BRAIN RUSH

PRESENTED BY NARGIZA ALIEVA AND
AIDAI KYDYKBEKOVA

INTRODUCTION

Brain Rush is a modern online learning platform aimed at providing students and instructors with an engaging and user-friendly environment. The platform offers a wide range of courses across various categories, helping users acquire new skills and knowledge.

Our focus is on delivering a personalized, interactive, and efficient learning experience tailored to individual needs.

ABOUT PROJECT

Brain Rush is designed as an innovative platform for course creation, enrollment, and feedback management. It allows instructors to manage their courses efficiently while enabling students to track their learning progress with ease.

From a technical perspective, Brain Rush uses a relational database structure to ensure data consistency and performance. Brain Rush is a step forward in transforming online education through innovative technology and user-centric solutions.

TECHNOLOGIES USED:

BACHEND:

JAVA

SPRING BOOT

SPRING DATA JPA

SPRING WEB

FLYWAY

FRONTEND

HTML

JS

THYMELEAF

DATABASE

POSTGRESQL

TASKS DESTRIBUTION

NARGIZA		AIDAI
Instructor, Course, Category	**DATABASE** logic together	Student, Enrollment, Feedback
Instructor, Course, Category	**BACKEND**	Student, Enrollment, Feedback
Instructor, Course, Category	**FRONTEND**	Student, Feedback
	OTHER presentation, documentations together	

DATABASE

you can learn more about the database in the
db_schema_documentation

LOGICAL DESIGN

The Brain Rush database is designed using a relational model to ensure efficiency, scalability, and data integrity. It includes six main tables:

1. Instructor - Stores instructor details.
2. Category - Organizes courses into categories for easy filtering.
3. Course - Contains course information, linked to instructors and categories.
4. Student - Tracks student data for enrollment and feedback.
5. Enrollment - Manages student-course relationships with a Many-to-Many structure.
6. Feedback - Captures course ratings and comments from students.

WHY WE CHOSE THIS LOGICAL DESIGN

Instructor can teach many Courses. (One-to-many)
Category can have many Courses. (One-to-many)
Course can be enrolled by many Students. Student can enroll in many Courses. (Many-to-many by enrollment table)
Feedback is linked to both Student and Course entities. (Many-to-one)

Built-in constraints, such as unique keys and checks on values (e.g., ratings 0-5), ensure data validation.

Cascading rules (e.g., ON DELETE CASCADE and SET NULL) simplify maintenance by preserving data integrity during updates or deletions.

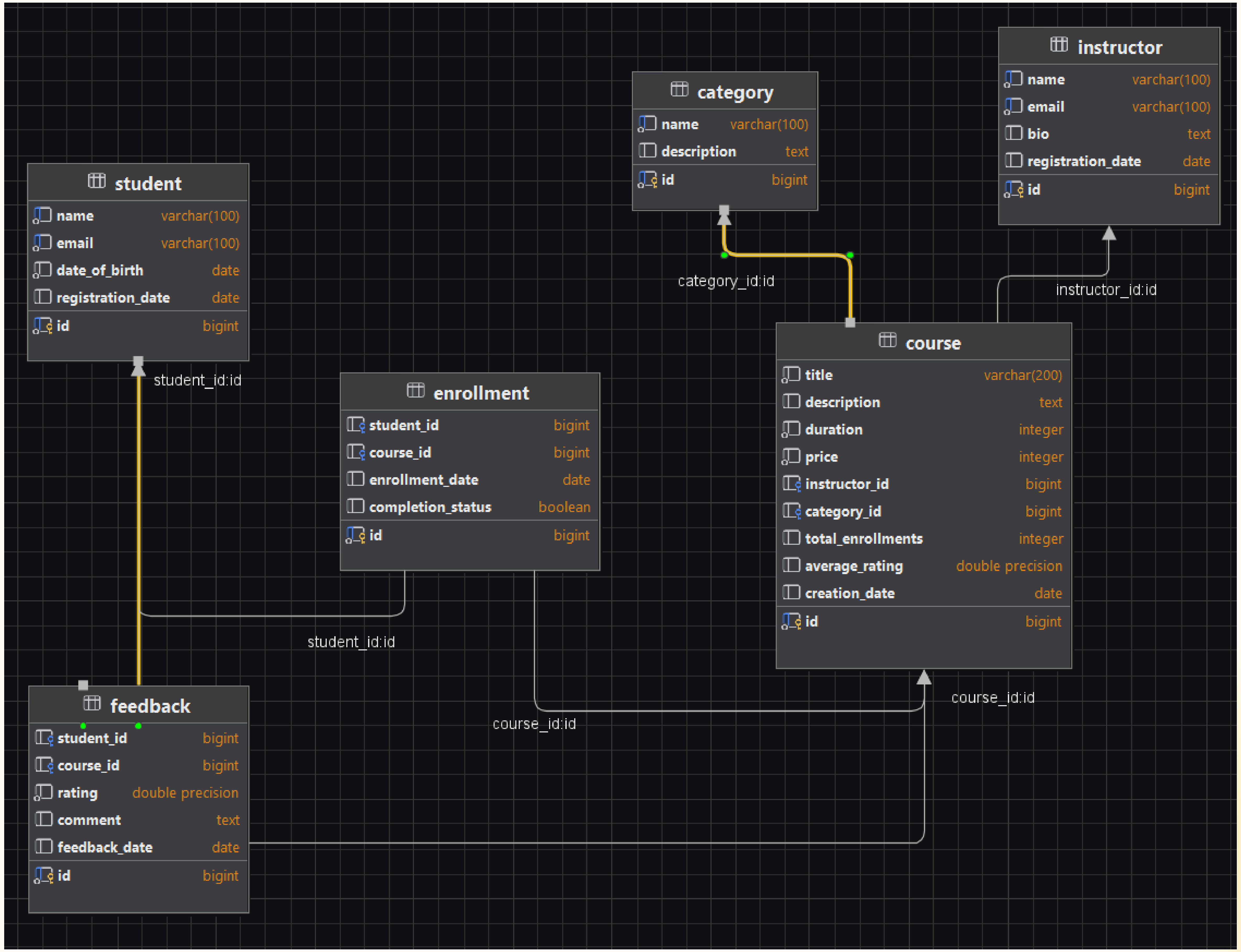
This modular design supports future expansion and provides a robust foundation for the Brain Rush platform.

NORMALIZATION

Normalization Analysis

1. First Normal Form (1NF):
   - Atomic values: All columns contain indivisible values (e.g., Name and Email in Instructor, Course, Student, and Feedback tables).
   - No repeating groups: No nested arrays or lists in any table.
   - Data type constraints: Each column has an appropriate data type (e.g., VARCHAR, BIGSERIAL, DATE).
2. Second Normal Form (2NF):
   - Full dependency on primary key: Every non-key column depends on the entire primary key (e.g., Instructor_ID and Category_ID depend on the ID in the Course table).
   - No partial dependencies: No column depends solely on a part of a composite key.
   - Normalization achieved: All non-key attributes fully depend on the primary key.
3. Third Normal Form (3NF):
   - No transitive dependencies: Non-key attributes are independent of other non-key attributes (e.g., Instructor_ID and Category_ID in the Course table do not rely on other non-key attributes).
   - Redundancy minimization: Foreign keys and ON DELETE SET NULL/CASCADE rules maintain data integrity and minimize redundancy.
   - Normalization achieved: Proper assignment of columns, avoiding transitive dependency issues.
4. Boyce-Codd Normal Form (BCNF):
   - Candidate key determination: Each determinant determines a candidate key (e.g., Instructor_ID and Category_ID in the Course table determine a key).

ENTITY-RELATIONSHIP (E-R) MODEL



student

name	varchar(100)
email	varchar(100)
date_of_birth	date
registration_date	date
id	bigint

category

name	varchar(100)
description	text
id	bigint

instructor

name	varchar(100)
email	varchar(100)
bio	text
registration_date	date
id	bigint

enrollment

student_id	bigint
course_id	bigint
enrollment_date	date
completion_status	boolean
id	bigint

course

title	varchar(200)
description	text
duration	integer
price	integer
instructor_id	bigint
category_id	bigint
total_enrollments	integer
average_rating	double precision
creation_date	date
id	bigint

feedback

student_id	bigint
course_id	bigint
rating	double precision
comment	text
feedback_date	date
id	bigint

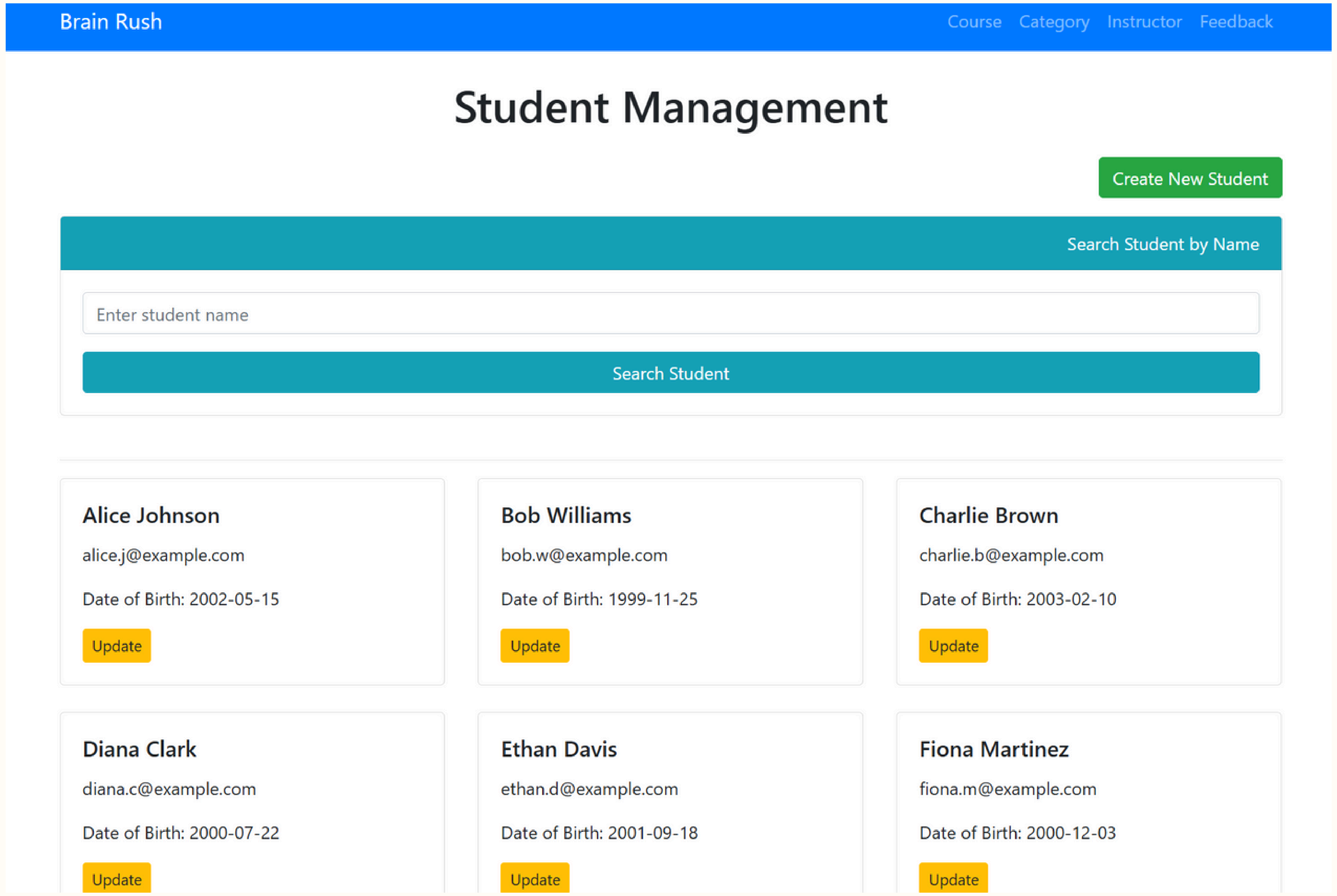
FRONTEND

you can learn more about the frontend in the
user_guide documentation

STUDENT PAGE

http://localhost:8888/student

The Student Page is designed to help users easily manage student records. Below is a comprehensive guide on how to navigate and use the page features.



Student Management

Alice Johnson
alice.j@example.com
Date of Birth: 2002-05-15

Bob Williams
bob.w@example.com
Date of Birth: 1999-11-25

Charlie Brown
charlie.b@example.com
Date of Birth: 2003-02-10

Diana Clark
diana.c@example.com
Date of Birth: 2000-07-22

Ethan Davis
ethan.d@example.com
Date of Birth: 2001-09-18

Fiona Martinez
fiona.m@example.com
Date of Birth: 2000-12-03

MAIN FEATURES

1. Create New Student;
2. Search Student by Name;
3. View All Students;
4. Update Student Information.

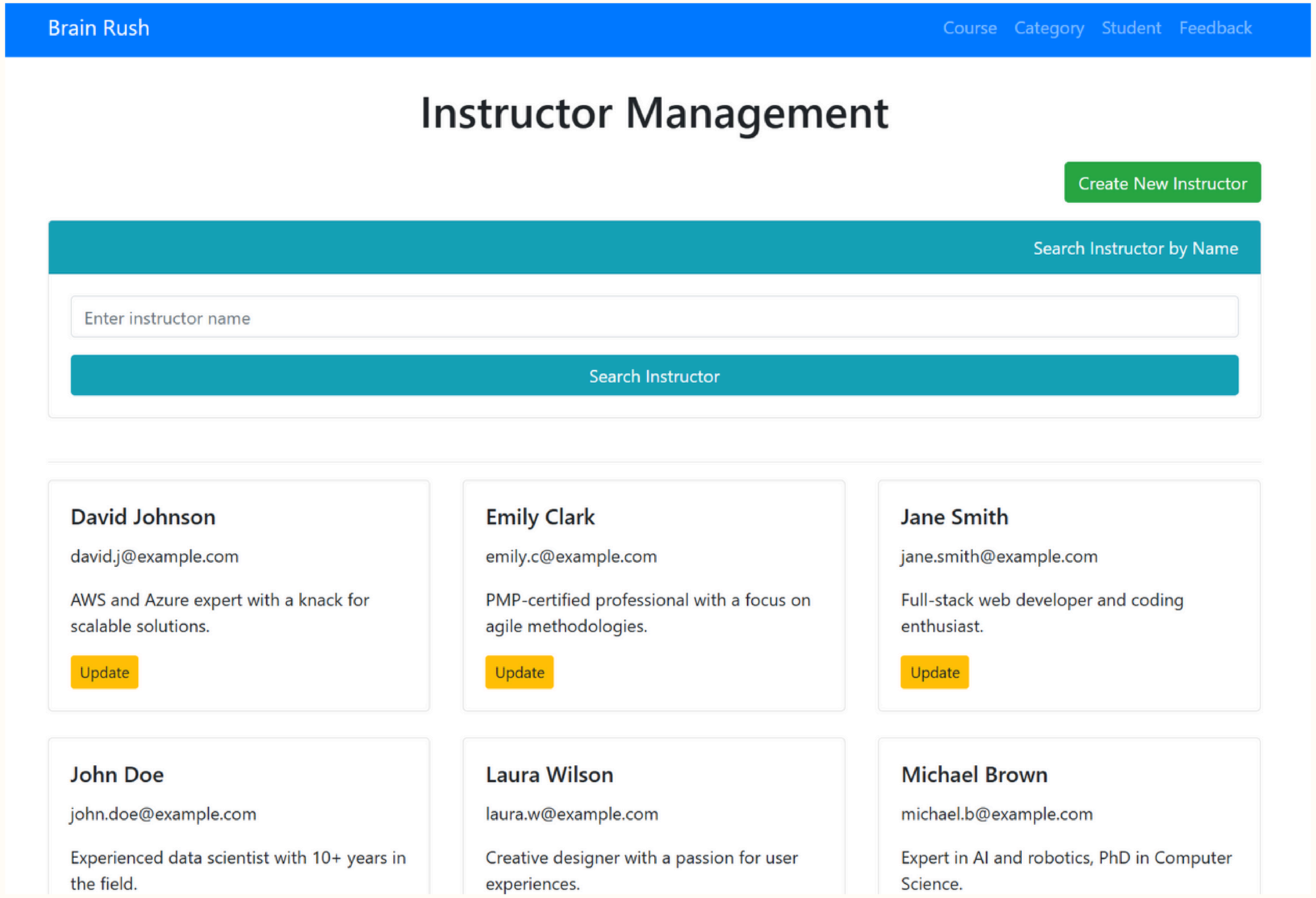
INSTRUCTOR PAGE

http://localhost:8888/instructor

The Instructor Page provides a centralized system for managing instructors in a course database. This user guide will help you navigate and use the various features available on this page.

MAIN FEATURES

1. Create New Instructor;
2. Search Instructor by Name;
3. View All Instructors;
4. Update Instructor Information.



Instructor Management

David Johnson
david.j@example.com
AWS and Azure expert with a knack for scalable solutions.

Emily Clark
emily.c@example.com
PMP-certified professional with a focus on agile methodologies.

Jane Smith
jane.smith@example.com
Full-stack web developer and coding enthusiast.

John Doe
john.doe@example.com
Experienced data scientist with 10+ years in the field.

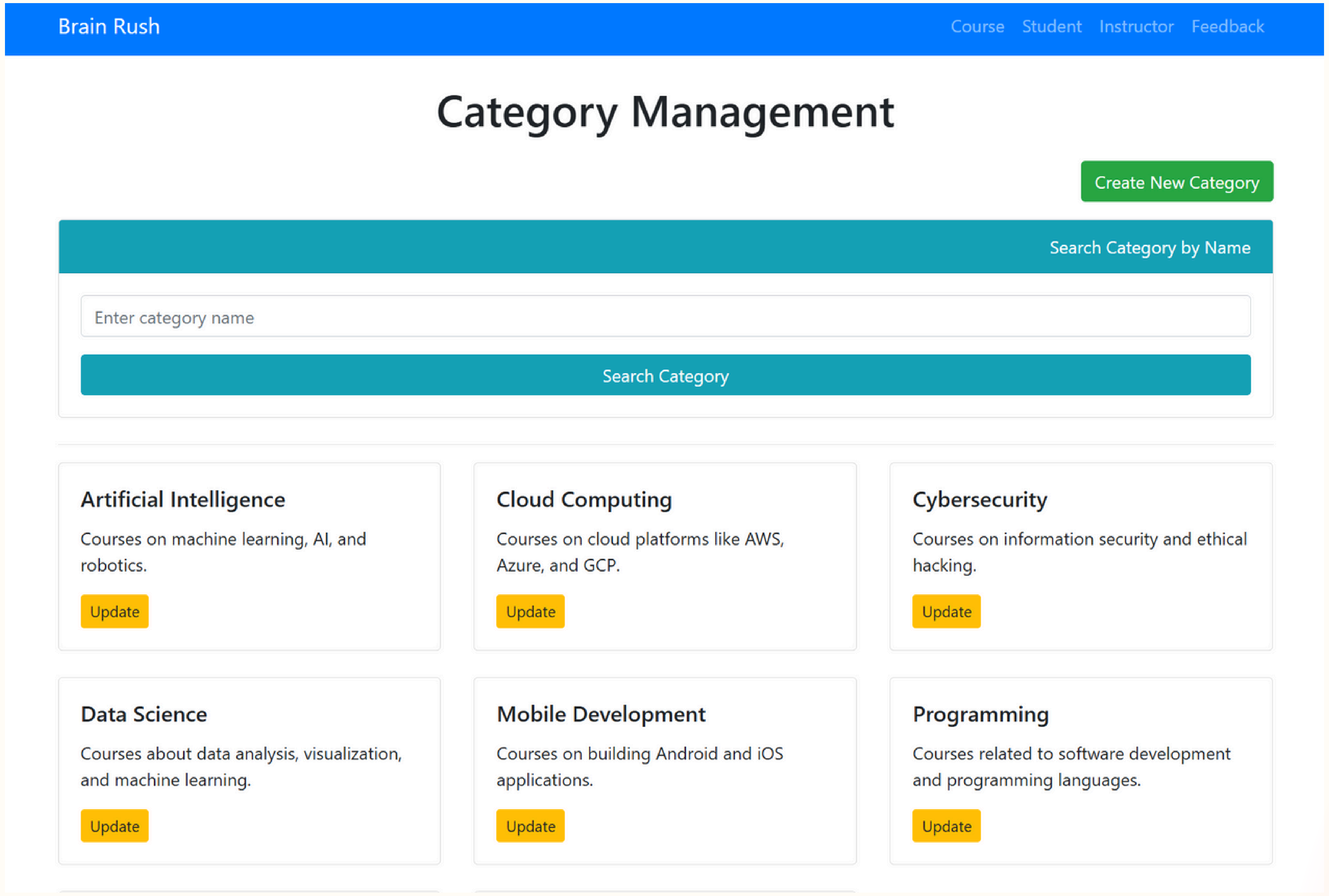
Laura Wilson
laura.w@example.com
Creative designer with a passion for user experiences.

Michael Brown
michael.b@example.com
Expert in AI and robotics, PhD in Computer Science.

CATEGORY PAGE

http://localhost:8888/category

The Category Page is designed to allow users to effectively manage categories within the application. Below is a detailed guide to help navigate and utilize its features.



Category Management

Artificial Intelligence
Courses on machine learning, AI, and robotics.

Cloud Computing
Courses on cloud platforms like AWS, Azure, and GCP.

Cybersecurity
Courses on information security and ethical hacking.

Data Science
Courses about data analysis, visualization, and machine learning.

Mobile Development
Courses on building Android and iOS applications.

Programming
Courses related to software development and programming languages.

MAIN FEATURES

1. Create New Category
2. Search Category by Name
3. View All Categories
4. Update Category Information

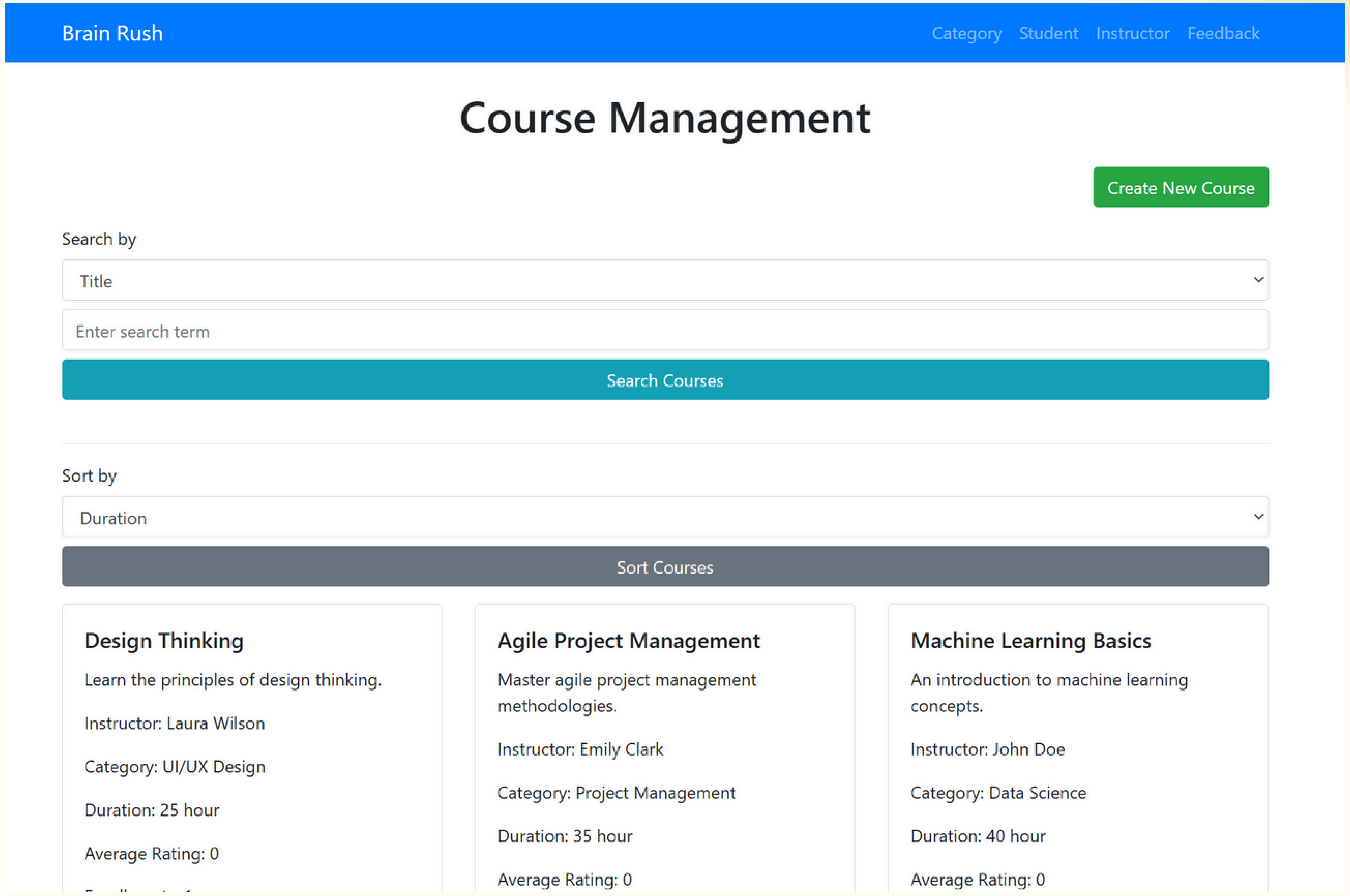
COURSE PAGE

http://localhost:8888/course

The Course Page is designed to help users efficiently manage and oversee courses within the system. This guide provides a comprehensive overview of the features available and how to use them effectively.

MAIN FEATURES

1. Create New Course;
2. Search Courses;
3. Sort Courses;
4. View All Courses;
5. Update Course Information;
6. Enroll Students;
7. Unenroll Students;
8. Write Feedback.



Course Management

Search by: Title

Sort by: Duration

Design Thinking
Learn the principles of design thinking.
Instructor: Laura Wilson
Category: UI/UX Design
Duration: 25 hour
Average Rating: 0

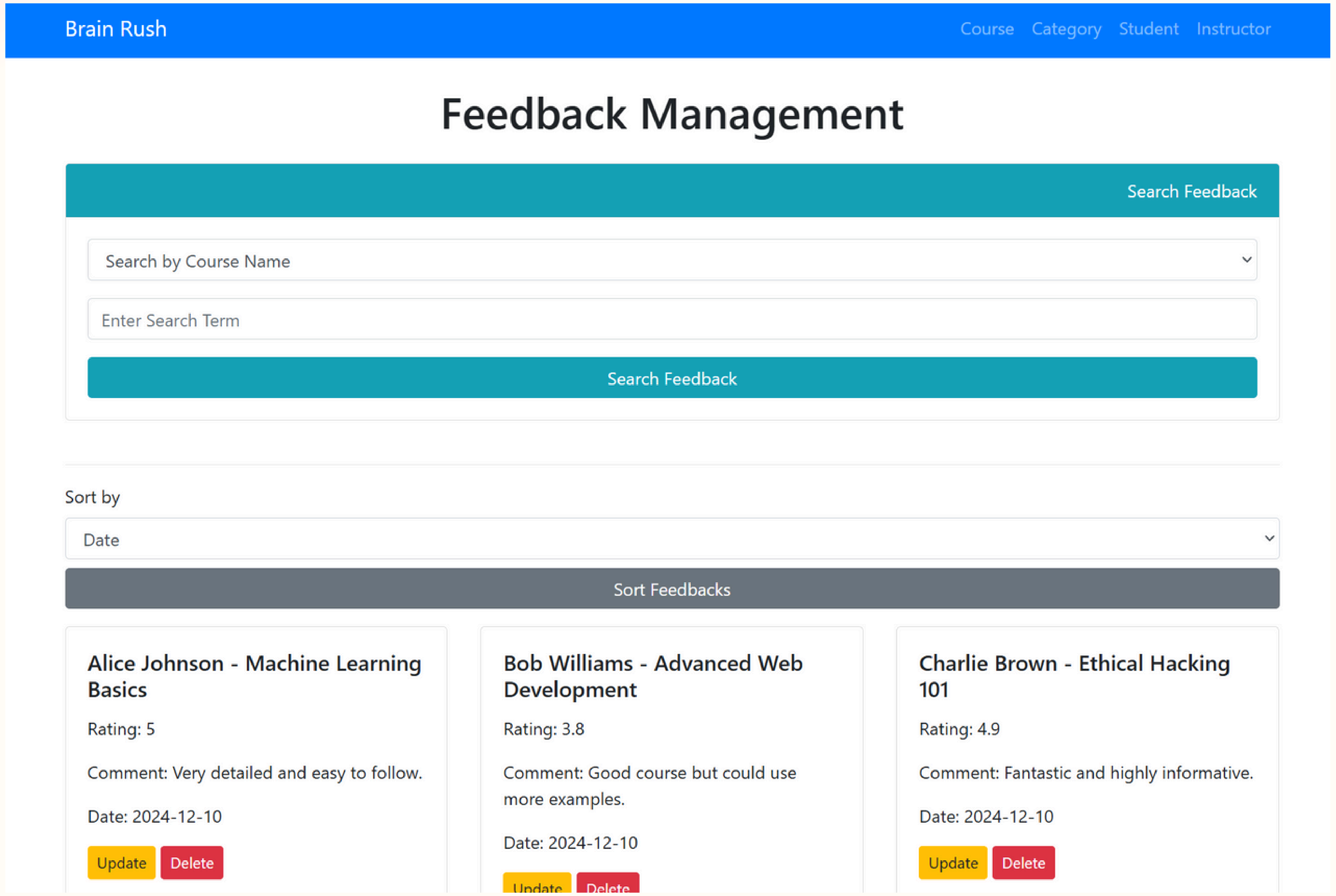
Agile Project Management
Master agile project management methodologies.
Instructor: Emily Clark
Category: Project Management
Duration: 35 hour
Average Rating: 0

Machine Learning Basics
An introduction to machine learning concepts.
Instructor: John Doe
Category: Data Science
Duration: 40 hour
Average Rating: 0

FEEDBACK PAGE

http://localhost:8888/feedback

The Feedback Management page is a tool designed to help users easily manage course and student feedback records. This guide provides an overview of the main features, instructions on how to navigate and use the page, error handling, and technical notes.


Brain Rush

Feedback Management

Alice Johnson - Machine Learning Basics

Rating: 5

Comment: Very detailed and easy to follow.

Date: 2024-12-10

Bob Williams - Advanced Web Development

Rating: 3.8

Comment: Good course but could use more examples.

Date: 2024-12-10

Charlie Brown - Ethical Hacking 101

Rating: 4.9

Comment: Fantastic and highly informative.

Date: 2024-12-10

MAIN FEATURES

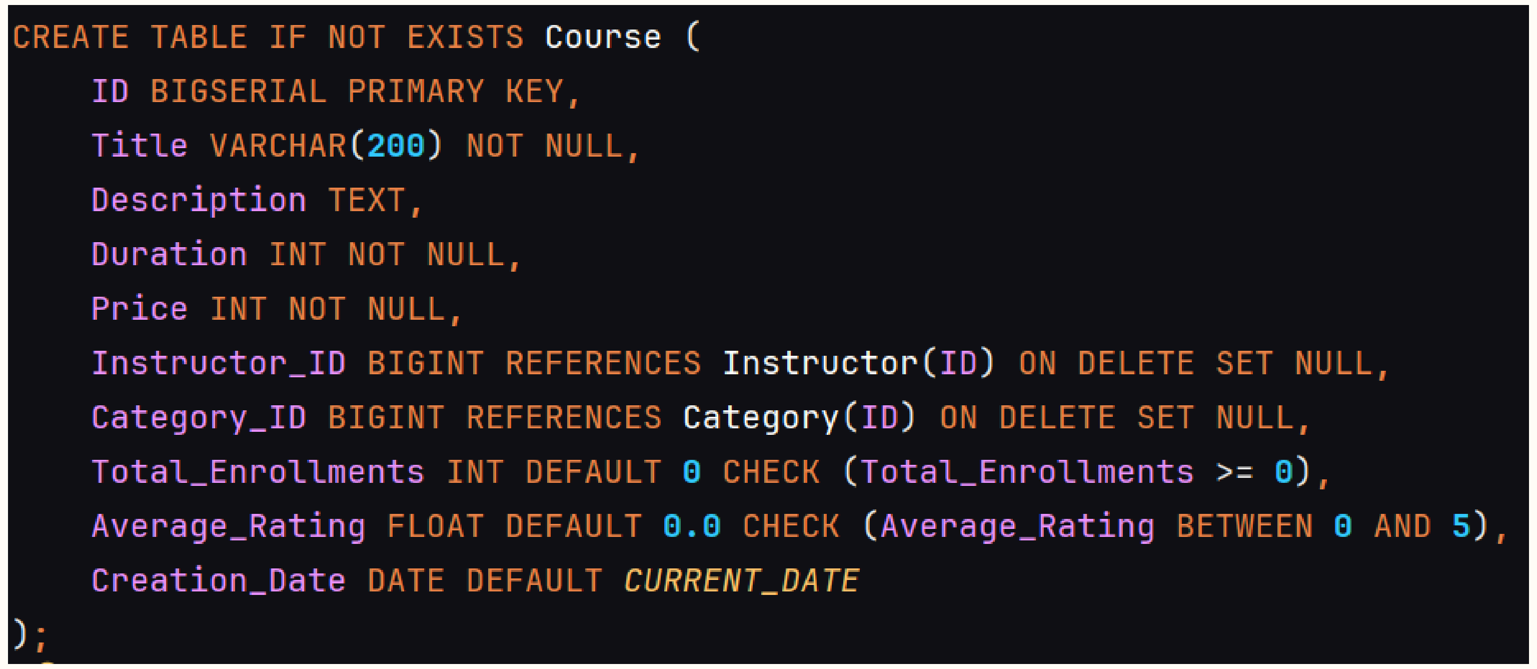
1. Search Feedback;
2. View All Feedbacks;
3. Update Feedback Information;
4. Delete Feedback;
3. Sort Feedback.

CHALLENGES

TOTAL_ENROLLMENTS

CHALLENGE

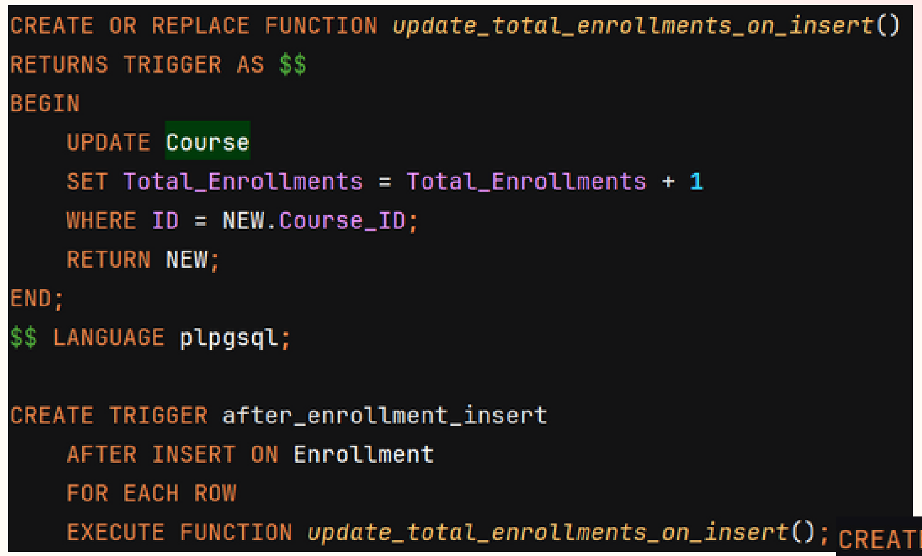
In our project, the Course table includes a Total_Enrollments attribute that tracks the number of students enrolled in each course. The challenge was ensuring this number updates automatically: increasing when a new student enrolls and decreasing when a student unenrolls.



```sql
CREATE TABLE IF NOT EXISTS Course (
    ID BIGSERIAL PRIMARY KEY,
    Title VARCHAR(200) NOT NULL,
    Description TEXT,
    Duration INT NOT NULL,
    Price INT NOT NULL,
    Instructor_ID BIGINT REFERENCES Instructor(ID) ON DELETE SET NULL,
    Category_ID BIGINT REFERENCES Category(ID) ON DELETE SET NULL,
    Total_Enrollments INT DEFAULT 0 CHECK (Total_Enrollments >= 0),
    Average_Rating FLOAT DEFAULT 0.0 CHECK (Average_Rating BETWEEN 0 AND 5),
    Creation_Date DATE DEFAULT CURRENT_DATE
);
```

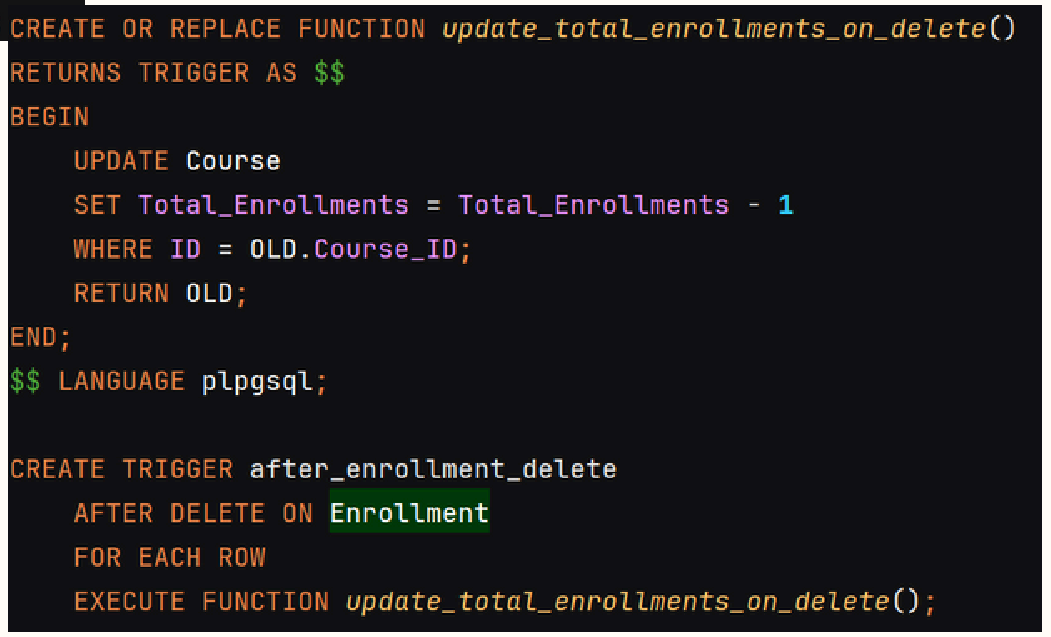
SOLUTION

To solve this, we implemented a trigger that adjusts the Total_Enrollments value whenever a record is added to or removed from the Enrollment table. This ensures the enrollment count stays accurate without requiring manual updates.



```sql
CREATE OR REPLACE FUNCTION update_total_enrollments_on_insert()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE Course
    SET Total_Enrollments = Total_Enrollments + 1
    WHERE ID = NEW.Course_ID;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER after_enrollment_insert
    AFTER INSERT ON Enrollment
    FOR EACH ROW
    EXECUTE FUNCTION update_total_enrollments_on_insert();
```



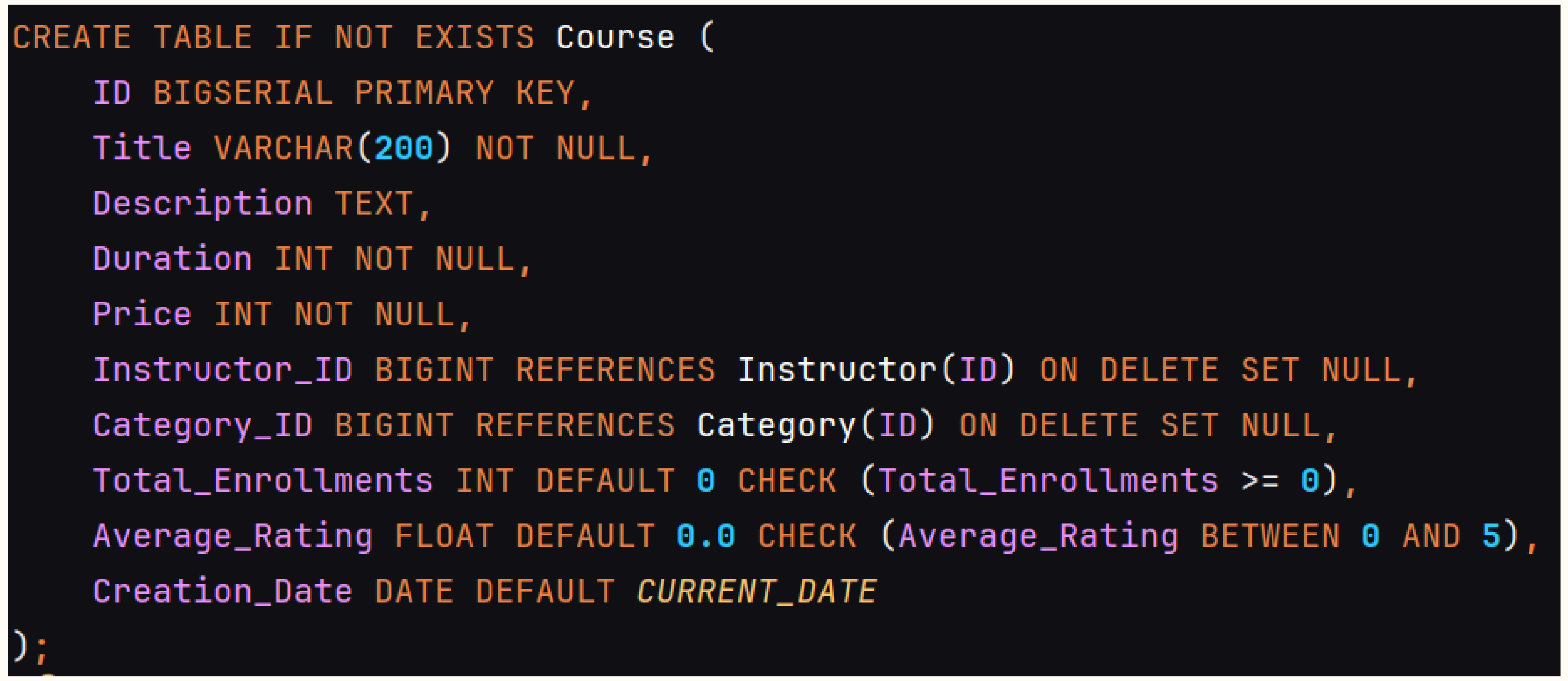
```sql
CREATE OR REPLACE FUNCTION update_total_enrollments_on_delete()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE Course
    SET Total_Enrollments = Total_Enrollments - 1
    WHERE ID = OLD.Course_ID;
    RETURN OLD;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER after_enrollment_delete
    AFTER DELETE ON Enrollment
    FOR EACH ROW
    EXECUTE FUNCTION update_total_enrollments_on_delete();
```

AVERAGE_RATING

CHALLENGE

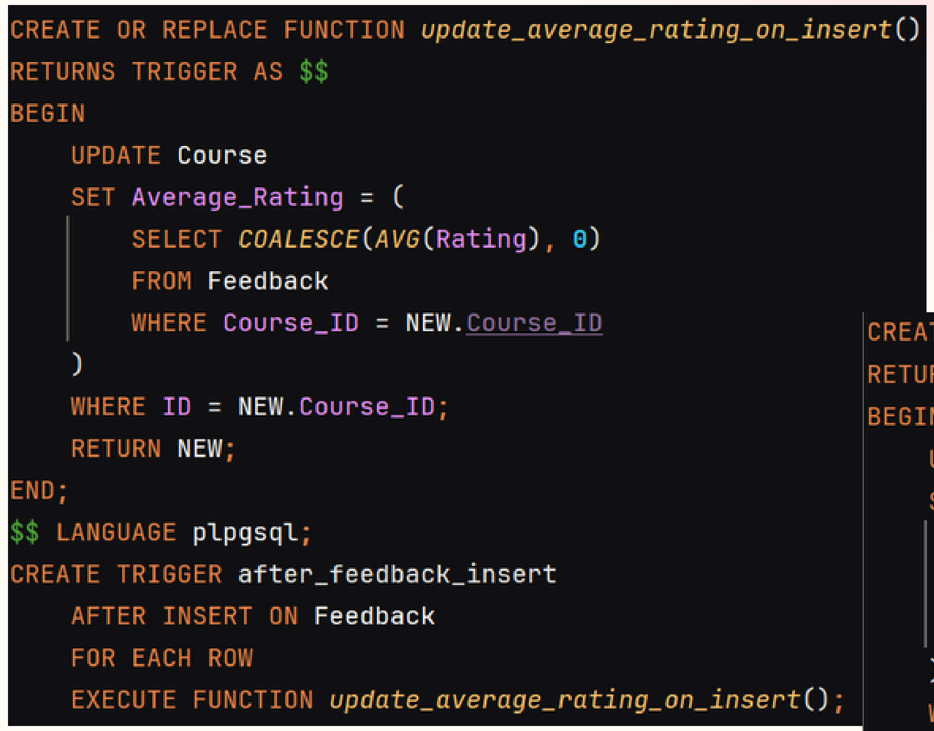
In our project, the Course table includes an Average_Rating attribute that stores the average rating of each course. The challenge was ensuring this value updates automatically whenever a student leaves feedback with a rating or deletes their feedback.



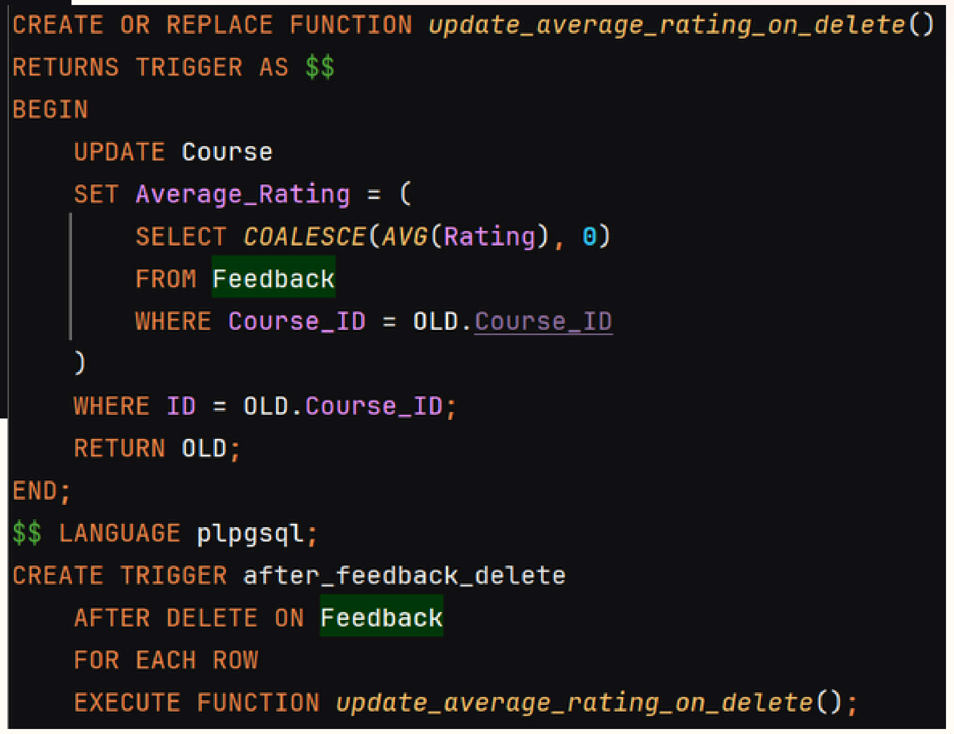
```sql
CREATE TABLE IF NOT EXISTS Course (
    ID BIGSERIAL PRIMARY KEY,
    Title VARCHAR(200) NOT NULL,
    Description TEXT,
    Duration INT NOT NULL,
    Price INT NOT NULL,
    Instructor_ID BIGINT REFERENCES Instructor(ID) ON DELETE SET NULL,
    Category_ID BIGINT REFERENCES Category(ID) ON DELETE SET NULL,
    Total_Enrollments INT DEFAULT 0 CHECK (Total_Enrollments >= 0),
    Average_Rating FLOAT DEFAULT 0.0 CHECK (Average_Rating BETWEEN 0 AND 5),
    Creation_Date DATE DEFAULT CURRENT_DATE
);
```

SOLUTION

To address this, we implemented a trigger that recalculates the Average_Rating whenever a record is added to or removed from the Feedback table. This ensures the rating remains accurate without requiring manual updates.



```sql
CREATE OR REPLACE FUNCTION update_average_rating_on_insert()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE Course
    SET Average_Rating = (
        SELECT COALESCE(AVG(Rating), 0)
        FROM Feedback
        WHERE Course_ID = NEW.Course_ID
    )
    WHERE ID = NEW.Course_ID;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER after_feedback_insert
    AFTER INSERT ON Feedback
    FOR EACH ROW
    EXECUTE FUNCTION update_average_rating_on_insert();
```



```sql
CREATE OR REPLACE FUNCTION update_average_rating_on_delete()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE Course
    SET Average_Rating = (
        SELECT COALESCE(AVG(Rating), 0)
        FROM Feedback
        WHERE Course_ID = OLD.Course_ID
    )
    WHERE ID = OLD.Course_ID;
    RETURN OLD;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER after_feedback_delete
    AFTER DELETE ON Feedback
    FOR EACH ROW
    EXECUTE FUNCTION update_average_rating_on_delete();
```

FUTURE IMPROVEMENTS

AUTHORIZATION

- **Enable Role-Based Authorization:** Implement a robust access control system to assign permissions based on user roles, ensuring secure and structured interactions.
- **Develop Login and Sign-Up Features:** Provide a seamless authentication process with secure email and password registration. Allow users to specify their roles during sign-up and enable token-based authentication for secure session management.
- **Define User Roles:** Establish distinct roles for Students, Instructors, and Admins, each with tailored access:
    - ***Students*** can enroll in courses, provide feedback, and track progress.
    - ***Instructors*** can create, manage, and monitor their courses.
    - ***Admins*** oversee platform management, including users, courses, and categories.

ENHANCED PERSONALIZATION

- Implement AI-driven recommendations to suggest courses based on student interests, performance, and learning history.
- Introduce personalized learning paths to guide students through progressive course levels.

GAMIFICATION FEATURES

- Add badges, points, and leaderboards to boost student engagement.
- Develop course completion certificates with customizable designs.

COURSE CREATION ENHANCEMENTS

- Offer instructors more tools to design engaging content, such as video editing and interactive modules.
- Enable peer reviews and collaboration for course development.

MOBILE APP DEVELOPMENT

- Build a mobile application for seamless access to courses on the go.
- Include offline learning capabilities for better accessibility.

MULTI-LANGUAGE SUPPORT

- Expand the platform’s reach by offering courses and interfaces in multiple languages.

SUBSCRIPTION AND PRICING MODELS

- Introduce flexible pricing options like subscriptions or pay-per-module systems.
- Offer discounts for students or bulk enrollments.

THANK YOU