DailyArt Art Gallery

Database Design (COP 4710)

Kamryn Suh, Akash Makati, Zhihao Zhang

1. About the Project

Our group designed and developed an online enterprise management system with nontrivial functionality which was backed by **SQLite** as our database. Essentially, our online enterprise management system focuses on a fabricated art gallery that serves users with multiple functionalities.

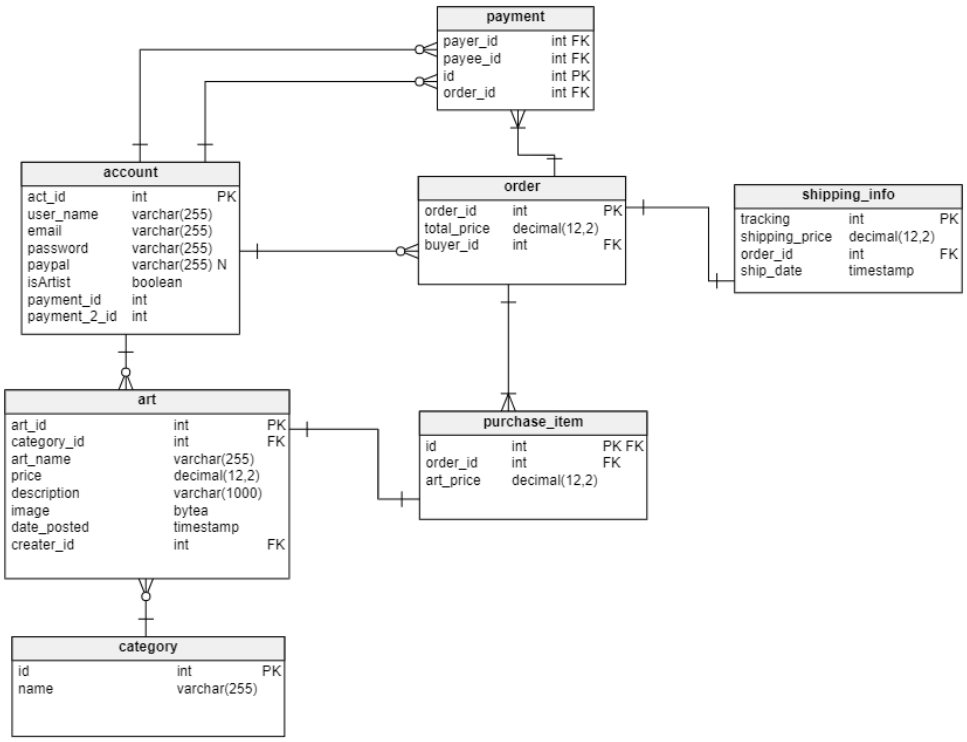
The idea we had was that our supposed website known as DailyArt would be able to serve as a social media platform for those interested in buying or selling art. We ended up not using a website due to some complications and used text-based pages. DailyArt has the options of buying and selling art, but also the options of making and talking to friends! With our added friends feature combined with the art-based enterprise, we thought it would be a great idea that would be both interesting and suitable for our project.

To implement DailyArt for our project, we used mainly **Python** to code all of our mainframe as well as all the functions with a combination of **SQL** statements to make any changes to the **SQLite** database that was connected. We initially attempted using **React** with **HTML** and **CSS** in combination with a **PostgreSQL** database but we were having difficulties with connecting the database and implementing the web pages so we veered off from it.

Directions to run:

1. Ensure you are in the correct directory. 2. Run the program by running 'login_main.py'
3. Need to run in python. No extra dependency is needed. We used PyCharm (highly recommended).

2. Database Design



Above is our ER-diagram, not much has to be explained about it but it shows the relational schemas for our enterprise and shows how each entity has relationships with other entities. We could not fully implement this into our project and ended up making a few changes to make things more simple which was to be expected.

3. Tables

1. **user**
   user(userID INTEGER PRIMARY KEY, username VARCHAR(20) NOT NULL, password VARCHAR(20) NOT NULL, first_name VARCHAR(20) NOT NULL, last_name VARCHAR(20) NOT NULL, tier_name VARCHAR(20) NOT NULL);
   Primary key: userID
   Foreign key: none
   This “user” table serves the purpose of storing the account management information for our users. It includes attributes of the username, password, first name, last name, and a tier name. The tier name correlates to what “member” they are when they create an account, which is either “default” or “core.” This table will act as a fundamental relation for the rest of our tables.
2. **profiles**
   profiles(userID INTEGER PRIMARY KEY, username VARCHAR(20) NOT NULL, title VARCHAR(20), location VARCHAR(20) NOT NULL, specialty VARCHAR(50) NOT NULL, about VARCHAR(250), FOREIGN KEY(userID) REFERENCES user(userID));
   Primary key: userID
   Foreign key: userID
   This “profiles” table serves the purpose of storing the user’s profile information. It includes attributes of the username, title, location, specialty, and about. The foreign key references the “user” table, creating a relation between the two.
3. **art**
   art(userID INTEGER PRIMARY KEY, username VARCHAR(20) NOT NULL, title VARCHAR(20), year INTEGER, medium VARCHAR(60), price INTEGER, description VARCHAR(250), FOREIGN KEY(userID) REFERENCES user(userID));
   Primary key: userID
   Foreign key: userID
   This “art” table serves the purpose of storing the art gallery information. It includes attributes of the username, title, year, medium, price, and description. The foreign key references the “user” table, creating a relation between the two.
4. **friends**
   friends(username VARCHAR(20) NOT NULL, stranger VARCHAR(20) NOT NULL, status VARCHAR(20));
   Primary key: none
   Foreign key: none

This “friends” table serves the purpose of storing the friends information between two users. It includes attributes of the username, stranger (person you are requesting a friend request), and status (which includes sentpending, acceptpending, and friend statuses).

5. **messageFriend**

   messageFriend(sender VARCHAR(20) NOT NULL, receiver VARCHAR(20) NOT NULL, message VARCHAR(20) NOT NULL, status VARCHAR(20));

   Primary key: none

   Foreign key: none

   This “messageFriend” table serves the purpose of storing the messaging information between two users/friends. It includes attributes of the sender, receiver, message, and status (which includes sent and deleted statuses).

6. **buyInfo**

   buyInfo(userID INTEGER PRIMARY KEY, username VARCHAR(20) NOT NULL, title VARCHAR(20) NOT NULL, price INTEGER, payment VARCHAR(20), comment VARCHAR(250), status VARCHAR(20), FOREIGN KEY(userID) REFERENCES user(userID));

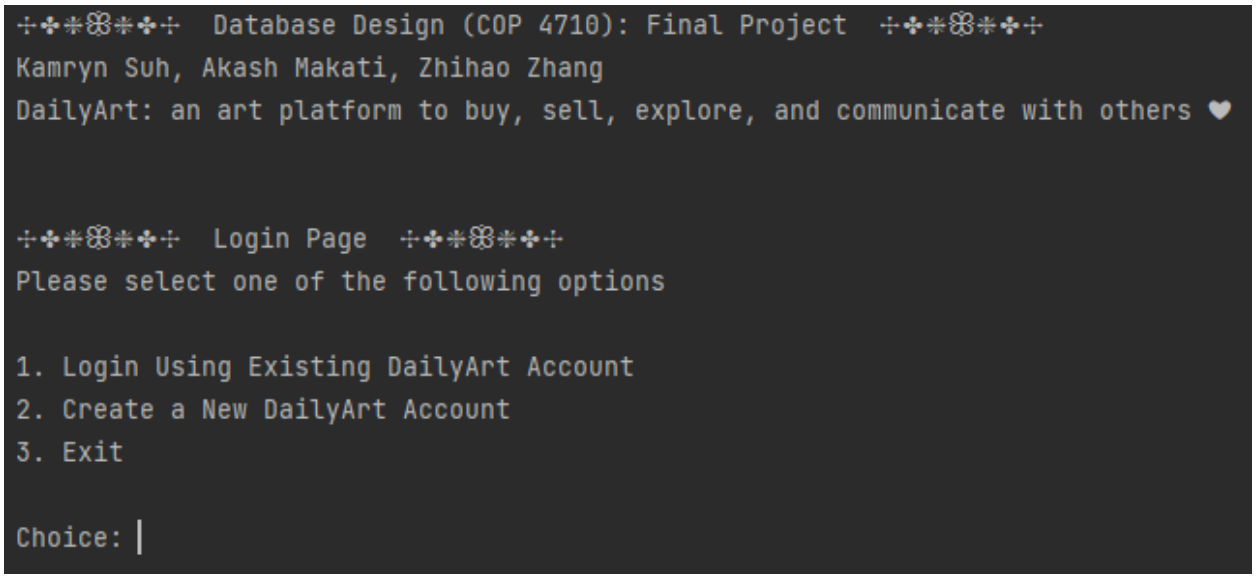
   Primary key: userID

   Foreign key: userID

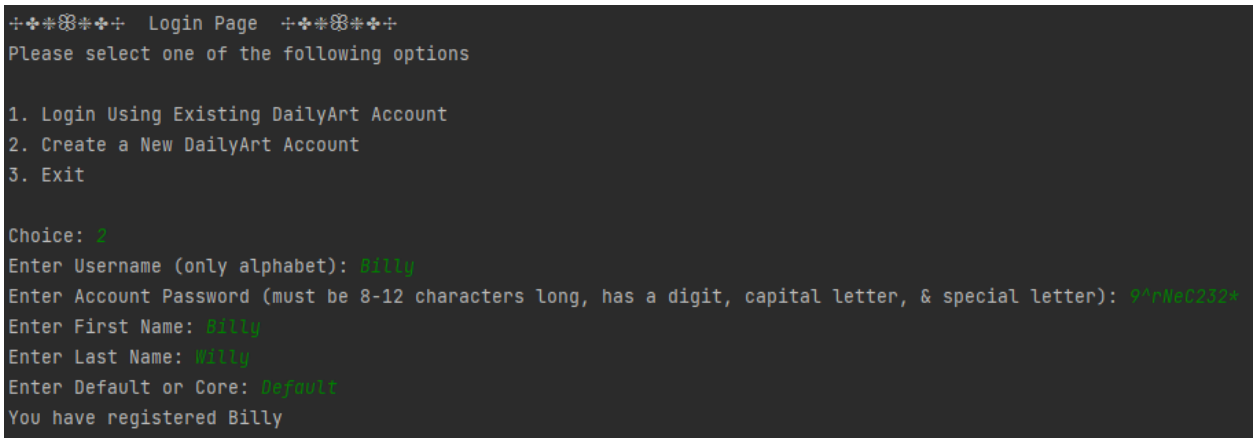
   This “buyInfo” table serves the purpose of storing the purchasing aspect of the art gallery information. It includes attributes of the username, title, price, payment, comment, and status (which includes the applied, deleted, and saved statuses). The foreign key references the “user” table, creating a relation between the two.

4. Interface

When running the program, all users are prompted to the login page UI and a header displaying our class information. This tells a short statement about our DailyArt website. The login page UI includes 3 options: login using existing DailyArt account, create a new DailyArt account, or exit the program completely.



You are unable to log into an account if there is no created account, so the user must create an account in order to interact with the program. When typing in choice 2 to create an account, the user will prompted with several questions:



For input validation, we made sure that the username is only alphabet and account password must follow the rules of: must be 8-12 characters long, has a digit, capital letter, and a special letter.

```
Choice: 2
Enter Username (only alphabet): 123

Please type a valid response.

Enter Username (only alphabet): Sam
Enter Account Password (must be 8-12 characters long, has a digit, capital letter, & special letter): Password
Make Sure Your Password has a Digit in it

Enter Account Password: Password5
Make Sure that the Password has a Special Letter in it
```

When logged into your user that you have selected, you will be prompted into the main menu interface that gives you a total of *16 options*:

```
Choice: 1
Existing Accounts (for demo purposes):
  1: Billy

Username: Billy
Password: 9^rNeC232*

1. Buy an Artwork Print
2. Sell an Artwork Print
3. Search for Artworks
4. List of Artworks on Market
5. List of Favorite Artworks
6. Create a Profile
7. Edit your Profile
8. Display a User Profile
9. Edit Artwork Information
10. Unlist an Artwork
11. Send a Friend Request
12. List Pending Requests
13. Show All my Friends
14. Send a Message
15. Find Someone You May Know
16. Exit

Pick an Option:
```

When selling an artwork print, we will choose *option 2*. This will prompt the user with several questions and if answered correctly (passes our input validation functions), then the print will be listed in the gallery.

```
Pick an Option: 2
Enter your artwork title: Shape
Enter the year it was made: 2005
Enter the medium you used: Watercolor
Enter your pricing (integer only): 900
Enter a description of the artwork: This contains so many shapes and sizes. Yay.

Your artwork has been listed.
```

If you choose *option 1* after, to purchase your own art, you will be prompted this following message:

```
Pick an Option: 1
1 . Shape

Select one of the artworks by number to purchase: 1
You cannot purchase an artwork you have created yourself, try purchasing for another artwork that you haven't created.
```

If you choose *option 1* on a different account, you will be prompted a different set of questions and can place your offer on the artwork.

```
Pick an Option: 1
1 . Starry

Select one of the artworks by number to purchase: 1
Enter the title of the artwork: Starry
Enter your pricing (integer only): 2500
Enter the payment option (E.g., Paypal, Cashapp, etc) : Paypal
Enter any additional comments/special requests: i love this artwork!

Your offer has been made.
```

If you choose *option 3*, to search for artworks, you will be prompted a few questions. First, it displays the “art gallery” where it will state the listed art prints and you can select which one you would like to learn more about. It will print the information that the seller inputted into the system. It will then prompt the user if they would like to favorite this artwork or go back to the main menu. If you favorite the art print, you can view it in *option 5*.

```
############## Art Gallery #############
You have currently purchased  0 art
1 . Shape

Select one of the artworks by number to learn more about: 1
Title:  Shape
Year:  2005
Medium:  Watercolor
Price:  900
Description:  This contains so many shapes and sizes. Yay.


Do you want to favorite this artwork to your list?
1. Favorite the artwork
2. Back to main menu
Type 1 or 2 here: 1
The artwork is favorited!
```

If you choose *option 5*, you can display the list of favorite artworks. It will also prompt you the option to unfavorite the artwork.

```
Pick an Option: 5
Your favorite artworks:

Title:  Shape

Do you want to unfavorite any artwork in your list?
1. Unfavorite the artwork
2. Back to main menu
Type 1 or 2 here: 1
Enter the artwork title you want to unfavorite: Shape
This artwork has been unfavorited.
```

If you choose *option 6*, it will prompt you several questions about making your own art profile. It also gives several examples for 2 of the questions to help the user when writing their profile.

```
Pick an Option: 6
Enter your level (E.g., Hobbyist, Professional, Student, None): Hobbyist
Enter your location: Asia
Enter your specialty (E.g., Digital Art, Traditional Art, Photography, None): Digital Art
Write your bio: I am a hobby artist. I make 0 money.

Your profile has been created.
```

If you choose *option 7*, you will be prompted for the profiles. It includes the attributes you can edit for your profile and an option to go back.

```
Pick an Option: 7
PROFILE:  (1, 'Billy', 'Hobbyist', 'Asia', 'Digital Art', 'I am a hobby artist. I make 0 money.\n')

1. Title
2. Location
3. Specialty
4. About
5. Go Back

Enter which to edit: 1
Enter your title (E.g., Student): Professional
Finished editing your user profile

1. Title
2. Location
3. Specialty
4. About
5. Go Back

Enter which to edit: 3
Enter your specialty: Traditional Art
Finished editing your user profile

1. Title
2. Location
3. Specialty
4. About
5. Go Back

Enter which to edit: 4
Rewrite your bio: I am now a professional artist. I still make 0 money.
```

If you choose *option 8*, it will display the user's selected profile. Since we edited/updated the previous iteration of Billy's profile, it will display the new, updated version of it. Before editing the profile, it would display how we originally inputted.

```
Pick an Option: 8
Type the username of the user you want to see the profile for or "exit" to return: Billy

---------- You are viewing  Billy Willy 's profile ----------
Title:  Professional
Location:  Asia
Specialty:  Traditional Art
About:  I am now a professional artist. I still make 0 money.
```

If you choose *option 9*, it will display the listed artworks in the gallery. It gives the prompt to select one of the following artworks to edit by number. This will update the information and we can check the updated version of the artwork by selecting option 3, which we will demonstrate in the next screenshot.

```
---------- Artworks ----------
Title:  Shape
Year:  2005
Medium:  Watercolor
Price:  900
Description of artwork:  This contains so many shapes and sizes. Yay.

Select one of the following to edit:
1 .  Shape
Select one of the artworks to edit by number: 1

1. Title
2. Year
3. Medium
4. Price
5. Description
6. Go Back

Enter which to edit: 2
Enter the year it was made: 2009

1. Title
2. Year
3. Medium
4. Price
5. Description
6. Go Back

Enter which to edit: 4
Enter your pricing (integer only): 1500
```

Updated version when selecting *option 3*:

```
############# Art Gallery #############
You have currently purchased  0 art
1 . Circle

Select one of the artworks by number to learn more about: 1
Title:  Circle
Year:  2009
Medium:  Watercolor
Price:  1500
Description:  This contains so many shapes and sizes. Yay.
```

If you choose *option 10*, you can unlist your artwork. This will prompt the user to select one of the artworks to delete by number (the screenshot shows an input validation as well).

```
---------- Artworks ----------
Title:  Circle
Year:  2009
Medium:  Watercolor
Price:  1500
Description of artwork:  This contains so many shapes and sizes. Yay.

Select one of the following to delete:
1 .  Circle
Select one of the artworks to delete by number: Circle

Please select a valid option.
Select one of the artworks to delete by number: 1
This artwork has been deleted:  Circle
```

Updated version when selecting *option 3*:

```
############# Art Gallery #############
You have currently purchased  0 art
No artworks have been posted
```

If you choose *option 11*, you can send a friend request to another user in the database.

```
Pick an Option: 11
Enter the username of the user you want to friend: Kamryn
You Have sent friend request to Kamryn
```

When logging into user Kamryn, we will be notified to accept or reject the friend request from Billy.

```
Username: Kamryn
Password: Password5!
You received a friend request.

Would you like to accept friend request from Billy
Enter 1 to Accept and 2 to Reject: 1
You have accepted friend request from, Billy
```

If you choose *option 12*, it will show you a list of your pending requests. This includes "your outgoing friend requests" and "your incoming friend requests." Since Billy sent a friend request to Kamryn, it will show Kamryn's name in the "your outgoing friend requests" list.

```
Pick an Option: 12
Your outgoing friend requests:
Kamryn
Your incoming friend requests:
```

If you choose *option 13*, you can show a list of all your friends. If there are no is friends, it will print out a statement "you have no connections." It will also prompt you if you want to remove a friend, you can select "yes" or "no."

```
Pick an Option: 13
Your friends:
      Kamryn
Would you like to remove a connection? (Yes / No): yes
Enter their username: Kamryn
You have removed  Kamryn
```

If you choose *option 14*, you can send a message. You will first be prompted to an option of "default member messaging" or "core member messaging." Default members can only send messages to friends. Core members can send messages to anyone in the system. In this example, Kamryn will send a message to Billy as a default member since she is friends with him.

```
Pick an Option: 14

1. Default Member Messaging
2. Core Member Messaging
3. Go back

Enter 1 as a Default member, Enter 2 as a Core member, Enter 3 to go back: 1
Enter the username of the user you want to send a message to: Billy
These are your friends that you can message:
      Billy
Enter your message: Hi Billy!!
You Have sent the message to Billy
```

If we log into Billy's account, we will receive a notification and display the message. You will be given 3 options: leave the message, send a message back, and delete the message. In this example, we will delete the message.

```
Username: Billy
Password: 9^rNeC232*
You have received a message in your inbox.
NOTIFICATION: You have received a message from Kamryn

Press 1 to read the message: 1
Message:  Hi Billy!!
Enter 1 to leave the message, 2 to send message back, 3 to delete the message: 3
You have removed the message from your inbox
```

If you choose option 15, you can find someone you may know in the database with 2 options, first & last name or username. For this example, we will do first & last name. It will prompt you several questions (if it can't find the first and last names, input validation will occur) and ask if you would like to send a friend request. For this example, Billy is already friends with Kamryn, so this will not work.

```
Pick an Option: 15

1. First & Last Name
2. Username
3. Exit

Select what you would like to search by: 1
Enter First Name: Billy
Enter Last Name: Willy
Users with that name  Willy :
Username:  Billy , Name:  Billy   Willy
Would you like to send a friend request? (Yes / No): yes
Enter their username: Billy
You've already sent a friend request to this user.
```

If you choose option 16, you simply exit out the program. This will print the statement “you have left DailyArt. Come back again!”

5. Queries

Here are 8 different queries we have implemented:

1. Query to select from the “friends” table where the status is equal to ‘friend’ so we are able to find where the “status” column has the ‘friend’ status

```
# Checking if they have them as a friend
find_friends = "SELECT * FROM friends WHERE username = ? AND stranger = ? AND status = 'friend'"
cursor.execute(find_friends, [(sender), (receiver)])
results = cursor.fetchall()
```

2. Query to insert the “buyInfo” table where it will insert the information that the user inputted including the inserted ‘applied’ status to be used later

```
cursor.execute(
    """
INSERT INTO buyInfo (username, title, price, payment, comment, status) VALUES(?,?,?,?,?,'applied')""",
    (username, number[choice - 1][2], price, payment, comment),
)
```

3. Query to delete from “buyInfo” table where it will unfavorite the artwork that is in the user’s list. It does this based on the “status” column where it has a ‘saved’ status

```
delete_save = "DELETE FROM buyInfo WHERE username = ? AND title = ? AND status = 'saved'"
cursor.execute(delete_save, [(username), (title1)])
db.commit()
```

4. Query to select from “buyInfo” table where the “status” column has a ‘deleted’ status. This serves as a deletion detector function

```
find_deleted = "SELECT * FROM buyInfo WHERE username = ? AND status = 'deleted'"
cursor.execute(find_deleted, [(username)])
```

5. Query to update from “buyInfo” table where it will update the status to ‘deleted’

```
update_status = ''' UPDATE buyInfo SET status = ? WHERE title = ?'''
cursor.execute(update_status, [("deleted"), (artTitle)])
db.commit()
```

6. Query to select from “friends” table where it will select the username with the status ‘acceptpending’. This is apart of the pending_friend function.

```
find_pending = "SELECT * FROM friends WHERE username = ? AND status = 'acceptpending'"
cursor.execute(find_pending, [(username)])
```

7. Query to select from “friends” table where it will select the username with the status ‘sentpending’. This is apart of the list_pending function.

```
find_sentRequests = "SELECT * FROM friends WHERE username = ? AND status = 'sentpending'"
cursor.execute(find_sentRequests, [(username)])
```

8. Query to update from “friends” table where it will set the status to ‘friend’ for both the users.

```
# both change status to "friend"
reject_request1 = ''' UPDATE friends SET status = ? WHERE username = ? AND stranger = ?'''
cursor.execute(reject_request1, [("friend"), (me), (stranger)])
reject_request2 = ''' UPDATE friends SET status = ? WHERE username = ? AND stranger = ?'''
cursor.execute(reject_request2, [("friend"), (stranger), (me)])
```

There are many other queries involved in the program but we selected a few that highlighted certain aspects of the tables.

6. Relations

We have three relations. These three relations include the tables: buyInfo, art, and profiles. These three tables use a foreign key (userID) that references the user table (userID). All of the tables mentioned use a primary integer key, which is userID. We have a one-to-many relationship with the user and profile tables, this is because many users are allowed to create their own profile, however, it can only be created once. We have a many-to-many relationship with the user and art tables, this is because many users can create/sell many art prints. We have a many-to-many relationship with the user and buyInfo table, this is because many users can purchase many art prints.