RV COLLEGE OF ENGINEERING ®

BENGALURU-560059

(Autonomous Institution Affiliated to VTU, Belagavi)

Faculty Feedback Management System

Report

Database Management Systems

(CD252IA)

Submitted By

Avinash Anish (1RV23IS145)

Mohammed Huzaif S (1RV23IS073)

Under the Guidance of

Padmashree T

Associate Professor

in partial fulfillment for the award of degree of

Bachelor of Engineering

in

INFORMATION SCIENCE AND ENGINEERING

2025-26

RV College of Engineering®

Mysore Road, RV Vidyaniketan Post, Bengaluru - 560059, Karnataka, India

CERTIFICATE

Certified that the project work titled ***"Faculty Feedback Management System"*** is carried out by **Avinash Anish (1RV23IS145)** and **Mohammed Huzaif S (1RV23IS073)** who are bonafide students of RV College of Engineering, Bengaluru, in partial fulfilment for the award of degree of **Bachelor of Engineering in Information Science and Engineering** of the Visvesvaraya Technological University, Belagavi during the academic year 2025-26. It is certified that all corrections/suggestions indicated for the Internal Assessment have been incorporated in the report deposited in the departmental library. The report has been approved as it satisfies the academic requirements in respect of experiential learning work prescribed by the institution for the said degree.

Padmashree T
Associate Professor
Department of ISE,
RVCE, Bengaluru-59

Dr Mamatha GS
Head of the Department
Department of ISE,
RVCE, Bengaluru-59

External Viva

Name of Examiners	**Signature with Date**
1.	
2.	

RV College of Engineering®

Mysore Road, RV Vidyaniketan Post, Bengaluru - 560059, Karnataka, India

DECLARATION

We, **Avinash Anish** and **Mohammed Huzaif S**, students of 5th semester B.E., Department of Information Science and Engineering, RV College of Engineering, Bengaluru, hereby declare that the Experiential Learning (EL) project titled **"Faculty Feedback Management System"** has been carried out by us and submitted in partial fulfillment for the award of degree of **Bachelor of Engineering in Information Science and Engineering** during the academic year 2025-26.

We also declare that any Intellectual Property Rights generated out of this project carried out at RVCE will be the property of RV College of Engineering, Bengaluru and we will be one of the authors of the same.

Place: Bengaluru

Date:

Name **Signature**

1. Avinash Anish (1RV23IS145)

2. Mohammed Huzaif S (1RV23IS073)

CONTENTS

INTRODUCTION

1.1 Terminology

The following terms and abbreviations are used throughout this report:

Term	Definition
RBAC	Role-Based Access Control – a method of regulating access to resources based on the roles of individual users within the system.
OAuth	Open Authorization – an open standard for access delegation, commonly used for token-based authentication.
OTP	One-Time Password – a password valid for a single login session or transaction.
PWA	Progressive Web Application – a web application that uses modern web technologies to deliver app-like experiences.
LLM	Large Language Model – an AI model trained on vast amounts of text data, capable of understanding and generating human-like text.
API	Application Programming Interface – a set of protocols and tools for building software applications.
NLP	Natural Language Processing – a branch of AI that enables computers to understand, interpret, and respond to human language.
USN	University Serial Number – a unique identifier assigned to each student in the institution.

Table 1.1: Terminology and Abbreviations

1.2 Purpose

The **Faculty Feedback Management System** is a web-based platform designed to automate student feedback collection and analysis at RV College of Engineering. The system enables faculty to dynamically create course-specific feedback forms and assign them to enrolled students. Students log in securely, submit feedback with complete anonymity, and receive confirmation. The platform automates the entire workflow, from form creation to AI-based result analysis, ensuring fairness, transparency, and efficiency in the feedback process.

1.3 Motivation

Currently, RVCE relies on platforms like Google Forms and Quiklrn for collecting student feedback. While these tools provide basic data collection capabilities, they present several limitations:

1. **Limited Analytics:** These platforms only provide raw data export without any automated analysis, summarization, or visualization of feedback trends.
2. **No AI-Powered Insights:** There is no capability for sentiment analysis or intelligent summarization of qualitative feedback, requiring manual effort to interpret open-ended responses.
3. **Weak Anonymity Guarantees:** Existing systems do not inherently guarantee student anonymity, potentially discouraging honest feedback.
4. **No Integrated Notifications:** Faculty must manually track submissions and send reminder emails, leading to lower response rates.
5. **Fragmented Workflow:** Form creation, distribution, collection, and analysis are disconnected processes requiring manual intervention at each step.

These limitations motivated the development of a dedicated system that integrates AI-powered analysis, guaranteed anonymity, and an automated end-to-end workflow.

1.4 Problem Statement

RVCE requires an efficient mechanism to collect, manage, and analyze student feedback for continuous improvement of teaching quality. The existing feedback collection through Google Forms and Quiklrn lacks:

- Automated sentiment analysis and summarization of feedback responses
- Guaranteed anonymity that encourages honest student participation
- Real-time analytics dashboards for faculty to monitor feedback trends
- AI assistance for creating effective feedback forms
- Integrated notification system for deadline reminders

There is a need for a unified platform that addresses these gaps while providing a seamless experience for administrators, faculty, and students.

1.5 Objectives

The objectives of the Faculty Feedback Management System are:

- To develop a web application with role-based access for administrators, faculty, and students.

- To implement AI-powered feedback analysis with sentiment detection and automated summarization.
- To ensure complete anonymity for students during feedback submission.
- To provide real-time analytics dashboards for faculty to visualize feedback trends.
- To automate the feedback workflow including form creation, distribution, and notifications.

1.6 Scope and Relevance

Scope

This project focuses on developing a feedback management system specifically for RVCE with the following boundaries:

- The system covers feedback collection for courses offered within the institution.
- Three user roles are supported: Administrator, Faculty, and Student.
- The platform is designed as a Progressive Web Application accessible on desktop and mobile devices.
- AI features are limited to form creation assistance, response summarization, and sentiment analysis.
- The system does not replace existing academic management systems but integrates as a standalone feedback platform.

Assumptions

- Users have access to devices with internet connectivity.
- Students have valid institutional USNs and email addresses for authentication.
- Faculty members and their course assignments are registered in the system by administrators.

Relevance

This project is relevant to RVCE as it:

- Addresses the current gaps in feedback collection and analysis processes.
- Provides actionable insights to faculty for improving teaching effectiveness.

- Encourages greater student participation through guaranteed anonymity and ease of access.
- Reduces administrative overhead by automating the feedback lifecycle.

REQUIREMENT SPECIFICATION

2.1 Specific Requirements

This section details the functional and non-functional requirements of the Faculty Feedback Management System. These requirements define the capabilities, constraints, and quality attributes that the system must satisfy.

2.1.1 Functional Requirements

Functional requirements describe the specific behaviors and functions that the system must perform. The requirements are organized by user role modules.

Administrator Module

The administrator has full control over system configuration and user management.

- The system shall allow administrators to create, update, and delete user accounts for faculty and students.
- The system shall provide a user approval workflow where new registrations require administrator approval before activation.
- The system shall allow administrators to define and manage academic structure including departments and courses.
- The system shall enable administrators to assign faculty members to specific courses.
- The system shall provide administrators with system-wide analytics and feedback reports across all departments.
- The system shall allow administrators to control feedback cycles by setting global deadlines and activation periods.
- The system shall provide administrators the ability to send announcements and notifications to all users.

Faculty Module

Faculty members can create feedback forms, distribute them to students, and analyze responses.

- The system shall allow faculty to create course-specific feedback forms with customizable question types (rating scale, multiple choice, text response).
- The system shall provide an AI-powered form builder that assists faculty in generating questions based on natural language prompts.

- The system shall allow faculty to assign feedback forms to specific student batches enrolled in their courses.
- The system shall enable faculty to set deadlines and activation periods for each feedback form.
- The system shall provide real-time tracking of submission status showing completed and pending responses.
- The system shall display an analytics dashboard with visualizations of feedback trends, ratings distribution, and comparative metrics.
- The system shall provide AI-powered summarization of qualitative feedback responses with sentiment analysis.
- The system shall allow faculty to export feedback reports in PDF format for academic reviews.

Student Module

Students can securely access and submit feedback for their enrolled courses.

- The system shall authenticate students using their USN and institutional email with OTP verification.
- The system shall display only active feedback forms for courses in which the student is enrolled.
- The system shall allow students to submit feedback responses with guaranteed anonymity.
- The system shall prevent duplicate submissions by tracking form completion status per student.
- The system shall send confirmation notifications upon successful feedback submission.
- The system shall send reminder notifications for pending feedback forms before deadlines.

AI Assistant Module

The AI assistant provides intelligent features for form creation and response analysis.

- The system shall provide a conversational AI interface for faculty to describe feedback requirements in natural language.
- The system shall generate structured feedback form questions based on AI interpretation of faculty requirements.

- The system shall analyze submitted text responses and provide sentiment classification (positive, negative, neutral).
- The system shall generate concise summaries of qualitative feedback highlighting key themes and suggestions.
- The system shall provide actionable insights and recommendations based on feedback analysis.

2.1.2 Non-Functional Requirements

Non-functional requirements specify the quality attributes and constraints under which the system must operate.

- **Performance:** The system shall handle a minimum of 500 concurrent users. Page load time shall not exceed 3 seconds, and API responses shall complete within 2 seconds.
- **Security:** All data transmission shall be encrypted using HTTPS/TLS. Role-based access control shall be enforced, and feedback responses shall be stored anonymously, decoupling student identity from responses.
- **Usability:** The interface shall be intuitive, responsive across devices (desktop, tablet, mobile), and installable as a Progressive Web Application (PWA).
- **Reliability:** The system shall maintain 99% uptime during academic sessions with daily automated backups and graceful error handling.
- **Maintainability:** The codebase shall follow modular architecture with the database designed in Third Normal Form (3NF) to ensure data integrity.

2.2 Hardware Requirements

The system requires standard computing resources suitable for web-based application deployment.

Component	Specification
Processor	2+ vCPUs (Intel/AMD x86_64)
RAM	4 GB minimum (8 GB recommended)
Storage	20 GB SSD (scalable based on data volume)
Network	100 Mbps stable internet connection

Table 2.1: Hardware Requirements

2.3 Software Requirements

The system is built using modern web technologies ensuring scalability, security, and developer productivity.

2.3.1 Development Environment

Category	Technology	Version
Runtime	Node.js	18.0 or higher
Package Manager	pnpm	8.0
Version Control	Git	2.40
IDE	Visual Studio Code	Latest

Table 2.2: Development Environment

2.3.2 Application Stack

Layer	Technology	Version	Purpose
Frontend Framework	Next.js	15.5.6	User interface and SSR
UI Components	Radix UI	1.2.x	Accessible component primitives
Styling	Tailwind CSS	4.0	Utility-first CSS framework
Data Fetching	TanStack React Query	5.90.9	Server state management
Backend/API	Next.js API Routes	15.5.6	Server-side logic and APIs
Database	PostgreSQL	15.0	Relational data storage
Database Client	Supabase JS	2.81.1	Backend-as-a-Service
Authentication	Supabase Auth	2.81.1	OAuth and OTP authentication
AI Integration	Google Gemini	2.0 Flash	LLM for analysis and generation
AI SDK	Vercel AI SDK	4.3.16	Streaming AI responses
Email Service	Resend	4.1.2	Transactional emails
Push Notifications	Web Push	3.6.7	Browser push notifications
Form Handling	React Hook Form	7.66.0	Form state management
Validation	Zod	3.23.8	Schema validation
Charts	Recharts	2.15.4	Data visualization
Animations	Framer Motion	12.23.24	UI animations

Table 2.3: Application Stack

2.3.3 Deployment

Component	Technology	Purpose
Hosting Platform	Vercel	Application deployment and serverless functions
Database Hosting	Supabase Cloud	Managed PostgreSQL database
CDN	Vercel Edge Network	Static asset delivery and caching
SSL Certificate	Automatic (Vercel)	HTTPS encryption

Table 2.4: Deployment Stack

DESIGN

3.1 E-R Diagram

Entity Relationship (ER) Diagram is a visual representation similar to a flowchart, depicting the relationships between ”entities” like people, objects, or concepts within a system. ER Diagrams are primarily utilized in designing or troubleshooting relational databases across various domains such as software engineering, business information systems, education, and research. Commonly referred to as ERDs or ER Models, these diagrams employ a set of symbols, including rectangles, diamonds, ovals, and connecting lines, to illustrate the connections between entities, relationships, and their attributes. They follow a structured format resembling grammar, with entities represented as nouns and relationships as verbs.

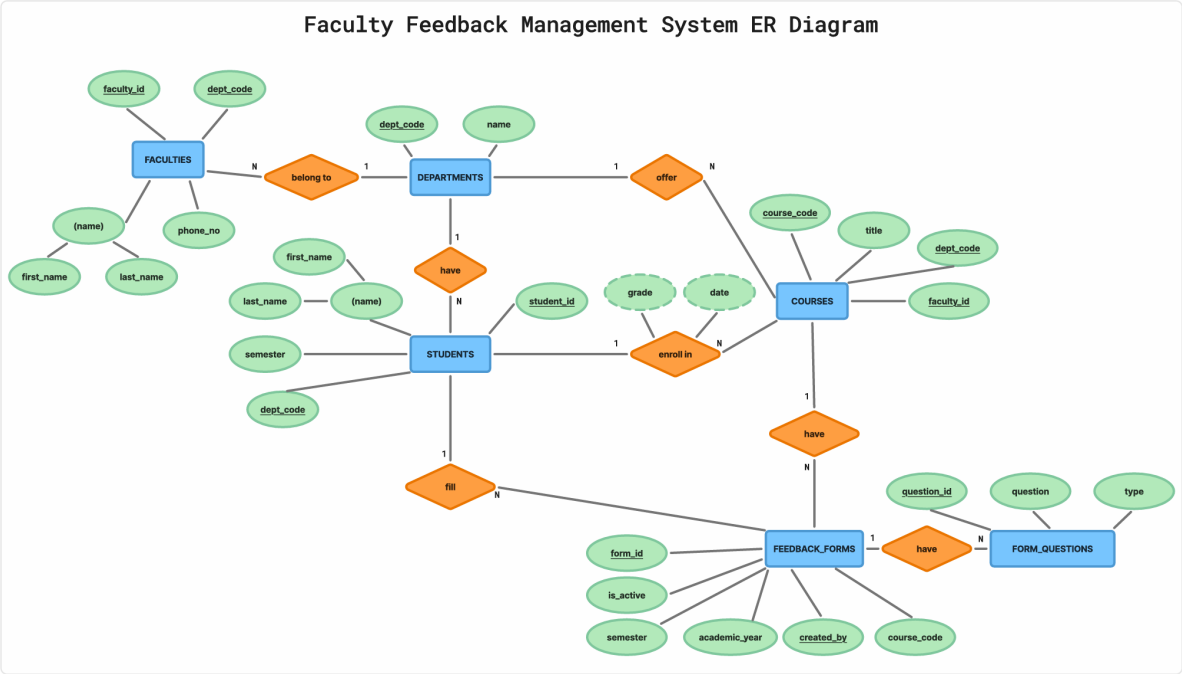

Figure 3.1: ER Diagram for Faculty Feedback Management System

Description of the ER diagram shown in Figure 3.1 is given as follows:

3.1.1 Entities and Attributes

1. **DEPARTMENTS**
   - Attributes:
     - `dept_code` (Primary Key)
     - `name`
2. **FACULTIES**

- Attributes:
  - `faculty_id` (Primary Key)
  - `dept_code` (Foreign Key referencing DEPARTMENTS)
  - `first_name`
  - `last_name`
  - `phone_no`

3. **STUDENTS**

- Attributes:
  - `student_id` (Primary Key)
  - `first_name`
  - `last_name`
  - `semester`
  - `dept_code` (Foreign Key referencing DEPARTMENTS)

4. **COURSES**

- Attributes:
  - `course_code` (Primary Key)
  - `title`
  - `dept_code` (Foreign Key referencing DEPARTMENTS)
  - `faculty_id` (Foreign Key referencing FACULTIES)

5. **FEEDBACK_FORMS**

- Attributes:
  - `form_id` (Primary Key)
  - `is_active`
  - `semester`
  - `academic_year`
  - `created_by`
  - `course_code` (Foreign Key referencing COURSES)

6. **FORM_QUESTIONS**

- Attributes:
  - `question_id` (Primary Key)
  - `question`
  - `type`

3.1.2 Relationships

1. **FACULTIES and DEPARTMENTS (belong to)**:
    - Many-to-One: Multiple faculty members can belong to one department, but each faculty member belongs to exactly one department.
2. **DEPARTMENTS and COURSES (offer)**:
    - One-to-Many: A department can offer multiple courses, but each course is offered by exactly one department.
3. **DEPARTMENTS and STUDENTS (have)**:
    - One-to-Many: A department can have multiple students enrolled, but each student belongs to exactly one department.
4. **STUDENTS and COURSES (enroll in)**:
    - Many-to-Many: A student can enroll in multiple courses, and a course can have multiple students enrolled. This relationship has additional attributes:
        - `grade` - The grade obtained by the student in the course
        - `date` - The date of enrollment
5. **STUDENTS and FEEDBACK_FORMS (fill)**:
    - One-to-Many: A student can fill multiple feedback forms, but each feedback submission is associated with one student.
6. **COURSES and FEEDBACK_FORMS (have)**:
    - One-to-Many: A course can have multiple feedback forms associated with it (for different semesters/years), but each feedback form is linked to exactly one course.
7. **FEEDBACK_FORMS and FORM_QUESTIONS (have)**:
    - One-to-Many: A feedback form can contain multiple questions, and questions can be reused across multiple forms.

3.1.3 Flow of Information

1. **Department Management:**
    - Departments form the organizational backbone of the system. Each department has a unique code and name, and serves as the parent entity for faculties, students, and courses.

2. **Faculty Management:**
    - Faculty members are registered under specific departments. Their personal information including name and contact details are stored for identification and communication purposes.
3. **Student Management:**
    - Students are enrolled in departments and their semester information is tracked. This allows the system to identify which courses a student should provide feedback for.
4. **Course Management:**
    - Courses are offered by departments and taught by faculty members. The relationship between courses and faculty enables the feedback system to link student responses to specific instructors.
5. **Enrollment and Grading:**
    - The enrollment relationship between students and courses tracks academic participation. Grade and date attributes ensure that feedback can be collected only from students who have completed the course.
6. **Feedback Collection:**
    - Feedback forms are created for specific courses and academic periods. The `is_active` flag controls whether a form is currently accepting responses. Students fill these forms to provide their evaluation of the course and instructor.
7. **Question Management:**
    - Form questions define the structure of feedback forms. The `type` attribute allows for different question formats (e.g., rating scale, multiple choice, text response), enabling comprehensive and structured feedback collection.
8. **Analytics and Reporting:**
    - Data from feedback submissions is aggregated to generate reports on faculty performance, course effectiveness, and overall student satisfaction. These insights help administration make informed decisions regarding curriculum and teaching quality.

3.2 Data Flow Diagram

A Data Flow Diagram (DFD) is a conventional visual tool that illustrates the flow of information within a system, providing a clear depiction of system requirements. It can encompass manual, automated, or mixed processes, showcasing how data enters and exits the system, undergoes

modifications, and where it's stored. The primary aim of a DFD is to delineate the scope and boundaries of a system comprehensively. DFDs are hierarchical, with each layer delving deeper into the system or data. Typically, these layers are represented as Levels 0 through 2, each progressively revealing more intricate details about the system's components and operations.

3.2.1 DFD Level 0

Figure 3.2 is a context diagram that provides a high-level overview of the system, showing interactions between external entities and the primary process (Faculty Feedback System).

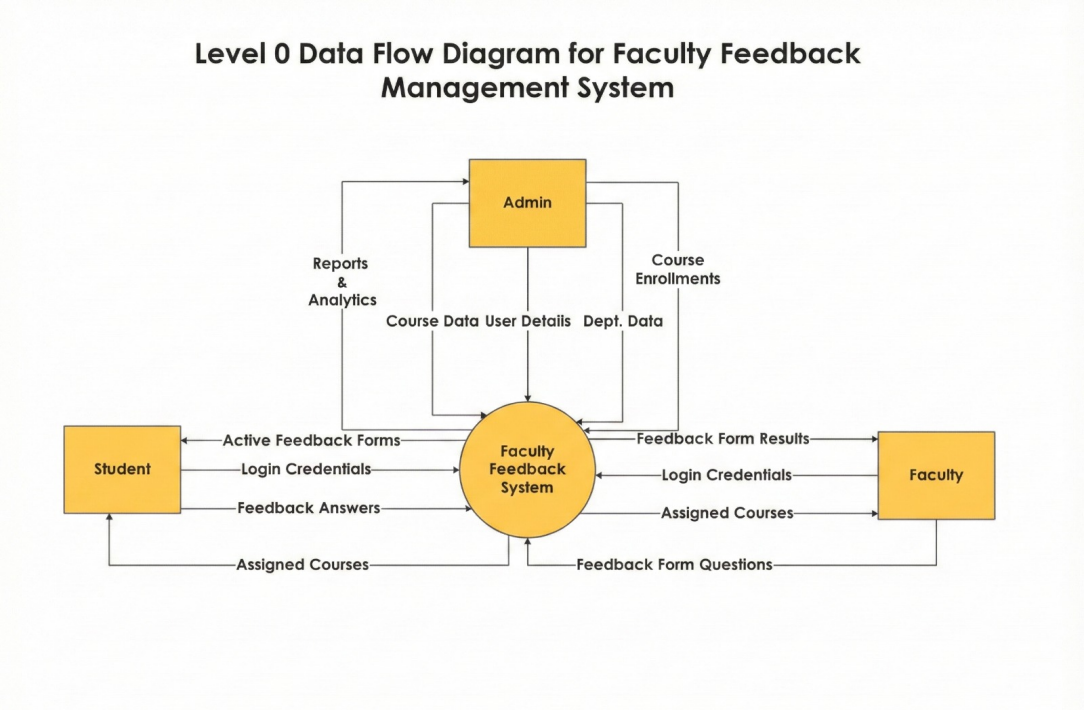

Figure 3.2: DFD Level 0 for Faculty Feedback Management System

1. **Student Interaction**
    - Students provide login credentials to authenticate with the system.
    - The system returns active feedback forms for courses the student is enrolled in.
    - Students submit feedback answers through the forms.
    - The system provides information about assigned courses to students.
2. **Faculty Interaction**
    - Faculty members provide login credentials for authentication.
    - The system returns feedback form results and analytics for their courses.
    - Faculty can view their assigned courses.
    - The system provides feedback form questions for review.

3. **Admin Interaction**
    - Admin provides course data, user details, and department data to the system.
    - Admin manages course enrollments for students.
    - The system generates reports and analytics for administrative review.

3.2.2 DFD Level 1

Figure 3.3 is a more detailed view, breaking the system into smaller, specific processes.

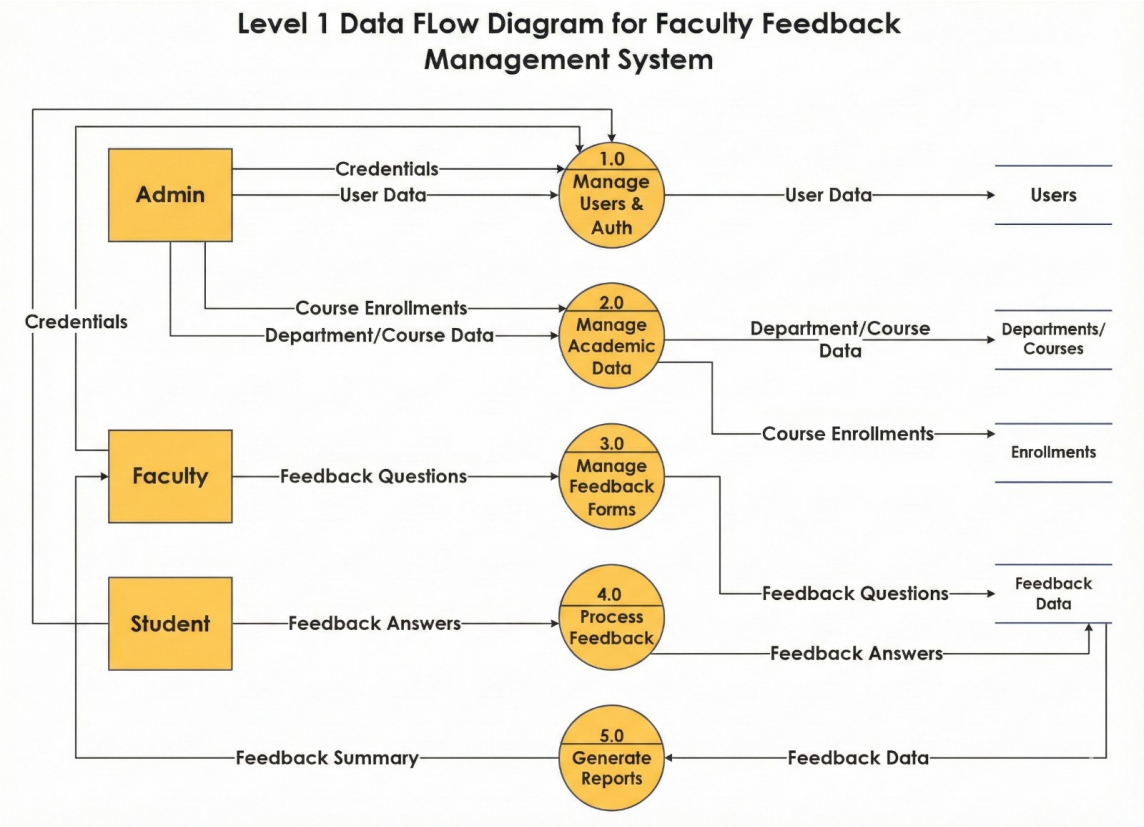

Figure 3.3: DFD Level 1 for Faculty Feedback Management System

1. **Process 1.1 - Manage Users & Auth**
    - Handles user authentication by receiving credentials from Admin, Faculty, and Students.
    - Manages user data including registration, profile updates, and role assignments.
    - Stores and retrieves user information from the Users data store.
2. **Process 1.2 - Manage Academic Data**
    - Admin provides course enrollments and department/course data.
    - The process manages the creation and modification of departments and courses.

- Data is stored in the Departments/Courses data store.
- Course enrollments are managed and stored in the Enrollments data store.

3. **Process 1.3 - Manage Feedback Forms**
   - Faculty can contribute feedback questions for their courses.
   - Admin creates and configures feedback forms with appropriate questions.
   - Feedback questions are stored and retrieved from the Feedback Data store.
4. **Process 1.4 - Process Feedback**
   - Students submit feedback answers through the active forms.
   - The process validates and stores feedback responses in the Feedback Data store.
   - Feedback questions are retrieved to display appropriate forms to students.
5. **Process 1.5 - Generate Reports**
   - Retrieves feedback data from the Feedback Data store.
   - Processes and aggregates feedback responses to generate meaningful insights.
   - Generates feedback summaries and analytical reports for Faculty and Admin review.

3.2.3 DFD Level 2

Figure 3.4 provides a detailed decomposition of the Level 1 processes, showing the sub-processes within each major function. Level 2 DFDs break down each Level 1 process into more granular operations, revealing the internal logic and data transformations.

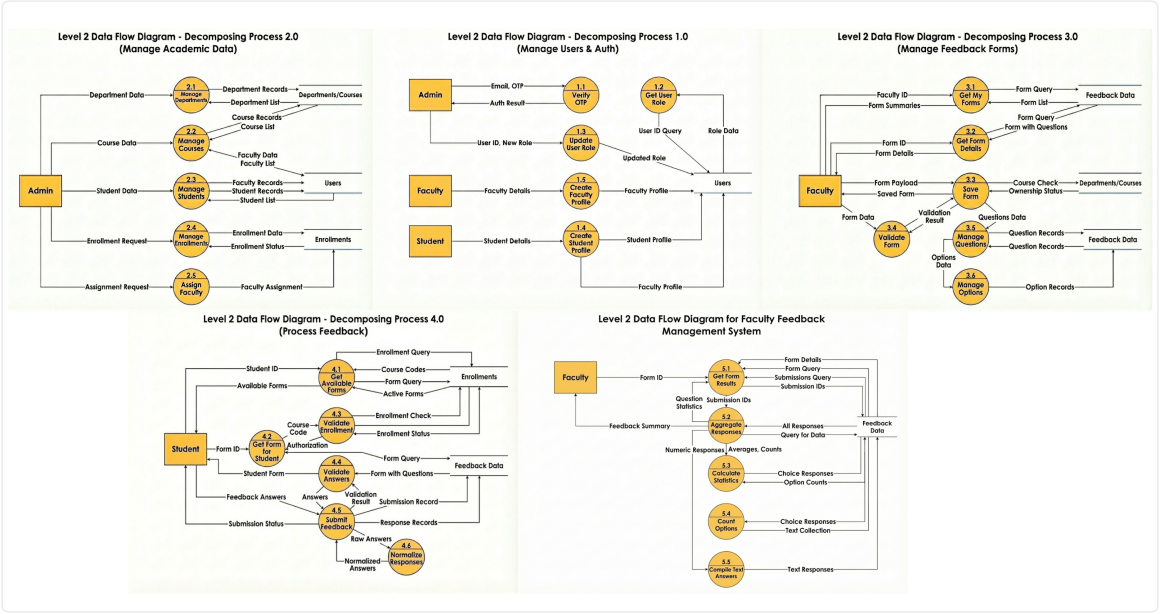

Figure 3.4: DFD Level 2 for Faculty Feedback Management System

1. **Process 1.0 - Manage Users & Auth (Sub-processes 1.1-1.5)**
   - **1.1 Verify OTP**: Verifies the one-time password sent to user email for authentication.
   - **1.2 Get User Role**: Retrieves the role (admin/faculty/student) of a user from the database.
   - **1.3 Update User Role**: Allows admin to update user roles in the system.
   - **1.4 Create Student Profile**: Creates a new student profile with semester and department info.
   - **1.5 Create Faculty Profile**: Creates a new faculty profile with department association.
2. **Process 2.0 - Manage Academic Data (Sub-processes 2.1-2.6)**
   - **2.1 Manage Departments**: CRUD operations for departments including students, courses, and faculty retrieval.
   - **2.2 Manage Courses**: Create, read, update, and delete courses with title and department association.
   - **2.3 Manage Faculty**: CRUD operations for faculty records with department assignment.
   - **2.4 Manage Students**: CRUD operations for student records with semester and department info.

- **2.5 Manage Enrollments**: Enroll and unenroll students in courses with date tracking.
- **2.6 Assign Faculty**: Assign faculty members to specific courses.

3. **Process 3.0 - Manage Feedback Forms (Sub-processes 3.1-3.6)**
   - **3.1 Get My Forms**: Faculty retrieves all feedback forms they have created.
   - **3.2 Get Form Details**: Retrieves a specific form with all questions and options.
   - **3.3 Save Form**: Creates new or updates existing feedback forms.
   - **3.4 Validate Form**: Validates form structure, required fields, and question configurations.
   - **3.5 Manage Questions**: Creates, updates, or deletes questions within a form.
   - **3.6 Manage Options**: Handles options for choice-type questions (radio, checkbox, dropdown).
4. **Process 4.0 - Process Feedback (Sub-processes 4.1-4.6)**
   - **4.1 Get Available Forms**: Retrieves all active feedback forms for enrolled courses.
   - **4.2 Get Form for Student**: Retrieves a specific form with questions for completion.
   - **4.3 Validate Enrollment**: Verifies that student is enrolled in the course.
   - **4.4 Validate Answers**: Validates that all required questions have been answered.
   - **4.5 Submit Feedback**: Creates submission record and stores all responses.
   - **4.6 Normalize Responses**: Normalizes response data (text trimming, number parsing).
5. **Process 5.0 - Generate Reports (Sub-processes 5.1-5.5)**
   - **5.1 Get Form Results**: Retrieves form details and all associated submissions.
   - **5.2 Aggregate Responses**: Coordinates aggregation of all response types into statistics.
   - **5.3 Calculate Statistics**: Computes statistics for numeric/rating type questions.
   - **5.4 Count Options**: Counts selections for each option in choice questions.
   - **5.5 Compile Text Answers**: Collects and compiles all text responses.

3.2.4 Data Stores

The system utilizes the following data stores as depicted in the Level 1 DFD:

1. **Users**: Stores information about all system users including students, faculty, and administrators along with their authentication credentials and roles.

2. **Departments/Courses**: Contains department information and course details including course codes, titles, and faculty assignments.
3. **Enrollments**: Maintains records of student enrollments in various courses, enabling the system to determine which feedback forms should be accessible to each student.
4. **Feedback Data**: Stores feedback form configurations, questions, and submitted responses. This is the central repository for all feedback-related information.

3.3 Relational Schema and Normalization

A relational schema is a structural framework that defines the organization of data in a relational database, detailing the logical design and relationships of the data. It specifies the tables that hold the data, the columns within those tables, the data types assigned to each column, and the constraints that govern the data. It also defines the primary keys, which uniquely identify each record within a table, and foreign keys, which create connections between tables by referencing the primary key of another table.

Normalization is the process of organizing data in a database. It includes creating tables and establishing relationships between those tables according to rules designed both to protect the data and to make the database more flexible by eliminating redundancy and inconsistent dependency.

Normalization works through a series of stages called Normal forms.

3.3.1 First Normal Form (1NF)

A table is in **1NF** if:

- It has a **primary key**.
- All attributes contain **atomic values**.
- Each column contains **values of a single type**.
- There are no **repeating groups** or **arrays**.

Example Violation (Before 1NF)
Consider the `FORM_QUESTIONS` table with options stored directly:

question_id	form_id	question_text	question_type	options
1	1	Rate teaching quality	rating	1,2,3,4,5
2	1	Select topics covered	mcq	A,B,C,D

- The column `options` contains multiple values (`1,2,3,4,5` or `A,B,C,D`).
- This **violates atomicity**, we need to separate options into another table `QUESTION_OPTIONS`.

Solution (After 1NF)

The `QUESTION_OPTIONS` table stores each option separately:

`FORM_QUESTIONS` (removes `options`):

question_id	form_id	question_text	question_type
1	1	Rate teaching quality	rating
2	1	Select topics covered	mcq

`QUESTION_OPTIONS` (stores options separately):

option_id	question_id	option_text	display_order
1	2	Option A	1
2	2	Option B	2
3	2	Option C	3
4	2	Option D	4

3.3.2 Second Normal Form (2NF)

A table is in **2NF** if:

- It is already in **1NF**.
- All non-key attributes are **fully functionally dependent** on the primary key (no partial dependencies).

Example Violation (Before 2NF)

Consider the `ENROLLMENTS` table with course details included:

enrollment_id	student_id	course_code	grade	enrolled_date	course_title
1	1RV23IS145	CD252IA	A	15-08-25	Database
2	1RV23IS146	CS354TA	B	16-08-25	ToC
3	1RV23IS147	CD252IA	A	15-08-25	Database

- The `course_title` depends only on `course_code`, not on the primary key `enrollment_id`.
- This is a **partial dependency** and must be removed.

Solution (After 2NF)

Move `course_title` to a separate `COURSES` table:

`ENROLLMENTS` (removes `course_title`):

enrollment_id	student_id	course_code	grade	enrolled_date
1	1RV23IS145	CD252IA	A	15-08-25
2	1RV23IS146	CS354TA	B	16-08-25
3	1RV23IS147	CD252IA	A	15-08-25

COURSES (stores title separately):

course_code	title	department_code	faculty_id
CD252IA	Database	ISE	F001
CS354TA	ToC	ISE	F002

3.3.3 Third Normal Form (3NF)

A table is in **3NF** if:

- It is already in **2NF**.
- There are no **transitive dependencies** (non-key attributes should not depend on other non-key attributes).

Example Violation (Before 3NF)
Consider the FACULTIES table:

faculty_id	user_id	first_name	last_name	department_code	dept_name
F001	U001	Padmashree	T	ISE	Information Science
F002	U002	Anala	M	ISE	Information Science

- dept_name depends on department_code, not directly on faculty_id.
- This is a **transitive dependency**: faculty_id → department_code → dept_name.

Solution (After 3NF)
Move dept_name to a separate DEPARTMENTS table:
FACULTIES (removes dept_name):

faculty_id	user_id	first_name	last_name	department_code
F001	U001	Padmashree	T	ISE
F002	U002	Anala	M	ISE

DEPARTMENTS (stores name separately):

department_code	name
ISE	Information Science
CSE	Computer Science

Now, there are **no transitive dependencies**, and the schema is in **3NF**.

3.3.4 Boyce-Codd Normal Form (BCNF)

A table is in **BCNF** if:

- It is already in **3NF**.
- For every functional dependency $X \rightarrow Y$, X is a superkey.

There are no dependencies where a non-superkey attribute determines another attribute in our schema. All determinants are superkeys.

3.3.5 Final Normalization Summary

Normal Form	Issue Fixed	Example Change
1NF	Remove repeating groups	Moved options to QUESTION_OPTIONS
2NF	Remove partial dependencies	Split course details into COURSES table
3NF	Remove transitive dependencies	Created DEPARTMENTS table
BCNF	Ensure all determinants are superkeys	Schema already satisfies BCNF

The schema is fully normalized up to 3NF and BCNF.

3.3.6 Schema after Normalization

The normalized schema for the Faculty Feedback Management System is shown in Figure 3.5.

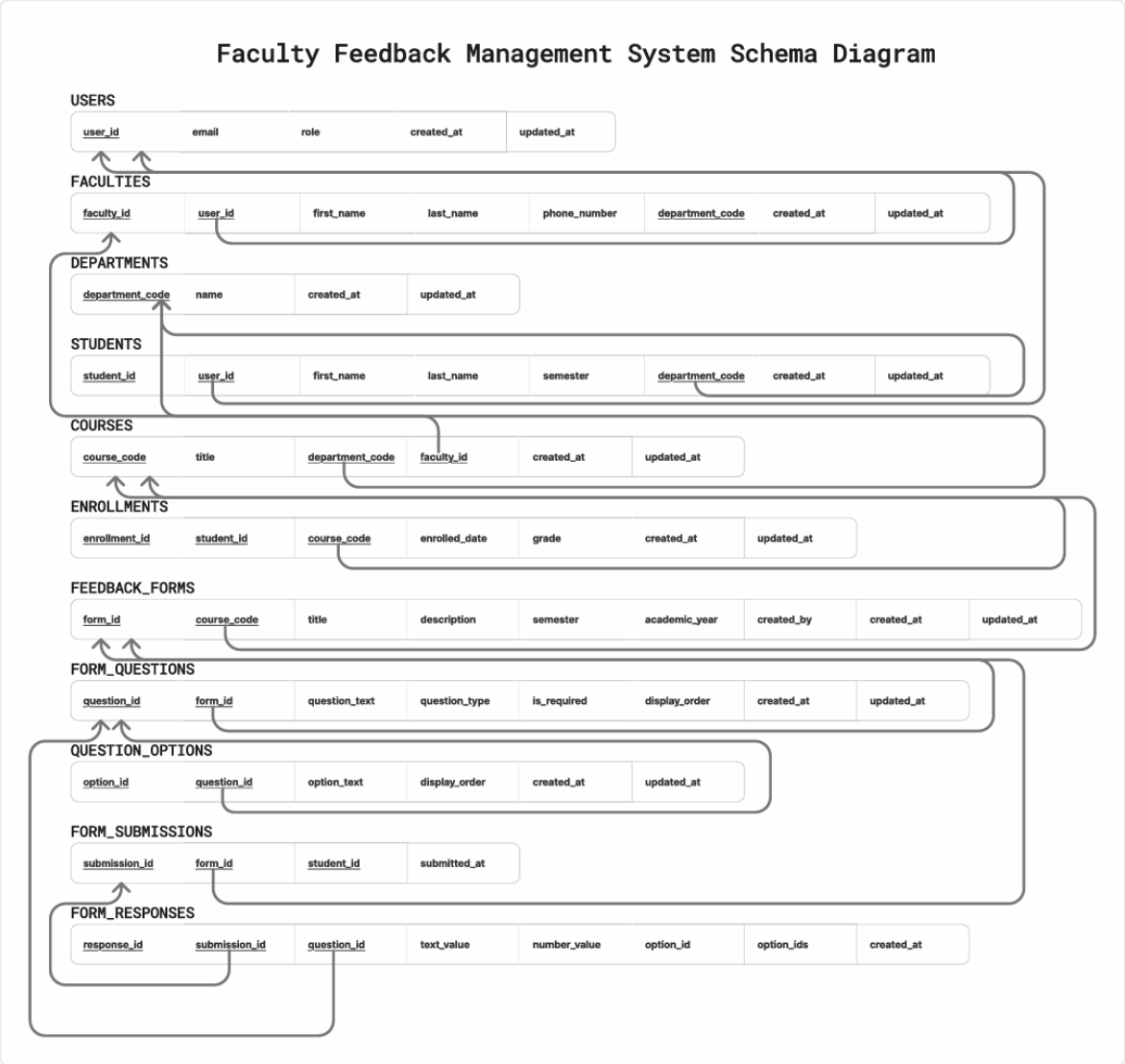

Figure 3.5: Normalized Relational Schema for Faculty Feedback Management System

3.4 Front End Design

The front-end of the Faculty Feedback Management System is designed with a focus on usability, accessibility, and modern aesthetics. The user interface caters to three primary user roles: Students, Faculty, and Administrators.

3.4.1 Design Principles

1. **Responsive Design:** The interface adapts seamlessly across different screen sizes, from desktop monitors to mobile devices.

2. **Intuitive Navigation:** Clear navigation menus and breadcrumbs ensure users can easily find and access features.

3. **Consistent Visual Language:** A unified color scheme, typography, and component styling maintains visual coherence throughout the application.

3.4.2 User Interface Components

1. **Login/Authentication Module:**
    - Secure login page with role-based authentication
    - Password recovery functionality
    - Session management
2. **Student Dashboard:**
    - List of enrolled courses
    - Pending feedback forms
    - Feedback submission history
    - Profile management
3. **Faculty Dashboard:**
    - Courses taught overview
    - Aggregated feedback reports
    - Performance analytics and trends
4. **Administrator Dashboard:**
    - User management (students, faculty)
    - Department and course management
    - Feedback form creation and configuration
    - System-wide analytics and reports
5. **Feedback Form Interface:**
    - Dynamic form rendering based on question types
    - Rating scales with visual indicators
    - Text input areas for qualitative feedback
    - Progress indicators for multi-page forms

3.4.3 Navigation Flow

The application follows a hierarchical navigation structure:

- **Primary Navigation:** Role-specific sidebar menu providing access to main features
- **Secondary Navigation:** Contextual tabs and sub-menus within each section
- **Quick Actions:** Floating action buttons and shortcuts for frequently used operations

IMPLEMENTATION DETAILS

4.1 Database Implementation

This section describes the implementation of the database layer for the Faculty Feedback Management System. The system uses PostgreSQL as the database management system, hosted on Supabase, a cloud-based backend-as-a-service platform that provides real-time database functionality.

4.1.1 Table Creation

The database schema consists of 12 tables organized into three logical groups: authentication, core entities, and dynamic form builder tables. Below are the SQL queries used to create all the tables.

```
CREATE TABLE users (
    user_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    email VARCHAR(255) UNIQUE NOT NULL,
    role VARCHAR(20) NOT NULL CHECK (role IN ('admin', 'faculty', '
       student')),
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW()
);
```

Create Table Query for users

```
CREATE TABLE departments (
    dept_code VARCHAR(10) PRIMARY KEY,
    dept_name VARCHAR(100) NOT NULL UNIQUE,
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW()
);
```

Create Table Query for departments

```
CREATE TABLE faculties (
    faculty_id VARCHAR(20) PRIMARY KEY,
    user_id UUID UNIQUE NOT NULL,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    phone_no VARCHAR(20),
    dept_code VARCHAR(10) NOT NULL,
```

```
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW(),
    FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE
       CASCADE,
    FOREIGN KEY (dept_code) REFERENCES departments(dept_code)
);
```

Create Table Query for faculties

```
CREATE TABLE courses (
    course_code VARCHAR(20) PRIMARY KEY,
    title VARCHAR(200) NOT NULL,
    dept_code VARCHAR(10) NOT NULL,
    faculty_id VARCHAR(20) NOT NULL,
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW(),
    FOREIGN KEY (dept_code) REFERENCES departments(dept_code),
    FOREIGN KEY (faculty_id) REFERENCES faculties(faculty_id)
);
```

Create Table Query for courses

```
CREATE TABLE students (
    student_id VARCHAR(20) PRIMARY KEY,
    user_id UUID UNIQUE NULL,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    semester INT NOT NULL,
    dept_code VARCHAR(10) NOT NULL,
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW(),
    FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE
       CASCADE,
    FOREIGN KEY (dept_code) REFERENCES departments(dept_code)
);
```

Create Table Query for students

```
CREATE TABLE enrollments (
    enrollment_id SERIAL PRIMARY KEY,
    student_id VARCHAR(20) NOT NULL,
    course_code VARCHAR(20) NOT NULL,
```

```
    enrolled_date DATE NOT NULL,
    grade INT NULL,
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW(),
    FOREIGN KEY (student_id) REFERENCES students(student_id) ON
       DELETE CASCADE,
    FOREIGN KEY (course_code) REFERENCES courses(course_code),
    UNIQUE (student_id, course_code)
);
```

Create Table Query for enrollments

```
CREATE TABLE feedback_forms (
    form_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    course_code VARCHAR(20) NOT NULL,
    title VARCHAR(200) NOT NULL,
    description TEXT,
    semester INT NOT NULL,
    academic_year INT NOT NULL,
    created_by UUID NOT NULL,
    is_active BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW(),
    FOREIGN KEY (course_code) REFERENCES courses(course_code),
    FOREIGN KEY (created_by) REFERENCES users(user_id)
);
```

Create Table Query for feedback_forms

```
CREATE TABLE form_questions (
    question_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    form_id UUID NOT NULL,
    question_text VARCHAR(500) NOT NULL,
    question_type VARCHAR(20) NOT NULL CHECK (
        question_type IN (
            'text',
            'textarea',
            'email',
            'number',
            'checkbox',
            'radio',
            'dropdown',
```

```
            'rating'
        )
    ),
    is_required BOOLEAN DEFAULT FALSE,
    display_order INT NOT NULL,
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW(),
    FOREIGN KEY (form_id) REFERENCES feedback_forms(form_id) ON
       DELETE CASCADE
);
```

Create Table Query for form_questions

```
CREATE TABLE question_options (
    option_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    question_id UUID NOT NULL,
    option_text VARCHAR(255) NOT NULL,
    display_order INT NOT NULL,
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW(),
    FOREIGN KEY (question_id) REFERENCES form_questions(question_id
       ) ON DELETE CASCADE
);
```

Create Table Query for question_options

```
CREATE TABLE feedback_submissions (
    submission_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    form_id UUID NOT NULL,
    student_id VARCHAR(20) NOT NULL,
    submitted_at TIMESTAMPTZ DEFAULT NOW(),
    FOREIGN KEY (form_id) REFERENCES feedback_forms(form_id) ON
       DELETE CASCADE,
    FOREIGN KEY (student_id) REFERENCES students(student_id) ON
       DELETE CASCADE,
    UNIQUE (form_id, student_id)
);
```

Create Table Query for feedback_submissions

```
CREATE TABLE feedback_responses (
    response_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
```

```
    submission_id UUID NOT NULL,
    question_id UUID NOT NULL,
    text_value TEXT NULL,
    number_value DOUBLE PRECISION NULL,
    option_id UUID NULL,
    option_ids UUID[] NULL,
    created_at TIMESTAMPTZ DEFAULT NOW(),
    FOREIGN KEY (submission_id) REFERENCES feedback_submissions(
       submission_id) ON DELETE CASCADE,
    FOREIGN KEY (question_id) REFERENCES form_questions(question_id
       ) ON DELETE CASCADE
);
```

Create Table Query for feedback_responses

```
CREATE TABLE push_subscriptions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID NOT NULL,
    endpoint TEXT NOT NULL,
    p256dh TEXT NOT NULL,
    auth TEXT NOT NULL,
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW(),
    FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE
       CASCADE,
    UNIQUE (user_id, endpoint)
);
```

Create Table Query for push_subscriptions

4.1.2 Table Population

The following queries demonstrate how data is inserted into the database tables.

```
INSERT INTO users (user_id, email, role) VALUES
  (gen_random_uuid(), 'avinashanish.is23@rvce.edu.in', 'student'),
  (gen_random_uuid(), 'mdhuzaifs.is23@rvce.edu.in', 'student'),
  (gen_random_uuid(), 'padmashreet@rvce.edu.in', 'faculty');
  (gen_random_uuid(), 'controllerofexaminations@rvce.edu.in', '
     admin');
```

Insert Queries for users

```
INSERT INTO departments (dept_code, dept_name) VALUES
  ('CS', 'Computer Science and Engineering'),
  ('IS', 'Information Science and Engineering'),
  ('CD', 'Computer Science and Engineering - DS');
```

Insert Queries for departments

```
INSERT INTO faculties (faculty_id, user_id, first_name, last_name,
   phone_no, dept_code) VALUES
  ('FAC001', (SELECT user_id FROM users WHERE email = '
     padmashreet@rvce.edu.in'), 'Padmashree', 'T', '9876543210', '
     IS')
```

Insert Queries for faculties

```
INSERT INTO courses (course_code, title, dept_code, faculty_id)
   VALUES
  ('CD252IA', 'Database Management Systems', 'CD', 'FAC001'),
  ('CS354TA', 'Theory of Computation', 'CD', 'FAC001');
```

Insert Queries for courses

```
INSERT INTO students (student_id, user_id, first_name, last_name,
   semester, dept_code) VALUES
  ('1RV23IS145', (SELECT user_id FROM users WHERE email = '
     avinashanish.is23@rvce.edu.in'), 'Avinash', 'Anish', 5, 'IS'),
  ('1RV23IS073', (SELECT user_id FROM users WHERE email = '
     mdhuzaifs.is23@rvce.edu.in'), 'Mohammed', 'Huzaif', 5, 'IS'),
```

Insert Queries for students

```
INSERT INTO enrollments (student_id, course_code, enrolled_date,
   grade) VALUES
  ('1RV23IS145', 'CD252IA', '2025-08-01', NULL),
  ('1RV23IS073', 'CS354TA', '2025-08-01', NULL);
```

Insert Queries for enrollments

4.1.3 Query Execution and Output

This section outlines the SQL queries used for various operations within the Faculty Feedback System, categorized by their functional modules.

Authentication & User Management

```
SELECT role FROM users WHERE user_id = '4cf9d93c-d5e0-4561-b121-431
    d91ad0fc9';
```

Query to retrieve user role

role
student

Table 4.1: Output: User Role

```
UPDATE users SET role = 'faculty' WHERE user_id = '4cf9d93c-d5e0
    -4561-b121-431d91ad0fc9';
```

Query to update user role

Profile Management

```
INSERT INTO students (student_id, user_id, first_name, last_name,
    semester, dept_code)
VALUES ('1RV23IS145', '4cf9d93c-d5e0-4561-b121-431d91ad0fc9', '
    Avinash', 'Anish', 5, 'IS');
```

Query to create student profile

```
INSERT INTO faculties (faculty_id, user_id, first_name, last_name,
    phone_no, dept_code)
VALUES ('FAC001', '7a8b9c0d-1e2f-4a3b-8c9d-0e1f2a3b4c5d', '
    Padmashree', 'T', '9876543210', 'IS');
```

Query to create faculty profile

Admin Dashboard

```
SELECT * FROM courses ORDER BY course_code ASC;
```

Query to list all courses

course_code	title	dept_code	faculty_id
CD252IA	Database Management Systems	CD	FAC001
CS354TA	Theory of Computation	CD	FAC001

Table 4.2: Output: List All Courses

```
SELECT * FROM students WHERE dept_code = 'IS' ORDER BY student_id
   ASC;
```

Query to get department students

student_id	first_name	last_name	semester
1RV23IS073	Mohammed	Huzaif	5
1RV23IS145	Avinash	Anish	5

Table 4.3: Output: Department Students

```
UPDATE courses SET faculty_id = 'FAC001' WHERE course_code = '
   CS354TA';
```

Query to assign faculty to course

```
UPDATE students
SET first_name = 'Avinash', last_name = 'A', semester = 6
WHERE student_id = '1RV23IS145';
```

Query to update student details

```
DELETE FROM courses WHERE course_code = 'CS354TA';
```

Query to delete a course

Faculty Dashboard & Feedback Management

```
SELECT * FROM courses
WHERE faculty_id = (SELECT faculty_id FROM faculties WHERE user_id
   = '7a8b9c0d-1e2f-4a3b-8c9d-0e1f2a3b4c5d')
```

```
ORDER BY course_code ASC;
```

Query to get courses assigned to faculty

course_code	title	dept_code	faculty_id
CD252IA	Database Management Systems	CD	FAC001
CS354TA	Theory of Computation	CD	FAC001

Table 4.4: Output: Faculty Courses

```
-- 1. Create the form header
INSERT INTO feedback_forms (form_id, course_code, title,
   description, semester, academic_year, created_by, is_active)
VALUES ('f47ac10b-58cc-4372-a567-0e02b2c3d479', 'CD252IA', 'Mid-
   Term Feedback', 'Please provide honest feedback', 5, 2025, '7
   a8b9c0d-1e2f-4a3b-8c9d-0e1f2a3b4c5d', TRUE);

-- 2. Add questions to the form
INSERT INTO form_questions (question_id, form_id, question_text,
   question_type, is_required, display_order)
VALUES ('7b8c9d0e-1a2b-3c4d-5e6f-7a8b9c0d1e2f', 'f47ac10b-58cc
   -4372-a567-0e02b2c3d479', 'How is the teaching pace?', 'rating',
    TRUE, 1);

-- 3. Add options for multiple choice questions
INSERT INTO question_options (option_id, question_id, option_text,
   display_order)
VALUES ('3d4e5f6a-7b8c-9d0e-1a2b-3c4d5e6f7a8b', '7b8c9d0e-1a2b-3c4d
   -5e6f-7a8b9c0d1e2f', 'Excellent', 1), ('5f6a7b8c-9d0e-1a2b-3c4d
   -5e6f7a8b9c0d', '7b8c9d0e-1a2b-3c4d-5e6f-7a8b9c0d1e2f', 'Good',
   2);
```

Queries to create feedback form, questions, and options

```
-- Get total submission count
SELECT COUNT(*) FROM feedback_submissions WHERE form_id = 'f47ac10b
   -58cc-4372-a567-0e02b2c3d479';

-- Get all responses for analysis
SELECT r.question_id, r.text_value, r.number_value, r.option_id, r.
   option_ids
FROM feedback_responses r
```

```
JOIN feedback_submissions s ON r.submission_id = s.submission_id
WHERE s.form_id = 'f47ac10b-58cc-4372-a567-0e02b2c3d479';
```

Queries to view form statistics and responses

question_id	text_value	number_value	option_id
7b8c9d0e-1a2b-..	NULL	4.5	NULL
1a2b3c4d-5e6f-..	"Great course!"	NULL	NULL

Table 4.5: Output: Feedback Responses

Student Dashboard & Feedback Participation

```
SELECT e.course_code, e.enrolled_date, e.grade, c.title, c.
   dept_code
FROM enrollments e
JOIN courses c ON e.course_code = c.course_code
WHERE e.student_id = '1RV23IS145'
ORDER BY e.enrolled_date DESC;
```

Query to get student's enrolled courses

course_code	enrolled_date	grade	title	dept_code
CD252IA	2025-01-10	NULL	DBMS	CD
CS354TA	2025-01-12	NULL	TOC	CD

Table 4.6: Output: Student Enrollments

```
SELECT f.form_id, f.title,
       (SELECT COUNT(*) FROM form_questions q WHERE q.form_id = f.
          form_id) as question_count,
       EXISTS (SELECT 1 FROM feedback_submissions s WHERE s.form_id
           = f.form_id AND s.student_id = '1RV23IS145') as
          is_submitted
FROM feedback_forms f
WHERE f.course_code IN (SELECT course_code FROM enrollments WHERE
   student_id = '1RV23IS145')
AND f.is_active = TRUE;
```

Query to fetch active feedback forms for a student

form_id	title	question_count	is_submitted
f47ac10b-58cc-..	Mid-Term Feedback	5	false

Table 4.7: Output: Active Feedback Forms

```
-- 1. Record the submission
INSERT INTO feedback_submissions (submission_id, form_id,
   student_id)
VALUES ('9e0f1a2b-3c4d-5e6f-7a8b-9c0d1e2f3a4b', 'f47ac10b-58cc
   -4372-a567-0e02b2c3d479', '1RV23IS145');

-- 2. Record individual responses
INSERT INTO feedback_responses (submission_id, question_id,
   number_value, text_value)
VALUES
('9e0f1a2b-3c4d-5e6f-7a8b-9c0d1e2f3a4b', '7b8c9d0e-1a2b-3c4d-5e6f-7
   a8b9c0d1e2f', 5, NULL),
('9e0f1a2b-3c4d-5e6f-7a8b-9c0d1e2f3a4b', '1a2b3c4d-5e6f-7a8b-9c0d-1
   e2f3a4b5c6d', NULL, 'The lab sessions are very helpful.');
```

Queries to submit feedback responses

Notifications

```
INSERT INTO push_subscriptions (user_id, endpoint, p256dh, auth)
VALUES ('8a3b8c2d-1e4f-4a5b-9c6d-7e8f9a0b1c2d', 'https://fcm.
   googleapis.com/...', 'p256-key', 'auth-key')
ON CONFLICT (user_id, endpoint)
DO UPDATE SET p256dh = EXCLUDED.p256dh, auth = EXCLUDED.auth;
```

Query to upsert push notification subscription

```
SELECT endpoint, p256dh, auth FROM push_subscriptions WHERE user_id
    = '8a3b8c2d-1e4f-4a5b-9c6d-7e8f9a0b1c2d';
```

Query to retrieve user push subscriptions

endpoint	p256dh	auth
https://fcm...	p256-key...	auth-key...

Table 4.8: Output: User Subscriptions

4.1.4 Security Features

The database implementation incorporates several security measures:

1. **UUID Primary Keys:** All primary keys use UUIDs generated by `gen_random_uuid()`, making them unpredictable and resistant to enumeration attacks.
2. **Role-Based Access Control:** The `users` table enforces role constraints using CHECK constraints, limiting roles to 'admin', 'faculty', or 'student'.
3. **Referential Integrity:** Foreign key constraints ensure data consistency across related tables. CASCADE delete operations are used where appropriate to maintain data integrity.
4. **Unique Constraints:** Strategic unique constraints prevent duplicate entries, such as:
   - Unique email addresses in the `users` table
   - Unique student-course combinations in `enrollments`
   - Unique form-student submissions in `feedback_submissions`
5. **Automatic Timestamps:** All tables include `created_at` and `updated_at` columns with triggers for automatic updates, providing audit trails.
6. **Row Level Security (RLS):** Supabase RLS policies restrict data access based on authenticated user roles and ownership.
7. **Prepared Statements:** The application uses parameterized queries through the Supabase client library, preventing SQL injection attacks.

4.2 Front End Implementation

The frontend is built using Next.js 15 with React, TypeScript, and Tailwind CSS. The application follows a modular component-based architecture with server-side rendering for optimal performance.

4.2.1 Form Creation

The system implements various forms for user authentication, profile management, and feedback collection. This section describes the design and fields of the key forms in the application.

Sign In / Register Form

The authentication form provides a unified interface for both new and returning users. The form adapts based on user actions and supports multiple authentication methods.

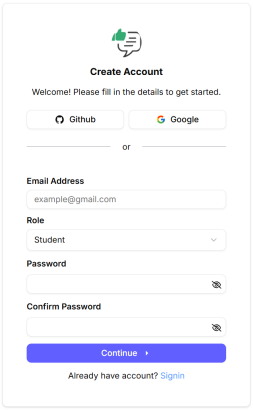

Figure 4.1: Sign In Form Interface

Form Fields:

- **Email Address:** A text input field for the user's institutional email address. The field includes validation for proper email format and domain restrictions (e.g., @rvce.edu.in).
- **OTP Input:** Upon email submission, users receive a 6-digit One-Time Password via email. A numeric input field is displayed for OTP entry with automatic focus progression.
- **Google OAuth Button:** A social login button that initiates the Google OAuth 2.0 authentication flow for users with Google accounts.

Form Behavior:

- New users are automatically registered upon first successful authentication
- Role selection (Student/Faculty) is prompted after initial registration
- Session tokens are securely stored and managed by Supabase Auth

Feedback Form Builder

The form builder allows faculty members to create customized feedback forms with a drag-and-drop interface and real-time preview.

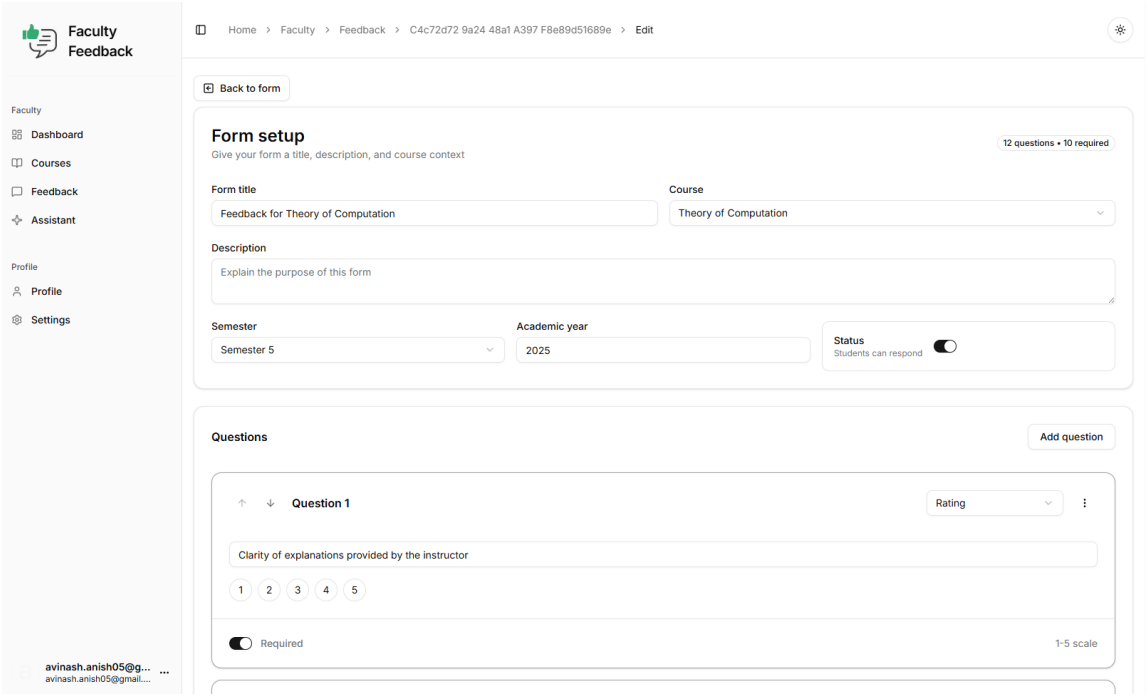

Figure 4.2: Feedback Form Builder Interface – An intuitive drag-and-drop interface allowing faculty to create customized feedback forms with various question types and real-time preview.

Form Metadata Fields:

- **Form Title:** Text input for the feedback form name (e.g., ”Mid-Term Feedback”)
- **Description:** Optional textarea for providing instructions to students
- **Course Selection:** Dropdown to select the associated course from faculty’s assigned courses
- **Semester:** Numeric selector for the current semester (1-8)
- **Academic Year:** Year selector defaulting to current year
- **Active Status:** Toggle switch to enable/disable form visibility to students

Question Builder Fields:

- **Question Text:** Text input for the question content
- **Question Type:** Dropdown selector with options:
  - Text (single line)
  - Textarea (multi-line)
  - Email
  - Number
  - Checkbox (multiple selection)

  - Radio (single selection)
  - Dropdown
  - Rating (1-5 scale)

- **Required Toggle:** Checkbox to mark question as mandatory
- **Options List:** Dynamic list for adding choices (for radio, checkbox, dropdown types)

Question Actions:

- Add new question
- Duplicate existing question
- Delete question
- Reorder questions (move up/down)

4.2.2 Connectivity to the Database

The application uses Supabase as the backend service, providing seamless database connectivity through the Supabase JavaScript client library.

Server Actions Architecture

Next.js Server Actions are used to handle database operations securely on the server side. This approach provides several benefits:

- Database credentials remain server-side only
- Reduced client-side bundle size
- Type-safe database operations with TypeScript
- Automatic request/response handling

```
"use server";

import { createSupabaseServer } from "@/lib/supabase/server";

export const getUserRole = async () => {
    const supabase = await createSupabaseServer();
    const { data: { user } } = await supabase.auth.getUser();

    if (!user) return null;
```

```
    const { data, error } = await supabase
        .from("users")
        .select("role")
        .eq("user_id", user.id)
        .single();

    if (error) {
        console.error("Error fetching user role:", error);
        return null;
    }

    return data?.role || null;
};
```

Server action for fetching user role

Real-time Data Synchronization

The application leverages Supabase's real-time capabilities for live updates. When a student submits feedback, faculty dashboards are automatically updated without requiring manual refresh.

4.2.3 Report Generation

The system provides comprehensive feedback analytics and report generation capabilities for faculty members.

Analytics Dashboard

Faculty members can view aggregated feedback statistics through an intuitive dashboard interface. The results page displays:

- Total submission count per form
- Average ratings for rating-type questions with visual indicators
- Response distribution for multiple-choice questions shown as bar charts
- Individual text responses in a scrollable list
- Overall form statistics and completion rates

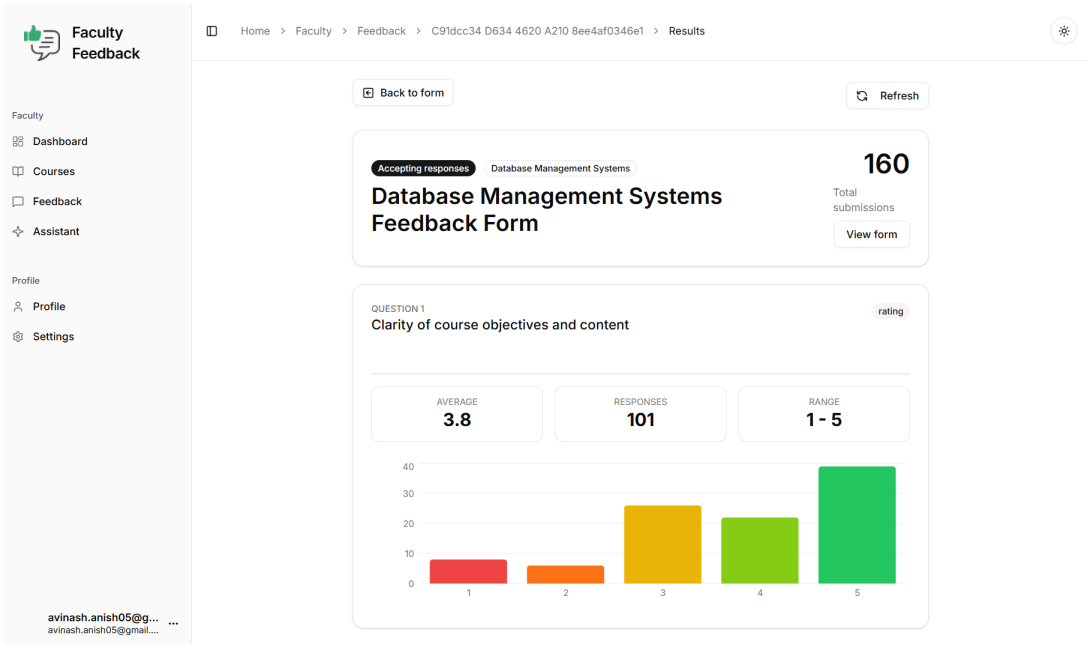

Figure 4.3: Feedback Results Dashboard

The dashboard provides real-time updates as students submit their feedback, allowing faculty to monitor response rates and preliminary results during an active feedback collection period.

AI-Assisted Analysis

The system integrates an AI assistant powered by Google's Generative AI that can:

- Summarize feedback responses across multiple submissions
- Identify common themes and sentiments in text responses
- Provide actionable suggestions based on feedback patterns
- Generate natural language reports from statistical data

Export Capabilities

Feedback data can be exported in multiple formats for further analysis:

- **JSON:** Complete structured data export
- **CSV:** Spreadsheet-compatible format for external analysis
- **PDF:** Formatted reports for documentation purposes

4.2.4 Security Features

The frontend implementation includes multiple layers of security:

Authentication

1. **OAuth 2.0:** Google OAuth integration for institutional accounts
2. **Email OTP:** One-time password authentication via email for users without Google accounts
3. **Session Management:** Secure server-side session handling with automatic token refresh

Authorization

1. **Role-Based Access Control:** Different interfaces for admin, faculty, and student roles
2. **Route Protection:** Middleware-based route guards prevent unauthorized access
3. **API Protection:** All server actions verify user authentication and role before processing requests

Data Protection

1. **HTTPS:** All communications encrypted in transit
2. **Input Sanitization:** All user inputs are validated and sanitized before processing
3. **Anonymous Feedback:** Student identities are protected in feedback submissions, with only aggregated data visible to faculty

TESTING AND RESULTS

This chapter presents the testing methodologies used to validate the Faculty Feedback Management System. The testing covers database integrity, front-end functionality, and system integration.

5.1 Database Testing

Database testing ensures the correctness and integrity of the PostgreSQL database. The following test cases validate schema constraints, referential integrity, and query correctness.

5.1.1 Test Cases

Test Description	Expected Result	Status
Insert user with invalid role value	CHECK constraint violation error	Pass
Insert user with duplicate email	UNIQUE constraint violation error	Pass
Insert student with missing required fields	NOT NULL constraint error	Pass
Insert enrollment with non-existent student	Foreign key violation error	Pass
Delete user and verify cascade to student profile	Student record deleted automatically	Pass
Delete feedback form and verify cascade to questions	Questions and options deleted	Pass
Update record and check updated_at trigger	Timestamp auto-updated to current time	Pass
Fetch active forms for enrolled student	Returns only relevant forms	Pass

Table 5.1: Database Test Cases

Sample Test: Constraint Validation

```
-- Test: Attempt to insert user with invalid role
INSERT INTO users (email, role) VALUES ('test@rvce.edu.in', '
   superuser');

-- Result: CHECK constraint "users_role_check" violated - PASS
```

Test: Insert user with invalid role

```
-- Setup: Create form with questions
INSERT INTO feedback_forms (...) VALUES (...);
INSERT INTO form_questions (...) VALUES (...);
```

```
-- Test: Delete the form
DELETE FROM feedback_forms WHERE form_id = 'test-form-uuid';

-- Verify: Questions should be deleted
SELECT COUNT(*) FROM form_questions WHERE form_id = 'test-form-uuid
    ';
-- Result: 0 rows - PASS
```

Test: Cascade delete verification

5.2 Front End Testing

Front-end testing validates the user interface, form handling, and user experience across different modules.

5.2.1 Test Cases

Authentication Module

Test Description	Expected Result	Status
Sign in with valid institutional email	OTP sent, OTP input displayed	Pass
Sign in with invalid email format	Validation error message shown	Pass
Enter correct OTP	User authenticated, redirected to dashboard	Pass
Google OAuth sign-in flow	User authenticated via Google	Pass
Access protected route without login	Redirected to sign-in page	Pass

Table 5.2: Authentication Test Cases

Admin Dashboard

Test Description	Expected Result	Status
Add new department	Department created, table updated	Pass
Add department with duplicate code	Error message displayed	Pass
Edit course details	Changes saved and reflected	Pass
Delete course with confirmation	Course removed from system	Pass
Assign faculty to course	Faculty assignment updated	Pass

Table 5.3: Admin Dashboard Test Cases

Feedback Form Builder

Test Description	Expected Result	Status
Create feedback form with all question types	Form saved with all questions	Pass
Add options to multiple-choice questions	Options saved correctly	Pass
Duplicate and delete questions	Operations performed correctly	Pass
Reorder questions using move buttons	Order updated correctly	Pass
Save form without required fields	Validation error, save disabled	Pass
Toggle form active status	Visibility updated for students	Pass

Table 5.4: Form Builder Test Cases

Student Feedback Submission

Test Description	Expected Result	Status
View available feedback forms	Only active forms for enrolled courses shown	Pass
Submit feedback with all required fields	Submission recorded, confirmation shown	Pass
Submit without required fields	Validation errors displayed	Pass
Attempt duplicate submission	Error or redirect to submitted page	Pass

Table 5.5: Student Submission Test Cases

Results and Analytics

Test Description	Expected Result	Status
View form results with submissions	Analytics and charts displayed	Pass
View average rating calculations	Correct averages shown	Pass
Use AI assistant for summarization	Summary generated correctly	Pass
Export results as JSON/CSV	Valid file downloaded	Pass

Table 5.6: Results and Analytics Test Cases

5.3 System Testing

System testing validates end-to-end workflows that span multiple modules.

5.3.1 Test Cases

Complete Feedback Lifecycle

This test verifies the entire feedback collection workflow:

1. **Admin:** Creates department, course, faculty, and enrolls students
2. **Faculty:** Creates and activates feedback form with multiple question types
3. **Student:** Views available forms, fills responses, and submits feedback
4. **Faculty:** Views submission count, analytics, and AI-generated summary

Result: All data flows correctly between modules. Submissions recorded, analytics calculated accurately. **PASS**

Role-Based Access Control

Test Description	Expected Result	Status
Student accessing admin dashboard	Redirected to student dashboard	Pass
Faculty accessing other faculty's forms	Only own forms visible	Pass
Unauthenticated access to API endpoints	401 Unauthorized response	Pass

Table 5.7: Access Control Test Cases

Security Tests

Test Description	Expected Result	Status
SQL injection in form input	Input sanitized, no manipulation	Pass
XSS attempt in text feedback	Script tags escaped	Pass
Session expiry after inactivity	User logged out automatically	Pass

Table 5.8: Security Test Cases

5.3.2 Test Summary

Test Category	Total Tests	Pass Rate
Database Testing	8	100%
Front End Testing	24	100%
System Testing	7	100%
Total	**39**	**100%**

Table 5.9: Test Execution Summary

All test cases passed successfully, demonstrating the reliability of the Faculty Feedback Management System.

CONCLUSION

6.1 Conclusion

The Faculty Feedback Management System has been successfully designed and implemented to streamline the feedback collection process at RVCE. The system addresses the limitations of traditional paper-based feedback methods by providing a digital, efficient, and user-friendly platform for all stakeholders.

The key achievements of this project include:

- **Role-Based Access Control:** Implementation of distinct portals for administrators, faculty, and students, ensuring secure and appropriate access to system features.
- **Dynamic Form Builder:** A flexible form creation tool that allows faculty to design custom feedback forms with multiple question types including text, ratings, and multiple-choice options.
- **Real-Time Analytics:** Comprehensive analytics dashboard providing faculty with instant insights into feedback responses, including average ratings, response distributions, and text response summaries.
- **AI-Powered Insights:** Integration of Google's Generative AI to provide intelligent summarization and analysis of feedback data, helping faculty identify key themes and actionable suggestions.
- **Progressive Web App:** Implementation of PWA features enabling mobile installation and push notifications for timely feedback collection.
- **Secure Authentication:** Multi-method authentication supporting Google OAuth and Email OTP, ensuring secure access while maintaining user convenience.

The system was built using modern web technologies including Next.js, React, TypeScript, and Tailwind CSS for the frontend, with PostgreSQL on Supabase providing a robust and scalable backend. All 39 test cases across database, frontend, and system testing passed successfully, validating the reliability of the implementation.

6.2 Limitations

While the system meets its primary objectives, some limitations exist:

- **Internet Dependency:** The system requires an active internet connection for all operations. Offline functionality is not currently supported.

- **Limited Report Formats:** Currently, feedback reports can only be exported as JSON and CSV. PDF report generation with custom formatting is not yet implemented.
- **Single Institution Focus:** The system is designed specifically for RVCE's structure. Adapting it for other institutions would require modifications to the department and course hierarchy.
- **No Comparative Analytics:** The system does not currently support comparing feedback across different semesters or academic years for trend analysis.
- **Limited Accessibility Features:** While the UI is responsive, comprehensive accessibility features for users with disabilities have not been fully implemented.

6.3 Future Enhancements

The following enhancements are proposed for future development:

- **Offline Support:** Implement service workers to enable offline form viewing and queued submission when connectivity is restored.
- **Trend Analysis:** Implement comparative analytics to track feedback trends across semesters and academic years, helping identify long-term improvements or concerns.
- **Multi-Language Support:** Implement internationalization to support feedback collection in multiple languages.
- **Integration with LMS:** Integrate with existing Learning Management Systems for seamless course and enrollment data synchronization.

In conclusion, the Faculty Feedback Management System successfully digitizes and enhances the feedback collection process, providing a foundation that can be extended with additional features to further improve the teaching-learning experience at educational institutions.

BIBLIOGRAPHY

[1] Vercel, "Next.js Documentation – The React Framework for the Web," 2024. Available: https://nextjs.org/docs

[2] Meta Platforms, Inc., "React – A JavaScript library for building user interfaces," 2024. Available: https://react.dev/

[3] Microsoft, "TypeScript: JavaScript With Syntax For Types," 2024. Available: https://www.typescriptlang.org/docs/

[4] Tailwind Labs, "Tailwind CSS – A utility-first CSS framework," 2024. Available: https://tailwindcss.com/docs

[5] shadcn, "shadcn/ui – Beautifully designed components built with Radix UI and Tailwind CSS," 2024. Available: https://ui.shadcn.com/docs

[6] Supabase, Inc., "Supabase Documentation – The Open Source Firebase Alternative," 2024. Available: https://supabase.com/docs

[7] The PostgreSQL Global Development Group, "PostgreSQL: The World's Most Advanced Open Source Relational Database," 2024. Available: https://www.postgresql.org/docs/

[8] Supabase, Inc., "Supabase Auth – User Management and Authentication," 2024. Available: https://supabase.com/docs/guides/auth

[9] Supabase, Inc., "Row Level Security," 2024. Available: https://supabase.com/docs/guides/database/postgres/row-level-security

[10] Google, "Gemini API Documentation – Google AI for Developers," 2024. Available: https://ai.google.dev/gemini-api/docs

[11] Vercel, "AI SDK – The AI Toolkit for TypeScript," 2024. Available: https://sdk.vercel.ai/docs

[12] Resend, Inc., "Resend – Email for developers," 2024. Available: https://resend.com/docs

[13] M. Thomson, E. Damaggio, and B. Raymor, "Generic Event Delivery Using HTTP Push," IETF RFC 8030, 2017. Available: https://datatracker.ietf.org/doc/html/rfc8030

[14] Google Developers, "Progressive Web Apps," 2024. Available: https://web.dev/progressive-web-apps/

[15] Mozilla Developer Network, "Service Worker API," 2024. Available: https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API

[16] D. Hardt, Ed., "The OAuth 2.0 Authorization Framework," IETF RFC 6749, Oct. 2012. Available: https://datatracker.ietf.org/doc/html/rfc6749

[17] Google, "Using OAuth 2.0 to Access Google APIs," 2024. Available: https://developers.google.com/identity/protocols/oauth2

[18] E. F. Codd, "A Relational Model of Data for Large Shared Data Banks," *Communications of the ACM*, vol. 13, no. 6, pp. 377–387, Jun. 1970.

[19] A. Silberschatz, H. F. Korth, and S. Sudarshan, *Database System Concepts*, 7th ed. McGraw-Hill Education, 2019.

[20] I. Sommerville, *Software Engineering*, 10th ed. Pearson, 2015.

[21] React DnD, "React DnD – Drag and Drop for React," 2024. Available: https://react-dnd.github.io/react-dnd/about

[22] Recharts, "Recharts – A composable charting library built on React components," 2024. Available: https://recharts.org/en-US/

[23] Vercel, "Vercel Documentation – Develop. Preview. Ship.," 2024. Available: https://vercel.com/docs

[24] Docker, Inc., "Docker Documentation," 2024. Available: https://docs.docker.com/

CODE SNIPPETS

A.1 Source Code Repository

The complete source code for the Faculty Feedback Management System is available on GitHub:

`https://github.com/Huzaif2309/faculty-feedback-system`

The repository contains the full implementation including the frontend application, backend API, and database schema.

A.2 Frontend Setup

A.2.1 Prerequisites

- Node.js (v18 or higher)
- Supabase Account (`https://supabase.com`)
- Google Cloud Console Project for OAuth and AI features
- Resend Account for email functionality (`https://resend.com`)

A.2.2 Installation

```
git clone https://github.com/Huzaif2309/faculty-feedback-system
cd faculty-feedback-system/frontend
npm install
```

Clone and Install Frontend

A.2.3 Environment Configuration

```
cp env.example .env
```

Configure Environment Variables

Required environment variables in `.env`:

- `NEXT_PUBLIC_SUPABASE_URL` - Supabase project URL
- `NEXT_PUBLIC_SUPABASE_ANON_KEY` - Supabase anonymous key
- `RESEND_API_KEY` - Resend API key for emails

- `GOOGLE_GENERATIVE_AI_API_KEY` - Google AI API key
- `NEXT_PUBLIC_VAPID_PUBLIC_KEY` - VAPID public key
- `VAPID_PRIVATE_KEY` - VAPID private key

Generate VAPID keys for push notifications:

```
npx web-push generate-vapid-keys
```

Generate VAPID Keys

A.2.4 Database Setup

Initialize the Supabase database by running the SQL schema in the Supabase Dashboard SQL Editor:

```
frontend/lib/supabase/migrations/schema.sql
```

Database Schema Location

A.2.5 Run the Application

```
npm run dev
# Application available at http://localhost:3000
```

Start Frontend Development Server

A.3 Backend Setup

The backend is a Flask API that provides additional REST endpoints for data management.

A.3.1 Prerequisites

- Python 3.10 or higher
- pip package manager

A.3.2 Installation

```
cd faculty-feedback-system/faculty-feedback-backend
pip install -r requirements.txt
```

Install Backend Dependencies

A.3.3 Environment Configuration

Create a `.env` file with the following variables:

- `SUPABASE_URL` - Supabase project URL
- `SUPABASE_KEY` - Supabase service role key

A.3.4 Run the Backend

```
python app.py
# API available at http://localhost:5000
```

Start Backend Server

A.4 Docker Deployment

Both frontend and backend can be deployed using Docker:

```
docker-compose up --build
```

Docker Deployment

SCREENSHOTS

This appendix provides a comprehensive visual walkthrough of the Faculty Feedback System, showcasing the key interfaces and features available to different user roles.

B.1 Landing and Authentication

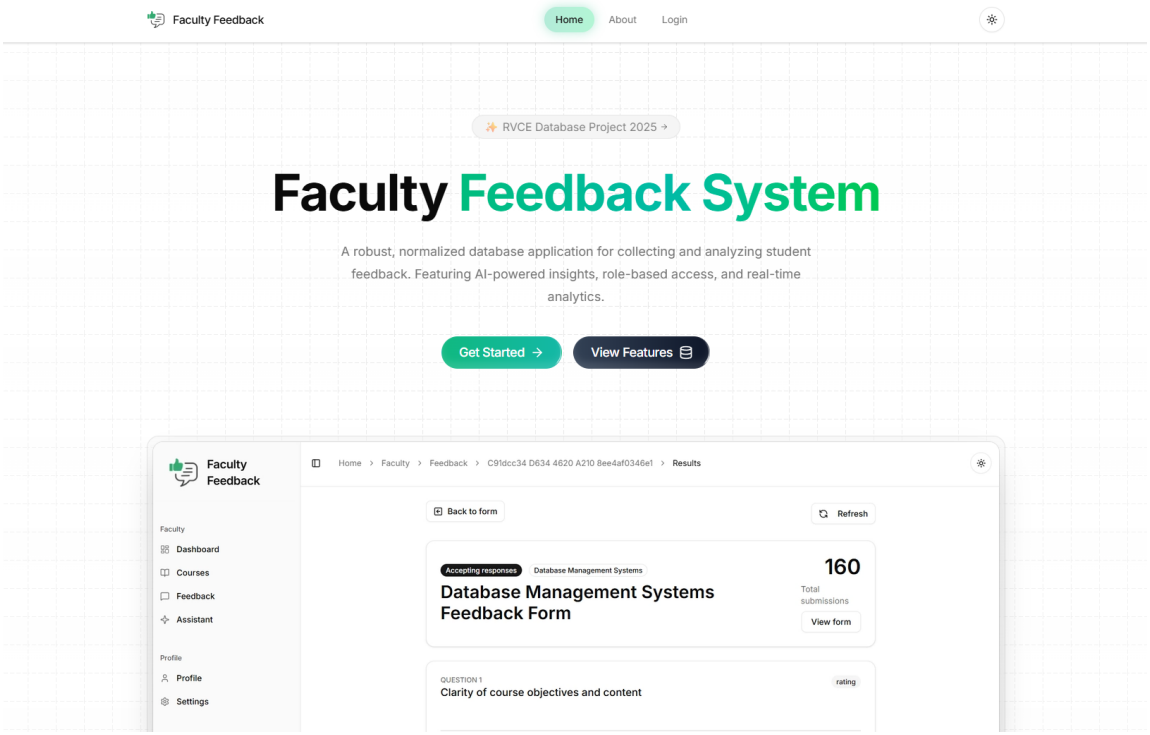

Figure B.1: Faculty Feedback System Landing Page – The main entry point featuring a modern, intuitive interface that welcomes users and provides navigation to sign-in functionality.

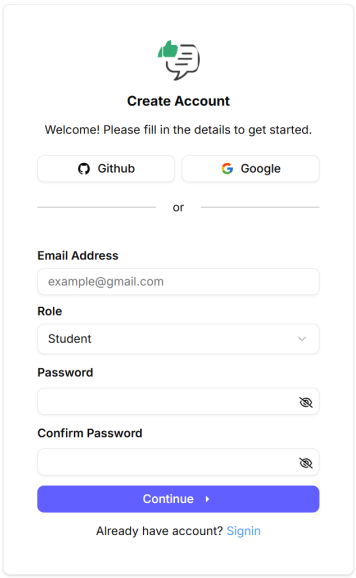

Figure B.2: User Authentication Interface – Secure sign-in page supporting role-based authentication for administrators, faculty members, and students.

B.2 Administrator Interface

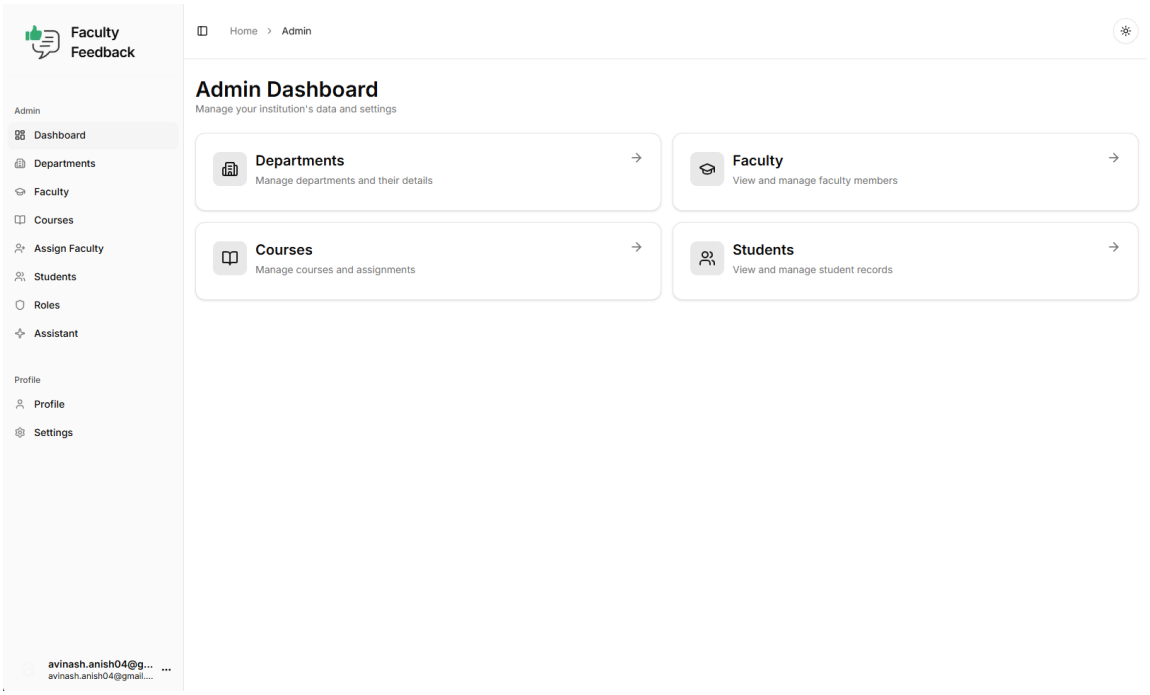

Figure B.3: Administrator Dashboard – The central control panel for system administrators, providing an overview of system metrics, user statistics, and quick access to management functions.

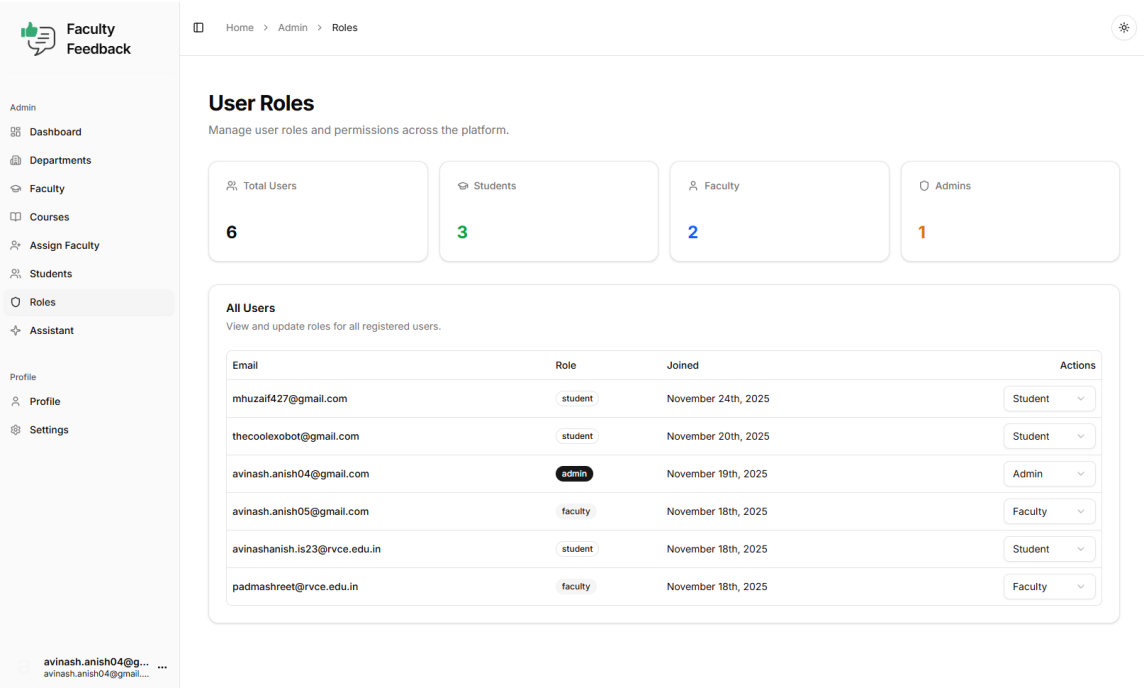

Figure B.4: Role Management Interface – Administrators can assign and manage user roles, ensuring proper access control across the system for faculty and student accounts.

B.3 Faculty Interface

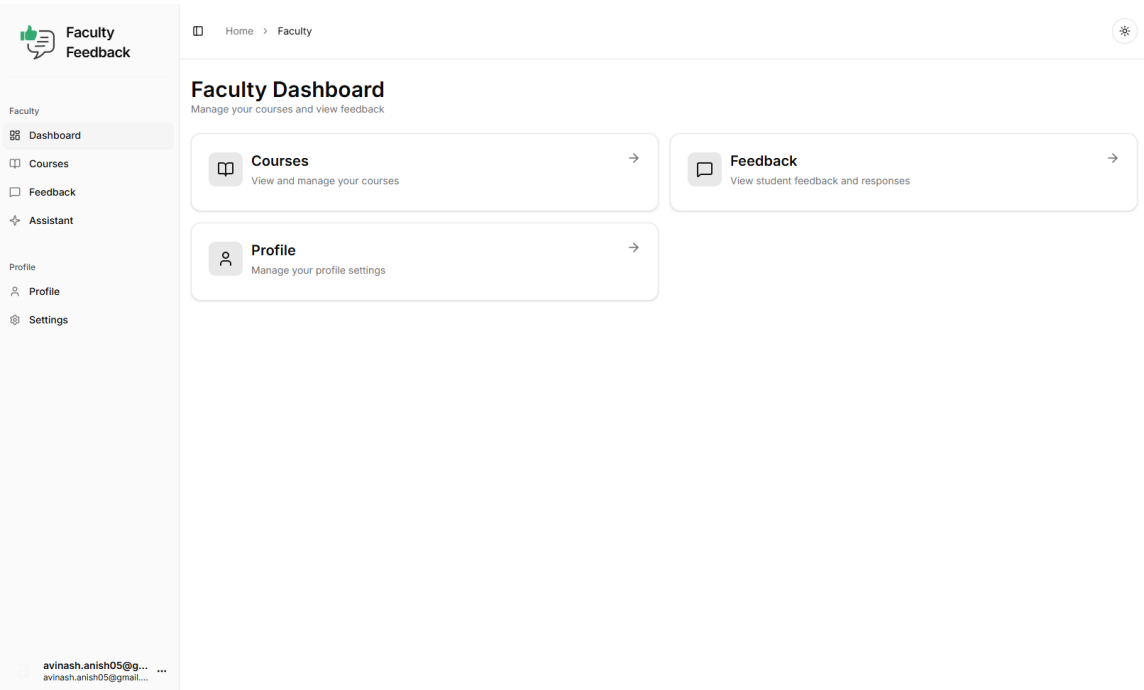

Figure B.5: Faculty Dashboard – A personalized workspace displaying an overview of created feedback forms, response statistics, and navigation to form management tools.

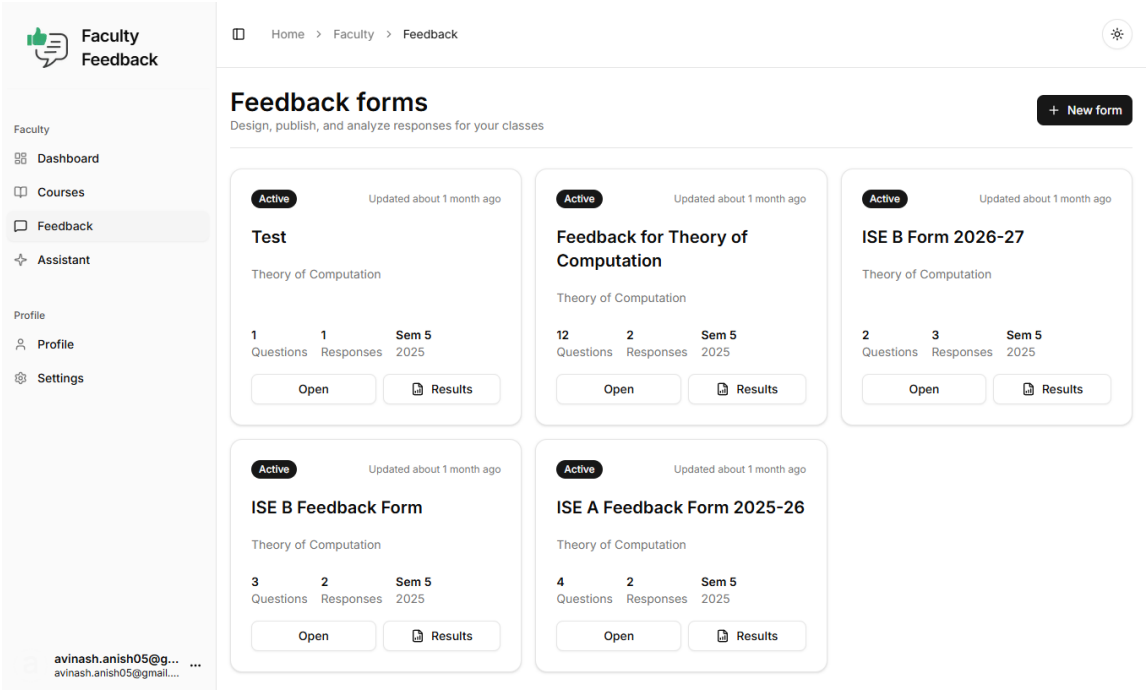

Figure B.6: Faculty Forms List – A comprehensive view of all feedback forms created by the faculty member, with options to view responses, edit, share, or delete forms.

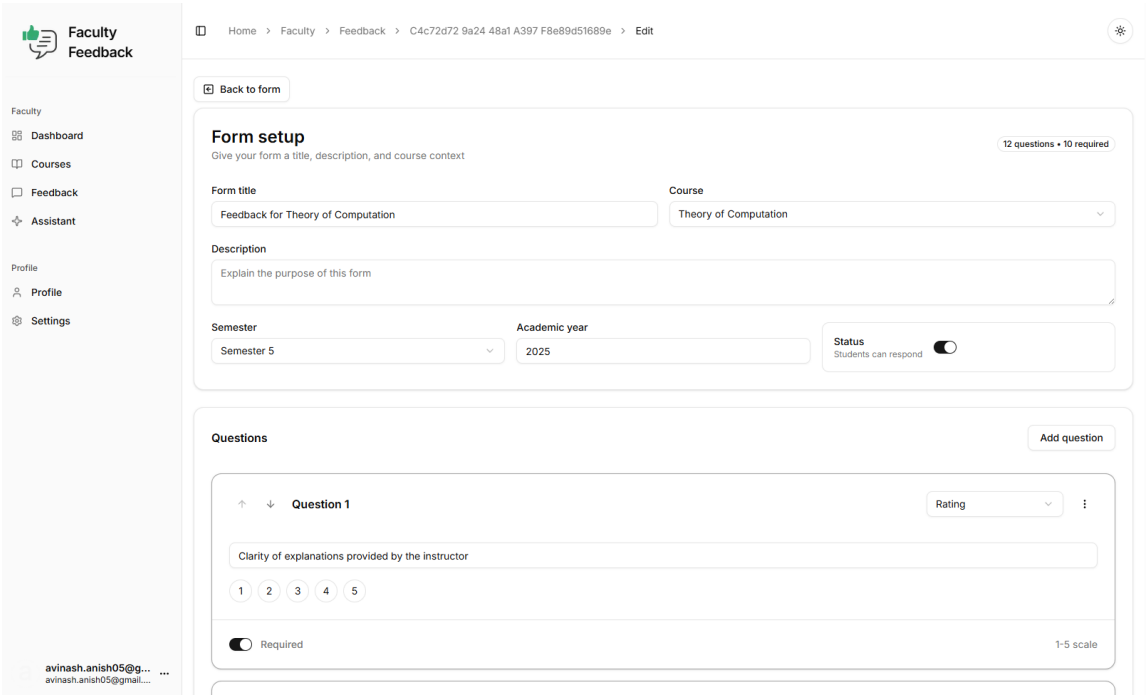

Figure B.7: Form Builder Interface – An intuitive drag-and-drop form builder allowing faculty to create custom feedback forms with various question types, including multiple choice, rating scales, and open-ended questions.

B.4 AI-Powered Form Assistant

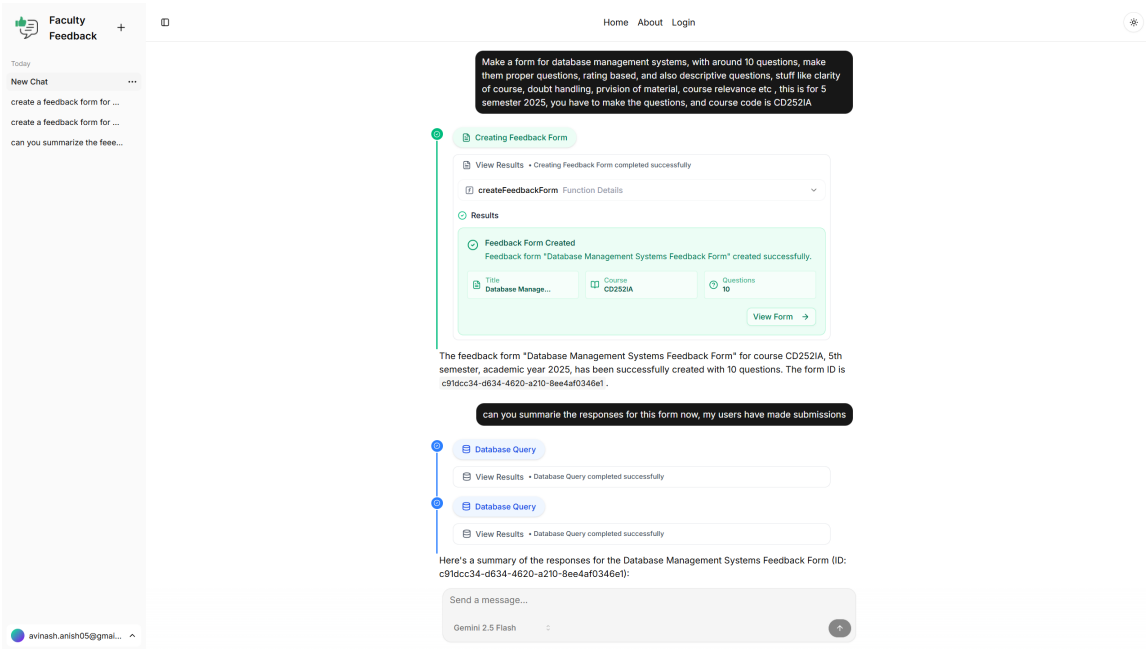

Figure B.8: AI Form Creation Assistant – The intelligent assistant interface that helps faculty members generate feedback forms through natural language prompts, streamlining the form creation process.

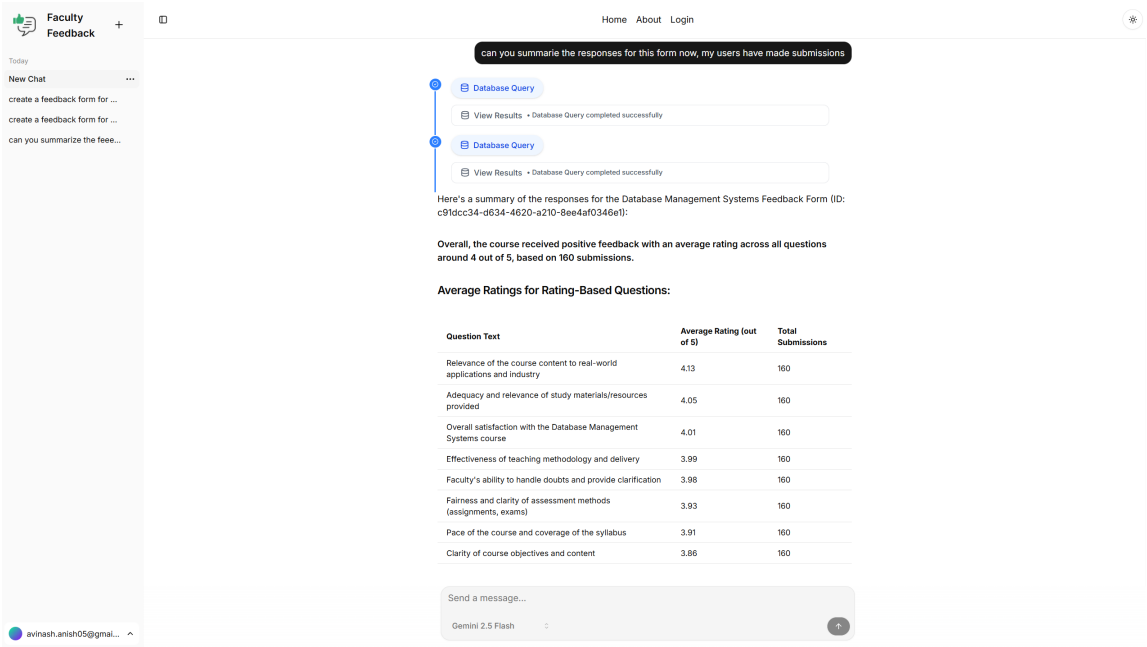

Figure B.9: AI Form Summary View – The assistant provides a comprehensive summary of the generated form, allowing faculty to review and refine questions before publishing.

B.5 Feedback Results and Analytics

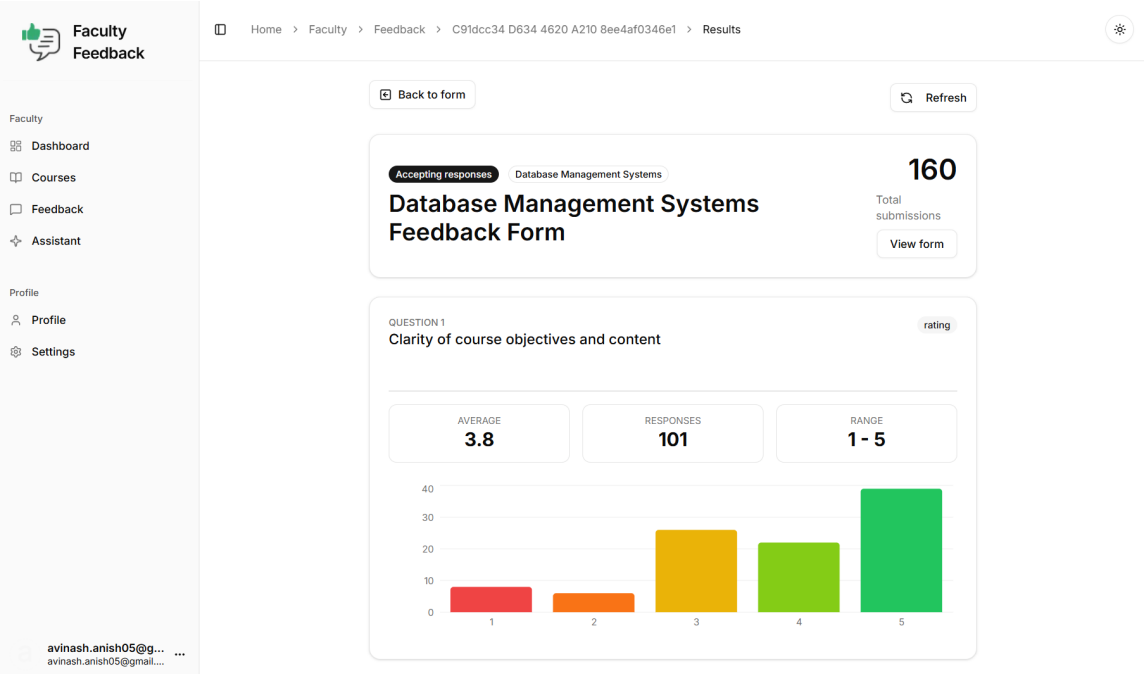

Figure B.10: Feedback Results Dashboard – Detailed analytics and visualizations of student responses, including response distributions, average ratings, and sentiment analysis of open-ended feedback.

B.6 Student Interface

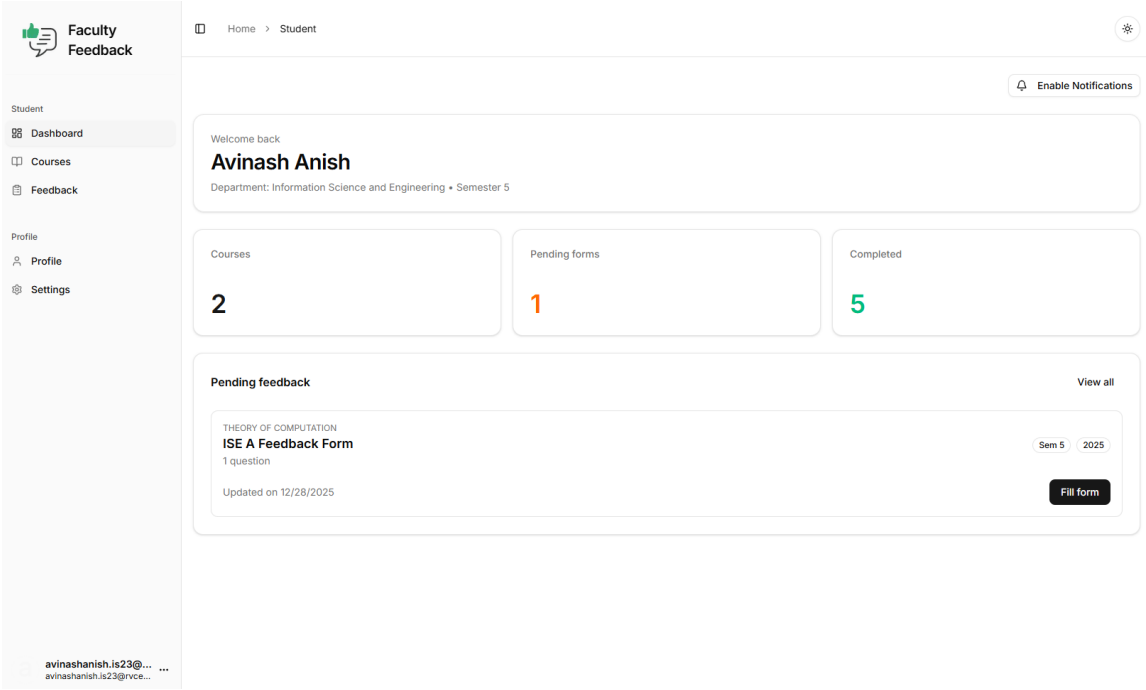

Figure B.11: Student Dashboard – The student's personalized view showing pending feedback forms assigned by faculty, completed submissions, and navigation options.

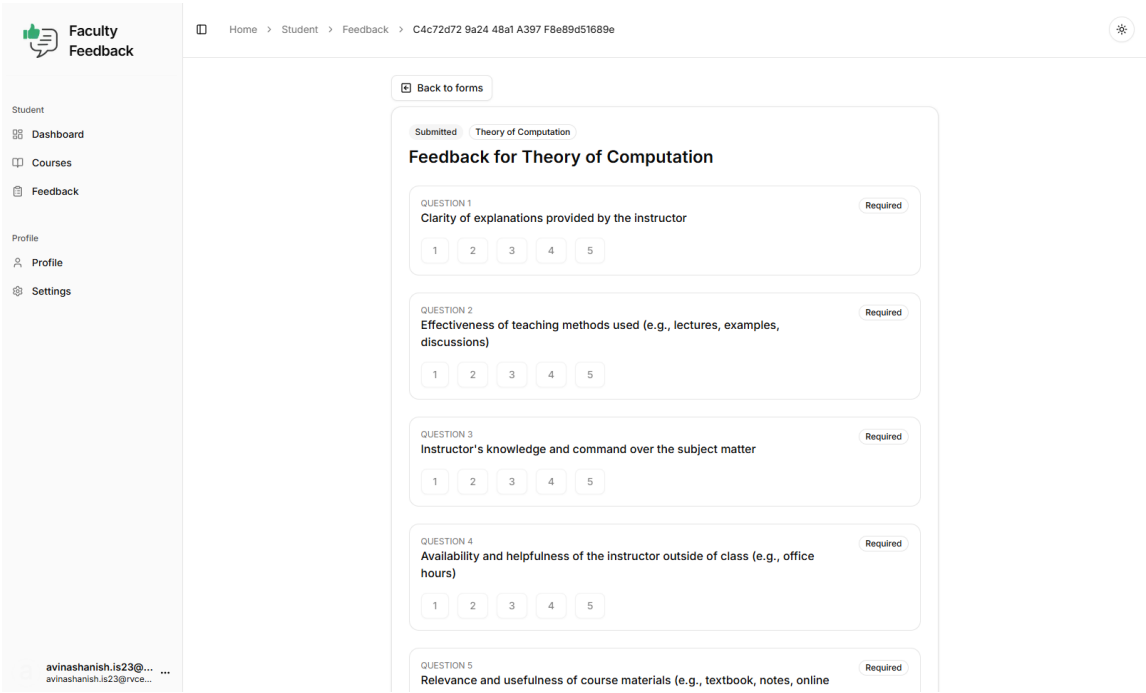

Figure B.12: Student Feedback Form – The clean, accessible interface students use to submit their feedback, featuring clear instructions, progress indicators, and anonymous submission options.