**COMP.3100 Database II Spring 2024**

**Term Paper**

Author: Alex Kashian

**I. Design Specification**

For our project this semester we had to recreate our Student Information Service(sis) using both a web a mobile application. Before I can get into either of those lets start from the begging of our design process. Our first order of business was to figure out how our database was going to look and flow. Once given our functionalities for our application we had to figure out what entities we needed, what attributes each entity needed, and the relations given for one entity to another. These entities, attributes, and relationships would all need to be displayed in an E-R, also know as entity relation, diagram. Lets take a look at section for example. Each section has a unique section number, year, and semester to it. These three attributes combined are what we call the primary key of the entity. These make it so they are unique and no other section will have the exact same values for all three attribute fields. Now the section has relationships to other entities such as a many to one relationship with course. This means that a course can have many sections but each section may only belong to one individual course. We can also see that section has a occupies relation with the classroom entity. This relation is one to one meaning each section is related to one classroom entity. The list goes on and on and below you can see a full completed version of our E-R. Your probably wondering why section has an extra box around it unlike all the other entities. This is because section is actually a weak entity meaning that in order to identify entities of section uniquely, we need to follow one or more many-one relationships from section and include the key of the related entities from the connected entity sets. This means section also has course number, room number, building ID and time slot ID as part of its uniqueness.

A diagram of a company

Description automatically generated

Once we completed our E-R diagram the next step was to translate these all into tables so when completed it would be easily implemented into our database. This step was crucial not only to help us visualize our database as a whole but also to address what attributes will be referencing other attributes from other entities. For example the takes relation would look something like this with its foreign key constraints.

takes(sid, course\_number, section\_number, semester, year, grade) foreign key constraint sid referencing student(sid), course\_number, section\_number, semester, year referencing section(course\_number, section\_number, semester, year)

As we can see the takes relation references both the student and the section entities to make up all of it attributes. Once all of our relationships were written out it was now time to implement them into our PHPMyAdmin database. This database was accessed with the help of XAMMP which is a web server solution stack package which let us start out MySQL module. In PHPMyAdmin we were able to host our database through its free and open source tools for MySQL. With the tables inserted into the database we then got to work on writing html and php code using Visual Studio Code for some of our first few tasks. We later on realized we needed some data to test our code so we populated to tables so we were able to have test samples for each functionality. One of the first tasks we were given to make is letting a newly signed up student view all the available courses offered in the current semester. We would also had to give the student the option to register for these courses if they met the prerequisite requirements and their was space in the class. When completing this task the first thing we needed to do was find out how many students were in each course. The query and additional php code looked like this.After we found the number of student in class we had to throw an error message if class already had 15 students. This was accomplished with a simple if statement.

A black background with orange and blue lines

Description automatically generated

Once that condition was handle the next was to check if it met the prerequisite requirements. To do so we needed to first collect all data from the takes tables for each student. After that we had to query out all the data from prerequisite courses table for each course. Finally we wrote a function that would check the classes you have taken and set a boolean condition to true if you meet the prerequisite conditions. If the boolean condition would return false then we would throw an error message letting the use know the prerequisite conditions we not satisfied. The code for that would look like this.

A computer screen shot of a program

Description automatically generated

With these two conditions covered all we had to do next is display out all the information for each course offered in the current semester. This was done with a simple one line query We used a table layout to do so with a create rows function as to easy display all the data. We also had to add the class that was just registered for to our takes table with a simple insert query upon completion of registry. This was only one of the ten functionalities we had to complete for a web application of this section of the project but it is a great example of how we are communicating with our database through out php code and querying out data to apply with some logic to make out website run smoothly. Once all ten tasks were complete, two of which were thought up of by my groupmates and I, we then had to provide a mobile application for our database. The concept of recreating our student information service was the same however we were given only 5 tasks to complete, one being one of the group inspired functionalities. For this mobile application we still used PHPmyAdmin for our database and also still used Visual Studio Code told hold our much needed php code. Our mobile app itself was made using around studio where we wrote the application in Java. If we look at the same task as we did for the web application we can see how its created as a mobile application. The first thing that was needed was to create a fragment that would be the blueprint for each course offered in the current semester. These were called course\_register\_items and displayed all the key information such as course name, section number, course number, and number of students. Now to display each course we made a list view of course\_register\_items. Once the UI is all situated we can then go to look at the java code. In our onResponse function we see all the magic happen in displaying our data onto the screen. In here we make an JSON objects to respond with our database and php code. If the response is successful then a JSONArray for all the data and we also make a arraylist of course register models which stores all the needed information for display. Once we have our JSONArray we then want to parse out each induvial piece of data and put them together to add to the course register model array list. Lastly we needed an adaptor that would set handle setting the fields of the fragment and also displaying error and success messages when attempting to enroll in a course. Lastly at the end we ran query request to run our PHP code that we moved around from our web application and the fragment was then A screenshot of a computer program

Description automatically generatedA computer screen shot of text

Description automatically generatedcomplete. The described code looks like this.

This final portion of creating the mobile app came with a lot of stress and confusion using java and android studio for the first time in some time. That being said our group was still able to complete all five tasks with flying colors.

**II. Individual Contribution**

When it came to the work the team put into this project Nate couldn’t have been a better partner to work with. Nate had the best understanding of the project and all the material needed to complete the project. I would go as far to say that without Nate this project may not have had all the functionality it currently does at this point. He was very easy to work with and very helpful at explaining things that may have confused me. Nate most likely helped with every page of our work whether it be to help look over a bug, or explain some concepts to implement, or just take over and do them all himself. He was most likely responsible for 50% of our overall work in all three stages. With that I was always trying to learn and help contribute in every way possible. I would help with getting pages finished or starting on new ones, logging data for entries, designing UI and navigation, and other small things hear and there. What ever Nate needed me to do at a given time before class or during a meeting I would try to complete it the best I could. Like I mentioned earlier some thing needed the assistance of Nate but I would get most of my assigned tasks completed with him. The most helpful part was that we were able to meet in class and over discord a few times a week. Here is where I was able to help and contribute the most because of the assistance of Nate. I would say I contributed to about 35% of our project. Lastly Dan was not there all the time and the times he was he was writing code that would have to be modified heavily or scrapped completely. He would miss classes and meetings down the stretch of the semester when we needed to get work done the most. Overall I would say he contributed the least and was responsible for 15% of the work.