Diary Management System

Database Management

Section 111

Coding Comrades

Marist College
School of Computer Science and Mathematics

Submitted To:
Dr. Reza Sadeghi

Spring 2023

Table of Contents

Table of Figures

1. Project Report of Diary Management System

Team Name

Coding Comrades

Team Members

1. Tania Hernandez-Martinez .....tania.hernandez-martinez1@marist.edu (Team Head)
2. Alexis Harman .....alexis.harman1@marist.edu (Team Member)
3. Daniella Boulos …..daniella.boulos1@marist.edu (Team Member)

Description of Team Members

1. Tania Hernandez-Martinez

My name is Tania Hernandez-Martinez. I am a computer science major. I wanted to work with my current team because all three of us were able to get along right away and I saw that each member is ready to collaborate with each other which shows me that we will be able to work together well moving forward. We selected the team head by asking if there was anyone that was interested in the role and if they felt comfortable with the extra responsibilities and tasks.

2. Alexis Harman

My name is Alexis Harman, and I am a sophomore. I am a software development major, minoring in cybersecurity and information systems. I wanted to work with the current team members because we have strong communication skills, and this will serve us well throughout the process of completing the project. We selected the team head by one of us asking the group if any of us had any aspirations to be the team head and was comfortable using GitHub.

3. Daniella Boulos

My name is Daniella Boulos. I am a sophomore who is majoring in cybersecurity, and minoring in computer science, information systems, and technology. I wanted to work with my current team members because we were able to get along quickly as well as we all have excellent communication skills. We managed to select the team head by asking if anyone had the desire to be the team head and had experience in GitHub.

2. Project Objective

This Diary Management System (DMS) records daily events and experiences for different users. The users can securely add records and update them. The DMS combines functions from various sources such as Google Calendar, Google Sheets, and Gmail. It will store the data of different user types in distinct SQL tables.

Logging in requires the admin's username and password. The admin has the capability to add a guest with their own username and password to view the DMS; however, the guest, unlike the admin, cannot add or remove other guests. The admin and guests can add a record with specified details such as time, place, duration, and description. They can also remove a record entry and edit its details. There will be a search function for all the users so that way they can find a record entry by searching up either the specific entry or one of its details.

Another function that our DMS will provide is a warning or error that will appear if the user tries to put in a new entry with an overlapping time to a previous entry. When a user first logs in, a welcome page will appear to greet the user. All the current entries can be viewed like a calendar. Finally, there will be a button to exit their DMS.

3. Review Related Work

[1]Google Calendar has many positive capabilities. Some include its ability to organize tasks, be accessible from any device, share capabilities, and add multiple calendars to the main calendar. It is also free. When placing a task within a specific day on the calendar, the user can do so within a specific time slot. To increase or decrease the time of the task the user can drag the task bubble up or down. This function creates a very easy and effective way for the user to create their task and make any adjustments when necessary. [2]Google Calendar can be accessed from any device. The user may not always have their laptop with them, but they will have their phone. The sharing capabilities allow many individuals to have access to one calendar. This feature is great for a family to manage everyone's schedules, a business to keep track of all meetings, and more. Google Calendar allows the user to create several calendars to the main calendar. Separate calendars can include school, work, home, birthdays, etc. The user selects how many calendars are visible on the main calendar. Lastly, Google Calendar is free. Many fancy calendar apps may ask for a subscription to their services or provide a free version with limited features. Google Calendar provides many useful capabilities without charging its users. Google Calendar provides the user with many positive aspects, although there are some negatives. One includes difficulty when granting access to others. Though the calendar can be shared with others, allowing multiple individuals access can be confusing. The user must be aware to select the box whether they wish to grant viewing or editing access to said individual. This extra step may be forgotten and may cause the owner of the calendar to have to change the specific access they wish the individual to have. A second negative aspect is that a user needs a Google account. The user may only want or need to use the calendar function, however, that can not be done until after creating an account. A third aspect is the lack of color diversity when selecting the different calendars. There are a handful of colors with different shades; however, the shades are not very different from one another. There is only a slight difference between the shades, which may confuse the user.

Google Sheets is another great example of a Diary Management System (DMS), even if it is not the first thing that comes to mind when an individual thinks of a DMS. Just like our DMS, Google Sheets can be shared with others. Unlike Google Sheets, our DMS can be shared with anyone, not just those with Google accounts. The admin user will be able to adjust everyone’s edit access. An admin can choose if someone can just view the DMS or if they will be able to edit it as well. Similar to Google Sheets, it will be a free application where the user can add and remove entries. The user will also be able to autofill cells. Auto-filling cells are appealing to many people. For example, it is great for mathematical calculations because it produces the calculations on its own. Also, if a user needs to type an entry multiple times, they can type it once and have it autofill the rest. While Google Sheets’ shortcuts are different from Microsoft

Excel, they are still very effective and there is a little learning curve if a particular user is used to Microsoft Excel; otherwise, it is highly intuitive.

A third example of a business version of our Diary Management System(DMS) is Gmail. Some of the positive capabilities of Gmail include the security it provides with 2-factor authentication, its ease of navigation without much support, its great compositional flow, and of course, Gmail is free. We find the security Gmail provides to be of utmost importance because, through the use of a second email or phone number for confirmation, the user is able to ensure that they are the only one that can get into their account. Gmail also makes it so that anyone, from the person always exposed to technology to the person who is not, can navigate the platform without much trouble[3] since everything is labeled clearly and placed intentionally on the screen. As it is with the other Google platforms, Gmail is also free which makes it accessible to anyone. Though Gmail is a great platform, it is not perfect. We mentioned above that the platform is initially free to use. However, a user who happens to be someone who sends or receives a lot of emails and attachments or stores them, can quickly use up all of the free storage provided and at that point, the user would be forced to pay for more storage. Another problem that does not aid the storage problem is the fact that Gmail allows numerous ads from various websites to be sent to a user's email thus overwhelming the user with countless emails that are not only useless the majority of the time, but take up valuable storage. Aside from that, Gmail also has a lack of customization options available. Part of the experience of a platform like this one is the ability to customize it and make the platform one's own, something that Gmail does not allow much of.

Researching and studying the way these versions of our DMS function has allowed us to determine what we would like to see and create for our DMS. It has also made us aware of the things we do not want in our Data Management System.

4. Merits

Our project will include many capabilities for the end user. It will provide organized tasks, accessibility from any device, and it will be secure. To ensure the security of our product we will require two-factor authentication or a Caesar Cipher. The admin will request a username and password to log in. The password will be a string of at least eight characters and the admin will have the capability to change their username and password. If the admin wishes to add a user to the DMS they must create a new username and password for the user. The admin can remove a user from the DMS, including his/her username, password, and other corresponding data. Each user should be able to add a diary record with the following data: Time, place, duration, description, and priority (low, medium, and high). The user should also be able to remove a diary record, edit details, and search through diaries. The search would be based on time, place, or duration. The list would result on the screen of the scheduled works for one day. The DMS will have user-friendly software. It shows a warning if a user tries to put in a new task with an overlapping time to the previous records. It will also display a welcome page and menu of all the functions to the user. It will provide the reports in a tabular form and an exit function. The end user should select our product because it will entail the functionality of a secure account, it can be accessed from any device, and organize tasks. We want to provide the user with one app that allows easy access to all their information that may be stored in one or more apps. This app merges their information that is stored in Google Calendar, Sheets, and Gmail. Our project will incorporate the functionality of all three apps into one.

5. GitHub Repository Address

https://github.com/T-H19/CMPT-308-111_DiaryManagementSystem_CodingComrades

6. Entity Relationship Model

The Entity Relationship (ER) Model was created on a website called lucidchart where we inserted shapes and text to create the diagram shown below. The entities were selected by figuring out what the main functions of our DMS are. Their attributes were chosen by figuring out what each entity entails. The relationships were selected by deciding how each entity was related to one another. The participation was decided upon by determining whether each relationship had zero or one interaction required. The cardinality was determined by figuring out whether each entity could have multiple relationships or not.

We have a total of ten entities and each entity holds about five attributes. All of the entities are connected through a relationship. The participation was determined by whether or not an entity needed another entity to operate. Say the entity needed another to operate, it would need total participation, and if the entity did not need another it would be partial participation. Cardinality is the ratio between the two entities.

We have a total of ten entities. The entities include description, guest, calendar, user, place, priority, time, date, duration, and reminder. The entity description has five attributes: text, add attachment, bold, italicize, and underline. Text is the primary key and has a total participation with calendar. The relationship between description to calendar is 1:1. The entity guest has five attributes: contact number, Fname, Lname, email, and quantity. There is no primary key and has partial participation with calendar. The relationship between guest to calendar is M:1. The entity calendar has five attributes: share, new, edit, view, and collaborative. New is the primary key. Calendar has partial participation with guest, partial participation with description, partial participation with place, total participation with user, and partial participation with priority. Calendar also has a 1:M relationship with guest, 1:1 relationship with description, M:N relationship with place, 1:M user, M:N relationship with priority. The relationship between calendar and guest is 1:M. The entity user has five attributes: recovery email, username, email, password, and name. Name has two composite attributes of Fname and Lname. Username is the primary key and has total participation and a 1:N relationship with calendar. The entity place has five attributes: restaurant, school, home, park, work. Home is the primary key. Place has total participation and a M:N relationship with calendar, partial participation with time and a 1:1 relationship. The entity time has six attributes: morning, afternoon, night, time zone, repeat, all day. There is no primary and time has a partial participation with place, partial participation with duration, and total participation with date. The entity priority has five attributes: high, medium, low, free, busy. There is no primary key. Priority has total participation with calendar, and total participation with date. Priority also has a M:N relationship with calendar and a M:N relationship with date. The entity date has five attributes: day, year, month, time/time zone, repeat. There is

no primary key and time/time zone is a multivalued attribute. Date has a total participation with priority, partial participation with time, and partial participation with reminder.The entity duration has five attributes: Minute, Hour, Day, Month, and Seconds. Day is the primary key. Duration has partial participation and a M:N relationship with time. The entity reminder has five attributes: Year, Month, Week, Day, and Hour. There is no primary key. Reminder has total participation and a M:N relationship with date.

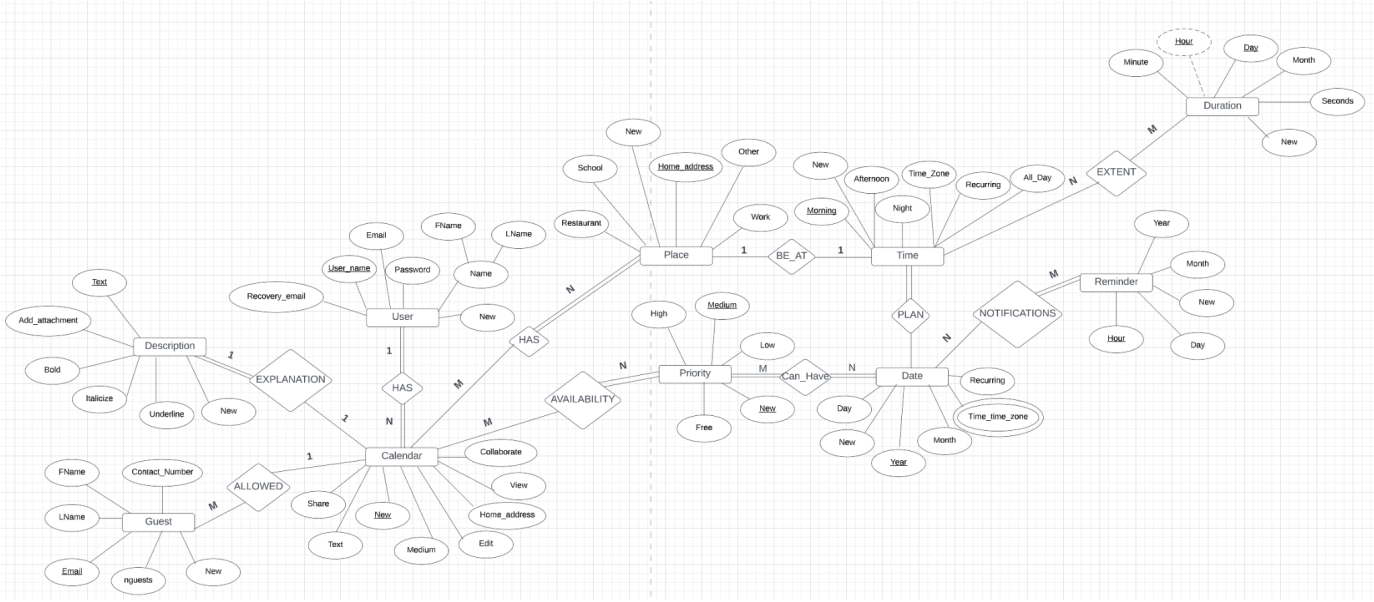


Figure 1: Entity Relationship Model
Designed By: [4]LucidCharts

7. Enhanced Entity Relationship Model

The primary keys from each table are unique identifiers for that specific table. However, the user can access that information from another table using what is called a foreign key. A foreign key allows the user to retrieve data from the other table. Therefore, a foreign key acts as a relationship between the two different tables.

In order to relate certain tables to one another we added a foreign key showing that the two tables are related. By creating these foreign keys we were able to identify the relationships between the tables and demonstrate how they are related by the use of the crow's feet icons. One example of our use of foreign keys is between the table's description and calendar. The foreign key in our calendar table, description_text, refers to the primary key, text, in the table description. If we need to join the tables of calendar and description to access certain information we can simply use the foreign key of description_text.

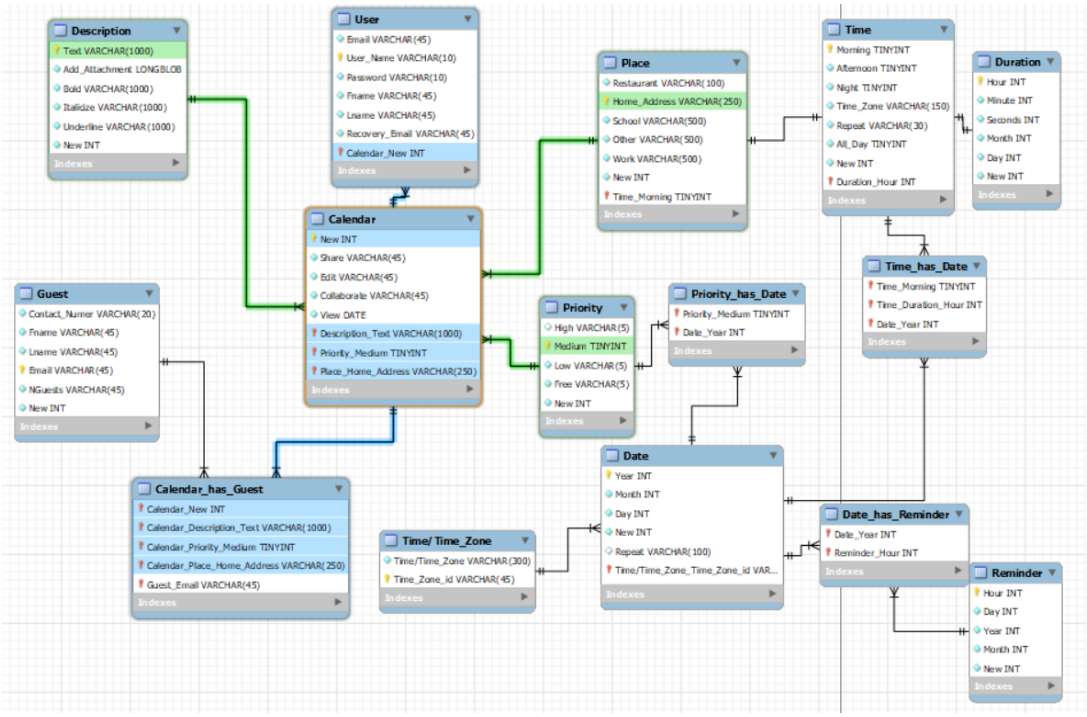


Figure 2: Enhanced Entity Relationship Model
Designed By: [5]MySQL

8. Database Development

To begin, we created a new database for the project, the code we used is shown below. After the database was created, we stated that we wanted to use that new database. Finally, we created the tables that were to be used in the database, which are described below.

```
create database dms_project;
use dms_project;
```

In the table titled Time, we included seven attributes with the titles and data types as follows. Morning tinyint, Afternoon tinyint, Night tinyint, Time_zone varchar(150), Recurring varchar(30), All_day tinyint, and New int. Our primary key and foreign key for this table were Morning and Hour respectively. This table is used to store/assign a time of day to an instance in a new entry. The attribute Repeat from the entity Time in the ER and EER diagrams has been changed to Recurring due to MySQL having a command that uses the same name thus causing MySQL to give us an error. The following code shows how the table was created in MySQL as well as the foreign keys used.

```
create table Time(Morning tinyint primary key, Afternoon tinyint,
Night tinyint, Time_zone varchar(150), Recurring varchar(30),
All_day tinyint, new int, hour int);
ALTER TABLE time
add constraint hour
FOREIGN KEY (hour)
REFERENCES duration(hour)
on update cascade
on delete cascade;
```

In the table titled Priority, We included five attributes with the titles and data types as follows. High tinyint, Medium tinyint, Low tinyint, Free tinyint, and New int. Our primary key and foreign key for this table were Medium and New respectively. This table is used to mark the urgency of the new entry created. The following code shows how the table was created in MySQL as well as the foreign keys used.

```
create table Priority(High varchar(5), Medium tinyint primary key,
Low varchar(5), Free varchar(5), new int,
FOREIGN KEY fk_cal(new)
```

```
REFERENCES calendar(new)
on update cascade
on delete cascade);
```

In the table titled Place, We included six attributes with the titles and data types as follows. Restaurant varchar(100), Home_address varchar(250), School varchar(500), Work varchar(500), Other varchar(500), and New int. Our primary key and foreign key for this table were Home_address and Morning respectively. This table is used to state the location of the entry created. The following code shows how the table was created as well as the foreign keys used.

```
create table Place(Restaurant varchar(100),
Home_address varchar(250) primary key, School varchar(500),
Work varchar(500), Other varchar(500), new int, morning tinyint,
FOREIGN KEY fk_t(morning)
REFERENCES time(morning)
on update cascade
on delete cascade);
```

In the table titled Date, we included five attributes with the titles and data types as follows. Year int, month int, day int, recurring varchar(100), time_time_zone varchar (300), and new int. Our primary key is year int and there is no foreign key for this table. This table states a date when the new entry is created. The following code shows how the table was created in MySQL.

```
create table Date (Year INT primary key, Month INT, Day INT,
Recurring VARCHAR(100), Time_Time_Zone VARCHAR(300), New INT);
```

In the table titled Reminder, we included five attributes with the titles and data types as follows. Hour int, day int, year int, month int, and new int. The primary key is hour and there is no foreign key for this table. This table is used to mark the repetitiveness of the new entry created. The following code shows how the table was created in MySQL.

```
create table Reminder (Hour INT, Day INT, Year INT,
Month INT, New INT);
```

In the table titled Duration, we included five attributes with the titles and data types as follows. Hour int, minute int, seconds int, month int, day int, and new int. The primary key is hour and there is no foreign key for this table. This table states the amount of time allotted for the new entry created. The following code shows how the table was created as well as what was needed to allow for foreign keys.

```
create table duration (Hour INT, Minute INT, Seconds INT,
Month INT, Day INT, New INT);
create index hour on duration(hour);
```

In the table titled Description, we included six attributes with the titles and data types as follows. Text varchar(100), add_attachment longblob, bold varchar(1000), italicize varchar(1000), underline varchar(1000), and new int. The primary key is text and there is no foreign key for this table. This table shows all the different ways the user can add a description or a note to the entry. The following code shows how the table was created.

```
create table Description(text varchar(100) primary key, add_attachment longblob,
bold varchar(1000), italicize varchar(1000), underline varchar(1000), new int);
```

In the table titled Calendar, we included 8 attributes with titles and data types as follows. New int, share varchar(45), edit varchar(45), collaborate varchar(45), view date, text varchar(100), medium tinyint, and home_address varchar(250). The primary key is new and the foreign keys are text, medium, and home_address. This table shows that the user can create new entries as well as edit each entry with specific details. The following code shows how the table was created as well as its foreign keys and the alterations made to the table.

```
create table Calendar(new int primary key, share varchar(45), edit varchar(45), collaborate
varchar(45),
view date, text varchar(100), medium tinyint, home_address varchar(250),
FOREIGN KEY fk_cal(text)
REFERENCES description(text)
on update cascade
on delete cascade);

ALTER TABLE calendar
add constraint medium
FOREIGN KEY (medium)
REFERENCES priority(medium)
on update cascade
on delete cascade;

ALTER TABLE calendar
add constraint home_address
FOREIGN KEY (home_address)
REFERENCES place(home_address)
on update cascade
```

```
on delete cascade;
```

In the table titled Guest, we included 6 attributes with titles and data types as follows. Email varchar(45), contact_number varchar(20), fname varchar(45), lname varchar(45), nguests varchar(45), and new int. The primary key is email and there is no foreign key for this table. This table allows a guest user to view the entries. The following code shows how the table was created.

```
create table Guest(email varchar(45) primary key, contact_number varchar(20),
fname varchar(45), lname varchar(45), nguests varchar(45), new int);
```

In the table titled User, we included seven attributes with titles and data types as follows. User_name varchar(10), email varchar(45), password varchar(10), fname varchar(45), lname varchar(45), recovery_email varchar(45), and new int. The primary key is user_name and the foreign key is new. This table allows a user to access and edit various entries as well as log into the DMS. The following code shows how the table was created as well as the foreign keys used.

```
create table User(user_name varchar(10) primary key, email varchar(45),
password varchar(10), fname varchar(45), lname varchar(45),
recovery_email varchar(45), new int,
FOREIGN KEY fk_cal(new)
REFERENCES calendar(new)
on update cascade
on delete cascade);
```

9. Loading Data and Performance Enhancements

As we were trying to insert the instances into the table place, we received an error due to the foreign key constraints. The following code is the error that was returned.
Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails(‘dms_project’. ‘Place’, CONTSRAINT ‘place_ibfk_1’ FOREIGN KEY(‘morning’) REFERENCES ‘time’ (‘Morning’) ON DELETE CASCADE ON UPDATE CASCADE)
We got around the foreign key constraints by setting the foreign key constraint checks equal to zero and then running the insert code. After the insert code ran, we then set the foreign key constraint checks equal to one.

Another insertion error that occurred was trying to have a duplicate primary key. However, the primary key is supposed to be unique and not repeated. The following code shows the error that was returned.
Error Code: 1062. Duplicate entry ‘3’ for key ‘duration.PRIMARY’
To fix the insertion error, we changed the values of the primary key, duration, to all be unique values.

Another insertion error that occurred was trying to run the calendar table. It was not reading the proper date started in the table.
Error Code: 1292. Incorrect date value: '2009' for column 'view' at row 1
To fix this insertion error we added quotation marks around the view attribute in order for the calendar table to run.

Relating to the unique key we did not run into any insertion errors when running the reminder table for example null values appeared for the new attribute.
10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0
Since the attribute new is not a primary key it will output a null. No issues were found in the other tables with attributes that are not primary keys.

Before we could begin inserting information, we had to specify what schema to use.
use dms_project;

Next, we began inserting information into the table user, guest, duration, and date. We started off by inserting each row individually. However, we decided to optimize our data insertion by inserting the rest of the rows at the same time. The code is shown below.

```
describe user;
insert into user(user_name, email, password,
fname, lname, recovery_email) values('Sam32', 'sam.weston32@gmail.com',
'Rocks_1!', 'Sam', 'Weston', 'sam32@yahoo.com');

insert into user(user_name, email, password,
fname, lname, recovery_email) values('John2002', 'john_da_bomb@gmail.com',
'Bombsquad', 'John', 'Doe', 'jman@hotmail.com');

insert into user(user_name, email, password,
fname, lname, recovery_email) values('Bob', 'bobthebuilder@gmail.com',
'Building10','Bob', 'Builder', 'builder101@outlook.com'),
('Greg1', 'gregthegreat@gmail.com', 'Gregrules', 'Greg', 'Duncan', 'duncandill@hotmail.com'),
('Jenny66', 'jennyday1@hotmail.com', 'Teaching6', 'Jenny', 'Day', 'day66@gamil.com'),
('Ellie3', 'ellieg3@gmail.com', 'fanfics33', 'Ellie', 'Gahan', 'elliereads@yahoo.com'),
('Alexis210', 'alexisharman@gmail.com', 'Cherries2', 'Alexis', 'Harman',
'aharman1@hotmail,com'),
('Tania518', 'tania518@hotmail.com', 'taniaAte1', 'Tania', 'Hernandez-Martinez',
'thm518@yahoo.com'),
('Daniella6', 'dhboulos@gmail.com', 'Cats4ever', 'Daniella', 'Boulos',
'daniella.boulos1@hotmail.com'),
('Hamish5', 'thebesthamish@gmail.com', 'Theater5', 'Hamish', 'Swanson',
'hstheater@yahoo.com');

insert into guest(email, contact_number, fname, lname) values(
'jttheater87@gmail.com', 973-829-5153, 'James', 'Thompson'),
('egpoggi1@gmail.com', 631-338-6985, 'Emma', 'Poggi'),
('jennastudios@gmail.com', 516-974-3548, 'Jenna', 'Staufenberg'),
('cfilms3@yahoo.com', 732-882-6380, 'Colin', 'Brennan'),
('colakuphotos@gmail.com', 201-496-1866, 'Adrian', 'Colaku'),
('kielinicholls@gmail.com', 201-989-3416, 'Kieli', 'Nicholls'),
('dantheman@yahoo.com', 973-201-732, 'Daniel', 'Mason'),
('phoebes87346@gmail.com', 603-552-4536, 'Phoebe', 'Castellano'),
('miad1297@gmail.com', 631-235-8942, 'Mia', 'DeSimone'),
('jonjeffrey2@gmail.com', 732-874-3295, 'Jonathon', 'Russo');

insert into duration (Hour, Minute, Seconds, Month, Day) values
```

```
(2, 30, 20, 01, 04),
(3, 32, 15, 01, 07),
(4, 55, 10, 03, 12),
(1, 22, 5, 03, 22),
(6, 42, 11, 04, 08),
(7, 10, 40, 04, 16),
(8, 15, 10, 11, 12),
(9, 40, 4, 11, 02),
(5, 0, 0, 12, 02),
(10, 26, 14, 12, 25);
select * from duration;

insert into date (Year, Month, Day, Recurring, Time_Time_Zone) values
(2014, 01, 04, "Daily", "8 EST"),
(2015, 01, 07, "Weekly", "10 EST"),
(2016, 03, 12, "Weekly", "7 EST"),
(2017, 03, 22, "Daily", "11 EST"),
(2018, 04, 08, "Weekly", "5 EST"),
(2019, 04, 16, "Daily", "9 EST"),
(2020, 11, 12, "Weekly", "4 EST"),
(2021, 11, 02, "Daily", "2 EST"),
(2022, 12, 02, "Annually", "6 EST"),
(2023, 12, 25, "Annually", "3 EST");
select * from date;
```

Next, we inserted information into the table description, but we inserted each row differently because for each query we inserted a different number of values.

```
Insert into Description(text, underline, new) values('This calendar contains vet visits for Ollie',
'This calendar contains vet visits for Ollie', '1');

Insert into Description(text, bold, italicize, underline, new ) values("Doctor's Appointments",
"Doctor's Appointments", "Doctor's Appointments", "Doctor's Appointments", "2");

Insert into Description(text, bold, underline, new) values('Hair Appointments for this year', 'this',
'Hair Appointments', '3');

Insert into Description(text, bold, italicize, new ) values('THIS IS MY SHOPPING
CALENDAR', 'THIS IS MY SHOPPING CALENDAR', 'THIS IS MY SHOPPING
CALENDAR', '4');
```

```
Insert into Description(text , italicize, underline, new ) values('These are my work meetings', 'work', 'These are my work meetings', '5');

Insert into Description(text , underline, new ) values('These are the days I have softball and ice skating trainings', 'These are the days I have softball and ice skating trainings', '6');

Insert into Description(text, bold , italicize , underline , new ) values('This is my vacation calendar', 'vacation', 'vacation', 'vacation', '7');

Insert into Description(text, bold, underline, new) values('This includes my work schedule', 'This includes my work schedule', 'This includes my work schedule', '8');

Insert into Description(text, bold, italicize, new ) values('My schedule for work this month', 'work', 'work', '9');

Insert into Description(text,bold, underline, new) values('Concerts I am going to', 'Concerts', 'Concerts I am going to', '10');
```

Next we began inserting information into the tables place, time, and calendar. When we originally attempted to insert our information into the tables it wasn't working because of the foreign key constraints. To get around this, we set the foreign key checks to zero then inserted the information and finally set the foreign key checks back to one.

```
set foreign_key_checks=0;
insert into place(restaurant, home_address, school, work, other, morning) values
('Shadows on the Hudson', '26 Main St.', 'Wayne Hills High School', 'Avis Budget Group', 'Yellowstone National Park', 8),
('Applebees', '33 Einfield Lane', 'Marist College', 'IBM', 'National History Musem', 10),
('Chanos', '776 Alpine Grove', 'N/A', 'Apple Inc.', 'N/A', 7),
('TGI Fridays', '84 Stewart Circle', 'DePaul High School', 'Amazon Inc.', 'Liberty Science Center', 11),
('Cosimos', '57 May Rd.', 'N/A', 'Google Inc.', 'Niagra Falls', 5),
('Outback Steakhouse', '12 Willowbrook Blvd.', 'New Jersey Institute of Technology', 'IBM', 'N/A', 9),
('Chick-fil-a', '123 Sesame St', 'American University', 'Los Angeles Police Department', 'N/A', 4),
('Stardust Diner', '70 Alter Ave', 'University of Pittsburgh', 'Hollister', 'N/A', 2),
('Anthony Francos', '928 Anchor Circle', 'N/A', 'Palace Diner', 'Hudson River', 6),
('Chipotle', '590 Gray RD', 'Quinnipac University', 'MidHudson Regional Hospital', 'N/A', 3);
set foreign_key_checks=1;
```

```
set foreign_key_checks=0;
insert into time ( Morning, Time_Zone, Recurring, All_day, hour) values
(8, "EST", "Daily", 24, 2),
(10, "EST", "Weekly", 24, 3),
(7, "EST", "Weekly", 0, 4),
(11, "EST", "Daily", 24, 1),
(5, "EST", "Weekly", 0, 3),
(9, "EST", "Daily", 24, 5),
(4, "EST", "Weekly", 0, 1),
(2, "EST", "Daily", 0, 3),
(6, "EST", "Annually", 0, 5),
(3, "EST", "Annually", 0, 2);
set foreign_key_checks=1;
select * from duration;

set foreign_key_checks=0;
insert into Calendar (new, share, edit, collaborate,
view , text , home_address) values
('1', 'Yes', 'Yes', 'Yes', '2014-01-04', 'vet visits', '33 Einfield lane'),
('2', 'Yes', 'Yes', 'Yes', '2015-01-07', 'Doctor Appointment', '26 Main St.'),
('3', 'No', 'Yes', 'No', '2016-03-12', 'Hair Appointment', '776 Alpine grove'),
('4', 'Yes', 'Yes', 'Yes', '2017-03-12', 'SHOPPING', '84 Stewart circle'),
('5', 'No', 'Yes', 'No', '2018-04-08', 'Meetings', '12 Willowbrook Blvd.'),
('6', 'No', 'Yes', 'No', '2019-04-16', 'Trainings', '57 May Rd.'),
('7','Yes', 'Yes', 'Yes', '2020-11-12', 'Vacation', '123 Sesame St'),
('8', 'No', 'Yes', 'No', '2021-11-02', 'work', '70 Alter Ave'),
('9', 'No', 'Yes', 'No', '2022-12-02', 'work', '928 Anchor Circle' ),
('10', 'Yes', 'Yes', 'Yes', '2023-12-25', 'Concert', '590 gray RD');
set foreign_key_checks=1;
select * from calendar;
```

Finally, we inserted the information into the table's reminder and priority. Once the information was inserted into the tables, we had finished inserting all of the information.

```
insert into Reminder(Hour, Day, Year, Month) values
(1, 03, 2014, 01),
(2, 07, 2015, 01),
(3, 11, 2016, 03),
(4, 22, 2017, 03),
(5, 07, 2018, 04),
(6, 16, 2019, 04),
```

```
(7, 11, 2020, 11),
(8, 02, 2021, 11),
(9, 01, 2022, 12),
(10, 25, 2023, 12);

insert into priority(high, medium, low, free, new) values
('N/A', 2, 'N/A', 'No', 1),
('N/A', 3, 'N/A', 'No', 2),
('N/A', 4, 'N/A', 'No', 3),
('N/A', 5, 'N/A', 'No', 4),
('N/A', 6, 'N/A', 'No', 5),
('N/A', 7, 'N/A', 'No', 6),
('N/A', 8, 'N/A', 'No', 7),
('N/A', 9, 'N/A', 'No', 8),
('N/A', 10, 'N/A', 'No', 9),
('N/A', 11, 'N/A', 'No', 10);
```

As we began to insert all of our information into each table, we realized that we had to insert ten different instances into ten different rows, which was going to be a lot of work. Instead we decided to insert all our information for each table at once. The following code is an example of one query where we optimized our data insertion by inserting every instance at once.

```
insert into time ( Morning, Time_Zone, Recurring, All_day, hour) values
(8, "EST", "Daily", 24, 2),
(10, "EST", "Weekly", 24, 3),
(7, "EST", "Weekly", 0, 4),
(11, "EST", "Daily", 24, 1),
(5, "EST", "Weekly", 0, 3),
(9, "EST", "Daily", 24, 5),
(4, "EST", "Weekly", 0, 1),
(2, "EST", "Daily", 0, 3),
(6, "EST", "Annually", 0, 5),
(3, "EST", "Annually", 0, 2);
```

When we were inserting each row individually it took 0.016 seconds. However, when we inserted each row at the same time it took 0.000 seconds.

The First Normal Form

For the first normal form we use the command select * from. We use this code to show what the tables consist of in our database. Each cell in the table contains a single value, and each record is unique which confirms it follows the first normal form.

The Second Normal Form

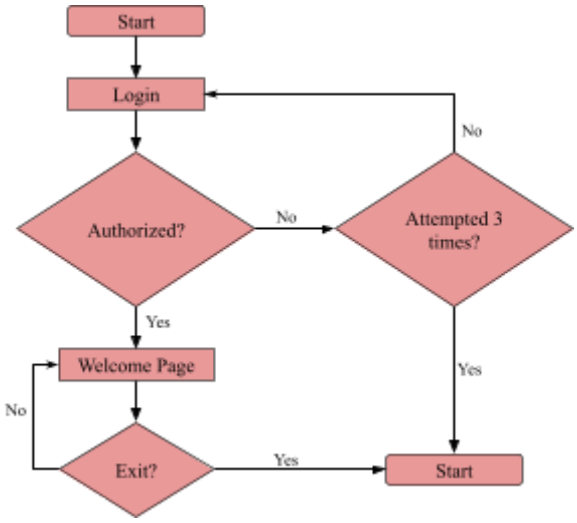
For the second normal form we check the tables using select * from. All of the attributes are dependent on the primary key. There are no attributes that are partially dependent on the primary key which confirms it follows the first and second normal form.

The Third normal form

For the third normal form all the attributes solely rely on the primary key. The EER diagram would not connect as it does if the attributes did not solely rely on a primary key. Without the primary key all other attributes cannot be defined.

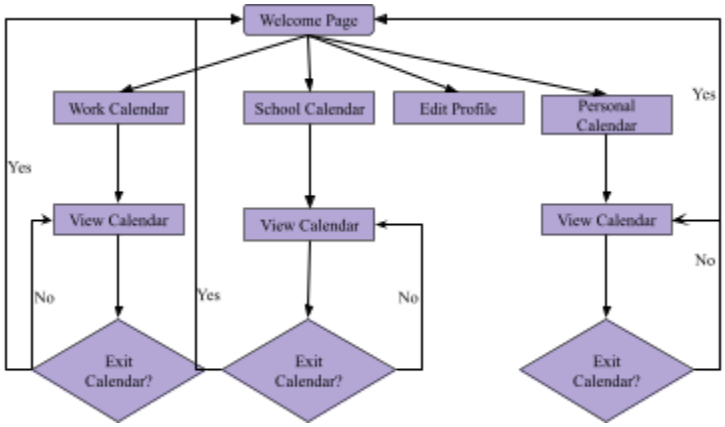
10. Application Development

Log in



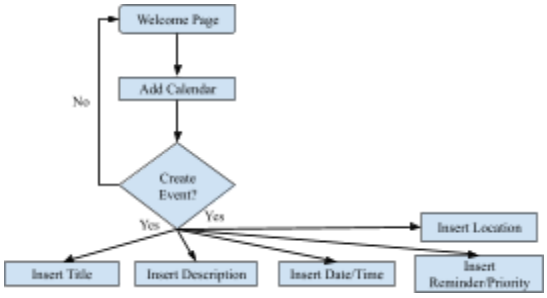
When starting the page the user will be prompted with a login page. The login page will ask for the username and password of the user in text. If the user types in the correct username and password they will be authorized to view the welcome page. The user can exit the welcome page when he or she desires and will be redirected to the login page. However if the user types in an incorrect username and or password they will be allowed to have 3 attempts. After each incorrect attempt, the user will be redirected to the login page.

Main Menu



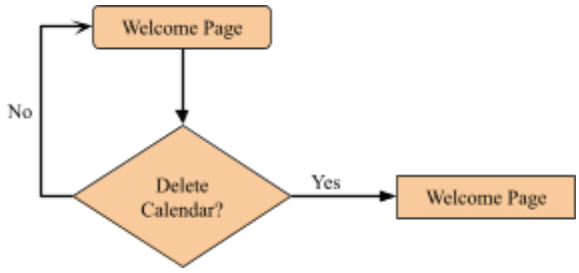
When the user is directed to the welcome page he or she will have the option to decide between three calendars; a work calendar, school calendar, and or a personal calendar and the option to edit their profile. For example if the user decides to choose the work calendar they will be directed to view calendar. If they decide to exit the calendar the user will be directed back to the welcome page.

Add Calendar



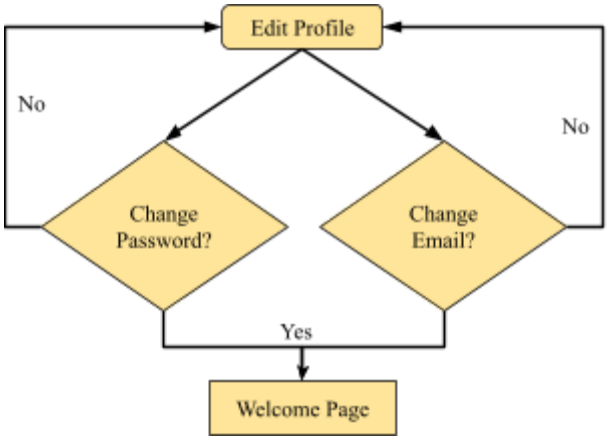
When the user is viewing the calendar they will then be prompted to add to the calendar. Once the user decides to add to their calendar they will be prompted to create an event. If the user does not decide to create an event they will be directed back to view calendar. If the user decides to create an event there will be directed to type in the title, description, date/time, reminder/priority, and location of the event.

Remove Calendar



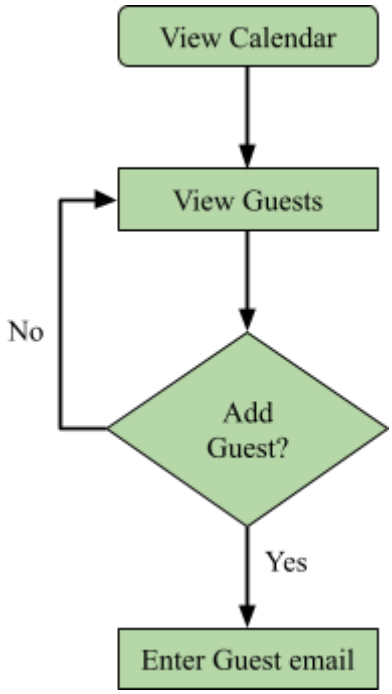
When the user is viewing the calendar they will have the option to delete the calendar. If they choose to delete the calendar they will be directed to the welcome page. If the user does not decide to delete the calendar they will be directed back to view calendar.

Edit Profile



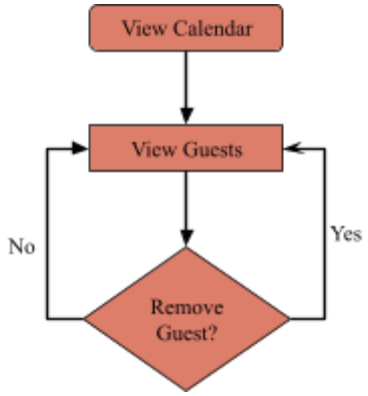
When the user edits their profile they will have the option to change their password and or their email. If the user decides to change their password and or email they will be brought to the welcome page. If the user does not decide to change their password and or email they will be brought to edit profile.

Create New User



When the user is viewing their calendar they can view the guests that have access to their calendar. If the user decides to add a guest to their calendar they will be directed to enter the guests email. If the user decides to not add a guest to their calendar they will be directed back to view guests.

Remove a User



When the user is viewing their calendar they can view the guests that have access to their calendar. If the user decides to delete a guest they will be prompted to do so and will be directed back to the view guests page.

While we have seven different graphical user experience designs, most of them only use one table. With that being said, we only had to create two different views since there were only two that required the use of two different tables. The first view we created was the view_guests, which combined the tables of calendar and guest so the user can view the guests for each calendar. This view returns the columns of fname, name, and email from the table of guest, and new from the table of calendar.

```
use dms_project;
create view view_guests as
select g.fname, g.lname, g.email, c.new from guest g
join calendar c on c.new = g.new;
```

The second view we created was view_user, which combined the tables of user and calendar so the user can view the users for each calendar and allow the user to be able to log in. This view returns the columns of user_name, email, and password from the table of user and new from the table of calendar..

```
create view view_user as
select u.user_name, u.email, concat(substr(u.password, 1,2), '*****'), c.new from user u
join calendar c on u.new = c.new;
```

11. Graphical User Interface

Connection to database

Before we were able to start coding each of our pages, we had to connect Python to MySQL. The following code was used in Python in order to connect it to MySQL. For MySQL, we just had to make sure that it allows paths, which we could check through the MySQL terminal.

```
import mysql.connector

db = mysql.connector.connect (
   host = "localhost",
   user = "root",
   passwd = "rootroot",
   database = "dms_project")

cursor = db.cursor()
cursor.execute('Select * from department')

for i in cursor:
   print(i)
```

Login page

Our login page consists of specifying a username and password to allow the user to log in into our database. We specified one of our user's username and password as an example to show how our user's would log in. The code is shown below.

```
from tkinter import *
from tkinter import messagebox

def everything():
   def Ok():
      uname = e1.get()
      password = e2.get()

      if(uname == "" and password == "") :
         messagebox.showinfo("", "Blank Not allowed")
```

```
    elif(uname == "Sam32" and password == "Rocks_1!"
        top.destroy()
        main.destroy()

    else :
        messagebox.showinfo("","Incorrect Username and Password")

main = Tk()
main.withdraw()
top = Toplevel()
top.title("Login")
top.geometry("300x200")

Label(top, text="UserName").place(x=10, y=10)
Label(top, text="Password").place(x=10, y=40)

e1 = Entry(top)
e1.place(x=140, y=10)

e2 = Entry(top)
e2.place(x=140, y=40)
e2.config(show="*")

Button(top, text="Login", command=Ok ,height = 3, width = 13).place(x=10, y=100)
top.mainloop()
```

Main menu page

Our Main Menu page can be accessed after the user logs in and it consists of nine buttons. The first three buttons are Work Calendar, School Calendar, and Personal Calendar. These buttons reference three different calendars that can be viewed by the user. The following five buttons are Edit Profile, Add Calendar, Delete Calendar, Create new user, and Remove a user. When the user clicks each of these buttons, it opens up a new window where the user can either complete the action or return to the Main Menu page. The last button is the Exit button. Once the user clicks this button, the whole module closes.

```
from tkinter import *
from tkinter import messagebox
from tkinter.ttk import *
import Login
import add_calendar
```

```
import EditProfile
import delete_calendar
import create_user
import remove_user
import WorkCalendar
import SchoolCalendar
import PersonalCalendar

Login.everything()

root = Tk()
root.title("Main")
root.geometry("500x500")

Label(root, text="Welcome!", font = ('times new roman', 12)).place(x=220, y=10)

# button style
style = Style()
style.configure('W.TButton', font = ('times new roman', 12),
                foreground= 'red', background = 'black', bd = ('black', 1))

#calendar buttons

bttn1 = Button(root, style = 'W.TButton',
               text="Work Calendar", command =
WorkCalendar.Work_Calendar_window).place(x=65, y=50)

bttn2 = Button(root, style = 'W.TButton',
               text="School Calendar", command=
SchoolCalendar.School_Calendar_Window).place(x=195, y=50)

bttn3 = Button(root, style = 'W.TButton',
               text="Personal Calendar", command=
PersonalCalendar.Personal_Calendar_Window).place(x=335, y=50)

#edit profile button
bttn4 = Button(root, style = 'W.TButton', text="Edit Profile",
               command = EditProfile.profile).place(x=65, y=90)

#add calendar button
```

```
bttn5 = Button(root, style = 'W.TButton', text="Add Calendar",
            command = add_calendar.add_calendar_window).place(x=195, y=90)

#delete calendar button
bttn6 = Button(root, style = 'W.TButton', text="Delete Calendar",
            command = delete_calendar.c_delete).place(x=335, y=90)
#create new user button
bttn7 = Button(root, style = 'W.TButton', text="Create New User",
            command = create_user.new_user).place(x=65, y=130)
#remove a user button
bttn8 = Button(root, style = 'W.TButton', text="Remove a User",
            command = remove_user.remove_user).place(x=195, y=130)
#exit button
bttn9 = Button(root, text="Exit", command=root.destroy).pack( side = BOTTOM )


root.mainloop()
```

Action Pages

Search Page

We have three different search pages, school calendar, work calendar, and personal calendar. The following code is the code for our school calendar, but they're all relatively the same. It allows you to type in a value for 'new', which is simply just a number that is in our calendar. After you type in the number, you click the button 'select' and every entry that goes with that number pops up for you to see in the textbox.

```
from tkinter import *
import tkinter as tk
from tkinter import messagebox
import mysql.connector

db = mysql.connector.connect(
host = "localhost",
user = "root",
passwd = "rootRoot@61",
database = "dms_project"
)
cur = db.cursor()

def everything():
    def school_calendar():
```

```
    new = entNew.get()

    select = f"select * from school_calendar where new='{new}';"
    print(select)
    cur.execute(select)

    txtOutcome.delete('1.0', END)
    head=["New","Share", "Edit", "Collaborate","Home_address","School"]
    head="\t".join(head)+"\n"
    txtOutcome.insert('1.0',head)
    for i in cur:
        string_list=[str(element) for element in i]
        print(string_list)
        txtOutcome.insert(END,"\t".join(string_list)+"\n")


main = Tk()
main.withdraw()
top = Toplevel()
top.title("School Calendar")
top.geometry("500x500")

lblNew = Label(top, text="New").place(x=10, y=10)

entNew = Entry(top)
entNew.place(x=140, y=10)

txtOutcome = Text(top, height=10, width = 50)
txtOutcome.place(x=10, y=115)


bttn1 = Button(top, text="Welcome Page", command= top.destroy,
            font = ('times new roman', 12), foreground= 'red',
            background = 'white').place(x=150, y=375)

bttn2 = Button(top, text="Search", command= school_calendar,
            font = ('times new roman', 12), foreground= 'red',
            background = 'white').place(x=175, y=290)

top.mainloop()
```

Insertion Page

We have two different add pages, add calendar and create a user. The following code is for adding a calendar, but, again, they are very similar. To add a calendar, you type a value into ‘new’, which is a number for a calendar. Then, you click the add button and that would add all the entries that correspond with that number calendar. You can even check to see if it was added by clicking the ‘search’ button and every entry will come up along with the one that was added.

```
from tkinter import *
import tkinter as tk
from tkinter import messagebox
import mysql.connector

db = mysql.connector.connect(
host = "localhost",
user = "root",
passwd = "rootRoot@61",
database = "dms_project"
)
cur = db.cursor()

def everything():
    def add():
        new = entNew.get()
        share = entShare.get()
        edit = entEdit.get()
        collaborate = entCollaborate.get()
        home_address = entHome_address.get()

        add = f"insert into calendar(new,share,edit,collaborate,home_address)
values(%s,%s,%s,%s,%s);"
        values = (new, share, edit, collaborate, home_address)
        print(add)
        cur.execute(add, values)
        db.commit()

    def select():
        select = f"select new,share,edit,collaborate,home_address from calendar"
        print(select)
        cur.execute(select)
```

```
    txtOutcome.delete('1.0', END)
    head=["New","Share", "Edit", "Collaborate","Home_address"]
    head="\t".join(head)+"\n"
    txtOutcome.insert('1.0',head)
    for i in cur:
        string_list=[str(element) for element in i]
        print(string_list)
        txtOutcome.insert(END,"\t".join(string_list)+"\n")


main = Tk()
main.withdraw()
top = Toplevel()
top.title("Add Calendar")
top.geometry("500x500")

lblNew = Label(top, text="New").place(x=10, y=10)
lblShare = Label(top, text="Share").place(x=10, y=40)
lblEdit = Label(top,text="Edit").place(x=10, y=70)
lblCollaborate = Label(top,text="Collaborate").place(x=10, y=100)
lblHome_address = Label(top,text="Home Address").place(x=10, y=140)

entNew = Entry(top)
entNew.place(x=140, y=10)
entShare = Entry(top)
entShare.place(x=140, y=40)
entEdit=Entry(top)
entEdit.place(x=140, y=70)
entCollaborate = Entry(top)
entCollaborate.place(x=140, y=100)
entHome_address = Entry(top)
entHome_address.place(x=140, y=140)

txtOutcome = Text(top, height=10, width = 50)
txtOutcome.place(x=10, y=170)


bttn1 = Button(top, text="Welcome Page", command= top.destroy,
            font = ('times new roman', 12), foreground= 'red',
```

```
            background = 'white').place(x=150, y=415)

    bttn2 = Button(top, text="Add", command= add,
            font = ('times new roman', 12), foreground= 'red',
            background = 'white').place(x=140, y=350)

    bttn3 = Button(top, text="View", command= select,
            font = ('times new roman', 12), foreground= 'red',
            background = 'white').place(x=230, y=350)

    top.mainloop()
```

Modification Page

Our modification page is called edit profile. On this page you can update your user information such as your first name. As shown by the code below, you type in the original first name that you want to update for the user and then you type in the first name that you want to change it to. After you fill that in, you click the ‘update’ button and your information will change. You can even hit the ‘view’ button to check if your information was updated.

```
from tkinter import *
import tkinter as tk
from tkinter import messagebox
import mysql.connector

db = mysql.connector.connect(
host = "localhost",
user = "root",
passwd = "rootRoot@61",
database = "dms_project"
)
cur = db.cursor()

def everything():
    def edit_profile():
        fname = entfname.get()
        fname1 = entfname1.get()

        modify = f"update user set fname='{fname1}' where fname='{fname}';"
        print(modify)
        cur.execute(modify)
```

```
def select():
    fname1 = entfname1.get()
    select=f"select * from user where fname='{fname1}';"
    print(select)
    cur.execute(select)

    txtOutcome.delete('1.0', END)
    head=["User Name", "Email", "Password", "First Name", "Last Name"]
    head="\t".join(head)+"\n"
    txtOutcome.insert('1.0',head)
    for i in cur:
        string_list=[str(element) for element in i]
        print(string_list)
        txtOutcome.insert(END,"\t".join(string_list)+"\n")


main = Tk()
main.withdraw()
top = Toplevel()
top.title("Edit Profile")
top.geometry("500x500")

lblfname = Label(top, text="First Name").place(x=10, y=10)
lblcolumnName = Label(top, text="New First Name").place(x=10, y=40)

entfname = Entry(top)
entfname.place(x=140, y=10)
entfname1 = Entry(top)
entfname1.place(x=140, y=40)


txtOutcome = Text(top, height=10, width = 50)
txtOutcome.place(x=10, y=115)


bttn1 = Button(top, text="Welcome Page", command= top.destroy,
            font = ('times new roman', 12), foreground= 'red',
            background = 'white').place(x=150, y=375)
```

```
    bttn2 = Button(top, text="Update", command= edit_profile,
                font = ('times new roman', 12), foreground= 'red',
                background = 'white').place(x=140, y=290)

    bttn3 = Button(top, text="View", command= select,
                font = ('times new roman', 12), foreground= 'red',
                background = 'white').place(x=230, y=290)

    top.mainloop()
```

Deletion Page

We have two different delete pages, delete calendar and delete user. The following code is for deleting a calendar, but, again, they are very similar. To delete a calendar, you type a value into ‘new’, which is a number for a calendar. Then, you click the ‘delete’ button and that would delete all the entries that correspond with that number calendar. You can even check to see if it was deleted by clicking the ‘search’ button and every entry will come up except for the one that was deleted.

```
from tkinter import *
import tkinter as tk
from tkinter import messagebox
import mysql.connector

db = mysql.connector.connect(
host = "localhost",
user = "root",
passwd = "rootRoot@61",
database = "dms_project"
)
cur = db.cursor()

def everything():
    def delete():
        new = entNew.get()

        delete = f"delete from calendar where new='{new}';"
        print(delete)
        cur.execute(delete)

    def select():
```

```
    select = f"select new,share,edit,collaborate,home_address from calendar;"
    print(select)
    cur.execute(select)

    txtOutcome.delete('1.0', END)
    head=["New","Share", "Edit", "Collaborate","Home_address"]
    head="\t".join(head)+"\n"
    txtOutcome.insert('1.0',head)
    for i in cur:
        string_list=[str(element) for element in i]
        print(string_list)
        txtOutcome.insert(END,"\t".join(string_list)+"\n")


main = Tk()
main.withdraw()
top = Toplevel()
top.title("Delete Calendar")
top.geometry("500x500")

lblNew = Label(top, text="New").place(x=10, y=10)

entNew = Entry(top)
entNew.place(x=140, y=10)

txtOutcome = Text(top, height=10, width = 50)
txtOutcome.place(x=10, y=115)


bttn1 = Button(top, text="Welcome Page", command= top.destroy,
            font = ('times new roman', 12), foreground= 'red',
            background = 'white').place(x=150, y=375)

bttn2 = Button(top, text="Delete", command= delete,
            font = ('times new roman', 12), foreground= 'red',
            background = 'white').place(x=140, y=290)

bttn3 = Button(top, text="View", command= select,
            font = ('times new roman', 12), foreground= 'red',
            background = 'white').place(x=230, y=290)
```

```
    top.mainloop()
```

Print All Data

The following code is the same as the search page. This code searches and prints the data that the user would like to view. It's multi-functional and allows for fewer action pages since it performs both functions.

```
from tkinter import *
import tkinter as tk
from tkinter import messagebox
import mysql.connector

db = mysql.connector.connect(
host = "localhost",
user = "root",
passwd = "rootRoot@61",
database = "dms_project"
)
cur = db.cursor()

def everything():
    def school_calendar():
        new = entNew.get()

        select = f"select * from school_calendar where new='{new}';"
        print(select)
        cur.execute(select)

        txtOutcome.delete('1.0', END)
        head=["New","Share", "Edit", "Collaborate","Home_address","School"]
        head="\t".join(head)+"\n"
        txtOutcome.insert('1.0',head)
        for i in cur:
            string_list=[str(element) for element in i]
            print(string_list)
            txtOutcome.insert(END,"\t".join(string_list)+"\n")
```

```
main = Tk()
main.withdraw()
top = Toplevel()
top.title("School Calendar")
top.geometry("500x500")

lblNew = Label(top, text="New").place(x=10, y=10)

entNew = Entry(top)
entNew.place(x=140, y=10)

txtOutcome = Text(top, height=10, width = 50)
txtOutcome.place(x=10, y=115)


bttn1 = Button(top, text="Welcome Page", command= top.destroy,
            font = ('times new roman', 12), foreground= 'red',
            background = 'white').place(x=150, y=375)

bttn2 = Button(top, text="Search", command= school_calendar,
            font = ('times new roman', 12), foreground= 'red',
            background = 'white').place(x=175, y=290)

top.mainloop()
```

12. Conclusion

The Diary Management System (DMS) stores calendar entries for the users. Throughout the semester we have learned how to create a database and graphical user interface using Python which has been implemented throughout our project. The first step in our diary management system was to create an Entity-Relationship Model (ER) and an Enhanced Entity-Relationship Model (EER). By way of the ER and EER diagrams, we were able to create our tables in MySQL with no problems. Once our tables were created it was time to enable a graphical user interface using Python. We learned how to write different lines of code in Python to connect our tables from MySQL. Insert queries, for one, create entry boxes for the user to insert data and design buttons that are user-friendly. Though we faced some problems while writing the queries that were inserted into Python, we learned how to resolve these issues. Collaborating with one another, referencing the examples the professor provided on his website, and utilizing the office hours, allowed us to resolve the problems and continue moving forward on our project.

Additionally, we considered three additional functionalities we would like to provide for our users. The first addition is adding images. Images would improve the users viewing experience and provide an aesthetically pleasing appearance. The second edition is changing the format of the page. This includes the page color, button style, and more. The format would add to providing the user with a great viewing experience. Lastly, we would like to implement notifications to be sent to the user. For example, reminders would appear on the notification banner of the user's device the system is installed on.

13. References

[1]Google Calendar
[2]Google Calendar Pros and Cons
[3]Gmail Reviews - Pros & Cons, Ratings & more | GetApp
[4]Lucidchart
[5]My SQL
[6]Python