Bangladesh University of Professionals

Faculty of Science and Technology
Department of Information & Communication Technology (ICT)

Course Title: Database Management System Laboratory
Course Code: ICE-2204

Project Report

Project Name: "ArtBlossoms" – An Art Commission Marketplace

Submitted to,

Sharmeen Jahan Seema Lecturer, Dept. of ICT Faculty of Science & Technology Bangladesh University of Professionals	Md. Sazzadul Islam Prottasha Lecturer, Dept. of ICT Faculty of Science & Technology Bangladesh University of Professionals

Submitted by,

ID	**Name**
23549009021	Ratrixmna Chakma
23549009041	Laiba Sumaiya Nazim
23549009093	Masuma Tasnim Nimo

Section A , Batch 2023

Submission Date : 8th August, 2025

Table of contents

Phase-01: Introduction

ArtBlossoms: Personalized ArtBlossoms is an online platform designed to connect artists and buyers through custom and ready-made artwork services. The purpose of this project is to simplify the process of art discovery, ordering, and commissioning by offering a centralized and interactive space for digital artists and their audiences. ArtBlossoms improves traditional art sale models by integrating user roles, artwork listings, commission management, and secure transactions. The system enhances accessibility, creativity, and transparency while ensuring that both artists and buyers have efficient tools for collaboration, showcasing, and feedback. The platform supports better profile management, artwork filtering, reviewing, and administrative control, making the digital art community more connected and streamlined.

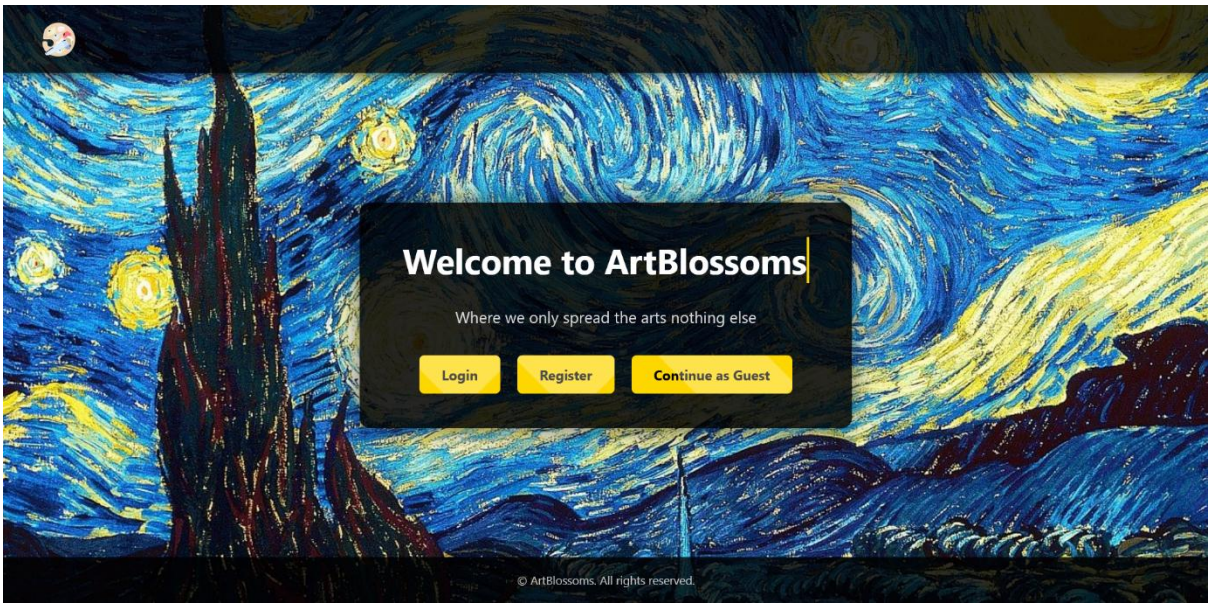
Summary

The *ArtBlossoms* is a web-based platform built to bring together artists and buyers in a creative and personalized environment. It allows users to register either as artists, buyers, or both, offering separate yet connected features tailored to their roles. Buyers can explore a diverse gallery of artworks, filter by categories, search by tags, and click on artworks to see detailed descriptions and images. Logged-in users can place orders for existing works or request custom commissions by specifying their own ideas, budget, and deadline. Artists, on the other hand, can upload and manage their artworks, include categories and tags for better visibility, control availability status, and view or respond to commission requests. Users can also update their profile pictures and leave reviews to help others. Admins have a dedicated panel to handle reports made on artworks and maintain the integrity of the platform. Guests can freely browse the gallery, but registration unlocks full functionality such as placing orders. The system is powered by a MySQL database and built with Python using the Flask web framework, ensuring smooth interactions, data handling, and a dynamic user experience across roles.

Functions & Operations

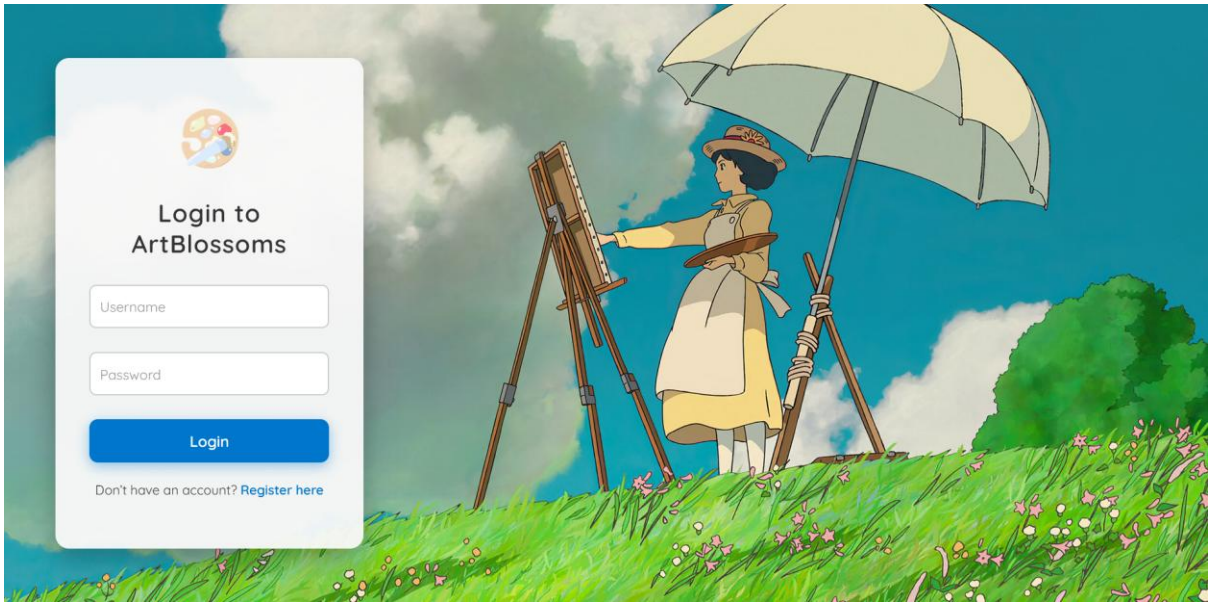
Below are the primary functions along with their descriptions that will be included in the project.

1. **Home :** The home page of *ArtBlossoms* serves as the welcoming interface for all users—registered or guests. It displays a visually appealing layout with a featured banner, quick navigation to core pages like "Login", "Register", and "Browse Artworks", and promotional artwork previews. The homepage also includes project branding, a brief mission statement, and links to learn more about the platform.



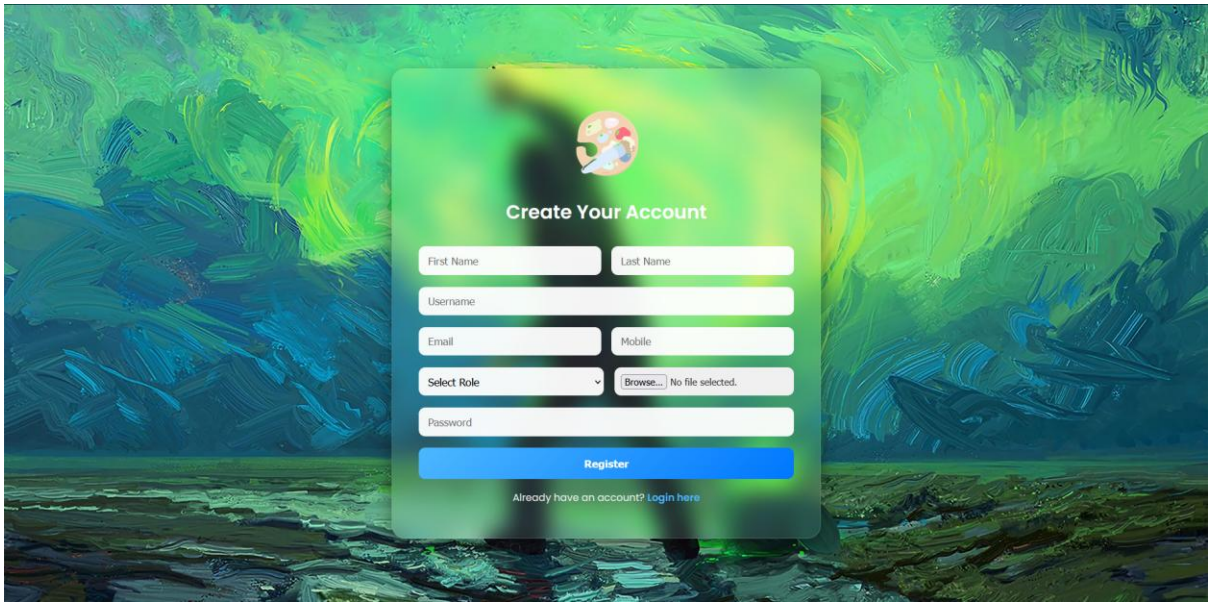
1: Home page

2. **Login :** The **Login Page** of *ArtBlossoms* allows registered users to securely access their accounts. Users are required to enter their username and password, which are verified against stored hashed values in the database. On successful login, users are redirected to their profile or dashboard based on the role they selected during registration. This page ensures secure and role-specific access to user functionalities.



***2**: Login page*

3. **Registration:** New users can register on *ArtBlossoms* by selecting their intended role: **Artist**, **Buyer**, or **Both**. During registration, users provide essential information including name, username, email, mobile number, password, and optional profile picture. Each role unlocks a different set of features tailored to their needs, and these can later be upgraded or changed.

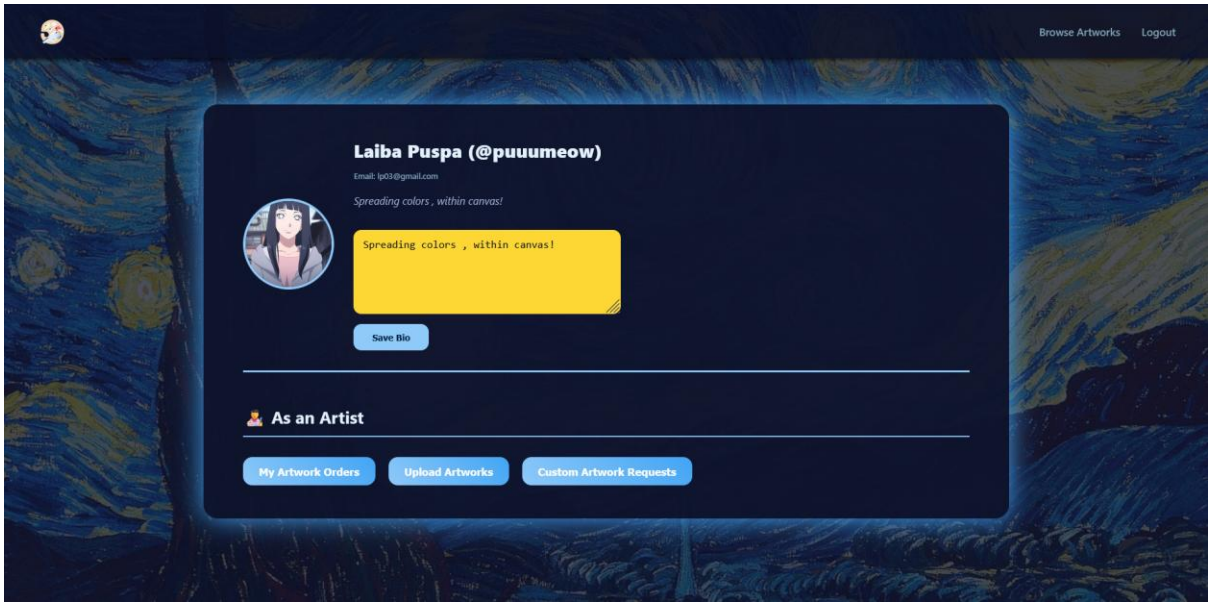


3: Register page

3. **User Profile :** The *ArtBlossoms* platform includes a role-based user profile system that supports three distinct roles: Artist, Buyer, and Both. Upon login, users are directed to personalized profile pages that dynamically adapt based on their role. Each profile provides access to relevant features—such as artwork uploads for artists, order and cart management for buyers, or a combination of both for users with dual roles—ensuring a tailored and intuitive user experience.

a. Artist Profile :

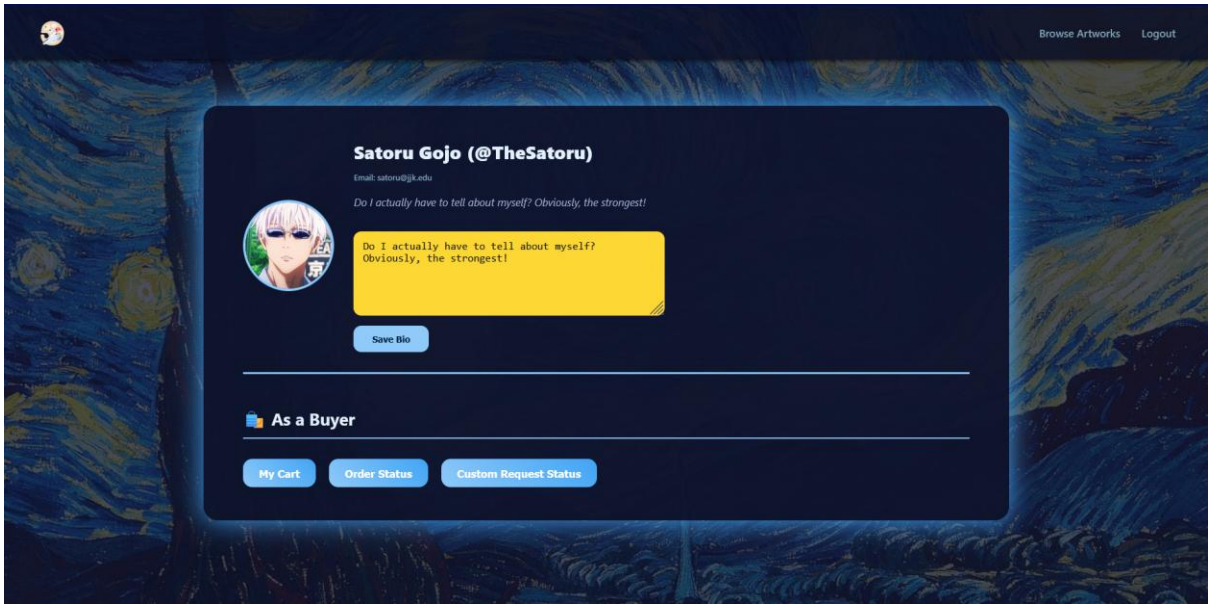
✓ View and manage their uploaded artworks.
✓ Upload new artwork with title, description, image, category, and tags.
✓ View incoming orders and respond to them.
✓ See reviews on their artworks.
✓ Update their bio and profile picture.
✓ Access commission requests.



4:Artsist Profile

b. <u>Buyer Profile:</u> Buyers can,

- ✓ View order history and current order statuses.
- ✓ Browse purchased artworks and reorder if desired.
- ✓ Submit reviews and ratings for completed orders.
- ✓ Manage and edit profile information such as bio and profile picture.
- ✓ Access saved cart items and continue with checkout.
- ✓ Track commission request status and responses.
- ✓ Cancel pending orders or commissions if needed.
- ✓ View recommended artworks based on past activity.

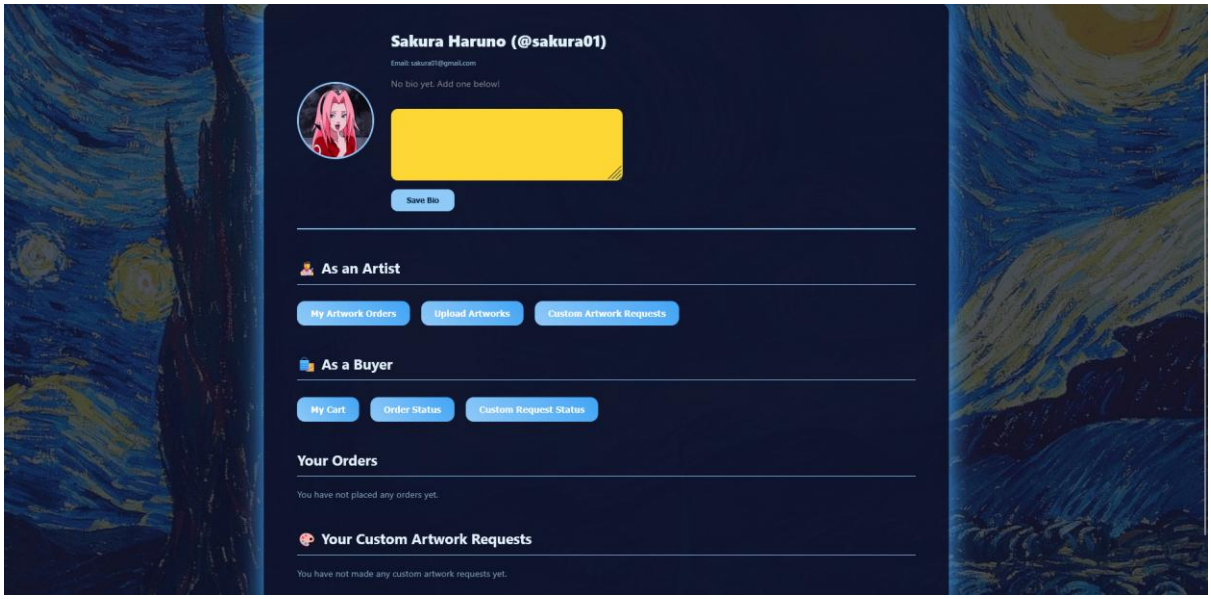


5: Buyer Profile

c. **<u>Both user profile (Buyer & Artist) :</u>**

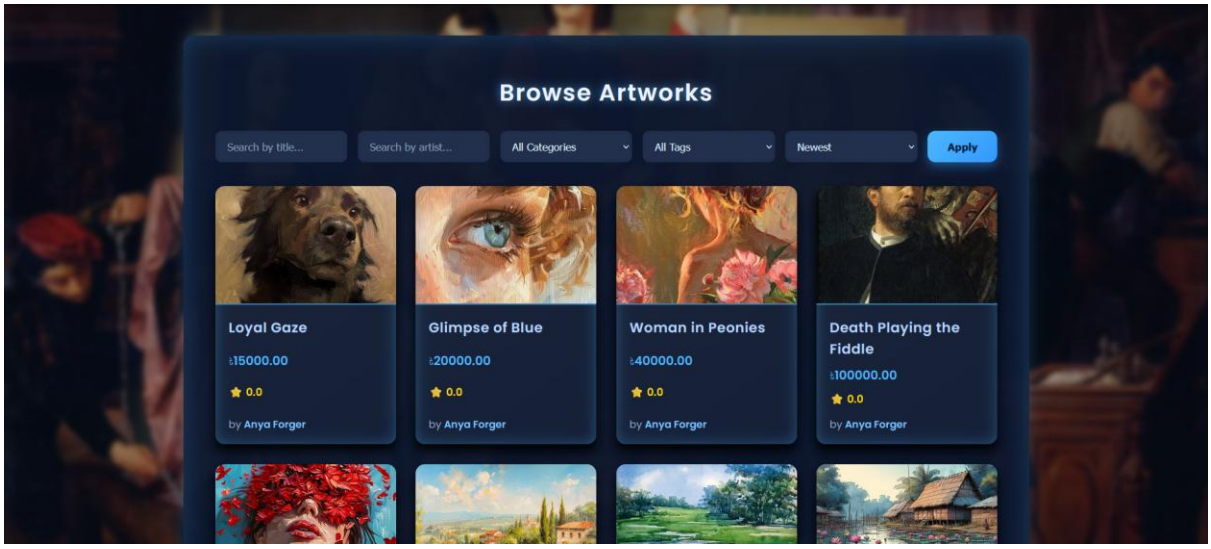
- ✓ Switch seamlessly between buyer and artist views in the dashboard.

- ✓ Upload and manage artworks as an artist.
- ✓ Place orders and request custom commissions as a buyer.
- ✓ View combined order history (received as artist, placed as buyer).
- ✓ Respond to commission requests and view sent commissions.
- ✓ Update personal profile, including bio and profile picture.
- ✓ See reviews given and received.
- ✓ Manage cart, uploads, and artwork availability from one unified page.

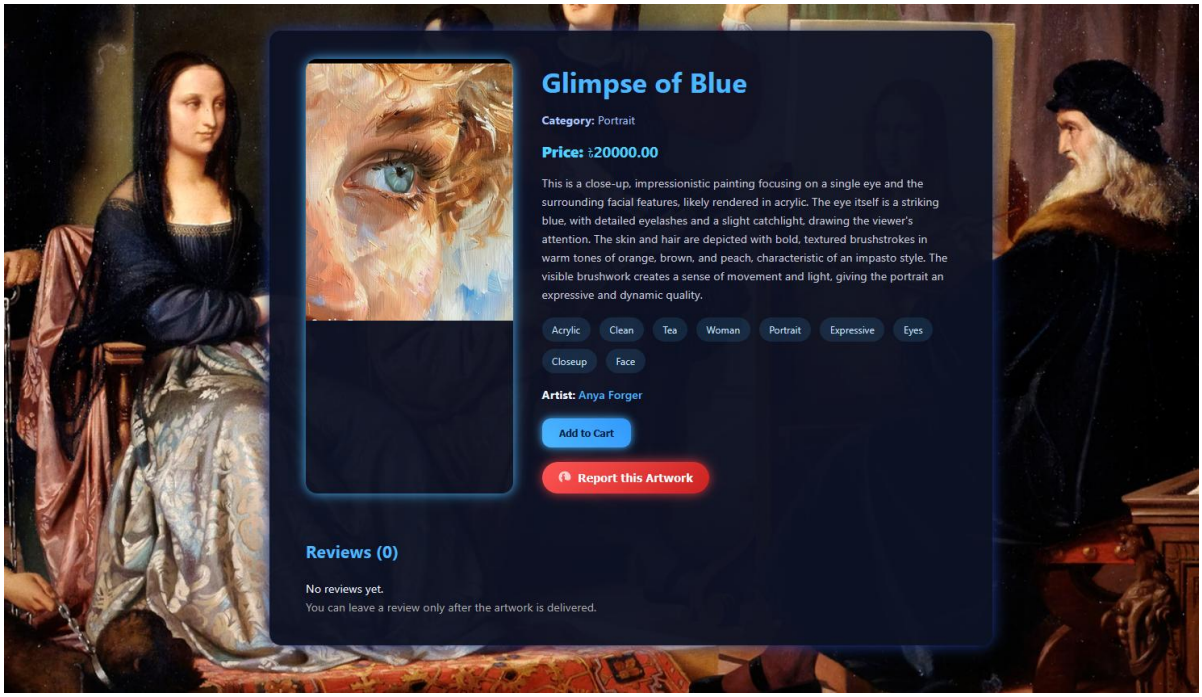


6: Both user profile

4. **Guest user:** Guests are users can browse the *ArtBlossoms* platform without logging in or registering. They have limited access but are still able to browse the gallery of artworks. They can view and explore artworks but can not add to cart or order without registering or logging in.
5. **Browse Artworks :** User can have their individual profiles and these profile has certains “role”. These are -
   a. **Browsing Artworks as registered users :** Registered users like buyers, artists, or both can browse artworks similarly to guests but have extra facilities. Logged-in users can add items to the cart, place direct orders, request commissions, and view personalized recommendations. Their interactions with artworks are stored and reflected in their dashboards.

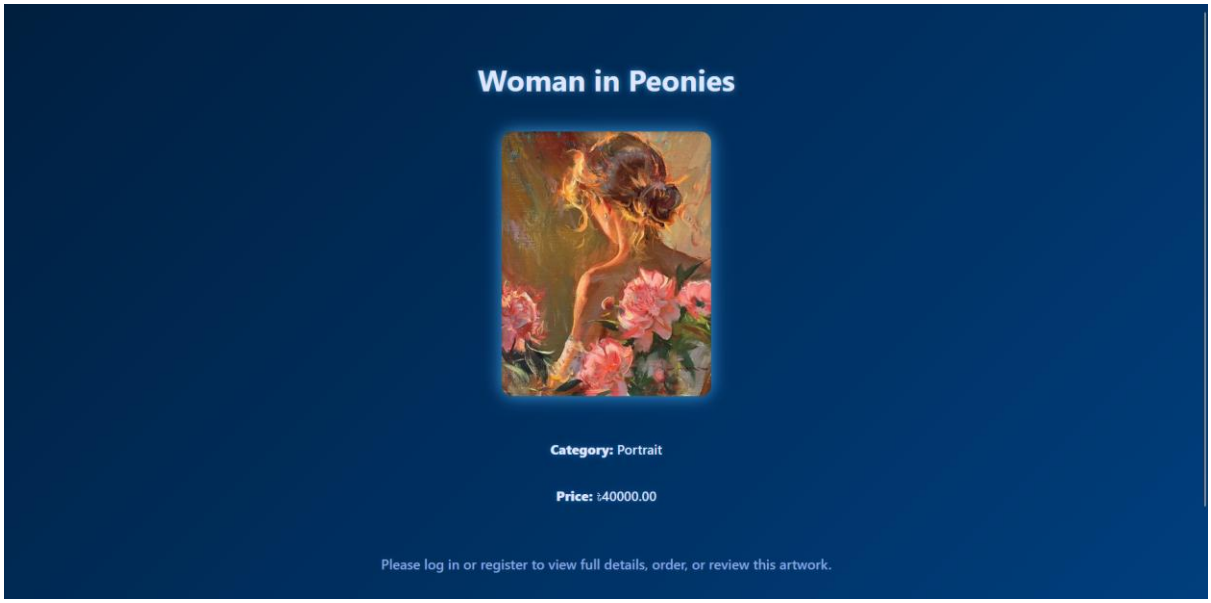


7: Browsing artworks at registered users



8: Artwork detail

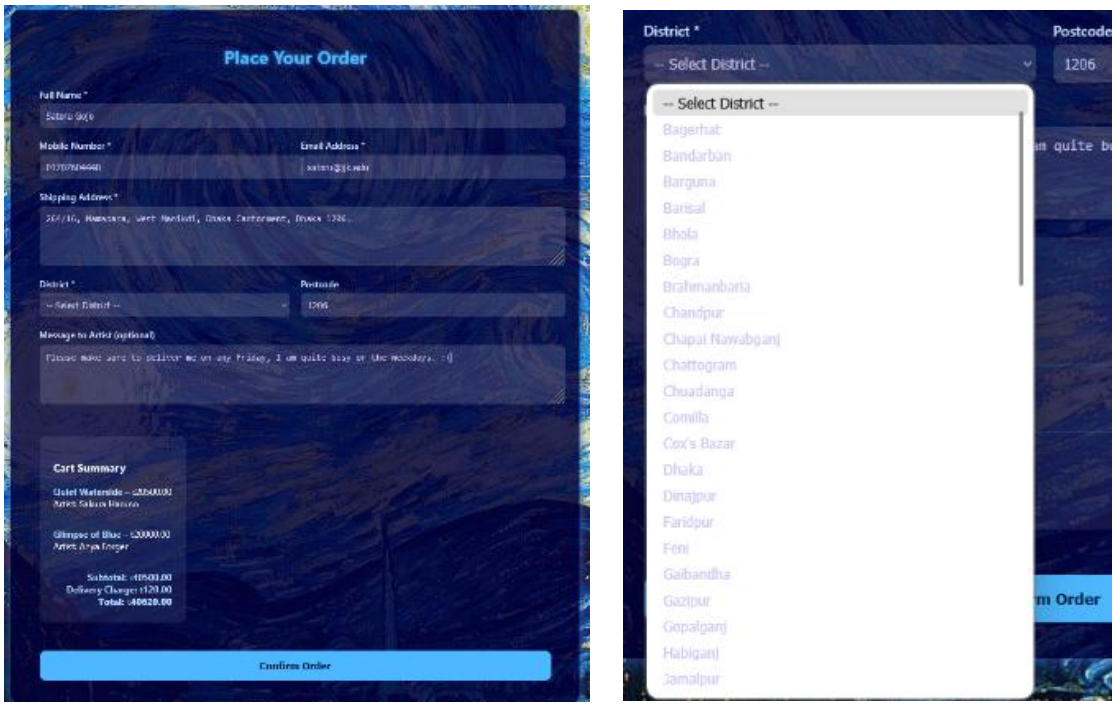
b. Browsing Artworks as guest : Guests can explore the artwork gallery by scrolling through all available listings. They can filter artworks by category or search by keywords and tags. Clicking on an artwork card allows them to view full details of the selected piece, including the title, artist name, price, and image. However, guests are restricted from adding items to the cart, placing orders, or writing reviews. They must register or log in to access those features.



6. **Artworks details :** The Artwork Details page provides a complete view of a selected artwork. This includes the image, title, description, price, category, tags, artist name that is linked to artist profile and existing reviews. Logged-in buyers will also see “Add to Cart” and “Order Now” options.

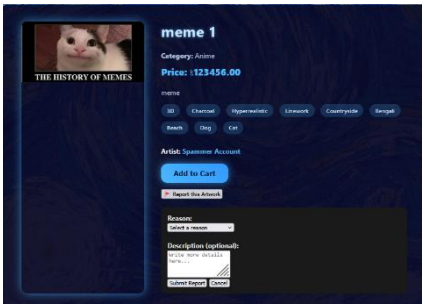
7. **Add to cart :** Registered buyers can add available artworks to their personal cart. This feature allows users to save items temporarily before placing a final order. Artworks can be added or removed from the cart at any time before checkout.

9: Cart list



10 : Order placing form and district selection

8. **Report Artwork :** Registered users can report artworks they find inappropriate, stolen, or misleading. A "Report" button appears on every artwork detail page. Users must select a reason and submit a short explanation. Reports are stored in the admin_reports table and can be reviewed later by an admin.

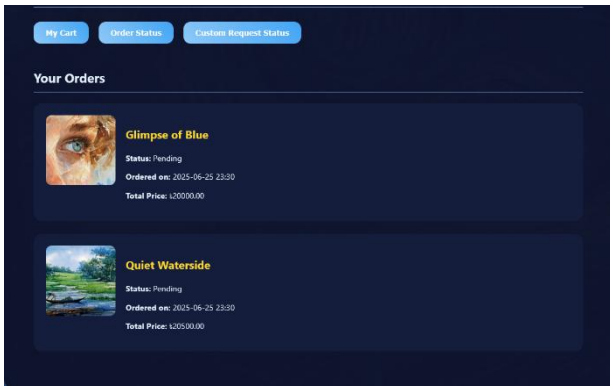


11: Spamming account reporting

9. **Order Status :** The order system in *ArtBlossoms* enables buyers to place individual or multiple artwork orders directly from their cart. Each order stores detailed buyer information and is linked to a specific artist through the ordered artwork. Once an order is placed, the artist receives a notification and can update the order status through their profile. The status flow includes stages such as Pending, Accepted, Processing, Sent to Delivery, Rejected, and Sold out, allowing buyers to track their order progress in real-time until completion.
   a. **Artist getting orders:** When a buyer places an order, the associated artist receives a notification in their dashboard. Artists can view order details such as buyer name,

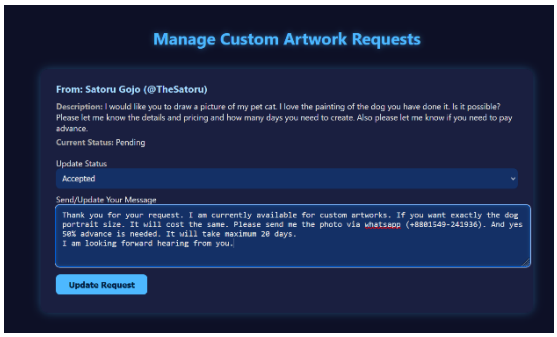
delivery information, and message. They can then process the order and update its status like accepted, sent to delivery or rejected from their profile dashboard.

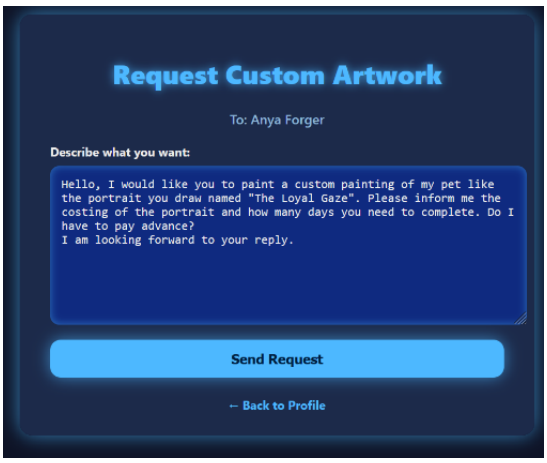
b. **Buyers checking order status:** Buyers can visit their order history page to view all their current and past orders. Each order displays its status like pending, processing, sent to delivery etc., total price, artwork details, and delivery information. This section also allows buyers to track whether their order has been accepted or shipped by the artist.



12: Cart list

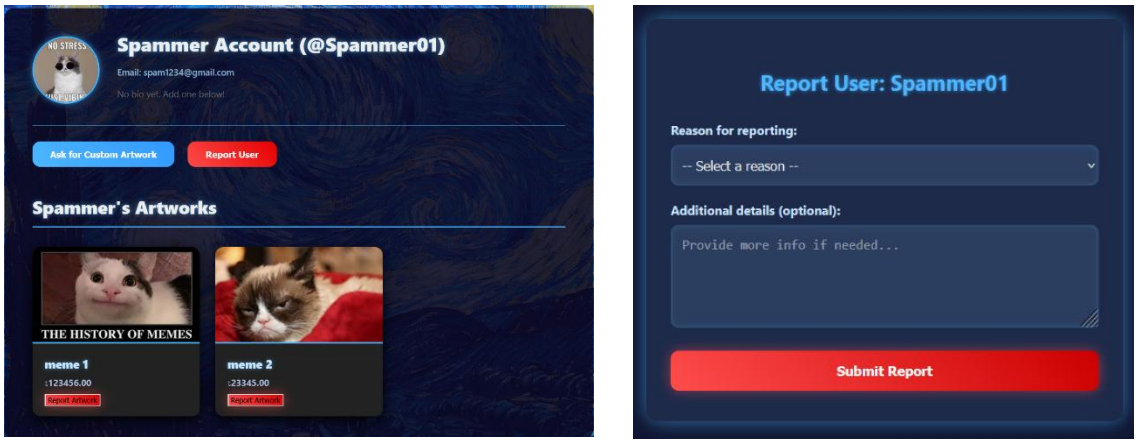
10. **Custom Commission request:** Registered users (especially buyers or both-role users) can submit a custom commission request to an artist by filling out a form with specific instructions, budget range, deadline, and reference images if applicable. Artists receive the request in their dashboard and can choose to accept, reject, or modify the proposal. Once accepted, the commission is added to the buyer's order list with a new status label.





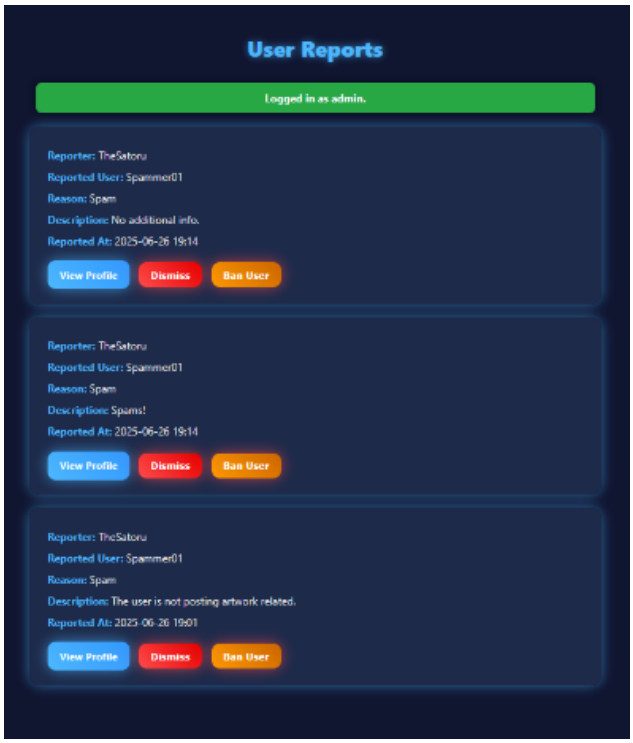
13: Artist getting custom orders and buyer rending custom order request

11. **Reporting user account:** Both artists and buyers can report inappropriate content, suspicious users, or system issues. Reports are submitted through a form that includes the report type, a description, and optional screenshots. Admins receive these reports in a dedicated admin panel where they can view, review, and update the status of each report (e.g., resolved, pending, dismissed). This feature ensures platform safety and ethical conduct.

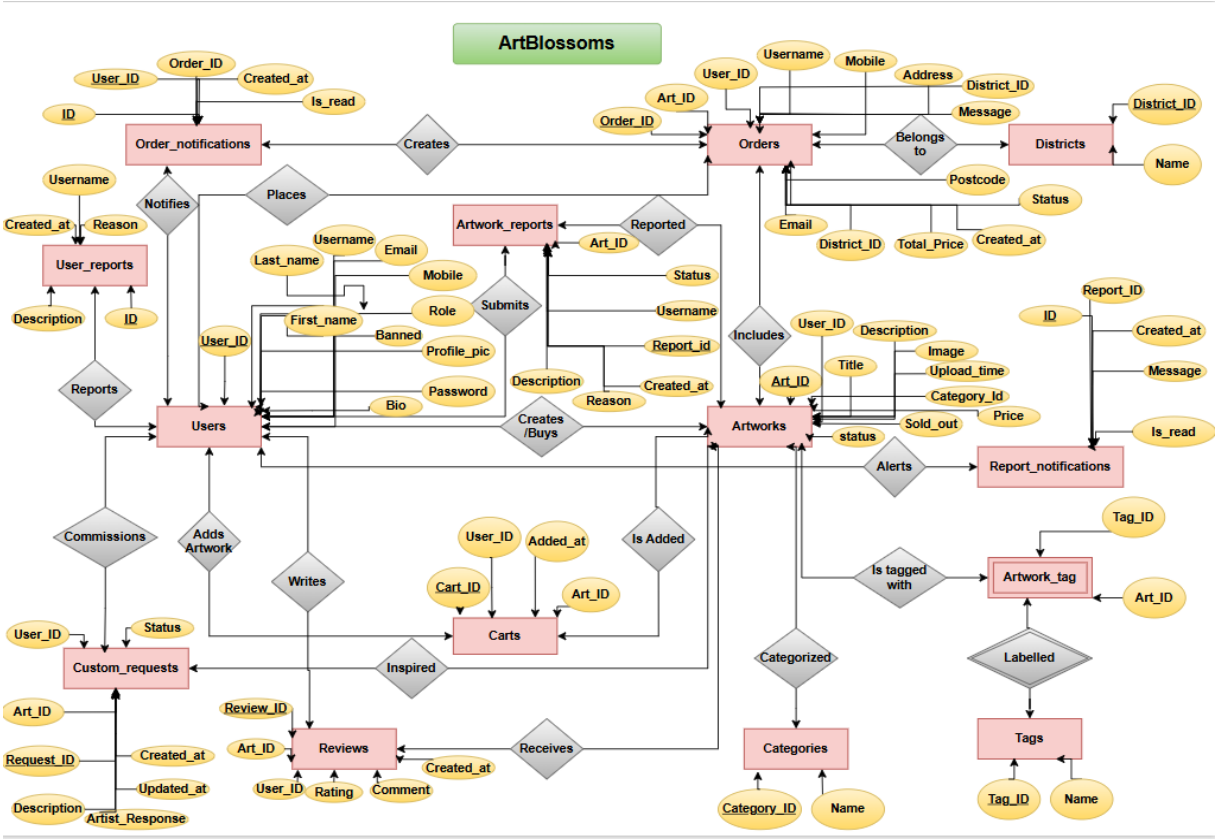


14: Reporting Spamming user profile

12. **Admin panel :**The Admin Panel in ArtBlossoms serves as the control center for managing users and platform content. Admins have the authority to review reported users and artworks, and take appropriate actions such as banning user accounts or deleting artworks that violate platform policies. The panel provides access to flagged content submitted by users through the reporting system, ensuring that moderation is fair and transparent. This feature helps maintain a safe and professional environment for both artists and buyers, making the platform more reliable and secure as it grows.



15: Admin interface of getting reports1



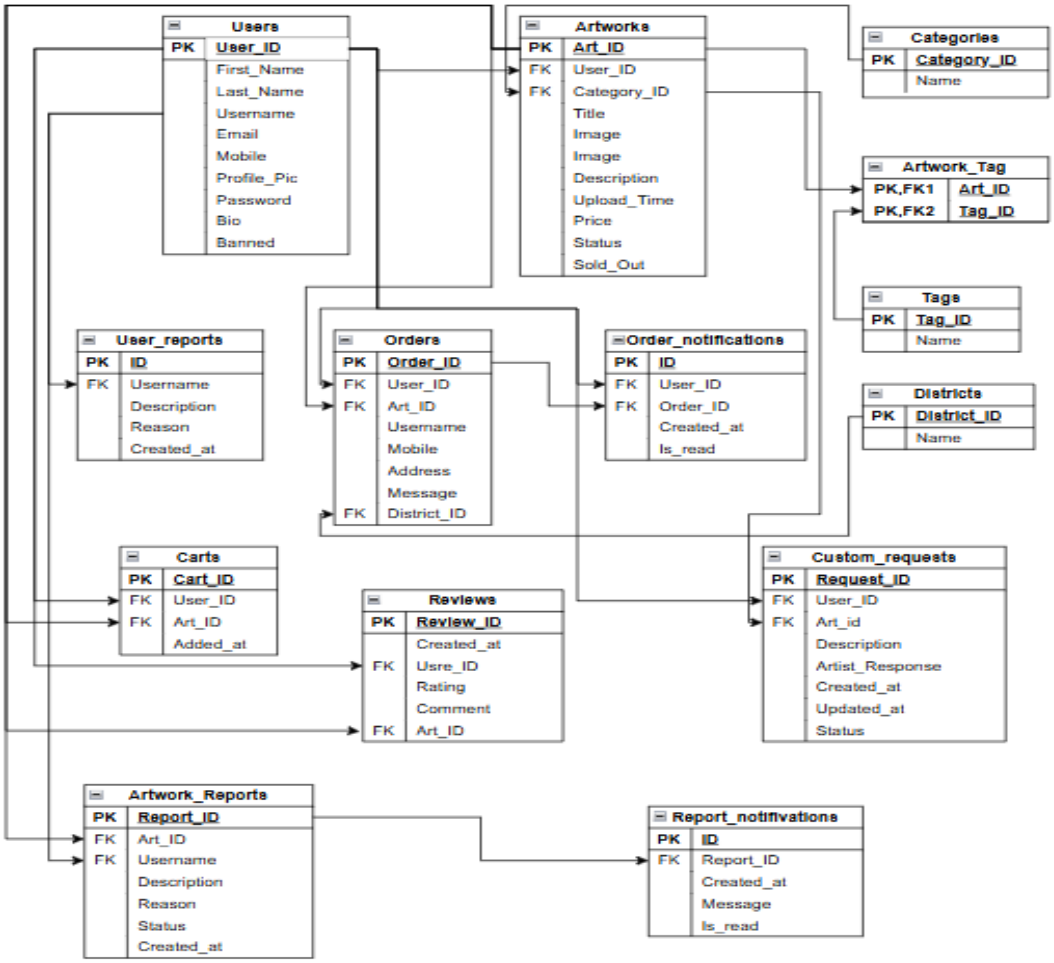
Entity Relation Diagram:

Narrative Description of the ER diagram:

1. **User :**Represents all users of the system, including buyers, artists, and admins. Each user has a unique ID, username, role, email, password (hashed), profile picture, and optional bio.
2. **Artworks :**Stores information about each artwork posted by artists. Includes title, description, price, category, tags, upload date, availability status, artist ID (foreign key), and image path.
3. **Carts :**Temporarily holds artworks that a buyer has selected before placing an order. Each cart is linked to a user and includes timestamps and cart item details.
4. **Categories :**Defines the different types or genres of artworks such as Portrait, Abstract, Digital, etc. Helps in filtering and organizing artworks.
5. **Artwork _tags :**Establishes a many-to-many relationship between artworks and tags for better filtering. Each entry links an artwork to a specific tag.
6. **Tags :**A collection of descriptive keywords like "Nature", "Fantasy", "Minimalist", etc., used to enhance search and discoverability of artworks.
7. **Order_notifications:**Stores messages sent to artists when an order or commission is placed. Includes buyer ID, artwork ID, timestamp, and message content.
8. **Orders :**Contains records of both regular and custom artwork orders. Tracks order status, total amount, buyer ID, artist ID, and delivery details.

9. **Districts :**Maintains the list of user districts (for delivery/localization), enhancing regional customization and logistics planning.

Schema Diagram :



Progress Report : The development of the ArtBlossoms platform followed a structured and well-organized progression. Initially, the project began with deciding on the topic, where we selected to build a web-based art marketplace that connects artists and buyers through a creative digital platform.

After finalizing the theme, we moved on to discussing the functionalities of the system. This included identifying the key roles (Artist, Buyer, Admin), outlining user interactions, and listing the core modules such as artwork uploads, cart management, order placement, review system, and admin moderation features.

Following the functionality breakdown, we started working on the database structure by adding tables and defining attributes for each entity such as users, artworks, orders, reviews, and notifications. We ensured that all necessary fields were included based on functional needs.

Next, we established the key constraints and participation constraints, which included setting primary keys, foreign key relationships, and defining optional/mandatory links between entities to maintain data integrity and proper referential relationships across the system.

Once the schema was clear, we proceeded to draw the ER diagram, representing all entities, relationships, and constraints visually. The ER model helped us validate the overall structure and logic before implementation.

Finally, we wrote a detailed description of the ER diagram, explaining each entity, its attributes, relationships, and cardinalities. This step ensured conceptual clarity and served as a blueprint during the actual database implementation and backend development.

Summary:

1. Deciding the topic of this project .
2. Discussing about the functionalities.
3. Tables and the attributes adding.
4. Key constraints and participation constraints adding.
5. Drawing ER diagram.
6. Writing the description of the ER diagram.
7. Learning related programming languages (Python and HTML) and related connection making.
8. Installing necessary software (Microsoft Visual Studio Code, Python version 13.0.2).
9. Learning Relational Schema for next phase.

Total Time Period and Progress Report of the Team Members

Group Progress Report :

Reporting period : 01.07.2025

Name and ID	**Total Hours**	**Remarks**
Ratrixmna Chakma (23549009021)	13.05.2025 08:00 PM – 10:30 PM (02 hours 30 minutes)	Introduction of the Project, Database Relations Discussion, Attribute and Key Constraints. Finalizing ER Diagram Design, Review and Edits, Document Formatting
Laiba Sumaiya Nazim (23549009041)	14.05.2025 07:00 PM – 10:30 PM (02 hours 30 minutes)	Reviewing ER Diagram Concepts, Attribute Analysis, Initial Report Drafting Group Discussion Notes and Overall Outcome of All Group Meetings Submission preparation of ER Diagram design, Document Formatting
Masuma Tasnim Nimo (23549009093)	13.05.2025 08:00 PM – 10:30 PM (02 hours 30 minutes)	Project Planning, Initial Description of ER Diagram, , Database schema overview. Writing Descriptions for ER Diagram, Writing Functional Descriptions, Group Meeting Notes Group Discussion Notes and Decision Making ER Diagram Creation, Detailing Relationships, and Keys, Group Discussion
Group total Time:	7 hours 30 minutes	

Individual Report :

Team member Name: Ratrixmna Chakma

Reporting Period : 01.07.2025

Date	Time period	Activities
21.05.2025	2 hours 30 minutes	Learning complex query in MySQL online to implement the project.
23.05.2025	3 hours	Browsing extra features which can be added in the project online.
01.06.2025	2 hours 30 minutes	Created a draft of tables with related attributes and constraints.
05.06.2025	1 hours 3 hours	Installing related software to implement the project. Watched YouTube videos regarding the connection between MySQL and Python.
06.06.2025	2 hours 30 minutes	Learn basic HTML language to create a sample login and registration page.
08.06.2025	4 hours	Connected MySQL and Python. Created a sample login and register page.
09.06.2025	2 hours	Created related tables on MySQL.
13.06.2025 14.06.2025	8 hours 7 hours	Group meeting for implementing project online.

Team Member Name : Laiba Sumaiya Nazim

Reporting Period: 01.07.2025

Date	Time Period	Activities
21.05.2025	2 hours 45 minutes	Learning complex query language online to implement the project.
24.05.2025	5 hours 27 minutes	Started to learn basic code language in Python.
01.05.2025	6 hours 45 minutes	Installing related software to implement the project. Watched YouTube videos regarding the connection between MySQL and Python.
03.06.2025	4 hours	Learn basic HTML language to create a sample login and registration page.
04.06.2025	4 hours 20 minutes	Connected MySQL and Python. Created a sample login and register page.
08.06.2025	5 hours	Learning about creating routes and connection between HTML and Python.
13.06.2025 14.06.2025	8 hours 7 hours	Group meeting for implementing project regarding interface (front end) and regarding the project inside workings.
15.06.2025	1 hour	Created a draft document on Words where we created the SRS.

Team Member Name : Masuma Tasnim Nimo

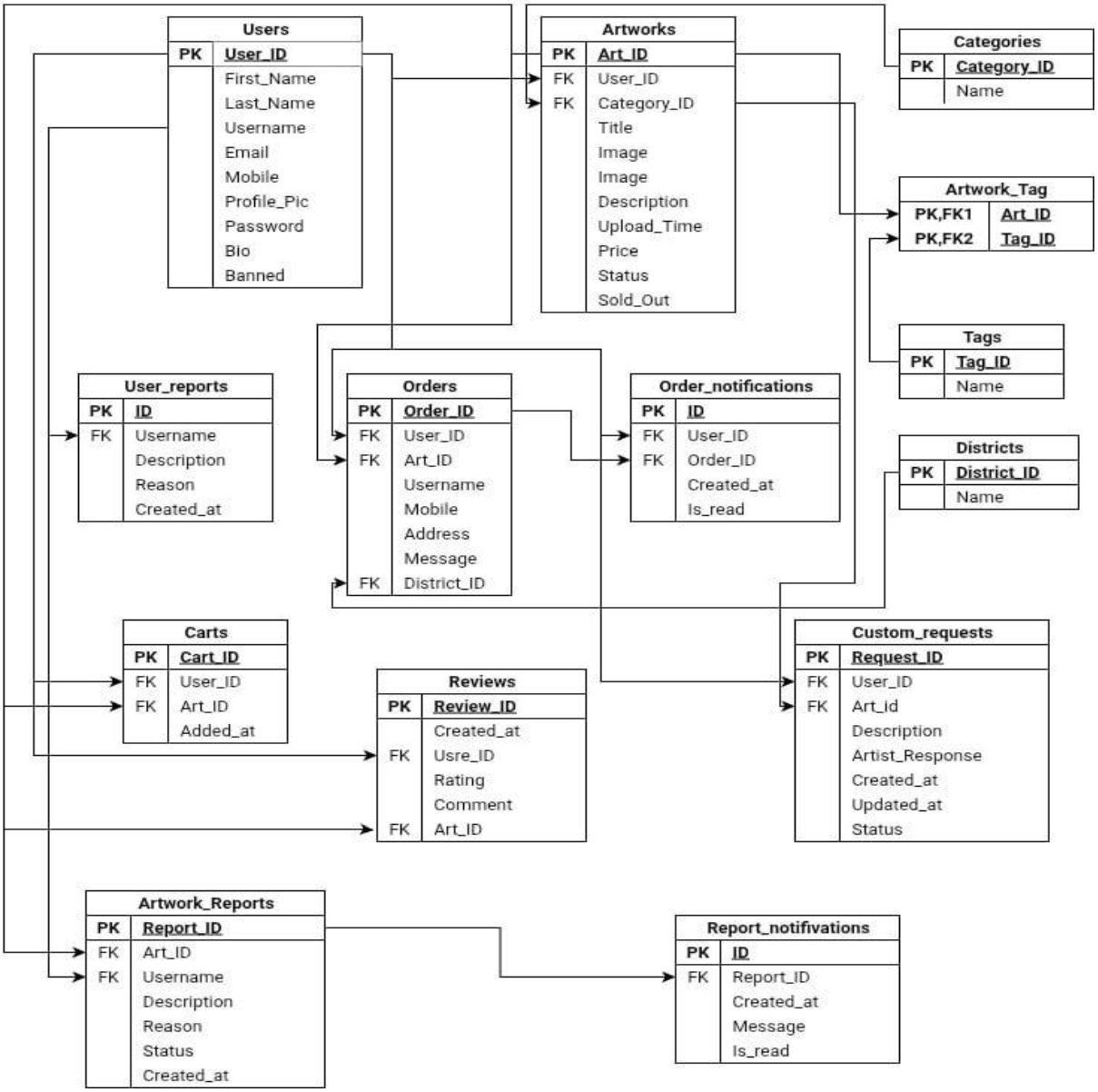
Reporting period : 01.07.2025

Date	**Time period**	**Activities**
21.05.2025	2 hours 30 minutes	Learning complex query in MySQL online to implement the project.
23.05.2025	5 hours 19 minutes	Started to learn basic code language in Python.
03.06.2025	1 hour 3 hours	Installing related software to implement the project. Watched YouTube videos regarding the connection between MySQL and Python.
09.06.2025	4 hours	Learn basic HTML language to create a sample login and registration page.
10.06.2025	4 hours	Connected MySQL and Python. Created a sample login and register page
12.06.2025	5 hours	Leaning more complex frontend using HTML online.
12.06.2025	2 hours	Learning about CSS online
13.06.2025 14.06.2025	8 hours 7 hours	Made a draft webpages for the project.
15.06.2025	5 hours	Created a draft ER Diagram on drawing.io

Phase 02

Building Schema Diagram:

The ER diagram submitted in Phase-1 has been modified and converted into relational schema which is given below:



Integrity Constraints:

User_ID(PK)	First_Name	Last_Name	Username	Email	Mobile	Profile_Pic	Password	Bio

Table 01 : User table

Art_ ID (PK)	User_ID (FK)	Category _ID (FK)	Title	Image	Description	Upload_Ti me	Price	Status	Sold_Out

Table 02 : Artworks table

Category_ID(PK)	Name

Table 03 : Categories

Art_ID(PK)(FK1)	Tag_ID(PK)(FK2)

Table 04 : Artwork_Tag

Tag_ID(PK)	Name

Table 05 : Tags

Cart_ID(PK)	User_ID(FK)	Art_ID(FK)	Added_at

Table 06 : Carts

Order_I D(PK)	User_ID(FK)	Art_ID(F K)	Usernam e	Mobile	Address	Message	District_ ID(FK)

Table 07 : Orders

ID(PK)	User_ID(FK)	Order_ID(FK)	Created_at	Is_read

Table 08 : Order_notifications

District_ID(PK)	Name

Table 09 : Districts

Review_ID(PK)	Created_at	User_ID(FK)	Rating	Comment	Art_ID(FK)

Table 10 : Reviews table

Request_ ID(PK)	User_ID(FK)	Art_ID(F K)	Descripti on	Artist_R esponse	Created_ at	Updated _at	Status

Table 11 : Custom_Requests table

ID(PK)	Username(FK)	Description	Reason	Created_at

Table 12 : User_Reports

Report_ID (PK)	Art_ID(F K)	Username(FK)	Descriptio n	Reason	Status	Created_at

Table 13 : Artwork_Reports

ID(PK)	Report_ID(FK)	Created_at	Message	is_read

Table 14 : Report_Notifications

Normalization:

1. First Normal Form(1NF) : The given relation is in 1st Normal Form because it meets the following criteria:
    a. All cells contain atomic (indivisible) values.
    b. Entries in each column are of the same type.
    c. Each tuple is unique; there are no duplicate rows.
    d. Column names are unique.

2. Second Normal Form(2NF) : The given relation is in 2nd Normal Form because it satisfies these conditions:
    a. The table is already in 1st Normal Form.
    b. There is no partial dependency; no proper subset of a candidate key can uniquely identify a non-prime attribute.
    c. All non-key attributes are fully dependent on the entire primary key.

3. Third Normal Form(3NF) : The given relation is in 3rd Normal Form because it meets the following requirements:
    a. The table is in 2nd Normal Form.
    b. There is no transitive dependency; no non-prime attribute can determine another non-prime attribute.
    c. For each functional dependency, either the left-hand side is a super key, or the right-hand side is a non-prime attribute.
    d. All columns are determined only by the primary key and no other column.

Normalization Check :

1. **Users Table:**

```
mysql> describe users;
+-------------+-------------------------------+------+-----+---------+----------------+
| Field       | Type                          | Null | Key | Default | Extra          |
+-------------+-------------------------------+------+-----+---------+----------------+
| id          | int                           | NO   | PRI | NULL    | auto_increment |
| first_name  | varchar(50)                   | YES  |     | NULL    |                |
| last_name   | varchar(50)                   | YES  |     | NULL    |                |
| username    | varchar(50)                   | YES  | UNI | NULL    |                |
| email       | varchar(100)                  | YES  | UNI | NULL    |                |
| mobile      | varchar(20)                   | YES  |     | NULL    |                |
| role        | enum('artist','buyer','both') | NO   |     | NULL    |                |
| profile_pic | varchar(255)                  | YES  |     | NULL    |                |
| password    | varchar(255)                  | YES  |     | NULL    |                |
| bio         | text                          | YES  |     | NULL    |                |
| banned      | tinyint(1)                    | YES  |     | 0       |                |
+-------------+-------------------------------+------+-----+---------+----------------+
11 rows in set (0.04 sec)
```

Normalization check:

Normal Form	Status	Description
First Normal Form	✅	All fields contain atomic values. No repeating groups.
Second Normal Form	✅	The primary key is User_ID. All non-key attributes (First_Name, Last_Name, Username, Email, etc.) are fully dependent on it.

Third Normal Form	✅	There are no transitive dependencies. Each non-key attribute is directly dependent on the primary key.

2. Artworks Table:

```
mysql> describe artworks;
+-------------+-------------------------------+------+-----+-------------------+-------------------+
| Field       | Type                          | Null | Key | Default           | Extra             |
+-------------+-------------------------------+------+-----+-------------------+-------------------+
| id          | int                           | NO   | PRI | NULL              | auto_increment    |
| user_id     | int                           | YES  | MUL | NULL              |                   |
| title       | varchar(100)                  | YES  |     | NULL              |                   |
| description | text                          | YES  |     | NULL              |                   |
| image       | varchar(255)                  | YES  |     | NULL              |                   |
| category_id | int                           | YES  | MUL | NULL              |                   |
| price       | decimal(10,2)                 | YES  |     | NULL              |                   |
| upload_time | timestamp                     | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
| status      | enum('available','sold out') | YES  |     | available         |                   |
| sold_out    | tinyint(1)                    | YES  |     | 0                 |                   |
+-------------+-------------------------------+------+-----+-------------------+-------------------+
10 rows in set (0.01 sec)
```

Normalization check :

Normal Form	Status	Description
First Normal Form	✅	All fields contain atomic and scalar values.
Second Normal Form	✅	The primary key is Art_ID All other attributes are fully dependent on it.
Third Normal Form	✅	No transitive dependencies exist. All non-key attributes are only dependent on the primary key.

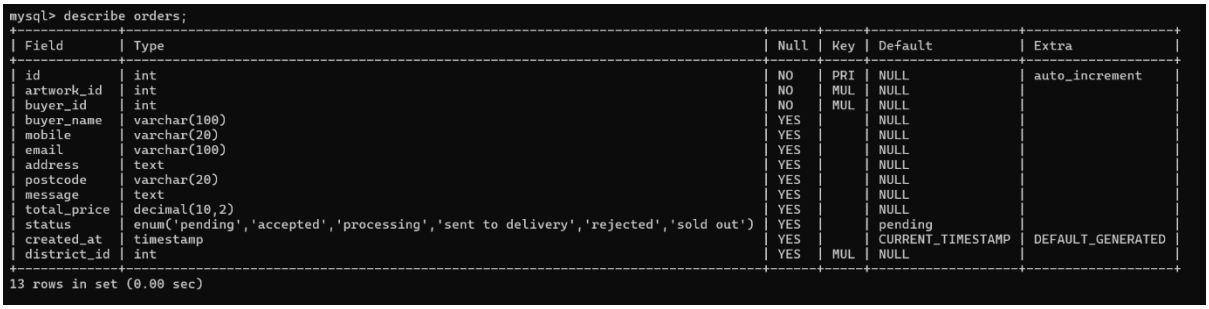
3. Order_notifications Table:

```
mysql> describe  order_notifications;
+------------+------------+------+-----+-------------------+-------------------+
| Field      | Type       | Null | Key | Default           | Extra             |
+------------+------------+------+-----+-------------------+-------------------+
| id         | int        | NO   | PRI | NULL              | auto_increment    |
| artist_id  | int        | YES  | MUL | NULL              |                   |
| order_id   | int        | YES  | MUL | NULL              |                   |
| is_read    | tinyint(1) | YES  |     | 0                 |                   |
| created_at | datetime   | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+------------+------------+------+-----+-------------------+-------------------+
5 rows in set (0.00 sec)
```

Normalization check :

Normal Form	Status	Description
First Normal Form	✅	All fields contain atomic values.
Second Normal Form	✅	All non-key attributes depend only on the primary key ID.
Third Normal Form	✅	No transitive dependencies are present.

4. Orders Table :

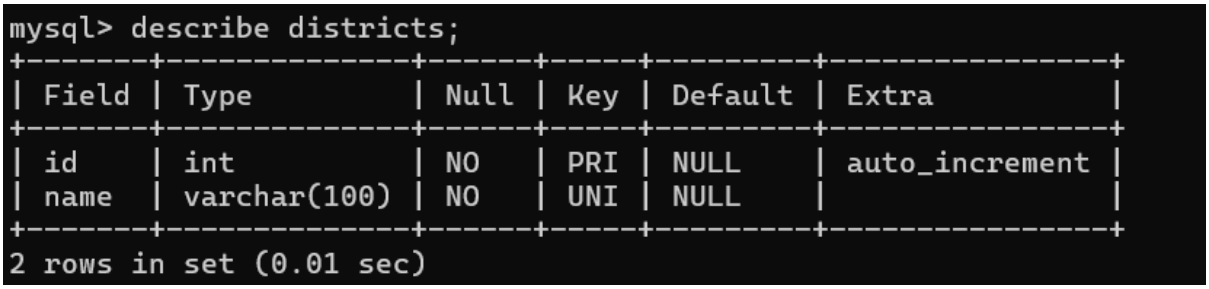


```
mysql> describe orders;
+-------------+-----------------------------------------------------------------------------+------+-----+-------------------+-------------------+
| Field       | Type                                                                        | Null | Key | Default           | Extra             |
+-------------+-----------------------------------------------------------------------------+------+-----+-------------------+-------------------+
| id          | int                                                                         | NO   | PRI | NULL              | auto_increment    |
| artwork_id  | int                                                                         | NO   | MUL | NULL              |                   |
| buyer_id    | int                                                                         | NO   | MUL | NULL              |                   |
| buyer_name  | varchar(100)                                                                | YES  |     | NULL              |                   |
| mobile      | varchar(20)                                                                 | YES  |     | NULL              |                   |
| email       | varchar(100)                                                                | YES  |     | NULL              |                   |
| address     | text                                                                        | YES  |     | NULL              |                   |
| postcode    | varchar(20)                                                                 | YES  |     | NULL              |                   |
| message     | text                                                                        | YES  |     | NULL              |                   |
| total_price | decimal(10,2)                                                               | YES  |     | NULL              |                   |
| status      | enum('pending','accepted','processing','sent to delivery','rejected','sold out') | YES  |     | pending           |                   |
| created_at  | timestamp                                                                   | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
| district_id | int                                                                         | YES  | MUL | NULL              |                   |
+-------------+-----------------------------------------------------------------------------+------+-----+-------------------+-------------------+
13 rows in set (0.00 sec)
```

Normalization check :

Normal Form	Status	Description
First Normal Form	✅	All columns are atomic. No multivalued attributes.
Second Normal Form	✅	All attributes are dependent on the primary key Order_ID.
Third Normal Form	✅	No transitive dependencies exist.

5. Districts Table :



```
mysql> describe districts;
+-------+--------------+------+-----+---------+----------------+
| Field | Type         | Null | Key | Default | Extra          |
+-------+--------------+------+-----+---------+----------------+
| id    | int          | NO   | PRI | NULL    | auto_increment |
| name  | varchar(100) | NO   | UNI | NULL    |                |
+-------+--------------+------+-----+---------+----------------+
2 rows in set (0.01 sec)
```

Normalization check :

Normal Form	Status	Description
First Normal Form	✅	Fields are atomic and consistent.
Second Normal Form	✅	There is only one attribute apart from the key; no partial dependency.
Third Normal Form	✅	No transitive dependency.

6. Artwork_Tag Table:

```
mysql> describe artwork_tags;
+------------+------+------+-----+---------+-------+
| Field      | Type | Null | Key | Default | Extra |
+------------+------+------+-----+---------+-------+
| artwork_id | int  | NO   | PRI | NULL    |       |
| tag_id     | int  | NO   | PRI | NULL    |       |
+------------+------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Normalization check :

Normal Form	Status	Description
First Normal Form	✅	Atomic values, no repeating groups.
Second Normal Form	✅	The composite key (Art_ID,Tag_ID) has no partial dependency.
Third Normal Form	✅	No transitive dependency among non-key attributes.

6. **Tags :**

```
mysql> describe tags;
+-------+-------------+------+-----+---------+----------------+
| Field | Type        | Null | Key | Default | Extra          |
+-------+-------------+------+-----+---------+----------------+
| id    | int         | NO   | PRI | NULL    | auto_increment |
| name  | varchar(50) | YES  | UNI | NULL    |                |
+-------+-------------+------+-----+---------+----------------+
2 rows in set (0.01 sec)
```

Normalization check :

Normal Form	Status	Description
First Normal Form	✅	Contains atomic values.
Second Normal Form	✅	Single-attribute primary key,no partial dependency.
Third Normal Form	✅	No transitive dependencies.

7. **Carts :**

```
mysql> describe carts;
+------------+-----------+------+-----+-------------------+-------------------+
| Field      | Type      | Null | Key | Default           | Extra             |
+------------+-----------+------+-----+-------------------+-------------------+
| id         | int       | NO   | PRI | NULL              | auto_increment    |
| buyer_id   | int       | NO   | MUL | NULL              |                   |
| artwork_id | int       | NO   | MUL | NULL              |                   |
| added_at   | timestamp | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+------------+-----------+------+-----+-------------------+-------------------+
4 rows in set (0.00 sec)
```

Normalization check :

Normal Form	Status	Description
First Normal Form	✅	All fields are atomic, such as added_at.
Second Normal Form	✅	All non-key attributes are fully functionally dependent on art_ID
Third Normal Form	✅	There is no transitive dependency among non-key attributes.

9.Categories Table :

```
mysql> describe categories;
+-------+-------------+------+-----+---------+----------------+
| Field | Type        | Null | Key | Default | Extra          |
+-------+-------------+------+-----+---------+----------------+
| id    | int         | NO   | PRI | NULL    | auto_increment |
| name  | varchar(50) | YES  | UNI | NULL    |                |
+-------+-------------+------+-----+---------+----------------+
2 rows in set (0.00 sec)
```

Normalization check :

Normal Form	Status	Description
First Normal Form	✅	All values are atomic.
Second Normal Form	✅	Single-attribute primary key. No partial dependency.
Third Normal Form	✅	No transitive dependencies.

10.Reviews table:

```
mysql> describe reviews;
+------------+-----------+------+-----+-------------------+-------------------+
| Field      | Type      | Null | Key | Default           | Extra             |
+------------+-----------+------+-----+-------------------+-------------------+
| id         | int       | NO   | PRI | NULL              | auto_increment    |
| artwork_id | int       | YES  | MUL | NULL              |                   |
| buyer_id   | int       | YES  | MUL | NULL              |                   |
| rating     | int       | YES  |     | NULL              |                   |
| comment    | text      | YES  |     | NULL              |                   |
| created_at | timestamp | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+------------+-----------+------+-----+-------------------+-------------------+
6 rows in set (0.01 sec)
```

Normalization check :

Normal Form	Status	Description
First Normal Form	✅	Atomic values are used (Rating, Comment).
Second Normal Form	✅	All fields depend on Review_ID which is the primary key.
Third Normal Form	✅	No non-key attribute determines another non-key attribute.

11. Custom_requests Table:

```
mysql> describe custom_requests;
+-----------------+-----------------------------------------------------------------+------+-----+-------------------+-----------------------------------------------
---------+
| Field           | Type                                                            | Null | Key | Default           | Extra
      |
+-----------------+-----------------------------------------------------------------+------+-----+-------------------+-----------------------------------------------
---------+
| id              | int                                                             | NO   | PRI | NULL              | auto_increment
      |
| buyer_id        | int                                                             | NO   | MUL | NULL              |
      |
| artist_id       | int                                                             | NO   | MUL | NULL              |
      |
| description     | text                                                            | NO   |     | NULL              |
      |
| status          | enum('pending','accepted','rejected','in progress','completed') | YES  |     | pending           |
      |
| created_at      | timestamp                                                       | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED
      |
| updated_at      | timestamp                                                       | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED on update CURRENT_T
IMESTAMP |
| artist_response | text                                                            | YES  |     | NULL              |
      |
+-----------------+-----------------------------------------------------------------+------+-----+-------------------+-----------------------------------------------
---------+
8 rows in set (0.01 sec)
```

Normalization check :

Normal Form	Status	Description
First Normal Form	✅	Each attribute holds atomic values (Status, Timestamps).
Second Normal Form	✅	Fully functionally dependent on Request_ID.
Third Normal Form	✅	No transitivve dependencies exist

12. Artwork_Reports Table:

```
mysql> describe artwork_reports;
+-------------------+-----------------------------------------+------+-----+-------------------+-------------------+
| Field             | Type                                    | Null | Key | Default           | Extra             |
+-------------------+-----------------------------------------+------+-----+-------------------+-------------------+
| id                | int                                     | NO   | PRI | NULL              | auto_increment    |
| reporter_username | varchar(100)                            | NO   | MUL | NULL              |                   |
| artwork_id        | int                                     | NO   | MUL | NULL              |                   |
| reason            | varchar(255)                            | NO   |     | NULL              |                   |
| description       | text                                    | YES  |     | NULL              |                   |
| status            | enum('pending','reviewed','dismissed')  | YES  |     | pending           |                   |
| created_at        | timestamp                               | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+-------------------+-----------------------------------------+------+-----+-------------------+-------------------+
7 rows in set (0.01 sec)
```

Normalization check :

Normal Form	Status	Description
First Normal Form	✅	All data fields are atomic.
Second Normal Form	✅	Attributes are fully dependent on Report_ID.
Third Normal Form	✅	No transitive dependency is observed.

13. User_reports Table:

```
mysql> describe user_reports;
+-------------------+--------------+------+-----+-------------------+-------------------+
| Field             | Type         | Null | Key | Default           | Extra             |
+-------------------+--------------+------+-----+-------------------+-------------------+
| id                | int          | NO   | PRI | NULL              | auto_increment    |
| reporter_username | varchar(255) | YES  | MUL | NULL              |                   |
| reported_username | varchar(255) | YES  | MUL | NULL              |                   |
| reason            | text         | YES  |     | NULL              |                   |
| description       | text         | YES  |     | NULL              |                   |
| created_at        | timestamp    | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+-------------------+--------------+------+-----+-------------------+-------------------+
6 rows in set (0.00 sec)
```

Normalization check :

Normal Form	Status	Description
First Normal Form	✅	Each field has atomic data.

Second Normal Form	✅	Attributes fully depend on ID.
Third Normal Form	✅	No derived or transitive dependencies.

14.Report_notifications:

```
mysql> describe report_notifications;
+------------+------------+------+-----+-------------------+-------------------+
| Field      | Type       | Null | Key | Default           | Extra             |
+------------+------------+------+-----+-------------------+-------------------+
| id         | int        | NO   | PRI | NULL              | auto_increment    |
| report_id  | int        | NO   | MUL | NULL              |                   |
| message    | text       | NO   |     | NULL              |                   |
| is_read    | tinyint(1) | YES  |     | 0                 |                   |
| created_at | timestamp  | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+------------+------------+------+-----+-------------------+-------------------+
5 rows in set (0.01 sec)
```

Normalization check :

Normal Form	Status	Description
First Normal Form	✅	Atomic values (Message, is_read).
Second Normal Form	✅	Fully dependent on ID primary key.
Third Normal Form	✅	No non-key to non-key attribute dependency.

SQL Statements

1. **Creation of User table :**

```
mysql> create table users(
    -> id int not null auto_increment primary key,
    -> email varchar(100) unique,
    -> password varchar(255) not null,
    -> first_name varchar(50),
    -> last_name varchar(50),
    -> profile_pic varchar(255),
    -> bio text,
    -> created_at timestamp default current_timestamp,
    -> updated_at timestamp default current_timestamp on update current_timestamp
    -> );
Query OK, 0 rows affected (0.09 sec)
```

---

2. **Creation of Categories:**

```
mysql> create table categories(
    -> id int not null auto_increment primary key,
    -> name varchar(50) unique
    -> );
Query OK, 0 rows affected (0.03 sec)
```

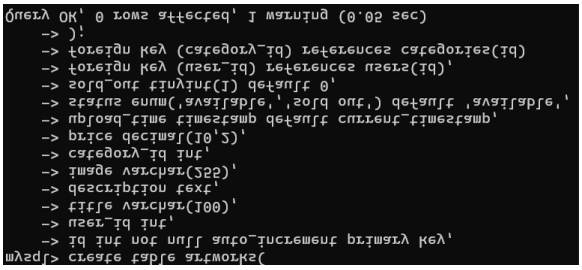
3. **Creation of tags table :**

```
mysql> create table tags(
    -> id int not null auto_increment primary key,
    -> name varchar(50) unique
    -> );
Query OK, 0 rows affected (0.03 sec)
```

4. **Creation of district table**

```
mysql> create table districts(
    -> id int not null auto_increment primary key,
    -> name varchar(100) not null unique
    -> );
Query OK, 0 rows affected (0.03 sec)
```

5. **Creation of Artworks table**



6. **Creation of Artwork table**

```
mysql> create table artwork_tags(
    -> artwork_id int not null,
    -> tag_id int not null,
    -> primary key (artwork_id, tag_id),
    -> foreign key (artwork_id) references artworks(id),
    -> foreign key (tag_id) references tags(id)
    -> );
Query OK, 0 rows affected (0.04 sec)
```

7. **Creation of reviews table**

```
mysql> create table artwork_tags(
    -> artwork_id int not null,
    -> tag_id int not null,
    -> primary key (artwork_id, tag_id),
    -> foreign key (artwork_id) references artworks(id),
    -> foreign key (tag_id) references tags(id)
    -> );
Query OK, 0 rows affected (0.04 sec)
```

8. **Creation of carts table**

```
mysql> create table carts(
    -> id int not null auto_increment primary key,
    -> buyer_id int not null,
    -> artwork_id int not null,
    -> added_at timestamp default current_timestamp,
    -> foreign key (buyer_id) references users(id),
    -> foreign key (artwork_id) references artworks(id)
    -> );
Query OK, 0 rows affected (0.04 sec)
```

9. **Creation of Orders table**

```
mysql> create table orders(
    -> id int not null auto_increment primary key,
    -> artwork_id int not null,
    -> buyer_id int not null,
    -> buyer_name varchar(100),
    -> mobile varchar(20),
    -> email varchar(100),
    -> address text,
    -> postcode varchar(20),
    -> message text,
    -> total_price decimal(10,2),
    -> status enum('pending','accepted','processing','sent to delivery','rejected','sold out') default 'pending',
    -> created_at timestamp default current_timestamp,
    -> district_id int,
    -> foreign key (artwork_id) references artworks(id),
    -> foreign key (buyer_id) references users(id),
    -> foreign key (district_id) references districts(id)
    -> );
Query OK, 0 rows affected (0.05 sec)
```

10. Creation of Custom_requests table :

```
mysql> create table custom_requests(
    -> id int not null auto_increment primary key,
    -> buyer_id int not null,
    -> artist_id int not null,
    -> description text not null,
    -> status enum('pending','accepted','rejected','in progress','completed') default 'pending',
    -> created_at timestamp default current_timestamp,
    -> updated_at timestamp default current_timestamp on update current_timestamp,
    -> artist_response text,
    -> foreign key (buyer_id) references users(id),
    -> foreign key (artist_id) references users(id)
    -> );
Query OK, 0 rows affected (0.04 sec)
```

11. Creation of artwork_reports:

```
mysql> create table artwork_reports(
    -> id int not null auto_increment primary key,
    -> reporter_username varchar(100) not null,
    -> artwork_id int not null,
    -> reason varchar(255) not null,
    -> description text,
    -> status enum('pending','reviewed','dismissed') default 'pending',
    -> created_at timestamp default current_timestamp,
    -> foreign key (reporter_username) references users(username),
    -> foreign key (artwork_id) references artworks(id)
    -> );
Query OK, 0 rows affected (0.04 sec)
```

12. Creation of user_reports table :

```
mysql> create table user_reports(
    -> id int not null auto_increment primary key,
    -> reporter_username varchar(255),
    -> reported_username varchar(255),
    -> reason text,
    -> description text,
    -> created_at timestamp default current_timestamp,
    -> foreign key (reporter_username) references users(username),
    -> foreign key (reported_username) references users(username)
    -> );
Query OK, 0 rows affected (0.14 sec)
```

13. Creation of report_notifications table :

```
mysql> create table report_notifications(
    -> id int not null auto_increment primary key,
    -> report_id int not null,
    -> message text not null,
    -> is_read tinyint(1) default 0,
    -> created_at timestamp default current_timestamp,
    -> foreign key (report_id) references artwork_reports(id)
    -> );
Query OK, 0 rows affected, 1 warning (0.13 sec)
```

14. Creation of order_notifications table :

```
mysql> create table order_notifications(
    -> id int not null auto_increment primary key,
    -> artist_id int,
    -> order_id int,
    -> is_read tinyint(1) default 0,
    -> created_at datetime default current_timestamp,
    -> foreign key (artist_id) references users(id),
    -> foreign key (order_id) references orders(id)
    -> );
Query OK, 0 rows affected, 1 warning (0.07 sec)
```

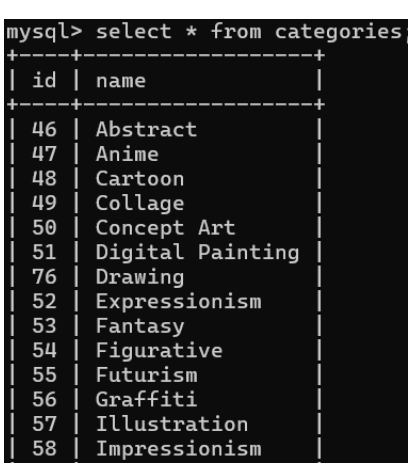
Population of Tables : Among all the tables we need to insert some tables –

1. District table :

id	name
1	Bagerhat
2	Bandarban
3	Barguna
4	Barisal
5	Bhola
6	Bogra
7	Brahmanbaria
8	Chandpur
9	Chapai Nawabganj
10	Chattogram
11	Chuadanga
12	Comilla
13	Cox's Bazar
14	Dhaka
15	Dinajpur
16	Faridpur
17	Feni
18	Gaibandha
19	Gazipur
20	Gopalganj
21	Habiganj
22	Jamalpur
23	Jashore
24	Jhalokathi
25	Jhenaidah
26	Joypurhat
27	Khagrachari
28	Khulna
29	Kishoreganj
30	Kurigram
31	Kushtia
32	Lakshmipur
33	Lalmonirhat
34	Madaripur
35	Magura
36	Manikganj
37	Meherpur
38	Moulvibazar
39	Munshiganj
40	Mymensingh
41	Naogaon
42	Narail
43	Narayanganj
44	Narsingdi
45	Natore
46	Netrokona
47	Nilphamari
48	Noakhali
49	Pabna
50	Panchagarh
51	Patuakhali
52	Pirojpur
53	Rajbari
54	Rajshahi
55	Rangamati
56	Rangpur

29	Kishoreganj
30	Kurigram
31	Kushtia
32	Lakshmipur
33	Lalmonirhat
34	Madaripur
35	Magura
36	Manikganj
37	Meherpur
38	Moulvibazar
39	Munshiganj
40	Mymensingh
41	Naogaon
42	Narail
43	Narayanganj
44	Narsingdi
45	Natore
46	Netrokona
47	Nilphamari
48	Noakhali
49	Pabna
50	Panchagarh
51	Patuakhali
52	Pirojpur
53	Rajbari
54	Rajshahi
55	Rangamati
56	Rangpur

1. **Categories Table :**



```
mysql> select * from categories;
+----+------------------+
| id | name             |
+----+------------------+
| 46 | Abstract         |
| 47 | Anime            |
| 48 | Cartoon          |
| 49 | Collage          |
| 50 | Concept Art      |
| 51 | Digital Painting |
| 76 | Drawing          |
| 52 | Expressionism    |
| 53 | Fantasy          |
| 54 | Figurative       |
| 55 | Futurism         |
| 56 | Graffiti         |
| 57 | Illustration     |
| 58 | Impressionism    |
| 60 | Line Art         |
| 79 | Manga Arts       |
| 61 | Minimalism       |
| 62 | Mixed Media      |
| 63 | Modern           |
| 64 | Nature           |
| 80 | Paintings        |
| 77 | Pen & Ink        |
| 65 | Pop Art          |
| 66 | Portrait         |
| 67 | Realism          |
| 68 | Sketch           |
| 69 | Still Life       |
| 70 | Street Art       |
| 71 | Surrealism       |
| 72 | Typography       |
| 73 | Urban            |
| 74 | Vector Art       |
| 75 | Vintage          |
+----+------------------+
```

2. **Tags table :**

```
mysql> select * from tags;
+-----+----------------+
| id  | name           |
+-----+----------------+
|  58 | 3D             |
|  59 | Acrylic        |
| 155 | Animal         |
|  60 | Animated       |
| 108 | Anime          |
| 106 | Apocalyptic    |
| 139 | Bangladesh     |
| 142 | Beach          |
| 140 | Bengali        |
|  61 | Black & White  |
| 110 | Blue Sky       |
|  62 | Bold           |
|  63 | Cartoonish     |
| 157 | Cat            |
|  64 | Charcoal       |
|  65 | Clean          |
| 153 | Closeup        |
| 131 | Coffee         |
|  67 | Collage        |
|  66 | Colorful       |
|  68 | Comic          |
| 119 | Countryside    |
|  77 | Impressionist  |
|  78 | Ink            |
| 101 | Ink Drawing    |
| 117 | Lake           |
|  97 | Landscape      |
|  79 | Linework       |
|  80 | Low Poly       |
| 143 | Man            |
|  81 | Matte          |
|  82 | Mixed Media    |
|  83 | Monochrome     |
| 113 | Moon           |
| 116 | Mountain       |
| 112 | Nature         |
|  84 | Oil            |
| 148 | Old            |
| 129 | Park           |
|  85 | Patterned      |
| 109 | Peaceful       |
| 128 | Pencil Drawing |
|  86 | Photorealistic |
| 146 | Portrait       |
| 133 | Realistic      |
| 102 | Rural          |
| 103 | Serene         |
| 114 | Sky            |
| 107 | Slice of Life  |
|  96 | Snow           |
| 126 | Still life     |
|  88 | Stylized       |
| 120 | Sunny          |
| 151 | Symbolism      |
| 132 | Tea            |
|  89 | Textured       |
|  90 | Traditional    |
|  91 | Transparent    |
|  99 | Trees          |
|  92 | Vector         |
| 135 | Vibrant        |
| 141 | Village        |
|  93 | Vintage        |
|  94 | Watercolor     |
| 123 | Waterfall      |
| 118 | Wild Flowers   |
|  95 | Winter         |
| 144 | Woman          |
+-----+----------------+
98 rows in set (0.13 sec)
```

Pattern Identification Queries : To understand user activity and system interaction in the ArtBlossoms platform, the following SQL queries were designed and applied. These pattern identification queries are directly aligned with the system's implemented features.

1. Shows all reported artworks with full details of the report, the reported artwork, and the artist's status.

```
SELECT ar.id, ar.reporter_username, ar.artwork_id, a.title AS
artwork_title,
            ar.reason, ar.description, ar.created_at,
u.username AS artist_username,
            u.banned AS is_banned
        FROM artwork_reports ar
        JOIN artworks a ON ar.artwork_id = a.id
        JOIN users u ON a.user_id = u.id
        ORDER BY ar.created_at DESC
```

2. Fetches full details of a specific artwork, including its artist and category.

```
SELECT a.*, u.username, u.first_name, u.last_name, c.name AS
category_name
                FROM artworks a
                JOIN users u ON a.user_id = u.id
                JOIN categories c ON a.category_id = c.id
                WHERE a.id = %s
```

3. Lists tags attached to a specific artwork.

```
SELECT t.name
               FROM tags t
               JOIN artwork_tags at ON t.id = at.tag_id
               WHERE at.artwork_id = %s
```

4. Fetches all reviews on a given artwork, with reviewer names, sorted by latest first.

```
SELECT r.*, u.first_name, u.last_name
                FROM reviews r
                JOIN users u ON r.buyer_id = u.id
                WHERE r.artwork_id = %s
                ORDER BY r.created_at DESC
```

5. Counts how many times a specific user has ordered a specific artwork and received it.

```
SELECT COUNT(*) AS count
                FROM orders o
                WHERE o.artwork_id = %s
                  AND o.buyer_username = %s
                  AND o.status = 'Sent to Delivery'
```

6. Fetch artwork details

```
SELECT a.*, u.username, u.first_name, u.last_name, c.name AS
category_name
                FROM artworks a
                JOIN users u ON a.user_id = u.id
                JOIN categories c ON a.category_id = c.id
                WHERE a.id = %s
```

7. Get tags for artwork

```
SELECT t.name
                FROM tags t
                JOIN artwork_tags at ON t.id = at.tag_id
                WHERE at.artwork_id = %s
```

8. List reviews on an artwork

```
SELECT r.*, u.first_name, u.last_name
                FROM reviews r
                JOIN users u ON r.buyer_id = u.id
                WHERE r.artwork_id = %s
                ORDER BY r.created_at DESC
```

9. Duplicate of query 5: Counts successful delivered orders for a specific user-artwork pair.

```
SELECT COUNT(*) AS count
                FROM orders o
                WHERE o.artwork_id = %s
                  AND o.buyer_username = %s
                  AND o.status = 'Sent to Delivery'
```

10. Returns full artwork data with artist name, category, and average rating (even if no reviews).

```
 SELECT
                a.id, a.title, a.image, a.price, a.upload_time,
                u.username, u.first_name, u.last_name,
                c.name AS category_name,
                COALESCE(AVG(r.rating), 0) AS avg_rating
            FROM artworks a
            JOIN users u ON a.user_id = u.id
            JOIN categories c ON a.category_id = c.id
            LEFT JOIN reviews r ON r.artwork_id = a.id
            LEFT JOIN artwork_tags at ON a.id = at.artwork_id
            LEFT JOIN tags t ON at.tag_id = t.id
            WHERE 1 = 1
```

11. Fetches the username of the artist who owns a specific artwork.

```
SELECT u.username
            FROM artworks a
            JOIN users u ON a.user_id = u.id
            WHERE a.id = %s
```

12. Shows items currently in a buyer's cart, including artwork and artist info.

```
            SELECT
                a.id, a.title, a.price, a.image,
                u.first_name, u.last_name
            FROM carts c
            JOIN artworks a ON c.artwork_id = a.id
            JOIN users u ON a.user_id = u.id
            WHERE c.buyer_username = %s
```

13. Shows all commission requests received by an artist, including buyer info.

```
            SELECT cr.id, cr.description, cr.status,
u.username AS buyer_username, u.first_name, u.last_name
            FROM custom_requests cr
            JOIN users u ON cr.buyer_id = u.id
            WHERE cr.artist_id = %s
            ORDER BY cr.id DESC
```

14. Counts unread report notifications for a user.

```
            SELECT COUNT(*) FROM report_notifications
            WHERE reporter_username = %s AND is_read = 0
```

15. Shows all orders for artworks owned by a specific artist, with buyer info.

```
            SELECT o.*, a.title AS artwork_title, a.image AS
artwork_image, u.username AS buyer_username
            FROM orders o
            JOIN artworks a ON o.artwork_id = a.id
            JOIN users u ON o.buyer_username = u.username
            WHERE a.user_id = %s
            ORDER BY o.created_at DESC
```

16. Lists all orders placed by a specific buyer, including artwork title and image.

```
                    SELECT o.*, a.title AS artwork_title,
a.image AS artwork_image
                    FROM orders o
                    JOIN artworks a ON o.artwork_id = a.id
                    WHERE o.buyer_username = %s
                    ORDER BY o.created_at DESC
```

17. Shows commission requests sent by a buyer, including the artist's info and response.

```
                    SELECT cr.description, cr.status,
cr.artist_response,
                           u.first_name AS artist_first,
u.last_name AS artist_last, u.username AS artist_username
                    FROM custom_requests cr
                    JOIN users u ON cr.artist_id = u.id
                    WHERE cr.buyer_id = %s
                    ORDER BY cr.id DESC
```

18. Displays order notifications received by an artist, showing buyer, artwork, and message.

```
                    SELECT n.id, o.message, o.status,
n.created_at, o.buyer_username, a.title
                    FROM order_notifications n
                    JOIN orders o ON n.order_id = o.id
                    JOIN artworks a ON o.artwork_id = a.id
                    WHERE n.artist_id = %s
                    ORDER BY n.created_at DESC
```

19. Shows report notifications submitted by a user, ordered by latest

```
                SELECT id, message, is_read, created_at
                FROM report_notifications
                WHERE reporter_username = %s
                ORDER BY created_at DESC
```

```
SELECT id, message, is_read, created_at
FROM report_notifications
WHERE reporter_username = %s
ORDER BY created_at DESC
```

20. Inserts a new user into the users table, used during registration.

```
INSERT INTO users
(first_name, last_name, username,
email, mobile, role, profile_pic, password)
VALUES (%s, %s, %s, %s, %s, %s, %s, %s)
""", (first, last, username, email, mobile,
role, filename, hashed_password))
```

21. Displays all reports submitted by a specific user.

```
SELECT * FROM user_reports
 WHERE reporter_username = %s
 ORDER BY created_at DESC
```

Progress Report:

Summary: Our team of three successfully collaborated to develop the *ArtBlossoms* project by dividing responsibilities across key areas. One member focused on designing and normalizing the database using MySQL, ensuring efficient data storage and relationships. Another handled the backend using Python and Flask, building the core logic, authentication, and database connectivity. The third member developed the frontend using HTML, CSS, and Bootstrap to create a user-friendly interface for buyers and artists. Together, we integrated all components into a fully functional art marketplace system.

Group members :

Name and ID	Total Hours	Remarks
Ratrixmna Chakma (23549009021)	20.06.2025, 07:00 PM – 10:00 PM (3 hours) 22.06.2025, 09:00 PM – 01:30 AM (4 hours 30 minutes) 26.06.2025, 08:00 PM – 12:00 AM (4 hours) 30.06.2025, 08:00 PM – 10:00 PM (2 hours)	Designed ER diagram, define entities, attributes, and relationships. Created and normalized tables added key constraints and foreign keys. Wrote ER diagram description and cross checked relational schema. Finalized table relationships and database schema in MySQL.
Laiba Sumaiya Nazim (23549009041)	21.06.2025, 07:30 PM – 10:30 PM (3 hours) 25.06.2025, 08:00 PM – 02:00 AM (6 hours) 28.06.2025, 08:00 PM – 11:30 PM (3 hours 30 minutes) 03.07.2025, 06:00 PM – 12:00 AM (6 hours)	Reviewed and corrected relational schema, ensured data consistency Connected to backend, handled order placement and status update Implemented flask backend routes, user login, and role based logic. Implemented review submission and cart checkout logic Integrated admin functionalities like banning users and deleting artworks.
Masuma Tasnim Nimo (23549009093)	23.06.2025, 06:00 PM – 10:00 PM (4 hours)	Designed user interface for profile pages, and artwork detail view

	27.06.2025, 07:30 PM – 01:30 AM (6 hours) 01.07.2025, 08:00 PM – 12:30 AM (4 hours 30 minutes) 04.07.2025, 08:00 PM – 10:30 PM (2 hours 30 minutes)	Created frontend for cart and order forms using HTML and CSS. Start admin panel and added interactive elements. Finalized responsive layout and validated from UI
Group total Time :	47 hours 30 minutes	

Individual report :

Team member Name: Ratrixmna Chakma

Reporting Period : 06.07.2025

Date	**Time period**	**Activities**
20.06.2025	3 hours	Created a database and related tables. Inserted data.
22.06.2025	2 hours 30 minutes	Helped in writing the summary and query related information.
23.06.2025 26.06.2025 30.06.2025 04.07.2025	6 hours 5 hours 8 hours 2 hours	Group meeting and discussion.
26.06.2025	3 hours	Worked on normalization of the tables within the database, ensuring all tables are 3NF.
30.06.2025	2 hours 30 minutes	Review ER diagram and relational schema, made necessary adjustments.
01.07.2025	2 hours	Finalized documentation and ensured all tables have individual records.

Team Member Name : Laiba Sumaiya Nazim

Reporting Period: 06.07.2025

Date	Time Period	Activities
21.06.2025		Studied the database and created logics for the backends.
25.06.2025		Created related files in the VS code and created routes using Python.
23.06.2025 26.06.2025 30.06.2025 04.07.2025	6 hours 5 hours 8 hours 2 hours	Group meeting and discussion
26.06.2025	6 hours	Connected MySQL and Python, installing flask. Checked if it is connected properly.
28.06.2025	4 hours	Connected the routes with the HTML files with the Python.

Team Member Name : Masuma Tasnim Nimo

Reporting period : 06.07.2025

Date	Time period	Activities
23.06.2025	3 hours	Reviewing the database and related python logics.
23.06.2025 26.06.2025 30.06.2025 04.07.2025	6 hours 5 hours 8 hours 2 hours	Group meeting and discussion
27.06.2025	3 hours	Created final schema and ER diagram.
28.06.2025	8 hours	Created related HTML files according to the routes of the project's file

01.06.2025	4 hours	Checked the HTML code is working properly and polished it if there is any scope. Lastly, finalized it.

Learning Outcomes :

1. **Technology skills**

1.1 Python (flask) : We gained hands-on experience in server-side development using Python and the Flask microframework. Through route creation, template rendering, and form processing, we developed a strong understanding of backend logic, RESTful design, and HTTP communication. Session management, error handling, and request validation were also effectively implemented using Flask.

1.2 MySQL :We designed a normalized relational database for the platform, incorporating foreign keys and indexing to ensure data integrity and optimal performance. Writing SQL queries to handle complex data retrieval, updates, and joins gave us deep insight into managing a multi-table application. Data constraints, normalization (up to 3NF), and relationship modeling were implemented in accordance with database design principles.

1.3 HTML & CSS : We created structured, responsive web pages using HTML, along with embedded Jinja2 syntax for dynamic rendering. CSS was used extensively to style components, ensuring a consistent and visually appealing interface. Layout designs using grid and flexbox helped us create a professional and intuitive user experience across different pages and user roles.

Features and explanations :

1. **User Authentication and Roles:** The system supports login, registration, and role-based access for three distinct user types — artist, buyer, and both. Depending on the role, users are shown appropriate sections and functionalities throughout the site, ensuring personalized interaction.
2. **Artwork Upload and Browsing:Artists** can upload artworks with title, description, category, tags, and image. These are displayed on a central browse page, accessible to all users and guests. Buyers can filter by category, search by name or artist, sort by price, and even search using tags.
3. **Artwork Details and Reviews:** Each artwork has a dedicated page showing full details, price, artist info, and reviews. Buyers who have completed an order (with status ‘Sent to Delivery’) can leave reviews, enabling transparent feedback.
4. **Cart and Order System:** Buyers can add artworks to their cart and either place single or grouped orders. Orders are stored artist-wise, and buyers fill in detailed order information including address and message. Artists are notified upon order placement.
5. **Order Status and Notifications:**

Artists can update the status of each order. This status change is visible to the buyer and also triggers notifications. The platform allows buyers to track delivery progress and leave reviews upon completion.

6. **Admin and Pattern Monitoring** : The backend allows for monitoring user activity trends through SQL queries, such as top buyers, most reviewed artworks, and district-wise order volume. These patterns help guide system improvements.

Practical Application of the project :

1. **Artist Marketplace:** ArtBlossoms provides an effective digital solution for artists to display and sell their work online. It functions as both a personal portfolio and a marketplace, reducing dependency on physical exhibitions.
2. **E-commerce Learning**: The cart, order, review, and delivery system closely resembles real-world e-commerce platforms. This gave us hands-on experience in developing and managing commercial workflows digitally.
3. **User Engagement Platform**: By allowing artist-buyer interactions (including messaging, reviews, and personalized profiles), the system encourages community building, which is relevant for many modern web platforms.
4. **Academic Value**: From a technical perspective, we applied principles of full-stack web development, database normalization, role-based access control, and SQL query optimization. This made it a valuable academic project grounded in real-world practice.

Limitations of this project :

1. **No Payment Integration**: The current platform lacks real payment gateway integration (e.g., Stripe, bKash, PayPal). Orders are placed manually without financial transactions, which limits real-world commercialization.
2. **Media Management**: There is no image optimization or upload size restriction, which may affect performance as more artworks are uploaded.
3. **Scalability Concerns**: With a growing user base, performance optimizations like caching, database indexing, and load handling mechanisms would be necessary.
4. **Security Enhancements**: Basic security like session-based authentication is implemented, but stronger security practices (e.g., CSRF protection, secure password hashing) need to be enforced for real-world deployment.

Conclusion :

The ArtBlossoms project has successfully demonstrated the design and development of a full-stack web-based art marketplace. It integrates multiple technologies such as Python (Flask), MySQL, HTML, CSS, and JavaScript to offer a seamless experience for artists and buyers. The platform provides core functionalities including user registration and authentication, artwork upload and browsing, cart management, order placement, delivery tracking, and review systems.

Throughout the development process, we applied critical database concepts such as normalization, foreign key constraints, and relational joins. At the same time, we gained practical skills in frontend design, responsive layouts, and user interaction. The implementation

of pattern identification queries added analytical depth to the system, helping to identify user behaviors and platform trends.

While the project meets its key objectives, certain limitations still exist, such as the absence of a payment gateway, admin controls, and scalability features. These can be addressed in future iterations of the system to improve functionality, security, and performance.

Overall, ArtBlossoms provided a comprehensive learning experience by blending theoretical knowledge with hands-on implementation. It serves not only as an academic achievement but also as a scalable foundation for a real-world digital art marketplace.

.