# Summer Internship Training

(June 2025 – July 2025)

On

# CODEQUERY: THE ULTIMATE PL/SQL AND DATA SCIENCE BOOTCAMP

## Project

On

## Student Feedback and Evaluation Portal


**Submitted By**:

**Name** : Dakshineswar.M
(**Reg. no**. : 12303280)

**Name** : Harish Ramesh
(**Reg. no**. : 12305986)

**Name** : Akshit Salgundi
(**Reg. no**.: 12306490)

**Name** : J Tharun Adithya
(**Reg. no**.: 12306086)

**Submitted To:**

Dr. Avinash Kaur
(**UID** : 14557)

Dr. Parminder Singh
(**UID** : 16479)


LOVELY PROFESSIONAL UNIVERSITY

*Transforming Education Transforming India*


**School of Computer Science and Engineering**

**Lovely Professional University**

Jalandhar - Delhi G.T. Road, Phagwara, Punjab (India) - 144411

# Acknowledgment

We would like to express my heartfelt gratitude to **Dr. Avinash Kaur** and **Dr. Parminder Singh**, our project guide, for their invaluable support, guidance, and encouragement throughout the development of this project, “Student Feedback Management System”. Their insights and expertise helped me overcome challenges and stay focused on my objectives.

We would also like to thank the faculty and staff of **Lovely Professional University** for providing the resources and infrastructure needed to successfully complete this work. A special thanks to **our teammates**, **our classmates** and our friends for their cooperation, and to **our families** for their unwavering motivation and support during this project.

This project has been a tremendous learning experience, and We are truly thankful to everyone who contributed to its successful completion.

# Table of Contents

# 1.Introduction

The Student Feedback Management System is an application that runs on a database and was made with PL/SQL and Oracle Database. It makes it easier for teachers to get feedback from students, check the data, and make summaries of their performance. This system makes it easier to handle feedback and makes sure that data is correct, safe, and easy to get to.

The goal is to get feedback from students online and let teachers and administrators look at it. Scope: This applies to schools that want to evaluate students' academic performance. Technologies Used: Oracle Database, PL/SQL, and Oracle APEX (for an optional GUI).

# 2.Problem Statement

Collecting and processing student feedback by hand takes a lot of time, is prone to mistakes, and doesn't give you enough information. This project solves these problems by creating an automated system that uses PL/SQL for backend tasks and Oracle APEX for the user interface.

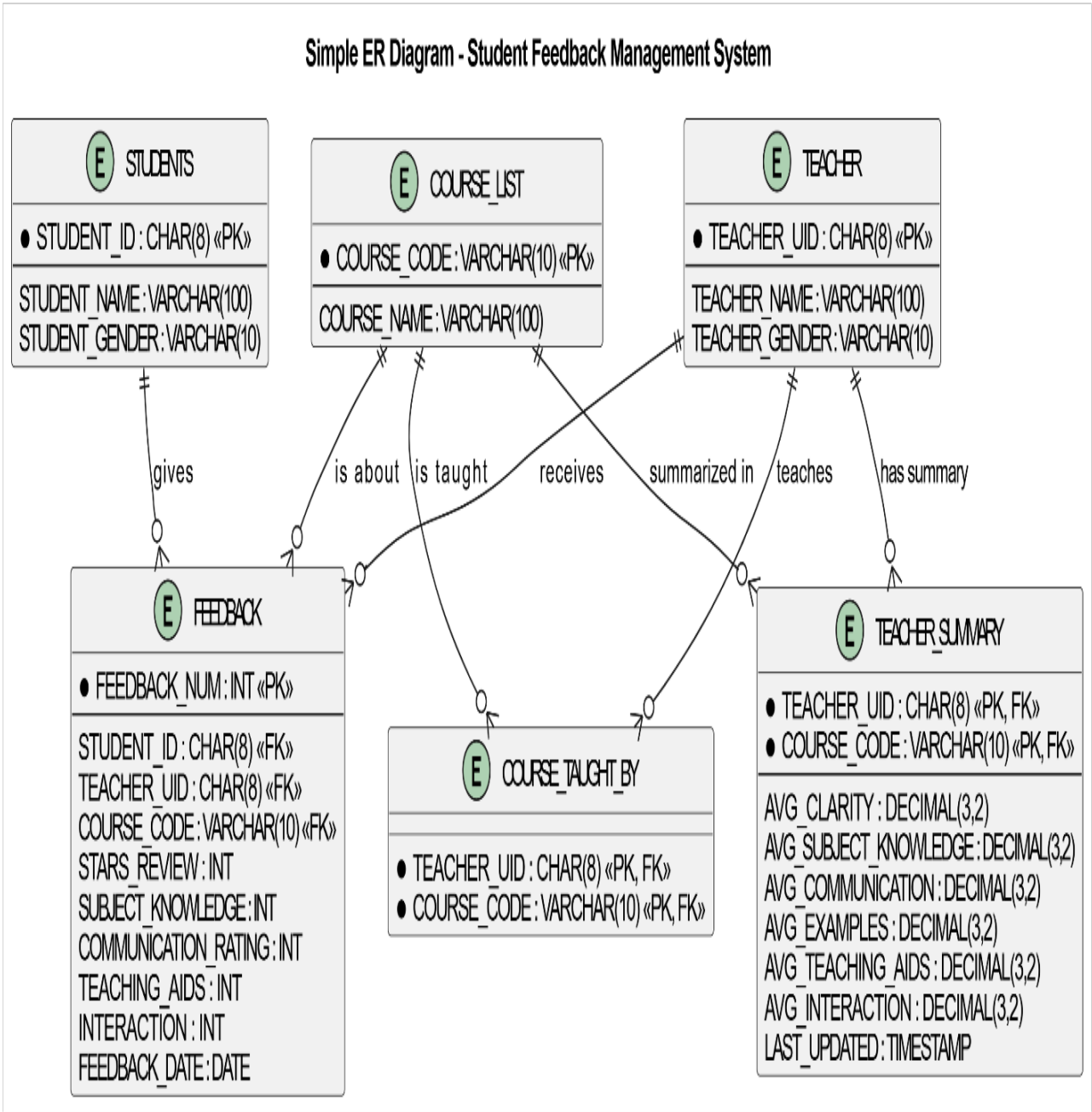
# 3.Objectives of the Project

- Automate student feedback collection.
- Validate student and course data before submission.
- Store feedback securely in an Oracle database.
- Enable teachers to view aggregated summaries.
- Maintain transparency and improve academic review systems.

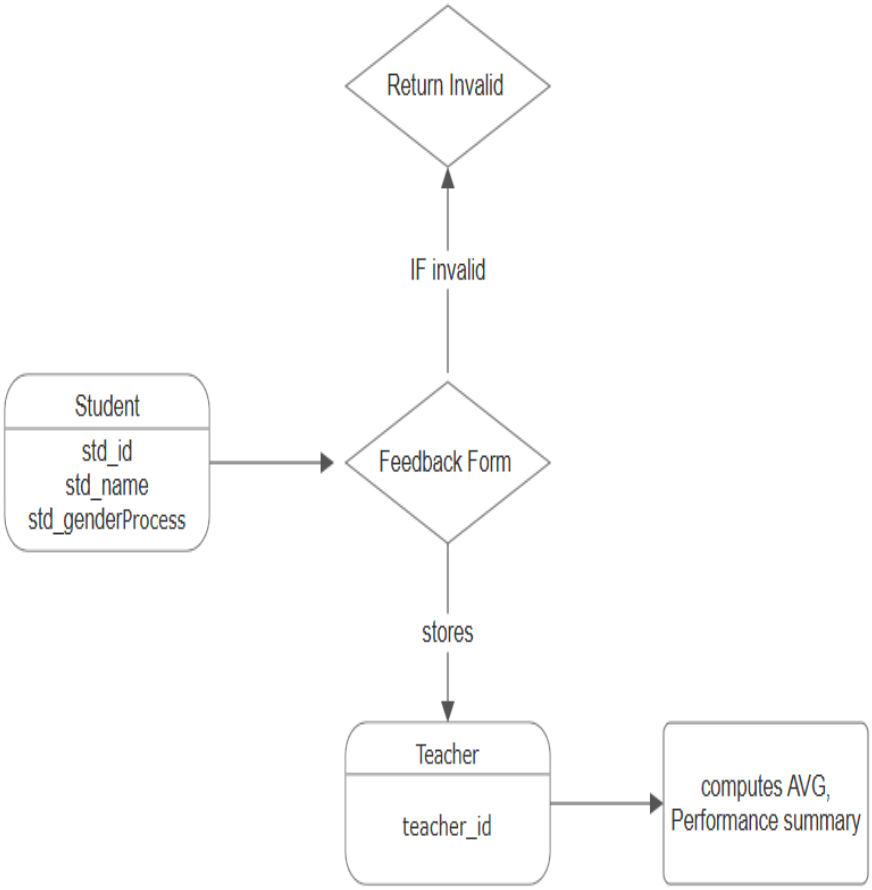
# 4. System Design

**ER Diagram:**

- Entities: Student, Teacher, Course, Feedback.
- Relations: Student submits Feedback, Feedback relates to Teacher and Course.



**Data Flow Diagram:**



# 5. PL/SQL Code

```
-- PROCEDIURES
CREATE PROCEDURE COUNT_FEEDBACK(
    P_TEACHER_UID IN VARCHAR2,P_COUNT OUT NUMBER)
)AS
BEGIN
    SELECT COUNT(*)
    INTO P_COUNT FROM STUDENT_FEEDBACK WHERE TEACHER_UID = P_TEACHER_UID;
    DBMS_OUTPUT.PUT_LINE ('TOTRAL FEEDBACK RECEIVED FROM STUDENTS : ' || P_COUNT);
    EXCEPTION
    WHEN OTHERS THEN P_COUNT := 0;
    DBMS_OUTPUT.PUT_LINE('ERROR HAS OCCUR');
    END;
    -- UPDATE AVERAGE RATING
    CREATE PROCEDURE UPDATE_AVG(
        P_TEACHER_UID IN VARCHAR2
    ) AS UPDTE_AVG NUMBER;
    BEGIN
        SELECT AVG(STARS_REVIEW) INTO V_AVG FROM FEEDBACK WHERE TEACHER_UID = P_TEACHER_UID;
        COMMIT ;
        DBMS_OUTPUT.PUT_LINE ('AVERAGE HAS BEEN UPDATAED , NEW AVG IS : ' || V_AVG);
        EXCEPTION
        WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE ('ERROR');
        ROLLBACK;
        END;

        CREATE FUNCTION TEACHER_SCORE (
            P_TEACHER_UID IN VARCHAR2
        )RETURN NUMBER AS TCHR_SCR NUMBER;
        BEGIN
            SELECT AVG((STARS_REVIEW + SUBJECT_KNOWLEDGE +RATING RATING_COMMUNICATION + RATING_TEACHING_AIDS + RATING_INTERACTIONS)/5)
            INTO TCHR_SCR FROM FEEDBACK WHERE TEACHER_UID = P_TEACHER_UID ;
            RETURN NVL(V_SCORE,0);
            EXCEPTION
            WHEN OTHERS THEN RETURN -1;
            END;
```

```
        CREATE FUNCTION TEACHER_SCORE (
            P_TEACHER_UID IN VARCHAR2
        )RETURN NUMBER AS TCHR_SCR NUMBER;
        BEGIN
            SELECT AVG((STARS_REVIEW + SUBJECT_KNOWLEDGE +RATING RATING_COMMUNICATION + RATING_TEACHING_AIDS + RATING_INTERACTIONS)/5)
            INTO TCHR_SCR FROM FEEDBACK WHERE TEACHER_UID = P_TEACHER_UID ;
            RETURN NVL(V_SCORE,0);
            EXCEPTION
            WHEN OTHERS THEN RETURN -1;
            END;
            --
            CREATE TRIGGER TCHR_SUMM
            AFTER INSERT OR UPDATE ON STUDENT_FEEDBACK FOR EACH ROW
            DECLARE V_ROWS NUMBER;
            BEGIN
                UPDATE TEACHER_SUMMARY TCHRSM SET
                TCHRSM.STARS_REVIEW = SELECT AVG(STARS_REVIEW)FROM FEEDBACK WHERE TEACHER_UID = :NEW.COURSE_CODE),
                TCHRSM.AVG_SUBJECT_KNOWLEDGE  = (SELECT AVG(SUBJECT_KNOWLEDGE) FROM FEEDBACK WHERE TEACHER_UID = :NEW.COURSE_CODE),
                TCHRSM.AVG_COMMUNICATION = (SELECT AVG(RATING_COMMUNICATION) FROM FEEDBACK WHERE TEACHER_UID =:NEW.TEACHER_UID AND COURSE_CODE =:NEW.COURSE_CODE),
                TCHRSM.AVG_TEACHING_AIDS  = (SELECT AVG(RATING_TEACHING_AIDS )FROM FEEDBACK WHERE TEACHER_UID =: NEW.TEACHER_UID AND TCHRSM.COURSE_CODE = :NEW.COURSE_CODE
                V_ROWS_UPDATED := SQL%ROWCOUNT;
                IF V_ROWS_UPDATED = 0 THEN
                INSERT INTO TEACHER_SUMMARY(TEACHER_UID,COURSE_CODE,AVG_STARS_REVIEW,AVG_SUBJECT_KNOWLEDGE,AVG_COMMUNICATION ,AVG_TEACHING_AIDS,AVG_INTERACTION, LAST_UPDA
                SELECT
                :NEW.TEACHER_ID, :NEW.COURSE_CODE,
                AVG(STARS_REVIEW),AVG(SUBJECT_KNOWLEDGE),AVG(RATING_COMMUNICATION),SYSTIMESTAMP FROM FEEDBACK WHERE TEACHER_UID = :NEW.TEACHER_UID AND COURSE_CODE = :NEW.
                END IF;
                END
```

# My SQL Code

```sql
-- STUDENT TABLE DEFINITION AND DECLARATION N
CREATE TABLE STUDENTS (
STUDENT_ID CHAR(8) PRIMARY KEY ,
STUDENT_NAME VARCHAR(100) NOT NULL,
STUDENT_GENDER VARCHAR(10),
CONSTRAINT VALID_STD_ID CHECK( LENGTH(STUDENT_ID) = 8
        AND STUDENT_ID NOT LIKE '%[^0-9]%')
);
-- TEACHERS TABLE CHECK
CREATE TABLE TEACHER(
TEACHER_UID CHAR(8) PRIMARY KEY,
TEACHER_NAME VARCHAR(100) NOT NULL,
TEACHER_GENDER VARCHAR(10),
CONSTRAINT TEACHER_UID_VALID CHECK (
        LENGTH(TEACHER_UID) = 8                          -- 1. Ensure the total length is 8 characters
        AND SUBSTR(TEACHER_UID, 1, 2) = 'TH'             -- 2. Ensure the first two characters are 'TH'
        AND SUBSTR(TEACHER_UID, 3, 6) NOT LIKE '%[^0-9]%' -- 3. Ensure the remaining 6 characters are digits
    )
);
 -- COURSE_LISTS TABLE
 CREATE TABLE COURSE_LIST (
COURSE_CODE VARCHAR(10) PRIMARY KEY, COURSE_NAME VARCHAR(100) NOT NULL
 );
--COURSE TAUGHT BY TEACHERS
CREATE TABLE COURSE_TAUGHT_BY (
    TEACHER_UID CHAR(8) REFERENCES TEACHER(TEACHER_UID),COURSE_CODE VARCHAR(10) REFERENCES COURSE_LIST(COURSE_CODE),
    PRIMARY KEY (TEACHER_UID, COURSE_CODE)
);
-- TABLE TO STORE STUDENT FEED BACK
CREATE TABLE FEEDBACK(
    FEEDBACK_NUM PRIMARY KEY,STUDENT_ID CHAR(8) REFERENCES STUDENTS(STUDENT_ID),
    COURSE_CODE VARCHAR(10) REFERENCES COURSE_LIST(COURSE_CODE),
    TEACHER_UID CHAR(8) REFERENCES TEACHER(TEACHER_UID),STARS_REVIEW INT NOT NULL,
    SUBJECT_KNOWLEDGE INT NOT NULL,
    COMMUNICATION_RATING INT NOT NULL,
    TEACHING_AIDS INT NOT NULL, INTERACTION INT NOT NULL, FEEDBACK_DATE DATE DEFAULT CURRENT_DATE,
```


```sql
 -- COURSE_LISTS TABLE
 CREATE TABLE COURSE_LIST (
COURSE_CODE VARCHAR(10) PRIMARY KEY, COURSE_NAME VARCHAR(100) NOT NULL
 );
--COURSE TAUGHT BY TEACHERS
CREATE TABLE COURSE_TAUGHT_BY (
    TEACHER_UID CHAR(8) REFERENCES TEACHER(TEACHER_UID),COURSE_CODE VARCHAR(10) REFERENCES COURSE_LIST(COURSE_CODE),
    PRIMARY KEY (TEACHER_UID, COURSE_CODE)
);
-- TABLE TO STORE STUDENT FEED BACK
CREATE TABLE FEEDBACK(
    FEEDBACK_NUM PRIMARY KEY,STUDENT_ID CHAR(8) REFERENCES STUDENTS(STUDENT_ID),
    COURSE_CODE VARCHAR(10) REFERENCES COURSE_LIST(COURSE_CODE),
    TEACHER_UID CHAR(8) REFERENCES TEACHER(TEACHER_UID),STARS_REVIEW INT NOT NULL,
    SUBJECT_KNOWLEDGE INT NOT NULL,
    COMMUNICATION_RATING INT NOT NULL,
    TEACHING_AIDS INT NOT NULL, INTERACTION INT NOT NULL, FEEDBACK_DATE DATE DEFAULT CURRENT_DATE,
```


```sql
    CONSTRAINT VALID_RATING CHECK (STARS_REVIEW BETWEEN 1 AND 5) AND SUBJECT_KNOWLEDGE BETWEEN 1 AND R COMMUNICATION BETWEEN 1 AND 5 AND TEACHING_AIDS BETWEEEN 1 AND 5
    AND INTERACTION BETWEEN 1 AND 5)
);
```

```sql
-- RANDOM DATA FOR ALL TABLES
INSERT INTO COURSE_LIST (COURSE_CODE, COURSE_NAME) VALUES
('CSE101', 'Introduction to Computer Science'),
('MAT201', 'Advanced Mathematics'),
('PHY301', 'Physics for Engineers'),
('ENG401', 'Technical Writing');
INSERT INTO UNLTEACHER (TEACHER_ID, TEACHER_NAME, TEACHER_GEN) VALUES
('TH000001', 'Dr. Smith', 'Male'),
('TH000002', 'Prof. Johnson', 'Female'),
('TH000003', 'Dr. Williams', 'Male');
INSERT INTO TEACHER_COURSE (TEACHER_ID, COURSE_CODE) VALUES
('TH000001', 'CSE101'),
('TH000001', 'MAT201'),
('TH000002', 'PHY301'),
('TH000003', 'ENG401');
INSERT INTO UNLSTUDENTS (STUD_ID, STUD_NAME, STUD_GEN) VALUES
('10000001', 'Alice Brown', 'Female'),
('10000002', 'Bob Green', 'Male'),
('10000003', 'Charlie White', 'Male'),
('10000004', 'Diana Black', 'Female');
INSERT INTO STUD_FEEDBACK (STUD_ID, COURSE_CODE, TEACHER_ID,
    RATING_CLARITY, RATING_SUBJECT_KNOWLEDGE, RATING_COMMUNICATION,
    RATING_EXAMPLES, RATING_TEACHING_AIDS, RATING_INTERACTION) VALUES
('10000001', 'CSE101', 'TH000001', 5, 4, 5, 4, 5, 4),
('10000002', 'CSE101', 'TH000001', 4, 5, 4, 5, 4, 5),
('10000003', 'PHY301', 'TH000002', 3, 4, 3, 4, 3, 4),
('10000004', 'ENG401', 'TH000003', 5, 5, 5, 5, 5, 5);
-- PROCEDIURES
CREATE PROCEDURE COUNT_FEEDBACK(
    P_TEACHER_UID IN VARCHAR2,P_COUNT OUT NUMBER)
AS
```

# Oracle APEX(Table Creation)

APEX App Builder SQL Workshop Team Development Gallery Search DM Dakshineswar M studentfeadback

SQL Commands Schema WKSP_STUDENTFEADBACI

Language SQL Rows 10 Clear Command Find Tables Save Run

```sql
--University Student Table
CREATE TABLE STUDENT_MASTER (
    STUDENT_ID VARCHAR2(8) PRIMARY KEY,
    STUDENT_NAME VARCHAR2(100),
    CONSTRAINT CK_STUDENT_ID_FORMAT CHECK (REGEXP_LIKE(STUDENT_ID, '^\d{8}$'))
);

--Course Table
CREATE TABLE COURSE_MASTER (
    COURSE_CODE VARCHAR2(6) PRIMARY KEY,
    COURSE_NAME VARCHAR2(100),
    CONSTRAINT CK_COURSE_FORMAT CHECK (REGEXP_LIKE(COURSE_CODE, '^[A-Za-z]{3}\d{3}$'))
);

--University teacher table
CREATE TABLE UNI_TEACHER (
    TEACHER_ID   VARCHAR2(8) PRIMARY KEY,
    TEACHER_NAME VARCHAR2(100),
    TEACHER_GEN  VARCHAR2(10),
    CONSTRAINT CK_TEACHER_ID_FORMAT CHECK (REGEXP_LIKE(TEACHER_ID, '^\d{8}$'))
);

--Teacher Course Table
CREATE TABLE TEACHER_COURSE (
    TEACHER_ID   VARCHAR2(8),
    COURSE_CODE  VARCHAR2(6),
    CONSTRAINT PK_TC PRIMARY KEY (TEACHER_ID, COURSE_CODE),
    CONSTRAINT FK_TC_TEACHER FOREIGN KEY (TEACHER_ID) REFERENCES UNI_TEACHER(TEACHER_ID),
    CONSTRAINT FK_TC_COURSE FOREIGN KEY (COURSE_CODE) REFERENCES COURSE_MASTER(COURSE_CODE)
);

--Student Feedback table
CREATE TABLE STUDENT_FEEDBACK (
    FEEDBACK_ID      NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
```

Results Explain Describe Saved SQL History

```
Table dropped.
```

dakshinm205@gmail.com studentfeadback en Copyright © 1999, 2024, Oracle and/or its affiliates. Oracle APEX 24.2.6

APEX App Builder SQL Workshop Team Development Gallery Search DM Dakshineswar M studentfeadback

SQL Commands Schema WKSP_STUDENTFEADBACI

Language SQL Rows 10 Clear Command Find Tables Save Run

```sql
--Student Feedback table
CREATE TABLE STUDENT_FEEDBACK (
    FEEDBACK_ID      NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    STUDENT_ID       VARCHAR2(8),
    TEACHER_ID       VARCHAR2(8),
    COURSE_CODE      VARCHAR2(6),
    RATING_CLARITY              NUMBER(1),
    RATING_SUBJECT_KNOWLEDGE NUMBER(1),
    RATING_COMMUNICATION     NUMBER(1),
    RATING_EXAMPLES          NUMBER(1),
    RATING_TEACHING_AIDS     NUMBER(1),
    RATING_INTERACTION       NUMBER(1),
    FEEDBACK_DATE    DATE DEFAULT SYSDATE,
    CONSTRAINT FK_FEEDBACK_STUDENT FOREIGN KEY (STUDENT_ID) REFERENCES STUDENT_MASTER(STUDENT_ID),
    CONSTRAINT FK_FEEDBACK_TEACHER FOREIGN KEY (TEACHER_ID) REFERENCES UNI_TEACHER(TEACHER_ID),
    CONSTRAINT FK_FEEDBACK_COURSE FOREIGN KEY (COURSE_CODE) REFERENCES COURSE_MASTER(COURSE_CODE),
    CONSTRAINT CK_CLARITY CHECK (RATING_CLARITY BETWEEN 1 AND 5),
    CONSTRAINT CK_SUBJECT_KNOWLEDGE CHECK (RATING_SUBJECT_KNOWLEDGE BETWEEN 1 AND 5),
    CONSTRAINT CK_COMMUNICATION CHECK (RATING_COMMUNICATION BETWEEN 1 AND 5),
    CONSTRAINT CK_EXAMPLES CHECK (RATING_EXAMPLES BETWEEN 1 AND 5),
    CONSTRAINT CK_TEACHING_AIDS CHECK (RATING_TEACHING_AIDS BETWEEN 1 AND 5),
    CONSTRAINT CK_INTERACTION CHECK (RATING_INTERACTION BETWEEN 1 AND 5),
    CONSTRAINT UNQ_FEEDBACK UNIQUE (STUDENT_ID, COURSE_CODE)
);

--Teacher Summary Table
CREATE TABLE TEACHER_SUMMARY (
    TEACHER_ID       VARCHAR2(8),
    COURSE_CODE      VARCHAR2(6),
    AVG_CLARITY           NUMBER(5,2),
    AVG_SUBJECT_KNOWLEDGE NUMBER(5,2),
    AVG_COMMUNICATION     NUMBER(5,2),
    AVG_EXAMPLES          NUMBER(5,2),
```

Results Explain Describe Saved SQL History

```
Table dropped.
```

dakshinm205@gmail.com studentfeadback en Copyright © 1999, 2024, Oracle and/or its affiliates. Oracle APEX 24.2.6

APEX App Builder SQL Workshop Team Development Gallery Search DM Dakshineswar M studentfeadback

SQL Commands Schema WKSP_STUDENTFEADBACI

Language SQL Rows 10 Clear Command Find Tables Save Run

```sql
    CONSTRAINT CK_SUBJECT_KNOWLEDGE CHECK (RATING_SUBJECT_KNOWLEDGE BETWEEN 1 AND 5),
    CONSTRAINT CK_COMMUNICATION CHECK (RATING_COMMUNICATION BETWEEN 1 AND 5),
    CONSTRAINT CK_EXAMPLES CHECK (RATING_EXAMPLES BETWEEN 1 AND 5),
    CONSTRAINT CK_TEACHING_AIDS CHECK (RATING_TEACHING_AIDS BETWEEN 1 AND 5),
    CONSTRAINT CK_INTERACTION CHECK (RATING_INTERACTION BETWEEN 1 AND 5),
    CONSTRAINT UNQ_FEEDBACK UNIQUE (STUDENT_ID, COURSE_CODE)
);

--Teacher Summary Table
CREATE TABLE TEACHER_SUMMARY (
    TEACHER_ID       VARCHAR2(8),
    COURSE_CODE      VARCHAR2(6),
    AVG_CLARITY          NUMBER(5,2),
    AVG_SUBJECT_KNOWLEDGE NUMBER(5,2),
    AVG_COMMUNICATION     NUMBER(5,2),
    AVG_EXAMPLES          NUMBER(5,2),
    AVG_TEACHING_AIDS     NUMBER(5,2),
    AVG_INTERACTION       NUMBER(5,2),
    LAST_UPDATED DATE DEFAULT SYSDATE,
    CONSTRAINT PK_TEACHER_SUMMARY PRIMARY KEY (TEACHER_ID, COURSE_CODE)
);
```

Results Explain Describe Saved SQL History

```
Table dropped.
```

dakshinm205@gmail.com studentfeadback en Copyright © 1999, 2024, Oracle and/or its affiliates. Oracle APEX 24.2.6

APEX App Builder SQL Workshop Team Development Gallery Search DM Dakshineswar M studentfeadback

SQL Commands Schema WKSP_STUDENTFEADBACI

Language SQL Rows 10 Clear Command Find Tables Save Run

```sql
-- Adding some content into STUDENT_MASTER table
INSERT INTO STUDENT_MASTER (STUDENT_ID, STUDENT_NAME) VALUES ('20230001', 'Anjali Sharma');
INSERT INTO STUDENT_MASTER (STUDENT_ID, STUDENT_NAME) VALUES ('20230002', 'Rahul Verma');
INSERT INTO STUDENT_MASTER (STUDENT_ID, STUDENT_NAME) VALUES ('20230003', 'Priya Mehta');

-- Course Master
INSERT INTO COURSE_MASTER (COURSE_CODE, COURSE_NAME) VALUES ('CSE101', 'Intro to Computer Science');
INSERT INTO COURSE_MASTER (COURSE_CODE, COURSE_NAME) VALUES ('MAT102', 'Engineering Mathematics');
INSERT INTO COURSE_MASTER (COURSE_CODE, COURSE_NAME) VALUES ('PHY103', 'Physics Fundamentals');

-- Uni teacher
INSERT INTO UNI_TEACHER (TEACHER_ID, TEACHER_NAME, TEACHER_GEN) VALUES ('10000001', 'Dr. Neha Singh', 'Female');
INSERT INTO UNI_TEACHER (TEACHER_ID, TEACHER_NAME, TEACHER_GEN) VALUES ('10000002', 'Prof. Arjun Rao', 'Male');
INSERT INTO UNI_TEACHER (TEACHER_ID, TEACHER_NAME, TEACHER_GEN) VALUES ('10000003', 'Dr. Kavita Jain', 'Female');

--Teacher course
INSERT INTO TEACHER_COURSE (TEACHER_ID, COURSE_CODE) VALUES ('10000001', 'CSE101');
INSERT INTO TEACHER_COURSE (TEACHER_ID, COURSE_CODE) VALUES ('10000002', 'MAT102');
INSERT INTO TEACHER_COURSE (TEACHER_ID, COURSE_CODE) VALUES ('10000003', 'PHY103');
INSERT INTO TEACHER_COURSE (TEACHER_ID, COURSE_CODE) VALUES ('10000001', 'MAT102');

--Student Feedback
INSERT INTO STUDENT_FEEDBACK (
    STUDENT_ID, TEACHER_ID, COURSE_CODE,
    RATING_CLARITY, RATING_SUBJECT_KNOWLEDGE,
    RATING_COMMUNICATION, RATING_EXAMPLES,
    RATING_TEACHING_AIDS, RATING_INTERACTION
) VALUES (
    '20230001', '10000001', 'CSE101',
```

Results Explain Describe Saved SQL History

```
1 row(s) inserted.

0.06 seconds
```

dakshinm205@gmail.com studentfeadback en Copyright © 1999, 2024, Oracle and/or its affiliates. Oracle APEX 24.2.6

# 6. Database Design

| Table Name | Purpose |
|---|---|
| STUDENTS | Stores student details like ID, name, and gender. |
| TEACHER | Stores teacher details with validation (ID must start with 'TH'). |
| COURSE_LIST | Holds list of available courses and their codes. |
| COURSE_TAUGHT_BY | Mapping between teachers and the courses they teach (many-to-many). |
| FEEDBACK | Stores feedback ratings given by students to teachers for specific courses. |
| TEACHER_SUMMARY | Aggregated/summary feedback data for each teacher-course combo. |

**Relationships**

1. **One-to-Many**:
   - One **student** can give many feedbacks.
   - One **teacher** can be associated with many feedbacks and courses.
   - One **course** can have many feedbacks and be taught by multiple teachers.
2. **Many-to-Many** (Handled via junction table COURSE_TAUGHT_BY):
   - Teachers ↔ Courses

**Key Constraints**

- **Primary Keys:**
  - STUDENT_ID, TEACHER_UID, COURSE_CODE
  - Composite PK in COURSE_TAUGHT_BY (TEACHER_UID + COURSE_CODE)
- **Foreign Keys:**
  - FEEDBACK references STUDENTS, TEACHER, COURSE_LIST

- TEACHER_SUMMARY references TEACHER, COURSE_LIST

**Check Constraints & Validations:**

- STUDENT_ID must be 8 digits long and numeric.
- TEACHER_UID must start with 'TH' followed by 6 digits.
- Ratings must be between 1 and 5.

# 7. Procedures & Triggers

- **Procedure**: COUNT_FEEDBACK
  **Purpose**: Returns total feedback count for a teacher.
- **Procedure**: UPDATE_AVG
  **Purpose**: Updates average star rating for a teacher.
- **Function**: TEACHER_SCORE
  **Purpose**: Calculates the average feedback score for a teacher.
- **Trigger**: TCHR_SUMM
  **Purpose**: After feedback is added/updated, it updates or inserts summary data into TEACHER_SUMMARY.

```oracle_plsql
CREATE PROCEDURE COUNT_FEEDBACK(
    P_TEACHER_UID IN VARCHAR2,P_COUNT OUT NUMBER)
)AS
BEGIN
    SELECT COUNT(*)
    INTO P_COUNT FROM STUDENT_FEEDBACK WHERE TEACHER_UID = P_TEACHER_UID;
    DBMS_OUTPUT.PUT_LINE ('TOTRAL FEEDBACK RECEIVED FROM STUDENTS : ' || P_COUNT);
    EXCEPTION
    WHEN OTHERS THEN P_COUNT := 0;
    DBMS_OUTPUT.PUT_LINE('ERROR HAS OCCUR');
    END;
    -- UPDATE AVERAGE RATING
    CREATE PROCEDURE UPDATE_AVG(
        P_TEACHER_UID IN VARCHAR2
    ) AS UPDTE_AVG NUMBER;
    BEGIN
        SELECT AVG(STARS_REVIEW) INTO V_AVG FROM FEEDBACK WHERE TEACHER_UID = P_TEACHER_UID;
        COMMIT ;
        DBMS_OUTPUT.PUT_LINE ('AVERAGE HAS BEEN UPDATAED , NEW AVG IS : ' || V_AVG);
        EXCEPTION
        WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE ('ERROR');
        ROLLBACK;
        END;

        CREATE FUNCTION TEACHER_SCORE (
            P_TEACHER_UID IN VARCHAR2
        )RETURN NUMBER AS TCHR_SCR NUMBER;
        BEGIN
            SELECT AVG((STARS_REVIEW + SUBJECT_KNOWLEDGE +RATING RATING_COMMUNICATION + RATING_TEACHING_AIDS + RATING_INTERACTIONS)/5)
            INTO TCHR_SCR FROM FEEDBACK WHERE TEACHER_UID = P_TEACHER_UID ;
            RETURN NVL(V_SCORE,0);
            EXCEPTION
            WHEN OTHERS THEN RETURN -1;
            END;
```

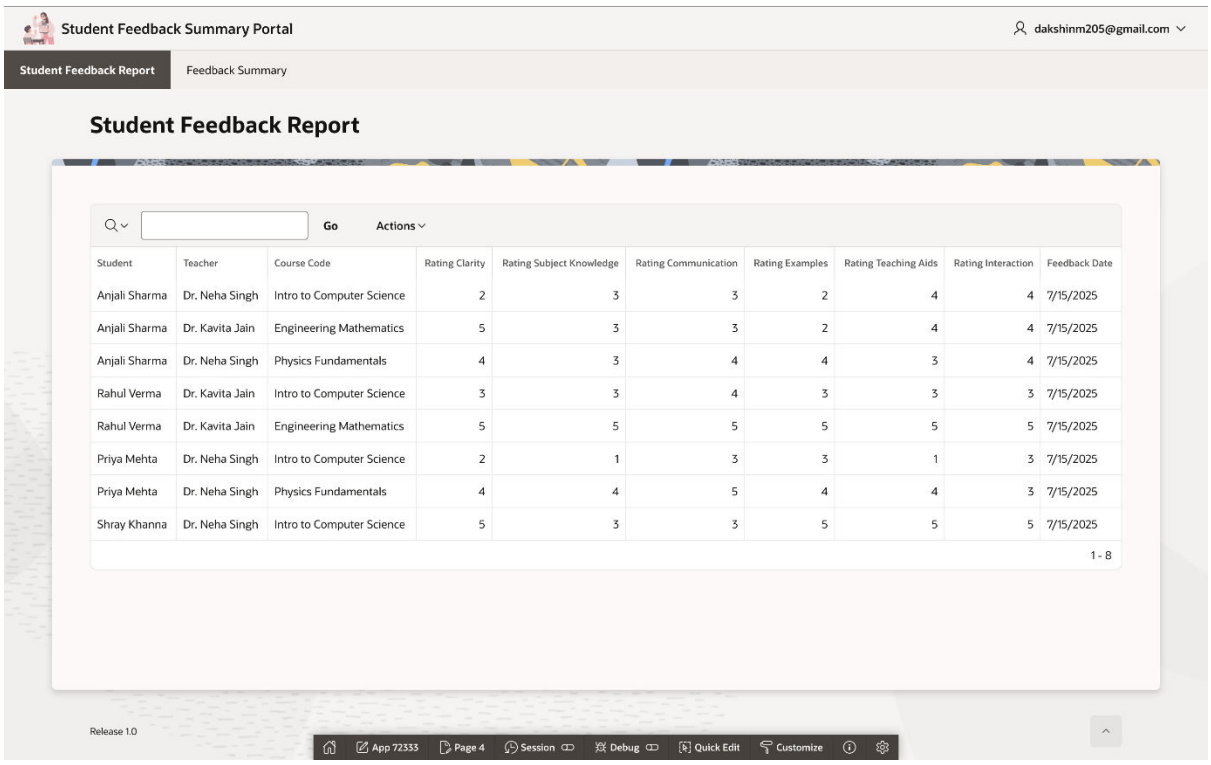
# 8. Output Screens

Students can easily submit feedback using a structured form thanks to the system's user-friendly interface, which was created with Oracle APEX. Important information like the course code, rating parameters, student and teacher IDs, and optional remarks are recorded in this feedback form. The system verifies that the entry has been entered into the database by displaying a confirmation message to the user upon successful submission.

The data is processed by the backend to produce summary reports for administrative and educational uses. These reports include metrics for overall satisfaction, feedback count, and average ratings for each teacher. These reports can be viewed by administrators and teachers to assess the efficacy of instruction and pinpoint areas in need of development. APEX reports or downloadable formats like CSV or PDF are used to present the condensed data in an understandable manner.

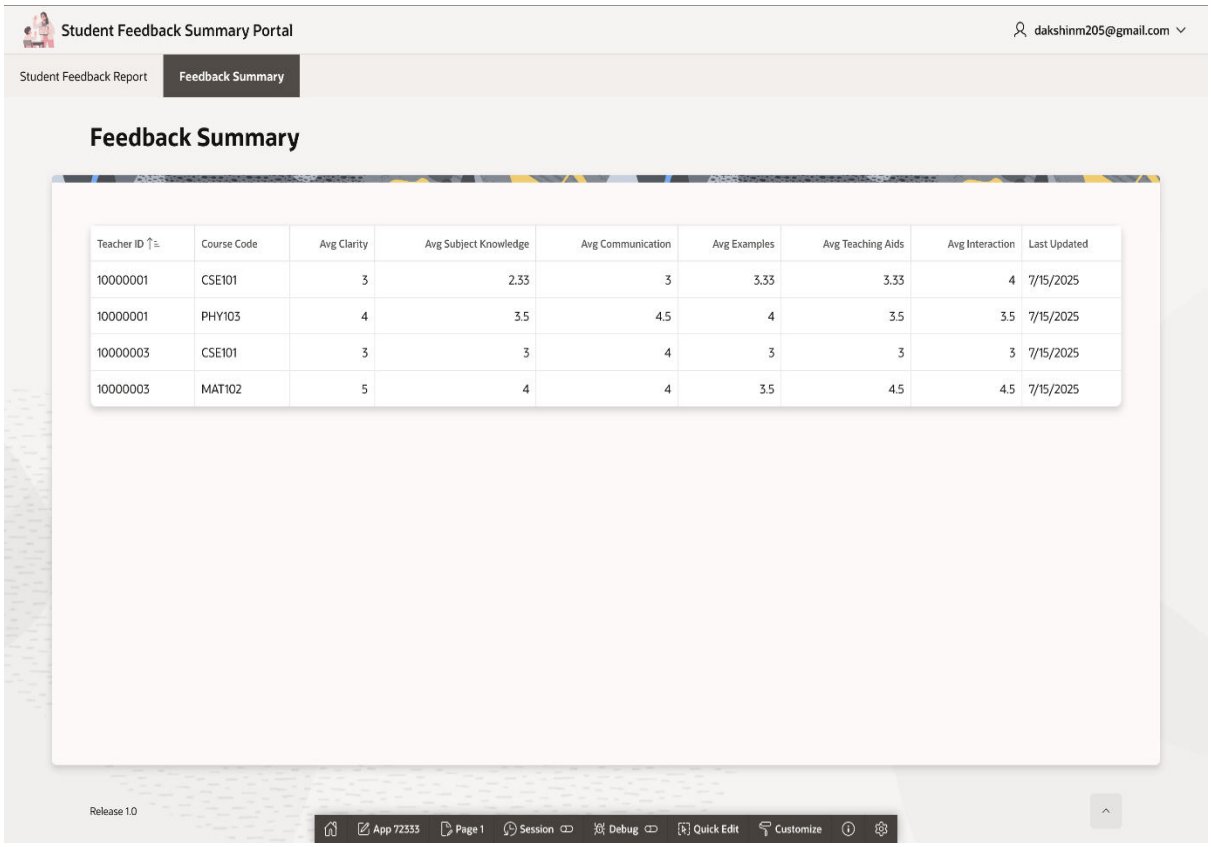
The Appendix section contains all relevant screenshots for visual reference, including the feedback form, confirmation message, and teacher rating summary. From user interaction to administrative analysis, these screens offer a comprehensive picture of the system's flow.

**Teacher Side Portal:**



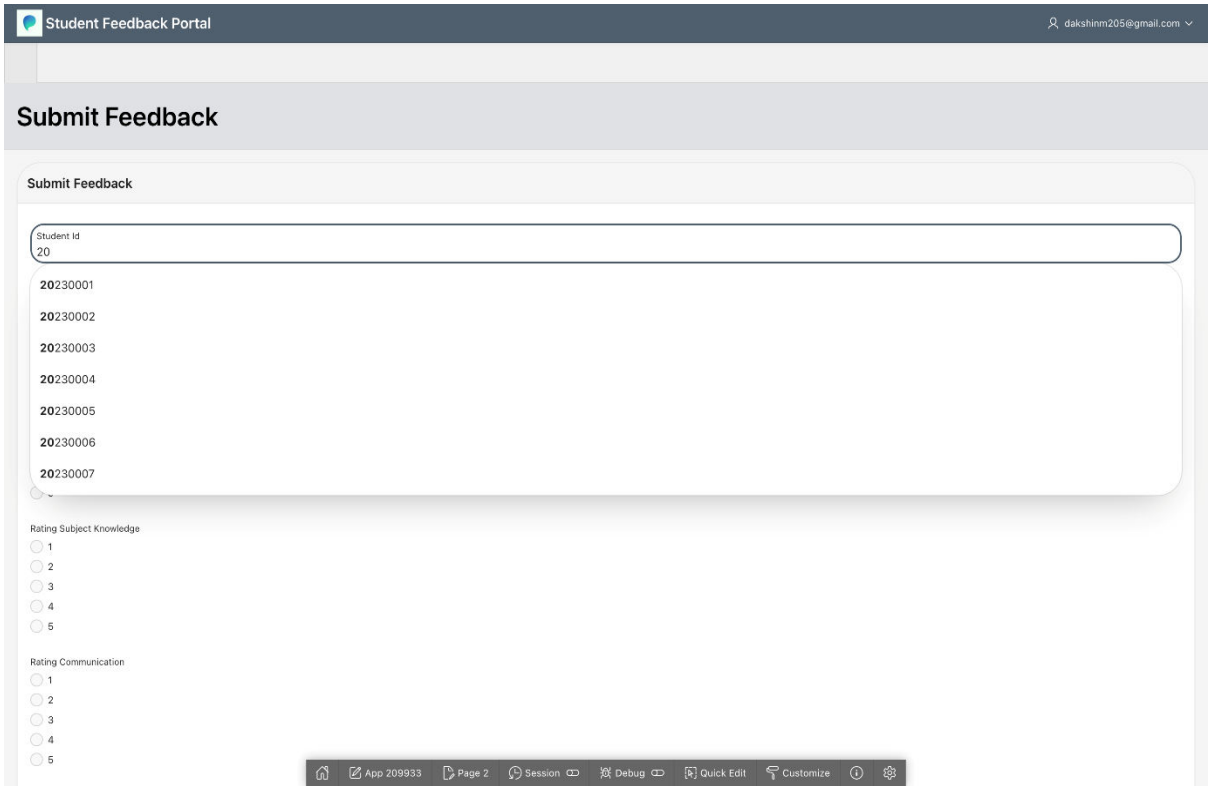

| Student | Teacher | Course Code | Rating Clarity | Rating Subject Knowledge | Rating Communication | Rating Examples | Rating Teaching Aids | Rating Interaction | Feedback Date |
|---|---|---|---|---|---|---|---|---|---|
| Anjali Sharma | Dr. Neha Singh | Intro to Computer Science | 2 | 3 | 3 | 2 | 4 | 4 | 7/15/2025 |
| Anjali Sharma | Dr. Kavita Jain | Engineering Mathematics | 5 | 3 | 3 | 2 | 4 | 4 | 7/15/2025 |
| Anjali Sharma | Dr. Neha Singh | Physics Fundamentals | 4 | 3 | 4 | 4 | 3 | 4 | 7/15/2025 |
| Rahul Verma | Dr. Kavita Jain | Intro to Computer Science | 3 | 3 | 4 | 3 | 3 | 3 | 7/15/2025 |
| Rahul Verma | Dr. Kavita Jain | Engineering Mathematics | 5 | 5 | 5 | 5 | 5 | 5 | 7/15/2025 |
| Priya Mehta | Dr. Neha Singh | Intro to Computer Science | 2 | 1 | 3 | 3 | 1 | 3 | 7/15/2025 |
| Priya Mehta | Dr. Neha Singh | Physics Fundamentals | 4 | 4 | 5 | 4 | 4 | 3 | 7/15/2025 |
| Shray Khanna | Dr. Neha Singh | Intro to Computer Science | 5 | 3 | 3 | 5 | 5 | 5 | 7/15/2025 |

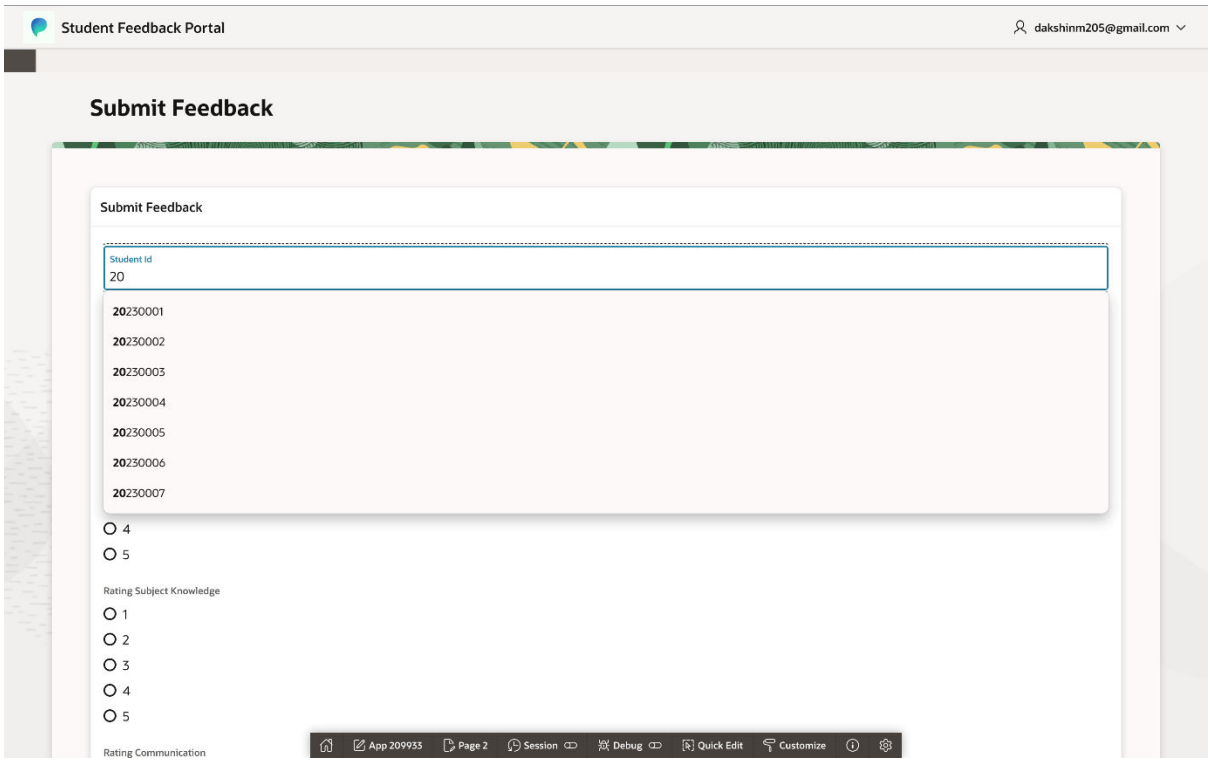
1 - 8

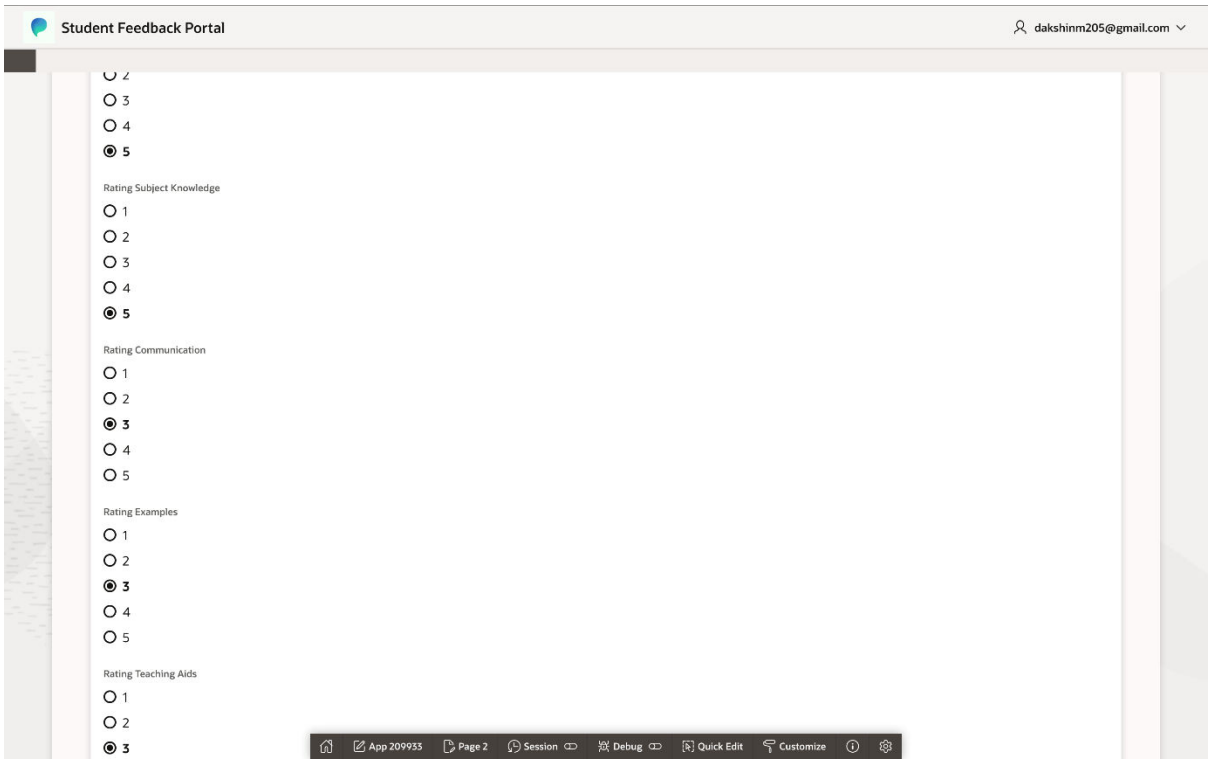


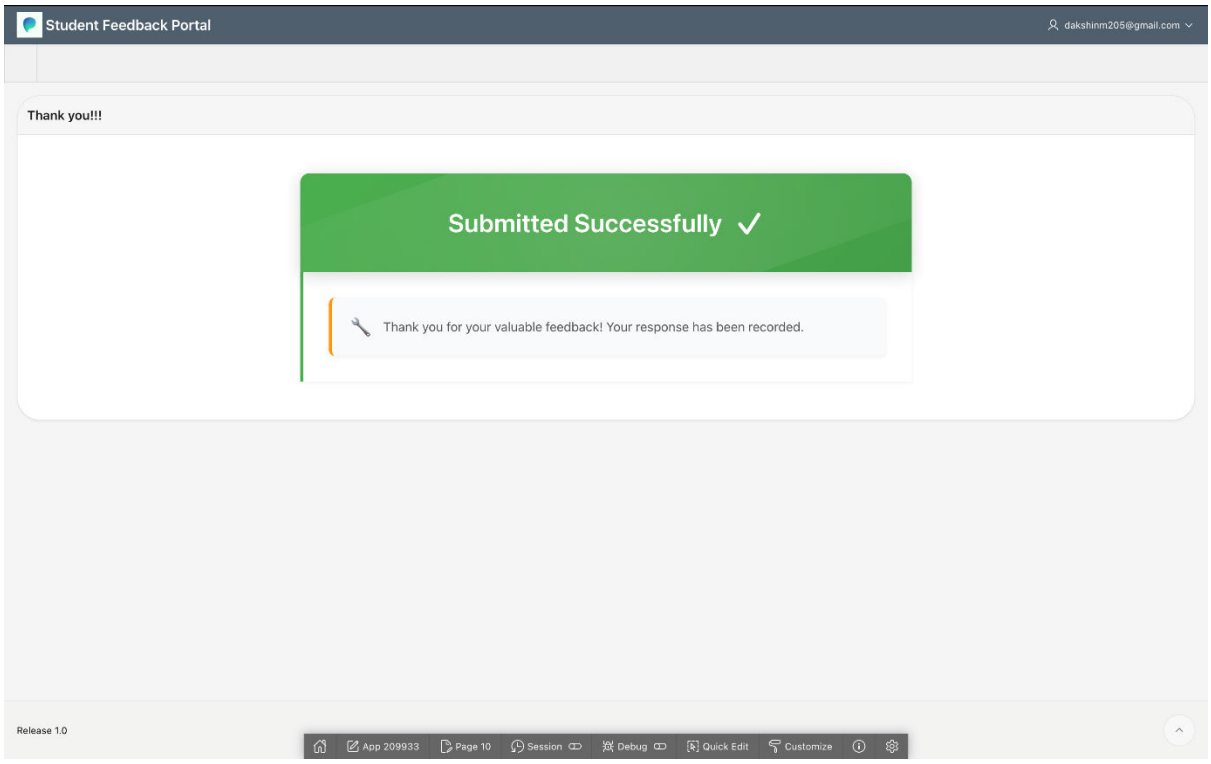

| Teacher ID | Course Code | Avg Clarity | Avg Subject Knowledge | Avg Communication | Avg Examples | Avg Teaching Aids | Avg Interaction | Last Updated |
|---|---|---|---|---|---|---|---|---|
| 10000001 | CSE101 | 3 | 2.33 | 3 | 3.33 | 3.33 | 4 | 7/15/2025 |
| 10000001 | PHY103 | 4 | 3.5 | 4.5 | 4 | 3.5 | 3.5 | 7/15/2025 |
| 10000003 | CSE101 | 3 | 3 | 4 | 3 | 3 | 3 | 7/15/2025 |
| 10000003 | MAT102 | 5 | 4 | 4 | 3.5 | 4.5 | 4.5 | 7/15/2025 |

# Student Side Portal:




Student Feedback Portal
dakshinm205@gmail.com
Submit Feedback
Submit Feedback
Student Id
20
20230001
20230002
20230003
20230004
20230005
20230006
20230007
4
5
Rating Subject Knowledge
1
2
3
4
5
Rating Communication
App 209933
Page 2
Session
Debug
Quick Edit
Customize


Student Feedback Portal
dakshinm205@gmail.com
3
4
5
Rating Subject Knowledge
1
2
3
4
5
Rating Communication
1
2
3
4
5
Rating Examples
1
2
3
4
5
Rating Teaching Aids
1
2
3
App 209933
Page 2
Session
Debug
Quick Edit
Customize


Student Feedback Portal
dakshinm205@gmail.com
Thank you!!!
Submitted Successfully
Thank you for your valuable feedback! Your response has been recorded.
Release 1.0
App 209933
Page 10
Session
Debug
Quick Edit
Customize

# 9. Conclusion

The process of gathering, storing, and evaluating student feedback is effectively streamlined and automated by the Student Feedback Management System. The system enables accurate validation, seamless data insertion, and quick retrieval of feedback summaries by utilizing PL/SQL procedures. This reduces the errors that come with manual data entry in addition to saving time. The system provides a methodical and structured way to handle student-teacher feedback, which can greatly enhance instructional quality and institutional effectiveness.

Learning how to use advanced PL/SQL features like error handling, cursors, triggers, and procedural logic was one of the biggest challenges encountered during development. Careful planning was also necessary to create a clear, standardized database schema that supports future integration and scalability. But conquering these obstacles aided in creating a strong

Overall, this project shows how PL/SQL can be used to create a useful application that can be implemented in academic settings. The system is scalable and a useful solution since it offers a solid basis for upcoming improvements like real-time analytics, authentication procedures, and AI-powered feedback analysis.

# 10. Future Scope

1. Include role-based access (student, teacher, administrator) and secure user authentication (login system).
2. Combine real-time feedback analytics with graphical dashboards.
3. Notify administrators and teachers automatically via SMS or email.
4. Feedback data can be exported to PDF and Excel formats.
5. Allow anonymous feedback submission to encourage truthful answers.
6. Apply sentiment analysis to feedback text using AI.
7. Create a version for a mobile app to facilitate access.
8. For greater accessibility, support a multilingual interface.
9. Include a chatbot to answer user questions.
10. Employ machine learning to identify trends and make recommendations for enhancements.

# 11. References


- MySQL Documentation : https://dev.mysql.com/doc/
- Oracle PL/SQL Documentation : https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/
- Oracle APEX Tutorials : https://apex.oracle.com/en/learn/documentation/
- GitHub : https://github.com/trancenoid/Student_Feedback_Management/blob/master/package_decl.sql