**Diary Management System**

**Advanced Database Design**

**CS-603-B**

##### TechME



Sacred Heart University

School of Computer Science & Engineering The Jack Welch College of Business & Technology

Submitted To Dr. Reza Sadeghi

Spring 2022

CS-603-B Project Progress Report Phase #6 TechME

**TechME**

# Project Phase 06 for

# Diary Management System (DMS)

### Team Members

1. Kiera Cutri cutrik@mail.sacredheart.edu
2. Ankush Chaudhari chaudharia@mail.sacredheart.edu

### Roles of Team Members

1. Kiera Cutri (Team Head)
2. Ankush Chaudhari (Team Member)

2

CS-603-B Project Progress Report Phase #6 TechME

## Table of Contents

Team Member Introduction 4

Project Summary 5

Purpose

Scope

Motivation

Literature Summary

Pros and Cons

Entities and Keys 6

Entity-Relationship (ER) to Enhanced Entity-Relationship (EER) Model

SQL Code Desc.

Models and Figures 7,8

ER Model

EER Model

Entities, Attributes, and Keys 9

SQL Code 10  
 Example of Insertion Error

Constraints Protect from Invalid Entry

Altering Our Code 11,12

Optimization 13

Graphical User Interface (GUI) 14,15

Future Work 16

Citations 17

**Table of Figures**

Fig. 1 – ER Model 7  
Fig. 2 – EER Model 8  
Fig. 3-12 – Alter Modifications 10  
Fig. 13-18 – Login GUI 14

3

**Team Member Introduction**

**Kiera Mariah Cutri**

SHU ID Number: 0813395

SHU Email: [cutrik@mail.sacredheart.edu](mailto:cutrik@mail.sacredheart.edu)

Kiera Cutri is a current Computer Science graduate student at Sacred Heart University; they have previous experience with C++, C#, Java, and Python. Kiera has previously worked at the Sacred Heart Factory as a level one technician for 3 years where she gained experience with troubleshooting various software issues; she has also performed various repairs and upgrades to their laptop.

I met Ankush after the first Advanced Database course and we conversed about interests, backgrounds, and hobbies. After finding common ground for interests, I thought that he would be a good partner for the project, he reached out to me first regarding group work. The team leader was decided through a discussion of responsibilities and through deciding who would prefer to take the role.

**Ankush Chaudhari**

SHU ID Number: 0882634

SHU Email: [chaudharia@mail.sacredheart.edu](mailto:chaudharia@mail.sacredheart.edu)

I am Ankush Chaudhari. Currently, I am a graduate student at Sacred Heart University. I have completed my Undergrad in Electronics and Telecommunication Engineering. After I completed my degree, I worked for MNC as an implementation consultant for 1.6 years. My role was to handle ERP systems, Client relations Project Management, Vendor Management, provide clients hands on training on the system, and design and integrate the system according to client requirements. I am very passionate about my work role. my team had received an extra mile reward for completing the project before the deadline.

It was the first day of my Database Management class. I was seated in the first row. Kiera was seated towards my right. We had little conversation regarding the software installation and some doubts. After class we had a brief conversation regarding our backgrounds. Kiera completed her undergraduate in gaming, she also told me about her interest in various programming languages which she has known. On that point I decided that Kiera is a perfect team partner and head, so I approached her and now we are a team.

4

**Purpose**

The purpose of the Diary Management System (DMS) is to aid staff members within a department to collaborate and organize in a school setting. This system allows faculty members to update their activity details within a diary that is accessible to relevant users. Permitted users will have access to the activity of specific events that have been logged within the system. This proposed system aims to minimize the time burden imposed on faculty members by allowing users to search for a colleague's log within their diary. The Head of Department (HOD) will be able to view each faculty member's diary logs.

**Scope**

TechME aims to produce an application that will provide greater ease for users by allowing users to effortlessly search and find records that pertain to other colleagues. This system provides users with an advanced and secure way to store documents that can be accessed by higher ups within a department.

**Motivation**

These days, everything tends to be stored within databases. Our staff continues to use more traditional logging methods when recording work information. To make the most out of current technology, we are creating the DMS which will allow faculty members to upgrade their work from traditional logging techniques to databases.

**Literature Survey**

Keeping logs to maintain work records daily allow faculty members to keep track of their work in their department. Each faculty member can keep and maintain a traditional diary which stores logs of their workday. Faculty members input various information relevant to their workday such as attendance, course number, semester taught, comments for the class, and content covered during lecture. The diary logs are later reviewed by the Department Head at the end of the day. Within our college, every faculty member keeps and maintains a work diary to enter logs of their workday daily. By maintaining a work diary, it allows faculty members to detail their workday and keep a log that would be available to other coworkers within their department. By having a diary system in place, it allows others to know the daily routines of faculty members and allows for easier assignment based on availability.

We were able to gauge an idea of how this system would work based on an online example of a diary management system on sourcecodester.com [2].

**Pros and Cons**

When considering the Diary Management System. We considered potential positives and negatives that would attribute to the system. The primary positives of the system would be that it would provide an accurate timeline of events and permit for easier scheduling of employees. The system would be able to support multiple team members at once and would allow coworkers to hold each other accountable for each other for specific actions preformed. The primary potential negatives we came across for this project would be a decrease in teamwork among members, large data size which use large pieces of software which would occupy multiple megabytes of diskspace, there could be an annual recurrent maintenance cost, and an ethical concern for privacy.

For more information regarding this project, please refer to the following github repository: <https://github.com/cutrik/CS603-FinalProject>

5

**Entities and Keys**

When selecting entities for our project, we examined how a university or college works and attempted to mimic how they operate within a database. We began by looking at the direct relationships among members of an institution; we noticed that there were individuals who worked for the institution which fell under faculty or staff members. We noticed that students interacted with both faculty and staff. Since faculty and staff members played a vital role in the Diary management System, we included them in our entity list.

We observed that the institution utilized multiple buildings and classrooms which members of the institution interacted with. When we looked to how faculty members use the building, we included aspects which were being used. Building name, classroom location, department, and the subject which the faculty member taught were all aspects of the institution which would allow for more detailed diary entries. Faculty members have information that pertain to themselves and to other aspects of the institution such as scheduling and teaching. Faculty members belong to different departments based on their specialized subjects which impact what they would teach to students.

We noticed that the faculty members schedules aligned with what students would take based on what the faculty member would teach. We thought it would be important to include aspects of the student and faculty interaction in our database by using student class status and student schedule. We noticed that students typically had declared one major but would still take a variety of subjects for their schedule. With each subject that a student would take, each class would occur at different times and different days for varied durations.

**ER to EER Model**

After considering what all our entities would be for the DMS, we broke down what attributes would be a part of each entity. We viewed each entity and determined which ones were strong and which ones would be weak; afterwards, we listed what type of relationship each of our entities would have with each other such as many to many, one to one, and one to many. After determining the relationships between entities, we transferred this into an ER Model. Once the ER Model was created, we transferred the information into an EER Model by adding datatypes to each attribute. We kept the names of entities and attributes across models.

For the visuals of these models, please refer to the figures on pages 7 and 8.

**SQL Code**

When writing the code for the database we referred to w3schools.com [5] and guru99.com [4] for how to write and format primary keys and foreign keys. The image above links to a copy of the SQL code for the Diary Management System.

6

**Models and Figures**

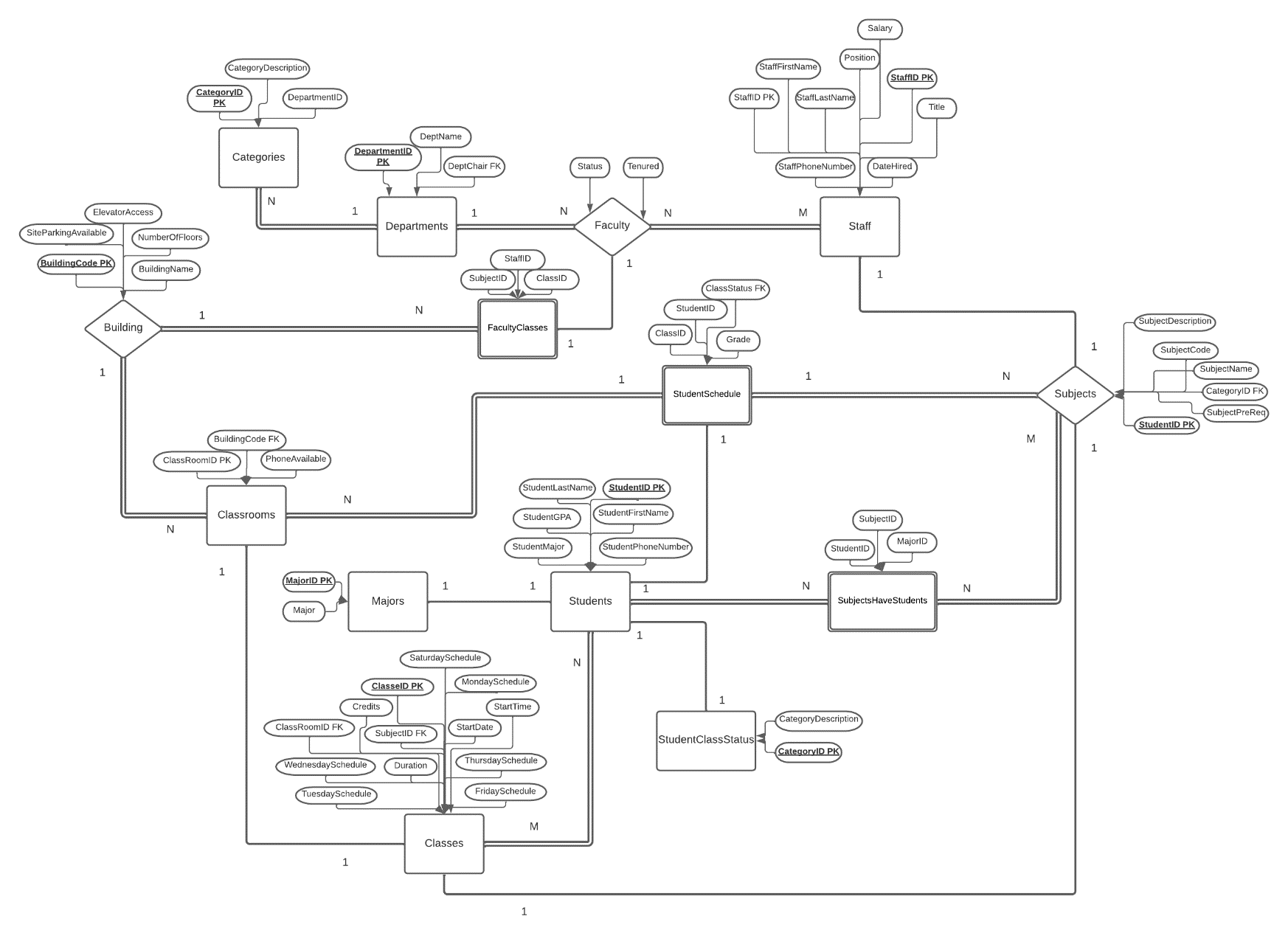
****

Figure 1: ER Model

Created in Lucidchart at [www.lucidchart.com](http://www.lucidchart.com) [3]

7

Timeline

Description automatically generated

[1]

Figure 2: EER Model

8

**All Entities, Attributes, and Keys**

|  |  |
| --- | --- |
| **Faculty**  StaffID PK  Title  Status  Tenured  **Staff**  StaffID PK  StaffFirstName  StaffLastName  StaffPhoneNumber  Salary  DateHired  Position  **Departments**  DepartmentID PK  DeptName  DeptChair FK  **FacultyClasses**  StaffID CPK  ClassID CPK  SubjectID CPK  **Subjects**  SubjectID PK  CategoryID FK  SubjectCode  SubjectName  SubjectPreReq  SubjectDescription  **Classes**  ClassID PK  SubjectID FK  ClassRoomID FK  Credits  StartDate  StartTime  Duration  MondaySchedule  TuesdaySchedule  WednesdaySchedule  ThursdaySchedule  FridaySchedule  SaturdaySchedule | **Classrooms**  ClassRoomID PK  BuildingCode FK  PhoneAvailable  **Building**  BuildingCode PK  BuildingName  NumberOfFloors  ElevatorAccess  SiteParkingAvailable  **StudentSchedule**  ClassID CPK  StudentID CPK  ClassStatus FK  Grade  **Students**  StudentID PK  StudentFirstName  StudentLastName  StudentPhoneNumber  StudentGPA  StudentMajor  **SubjectsHaveStudents**  SubjectID  StudentID  MajorID  **StudentClassStatus**  CategoryID PK  CategoryDescription  **Majors**  MajorID PK  Major  **Categories**  CategoryID PK  CategoryDescription  DepartmentID |

9

**SQL Code**

When creating our database, we observed how most systems work in a school setting. When looking to things like if a faculty member were tenured, if a phone would be available, or if an elevator would be available in the building, we decided to use a single character to indicate if these things were available by having a yes (‘y’) value or a no (‘n’) value. Information such as titles, names, positions, and schedule we were unsure how many characters the user would input so we made sure to make these attributes variable characters. Initially, for attributes such as date hired, phone number, salary, and credits we thought that these values would be stored as integers and have allotted a value for how many digits would be input by the user. We later changed these values to reflect the variables more accurately in the database. As an example, we changed phone number from an integer value to a character value based on how this does not change, we kept the number of digits the same based on phone numbers containing ten digits in the United States. The variable ‘GPA’ was changed from integer to a float value based on how the value for a grade point average is stored on a four-point scale at Sacred Heart University. For attributes such as subject code, subject name, and other class identifying attributes we had adjusted these to be character values based on a mixture of a set number of letters and numbers.

**Example of Insertion Error**

INSERT INTO subjectsHaveStudents (subjectID, studentID, majorID) VALUES ("ASTO" , "0835019", 11650)

Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`universitymanagement`.`subjectshavestudents`, CONSTRAINT `fksubjectID` FOREIGN KEY (`subjectID`) REFERENCES `subjects` (`subjectID`)) 0.016 sec

INSERT INTO subjectsHaveStudents (subjectID, studentID, majorID)

VALUES ("ASTO", "0835019", 11650);

**Constraints Protect from Invalid Entry**

Different constraints were used in our database to prevent invalid inputs; some such constraints were in the form of primary keys and foreign keys. Some examples that we had used in our database were through the datatype float, datetime, and time. Float was used for the attribute “studentGPA”, by having this value be a limited decimal value, it allows the data to be specified as a numeric value while if this value were a different datatype, it would open the ability for incorrect data. Another datatype that was used was time and datetime; these datatypes require input in a valid format and will reject the input if the formatting is incorrect, this decreases the input of incorrect data through mandatory formatting and only acceptance of numeric values. We had used the time and datetime through the “startDate” and “startTime” attributes. Additionally, we used multiple constraints through keys. Some such examples of keys we had used were primary and foreign keys. By using these keys, it assures that the data must remain consistent with the primary key to run without error; an example of a primary key we had used was “studentID”, we had referenced this through foreign keys in tables such as “subjectID”. As a foreign key, if the information about the datatype, the information input, or the length allotted did not match the primary key, the data would not store and would need to be edited.

10

**Altering Our Code**

When adjusting our code, we had to either add columns or adjust datatypes of attributes. We preformed these changes through a sequence of alter commands in our code. For the table “staff” we added the column for birth date and used an update command to input the data per identifying number for staff members in the database. We additionally preformed multiple modify commands to change previous datatypes to more efficient and relevant ones. The following are some code excepts for where these changes were made:

1. ALTER TABLE staff

ADD dateOfBirth datetime(6);

UPDATE staff

SET dateOfBirth = CAST('1989/01/22' AS DATETIME)

WHERE staffID = 0216659;

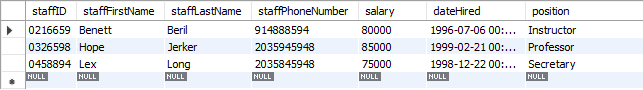


Figure 3: Before Alter staff

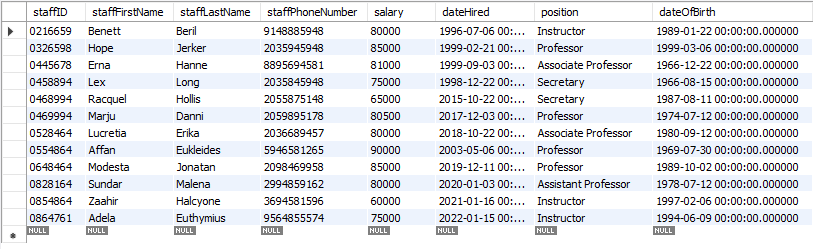


Figure 4: After Alter Staff

1. ALTER TABLE falcultyClasses

ADD subjectID varchar(8);



Figure 5: Before Alter falcultyClasses

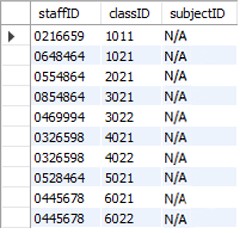


Figure 6: After Alter falcultyClasses

11

1. ALTER TABLE categories

MODIFY subjectID char(8);

UPDATE categories

SET subjectID = CAST('ASTRO' as char)

WHERE categoryID = 'ASTR1265';



Figure 7: Before Alter categories

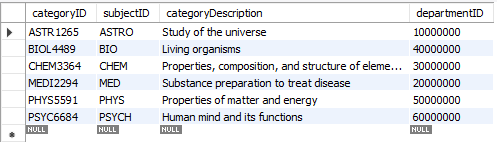


Figure 8: After Alter categories

1. ALTER TABLE building

MODIFY buildingCode varchar(6);

UPDATE building

SET buildingCode = CAST('1945-A' as varchar)

WHERE buildingName = 'Astronomy Building';

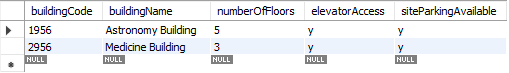


Figure 9: Before Alter building

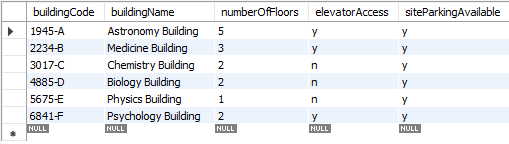


Figure 10: After Alter building

1. ALTER TABLE studentSchedule

MODIFY studentID char(8);



Figure 11: Before Alter studentSchedule

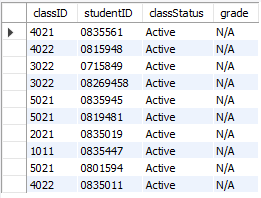


Figure 12: After Alter studentSchedule

12

**Optimization**

When optimizing our database, we carefully reviewed our code to see how best to optimize for the best performance. We had already produced some good optimization techniques from the beginning of the creation of our database such as selecting specific columns for insertion of values instead of calling the entire table, we avoided running queries in loops, we used ‘create table if exists’, and used the ‘where’ command instead of ‘having’. Based on good coding practices, we did not need to adjust the database to further optimize.

By using the ‘where’ command instead of the ‘having’ command it allows for faster execution of the database since the database will perform the command immediately while the ‘having’ command would wait to preform the command until the end of execution of the database. This can be seen when the data for some tables has been updated.

UPDATE categories  
SET subjectID = CAST('ASTRO' as char)  
**WHERE** categoryID = 'ASTR1265';

By using the command ‘create table if not exists’, allows for the database to overlook tables if they already exist and immediately go to the next command; this allows the program to continue running without the risk of duplication.

**create table if not exists** departments(departmentID int(8),

deptName varchar(25),

deptChair varchar(60),

CONSTRAINT pk\_departments\_SID PRIMARY KEY(departmentID));

13

**GUI**

When in the process of creating our graphical user interface (GUI), we referenced multiple projects to gain an idea of how to produce a proper and fully functioning GUI. We have attempted to manipulate and edit various projects to attempt to incorporate them into our project. For the login page of our application, we borrowed code from an online resource [6], we have attempted to adjust the login page to redirect to another GUI application but received errors and are in the process of fixing this. In the login GUI, there are multiple functions for each portion of the login process; each of these functions are called in a main function which is then called at the end of the code, the main function calls all previous functions and triggers them based on user interaction with buttons on the GUI. The functions are first designed then an additional function is preformed to create the window, each portion of the login GUI has either a functionality implementation, popup implementation, or event implementation.

The following figures are from the login GUI.

|  |  |
| --- | --- |
| Figure 13: Welcome Page | Figure 14: Login |
| Figure 15: Registration | Figure 16: Registration Success |
| Figure 17: Login Success | Figure 18: User Not Found, Denied Login |

We attempted to rework our login code to connect to our database, we also made sure to allow the GUI to obtain the username and password from the database instead of reading from a text file. 14

The following excerpt is the connection to the database:

import mysql.connector

db = mysql.connector.connect(

host="localhost",

user="root",

passwd="Ankush@11",

database="universityManagement")

To obtain the username and password we can obtain these two fields through the following code in the GUI:

uname = e1.get()

password = e2.get()

cursor = db.cursor()

15

**Future Work and Conclusions**

This project has allowed us to further develop our communication skills, time management, peer review, peer accountability, the ability to delegate responsibilities, and refine our understandings of concepts through careful discussion and explanation with each other.

For the future of this project, we would like to create a fully working graphical user interface. We would also like to polish the aesthetic appearance of the application; one such way we would attempt this is by taking original photos and creating an original logo to incorporate into the application. We would like to optimize the graphical user interface to run more efficiently after having a more complete implementation. We would like to add security features to prevent malicious attacks on the database. Additionally, we would like to attempt to create a single executable file for the application rather than having multiple python files.

16

**Citations**

1. Viescas, John L. “School Scheduling Modify Database Page 883.” *SQL Queries for Mere Mortals: A Hands-on Guide to Data Manipulation in SQL*, Pearson Technology Group Canada, 2018.
2. elinet. “Elinet.” *Free Source Code, Projects & Tutorials*, 11 Sept. 2014, https://www.sourcecodester.com/php/7943/online-diary-management-system.html.
3. W, Megan. “Urgent! -- Cite LudicChart – Lucidchart.” *Lucidchart*, 20 Sept. 2018, https://lucidchart.zendesk.com/hc/en-us/community/posts/360018227346-URGENT-Cite-LudicChart.
4. Peterson, Richard. “SQL Primary Key: How to Create & Add to Existing Table.” *Guru99*, 29 Jan. 2022, https://www.guru99.com/sql-server-primary-key.html.
5. “SQL FOREIGN KEY Constraint.” *SQL Foreign Key Constraint*, https://www.w3schools.com/sql/sql\_foreignkey.asp.
6. Saba, Gulsanober, et al. “Python GUI Login - Graphical Registration and Login System in Python.” *Simplified Python*, 4 Mar. 2020, https://www.simplifiedpython.net/python-gui-login/.

17