Student Record Keeping System

Table of Contents

01

Intro and Background

Introduction

Brief Overview

- Centralized platform for managing academics, finances, housing, dining, and extracurriculars.
- User-friendly interface designed to simplify college life and reduce stress.

Motivation

- Wanted to simplify administrative processes at U.S. universities, benefiting both students and institutions.
- We have a personal connection to the project as students ourselves, so we wanted to address real challenges we face daily as college students.

Background and Related Work

Related work 1

Eludire (2011):

- Focuses on academic record management, emphasizing grades and class schedules, but does not address non-academic aspects like housing and dining.
- Our platform expands on this by integrating these services into a unified system.

Related work 2

Tamboli (2017):

- Automates institutional processes like course registration and student records, but lacks integration with student life services.
- Our platform enhances this by adding housing, dining, and financial features with personalized options.

Team Members

Frontend

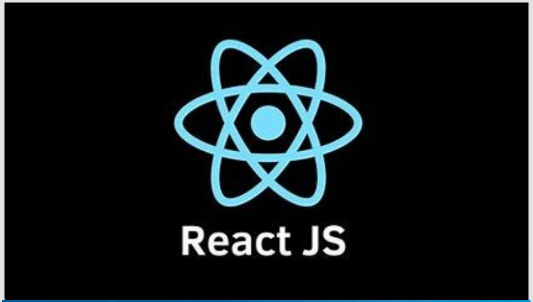
- Paribesh
- Justin
- Shubham
- Vaishnavi

Backend

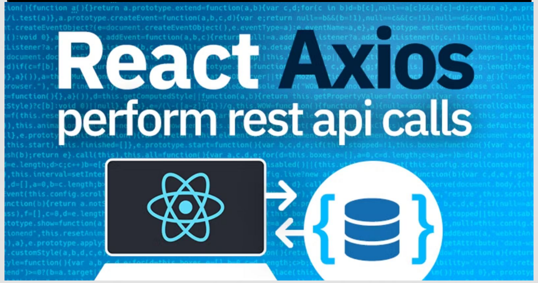
- Abhishek
- Anveetha
- Mia
- Abbas
- Ahimsa

Tech Stack

Frontend



User Interface



Connection

Backend

Google Authorization

Database

02

EER Diagrams

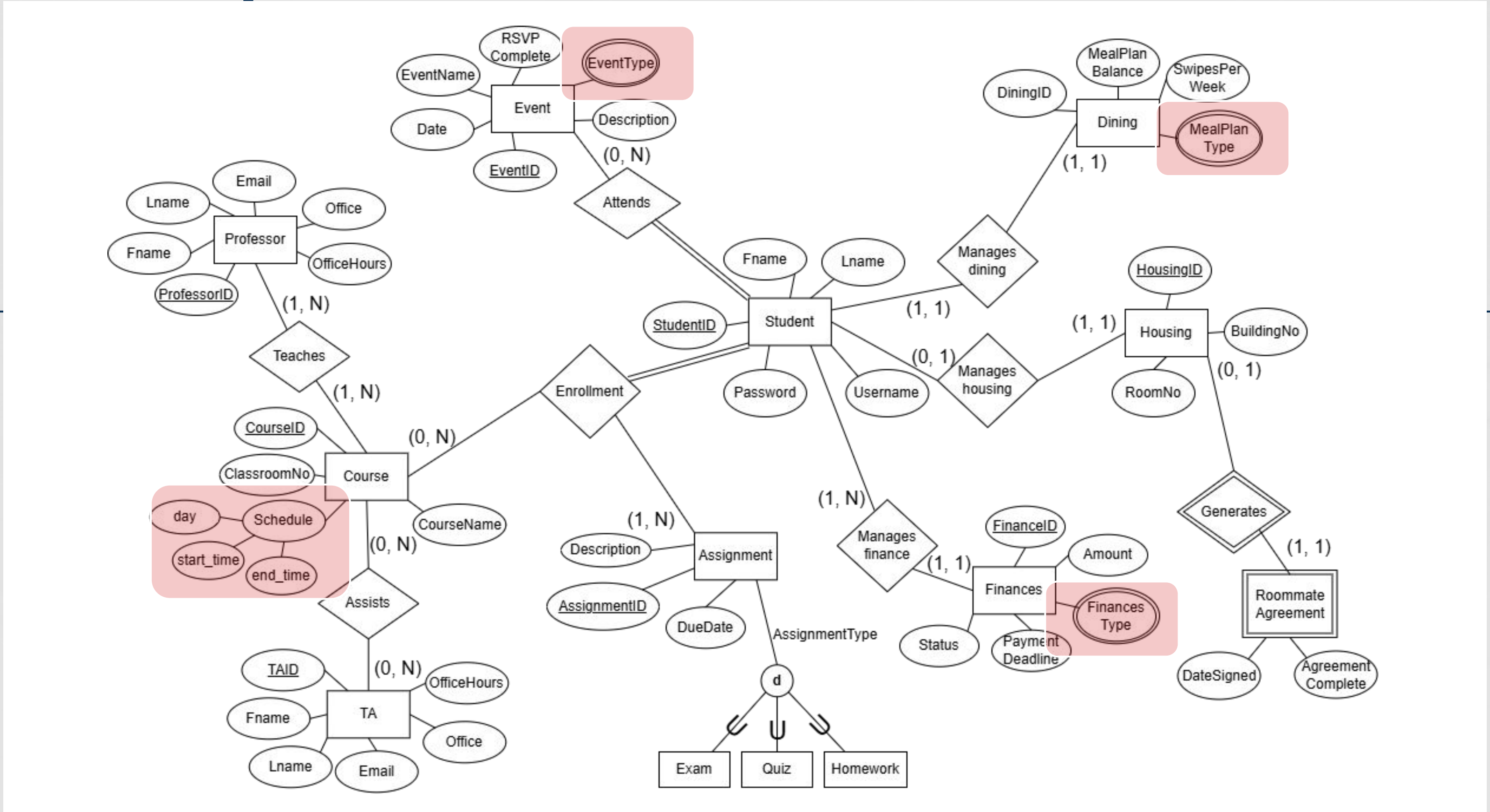
Data Model Overview

A record keeping system for students should contain these tables:

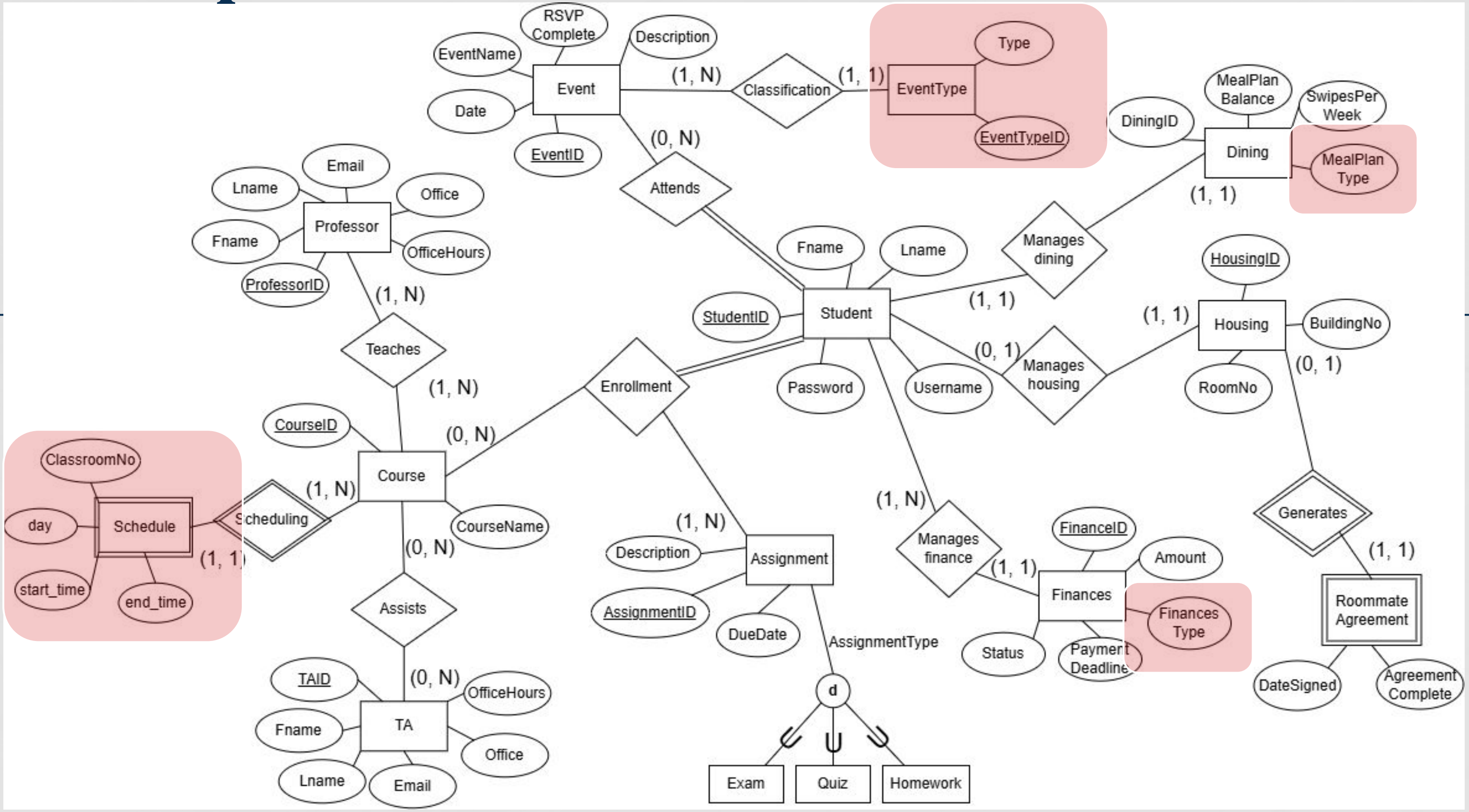
1. **Event**
    a. Club, volunteering, etc events that a student plans to attend
2. **Course**
3. **Professor** and **TA**
4. **Assignment**
5. **Finances**
6. **Dining**
7. **Housing**
8. **Student**
    a. Account information per user so that their information is associated with their specific account

EER Diagrams (Conceptual)

Conceptual Data Model Before Normalization

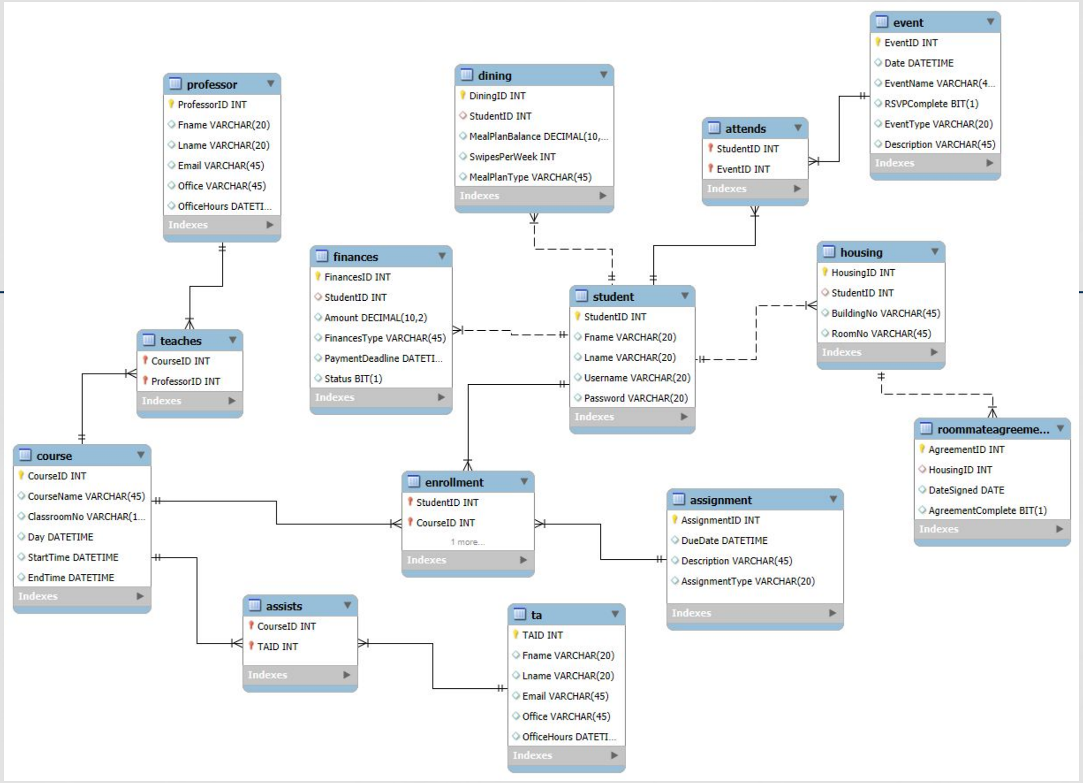


Conceptual Data Model After Normalization

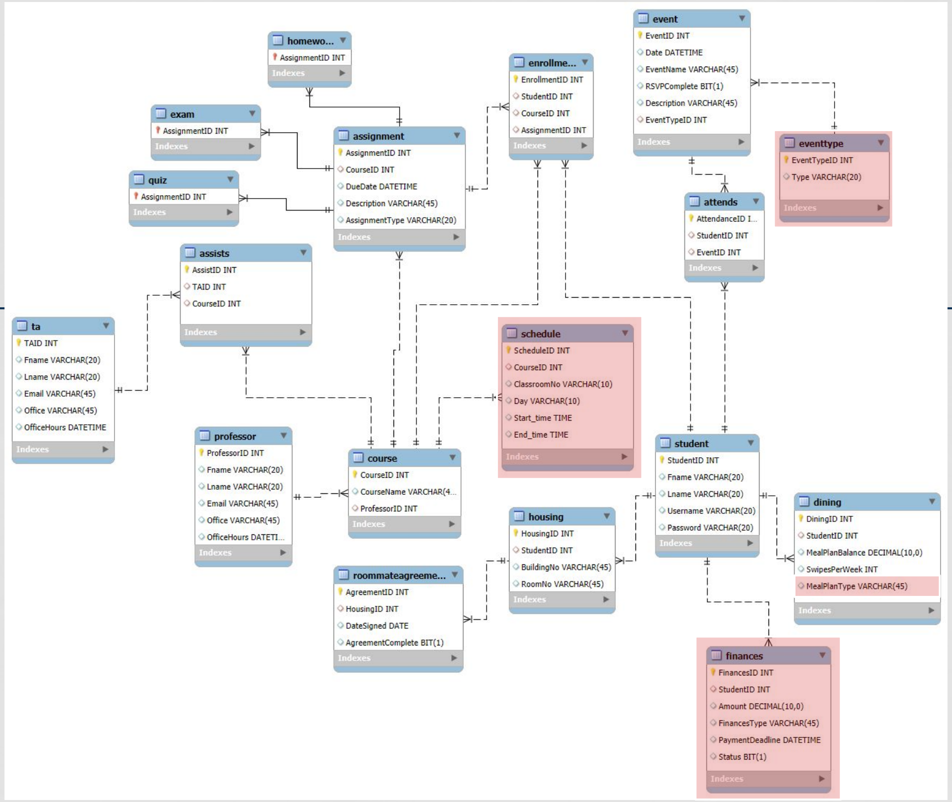


EER Diagrams (Relational)

Relational Data Model Before Normalization



Relational Data Model After Normalization

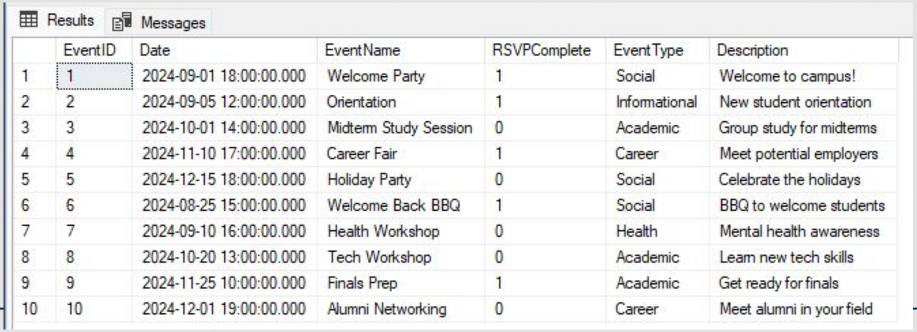


03

Tables and Queries

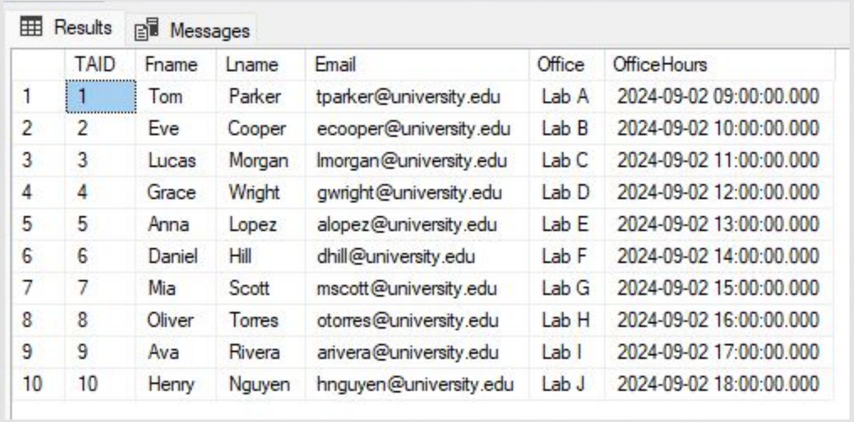
Populated Tables Before Normalization

- These tables have either multivalued attributes or have no relation to the other tables in the database



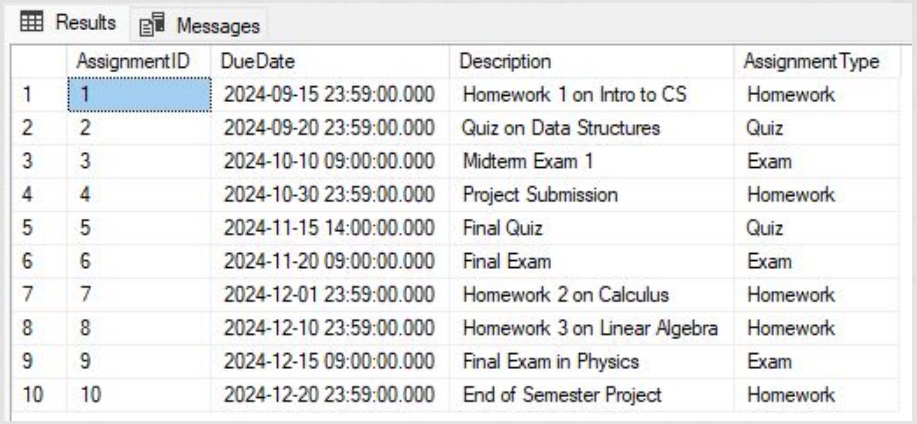
Results | Messages

	EventID	Date	EventName	RSVPComplete	EventType	Description
1	1	2024-09-01 18:00:00.000	Welcome Party	1	Social	Welcome to campus!
2	2	2024-09-05 12:00:00.000	Orientation	1	Informational	New student orientation
3	3	2024-10-01 14:00:00.000	Midterm Study Session	0	Academic	Group study for midterms
4	4	2024-11-10 17:00:00.000	Career Fair	1	Career	Meet potential employers
5	5	2024-12-15 18:00:00.000	Holiday Party	0	Social	Celebrate the holidays
6	6	2024-08-25 15:00:00.000	Welcome Back BBQ	1	Social	BBQ to welcome students
7	7	2024-09-10 16:00:00.000	Health Workshop	0	Health	Mental health awareness
8	8	2024-10-20 13:00:00.000	Tech Workshop	0	Academic	Learn new tech skills
9	9	2024-11-25 10:00:00.000	Finals Prep	1	Academic	Get ready for finals
10	10	2024-12-01 19:00:00.000	Alumni Networking	0	Career	Meet alumni in your field



Results | Messages

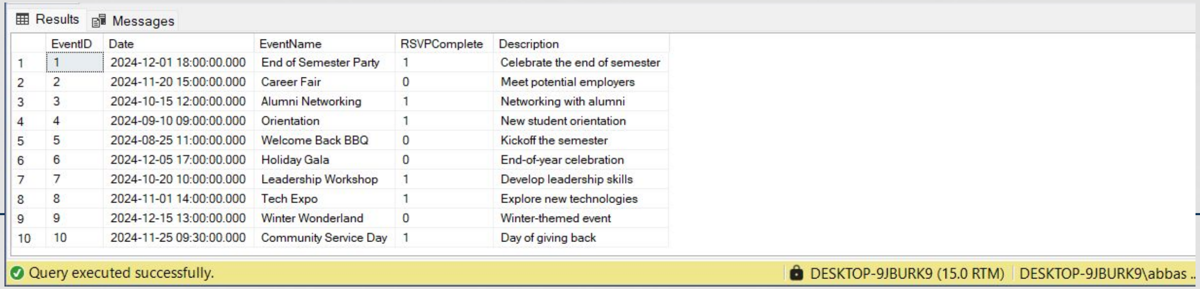
	TAID	Fname	Lname	Email	Office	OfficeHours
1	1	Tom	Parker	tparker@university.edu	Lab A	2024-09-02 09:00:00.000
2	2	Eve	Cooper	ecooper@university.edu	Lab B	2024-09-02 10:00:00.000
3	3	Lucas	Morgan	lmorgan@university.edu	Lab C	2024-09-02 11:00:00.000
4	4	Grace	Wright	gwright@university.edu	Lab D	2024-09-02 12:00:00.000
5	5	Anna	Lopez	alopez@university.edu	Lab E	2024-09-02 13:00:00.000
6	6	Daniel	Hill	dhill@university.edu	Lab F	2024-09-02 14:00:00.000
7	7	Mia	Scott	mscott@university.edu	Lab G	2024-09-02 15:00:00.000
8	8	Oliver	Torres	otorres@university.edu	Lab H	2024-09-02 16:00:00.000
9	9	Ava	Rivera	arivera@university.edu	Lab I	2024-09-02 17:00:00.000
10	10	Henry	Nguyen	hnguyen@university.edu	Lab J	2024-09-02 18:00:00.000



Results | Messages

	AssignmentID	DueDate	Description	AssignmentType
1	1	2024-09-15 23:59:00.000	Homework 1 on Intro to CS	Homework
2	2	2024-09-20 23:59:00.000	Quiz on Data Structures	Quiz
3	3	2024-10-10 09:00:00.000	Midterm Exam 1	Exam
4	4	2024-10-30 23:59:00.000	Project Submission	Homework
5	5	2024-11-15 14:00:00.000	Final Quiz	Quiz
6	6	2024-11-20 09:00:00.000	Final Exam	Exam
7	7	2024-12-01 23:59:00.000	Homework 2 on Calculus	Homework
8	8	2024-12-10 23:59:00.000	Homework 3 on Linear Algebra	Homework
9	9	2024-12-15 09:00:00.000	Final Exam in Physics	Exam
10	10	2024-12-20 23:59:00.000	End of Semester Project	Homework

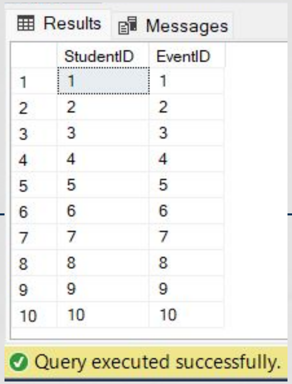
Populated Tables After Normalization



Results | Messages

	EventID	Date	EventName	RSVPComplete	Description
1	1	2024-12-01 18:00:00.000	End of Semester Party	1	Celebrate the end of semester
2	2	2024-11-20 15:00:00.000	Career Fair	0	Meet potential employers
3	3	2024-10-15 12:00:00.000	Alumni Networking	1	Networking with alumni
4	4	2024-09-10 09:00:00.000	Orientation	1	New student orientation
5	5	2024-08-25 11:00:00.000	Welcome Back BBQ	0	Kickoff the semester
6	6	2024-12-05 17:00:00.000	Holiday Gala	0	End-of-year celebration
7	7	2024-10-20 10:00:00.000	Leadership Workshop	1	Develop leadership skills
8	8	2024-11-01 14:00:00.000	Tech Expo	1	Explore new technologies
9	9	2024-12-15 13:00:00.000	Winter Wonderland	0	Winter-themed event
10	10	2024-11-25 09:30:00.000	Community Service Day	1	Day of giving back

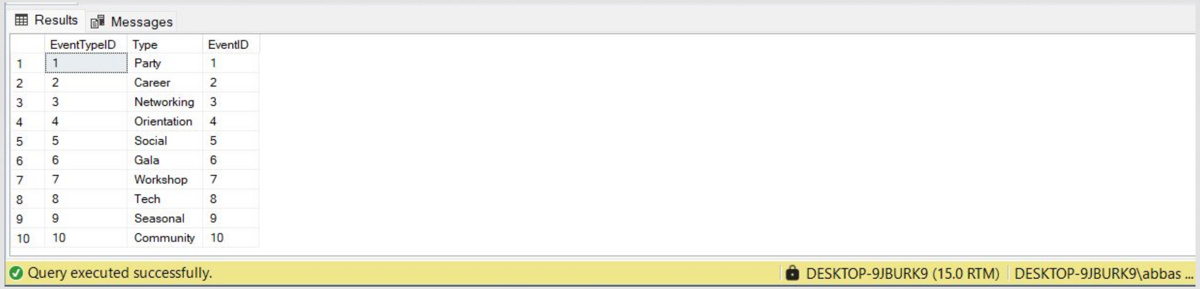
Query executed successfully. DESKTOP-9JBURK9 (15.0 RTM) DESKTOP-9JBURK9\abbas ...



Results | Messages

	StudentID	EventID
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	9	9
10	10	10

Query executed successfully.



Results | Messages

	EventTypeID	Type	EventID
1	1	Party	1
2	2	Career	2
3	3	Networking	3
4	4	Orientation	4
5	5	Social	5
6	6	Gala	6
7	7	Workshop	7
8	8	Tech	8
9	9	Seasonal	9
10	10	Community	10

Query executed successfully. DESKTOP-9JBURK9 (15.0 RTM) DESKTOP-9JBURK9\abbas ...

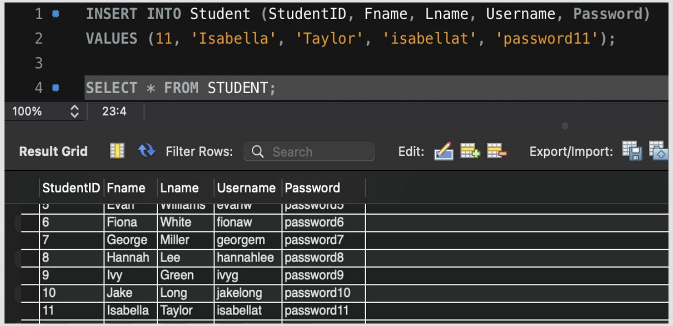
- After, normalization, there is less redundancy and more defined relations between Student and Event table

Query Executions

These operations ensure that the database handles critical actions like retrieving, inserting, updating, and deleting data efficiently for each table:

Data Retrieval (Query):
Retrieve specific data from tables using SELECT statements to ensure the database structure supports optimized access to required information.

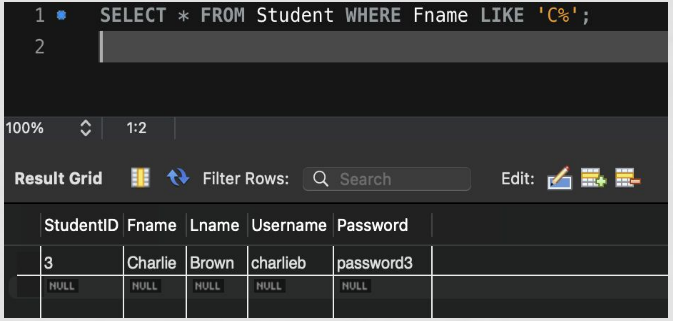
Data Insertion:
INSERT statements to add new records to the database tables, demonstrating the ability to expand the dataset.



```sql
INSERT INTO Student (StudentID, Fname, Lname, Username, Password)
VALUES (11, 'Isabella', 'Taylor', 'isabellat', 'password11');

SELECT * FROM STUDENT;
```

StudentID	Fname	Lname	Username	Password
5	Evan	Williams	evanw	password5
6	Fiona	White	fionaw	password6
7	George	Miller	georgem	password7
8	Hannah	Lee	hannahlee	password8
9	Ivy	Green	ivyg	password9
10	Jake	Long	jakelong	password10
11	Isabella	Taylor	isabellat	password11



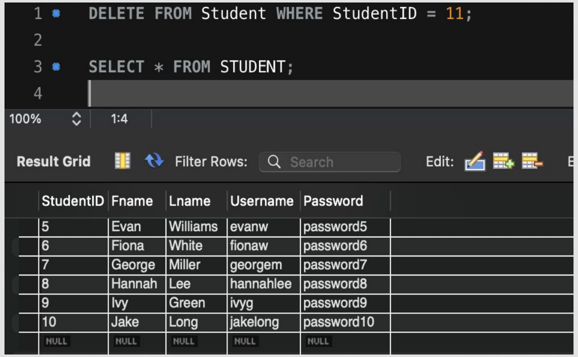
```sql
SELECT * FROM Student WHERE Fname LIKE 'C%';
```

StudentID	Fname	Lname	Username	Password
3	Charlie	Brown	charlieb	password3
NULL	NULL	NULL	NULL	NULL

Query Executions

Data Modification (Update):
Update existing records to reflect changes or correct inaccuracies in the stored information.

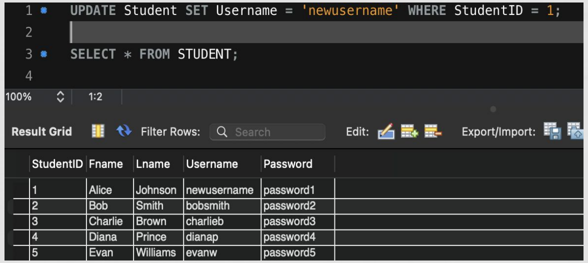
Data Deletion:
DELETE statements to remove unnecessary or outdated records, maintaining data relevance and database performance.



```
DELETE FROM Student WHERE StudentID = 11;

SELECT * FROM STUDENT;
```

StudentID	Fname	Lname	Username	Password
5	Evan	Williams	evanw	password5
6	Fiona	White	fionaw	password6
7	George	Miller	georgem	password7
8	Hannah	Lee	hannahlee	password8
9	Ivy	Green	ivyg	password9
10	Jake	Long	jakelong	password10
NULL	NULL	NULL	NULL	NULL



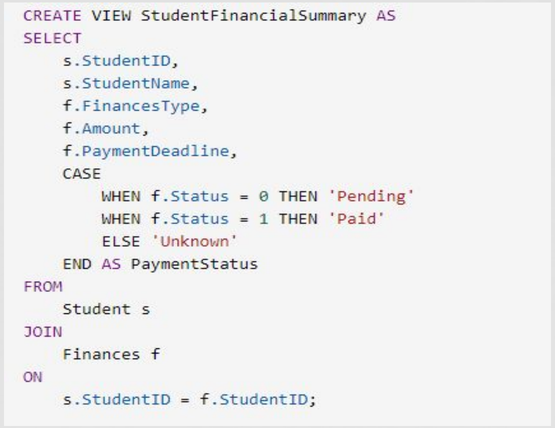
```
UPDATE Student SET Username = 'newusername' WHERE StudentID = 1;

SELECT * FROM STUDENT;
```

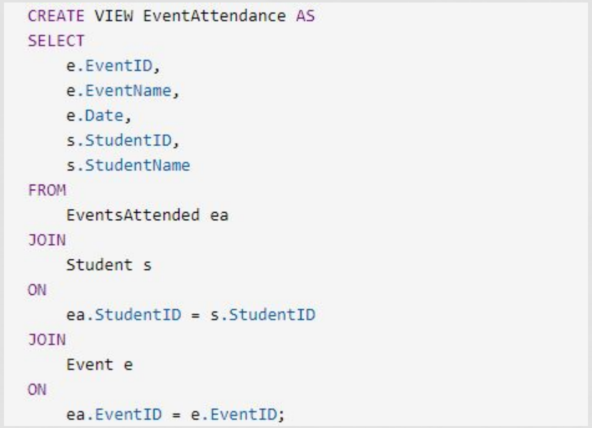
StudentID	Fname	Lname	Username	Password
1	Alice	Johnson	newusername	password1
2	Bob	Smith	bobsmith	password2
3	Charlie	Brown	charlieb	password3
4	Diana	Prince	dianap	password4
5	Evan	Williams	evanw	password5

Views

- Student financial summary - This view simplifies tracking student financial records and is useful for administrators or financial officers.
- Event Attendance - This view helps in monitoring student participation in campus events and is useful for event organizers to analyze attendance trends.



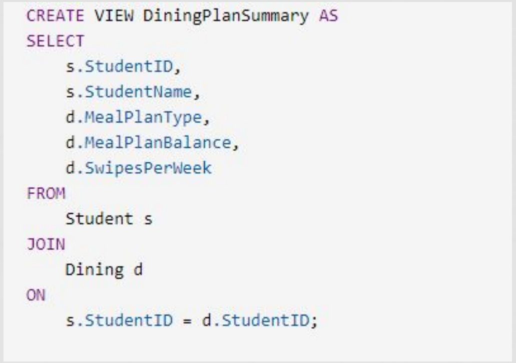
```sql
CREATE VIEW StudentFinancialSummary AS
SELECT
    s.StudentID,
    s.StudentName,
    f.FinancesType,
    f.Amount,
    f.PaymentDeadline,
    CASE
        WHEN f.Status = 0 THEN 'Pending'
        WHEN f.Status = 1 THEN 'Paid'
        ELSE 'Unknown'
    END AS PaymentStatus
FROM
    Student s
JOIN
    Finances f
ON
    s.StudentID = f.StudentID;
```



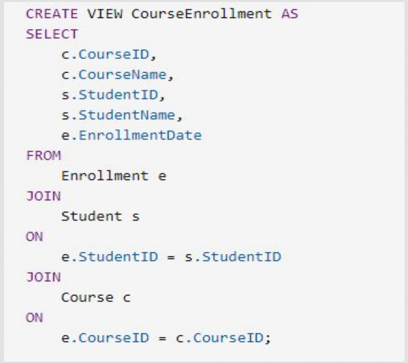
```sql
CREATE VIEW EventAttendance AS
SELECT
    e.EventID,
    e.EventName,
    e.Date,
    s.StudentID,
    s.StudentName
FROM
    EventsAttended ea
JOIN
    Student s
ON
    ea.StudentID = s.StudentID
JOIN
    Event e
ON
    ea.EventID = e.EventID;
```

Views

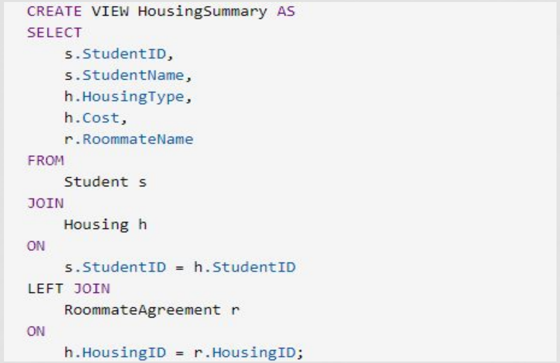
- Dining Plan Summary - This view is helpful for dining services to manage and assess the usage of meal plans by students.
- Course Enrollment - This view is useful for professors or administrative staff to get a quick overview of student enrollments and manage course rosters effectively.
- Housing Summary - This view is valuable for housing staff to monitor student accommodations and resolve housing-related inquiries.



```
CREATE VIEW DiningPlanSummary AS
SELECT
    s.StudentID,
    s.StudentName,
    d.MealPlanType,
    d.MealPlanBalance,
    d.SwipesPerWeek
FROM
    Student s
JOIN
    Dining d
ON
    s.StudentID = d.StudentID;
```



```
CREATE VIEW CourseEnrollment AS
SELECT
    c.CourseID,
    c.CourseName,
    s.StudentID,
    s.StudentName,
    e.EnrollmentDate
FROM
    Enrollment e
JOIN
    Student s
ON
    e.StudentID = s.StudentID
JOIN
    Course c
ON
    e.CourseID = c.CourseID;
```



```
CREATE VIEW HousingSummary AS
SELECT
    s.StudentID,
    s.StudentName,
    h.HousingType,
    h.Cost,
    r.RoommateName
FROM
    Student s
JOIN
    Housing h
ON
    s.StudentID = h.StudentID
LEFT JOIN
    RoommateAgreement r
ON
    h.HousingID = r.HousingID;
```

04

Frontend

Explanation of Frontend Design

The frontend design of our application focuses on creating a clean, interactive, and user-friendly interface for managing a database. It allows a user to perform CRUD (Create, Read, Update, Delete) operations on a database using SQL queries through dedicated pages. Each page is designed with simplicity in mind, ensuring smooth navigation and interaction, supported by a responsive layout. Here's a breakdown:

1. **Separation of Concerns:**
   - Each page (Query, Insert, Update, Delete) handles a specific operation, with individual components and CSS files for styling.
   - React Router is used to define routes and manage navigation between pages.
2. **Dynamic Interaction:**
   - Textareas are used for inputting SQL queries, providing placeholders to guide users.
   - Form submissions are handled using useState for real-time feedback and Axios for communication with the backend.
3. **Error and Success Feedback:**
   - Users receive clear success or error messages based on query execution results.
   - Query results (for **SELECT**) are displayed in a dynamic table format.

Core Features

1. **Welcome Page**
   - Displays user details after login (profile picture, name).
   - Includes a **Google Sign-In** button for login and logout functionality.
2. **Query Page**
   - Executes SELECT queries to fetch data from the database.
   - Results are dynamically rendered in a table with proper headings.
3. **Insert Page**
   - Executes INSERT queries to add new rows to the database.
   - Displays the result of the insertion and resets the input field.
4. **Update Page**
   - Executes UPDATE queries to modify existing data.
   - Shows the before and after for all updated rows.
5. **Delete Page**
   - Executes DELETE queries to remove records from the database.
   - Displays the number of rows deleted after execution and confirms the deletion.
6. **Logout (Ouit Page)**
   - Logs the user out of the system and redirects them to the Quit page.

Google Authentication (Login and Logout)

Why We Included Google Authentication

1. **Secure Access Control:**
   - Ensures that only authorized users can access the application, protecting the database from unauthorized queries or malicious users.
2. **Ease of Use:**
   - Google Sign-In simplifies the login process by eliminating the need for users to remember additional credentials.
3. **Enhanced User Experience:**
   - Displays user-specific details (such as name, profile picture), creating a more personalized experience.
4. **Session Management:**
   - Simplifies user session handling with Firebase, allowing for seamless login/logout across sessions.

Execution of Queries

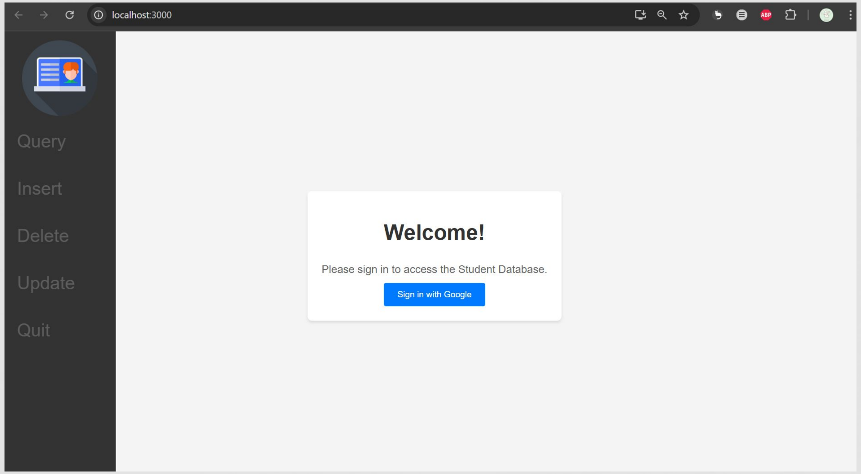
How It Works

1. **Frontend (React):**
   - Users input SQL queries (e.g., SELECT, INSERT, UPDATE, DELETE) into textareas on respective pages.
   - The form submission triggers an Axios POST request to the backend.
2. **Backend (Node.js/Express):**
   - The backend endpoint (e.g., /execute-query) receives the SQL query from the frontend.
   - It executes the query on the connected database (e.g., MySQL) using a database driver.
3. **Database Interaction:**
   - For SELECT: Retrieves results and sends them back to the frontend.
   - For INSERT, UPDATE, DELETE: Returns the number of affected rows.
4. **Frontend Feedback:**
   - The response is displayed to the user as:
     - **Query Results** (for SELECT queries).
     - **Success Messages** (e.g., "Successfully deleted X rows").
     - **Error Messages** (e.g., "Syntax error in the query").

05

Demo

Home page

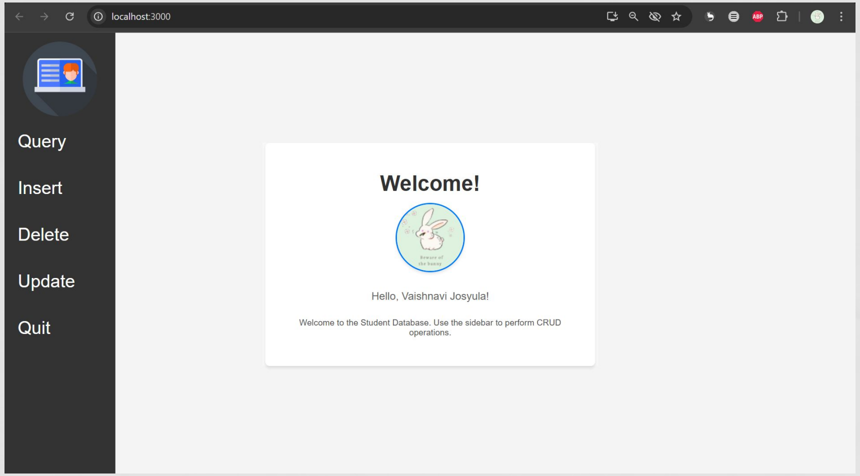

localhost:3000

Query
Insert
Delete
Update
Quit

Welcome!

Please sign in to access the Student Database.

Sign in with Google


localhost:3000

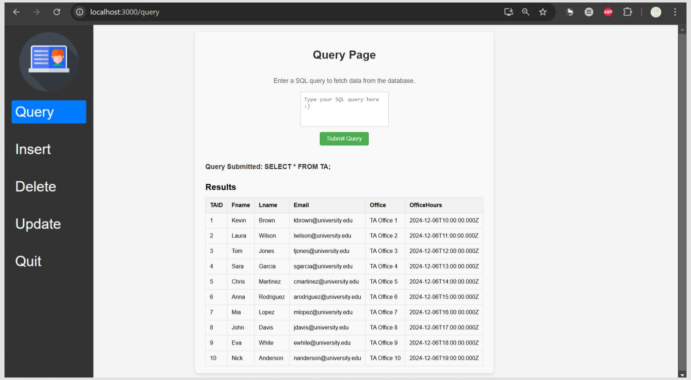
Query
Insert
Delete
Update
Quit

Welcome!

Hello, Vaishnavi Josyula!

Welcome to the Student Database. Use the sidebar to perform CRUD operations.

Query page



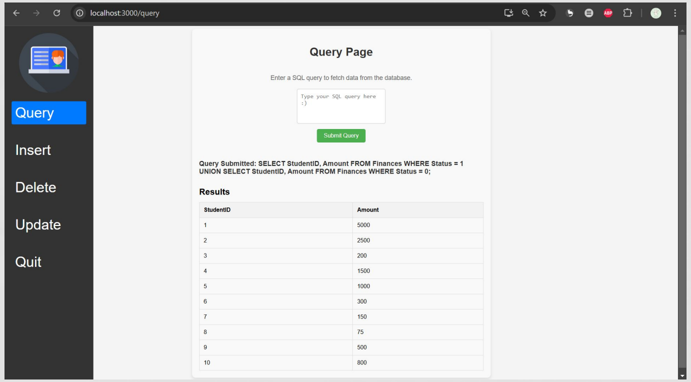
Query Page

Enter a SQL query to fetch data from the database.

Query Submitted: SELECT * FROM TA;

Results

TAID	Fname	Lname	Email	Office	OfficeHours
1	Kevin	Brown	kbrown@university.edu	TA Office 1	2024-12-06T10:00:00.000Z
2	Laura	Wilson	lwilson@university.edu	TA Office 2	2024-12-06T11:00:00.000Z
3	Tom	Jones	tjones@university.edu	TA Office 3	2024-12-06T12:00:00.000Z
4	Sara	Garcia	sgarcia@university.edu	TA Office 4	2024-12-06T13:00:00.000Z
5	Chris	Martinez	cmartinez@university.edu	TA Office 5	2024-12-06T14:00:00.000Z
6	Anna	Rodriguez	arodriguez@university.edu	TA Office 6	2024-12-06T15:00:00.000Z
7	Mia	Lopez	mlopez@university.edu	TA Office 7	2024-12-06T16:00:00.000Z
8	John	Davis	jdavis@university.edu	TA Office 8	2024-12-06T17:00:00.000Z
9	Eva	White	ewhite@university.edu	TA Office 9	2024-12-06T18:00:00.000Z
10	Nick	Anderson	nanderson@university.edu	TA Office 10	2024-12-06T19:00:00.000Z



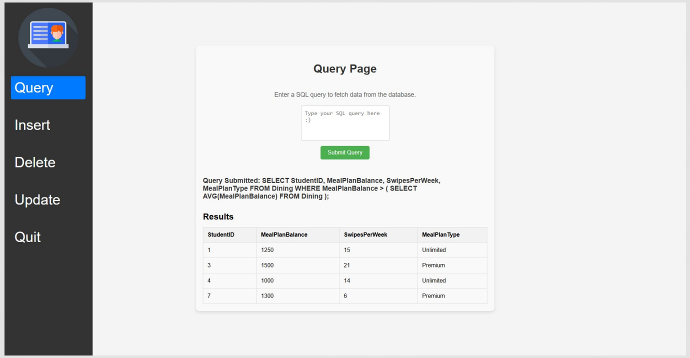
Query Page

Enter a SQL query to fetch data from the database.

Query Submitted: SELECT StudentID, Amount FROM Finances WHERE Status = 1 UNION SELECT StudentID, Amount FROM Finances WHERE Status = 0;

Results

StudentID	Amount
1	5000
2	2500
3	200
4	1500
5	1000
6	300
7	150
8	75
9	500
10	800



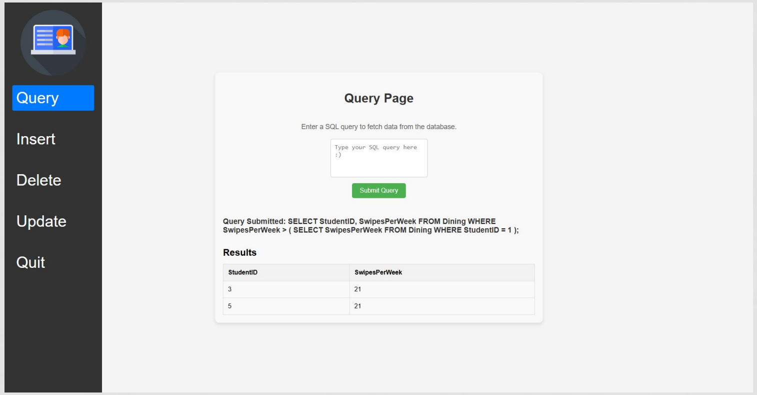
Query Page

Enter a SQL query to fetch data from the database.

Query Submitted: SELECT StudentID, MealPlanBalance, SwipesPerWeek, MealPlanType FROM Dining WHERE MealPlanBalance > (SELECT AVG(MealPlanBalance) FROM Dining);

Results

StudentID	MealPlanBalance	SwipesPerWeek	MealPlanType
1	1250	15	Unlimited
3	1500	21	Premium
4	1000	14	Unlimited
7	1300	6	Premium



Query Page

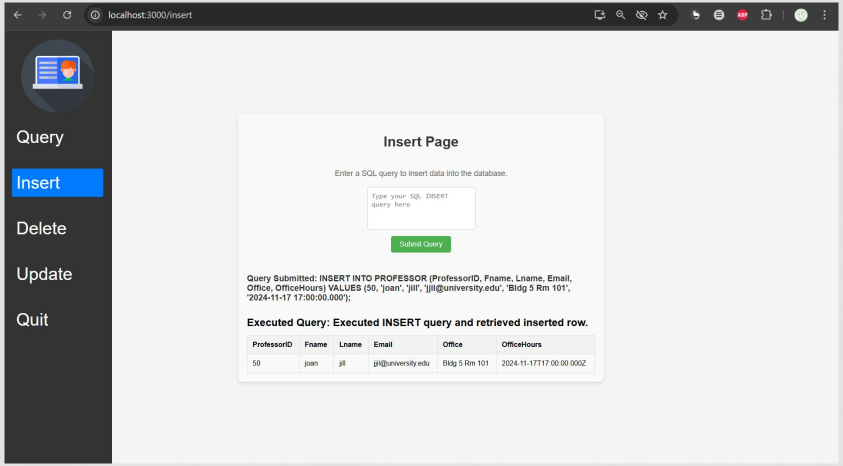
Enter a SQL query to fetch data from the database.

Query Submitted: SELECT StudentID, SwipesPerWeek FROM Dining WHERE SwipesPerWeek > (SELECT SwipesPerWeek FROM Dining WHERE StudentID = 1);

Results

StudentID	SwipesPerWeek
3	21
5	21

Insert page



Insert Page

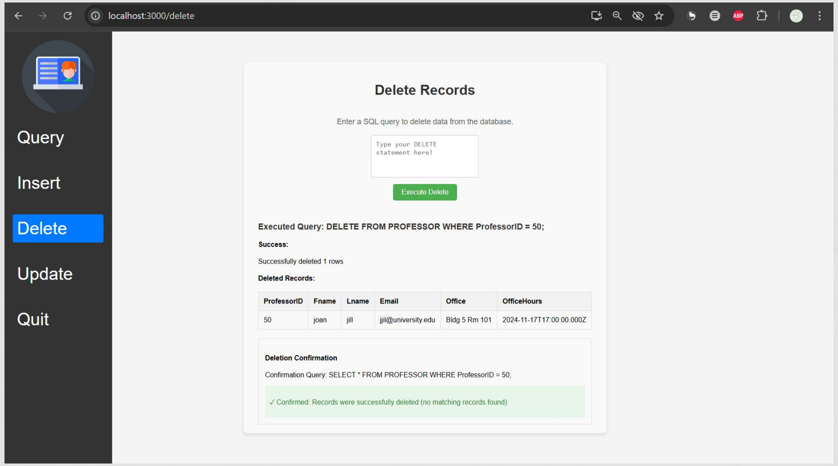
Enter a SQL query to insert data into the database.

Query Submitted: INSERT INTO PROFESSOR (ProfessorID, Fname, Lname, Email, Office, OfficeHours) VALUES (50, 'joan', 'jill', 'jjil@university.edu', 'Bldg 5 Rm 101', '2024-11-17 17:00:00.000');

Executed Query: Executed INSERT query and retrieved inserted row.

ProfessorID	Fname	Lname	Email	Office	OfficeHours
50	joan	jill	jjil@university.edu	Bldg 5 Rm 101	2024-11-17T17:00:00.000Z

Delete page



Delete Records

Enter a SQL query to delete data from the database.

Executed Query: DELETE FROM PROFESSOR WHERE ProfessorID = 50;

Success:

Successfully deleted 1 rows

Deleted Records:

ProfessorID	Fname	Lname	Email	Office	OfficeHours
50	joan	jill	jjil@university.edu	Bldg 5 Rm 101	2024-11-17T17:00:00.000Z

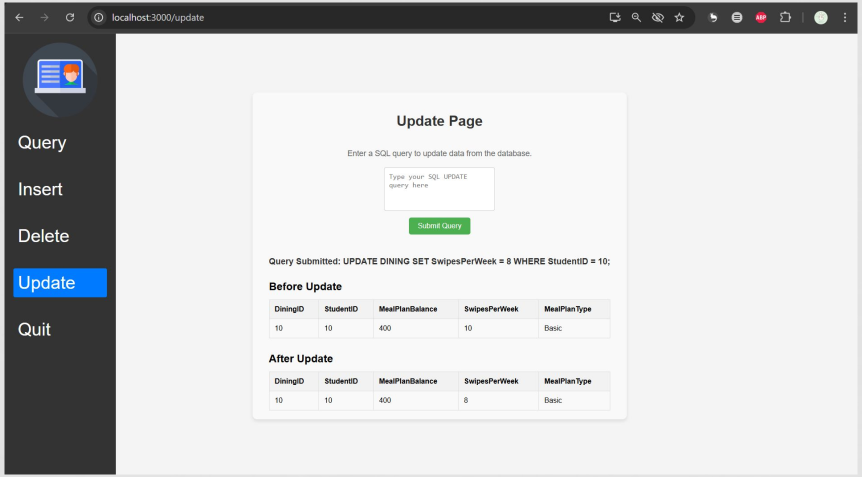
Deletion Confirmation

Confirmation Query: SELECT * FROM PROFESSOR WHERE ProfessorID = 50;

✓ Confirmed: Records were successfully deleted (no matching records found)

Update page

Quit page



Update Page

Enter a SQL query to update data from the database.

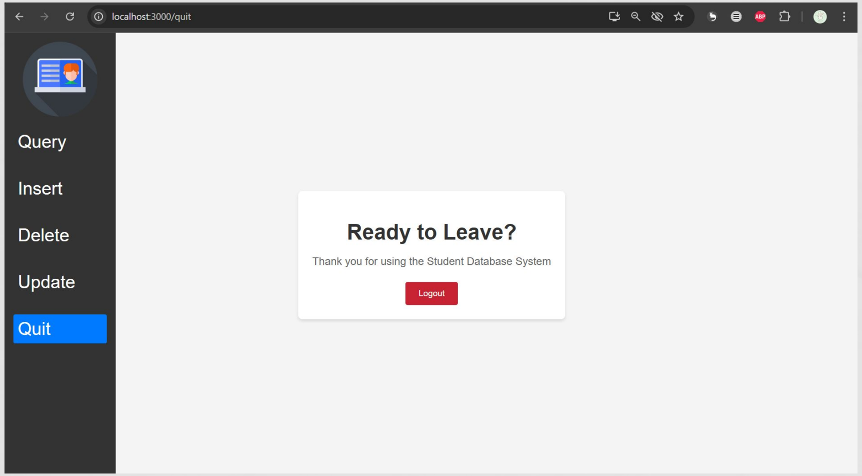
Query Submitted: UPDATE DINING SET SwipesPerWeek = 8 WHERE StudentID = 10;

Before Update

DiningID	StudentID	MealPlanBalance	SwipesPerWeek	MealPlanType
10	10	400	10	Basic

After Update

DiningID	StudentID	MealPlanBalance	SwipesPerWeek	MealPlanType
10	10	400	8	Basic



Ready to Leave?

Thank you for using the Student Database System

06

Conclusion

Competitors

Skyward

Popular student record management system that has unique features for administrators, teachers, and parents that is widely used by high schools alike.

They align user permissions across the board, while ours focuses on prioritizing data synchrony.

Powerschool

Comprehensive solution for managing student information that also has features for all parties to be able to engage in their needs with the information and is used by districts worldwide.

Roadblocks

Integration Issues

- Initial difficulty in understanding React.js concepts and workflows led to slower development and troubleshooting.
- Challenges in configuring and running the MSSQL server locally or in a suitable testing environment, especially for backend integrations with React.

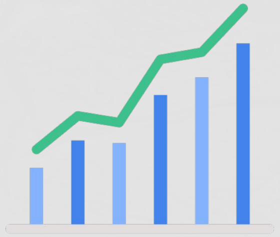
User Interface Design

- Organize the UI logically, ensuring that users can easily navigate between core features.
- Use a minimalist design to avoid clutter, making it easier for users to focus on the primary actions and content.

Future Goals

UI improvement

- More widgets/components to each page
  - Images, Graphs, Tables, etc.
- Links to each query page on the homepage.



Mobile App

- Alternative appeal to phones
- Competition against other UTD applications

Thanks!

Do you have any questions?

Work Cited

1. A. Eludire, “The Design and Implementation of Student Academic Record Management System,” International Journal of Computing and ICT Research, vol. 5, no. 2, pp. 20-25, 2011.

2. A. Tamboli, “Institute Administration Automation and Student Database Management System,” Journal of Automation and Control Engineering, vol. 5, no. 4, pp. 210-216, 2017.