WENZHOU-KEAN UNIVERSITY

Final Project

CPS3740: Database Management System

Student Name:	Chun Chen
Student ID:	1234930
Lecture:	Dr. Hemn Barzan Abdalla

Fall 2025

1. Abstract

This project built a simple system for students' university course scheduling and grade calculation. The system integrates the course scheduling and grade calculation systems and uses triggers to automatically record grade changes, ultimately enabling students to quickly manage their studies.

2. Introduction

Background: The lack of integration between the student course scheduling system, grade calculation system, and course syllabus creates inconvenience for students' course selection planning.

Problem: Students often need to switch between different systems to make a final decision, and this separation often hinders the decision-making process. For example, some courses require a prerequisite grade of >=2.4 or be in progress before selection is allowed, but the separation of the grading system and course selection system prevents users from quickly determining whether they meet the requirements.

This project primarily targets **students with course selection needs**, aiming to enable them to **quickly make decisions about course scheduling**. And also export GPA csv

3. System Requirement

- ## Functional Requirements

Frontend:

a) A rapid presentation of the courses in the four-year plan.

b) Quickly view prerequisite courses for selected courses.

c) Enter or modify course grades.

d) Record course grade entry or modification.

e) Unlock courses that have met prerequisite requirements.

f) Delete courses in courses table.

Backend:

a) Quick entry of course schedules

b) Quick entry of prerequisite course relationship tables

c) Two triggers for recording changes in course grades

- ## Non-Functional Requirements

a) Quick interaction with the UI

b) A simple and logically arranged UI

4. System Design

● System Architecture

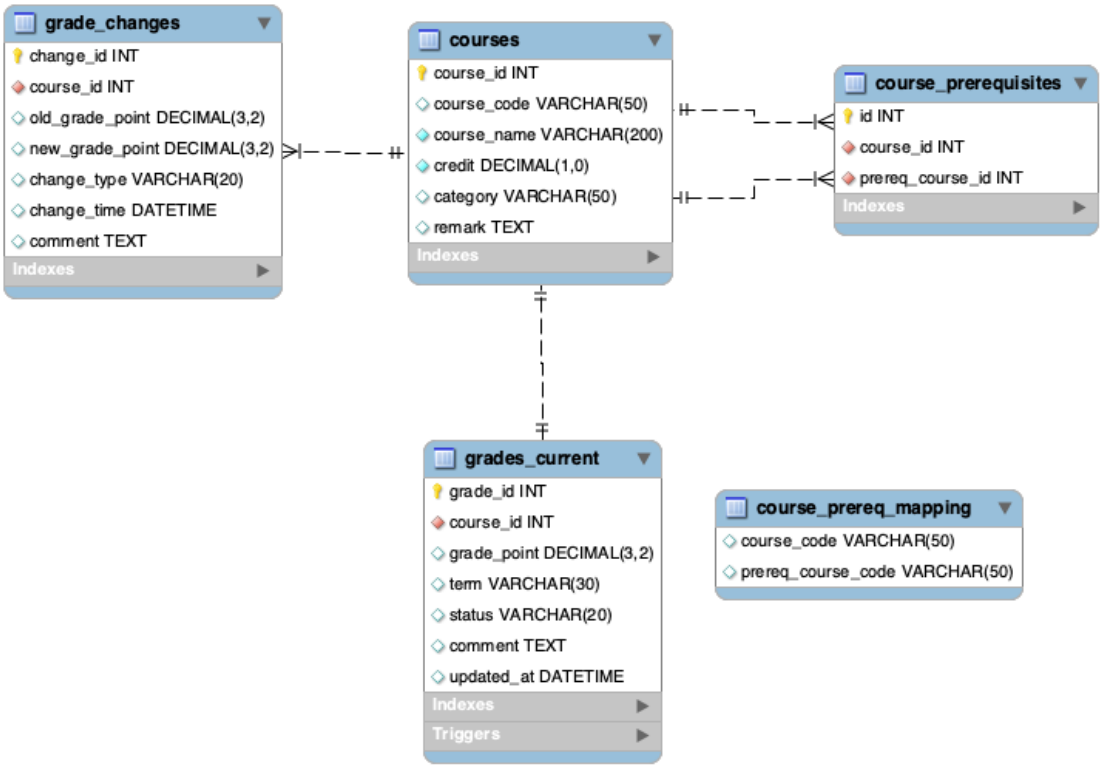
Python(Tkinter) for GUI:

Class_gpa_management_app.py

SQLWorkbench for localhost:

Class_GPA_Management_SYS_db. Sql

● Database Design



Entity Relationship (ER) Modeling in Ternary relationship
Class_GPA_Management_SYS_db

courses table:

The main table in the schema, with a non-identifying relationship with three other tables, the **course_id** key is a primary key in the courses table, and other attributes in the courses key provide other relevant information about the course is described.

course_code: The course's Abbreviation in the four-year plan, like 'ESL3035'.

course_name: The course's name(English description) in the four-year plan, like 'ESL 0303 Academic oral presentation I'.

credit: The GPA credits scope between 0 to 9.

category: This time, only divide the category between major and required.

Text: Other word description for this course.

The **relationship** between the other tables:

For the course_prerequisites table: One course can have one or many course_id, and prereq_course_id in the prerequisite courses table, and each course_id and prereq_course_id have relevant course_id in the courses table.

For grades_current table: One course can have one grades_current record and one grades_current record only related to one course.

For grade_changes table: One course can have multiple GPA change records, and one change record can only have one course of information.

grades_current table:

we have a primary key **grade_id** and a Foreign key **course_id**. The other attributes in this table represent the grade and credit for the course.

grade_point: the GPA between (0.00 – 9.00).

term: the semester we take the course.

status: the type contains (normal/ retake/ correction/ in progress).

comment: the comment for this grade.

updated_at: the time we add this grade.

We also added two triggers to record the changes:

When inserting a new grade into the **grades_current table**, the **Insert_trigger** catches the change into **grade_changes.**

When updating a grade in **grades_current table,** the **Update_trigger** catches the change into **grade_changes**.

The difference between the two triggers is that we do not record **old_grade_point** into **grade_changes table** because we do not have old GPA before.

grade_changes table:

This table records the grade change for each course from the trigger. The primary key change_id, with Foreign key course_id.

old_grade_point: the old GPA before(if no GPA before enter 'NULL').

new_grade_point: the new GPA when we Normal/retake/correction/In progress the class.

change_type: the four statuses: Normal/retake/correction/In progress for the course.

change_time: the time we record the change.

course_prerequisites table:

This table records between the course and the prerequisite course, the primary key is 'id', and the two foreign keys are 'course_id' and 'prereq_course_id', which reference to courses table.

course_prereq_mapping table:

This table helps us have a quick correspondence between the course and the prerequisite course.

- # GUI Design

The GUI design contains three parts

| ----- UI Construction (Construct the Arrange the specific images and button layout of the user interface.)

| ----- Event handlers (Define the programs that different buttons in the UI can trigger.)

| ----- Refresh functions (Define the specific statements of the program required for the response button.)

5. Implementation

- ## Class_GPA_Management_SYS_db. Sql

  First, we create and use the schema:

```
1  CREATE DATABASE IF NOT EXISTS Class_GPA_Management_SYS_db;
2  USE Class_GPA_Management_SYS_db;
```

  Second, we create five tables:

```
5   CREATE TABLE IF NOT EXISTS courses (
20  CREATE TABLE IF NOT EXISTS grades_current (
36  CREATE TABLE IF NOT EXISTS grade_changes (
53  CREATE TABLE IF NOT EXISTS course_prerequisites (
182 CREATE TABLE IF NOT EXISTS course_prereq_mapping (
```

  with two triggers:

```
72      CREATE TRIGGER insert_trigger
73      AFTER INSERT ON grades_current

94      CREATE TRIGGER update_trigger
95      BEFORE UPDATE ON grades_current
```

  Then, we insert values into the courses and course_prereq_mapping table to

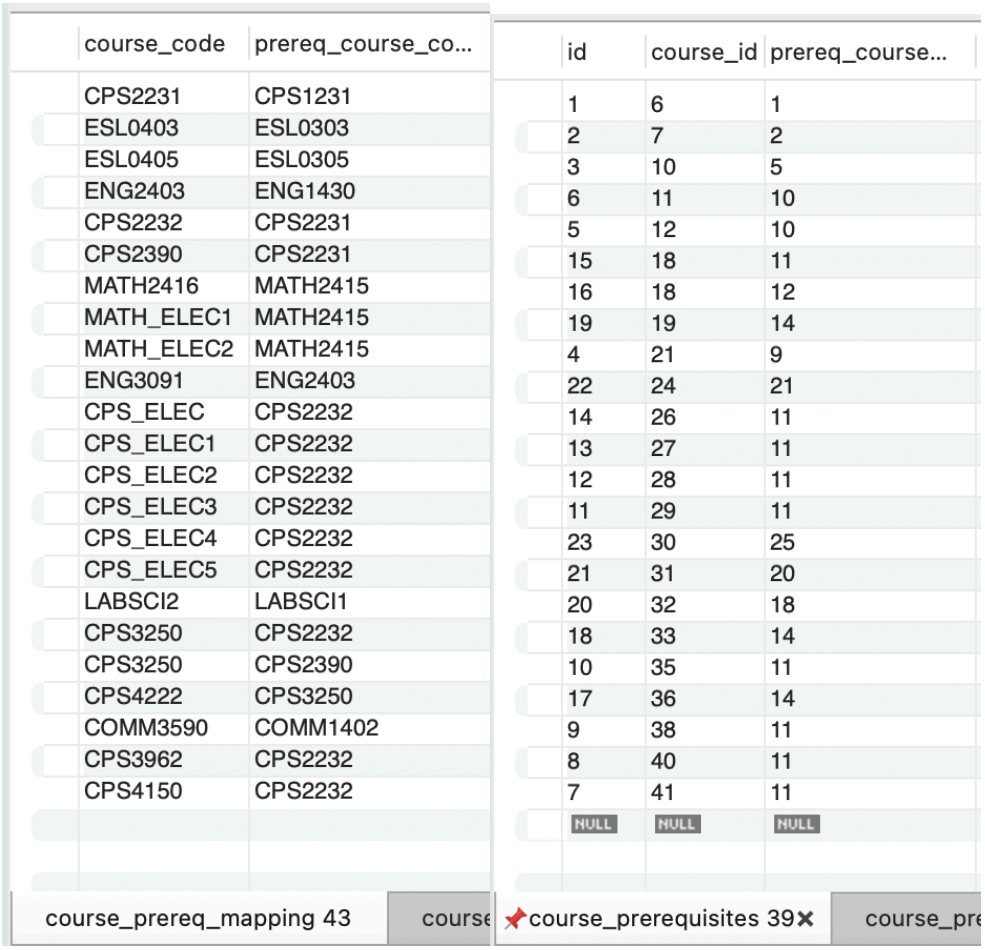
  construct a basic course framework

```
128  INSERT INTO courses (course_code, course_name, credit, category)
129  VALUES

187  INSERT INTO course_prereq_mapping (prereq_course_code,course_code) VALUES
```

Finally, we select all the tables attributes

```
224 •   select * from course_prerequisites;
225 •   select * from courses;
226 •   select * from grade_changes;
227 •   select * from grades_current;
228 •   select * from course_prereq_mapping;
```



course_code	prereq_course_co...
CPS2231	CPS1231
ESL0403	ESL0303
ESL0405	ESL0305
ENG2403	ENG1430
CPS2232	CPS2231
CPS2390	CPS2231
MATH2416	MATH2415
MATH_ELEC1	MATH2415
MATH_ELEC2	MATH2415
ENG3091	ENG2403
CPS_ELEC	CPS2232
CPS_ELEC1	CPS2232
CPS_ELEC2	CPS2232
CPS_ELEC3	CPS2232
CPS_ELEC4	CPS2232
CPS_ELEC5	CPS2232
LABSCI2	LABSCI1
CPS3250	CPS2232
CPS3250	CPS2390
CPS4222	CPS3250
COMM3590	COMM1402
CPS3962	CPS2232
CPS4150	CPS2232

course_prereq_mapping 43 | course

id	course_id	prereq_course...
1	6	1
2	7	2
3	10	5
6	11	10
5	12	10
15	18	11
16	18	12
19	19	14
4	21	9
22	24	21
14	26	11
13	27	11
12	28	11
11	29	11
23	30	25
21	31	20
20	32	18
18	33	14
10	35	11
17	36	14
9	38	11
8	40	11
7	41	11
NULL	NULL	NULL

course_prerequisites 39 | course_pre

course_id	course_code	course_name	credit	category	remark
1	ESL0303	ESL 0303 Academic oral presentation I	3	Required	NULL
2	ESL0305	ESL 0305 Academic written terminology I	3	Required	NULL
3	GE1000	GE 1000 Transition to Kean	1	Required	NULL
4	ENG1300	ENG 1300 English essays by non-native speak…	6	Required	NULL
5	CPS1231	CPS 1231 Foundations of Computer Science	4	Major	NULL
6	ESL0403	ESL 0403 Academic oral presentation II	3	Required	NULL
7	ESL0405	ESL 0405 Academic written terminology II	3	Required	NULL
8	MATH2110	MATH 2110 Discrete structure	3	Required	NULL
9	ENG1430	ENG 1430 English essays by non-native speak…	6	Required	NULL
10	CPS2231	CPS 2231 Computer Programming	4	Major	NULL
11	CPS2232	CPS 2232 Data Structures	4	Major	NULL
12	CPS2390	CPS 2390 Organization and Structure	3	Major	NULL
13	MATH2995	MATH 2995 Matrices and Linear Algebra	3	Required	NULL
14	MATH2415	MATH 2415 Calculus I	4	Required	NULL
15	GE2024	GE 2024 Research and Technology	3	Required	NULL
16	HIST1062	HIST 1062 World History	3	Required	NULL
17	TECH2920	TECH 2920 Computer System	3	Major	NULL
18	CPS3250	CPS 3250 Computer operating system	3	Major	NULL
19	MATH2416	MATH 2416 Calculus II	4	Required	NULL
20	COMM1402	COMM 1402 Verbal communication	3	Required	NULL
21	ENG2403	ENG 2403 English essays by non-native speak…	3	Required	NULL
22	CPS3440	CPS 3440 Algorithm Analysis	3	Major	NULL
23	MATH2526	MATH 2526 Applied Statistics	3	Required	NULL
24	ENG3091	ENG 3091 Technical writing	3	Required	NULL
25	LABSCI1	Lab Science I	4	Required	NULL
26	CPS_ELEC	CPS Elective 3xxx/4xxx	3	Major	NULL

courses 40 | course_prerequisites 39 | course_prereq_mapping 43 | grade_changes

The grades_current and grade_changes tables no value available at the moment.

- # Class_gpa_management_app. Py

In the program, we defined a function to let other functions have a quick connection with the database.

```
8   # --------------- used to quick connect with datatbase --------------------------
9   def get_connection():
10
11      return pymysql.connect(
12          host='localhost',
13          user='root',
14          password='12345678',
15          database='Class_GPA_Management_SYS_db',
16          charset='utf8mb4',
17      )
```

The most basic operating logic is:

a) we defined several functions to get a connection with the database and write SQL code to deal with different Event handlers.

b) Different methods are applied to handle different refresh requests.

c) Different update requests were used to process different entries and UI commands.

For example, we define the fetch_unlocked_courses function to connect with the database and **tuple** the courses that satisfy the prerequisite courses' GPA >= 2.4 or the status = In progress, and the course grade must is NULL.

```
# ---------------- return the unlock course row(tuple) with a list satisfy
# ---------------- prerequisite courses GPA >= 2.4 or 'InProgress'
# ---------------- and the grades_current.grade_id = NULL means no course record before
def fetch_unlocked_courses():
    conn = get_connection()
    cur = conn.cursor()
    cur.execute("""
    SELECT
        c.course_id,
        c.course_code,
        c.course_name,
        c.credit
    FROM
        courses AS c
    LEFT JOIN
        course_prerequisites AS p
        ON c.course_id = p.course_id
    LEFT JOIN
        grades_current AS g_pre
        ON p.prereq_course_id = g_pre.course_id
    LEFT JOIN
        grades_current AS g_self
        ON c.course_id = g_self.course_id
    WHERE
        g_self.grade_id IS NULL
    GROUP BY
        c.course_id
    HAVING
        COUNT(p.prereq_course_id) =
        SUM(
            CASE
                WHEN p.prereq_course_id IS NULL THEN 1
                WHEN g_pre.grade_point >= 2.4 OR g_pre.status = 'InProgress' THEN 1
                ELSE 0
            END
        )
    ORDER BY
        c.course_id;
    """)
    rows = cur.fetchall()
    conn.close()
    return rows
```

And in one refresh function we use tuple return from fetch_unlocked_courses function to show the content.

```
401     def refresh_unlocked_courses(self):
402         for item in self.tree_unlock.get_children():
403             self.tree_unlock.delete(item)
404         rows = fetch_unlocked_courses()
405         for (cid, code, name, credit) in rows:
406             self.tree_unlock.insert(
407                 "", END, iid=str(cid),
408                 values=(code or "", name, float(credit) if credit is not None else "")
409             )
```

```
378     def refresh_all(self):
379         self.refresh_course_table()
380         self.refresh_unlocked_courses()
381         self.refresh_grade_history()
382         self.refresh_prereq_table()
```

Here is the command that we call the refresh command.

```
250         btn_refresh = Button(left_frame, text="Refresh", command=self.refresh_all)
251         btn_refresh.pack(pady=5)
```

Accessibility Features:

In SQL we define functions to insert course information and prerequisite courses information individually.

Also, some functions to drop values in the schema.

```
231     # --------------------------- delete function --------------
232     # delete four tables
233     -- TRUNCATE table course_prerequisites;
234     -- TRUNCATE table grade_changes;
235     -- TRUNCATE table grades_current;
236     -- TRUNCATE table course_prereq_mapping;
237
238     # drop all the row in courses table
239     -- SET SQL_SAFE_UPDATES = 0;
240     -- DELETE FROM courses;
241     -- ALTER TABLE courses AUTO_INCREMENT = 1;
242     -- SET SQL_SAFE_UPDATES = 1;
243
244
245     # ------------------ insert course or prerequisites course individual
246     # assume we don't konw the individual id in courses table we only
247     # konw the relationship between each course
248     -- INSERT INTO course_prerequisites (course_id, prereq_course_id)
249     -- SELECT
250     --        course.course_id,
251     --        pre.course_id
252     -- FROM courses AS pre, courses AS course
253     -- WHERE pre.course_code = 'CPS2232' AND course.course_code = 'CPS4150';
254
255     # insert value into courses
256     # INSERT INTO courses (course_code, course_name, credit, category)
257     # VALUES('','',0,'');
258
259     # don't run it
260     # DROP schema Class_GPA_Management_SYS_db;
```

In the GUI, we also built some functions to deal with some illegal requests.

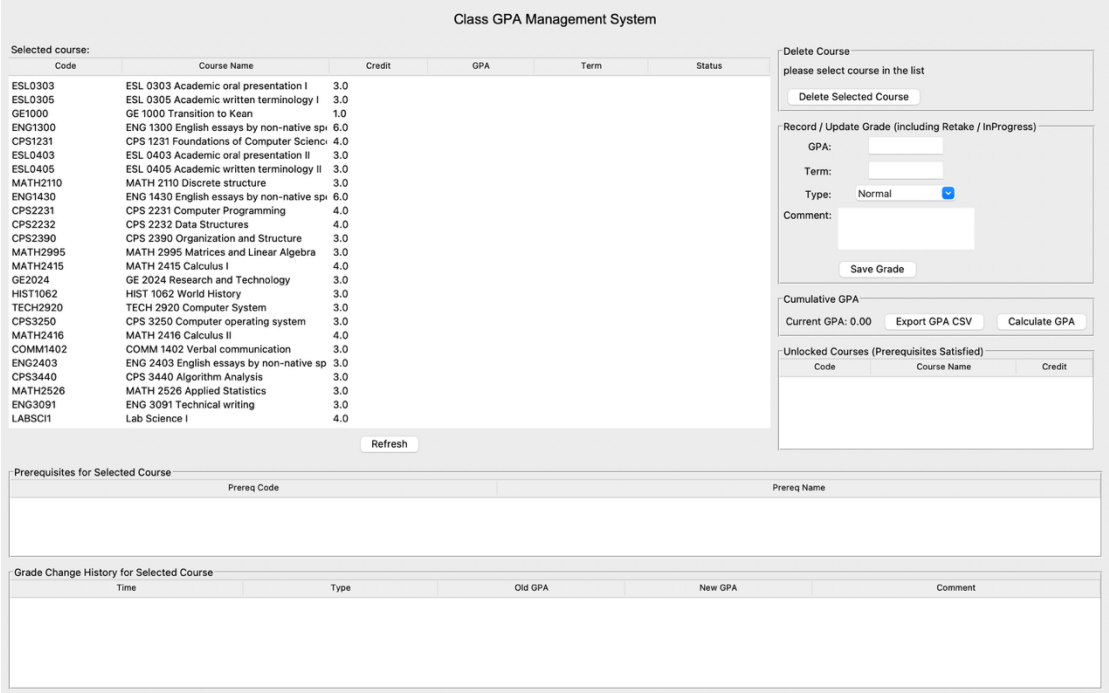
```
484         if not name or not credit_str:
485             messagebox.showwarning("Warning", "Course name and credit are required.")
486             return
```

6. Testing

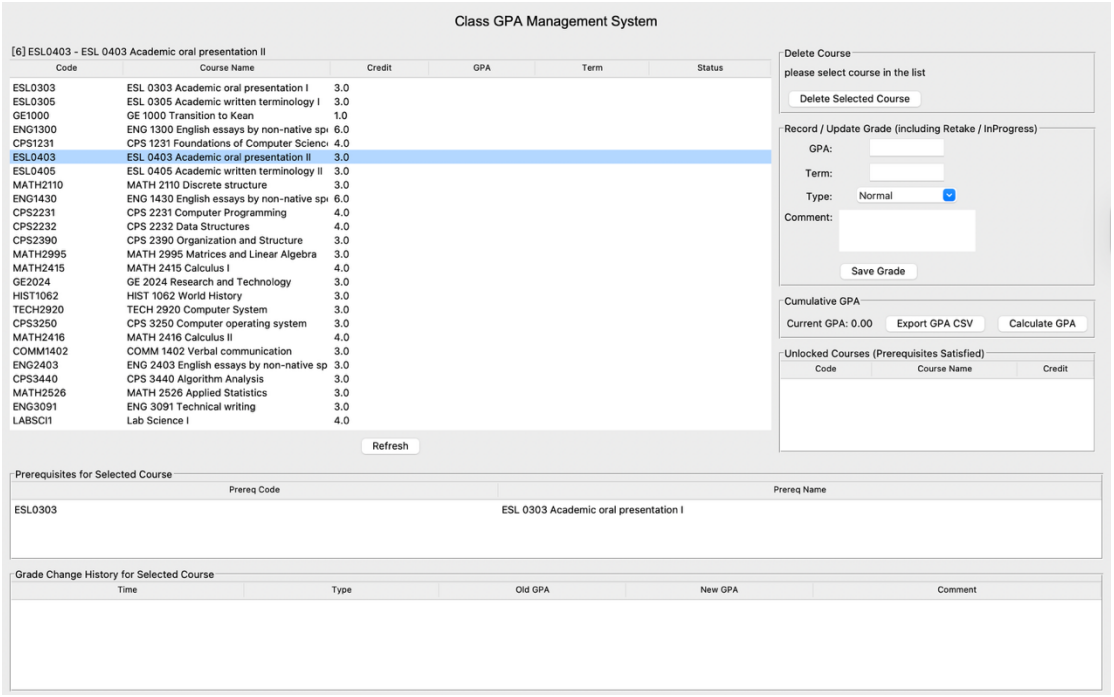
We first start the workbench, then we run the code in the terminal:

```
(base) chenchenchun@chenchenchundeMacBook-Air-2 Cps 3740 % python class_gpa_mana
gement_app.py
```

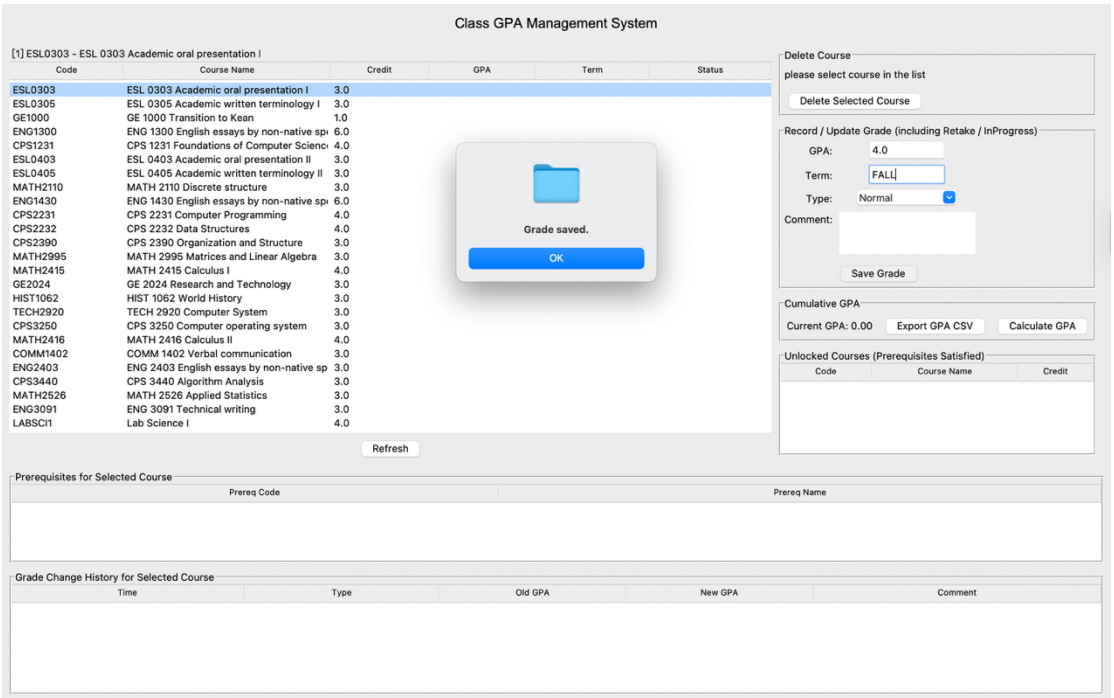
Then we get the GUI with courses and grades_current list:

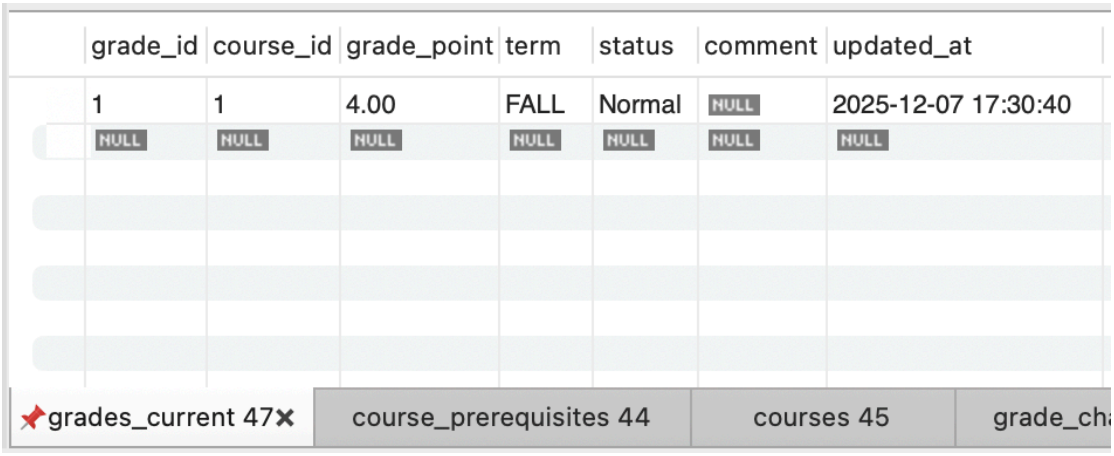


If we select one course, we will get the prerequisites and grade change history for the selected course.



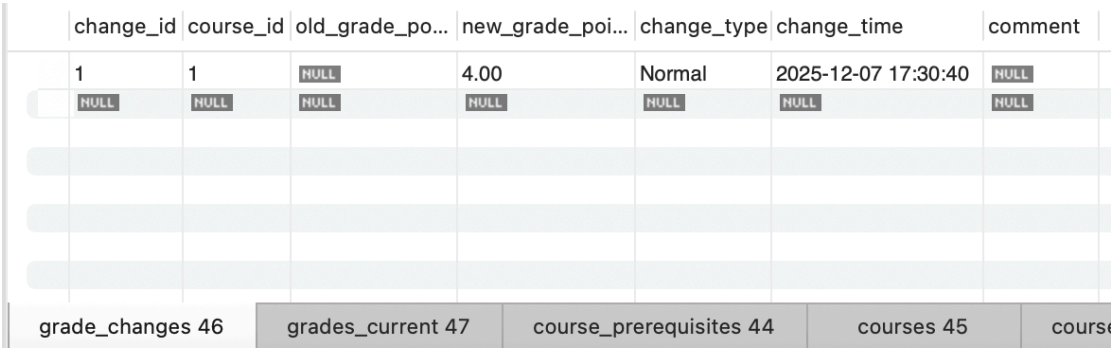
After we enter the grade into the GUI we save the data into grades_current and grade_changes(by trigger),





grade_id	course_id	grade_point	term	status	comment	updated_at
1	1	4.00	FALL	Normal	NULL	2025-12-07 17:30:40
NULL	NULL	NULL	NULL	NULL	NULL	NULL

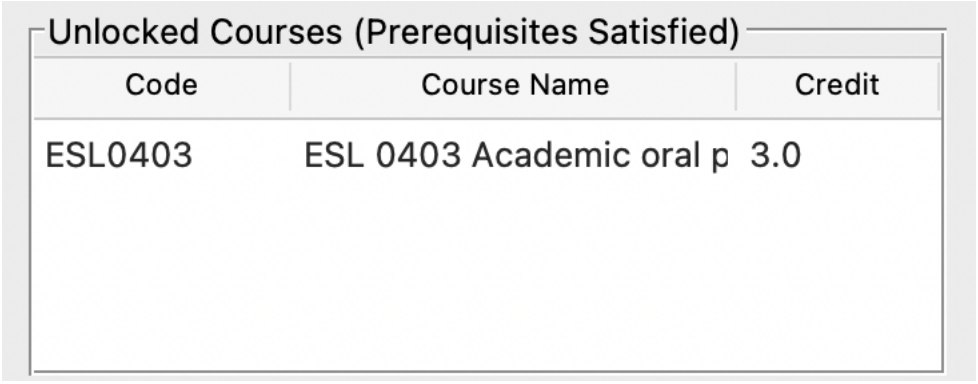
grades_current 47 | course_prerequisites 44 | courses 45 | grade_ch



change_id	course_id	old_grade_po...	new_grade_poi...	change_type	change_time	comment
1	1	NULL	4.00	Normal	2025-12-07 17:30:40	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL

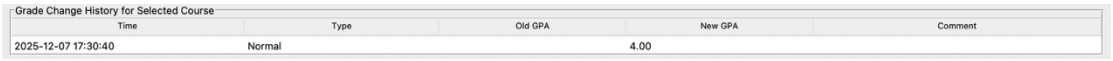
grade_changes 46 | grades_current 47 | course_prerequisites 44 | courses 45 | course

Because the refresh function, the unlocked and grade change history list also refreshed.



Unlocked Courses (Prerequisites Satisfied)

Code	Course Name	Credit
ESL0403	ESL 0403 Academic oral p	3.0



Grade Change History for Selected Course

Time	Type	Old GPA	New GPA	Comment
2025-12-07 17:30:40	Normal		4.00	

7. Conclusion

This project initially integrates a four-year plan, a grading system, and a course scheduling system, enabling rapid entry of grades and course planning. However, some course selection logic is still incomplete and requires further improvement.