LONDON METROPOLITAN UNIVERSITY

Islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CC6001NI - Advanced Database System Development

Assessment Weightage & Type

40% Individual Coursework

Year and Semester

2020-21 Autumn

Student Name: Suyogya Luitel

Group: C3

London Met ID: 19031784

College ID: NP01CP4A190035

Assignment Due Date: 23rd March 2022

Assignment Submission Date: 23rd March 2022

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

Table of Contents

Table of Figures

Table of Tables

1. Introduction

This is a report of the coursework involving analysis, design, and implementation of a web-based database application in accordance with the given business case study with the help of tools such as Oracle SQL Developer Data Modeler, Oracle SQL Developer and ASP.NET. According to the presented business scenario, the database should contain information regarding college departments; students, including fee status and attendance; assignments, and their results.

Initially the database is normalized based on given example data and integrated along with assumptions to better fit the given business scenario. The final entities and relations obtained from integration are visualized with Oracle Developer Data Modeler and a DDL script is generated. The generated DDL is then used to generate a database through Oracle SQL, which is then populated with sample data. Webforms are then created to perform CRUD operations on some tables and selective filtration operations on joined tables. Testing is then performed to make sure the system works fine.

2. Textual Analysis

From the given case study, I derived the following relations between entities:

2.1. Teacher-Module

A teacher is allowed to teach multiple modules.

A module can be taught by multiple teachers.

Hence, a many-to-many relation seems to be established.

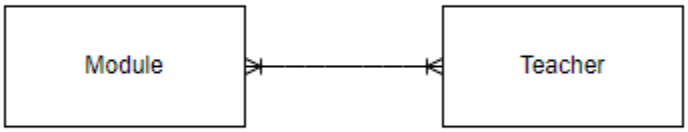


Figure 1: Relation between Module and Teacher

2.2. Student-Module

A student can study multiple modules.

A module can be studied by multiple students.

Hence a many-to-many relation seems to be established.

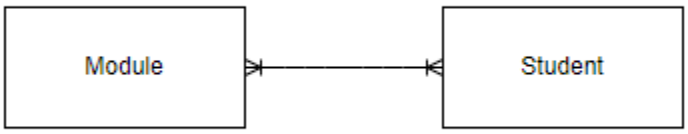


Figure 2: Relation between Module and Student

2.3. Student-Assignment

A student can partake in multiple assignments.

An assignment is submitted by multiple students.

Hence, a many-to-many relation is established.

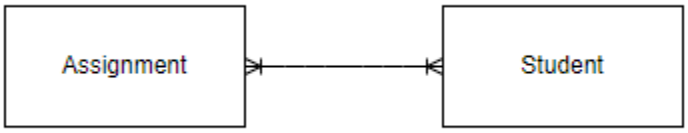


Figure 3: Relation between Student and Assignment

2.4. Department-Fees-Assignment-Result

A department manages multiple student fees.

A department manages multiple assignments.

A department manages multiple results.

Student fee is managed by a single department.

Assignment is managed by a single department.

Result is managed by a single department.

Hence, department establishes a one-to-many relation with fees, assignments, and results.

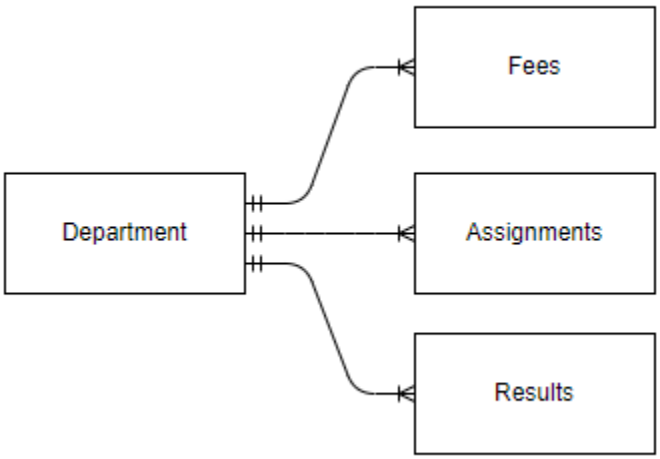


Figure 4: Relation between Department with Fees, Assignments and Results

3. ERD From Case Study

Based on the relations derived in the textual analysis of the given case study and some general assumptions, the following ERD is generated.

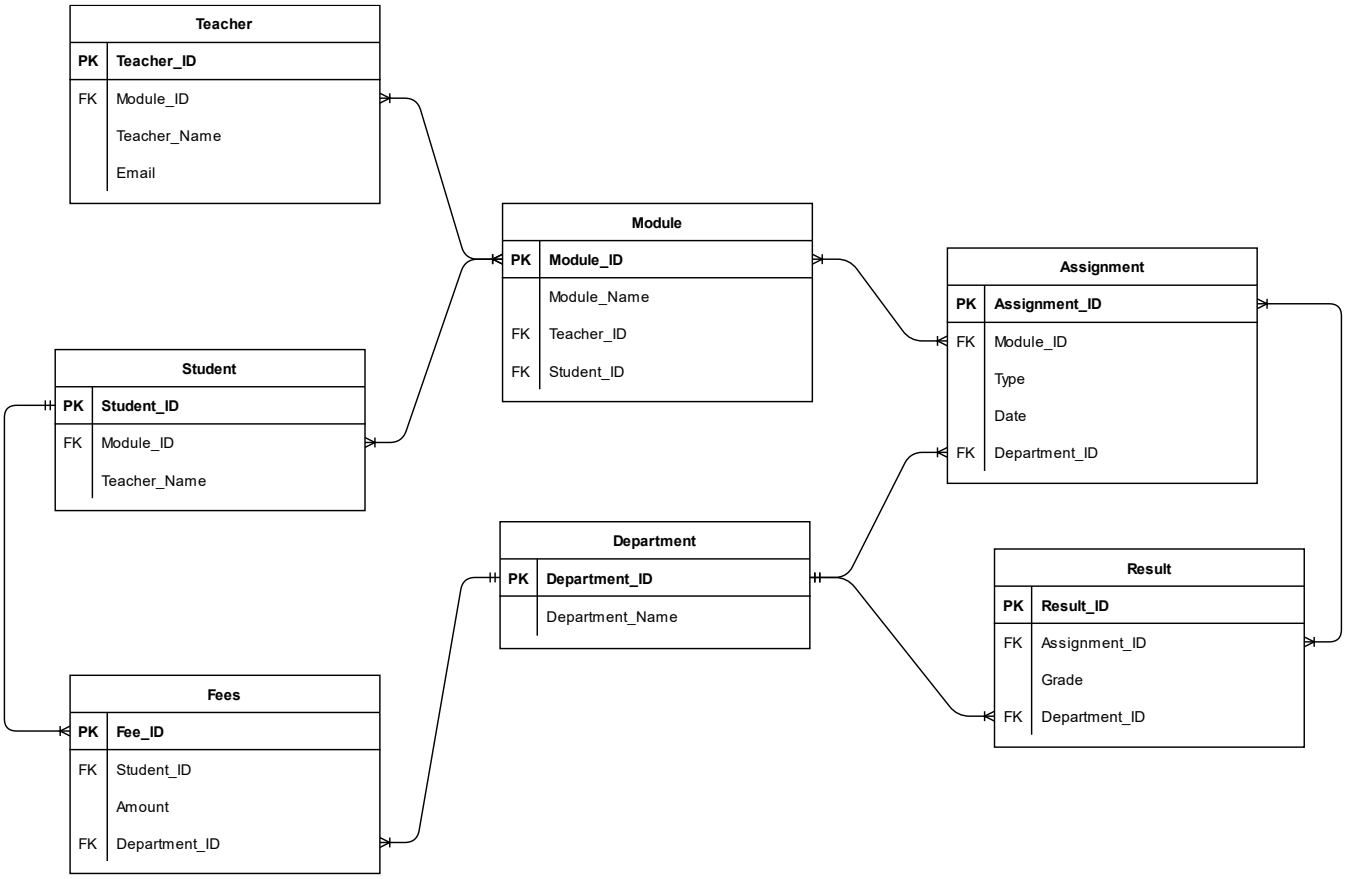


Figure 5: Initial ERD from Case Study

Assumptions:

- A student pays multiple fees throughout their stay in the college, and a fee belongs to only one student.
- Multiple results are published for the same assignment and a result relates to multiple assignments.
- A module may be included in multiple assignments and an assignment type may be implemented for multiple modules.

4. Normalization

4.1. Figure 1: Example of Teacher allocation list

S.N.	Teacher Name	Address	Email	Module Code	Module Name	Credit Hours
1	Saul Goodman	595 Green Lake Road Black Lake 9115 Lake Street Harrietsfield	Saulthegoodman@abc.edu.np	CC12	Data Structureand Algorithm	**30**
2	Walter White	696 Madison St. Pierrefonds	whitywalker@abc.edu.np	CC12	Data Structureand Algorithm	**30**
3	Santana Lopez	6 Valley View Street Griffintown	Santanalopez@abc.edu.np	CC49	Engineering Thermodynamic	**60**
4	Rust Cohle	89 Coffee Dr. Plaster Rock	rustycohle@abc.edu.np	SG101	Softwareengineer	**30**
				TG405	**Data Analysis**	**50**

Table 1: Example of Teacher allocation list

Normalization of teacher allocation:

4.1.1. UNF

Identifying the repeating groups among the given figure fields, we get the following UNF:

Teacher (Teacher_ID, Teacher_Name, {Street_No., Street_Name, State_Name}, Email, {Module_Code, Module_Name, Credit_Hours,})

4.1.2. 1NF

Separating the repeating groups identified in the UNF above and assigning composite keys, we get the following entities:

Teacher-1 (Teacher_ID, Teacher_Name, Email)
Address-1 (Address_ID, Teacher_ID*, Street_No., Street_Name, State_Name)
Module-1 (Module_Code, Teacher_ID*, Module_Name, Credit_Hours)

4.1.3. 2NF

After the repeating groups were separated into different entities and assigned new composite keys, they needed to be checked for partial dependencies.

For Teacher-1:

- No composite keys were present and hence there were no partial dependencies.

  Teacher_ID → Teacher_Name, Email

  Teacher-2 (Teacher_ID, Teacher_Name, Email)

For Address-1:

- Address_ID key determines Street_No., Street_Name, State_Name
- Address_ID, Teacher_ID* composite key determines nothing
- Teacher_ID* foreign key determines nothing

Address_ID → (Street_No., Street_Name, State_Name)
Address_ID, Teacher_ID* → ()
Teacher_ID → ()

The partial dependencies are separated into new entities as:

Address-2 (Address_ID*, Street_No., Street_Name, State_Name)
Address-Teacher-2 (Address_ID, Teacher_ID*)

For Module-1:

- Module_Code key determines Module_Name, Credit_Hours
- Module_Code, Teacher_ID* composite key determines nothing

  Module_Code → (Module_Name, Credit_Hours)

Module_Code, Teacher_ID* → ()

Module -2 (Module_Code, Module_Name, Credit_Hours)

Module-Teacher -2 (Module_Code, Teacher_ID*)

Hence, the results of 2NF are the following entities:

Teacher-2 (Teacher_ID, Teacher_Name, Email)

Address-2 (Address_ID→ Street_No., Street_Name, State_Name)

Address-Teacher-2 (Address_ID, Teacher_ID*)

Module -2 (Module_Code, Module_Name, Credit_Hours)

Module-Teacher -2 (Module_Code, Teacher_ID*)

4.1.4. 3NF

After checking and removing any partial dependencies, transitive dependencies are required to be checked and removed.

For Teacher-2:

- Teacher_ID determines Teacher_Name and Email
- Teacher_Name determines nothing
- Email determines nothing

There are no transitive dependencies.

Teacher-3 (Teacher_ID, Teacher_Name, Email)

For Address-2:

- Address_ID determines Street_No., Street_Name, State_Name
- Address determines nothing

There are no transitive dependencies.

Address-3 (Address_ID, Street_No., Street_Name, State_Name)

For Address-Teacher 2:

There are no transitive dependencies, the entity contains only a composite primary key.

Address-Teacher-3 (Address_ID, Teacher_ID*)

For Module-Teacher 2:

There are no transitive dependencies, the entity contains only a composite primary key.

Module-Teacher-3 (Module_ID, Teacher_ID*)

For Module-2:

- Module_Code determines Module_Name, Credit_Hours
- Module_Name determines nothing
- Credit_Hours determines nothing

Module_Code → (Module_Name, Credit_Hours)

The transitive dependencies are separated into entities as follows:

Module-3 → (Module_Code, Module_Name, Credit_Hours)

Hence, the results of 3NF are the following entities:

Teacher-3 (Teacher_ID, Teacher_Name, Email)
Address-3 (Address_ID, Address)
Address-Teacher-3 (Address_ID, Teacher_ID*)
Module-3 (Module_Code, Module_Name, Credit_Hours)
Module-Teacher-3 (Module_ID, Teacher_ID*)

4.1.5. Final entities from figure 1

After normalizing figure 1 up to third normal form, the following entities are obtained:

Teacher (Teacher_ID, Teacher_Name, Email)

Address (Address_ID, Street_No., Street_Name, State_Name)

Address-Teacher (Address_ID, Teacher_ID*)

Module (Module_Code, Module_Name, Credit_Hours)

Module-Teacher (Module_ID, Teacher_ID*)

4.2. Fig 2: Example of Assignment and Examination Results

Student ID: 149893
Student Name: Mr. William Ishee
Student Address: 2508 Shinn Street New York

Module Code	Module Name	Assignment Type	Grade	Status
CC12	Data Structure and Algorithm	Coursework	A	Pass
CC49	Engineering Thermodynamic	Coursework	B	Pass
CC49	Engineering Thermodynamic	Written Exam	F	Fail
SG101	Software engineer	Individual Assignment	B+	Pass
SG101	Software engineer	Group Assignment	B	Pass
SG101	Software engineer	Unseen Examination	A	Pass

Assumptions:

- A student cannot have multiple addresses.

4.2.1. UNF

Adding the above-mentioned fields to the given Figure 2 fields, and identifying the repeating groups among them, we get the following UNF:

Student (Student_ID, Sudent_Name, Street_No., Street_Name, State_Name, {Module_Code, Module_Name {Assignment_Type, Grade, Status}})

4.2.2. 1NF

Separating the repeating groups identified in the UNF above and assigning composite keys, we get the following entities:

Student-1 (Student_ID, Student_Name, Street_No., Street_Name, State_Name)
Module-1 (Module_Code, Student_ID*, Module_Name)
Assignment-1 (Assignment_ID, Student_ID*, Module_Code*, Assignment_Type, Grade, Status)

4.2.3. 2NF

After the repeating groups were separated into different entities and assigned new composite keys, they needed to be checked for partial dependencies.

For Student-1:

- No composite keys were present and hence there were no partial dependencies.

Student_ID → (Student_Name, Street_No., Street_Name, State_Name)

Student-2 (Student_ID, Student_Name, Street_No., Street_Name, State_Name)

For Module-1:

- Module_Code key determines Module_Name
- Module_Code, Student_ID* composite key determines nothing
- Student _ID* foreign key determines nothing

Student_ID, Module_Code → ()
Student_ID → ()
Module_Code → (Module_Name)

Module-Student-2 (Student_ID*, Module_Code*)
Module-2 (Module_Code, Module_Name)

For Assignment-1:

- Assignment_ID key determines Assigment_Type
- Assignment_ID, Student_ID*, Module_Code* composite key determines Grade, Status
- Student _ID* foreign key determines nothing
- Module _Code* foreign key determines nothing

Assignment_ID → (Assignment_Type)

Assignment_ID, Student_ID, Module_Code → (Grade, Status)

Student_ID → ()

Module_Code → ()

The partial dependencies are separated into new entities as:

Assignment-2 (Assignment_ID, Assignment_Type)

Student-Assignment-2 (Assignment_ID*, Student_ID*, Module_Code*, Grade, Status)

Hence, the results of 2NF are the following entities:

Student-2 (Student_ID, Student_Name, Street_No., Street_Name, State_Name)

Module-Student-2 (Student_ID*, Module_Code*)

Module-2 (Module_Code, Module_Name)

Assignment-2 (Assignment_ID, Assignment_Type)

Student-Assignment-2 (Assignment_ID*, Student_ID*, Module_Code*, Grade, Status)

4.2.4. 3NF

After checking and removing any partial dependencies, transitive dependencies are required to be checked and removed.

For Student-2 :

- Student_ID determines Student_Name, Street_No., Street_Name, State_Name Student_Name determines nothing
- Student_Address determines nothing

Student_ID → (Student_Name, Street_No., Street_Name, State_Name)

There are no transitive dependencies.

Student-3 (Student_ID, Student_Name, Street_No., Street_Name, State_Name)

For Module-Student-2:

There are no transitive dependencies, the entity contains only a composite primary key.

Module-Student-3 (Student_ID*, Module_Code*)

For Module-2:

- Module_Code determines Module_Name
- Module_Name determines nothing

There are no transitive dependencies.

Module-3 (Module_Code, Module_Name)

For Assignment-2:

- Assignment_ID determines Assigment_Type
- Assignment_Type determines nothing
- Module_Name determines nothing

There are no transitive dependencies.

Assignment-3 (Assignment_ID, Assignment_Type)

For Student-Assignment-2:

- Assignment_ID, Student_ID, Module_Code composite key determines Grade
- Grade determines Status
- Status determines nothing

Assignment_ID*, Student_ID*, Module_Code* → Grade
Grade → Status

The transitive dependencies are separated as:

Student-Assignment-3 (Assignment_ID*, Student_ID*, Module_Code*, Grade_ID*)
Grade-3 (Grade_ID, Grade, Status)

Hence, the results of 3NF are the following entities :

Student-3 (Student_ID, Student_Name, Street_No., Street_Name, State_Name)
Module-Student-3 (Student_ID*, Module_Code*)
Module-3 (Module_Code, Module_Name)
Assignment-3 (Assignment_ID, Assignment_Type)
Student-Assignment-3 (Assignment_ID*, Student_ID*, Module_Code*, Grade_ID*)
Grade-3 (Grade_ID, Grade, Status)

4.2.5. Final entities from figure 2

After normalizing figure 2 up to third normal form, the following entities are obtained:

Student (Student_ID, Student_Name, Street_No., Street_Name, State_Name)

Module-Student (Student_ID*, Module_Code*)

Module (Module_Code, Module_Name)

Assignment (Assignment_ID, Assignment_Type)

Student-Assignment (Assignment_ID*, Student_ID*, Module_Code*, Grade_ID*)

Grade (Grade_ID, Grade, Status)

5. Integration and Assumptions

Assumptions:

- A teacher may have multiple addresses, but a student can only have one address recorded.
- Grade is evaluated in a single alphabet optionally followed by a sign (Ex: A, A+).
- College has additional departments for managing examinations, assessments, results, as well as fees.
- Attendance of students is managed by a different system and recorded for each semester in percentage value in this database at the end of each semester right before examinations are conducted.
- Each semester has a fee associated with it which may or may not be equal for all semesters.
- Semester fees of the students are common regardless of the modules they study. This implies that students A and B studying module sets C and D have the same semester fees if they both are in semester E.
- Each semester, students are required to pay the allocated semester fees in a single installment.
- When a student pays semester fees, their fee status is updated to paid.
- To store the fees paid amount, the fee amount is redundantly added to the student_fees table so it always remains the same even if the semester fees change over time.
- Assignments and their results, attendance, and fees are managed by departments
- A student can study any three module they want.
- A teacher can teach one or multiple modules.
- Different semesters may have different fees.
- Student assignments for each module are recorded only after their grades have been evaluated.
- A student, after graduation, may become a teacher. Hence, a reference for student_id is added to the teacher table in case a student does become a teacher after graduation.

While the given example tables are very informative, they do not contain information regarding college departments. They also do not contain information regarding semester, semester fees, attendance, and fee status. Thus, the entities "Department", and "Semester", "Student_Fees" and "Student_Attendance" are added to the result of normalizing figures one and two with appropriate relations to existing entities.

Teacher (Teacher_ID, Teacher_Name, Email, Student_ID*)
Address (Address_ID, Street_No., Street_Name, State_Name)
Address-Teacher (Address_ID, Teacher_ID*)
Module (Module_Code, Module_Name, Credit_Hours)
Module-Teacher (Module_ID*, Teacher_ID*)
Student (Student_ID, Student_Name, Street_No., Street_Name, State_Name)
Module-Student (Student_ID*, Module_Code*)
Department (Department_ID, Department_Name)
Assignment (Assignment_ID, Assignment_Type, Department_ID*)
Student-Assignment (Assignment_ID*, Student_ID*, Module_Code*, Grade_ID*)
Grade (Grade_ID, Grade, Status)
Semester (Semester_ID, Semester, Semester_Fees)
Student_Fees (Semester_ID*, Student_ID*, Fee_Status, Department_ID*, Fee_Amount)
Student_Attendance (Semester_ID*, Student_ID*, Attendance_Percentage)

6. Final ERD

After normalization and integration of the given two example figures, the obtained entities form the following ERD:

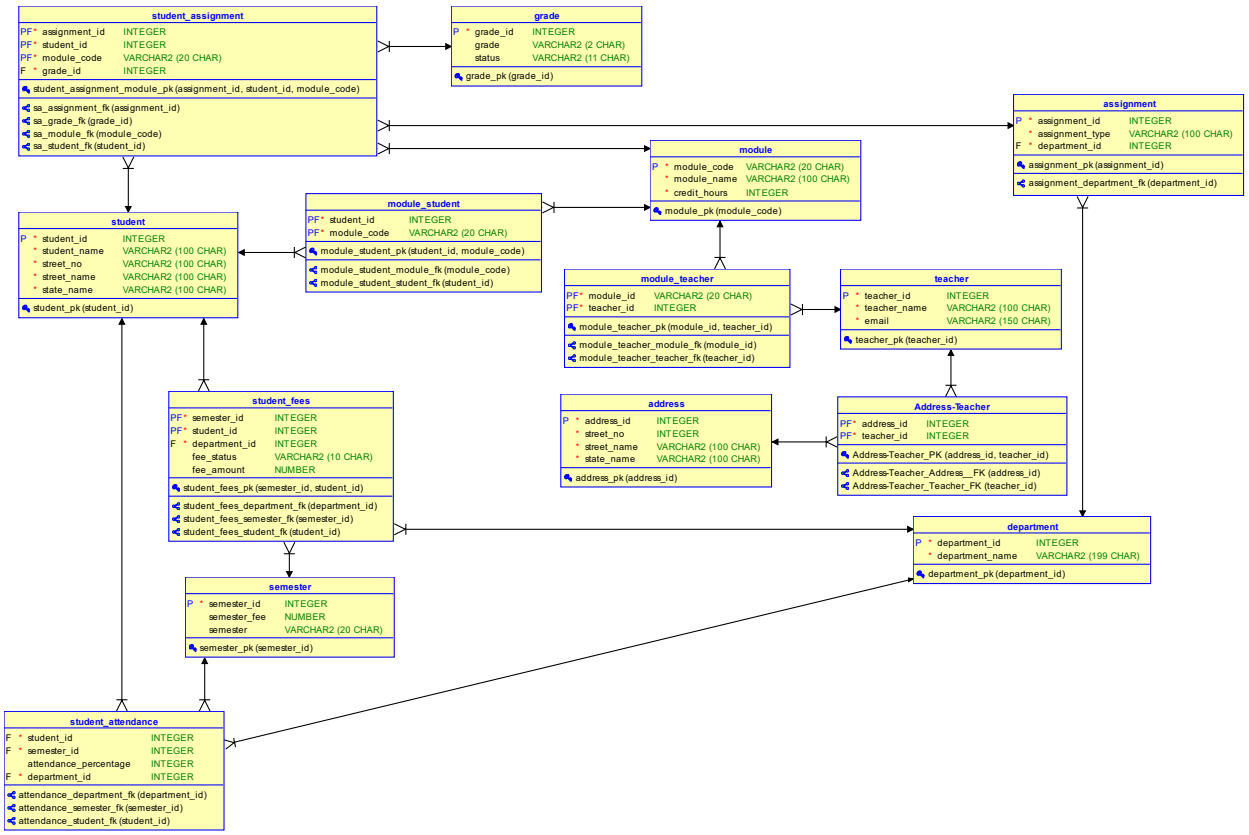


Figure 6: Final ERD After Integration

7. Data Dictionary

7.1. Table: Teacher

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Sample Data
Teacher_ID	INTEGER	38	Primary Key			Uniquely identifies each teacher	9
Teacher_Name	VARCHAR	100	NOT NULL			Stores the name of the teacher	4251
Email	VARCHAR	100	NOT NULL, UNIQUE			Stores the active email address of the teacher	Ethan_ Cunningham 2529@brety .org
Student_ID	INTEGER	38				Stores student id of the person if they were a graduated student	6

Table 2: Teacher Table Dictionary

7.2. Table: Student

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Sample Data
Student_ID	INTEGER	38	Primary Key			Uniquely identifies each student	5
Student_Name	VARCHAR	100	NOT NULL			Stores the name of the student	Ethan Shrestha
Street_No	INTEGER	38	NOT NULL			Stores street number of an address	4251
Street_Name	VARCHAR	100	NOT NULL			Stores street name of an address	Sheffield Walk
State_Name	VARCHAR	100	NOT NULL			Stores state name of an address	Hawaii

Table 3: Student Table Dictionary

7.3. Table: Address

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Sample Data
Address_ID	INTEGER	38	Primary Key			Uniquely identifies each address	3
Street_No	INTEGER	38	NOT NULL			Stores street number of an address	4251
Street_Name	VARCHAR	100	NOT NULL			Stores street name of an address	Sheffield Walk
State_Name	VARCHAR	100	NOT NULL			Stores state name of an address	Hawaii

Table 4: Address Table Dictionary

7.4. Table: Department

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Sample Data
Department_ID	INTEGER	38	Primary Key			Uniquely identifies each department	9
Department_Name	VARCHAR	100	NOT NULL, UNIQUE			Stores the name of the department	RTE

Table 5: Department Table Dictionary

7.5. Table: Assignment

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Sample Data
Assignment_ID	INTEGER	38	Primary Key			Uniquely identifies each assignment	2
Assignment_Type	VARCHAR	100	NOT NULL, UNIQUE			Stores the type of current assignment	4251
Department_ID	VARCHAR	100	NOT NULL	Department	Department_ID	Stores the department ID that manages assignments	Sheffield Walk

Table 6: Assignment Table Dictionary

7.6. Table: Grade

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Sample Data
Grade_ID	INTEGER	38	Primary Key			Uniquely identifies each grade as a number	2
Grade	VARCHAR	100	NOT NULL, UNIQUE			Stores the grade	D+
Status	VARCHAR	100	NOT NULL			Stores the pass or fail status of the grade	Pass

Table 7: Grade Table Dictionary

7.7. Table: Module

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Sample Data
Module_Code	VARCHAR	20	Primary Key			Uniquely identifies each assignment	CC12
Module_Name	VARCHAR	100	NOT NULL, UNIQUE			Stores the name of the module	Data Structure and Algorithm
Credit_Hours	INTEGER	38	NOT NULL			Stores the credit hours of the module	30

Table 8: Module Table Dictionary

7.8. Table: Module_Student

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Sample Data
Student_ID	VARCHAR	100	FOREIGN KEY, PRIMARY KEY (composite primary key)	Student	Student_ID	Stores the student id of the student associated with a module	6
Module_Code	VARCHAR	20	FOREIGN KEY, PRIMARY KEY (composite primary key)	Module	Module_Code	Stores the module code of a module associated with the student	CC12

Table 9: Module_Student Table Dictionary

7.9. Table: Module_Teacher

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Sample Data
Teacher_ID	VARCHAR	100	FOREIGN KEY, PRIMARY KEY (composite primary key)	Teacher	Teacher_ID	Stores the teacher id of the teacher associated with a module	9
Module_ID	VARCHAR	20	FOREIGN KEY, PRIMARY KEY (composite primary key)	Module	Module_Code	Stores the module code of a module associated with the teacher	CC12

Table 10: Module_Teacher Table Dictionary

7.10. Table: Semester

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Sample Data
Semester_ID	INTEGER	38	Primary Key			Uniquely identifies each semester	6
Semester	INTEGER	38	NOT NULL, UNIQUE			Stores the semester number	9
Semester_Fees	INTEGER	38	NOT NULL			Stores the fee for the semester	650

Table 11: Semester Table Dictionary

7.11. Table: Student_Assignment

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Sample Data
Assignment_ID	INTEGER	38	FOREIGN KEY, PRIMARY KEY (composite primary key)	Assignment	Assignment_ID	Stores the assignment id of the assignment	6
Student_ID	INTEGER	38	FOREIGN KEY, PRIMARY KEY (composite primary key)	Student	Student_ID	Stores the student id of the student associated with the assignment	9
Module_Code	VARCHAR	20	FOREIGN KEY, PRIMARY KEY (composite primary key)	Module	Module_Code	Stores the module id of the module of which thew assignment is	CC12
Grade_ID	INTEGER	38	FOREIGN KEY	Grade	Grade_ID	Stores the grade id of the grade obtained by the student	3

Table 12: Student_Assignment Table Dictionary

7.12. Table: Student_Attendance

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Sample Data
Student_ID	INTEGER	38	FOREIGN KEY, PRIMARY KEY (composite primary key)	Student	Student_ID	Stores the student id of the student whose attendance is being recorded	6
Semester_ID	INTEGER	38	FOREIGN KEY, PRIMARY KEY (composite primary key)	Semester	Semester_ID	Stores the semester id of the semester in which the attendance was recorded	3
Department_ID	INTEGER	38	FOREIGN KEY, NOT NULL	Department	Department_ID	Stores the department id of the department responsible for managing attendance	1

Table 13: Student_Attendance Table Dictionary

7.13. Table: Student_Fees

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Sample Data
Student_ID	INTEGER	38	FOREIGN KEY, PRIMARY KEY (composite primary key)	Student	Student_ID	Stores the student id of the student whose fee detail is being recorded	6
Semester_ID	INTEGER	38	FOREIGN KEY, PRIMARY KEY (composite primary key)	Semester	Semester_ID	Stores the semester id of the semester for which the fee detail was recorded	3
Department_ID	INTEGER	38	FOREIGN KEY, NOT NULL	Department	Department_ID	Stores the department id of the department responsible for managing fees	2
Fee_Amount	NUMBER	(30,8)				Stores the amount of fee paid by the student	700

Fee_Status	VARCHAR	10				Stores the status of fees determining if the fee has been paid or not	Due
Payment_Date	DATE					Stores the date of payment of	TO_DATE('09, 01, 2017', 'MM, DD, YYYY')

Table 14: Student_Attendance Table Dictionary

7.14. Table: Address_Teacher

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Sample Data
Teacher_ID	INTEGER	38	FOREIGN KEY, PRIMARY KEY (composite primary key)	Teacher	Teacher_ID	Stores the teacher id of the teacher associated with an address	6
Address_ID	INTEGER	38	FOREIGN KEY, PRIMARY KEY (composite primary key)	Address	Address_ID	Stores the address id of the address that is associated with the teacher	9

Table 15: Address_Teacher Table Dictionary

8. DDL Script

8.1. Script

```
-- Generated by Oracle SQL Developer Data Modeler 21.4.1.349.1605
--   at:        2022-03-12 18:05:35 NPT
--   site:      Oracle Database 11g
--   type:      Oracle Database 11g



-- predefined type, no DDL - MDSYS.SDO_GEOMETRY

-- predefined type, no DDL - XMLTYPE

CREATE SEQUENCE address_address_id_seq START WITH 1 NOCACHE ORDER;

CREATE SEQUENCE assignment_assignment_id_seq START WITH 1 NOCACHE ORDER;

CREATE SEQUENCE department_department_id_seq START WITH 1 NOCACHE ORDER;

CREATE SEQUENCE grade_grade_id_seq START WITH 1 NOCACHE ORDER;

CREATE SEQUENCE semester_semester_id_seq START WITH 1 NOCACHE ORDER;

CREATE SEQUENCE student_student_id_seq START WITH 1 NOCACHE ORDER;

CREATE SEQUENCE teacher_teacher_id_seq START WITH 1 NOCACHE ORDER;

CREATE TABLE address (
    address_id  INTEGER NOT NULL,
    street_no   INTEGER NOT NULL,
    street_name VARCHAR2(100 CHAR) NOT NULL,
    state_name  VARCHAR2(100 CHAR) NOT NULL
)
;

ALTER TABLE address ADD CONSTRAINT address_pk PRIMARY KEY ( address_id );

CREATE TABLE address_teacher (
    address_id INTEGER NOT NULL,
    teacher_id INTEGER NOT NULL
)
;
```

```
ALTER TABLE address_teacher ADD CONSTRAINT "Address_Teacher_PK" PRIMARY KEY (
address_id,
                                                                        t
eacher_id );

CREATE TABLE assignment (
    assignment_id   INTEGER NOT NULL,
    assignment_type VARCHAR2(100 CHAR) NOT NULL,
    department_id   INTEGER DEFAULT 1 NOT NULL
)
;

ALTER TABLE assignment ADD CONSTRAINT assignment_pk PRIMARY KEY ( assignment_id
);

ALTER TABLE assignment ADD CONSTRAINT assignment_assignment_type_un UNIQUE (
assignment_type );

CREATE TABLE department (
    department_id   INTEGER NOT NULL,
    department_name VARCHAR2(199 CHAR) NOT NULL
)
;

ALTER TABLE department ADD CONSTRAINT department_pk PRIMARY KEY ( department_id
);
ALTER TABLE department ADD CONSTRAINT department_department_name_un UNIQUE (
department_name );

CREATE TABLE grade (
    grade_id INTEGER NOT NULL,
    grade    VARCHAR2(2 CHAR),
    status   VARCHAR2(11 CHAR)
)
;

ALTER TABLE grade ADD CONSTRAINT grade_pk PRIMARY KEY ( grade_id );

ALTER TABLE grade ADD CONSTRAINT grade__un UNIQUE ( grade );

CREATE TABLE module (
    module_code  VARCHAR2(20 CHAR) NOT NULL,
    module_name  VARCHAR2(100 CHAR) NOT NULL,
    credit_hours INTEGER NOT NULL
)
```

```
;

ALTER TABLE module ADD CONSTRAINT module_pk PRIMARY KEY ( module_code );

ALTER TABLE module ADD CONSTRAINT module_module_name_un UNIQUE ( module_name );

CREATE TABLE module_student (
    student_id  INTEGER NOT NULL,
    module_code VARCHAR2(20 CHAR) NOT NULL
)
;

ALTER TABLE module_student ADD CONSTRAINT module_student_pk PRIMARY KEY (
student_id,
                                                                    module_
code );

CREATE TABLE module_teacher (
    module_id  VARCHAR2(20 CHAR) NOT NULL,
    teacher_id INTEGER NOT NULL
)
;

ALTER TABLE module_teacher ADD CONSTRAINT module_teacher_pk PRIMARY KEY (
module_id,
                                                                    teacher
_id );

CREATE TABLE semester (
    semester_id  INTEGER NOT NULL,
    semester_fee NUMBER,
    semester     VARCHAR2(20 CHAR)
)
;

ALTER TABLE semester ADD CONSTRAINT semester_pk PRIMARY KEY ( semester_id );
ALTER TABLE semester ADD CONSTRAINT semester_semester_un UNIQUE ( semester );

CREATE TABLE student (
    student_id   INTEGER NOT NULL,
    student_name VARCHAR2(100 CHAR) NOT NULL,
    street_no    VARCHAR2(100 CHAR) NOT NULL,
    street_name  VARCHAR2(100 CHAR) NOT NULL,
    state_name   VARCHAR2(100 CHAR) NOT NULL
)
```

```
;

ALTER TABLE student ADD CONSTRAINT student_pk PRIMARY KEY ( student_id );

CREATE TABLE student_assignment (
    assignment_id INTEGER NOT NULL,
    student_id    INTEGER NOT NULL,
    module_code   VARCHAR2(20 CHAR) NOT NULL,
    grade_id      INTEGER NOT NULL
)
;

ALTER TABLE student_assignment
    ADD CONSTRAINT student_assignment_module_pk PRIMARY KEY ( assignment_id,
                                                              student_id,
                                                              module_code );

CREATE TABLE student_attendance (
    student_id            INTEGER NOT NULL,
    semester_id           INTEGER NOT NULL,
    attendance_percentage INTEGER,
    department_id         INTEGER DEFAULT 1 NOT NULL
)
;

ALTER TABLE student_attendance ADD CONSTRAINT student_attendance_pk PRIMARY KEY
(student_id,

  semester_id);

CREATE TABLE student_fees (
    semester_id   INTEGER NOT NULL,
    student_id    INTEGER NOT NULL,
    department_id INTEGER DEFAULT 2 NOT NULL,
    fee_status    VARCHAR2(10 CHAR),
    fee_amount    NUMBER(30,8),
    payment_date  DATE
)
;

ALTER TABLE student_fees ADD CONSTRAINT student_fees_pk PRIMARY KEY (
semester_id,
                                                                    student_id
);
```

```
CREATE TABLE teacher (
    teacher_id   INTEGER NOT NULL,
    teacher_name VARCHAR2(100 CHAR) NOT NULL,
    email        VARCHAR2(150 CHAR) NOT NULL,
    student_id   INTEGER DEFAULT NULL
)
;

ALTER TABLE teacher ADD CONSTRAINT teacher_pk PRIMARY KEY ( teacher_id );

ALTER TABLE teacher ADD CONSTRAINT teacher_email_un UNIQUE ( email );

ALTER TABLE address_teacher
    ADD CONSTRAINT "address_teacher_Address__FK" FOREIGN KEY ( address_id )
        REFERENCES address ( address_id )
                ON DELETE CASCADE
    NOT DEFERRABLE;

ALTER TABLE address_teacher
    ADD CONSTRAINT "address_teacher_Teacher_FK" FOREIGN KEY ( teacher_id )
        REFERENCES teacher ( teacher_id )
                ON DELETE CASCADE
    NOT DEFERRABLE;

ALTER TABLE assignment
    ADD CONSTRAINT assignment_department_fk FOREIGN KEY ( department_id )
        REFERENCES department ( department_id )
                ON DELETE CASCADE
    NOT DEFERRABLE;

ALTER TABLE student_attendance
    ADD CONSTRAINT attendance_department_fk FOREIGN KEY ( department_id )
        REFERENCES department ( department_id )
                ON DELETE CASCADE
    NOT DEFERRABLE;

ALTER TABLE student_attendance
    ADD CONSTRAINT attendance_semester_fk FOREIGN KEY ( semester_id )
        REFERENCES semester ( semester_id )
                ON DELETE CASCADE
    NOT DEFERRABLE;

ALTER TABLE student_attendance
    ADD CONSTRAINT attendance_student_fk FOREIGN KEY ( student_id )
        REFERENCES student ( student_id )
```

```
                ON DELETE CASCADE
    NOT DEFERRABLE;

ALTER TABLE module_student
    ADD CONSTRAINT module_student_module_fk FOREIGN KEY ( module_code )
        REFERENCES module ( module_code )
                ON DELETE CASCADE
    NOT DEFERRABLE;

ALTER TABLE module_student
    ADD CONSTRAINT module_student_student_fk FOREIGN KEY ( student_id )
        REFERENCES student ( student_id )
                ON DELETE CASCADE
    NOT DEFERRABLE;

ALTER TABLE module_teacher
    ADD CONSTRAINT module_teacher_module_fk FOREIGN KEY ( module_id )
        REFERENCES module ( module_code )
                ON DELETE CASCADE
    NOT DEFERRABLE;

ALTER TABLE module_teacher
    ADD CONSTRAINT module_teacher_teacher_fk FOREIGN KEY ( teacher_id )
        REFERENCES teacher ( teacher_id )
                ON DELETE CASCADE
    NOT DEFERRABLE;

ALTER TABLE student_assignment
    ADD CONSTRAINT sa_assignment_fk FOREIGN KEY ( assignment_id )
        REFERENCES assignment ( assignment_id )
                ON DELETE CASCADE
    NOT DEFERRABLE;

ALTER TABLE student_assignment
    ADD CONSTRAINT sa_grade_fk FOREIGN KEY ( grade_id )
        REFERENCES grade ( grade_id )
                ON DELETE CASCADE
    NOT DEFERRABLE;

ALTER TABLE student_assignment
    ADD CONSTRAINT sa_module_fk FOREIGN KEY ( module_code )
        REFERENCES module ( module_code )
                ON DELETE CASCADE
    NOT DEFERRABLE;
```

```
ALTER TABLE student_assignment
    ADD CONSTRAINT sa_student_fk FOREIGN KEY ( student_id )
        REFERENCES student ( student_id )
                ON DELETE CASCADE
    NOT DEFERRABLE;

ALTER TABLE student_fees
    ADD CONSTRAINT student_fees_department_fk FOREIGN KEY ( department_id )
        REFERENCES department ( department_id )
                ON DELETE CASCADE
    NOT DEFERRABLE;

ALTER TABLE student_fees
    ADD CONSTRAINT student_fees_semester_fk FOREIGN KEY ( semester_id )
        REFERENCES semester ( semester_id )
                ON DELETE CASCADE
    NOT DEFERRABLE;

ALTER TABLE student_fees
    ADD CONSTRAINT student_fees_student_fk FOREIGN KEY ( student_id )
        REFERENCES student ( student_id )
                ON DELETE CASCADE
    NOT DEFERRABLE;

ALTER TABLE teacher
    ADD CONSTRAINT teacher_student_fk FOREIGN KEY ( student_id )
        REFERENCES student ( student_id )
                ON DELETE CASCADE
    NOT DEFERRABLE;

CREATE OR REPLACE TRIGGER address_address_id_trg BEFORE
    INSERT ON address
    FOR EACH ROW
    WHEN ( new.address_id IS NULL )
BEGIN
    :new.address_id := address_address_id_seq.nextval;
END;
/

CREATE OR REPLACE TRIGGER assignment_assignment_id_trg BEFORE
    INSERT ON assignment
    FOR EACH ROW
    WHEN ( new.assignment_id IS NULL )
BEGIN
    :new.assignment_id := assignment_assignment_id_seq.nextval;
```

```
END;
/

CREATE OR REPLACE TRIGGER department_department_id_trg BEFORE
    INSERT ON department
    FOR EACH ROW
    WHEN ( new.department_id IS NULL )
BEGIN
    :new.department_id := department_department_id_seq.nextval;
END;
/

CREATE OR REPLACE TRIGGER grade_grade_id_trg BEFORE
    INSERT ON grade
    FOR EACH ROW
    WHEN ( new.grade_id IS NULL )
BEGIN
    :new.grade_id := grade_grade_id_seq.nextval;
END;
/

CREATE OR REPLACE TRIGGER semester_semester_id_trg BEFORE
    INSERT ON semester
    FOR EACH ROW
    WHEN ( new.semester_id IS NULL )
BEGIN
    :new.semester_id := semester_semester_id_seq.nextval;
END;
/

CREATE OR REPLACE TRIGGER student_student_id_trg BEFORE
    INSERT ON student
    FOR EACH ROW
    WHEN ( new.student_id IS NULL )
BEGIN
    :new.student_id := student_student_id_seq.nextval;
END;
/

CREATE OR REPLACE TRIGGER teacher_teacher_id_trg BEFORE
    INSERT ON teacher
    FOR EACH ROW
    WHEN ( new.teacher_id IS NULL )
BEGIN
    :new.teacher_id := teacher_teacher_id_seq.nextval;
```

```
END;
/



-- Oracle SQL Developer Data Modeler Summary Report:
--
-- CREATE TABLE                            14
-- CREATE INDEX                             0
-- ALTER TABLE                             30
-- CREATE VIEW                              0
-- ALTER VIEW                               0
-- CREATE PACKAGE                           0
-- CREATE PACKAGE BODY                      0
-- CREATE PROCEDURE                         0
-- CREATE FUNCTION                          0
-- CREATE TRIGGER                           7
-- ALTER TRIGGER                            0
-- CREATE COLLECTION TYPE                   0
-- CREATE STRUCTURED TYPE                   0
-- CREATE STRUCTURED TYPE BODY              0
-- CREATE CLUSTER                           0
-- CREATE CONTEXT                           0
-- CREATE DATABASE                          0
-- CREATE DIMENSION                         0
-- CREATE DIRECTORY                         0
-- CREATE DISK GROUP                        0
-- CREATE ROLE                              0
-- CREATE ROLLBACK SEGMENT                  0
-- CREATE SEQUENCE                          7
-- CREATE MATERIALIZED VIEW                 0
-- CREATE MATERIALIZED VIEW LOG             0
-- CREATE SYNONYM                           0
-- CREATE TABLESPACE                        0
-- CREATE USER                              0
--
-- DROP TABLESPACE                          0
-- DROP DATABASE                            0
--
-- REDACTION POLICY                         0
--
-- ORDS DROP SCHEMA                         0
-- ORDS ENABLE SCHEMA                       0
-- ORDS ENABLE OBJECT                       0
--
```

```
-- ERRORS                                    0
-- WARNINGS                                  0
```

8.2. Output

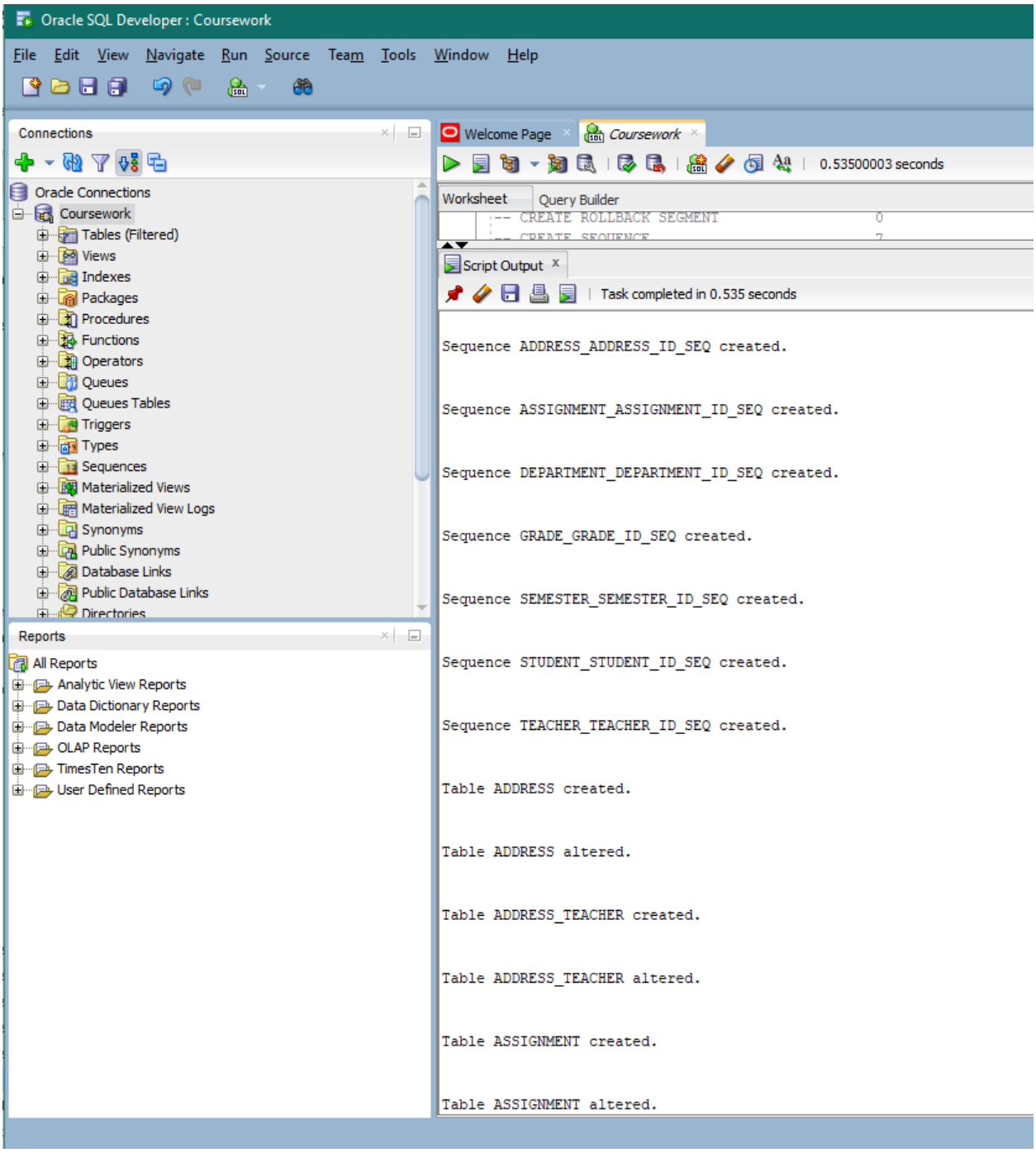


Figure 7:DDL Script Output 1

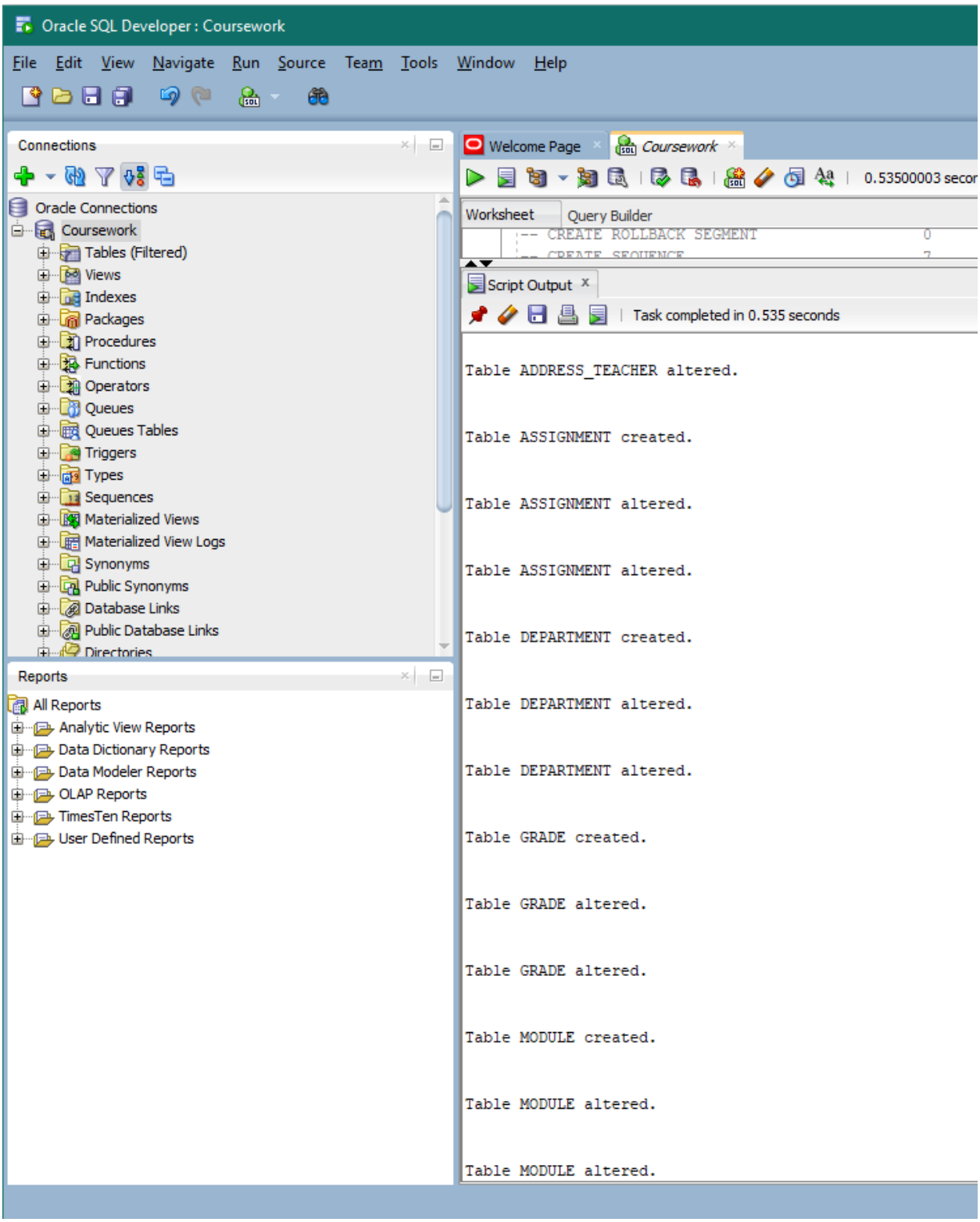


Figure 8:DDL Script Output 2

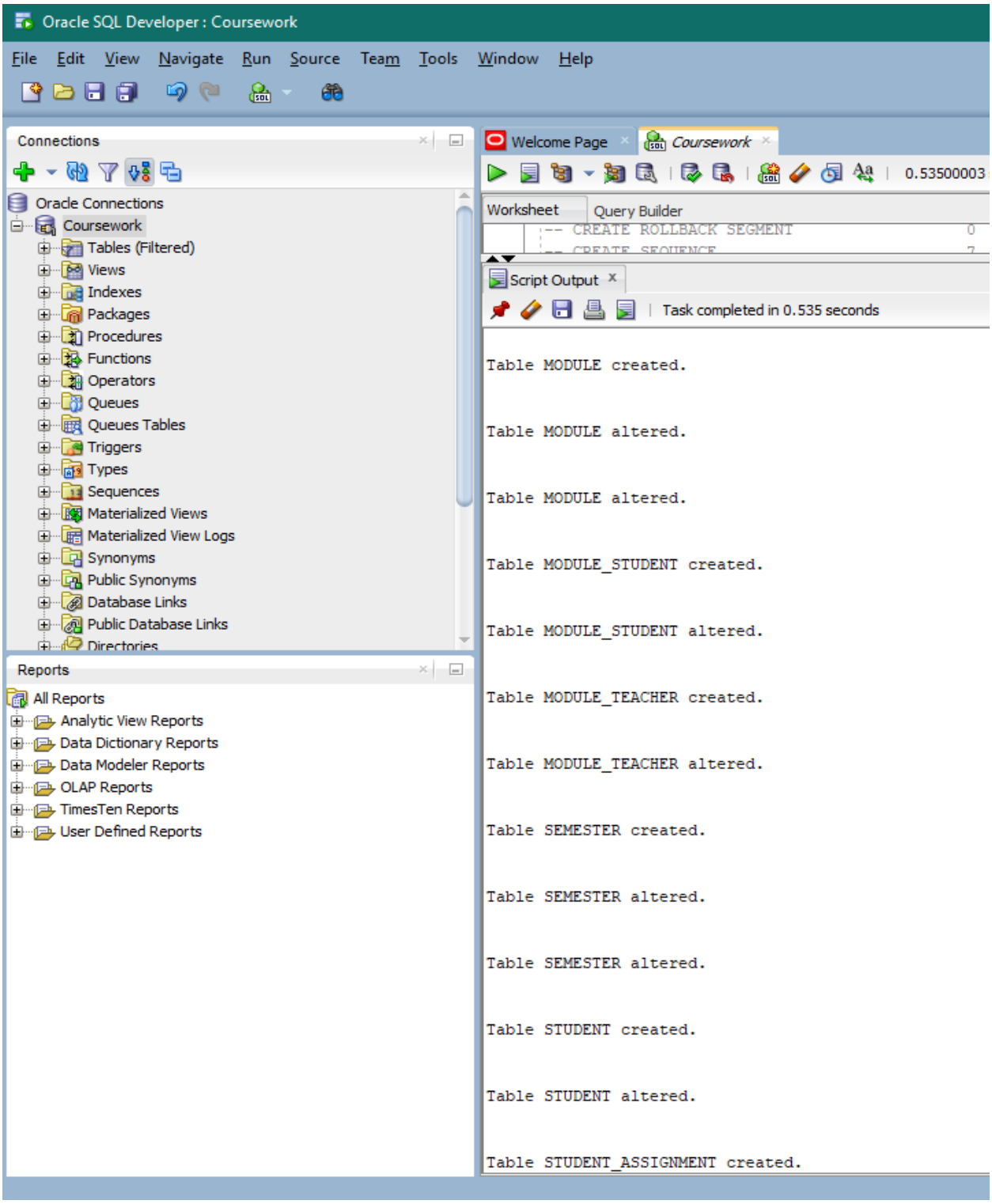


Figure 9:DDL Script Output 3

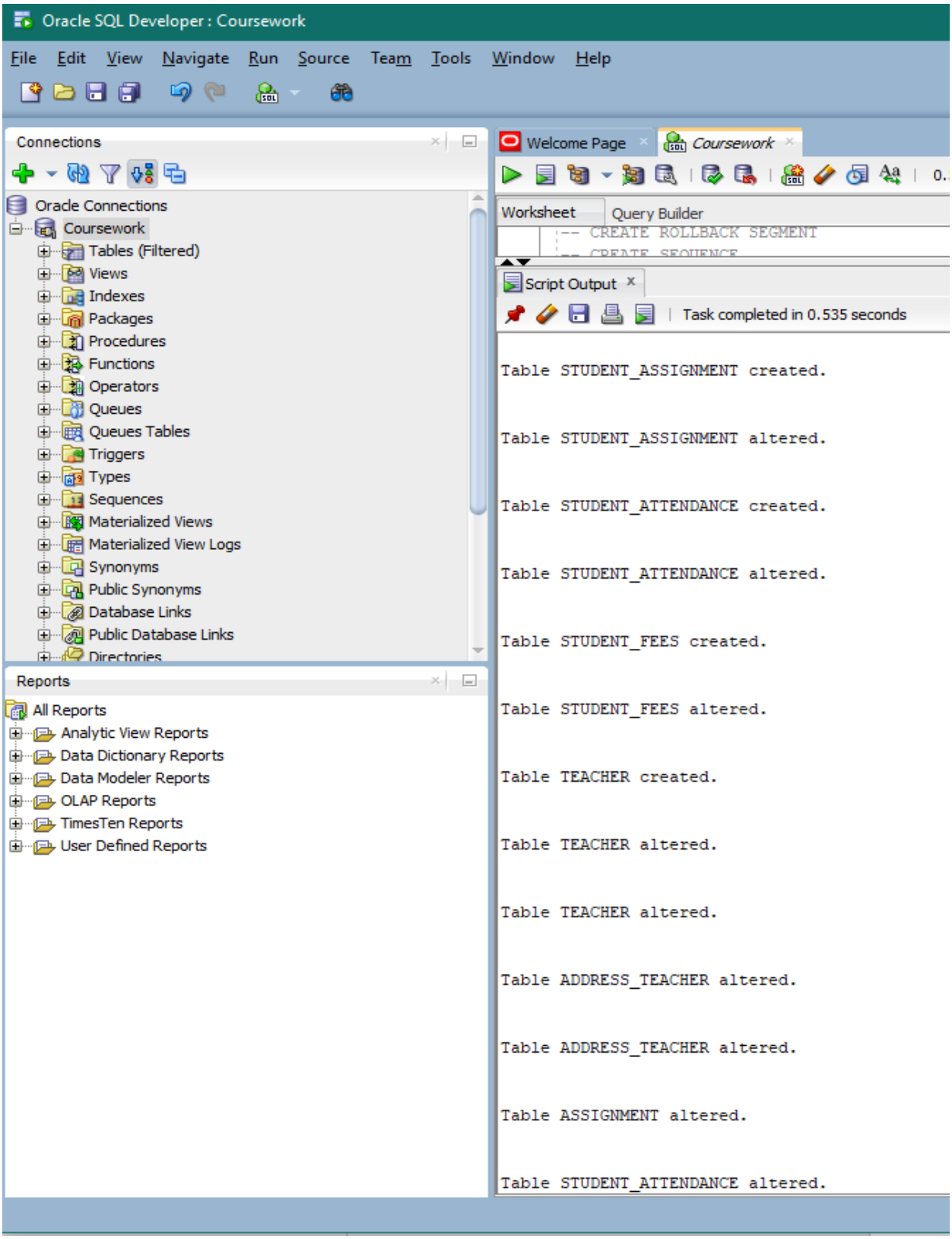


Figure 10:DDL Script Output 4

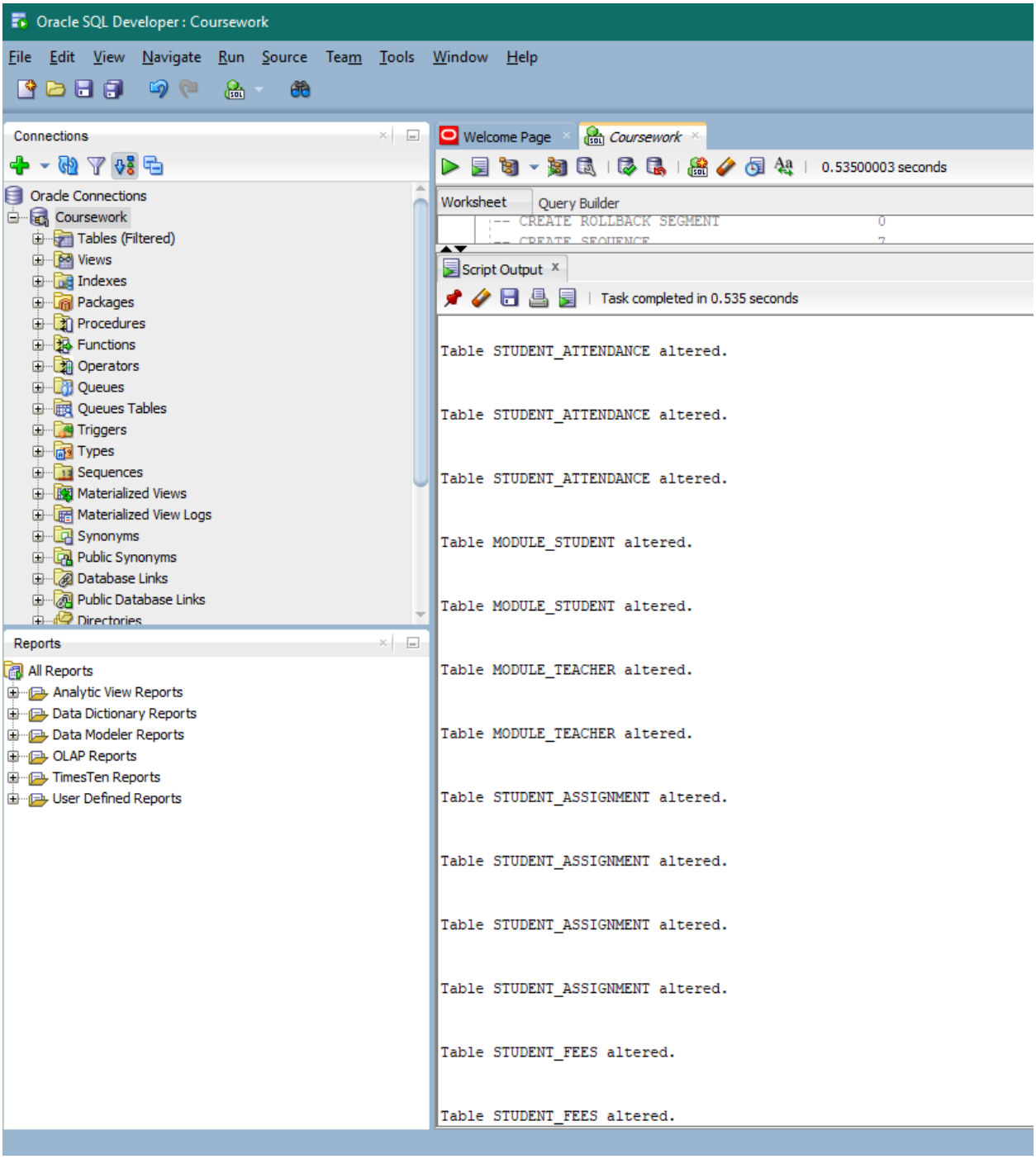


Figure 11:DDL Script Output 5

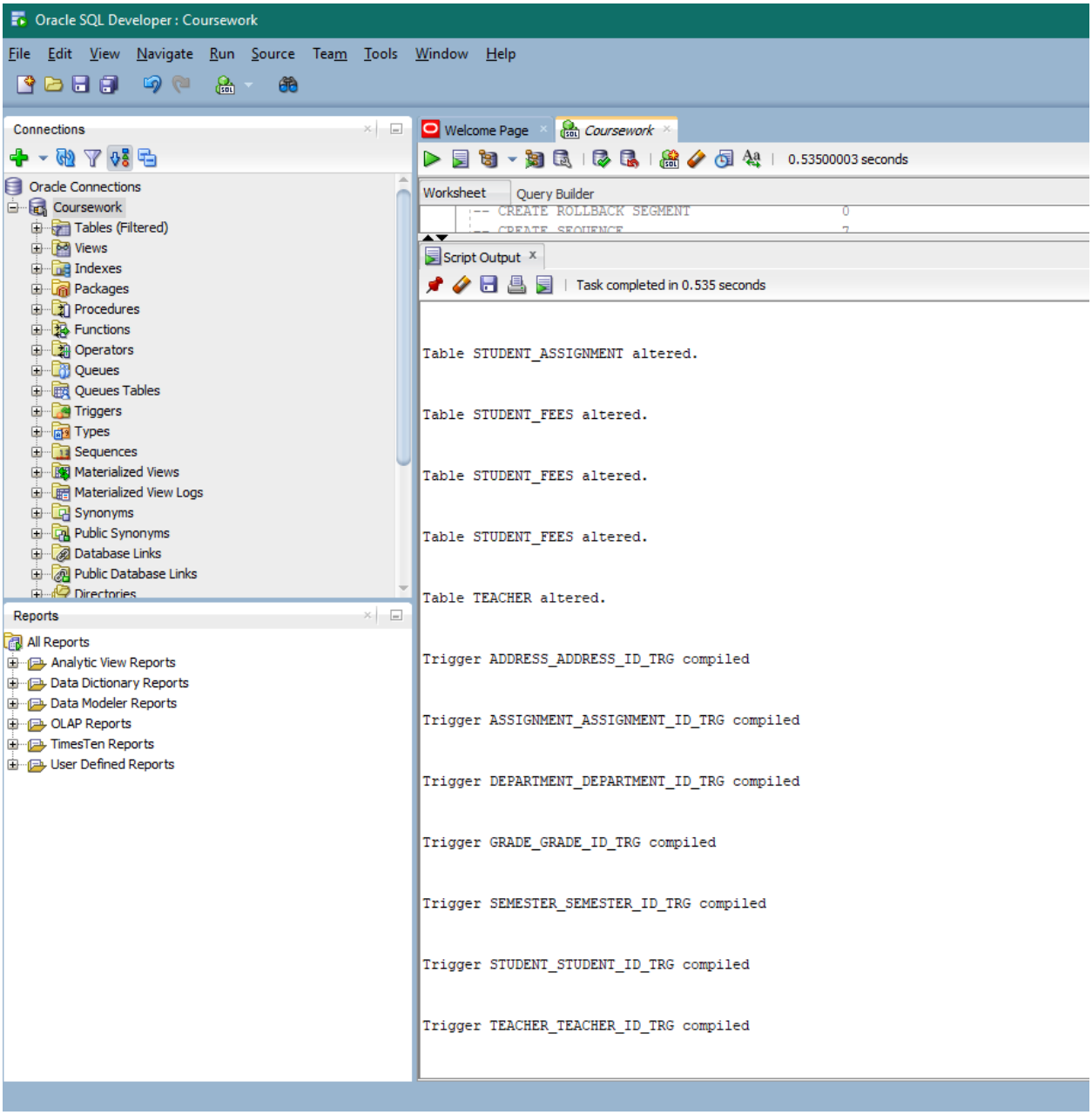


Figure 12:DDL Script Output 6

9. INSERT Statements

9.1. Teacher

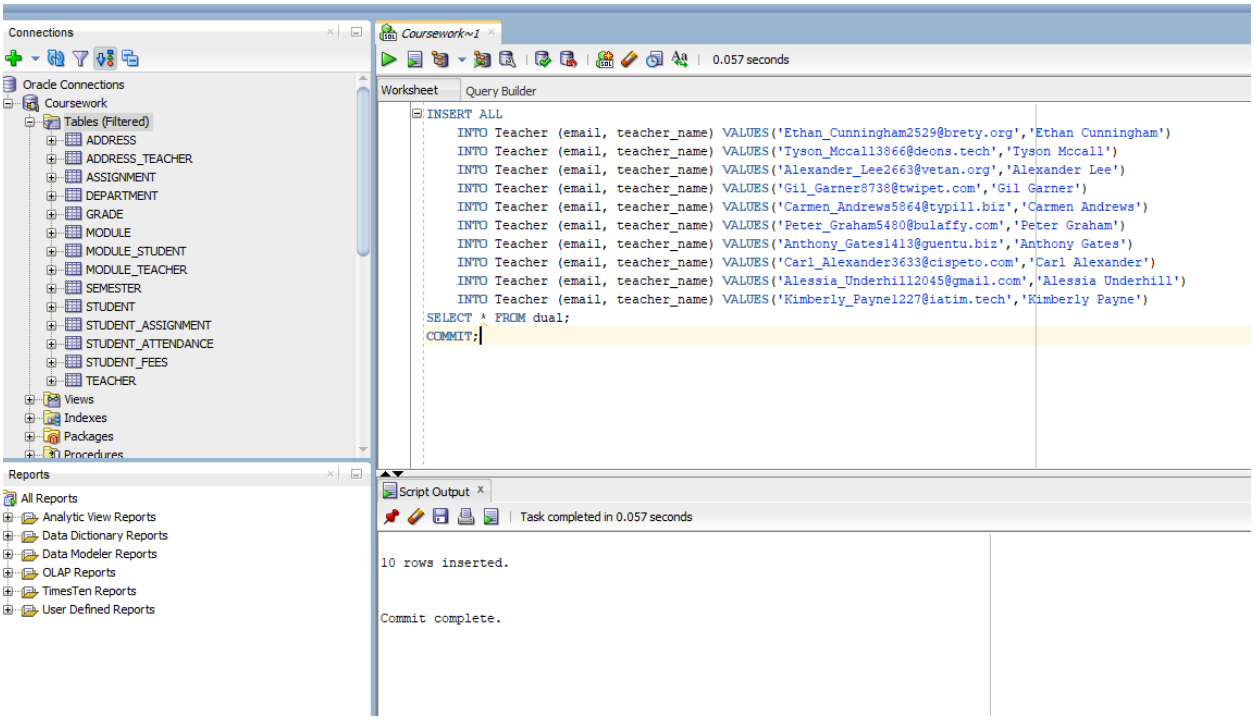


Figure 13: INSERT Teacher

9.2. Student

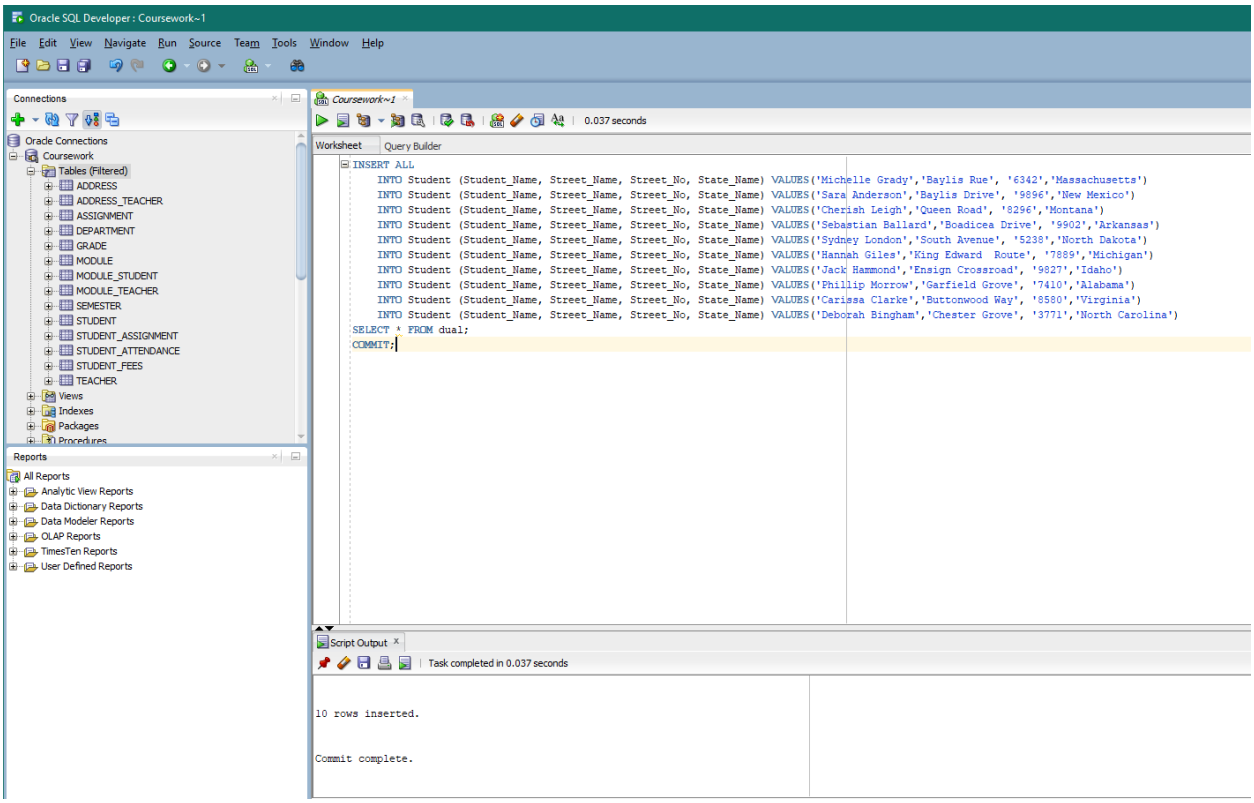


Figure 14: INSERT Student

9.3. Semester

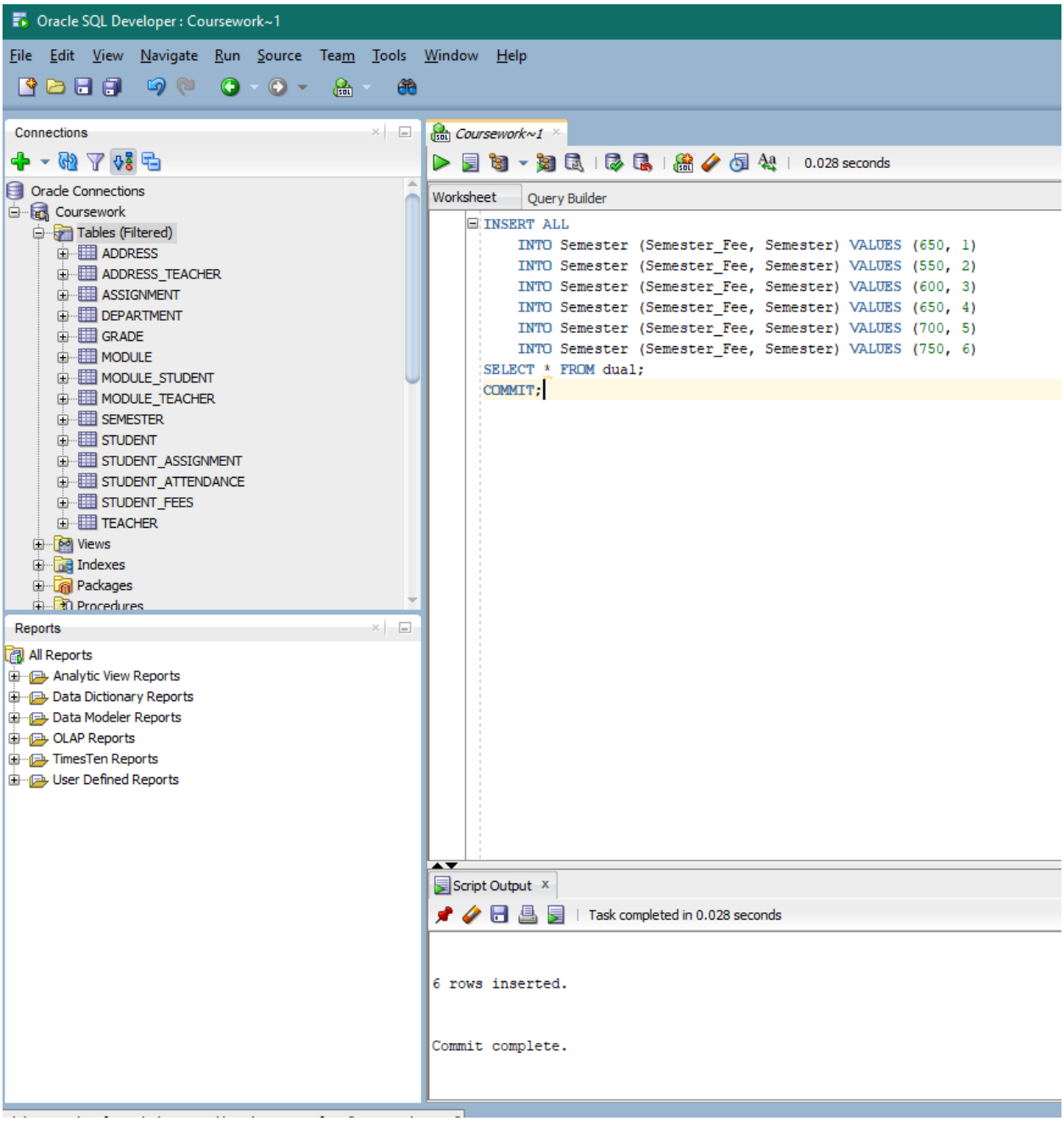


Figure 15: INSERT Semester

9.4. Module

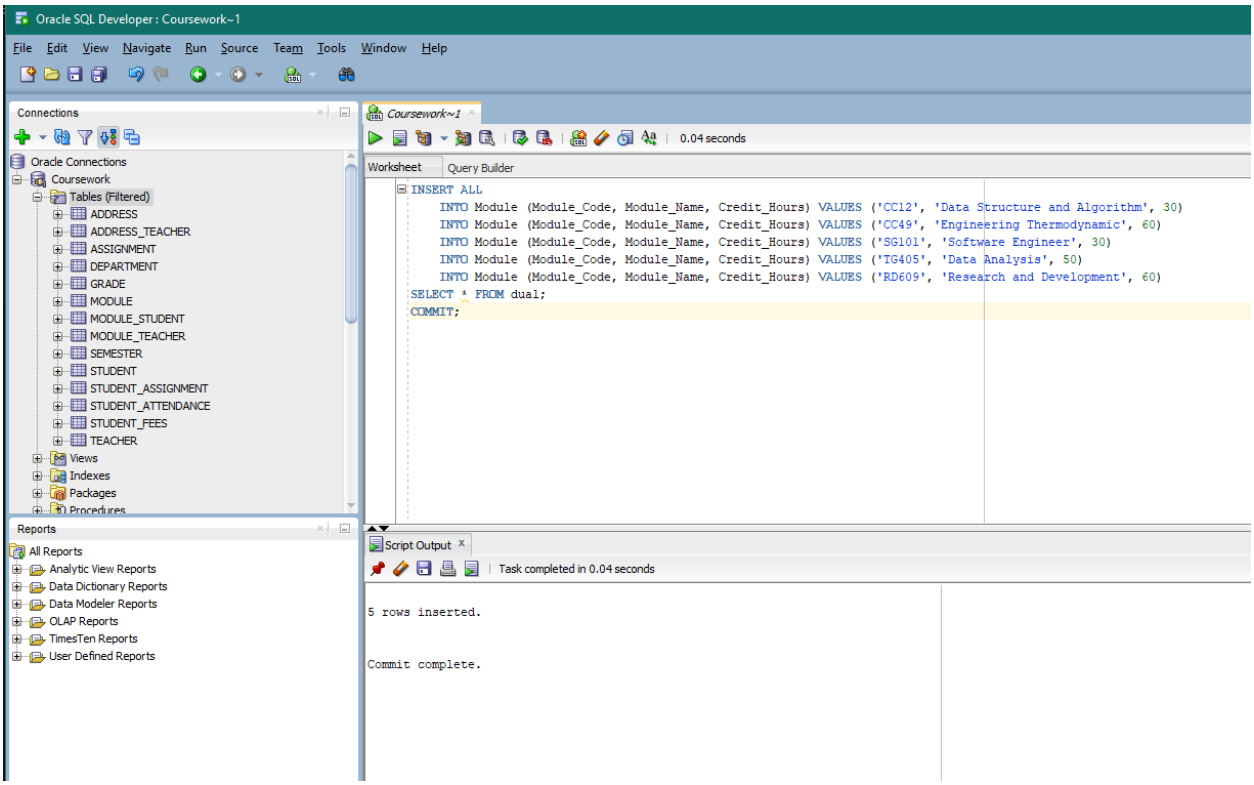


Figure 16: INSERT Module

9.5. Grade

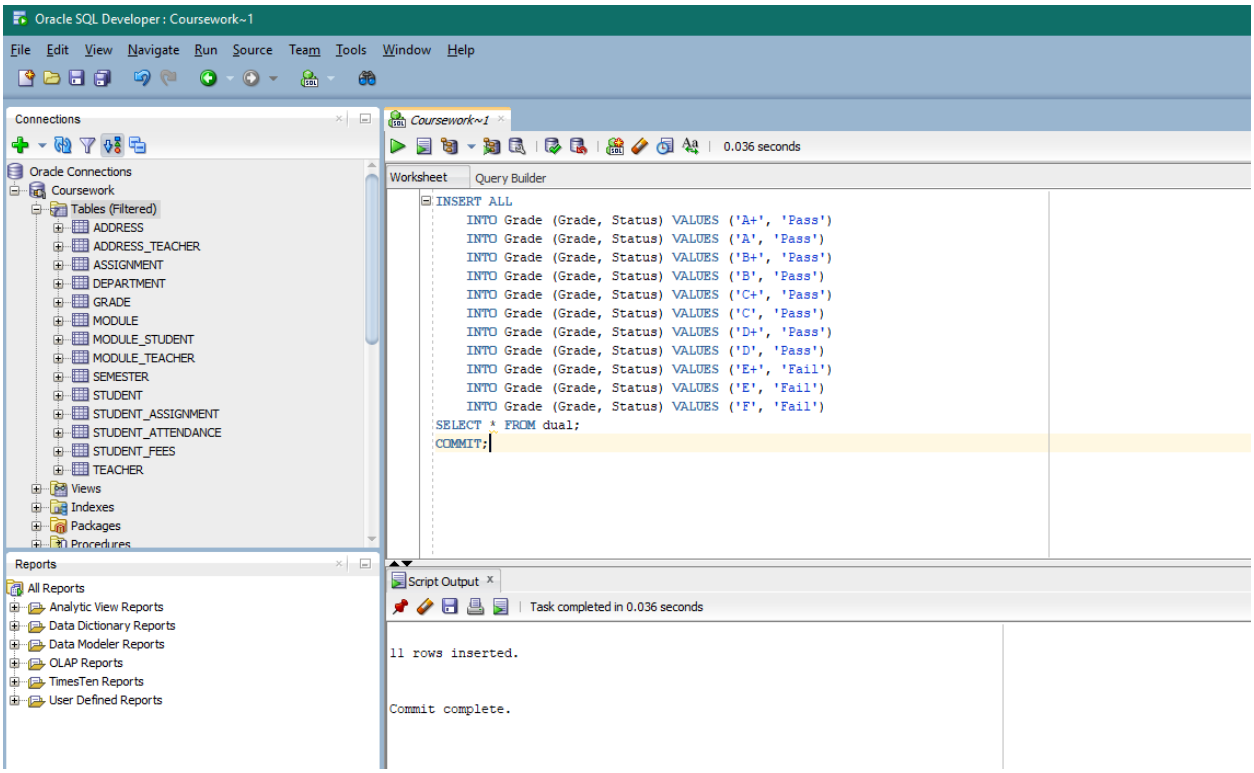


Figure 17: INSERT Grade

9.6. Department

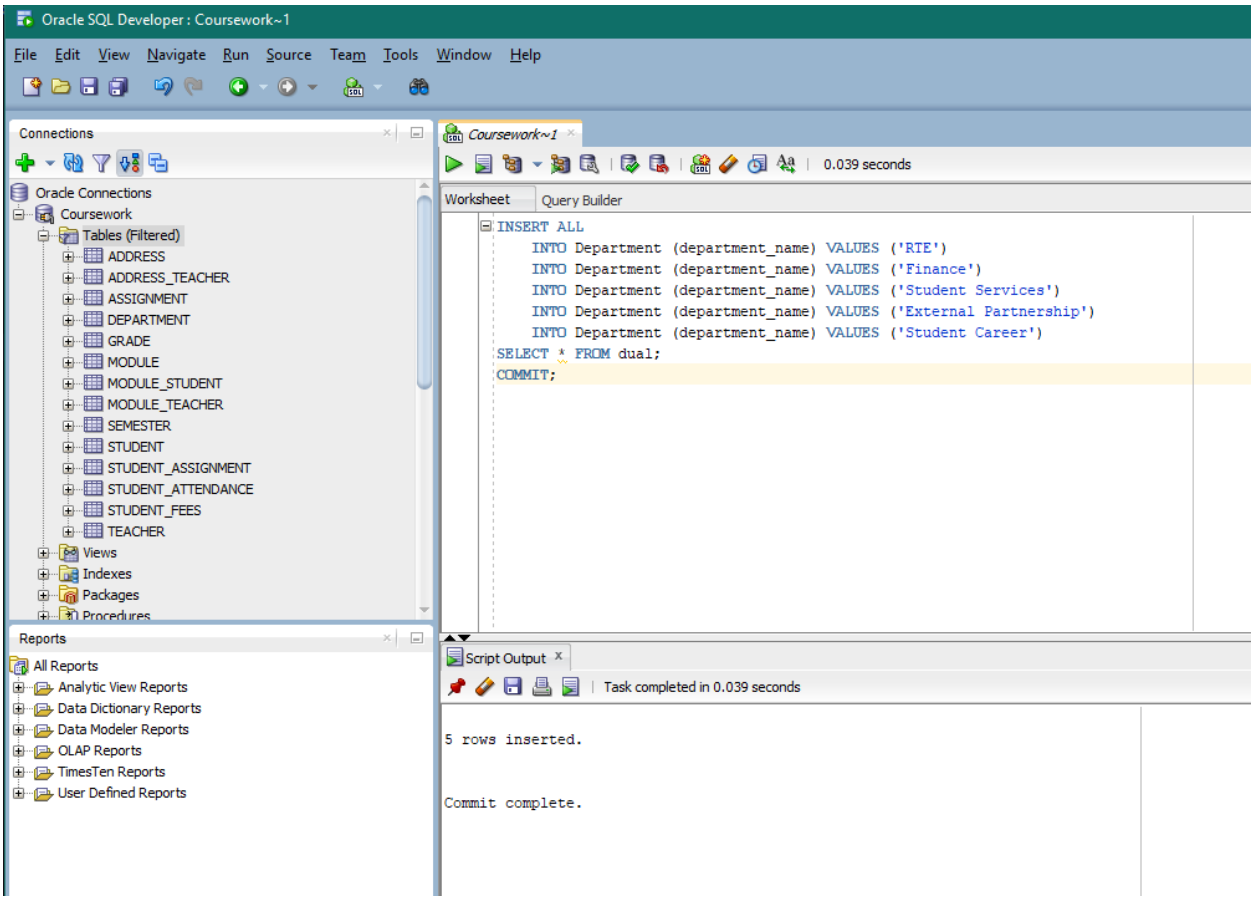


Figure 18: INSERT Department

9.7. Address

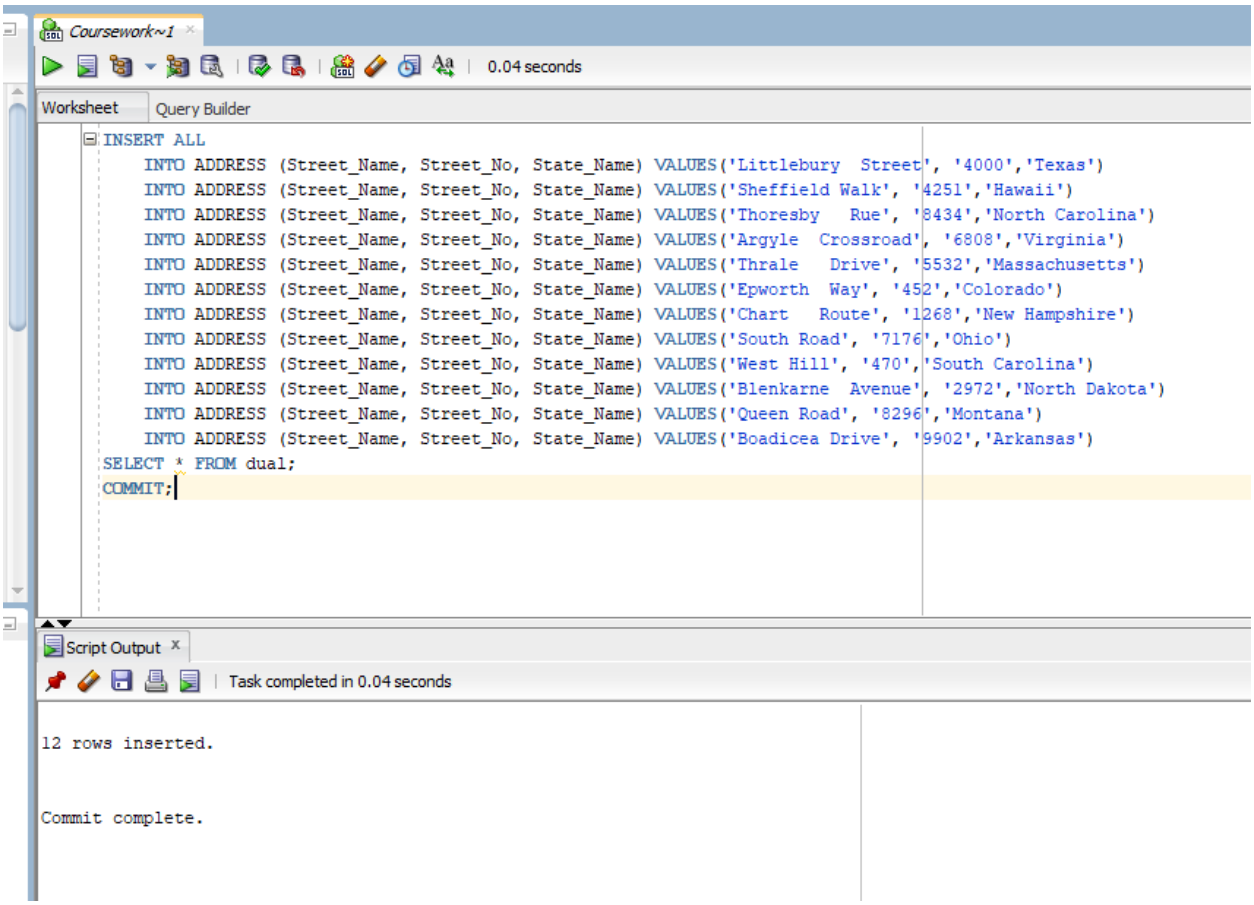


Figure 19: INSERT Address

9.8. Assignment

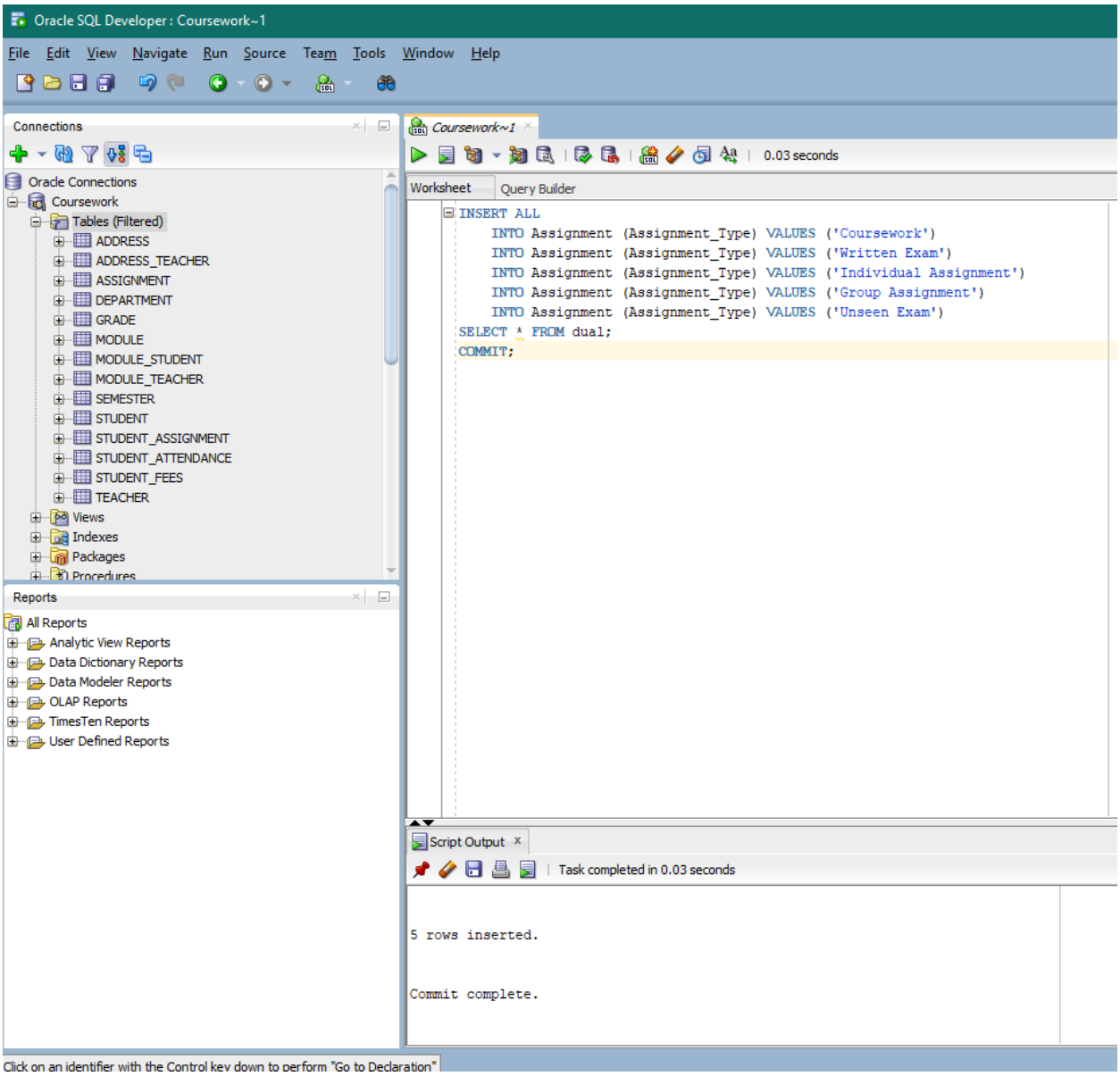


Figure 20: INSERT Assignment

9.9. Student_Fees

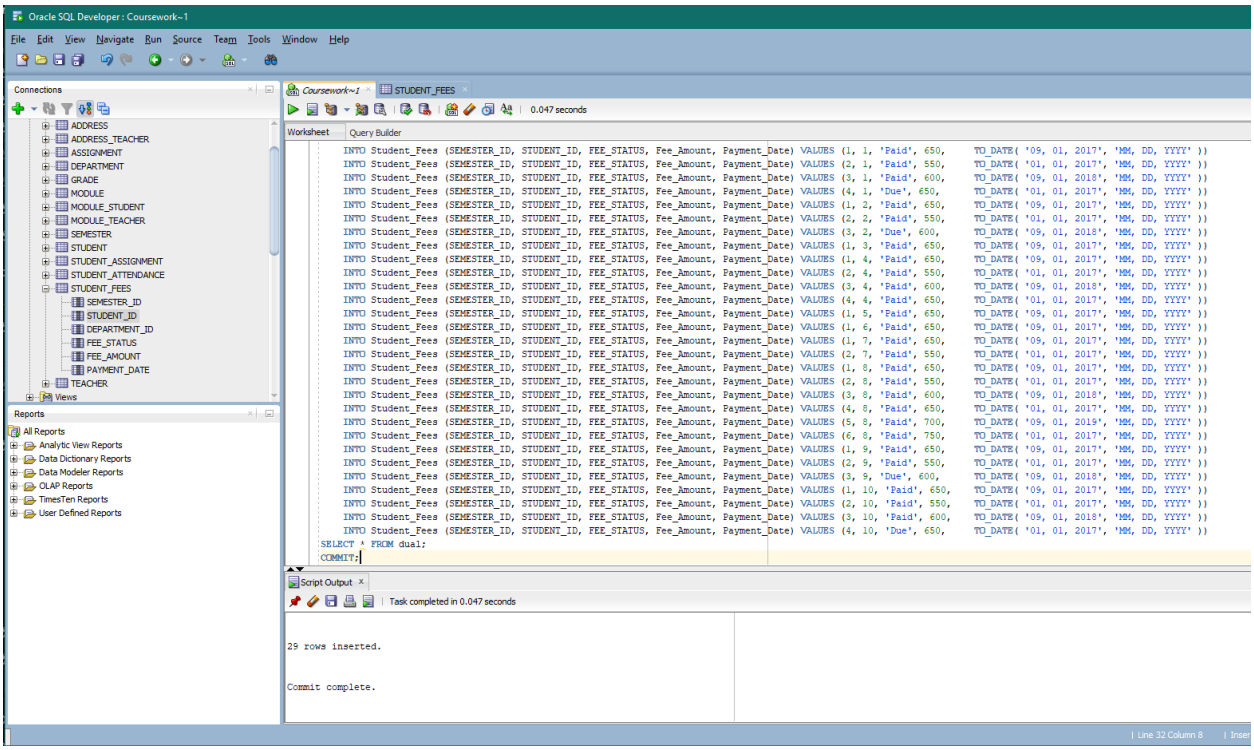


Figure 21: INSERT Student_Fees

9.10. Student_Attendance

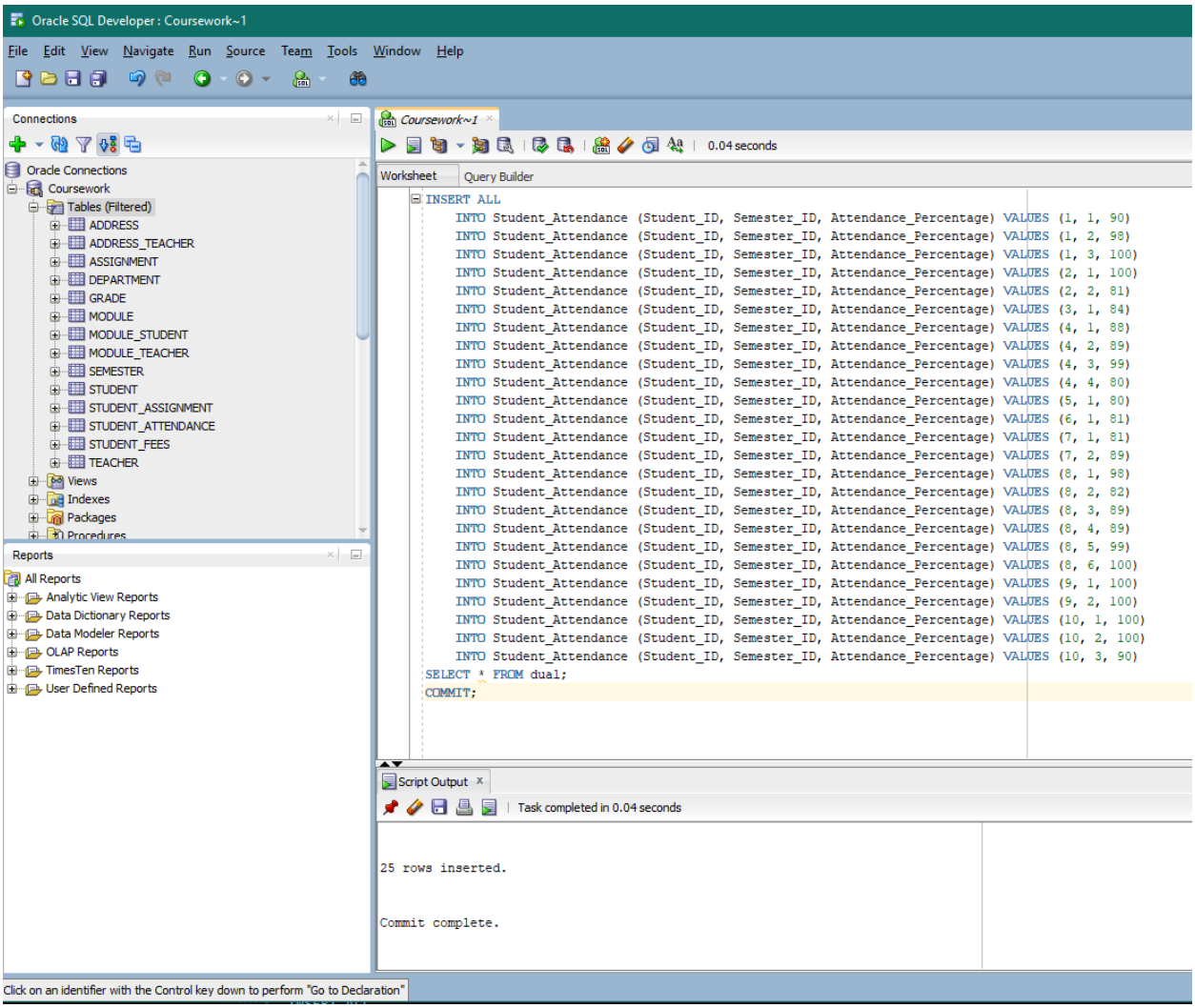


Figure 22: INSERT Student_Attendance

9.11. Student_Assignment

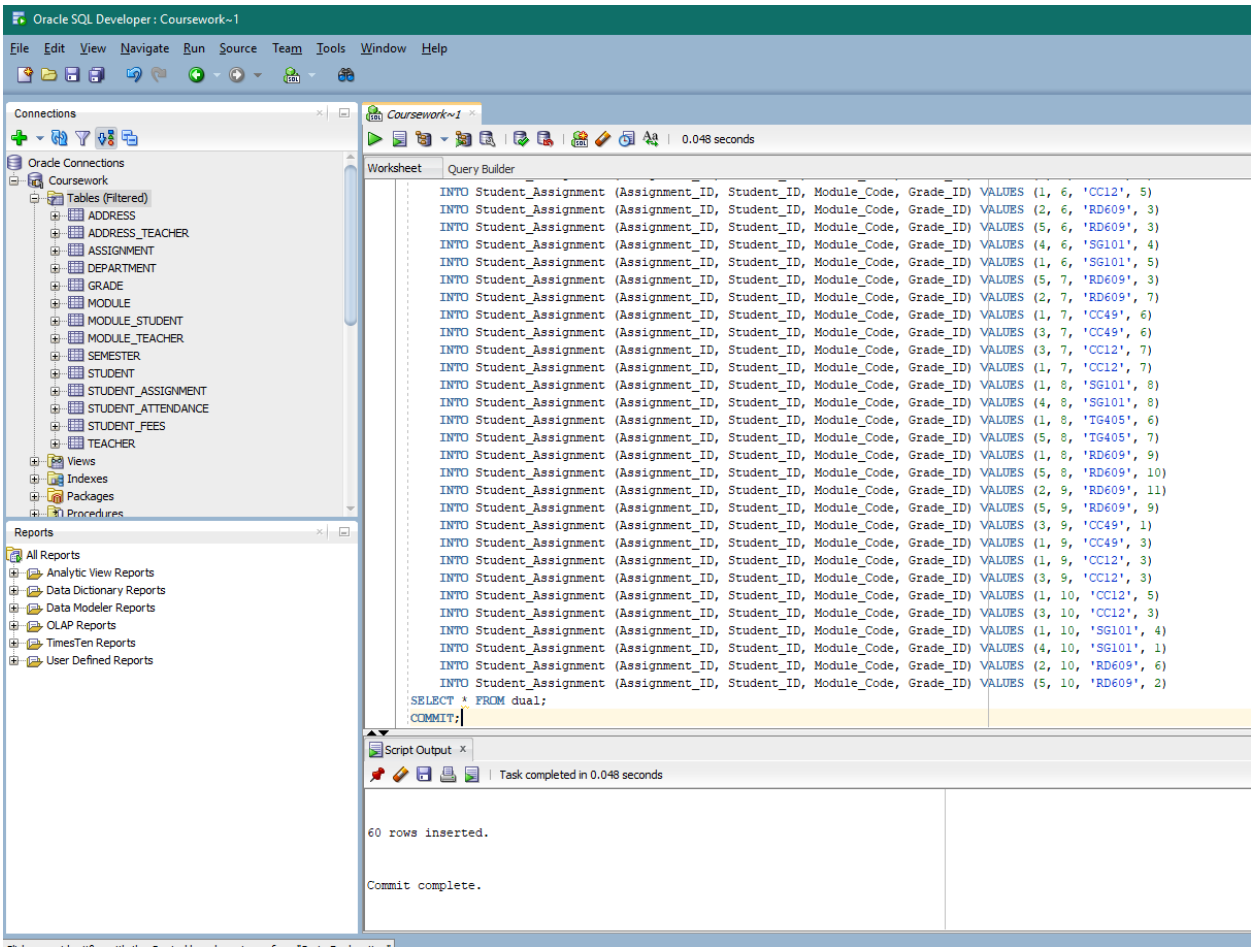


Figure 23: INSERT Student_Assignment

9.12. Module_Teacher

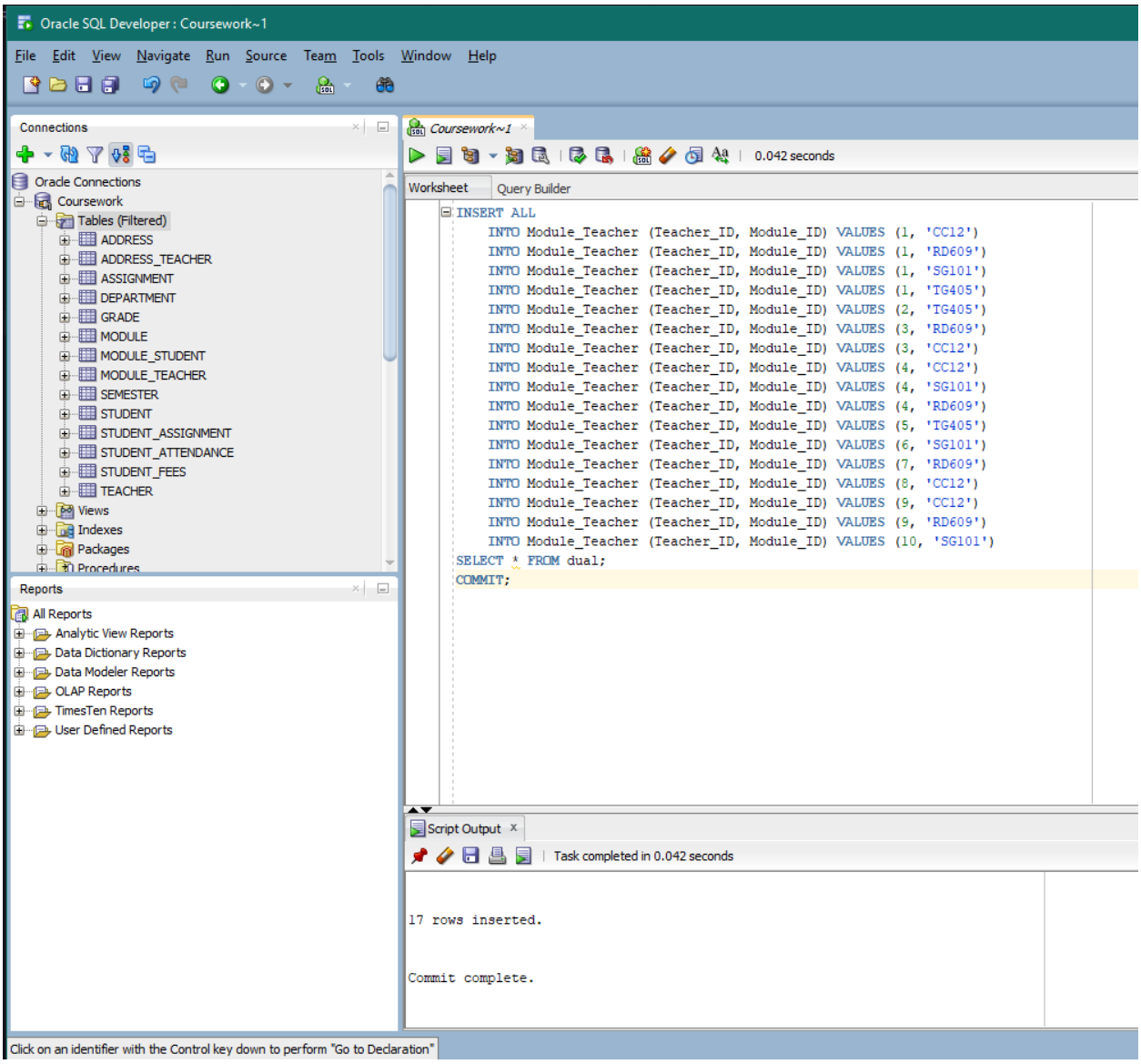


Figure 24: INSERT Module_Teacher

9.13. Module_Student

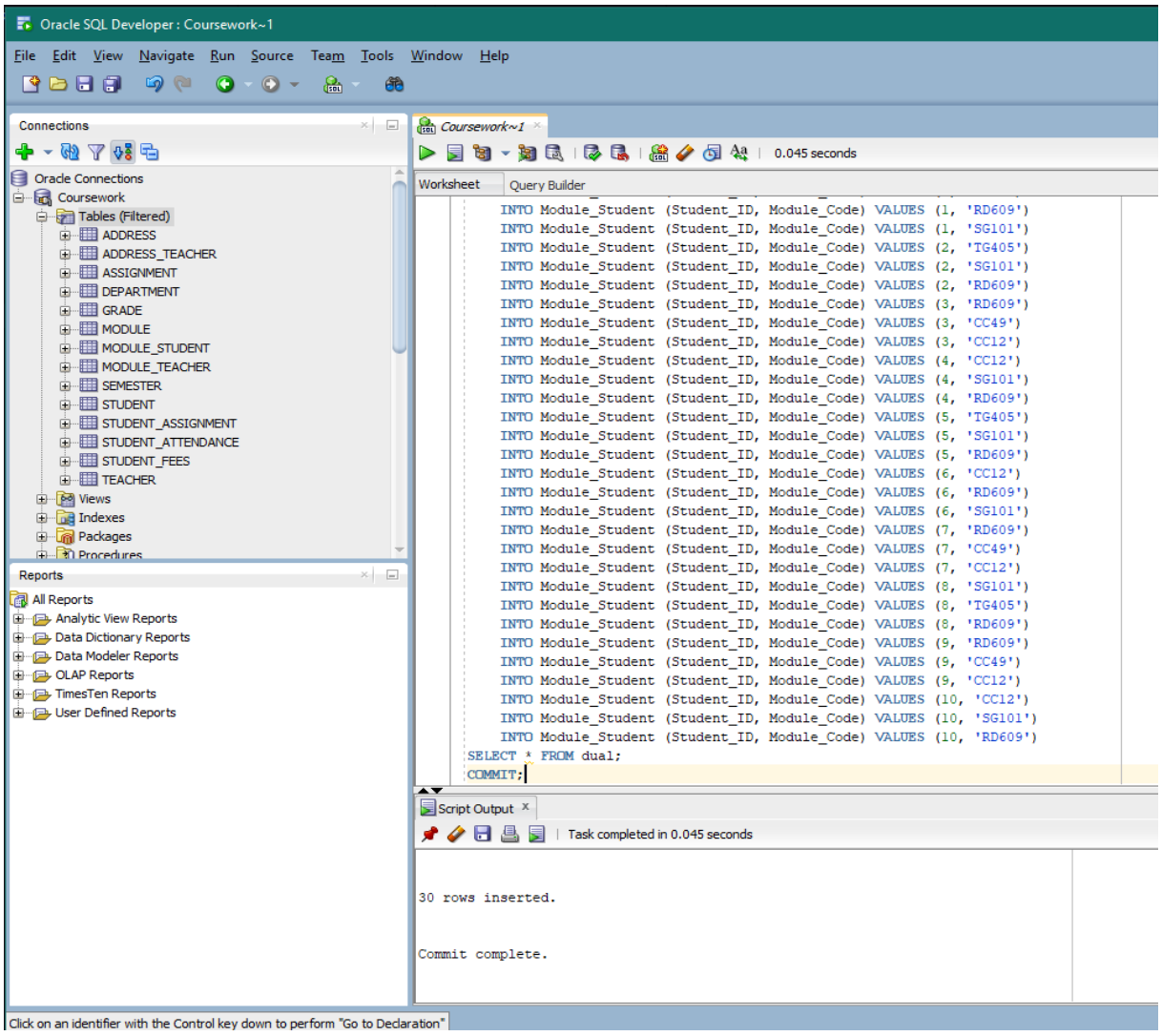


Figure 25: INSERT Module_Student

9.14. Address_Teacher

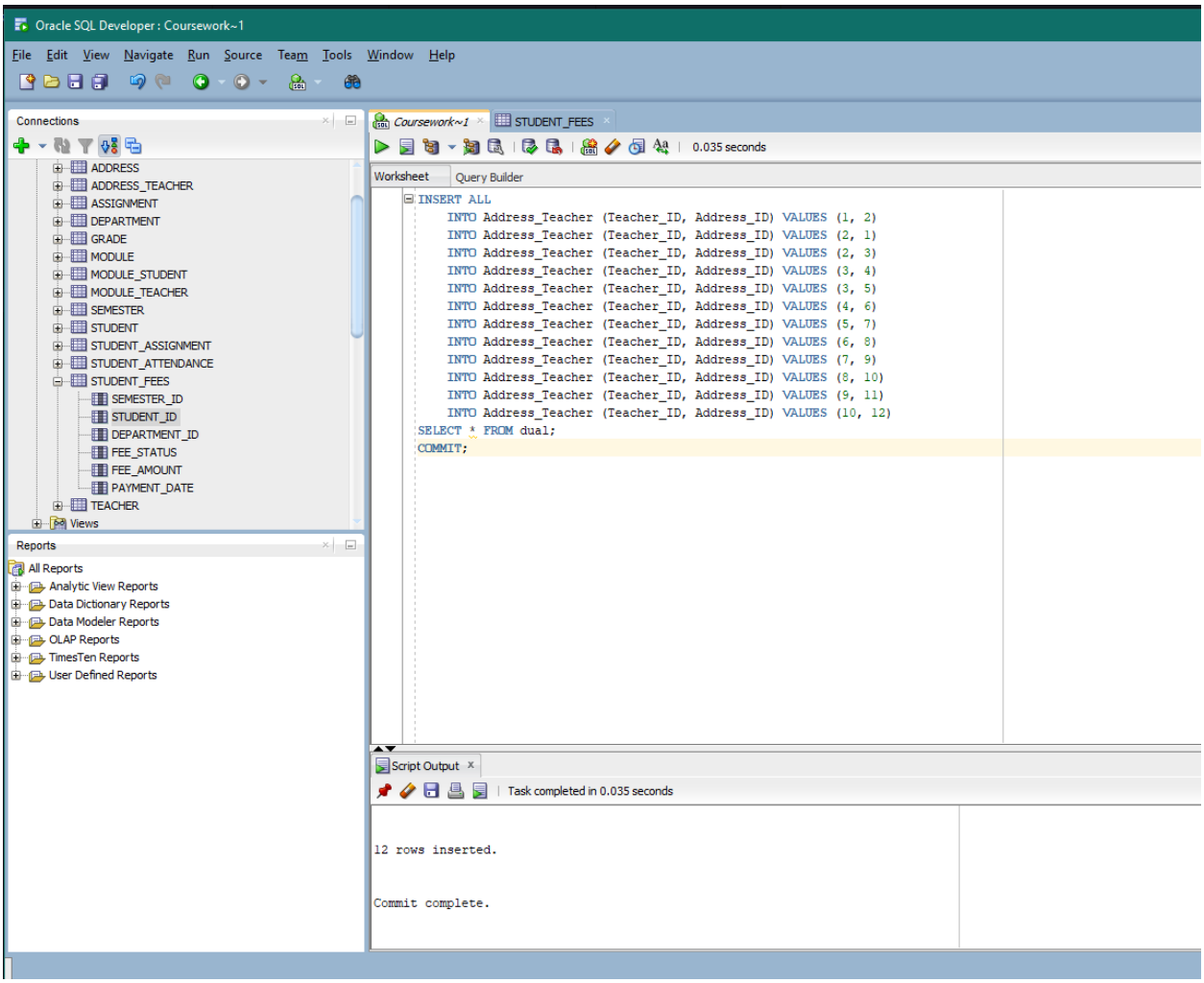


Figure 26: INSERT Address_Teacher

10. SELECT Statements

10.1. Address

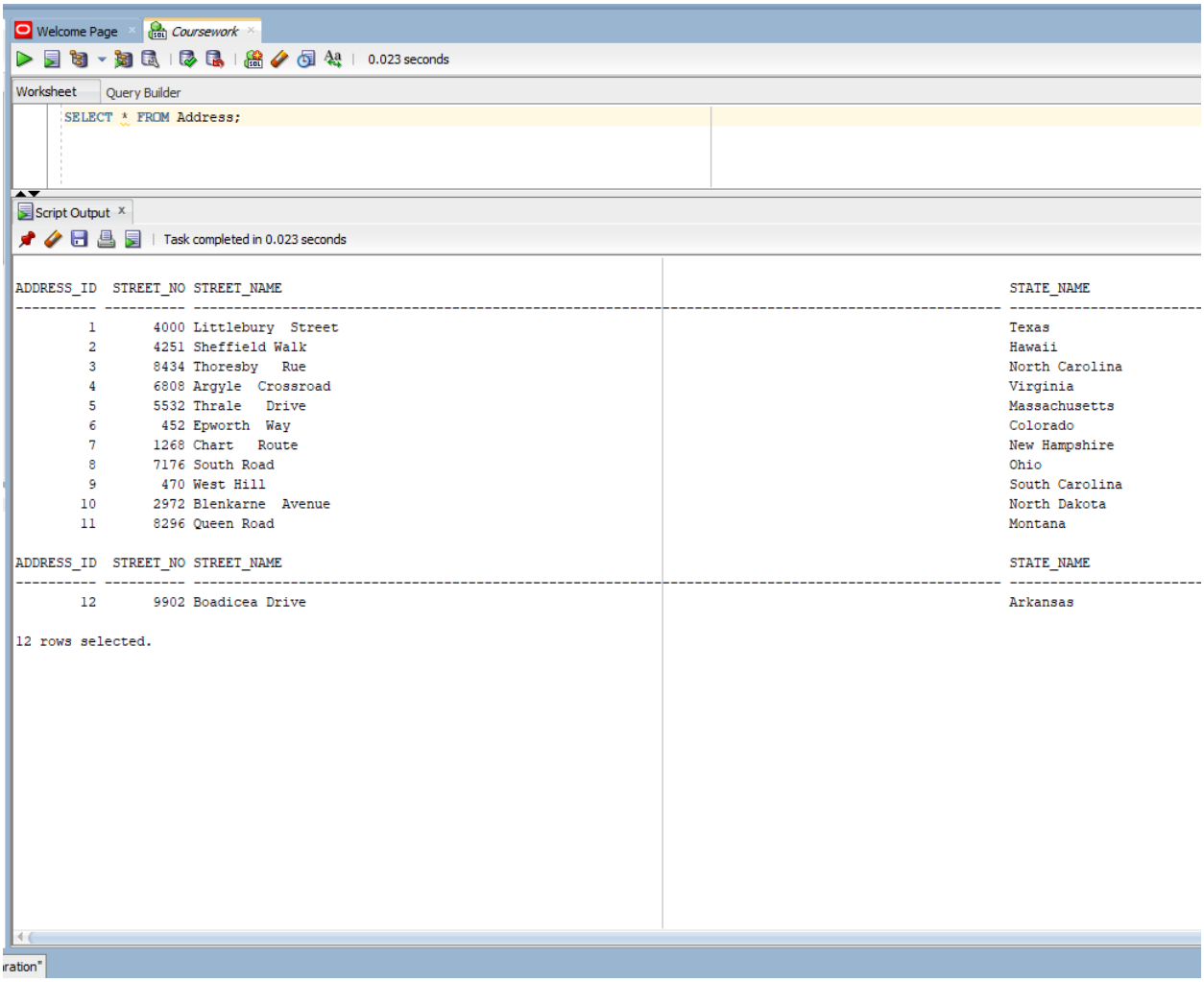


Figure 27:Select Address

10.2. Address_Teacher

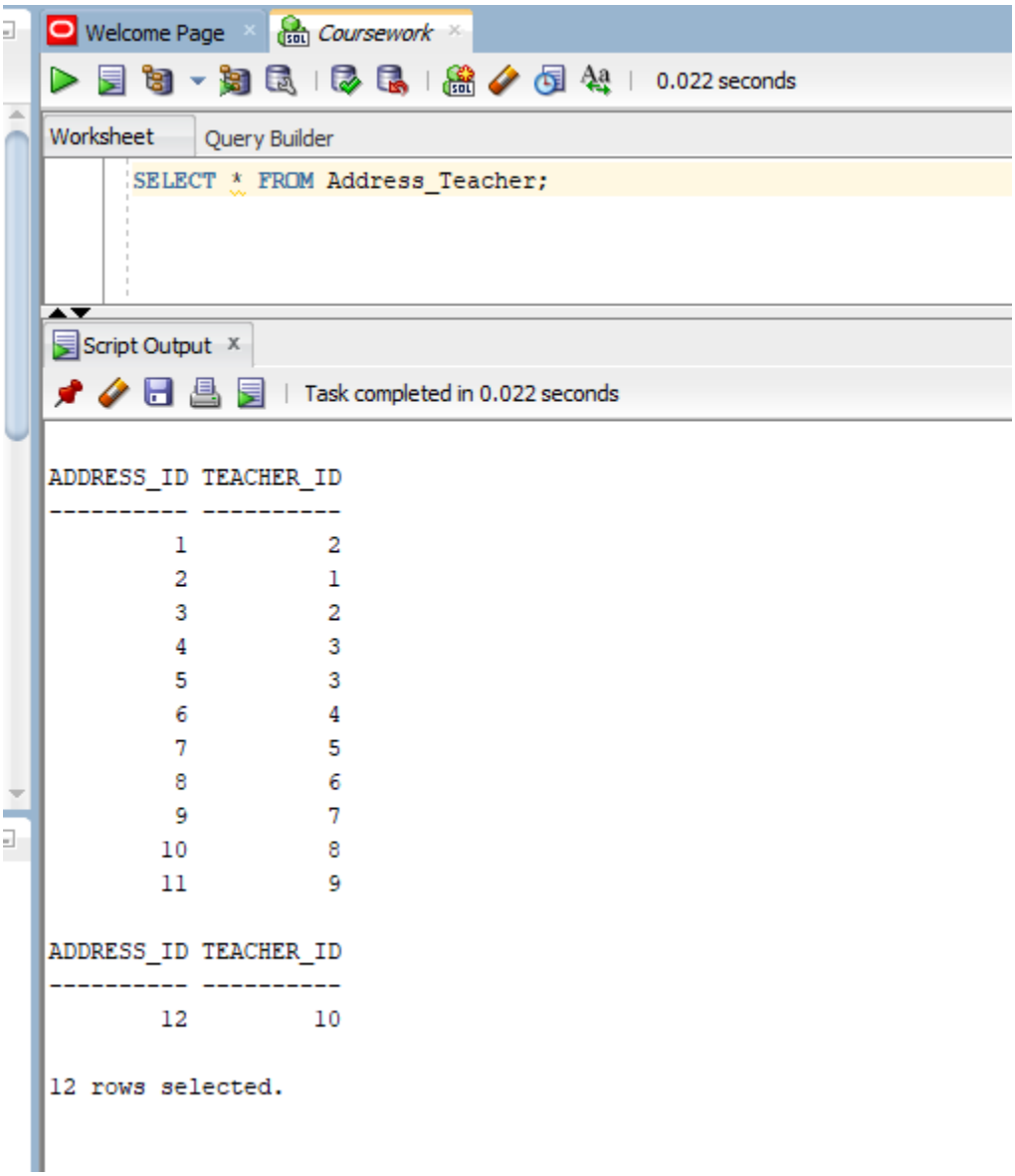


Figure 28:Select Address_Teacher

10.3. Assignment

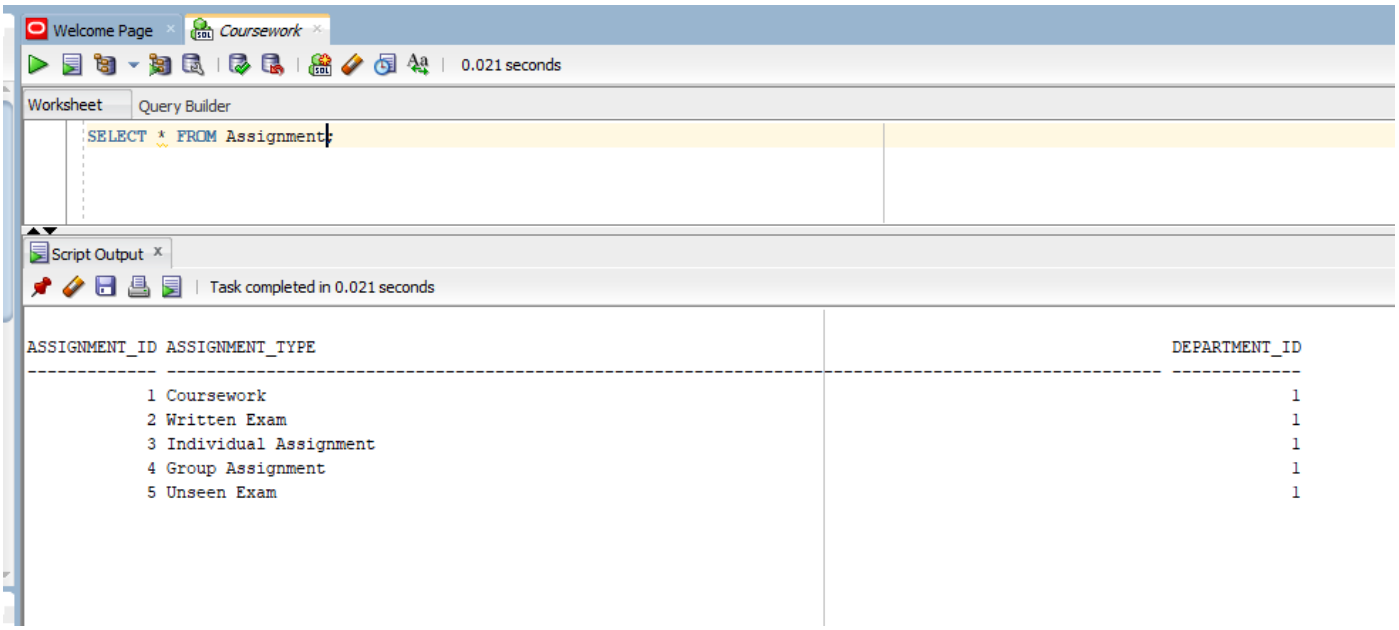


Figure 29:Select Assignment

10.4. Grade

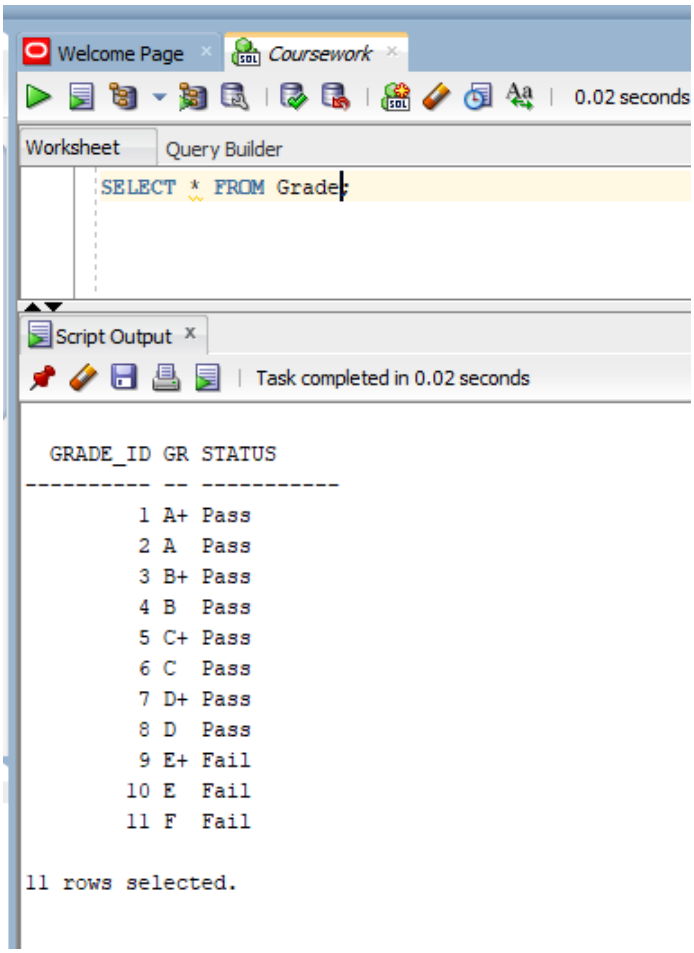


Figure 30:Select Grade

10.5. Module

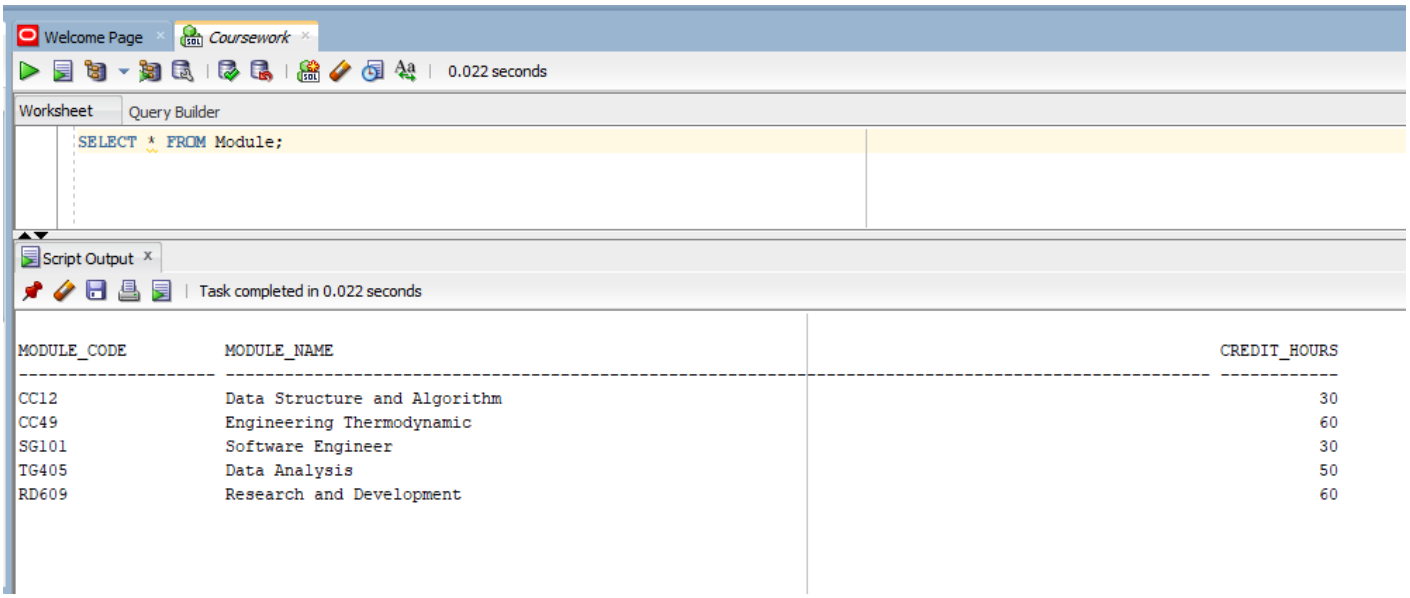


Figure 31:Select Module

10.6. Module_Student

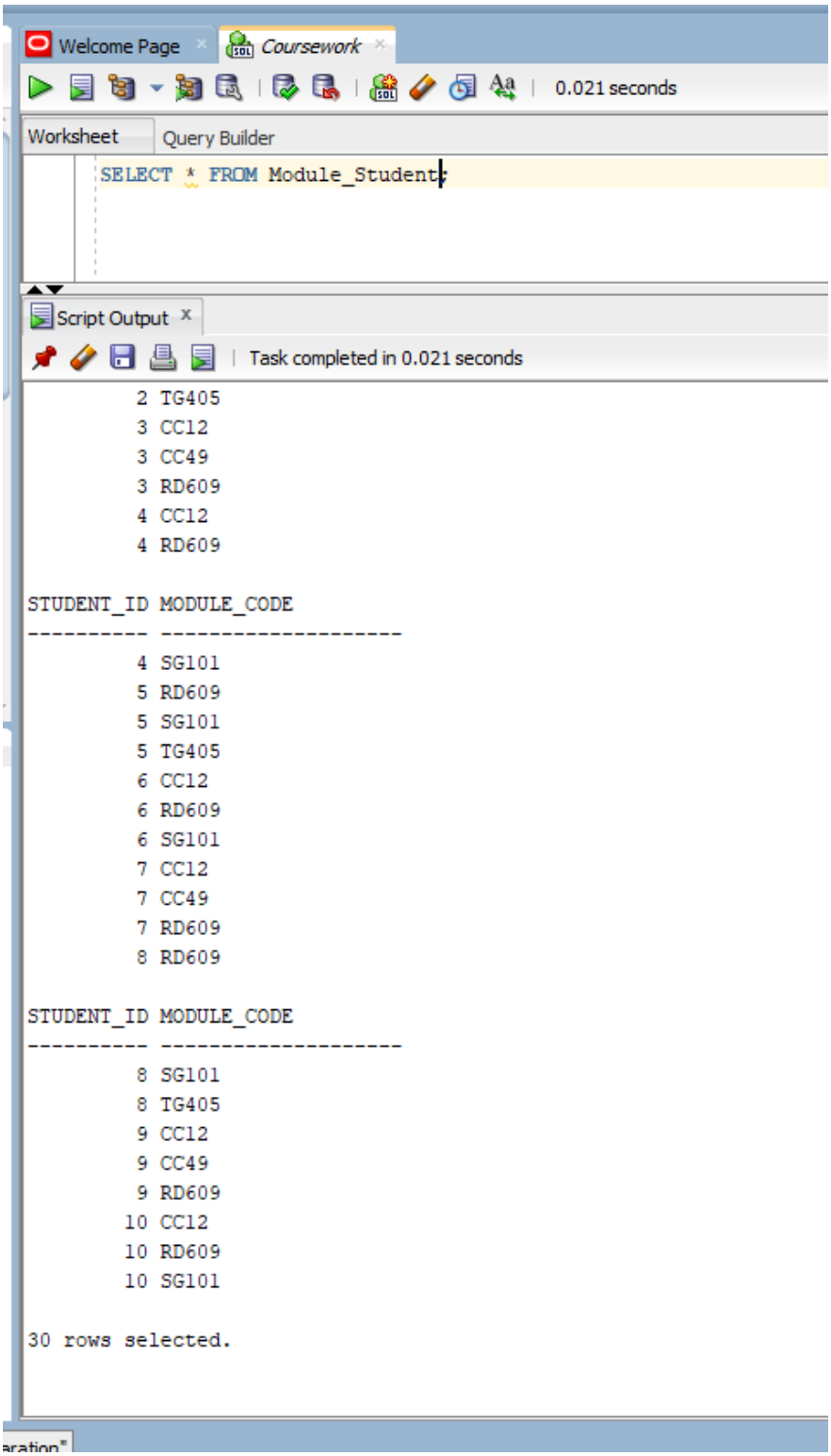


Figure 32:Select Module_Student

10.1. Module_Teacher

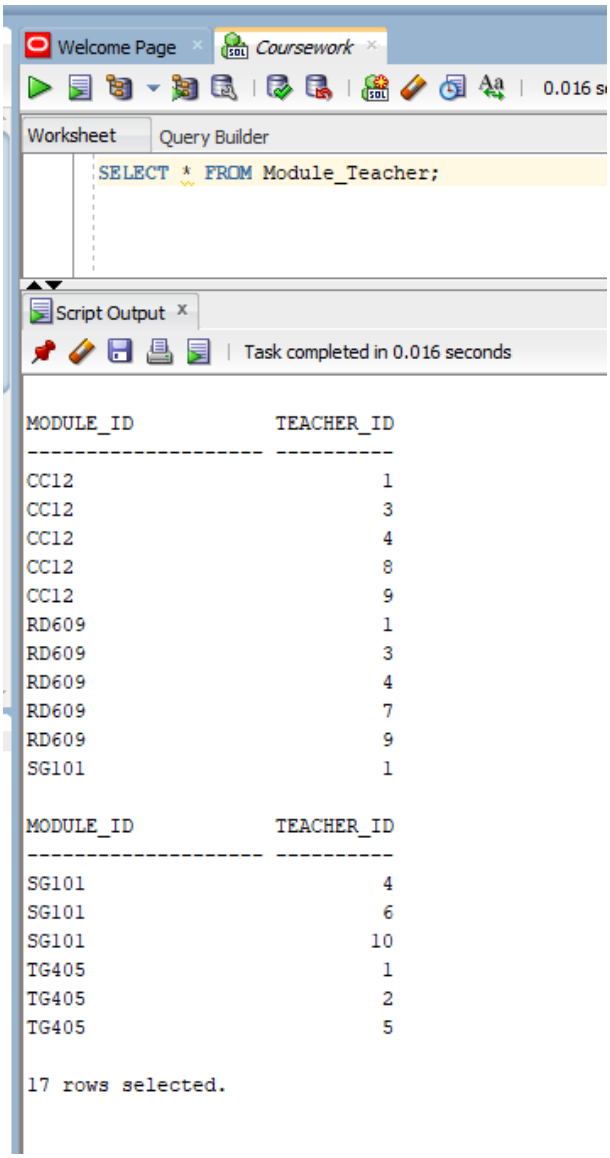


Figure 33:Select Module_Teacher

10.1. Semester

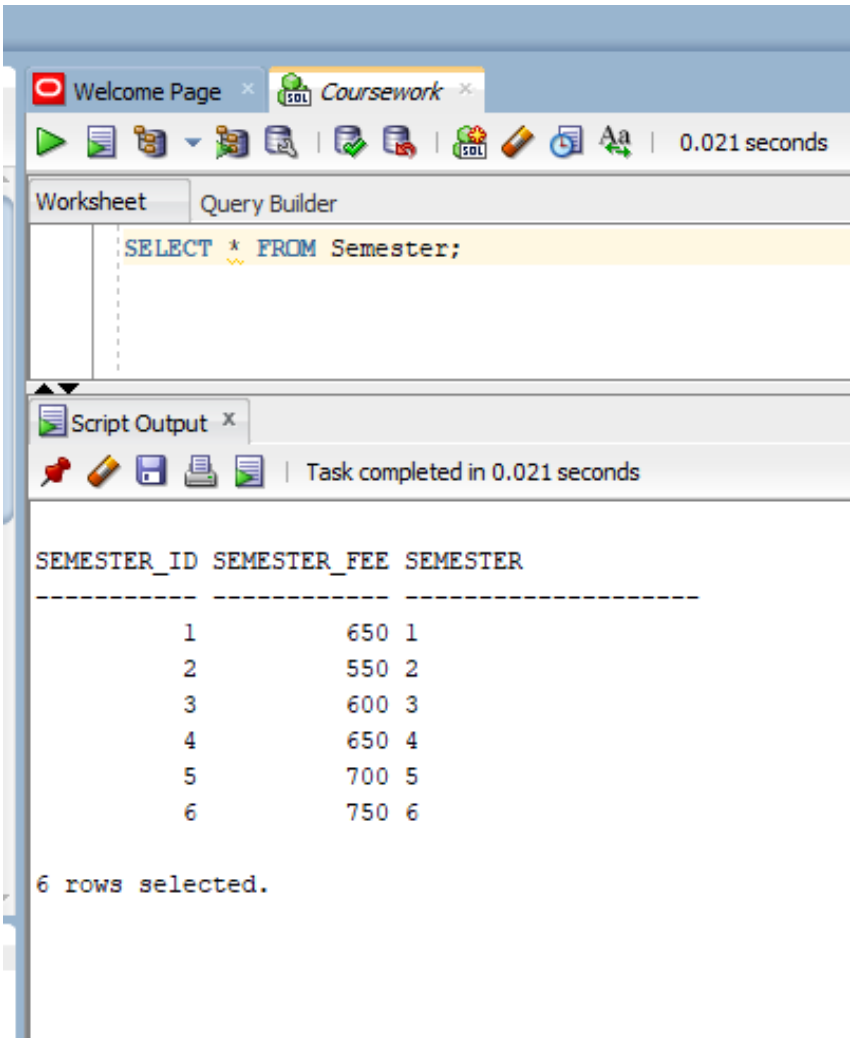


Figure 34:Select Semester

10.1. Student

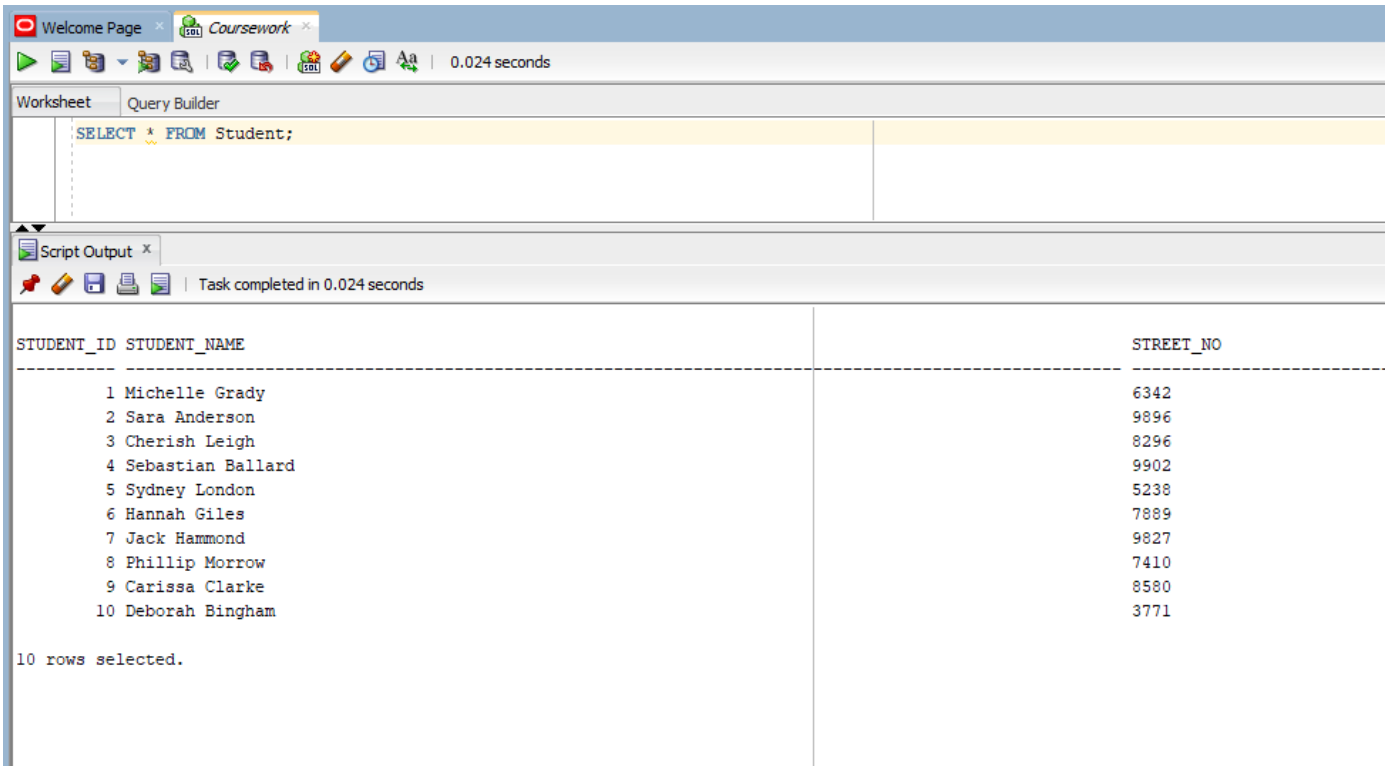


Figure 35:Select Student

10.1. Student_Assignment

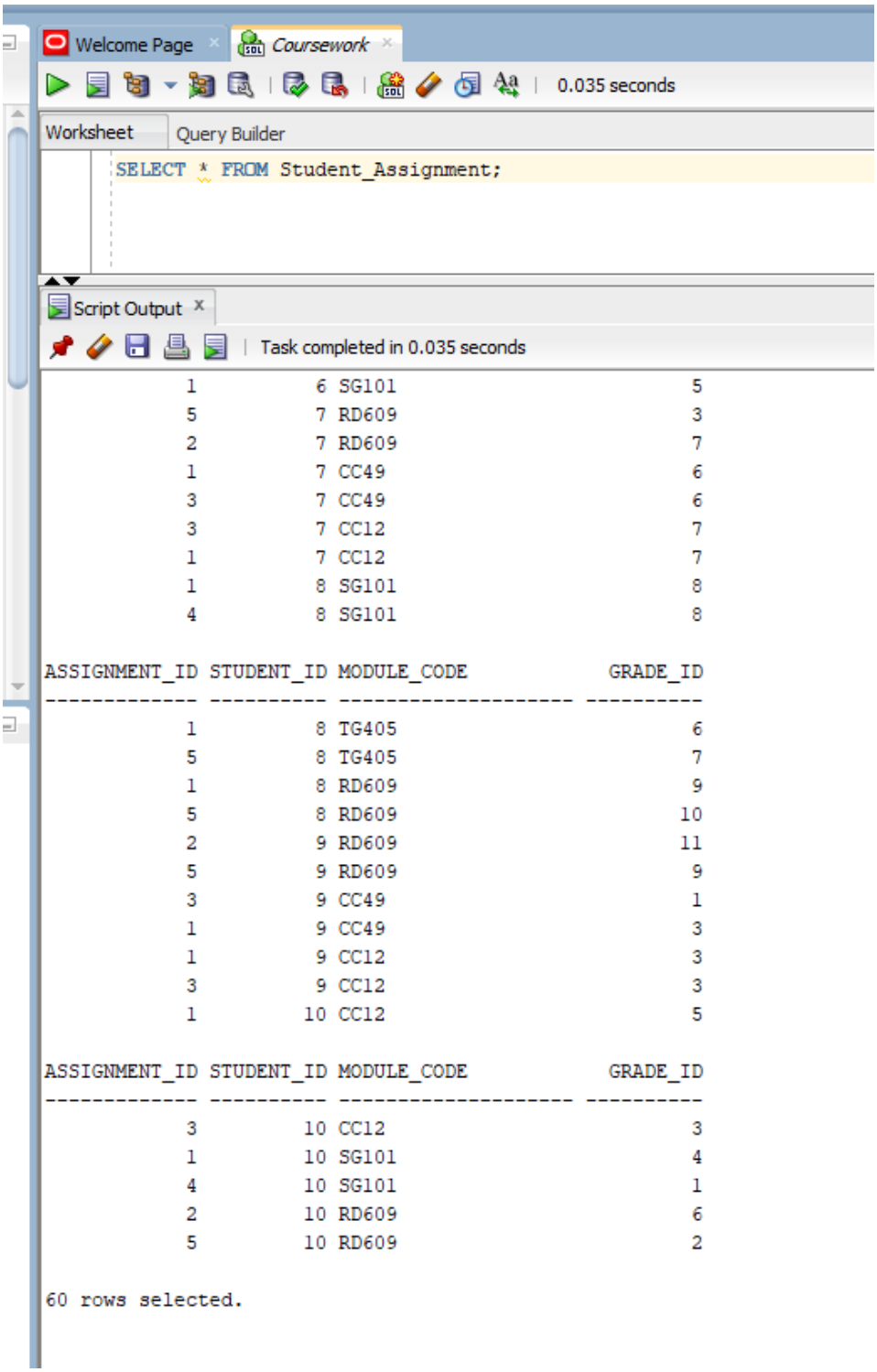


Figure 36:Select Student_Assignment

10.1. Student_Attendance

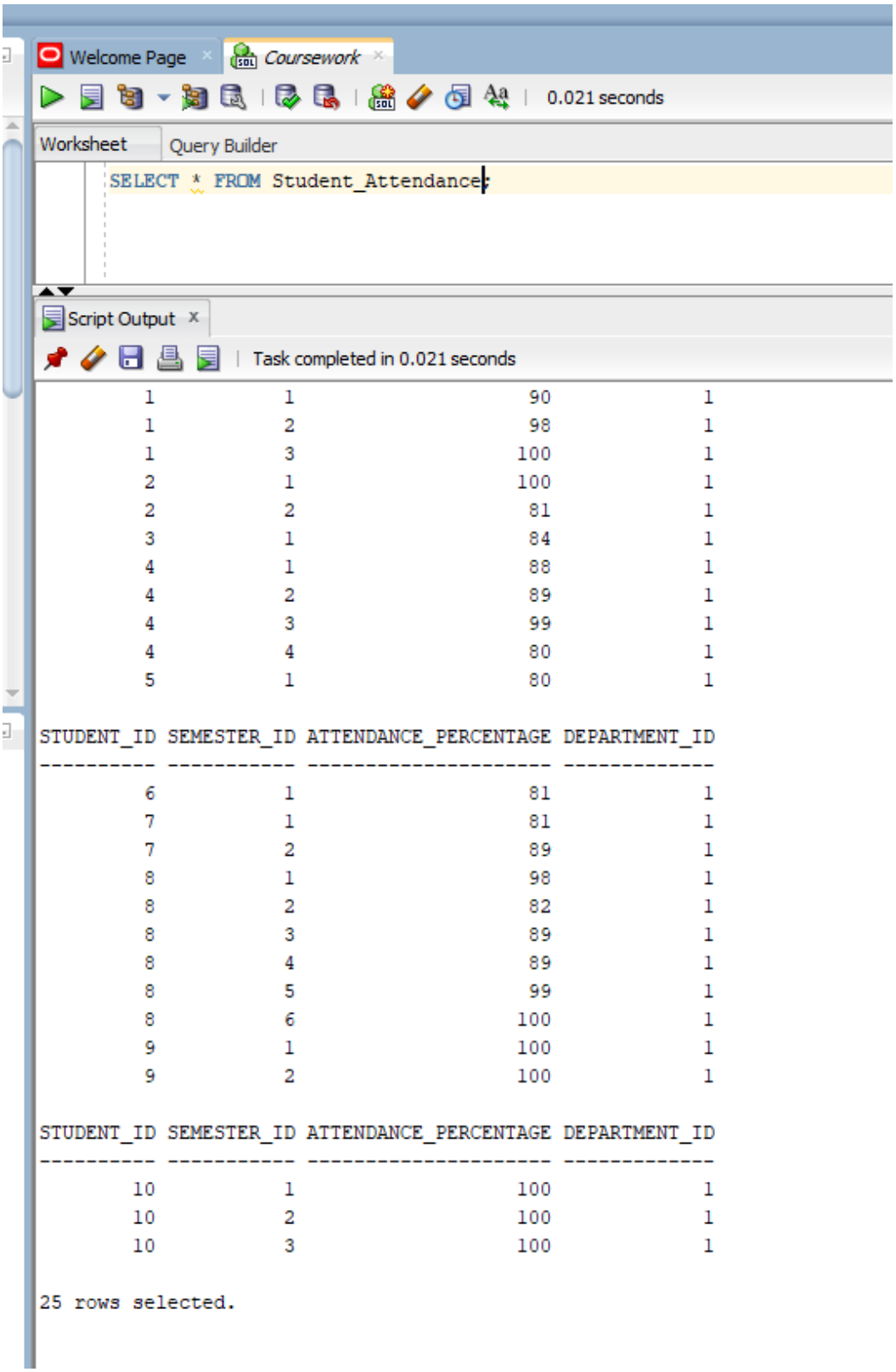


Figure 37:Select Attendance

10.1. Student_Fees

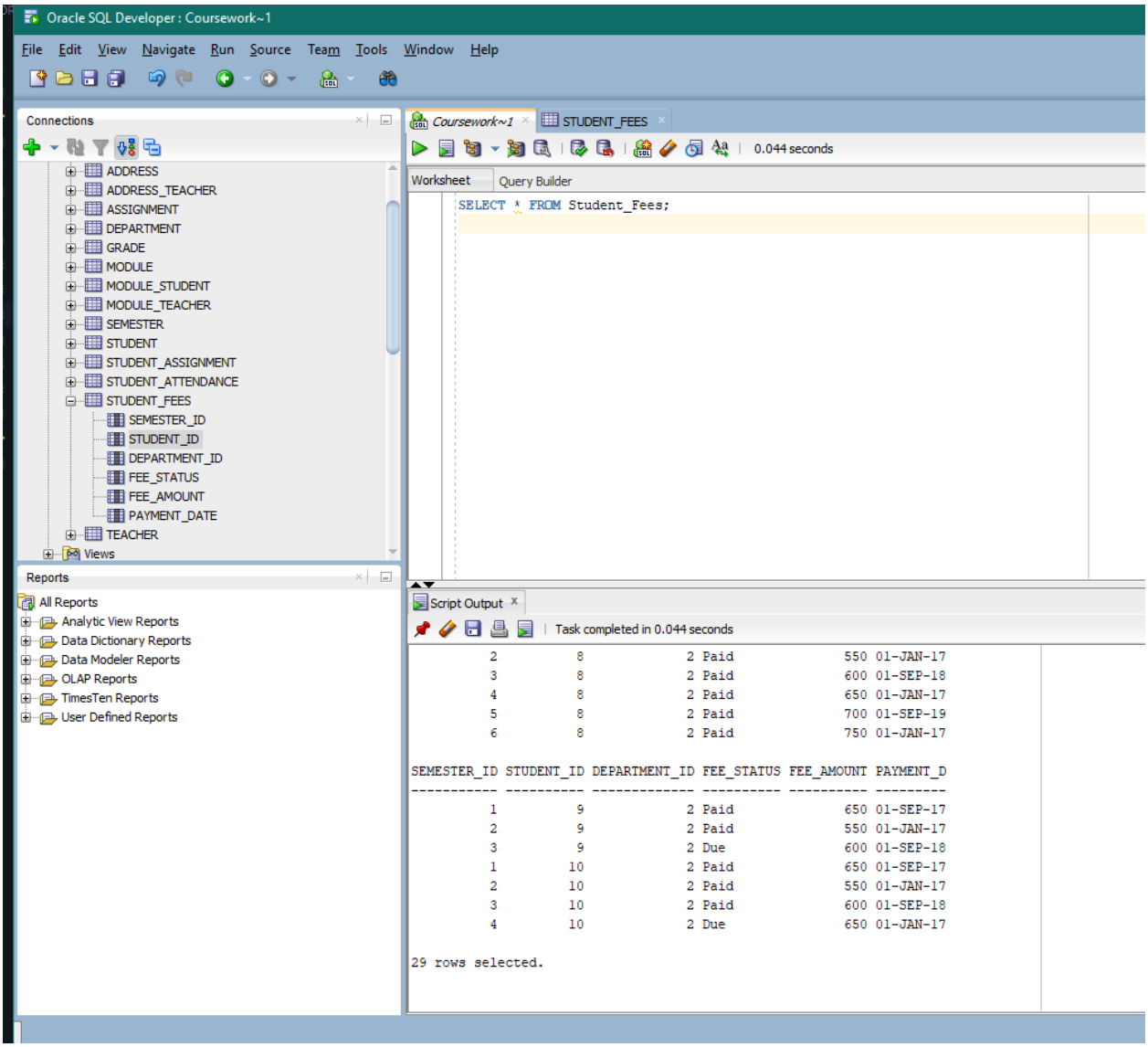


Figure 38:Select Fees

10.1. Teacher

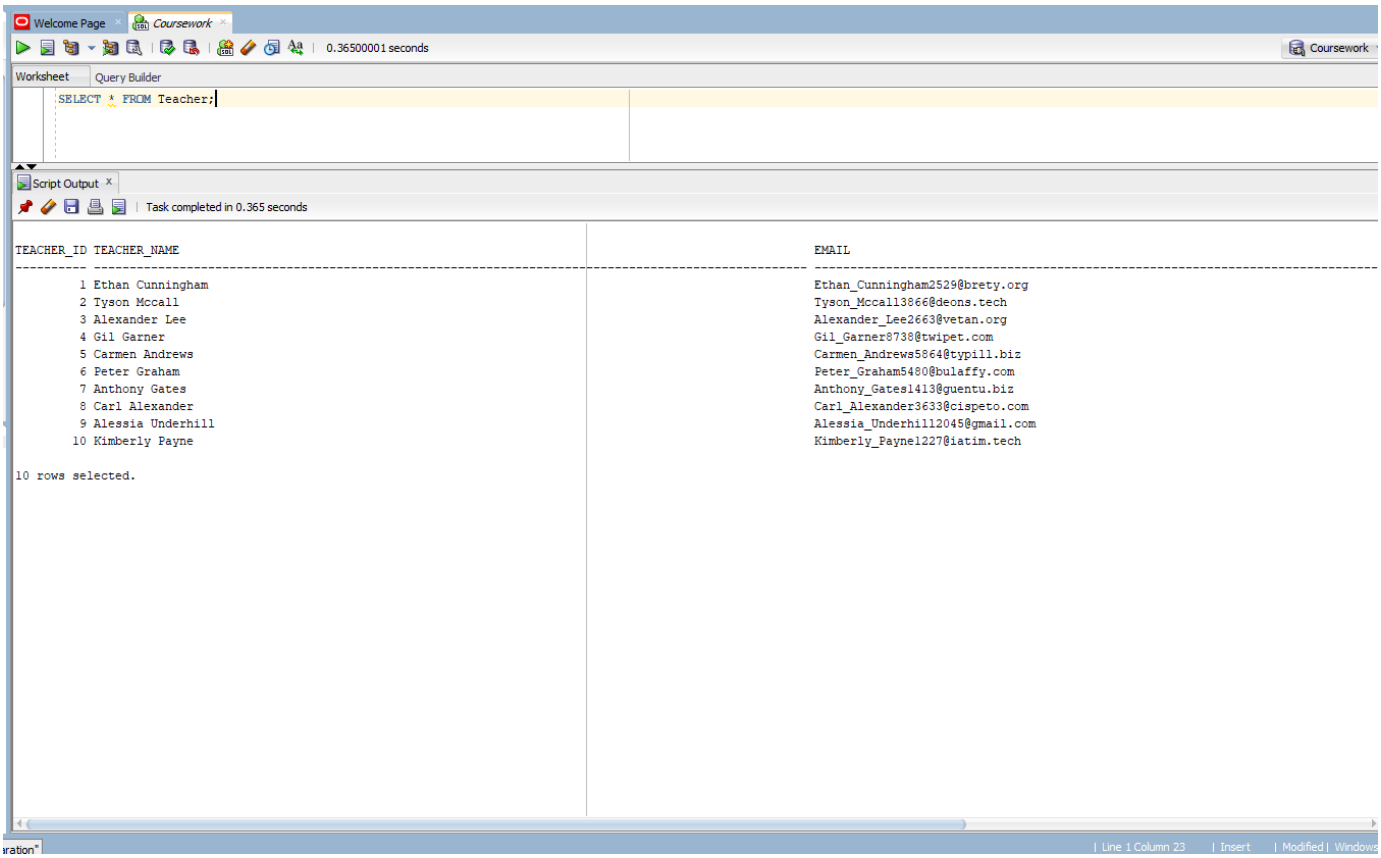


Figure 39:Select Teacher

11. Forms

11.1. Dashboard

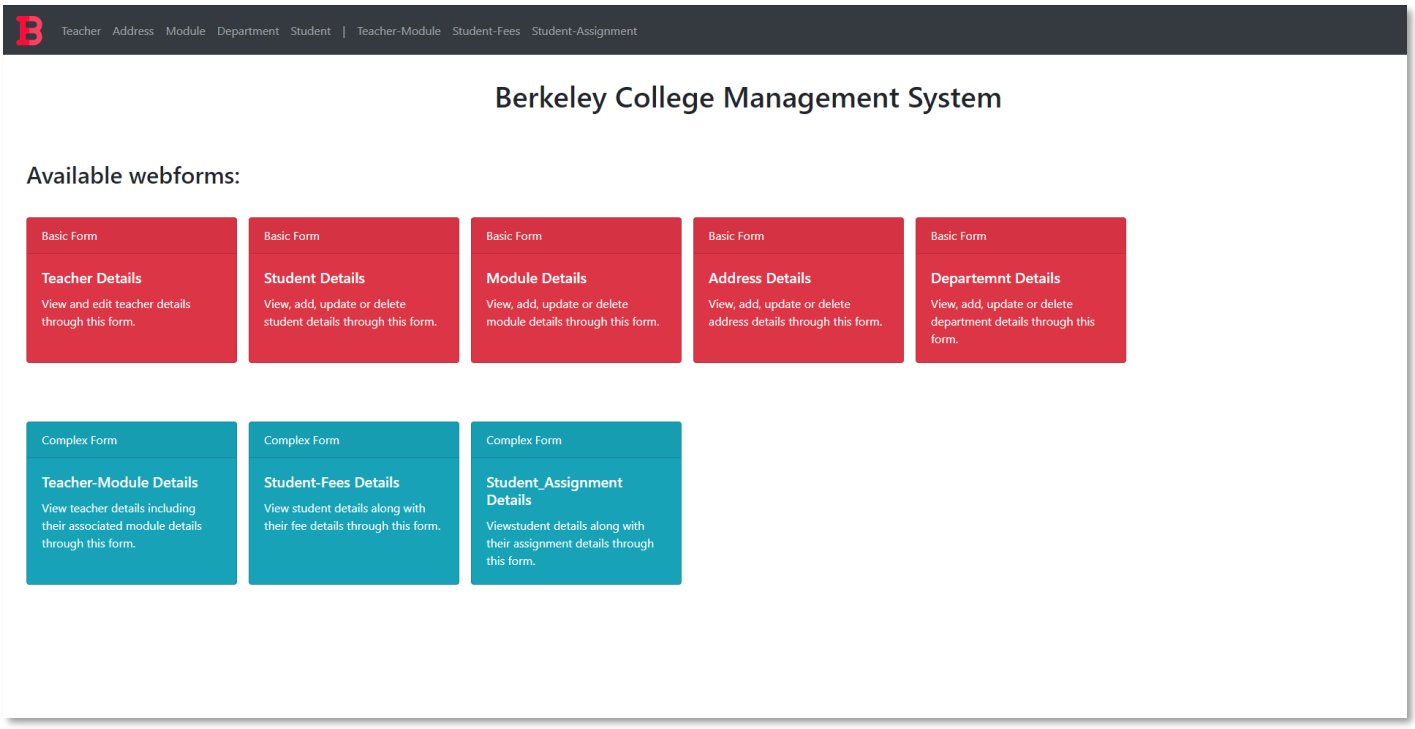


Figure 40: WebForms Dashboard

11.2. Basic Forms

11.2.1. Teacher Form

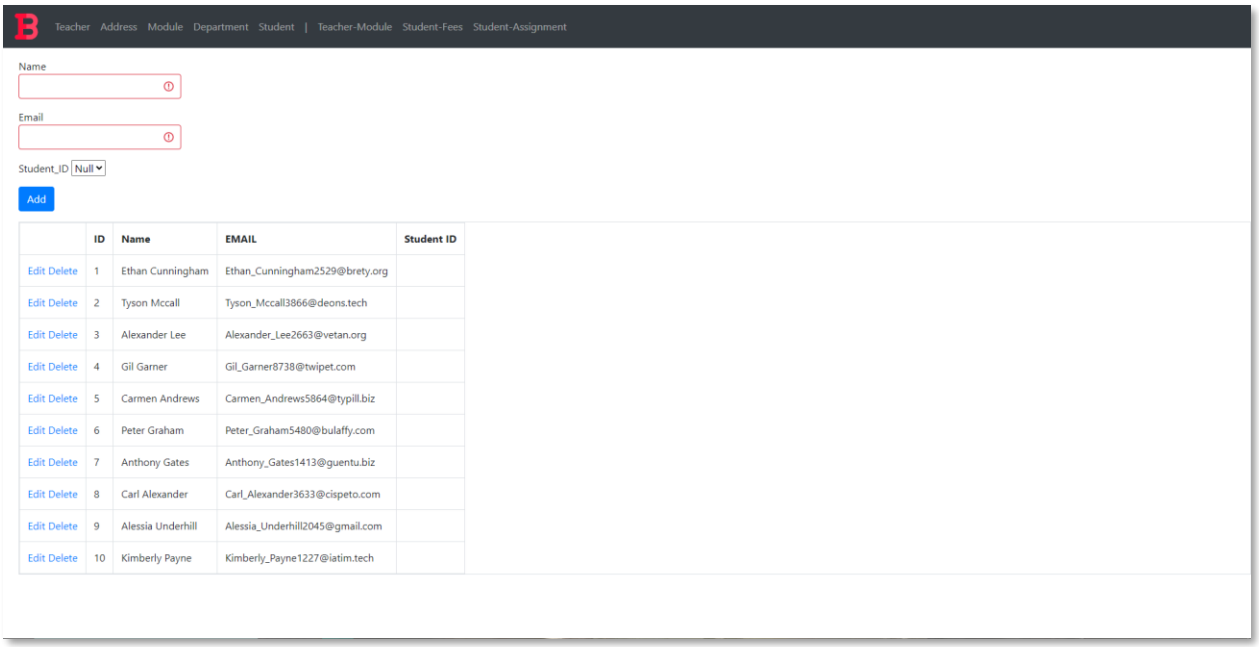


Figure 41: Teacher Form

11.2.2. Address Form

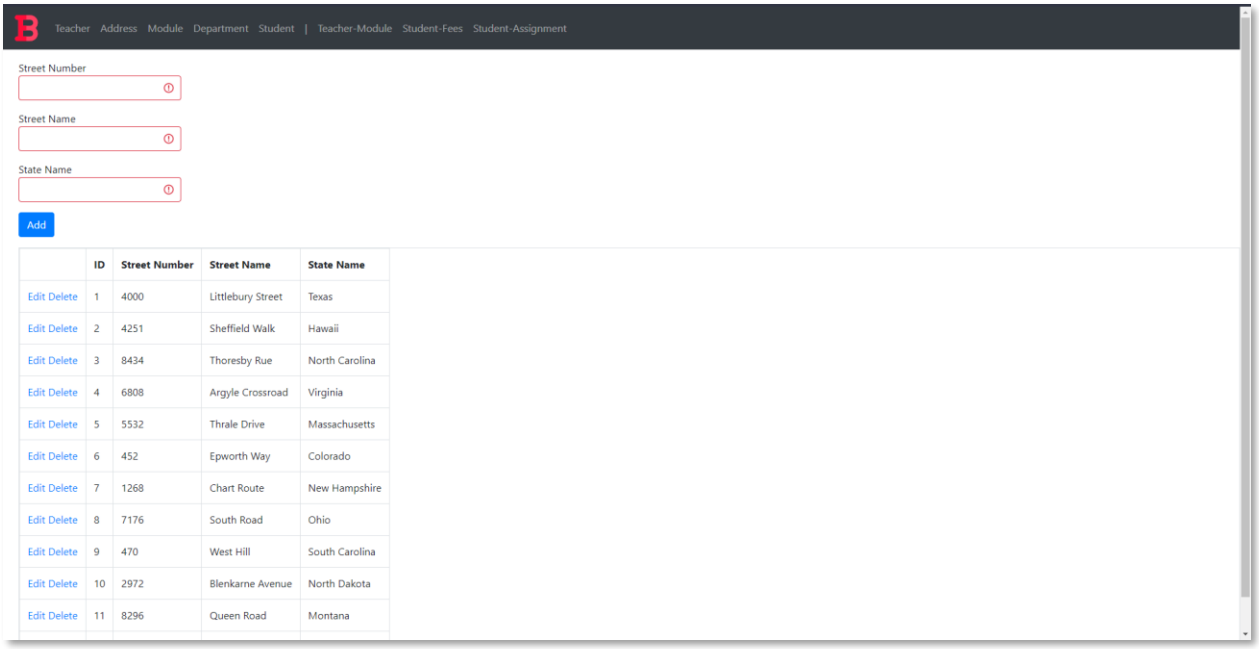


Figure 42: Teacher Form

11.2.3. Module Form

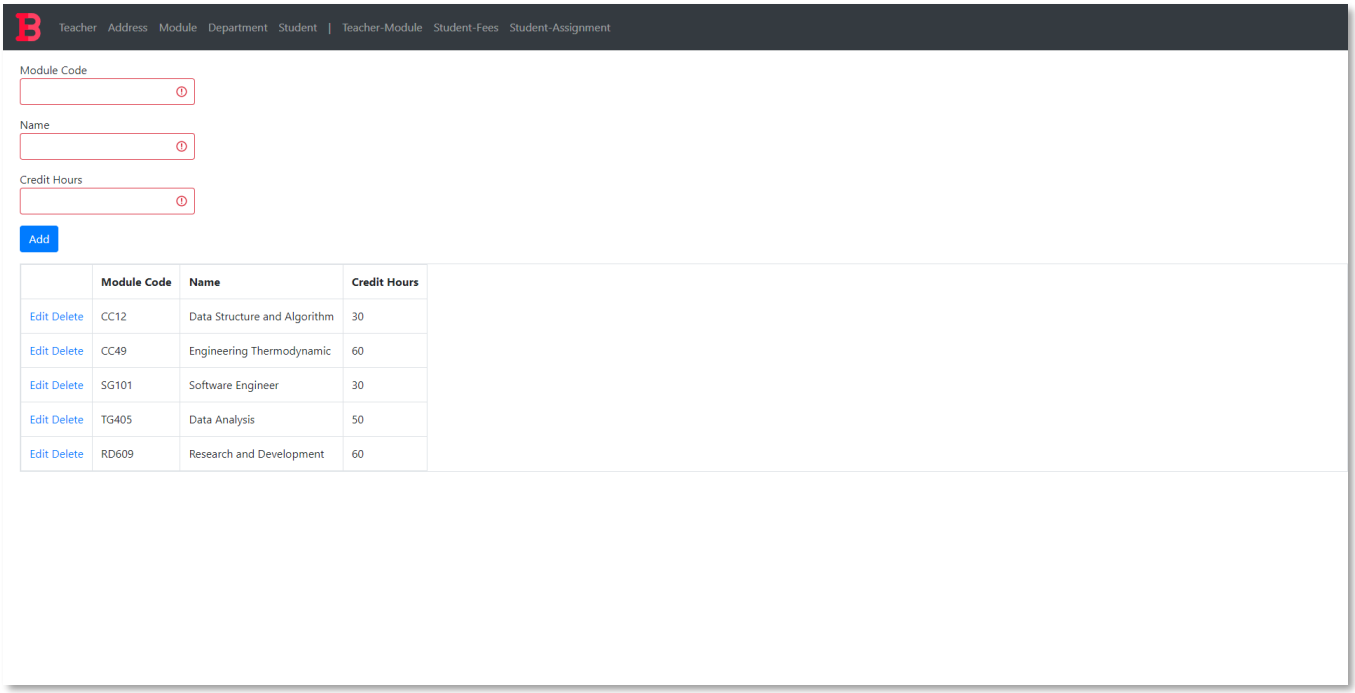


Figure 43: Module Form

11.2.4. Department Form

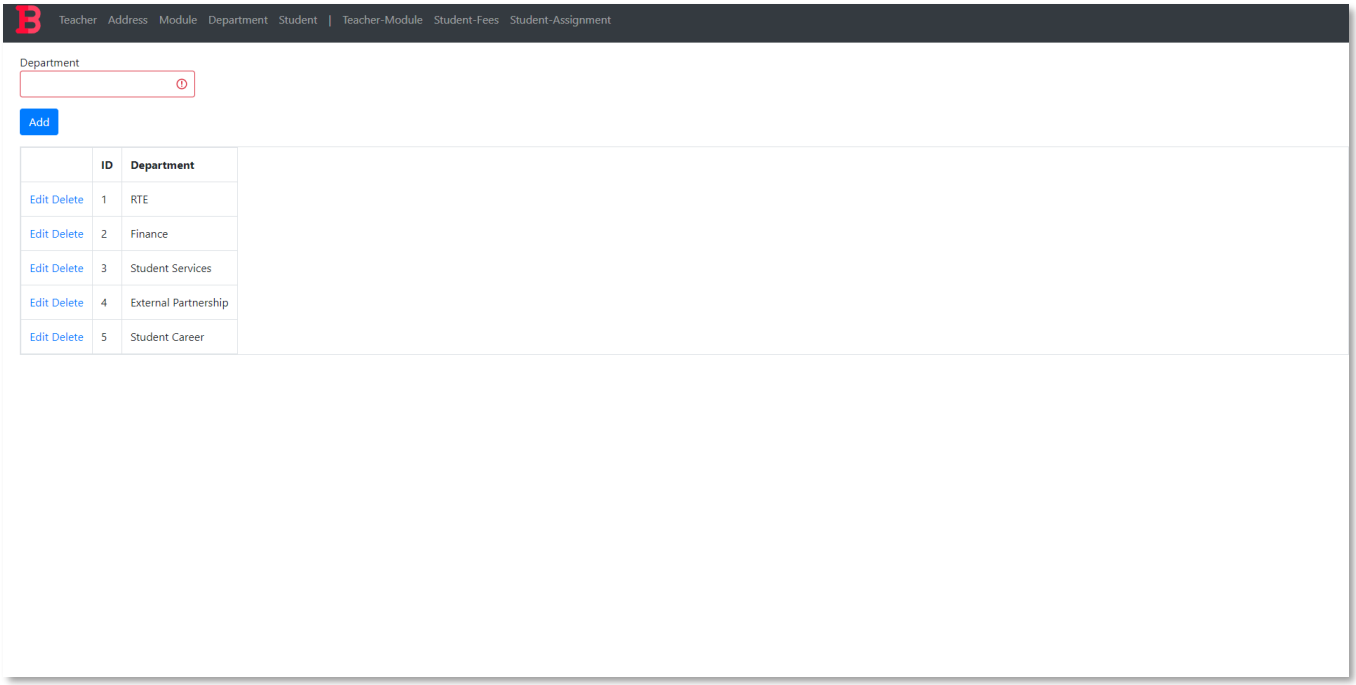


Figure 44: Department Form

11.2.5. Student Form

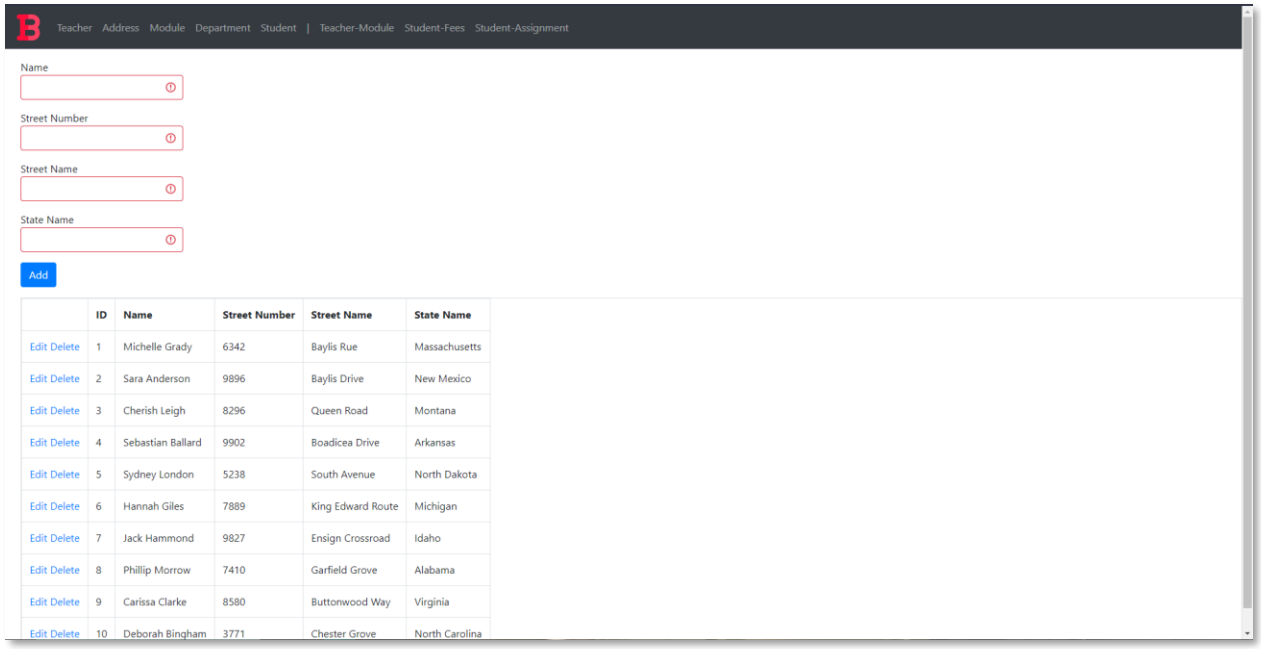


Figure 45: Student Form

11.3. Complex Forms

11.3.1. Teacher-Module Mapping

11.3.1.1. Form

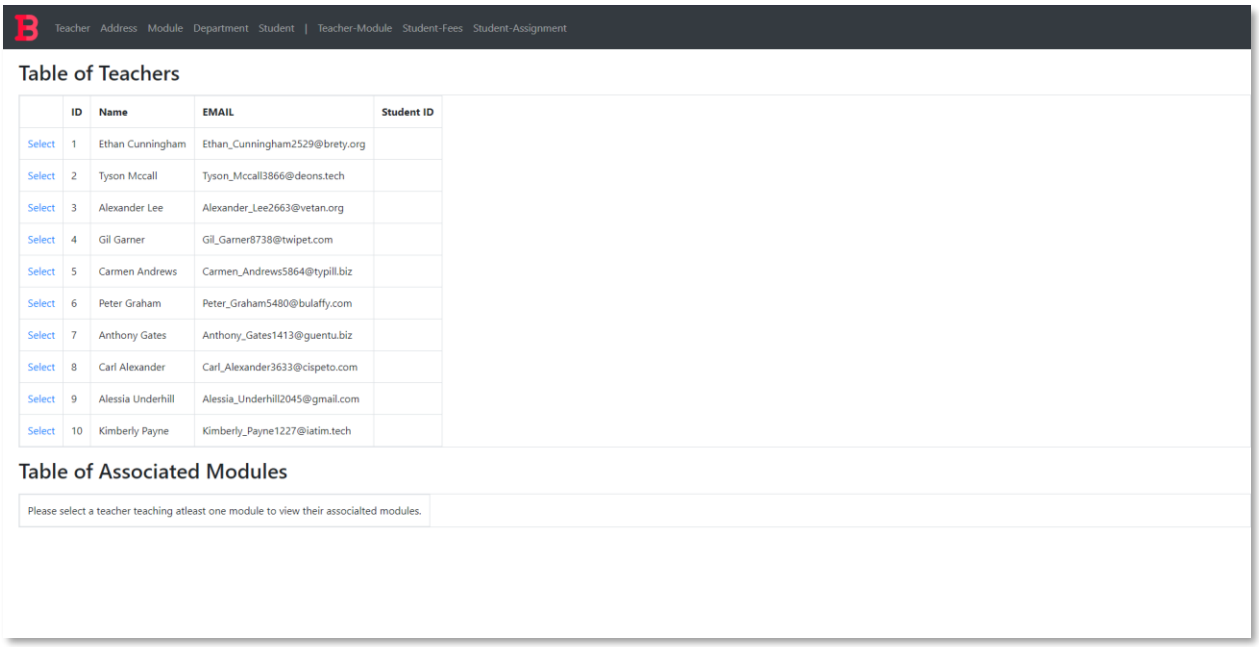


Figure 46: Teacher Module Mapping Form Before Selection

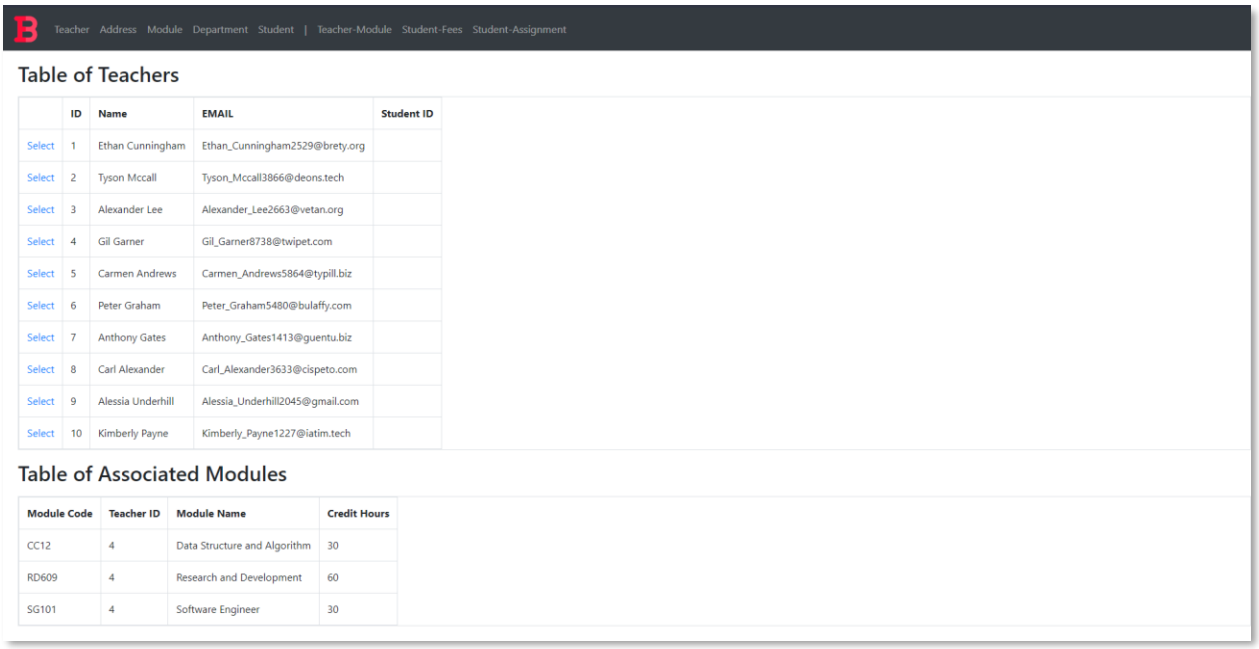


Figure 47: Teacher Module Mapping Form After Selection

11.3.1.2. Query

Assuming the selected student has student_id 1, the query is as following:

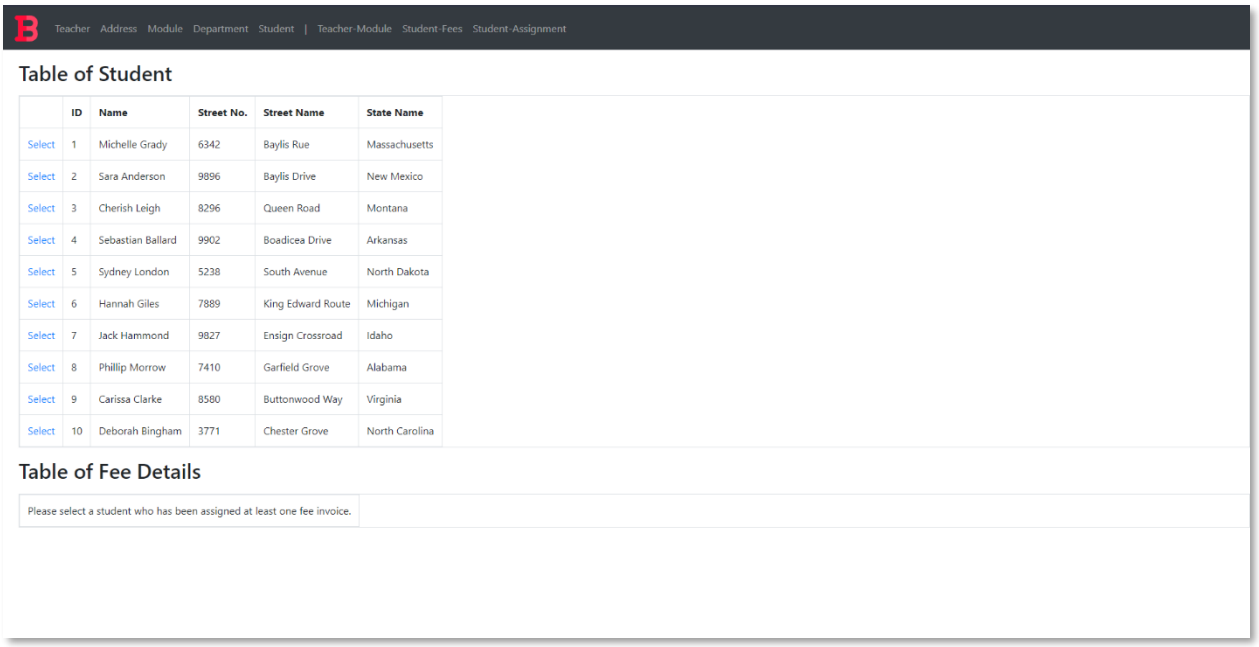
```
SELECT m.Module_Code as "Module Code", mt.Teacher_ID "Teacher ID", m.Module_Name as "Module Name", m.Credit_Hours "Credit Hours"
                                FROM Module m
                                INNER JOIN Module_Teacher mt ON mt.Module_ID = m.Module_Code
                                WHERE mt.Teacher_ID = 1
                                ;
```

In the webform the query string is written as:

```
$@"SELECT m.Module_Code as ""Module Code"", mt.Teacher_ID ""Teacher ID"", m.Module_Name as ""Module Name"", m.Credit_Hours ""Credit Hours""
                                FROM Module m
                                INNER JOIN Module_Teacher mt ON mt.Module_ID = m.Module_Code
                                WHERE mt.Teacher_ID = {teacherID}
                                ";
```

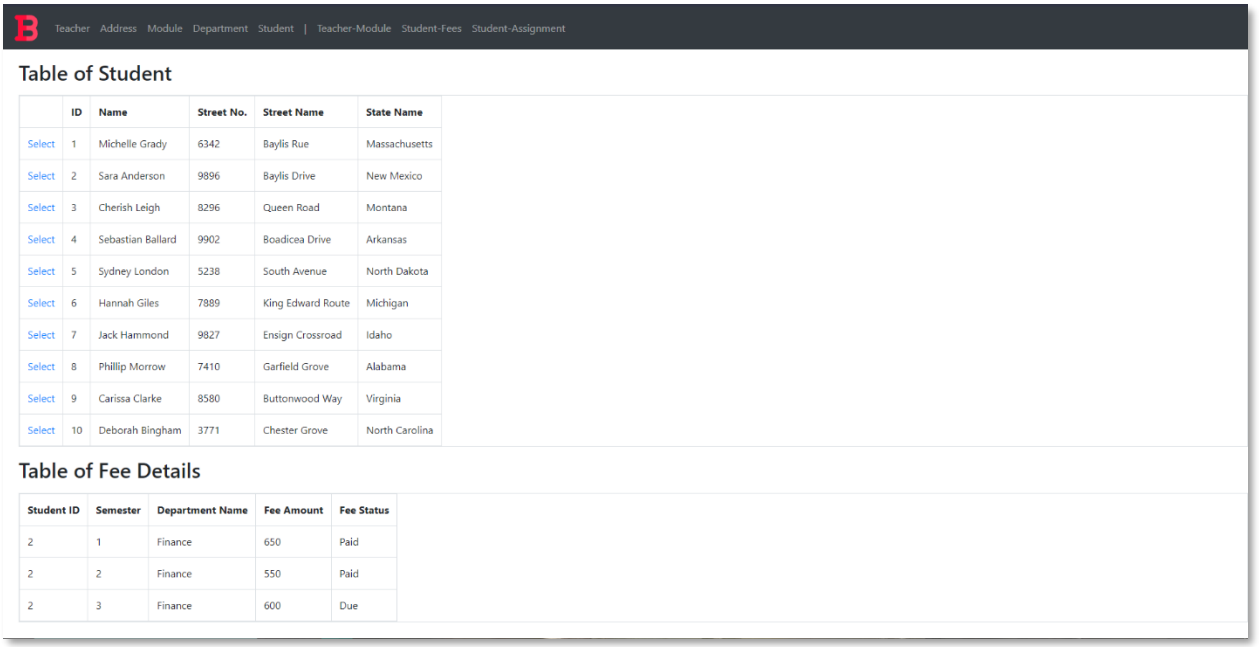
11.4. Student-Fees Mapping

11.4.1.1. Form



	ID	Name	Street No.	Street Name	State Name
Select	1	Michelle Grady	6342	Baylis Rue	Massachusetts
Select	2	Sara Anderson	9896	Baylis Drive	New Mexico
Select	3	Cherish Leigh	8296	Queen Road	Montana
Select	4	Sebastian Ballard	9902	Boadicea Drive	Arkansas
Select	5	Sydney London	5238	South Avenue	North Dakota
Select	6	Hannah Giles	7889	King Edward Route	Michigan
Select	7	Jack Hammond	9827	Ensign Crossroad	Idaho
Select	8	Phillip Morrow	7410	Garfield Grove	Alabama
Select	9	Carissa Clarke	8580	Buttonwood Way	Virginia
Select	10	Deborah Bingham	3771	Chester Grove	North Carolina

Figure 48: Student-Fees Mapping Form Before Selection



	ID	Name	Street No.	Street Name	State Name
Select	1	Michelle Grady	6342	Baylis Rue	Massachusetts
Select	2	Sara Anderson	9896	Baylis Drive	New Mexico
Select	3	Cherish Leigh	8296	Queen Road	Montana
Select	4	Sebastian Ballard	9902	Boadicea Drive	Arkansas
Select	5	Sydney London	5238	South Avenue	North Dakota
Select	6	Hannah Giles	7889	King Edward Route	Michigan
Select	7	Jack Hammond	9827	Ensign Crossroad	Idaho
Select	8	Phillip Morrow	7410	Garfield Grove	Alabama
Select	9	Carissa Clarke	8580	Buttonwood Way	Virginia
Select	10	Deborah Bingham	3771	Chester Grove	North Carolina

Student ID	Semester	Department Name	Fee Amount	Fee Status
2	1	Finance	650	Paid
2	2	Finance	550	Paid
2	3	Finance	600	Due

Figure 49: Student-Fees Mapping Form After Selection

11.4.1.2. Query

Assuming the selected student has student_id 1, the query is as following:

```
SELECT sf.Student_ID "Student ID", se.Semester "Semester",  d.Department_Name as "Department Name", sf.Fee_Amount "Fee Amount", sf.Fee_Status "Fee Status"
                                FROM Student s
                                INNER JOIN Student_Fees sf ON sf.Student_ID = s.Student_ID
                                INNER JOIN Semester se ON se.Semester_ID = sf.Semester_ID
                                INNER JOIN Department d ON d.Department_ID = sf.Department_ID
                                WHERE sf.Student_ID = 1
                                ;

```

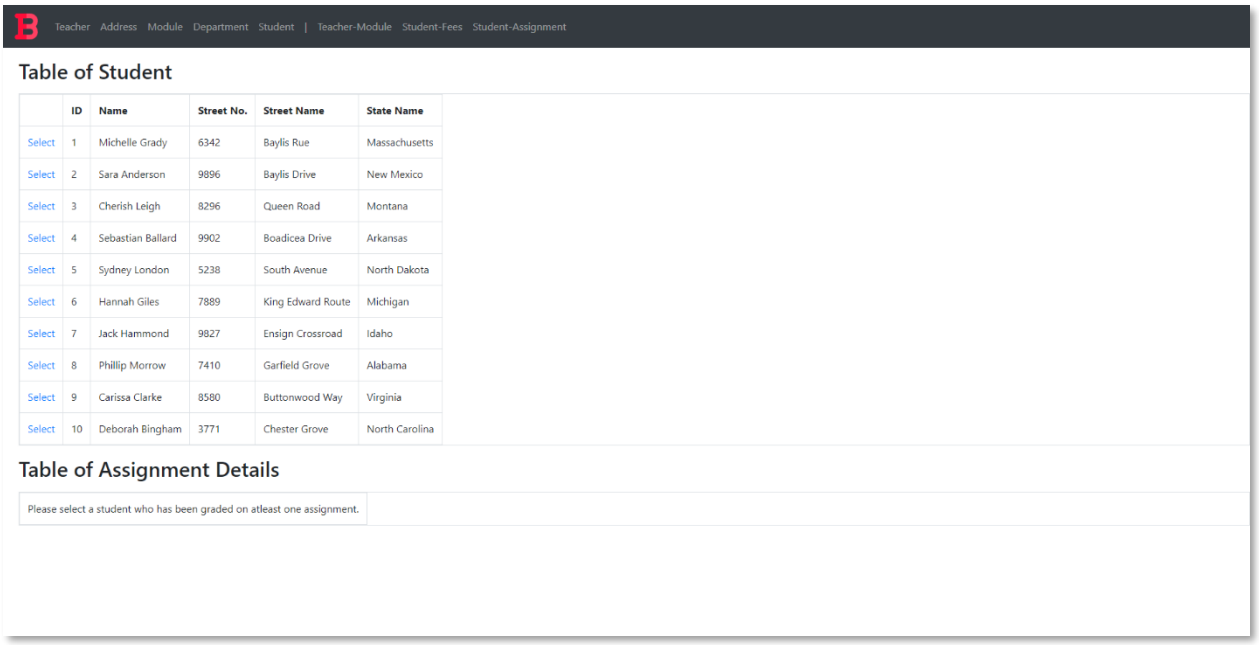
In the webform the query string is written as:

```
$@"SELECT sf.Student_ID ""Student ID"", se.Semester ""Semester"",  d.Department_Name as ""Department Name"", sf.Fee_Amount ""Fee Amount"", sf.Fee_Status ""Fee Status""
                                FROM Student s
                                INNER JOIN Student_Fees sf ON sf.Student_ID = s.Student_ID
                                INNER JOIN Semester se ON se.Semester_ID = sf.Semester_ID
                                INNER JOIN Department d ON d.Department_ID = sf.Department_ID
                                WHERE sf.Student_ID = {studentID}
                                ";

```

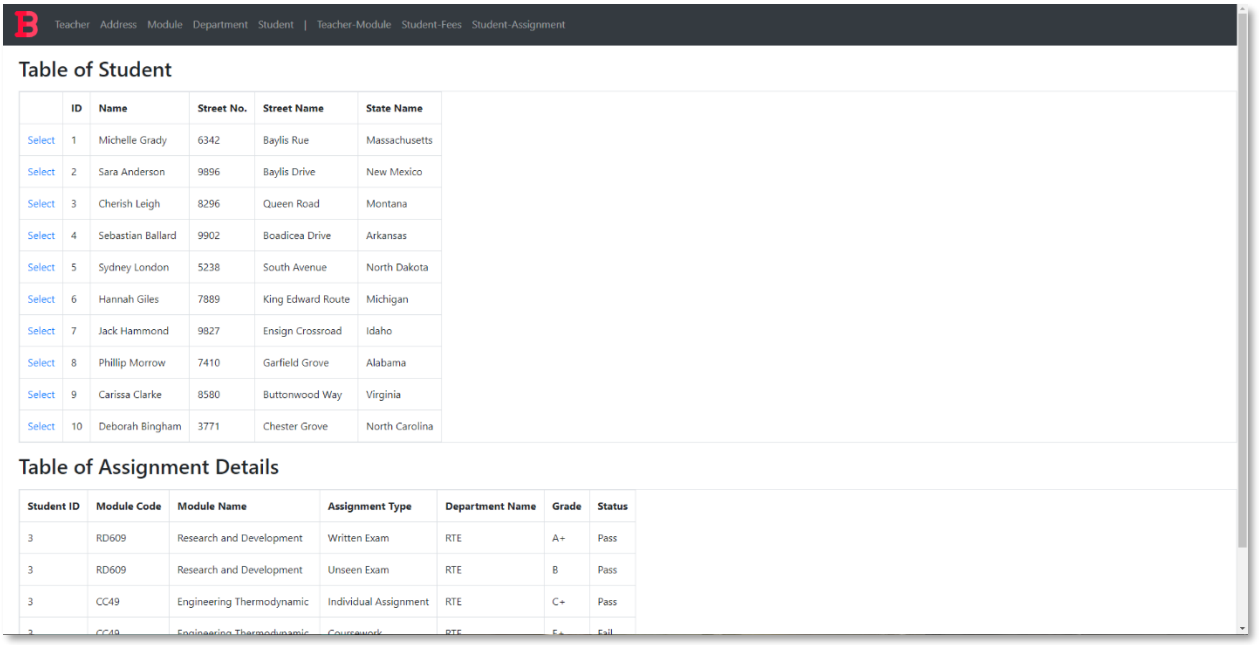
11.5. Student-Assignment Mapping

11.5.1.1. Form



	ID	Name	Street No.	Street Name	State Name
Select	1	Michelle Grady	6342	Baylis Rue	Massachusetts
Select	2	Sara Anderson	9896	Baylis Drive	New Mexico
Select	3	Cherish Leigh	8296	Queen Road	Montana
Select	4	Sebastian Ballard	9902	Boadicea Drive	Arkansas
Select	5	Sydney London	5238	South Avenue	North Dakota
Select	6	Hannah Giles	7889	King Edward Route	Michigan
Select	7	Jack Hammond	9827	Ensign Crossroad	Idaho
Select	8	Phillip Morrow	7410	Garfield Grove	Alabama
Select	9	Carissa Clarke	8580	Buttonwood Way	Virginia
Select	10	Deborah Bingham	3771	Chester Grove	North Carolina

Figure 50: Student-Assignment Mapping Form Before Selection



	ID	Name	Street No.	Street Name	State Name
Select	1	Michelle Grady	6342	Baylis Rue	Massachusetts
Select	2	Sara Anderson	9896	Baylis Drive	New Mexico
Select	3	Cherish Leigh	8296	Queen Road	Montana
Select	4	Sebastian Ballard	9902	Boadicea Drive	Arkansas
Select	5	Sydney London	5238	South Avenue	North Dakota
Select	6	Hannah Giles	7889	King Edward Route	Michigan
Select	7	Jack Hammond	9827	Ensign Crossroad	Idaho
Select	8	Phillip Morrow	7410	Garfield Grove	Alabama
Select	9	Carissa Clarke	8580	Buttonwood Way	Virginia
Select	10	Deborah Bingham	3771	Chester Grove	North Carolina

Student ID	Module Code	Module Name	Assignment Type	Department Name	Grade	Status
3	RD609	Research and Development	Written Exam	RTE	A+	Pass
3	RD609	Research and Development	Unseen Exam	RTE	B	Pass
3	CC49	Engineering Thermodynamic	Individual Assignment	RTE	C+	Pass

Figure 51: Student-Fees Mapping Form After Selection

11.5.1.2. Query

Assuming the selected student has student_id 1, the query is as following:

```
SELECT sa.Student_ID "Student ID", m.Module_Code "Module Code" , m.Module_Name "Module Name", a.Assignment_Type "Assignment Type" , d.Department_Name "Department Name" , g.Grade "Grade", g.Status "Status"
                                FROM Student s
                                INNER JOIN Student_Assignment sa ON sa.Student_ID = s.Student_ID
                                INNER JOIN Grade g ON sa.Grade_ID = g.Grade_ID
                                INNER JOIN Module m ON sa.Module_Code = m.Module_Code
                                INNER JOIN Assignment a ON sa.Assignment_ID = a.Assignment_ID
                                INNER JOIN Department d ON a.Department_ID = d.Department_ID
                                WHERE sa.Student_ID = 1;

```

In the webform the query string is written as:

```
$@"SELECT sa.Student_ID ""Student ID"", m.Module_Code ""Module Code"" , m.Module_Name ""Module Name"", a.Assignment_Type ""Assignment Type"" ,
                                d.Department_Name ""Department Name"" , g.Grade ""Grade"", g.Status ""Status""
                                FROM Student s
                                INNER JOIN Student_Assignment sa ON sa.Student_ID = s.Student_ID
                                INNER JOIN Grade g ON sa.Grade_ID = g.Grade_ID
                                INNER JOIN Module m ON sa.Module_Code = m.Module_Code
                                INNER JOIN Assignment a ON sa.Assignment_ID = a.Assignment_ID
                                INNER JOIN Department d ON a.Department_ID = d.Department_ID
                                WHERE sa.Student_ID = {studentID}
                                ";

```

12. User Manual

If the project is run without changing the port settings, then the url for the project is:
https://localhost:44370/Index.aspx

12.1. Included Content

1. Landing Page
2. Teacher Form
3. Address Form
4. Module Form
5. Teacher-Module Form
6. Student-Fees Form
7. Student-Assignment Form

12.2. Landing Page

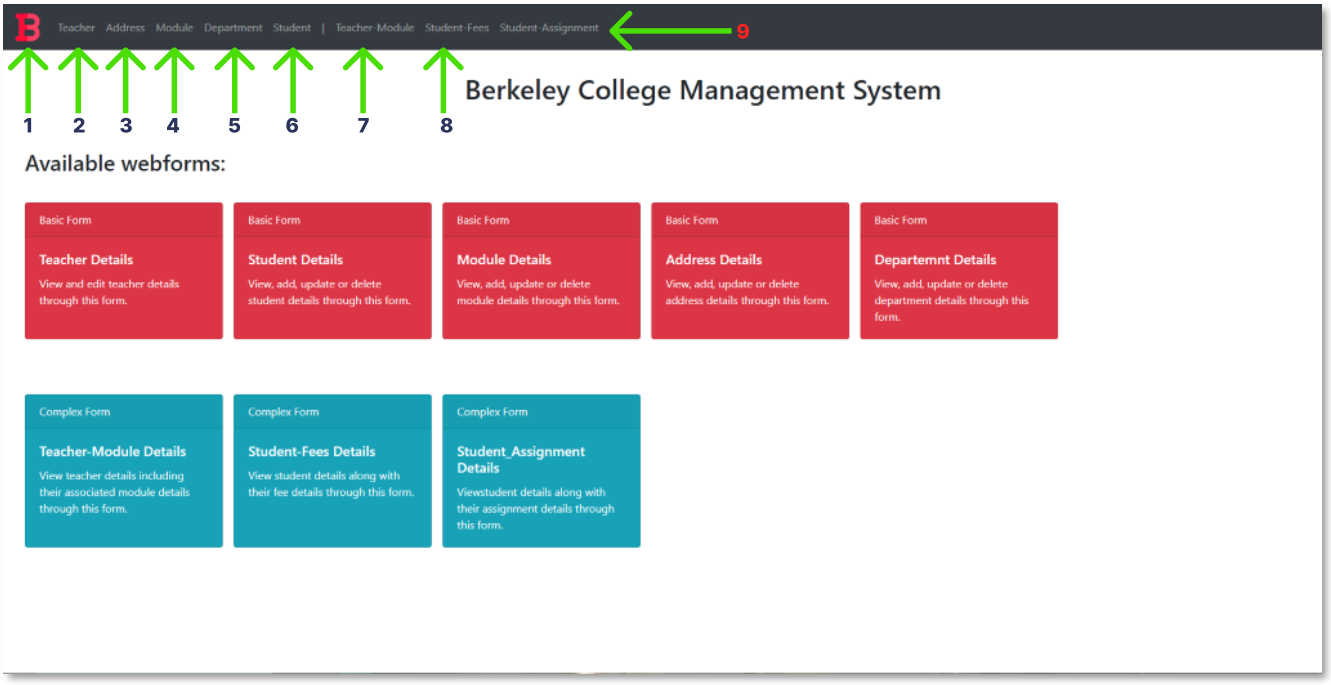


Figure 52: Landing Page

1. The logo can be pressed to navigate to the dashboard
2. The Teacher text in the navbar can be pressed to navigate to the Teacher form
3. The Module text in the navbar can be pressed to navigate to the Module form
4. The Department text in the navbar can be pressed to navigate to the Department form
5. The Student text in the navbar can be pressed to navigate to the Student form
6. The Teacher-Module text in the navbar can be pressed to navigate to the Teacher-Module form
7. The Student-Fees text in the navbar can be pressed to navigate to the Student-Fees form
8. The Student-Assignment text in the navbar can be pressed to navigate to the Student-Assignment form

12.3. Teacher Form

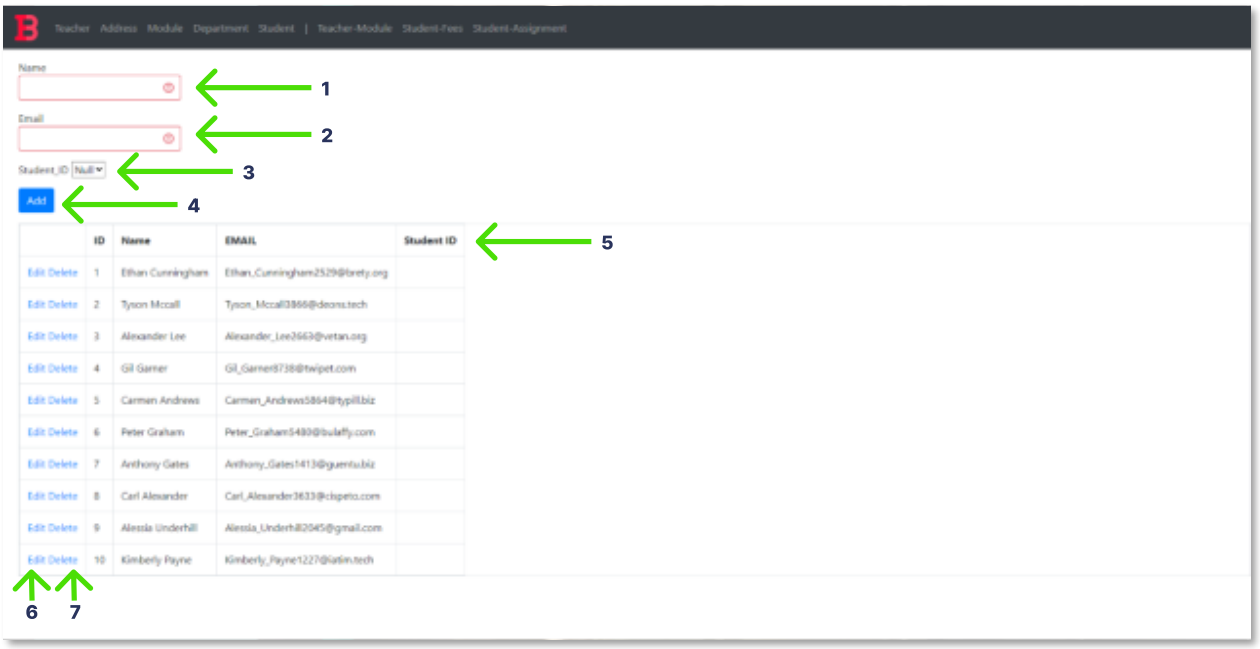


Figure 53: Teacher Form

1. Enter the name of teacher to be added in the name text field. This text field cannot be left empty.
2. Enter the email address of teacher to be added in the email text field. This text field cannot be left empty.
3. If the teacher was a former student, select their student ID from this dropdown. Otherwise, leave it as null.
4. Press the Add button to add data.
5. The teacher details data is displayed in this grid.
6. Press this button to edit data in the grid.
7. Press this button to delete data in the grid.

12.4. Address Form

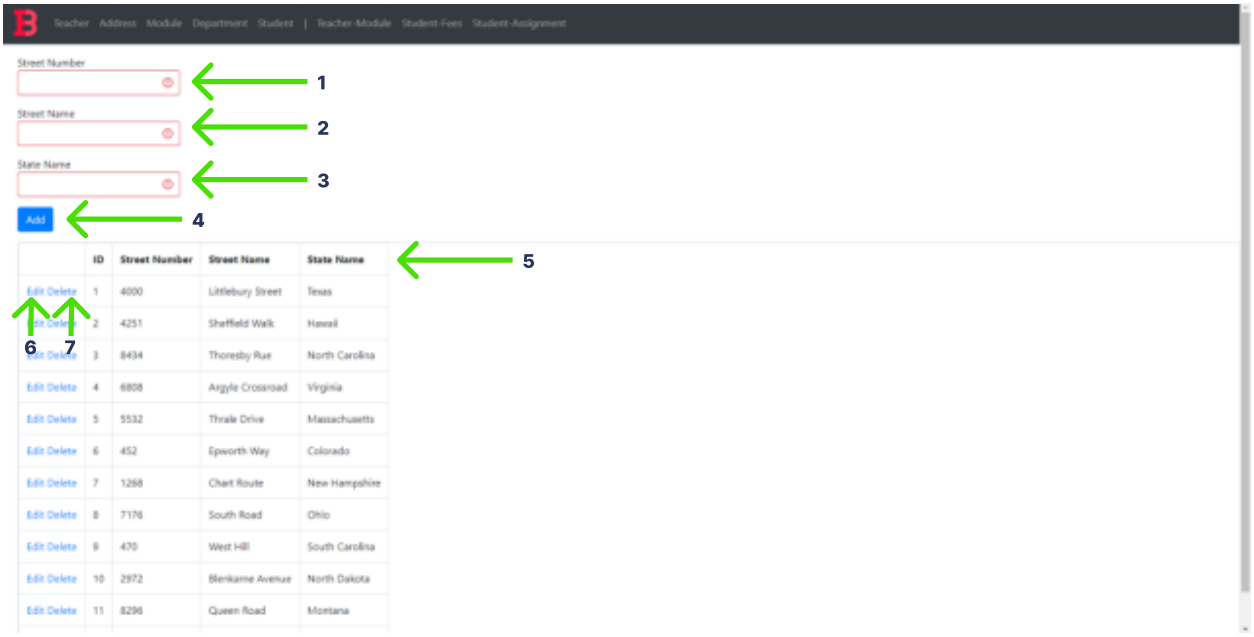


Figure 54: Address Form

1. Enter the Street number of the address in this text field. This text field cannot be left empty.
2. Enter the name of street to be added in this text field. This text field cannot be left empty.
3. Enter the street name of the address to be added in this text field. This text field cannot be left empty.
4. Press the Add button to add data.
5. The teacher details data is displayed in this grid.
6. Press this button to edit data in the grid.
7. Press this button to delete data in the grid.

12.5. Module Form

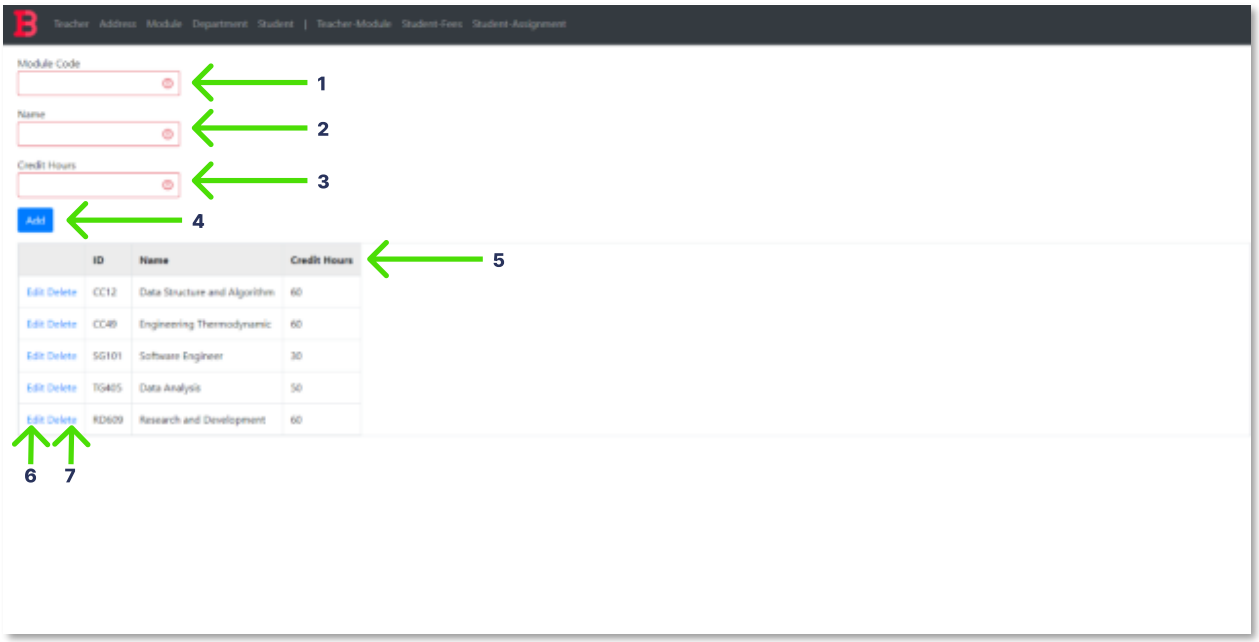


Figure 55: Module Form

1. Enter the code of module to be added in this text field. This text field cannot be left empty.
2. Enter the name of module to be added in this text field. This text field cannot be left empty.
3. Enter the Credit hours of the module in numeric form. This text field cannot be left empty.
4. Press the Add button to add data.
5. The module details data is displayed in this grid.
6. Press this button to edit data in the grid.
7. Press this button to delete data in the grid.

12.6. Department Form

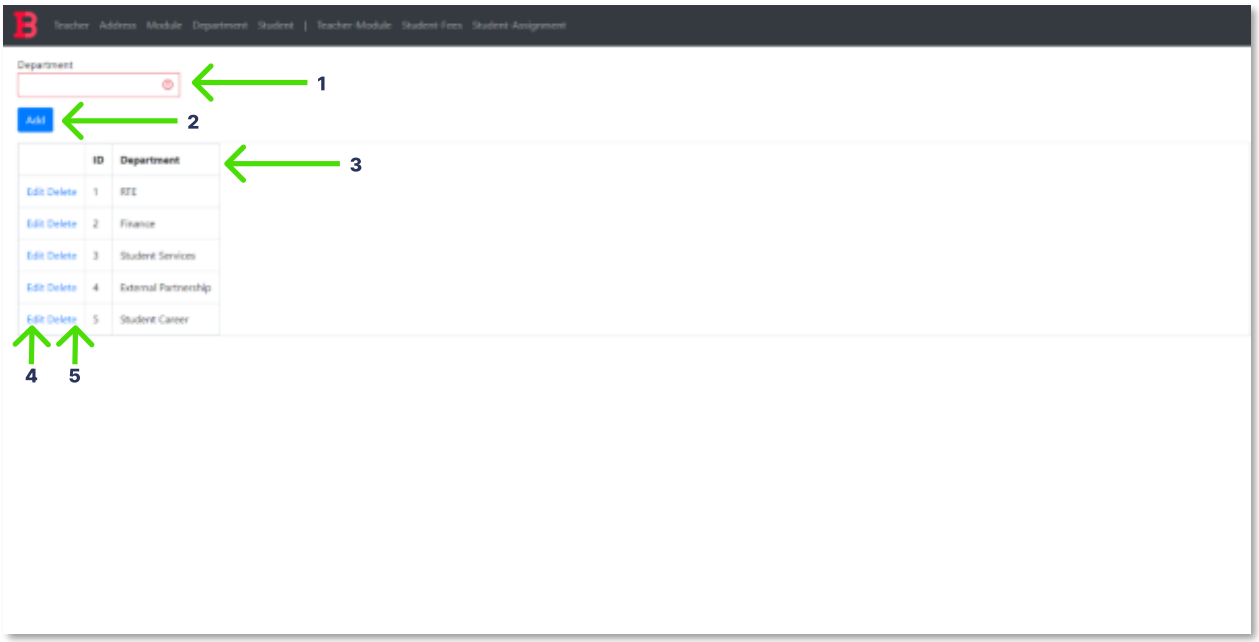


Figure 56: Department Form

1. Enter the name of the department to be added in this text field. This text field cannot be left empty.
2. Press the Add button to add data.
3. The department details data is displayed in this grid.
4. Press this button to edit data in the grid.
5. Press this button to delete data in the grid.

12.7. Student Form

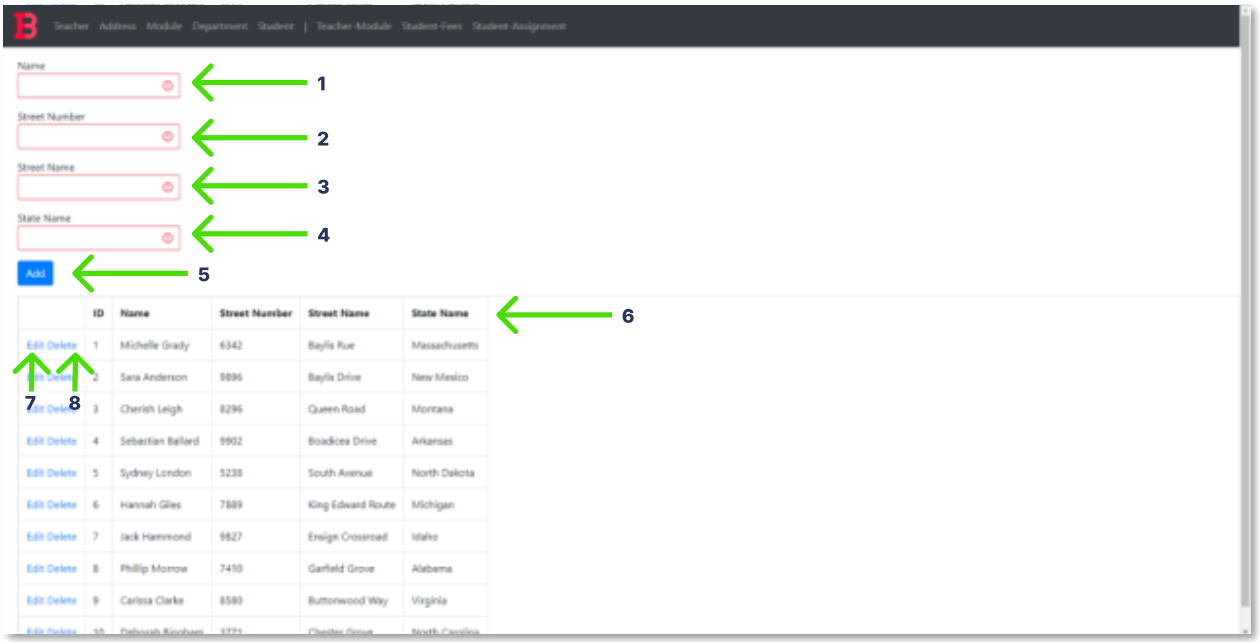


Figure 57: Student Form

1. Enter the name of the student to be added in this text field. This text field cannot be left empty.
2. Enter the street number of the student's address to be added in this text field in numeric form. This text field cannot be left empty.
3. Enter the street name of the student's address to be added in this text field. This text field cannot be left empty.
4. Enter the state name of the student's address to be added in this text field. This text field cannot be left empty.
5. Press the Add button to add data.
6. The student details data is displayed in this grid.
7. Press this button to edit data in the grid.
8. Press this button to delete data in the grid.

12.8. Teacher-Module Form

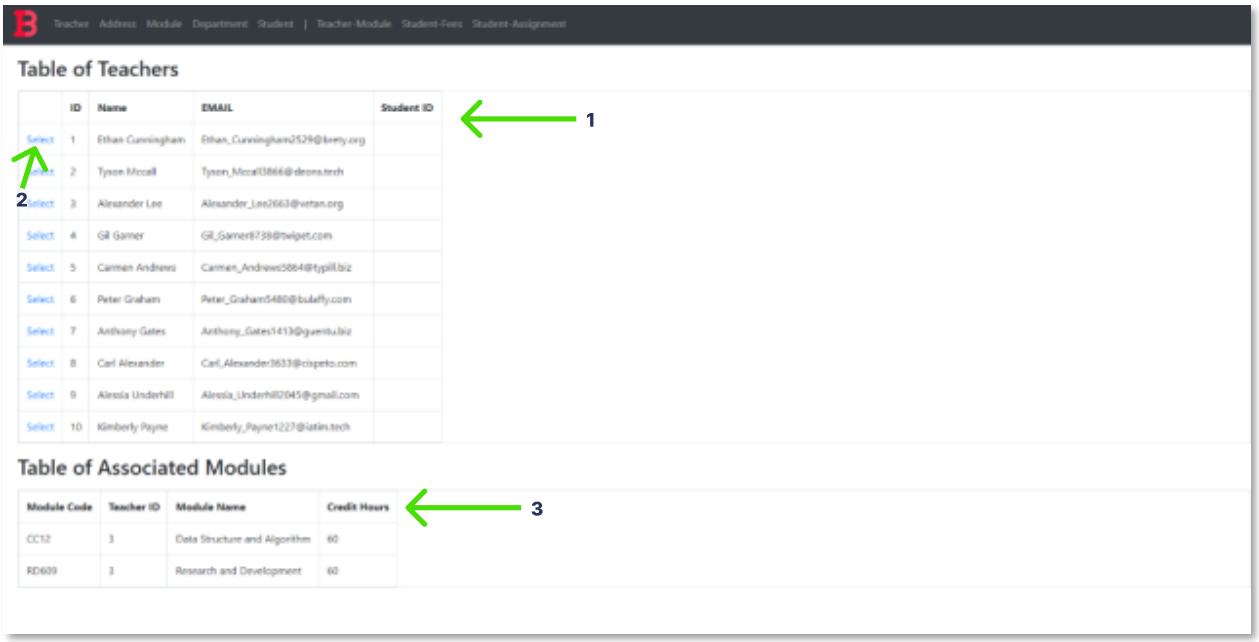


Figure 58: Teacher-Module Form

1. The teacher details data is displayed in this grid.
2. Press this button to select this teacher data and display the modules taught by this teacher.
3. The module details data is displayed in this grid.

12.9. Student-Fees Form

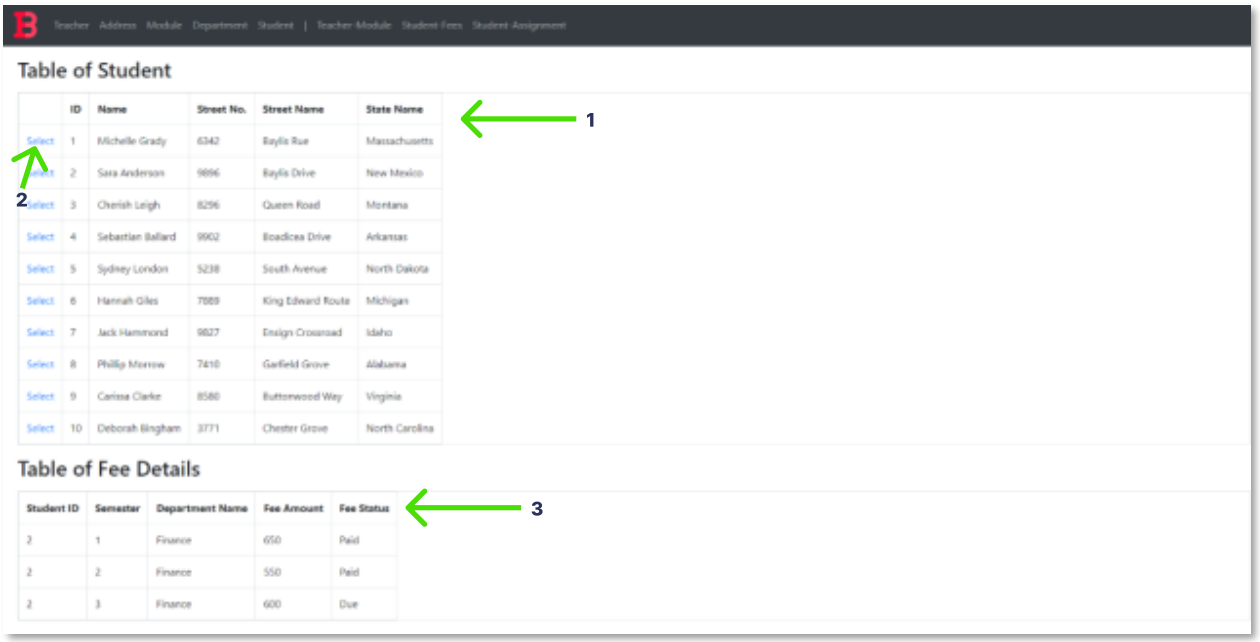


Figure 59: Student_Fees Form

1. The student details data is displayed in this grid.
2. Press this button to select this student data and display the fee details of this student.
3. The fee details data is displayed in this grid.

12.10. Student-Assignment Form

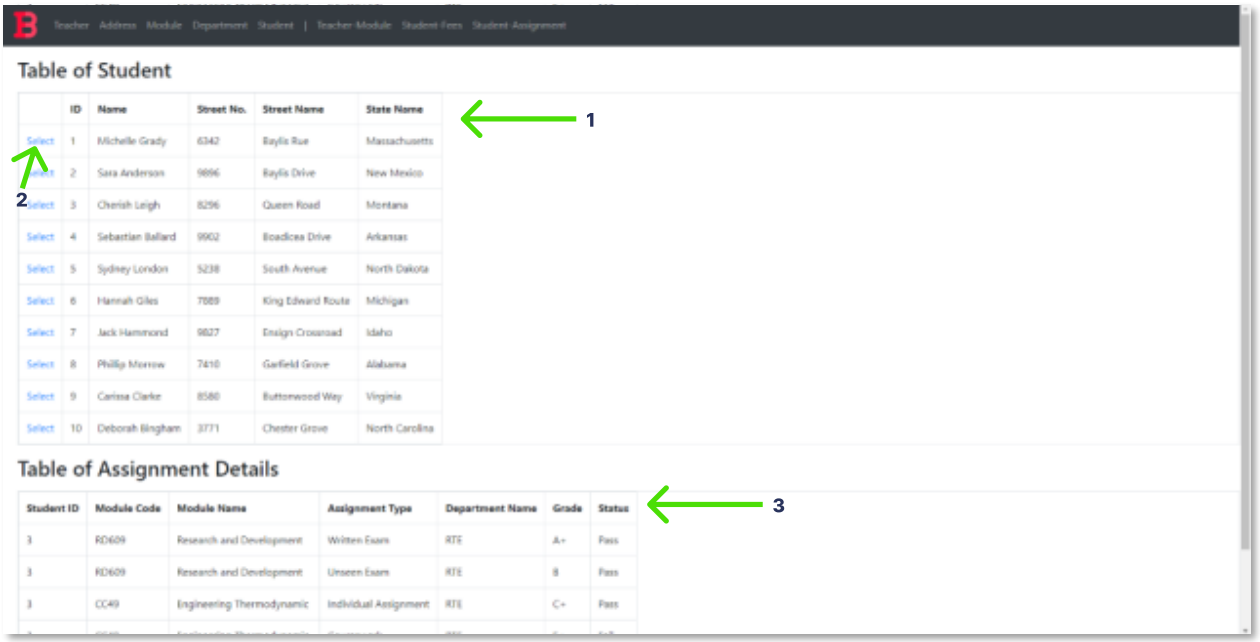


Figure 60: Student_Assignment Form

1. The student details data is displayed in this grid.
2. Press this button to select this student data and display the assignment details of this student.
3. The assignment details data is displayed in this grid.

13. Testing

13.1. Basic Forms

13.1.1. Teacher Form

13.1.1.1. Test 1: View

Test Condition	Expected Outcome	Obtained Outcome
Load the teacher form and check if data is loaded in the grid.	The data is loaded on page load.	The data was loaded on page load.

Table 16: Teacher Test 1

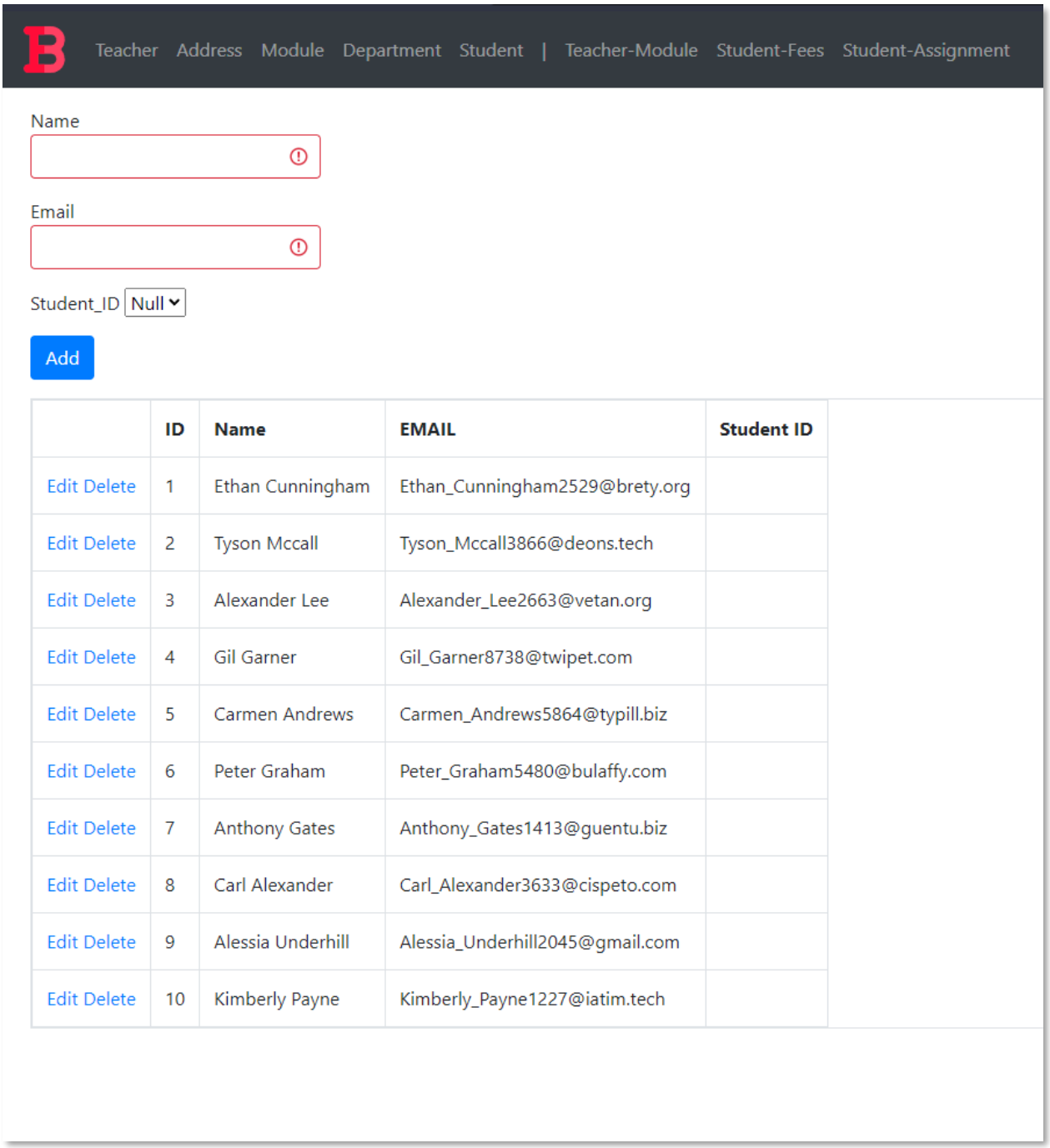


Figure 61: Teacher Test 1 evidence

13.1.1.2. Test 2: Add

Test Condition	Expected Outcome	Obtained Outcome
Fill the teacher form appropriately and press the add button.	The data is added, and the grid is refreshed with the new data.	The data was added without any errors and the grid was refreshed with the new data.

Table 17: Teacher Test 2

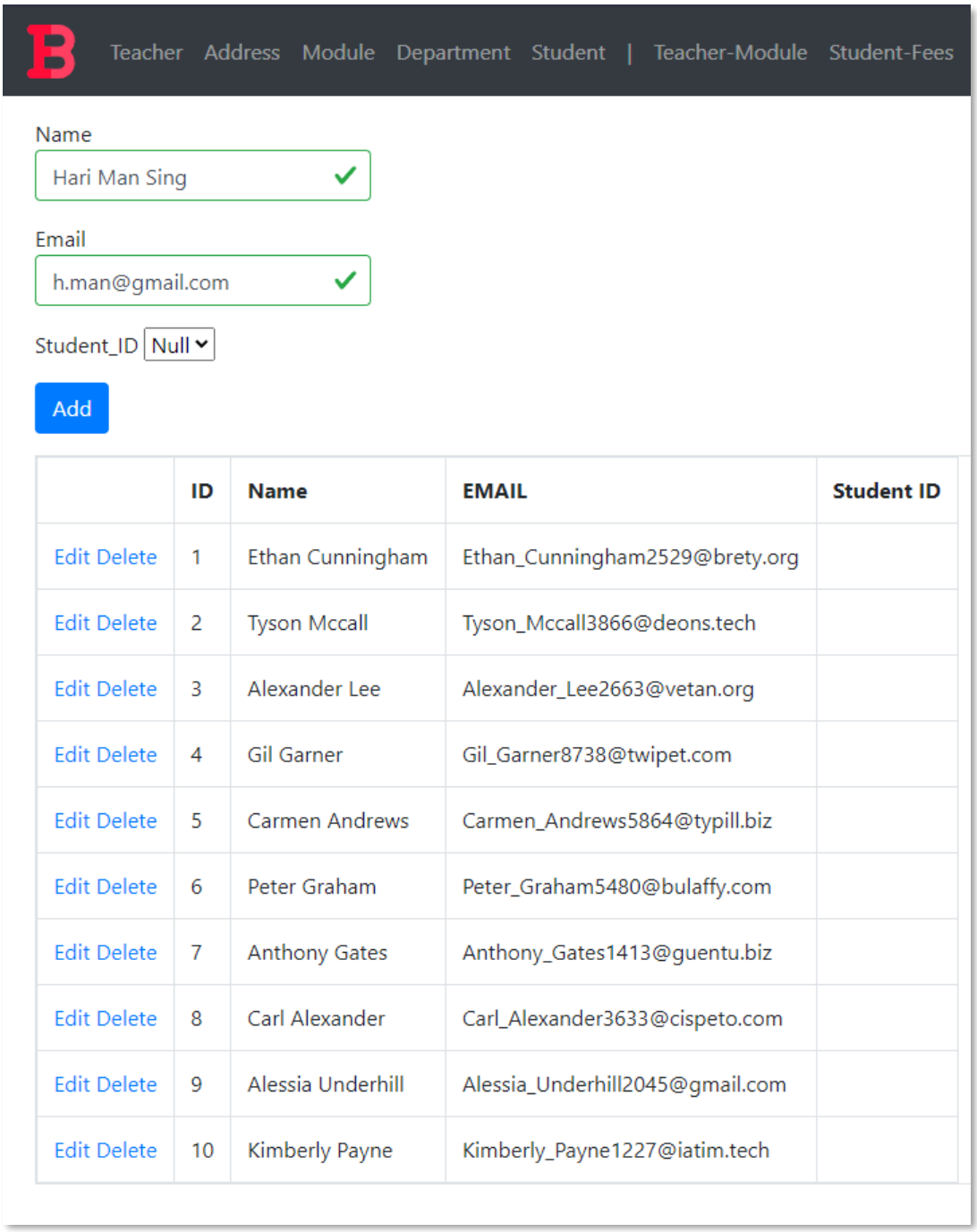


Figure 62: Teacher Test 2 Added Data

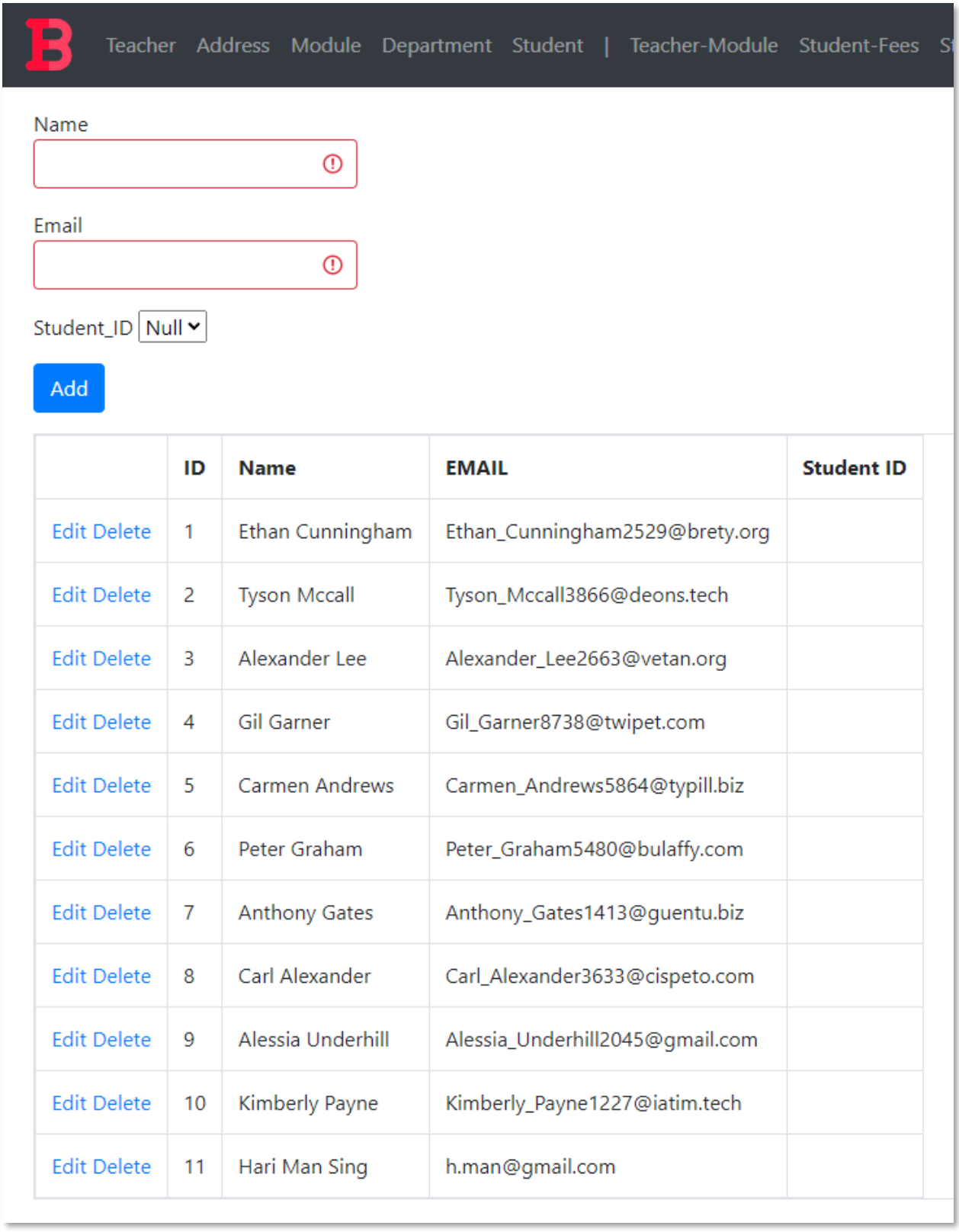


Figure 63: Teacher Test 2 Addition Result

13.1.1.3. Test 3: Edit

Test Condition	Expected Outcome	Obtained Outcome
Click the edit button	• The add button disappears and an update button appears. • The data in the row gets filled in the available textboxes.	• The add button disappeared and an update button appeared. • The data in the field got filled in the available textboxes.
Click the update button	• The update button disappears and the add button reappears • The data in the grid is refreshed with the edited data.	• The update button disappears and the add button reappears • The data in the grid is refreshed with the edited data.

Table 18: Teacher Test 3

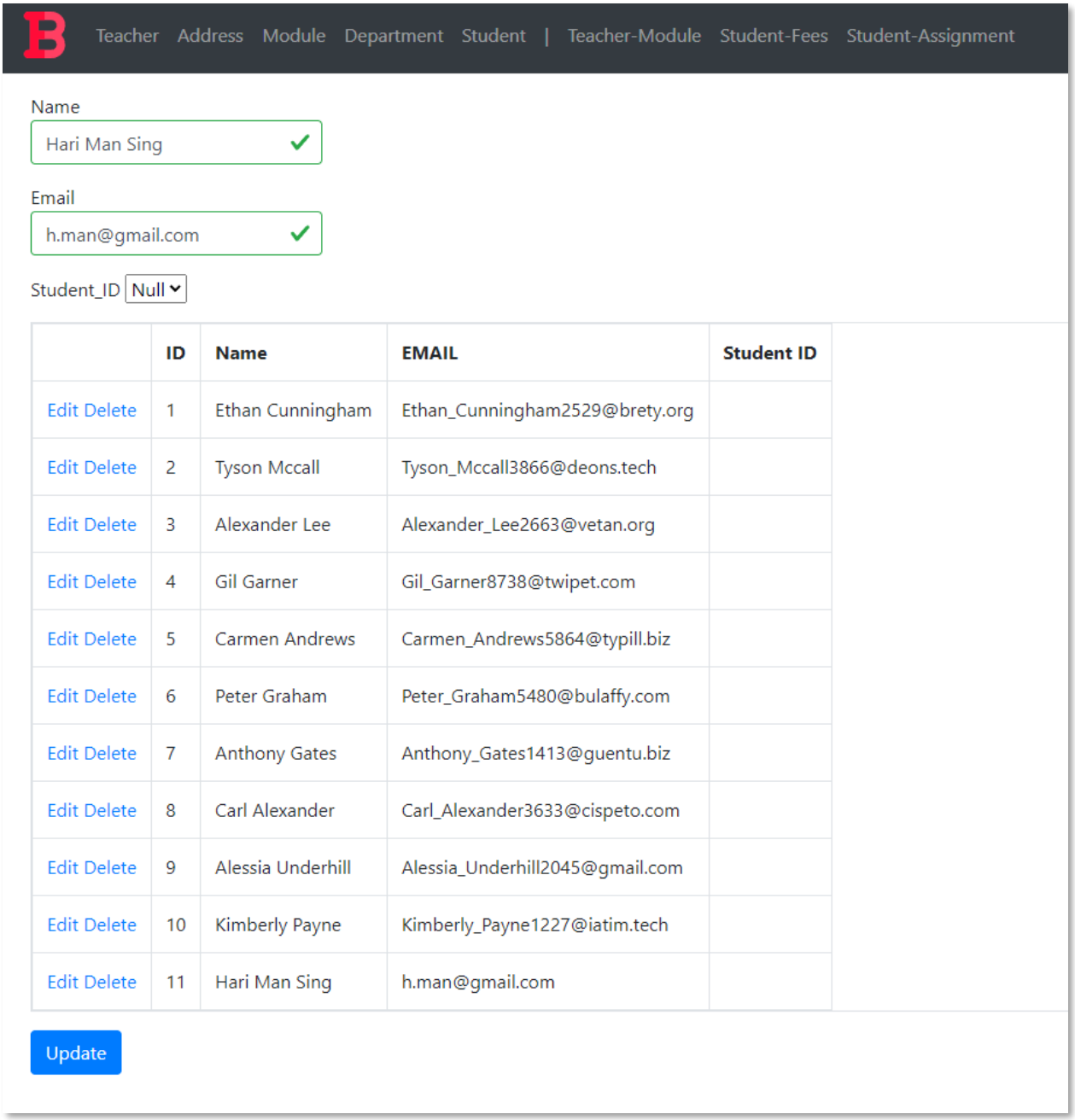


Figure 64: Teacher Test 3 condition 1

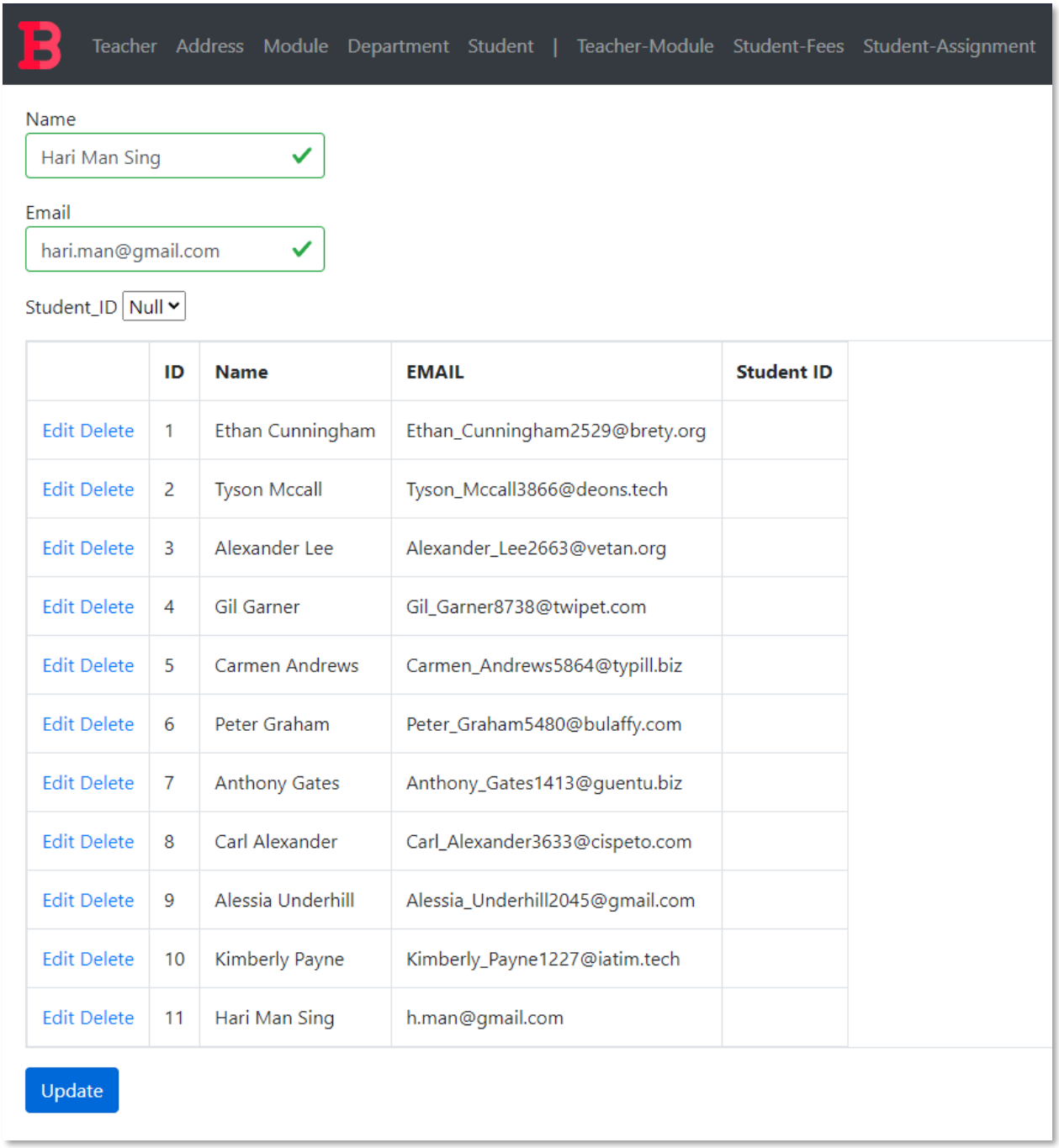


Figure 65: Edited value in Email Textbox

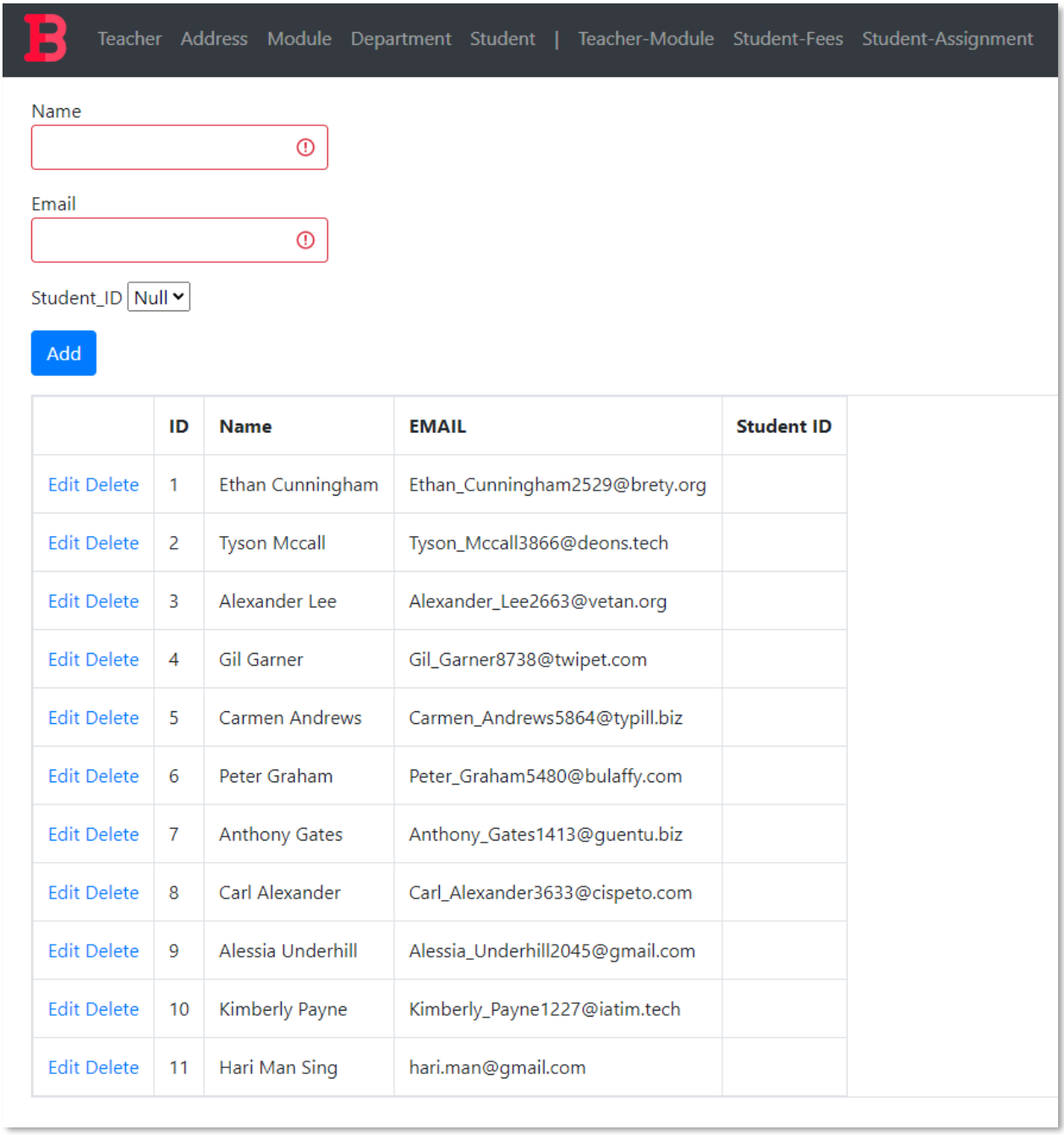


Figure 66: Teacher Test 3 Condition 2

13.1.1.4. Test 4: Delete

Test Condition	Expected Outcome	Obtained Outcome
Click the delete button.	The grid is refreshed with the data row deleted.	The grid was refreshed with the data row deleted.

Table 19: Teacher Test 4

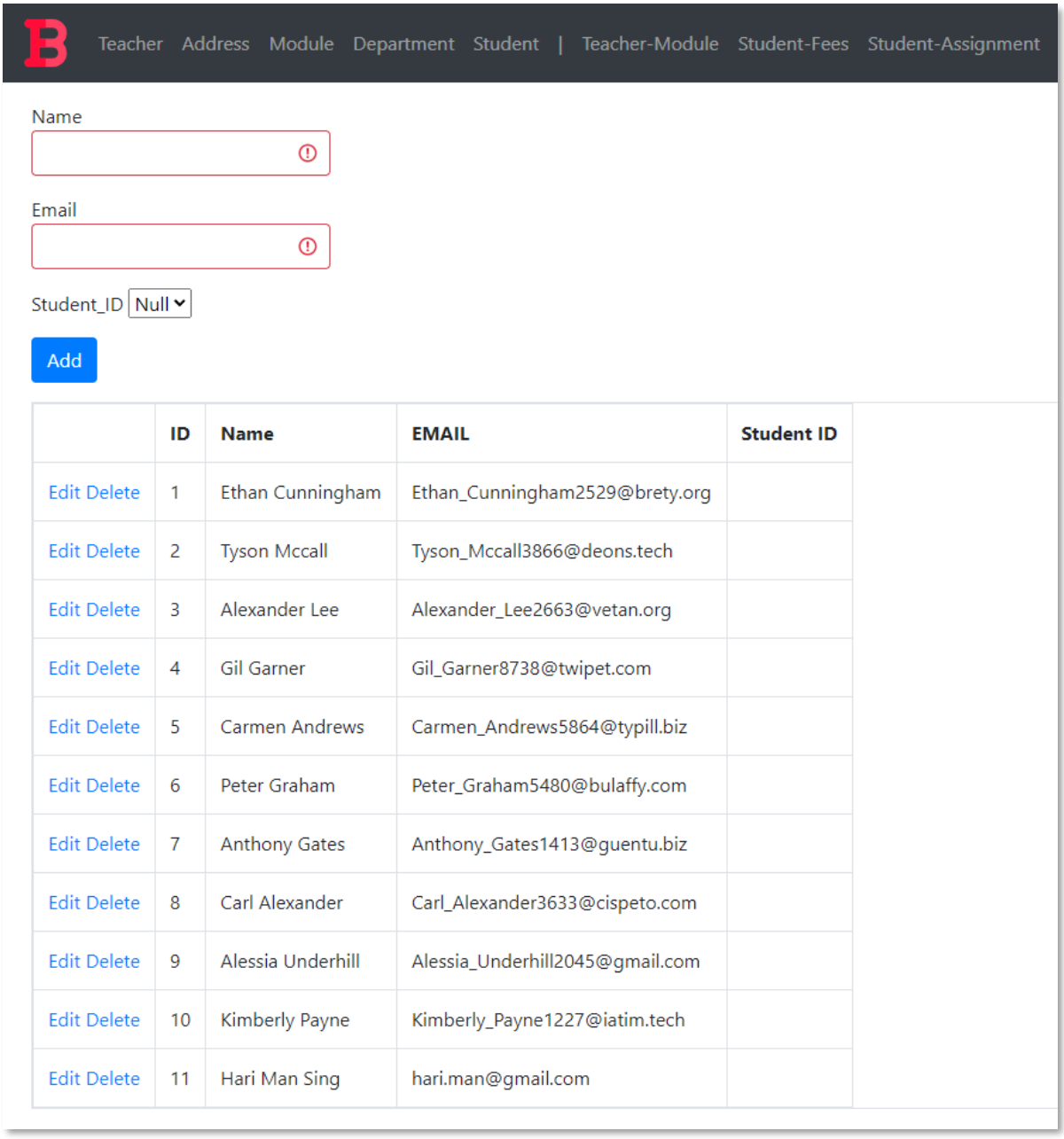


Figure 67: Teacher Grid Pre-Deletion

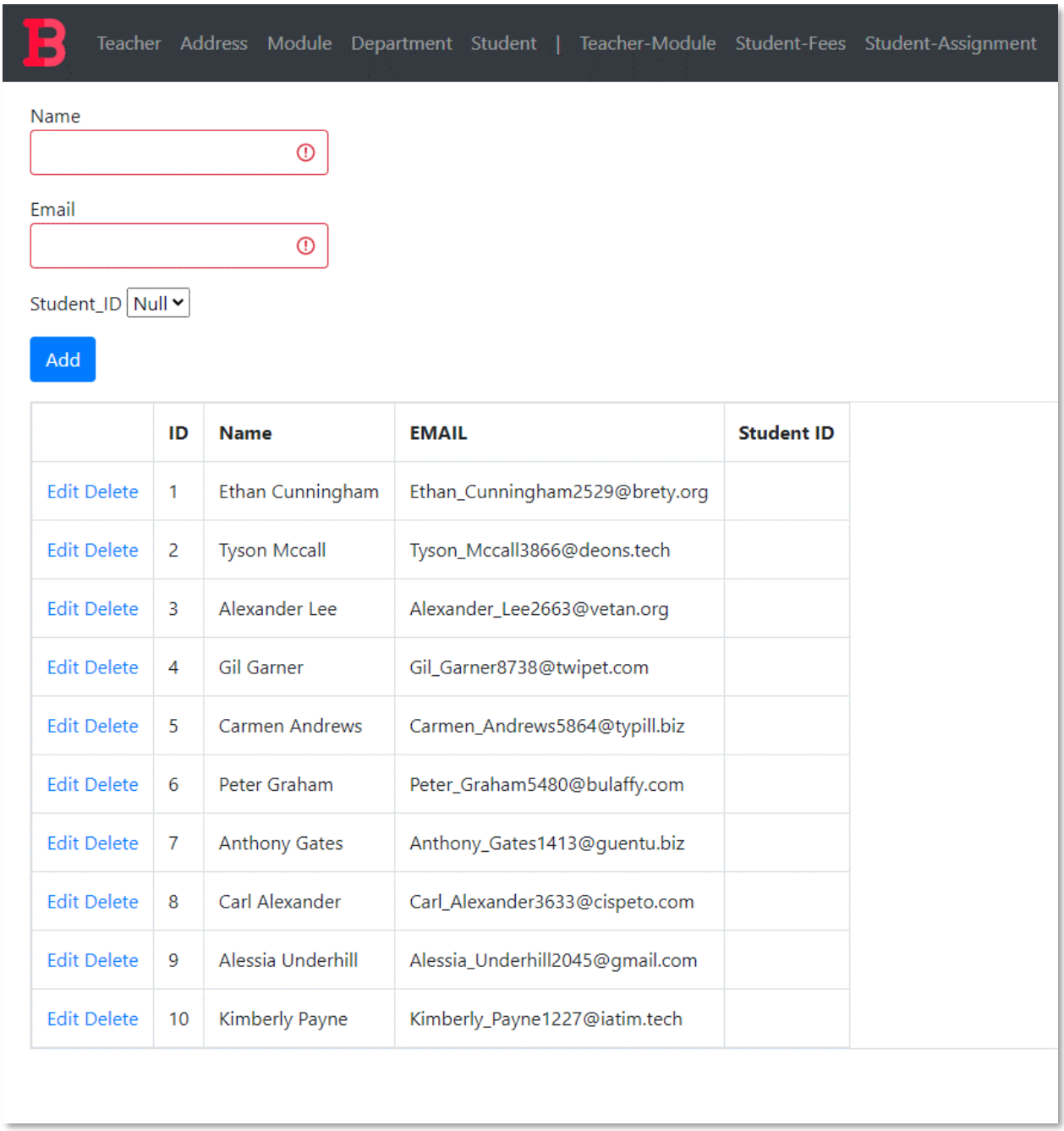


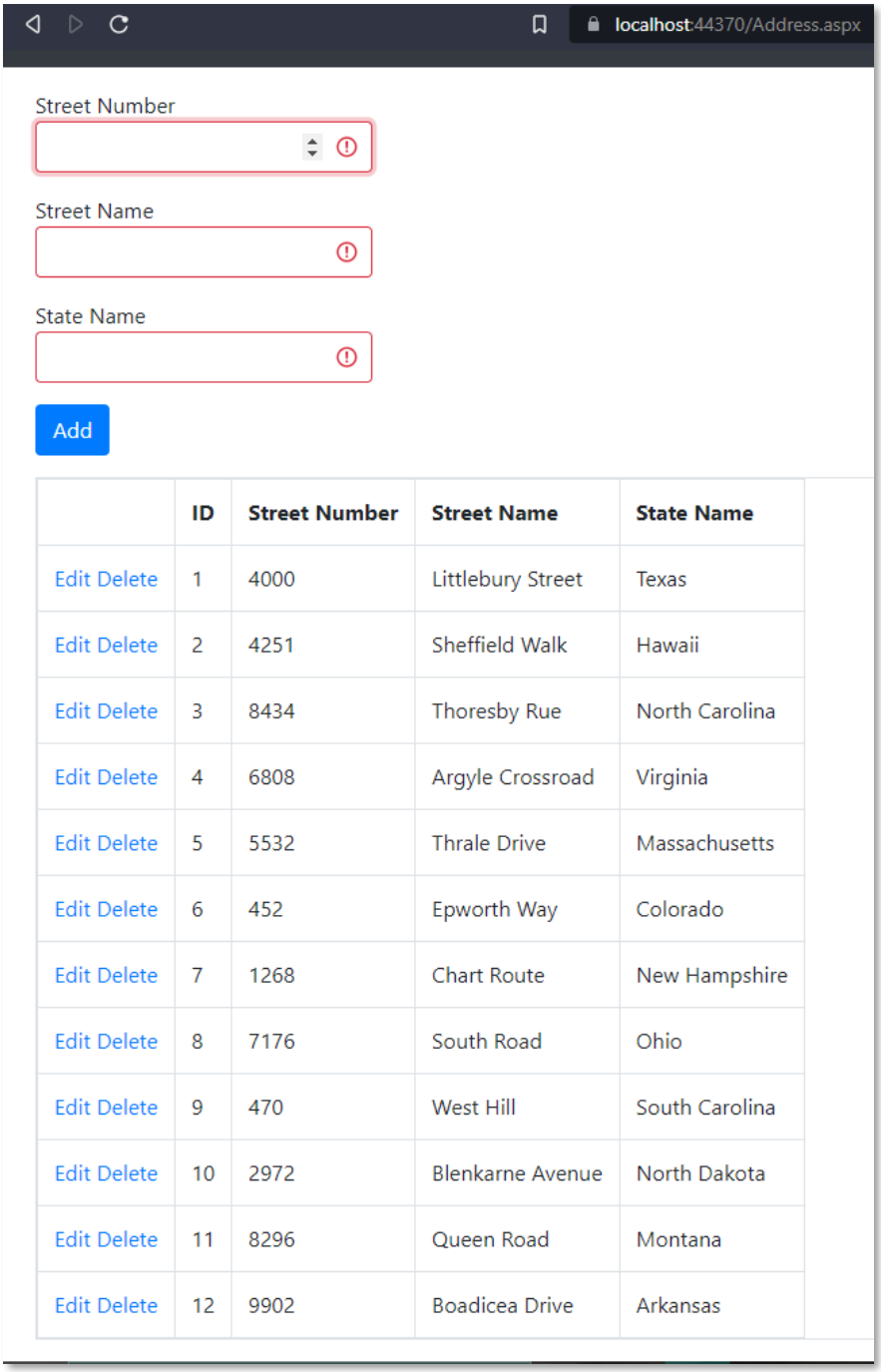
Figure 68: Teacher Grid Post-Deletion

13.1.2. Address Form

13.1.2.1. Test 1: View

Test Condition	Expected Outcome	Obtained Outcome
Load the address form and check if data is loaded in the grid.	The data is loaded on page load.	The data was loaded on page load.

Table 20: Address Test 1



	ID	Street Number	Street Name	State Name
Edit Delete	1	4000	Littlebury Street	Texas
Edit Delete	2	4251	Sheffield Walk	Hawaii
Edit Delete	3	8434	Thoresby Rue	North Carolina
Edit Delete	4	6808	Argyle Crossroad	Virginia
Edit Delete	5	5532	Thrale Drive	Massachusetts
Edit Delete	6	452	Epworth Way	Colorado
Edit Delete	7	1268	Chart Route	New Hampshire
Edit Delete	8	7176	South Road	Ohio
Edit Delete	9	470	West Hill	South Carolina
Edit Delete	10	2972	Blenkarne Avenue	North Dakota
Edit Delete	11	8296	Queen Road	Montana
Edit Delete	12	9902	Boadicea Drive	Arkansas

Figure 69: Address Test 1 evidence

13.1.2.2. Test 2: Add

Test Condition	Expected Outcome	Obtained Outcome
Fill the address form appropriately and press the add button.	The data is added, and the grid is refreshed with the new data.	The data was added without any errors and the grid was refreshed with the new data.

Table 21: Address Test 2

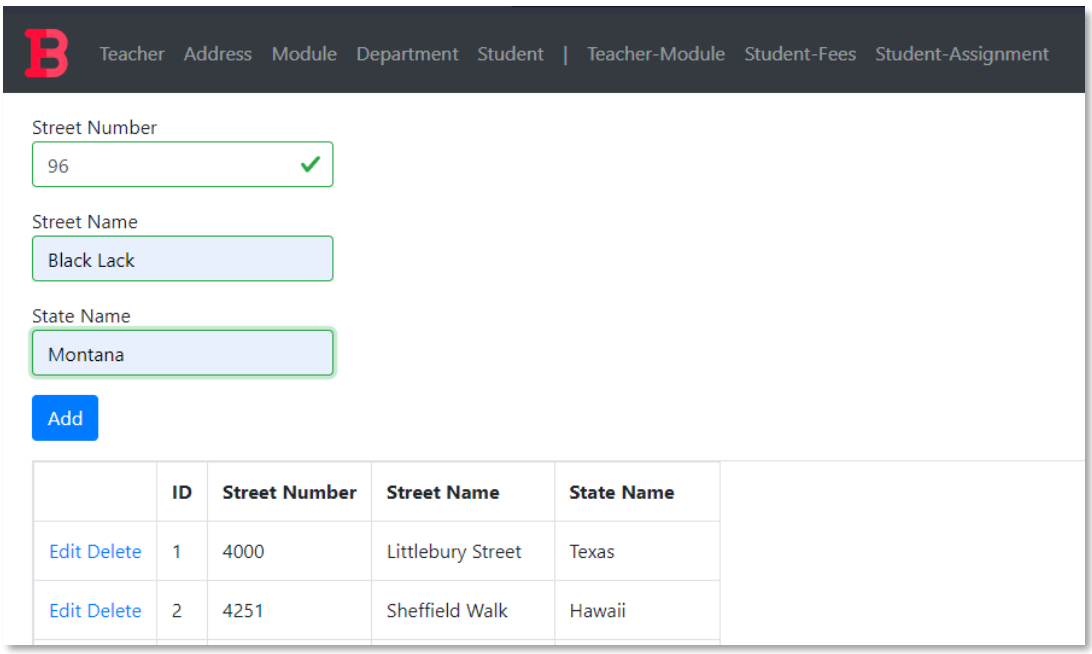
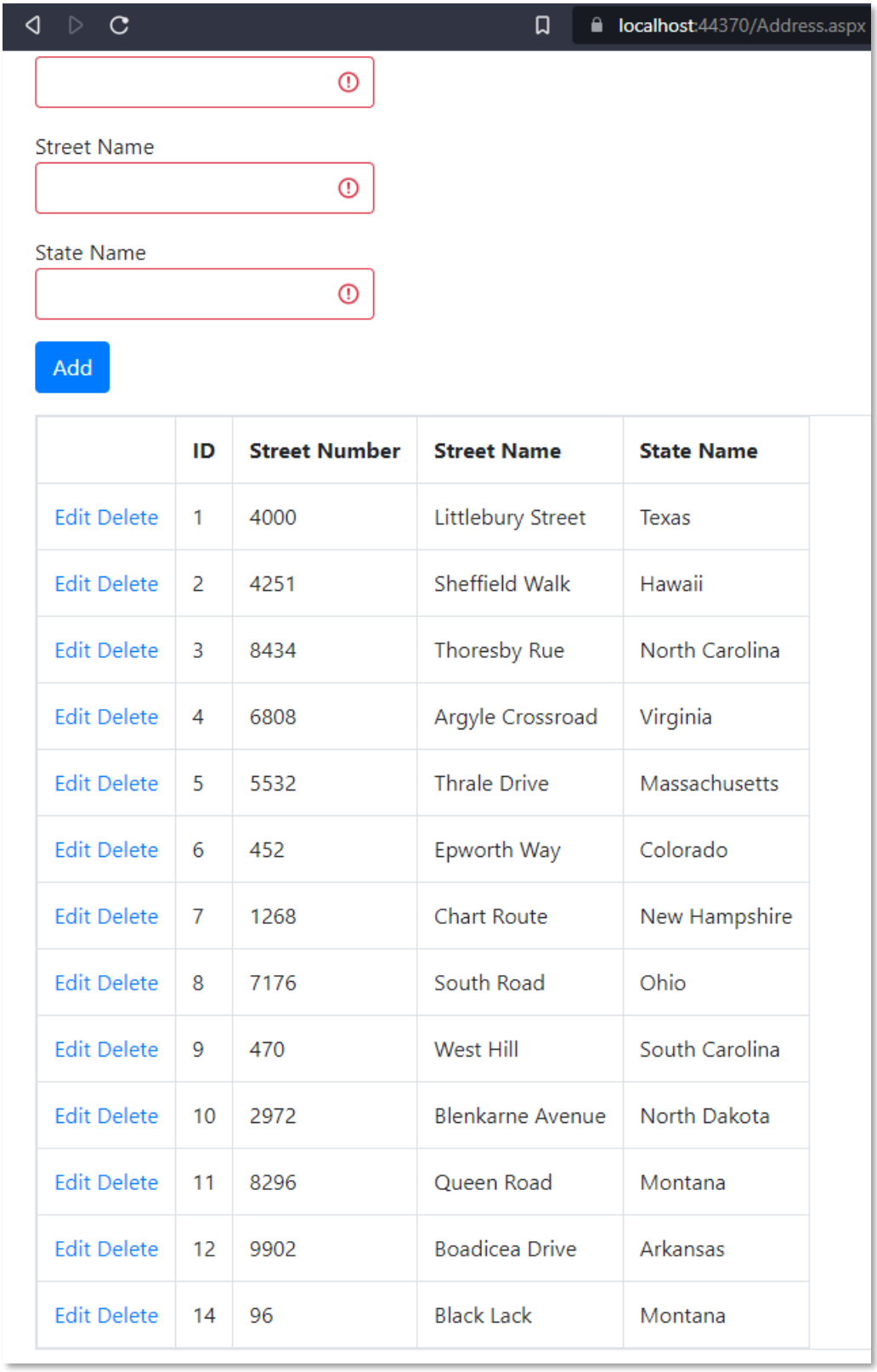


Figure 70: Address Test 2 Data to be Added



	ID	Street Number	Street Name	State Name
Edit Delete	1	4000	Littlebury Street	Texas
Edit Delete	2	4251	Sheffield Walk	Hawaii
Edit Delete	3	8434	Thoresby Rue	North Carolina
Edit Delete	4	6808	Argyle Crossroad	Virginia
Edit Delete	5	5532	Thrale Drive	Massachusetts
Edit Delete	6	452	Epworth Way	Colorado
Edit Delete	7	1268	Chart Route	New Hampshire
Edit Delete	8	7176	South Road	Ohio
Edit Delete	9	470	West Hill	South Carolina
Edit Delete	10	2972	Blenkarne Avenue	North Dakota
Edit Delete	11	8296	Queen Road	Montana
Edit Delete	12	9902	Boadicea Drive	Arkansas
Edit Delete	14	96	Black Lack	Montana

Figure 71: Address Test 2 Addition Result

13.1.2.3. Test 3: Edit

Test Condition	Expected Outcome	Obtained Outcome
Click the edit button	• The add button disappears and an update button appears. • The data in the row gets filled in the available textboxes.	• The add button disappeared and an update button appeared. • The data in the field got filled in the available textboxes. However, the street number got loaded into all textboxes

Table 22: Address Test 3 Fail Case

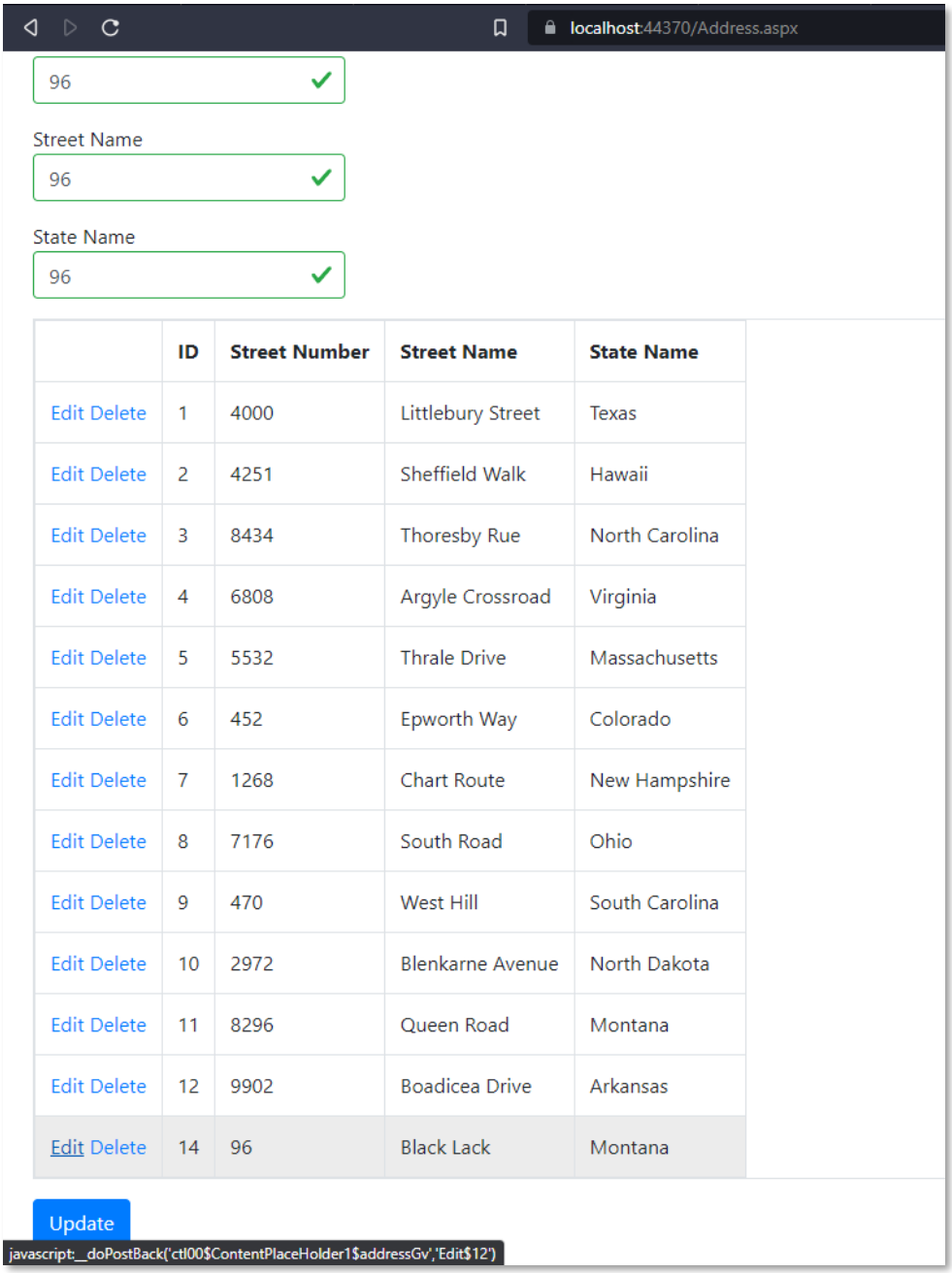


Figure 72: Address Test 3 condition 1 fail case

13.1.2.3.1. What caused the issue and how it was rectified:

```
protected void OnRowEditing(object sender, GridViewEditEventArgs e)
{
    IDTxt.Text = this.addressGv.Rows[e.NewEditIndex].Cells[1].Text;
    streetNoTxt.Text = this.addressGv.Rows[e.NewEditIndex].Cells[2].Text.ToString();
    streetNameTxt.Text = this.addressGv.Rows[e.NewEditIndex].Cells[2].Text.ToString();
    stateNameTxt.Text = this.addressGv.Rows[e.NewEditIndex].Cells[2].Text.ToString();
    updateBtn.Visible = true;
    submitBtn.Visible = false;

}
```

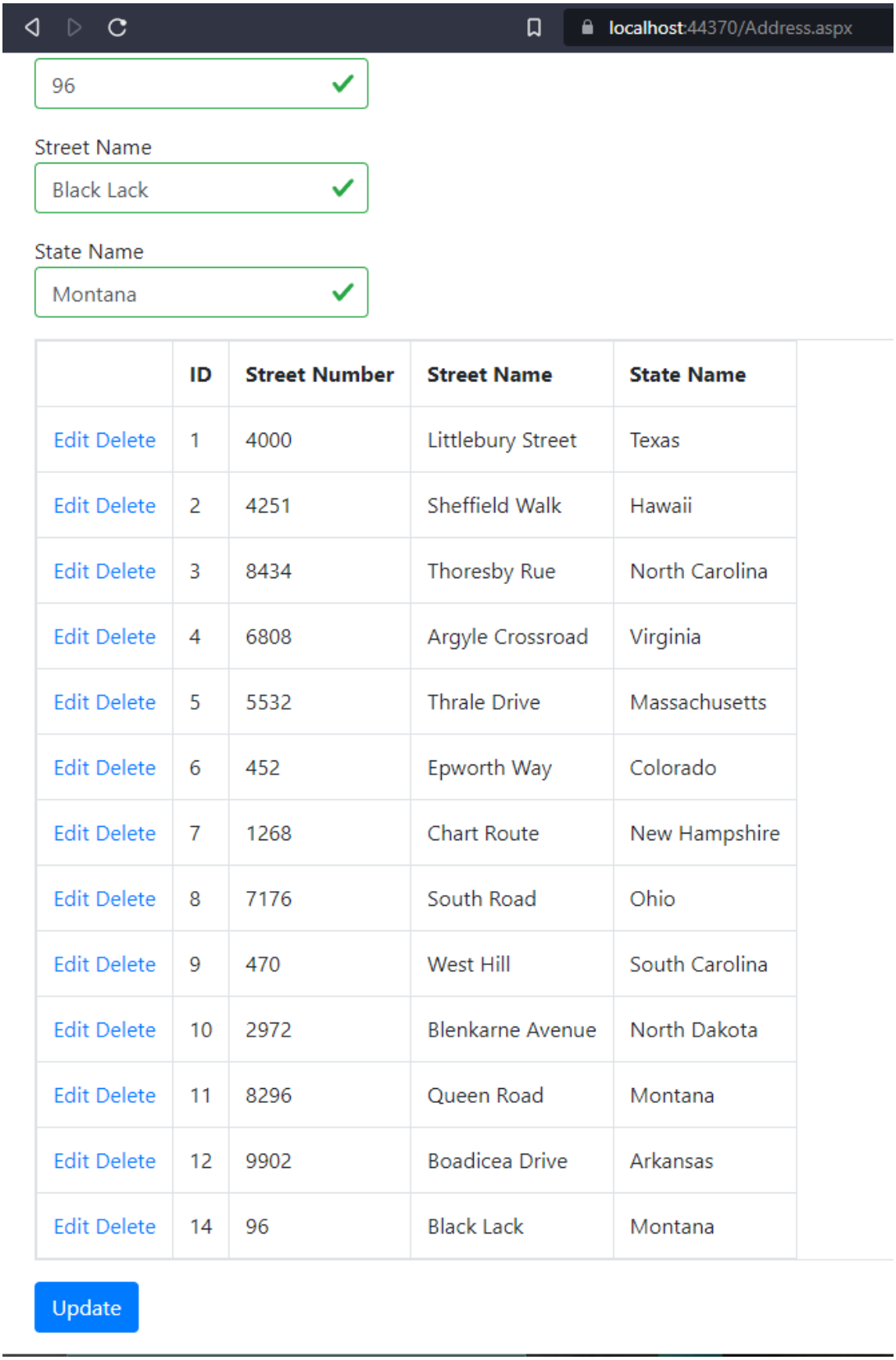
Figure 73: Issue causing the fail case

The textboxes were updated with the grid row cell containing the street number value. Once they were changed, relevant values got added to the respective textboxes.

```
protected void OnRowEditing(object sender, GridViewEditEventArgs e)
{
    IDTxt.Text = this.addressGv.Rows[e.NewEditIndex].Cells[1].Text;
    streetNoTxt.Text = this.addressGv.Rows[e.NewEditIndex].Cells[2].Text.ToString();
    streetNameTxt.Text = this.addressGv.Rows[e.NewEditIndex].Cells[3].Text.ToString();
    stateNameTxt.Text = this.addressGv.Rows[e.NewEditIndex].Cells[4].Text.ToString();
    updateBtn.Visible = true;
    submitBtn.Visible = false;

}
```

Figure 74: Rectification of the Issue causing the fail case



	ID	Street Number	Street Name	State Name
Edit Delete	1	4000	Littlebury Street	Texas
Edit Delete	2	4251	Sheffield Walk	Hawaii
Edit Delete	3	8434	Thoresby Rue	North Carolina
Edit Delete	4	6808	Argyle Crossroad	Virginia
Edit Delete	5	5532	Thrale Drive	Massachusetts
Edit Delete	6	452	Epworth Way	Colorado
Edit Delete	7	1268	Chart Route	New Hampshire
Edit Delete	8	7176	South Road	Ohio
Edit Delete	9	470	West Hill	South Carolina
Edit Delete	10	2972	Blenkarne Avenue	North Dakota
Edit Delete	11	8296	Queen Road	Montana
Edit Delete	12	9902	Boadicea Drive	Arkansas
Edit Delete	14	96	Black Lack	Montana

Figure 75: Test 3 Condition 1 post rectification

Test Condition	Expected Outcome	Obtained Outcome
Click the update button	• The update button disappears and the add button reappears • The data in the grid is refreshed with the edited data.	• The update button disappears and the add button reappears • The data in the grid is refreshed with the edited data.

Table 23: Address Test 3 condition 2

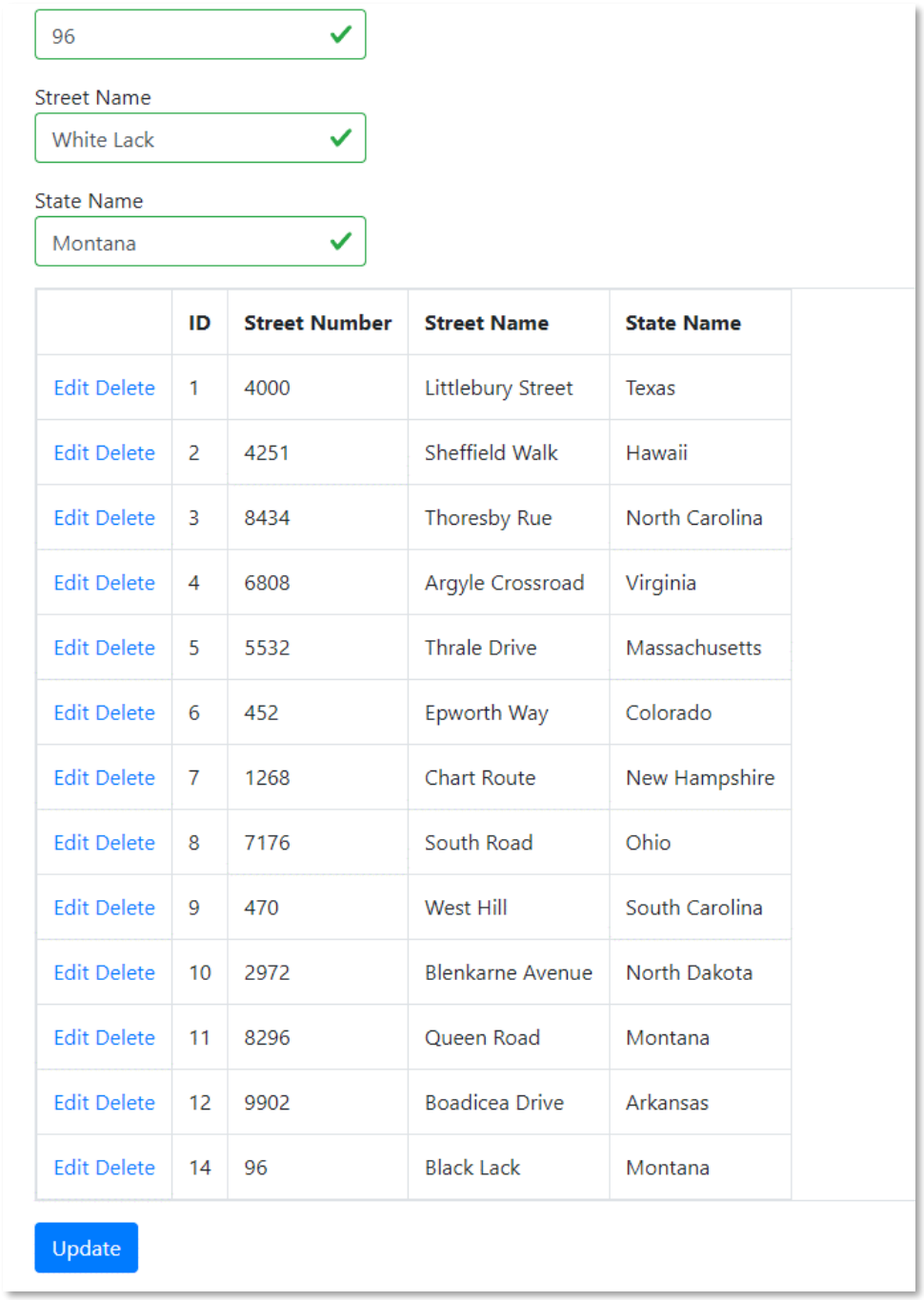
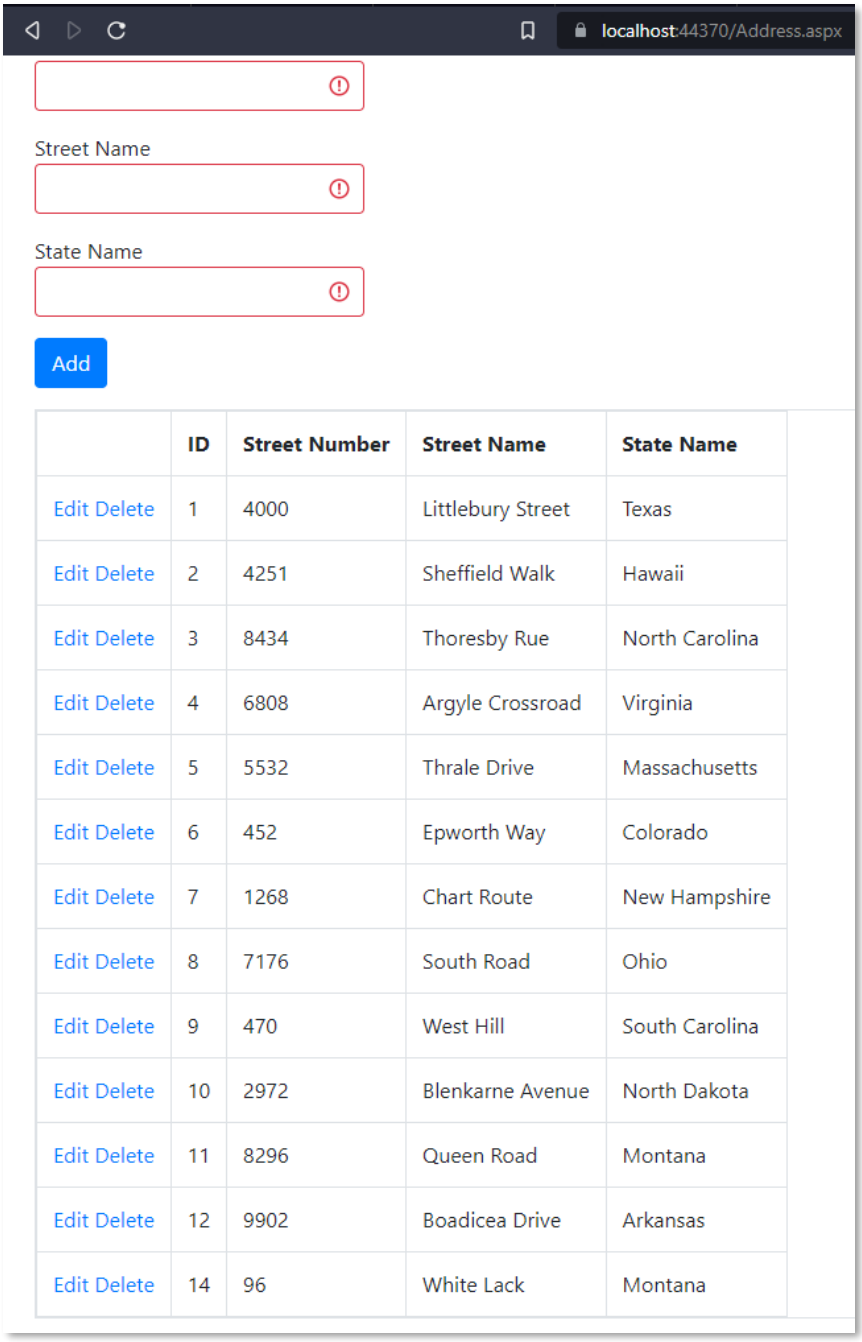


Figure 76: Edited value in Street Name Textbox



	ID	Street Number	Street Name	State Name
Edit Delete	1	4000	Littlebury Street	Texas
Edit Delete	2	4251	Sheffield Walk	Hawaii
Edit Delete	3	8434	Thoresby Rue	North Carolina
Edit Delete	4	6808	Argyle Crossroad	Virginia
Edit Delete	5	5532	Thrale Drive	Massachusetts
Edit Delete	6	452	Epworth Way	Colorado
Edit Delete	7	1268	Chart Route	New Hampshire
Edit Delete	8	7176	South Road	Ohio
Edit Delete	9	470	West Hill	South Carolina
Edit Delete	10	2972	Blenkarne Avenue	North Dakota
Edit Delete	11	8296	Queen Road	Montana
Edit Delete	12	9902	Boadicea Drive	Arkansas
Edit Delete	14	96	White Lack	Montana

Figure 77: Address Test 3 Condition 2

13.1.2.4. Test 4: Delete

Test Condition	Expected Outcome	Obtained Outcome
Click the delete button.	The grid is refreshed with the data row deleted.	The grid was refreshed with the data row deleted.

Table 24: Address Test 4

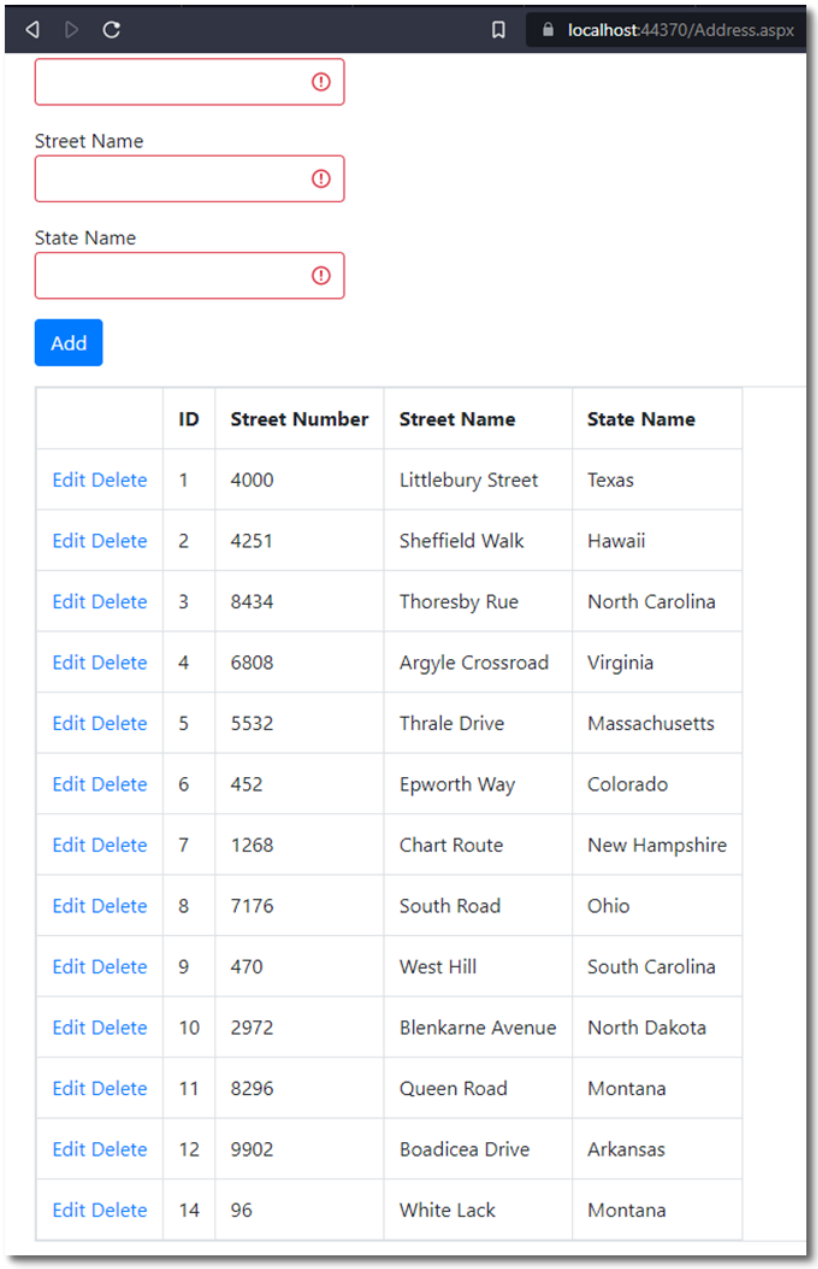
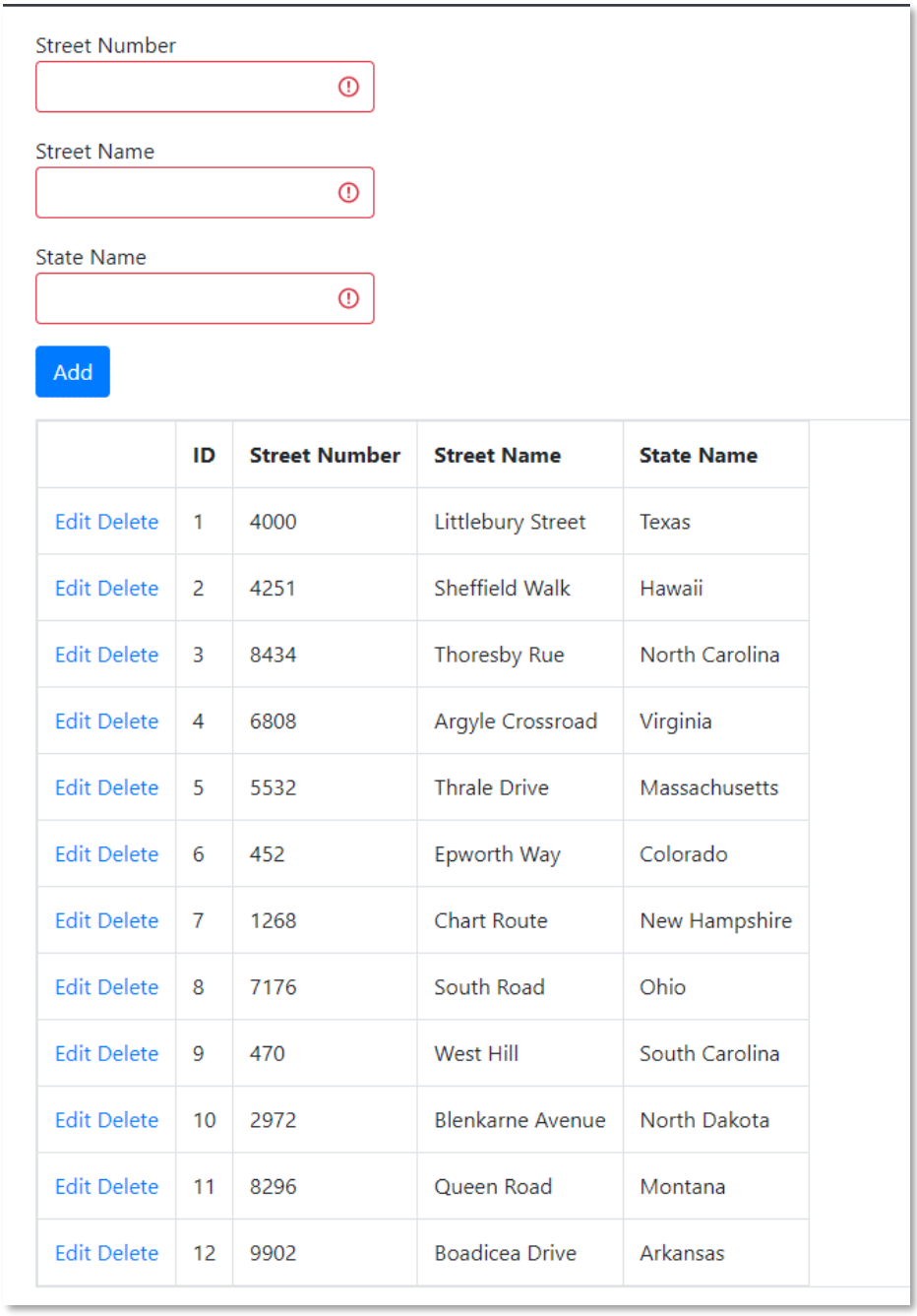


Figure 78: Address Grid Pre-Deletion



	ID	Street Number	Street Name	State Name
Edit Delete	1	4000	Littlebury Street	Texas
Edit Delete	2	4251	Sheffield Walk	Hawaii
Edit Delete	3	8434	Thoresby Rue	North Carolina
Edit Delete	4	6808	Argyle Crossroad	Virginia
Edit Delete	5	5532	Thrale Drive	Massachusetts
Edit Delete	6	452	Epworth Way	Colorado
Edit Delete	7	1268	Chart Route	New Hampshire
Edit Delete	8	7176	South Road	Ohio
Edit Delete	9	470	West Hill	South Carolina
Edit Delete	10	2972	Blenkarne Avenue	North Dakota
Edit Delete	11	8296	Queen Road	Montana
Edit Delete	12	9902	Boadicea Drive	Arkansas

Figure 79: Address Grid Post-Deletion

13.1.3. Module Form

13.1.3.1. Test 1: View

Test Condition	Expected Outcome	Obtained Outcome
Load the module form and check if data is loaded in the grid.	The data is loaded on page load.	The data was loaded on page load.

Table 25: Module Test 1

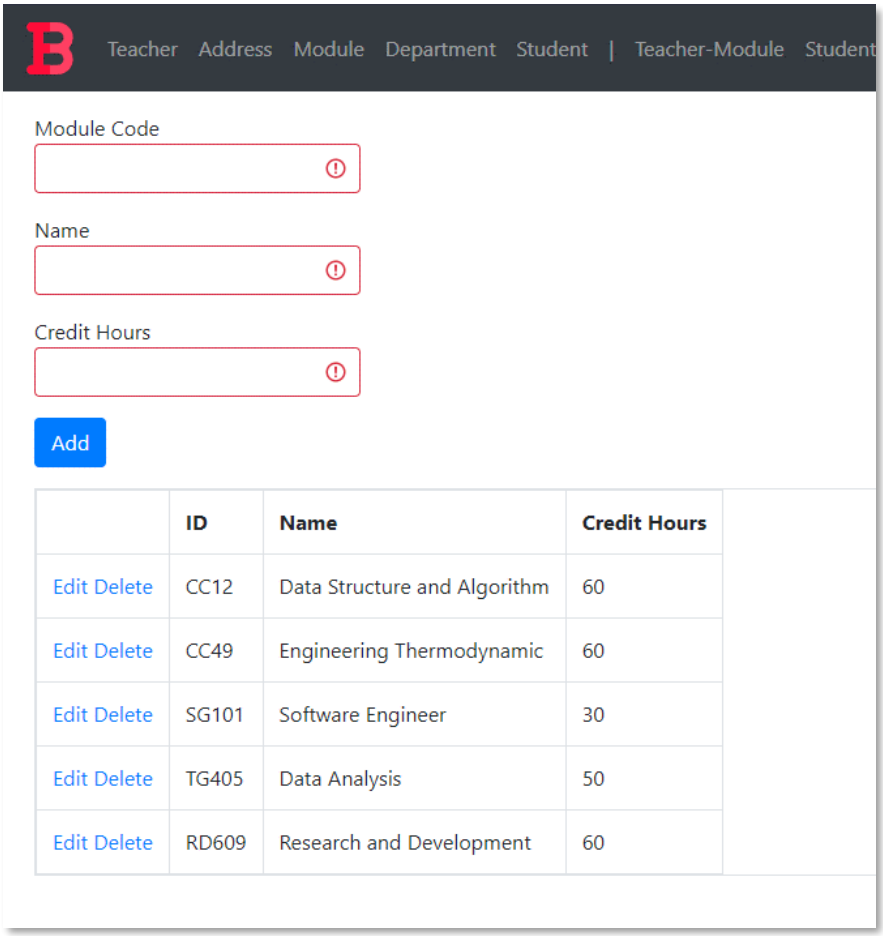


Figure 80: Module Test 1 evidence

13.1.3.2. Test 2: Add

Test Condition	Expected Outcome	Obtained Outcome
Fill the module form appropriately and press the add button.	The data is added, and the grid is refreshed with the new data.	The data was added without any errors and the grid was refreshed with the new data.

Table 26: Module Test 2

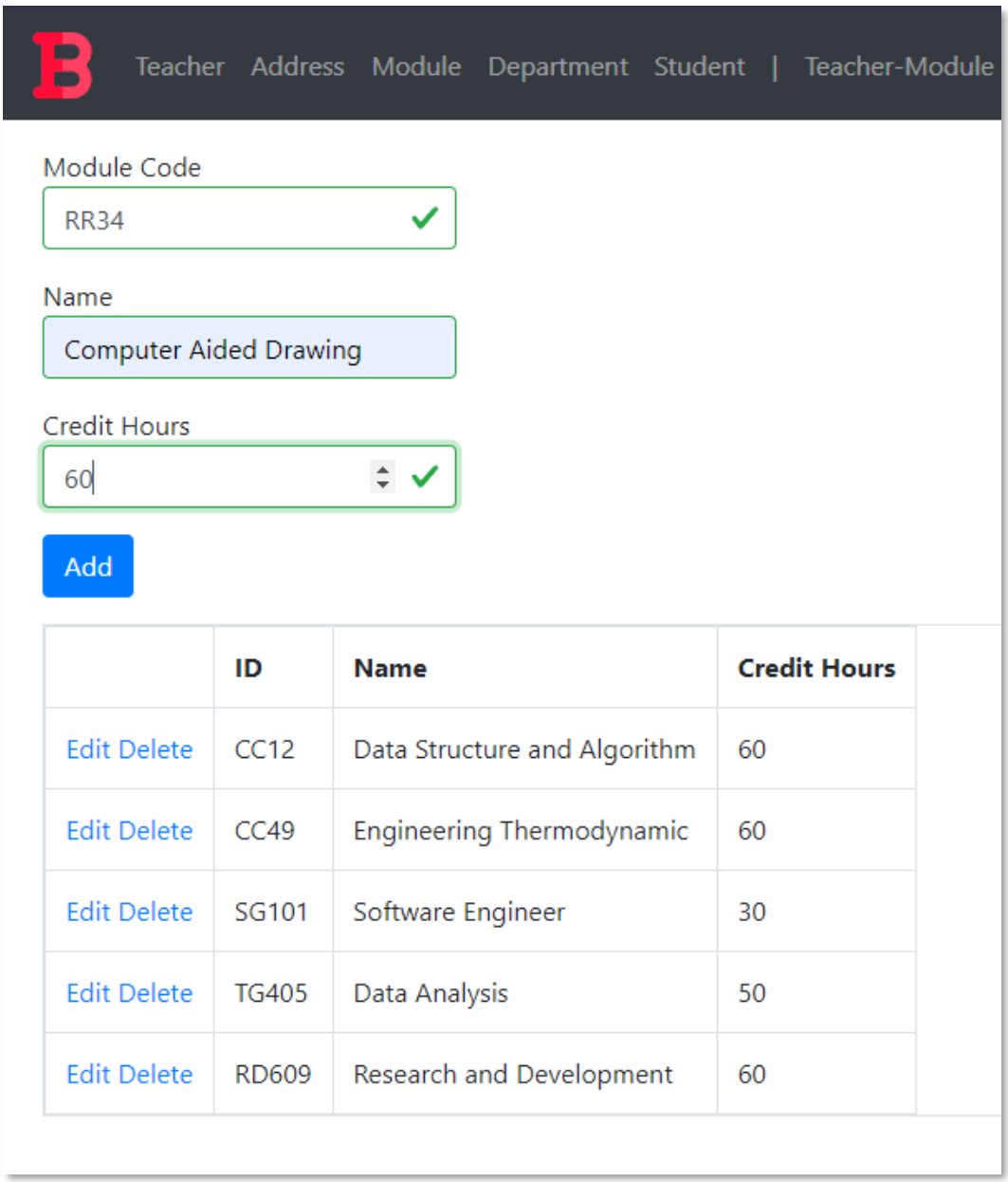


Figure 81: Module Test 2 Data to be Added

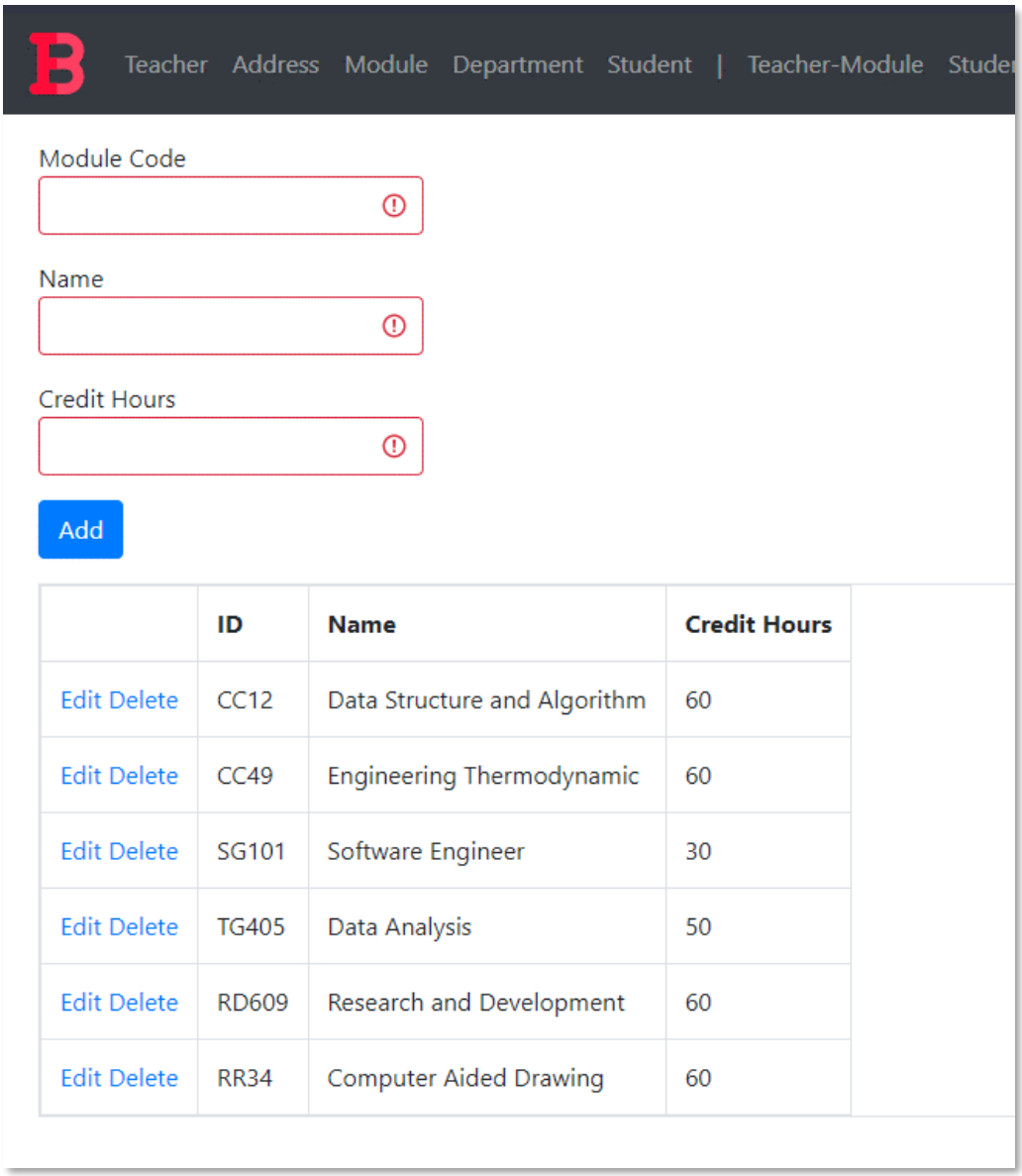


Figure 82: Module Test 2 Addition Result

13.1.3.3. Test 3: Edit

Test Condition	Expected Outcome	Obtained Outcome
Click the edit button	• The add button disappears and an update button appears. • The data in the row gets filled in the available textboxes.	• The add button disappeared and an update button appeared. • The data in the field got filled in the available textboxes.
Click the update button	• The update button disappears and the add button reappears • The data in the grid is refreshed with the edited data.	• The update button disappears and the add button reappears • The data in the grid is refreshed with the edited data.

Table 27: Module Test 3

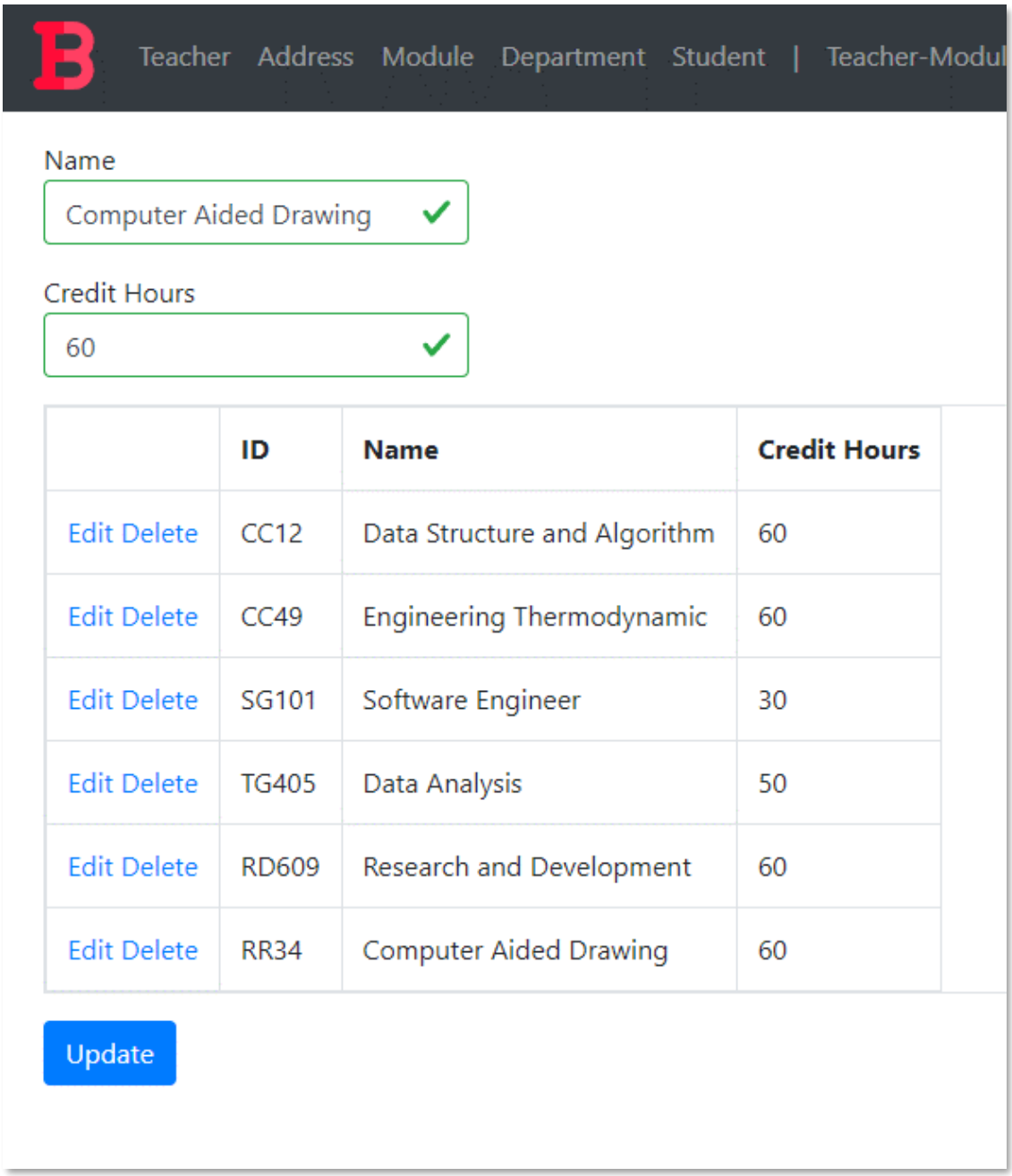


Figure 83: Module Test 3 condition 1

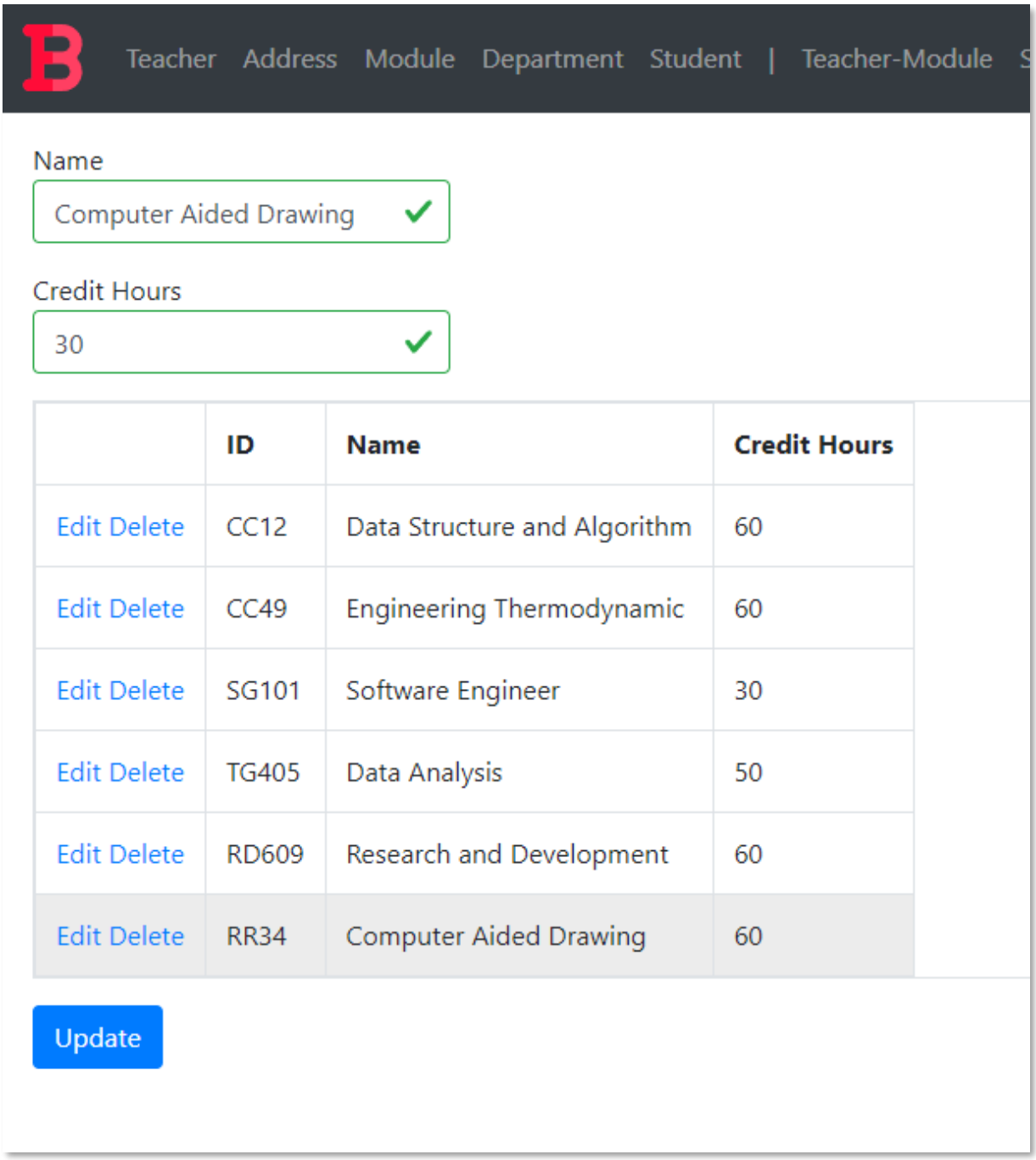


Figure 84: Edited value in Credit Hours Textbox

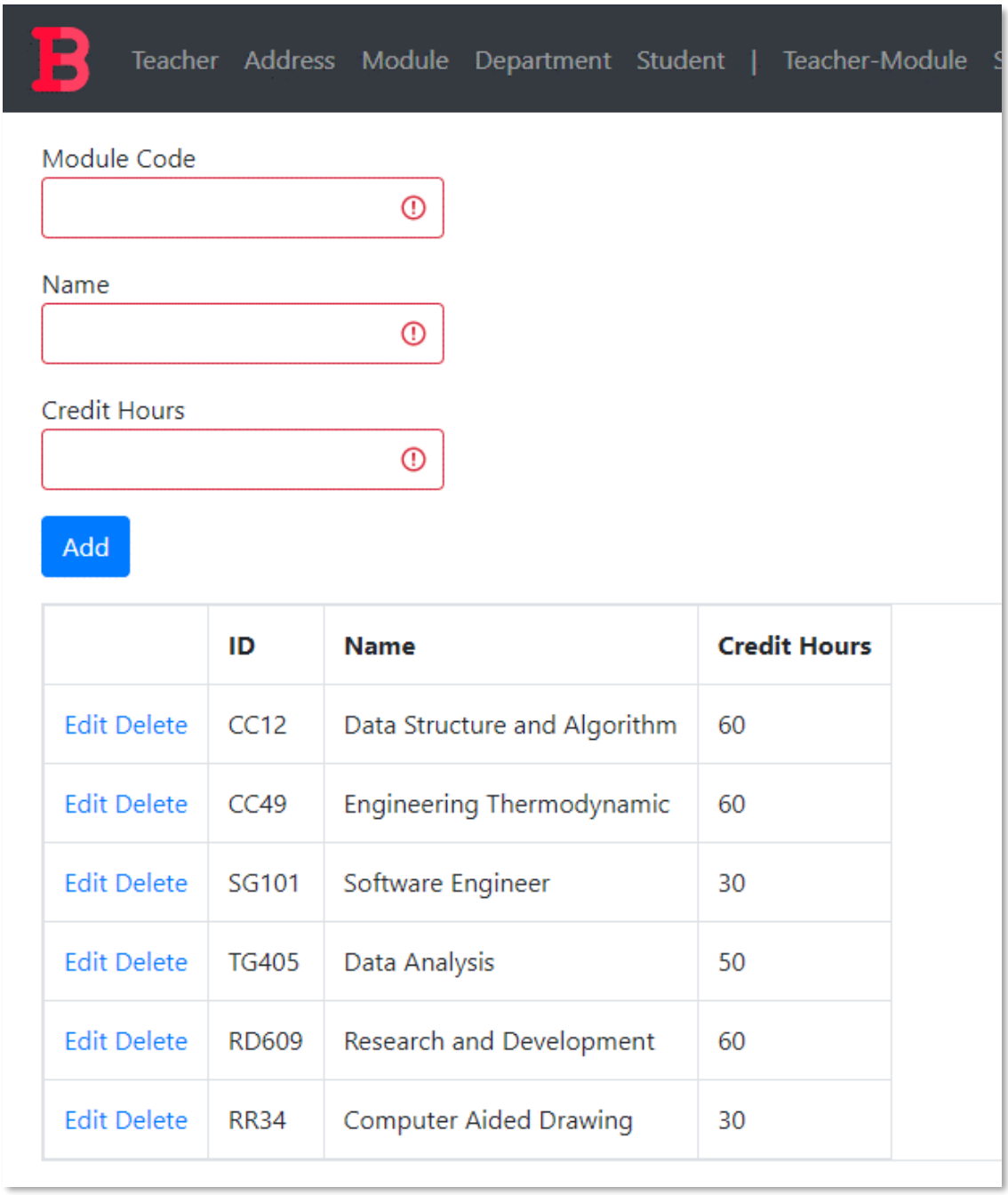


Figure 85: Module Test 3 Condition 2

13.1.3.4. Test 4: Delete

Test Condition	Expected Outcome	Obtained Outcome
Click the delete button.	The grid is refreshed with the data row deleted.	The grid was refreshed with the data row deleted.

Table 28: Module Test 4

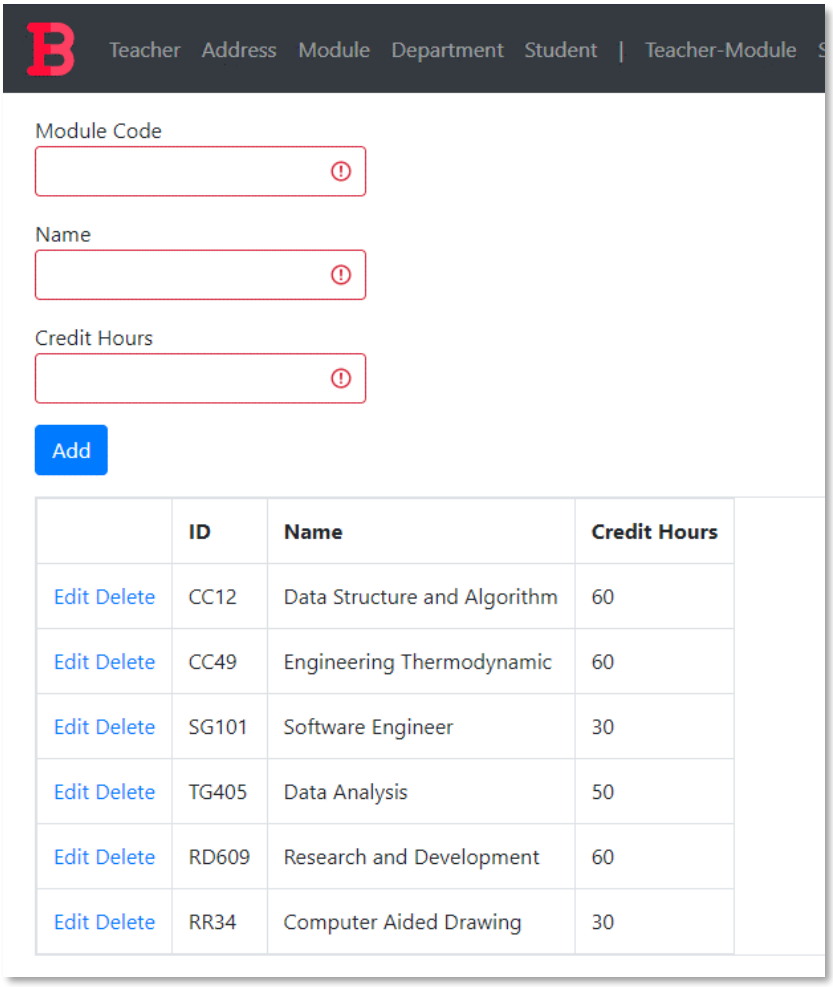


Figure 86: Address Grid Pre-Deletion

13.1.3.4.1. What caused the issue and how it was rectified:

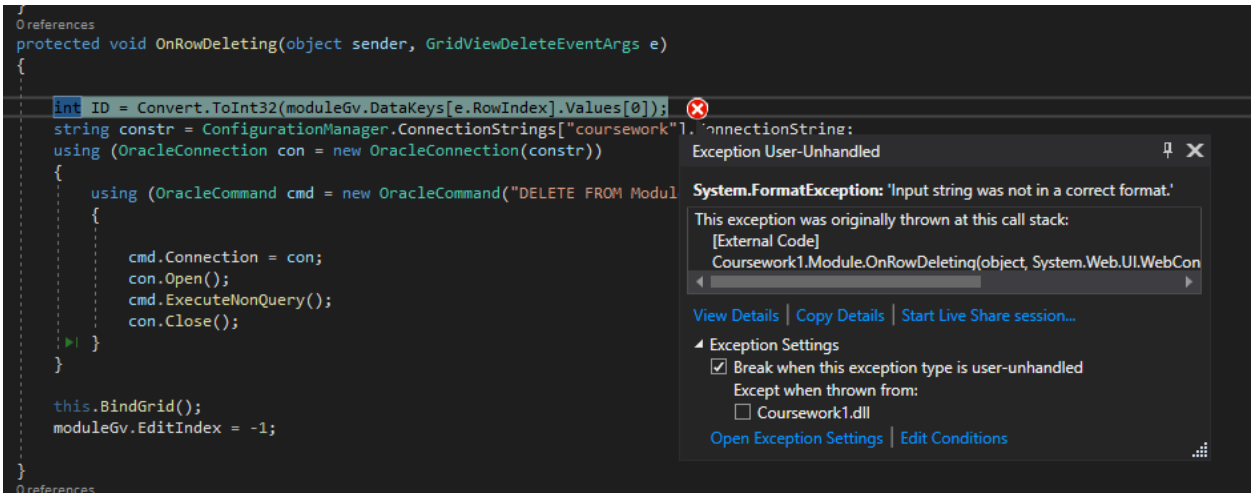


Figure 87: Issue causing the fail case

Unlike the primary keys from other form table, module table had a VARCHAR primary key. Here, I was trying to convert the string into an integer in order to obtain the primary key for the row of data that needed to be deleted. Hence, it caused a format exception and did not carry out the delete procedure.

I set the variable to string and used the value obtained from the grid view row as a string and rectified the issue. This made the deletion procedure go smoothly and uninterrupted.

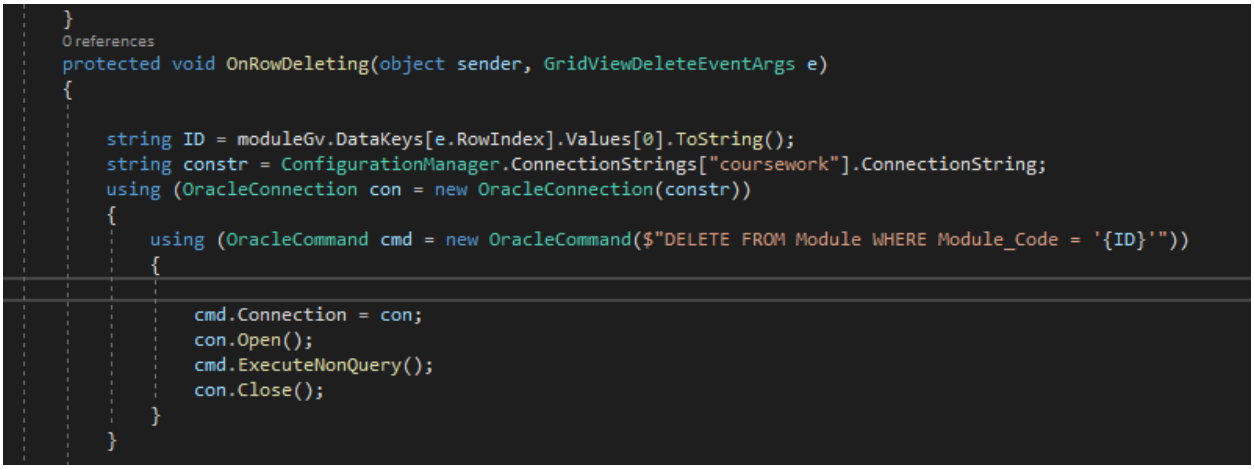


Figure 88: Rectification of the Issue causing the fail case

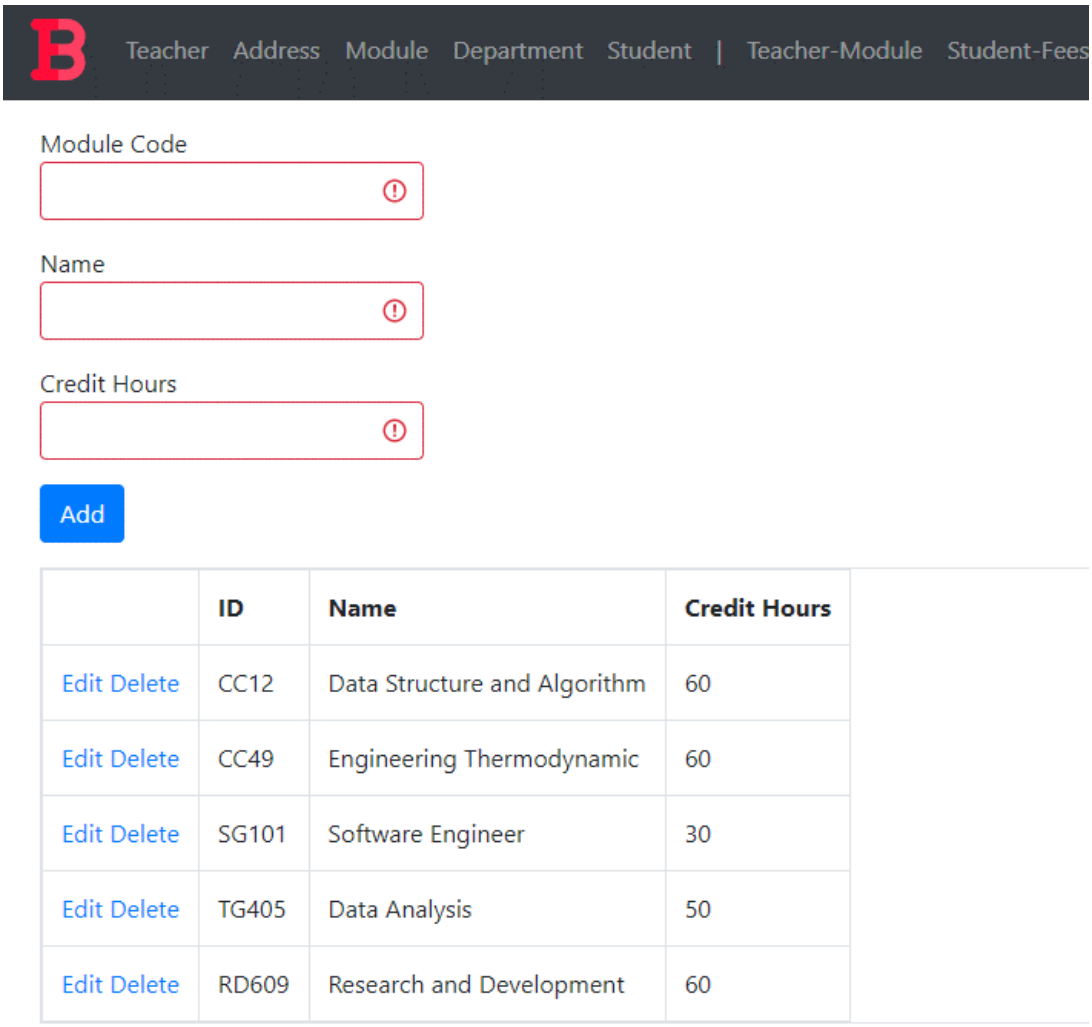


Figure 89: Test 4 post deletion

13.1.4. Department Form

13.1.4.1. Test 1: View

Test Condition	Expected Outcome	Obtained Outcome
Load the department form and check if data is loaded in the grid.	The data is loaded on page load.	The data was loaded on page load.

Table 29: Department Test 1

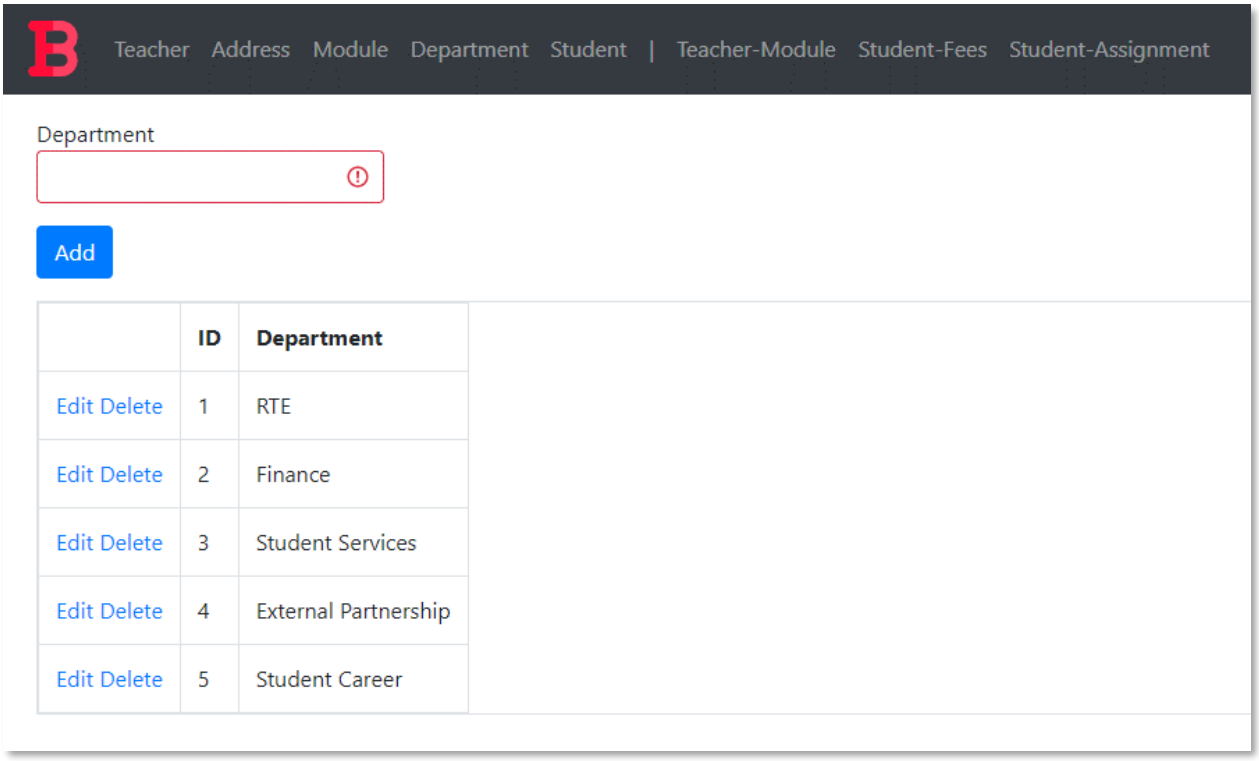


Figure 90: Department Test 1 evidence

13.1.4.2. Test 2: Add

Test Condition	Expected Outcome	Obtained Outcome
Fill the department form appropriately and press the add button.	The data is added, and the grid is refreshed with the new data.	The data was added without any errors and the grid was refreshed with the new data.

Table 30: Department Test 2

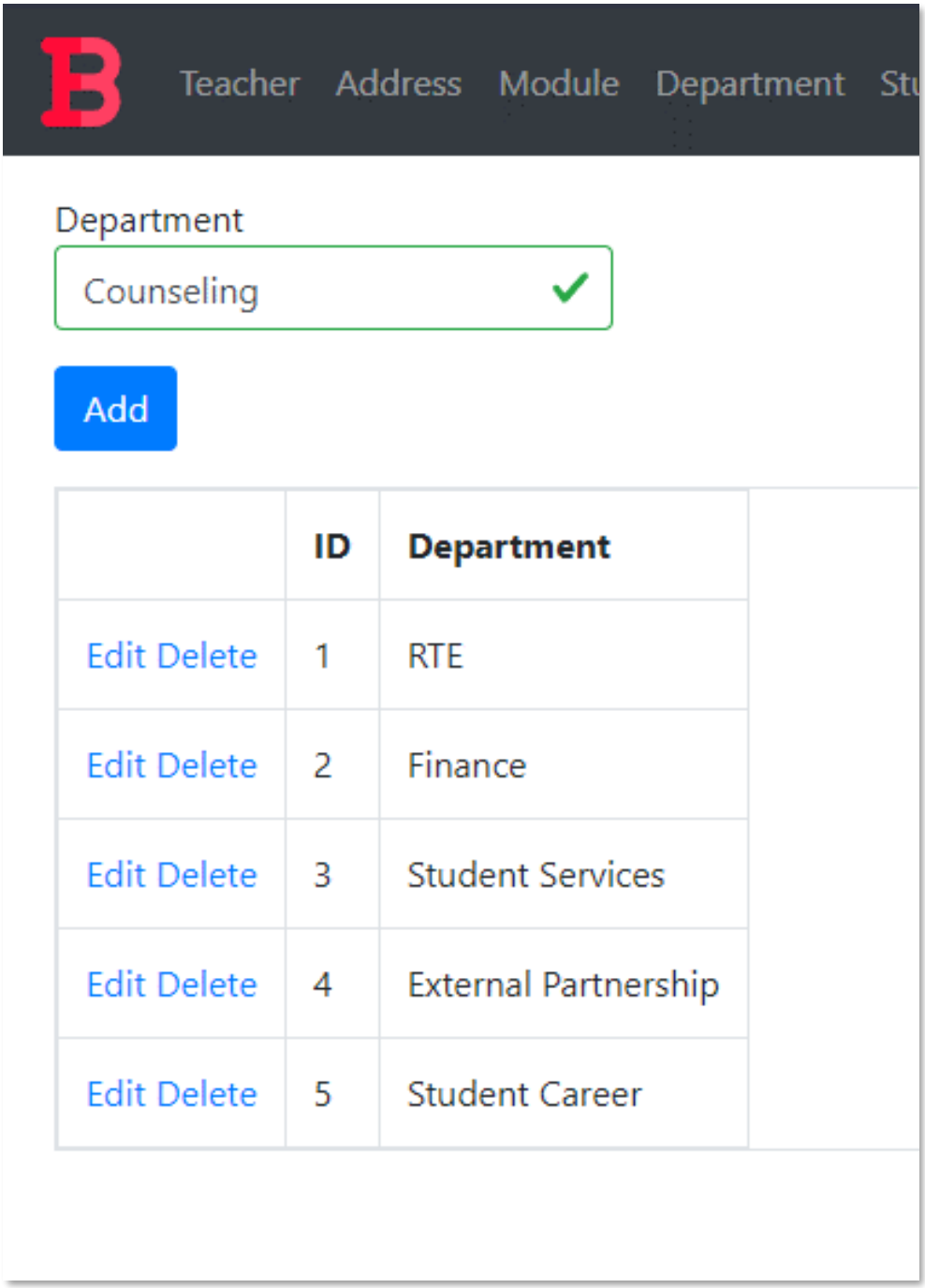


Figure 91: Department Test 2 Added Data

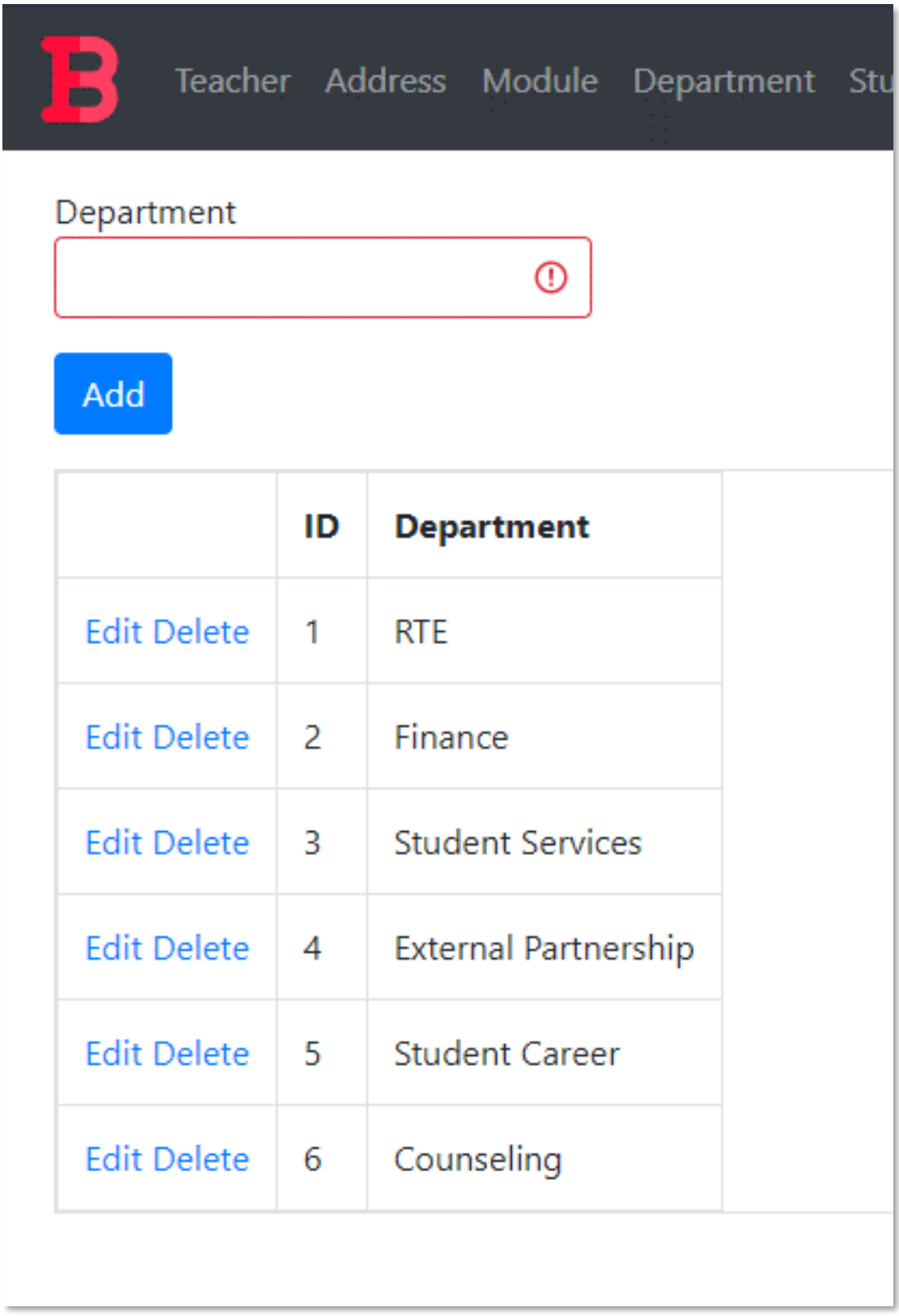


Figure 92: Department Test 2 Addition Result

13.1.4.3. Test 3: Edit

Test Condition	Expected Outcome	Obtained Outcome
Click the edit button	• The add button disappears and an update button appears. • The data in the row gets filled in the available textboxes.	• The add button disappeared and an update button appeared. • The data in the field got filled in the available textboxes.
Click the update button	• The update button disappears and the add button reappears • The data in the grid is refreshed with the edited data.	• The update button disappears and the add button reappears • The data in the grid is refreshed with the edited data.

Table 31: Department Test 3

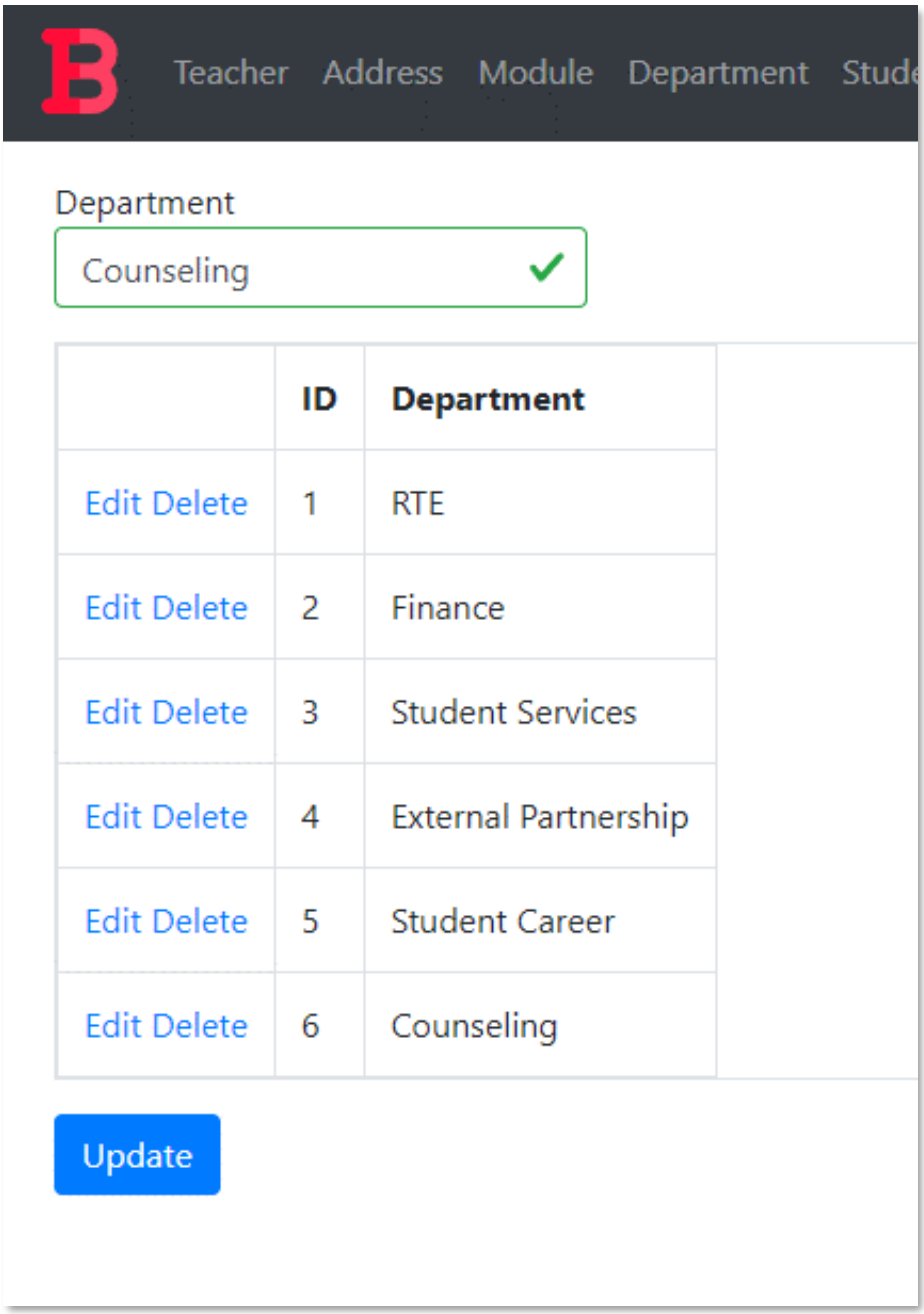


Figure 93: Department Test 3 condition 1

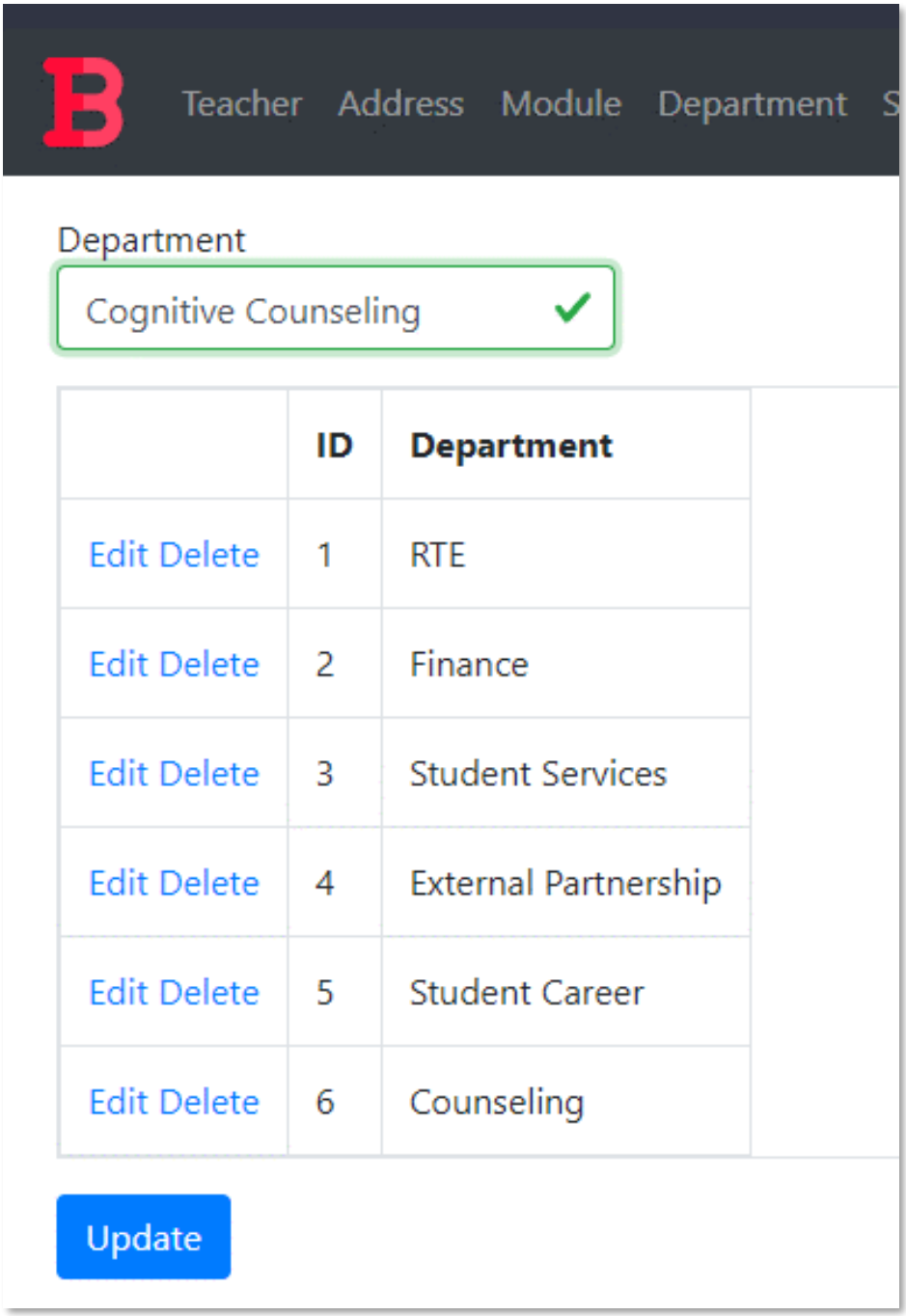


Figure 94: Edited value in Department Textbox

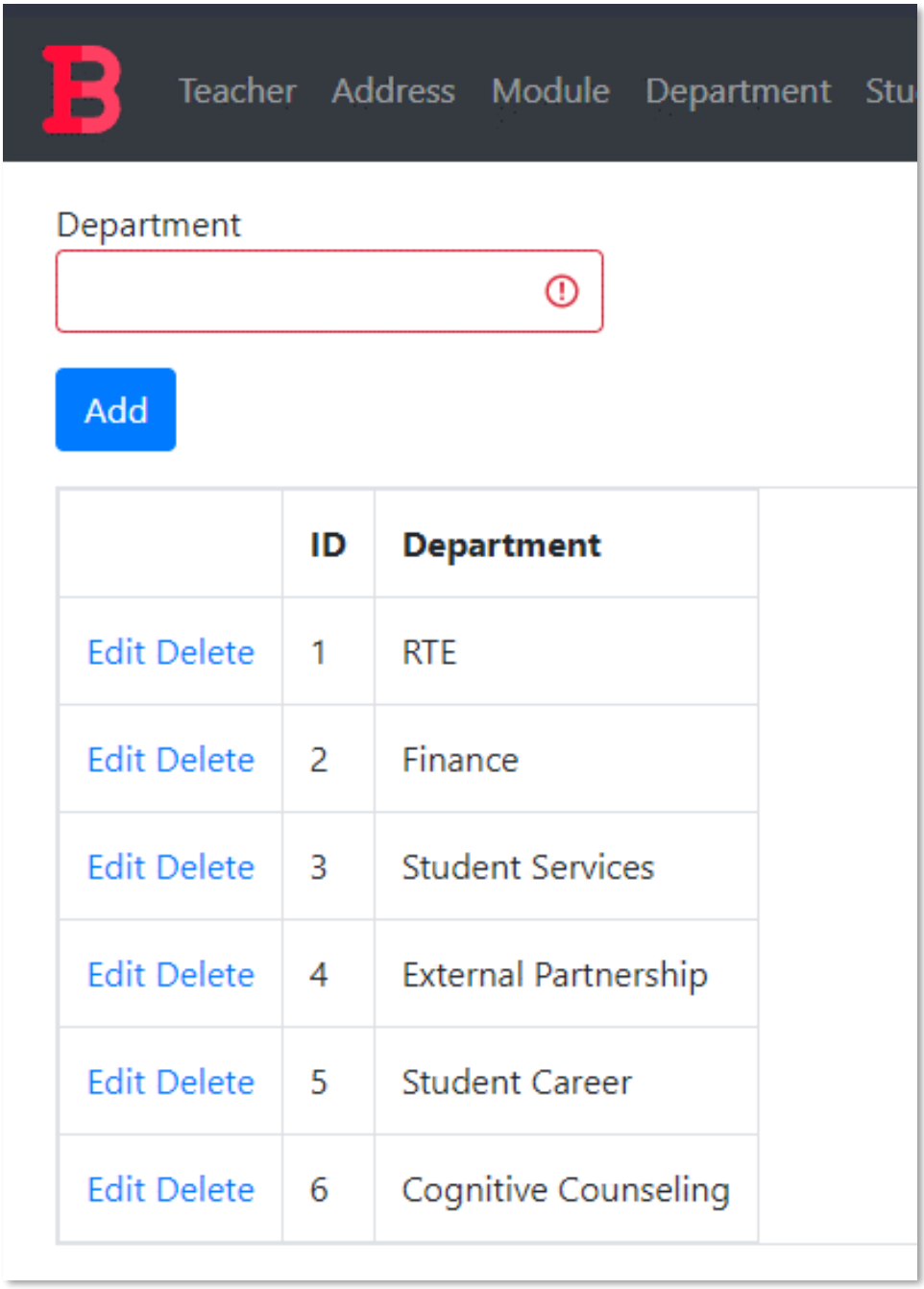


Figure 95: Department Test 3 Condition 2

13.1.4.4. Test 4: Delete

Test Condition	Expected Outcome	Obtained Outcome
Click the delete button.	The grid is refreshed with the data row deleted.	The grid was refreshed with the data row deleted.

Table 32: Department Test 4

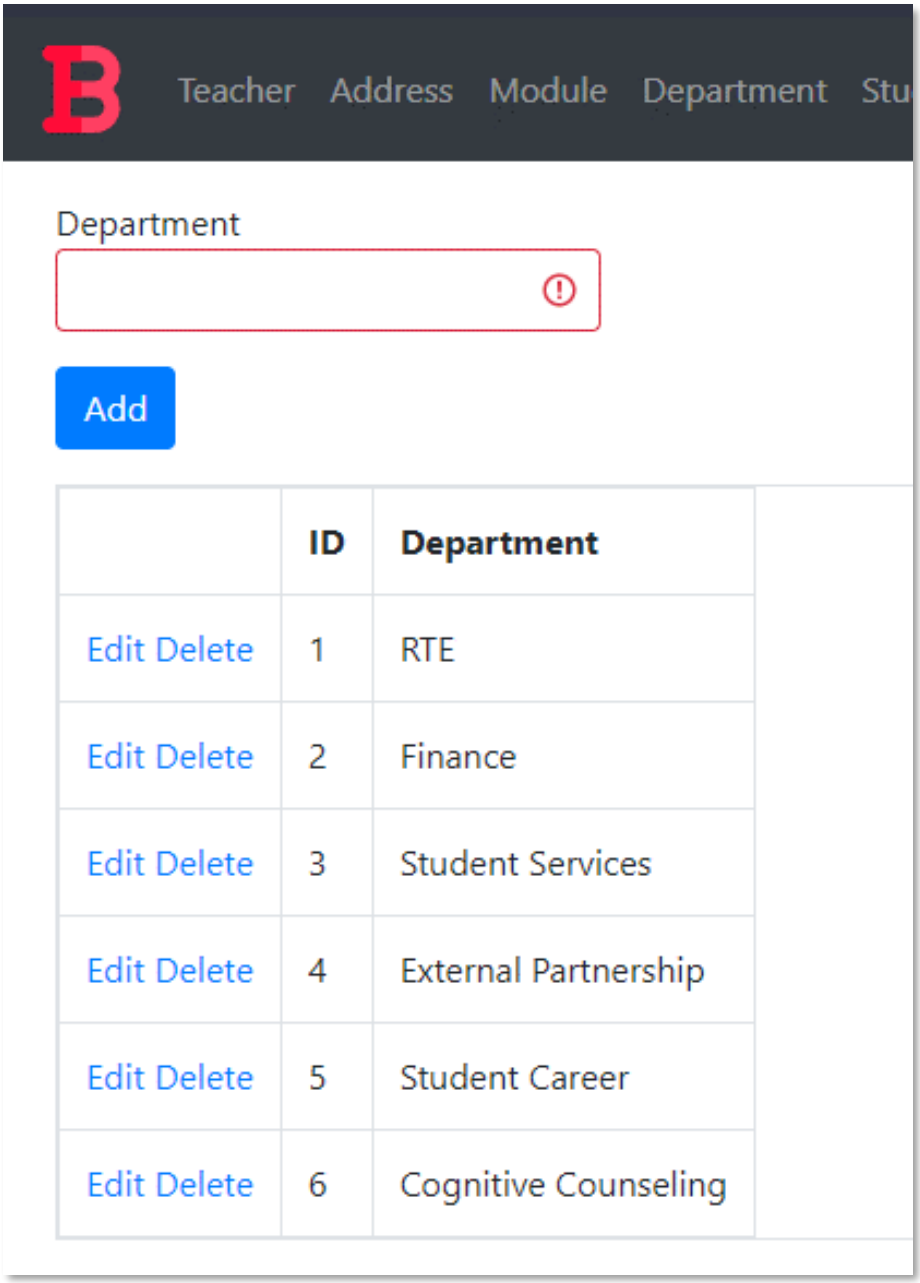


Figure 96: Department Grid Pre-Deletion

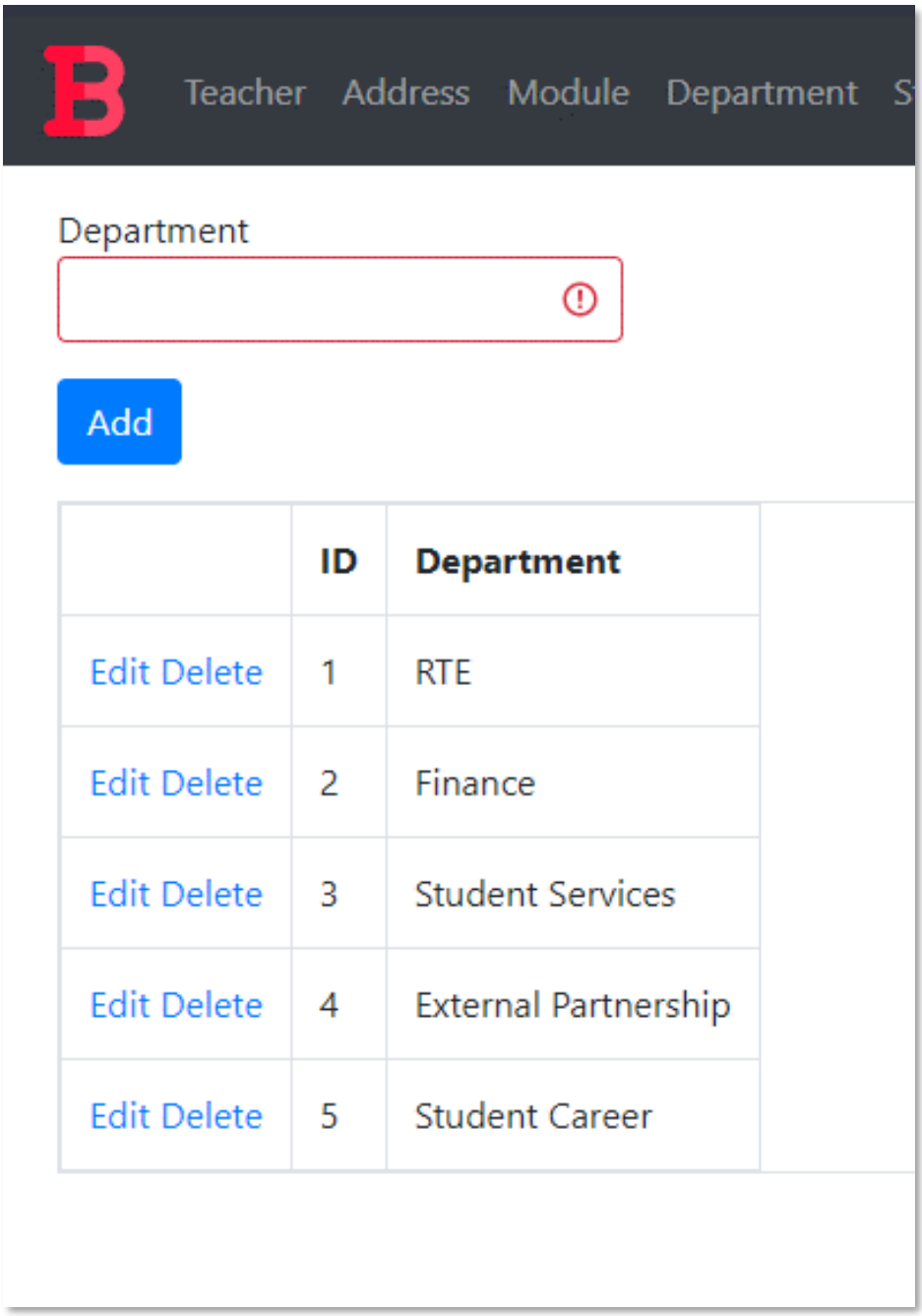


Figure 97: Department Grid Post-Deletion

13.1.5. Student Form

13.1.5.1. Test 1: View

Test Condition	Expected Outcome	Obtained Outcome
Load the student form and check if data is loaded in the grid.	The data is loaded on page load.	The data was loaded on page load.

Table 33: Student Test 1

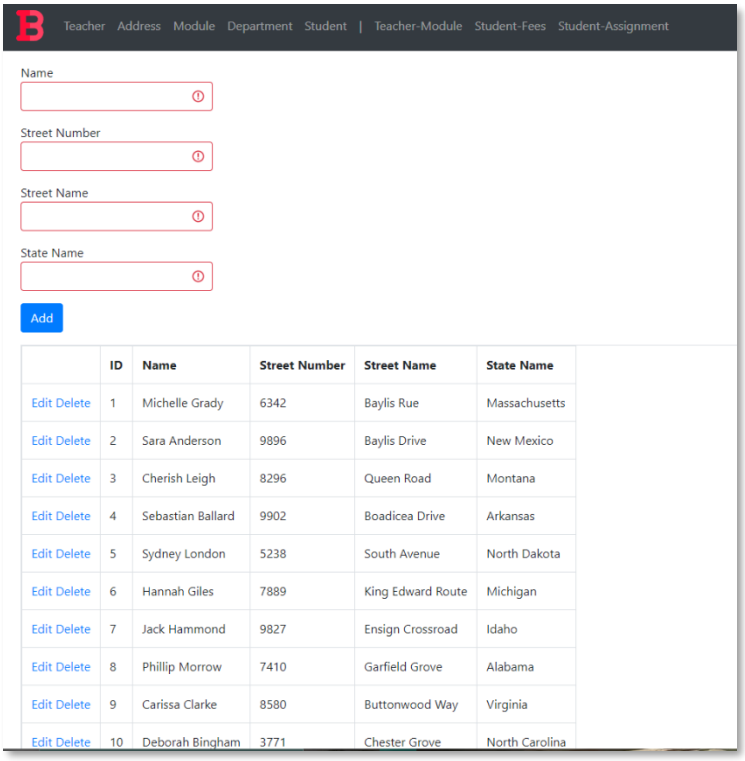


Figure 98: Student Test 1 evidence

13.1.5.2. Test 2: Add

Test Condition	Expected Outcome	Obtained Outcome
Fill the student form appropriately and press the add button.	The data is added, and the grid is refreshed with the new data.	The data was added without any errors and the grid was refreshed with the new data.

Table 34: Student Test 2

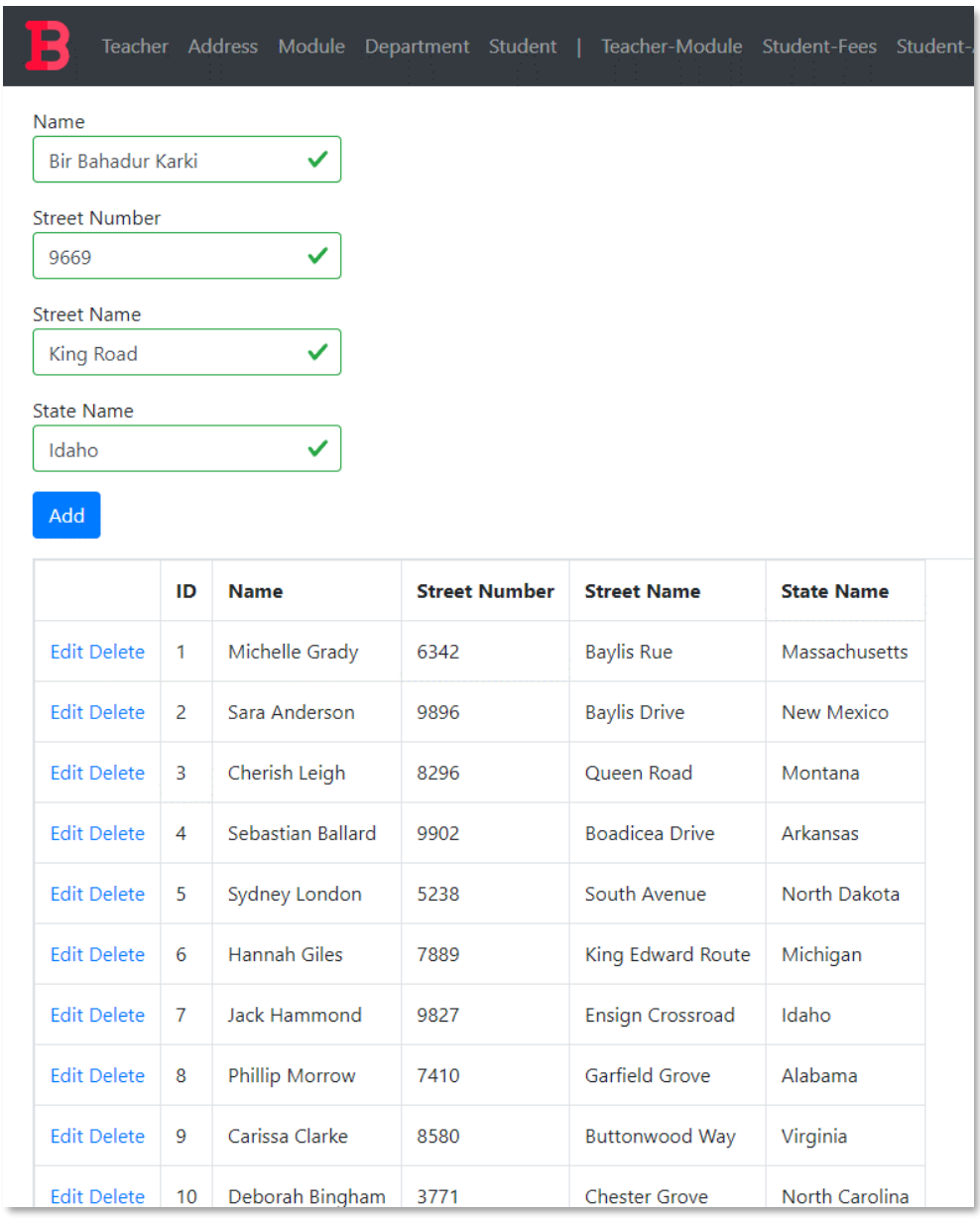
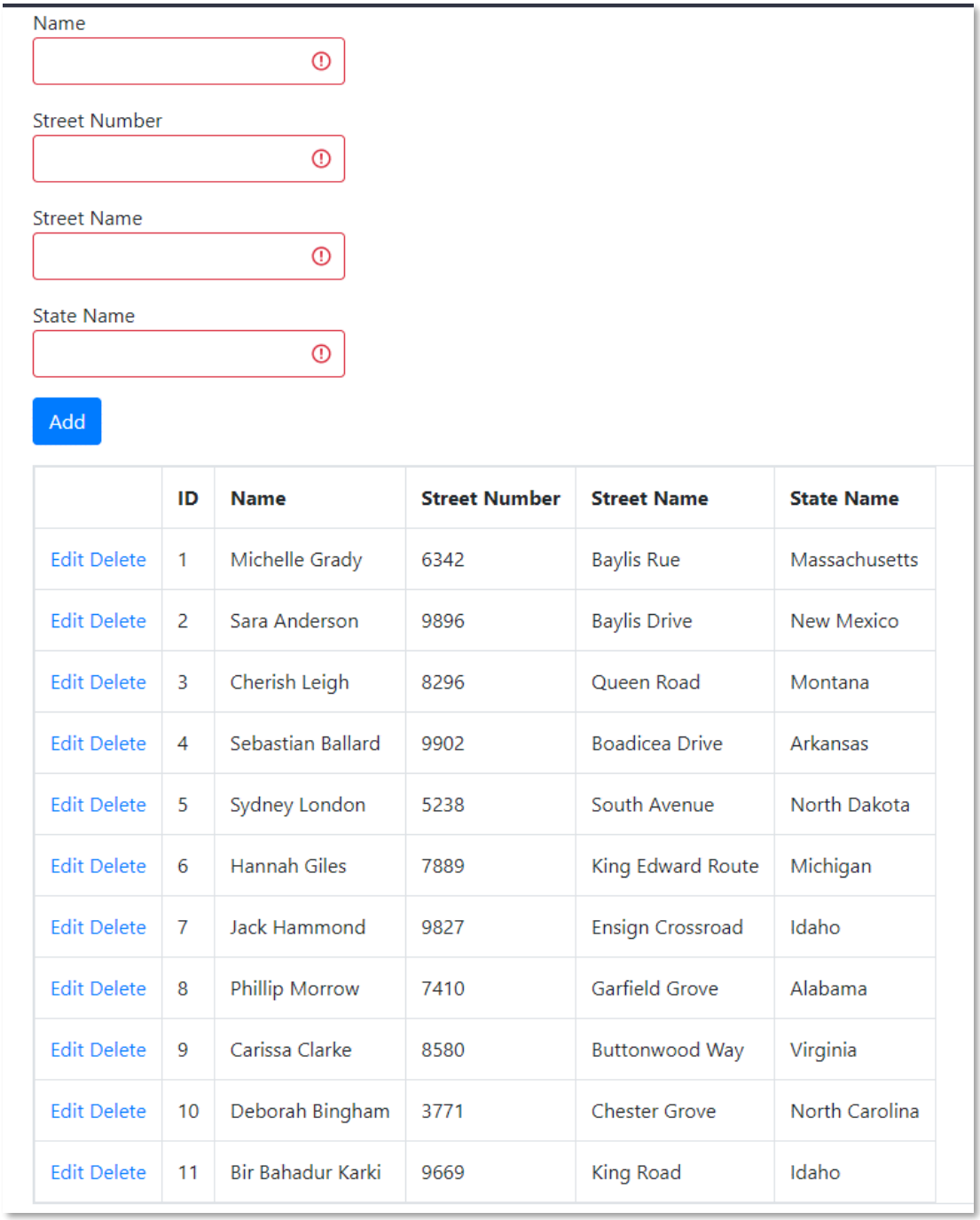


Figure 99: Student Test 2 Added Data



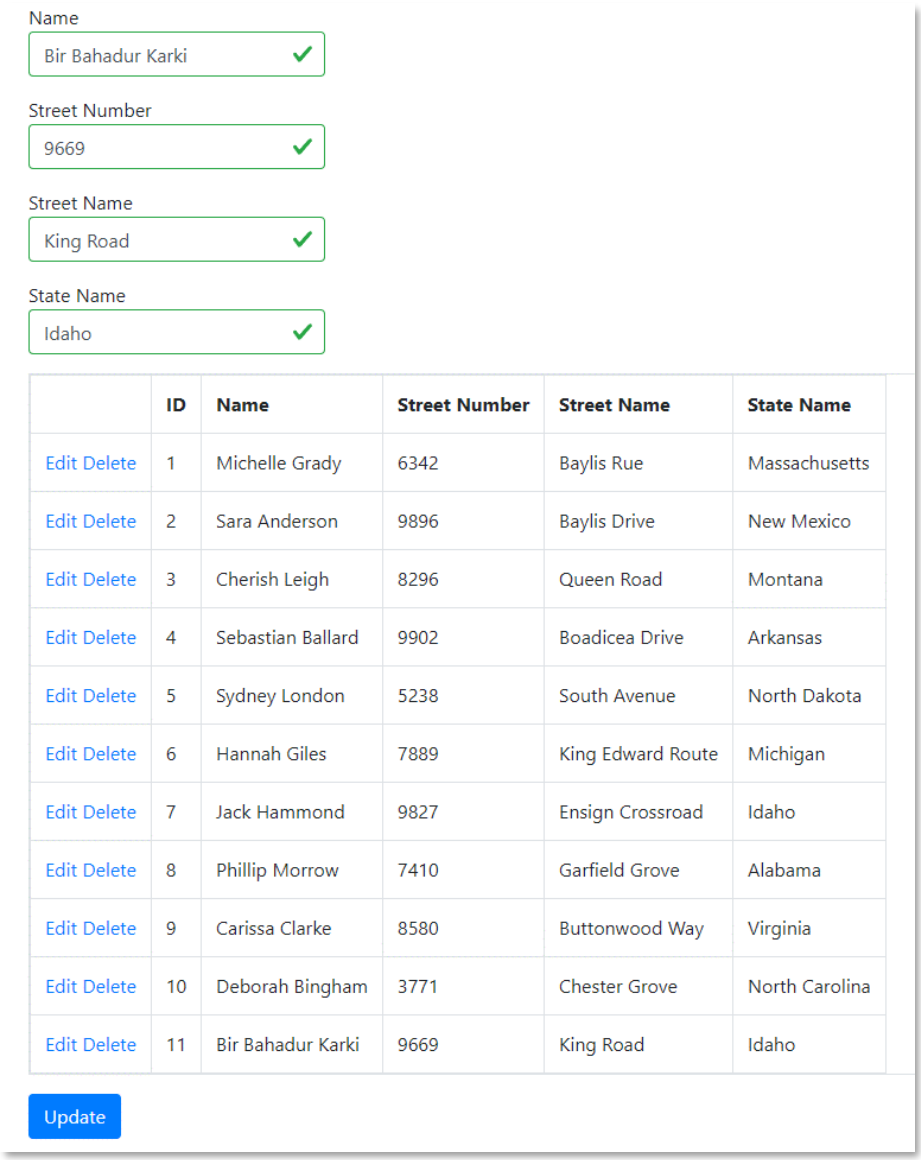
	ID	Name	Street Number	Street Name	State Name
Edit Delete	1	Michelle Grady	6342	Baylis Rue	Massachusetts
Edit Delete	2	Sara Anderson	9896	Baylis Drive	New Mexico
Edit Delete	3	Cherish Leigh	8296	Queen Road	Montana
Edit Delete	4	Sebastian Ballard	9902	Boadicea Drive	Arkansas
Edit Delete	5	Sydney London	5238	South Avenue	North Dakota
Edit Delete	6	Hannah Giles	7889	King Edward Route	Michigan
Edit Delete	7	Jack Hammond	9827	Ensign Crossroad	Idaho
Edit Delete	8	Phillip Morrow	7410	Garfield Grove	Alabama
Edit Delete	9	Carissa Clarke	8580	Buttonwood Way	Virginia
Edit Delete	10	Deborah Bingham	3771	Chester Grove	North Carolina
Edit Delete	11	Bir Bahadur Karki	9669	King Road	Idaho

Figure 100: Student Test 2 Addition Result

13.1.5.3. Test 3: Edit

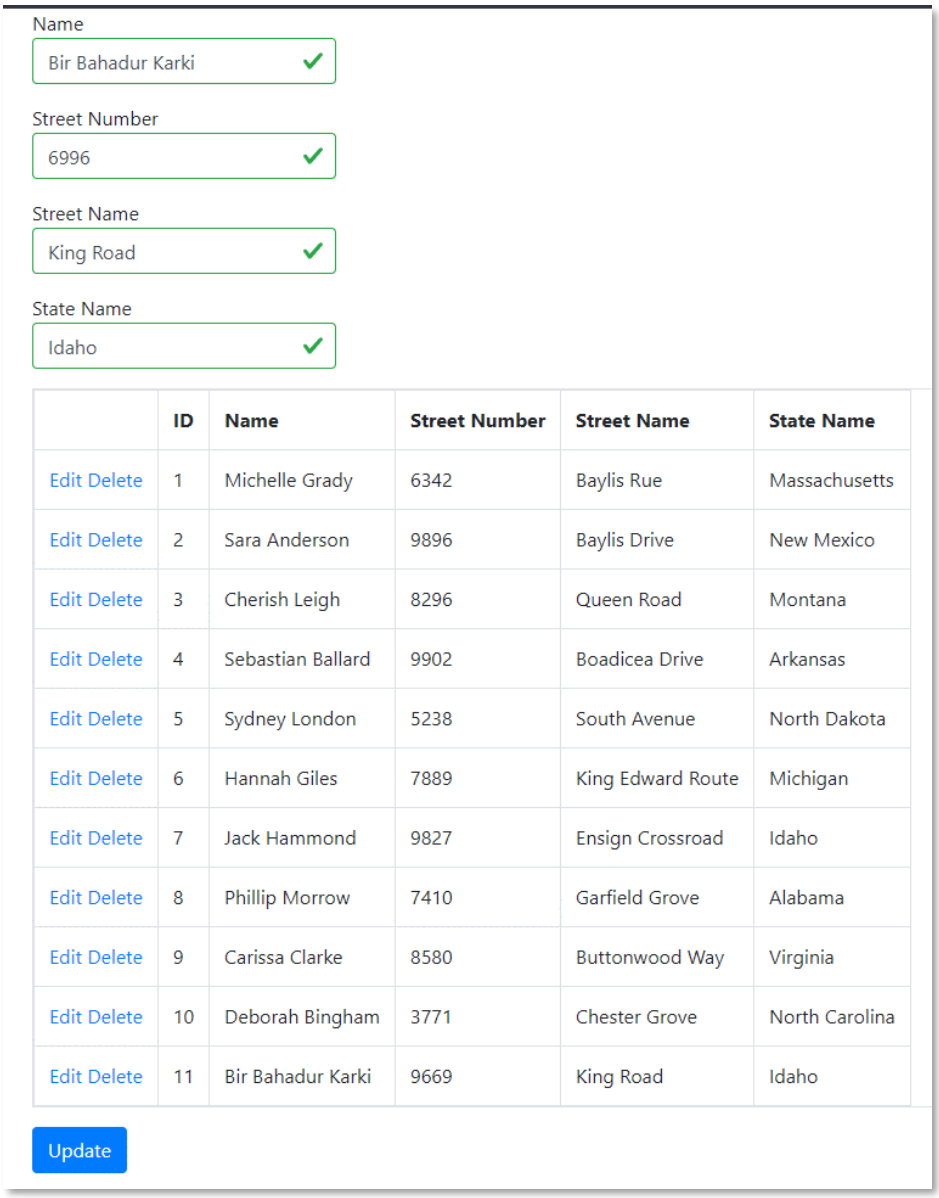
Test Condition	Expected Outcome	Obtained Outcome
Click the edit button	• The add button disappears and an update button appears. • The data in the row gets filled in the available textboxes.	• The add button disappeared and an update button appeared. • The data in the field got filled in the available textboxes.
Click the update button	• The update button disappears and the add button reappears • The data in the grid is refreshed with the edited data.	• The update button disappears and the add button reappears • The data in the grid is refreshed with the edited data.

Table 35: Student Test 3



	ID	Name	Street Number	Street Name	State Name
Edit Delete	1	Michelle Grady	6342	Baylis Rue	Massachusetts
Edit Delete	2	Sara Anderson	9896	Baylis Drive	New Mexico
Edit Delete	3	Cherish Leigh	8296	Queen Road	Montana
Edit Delete	4	Sebastian Ballard	9902	Boadicea Drive	Arkansas
Edit Delete	5	Sydney London	5238	South Avenue	North Dakota
Edit Delete	6	Hannah Giles	7889	King Edward Route	Michigan
Edit Delete	7	Jack Hammond	9827	Ensign Crossroad	Idaho
Edit Delete	8	Phillip Morrow	7410	Garfield Grove	Alabama
Edit Delete	9	Carissa Clarke	8580	Buttonwood Way	Virginia
Edit Delete	10	Deborah Bingham	3771	Chester Grove	North Carolina
Edit Delete	11	Bir Bahadur Karki	9669	King Road	Idaho

Figure 101: Student Test 3 condition 1



	ID	Name	Street Number	Street Name	State Name
Edit Delete	1	Michelle Grady	6342	Baylis Rue	Massachusetts
Edit Delete	2	Sara Anderson	9896	Baylis Drive	New Mexico
Edit Delete	3	Cherish Leigh	8296	Queen Road	Montana
Edit Delete	4	Sebastian Ballard	9902	Boadicea Drive	Arkansas
Edit Delete	5	Sydney London	5238	South Avenue	North Dakota
Edit Delete	6	Hannah Giles	7889	King Edward Route	Michigan
Edit Delete	7	Jack Hammond	9827	Ensign Crossroad	Idaho
Edit Delete	8	Phillip Morrow	7410	Garfield Grove	Alabama
Edit Delete	9	Carissa Clarke	8580	Buttonwood Way	Virginia
Edit Delete	10	Deborah Bingham	3771	Chester Grove	North Carolina
Edit Delete	11	Bir Bahadur Karki	9669	King Road	Idaho

Figure 102: Edited value in Street Number Textbox

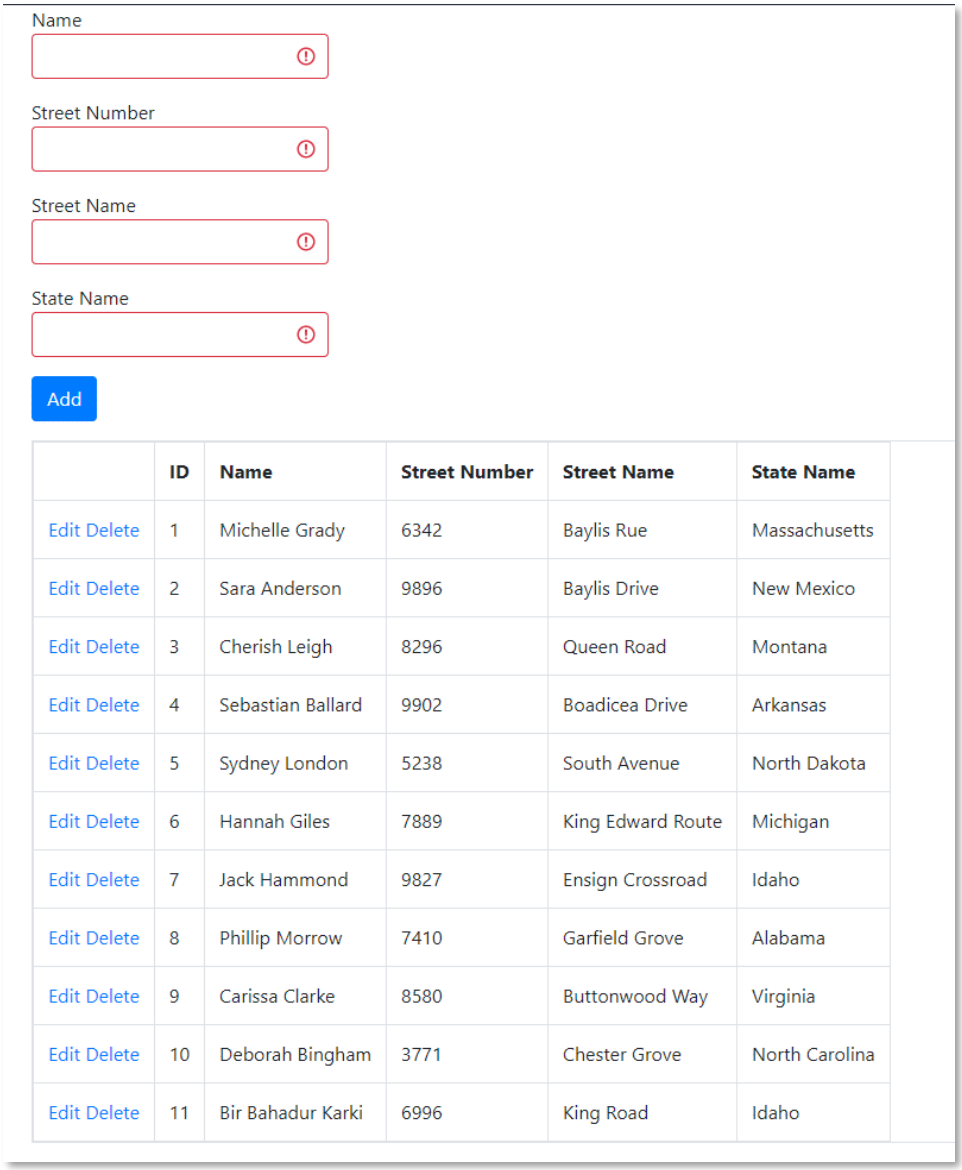


Figure 103: Student Test 3 Condition 2

13.1.5.4. Test 4: Delete

Test Condition	Expected Outcome	Obtained Outcome
Click the delete button.	The grid is refreshed with the data row deleted.	The grid was refreshed with the data row deleted.

Table 36: Student Test 4

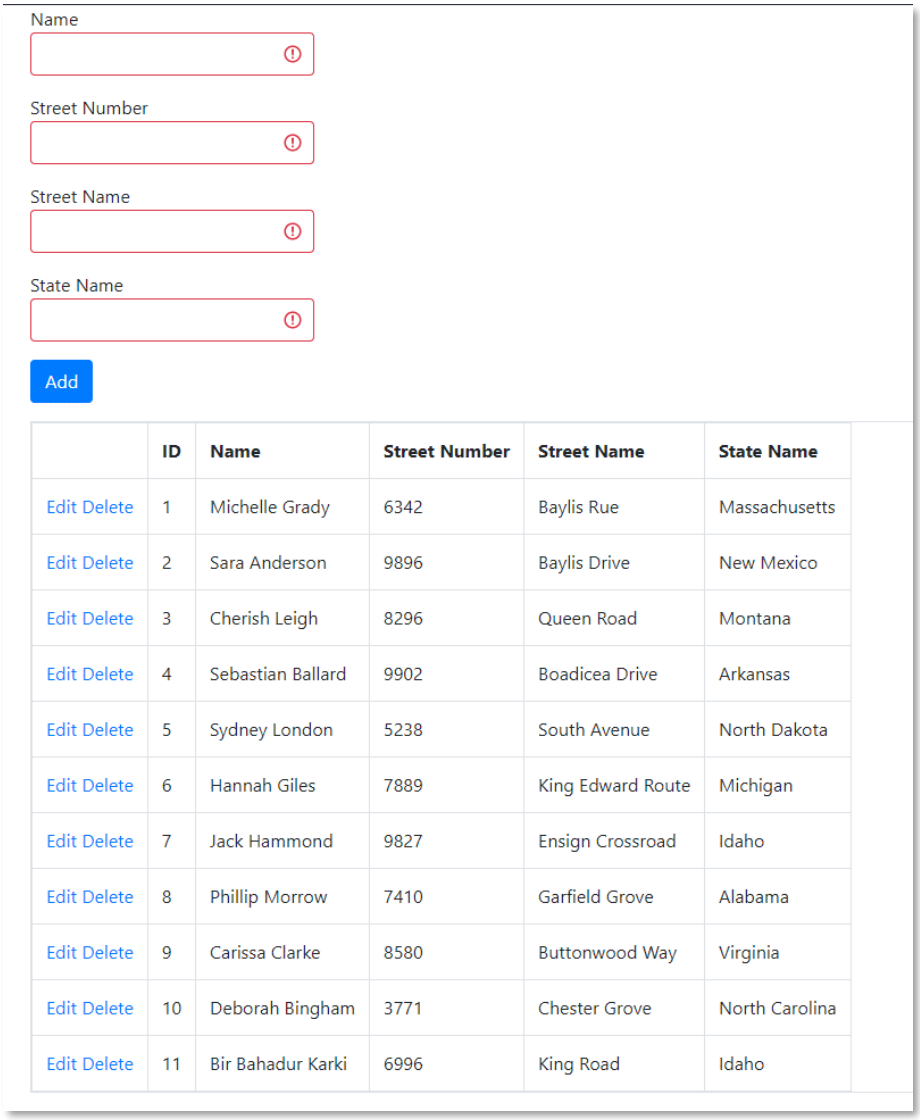


Figure 104: Student Grid Pre-Deletion

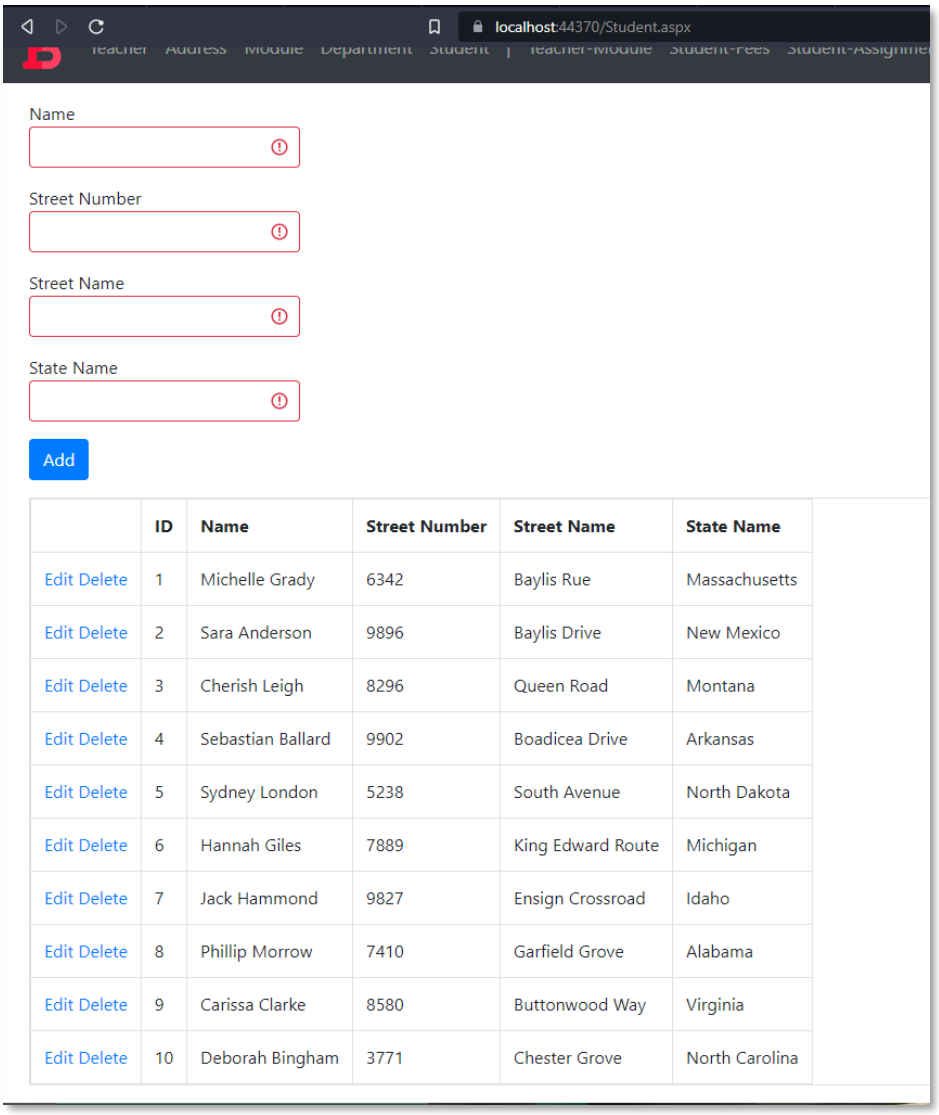


Figure 105: Student Grid Post-Deletion

13.2. Complex Forms

13.2.1. Teacher-Module Mapping Form

13.2.1.1. Test 1: Page Load

Test Condition	Expected Outcome	Obtained Outcome
Load the complex form and check if data is loaded in the initial grid.	The data is loaded to the initial grid on page load.	The data was loaded to the initial grid on page load.

Table 37: Teacher-Module Test 1

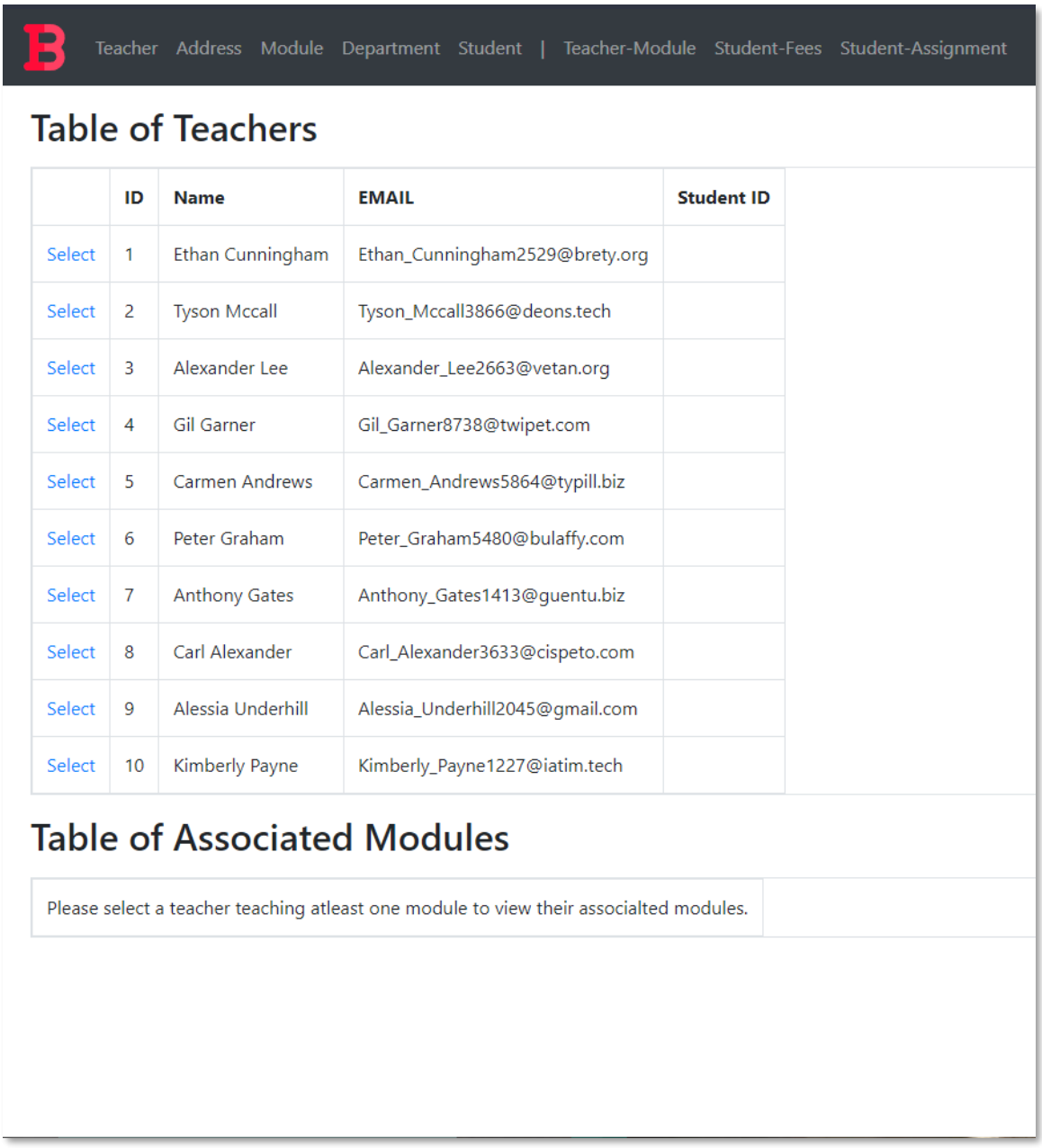


Figure 106: Teacher-Module Test 1 evidence

13.2.1.2. Test 2: Data Load on Selection

Test Condition	Expected Outcome	Obtained Outcome
Select a data row from the initial grid view and check if the other grid gets loaded with relevant data.	The other grid is loaded with relevant data.	The other grid was loaded with relevant data.

Table 38: Teacher-Module Test 2

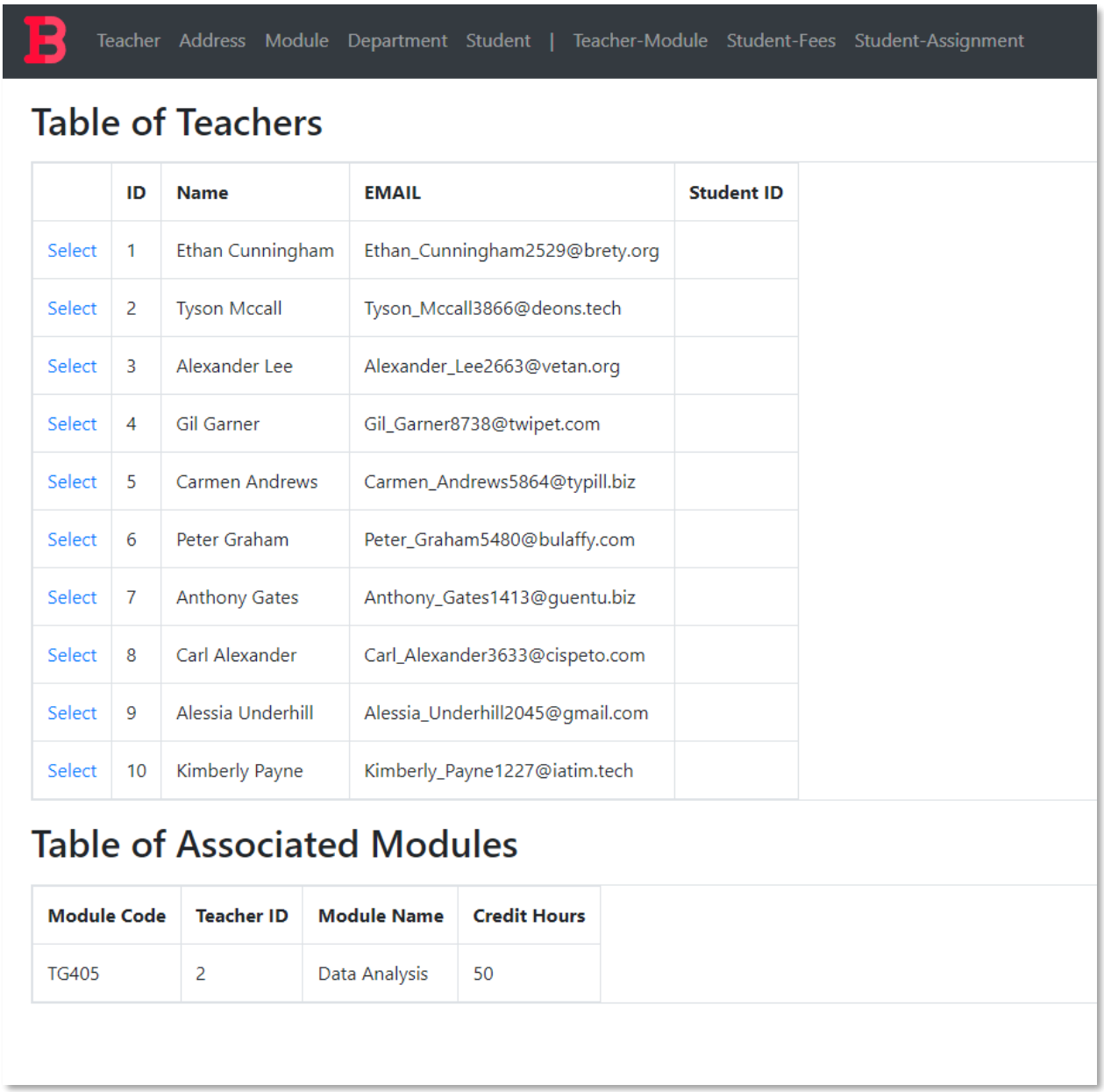


Figure 107: Teacher-Module Test 2 evidence

13.2.2. Student-Fees Mapping Form

13.2.2.1. Test 1: Page Load

Test Condition	Expected Outcome	Obtained Outcome
Load the complex form and check if data is loaded in the initial grid.	The data is loaded to the initial grid on page load.	The data was loaded to the initial grid on page load.

Table 39: Student-Fees Test 1

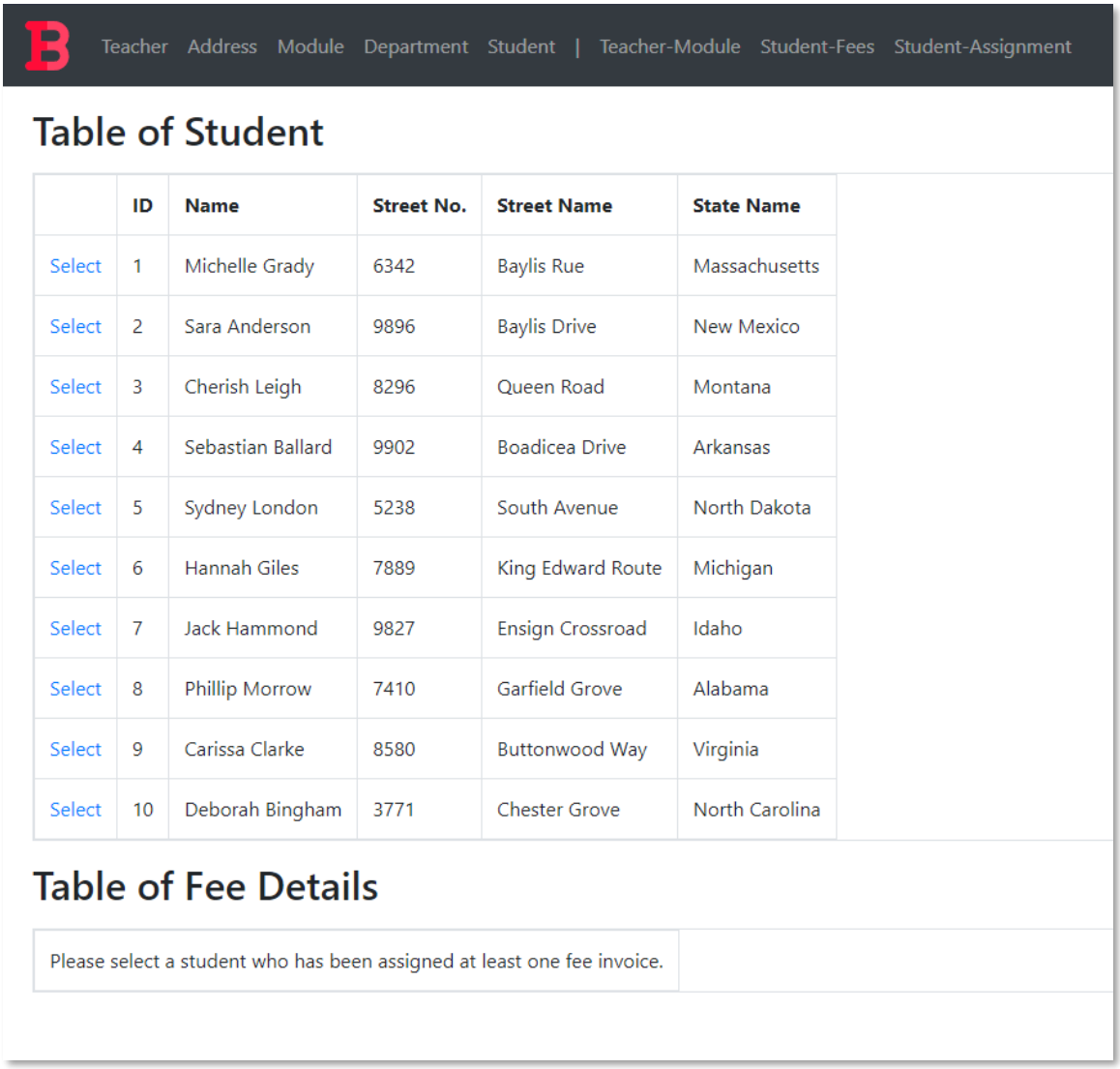
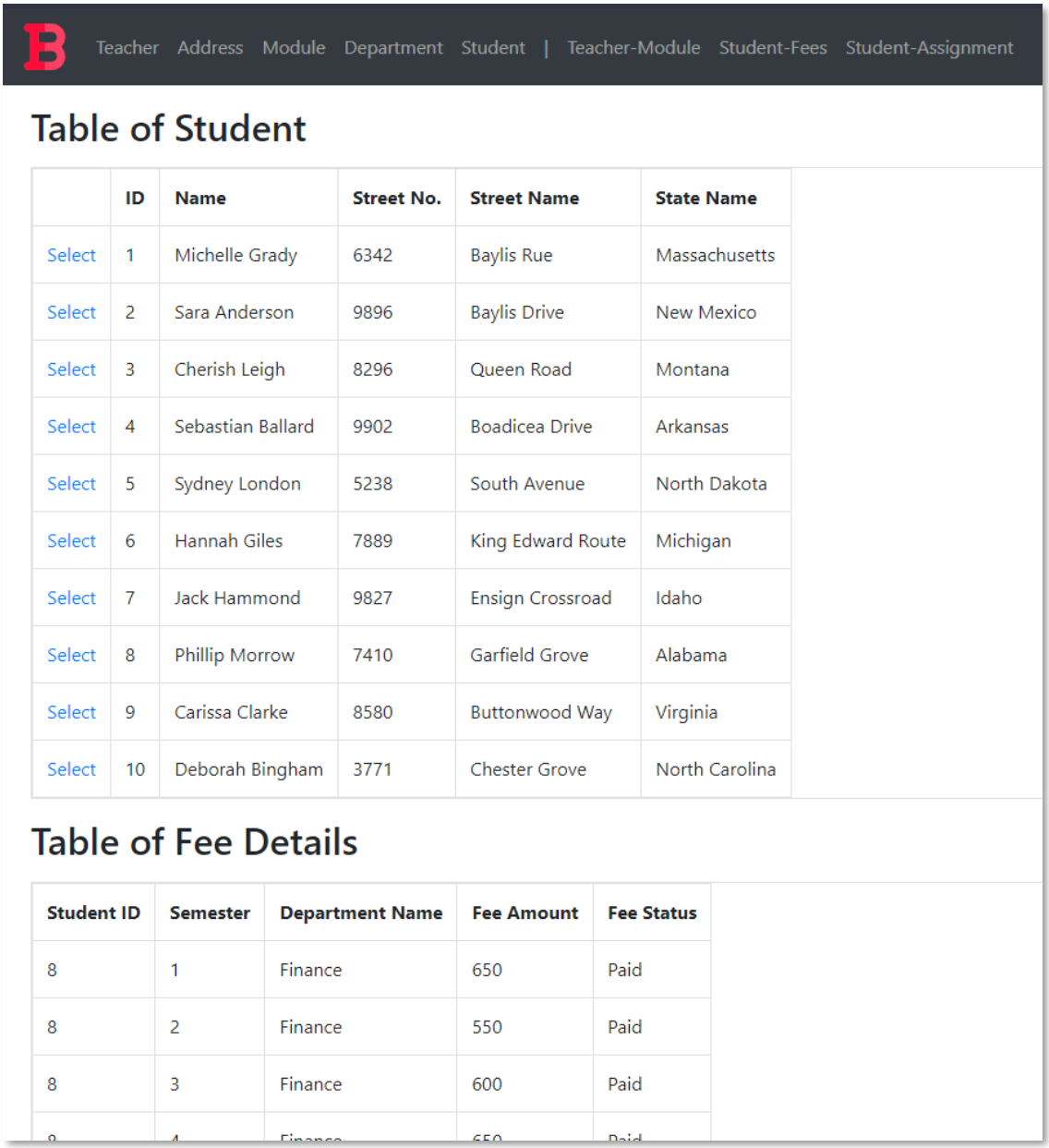


Figure 108: Student-Fees Test 1 evidence

13.2.2.2. Test 2: Data Load on Selection

Test Condition	Expected Outcome	Obtained Outcome
Select a data row from the initial grid view and check if the other grid gets loaded with relevant data.	The other grid is loaded with relevant data.	The other grid was loaded with relevant data.

Table 40: Student-Fees Test 2



	ID	Name	Street No.	Street Name	State Name
Select	1	Michelle Grady	6342	Baylis Rue	Massachusetts
Select	2	Sara Anderson	9896	Baylis Drive	New Mexico
Select	3	Cherish Leigh	8296	Queen Road	Montana
Select	4	Sebastian Ballard	9902	Boadicea Drive	Arkansas
Select	5	Sydney London	5238	South Avenue	North Dakota
Select	6	Hannah Giles	7889	King Edward Route	Michigan
Select	7	Jack Hammond	9827	Ensign Crossroad	Idaho
Select	8	Phillip Morrow	7410	Garfield Grove	Alabama
Select	9	Carissa Clarke	8580	Buttonwood Way	Virginia
Select	10	Deborah Bingham	3771	Chester Grove	North Carolina

Student ID	Semester	Department Name	Fee Amount	Fee Status
8	1	Finance	650	Paid
8	2	Finance	550	Paid
8	3	Finance	600	Paid

Figure 109: Student-Fees Test 2 evidence

13.2.3. Student-Assignment Mapping Form

13.2.3.1. Test 1: Page Load

Test Condition	Expected Outcome	Obtained Outcome
Load the complex form and check if data is loaded in the initial grid.	The data is loaded to the initial grid on page load.	The data was loaded to the initial grid on page load.

Table 41: Student-Assignment Test 1

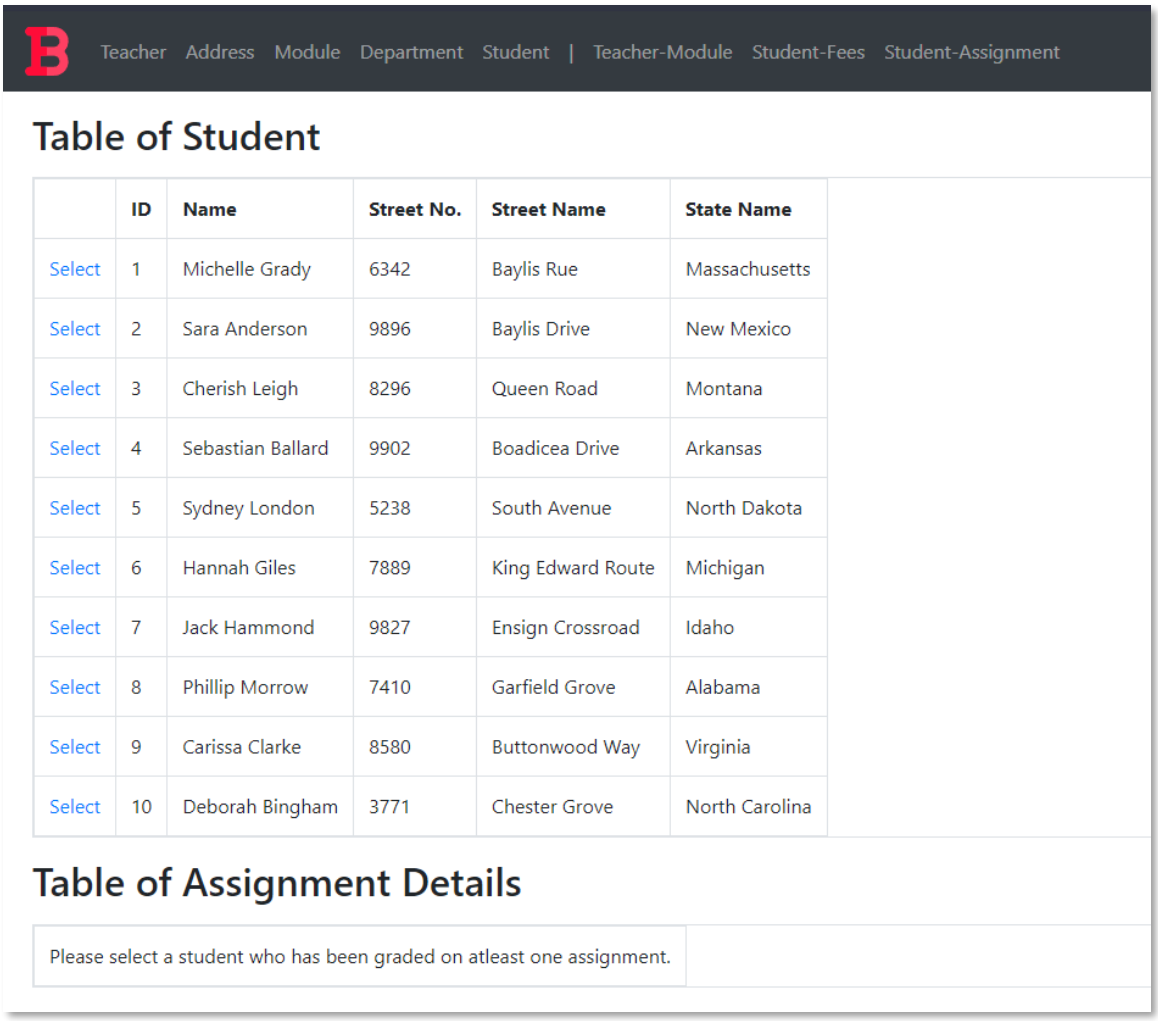


Figure 110: Student-Assignment Test 1 evidence

13.2.3.2. Test 2: Data Load on Selection

Test Condition	Expected Outcome	Obtained Outcome
Select a data row from the initial grid view and check if the other grid gets loaded with relevant data.	The other grid is loaded with relevant data.	The other grid was loaded with relevant data.

Table 42: Student-Assignment Test 2

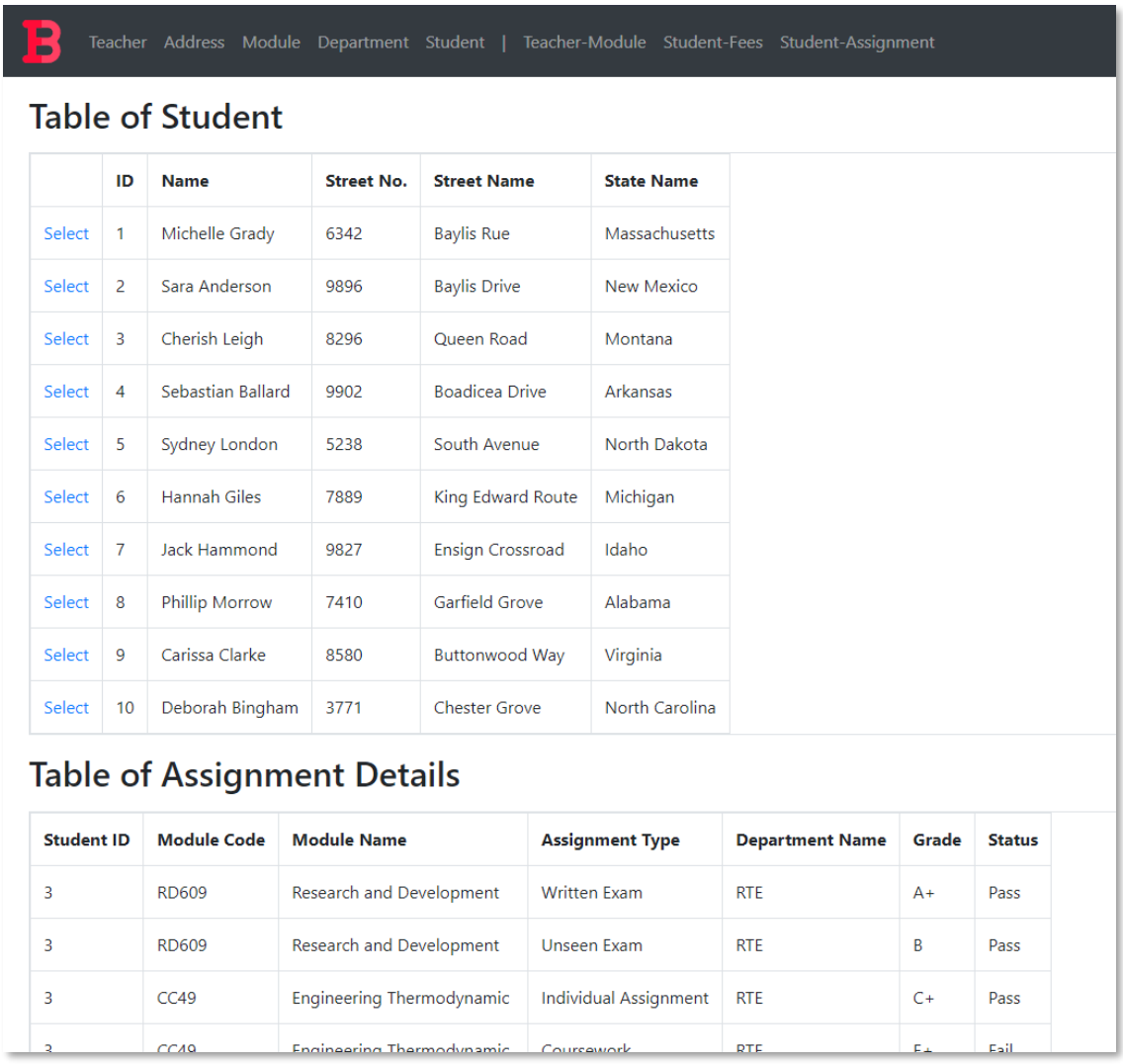


Figure 111: Student-Assignment Test 2 evidence

14. Further Discussion

The project was a simple one, but it provided a lot of insight on oracle databases and dotnet webforms. While the database normalization, integration with assumptions, creation and data population portions were familiar to what we had already been taught, the dotnet webform creation was a unique experience in this project. As dotnet was used previously for another module to create a windows form application, I was already familiar with the dotnet syntax. Similarly, web development was also quite familiar since I had experience working with JavaScript frontend frameworks like React.

However, web development with dotnet was almost completely different than what I had experience with. The project taught me about real world business scenarios. I learned how normalization, being the useful tool that it is, sometimes does not give what we need, about how redundancy must be sometimes added artificially to keep record of things. The project also helped me better understand dotnet as a framework for web development, about how master classes can be used for page layouts, about how libraries like bootstrap can be used to visually enhance a webpage without too much effort and about HTML5 technologies such as input validation. The project also taught me about Oracle SQL sequences and triggers for auto generation of keys, about how Oracle Developer and Oracle Developer Data Modeler can be used to visualize databases, easily edit, or create tables, and generate DDL scripts. It also deepened my understanding about user permissions and Oracle database connection with third parties such as visual studio.

15. Conclusion

The coursework completion fueled my feel confidence to create a database based on real world business scenario to store data in a professional manner. Since the coursework was about creating a database record system and a web application for a college, it has lots of applicability in the real world as most colleges and schools use a database to record their data. Not only schools but a lot of management systems are built the same way with different business scenarios.

All in all, the help from teachers and timely self-research turned the project from what I believe would be intimidating to a simple and fun project that taught me a ton of techniques. While the project was a simple one it taught me more about oracle, dotnet for web development and more about problem solving in general.