feedback/5">http://localhost:8080/admin/feedback/5</a> | Deletes the feedback of feedback ID 5          |

# DATABASES

## USER DETAILS DATABASE

SCHEMAS

Filter objects

amazon

eshop

feedback\_db

Tables

event

feedback

users

Views

Stored Procedures

Functions

joins

nji

sakila

sathyabama

saturday

securefeedback

Administration Schemas

No object selected

58 • SELECT \* FROM users;

59 • UPDATE users SET role = 'ADMIN' WHERE username = 'pankaj';

60 • SELECT username, password, role FROM users WHERE username='pankaj';

61 • SELECT \* FROM feedback;

62 • SELECT \* FROM event;

63 • SELECT \* FROM users;

64

65

66

67

68

69

Result Grid

Filter Rows:

id username password role

1 pankaj \$2a\$10\$fCqZDe3Cp8Eg5Z7jRoV.pUjZn5doN... ADMIN

2 nju \$2a\$10\$3EhD6ypHJMeYs4VvLT.OsRTO4WDv... USER

3 user \$2a\$10\$WChRe7zyFddCh1XPeO1Fq.3yLaJ2ve... USER

\* NULL NULL NULL NULL

## EVENT LIST DATABASE

SCHEMAS

Filter objects

amazon

eshop

feedback\_db

Tables

event

feedback

users

Views

Stored Procedures

Functions

joins

nji

sakila

sathyabama

saturday

securefeedback

Administration Schemas

No object selected

54 username VARCHAR(50) UNIQUE NOT NULL,

55 password VARCHAR(255) NOT NULL,

56 role VARCHAR(20) NOT NULL

57 ;

58 • SELECT \* FROM users;

59 • UPDATE users SET role = 'ADMIN' WHERE username = 'pankaj';

60 • SELECT username, password, role FROM users WHERE username='pankaj';

61 • SELECT \* FROM feedback;

62 • SELECT \* FROM event;

63 • SELECT \* FROM users;

64

65

Result Grid

Filter Rows:

id event\_name published

1 techno submit 1

2 voyage2025 1

\* NULL NULL NULL

## FEEDBACK DATABASE

SCHEMAS

Filter objects

amazon

eshop

feedback\_db

Tables

event

feedback

users

Views

Stored Procedures

Functions

joins

nji

sakila

sathyabama

saturday

securefeedback

Administration Schemas

No object selected

54 username VARCHAR(50) UNIQUE NOT NULL,

55 password VARCHAR(255) NOT NULL,

56 role VARCHAR(20) NOT NULL

57 ;

58 • SELECT \* FROM users;

59 • UPDATE users SET role = 'ADMIN' WHERE username = 'pankaj';

60 • SELECT username, password, role FROM users WHERE username='pankaj';

61 • SELECT \* FROM feedback;

62 • SELECT \* FROM event;

63 • SELECT \* FROM users;

64

65

Result Grid

Filter Rows:

id comments email name rating event\_id

21 vfvvff eggw@fvfv fregfergvg 3 1

22 csdcda fcvac@fvfv vfvfv 5 1

23 ffrfrgjbcb cngjcb@bcjcb hcbjcb 4 1

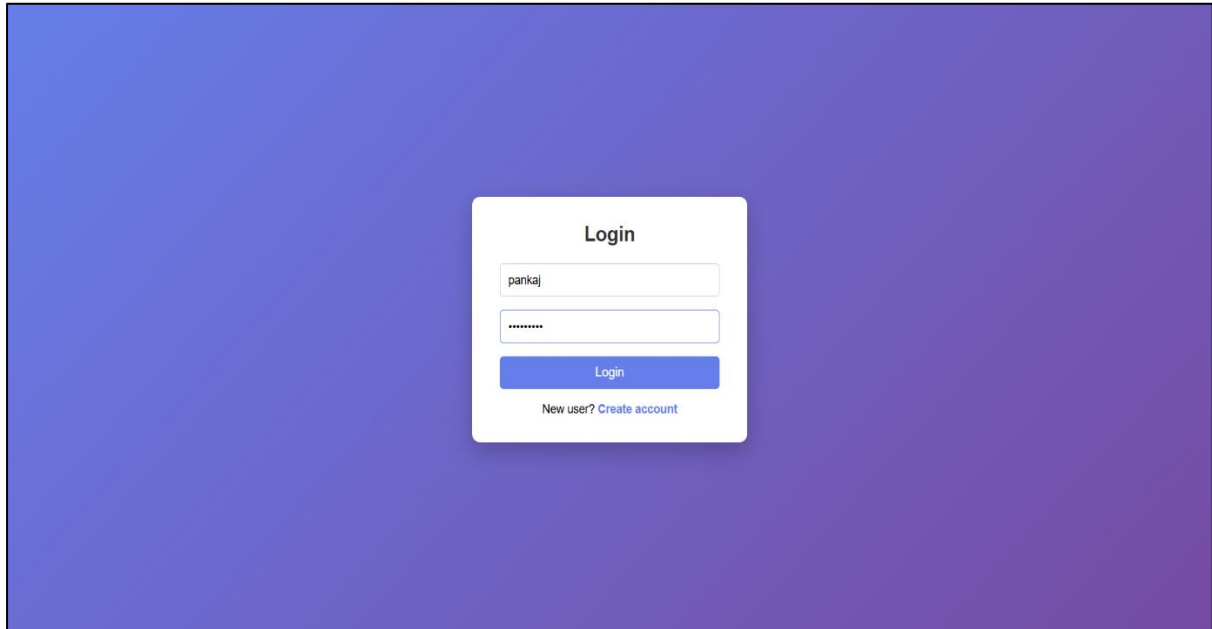
25 dcaca ca@dcac vfc 4 2

27 cdqcdcd@csad dcdac 3 1

\* NULL NULL NULL NULL NULL

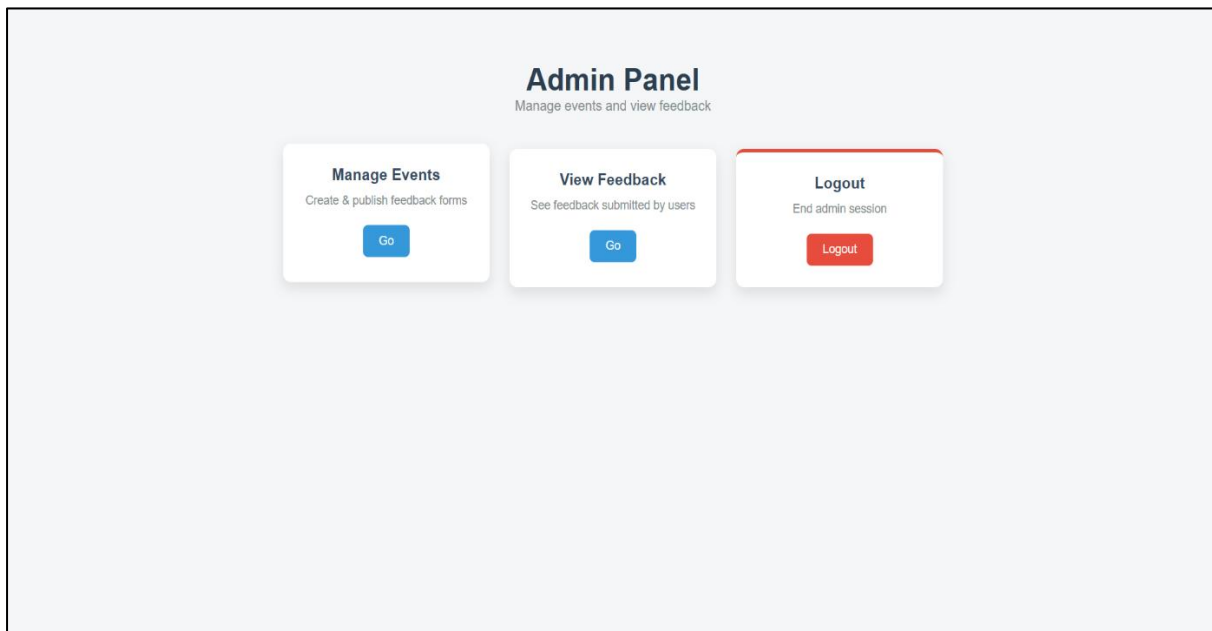
# ADMIN OUTPUTS

## ADMIN LOGIN PAGE



The image shows a login form centered on a purple gradient background. The form is titled "Login" and contains two input fields: one for the username "pankaj" and one for a password represented by seven dots. Below the password field is a blue "Login" button. At the bottom of the form, there is a link that says "New user? Create account".

## ADMIN PANEL



The image shows the Admin Panel interface. At the top, it says "Admin Panel" with the subtitle "Manage events and view feedback". Below this, there are three main sections: "Manage Events" with the description "Create & publish feedback forms" and a blue "Go" button; "View Feedback" with the description "See feedback submitted by users" and a blue "Go" button; and "Logout" with the description "End admin session" and a red "Logout" button.

# EVENT CREATION BY ADMIN

Create Event (Admin)

Event Name

Event Name

☐ Publish Event

Save Event

All Events

| ID | Event Name    | Published |
|----|---------------|-----------|
| 1  | techno submit | YES       |
| 2  | voyage2025    | YES       |

[← Back to Admin Panel](#)

# VIEW ALL FEEDBACK

All Feedback

Search by name or email

Search

Rating

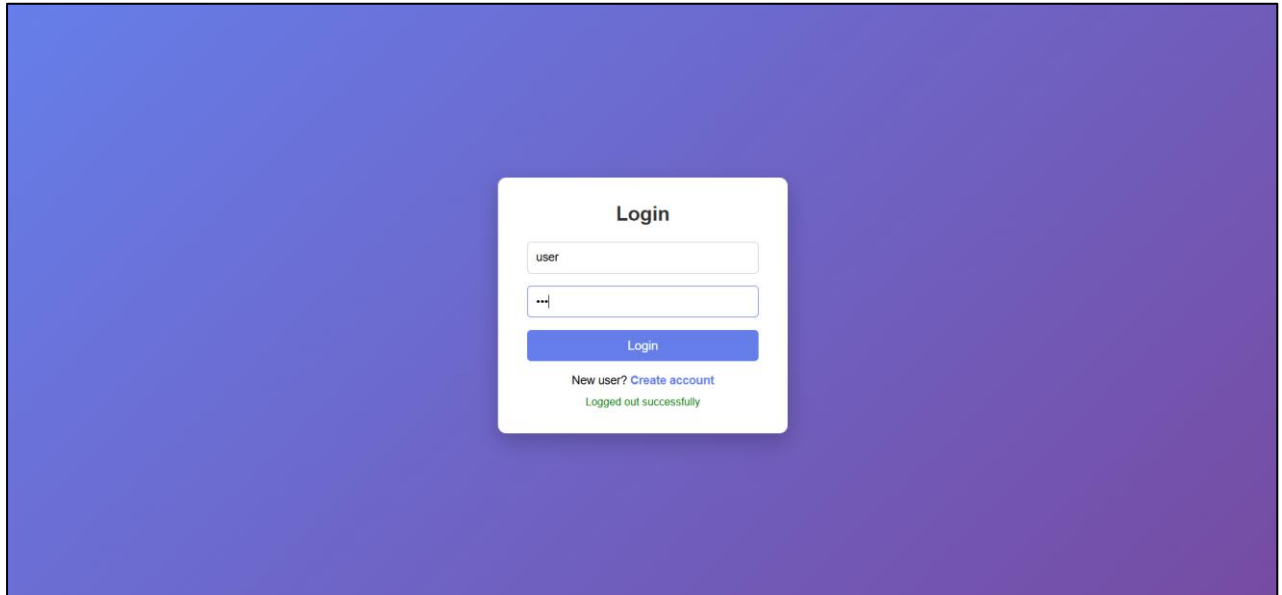
Filter

Back

| ID | Name      | Email         | Rating | Comments     | Action |
|----|-----------|---------------|--------|--------------|--------|
| 21 | fregfervg | eggw@fvfv     | 3      | vfvvff       | Delete |
| 22 | vfvvf     | fcvac@vfv     | 5      | csodca       | Delete |
| 23 | hcbhjb    | cjnjbcb@bcjbc | 4      | frfrfrgjcjbc | Delete |
| 25 | vfcc      | ca@dodc       | 4      | dcaca        | Delete |
| 27 | dcdac     | oddq2cd@csad  | 3      | odcacxxsdxsx | Delete |

# USER OUTPUTS

## USER LOGIN PAGE



A login form titled "Login" is centered on a purple gradient background. The form contains two input fields: the first is labeled "user" and contains the text "user"; the second is labeled "..." and contains a password mask "···". Below the inputs is a blue "Login" button. Underneath the button, there are two links: "New user? Create account" and "Logged out successfully".

**Login**

user

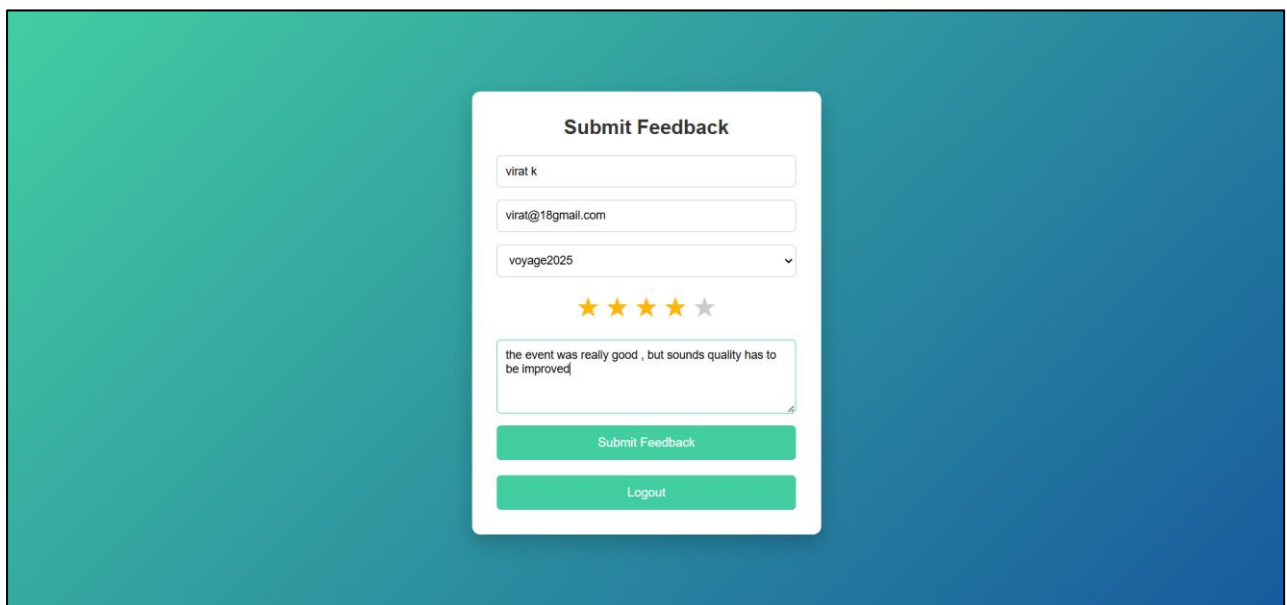
···

Login

New user? [Create account](#)

[Logged out successfully](#)

## FEEDBACK ENTER INTERFACE



A feedback form titled "Submit Feedback" is centered on a teal-to-blue gradient background. The form includes three input fields: the first for the name "virat k", the second for the email "virat@18gmail.com", and the third is a dropdown menu showing "voyage2025". Below these is a five-star rating system with four yellow stars and one grey star. A text area contains the feedback: "the event was really good , but sounds quality has to be improved". At the bottom are two green buttons: "Submit Feedback" and "Logout".

**Submit Feedback**

virat k

virat@18gmail.com

voyage2025

★★★★☆

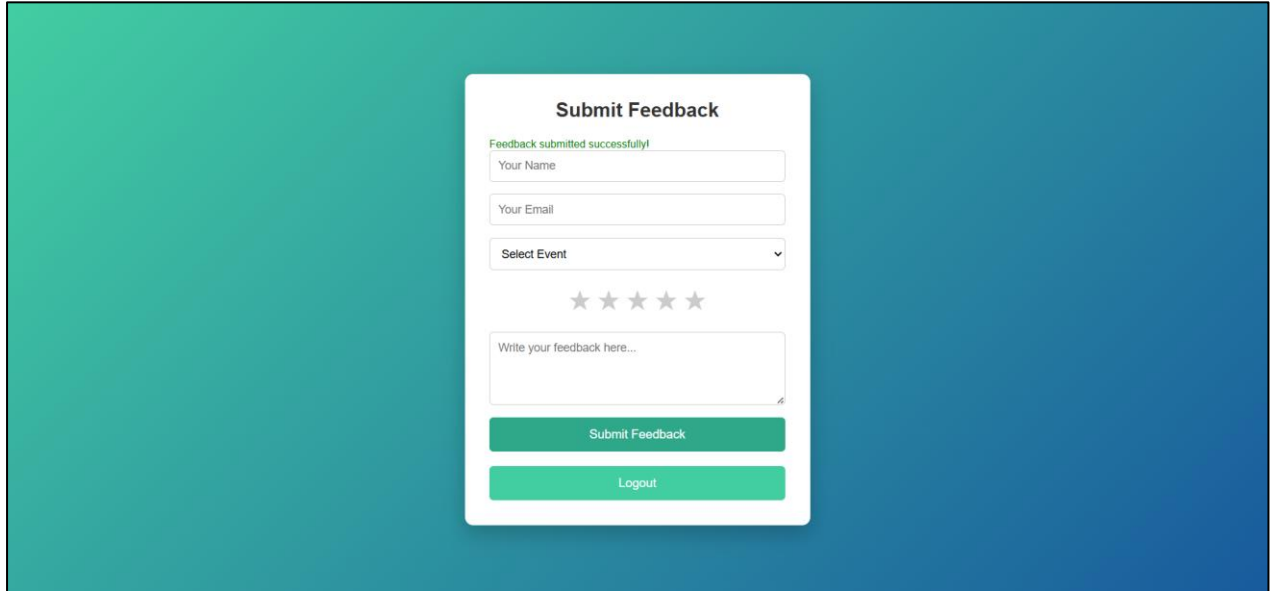
the event was really good , but sounds quality has to be improved

Submit Feedback

Logout



## FEEDBACK SUBMITTED SUCCESSFULLY



The image shows a 'Submit Feedback' form centered on a teal-to-blue gradient background. The form is white with a thin border and contains the following elements: a title 'Submit Feedback', a green success message 'Feedback submitted successfully!', input fields for 'Your Name' and 'Your Email', a 'Select Event' dropdown menu, a five-star rating system, a text area for feedback, and two buttons: 'Submit Feedback' (green) and 'Logout' (light green).

**Submit Feedback**

Feedback submitted successfully!

Your Name

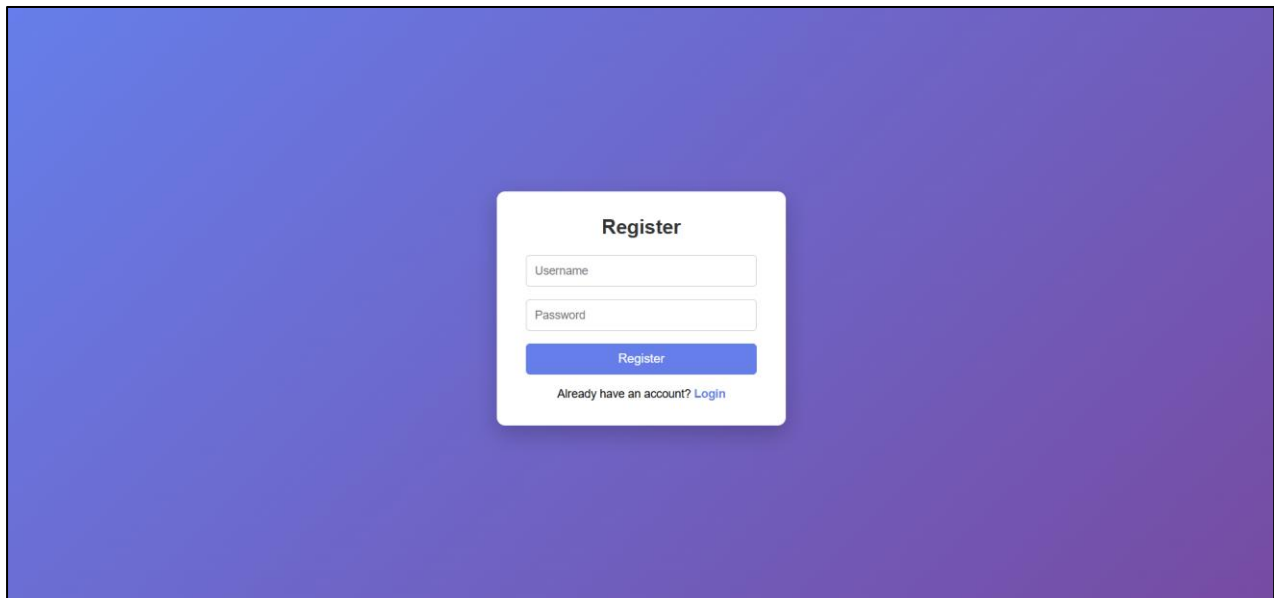
Your Email

Select Event

★ ★ ★ ★ ★

Write your feedback here...

## NEW REGISTRATION INTERFACE



The image shows a 'Register' form centered on a purple-to-blue gradient background. The form is white with a thin border and contains the following elements: a title 'Register', input fields for 'Username' and 'Password', a blue 'Register' button, and a link 'Already have an account? Login'.

**Register**

Username

Password

Already have an account? [Login](#)