

COMP.3100 Database II Spring 2024

Term Paper

Author: Paige Labbe

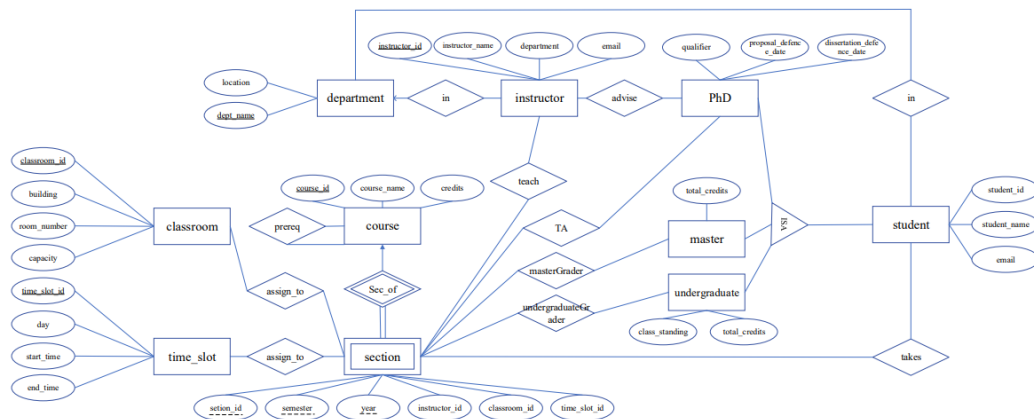
I. Design Specification

Phase I: Design a database

To complete this project, XAMPP and Android Studio software were used. In Phase I each team was tasked with designing an entity-relationship diagram for a database. The information is stored as entity and relationship sets with cardinality and participation constraints. Attributes and primary keys are labeled for each set. After the diagram is designed it is translated into relationship tables with primary and foreign key constraints (if used). The following relationships were given and to be accounted for:

- The database should include details about courses, students, and instructors.
- Each course can have several sections.
- Students can be undergraduates, MS students, or PhD students.
- PhD students can serve as TAs for a course, while MS students and undergraduates (who have scored A- or higher in the course) can be graders.
- MS and PhD students may have one or more advisors who are instructors.
- Each section should have exactly one instructor and can have none, one, or multiple TAs and graders.
- An instructor may teach up to four sections, ranging from none to four.
- Each section is assigned a specific time slot in a particular classroom, with the restriction that only one section can occupy a classroom at any given time slot.
- No more than four sections can be scheduled concurrently in different classrooms during the same time slot.

Pictured below is the completed entity-relationship diagram based on the above relationships. The rectangles are tables, the circles connected are their attributes, the diamonds are actions. Underlined solid-line attributes are primary keys while dotted-lines are foreign keys. A double border of a shape means it can have multiple values.



Phase II: Implement a web-based database

The first part of Phase II consisted of taking the entity-relationship diagram and the tables made from it's logic and encoding them into a database in PhpMyAdmin. The database name is 'DB2'. After creating the database and inserting the tables, at least five records for each table must be added. Example code for adding a table and populating it with five records is shown below for the table 'course':

```
CREATE TABLE `course` (
  `course_id` varchar(20) NOT NULL,
  `course_name` varchar(50) NOT NULL,
  `credits` decimal(2,0) DEFAULT NULL CHECK (`credits` > 0)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

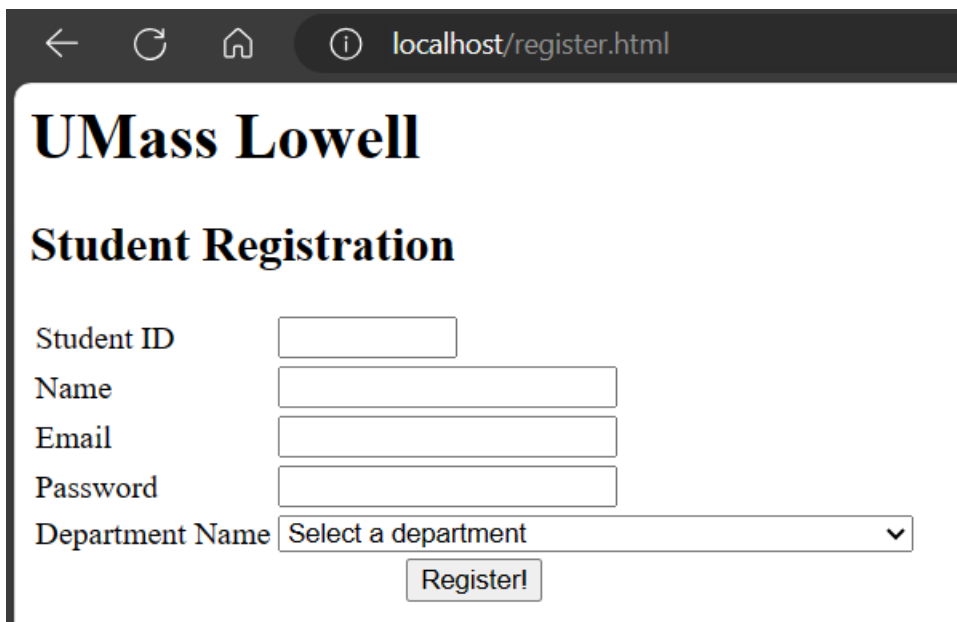
```
INSERT INTO `course` (`course_id`, `course_name`, `credits`) VALUES
('COMP1010', 'Computing I', 3),
('COMP1020', 'Computing II', 3),
('COMP2010', 'Computing III', 3),
('COMP2040', 'Computing IV', 3),
('MATH1310', 'Calculus I', 3);
```

The second part of Phase II involved performing the following specific queries on the database 'DB2' using only plain HTML and PHP.

1. A student can create an account and modify their information later. (The accounts for admin and instructors are created in advance).
2. The admin will be able to create a new course section and appoint an instructor to teach the section. Every course section is scheduled to meet at a specific time slot, with a limit of two sections per time slot. Each instructor teaches one or two sections per semester. Should an instructor be assigned two sections, the two sections must be scheduled in consecutive time slots.
3. A student can browse all the courses offered in the current semester and can register for a specific section of a course if they satisfy the prerequisite conditions and there is available space in the section. (Assume each section is limited to 15 students).
4. A student can view a list of all courses they have taken and are currently taking, along with the total number of credits earned and their cumulative GPA.
5. Instructors have access to records of all course sections they have taught, including names of current semester's enrolled students and the names and grades of students from past semesters.
6. Teaching Assistants (TAs), who are PhD students, will be assigned by the admin to sections with more than 10 students. A PhD student is eligible to be a TA for only one section.
7. Grader positions for sections with 5 to 10 students will be assigned by the admin with either MS students or undergraduate students who have got A- or A in the course. If there is more than one qualified candidate, the admin will choose one as the grader. A student may serve as a grader for only one section.
8. The admin or instructor can appoint one or two instructors as advisor(s) for PhD students, including a start date, and optional end date. The advisor will be able to view the course history of their advisee's and update their advisee's information.
9. Student-proposed functionality #1
10. Student-proposed functionality #2

Using the XAMPP and PhpMyAdmin, the PHP and HTML files must be saved under the XAMPP/htdocs file. To begin we made a 'home.html' to be a home page where the user can choose to create a new account (for students only) or to login. Choosing to create a new account sends the user to 'register.html'. Login will login the user based on email and password, then query the database to find their 'type' (student, admin, instructor) so

it can send them to their appropriate menu, as each type has its specific tasks. Starting with query 1, to create a student account user-input must be taken in so it can be inserted into the database. A record must be inserted into 'undergraduate', 'student' and 'account' tables. The entity-relationship diagram shows that 'undergraduate' needs {student_id}, 'account' needs {email, password, type} and 'student' needs {student_id, name, email, dept_name}. An HTML file must be created to use a webpage to ask the user for this information, then pass it along through a 'post' command to a PHP page. The PHP page will take the data and insert it into the database with SQL commands. The below picture is what the 'register.html' page looks like to collect user data, the code following it (register.php) demonstrates the PHP code of the first query, after the data is sent from the HTML code.



The screenshot shows a web browser window with the address bar displaying 'localhost/register.html'. The page has a dark header with navigation icons (back, refresh, home, info) and the URL. The main content area has a white background with the text 'UMass Lowell' in a large, bold, serif font. Below it is the title 'Student Registration' in a bold, serif font. The registration form consists of five labeled input fields: 'Student ID' (a small text box), 'Name' (a medium text box), 'Email' (a medium text box), 'Password' (a medium text box), and 'Department Name' (a dropdown menu with the text 'Select a department' and a downward arrow). Below the dropdown is a 'Register!' button.

```
<?php

$student_id = $_POST['student_id'];

$name = $_POST['name'];

$email = $_POST['email'];

$dept_name = $_POST['dept_name'];

$password = $_POST['password'];

$type = "student";

$DOCUMENT_ROOT = $_SERVER['DOCUMENT_ROOT'];
```

```
$myconnection = mysqli_connect('localhost', 'root', '') or die ('Could not connect: ' .  
mysqli_error());
```

```
$mydb = mysqli_select_db ($myconnection, 'db2') or die ('Could not select database');
```

```
$query = 'INSERT INTO student (student_id, name, email, dept_name, password)  
VALUES (?, ?, ?, ?, ?)';
```

```
$stmt = $myconnection->prepare($query);
```

```
$stmt->bind_param("issss", $student_id, $name, $email, $dept_name, $password);
```

```
$stmt->execute();
```

```
$stmt->close();
```

```
$query = 'INSERT INTO account (email, password, type) VALUES (?, ?, ?)';
```

```
$stmt = $myconnection->prepare($query);
```

```
$stmt->bind_param("sss", $email, $password, $type);
```

```
$stmt->execute();
```

```
$stmt->close();
```

```
$query = 'INSERT INTO undergraduate (student_id) VALUES (?)';
```

```
$stmt = $myconnection->prepare($query);
```

```
$stmt->bind_param("i", $student_id);
```

```
$stmt->execute();
```

```
$stmt->close();
```

```
mysqli_close($myconnection);
```

```
echo '<script>alert("Student successfully registered!")</script>';
```

```
header("Refresh: 0; URL=home.html");
```

```
?>
```

It uses the 'POST' commands to take in the data, then establishes a connection to the database, then uses insert statements to insert the information into the tables. Finally closes the database and sends the user back to the home page, so they can login.

Query 2 consists of having an admin be able to create a new course section and appoint an instructor to it. Since each course section requires a time slot (which has its own restriction of two sections per time slot), the instructor can only teach one section per time slot, and they must be consecutive. This requires retrieving information from the database, displaying choices to the user based on that information, then inserting information into the database based on their selections. Below is a picture of the 'createSection.php' page:

← ↻ 🏠 ⓘ localhost/createSection.php

University of Massachusetts Lowell

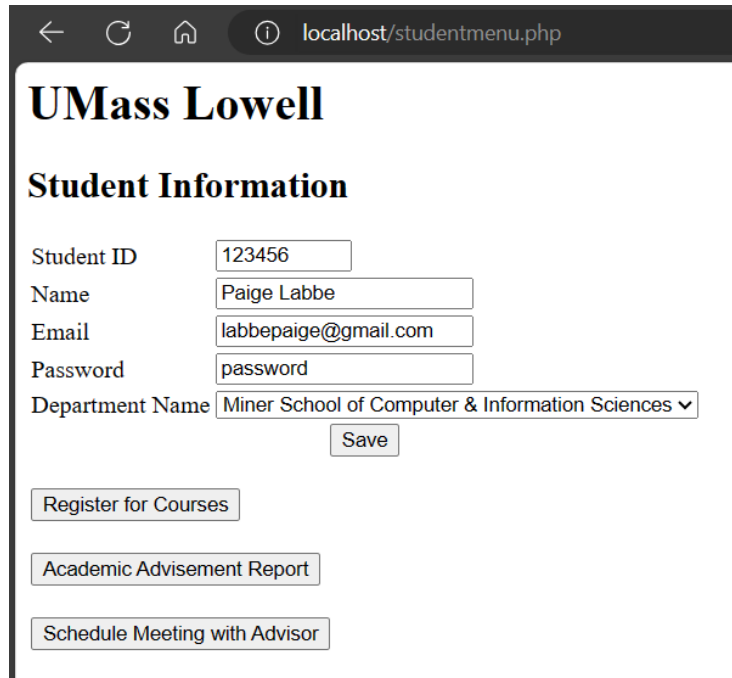
Create Section

instructor_id ▼
classroom_id ▼
course_id ▼
Section:
Semester: ▼
Year:
Time Start: ⌚
Time End: ⌚
Monday ☐
Tuesday ☐
Wednesday ☐
Thursday ☐
Friday ☐
Saturday ☐
Sunday ☐

Before this page is displayed, a query to the database is performed to retrieve the 'instructor_id's from 'instructor' table, this populates the drop-down menu seen above. The same is done for classroom_id from the 'classroom' table and course_id from the 'course' table. The rest of the values are hard-coded in. Section and Year must be manually typed in. Once all the fields are filled out and the 'submit' button is clicked, the data is sent to 'createSectionAction.php' where it can be evaluated and potentially inserted into the database (if it meets the requirements). First a check is done to see if the course and section for the selected semester and year already exists, it will send an error if so. The next check is to ensure the selected times are valid and do not overlap with one another. Section and time_slot tables are used next to check if the classroom is occupied at the specific time. Now it must check if and how many sections the instructor is already signed up to teach that semester, if none, the section can now be inserted. If they are

teaching one course, the section's time must be checked against the new section's time, as they must be consecutive. If the instructor is already teaching two sections for that semester then an error will be thrown to indicate they can only teach two sections per semester.

Query 3 and 4 (and part of 1) are accessed through the 'studentmenu.php' screen. Once a student logs in from the 'login.html' screen, they are displayed the following page (specific to each user who is logged in):



The screenshot shows a web browser window with the address bar displaying 'localhost/studentmenu.php'. The page title is 'UMass Lowell'. Below the title is the heading 'Student Information'. The form contains the following fields and values:

- Student ID: 123456
- Name: Paige Labbe
- Email: labbepaige@gmail.com
- Password: password
- Department Name: Miner School of Computer & Information Sciences (dropdown menu)

Below the form is a 'Save' button. At the bottom of the page, there are three buttons: 'Register for Courses', 'Academic Advisement Report', and 'Schedule Meeting with Advisor'.

Query 1 requested that the student can later edit their information, which can be done on this screen. Everything but the student_id (it is a primary key) can be edited, then when 'save' is clicked, 'editinfo.php' is sent the data from this page and it updates the 'account' and 'student' tables. Query 3 is accessed via the 'Register for Courses' button. This page first queries the database to find the available courses for the next semester and displays them on the page. Once a section is chosen and the 'register' button is clicked, the 'checkPreReq.php' is sent the data. This page first checks if there is room in the section (only 15 students per section), if it passes the check it must check for prerequisites. It searches the 'prereq' table based on the course_id. If one of more exists, it checks the 'take' table to see if that specific student (student_id) has taken and passed the course. Below is example code of just the SQL statements and if statements for checking the prerequisites and potentially inserting the new section:

```

<?php

$query = 'SELECT numEnrolled FROM section WHERE section_id = ? AND semester
        = ? AND year = ?';

if($numEnrolled < 15){

    $query = 'UPDATE section SET numEnrolled = ? WHERE section_id = ? AND
            course_id = ? AND year = ? AND semester = ?';

    $query = 'SELECT prereq_id FROM prereq WHERE course_id = ?';

    if($prereq_id){

        $query = 'SELECT course_id, semester, year, section_id FROM take
                WHERE course_id = ? AND student_id = ?';

        if($result_course_id){

            $query = 'INSERT INTO take (student_id, course_id, section_id,
                    semester, year) VALUES (?, ?, ?, ?, ?)';

        }

        else{ echo '<script>alert("Student does not meet prerequisite
                requirement!")</script>';    }

    }

    else{ $query = 'INSERT INTO take (student_id, course_id, section_id,
            semester, year) VALUES (?, ?, ?, ?, ?)'; }

}

else{

    echo '<script>alert("This section is at capacity!")</script>';

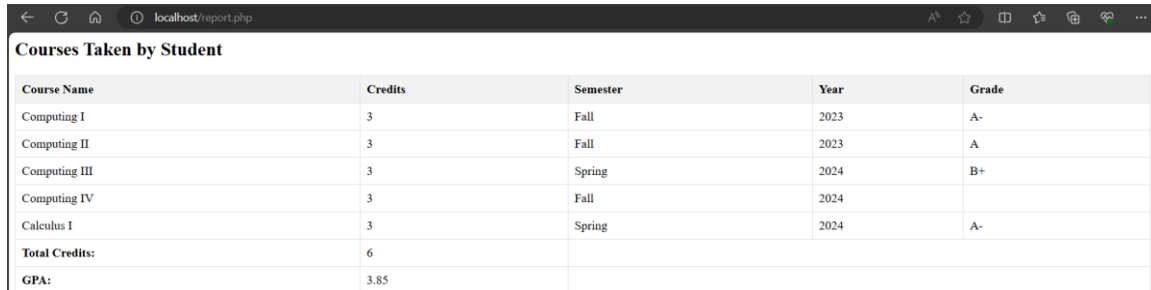
    header("Refresh: 0; URL=studentmenu.php");

}

?>

```


Upon clicking ‘Academci Advisement Report’ Query 4 is commenced as ‘report.php’ is loaded, ‘A student can view a list of all courses they have taken and are currently taking, along with the total number of credits earned and their cumulative GPA.’:



Course Name	Credits	Semester	Year	Grade
Computing I	3	Fall	2023	A-
Computing II	3	Fall	2023	A
Computing III	3	Spring	2024	B+
Computing IV	3	Fall	2024	
Calculus I	3	Spring	2024	A-
Total Credits:	6			
GPA:	3.85			

The student_id is passed from the ‘studentmenu.php’ to ‘report.php’. It queries the database table ‘take’ in the following way:

```
'SELECT c.course_name, c.credits, t.semester, t.year, t.grade FROM take t INNER JOIN
course c ON t.course_id = c.course_id WHERE t.student_id = ?'
```

Using the student_id it finds all courses taking by the student, displays them and also tracks the credits and converts the letter grade to a grade-point, later computing the GPA.

Upon an instructor logging in they are sent to the ‘instructormenu.php’ where they have three options ‘View Record’, ‘Appoint’ and ‘Advisee page’. Query 5 shows that ‘Instructors have access to records of all course sections they have taught, including names of current semester's enrolled students and the names and grades of students from past semesters.’ Clicking ‘View Record’ will send the user to the page ‘irecord.php’ which using the instructor_id, queries the database for the course_names, course and section ids, years and semesters of classes that specific instructor has taught:

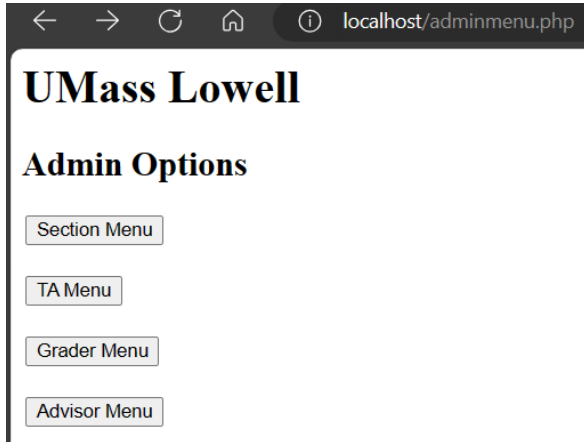
```
'SELECT course_name, section.course_id, section_id, section.year, semester FROM
section, course, instructor WHERE ? = email AND instructor.instructor_id =
section.instructor_id AND section.course_id = course.course_id
ORDER BY course_name ASC';
```

Then it sends this data to be displayed as ‘Course list’. The next query will retrieve the student’s data (their name and grade) to display it under ‘Student list’:

```
"SELECT DISTINCT student.name, take.grade, course_name, section.course_id,
section.section_id, section.year, section.semester FROM take, student, section, course,
instructor WHERE ? = instructor.email AND instructor.instructor_id =
section.instructor_id AND section.course_id = course.course_id AND take.student_id =
student.student_id AND take.section_id = section.section_id AND take.course_id =
```

section.course_id AND take.semester = section.semester AND take.year = section.year
ORDER BY student.name ASC";

Query 6, 7 and 8 involve navigating from the 'adminmenu.php':



Clicking on 'TA Menu' will bring you to a page that you see the current assigned TAs. There is a button as well to assign TAs. This will bring you to 'taselection.php' where a query is first done to retrieve the PhD students who are available to be assigned to be a TA (they are not currently a TA already):

```
"SELECT DISTINCT PhD.student_id, student.name FROM PhD, student WHERE  
student.student_id = PhD.student_id and PhD.student_id NOT IN (select student_id  
FROM TA)";
```

Another query is completed to determine the courses that the TA can be assigned to (must have at least 10 students enrolled in section):

```
"SELECT course_id, section_id, semester, year FROM section WHERE numEnrolled >=  
10 AND section_id NOT IN (SELECT section_id from TA)"
```

The submit button does not go to a new page to insert the TA, so this functionality does not work. Query 7 involves assigning graders who are MS students or undergrad students that achieved an A- or A in a specific course. A student can only be a grader for one section per semester. This is accessed via the 'Grader Menu' button. This query does not work, as when the button is clicked it only brings up courses that need a grader assigned, only based on number of students enrolled. The following is the query for this:

```
"SELECT course_id, section_id, semester, year FROM section WHERE numEnrolled >=  
5 AND numEnrolled <= 10";
```

Query 8 is accessed via the 'Advisor Menu' button from 'adminmenu.php' or via 'Advisor Page' from the 'instructormenu.php'. This query requests that the

admin/instructor appoint an instructor as an advisor for PhD students. It must include the start date, and potentially an end date. From 'instructormenu.php' 'Advisor Page' will bring you to 'advisor.php'. This page queries the database to see if there already exists any advisors for this instructor, if so it will display them on this page:

```
"SELECT DISTINCT student.name FROM student, instructor, advise WHERE ? =  
instructor.email AND instructor.instructor_id = advise.instructor_id AND  
student.student_id = advise.student_id ORDER BY student.name ASC";
```

The page has 'Update Advisee' that brings you to 'updateAdvisee.php'. This page displays the advisee_id, start date and optional end date for the instructor to choose. Upon clicking 'Submit' the data is sent to 'updateAdviseeAction.php' where it will be inserted into the database with this statement (as long as the dates do not overlap):

```
"INSERT INTO advise (instructor_id, student_id, start_date, end_date) VALUES  
(?, ?, ?, ?) ON DUPLICATE KEY UPDATE start_date = VALUES(start_date), end_date  
= VALUES(end_date)"
```

Query 9 was a student-proposed functionality. We chose to add scheduling a meeting with an advisor to be this functionality. At the 'studentmenu.php' there is a button to 'Schedule Meeting with Advisor' which goes to page 'schedule.php'. This page displays the available days and times of meetings upon querying 'advising_time':

```
'SELECT time_id, day, start_time FROM advising_time'
```

Upon selection and clicking 'Submit' the data will be inserted into the table 'advise_appt' with this query:

```
'INSERT INTO advise_appt (student_id, instructor_id, time_id) VALUES (?, ?, ?)';
```

Phase III: Implement an Android app connected to a database

Phase III involved performing the following specific queries on the database 'DB2':

1. A student can create an account.
2. A student can browse all the courses offered in the current semester and can register for a specific section of a course if there is available space in the section. (Assume each section is limited to 15 students).
3. A student can view a list of all courses they have taken and are currently taking.

4. Instructors have access to records of all course sections they have taught, including names of current semester's enrolled students and the names and grades of students from past semesters.

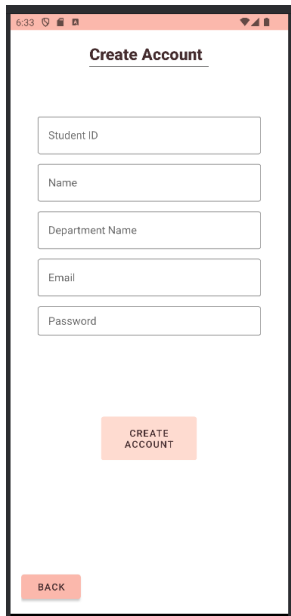
5. Student-proposed functionality #1

After starting XAMPP (which is needed to access the database) click on 'Netstat' in the app then record the I.P. address, this needs to be adding into the strings.xml file in Android Studio so this line will be updated with the current IP you just recorded:

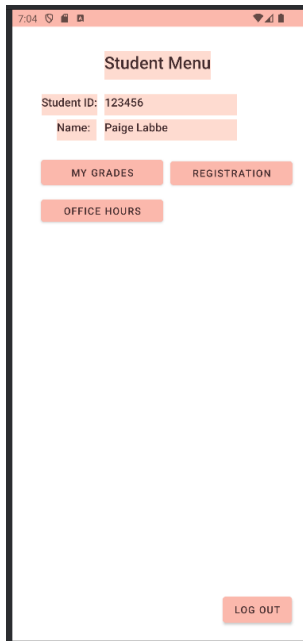
```
<string name="url">http://192.168.40.6:80/Phase3/</string>
```

Now the android app should be able to connect to the XAMPP software.

Query 1, to create a new student account, is done from the 'activity_movie_query.xml' and 'MovieQuery.java' pages. MovieQuery.java reads in the data from the fields seen on the screen from the activity page (seen below).



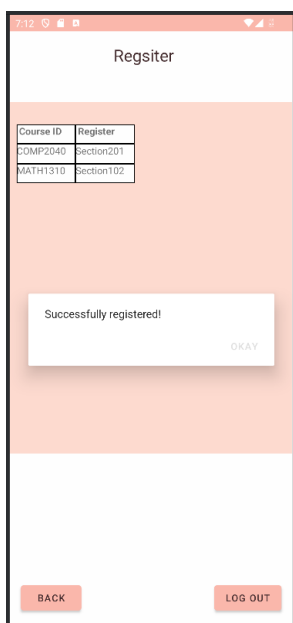
After reading in the data MovieQuery.java calls the PHP file 'create_account.php' and sends the collected data there, to be inserted into the database with the same code used to insert in Phase II Query 1. Logging in will redirect each 'type' to their respective menu. A student logging in will be sent to 'activity_page_menu.xml' which is pictured below. Clicking on the 'MY GRADES' button will bring you to Query 3, 'REGISTRATION' will bring you to Query 2, and 'OFFICE HOURS' will go to Query 5.



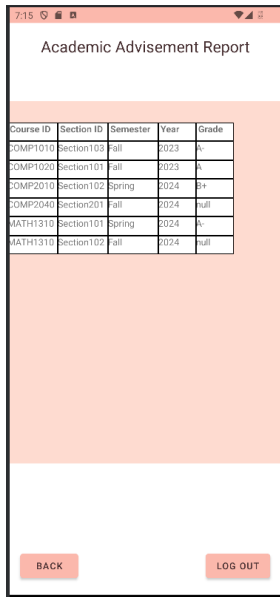
Query 2 requires a student being able to register for a course in the next semester. This is accomplished by navigating to 'activity_page_register.xml' which is populated by querying the database for available courses next semester with the following code:

```
"SELECT course_id, section_id FROM section WHERE semester = ? AND year = ?"
```

Once a row with the desired course is clicked, a message describing if the query was completed or not is displayed, indicating if the student was enrolled or not. The same logic and SQL statements from Phase II, Query 2 are used. Below is an example of what appears after a student is enrolled in a course:



Query 3 requires a student to be able to access a record of their current and past courses.



The screenshot shows a mobile application interface with a status bar at the top displaying '7:15' and various icons. The main title is 'Academic Advisement Report'. Below the title is a table with the following data:

Course ID	Section ID	Semester	Year	Grade
COMP1010	Section103	Fall	2023	A-
COMP1020	Section101	Fall	2023	A
COMP2010	Section102	Spring	2024	B+
COMP2040	Section201	Fall	2024	null
MATH1310	Section101	Spring	2024	A-
MATH1310	Section102	Fall	2024	null

Below the table is a large orange rectangular area. At the bottom of the screen are two buttons: 'BACK' and 'LOG OUT'.

This is accomplished by calling 'grades.php' which is called from 'PageGrades.java' which performs the following query:

```
"SELECT course_id, section_id, semester, year, grade FROM take WHERE student_id = ?"
```

```
if ($result) {  
    while ($row = $result->fetch_assoc()) {  
        $response["courses"][] = [  
            'course_id' => $row['course_id'],  
            'section_id' => $row['section_id'],  
            'semester' => $row['semester'],  
            'year' => $row['year'],  
            'grade' => $row['grade']  
        ];  
    }  
}
```

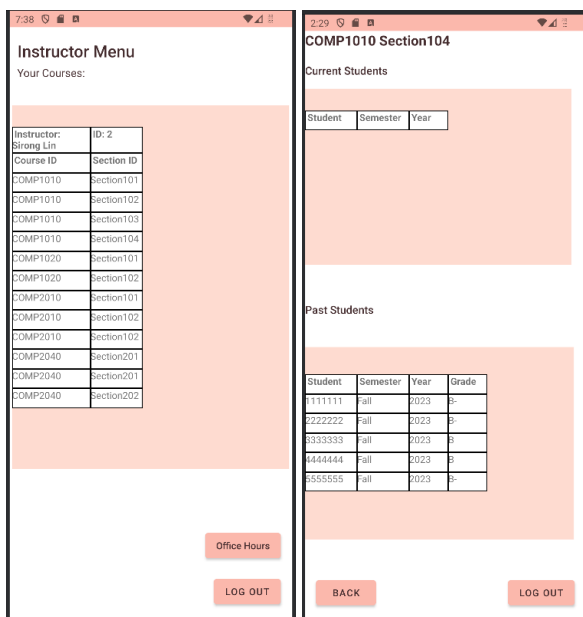
It uses the data sent via 'intent' from 'PageMenu.java'.

```
Intent intent = getIntent();  
final String email2 = intent.getStringExtra("email2");  
final String password2 = intent.getStringExtra("password2");
```

It then sends the data back to 'PageGrades.java' which calls 'activity_page_grades.xml' to display the information (setContentView(R.layout.activity_page_grades)). The java page processes the JSON data taken back from the query and configures it to be sent to the activity page.

```
String jsonString = intent.getStringExtra("courses");  
  
for (int i = 0; i < jsonArray.length(); i++) {  
  
    JSONObject jsonObject = jsonArray.getJSONObject(i);  
  
    String course_id = jsonObject.getString("course_id");  
    String section_id = jsonObject.getString("section_id");  
    String semester = jsonObject.getString("semester");  
    String grade = jsonObject.getString("grade");  
    int year = jsonObject.getInt("year");  
  
}
```

Query 4 requires to login as an instructor to have access to records of all course sections they have taught, including names of current semester's enrolled students and the names and grades of students from past semesters. Upon logging in as an instructor, the query information is displayed, which is the courses and sections the specific instructor has taught. The screen looks like the following picture:

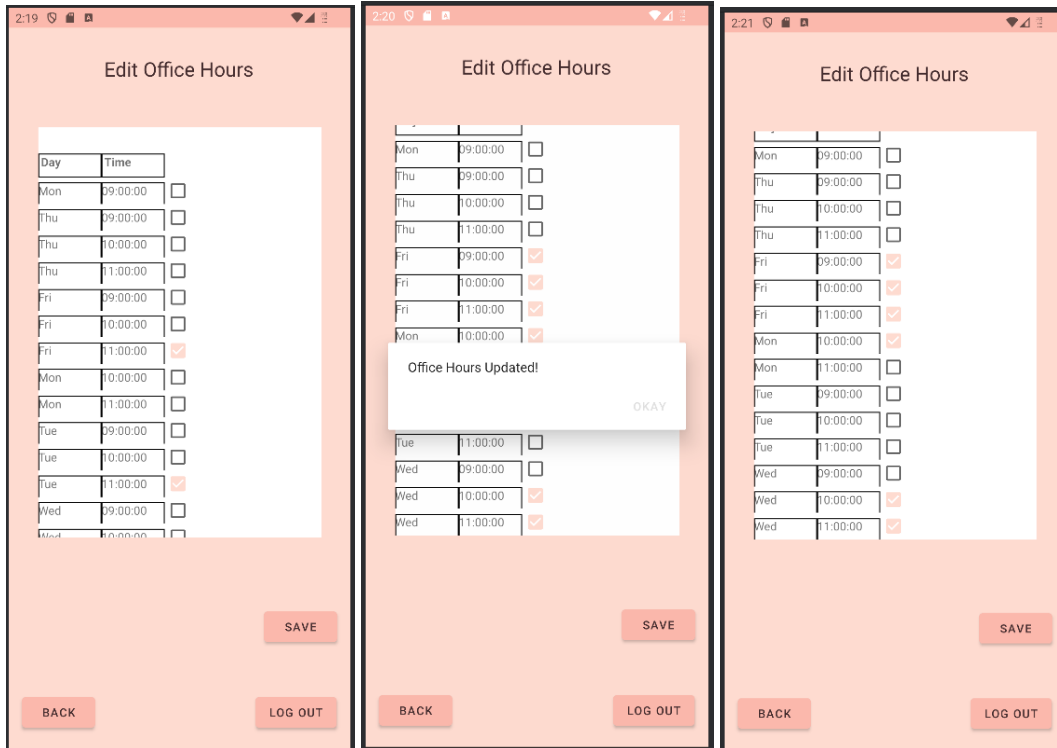


When a section row is clicked, it takes the data from the selected row from 'activity_page_instructor_menu.xml' as this is the current page being viewed. 'PageMenu.java' takes in this data and sends it to a query request at the PHP page called 'sectionInfo.php':

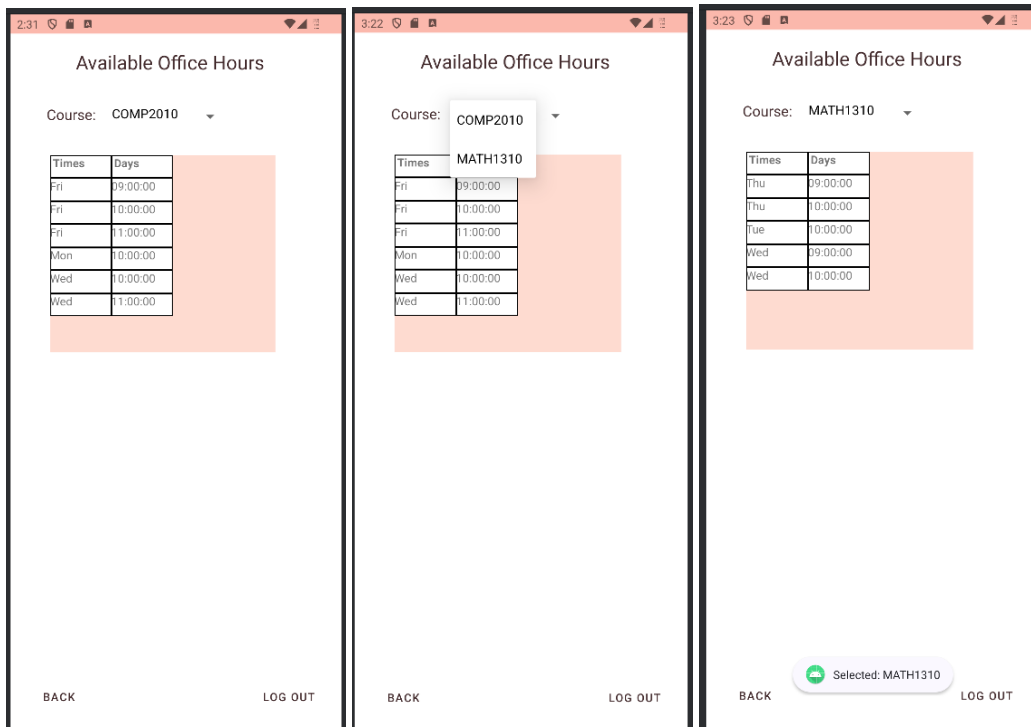
```
QueryRequest queryRequest = new QueryRequest(id, c_id, s_id,
getString(R.string.url) + "sectionInfo.php", responseListener);
RequestQueue queue = Volley.newRequestQueue(PageMenu.this);
queue.add(queryRequest);
```

The program then waits for an onResponse(string response) to know that the query has been completed and the status of the attempt. If a success, it can then take the JSON information returned from the query and pass it to 'PageSectionInfo.java'. Upon entering this page, the view is set to 'activity_page_sectioninfo.xml', the JSON data from the query is taken in from the last page via 'intent' and finally displayed to the view. It is displayed similar to the courses that the instructor has taught except the current semester students and the past students are in their own individual tables.

The final Query, 5, was a student-proposed query. We chose to add the functionality for an instructor to add their availability for office hours, specific to the courses they are teaching that semester. The times that they are already teaching a class will not be displayed for the instructor to pick from to add to their availability. A student is also able to see the office hours available for the specific courses they are enrolled in. This required adding two new tables to the database, 'office_hours' and 'office_time_slot'. 'Office_hours' contains the added availability for the instructors, it has two attributes: instructor_id and time_id. 'Office_time_slot' has attributes: office_id, day, start_time, end_time, it also is pre-populated with a set of office hours for the instructors to choose from. From the 'officeTimes.php' the available times that the instructor has for the semester are calculated, as well as retrieving their already selected available times. This information is posted to an activity page. Pictured below is a demonstration of accessing this page, checking new times and clicking save.



Finally the end of Query 5 allows the student to see the available times for a specific course they are taking that semester. A drop-down menu displays the classes they are taking the current semester, when a course is selected the available office times are shown.



II. Individual Contribution

Phase I of the project was an equal 33.3% for each team member. We completed it fully in class together. For Phase II and III, we each decided on queries we would individually complete. Phase II of the project I completed the home page and login page, to combine all the queries into one cohesive project. Queries 1, 3, 4 and 9 were completely done by me. Query 2 was tough, Kriston Theng completed it as well as Query 5 and 8. Queries 6 and 7 were completed by Felipe Oliveira, unfortunately he was not able to complete Query 10. I did 50%, Kriston 35% and Felipe 15%. Phase III of the project was done completely by me. Felipe finished query 5, but he did not communicate that with me until two hours before the due date. Anticipating this I completed the entire project early, as merging the projects is not always an easy task. As well as having to submit the video submission which takes a few minutes to get perfect. The average of the entire project, I did 61%, Kriston 23% and Felipe 16%.