# Art Club Management System Project Report

## Submitted by:

Mounota SIKDER(2024200000662)

Farhana Akter(2024200000651)

Al Nahian Nafi(2024200000680)

Nafisa Hossain Lamisa(2024200000672)

Md.Torikul Islam(2024200000644)

Hasna Akter Mim(2024200000665)

## Submitted to:

Mr.Mahabubul Haq Bhuiyan(MBH)

# Introduction

An Art Club Management System is a digital tool designed to help manage an art club. It helps club members and organizers keep doing of important activities such as adding members, organizing events, storing artwork records, and managing finances. By using this system users store everything in one place and make it easy to access and update information when needed.

## Purpose of the Art Club Management System:

The main purpose of this system is to help organize and manage an art club. It ensures that all club activities run effortlessly by keeping proper records and making club management easier for the users.

## 1. Member Managment:

This feature keep a list of all members with their name, age, and art work.This feature also add new members easily and store their information for future and remove members who leave the club.Additionally this feature view all members in a  list.

## 2.Event Management:

This feature keep a record of upcoming art events like workshops, exhibitions, and competitions and store details such as event name, date, and description and view a list of all planned events.Additionally remove past or canceled events from the system.

## 3.Artwork Management:

This feature record artworks created by members and store details like artwork name, type, and the artist's name.Additionally view a collection of all submitted artworks.

## 4.Finance Management:

This feature record the income of the club such as membership fees, donations, or event earnings.It also

record expenses such as event costs, materials, or venue fees and calculate the total balance to understand the financial condtion of the art club.

# Features:

## 1. Member Management

•Add Member

•View Members

•Remove Member

## 2. Event Management

•Add Event

•View Events

•Remove Event

## 3. Artwork Management

•Add Artwork

•View Artworks

## 4. Finance Managemen

•Manage Finance (add income, add expense & view balance)

# Code Explanation:

## #Nafisha's Part

## 1. Add Member

At first I take an empty list for store information about members.Each member will be stored as a list.

```
def add_member():
    name = input("Enter member's name: ")
    age = input("Enter age: ")
    art_form = input("Enter art form: ")
    members.append([name, age, art_form])
    print(f"{name} added successfully!")
```

The 1st line defines the function add_member, which will add a member in the club.

The 2nd line of the code asks the user to input the name of the member they want to add.

The 3rd line of the code asks the user to input the age of the member.

The 4th line of the code asks the user to input the art form.

The 5th line add the member in the members list by using the .append() method

The 6th line displays a confirmation message that the member has been added using print() statement.

## #Mim's Part

### 2. View Member

```
def view_members():
    if len(members) == 0:
        print("No members in the club.")
    else:
        print("\nArt Club Members:")
        print("--------------------------------------------------")
        print("No. | Name        | Age | Art Form")
        print("-------------------------------------------")
        for i in range(len(members)):
            print(f"{i+1}.  | {members[i][0]:12} | {members[i][1]:3} | {members[i][2]}")
        print("-------------------------------------------")
```

Explanation of the view_members() Function:

The function first checks if the members list is empty using if len(members) == 0:

If it is empty, it prints "No members in the club."

If the members list is not empty, the function prints a header "Art Club Members:" followed by a line of dashes to separate the header from the data.

It then prints the column headers "No. | Name | Age | Art Form" along with another line of dashes to mark the start of the data.

The function uses a for loop (for i in range(len(members)):) to iterate over each member in the list.

For each member, it prints:

The member's number (i.e., the index + 1).

The member's name, formatted to 12 characters wide.

The member's age, formatted to 3 characters wide.The member's art form.

After printing all members, the function prints a final line of dashes to signal the end of the table.

## #Farhana's Part

## 3. Remove Member

```
def remove_member():
    name = input("Enter the name of the       member to remove: ")
    for i in range(len(members)):
        if members[i][0] == name:
            members.pop(i)
            print(f"Member '{name}' removed successfully!")
            return
    print("Member not found!")
```

The 1st line defines the function remove_member, which will remove a member from the club.

The 2nd line of the code asks the user to input the name of the member they want to remove.

Then in the 3rd line, for loop checks every member in the members list.

In the 4th line, if condition checkd if the index of member's name matches the name user entered.

The 5th line removes the member from the members list by using the pop() method. i is the index value of the member user wanted to remove.

The 6th line displays a confirmation message that the member has been removed using print() statement.

And then the break statement ends the function if the member is found and removed.

If the member's name is not in the list, it will display not found using print statement which is stated in the last line of the code.

## #Mounota's Part

## 4. Add Event

At first I take an empty list for store information about events.Each event will be stored as a list.

```
def add_event():
    name = input("Enter event name: ")
    date = input("Enter event date (YYYY-MM-DD):")
    description = input("Enter event description: ")
    events.append([name, date, description])
    print("Event {0} added
successfully!".format(name))
```

The 1st line defines the function add_event, which will add a event in the club.

The 2nd line of the code asks the user to input the name of the event they want to add.

The 3rd line of the code asks the user to input the date of the event including year,month and date

The 4th line of the code asks the user to input the description about the event.

The 5th line add the event in the enents list by using the .append() method.

The 6th line displays a confirmation message that the event has been added using print() statement.

## 5. View Event

```
def view_events():
    if len(events) == 0:
        print("No upcoming events.")
    else:
        print("\nUpcoming Events:")
        print("-----------------------------------------------")
        print("No.  Event Name    Date     Description")
        print("-----------------------------------------------")
        for i in range(len(events)):
            print("{0:3}  {1:10}  {2:24}  {3:4}".format(i + 1,
events[i][0], events[i][1], events[i][2]))
        print("---------------------------------------------)
```

The 1st line defines the function view_events, which will view the events as a event list.

The function first checks if the events list is empty using if len(events) == 0:

If it is empty, it prints "No upcoming events." by using print() statement.

If the events list is not empty, the function prints a header "Upcoming Events:" and print  a line of dashes to separate the header from the data by using print statement.

It then prints the column headers "No. | Event Name | Date | Drscription" along with another line of dashes to mark the start of the data.

The function uses a for loop (for i in range(len(events)):) to print over each event in the list.

Then I use spacing and .format method for organizing the event list.

After printing all events, the function prints a final line of dashes to signal the end of the table.

## 6. Remove Event

```
def remove_event():
name = input("Enter the name of the event to remove: ")
   for i in range(len(events)):
      if events[i][0] == name:
            events.pop(i)
            print(f"Event '{name}' removed
successfully!")
    else
      print("Event not found!")
```

The 1st line defines the function remove_event, which will remove a event from the event list.

The 2nd line of the code asks the user to input the name of the event they want to remove.

Then in the 3rd line, for loop checks every evert in the events list.

In the 4th line, if condition checked if the index of event's name matches the name user entered.

The 5th line removes the event from the events list by using the .pop() method. i is the index value of the event user wanted to remove.

The 6th line displays a confirmation message that event has been removed using print() statement.

The 7th line means that if the name of the event is not found in the list, the if condition does not work and for this we use the else condition.

If the event's name is not in the list, it will display not found using print statement which is stated in the last line of the code.

## #Nafi's part

### 7. Add Artwork

```
def add_artwork():
    artwork_name = input("Enter artwork name: ")
    artwork_type = input("Enter artwork type: ")
    member_name = input("Enter the member's name who created the artwork: ")
    artworks.append([artwork_name, artwork_type, member_name])
    print(f"Artwork '{artwork_name}' added successfully!")
```

### 8. View Artwork

```
def view_artworks():
   if len(artworks) == 0:
      print("No artworks available.")
   else:
      print("\nArt Club Artworks:")
      print("----------------------------------------------")
      print("No. | Artwork Name  | Type      | Created by")
      print("----------------------------------------------")
      for i in range(len(artworks)):
          print(f"{i+1}.  | {artworks[i][0]:12} | {artworks[i][1]:10} |
{artworks[i][2]}")
      print("----------------------------------------------")
```

The view_artworks function is designed to display a list of artworks from the artworks list in a neatly formatted table.If the list is empty, it informs the user that no artworks are available. Otherwise, it prints a header, column titles, and each artwork's details in a well-organized manner.

And in the code, firstly I take An empty list for store information about artworks. Each artwork will be stored as a list of three items: the artwork name, its type, and the name of the creator (member).

Then i take a function add_artwork(), it will allow the user to input details about a new artwork and add it to the artworks list.

Then i take input from user artwork name, type and creator name. This line adds a new list containing the three details (name, type, and creator) into the artworks list.

The append() method is used to add the new list to the end of artworks.

After the artwork has been added to the list, this line prints a confirmation message that includes the name of the artwork that was just added.

For viewing,I take a new function view_artworks() that is responsible for displaying all the artworks in the artworks list.

Then I use if conditions,it checks if the artworks list is empty. len(artworks) returns the number of elements in the list, and if it equals 0, it means no artworks have been added yet.

If the list is empty (no artworks have been added), this message will be displayed.

If the artworks list is not empty, this prints a header saying "Art Club Artworks" to indicate the list of artworks is about to be shown.

Then printing a table header with labels for each column: "No." (for the index), "Artwork Name", "Type", and "Created by" (the member's name).

The "_" lines help separate the header from the content below.

Then using a for loop that will go through each element in the artworks list.

Here, len(artworks) returns the number of artworks, so range(len(artworks)) will loop from 0 to len(artworks) - 1.

At the last line ,If the list artworks is not empty, it prints the header "Art Club Artworks" followed by a decorative line and column headers ("No.", "Artwork Name", "Type", "Created by").

These headers are separated by lines to make it look neat and organized.

This line uses a for loop to iterate over each artwork in the artworks list. enumerate(artworks, 1) is used to get both the index (i) starting from 1 and the elements of the artworks list (name, art_type, creator).

Then printing each artwork's details in a formatted way, with name, art_type, and creator aligned properly using formatting options (:12, :10).

And the last print statement prints another decorative line at the end to mark the conclusion of the table of artworks.

## #Farhana's Part

## 9.Manage Finance

At first I take an empty list for store information about finances.Each financial statement  will be stored as a list.

```
def manage_finance():
    print("1. Add Income")
    print("2. Add Expense")
    print("3. View Balance")
    choice = input("Enter your choice (1-3): ")
    if choice == '1':
        amount = float(input("Enter income amount (in Tk): "))
        financials.append(["Income", amount])
        print(f"Income of {amount:.2f} Tk added successfully!")
    elif choice == '2':
        amount = float(input("Enter expense amount (in Tk): "))
        financials.append(["Expense", amount])
        print(f"Expense of {amount:.2f} Tk added
successfully!")
    elif choice == '3':
        total_income = sum([entry[1] for entry in
financials if entry[0] == "Income"])
        total_expense = sum([entry[1] for entry in
financials if entry[0] == "Expense"])
        balance = total_income - total_expense
        print(f"Current balance: {balance:.2f} Tk")
    else:
        print("Invalid choice!")
```

At first of this part of the program, defining a function called manage_finance where the user can add income, expenses, or check the current balance.

The print statements display options for the user to choose from adding income, adding expense or viewing the balance.

And then the input() function asks the user to enter their choice and stores it in the choice variable.

After that the if condition checks if the user chose option 1 (add income).

The next line of the code asks the user to enter the income amount and stores it against variable amount after converting it to a float.

And then adding the income to the financials list with a label "Income" and the corresponding amount using (.append) method.

And the print statement prints a confirmation message with the added income which is formatted to two decimal places by (.2f).

After if condition, using elif choice == "2": condition which checks if the user chose option 2 (add expense).

And the same way using amount variable to store the input from the user for the expense amount and converts it to a float.

Then again using (.append) method to add the expense to the financials list with a label "Expense" and the amount.

After that printing a confirmation message for the expense using print statement.

And the using elif choice == '3': condition so that if the user chose to view the balance, this block executes.

total_income = sum([entry[1] for entry in financials if entry[0] == "Income"]):

This line calculates the total income by filtering the financials list for "Income" and summing the amounts.

total_expense = sum([entry[1] for entry in financials if entry[0] == "Expense"]):

Similarly, it calculates the total expenses.

And then in the next line, the balance is calculated by subtracting total expenses from total income and stored in balance variable.

Then using print() to print the balance.

And lastly using else, so if the user entered an choice except 1,2,3 then this prints a message saying "Invalid choice!"

## #Torikul's Part

## Main Functionality

```
def main():
    while True:
        print("\n==== Art Club Management System ====")
        print("1. Add Member")
        print("2. View Members")
        print("3. Remove Member")
        print("4. Add Event")
        print("5. View Events")
        print("6. Remove Event")
        print("7. Add Artwork")
        print("8. View Artworks")
        print("9. Manage Finance")
        print("10. Exit")
        choice = input("Enter your choice (1-10): ")
        if choice == '1':
            add_member()
        elif choice == '2':
            view_members()
        elif choice == '3':
            remove_member()
        elif choice == '4':
            add_event()
        elif choice == '5':
```

```
            view_events()
        elif choice == '6':
            remove_event()
        elif choice == '7':
            add_artwork()
        elif choice == '8':
            view_artworks()
        elif choice == '9':
            manage_finance()
        elif choice == '10':
            print("Exiting the program. Goodbye!")
            break
        else:
            print("Invalid choice! Please enter a number
between 1 and 10.")
main()
```

The 1st line defines the function main.It is the main part of my program, where everything starts running.

The 2nd line of the code while true: starts an infinite loop.

The program will keep running until the user chooses to exit.

The 3rd line of the code indicates the title of my code and I use print statement for print title.

Then these 10 lines print the menu options.

The numbers (1 to 10) help users choose what they want to do.

This asks the user to enter a number between 1 and 10.

The input() function takes user input as a string.

The result is stored in a variable called choice.

Then the program checks what the user entered and runs the correct function.

If the user enters "1", the program calls the add_member() function.

This function should add a new member to the system.

If the user enters "2", it calls view_members().

This should display the list of members.

If the user enters "3", it calls remove_member().

This function should delete a member from the system.

If the user enters "4", the program calls the add_event() function.

This function should add a new event to the system.

If the user enters "5", it calls view_events().

This should display the list of events.

If the user enters "6", it calls remove_event().

This function should delete a event from the event list.

If the user enters "7", the program calls the add_artwork() function.

This function should add a artwork to the system.

If the user enters "8", it calls view_artwork().

This should display the list of members.

If the user enters "9", it calls manage_finance().

This function should handle the club’s money records.

If the user enters "10", the program prints a goodbye message.

Then, it uses break to stop the while True loop, ending the program.

If the user enters something other than 1-10.Than the esle condition works and

the program t asks the user again for input.

The last line calls the main() function, so the program starts running.

## Challenges that We Faced:

While working on the Art Club Management System, we faced several challenges as first semester students but overcoming .

One of the first problems we faced was understanding how to structure a program with multiple functions. At first, our code was not correct because we were not sure how to separate different works into functions. We solved this by discussing the project as a team and dividing the work into clear parts like Member Management,Event Management, Artwork Management and Finance Management.

Another big challenge was handling user input properly. Sometimes, users entered invalid data, like string type data instead of numbers for age or financial amounts. This caused our program run incorrectly. We solved this by adding type casting like int().

We also struggled with removing members and events from lists. At first, our code didn't work correctly because we did not use the correct loop index for removing an item.Then we read the articles from W3 School about index number.

Another difficulty was using spacing and .format method.We practiced more and more and somehow overcame the issue.

Additionally, we faced challenges in keeping track of financial transactions. At first, we only stored the current transaction, but later, we realized we needed to keep a full record of all incomes and expenses. We fixed this by using a list to store all transactions and calculating the balance.

Moreover, we had difficulty testing and debugging our code. Sometimes, the program didn’t work and we didn’t know where the mistake was. We learned to use print statements to check  our code and find errors step by step.

Finally,we had a problem to run the code when the six members of our were running together.So we did not run the code together.

Through these challenges, we learned important programming skills like problem-solving, debugging, and writing cleaner code. This project was a great learning experience that helped us understand how to work as a team and build a art club management system by using python.

## Conclusion:

Working on the Art Club Management System as beginners in Python taught us many important programming skills. We learned how to use functions to organize our code, making it more  easier to read.We work by using 2D list that helped us store and manage data like members, events, and financial records. We also improved our understanding of loops which allowed us to display and modify data.We handle user input correctly that was another important lesson. We had to ensure the program did not work when incorrect data was entered. Through error handling , we learned how to prevent solve errors and make the program more useful. Formatting output properly using string formatting  helped us create well-structured tables for displaying information. Debugging errors and solving problems step by step made us better at logical thinking. Additionally, this project helped us understand the importance of teamwork as we had to work different parts of the code while maintaining consistency. Overall, this project gave us hands-on experience and grow our confidence in writing Python programs. Now, we feel more prepared to solve any code.Art Club Management System

Project Report