Criterion C

Overall Structure of Logic

Normalization

First Normal Form

Student First, Student Last, Student Email, Current Grade, Current School, Siblings, How They Learned About Us, Comments, Internal Comments, Course 1, Course 2, Course 3, Course 4, Semester, Paid, Payment Date, Payment Method

Third Normal Form

Student: Student ID, Student First, Student Last, Student Email, Current Grade, Current School, Siblings, How They Learned About Us, Comments, Internal Comments

Course: Course ID, Course Name, Teacher ID, Course Subject, Course Year, Course Semester, Course Time, Course Cost

Enrollment: Enrollment ID, Student ID, Course ID, Paid

Payment: Payment ID, Student ID, Paid Amount, Payment Date, Payment Method

Teacher: Teacher ID, Teacher First, Teacher Last, Teacher Email

Entity Relationship Diagram

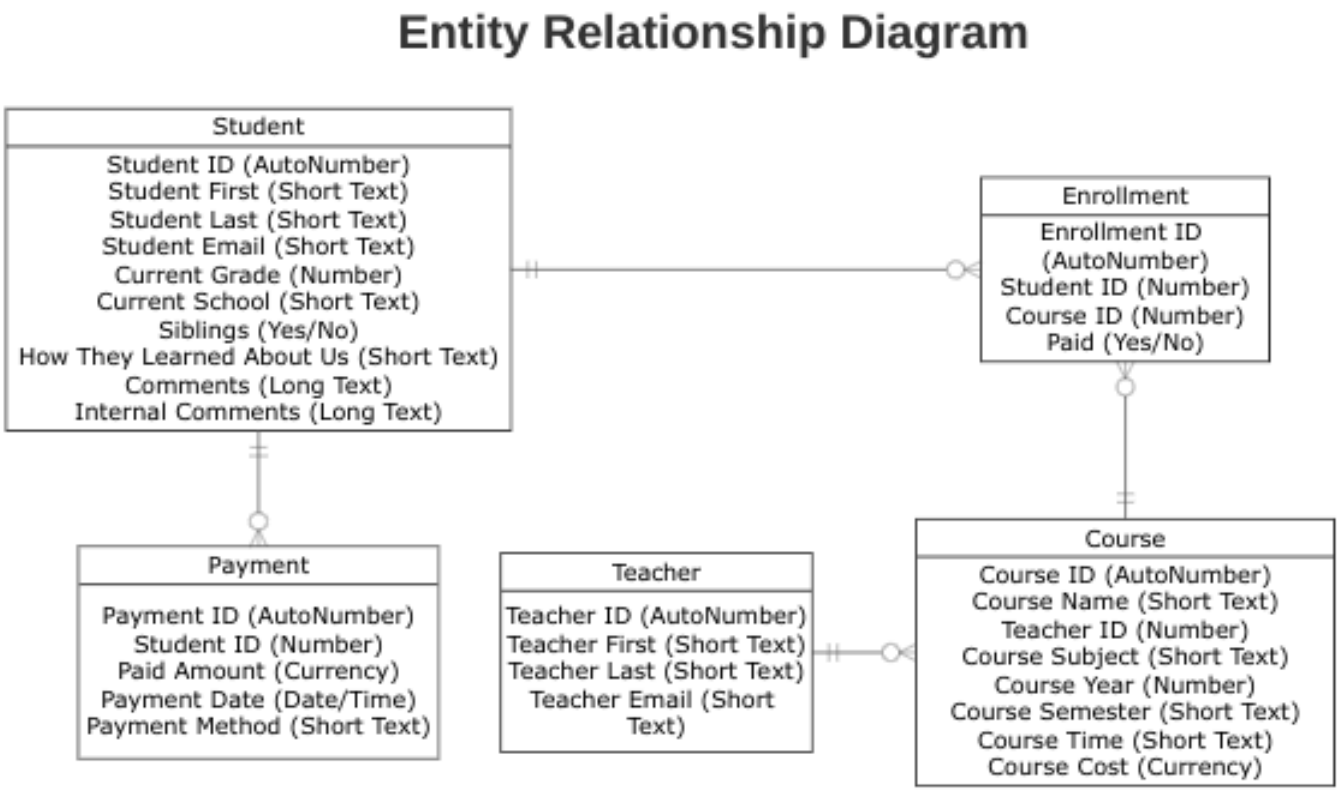


Figure 1: Entity Relationship Diagram

Table Relationship Diagram

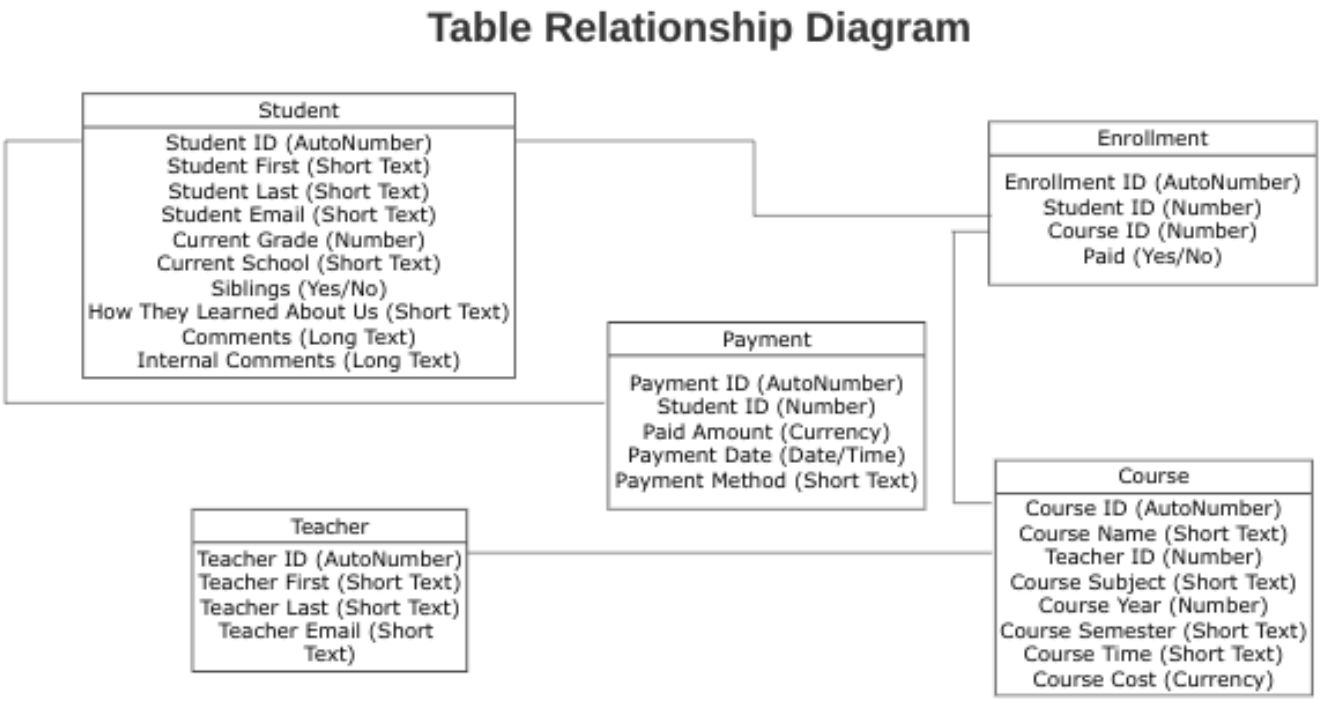


Figure 2: Table Relationship Diagram

Overall Structure of UI

	Home Page		
Pages	Student	Teacher	Administrator
Forms and Reports under each Page	Student Registration	Teacher Registration	Course Creation
	Enrollment	Teacher Assignments	Course History
	Course Catalog		Invoice Generator
			Payment Entry
			Enrollment Summary

Figure 3: User Interface Structure

Design of Individual Features

Adding Records (Student, Teacher, Enrollment, Course, Payment)

The feature of adding records to all tables is necessary for a smooth user experience. It prevents users from needing to directly access tables to add or change information and creates a more appealing view for students, teachers, and administrators. To implement

this feature, I used forms allowing users to input all necessary information to add a new record to a given table, then included a button at the bottom of each form to have a layer of "confirmation" as additions or edits are made to the data. This feature can be seen in the "Course Creation" form, which adds records to the Course table (Figure 4). This is one of five forms total in this database management system that adds records to a table. Once the administrator fills in all course information and clicks "Add Course", a new course will be added to the Course Table. In Figure 5, it can be seen that the course information is successfully added to the Course table.

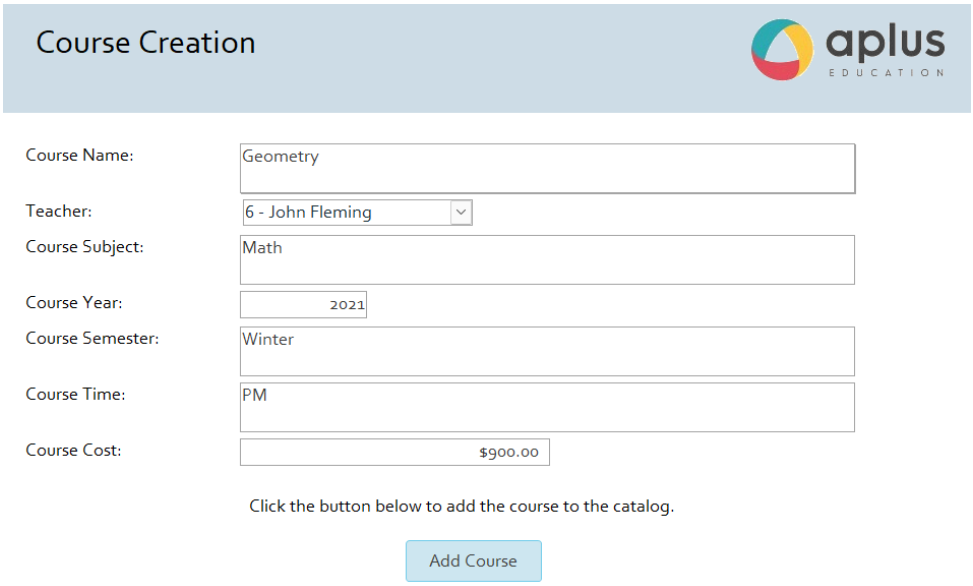


Figure 4: Course Creation (Adding Records)

Course ID	Course Name	Teacher ID	Course Subje	Course Year	Course Seme	Course Time	Course Cost
1	Critical Reading/Writing - Grade 5	3	English	2022	Summer	AM	$900.00
2	SAT/PSAT - English	2	English	2022	Summer	AM	$1,000.00
3	Grammar 2	7	English	2022	Summer	PM	$900.00
4	Reading & Vocab 1	3	English	2022	Summer	PM	$850.00
5	Geometry	6	Math	2022	Summer	AM	$900.00
6	Math 5	4	Math	2022	Summer	AM	$850.00
7	Critical Reading/Writing - Grade 3	7	English	2022	Summer	AM	$850.00
8	Composition 1	8	English	2022	Summer	PM	$900.00
9	SAT/PSAT - English	2	English	2021	Summer	AM	$1,000.00
10	SAT/PSAT - Math	9	Math	2021	Winter	PM	$1,000.00
11	Geometry	6	Math	2021	Winter	PM	$900.00

Figure 5: Course Table (Adding Records)

The presence of a button also fulfills the idea of protection against data loss; there is an extra "confirmation" before data is added or changed. This makes this implementation of this feature ideal. Figure 6 highlights the logic behind the functionality of the button, where the command to add a new record and display an error message if there is a mistake or incomplete information.

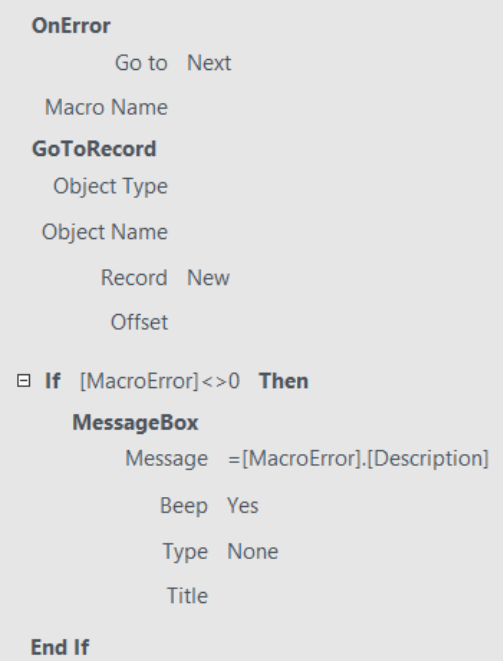


Figure 6: On Click of "Add Course" Button

Course Catalog

The course catalog feature aims to display all courses offered by APLUS Learning Center, organized by semester. It is necessary as a tool for students to view all the offerings that APLUS has in their process of enrolling in classes. The "Course Catalog" report uses the "Course Query" query as a source, utilizing the concatenated Course Year and Course Semester as category headings and listing Course Names underneath their respective semesters. The SQL code for the "Course Query" query is found in Figure 7.

```
SELECT [Course Name] & " (" & [Course Semester ] & " " &
[Course Year] & ", " & [Course Time] & ")" & " - $" & [Course
Cost] AS [Catalog], Course.[Course ID], [Course Year] & " " &
[Course Semester] AS Semester, Course.[Course Name]
FROM Course;
```

Figure 7: Course Query (SQL Code)

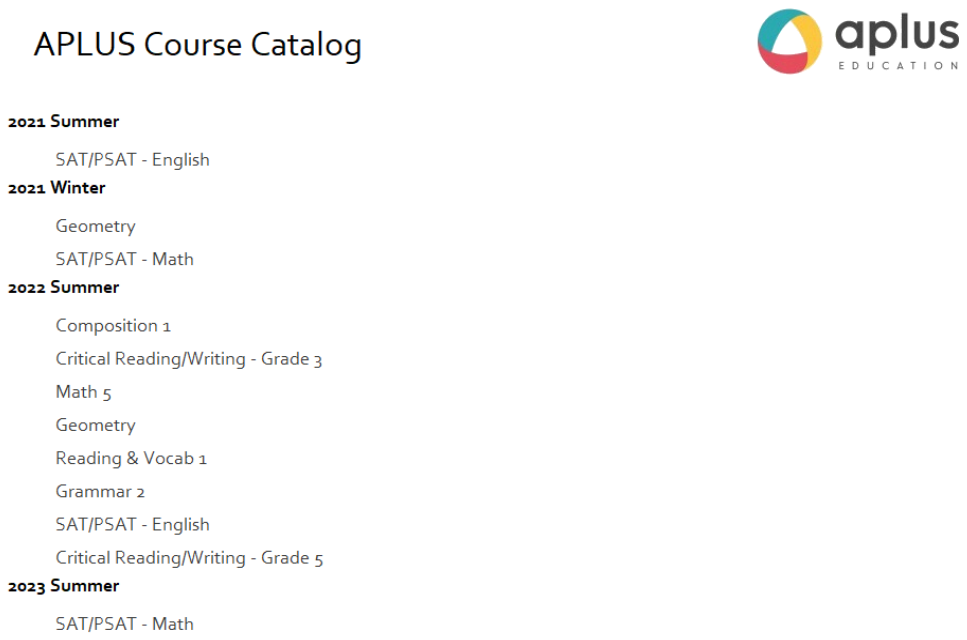


Figure 8: Course Catalog Report

Course History

The course history feature aims to display the course history of a selected student, organized by semester. This is a necessary feature as it addresses a client request (success criterion) to be able to view the history of a student's classes in an intuitively generated report that is easy to access. Because of its simple design, it allows for a birds-eye overview of a student's history at APLUS. The report is generated through a form and a query. A form where users can select a student (Figure 9). This feeds into the "Course History" query, which generates the selected student's enrollment information from the enrollment table: Course Semester, Course Name, Student ID, Student First, and Student Last. Figure 10 highlights the logic of this segment in SQL form. This is the most appropriate technique because it links the form with the report seamlessly through a query acting as the "middle man".

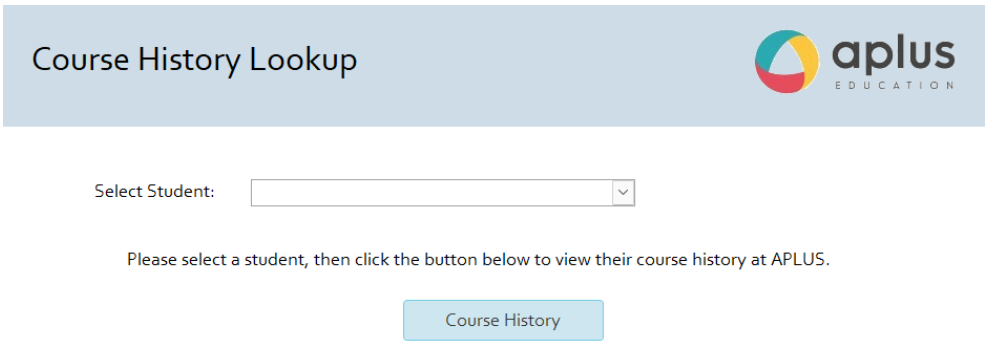


Figure 9: Course History Form

```
SELECT Student.[Student First], Student.[Student Last], [Course Semester] & " " & [Course Year] AS Semester,
Course.[Course Name]
FROM Course INNER JOIN (Student INNER JOIN Enrollment ON Student.[Student ID] = Enrollment.[Student ID])
ON Course.[Course ID] = Enrollment.[Course ID]
GROUP BY Student.[Student First], Student.[Student Last], [Course Semester] & " " & [Course Year],
Course.[Course Name], Student.[Student ID]
HAVING (((Student.[Student ID])=[Forms]![Course History (Admin Only)]![cbochstudent]));
```

Figure 10: Course History Query (SQL Code)

Upon clicking the “Course History” button, a report is opened which sources its data from the “Course History” query. This report categorizes courses by semester and displays the selected student’s first and last name at the top of the screen. A sample report is seen in Figure 11.

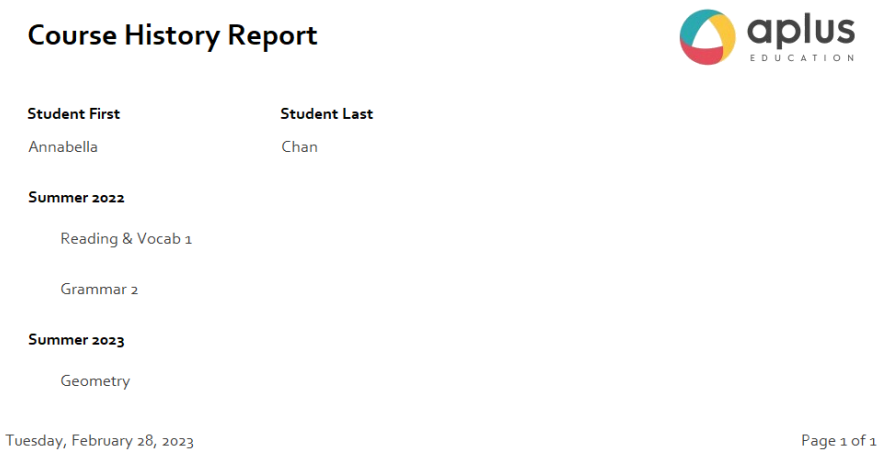


Figure 11: Course History Report

Enrollment Summary

The enrollment summary feature aims to display the number of students enrolled in each class in the current year. The report displayed simply shows the number of enrolled students next to the name of the class, and highlights classes with more than four people in green, and classes with less than four people in red. This allows for administrators to conveniently access an at-a-glance view of how many classes are qualified to run. The report is generated using data from the “Enrollment Summary” query, which calculates the number of students enrolled in each class from the Course and Enrollment tables. The SQL code used for this query can be viewed in Figure 12.

```
SELECT Course.[Course ID], Count(Enrollment.[Enrollment ID]) AS Enrolled, [Course Query].Catalog,
Course.[Course Year]
FROM ([Course Query] INNER JOIN Course ON [Course Query].[Course ID] = Course.[Course ID]) INNER JOIN
Enrollment ON Course.[Course ID] = Enrollment.[Course ID]
GROUP BY Course.[Course ID], [Course Query].Catalog, Course.[Course Year]
HAVING (((Course.[Course Year])=Year(Now())));
```

Figure 12: Enrollment Summary (SQL Code)

The query also pulls the course names from the "Course Query", which simplifies the process of generating the full name of a course. For example, instead of having to type out the concatenation of the Course Name, Course Semester, Course Year, and Course Cost, the query sourced this text from a pre-existing query. The final report is shown in Figure 13. Because this is meant to be an overview, I chose to go directly from a query to a report (rather than use form input) because the report is meant to show the same information every time.

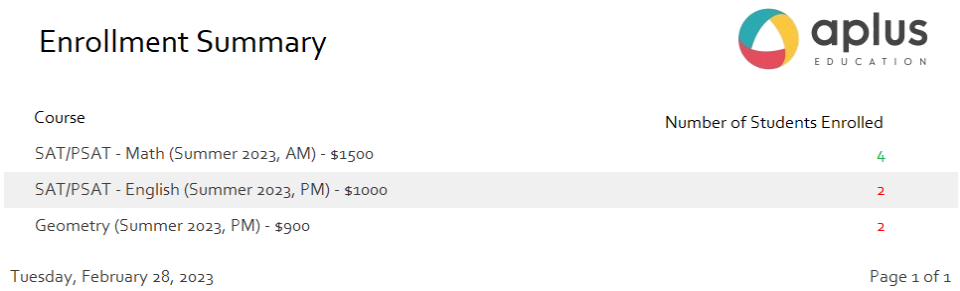


Figure 13: Enrollment Summary Report

Invoice

The invoice feature draws directly from the data to generate an invoice for a selected student, including the cost for each class, discount offered, and total cost. The invoice also offers a customized and aesthetically pleasing format for both administrators and students. Most importantly, it fulfills the success criterion of having the ability to easily generate invoices for a student. Like the Course History Report, the Invoice Report draws from an Invoice Query, which takes into account a selected student as input from the "Invoice Generator" form. A screen capture of the "Invoice Generator" form is seen below in Figure 14. There is a combo box available to select a student, which automatically runs the Invoice Query, and upon clicking "Generate Invoice", the Invoice Report is opened. The logic behind the button's function can be seen in Figure 15.

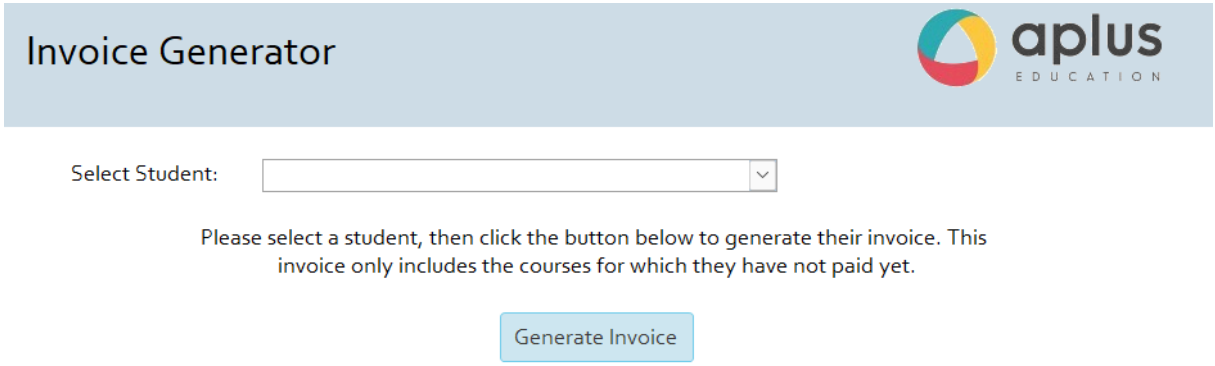


Figure 14: Invoice Generator Form

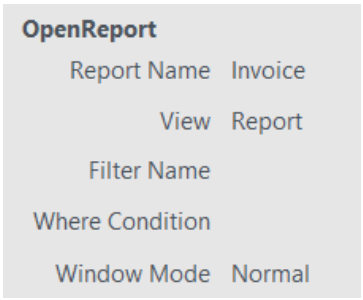
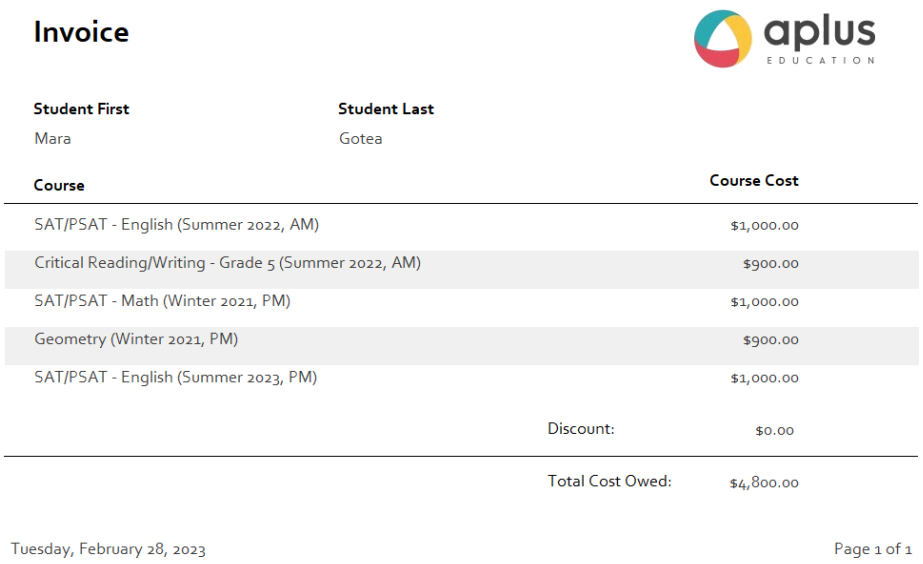


Figure 15: “Generate Invoice” Button Logic

A screen capture of the final report view is shown below in Figure 16. It lists the student name, along with course names and costs, a discount, and total cost owed.



Invoice

aplus EDUCATION

Student First	Student Last
Mara	Gotea

Course	Course Cost
SAT/PSAT - English (Summer 2022, AM)	$1,000.00
Critical Reading/Writing - Grade 5 (Summer 2022, AM)	$900.00
SAT/PSAT - Math (Winter 2021, PM)	$1,000.00
Geometry (Winter 2021, PM)	$900.00
SAT/PSAT - English (Summer 2023, PM)	$1,000.00
Discount:	$0.00
Total Cost Owed:	$4,800.00

Tuesday, February 28, 2023 Page 1 of 1

Figure 16: Invoice Report

Teacher Assignments

The teacher assignments report is generated to list the courses that each teacher is teaching under their name. Only courses taught in the current year appear in this report so as to display only relevant information. This report is not only to make it easier for teachers to view what classes they have been assigned, but also to contain all teacher assignments in a designated area. This also goes to fulfill the success criterion of having the ability to easily view teacher assignments. This feature works through a query that pulls information from the Teacher and Course tables and filters out classes of years other than the current one. Because this is meant to be an overview, I chose to go directly from a query to a report (rather than use form input) because the report is meant to show the same information every time. The design view and SQL view of the logic of the query

can be seen in Figures 17 and 18, respectively. Finally, a view of the Teacher Assignments Report can be seen in Figure 19.

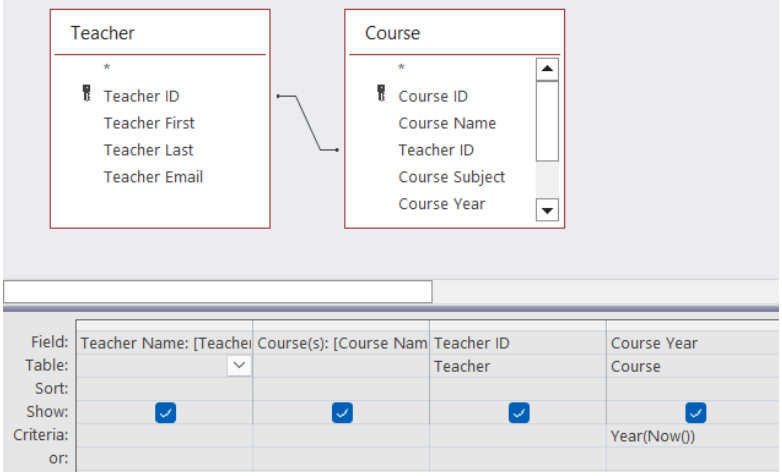


Figure 17: Teacher Assignments Query Design View

```
SELECT [Teacher First] & " " & [Teacher Last] AS [Teacher Name], [Course Name] & " (" & [Course Semester ] & "
" & [Course Year] & ", " & [Course Time] & ")" AS [Course(s)], Teacher.[Teacher ID], Course.[Course Year]
FROM Teacher INNER JOIN Course ON Teacher.[Teacher ID] = Course.[Teacher ID]
WHERE (((Course.[Course Year])=Year(Now())));
```

Figure 18: Teacher Assignments Query SQL View

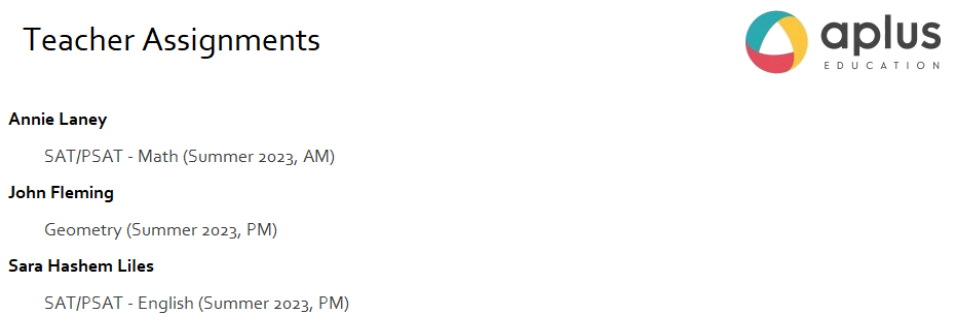


Figure 19: Teacher Assignments Report