Final Project Report for TKimTMorril

Programming Language: Java (JDK 8 or higher recommended for modern features).

Database: MySQL (version 8 or later recommended).

IDE: IntelliJ IDEA

Database Client Tool: MySQL Workbench

Design: Used observer pattern for delegating tasks from an observer to its many listeners for separating UI logic, business logic, and database interactions.

User Interface: Contains Swing GUI classes for user interface components.

model: Represents data entities corresponding to database tables.

Data Access Objects: database interactions using JDBC.

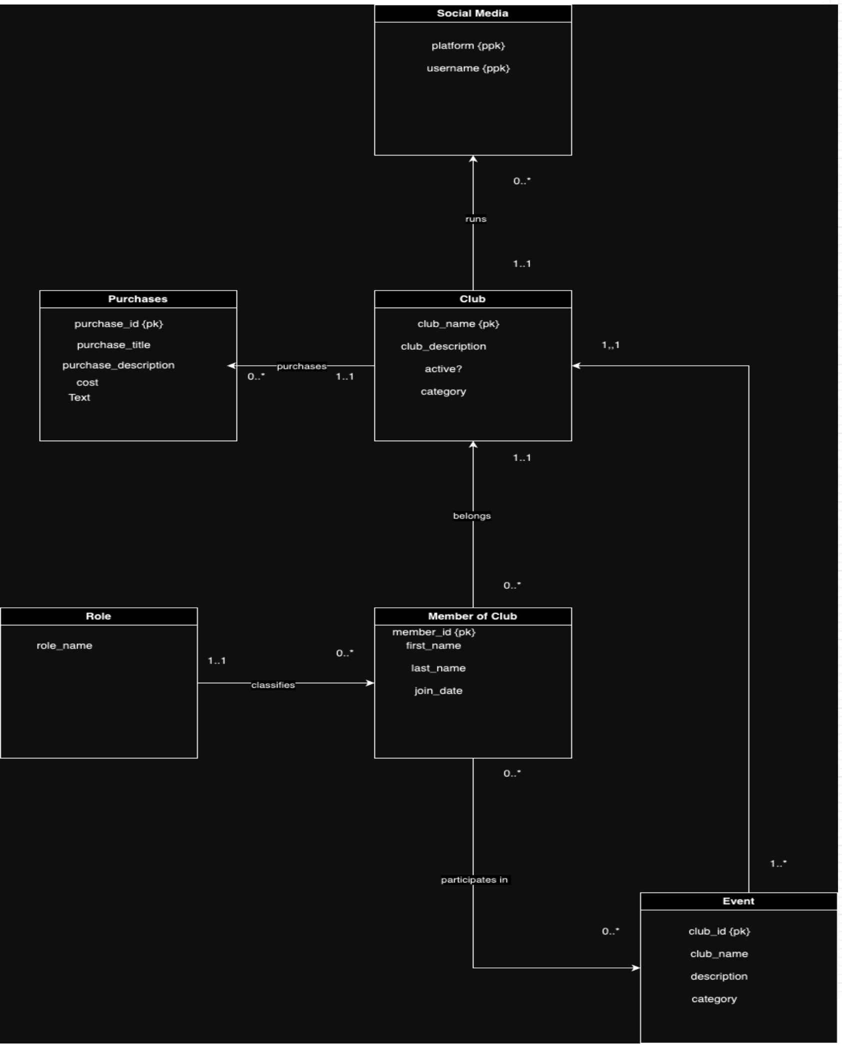
Framework: Java Swing

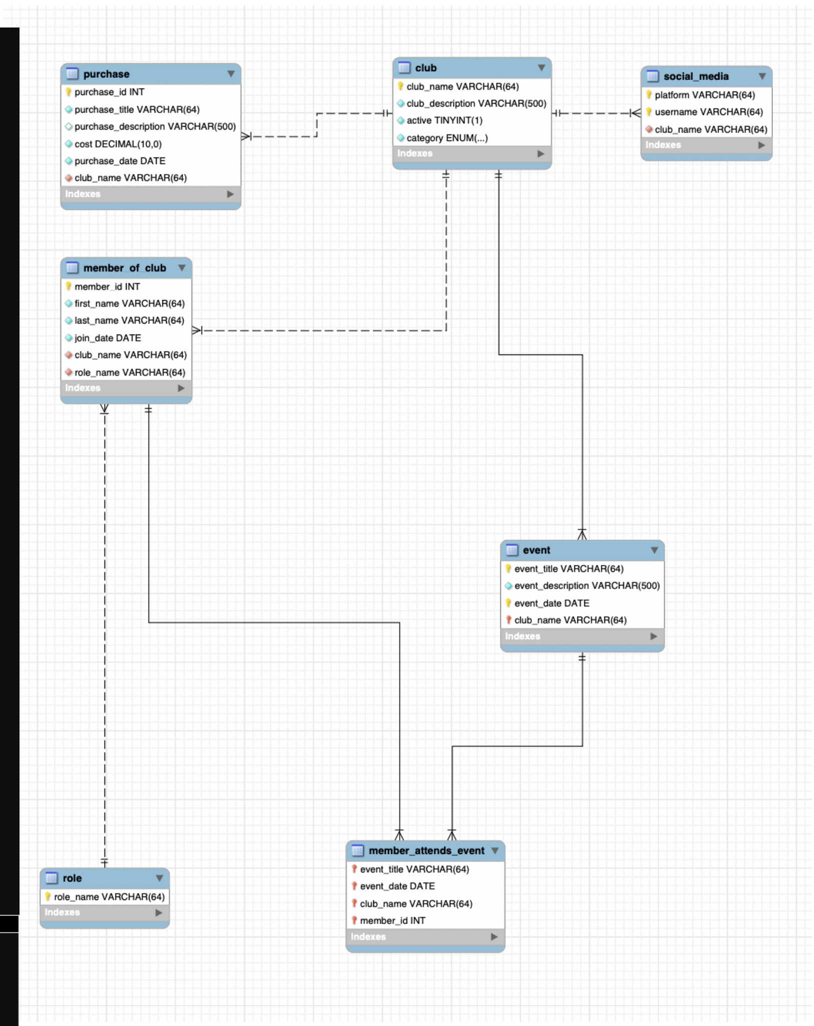
UI Components:

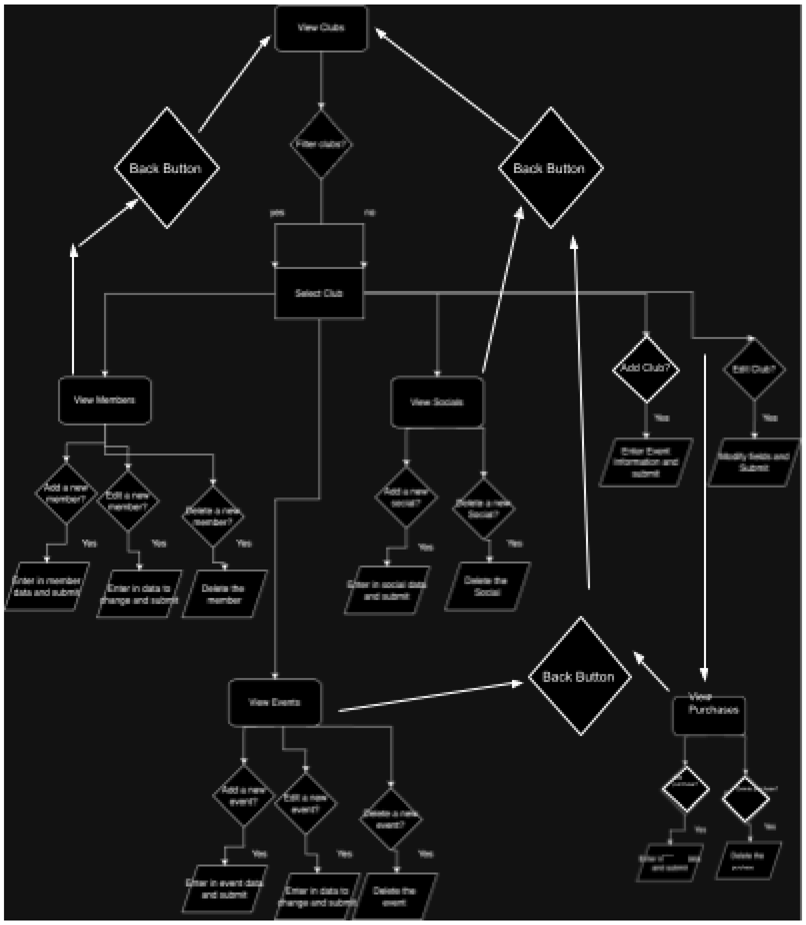
- **JFrame**: Main application window.
- **JPanel**: Used for organizing UI elements into logical sections.
- **JLabel, JButton, JTextField, JTable**: Core UI elements for input/output and interactivity.
- **Layouts**: BorderLayout, GridLayout, and BoxLayout for flexible designs. Custom styling with UIManager for consistent look and feel.
- **Icons**: Embedded resources for visual elements like buttons (e.g., delete, add).

UML Project Description:

Our project plans to provide Northeastern clubs a centralized place to communicate resources and events to students. Once you are in the app, you can then interact with the clubs on the site and make edits and add to it and its many components as you like. A club can make 0 to many purchases using its money. A purchase has an id to identify it, title, description, cost in decimals and a purchase date. A club has a name, description, a category (which is one of "STEM", "Arts and Culture", "Sports and Recreation", "Community Service and Social Impact", "Social and Special Interests", "Media and Communications", "Religious and Spiritual"), and knows whether or not it is active. A club also maintains zero to many social media accounts which are identified by its platform and username handle. There are also members for these clubs and a member can belong to only one club and a club can have many members. A member has an id number, first and last name, and a join date. A member can also have 0 to many roles in their specific club. A role can be anything specified by the user. Clubs also keep their events. An event can only belong to one and only one club. It has a title, a description, and a date. The database also keeps track of which members attend events so that clubs can measure the commitment of their members. A member can attend many events and an event can be attended by many members.


Social Media
platform {ppk}
username {ppk}
0..*
runs
1..1
Purchases
purchase_id {pk}
purchase_title
purchase_description
cost
Text
purchases
0..*
1..1
Club
club_name {pk}
club_description
active?
category
1,,1
1..1
belongs
0..*
Role
role_name
1..1
classifies
0..*
Member of Club
member_id {pk}
first_name
last_name
join_date
0..*
participates in
1..*
0..*
Event
club_id {pk}
club_name
description
category


purchase
purchase_id INT
purchase_title VARCHAR(64)
purchase_description VARCHAR(500)
cost DECIMAL(10,0)
purchase_date DATE
club_name VARCHAR(64)
Indexes
club
club_name VARCHAR(64)
club_description VARCHAR(500)
active TINYINT(1)
category ENUM(...)
Indexes
social_media
platform VARCHAR(64)
username VARCHAR(64)
club_name VARCHAR(64)
Indexes
member_of_club
member_id INT
first_name VARCHAR(64)
last_name VARCHAR(64)
join_date DATE
club_name VARCHAR(64)
role_name VARCHAR(64)
Indexes
event
event_title VARCHAR(64)
event_description VARCHAR(500)
event_date DATE
club_name VARCHAR(64)
Indexes
role
role_name VARCHAR(64)
Indexes
member_attends_event
event_title VARCHAR(64)
event_date DATE
club_name VARCHAR(64)
member_id INT
Indexes



User Actions

- Filters
  - A user can filter between non active and active clubs
- A user can select a club
  - A user can edit that club
  - A user can add a new club
  - A user can view the socials of that club
    - A user can delete a social
    - A user can create a new social
  - A user can View the events of that club
    - A user can add a new event
    - A user can update a new event
    - A user can delete a new event
  - A user can view the members of that club
    - A user can add a new member
    - A user can update a new member
    - A user can delete a new member

Bugs

We only found two small errors, one where, when you delete a Social, it creates a new pop up with the deleted view, and you have to exit out of that and the view with the original socials on it.

We also found that when editing clubs, it will always by default change the category to STEM unless you specify otherwise.

Lessons Learned

We gained a much better understanding of Swing and its components listed in the technical description. We also learned how to handle errors and read them. In using try - catch blocks, we began to understand how to write more helpful error messages to help with quicker debugging. We also made our design modular as we went on, often delegating long and winding and breaking them up into helper methods. Or we would create interfaces for easy access to our many SQL CallableStatement methods to organize them in one place and also know exactly what we have done and where to find it. Overall, our design and knowledge of java swing have greatly improved over the course of this assignment.

I think it would've been a much better design decision to create a robust MVC design had this project been more scalable and larger in scope. Even with our current design and scope of segregating observers with its listeners, our design became muddy and difficult to traverse at times. Creating a concrete, modular, and straightforward MVC design would have helped a lot with clarity and focus in our code. The view would contain all of our panels and classes, our controller would handle user input and emit user actions that the view would eventually receive and act upon (or just generally refresh). Then our model would control all the connections to our database and create our object models for different entities in our database. This way, we have more clarity by segregating different functions and responsibilities to different directories which could've potentially aided in the project's earlier phases!

Future Work

We plan on expanding the database to contain sections that allow different users to have different access to certain information. First we would add passwords and usernames to be stored in the database for you to sign up with or log in. Then we would add a user Entity that has information surrounding its profile and it is the superclass of a member user and club organizer user.

For functionality, we would then provide different views. The relevant important differences are that as a member you wouldn't be able to create, update, or delete existing events, socials, purchases, or existing members. Members could also know what clubs they belong to. They would only be able to read that data in a view only display of the data in our database. On the club organizer member's end, they would be able to have all the missing functionality from a normal member.