Republic of the Philippines
Cagayan State University
CARIG CAMPUS
Carig Sur, Tuguegarao City
www.csucarig.edu.ph

COLLEGE of INFORMATION and COMPUTING SCIENCES

VIVIDLY: ELEVATING BMMA ARTISTRY IN A DIGITAL REALM

A Software Project Proposal
to the Faculty of College of Information and Computing Sciences
Cagayan State University – Carig Campus, Tuguegarao City, Cagayan 3500

In Partial fulfilment
of the Course Requirements for
SOFTWARE ENGINNERING 1

Led by:
Quegan, Jaren G.

Members:
Castañeda, Stephen Keant N.
Navarette, Rovin B.
Tugas, Jephunneh Jediael A.

January 15, 2024

ABSTRACT

In their pursuit of creating a dynamic platform tailored for the visual arts community, the developers embarked on the journey to conceive, design, and implement VIVIDLY. This project, a culmination of extensive research and development, is a digital space dedicated to BMMA students—a virtual canvas that seamlessly integrates artistic expression, user interaction, and immersive experiences.

VIVIDLY is not merely an art showcase; it is an ecosystem designed to empower artists, allowing them to upload their creations effortlessly and curate expressive online profiles. The platform's intuitive features, ranging from interactive artworks and profiles to robust user interactions, foster a sense of community and provide a vibrant canvas for artistic growth.

The developers have carefully woven together elements of user authentication, artwork upload functionalities, and interactive profiles to create a visually compelling and immersive experience. The platform goes beyond mere presentation, allowing users to engage with artworks through comments and likes, while also affording artists the flexibility to edit and manage their portfolios.

As a testament to its versatility, VIVIDLY incorporates an administrator page, ensuring effective oversight and moderation. This page equips administrators with the tools needed to maintain a secure and vibrant community, emphasizing the commitment to creating a platform that not only serves the artistic community but also stands as a beacon of user-centric design.

In essence, VIVIDLY emerges as a result of the developers' dedication to providing a tailored and engaging space for BMMA students. It is not just a digital platform but a vivid and immersive gallery—a testament to the team's passion for fostering artistic expression and community engagement within the realm of visual arts.

TABLE OF CONTENTS

INTRODUCTION

In the ever-evolving landscape of digital platforms, VIVIDLY stands as a testament to the collaborative vision of The Developers. This project is a thoughtful response to the dynamic needs of Bachelor of Multimedia Arts students at Cagayan State University – Carig Campus, aiming to provide a tailored space where creativity flourishes and artistic expressions find a vibrant digital canvas.

VIVIDLY is a web-based platform written in PHP designed to facilitate the sharing, exploration, and interaction with visual artworks. It includes features such as user authentication, artwork uploading, interactive profiles, search functionality, and administrative tools.

The Developers set out on this journey with the specific goal of building a platform that goes beyond the conventions of the traditional creative exhibition. VIVIDLY is intended to be more than just a digital gallery; rather, it is a carefully constructed virtual environment designed to enhance BMMA students' creativity. The Developers want to provide a platform where artists can easily share their creations and immerse viewers in an aesthetically gorgeous journey by seamlessly combining visually appealing interface with user-friendly features.

This document encapsulates the essence of "VIVIDLY," detailing the careful actions that The Developers took in response to the requests made by BMMA students. The goal is apparent: to empower students by providing a seamless and immersive environment for the expression, sharing, and development of visual artworks. This paper unveils the specifics of user account creation, artwork upload processes, user interactions, and other important features that all work together to give VIVIDLY its distinct identity.

The Developers envision VIVIDLY as an integral tool in shaping the future of BMMA education, providing students with a digital space that not only elevates their artistic expression but also facilitates a fun and supportive community.

OBJECTIVES

General Objective

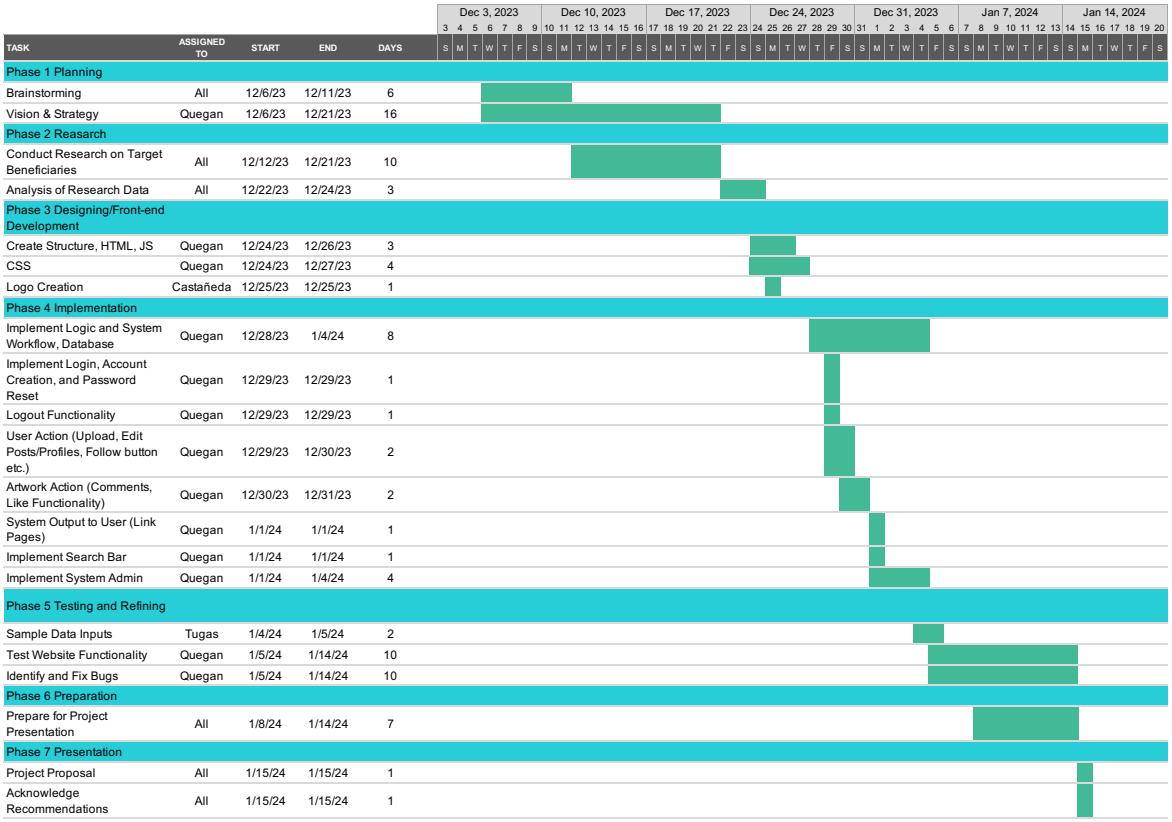
VIVIDLY aims to create a comprehensive and enriching experience for BMMA students, promoting creativity, skill development, and a supportive community within the digital realm.

Specific Objectives

1. Elevated Artistic Expression & Immersive Showcase:
    - VIVIDLY will empower BMMA students to articulate their creative visions, offering a captivating virtual gallery space for the seamless display of visual artworks. This feature ensures an immersive and engaging experience, allowing viewers to explore the depth of artistic expression within a dynamic showcase environment.
2. Seamless Artwork Uploads:
    - The platform will simplify the process of uploading visual artworks, ensuring it becomes effortless for BMMA students to share their creations on VIVIDLY. This streamlined approach facilitates efficient content sharing within the BMMA community.
3. Community Interaction:
    - VIVIDLY will foster a sense of community among BMMA students by facilitating interactions within the platform. This includes features that encourage connections, feedback, and a supportive environment for artistic growth, creating a vibrant digital community.
4. Expressive Artistic Profiles:
    - The platform will cultivate vibrant and visually engaging online portfolios for BMMA students. As visual artworks seamlessly populate their profiles, these portfolios will not only showcase the ongoing creative journey of each artist but also stand as vivid testaments to their versatility and proficiency in the realm of visual arts.
5. Technical Skill Development:
    - VIVIDLY will integrate a robust feedback and critique system, facilitating constructive input from peers. This mechanism aims to enhance technical skills and foster a culture of continuous improvement among BMMA students as they explore and refine their artistic capabilities.
6. User-Friendly Interface:
    - The platform will prioritize a user-friendly interface, ensuring easy navigation and accessibility for BMMA students with varying technical abilities. This design philosophy ensures that all students can maximize their experience on the VIVIDLY platform.

The proposed features are a direct reflection of the desires expressed by BMMA students during the research phase, ensuring that VIVIDLY aligns seamlessly with their preferences and expectations, specifically focusing on the realm of visual arts.

GANTT CHART



TASK	ASSIGNED TO	START	END	DAYS
Phase 1 Planning				
Brainstorming	All	12/6/23	12/11/23	6
Vision & Strategy	Quegan	12/6/23	12/21/23	16
Phase 2 Reasarch				
Conduct Research on Target Beneficiaries	All	12/12/23	12/21/23	10
Analysis of Research Data	All	12/22/23	12/24/23	3
Phase 3 Designing/Front-end Development				
Create Structure, HTML, JS	Quegan	12/24/23	12/26/23	3
CSS	Quegan	12/24/23	12/27/23	4
Logo Creation	Castañeda	12/25/23	12/25/23	1
Phase 4 Implementation				
Implement Logic and System Workflow, Database	Quegan	12/28/23	1/4/24	8
Implement Login, Account Creation, and Password Reset	Quegan	12/29/23	12/29/23	1
Logout Functionality	Quegan	12/29/23	12/29/23	1
User Action (Upload, Edit Posts/Profiles, Follow button etc.)	Quegan	12/29/23	12/30/23	2
Artwork Action (Comments, Like Functionality)	Quegan	12/30/23	12/31/23	2
System Output to User (Link Pages)	Quegan	1/1/24	1/1/24	1
Implement Search Bar	Quegan	1/1/24	1/1/24	1
Implement System Admin	Quegan	1/1/24	1/4/24	4
Phase 5 Testing and Refining				
Sample Data Inputs	Tugas	1/4/24	1/5/24	2
Test Website Functionality	Quegan	1/5/24	1/14/24	10
Identify and Fix Bugs	Quegan	1/5/24	1/14/24	10
Phase 6 Preparation				
Prepare for Project Presentation	All	1/8/24	1/14/24	7
Phase 7 Presentation				
Project Proposal	All	1/15/24	1/15/24	1
Acknowledge Recommendations	All	1/15/24	1/15/24	1

SOFTWARE REQUIREMENT SPECIFICATIONS

System Features

1. User Authentication
2. Artwork Upload
3. Interactive Profiles/Portfolios
4. Search Functionality
5. Artwork Interaction
6. User Interaction
7. Administrator Page

Functional requirements

1. User Authentication
    - Users can create accounts with unique usernames and passwords.
    - Users can log in securely to access the platform.

2. Artwork Upload
    - Users can upload visual artworks to their profiles.
    - Artwork uploads include metadata such as title, description, and tags.
    - Artworks are associated with the user's profile.
3. Interactive Profiles/Portfolios
    - User profiles display uploaded artworks.
    - Users can edit their profiles, including profile pictures and bio.
    - Users can follow other artists and be followed.
4. Search Functionality
    - Users can search for specific artists or artworks.
    - Search results are displayed in a user-friendly manner.
5. Artwork Interaction
    - Users can view full-size artworks with detailed descriptions.
    - Users can leave comments on artworks.
    - Users can like artworks.
6. User Interaction
    - Users can edit their artworks and profiles.
    - Users can make comments, upload, or update profile picture, change passwords, like artworks, follow artists, delete comments, search for artists or artworks, delete uploaded artworks, delete profile, send feedback, send reviews, etc.
7. Administrator Page
    - Admins can manage the system, including deleting artworks and user accounts.
    - Admins can track user reviews and feedback.

Non-Functional requirements

1. Performance
    - Artwork uploads and page loads should not exceed 5 seconds.
2. Security
    - User authentication and data transmission should be secure.
    - Admin functionalities should be accessible only to authorized personnel.
3. Usability
    - The user interface should be intuitive and user-friendly.
    - The platform should be accessible on major web browsers.

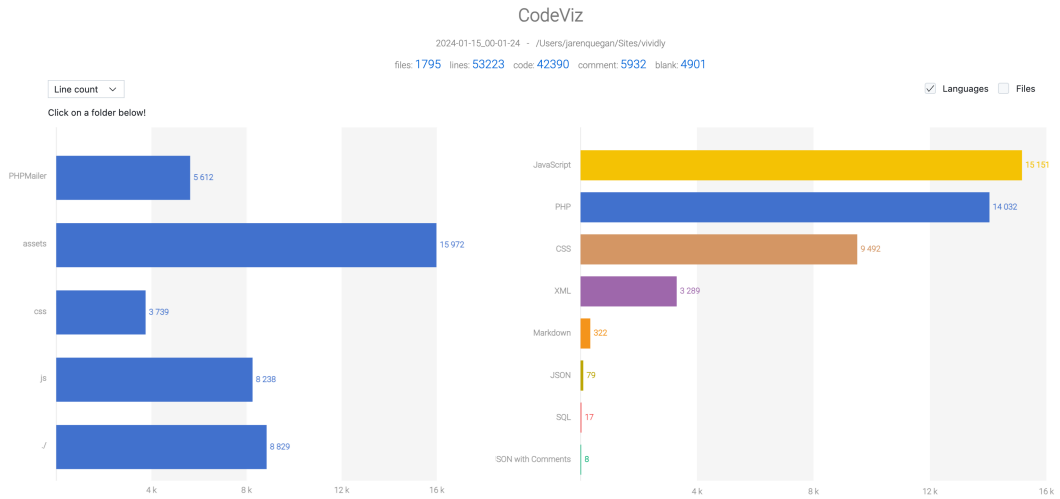
Requirement Elicitation Techniques

The Developers recognize the critical importance of accurately gathering and understanding the requirements for VIVIDLY. To ensure a comprehensive and effective requirement elicitation process, the team will employ specific techniques aimed at extracting relevant information from stakeholders.

1. Surveys and Questionnaires
   The Developers will leverage surveys and questionnaires as a primary method for requirement elicitation. The process involves:
   - Survey Creation: Develop structured surveys using Google Forms (Refer to Exhibit A).
   - Distribution: Disseminate surveys among BMMA students, the potential users.
   - Data Gathering: Collect quantitative data on user preferences, expectations, and desired features (Refer to Exhibit B for the results).
2. Brainstorming Sessions
   Organized brainstorming sessions will encourage open discussions among team members. The process involves:
   - Session Facilitation: Conduct brainstorming sessions to generate creative ideas.
   - Prioritization: Prioritize ideas based on feasibility and stakeholder relevance.

VIVIDLY SOFTWARE METRICS

Code Metrics



In their pursuit of creating a robust and dynamic digital platform, the developers conducted a comprehensive code analysis using the CodeViz Visual Studio Code extension for VIVIDLY. The results reveal a total of 53,223 lines of code distributed across various programming languages.

The breakdown includes a total of 1,795 files, encompassing 42,390 lines of actual code, 5,932 lines of comments, and 4,901 blank lines. The language distribution within the codebase highlights the following:

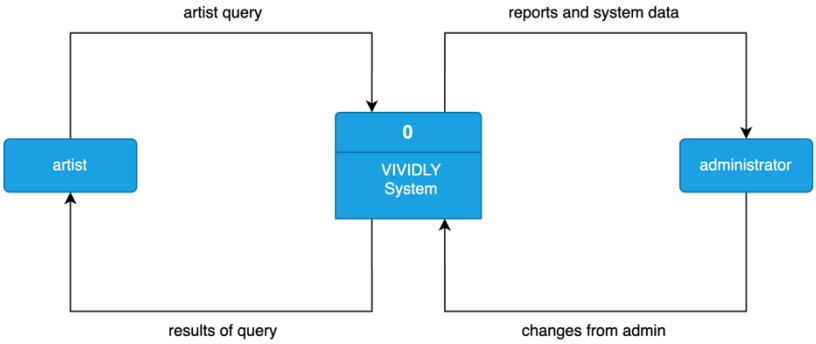
- JavaScript: 15,151 lines
- PHP: 14,032 lines
- CSS: 9,492 lines
- XML: 3,289 lines
- Markdown: 322 lines
- JSON: 79 lines
- SQL: 17 lines
- SON with comments: 8 lines

These metrics not only quantify the scale of the development effort invested by the developers but also provide insights into the diverse programming languages contributing to the functionality and aesthetics of the VIVIDLY platform. The breakdown emphasizes the comprehensive nature of the development, encompassing both frontend and backend components to create a rich and engaging user experience.

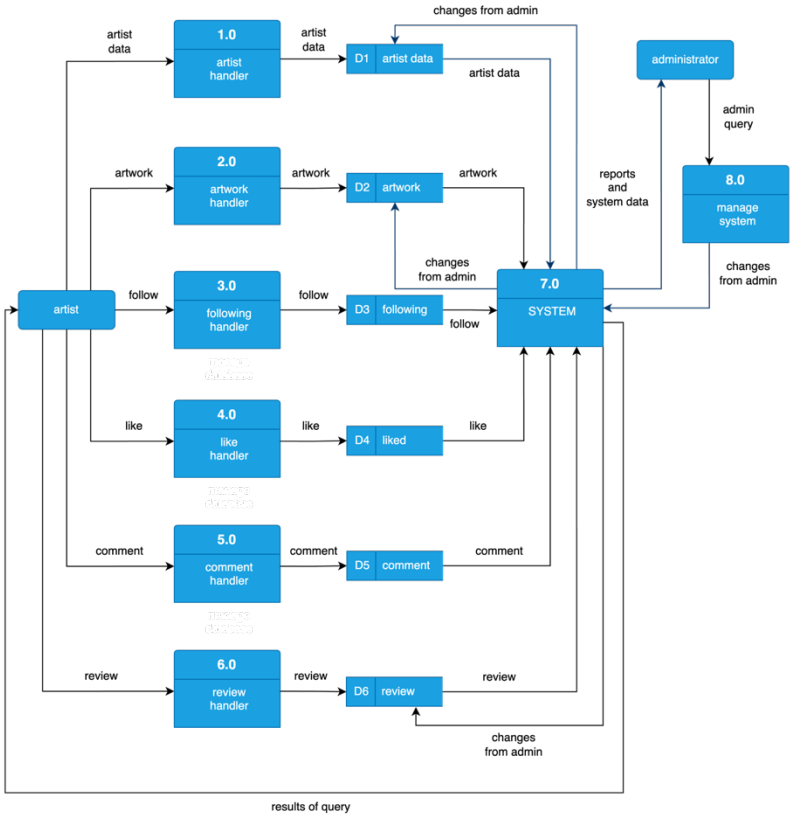
VIVIDLY DIAGRAM

Data Flow Diagram

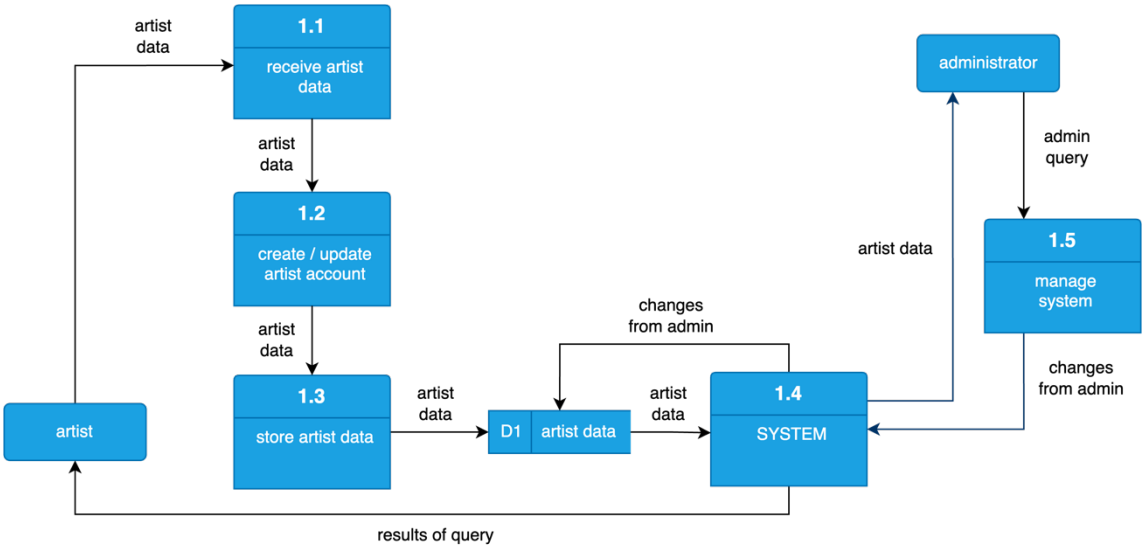
Context Diagram



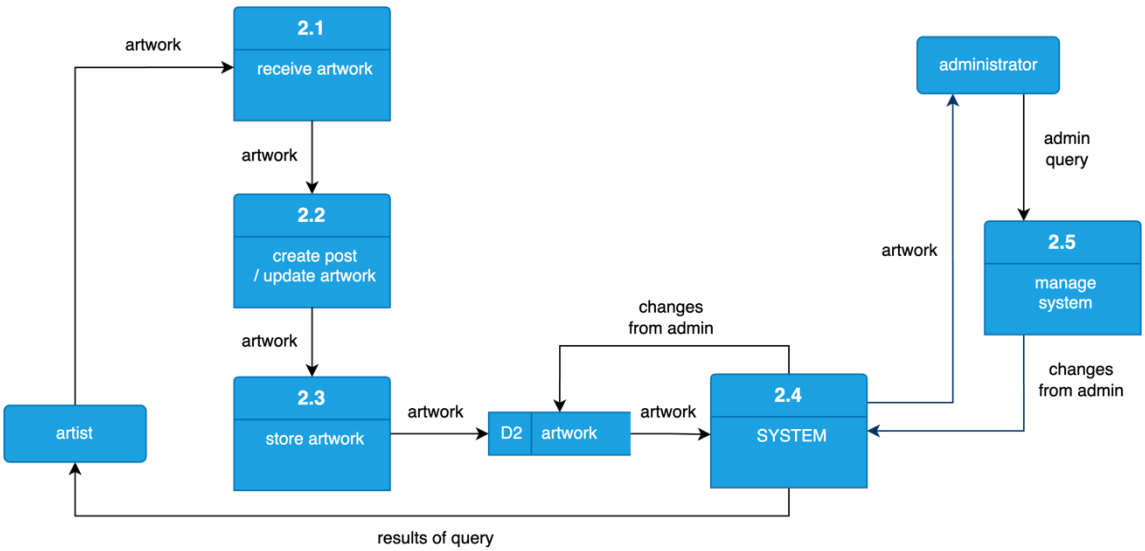
Level 1 Data Flow Diagram



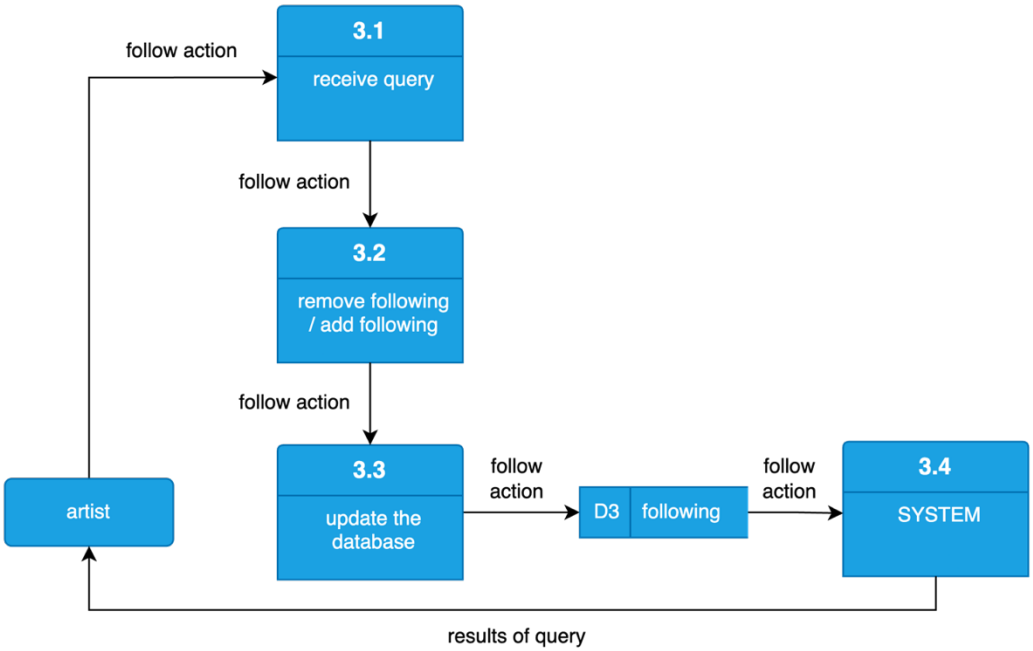
Child of Artist Handler



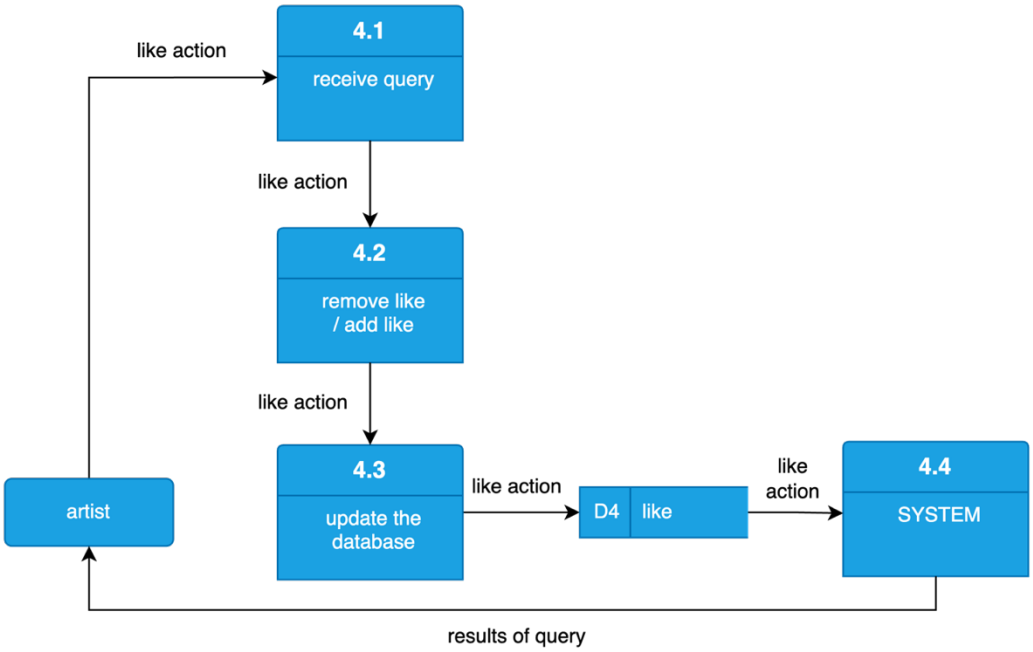
Child of Artwork Handler



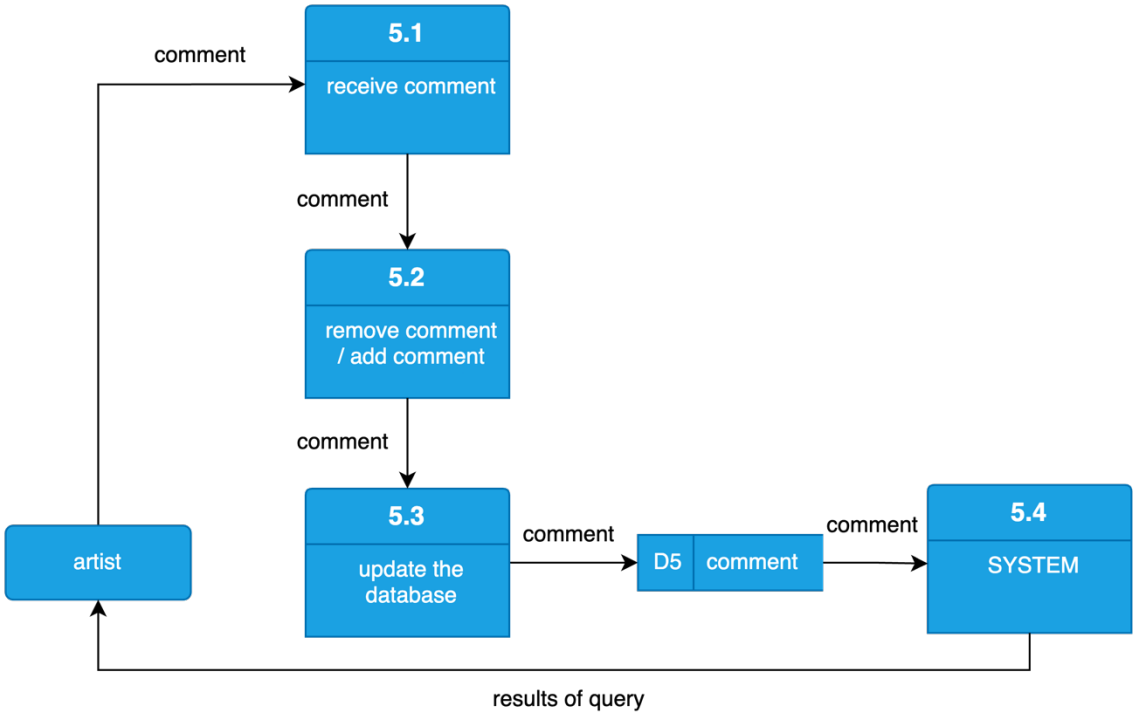
Child of Following Handler



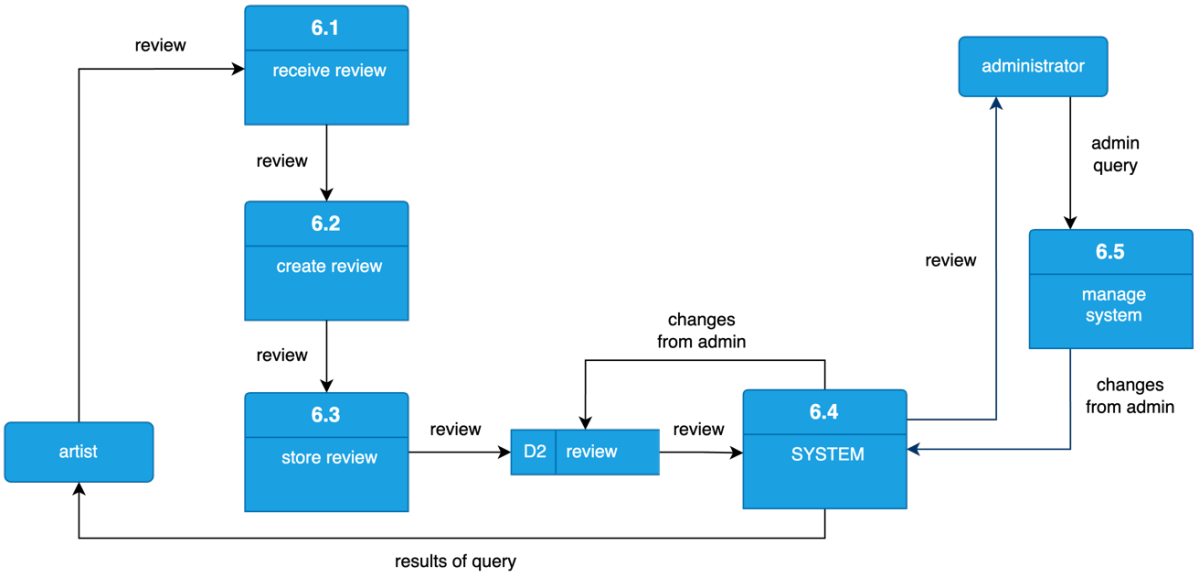
Child of Like Handler



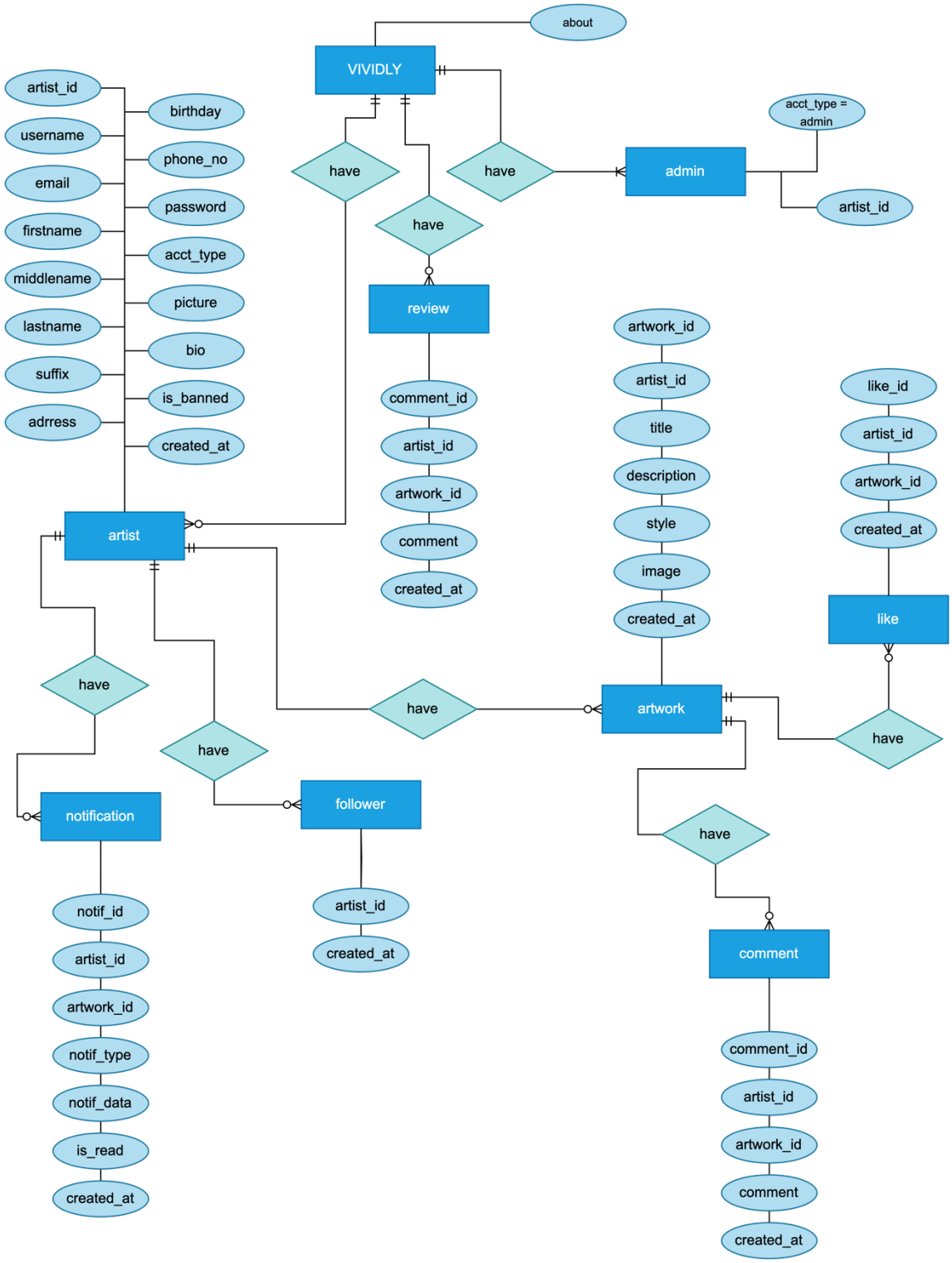
Child Comment Handler



Child of Review Handler



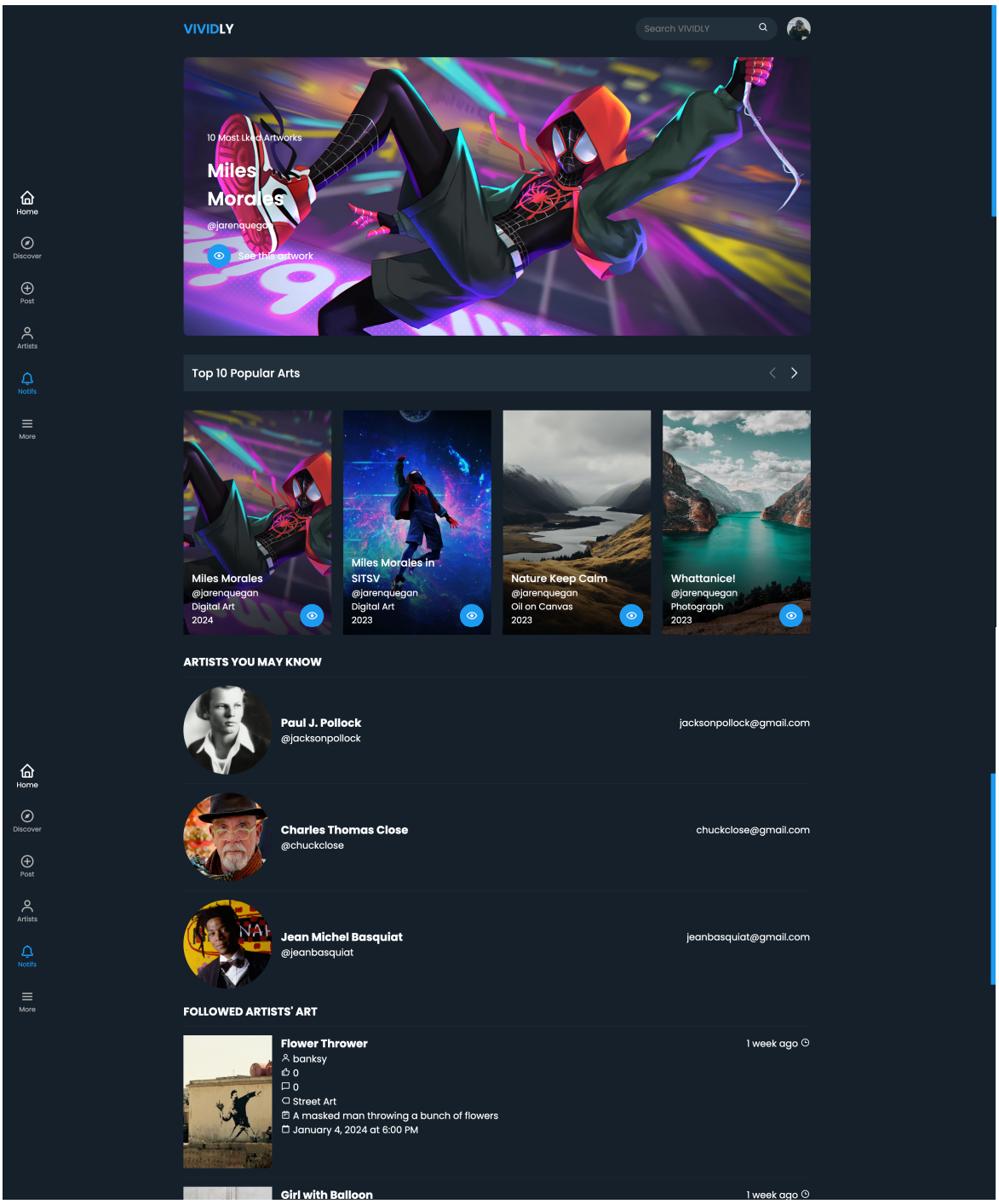
Entity Relationship Diagram



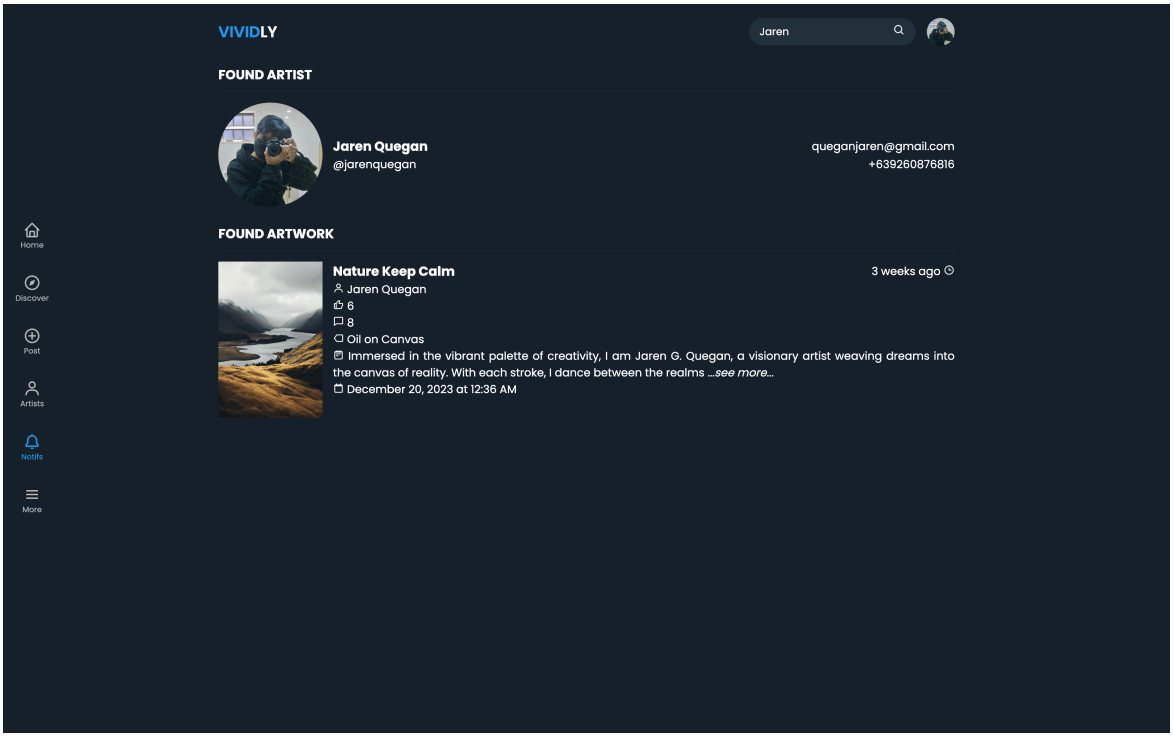
VIVIDLY INTERFACE

Regular User

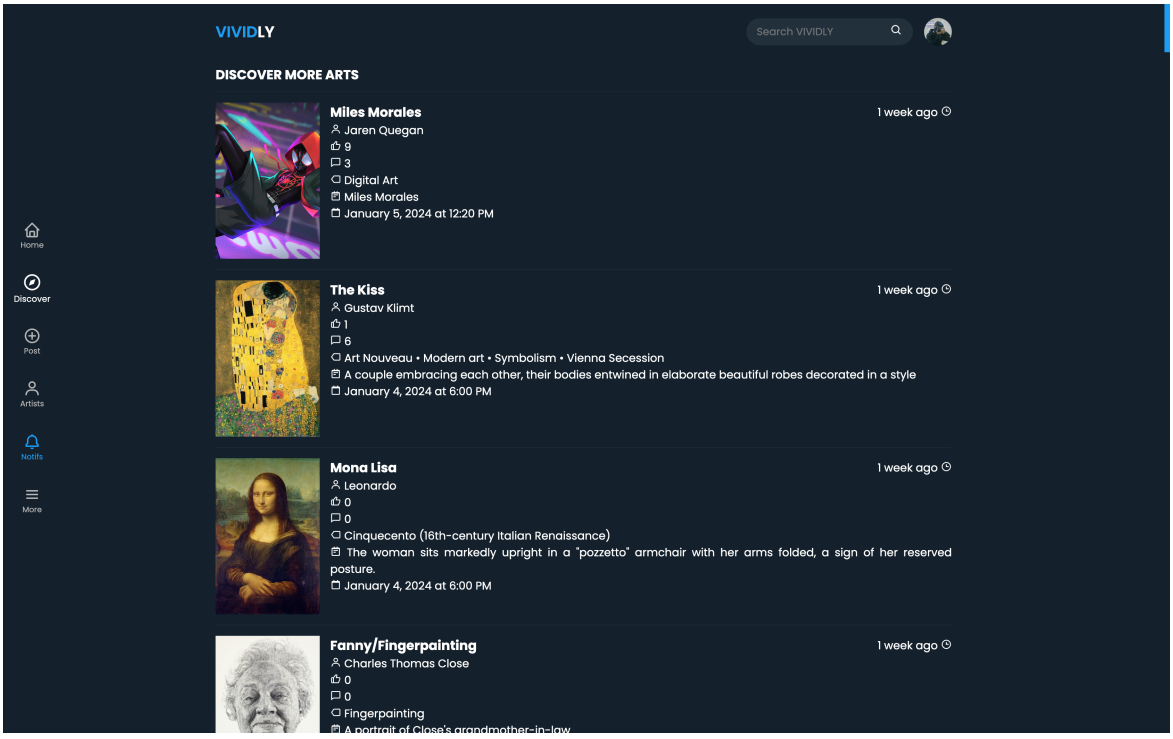
Home Page



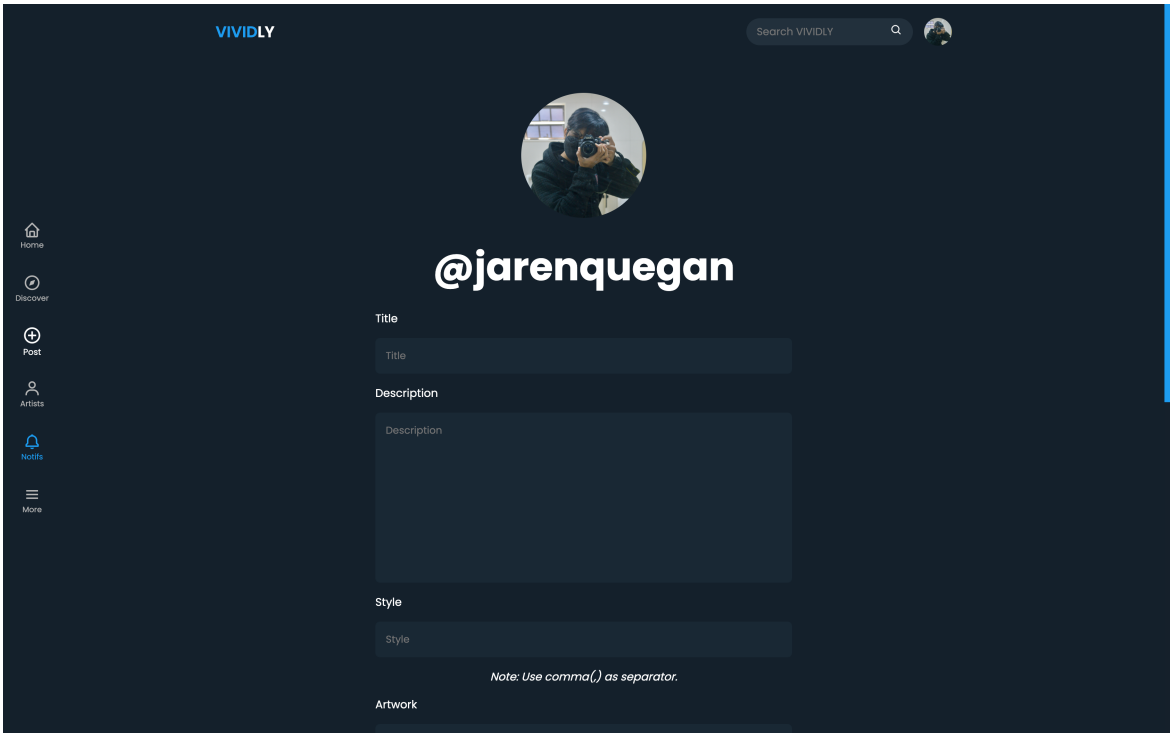
Search Results



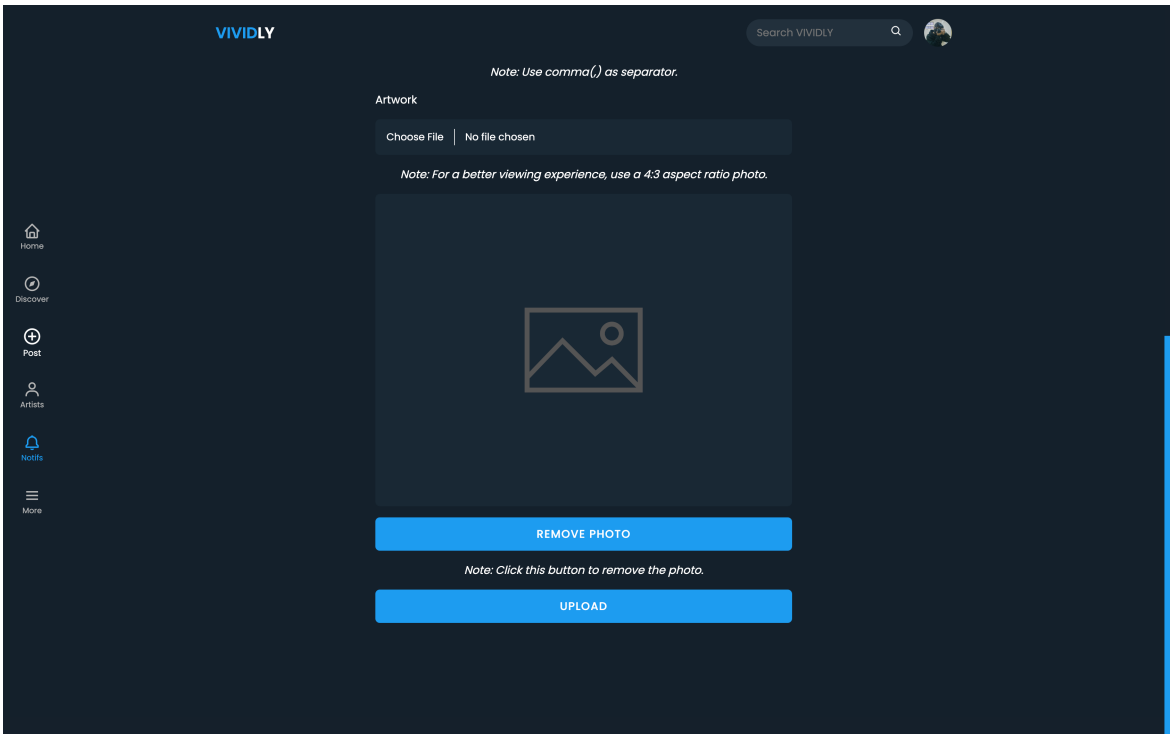
Discover Page



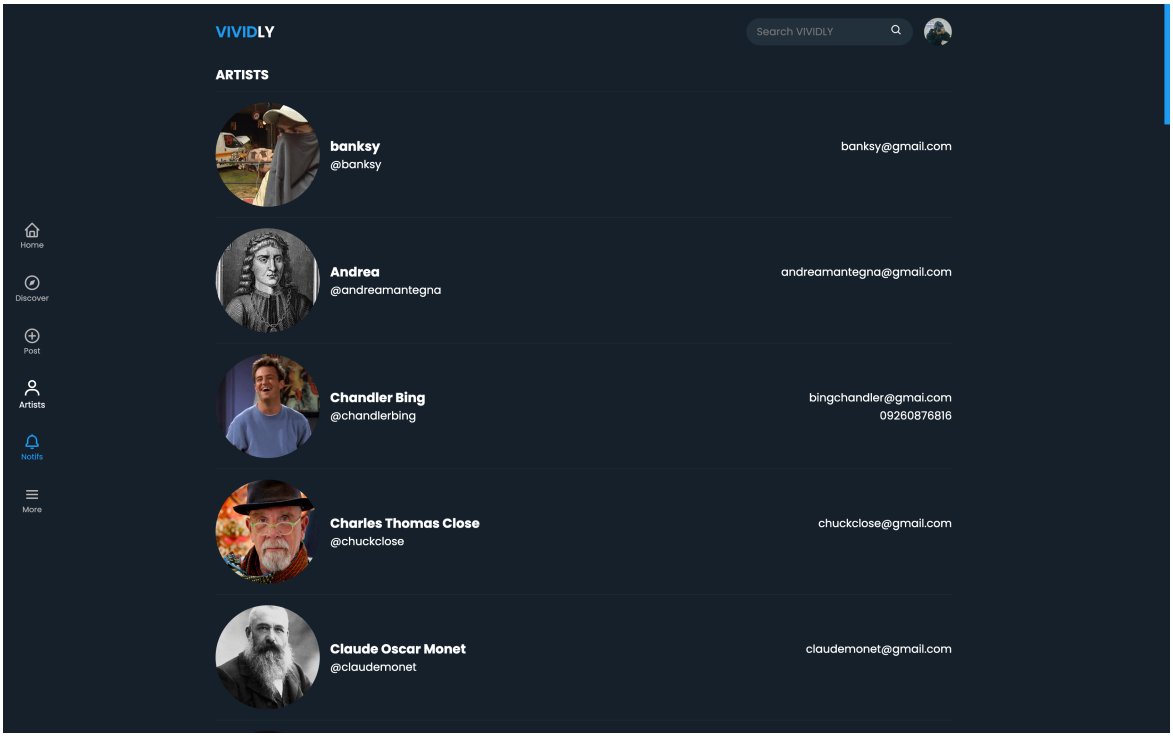
Artwork Posting



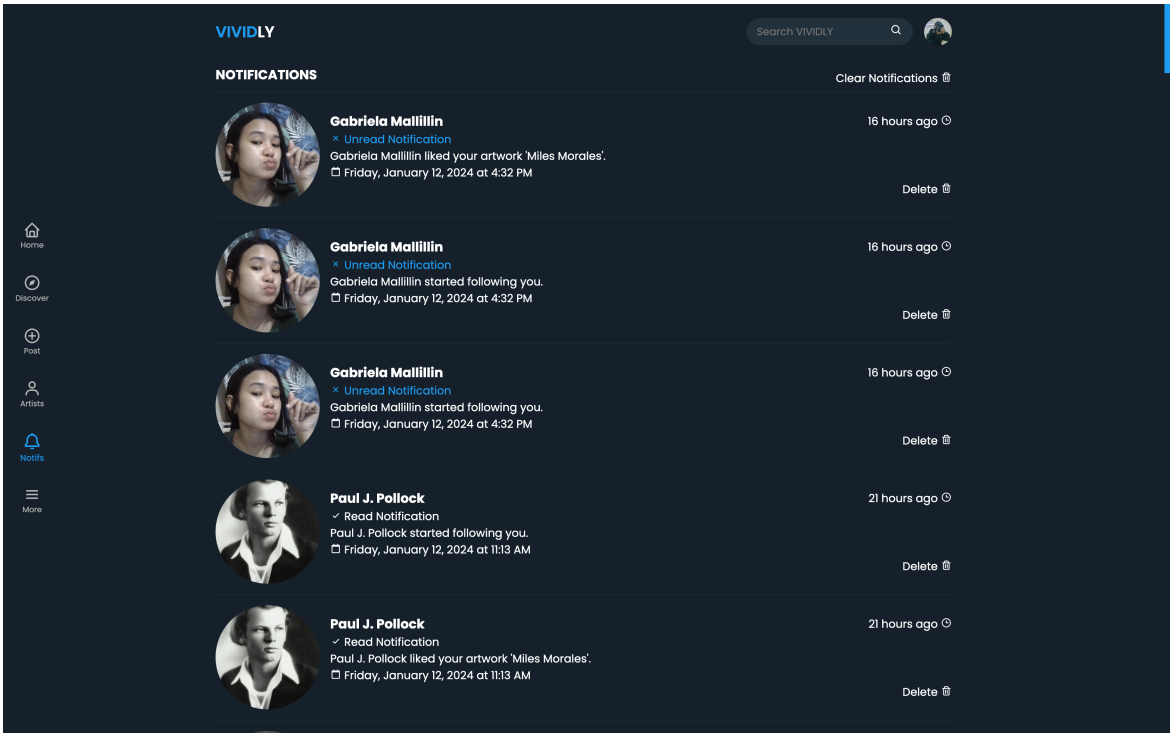
Continuation…



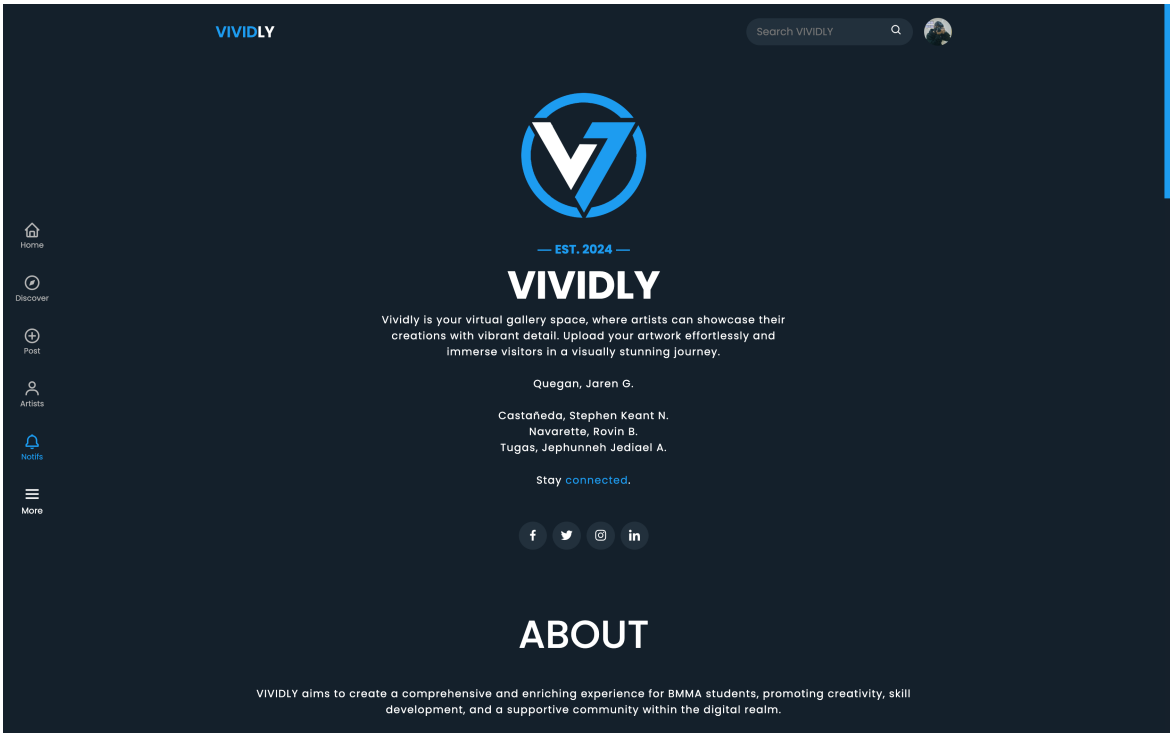
Artists Page



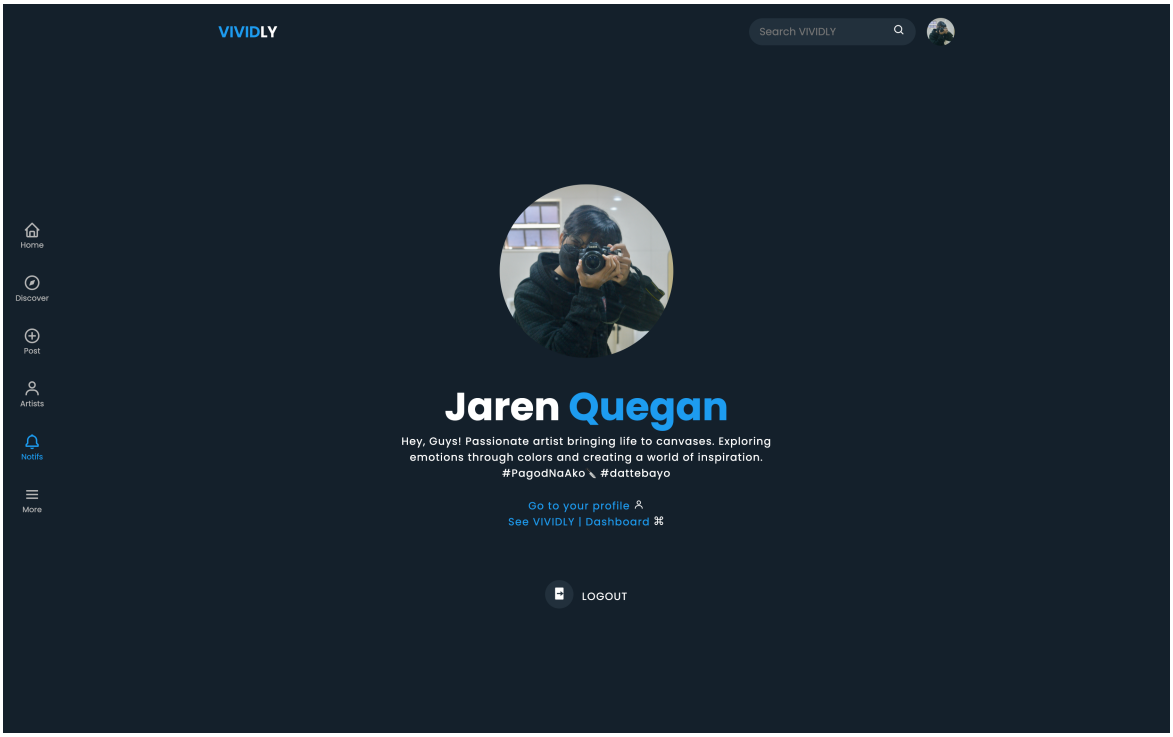
Notifications



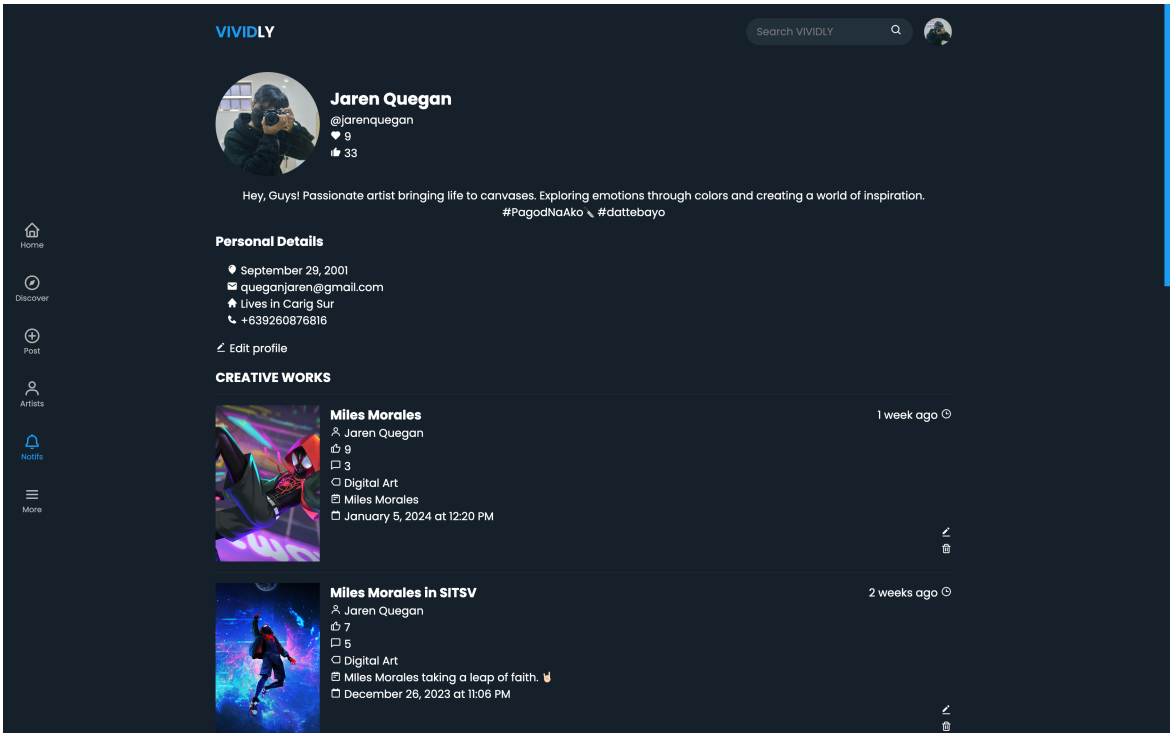
More



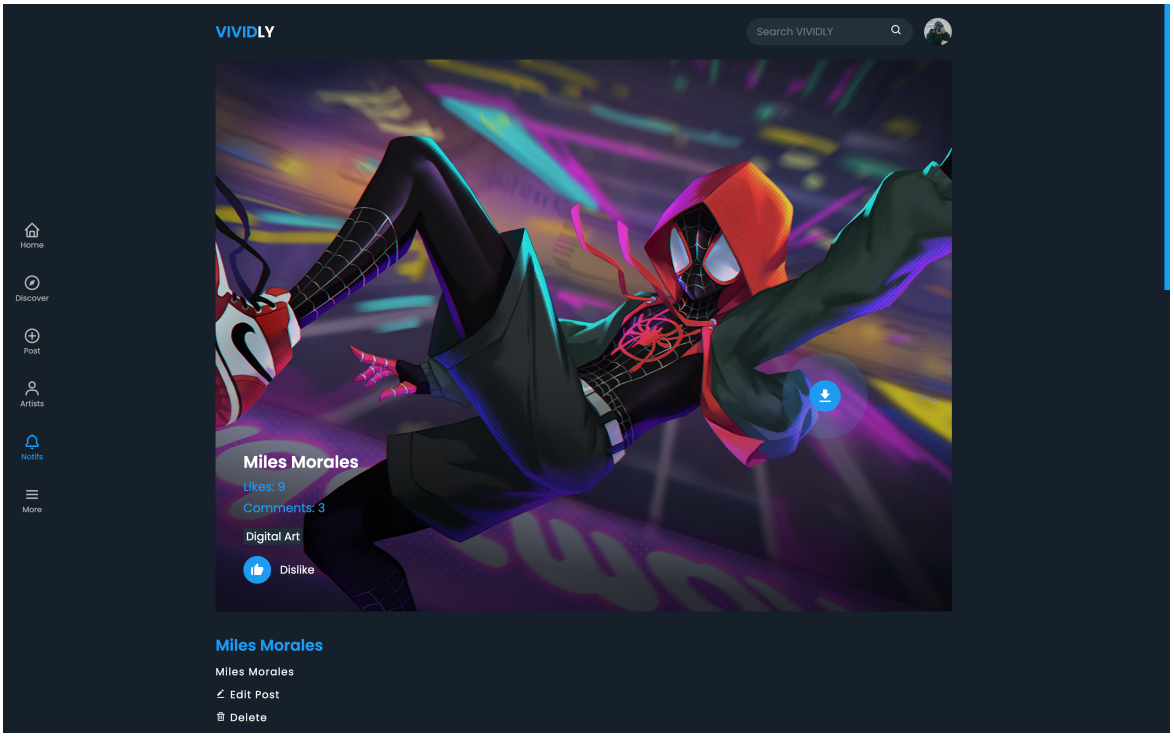
Account Center



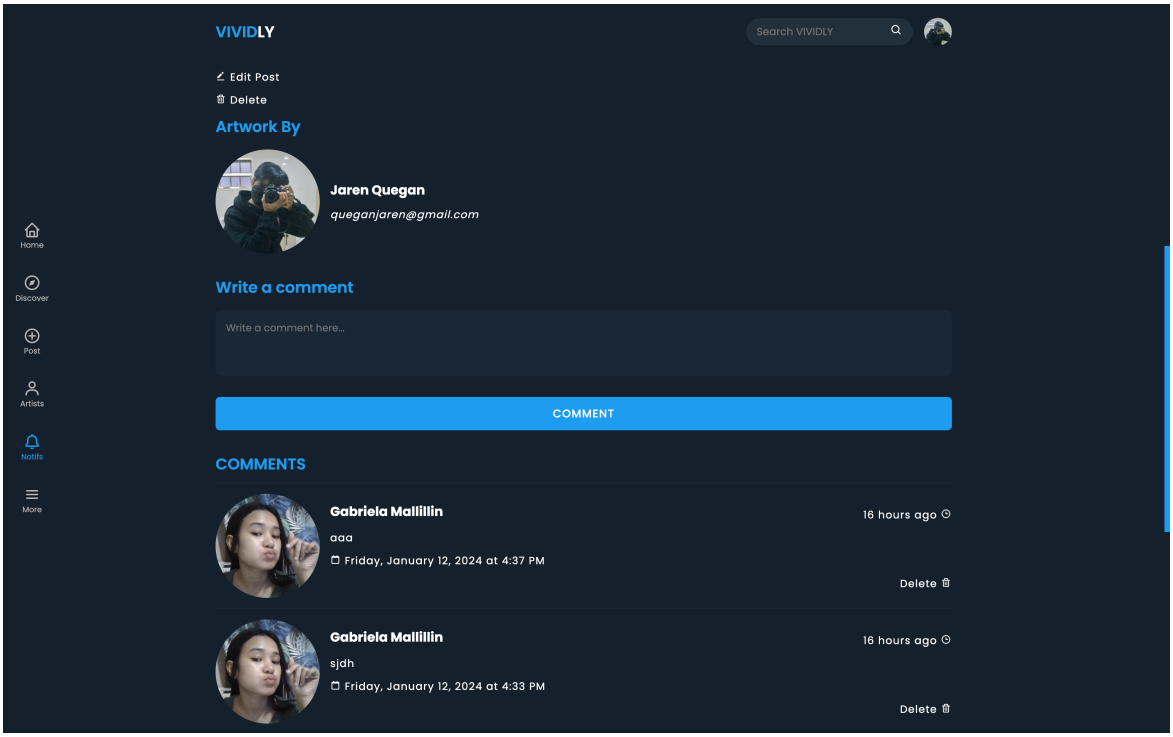
Profile / Portfolio



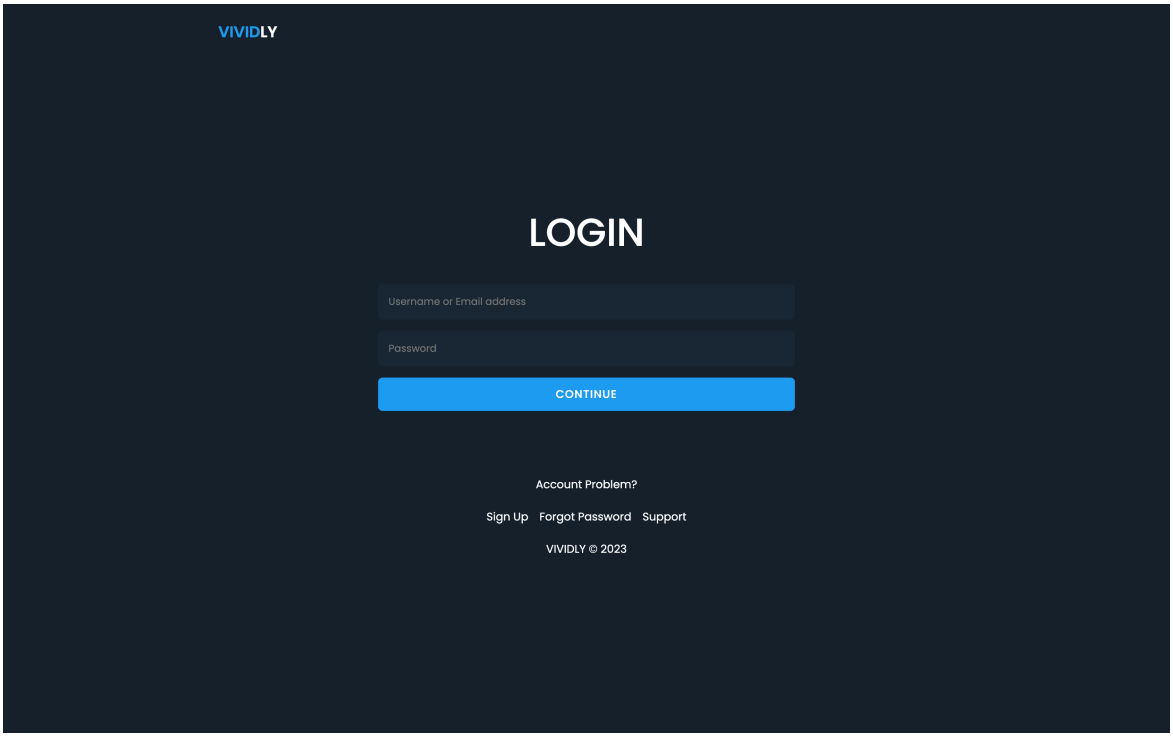
Previewing Arts



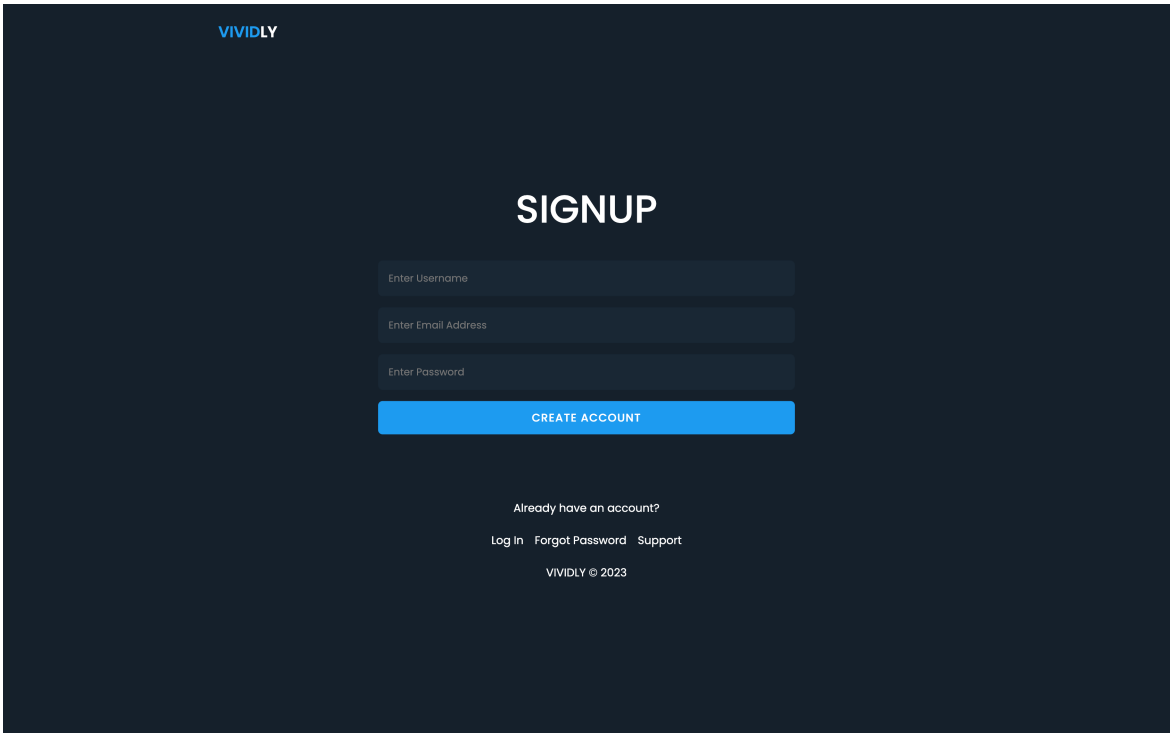
Comment Section



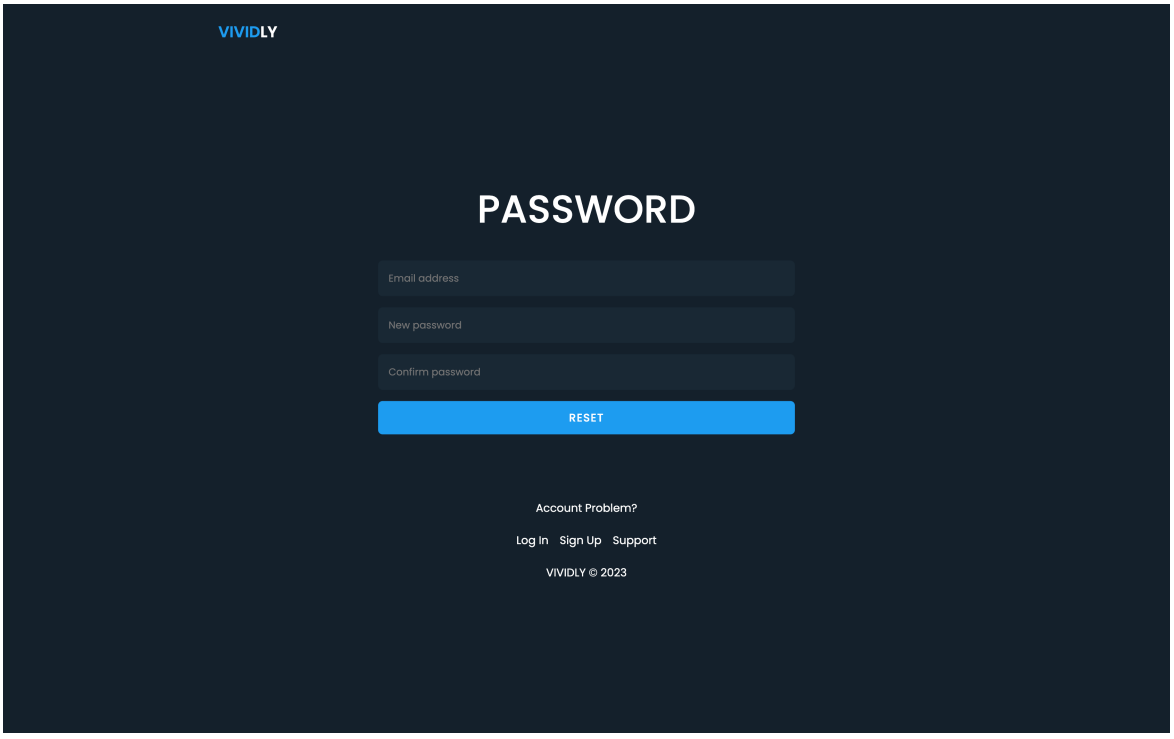
User Login



User Signup

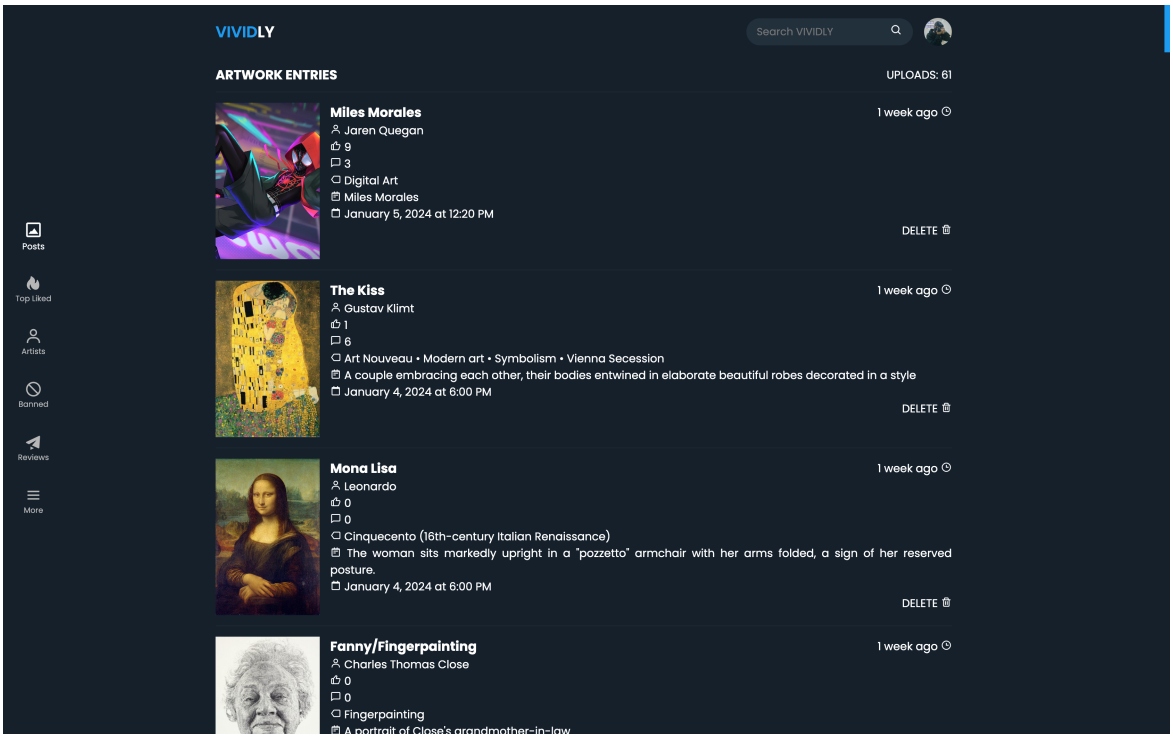


User Password Reset

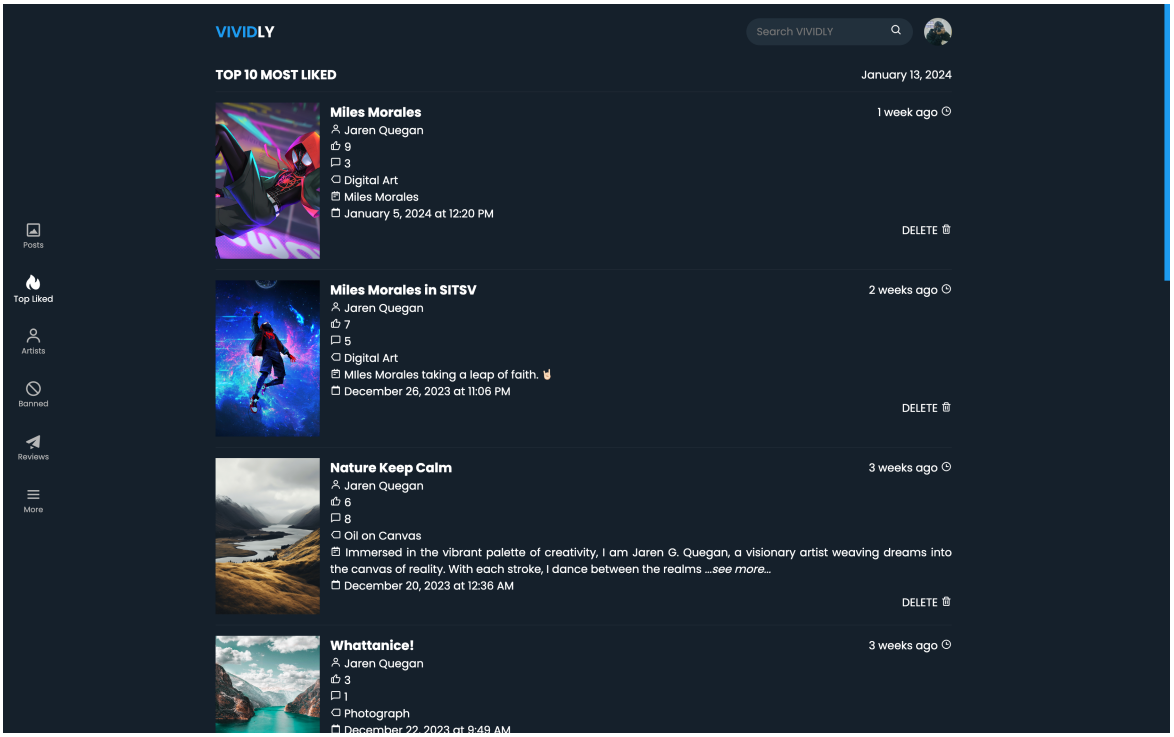


Admin

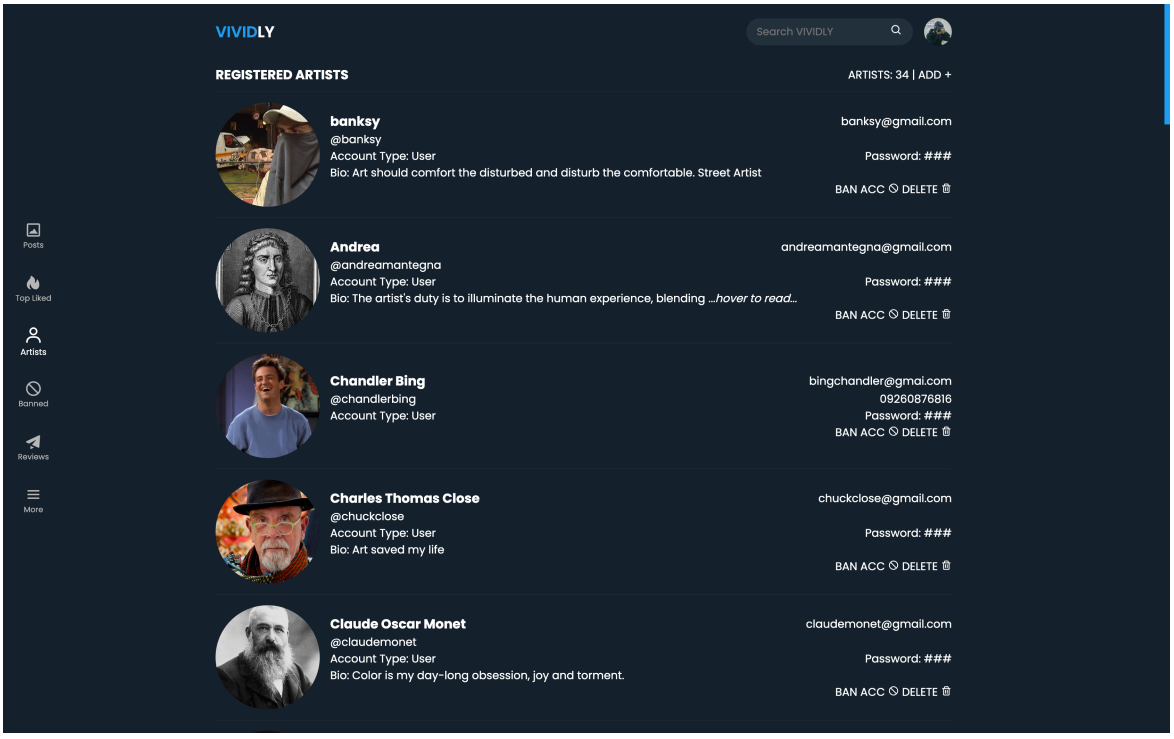
Posts


VIVIDLY
Search VIVIDLY
ARTWORK ENTRIES
UPLOADS: 61
Posts
Top Liked
Artists
Banned
Reviews
More
Miles Morales
1 week ago
Jaren Quegan
9
3
Digital Art
Miles Morales
January 5, 2024 at 12:20 PM
DELETE
The Kiss
1 week ago
Gustav Klimt
1
6
Art Nouveau • Modern art • Symbolism • Vienna Secession
A couple embracing each other, their bodies entwined in elaborate beautiful robes decorated in a style
January 4, 2024 at 6:00 PM
DELETE
Mona Lisa
1 week ago
Leonardo
0
0
Cinquecento (16th-century Italian Renaissance)
The woman sits markedly upright in a "pozzetto" armchair with her arms folded, a sign of her reserved posture.
January 4, 2024 at 6:00 PM
DELETE
Fanny/Fingerpainting
1 week ago
Charles Thomas Close
0
0
Fingerpainting

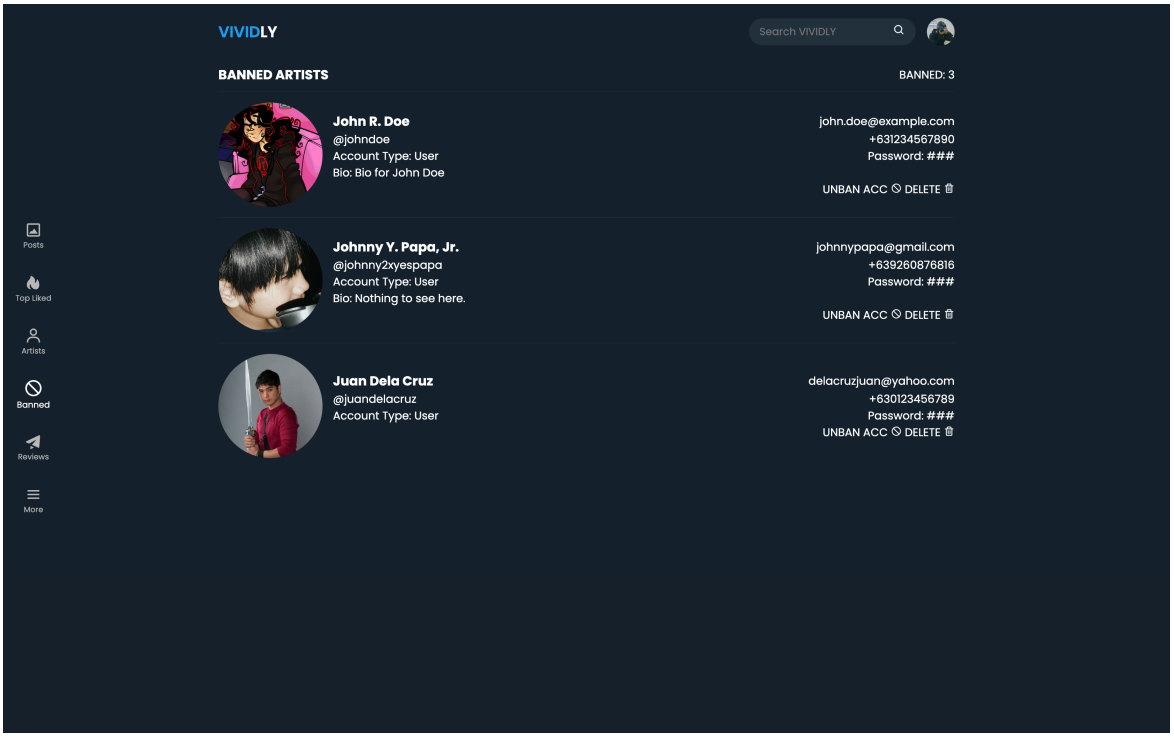
Top Liked



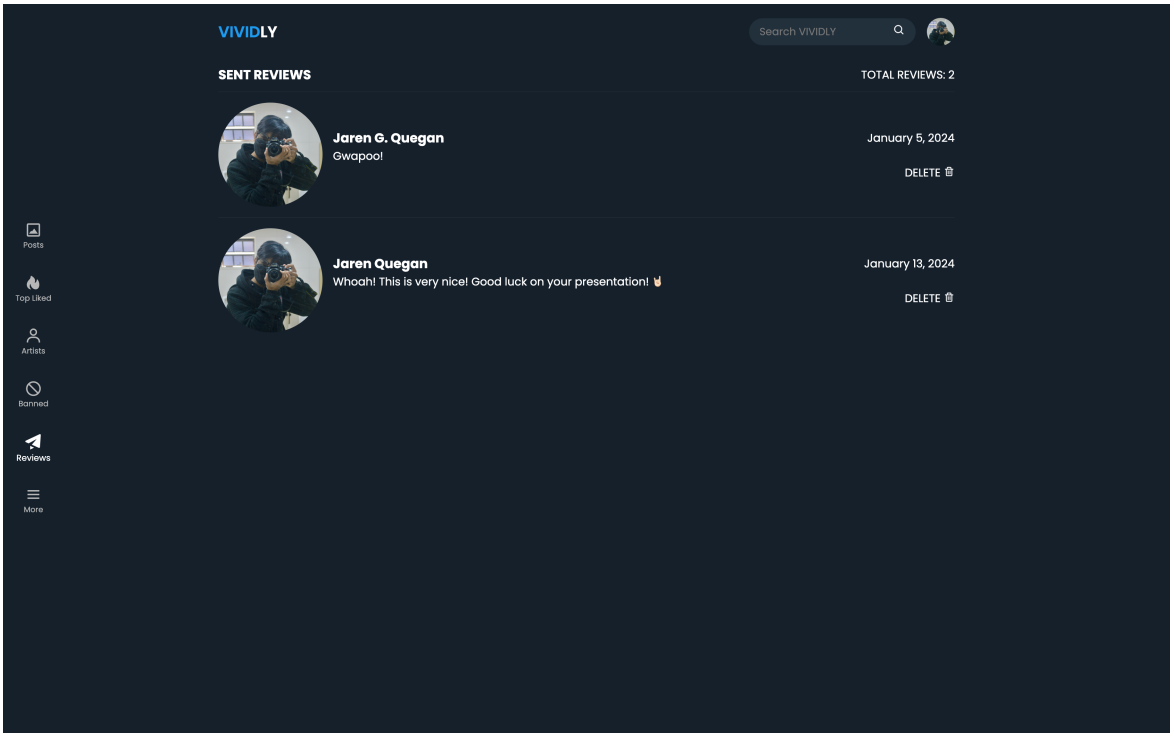
Artists



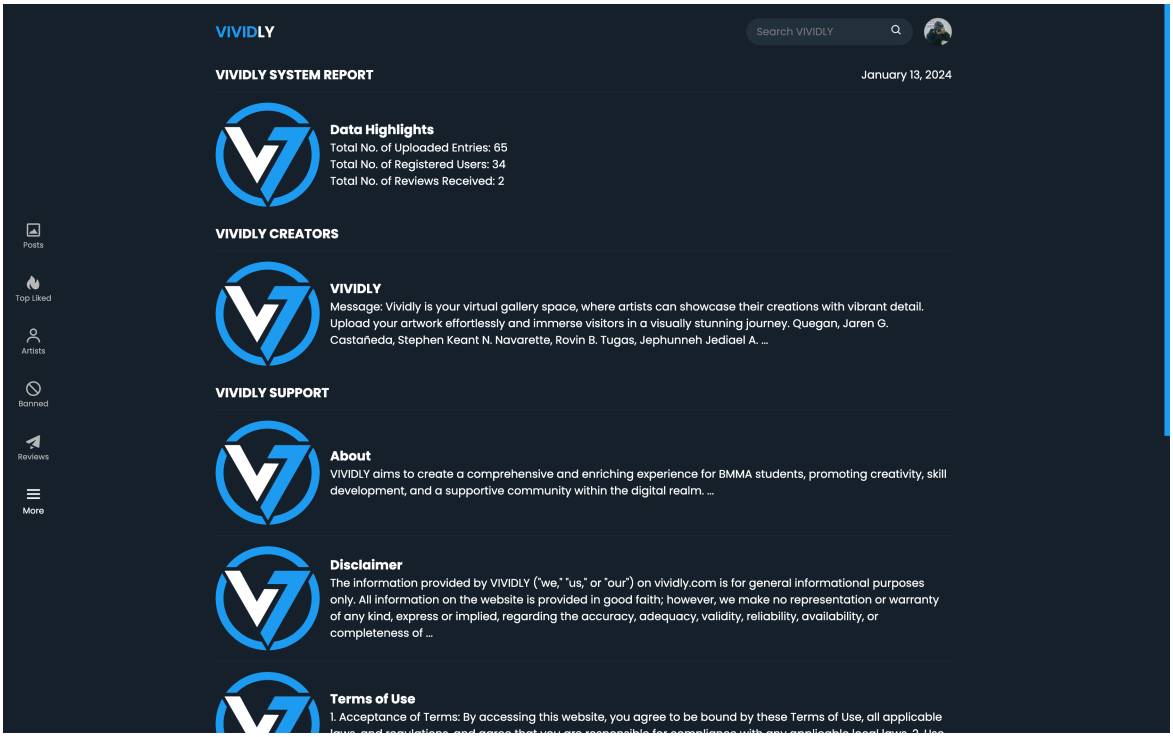
Banned Account



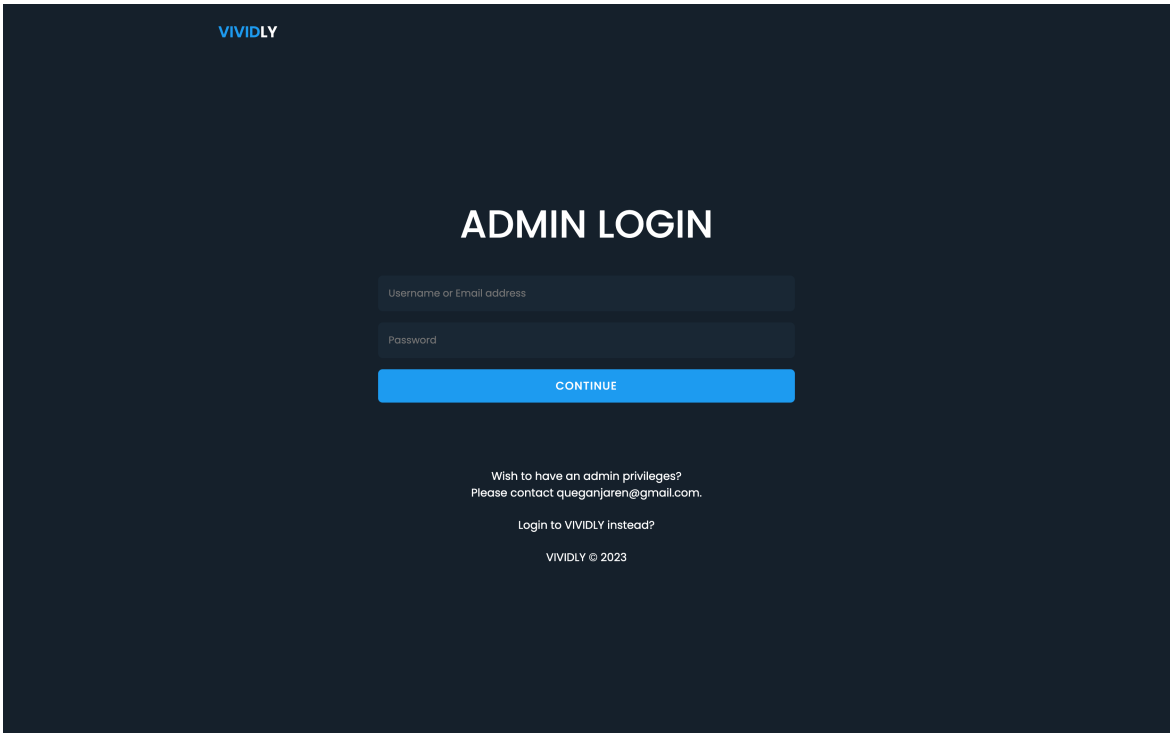
Reviews



More



Admin Login



VIVIDLY PSEUDOCODE

Pseudocode for Sign Up

```
// Initialize variables
$error = ""
$username = ""
$password = ""
$email = ""

// Add User
if (POST request is received with key 'addUser') {
    // Extract user data from the form
    username = value of 'uname' field in the form
    password = value of 'pswd' field in the form
    email = value of 'email' field in the form
    newFilename = "user_default.png"

    // Check if username or email is already taken
    query = "SELECT * FROM artists WHERE username = :uname OR
emailaddress = :email"
    stmt = prepare(query)
    execute stmt with parameters [':uname' => username, ':email' => email]
    result = fetch all rows from stmt

    isUsernameTaken = false
    isEmailTaken = false

    // Iterate through the results to check for existing username or email
    for each row in result {
        if (row's username is equal to username) {
            isUsernameTaken = true
        }
        if (row's email is equal to email) {
            isEmailTaken = true
        }
    }

    // Handle cases where username or email is already taken
```

```
    if (isUsernameTaken and isEmailTaken) {
        error = "Username and email are already taken. Please choose different
ones."
    } elseif (isUsernameTaken) {
        error = "Username is already taken. Please choose a different one."
    } elseif (isEmailTaken) {
        error = "Email is already taken. Please choose a different one."
    } else {
        // Username and email are not taken, proceed with adding the user
        sql = "INSERT INTO artists (username, password, emailaddress, artist_pic)
VALUES (:uname, :pwd, :email, :artist_pic)"
        stmt = prepare(sql)
        execute stmt with parameters [':uname' => username, ':pwd' => password,
':email' => email, ':artist_pic' => newFilename]

        // Retrieve the added user for session setup
        sql = "SELECT * FROM artists WHERE username = :username"
        stmt = prepare(sql)
        bind parameter ':username' with value username
        execute stmt
        artist = fetch a single row from stmt

        // Set up session for the added user
        SESSION['username'] = artist's username

        // Redirect to a confirmation page
        redirect to "sign_up2.php"
        exit
    }
}
```

Pseudocode for Sign Up 2

```
// Initialize variables
firstname = ""
lastname = ""
error = ""

// Check if the signup form is submitted
```

```
if (HTTP request method is POST and 'addUser' is set in the form) {
    // Retrieve the entered first name and last name
    firstname = value of 'firstname' field in the form
    lastname = value of 'lastname' field in the form

    // Handle uploaded avatar
    if (file upload error is 0) {
        filename = name of the uploaded file
        extension = extract file extension from filename
        newFilename = concatenate session username, ".", and extension
        tempname = temporary name of the uploaded file
        folder = concatenate "./images/" and newFilename

        // Move uploaded avatar to folder
        move uploaded file from tempname to folder
    } else {
        // Use a default photo if no image is uploaded
        newFilename = "user_default.png"
    }

    // Update user's first name, last name, and avatar in the database
    sql = "UPDATE artists SET firstname = :firstname, lastname = :lastname,
artist_pic = :pic WHERE username = :username"
    stmt = prepare sql
    execute stmt with parameters [':firstname' => firstname, ':lastname' =>
lastname, ':pic' => newFilename, ':username' => session username]

    // Redirect to the login page
    redirect to "success_page.php?success=true&registered=true"
    exit
} elseif (HTTP request method is POST and 'addUser' is not set in the form) {
    // Redirect back to the signup page
    redirect to "sign_up.php"
    exit
}
```

Pseudocode for Logging In

```
// Initialize variables
```

```
username = ""
emailaddress = ""
usernameOrEmail = ""
password = ""
error = ""

// Check if the login form is submitted
if (HTTP request method is POST) {
    // Retrieve the entered username or email address and password
    usernameOrEmail = value of 'username_or_email' field in the form
    password = value of 'password' field in the form

    // Query the database to check if the user exists and is not banned
    sql = "SELECT * FROM artists WHERE (username = :username OR
emailaddress = :emailaddress) AND password = :password"
    stmt = prepare sql
    bind parameter ':username' with value usernameOrEmail
    bind parameter ':emailaddress' with value usernameOrEmail
    bind parameter ':password' with value password
    execute stmt
    artist = fetch a single row from stmt

    if (artist exists) {
        if (artist's 'is_banned' is 0) {
            // Set session variables
            regenerate session ID
            SESSION['artist_id'] = artist's 'artist_id'
            SESSION['username'] = artist's 'username'
            SESSION['password'] = password
            SESSION['artist_pic'] = artist's 'artist_pic'
            SESSION['emailaddress'] = artist's 'emailaddress'
            SESSION['firstname'] = artist's 'firstname'
            SESSION['lastname'] = artist's 'lastname'
            SESSION['bio'] = artist's 'bio'

            if (artist's 'acct_type' is 'Admin') {
                // Redirect to admin page
                redirect to "whereTo.php"
                exit
```

```
            } else {
                // Redirect to user page
                redirect to "index.php"
                exit
            }
        } else {
            // User is banned, display error message
            error = "Account banned. Contact the admin at queganjaren@gmail.com."
        }
    } else {
        // User not found, display error message
        error = "Invalid username or password."
    }
}
```

Pseudocode for Password Reset

```
// Initialize variables
error = ""
success = ""

// Check if the password reset form is submitted
if (HTTP request method is POST) {
    // Retrieve the entered email address and new password
    email = value of 'email' field in the form
    newPassword = value of 'new_password' field in the form
    confirmPassword = value of 'confirm_password' field in the form

    // Check if the new password and confirm password match
    if (newPassword is equal to confirmPassword) {
        // Check if the user exists in the database
        sql = "SELECT * FROM artists WHERE emailaddress = :email"
        stmt = prepare sql
        execute stmt with parameters [':email' => email]
        artist = fetch a single row from stmt

        if (artist exists) {
            // Update the user's password in the database
```

```
        updateSql = "UPDATE artists SET password = :password WHERE
emailaddress = :email"
        updateStmt = prepare updateSql
        execute updateStmt with parameters [':password' => newPassword,
':email' => email]

        // Set the success message
        success = "Password reset successfully. Redirecting to the login page.
Please wait..."

        // Redirect to login page after 2 seconds using JavaScript
        echo "<script>setTimeout(function() {
            window.location.href = 'login.php';
        }, 2000);</script>"
    } else {
        // User doesn't exist, display an error message
        error = "User does not exist."
    }
} else {
    // Passwords don't match, display an error message
    error = "Passwords do not match."
}
```

Pseudocode for Redirecting the User to the Login Page If Not Logged In

```
// Check if session variables are not set for username, emailaddress, and
password
if (session variable 'username' is not set OR session variable 'emailaddress' is not
set AND session variable 'password' is not set) {
    // Redirect to the login page
    header("Location: login.php");
    exit;
}
```

Pseudocode for Uploading An Artwork

```
// Check if the form for inserting an entry is submitted
if (POST request contains key 'insertEntry') {
    // Get form data
```

```
    title = value of 'title' field in the form
    description = value of 'description' field in the form
    style = value of 'style' field in the form
    artistId = value of 'artistId' field in the form

    // File upload
    artwork = value of 'artwork' field in the form

    // Extract information from the uploaded artwork file
    artwork_filename = name of the uploaded artwork file
    artwork_tmp_path = temporary path of the uploaded artwork file
    artwork_destination = concatenate "artworks/" and artwork_filename
    move uploaded file from artwork_tmp_path to artwork_destination

    // Insert into 'artworks' table
    stmt = prepare "INSERT INTO artworks (title, description, style, image_url)
VALUES (?, ?, ?, ?)"
    execute stmt with parameters [title, description, style, artwork]

    // Get the last inserted artwork ID
    artworkId = last inserted ID in the 'artworks' table

    // Insert into 'artists_artworks' table
    stmt = prepare "INSERT INTO artists_artworks (artist_id, artwork_id) VALUES
(?, ?)"
    execute stmt with parameters [artistId, artworkId]

    // Redirect to success page or display a success message
    redirect to "success_page.php?success=true&post=true"
    exit
}
```

Pseudocode for Editing Uploaded Artworks

```
// Check if 'id' is set in the query parameters
if (GET request contains key 'id') {
    // Retrieve the artwork details for the given artwork_id
    artwork_id = value of 'id' in the query parameters
    sql = "SELECT * FROM artworks WHERE artwork_id = :artwork_id"
```

```
    stmt = prepare sql
    bind parameter ':artwork_id' with value artwork_id
    execute stmt
    artwork = fetch a single row from stmt

    // Check if the form for editing an entry is submitted
    if (POST request contains key 'editEntry') {
        // Get form data
        artworkId = value of 'artworkId' field in the form
        title = value of 'title' field in the form
        description = value of 'description' field in the form
        style = value of 'style' field in the form
        artistId = value of 'artistId' field in the form

        // Check if a new artwork file is uploaded
        if (size of 'artwork' file in the form is greater than 0) {
            // User has uploaded a new artwork
            artwork_filename = name of the uploaded artwork file
            artwork_tmp_path = temporary path of the uploaded artwork file
            artwork_destination = concatenate "artworks/" and artwork_filename
            move uploaded file from artwork_tmp_path to artwork_destination

            // Update 'artworks' table with new image URL
            stmt = prepare "UPDATE artworks SET title = ?, description = ?, style = ?,
image_url = ? WHERE artwork_id = ?"
            execute stmt with parameters [title, description, style, artwork_filename,
artworkId]
        } else {
            // User did not upload a new artwork, retain the existing image URL
            stmt = prepare "UPDATE artworks SET title = ?, description = ?, style = ?
WHERE artwork_id = ?"
            execute stmt with parameters [title, description, style, artworkId]
        }

        // Update 'artists_artworks' table
        stmt = prepare "UPDATE artists_artworks SET artist_id = ? WHERE
artwork_id = ?"
        execute stmt with parameters [artistId, artworkId]
```

```
        // Redirect to success page or display a success message
        header("Location: success_page.php?success=true&editpost=true")
        exit
    }
}
```

Pseudocode for Updating Profile

```
// Check if the update profile form is submitted
if (POST request contains key 'updateProfile') {
    // Get the submitted form data
    newUsername = value of 'username' field in the form
    newFirstname = value of 'firstname' field in the form
    newMiddlename = value of 'middlename' field in the form
    newLastname = value of 'lastname' field in the form
    newSuffix = value of 'suffix' field in the form
    newEmail = value of 'email' field in the form
    newBirthdate = value of 'birthdate' field in the form
    newAddress = value of 'address' field in the form
    newPhoneNumber = value of 'phone_number' field in the form
    newPassword = value of 'password' field in the form
    newAvatar = value of 'defaultImage' field in the form
    newBio = value of 'bio' field in the form

    // Check if a new avatar is uploaded
    if (name of 'uPic' file in the form is not empty) {
        // Handle the uploaded avatar
        newUserPic = name of 'uPic' file in the form
        uploadPath = concatenate "images/" and newUserPic
        move uploaded file from temporary path of 'uPic' to uploadPath
    } else {
        // No new avatar uploaded, check if the remove button is clicked
        if (isset 'defaultImage' in the form and 'defaultImage' field is not empty) {
            // User wants to remove the avatar, use the default image
            newUserPic = 'user_default.png'
        } else {
            // Keep the existing avatar
            newUserPic = value of 'artist_pic' from session
        }
```

```
    }

    // Check if username or email is already taken
    query = "SELECT * FROM artists WHERE (username = :newUsername OR
emailaddress = :newEmail) AND artist_id != :artist_id"
    stmt = prepare query
    execute stmt with parameters [':newUsername' => newUsername, ':newEmail'
=> newEmail, ':artist_id' => artist_id]
    result = fetch all rows from stmt

    isUsernameTaken = false
    isEmailTaken = false

    // Iterate through the results to check for existing username or email
    for each row in result {
        if (row's username is equal to newUsername) {
            isUsernameTaken = true
        }
        if (row's emailaddress is equal to newEmail) {
            isEmailTaken = true
        }
    }

    if (isUsernameTaken and isEmailTaken) {
        // Both username and email are taken
        error = "Username and email are already taken. Please choose different
ones."
    } elseif (isUsernameTaken) {
        // Username is taken
        error = "Username is already taken. Please choose a different one."
    } elseif (isEmailTaken) {
        // Email is taken
        error = "Email is already taken. Please choose a different one."
    } else {
        // Update the user's profile in the database
        sql = "UPDATE artists SET
                username = :username,
                firstname = :firstname,
                middlename = :middlename,
```

```
            lastname = :lastname,
            suffix = :suffix,
            emailaddress = :email,
            birthdate = :birthdate,
            address = :address,
            phone_number = :phone_number,
            password = :password,
            bio = :bio,
            artist_pic = :userPic
            WHERE artist_id = :artist_id"

        stmt = prepare sql
        execute stmt with parameters [
            ':username' => newUsername,
            ':firstname' => newFirstname,
            ':middlename' => newMiddlename,
            ':lastname' => newLastname,
            ':suffix' => newSuffix,
            ':email' => newEmail,
            ':birthdate' => newBirthdate,
            ':address' => newAddress,
            ':phone_number' => newPhoneNumber,
            ':password' => newPassword,
            ':bio' => newBio,
            ':userPic' => newUserPic,
            ':artist_id' => artist_id,
        ]

        // Redirect to the edit page after updating
        header("Location: success_page.php?success=true&updateUser=true")
        exit
    }
}
```

Pseudocode for Inserting Comment with Notification

```
// Check if the comment form is submitted
if (POST request contains key 'comment') {
    // Retrieve user and artwork information
```

```
    artist_id = value of session variable 'artist_id'
    artwork_id = value of 'artworkId' field in the form
    review = value of 'review' field in the form

    // Insert the new comment into the comments table
    sqlAddComment = "INSERT INTO comments (artist_id, artwork_id,
comment_text, created_at) VALUES (:artist_id, :artwork_id, :comment_text,
NOW())"
    stmtAddComment = prepare sqlAddComment
    bind parameters in stmtAddComment [':artist_id' => artist_id, ':artwork_id' =>
artwork_id, ':comment_text' => review]
    execute stmtAddComment

    // Fetch the commented artwork title
    sqlArtworkTitle = "SELECT title FROM artworks WHERE artwork_id =
:artwork_id"
    stmtArtworkTitle = prepare sqlArtworkTitle
    bind parameter ':artwork_id' with value artwork_id
    execute stmtArtworkTitle
    artworkTitle = fetch a single column from stmtArtworkTitle

    // Retrieve the associated artists for the artwork
    sql = "SELECT artists.*
           FROM artists
           INNER JOIN artists_artworks ON artists.artist_id =
artists_artworks.artist_id
           WHERE artists_artworks.artwork_id = :artwork_id"
    stmt = prepare sql
    bind parameter ':artwork_id' with value artwork_id
    execute stmt
    associatedArtists = fetch all rows from stmt

    // Fetch artist name of the follower
    sqlCommenterName = "SELECT * FROM artists WHERE artist_id =
:commenter_id"
    stmtCommenterName = prepare sqlCommenterName
    bind parameter ':commenter_id' with value artist_id
    execute stmtCommenterName
    followerData = fetch a single row from stmtCommenterName
```

```
    // Format the follower name
    if (followerData has non-empty 'firstname', 'middlename', 'lastname', or 'suffix') {
        middleInitial = get first character of followerData's 'middlename'
        CommenterName = concatenate followerData's 'firstname', middleInitial,
followerData's 'lastname'
        if followerData has non-empty 'suffix', concatenate ", " and followerData's
'suffix' to CommenterName
    } else {
        CommenterName = followerData's 'username'
    }

    // Initialize receiver_id
    receiver_id = null

    // Iterate through associatedArtists to get the receiver_id
    for each artist in associatedArtists {
        receiver_id = artist's 'artist_id'
    }

    // Set commenter_id
    commenter_id = value of session variable 'artist_id'

    // Create notification message
    notificationMessage = "{$CommenterName} commented on your artwork
'{$artworkTitle}'."

    // Check if the commenter_id is not the same as receiver_id to avoid self-
notification
    if commenter_id is not equal to receiver_id {
        // Insert a notification into the notifications table
        sqlInsertNotification = "INSERT INTO notifications (sender_id, receiver_id,
artwork_id, notification_type, notification_data)
                                 VALUES (:commenter_id, :artist_id, :artwork_id, 'Like',
:notification_data)"
        stmtInsertNotification = prepare sqlInsertNotification
        bind parameters in stmtInsertNotification [
            ':commenter_id' => commenter_id,
            ':artist_id' => receiver_id,
```

```
            ':artwork_id' => artwork_id,
            ':notification_data' => notificationMessage
        ]
        execute stmtInsertNotification
    }
}
```

Pseudocode for Liking and Disliking an Artwork with Notification

```
// Check if the like/dislike form is submitted
if (POST request contains keys 'artworkId' and 'isFavorite') {
    // Retrieve user and artwork information
    artist_id = value of session variable 'artist_id'
    artwork_id = value of 'artworkId' field in the form
    isFavorite = value of 'isFavorite' field in the form

    // Fetch the artwork title
    sqlArtworkTitle = "SELECT title FROM artworks WHERE artwork_id =
:artwork_id"
    stmtArtworkTitle = prepare sqlArtworkTitle
    bind parameter ':artwork_id' with value artwork_id
    execute stmtArtworkTitle
    artworkTitle = fetch a single column from stmtArtworkTitle

    // Retrieve the associated artists for the artwork
    sql = "SELECT artists.*
        FROM artists
        INNER JOIN artists_artworks ON artists.artist_id = artists_artworks.artist_id
        WHERE artists_artworks.artwork_id = :artwork_id"
    stmt = prepare sql
    bind parameter ':artwork_id' with value artwork_id
    execute stmt
    associatedArtists = fetch all rows from stmt

    // Fetch the artist name of the follower
    sqlLikerName = "SELECT * FROM artists WHERE artist_id = :liker_id"
    stmtLikerName = prepare sqlLikerName
    bind parameter ':liker_id' with value artist_id
    execute stmtLikerName
```

```
    followerData = fetch a single row from stmtLikerName

    // Format the follower name and store it in a variable
    if followerData has non-empty 'firstname', 'middlename', 'lastname', or 'suffix' {
        middleInitial = get first character of followerData's 'middlename'
        LikerName = concatenate followerData's 'firstname', middleInitial,
followerData's 'lastname'
        if followerData has non-empty 'suffix', concatenate ", " and followerData's
'suffix' to LikerName
    } else {
        LikerName = followerData's 'username'
    }

    // Check if the artist has already liked the artwork
    sqlCheckLiked = "SELECT COUNT(*) AS liked_count
                    FROM artist_liked_artworks
                    WHERE artist_id = :artist_id AND artwork_id = :artwork_id"
    stmtCheckLiked = prepare sqlCheckLiked
    bind parameters in stmtCheckLiked [':artist_id' => artist_id, ':artwork_id' =>
artwork_id]
    execute stmtCheckLiked
    likedCount = fetch a single column from stmtCheckLiked

    // Check if the artist is disliking the artwork
    if likedCount > 0 {
        // Artist has already liked the artwork, so dislike it
        sqlDislike = "DELETE FROM artist_liked_artworks
                      WHERE artist_id = :artist_id AND artwork_id = :artwork_id"
        stmtDislike = prepare sqlDislike
        bind parameters in stmtDislike [':artist_id' => artist_id, ':artwork_id' =>
artwork_id]
        execute stmtDislike
        isFavorite = false
    } else {
        // Artist has not liked the artwork, so like it
        sqlLike = "INSERT INTO artist_liked_artworks (artist_id, artwork_id)
                   VALUES (:artist_id, :artwork_id)"
        stmtLike = prepare sqlLike
```

```
        bind parameters in stmtLike [':artist_id' => artist_id, ':artwork_id' =>
artwork_id]
        execute stmtLike
        isFavorite = true

        // Send like notification
        notificationMessage = "{$LikerName} liked your artwork '{$artworkTitle}'."

        // Initialize receiver_id
        receiver_id = null

        // Iterate through associatedArtists to get the receiver_id
        for each artist in associatedArtists {
            receiver_id = artist's 'artist_id'
        }

        // Set liker_id
        liker_id = value of session variable 'artist_id'

        // Check if the liker_id is not the same as receiver_id to avoid self-notification
        if liker_id is not equal to receiver_id {
            // Insert a notification into the notifications table
            sqlInsertNotification = "INSERT INTO notifications (sender_id, receiver_id,
artwork_id, notification_type, notification_data)
                                    VALUES (:liker_id, :artist_id, :artwork_id, 'Like',
:notification_data)"
            stmtInsertNotification = prepare sqlInsertNotification
            bind parameters in stmtInsertNotification [
                ':liker_id' => liker_id,
                ':artist_id' => receiver_id,
                ':artwork_id' => artwork_id,
                ':notification_data' => notificationMessage
            ]
            execute stmtInsertNotification
        }
    }

    // Fetch the updated total_likes
    sqlTotalLikes = "SELECT COUNT(*) AS total_likes
```

```
                FROM artist_liked_artworks
                WHERE artwork_id = :artwork_id"
    stmtTotalLikes = prepare sqlTotalLikes
    bind parameter ':artwork_id' with value artwork_id
    execute stmtTotalLikes
    result = fetch a single row from stmtTotalLikes

    // Return the updated total_likes and isFavorite status as JSON
    header('Content-Type: application/json')
    echo json_encode(['status' => 'success', 'total_likes' => result['total_likes'],
'isFavorite' => isFavorite])
    exit()
}
```

Pseudocode for Following and Unfollowing with Notification

```
// Check if the follow/unfollow form is submitted
if (POST request contains keys 'artistId' and 'isFollowing') {
    // Retrieve user and artist information
    follower_id = value of session variable 'artist_id'
    artist_id = value of 'artistId' field in the form
    isFollowing = value of 'isFollowing' field in the form

    // Fetch the artist name of the follower
    sqlFollowerName = "SELECT username, firstname, middlename, lastname,
suffix FROM artists WHERE artist_id = :follower_id"
    stmtFollowerName = prepare sqlFollowerName
    bind parameter ':follower_id' with value follower_id
    execute stmtFollowerName
    followerData = fetch a single row from stmtFollowerName

    // Format the follower name and store it in a variable
    if followerData has non-empty 'firstname', 'middlename', 'lastname', or 'suffix' {
        middleInitial = get first character of followerData's 'middlename'
        followerName = concatenate followerData's 'firstname', middleInitial,
followerData's 'lastname'
        if followerData has non-empty 'suffix', concatenate ", " and followerData's
'suffix' to followerName
    } else {
```

```
    followerName = followerData's 'username'
  }

  // Check if the follower is already following the artist
  sqlCheckFollowing = "SELECT COUNT(*) AS following_count
                       FROM artist_followers
                       WHERE follower_id = :follower_id AND artist_id = :artist_id"
  stmtCheckFollowing = prepare sqlCheckFollowing
  bind parameters in stmtCheckFollowing [':follower_id' => follower_id, ':artist_id'
=> artist_id]
  execute stmtCheckFollowing
  followingCount = fetch a single column from stmtCheckFollowing

  // Check if the follower is unfollowing the artist
  if followingCount > 0 {
    // Follower is already following the artist, so unfollow
    sqlUnfollow = "DELETE FROM artist_followers
                   WHERE follower_id = :follower_id AND artist_id = :artist_id"
    stmtUnfollow = prepare sqlUnfollow
    bind parameters in stmtUnfollow [':follower_id' => follower_id, ':artist_id' =>
artist_id]
    execute stmtUnfollow
    isFollowing = false

    // Send unfollow notification
    notificationMessage = "{$followerName} unfollowed you."
  } else {
    // Follower is not following the artist, so follow
    sqlFollow = "INSERT INTO artist_followers (follower_id, artist_id)
                 VALUES (:follower_id, :artist_id)"
    stmtFollow = prepare sqlFollow
    bind parameters in stmtFollow [':follower_id' => follower_id, ':artist_id' =>
artist_id]
    execute stmtFollow
    isFollowing = true

    // Send follow notification
    notificationMessage = "{$followerName} started following you."
  }
```

```
    // Fetch the updated total_followers
    sqlTotalFollowers = "SELECT COUNT(*) AS total_followers
                         FROM artist_followers
                         WHERE artist_id = :artist_id"
    stmtTotalFollowers = prepare sqlTotalFollowers
    bind parameter ':artist_id' with value artist_id
    execute stmtTotalFollowers
    result = fetch a single row from stmtTotalFollowers

    // Return the updated total_followers and isFollowing status as JSON
    header('Content-Type: application/json')
    echo json_encode(['status' => 'success', 'total_followers' =>
result['total_followers'], 'isFollowing' => isFollowing])

    // Insert notification into the notifications table
    sqlInsertNotification = "INSERT INTO notifications (sender_id, receiver_id,
notification_type, notification_data)
                             VALUES (:follower_id, :artist_id, 'follow', :notification_data)"
    stmtInsertNotification = prepare sqlInsertNotification
    bind parameters in stmtInsertNotification [
        ':follower_id' => follower_id,
        ':artist_id' => artist_id,
        ':notification_data' => notificationMessage
    ]
    execute stmtInsertNotification
    exit()
}
```

Pseudocode for Searching

```
// Initialize variables
searchActor = []
searchResults = []
input = ""

// Search Database
if GET parameter 'search' is set and not empty:
    input = GET parameter 'search'
```

```
    search = '%' + sanitize_and_format(input) + '%'

    // Search Artists
    sql = "SELECT * FROM artists WHERE various_fields_like_search ORDER BY
created_at DESC"
    Execute SQL query with search parameter
    Store results in searchActor

    // Search Artworks
    sql = "SELECT artworks.*, artists.*, COUNT(likes) AS total_likes,
COUNT(comments) AS total_comments FROM artworks
          JOIN artists_artworks ON artworks.artwork_id = artists_artworks.artwork_id
          JOIN artists ON artists_artworks.artist_id = artists.artist_id
          LEFT JOIN artist_liked_artworks ON artworks.artwork_id =
artist_liked_artworks.artwork_id
          LEFT JOIN comments ON artworks.artwork_id = comments.artwork_id
          WHERE various_fields_like_search GROUP BY artworks.artwork_id
ORDER BY artworks.created_at DESC"
    Execute SQL query with search parameter
    Store results in searchResults

// If no search query is provided
else:
    // Fetch all artworks
    sql = "SELECT artworks.artwork_id, artworks.*, artists.artist_id,
artists.username, artists.firstname,
               artists.middlename, artists.lastname, artists.suffix,
               COUNT(DISTINCT artist_liked_artworks.id) AS total_likes,
               COUNT(DISTINCT comments.comment_id) AS total_comments
           FROM artworks
           INNER JOIN artists_artworks ON artworks.artwork_id =
artists_artworks.artwork_id
           INNER JOIN artists ON artists_artworks.artist_id = artists.artist_id
           LEFT JOIN artist_liked_artworks ON artworks.artwork_id =
artist_liked_artworks.artwork_id
           LEFT JOIN comments ON artworks.artwork_id = comments.artwork_id
           GROUP BY artworks.artwork_id
           ORDER BY artworks.created_at DESC"
```

Execute SQL query
Store results in searchResults

Pseudocode for Fetching the Top 10 most Liked Artwork

```
// Define the SQL query
sql = "SELECT
          artworks.*,
          artists.*,
          COUNT(DISTINCT artist_liked_artworks.id) AS total_likes
       FROM artworks
       LEFT JOIN artists_artworks ON artworks.artwork_id =
artists_artworks.artwork_id
       LEFT JOIN artists ON artists.artist_id = artists_artworks.artist_id
       LEFT JOIN artist_liked_artworks ON artworks.artwork_id =
artist_liked_artworks.artwork_id
       GROUP BY artworks.artwork_id, artists.artist_id
       ORDER BY total_likes DESC
       LIMIT 10";

// Prepare the SQL query
stmt = prepare sql

// Execute the SQL query
execute stmt

// Fetch the results
results = fetchAll from stmt

// Return the results as JSON
header('Content-Type: application/json');
echo json_encode(results);
```

Pseudocode for Adding Review

```
// Check if the review form is submitted via POST and the 'review' field is set
if (POST request contains key 'review' and 'artist' is set) {
   // Retrieve review information
   review = value of 'review' field in the form
   artist_id = artist's 'artist_id'
```

```
    username = artist's 'username'
    firstname = artist's 'firstname'
    middlename = artist's 'middlename'
    lastname = artist's 'lastname'
    suffix = artist's 'suffix'
    email = artist's 'emailaddress'
    artist_pic = artist's 'artist_pic'

    // Prepare and execute SQL query to insert the review into the 'reviews' table
    sql = "INSERT INTO reviews (artist_id, username, firstname, middlename,
lastname, suffix, emailaddress, review_content, artist_pic)
        VALUES (:artist_id, :username, :firstname, :middlename, :lastname, :suffix,
:emailaddress, :review_content, :artist_pic)";
    stmt = prepare sql
    bind parameters in stmt [
        ':artist_id' => artist_id,
        ':username' => username,
        ':firstname' => firstname,
        ':middlename' => middlename,
        ':lastname' => lastname,
        ':suffix' => suffix,
        ':emailaddress' => email,
        ':review_content' => review,
        ':artist_pic' => artist_pic
    ]
    execute stmt

    // Set feedback message
    feedback_result = 'Your feedback has been added successfully. Thank you for
taking the time to provide your review!';

    // Redirect to 'more.php' after 2 seconds
    echo "<script>setTimeout(function() {
            window.location.href = 'more.php';
        }, 2000);</script>";
}
```

Pseudocode for Banning and Unbanning a User

```
// Check if artist_id is set in the query parameters
if artist_id is set in $_GET {
    // Check if artist_id and 'unban' parameter are set in the query parameters
    if artist_id and 'unban' parameter are set in $_GET {
        artist_idToEdit = value of 'artist_id' in $_GET
        // Update the user's profile in the database to unban
        sql = "UPDATE artists SET
            is_banned = :is_banned
            WHERE artist_id = :artist_id";
        stmt = $conn->prepare(sql);
        stmt->execute([
            ':is_banned' => 0,
            ':artist_id' => artist_idToEdit,
        ]);
        // Redirect to success page with appropriate message
        header("Location: success_page.php?success=true&ban=true");
        exit;
    }

    // Check if artist_id and 'ban' parameter are set in the query parameters
    if artist_id and 'ban' parameter are set in $_GET {
        artist_idToEdit = value of 'artist_id' in $_GET
        // Update the user's profile in the database to ban
        sql = "UPDATE artists SET
            is_banned = :is_banned
            WHERE artist_id = :artist_id";
        stmt = $conn->prepare(sql);
        stmt->execute([
            ':is_banned' => 1,
            ':artist_id' => artist_idToEdit,
        ]);
        // Redirect to success page with appropriate message
        header("Location: success_page.php?success=true&ban=true");
        exit;
    }
}
```

Pseudocode for Deleting a User

```
// Check if the user confirmed the deletion
if (GET request contains keys 'confirm' and 'artists' and their values are 'true' and
'true' respectively) {
    // Prepare and execute the query to delete the user
    deleteUserId = value of session variable 'artist_id'
    deleteUserQuery = "DELETE FROM artists WHERE artist_id = :artist_id"
    deleteUserStmt = prepare deleteUserQuery
    bind parameter ':artist_id' with deleteUserId
    execute deleteUserStmt

    // Redirect to the success page with a deletion success message
    header("Location: success_page.php?success=true&deleted=true")
    exit
}
```

Pseudocode for Deleting an Artwork

```
// Check if the user confirmed the deletion and provided necessary parameters
if (GET request contains keys 'confirm' with value 'true', 'post' with value 'true', and
'id') {
    // Retrieve the artwork ID from the GET parameters
    artwork_id = value of 'id'

    // Prepare and execute the query to delete the artwork
    deletePostQuery = "DELETE FROM artworks WHERE artwork_id =
:artwork_id"
    deletePostStmt = prepare deletePostQuery
    bind parameter ':artwork_id' with artwork_id
    execute deletePostStmt

    // Redirect to the success page with a deletion success message
    header("Location: success_page.php?success=true&post=true")
    exit
}
```

Pseudocode for Deleting a Comment

```
// Check if the user confirmed the deletion and provided necessary parameters
if (GET request contains keys 'confirm' with value 'true', 'comment' with value
'true', and 'comment_id') {
```

```
    // Retrieve the comment ID from the GET parameters
    comment_id = value of 'comment_id'

    // Prepare and execute the query to delete the comment
    deleteCommentQuery = "DELETE FROM comments WHERE comment_id =
:comment_id"
    deleteCommentStmt = prepare deleteCommentQuery
    bind parameter ':comment_id' with comment_id
    execute deleteCommentStmt

    // Redirect to the success page with a deletion success message
    header("Location: success_page.php?success=true&comment=true")
    exit
}
```

Pseudocode for Deleting a Notification

```
// Check if the user confirmed the deletion and provided necessary parameters
if (GET request contains keys 'confirm' with value 'true', 'notif' with value 'true', and
'notif_id') {
    // Retrieve the notification ID from the GET parameters
    notif_id = value of 'notif_id'

    // Prepare and execute the query to delete the notification
    deleteNotifQuery = "DELETE FROM notifications WHERE notification_id =
:notification_id"
    deleteNotifStmt = prepare deleteNotifQuery
    bind parameter ':notification_id' with notif_id
    execute deleteNotifStmt

    // Redirect to the success page with a deletion success message
    header("Location: success_page.php?success=true&notif=true")
    exit
}
```

Pseudocode for Clearing Notifications

```
// Check if the user confirmed the deletion and provided necessary parameters
if (GET request contains keys 'confirm' with value 'true', 'clear' with value 'true',
and 'artist_id') {
```

```
    // Retrieve the artist ID from the GET parameters
    receiver_id = value of 'artist_id'

    // Prepare and execute the query to delete notifications for the specified artist
    deleteNotifQuery = "DELETE FROM notifications WHERE receiver_id =
:receiver_id"
    deleteNotifStmt = prepare deleteNotifQuery
    bind parameter ':receiver_id' with receiver_id
    execute deleteNotifStmt

    // Redirect to the success page with a deletion success message
    header("Location: success_page.php?success=true&notif=true")
    exit
}
```

Pseudocode for Deleting a Review

```
// Check if 'review_id' is set in the GET parameters and 'delete' is set to 'true'
if (GET parameters contain 'review_id' and 'delete' is 'true') {
    // Retrieve 'review_id' to delete
    review_idToDelete = value of 'review_id' in GET parameters

    // Prepare and execute SQL query to delete the review from the 'reviews' table
    deleteReviewQuery = "DELETE FROM reviews WHERE review_id =
:review_id";
    deleteReviewStmt = prepare deleteReviewQuery
    bind parameters in deleteReviewStmt [':review_id' => review_idToDelete]
    execute deleteReviewStmt

    // Redirect to 'success_page.php' with success and deleteReview parameters
after deletion
    header("Location: success_page.php?success=true&deleteReview=true");
    exit;
}
```

APPENDICES

Appendix A. Exhibits

Exhibit A.

Designing Your Ideal Multimedia Space: Student Survey

Welcome to our Multimedia Arts Student Survey!

We're excited to hear your thoughts and preferences on the development of a new online platform tailored for BMMA students. Your input is invaluable in creating a space that enhances your multimedia experience and showcases your talents effectively.

This short survey will only take a few minutes to complete. Please share your insights on platform features, what makes for an engaging experience, and other aspects of a user-friendly interface. Your feedback will shape the future of our multimedia community.

* Indicates required question

Name: (Optional)
(Family name, Given name M.I.)

Your answer

Continuation…

Year Level: *

- ◯ First Year
- ◯ Second Year
- ◯ Third Year
- ◯ Fourth Year

Questions

Share your preferences and shape our Multimedia Arts Platform! Your insights guide us in creating a platform tailored to your desires.

As an artist or BMMA student, which type of online platform or tool would you find most useful for showcasing your multimedia work? You can choose from the provided options, but if none of the choices apply, please specify. *

- ◯ LikHub is a platform that offers users a daily prompt or creative challenge spanning various multimedia disciplines, including graphic design, illustration, animation, video editing, audio creation, etc.
- ◯ Vividly: Vividly is your virtual gallery space, where artists can showcase their creations with vibrant detail. Upload your artwork effortlessly and immerse visitors in a visually stunning journey.
- ◯ ArtBenta: Your gateway to an artistic marketplace. Tailored for artists and BMMA students, ArtBenta is a fusion of art and commerce, where creators can effortlessly showcase and sell their multimedia treasures.
- ◯ Other: _____

Continuation…

What specific features would you like to see in a website or app designed for multimedia students? *(You can write as many as you want)* *

Your answer

What qualities do you consider important for a user-friendly experience? *(Select all that apply)* *

- [ ] Intuitive navigation
- [ ] Clean and visually appealing design
- [ ] Quick and easy access to features
- [ ] Mobile responsiveness
- [ ] Other: _____

What elements would be crucial for creating an engaging experience? *(Select all that apply)* *

- [ ] Interactive features
- [ ] Artist statements and descriptions
- [ ] User comments and feedback
- [ ] Other: _____

Continuation…

On which devices would you primarily access a multimedia platform? *

- [ ] Desktop computer
- [ ] Laptop
- [ ] Tablet
- [ ] Smartphone

If you have any creative ideas, what would you suggest as a name for the multimedia platform? Feel free to share your naming ideas. *(If none, please leave it blank)*

Your answer

Is there any specific concept or idea related to multimedia arts platforms that you would like to suggest, and that hasn't been addressed in the previous questions? Feel free to share your unique thoughts and preferences. *(If none, please leave it blank)*

Your answer

Exhibit B.

Designing Your Ideal Multimedia Space: Student Survey

51 responses

Publish analytics

Name: (Optional)
(Family name, Given name M.I.)

10 responses

De Rivera, Alexandra, N.

Mendoza, Carla T.

Juliano, Jade

Caspian

Yanga, Kal P.

Julian, Abbiegail

PIDO , IRISH LAYE P.

Luna

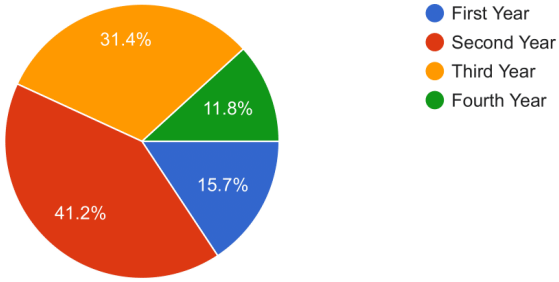
Zer

Peter

Continuation…

Year Level: Copy

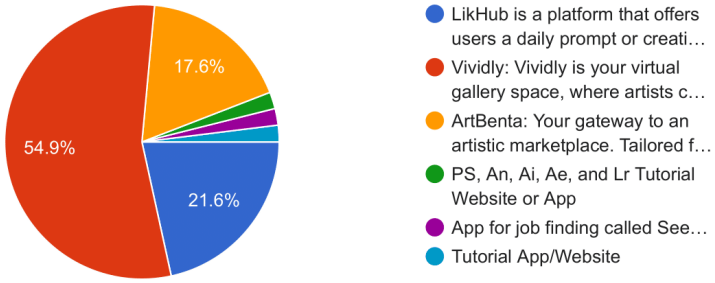
51 responses



Questions

As an artist or BMMA student, which type of online platform or tool would you find most useful for showcasing your multimedia work? You can choose from the provided options, but if none of the choices apply, please specify. Copy

51 responses



What specific features would you like to see in a website or app designed for multimedia students? *(You can write as many as you want)*

51 responses

Basta maganda ata madaling matutunan.

A built-in notify feature for websites that can get you a client.

Users can add and remove their favorites.

Continuation…

An app where I can showcase my work and also find clients.
Good graphics
Something new.
Tutorials or anything that make us work easily.
Search feature. Where you can search artists and their work of art.
The artist(s) name, background, artworks and their rates.
As a student I would appreciate if there was a free brushes to use to sketch our Art works and etc also I want to improve our skill when it comes of animation most of us enjoy making a character design or making animation just for fun or to improve our knowledge to it.
Favorite or like button.
Easy to navigate, good page speed, simple and easy to reach.
User-Friendly Interface, Course Organization, Interactive Learning Materials
Anything that is good is always welcome.
Stable
Stripped down version of Behance.
The ability to rate artworks.
I would like to see a user-friendly interface, tutorials covering animation, video and photo editing, graphic design along with features like interactive quizzes and hands-on exercises within the tutorials, and a forum discussion for students to share their insights, seek advice, and collaborate with each other.
Anything that works
Minimal but elegant
A different kind of art.

Continuation…

a proper client payment secured system to avoid scam or bogus buyers and fake artist scams
Search feature. I want to search artists and their artworks.
Simple features lang. May search, like or favorite button mga ganun.
Like button. Like count. I want to see a search button also.
I think encouraging yung mga colors para makapag attract ng mga users, syempre madali gamitin.
Maganda pag may search, hindi siya case sensitive ganun ganun. Tapos maganda rin gamitin niyo yung dim color ng Twitter. Astig siya para sa Vividly.
Simpe lang,parang social media siya for artists and their artworks.
Gusto ko aesthetic yung website, may like and dislike button pwede rin yung may artist na profile, yung about sakanya like contact information
May report button rin po sana, tsaka dapat verified lang po yung pwede mag upload hehe
Gusto ko po may save button at comment para makapag suggest and ma judge yung artwork na inupload
May search button po
May like button po
To what I have chosen platform I want features like I can track pending orders, have reviews, I can see what is my ROI.
i want colorful, at may meaning talaga yung mga pinoportrait at shinoshow nilang arts
Parang social media siya. May like comment follow ect.
Artist name, bio, content information.
Twitter of creators. Virtual Gallery Experience.
Edit features, add, delete artworks.

Continuation…

Kung kayavgumawa ng chatbox much better.

Add some features from the LikHub.

Easy to use. Clean UI.

Feature mode. Feature the artwork with the most likes.

Fast loading time. No bogus buyer.

Anything, basta pagkakakitaan.

Follow fellow artists.

Pasiklaban ng nga BMMA students.

Pwedeng maglike and comment.

Customer service.

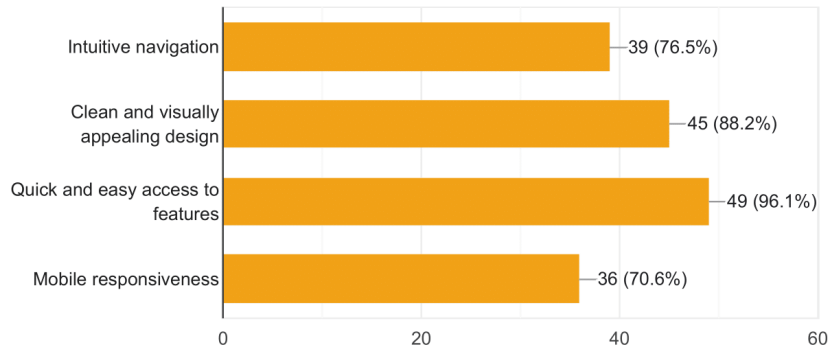
Less bugs if not no bugs at all.

Anything na maisip niyong mga researchers.

What qualities do you consider important for a user-friendly experience? *(Select all that apply)*

Copy

51 responses

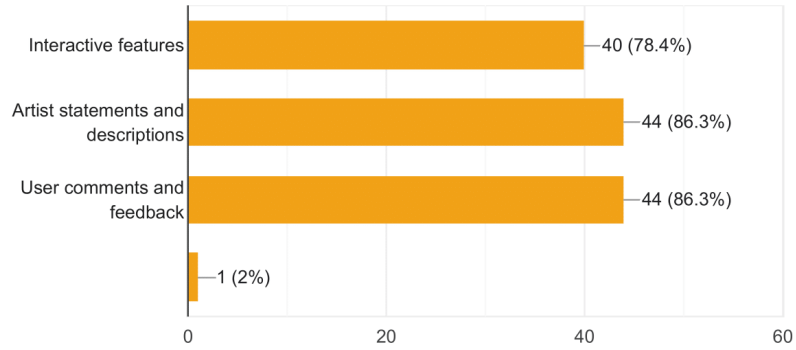


Continuation…

What elements would be crucial for creating an engaging experience? *(Select all that apply)*

Copy

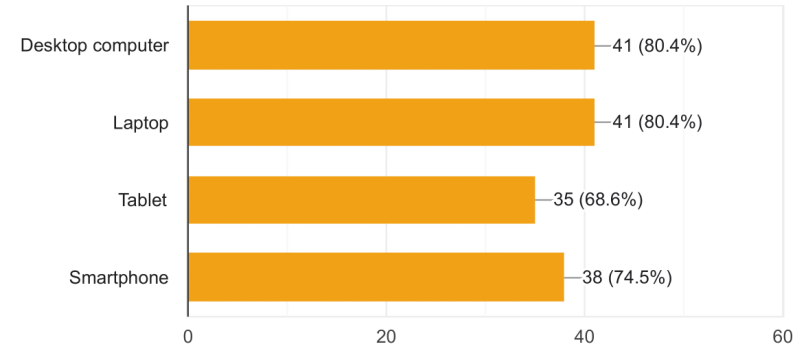
51 responses



On which devices would you primarily access a multimedia platform?

Copy

51 responses

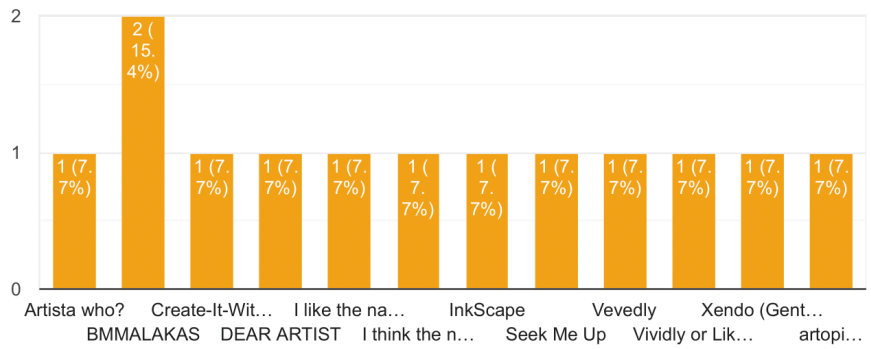


Continuation…

If you have any creative ideas, what would you suggest as a name for the multimedia platform? Feel free to share your naming ideas. *(If none, please leave it blank)*

Copy

13 responses



Is there any specific concept or idea related to multimedia arts platforms that you would like to suggest, and that hasn't been addressed in the previous questions? Feel free to share your unique thoughts and preferences. *(If none, please leave it blank)*

7 responses

Well, if you could combine the LikHub and Vividly together, I think that would be better.

App for job finding called Seek Me Up.

AR-enhanced Learning Materials

I think Vividly would be better if it includes the features of LikHub.

none

None

Parang pasiklaban sila pag may bagong challenge. Magupload sila all they want walang limit.