Michael Kearns

Tim Miller

ELEC-3225-01

Professor Carpenter

07/22/20

CURSE

The CURSE program is a registration system for a university that allows students, instructors, and administrators to interact and modify course schedules. Each user must login with their respective IDs. Each user can also search the course catalog by some parameter or view the entire catalog. Students can register for courses, drop courses, print their individual schedule, and check for conflicts in their schedule. Instructors can print their teaching schedule and print course rosters. Finally, administrators can add and remove courses, add instructors and students, and link instructors or students to a course.

The main functionality of this program comes from the database organization and how that information is used to allow the users to do what they need. Most user functions are simple queries to fetch or update data in the database. For example, a student adding a course queries the course table to print a list to the student of valid CRNs. When the student adds a course to their schedule, an update statement is created to update a table that keeps track of each students' and instructors' courses.

The process for completing this project was a mix between waterfall and incremental design. The waterfall portion of this project focused on getting each component working before having a simplified version of the final product working. Once the program worked in its simplified version, functionality was incrementally added along the way. Models were created for the different aspects of the design. These models include a context model of the entire system, activity models for the different user functions, sequence models for functionality such as login, logout, searching, and use case models the student, instructor, and admin. These models can be found on the GitHub repository at the link provided at the end of this report.

Overall, this program is very successful. Each user can successfully go through each of their functions and the database will be updated accordingly. When an admin makes changes to the system, the next time a student or instructor logs in, those changes can be seen in the system. One area for improvement would be the student checking for scheduling conflicts. As of right now, they can see all courses that conflict, but the program does not tackle multiple courses

overlapping at the same time. It will, however, still give the student an idea of which courses overlap. If the student drops one of the courses, it will still show the other two overlapping.

Most functions are very efficient as they are simply database queries to either create tables, insert data, return data, or update records. Examples of these functions are the setupDB function, populateSchedule function, and admin create student function. These functions focus on inserting data to the pre-existing tables in the database. More complicated functions such as a student or admin registering a student for a course are less efficient. This is because they have to go through multiple cases to check if the student is allowed to register for that course. Another function that is less efficient is checking for any scheduling conflicts for a student's courses. The least efficient portion of the program is checking for schedule conflicts. There are four nested for loops in that function, with if statements checking if the two innermost need to run. Even with these inefficiencies the program is able to make changes and show them to the user seemingly instantly. A way to improve its efficiency is to run multiple SQL statements under one python execution statement. The setup/reset database function could also benefit from having a stored procedure that would run a SQL script all at once. Stored procedures are good for running large chunks of SQL statements that need to be completed on a regular basis. In conclusion, the best way to increase the efficiency of this program would be to clean up the SQL statements and how and when they are executed. With that in mind, this program scores a 75% for efficiency. The Python code does not add much for inefficiency and around 25% of the SQL statements could be improved upon.

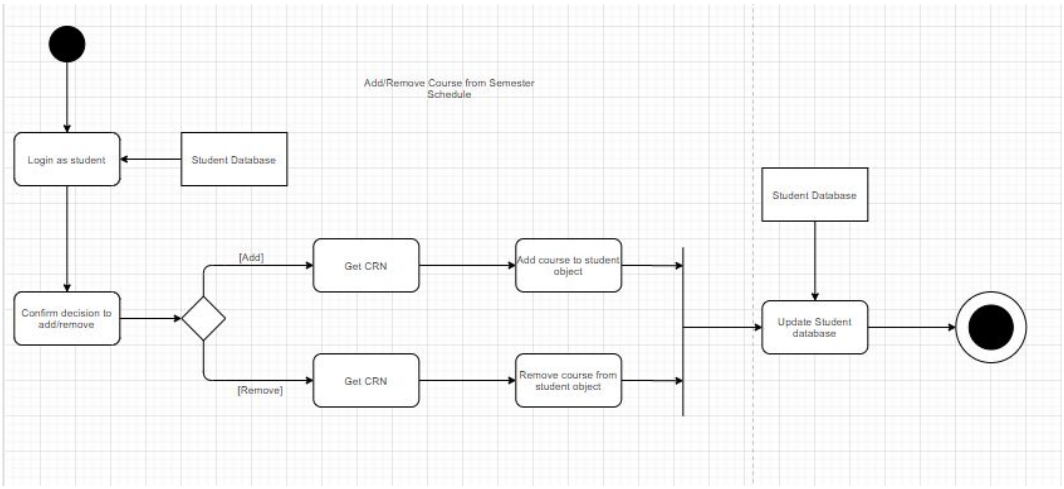
The CURSE program is a great foundation for a simplified University registration system. On this scale, it operates extremely well for allowing each user to carry out their functionality. In the future, scalability and redundancy would greatly increase the usability of this program on a larger scale.

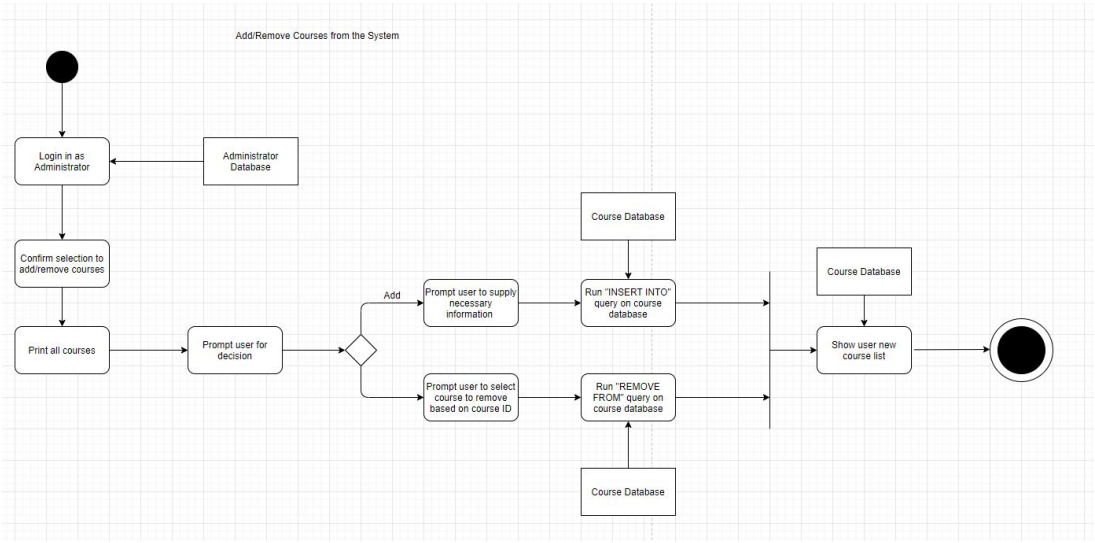
For work distribution, We originally started by sharing a single file on discord that everyone was able to work on, and reupload when they were done, and changes were merged manually. After about a week of that, the project transitioned to github. Editing the file was often done while the members were in a voice call and or screen sharing to facilitate easier coding and communication. For the classes, Tim wrote all of the admin functionality, Michael wrote most of the user, student, and admin code. Aside from those, Tim took care of the login/logout, menu, and Michael wrote all of the helper functions.

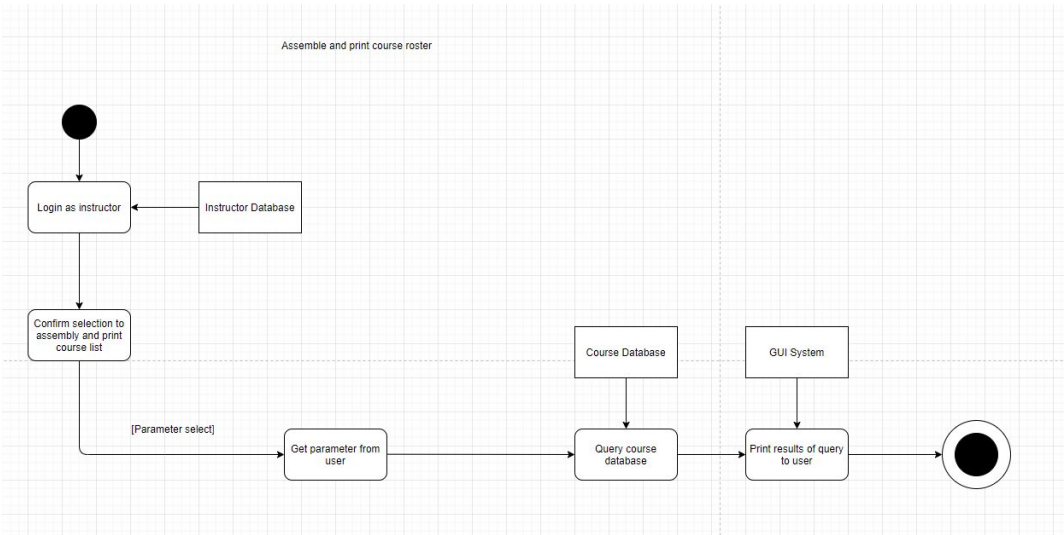
GitHub Link: https://github.com/kearnsm2atwit/APC_Assignment04

The code for the assignment is CURSE.py

Video Link: https://www.youtube.com/watch?v=1ZtWk7kKYYM&feature=youtu.be

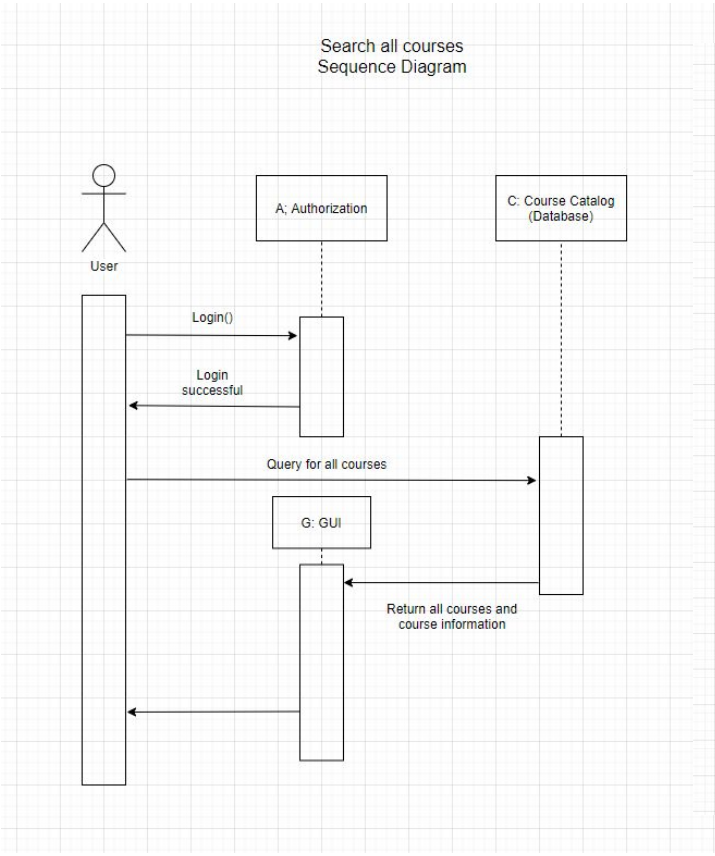

Add/Remove Course from Semester Schedule
Login as student
Student Database
Confirm decision to add/remove
[Add]
Get CRN
Add course to student object
[Remove]
Get CRN
Remove course from student object
Student Database
Update Student database

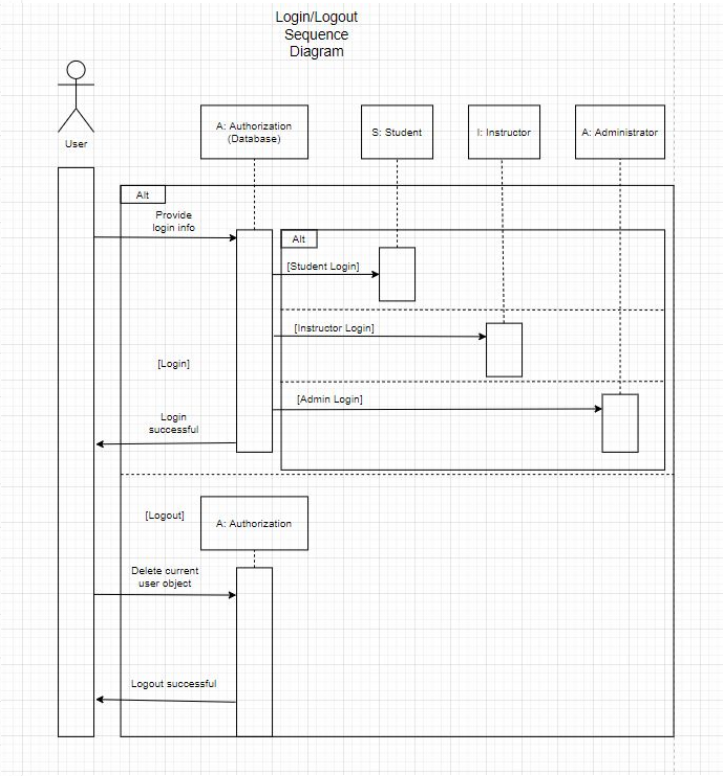

Add/Remove Courses from the System
Login in as Administrator
Administrator Database
Confirm selection to add/remove courses
Print all courses
Prompt user for decision
Add
Prompt user to supply necessary information
Run "INSERT INTO" query on course database
Course Database
Prompt user to select course to remove based on course ID
Run "REMOVE FROM" query on course database
Course Database
Course Database
Show user new course list

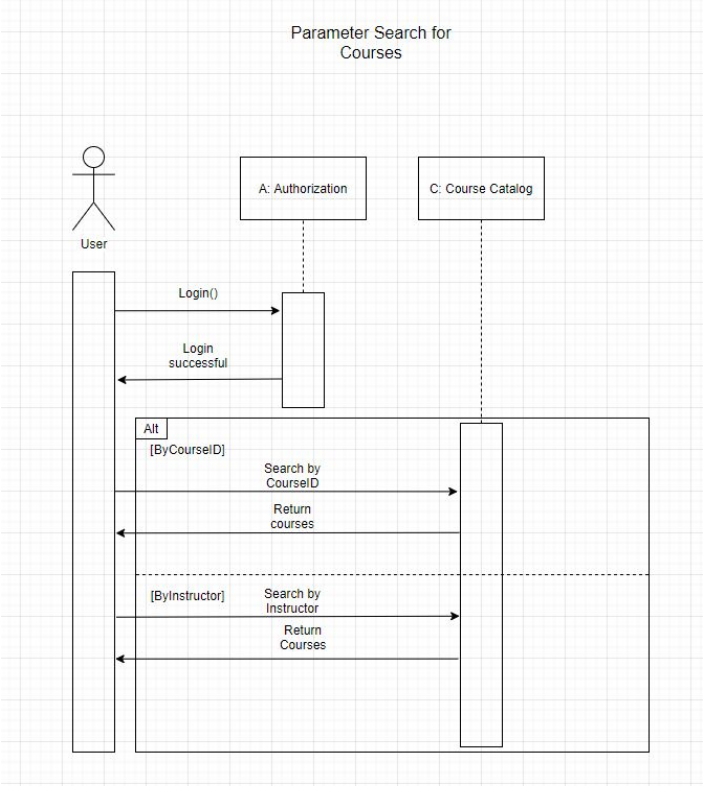

Assemble and print course roster
Login as instructor
Instructor Database
Confirm selection to assembly and print course list
[Parameter select]
Get parameter from user
Course Database
Query course database
GUI System
Print results of query to user

Use Cases Models

Actors:	Student, Instructor, Administrator
Description:	A student has the lowest level of access to the information held within the CURSE system. Students can interact directly with the systems to form and view their schedules. They cannot make significant changes to the system (Courses, shcedules, etc)
Data:	Student, Instructor, Administrator information/logins. Information on courses are also available.
Stimulus:	User commands. Stimulus varies between what type of user is logged into the system.
Response:	Confirmation screens for any changes made. Print outs to screen for strictly viewing functions
Comments:	User misuse will print a screen with an error and some description of what the error was. Examples of user misuse include incorrect login, unauthorized access to functions, etc.


Search all courses
Sequence Diagram
User
A; Authorization
C: Course Catalog
(Database)
Login()
Login
successful
Query for all courses
G: GUI
Return all courses and
course information


Login/Logout
Sequence
Diagram
User
A: Authorization
(Database)
S: Student
I: Instructor
A: Administrator
Alt
Provide
login info
Alt
[Student Login]
[Instructor Login]
[Login]
[Admin Login]
Login
successful
[Logout]
A: Authorization
Delete current
user object
Logout successful


Parameter Search for
Courses
User
A: Authorization
C: Course Catalog
Login()
Login
successful
Alt
[ByCourseID]
Search by
CourseID
Return
courses
[ByInstructor]
Search by
Instructor
Return
Courses