ArtVision: Collaborative E-Commerce Art Gallery Platform:

Project Idea:

Our goal is to build a fully integrated art gallery website that allows users to create accounts, share artwork, and make purchases. The website will provide a platform for artists and art enthusiasts to connect, explore, and transact with ease and security. It's a project inspired by popular art platforms such as DeviantArt, Behance, Dribbble, and ArtStation.

Team Members:

Jesus Zubia, Obie Carnathan, Katrina Murrell, and Daniela Chavez

Technologies Planned for Use:

Front-End Development:

- HTML/CSS: To create the structure and design of the website.
- JavaScript: To add interactivity and dynamic features.
- React: To simplify the development process and enhance user experience.

Back-End Development:

- Python/Django: Our chosen server-side programming language and framework.
- PostgreSQL: Our chosen database to store user information and artwork details.

User Authentication and Account Management:

- Firebase Authentication: To handle user registration, login, and account management.
- User Profile Features: To allow users to create and manage their profiles and update personal information.

Artwork Display and Gallery Functionality:

- Art Upload: All users, regardless of account status, will have the ability to upload artwork to the platform. This not only simplifies the uploading process but also provides a more inclusive environment for creativity. Additionally, each user will have a personal gallery on their profile, where their uploaded artwork will be showcased.
- CSS Grids/Masonry.js: To create a responsive and organized gallery layout.

- Art Discovery: Implement basic tags or categories for easy discovery and filtering of artwork.

Art 'Purchase' Functionality:

- Purchase's Functionality: Users will have the option to make their artwork available for purchase. By clicking a 'purchase' button, the system will deduct from the artwork's quantity. The original uploader will be notified of this action, ensuring they remain updated about their listings. No actual payment processing will be implemented for this project.
- Seller Interface: To complement the Purchase Functionality, a user-friendly interface will be created for sellers. This interface will allow them to view, edit, and delete their art listings as per their requirements, providing them with full control over their content and sales.

Security and Performance:

- SSL Certificate: To enable secure connections (HTTPS) and protect user data during transmission.

- Performance Optimization Techniques: Such as caching and code minification to improve website performance and load times.

Meeting Notes: ArtVision - Collaborative E-Commerce Art Gallery Platform

Date: July 13, 2023

Participants: Jesus Zubia, Obie Carnathan, Katrina Murrell, and Daniela Chavez

Agenda:

1. Website Structure
2. Technology Stack
3. Work Division
4. Meeting Schedule
5. Goals and Deadlines

Notes:

1. Website Structure

Our proposed website will be comprised of six key pages:

Information Pages

Main Page: Features art pieces available for purchase, organized into rolls/cells. Users can click on an art piece to view and/or buy.

User Profile Page: Hosts relevant user information, including art pieces bought, sold, and uploaded.

A user's profile from the eyes of another user should show the user's bio/photo and all art uploaded by that user. The art will either be sellable or just simply available for viewing.

Art Information Page: Displays specific details about the selected art piece (i.e., upload date, creator, title, description) and provides the option to buy.

Functionality Pages

Art Purchase Procedure Page: Outlines the procedure for purchasing an art piece.

Art Selling Procedure Page: Details the steps required for selling art pieces.

Art Upload Procedure Page: Guides users on how to upload art pieces.

2. Technology Stack

After deliberation, we have proposed the following stack for our project:

Front-end: JavaScript, HTML/CSS

Back-end: Python

Database: PostgreSQL

Note: We will double-check the acceptability of these technologies before our next meeting.

3. Work Division

Team roles have been allocated as follows:

Front-end Development: Obie Carnathan, Daniela Chavez

Web Design: Daniela Chavez, Katrina Murrell

Back-end Development: Jesus Zubia, Katrina Murrell

Database Management: Jesus Zubia, Katrina Murrell

Note: All roles are flexible and may be adjusted as needed.

4. Meeting Schedule

Team meetings will be held twice a week at 7:30 CT/8:30 ET every Monday and Thursday.

5. Goals and Deadlines

For the first week, we aim to complete a basic version of the six key pages. The design and full functionality will be refined in subsequent weeks. Please note that this does not mean they need to be interconnected or have working functionalities at this point, but should exist in their basic format.

Our timeline is as follows:

- **Week 1:** Basic structure for all six pages
- **Week 2:** Design and functionality improvement (TBD)
- **Week 3:** Finalization and Testing (TBD)

Next Steps:

1. Each team member begins work on their assigned tasks.
2. Validate the technology stack.
3. Prepare for the first weekly goal of creating the basic page structure.

ArtVision:
A Virtual Art Gallery & Social Platform

developed by

Obie Carnathan
Daniela Chavez
Katrina Murrell
Jesus Zubia

CS 421/621 Advanced Web Development
August 4, 2023

1. Introduction

"See the world in new colors."

The purpose of this project was to develop ArtVision, a virtual art gallery and social platform for artists—hobbyists and professionals alike—to share their work, sell prints, and above all else immerse themselves in a community of fellow art enthusiasts. This report encapsulates the processes, technologies, and crucial functionalities that contributed to the realization of ArtVision.

2. Project Planning and Communication

Active participation and consistent communication were key to the success of ArtVision's development. Each team member attended all meetings and was involved in decision-making processes every step of the way. GroupMe and Discord served as our main channels of communication, fostering swift idea exchanges and problem resolution.

a. Meeting Highlights:

- **Meeting 1: Week of July 10, 2023**
  - Determined the structure of the website comprising six key pages.
  - Choose the technology stack, including Django as the web framework and SQLAlchemy/Flask_SQLAlchemy for the database.
  - Allocated team roles for frontend and backend development, web design, and database management.
  - Set a meeting schedule along with project goals and deadlines.

- **Meeting 2: Week of July 17, 2023**
  - Created a basic structure of the web pages using HTML and CSS, without incorporating any functionalities.

    - Finalized the decision to transition from SQL database technology to SQLAlchemy/Flask_SQLAlchemy for the application's database structure, and initiated work on it.
    - Created a draft of the website's design using Figma, including finalized logos. Discussed the implementation of functionalities.

- **Meeting 3: Week of July 24, 2023**
    - Began the implementation of CSS/HTML designs, importing the necessary modules for functionalities.
    - Initiated the implementation of functionalities for various operations, such as uploading, selling, and buying artworks, along with the user login/logout mechanism. Included a view option for art pieces.
    - Integrated Amazon Web Services (AWS) for hosting our application and managing resources. Utilized key services such as Amazon S3 for storage, and this step ensured the scalability and robustness of our platform.

- **Meeting 4: Week of July 31, 2023**
    - Commenced work on the project report.
    - Fixed the logout issue and updated the navigation bar functionality.
    - Scheduled a meeting with Unan to discuss project requirements.
    - Outlined future tasks including development of user profile page, checkout functionality, and design enhancements.
    - Decided to finalize the application with comprehensive error testing.
    - Planned to work on a presentation video that includes a short demo of our application.

3. Website Features

**Features that were not implemented by the project deadline or are unstable are highlighted.*

- **Landing Page**
    - Options to create an account, log in, or continue to the explore page without an account
    - Features a slideshow background of randomly selected artwork, as well as details about the artwork in the lower right-hand corner of page

- **Explore Page**
    - Search
    - Options to view artwork by popularity, recent upload date, and category
    - Link to a random artwork

- **User Profiles**
    - Show info about the user (display name, pronouns, title, bio, date joined)

    - Gallery tab - art user has submitted
    - Shop tab - prints the user has made available for purchase
    - Favorites tab - art user has added to their favorites
    - If user is logged in, shows options to edit account and manage gallery (delete artwork)

- **Edit Account**
    - Update user profile information
        - Change username, display name, pronouns, title, or bio
    - Update account information
        - Change email or password
        - Requires users to verify their current email and password
    - Manage account
        - Option to delete account

- **Artwork Detail Page**
    - Show info about the artwork (artist name, description, category, upload date)
    - If a print of the artwork is available for purchase, display price details and purchase option
    - If user is logged in, shows option to add the artwork to their favorites
    - Comment section
        - Users must be registered and logged in to leave a comment

- **Submit Artwork**
    - File upload
    - Provide artwork information (title, description, category)
    - View a thumbnail of file upload
    - Users must be registered and logged in to submit art

- **Sell Prints**
    - File upload
    - Provide artwork information (title, description, category, price)
    - View a thumbnail of file upload
    - Users must be registered and logged in to sell art

- **Purchase Prints**
    - View selected artwork labeled to be sell within the shop page
    - Within each artwork will redirect to the artwork information page
    - Provide artwork information (author, date of publication, description, title, price)
    - Users must be registered and logged in to purchase prints

    - After artwork has been added to cart, users will be able to finalize purchase through the shopping icon demonstrating their total amount

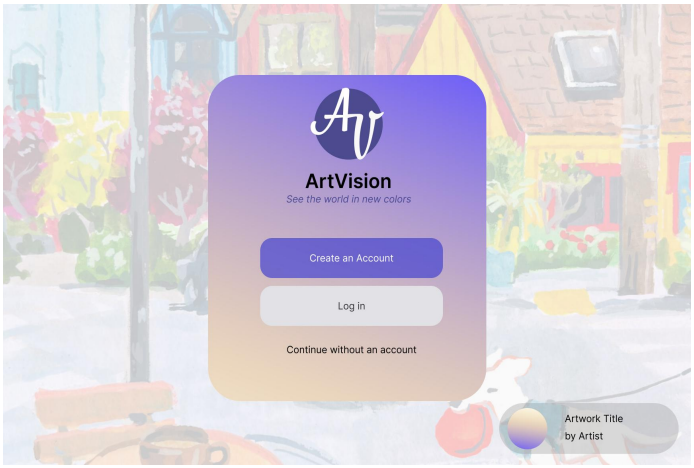
- **Administrative Role**
    - There are no admin users at this time but we do have a page to access important site data.
    - User Management
        - View all ArtVision users and their account information
    - Artwork Management
        - View all artwork submitted to the site
    - View all transaction information

4. Website Mockup

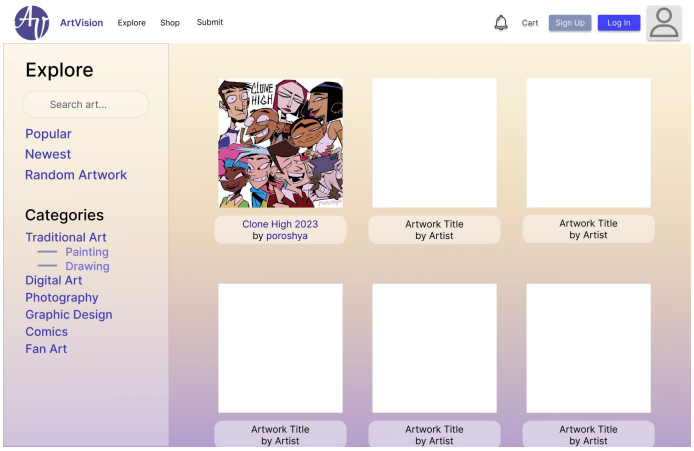
To view designs for each of ArtVision's (initially planned) pages, visit the Figma board: https://www.figma.com/file/jYv5IE4s6qmNwuADLbjczi/ArtVision?type=design&node-id=0-1&mode=design

a. Selected Pages:

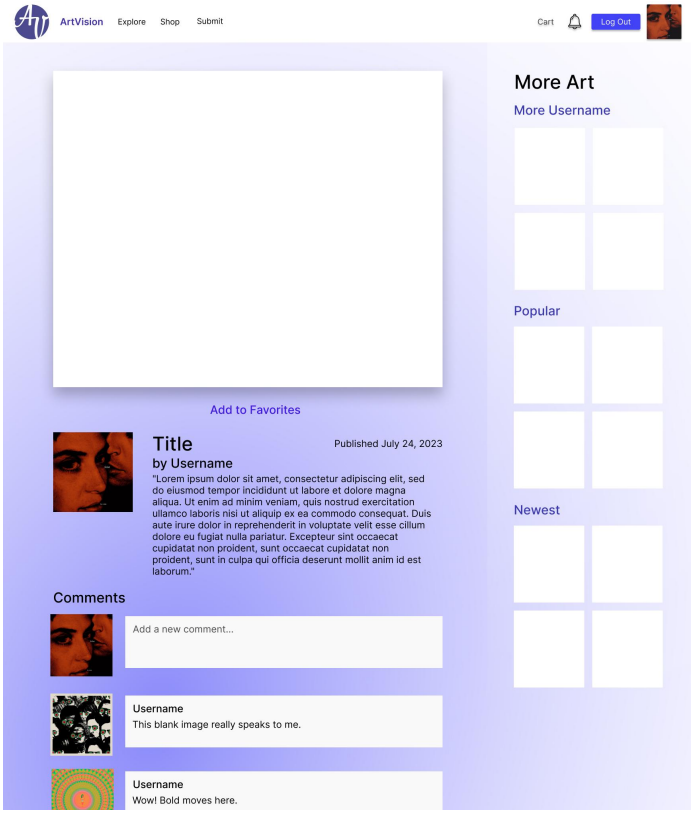
Landing Page



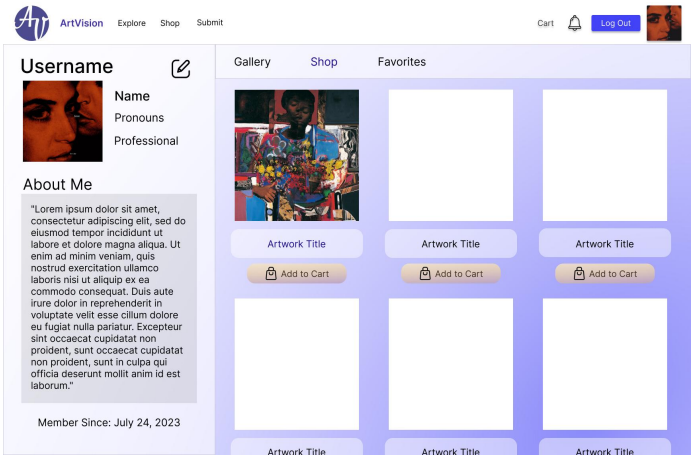
Explore Page



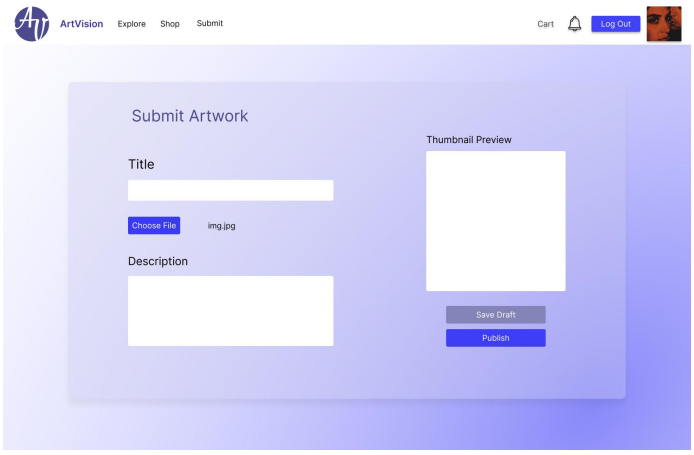
Artwork Detail Page



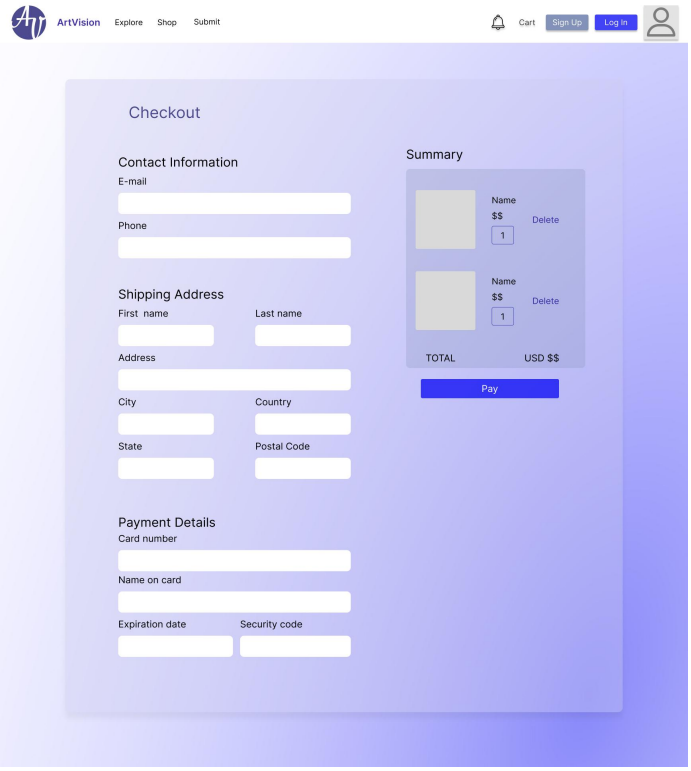
User Profile Page - Shop Tab



Submit Art Page



Shop Checkout Page



5. Technologies:

The following technologies were carefully chosen to ensure optimal performance and a seamless user experience:

- **Design:** Figma was used for our user interface design process. Its collaborative features allowed for a streamlined approach. The website mockups created in Figma guided our frontend development process, facilitating consistency across our application and ensuring a unified aesthetic by providing reusable design templates.

- **Frontend:** HTML, CSS, and JavaScript—in tandem with the jQuery library and Bootstrap framework—were employed to create an engaging user interface.

- **Backend:** Python was chosen to manage server-side functionality. Initially, we considered Django for the web framework, but we eventually settled on Flask because of its simplicity, flexibility, and fine-grained control over the components of the application. SQLAlchemy was used to facilitate interaction with the database, and Boto3 was employed to interface with AWS.

- **Database:** Initially, we selected PostgreSQL for use as our database system because of its comprehensive feature set and powerful SQL capabilities. However,

as the project progressed, we realized the benefits of using SQLAlchemy with Flask_SQLAlchemy. This combination offered a much more seamless integration with our Python-based backend. SQLAlchemy abstracted low-level SQL commands and provided a high-level, pythonic interface to interact with our database. This switch not only increased our development speed but also greatly improved the maintainability and scalability of our application.

- **Cloud:** We decided to use Amazon Web Services (AWS) for hosting and managing user-submitted images. Specifically, we employed Boto3, the AWS SDK for Python, to create, configure, and manage an Amazon Simple Storage Service (Amazon S3) bucket. This not only allowed us to leverage AWS's powerful infrastructure but also ensured seamless integration with our Python-based backend. AWS also offers top-notch security measures, ensuring the safety and integrity of our user data.

6. Team Contributions

Each member of our team demonstrated remarkable flexibility and a readiness to assume different roles as needed. Our collaborative efforts and individual initiatives were equally crucial to the success of ArtVision's development. A detailed overview of each team member's contributions is as follows:

- **Obie Carnathan:** Obie was our resident cloud technology maestro. While he made significant contributions in back-end development and database management, his most impactful role was the integration of Amazon Web Services (AWS). Obie also demonstrated exceptional problem-solving skills when the team faced challenges with GitHub. His ability to resolve complex merge conflicts and other issues was critical to our team's ability to maintain a smooth development process.

- **Daniela Chavez:** Daniela collaborated with Katrina in creating the website's user interface design template in Figma. She was involved in both front-end and back-end development as well as database management and debugging, playing a significant role in the introduction of several key website functionalities. Her efforts were particularly noteworthy in implementing the art submission pages, checkout page and enhancing users' overall experience.

- **Katrina Murrell:** Katrina demonstrated her design prowess by spearheading creation of the website's user interface design template in Figma. She worked hand in hand with Jesus, Daniela, and Obie on both front-end and back-end development with significant contributions to database management, debugging and code revamping, and new feature implementation. Selected contributions of note include comment functionality, a robust search feature, and account deletion. These multifaceted contributions greatly enhanced the user experience,

underlining her valuable role within the team.

- **Jesus Zubia:** Jesus was instrumental in both front-end and back-end development of the platform, demonstrating his proficiency in JavaScript and Python. Besides building a user-friendly interface and efficient server-side functionalities, he also spearheaded the design and management of ArtVision's database architecture, ensuring smooth data retrieval within the platform. Jesus also contributed significantly to the efficiency of our team outside of development by keeping extensive meeting notes and drafting our final report.

7. Results

- **HTML:**
    - **Static:**
        - **css:**
            - **Profile.css:**
            - **Purchase.css:**
            - **registrationstyles.css:**
        - **js:**
        - **env.py:**

```python
import logging
from logging.config import fileConfig

from flask import current_app

from alembic import context

# this is the Alembic Config object, which provides
# access to the values within the .ini file in use.
config = context.config

# Interpret the config file for Python logging.
# This line sets up loggers basically.
fileConfig(config.config_file_name)
logger = logging.getLogger('alembic.env')


def get_engine():
    try:
        # this works with Flask-SQLAlchemy<3 and Alchemical
        return current_app.extensions['migrate'].db.get_engine()
    except TypeError:
        # this works with Flask-SQLAlchemy>=3
        return current_app.extensions['migrate'].db.engine


def get_engine_url():
    try:
        return get_engine().url.render_as_string(hide_password=False).replace(
            '%', '%%')
    except AttributeError:
        return str(get_engine().url).replace('%', '%%')


# add your model's MetaData object here
# for 'autogenerate' support
# from myapp import mymodel
# target_metadata = mymodel.Base.metadata
config.set_main_option('sqlalchemy.url', get_engine_url())
target_db = current_app.extensions['migrate'].db

# other values from the config, defined by the needs of env.py,
# can be acquired:
# my_important_option = config.get_main_option("my_important_option")
# ... etc.


def get_metadata():
    if hasattr(target_db, 'metadatas'):
        return target_db.metadatas[None]
    return target_db.metadata


def run_migrations_offline():
    """Run migrations in 'offline' mode.

    This configures the context with just a URL
    and not an Engine, though an Engine is acceptable
    here as well.  By skipping the Engine creation
    we don't even need a DBAPI to be available.

    Calls to context.execute() here emit the given string to the
    script output.

    """
    url = config.get_main_option("sqlalchemy.url")
    context.configure(
        url=url, target_metadata=get_metadata(), literal_binds=True
    )

    with context.begin_transaction():
        context.run_migrations()


def run_migrations_online():
    """Run migrations in 'online' mode.

    In this scenario we need to create an Engine
    and associate a connection with the context.

    """

    # this callback is used to prevent an auto-migration from being generated
    # when there are no changes to the schema
    # reference: http://alembic.zzzcomputing.com/en/latest/cookbook.html
    def process_revision_directives(context, revision, directives):
        if getattr(config.cmd_opts, 'autogenerate', False):
            script = directives[0]
            if script.upgrade_ops.is_empty():
```

```python
def run_migrations_online():
    """Run migrations in 'online' mode.

    In this scenario we need to create an Engine
    and associate a connection with the context.

    """

    # this callback is used to prevent an auto-migration from being generated
    # when there are no changes to the schema
    # reference: http://alembic.zzzcomputing.com/en/latest/cookbook.html
    def process_revision_directives(context, revision, directives):
        if getattr(config.cmd_opts, 'autogenerate', False):
            script = directives[0]
            if script.upgrade_ops.is_empty():
                directives[:] = []
                logger.info('No changes in schema detected.')

    connectable = get_engine()

    with connectable.connect() as connection:
        context.configure(
            connection=connection,
            target_metadata=get_metadata(),
            process_revision_directives=process_revision_directives,
            **current_app.extensions['migrate'].configure_args
        )

        with context.begin_transaction():
            context.run_migrations()


if context.is_offline_mode():
    run_migrations_offline()
else:
    run_migrations_online()
```

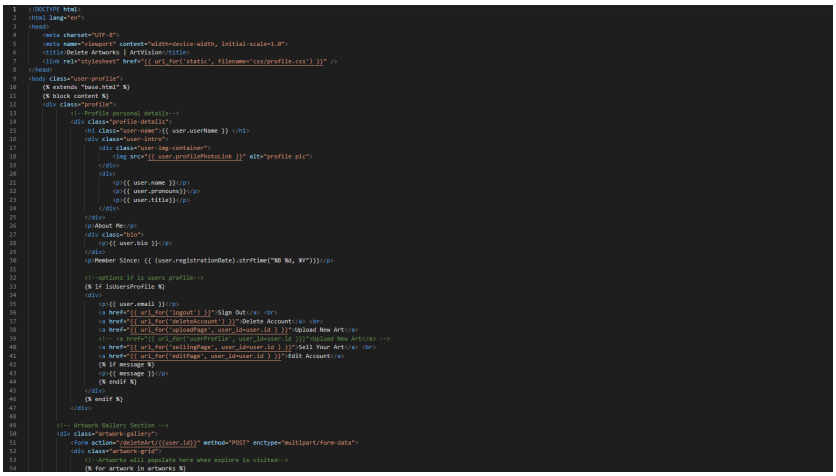
- **Templates:**
  - **artworkDetails.html:**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{{artwork.title}} by {{artist.userName}}</title>
    <!-- Add your CSS file link here -->
    <link rel="stylesheet" href="{{ url_for('static', filename='css/styles.css') }}" />
</head>
<body class="details-page gradient-bg-1">
    <!-- Navbar (You can create a separate header file and include it on all pages) -->
    <header>
        {% extends "base.html" %}
     </header>
    {% block content %}

        <!-- Artwork Gallery Section -->
        <div class="artwork">

            <!-- Display the user's artworks here -->
            <!-- You can use a loop to generate the artwork thumbnails -->
            <!-- Example of a single artwork thumbnail -->
            <div class="full-img">
                <a href="{{ artwork.url }}">
                    <img src="{{ artwork.url }}" alt="{{ artwork.title }}">
                </a>
            </div>

            <div class="info">
                <a href="{{ url_for('userProfile', user_id=artwork.user_id) }}"><img class="user-pic"  src="{{ artist.profilePhotoLink }}" alt="profile pic"></a>
                <div class="details">
                    <h3>{{artwork.title}}</h3>
                    <p>by <a href="{{ url_for('userProfile', user_id=artwork.user_id) }}">{{artist.userName}}</a></p>
                    <p>{{artwork.description}}</p>
                    <p>Published {{artwork.uploadDate.strftime("%B %d, %Y")}}</p>
                </div>
            </div>

            <div>
                {% if artwork.shop_item %}
                <h4>Buy a print?</h4>
                <p>$ {{"%.2f"|format(artwork.price)}}</p>
                <p>({artwork.status})</p>
                <form action="{{ url_for('addCart', artwork_id=artwork.id) }}" method="POST"
                enctype="multipart/form-data">
                    <label for="quantity">Quantity:</label>
                    <input type="number" name="quantity" value="1" min="1">
                    <button type="submit">Add to cart</button>
                </form>
                {% endif %}
            </div>
        </div>
```

```
        <!-- Comment Section -->
        <!-- <form class="ui form" action="/artwork/{{artwork.id}}" method="post" enctype="multipart/form-data">
            <img src="{{user.profilePhotoLink}}" alt="{{user.userName}}" width="50px">
            <input type="text" name="text" placeholder="Add a new comment...">
            <button class="ui blue button custom-button" type="submit">Comment</button>
        </form>

        <div comment-section>
        {% for comment in comments %}
            <div comment>
                <a href="{{ url_for('userProfile', user_id=comment.author_id) }}"> <img src="{{comment.profile_pic}}" alt="{{comment.author}}" width="50px"></a>
            <p><a href="{{ url_for('userProfile', user_id=comment.author_id) }}">{{comment.author}}</a> says: {{comment.text}}</p>
            {% if user.userName == comment.author %}
                <a href="{{ url_for('deleteComment', comment_id=comment.id, artwork_id=artwork.id)}}" class="ui blue button custom-button">Delete</a>
            {% endif %}
            </div>
        {% endfor %}
        </div> -->

                <b>Comments</b> {{commentsCount}}

                <form class="ui form" action="/artwork/{{artwork.id}}" method="post" enctype="multipart/form-data">
                    <img src="{{user.profilePhotoLink}}" alt="{{user.userName}}" width="50px">
                    {% if userLoggedIn %}
                    <input type="text" name="text" placeholder="Add a new comment...">
                    <button class="btn btn-primary" role="button" type="submit">Comment</button>

                    {% endif %}

                    {% if userLoggedIn == False %}
                    <input type="text" name="text" placeholder="Log in to comment...">
                    <a href="{{url_for('loginPage')}}" class="btn btn-primary" role="button">Comment</a>
                    {% endif %}

                </form>

                <br>
                <div comment-section>
                {% for comment in comments %}
                    <div comment>
                        <p> <a href="{{ url_for('userProfile', user_id=comment.author_id) }}"> <img src="{{comment.profile_pic}}" alt="{{comment.author}}" width="50px"></a>
                    <a href="{{ url_for('userProfile', user_id=comment.author_id) }}">{{comment.author}}</a> says: {{comment.text}}
                    {% if user and user.userName == comment.author %}
                        <a href="{{ url_for('deleteComment', comment_id=comment.id, artwork_id=artwork.id)}}" class="btn btn-primary" role="button">Delete</a></p>
                    {% endif %}
                    </div>
                {% endfor %}
                </div>
    <!-- Footer -->


    <!-- Add your JavaScript file link here if needed -->
    <script src="scripts.js"></script>
    {% endblock %}
```

  - **Base.html:**

- **deleteArt.html:**



```
                    <a href="{{ url_for('sellingPage', user_id=user.id ) }}">Sell Your Art</a> <br>
                    <a href="{{ url_for('editPage', user_id=user.id ) }}">Edit Account</a>
                    {% if message %}
                    <p>{{ message }}</p>
                    {% endif %}
                </div>
                {% endif %}
            </div>

        <!-- Artwork Gallery Section -->
        <div class="artwork-gallery">
            <form action="/deleteArt/{{user.id}}" method="POST" enctype="multipart/form-data">
            <div class="artwork-grid">
                <!-- Artworks will populate here when explore is visited-->
                {% for artwork in artworks %}
                <!-- Display the user's artworks here -->
                <!-- You can use a loop to generate the artwork thumbnails -->
                <!-- Example of a single artwork thumbnail -->
                <div class="artwork-thumbnail">
                    <input type="checkbox" name="artworkToDelete" value="{{artwork.id}}">
                    <a href="{{ url_for('artworkDetails', artwork_id=artwork.id) }}">
                        <img src="{{ artwork.url }}" alt="{{ artwork.title }}">
                    </a>
                    <div class="art-title"><p>{{ artwork.title}}</p></div>
                </div>
                {% endfor %}
            </div>
            <button class="ui blue button custom-button" type="submit">Delete Selected Artwork</button>
            </form>
        </div>
    </div>

    <!-- Footer -->
    <footer>
        <!-- Your footer content here -->
    </footer>

    <script src="{{ url_for('static', filename='js/user_data_fetch.js') }}"></script>
    {% endblock %}
</body>
</html>
```

- **Error404.html:**

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
    <head>
        <meta charset="utf-8">
        <title>Error Page</title>
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js" integrity="sha384-geWF76RCwLtnZ8qwWowPQNguL3RmwHVBC9FhGdlKrxdiJJigb/j/68SIy3Te4Bkz" crossorigin="anonymous"></script>
        <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
        <div class="jumbotron">
      <h3> Sorry, we couldn't find the page you were looking for.</h3>
    </div>
</html>
```

- **Homepage.html:**

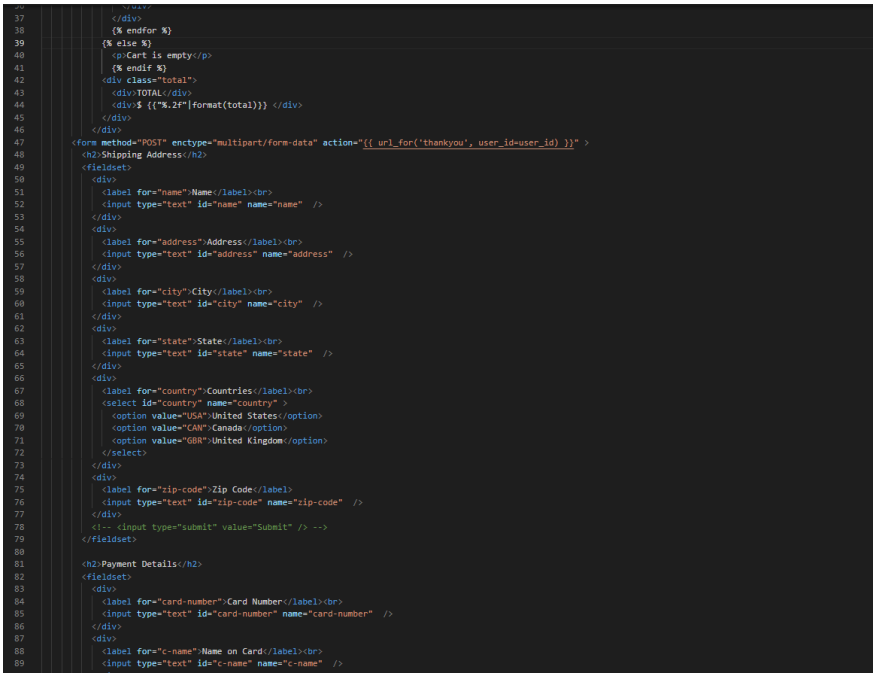
```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <title>ArtVision</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/registrationstyles.css') }}" />
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.0/jquery.min.js"></script>
    <script src="../static/js/scripts.js"></script>
    <script>
        var all_art = {{ all_art|tojson }}
        var all_art_objects = {{ all_art_objects|tojson }}
        var all_user_objects = {{all_user_objects|tojson}}
    </script>
</head>

<body id="background" style="background-image: url({{all_art[0]}});">
    <div class="fd-container">
            <img src="{{ url_for('static', filename='img/ArtVision_Logo.svg')}}" width="80" alt="Logo">
            <h1>ArtVision</h1>
            <p><em>See the world in new colors</em></p>
            <div class="home-buttons">
                <button type="buttons" class="btn btn-primary" onclick="window.location.href='{{ url_for('register') }}';">Create an Account</button>
                <button type="button" class="btn btn-secondary" onclick="window.location.href='{{ url_for('loginPage') }}';">Log In</button><!-- Needs to be updated to Sign Up page -->
            </div>
            <p><a href="{{url_for('explore', sort='random')}}">Continue without an Account</a></p>
        </div>

        <div class="artwork-info">
            <img class="user" src="{{ url_for('static', filename='img/ArtVision_Logo.svg')}}" id="artwork_pfp">
            <div class="artist-details">
            <p id="artwork_title">Artwork title</p>
            <p>by <span id="artwork_artist">Artist</span></p>
        </div>
        </div>
        <script src="../static/js/scripts.js"></script>
    </body>
</html>
```

- **loginPage.html:**

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>ArtVision | Login</title>
        <link rel="stylesheet" href="{{ url_for('static', filename='css/registrationstyles.css') }}" />
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.0/jquery.min.js"></script>
        <script>
            var all_art = {{ all_art|tojson }}
            var all_art_objects = {{ all_art_objects|tojson }}
            var all_user_objects = {{all_user_objects|tojson}}

        </script>
    </head>

    <body id= "background" style="background-image: url({{all_art[0]}});">
            <div class="fd-container">
              <a href="{{url_for('index')}}"><img src="{{ url_for('static', filename='img/ArtVision_Logo.svg')}}" width="80" alt="Logo"></a>
                <h1>Login to Your Account</h1>
                <form method="POST">
                    {{form.hidden_tag()}}
                    <br>
                    {{form.userIdentity(placeholder="Email address or username")}}
                    <br>
                    {{form.password(placeholder="Password", type="password")}}
                    <br>
                    {{form.submit(class="custom-button")}}
                </form>
                {% if error %}
                <p>{{ error}}</p>
                {% endif %}
                <p>Not a member yet? <a class="createAccount" href="{{url_for('register')}}">Create an Account</a></p>
            </div>
            <div class="artwork-info">
                <img class="user" src="{{ url_for('static', filename='img/ArtVision_Logo.svg')}}" id="artwork_pfp">
                <div class="artist-details">
                <p id="artwork_title">Artwork title</p>
                <p>by <span id="artwork_artist">Artist</span></p>
            </div>
            </div>

        <script src="../static/js/scripts.js"></script>
    </body>
</html>
```

- **Purchase.html:**

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <link rel="stylesheet" href="{{ url_for('static', filename='css/styles.css') }}" />
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet"
    integrity="sha384-9ndCyUaIbzAi2FUVXJi0CjmCapSmO7SnpJef0486qhLnuZ2cdeRhO02iuK6FUUVM" crossorigin="anonymous">
  <title>Purchase</title>
 </head>
 <body class="purchase-page gradient-bg">
  <header>
   {% extends "base.html" %}
  </header>
{% block content %}

  <main class="main-content">
   <div class="wrapper">
    <h1>Checkout</h1>
    <div class="form-container">
     <div class="checkout-summary">
      <h2>Summary</h2>
      {% if session['cart'] %}
       {% for item, art in cart.items() %}
        <div class="card">
         <a href="{{ url_for('artworkDetails', artwork_id=item) }}">
         <div class="card-img" style="width: 100px; height: 100px; background-color: grey">
          <img style="width: 100%; height: 100%; background-size: cover; background-position: center center; " src="{{art.image_url}}">
         </div>
        </a>
         <div class="card-details">
          <p>{{art.title}}</p>
          <p>$ {{"%.2f"|format(art.price)}}</p>
          <div class="quantity" id="{{art.quantity}}"><button class="decrease" value="{{art.id}}">-</button>{{ art.quantity }}<button class="increase" value="{{art.id}}">+</button></div>
          <a href="{{ url_for('deleteCartItem', artwork_id=item ) }}">Delete</a> <br>
         </div>
        </div>
        {% endfor %}
       {% else %}
        <p>Cart is empty</p>
       {% endif %}
       <div class="total">
        <div>TOTAL</div>
        <div>$ {{"%.2f"|format(total)}} </div>
       </div>
      </div>
     <form method="POST" enctype="multipart/form-data" action="{{ url_for('thankyou', user_id=user_id) }}" >
      <h2>Shipping Address</h2>
      <fieldset>
       <div>
        <label for="name">Name</label><br>
        <input type="text" id="name" name="name"  />
       </div>
       <div>
```



```
                </div>
                {% endfor %}
               {% else %}
                <p>Cart is empty</p>
               {% endif %}
               <div class="total">
                <div>TOTAL</div>
                <div>$ {{"%.2f"|format(total)}} </div>
               </div>
              </div>
             <form method="POST" enctype="multipart/form-data" action="{{ url_for('thankyou', user_id=user_id) }}" >
              <h2>Shipping Address</h2>
              <fieldset>
               <div>
                <label for="name">Name</label><br>
                <input type="text" id="name" name="name"  />
               </div>
               <div>
                <label for="address">Address</label><br>
                <input type="text" id="address" name="address"  />
               </div>
               <div>
                <label for="city">City</label><br>
                <input type="text" id="city" name="city"  />
               </div>
               <div>
                <label for="state">State</label><br>
                <input type="text" id="state" name="state"  />
               </div>
               <div>
                <label for="country">Countries</label><br>
                <select id="country" name="country" >
                 <option value="USA">United States</option>
                 <option value="CAN">Canada</option>
                 <option value="GBR">United Kingdom</option>
                </select>
               </div>
               <div>
                <label for="zip-code">Zip Code</label>
                <input type="text" id="zip-code" name="zip-code"  />
               </div>
               <!-- <input type="submit" value="Submit" /> -->
              </fieldset>

              <h2>Payment Details</h2>
              <fieldset>
               <div>
                <label for="card-number">Card Number</label><br>
                <input type="text" id="card-number" name="card-number"  />
               </div>
               <div>
                <label for="c-name">Name on Card</label><br>
                <input type="text" id="c-name" name="c-name"  />
```

```
                <select id="country" name="country" >
                  <option value="USA">United States</option>
                  <option value="CAN">Canada</option>
                  <option value="GBR">United Kingdom</option>
                </select>
              </div>
              <div>
                <label for="zip-code">Zip Code</label>
                <input type="text" id="zip-code" name="zip-code"  />
              </div>
              <!-- <input type="submit" value="Submit" /> -->
            </fieldset>

            <h2>Payment Details</h2>
            <fieldset>
              <div>
                <label for="card-number">Card Number</label><br>
                <input type="text" id="card-number" name="card-number"  />
              </div>
              <div>
                <label for="c-name">Name on Card</label><br>
                <input type="text" id="c-name" name="c-name"  />
              </div>
              <div>
                <label for="date">Expiration Date</label>
                <input
                  type="text"
                  id="date"
                  name="date"
                  placeholder="MM/YYYY"

                />
              </div>
              <div>
                <label for="cvv">CVV</label>
                <input type="text" id="cvv" name="cvv" placeholder="***"  />
                </div>
            </fieldset>
            <button class="btn custom-button" type="submit">Submit</button>
          </form>
        </div>
        </div>
        </main>
        <script src="{{ url_for('static', filename='js/update_cart.js') }}"></script>
        {% endblock %}
      </body>
    </html>
```

- **Register.html**

```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE-edge">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>ArtVision | Register
        </title>
        <link rel="stylesheet" href="{{ url_for('static', filename='css/registrationstyles.css') }}">
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.0/jquery.min.js"></script>
        <script>
            var all_art = {{ all_art|tojson }}
            var all_art_objects = {{ all_art_objects|tojson }}
            var all_user_objects = {{all_user_objects|tojson}}
        </script>
    </head>

    <body id= "background" style="background-image: url({{all_art[0]}});">
        <div class="register fd-container">
            <a href="{{url_for('index')}}"><img src="{{ url_for('static', filename='img/ArtVision_Logo.svg')}}" width="80" alt="Logo"></a>
            <h1>Create an Account</h1>

            <form class="ui form" action="/add" method="post" enctype="multipart/form-data">
                    <input type="email" name="email" placeholder="Email"> <br>
                    <input type="text" name="username" placeholder="Username"> <br>
                    <input type="password" name="password" placeholder="Password"> <br>
                    <input type="password" name="confirmpassword" placeholder="Confirm password"> <br>
                    <input type="text" name="name" placeholder="Name"> <br>
                    <!--Move bio and profile photo to user profile page-->
                    <input type="text" name="bio" placeholder="Bio"> <br>

                    <div class="pronouns">
                        <p><label for="pronouns">Select your pronouns:</label>
                        <select id="pronouns" name="pronouns">
                            <option value="she/her">she/her</option>
                            <option value="he/him">he/him</option>
                            <option value="they/them">they/them</option>
                            <option value="she/they">she/they</option>
                            <option value="he/they">he/they</option>
                            <option value="any">any pronouns</option>
                        </select></p>
                    </div>

                    <div class="title">
                        <p><label for="title">Select your title:</label>
                        <select id="title" name="title">
                            <option value="Professional">Professional</option>
                            <option value="Student">Student</option>
                            <option value="Hobbyist">Hobbyist</option>
                        </select>
                        </p>
                    </div>
                    <input type="file" name="file">
                    <br>
```

```
22
23          <form class="ui form" action="/add" method="post" enctype="multipart/form-data">
24                  <input type="email" name="email" placeholder="Email"> <br>
25                  <input type="text" name="username" placeholder="Username"> <br>
26                  <input type="password" name="password" placeholder="Password"> <br>
27                  <input type="password" name="confirmpassword" placeholder="Confirm password"> <br>
28                  <input type="text" name="name" placeholder="Name"> <br>
29                  <!--Move bio and profile photo to user profile page-->
30                  <input type="text" name="bio" placeholder="Bio"> <br>
31
32                  <div class="pronouns">
33                      <p><label for="pronouns">Select your pronouns:</label>
34                      <select id="pronouns" name="pronouns">
35                          <option value="she/her">she/her</option>
36                          <option value="he/him">he/him</option>
37                          <option value="they/them">they/them</option>
38                          <option value="she/they">she/they</option>
39                          <option value="he/they">he/they</option>
40                          <option value="any">any pronouns</option>
41                      </select></p>
42                  </div>
43
44                  <div class="title">
45                      <p><label for="title">Select your title:</label>
46                      <select id="title" name="title">
47                          <option value="Professional">Professional</option>
48                          <option value="Student">Student</option>
49                          <option value="Hobbyist">Hobbyist</option>
50                      </select>
51                      </p>
52                  </div>
53                  <input type="file" name="file">
54                  <br>
55              <button class="ui blue button custom-button" type="submit">Sign Up</button>
56          </form>
57          {% if error %}
58          <p>{{ error}}</p>
59          {% endif %}
60          <p>Already have an account? <a class="createAccount" href="{{url_for('loginPage')}}">Log in</a></p>
61      </div>
62
63      <div class="artwork-info">
64          <img class="user" src="{{ url_for('static', filename='img/ArtVision_Logo.svg')}}" id="artwork_pfp">
65          <div class="artist-details">
66          <p id="artwork_title">Artwork title</p>
67          <p>by <span id="artwork_artist">Artist</span></p>
68        </div>
69      </div>
70
71      <script type="text/javascript" src="../static/js/scripts.js"></script>
72  </body>
73  </html>
```

- **Selling.html**

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Selling</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/styles.css') }}" />
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-9ndCyUaIbzAi2FUVXJi0CjmCapSmO7SnpJef0486qhLnuZ2cdeRhO02iuK6FUUVM" crossorigin="anonymous">
  </head>
  <body class="selling-page gradient-bg">
    <header>
      {% extends "base.html" %}
    </header>
  {% block content %}
    <main class="main-content">
      <div class="wrapper">
        <h1>New listing for {{ user.name }} </h1>
      <div class="form-container">
      <form method="POST" enctype="multipart/form-data" action="{{ url_for('addArt', user_id=user_id) }}">
          <div>
          <label>Title</label>
          <br />
          <input type="text" name="title" />
          </div>
          <div>
            <input type="file" name="file" class="fileInput" />
          </div>
          <div class="submit-category">
            <p><label for="category">Select artwork category:</label>
            <select id="category" name="category" required>
                <option value="Traditional Art">Traditional Art</option>
                <option value="Digital Art">Digital Art</option>
                <option value="Mixed Media">Mixed Media</option>
                <option value="Graphic Design">Graphic Design</option>
                <option value="Photography">Photography</option>
                <option value="Comics">Comics</option>
                <option value="Fan Art">Fan Art</option>
                <option value="Other">Other</option>
            </select>
            </p>
        </div>
          <div>
          <label>Description</label> <br />
          <textarea id="description" name="description" rows="5" cols="40"></textarea>
          </div>
        <div>
          <label>Price</label>
          <br />
          <input class="price-input" type="number" step="0.01" min="0.01" name="price" placeholder="$"
          />
          <p class="estimate">Estimated earnings:</p>
        </div>
        <button class="btn custom-button" type="submit">Submit</button>
```

```
                </div>
                <div>
                  <input type="file" name="file" class="fileInput" />
                </div>
                <div class="submit-category">
                  <p><label for="category">Select artwork category:</label>
                  <select id="category" name="category" required>
                      <option value="Traditional Art">Traditional Art</option>
                      <option value="Digital Art">Digital Art</option>
                      <option value="Mixed Media">Mixed Media</option>
                      <option value="Graphic Design">Graphic Design</option>
                      <option value="Photography">Photography</option>
                      <option value="Comics">Comics</option>
                      <option value="Fan Art">Fan Art</option>
                      <option value="Other">Other</option>
                  </select>
                  </p>
                </div>
                  <div>
                  <label>Description</label> <br />
                  <textarea id="description" name="description" rows="5" cols="40"></textarea>
                  </div>
                <div>
                  <label>Price</label>
                  <br />
                  <input class="price-input" type="number" step="0.01" min="0.01" name="price" placeholder="$"
                  />
                  <p class="estimate">Estimated earnings:</p>
                </div>
                <button class="btn custom-button" type="submit">Submit</button>
              </form>

              <div class="preview">
                Preview
                <div class="image-preview"></div>
                </div>
              </div>
              <div>
            </main>
            <script>
              const estimatedEarnings = document.querySelector('.estimate');
              document.querySelector('.price-input').addEventListener('input', function(){
                let inputPrice = parseFloat(this.value)
                let total;
                if (inputPrice >= 0){
                  total = inputPrice * 0.8;
                  estimatedEarnings.textContent = `Estimated earnings: ${total.toFixed(2)}`
                }
              });
            </script>
            <script src="{{ url_for('static', filename='js/file_upload_preview.js') }}"></script>
            {% endblock %}
          </body>
```

- **Shop.html**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Shope ArtVision</title>
    <!-- Add your CSS file link here -->
    <link rel="stylesheet" href="{{ url_for('static', filename='css/styles.css') }}" />
</head>
<body>
    <!-- Navbar (You can create a separate header file and include it on all pages) -->
    <header>
        {% extends "base.html" %}
    </header>
    {% block content %}

        <!-- Artwork Gallery Section -->
        <div class="artwork-gallery">
            <h2>ArtVision Gallery</h2>
            <div class="artwork-grid">
                {% for artwork in artworks %}
                <!-- Display the sitewide artworks here -->
                <!-- Grab random/popular artwork from within a certain time period -->
                <!-- Example of a single artwork thumbnail -->
                <div class="artwork-thumbnail">
                    <a href="{{ url_for('artworkDetails', artwork_id=artwork.id) }}">
                        <img src="{{ artwork.url }}" alt="{{ artwork.title }}">
                    </a>
                    <div class="artwork-details">
                      <p><a href="{{ url_for('artworkDetails', artwork_id=artwork.id) }}">{{artwork.title}}</a> by <a href="{{ url_for('userProfile', user_id=artwork.user_id) }}">{{artwork.artist}}</a></p>
                    </div>
                    <p>${{"%.2f"|format(artwork.price)}}</p>
                </div>
                {% endfor %}
            </div>
        </div>

    <!-- Footer -->
    <footer>
        <!-- Your footer content here -->
    </footer>

    <!-- Add your JavaScript file link here if needed -->
    <script src="{{ url_for('static', filename='scripts.js') }}"></script>
    {% endblock %}
</body>
</html>
```

- **Thankyou.html**

```html
<!DOCTYPE html>
<html lang="en" dir="ltr">
    <head>
        <meta charset="utf-8">
            <title>Thank You</title>
            <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js" integrity="sha384-geWF76RCwLtnZ8qwWowPQNguL3RmwHVBC9FhGdlKrxdiJJigb/j/68SIy3Te4Bkz" crossorigin="anonymous"></script>
            <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
</html>
<body>
    <header>
        {% extends "base.html" %}
    </header>
    {% block content %}
        <h3>{{message}}</h3>
    {% endblock %}
</body>
```

- **Upload.html**

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Upload</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/styles.css') }}" />
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-9ndCyUaIbzAi2FUVXJi0CjmCapSmO7SnpJef0486qhLnuZ2cdeRhO02iuK6FUUVM" crossorigin="anonymous">
  </head>

<body class="upload-page gradient-bg">
  <header>
    {% extends "base.html" %}
  </header>
  {% block content %}
  <main class="main-content">
    <div class="wrapper">
      <h1>Hi {{ user.name }}, submit your artwork</h1>
      <div class="form-container">
        <form class="ui form" action="{{ url_for('addArt', user_id=user_id) }}" method="POST"
          enctype="multipart/form-data">
          <div>
            <label>Title</label>
            <br />
            <input type="text" name="title" />
          </div>
          <div>
            <input type="file" name="file" class="fileInput" />
          </div>
          <div class="submit-category">
            <p><label for="category">Select artwork category:</label>
            <select id="category" name="category" required>
                <option value="Traditional Art">Traditional Art</option>
                <option value="Digital Art">Digital Art</option>
                <option value="Mixed Media">Mixed Media</option>
                <option value="Graphic Design">Graphic Design</option>
                <option value="Photography">Photography</option>
                <option value="Comics">Comics</option>
                <option value="Fan Art">Fan Art</option>
                <option value="Other">Other</option>
            </select>
            </p>
        </div>
          <div>
            <label>Description</label> <br />
            <textarea id="description" name="description" rows="5" cols="40"></textarea>
          </div>
          <!-- <div>
          <label>price</label>
          <br />
          <input
            type="number"
            step="0.01"
```

```
            <input type="file" name="file" class="fileInput" />
          </div>
          <div class="submit-category">
            <p><label for="category">Select artwork category:</label>
            <select id="category" name="category" required>
                <option value="Traditional Art">Traditional Art</option>
                <option value="Digital Art">Digital Art</option>
                <option value="Mixed Media">Mixed Media</option>
                <option value="Graphic Design">Graphic Design</option>
                <option value="Photography">Photography</option>
                <option value="Comics">Comics</option>
                <option value="Fan Art">Fan Art</option>
                <option value="Other">Other</option>
            </select>
            </p>
        </div>
          <div>
            <label>Description</label> <br />
            <textarea id="description" name="description" rows="5" cols="40"></textarea>
          </div>
          <!-- <div>
          <label>price</label>
          <br />
          <input
            type="number"
            step="0.01"
            min="0.01"
            name="price"
            placeholder="price"
          />
        </div> -->

          <!-- <div>
          <label>Status</label>
          <br />
          <input type="text" name="status" placeholder="status" />
        </div> -->
          <button class="btn custom-button" type="submit">Submit</button>
        </form>

        <div class="preview">
          Preview
          <div class="image-preview"></div>
        </div>
      </div>
      <div>
    </main>
    <script src="{{ url_for('static', filename='js/file_upload_preview.js') }}"></script>
    {% endblock %}
  </body>
</html>
```

- **userProfile.html**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{{user.userName}} ({{user.name}}) | ArtVision</title>
    <!-- Add your CSS file link here -->
    <link rel="stylesheet" href="{{ url_for('static', filename='css/profile.css') }}" />

</head>
<body class="user-profile">
    <!-- Navbar (You can create a separate header file and include it on all pages) -->
        {% extends "base.html" %}
    <!-- User Profile Section -->
    {% block content %}
    <div class="profile">
            <!--Profile personal details-->
            <div class="profile-details">
                <h1 class="user-name">{{ user.userName }} </h1>
                <div class="user-intro">
                    <div class="user-img-container">
                        <img src="{{ user.profilePhotoLink }}" alt="profile pic">
                    </div>
                    <div>
                        <p>{{ user.name }}</p>
                        <p>{{ user.pronouns}}</p>
                        <p>{{ user.title}}</p>
                    </div>
                </div>
                <p><b>About Me</b></p>
                <div class="bio">
                    <p>{{ user.bio }}</p>
                </div>
                <p>Member Since: {{ (user.registrationDate).strftime("%B %d, %Y")}}</p>

                <!--options if is users profile-->
                {% if isUsersProfile %}
                <div>
                    <p><a href="{{ url_for('logout') }}" class="btn btn-primary">Log Out</a></p>
                    <!--<p><a href="{{ url_for('uploadPage', user_id=user.id ) }}" class="btn btn-primary">Submit Art</a></p>-->
                    <!--<p><a href="{{ url_for('sellingPage', user_id=user.id ) }}" class="btn btn-primary">Sell Your Art</a></p>-->
                    <p><a href="{{ url_for('deletePage', user_id=user.id ) }}" class="btn btn-primary">Manage Gallery</a></p>
                    <p><a href="{{ url_for('editPage', user_id=user.id ) }}" class="btn btn-primary">Edit Account</a></p>
                    {% if message %}
                    <p>{{ message }}</p>
                    {% endif %}
                </div>
                {% endif %}
            </div>
```

```
                {% if isUsersProfile %}
                <div>
                    <p><a href="{{ url_for('logout') }}" class="btn btn-primary">Log Out</a></p>
                    <!--<p><a href="{{ url_for('uploadPage', user_id=user.id ) }}" class="btn btn-primary">Submit Art</a></p>-->
                    <!--<p><a href="{{ url_for('sellingPage', user_id=user.id ) }}" class="btn btn-primary">Sell Your Art</a></p>-->
                    <p><a href="{{ url_for('deletePage', user_id=user.id ) }}" class="btn btn-primary">Manage Gallery</a></p>
                    <p><a href="{{ url_for('editPage', user_id=user.id ) }}" class="btn btn-primary">Edit Account</a></p>
                    {% if message %}
                    <p>{{ message }}</p>
                    {% endif %}
                </div>
                {% endif %}
            </div>

        <!-- Artwork Gallery Section -->
        <div class="artwork-gallery">
            <nav>
                <a href="#" class="user-data" onclick="loadContent('gallery')" value="{{user.id}}">Gallery</a>
                <a href="#" class="user-data" onclick="loadContent('shop')" value="{{user.id}}" >Shop</a>
                <a href="#" class="user-data" onclick="loadContent('favorites')" value="{{user.id}}">Favorites</a>
            </nav>
            <div class="artwork-grid">
                <!--Artworks will populate here when explore is visited-->
                {% for artwork in artworks %}
                {% if not artwork.shop_item %}
                <!-- Display the user's artworks here -->
                <!-- You can use a loop to generate the artwork thumbnails -->
                <!-- Example of a single artwork thumbnail -->
                <div class="artwork-thumbnail">
                    <a href="{{ url_for('artworkDetails', artwork_id=artwork.id) }}">
                        <img src="{{ artwork.url }}" alt="{{ artwork.title }}">
                    </a>
                    <div class="art-title"><a href="{{ url_for('artworkDetails', artwork_id=artwork.id) }}">{{ artwork.title }}</a></div>
                </div>
                {% endif %}
                {% endfor %}
                </div>

            </div>
        </div>

    <!-- Footer -->
    <footer>
        <!-- Your footer content here -->
    </footer>

    <!-- Add your JavaScript file link here if needed -->
    <script src="{{ url_for('static', filename='js/user_data_fetch.js') }}"></script>
    {% endblock %}
</body>
</html>
```

- **Crud.py:**

```python
from main import app, db, User, Artwork
import os
import boto3

# ###
# GRAB ACCESS_KEY and SECRET_KEY FROM GITHUB. DO NOT COMMIT TO GITHUB WITH ACCESS KEYS IN CODE

ACCESS_KEY = os.getenv("ACCESS_KEY")
SECRET_KEY = os.getenv("SECRET_KEY")
AWS_REGION = os.getenv("AWS_REGION")


client = boto3.client(
    's3',
    aws_access_key_id=ACCESS_KEY,
    aws_secret_access_key=SECRET_KEY,
    region_name=AWS_REGION
)
    # s3 = boto3.resource(
    #     's3',
    #     aws_access_key_id=ACCESS_KEY,
    #     aws_secret_access_key=SECRET_KEY,
    #     region_name=AWS_REGION
    # )
    # Currently not working!!!

def delete_photo_from_s3(photo_url):
    if not photo_url:
        return
    url_parts = photo_url.split("/")

    # Get the elements we need from the URL parts
    userName = url_parts[4]
    file_name = url_parts[-1]
    file_name_parts = file_name.split("?")
    fileName = file_name_parts[0]

    # print("User name:", userName)
    # print("File name:", fileName)

    # Create a Boto3 client for S3
    client = boto3.client(
        's3',
        aws_access_key_id=ACCESS_KEY,
        aws_secret_access_key=SECRET_KEY,
        region_name=AWS_REGION
    )
```

```
    # s3 = boto3.resource(
    #     's3',
    #     aws_access_key_id=ACCESS_KEY,
    #     aws_secret_access_key=SECRET_KEY,
    #     region_name=AWS_REGION
    # )
    # Currently not working!!!
    try:
        client.delete_object(Bucket="artvisionbucket", Key="profilephoto/"+ userName + "/" +fileName)
        # s3.Object("artvisionbucket", "profilephoto/" + filename).delete()
        print(f"Photo {fileName} deleted from S3 bucket")
    except Exception as e:
        print(f"Error deleting photo {fileName} from S3 bucket: {e}")


def delete_artwork_from_s3(artwork_url):
    if not artwork_url:
        return

    url_parts = artwork_url.split("/")

    # Get the elements we need from the URL parts
    userName = url_parts[4]
    file_name = url_parts[-1]
    file_name_parts = file_name.split("?")
    fileName = file_name_parts[0]
    # Create a Boto3 client for S3
    client = boto3.client(
        's3',
        aws_access_key_id=ACCESS_KEY,
        aws_secret_access_key=SECRET_KEY,
        region_name=AWS_REGION
    )

    try:
        client.delete_object(Bucket="artvisionbucket", Key="artgallery/" + userName +"/"+ fileName)
        print(f"Artwork {fileName} deleted from S3 bucket")
    except Exception as e:
        print(f"Error deleting artwork {fileName} from S3 bucket: {e}")

# Now, run the database operations within the Flask application context
# Reference:
# https://stackoverflow.com/questions/3140779/how-to-delete-files-from-amazon-s3-bucket
with app.app_context():
```

```
# NOTE THIS CODE BELOW WILL RESET THE DATABASE. COMMENT OUT PARTS IF NECCESSARY
    # all_users = User.query.all()
    # for user in all_users:
    #     delete_photo_from_s3(user.profilePhotoLink)
    #     print(user.profilePhotoLink)
    #     # db.session.delete(user)
    #     # print(user.profilePhotoLink)
    #     # print(f"User {user.userName} deleted")

    #     artworks = Artwork.query.filter_by(user_id=user.id).all()
    #     for artwork in artworks:
    #         delete_artwork_from_s3(artwork.url)
    #         print(user.profilePhotoLink)
    #         delete_artwork_from_s3(artwork.url)
    #         db.session.delete(artwork)

    #     # Delete the user
    #     db.session.delete(user)
    #     print(f"User {user.userName} deleted")


    # db.session.commit()
    # List all Users
    all_Users = User.query.all()
    for user in all_Users:
        print(user.userName)
        print(user.email)

    userNameCheck = User.query.filter_by(userName="Ambition2015").first()
    if userNameCheck:
        userNameCheck.userName = "Ambition0516"
        artworks = Artwork.query.filter_by(user_id=1).all()
        for artwork in artworks:
            url = artwork.url
            url_parts = url.split("/")

            url_parts[4] = "Ambition0516"
            newUrl = "/".join(url_parts)
            artwork.url = newUrl
        db.session.commit()
    # List all Art
    # all_Art = Artwork.query.all()
    # for art in all_Art:
    #     print(art.url)
    # for art in all_Art:
    #     print(art.url)

    # new_User = User('John', 'James','j@gmail.com','JJsmoove','jjsmoove123','I like basketball','profile_photo.jpeg')
    # db.session.add(new_User)
    # db.session.commit()

    # # List all Users
    # all_Users = User.query.all()
    # print(all_Users)

    # Select the User by id
    # first_User = User.query.get(1)
    # if first_User:
    #     print(first_User.userName)
    #     print(first_User.profilePhotoLink)

    # # Update
    # first_User = User.query.get(1)
    # first_User.lastName = "Charles"
    # db.session.add(first_student)
    # db.session.commit()

    # # delete
    # second_User = User.query.get(2)
    # db.session.delete(second_User)
    # db.session.commit()


    # new_User = User('Obie', 'C','o@gmail.com','Obie2023','Ambition2023','I like basketball','image/profile_photo.jpeg')
    # db.session.add(new_User)
    # db.session.commit()
    # print(new_User.id)
    # print(new_User.userName)
    # print(new_User.password)
    # print(new_User.profilePhotoLink)

# NOTE CODE BELOW IS FOR DELETING A SINGLE USER
    # user = User.query.filter_by(userName="Ambition0516").first()

    # if user:
```

- **Main.py**

```python
import os
import re
import datetime
import random
from flask import Flask, render_template, session, redirect, url_for, request, jsonify
from flask_wtf import FlaskForm
from wtforms import (StringField, SubmitField, BooleanField, DateTimeField,
                     RadioField, SelectField, TextAreaField, DecimalField)
from flask_wtf.file import FileField, FileAllowed, FileRequired
from wtforms.validators import DataRequired, Length
from flask_sqlalchemy import SQLAlchemy
from flask_migrate import Migrate
from sqlalchemy import LargeBinary, func, desc
import boto3
from werkzeug.utils import secure_filename
import random
from dotenv import load_dotenv
import requests # need to pip install requests
import json
from json import dumps
from sqlalchemy.orm import class_mapper

load_dotenv()

# ###
# GRAB ACCESS_KEY and SECRET_KEY FROM DISCORD. DO NOT COMMIT TO GITHUB WITH ACCESS KEYS IN CODE
ACCESS_KEY = os.getenv("ACCESS_KEY")
SECRET_KEY = os.getenv("SECRET_KEY")
AWS_REGION = os.getenv("AWS_REGION")


#artvisionbucket

basedir = os.path.abspath(os.path.dirname(__file__))
#__file__ refers to main.py
# abspath -> absolute path -> provides the full directory path
```

```python

basedir = os.path.abspath(os.path.dirname(__file__))
#__file__ refers to main.py
# abspath -> absolute path -> provides the full directory path

# We render templates by importing the render_template function from flask and
# returning an .html file from our view function
app = Flask (__name__)

app.config['SQLALCHEMY_DATABASE_URI']='sqlite:///'+os.path.join(basedir,'data.sqlite')
app.config['SQLALCHEMY_TRAC_MODIFICATIONS']=False

app.config['SECRET_KEY'] = 'oursecretkey'

db = SQLAlchemy(app)
Migrate(app,db)

client = boto3.client(
    's3',
    aws_access_key_id = ACCESS_KEY,
    aws_secret_access_key = SECRET_KEY ,
    region_name=AWS_REGION
    )


def passwordValidation(PWD):
    regexCapLetter = r'[A-Z]'
    regexLowLetter = r'[a-z]'
    regexEndNumber = r'[0-9]$'
    regexList = [regexCapLetter, regexLowLetter, regexEndNumber]
    count = 0
    for regex in range(0,3):
        match = re.search(regexList[regex],PWD)
        if match:
            count+=1
    if count == 3:
```

```python

def passwordValidation(PWD):
    regexCapLetter = r'[A-Z]'
    regexLowLetter = r'[a-z]'
    regexEndNumber = r'[0-9]$'
    regexList = [regexCapLetter, regexLowLetter, regexEndNumber]
    count = 0
    for regex in range(0,3):
        match = re.search(regexList[regex],PWD)
        if match:
            count+=1
    if count == 3:
        return True
    else:
        return False

def delete_photo_from_s3(photo_url):
    if not photo_url:
        return
    url_parts = photo_url.split("/")

    # Get the elements we need from the URL parts
    userName = url_parts[4]
    file_name = url_parts[-1]
    file_name_parts = file_name.split("?")
    fileName = file_name_parts[0]

    # print("User name:", userName)
    # print("File name:", fileName)

    # Create a Boto3 client for S3
    # client = boto3.client(
    #     's3',
    #     aws_access_key_id=ACCESS_KEY,
    #     aws_secret_access_key=SECRET_KEY,
```

```python
def delete_photo_from_s3(photo_url):
    if not photo_url:
        return
    url_parts = photo_url.split("/")

    # Get the elements we need from the URL parts
    userName = url_parts[4]
    file_name = url_parts[-1]
    file_name_parts = file_name.split("?")
    fileName = file_name_parts[0]

    # print("User name:", userName)
    # print("File name:", fileName)

    # Create a Boto3 client for S3
    # client = boto3.client(
    #     's3',
    #     aws_access_key_id=ACCESS_KEY,
    #     aws_secret_access_key=SECRET_KEY,
    #     region_name=AWS_REGION
    # )

    # s3 = boto3.resource(
    #     's3',
    #     aws_access_key_id=ACCESS_KEY,
    #     aws_secret_access_key=SECRET_KEY,
    #     region_name=AWS_REGION
    # )
    # Currently not working!!!
    try:
        client.delete_object(Bucket="artvisionbucket", Key="profilephoto/"+ userName + "/" +fileName)
        # s3.Object("artvisionbucket", "profilephoto/" + filename).delete()
        print(f"Photo {fileName} deleted from S3 bucket")
    except Exception as e:
        print(f"Error deleting photo {fileName} from S3 bucket: {e}")
```

```
    # Currently not working!!!
    try:
        client.delete_object(Bucket="artvisionbucket", Key="profilephoto/"+ userName + "/" +fileName)
        # s3.Object("artvisionbucket", "profilephoto/" + filename).delete()
        print(f"Photo {fileName} deleted from S3 bucket")
    except Exception as e:
        print(f"Error deleting photo {fileName} from S3 bucket: {e}")


def delete_artwork_from_s3(artwork_url):
    if not artwork_url:
        return

    url_parts = artwork_url.split("/")

    # Get the elements we need from the URL parts
    userName = url_parts[4]
    file_name = url_parts[-1]
    file_name_parts = file_name.split("?")
    fileName = file_name_parts[0]
    # Create a Boto3 client for S3
    # client = boto3.client(
    #     's3',
    #     aws_access_key_id=ACCESS_KEY,
    #     aws_secret_access_key=SECRET_KEY,
    #     region_name=AWS_REGION
    # )

    try:
        client.delete_object(Bucket="artvisionbucket", Key="artgallery/" + userName +"/"+ fileName)
        print(f"Artwork {fileName} deleted from S3 bucket")
    except Exception as e:
        print(f"Error deleting artwork {fileName} from S3 bucket: {e}")
```

```
class LoginForm(FlaskForm):
    userIdentity = StringField(validators = [DataRequired()])
    password = StringField(validators = [DataRequired()])
    submit = SubmitField('Login')

class RegistrationForm(FlaskForm):
    name = StringField(validators = [DataRequired()])
    email = StringField(validators = [DataRequired()])
    password = StringField(validators = [DataRequired()])
    confirmPassword = StringField(validators = [DataRequired()])
    userName = StringField(validators = [DataRequired()])
    #profilePhoto = FileField('Profile Photo', validators=[FileAllowed(['jpg', 'jpeg', 'png']), FileSize(max_size=2 * 1024 * 1024, message='No photos larger than 2MB.'),
    bio = TextAreaField()
    pronouns = SelectField('Choose your pronouns', choices=[('option1', 'she/her'),('option2', 'he/him'), ('option3', 'they/them'), ('option4', 'she/they'), ('option5',
    title = SelectField('Choose a title', choices=[('title1', 'Professional'), ('title2', 'Student'), ('title3', 'Hobbyist')])
    submit = SubmitField('Sign Up')

#Upload File Form
class UploadForm(FlaskForm):
    title = StringField('Title', validators = [DataRequired(), Length(max=150)])
    fileInput = FileField('Upload File', validators = [FileRequired(), FileAllowed(['png', 'jpg', 'jpeg', 'gif'])])
    description = TextAreaField('Artwork Description', validators = [DataRequired(), Length(max=400)])
    category = SelectField('Select artwork category', validators = [DataRequired()], choices=[('option1', 'Traditional Art'),('option2', 'Digital Art'), ('option3', 'Mix
    saveDraft = SubmitField('Save Draft')
    submit = SubmitField('Submit')


#Selling Form inherits from UploadForm
class SellingForm(UploadForm):
    price = DecimalField('Price', validators=[DataRequired()])
```

```python
class CommentForm(FlaskForm):
    text = StringField()
    submit = SubmitField("Comment")


def serialize(model):
    columns = [c.key for c in class_mapper(model.__class__).columns]
    return dict((c, getattr(model, c)) for c in columns)

class User(db.Model):
    __tablename__="users"

    id = db.Column(db.Integer, primary_key = True)
    name = db.Column(db.Text)
    email = db.Column(db.Text)
    userName = db.Column(db.Text)
    password = db.Column(db.Text)
    bio = db.Column(db.Text)
    profilePhotoLink = db.Column(db.Text)
    pronouns = db.Column(db.Text)
    title = db.Column(db.Text)
    registrationDate = db.Column(db.DateTime)

    def __init__(self, name, email, userName, password, bio, profilePhotoLink, pronouns, title, registrationDate):
        self.name = name
        self.email = email
        self.userName = userName
        self.password = password
        self.bio = bio
        self.profilePhotoLink = profilePhotoLink
        self.pronouns = pronouns
        self.title = title
        self.registrationDate = registrationDate
```

```python
class Artwork(db.Model):
    __tablename__="artworks"

    id = db.Column(db.Integer, primary_key = True)
    title = db.Column(db.String(80), nullable = False)
    description = db.Column(db.String(120), nullable = False)
    category = db.Column(db.String(32))
    price = db.Column(db.Float, nullable = True)
    status = db.Column(db.String(20), nullable = True)
    url = db.Column(db.Text)
    artist = db.Column(db.String(32))
    user_id = db.Column(db.Integer, db.ForeignKey('users.id'), nullable = False)
    uploadDate = db.Column(db.DateTime)
    shop_item = db.Column(db.Boolean, default=False)


    def __init__(self, title, description, category, price, status, url, artist, user_id, uploadDate, shop_item):
        self.title = title
        self.description = description
        self.category = category
        self.price = price
        self.status = status
        self.url = url
        self.artist = artist
        self.user_id = user_id
        self.uploadDate = uploadDate
        self.shop_item = shop_item

    def __repr__(self):
        return f"<Artwork {self.title}>"
```

```python
class Comment(db.Model):
    __tablename__="comments"
    id = db.Column(db.Integer, primary_key=True)
    artwork_id = db.Column(db.Integer)
    text = db.Column(db.String(140))
    profile_pic = db.Column(db.Text)
    author = db.Column(db.String(32))
    author_id = db.Column(db.Text)
    timestamp = db.Column(db.DateTime(), index=True)

    def __init__(self, artwork_id, text, profile_pic, author, author_id, timestamp):
        self.artwork_id = artwork_id
        self.text = text
        self.profile_pic = profile_pic
        self.author = author
        self.author_id = author_id
        self.timestamp = timestamp

    def __repr__(self):
        return f"<Comment {self.text}>"

with app.app_context():
    # Create the tables (if not already created)
    db.create_all()

# success = False # Login variable
```

```python
@app.route('/loginPage', methods = ['GET', 'POST'])
def loginPage():
    form = LoginForm()
    if form.validate_on_submit():
        userIdentity = form.userIdentity.data
        passwordInput = form.password.data

        user = User.query.filter((User.userName==userIdentity) | (User.email==userIdentity)).first()
        if user and user.password==passwordInput:
            # global success
            # success = True
            session['logged_in'] = True
            session['user_id'] = user.id
            session['cart'] = {}
            return redirect(url_for('userProfile', user_id=user.id)) #User's home page
        else:
            error = "Incorrect Login Info"

    all_art = []

    all_art_objects = []
    all_user_objects = []

    for art in Artwork.query.all():
        all_art.append(art.url)
        all_art_objects.append(serialize(art))

    random.shuffle(all_art)

    for user in User.query.all():
        all_user_objects.append(serialize(user))

    return render_template ('loginPage.html', form=form, error=error if 'error' in locals() else None, all_art=all_art, all_art_objects=all_art_objects, all_user_objects
```

```python
@app.route("/register")
def register():
    all_art = []

    all_art_objects = []
    all_user_objects = []

    for art in Artwork.query.all():
        all_art.append(art.url)
        all_art_objects.append(serialize(art))

    for user in User.query.all():
        all_user_objects.append(serialize(user))

    random.shuffle(all_art)

    return render_template("register.html", all_art=all_art, all_art_objects=all_art_objects, all_user_objects=all_user_objects)

# Adds
@app.route("/add", methods = ["POST"])
def add():
    all_art = []

    all_art_objects = []
    all_user_objects = []


    for art in Artwork.query.all():
        all_art.append(art.url)
        all_art_objects.append(serialize(art))

    for user in User.query.all():
        all_user_objects.append(serialize(user))

    random.shuffle(all_art)

    genericPhotoLink = 'image/profile_photo.jpeg'
```

```python
    genericPhotoLink = 'image/profile_photo.jpeg'

    name = request.form.get("name")
    email = request.form.get("email")
    username = request.form.get("username")
    password = request.form.get("password")
    confirmpassword = request.form.get("confirmpassword")
    bio = request.form.get("bio")
    f = request.files["file"]
    pronouns = request.form.get("pronouns")
    title = request.form.get("title")
    # filename = f.filename.split("\\")[-1]
    filename = f"{username}profilepicture.{f.filename.split('.')[-1]}"

    emailCheck = User.query.filter_by(email=email).first()
    if emailCheck: # If email already exists in database
        error = "The email you entered is already taken."
        return render_template ('register.html', error=error, all_art=all_art, all_art_objects=all_art_objects, all_user_objects=all_user_objects)

    userNameCheck = User.query.filter_by(userName=username).first()
    if userNameCheck: # If email already exists in database
        error = "The username you entered is already taken."
        return render_template ('register.html', error=error, all_art=all_art, all_art_objects=all_art_objects, all_user_objects=all_user_objects)

    if not passwordValidation(password):
        error = "Password must contain at least one capital letter, one lowercase letter, and end with a number."
        return render_template('register.html', error=error, all_art=all_art, all_art_objects=all_art_objects, all_user_objects=all_user_objects)
```

```python
    userNameCheck = User.query.filter_by(userName=username).first()
    if userNameCheck: # If email already exists in database
        error = "The username you entered is already taken."
        return render_template ('register.html', error=error, all_art=all_art, all_art_objects=all_art_objects, all_user_objects=all_user_objects)

    if not passwordValidation(password):
        error = "Password must contain at least one capital letter, one lowercase letter, and end with a number."
        return render_template('register.html', error=error, all_art=all_art, all_art_objects=all_art_objects, all_user_objects=all_user_objects)

    if password != confirmpassword:
        error = "Passwords do not match."
        return render_template('register.html', error=error, all_art=all_art, all_art_objects=all_art_objects, all_user_objects=all_user_objects)

    # if ' ' in filename:
    #     error = "photo name cannot contain a space."
    #     return render_template('register.html', error=error)

    f.save(secure_filename(filename))

    # client = boto3.client(
    #     's3',
    #     aws_access_key_id = ACCESS_KEY,
    #     aws_secret_access_key = SECRET_KEY ,
    #     region_name=AWS_REGION
    #     )
```

```python
    if password != confirmpassword:
        error = "Passwords do not match."
        return render_template('register.html', error=error, all_art=all_art, all_art_objects=all_art_objects, all_user_objects=all_user_objects)

    # if ' ' in filename:
    #     error = "photo name cannot contain a space."
    #     return render_template('register.html', error=error)

    f.save(secure_filename(filename))

    # client = boto3.client(
    #     's3',
    #     aws_access_key_id = ACCESS_KEY,
    #     aws_secret_access_key = SECRET_KEY ,
    #     region_name=AWS_REGION
    #     )
    client.upload_file(filename,"artvisionbucket", "profilephoto/"+ username + "/" +filename)
    # presigned_url = client.generate_presigned_url("get_object",
    #     Params={
    #         "Bucket":"artvisionbucket",
    #         "Key":"profilephoto/"+ username + "/" +filename
    #     },
    #     )
    bucket_name = "artvisionbucket"
    s3_key = "profilephoto/" + username + "/" + filename
    url = f"https://{bucket_name}.s3.{AWS_REGION}.amazonaws.com/{s3_key}"
    os.remove(filename)

    newUser = User(name=name, email=email,
        userName=username, password=password, bio=bio, profilePhotoLink=url, pronouns=pronouns, title=title, registrationDate=datetime.date.today())

    db.session.add(newUser)
    db.session.commit()
    return redirect('loginPage')
    # return redirect("/")


@app.route('/user/<int:user_id>')
def userProfile(user_id):
    if session['logged_in'] == True and 'user_id' in session and session['user_id'] == user_id: # 1st person profile visit
        user = User.query.get(user_id)
        artworks = Artwork.query.filter_by(user_id=user_id).all()

        return render_template ('userProfile.html', user=user, currentUser=user, isUsersProfile=True, artworks=artworks)
```

```python
@app.route('/user/<int:user_id>')
def userProfile(user_id):
    if session['logged_in'] == True and 'user_id' in session and session['user_id'] == user_id: # 1st person profile visit
        user = User.query.get(user_id)
        artworks = Artwork.query.filter_by(user_id=user_id).all()

        return render_template ('userProfile.html', user=user, currentUser=user, isUsersProfile=True, artworks=artworks)

    elif session['logged_in'] == True and 'user_id' in session and session['user_id'] != user_id:
        # 3rd person profile visit
        currentUser = User.query.get(session['user_id'])
        user = User.query.get(user_id)
        artworks = Artwork.query.filter_by(user_id=user_id).all()

        return render_template('userProfile.html', user=user, currentUser=currentUser, isUsersProfile=False, artworks=artworks)

    else:
        currentUser=None
        user = User.query.get(user_id)
        artworks = Artwork.query.filter_by(user_id=user_id).all()
        return render_template('userProfile.html', user=user, currentUser=currentUser, isUsersProfile=False, artworks=artworks)

    # else:
    #     return redirect(url_for('error404'))


@app.route('/explore', methods=["GET", "POST"])
def explore():
    artworks = Artwork.query.all()
    random.shuffle(artworks)

    randomArtwork = random.choice(Artwork.query.all())

    if 'user_id' in session and session['logged_in'] == True:
        currentUser = User.query.get(session['user_id'])
        user = User.query.get(session['user_id'])

        if request.method == "POST":
            search = request.form.get("search_term")
            print(search)
            artworks = Artwork.query.filter((Artwork.title.like("%"+search+"%")) | (Artwork.artist.like("%"+search+"%")) | (Artwork.description.like("%"+search+"%"))).all()

            random.shuffle(artworks)
            return render_template('explore.html', artworks=artworks, currentUser=currentUser, randomArtwork=randomArtwork, user=user, userLoggedIn = True)
```

```python
@app.route('/explore', methods=["GET", "POST"])
def explore():
    artworks = Artwork.query.all()
    random.shuffle(artworks)

    randomArtwork = random.choice(Artwork.query.all())

    if 'user_id' in session and session['logged_in'] == True:
        currentUser = User.query.get(session['user_id'])
        user = User.query.get(session['user_id'])

        if request.method == "POST":
            search = request.form.get("search_term")
            print(search)
            artworks = Artwork.query.filter((Artwork.title.like("%"+search+"%")) | (Artwork.artist.like("%"+search+"%")) | (Artwork.description.like("%"+search+"%"))).all()

            random.shuffle(artworks)
            return render_template('explore.html', artworks=artworks, currentUser=currentUser, randomArtwork=randomArtwork, user=user, userLoggedIn = True)

        else:
            return render_template('explore.html', artworks=artworks, currentUser=currentUser, randomArtwork=randomArtwork, user=user, userLoggedIn = True)
    else:
        if request.method == "POST":
            search = request.form.get("search_term")
            random.shuffle(artworks)
            return render_template('explore.html', artworks=artworks, randomArtwork=randomArtwork, userLoggedIn = False)
        else:
            return render_template('explore.html', artworks=artworks, randomArtwork=randomArtwork, userLoggedIn = False)
```

```python
#shop page
@app.route('/shop')
def shop():
    artworks = Artwork.query.filter_by(shop_item=True).all()
    random.shuffle(artworks)
    if 'user_id' in session and session['logged_in'] == True:
        currentUser = User.query.get(session['user_id'])
        user = User.query.get(session['user_id'])
        return render_template('shop.html', artworks=artworks, user=user, userLoggedIn = True, currentUser=currentUser)

@app.route('/artwork/<int:artwork_id>', methods=["GET", "POST"])
def artworkDetails(artwork_id):
    form=CommentForm()
    artwork = Artwork.query.get(artwork_id)
    print(artwork_id)
    artist = User.query.get(artwork.user_id)

    if 'user_id' not in session:  # add this line
        # handle the case when the user is not logged in
        user = None  # or whatever is appropriate in your case
        userLoggedIn = False  # add this line

    else:
        user = User.query.get(session['user_id'])
        userLoggedIn = True   # add this line

    comments = Comment.query.filter_by(artwork_id = artwork.id).all()
    commentsCount = db.session.execute(Comment.query.filter_by(artwork_id=artwork.id).statement.with_only_columns(func.count()).order_by(None)).scalar()

    if not artwork:
        return render_template('error.html', message='Artwork not found.')

    if request.method == 'POST':
        if user:  # add this line
            text = request.form.get("text")
            newComment = Comment(artwork_id=artwork_id, text=text, author=user.userName, profile_pic=user.profilePhotoLink, author_id=user.id, timestamp=datetime.date.today())

            db.session.add(newComment)
            db.session.commit()
            return redirect(url_for('artworkDetails', artwork=artwork, user=user, currentUser=user, artist=artist, form=form, comments=comments, commentsCount=commentsCount, artwork_id=artwork_id, userLogge

    return render_template('artworkDetails.html', artwork=artwork, user=user, currentUser=user, artist=artist, form=form, comments=comments, commentsCount=commentsCount, userLoggedIn=userLoggedIn)
```

```python
#shop page
@app.route('/shop')
def shop():
    artworks = Artwork.query.filter_by(shop_item=True).all()
    random.shuffle(artworks)
    if 'user_id' in session and session['logged_in'] == True:
        currentUser = User.query.get(session['user_id'])
        user = User.query.get(session['user_id'])
        return render_template('shop.html', artworks=artworks, user=user, userLoggedIn = True, currentUser=currentUser)

@app.route('/artwork/<int:artwork_id>', methods=["GET", "POST"])
def artworkDetails(artwork_id):
    form=CommentForm()
    artwork = Artwork.query.get(artwork_id)
    print(artwork_id)
    artist = User.query.get(artwork.user_id)

    if 'user_id' not in session:  # add this line
        # handle the case when the user is not logged in
        user = None  # or whatever is appropriate in your case
        userLoggedIn = False  # add this line

    else:
        user = User.query.get(session['user_id'])
        userLoggedIn = True   # add this line

    comments = Comment.query.filter_by(artwork_id = artwork.id).all()
    commentsCount = db.session.execute(Comment.query.filter_by(artwork_id=artwork.id).statement.with_only_columns(func.count()).order_by(None)).scalar()

    if not artwork:
        return render_template('error.html', message='Artwork not found.')

    if request.method == 'POST':
        if user:  # add this line
            text = request.form.get("text")
            newComment = Comment(artwork_id=artwork_id, text=text, author=user.userName, profile_pic=user.profilePhotoLink, author_id=user.id, timestamp=datetime.date.today())

            db.session.add(newComment)
            db.session.commit()
            return redirect(url_for('artworkDetails', artwork=artwork, user=user, currentUser=user, artist=artist, form=form, comments=comments, commentsCount=commentsCount, artwork_id=artwork_id, userLoggedIn=userLoggedIn))

    return render_template('artworkDetails.html', artwork=artwork, user=user, currentUser=user, artist=artist, form=form, comments=comments, commentsCount=commentsCount, userLoggedIn=userLoggedIn)

#Upload File Page
@app.route('/uploadPage/<int:user_id>', methods = ['GET', 'POST'])
def uploadPage(user_id):
    user = User.query.get(user_id)
    return render_template('upload.html', user=user, user_id=user_id, currentUser=user)
```

```python
#Upload File Page
@app.route('/uploadPage/<int:user_id>', methods = ['GET', 'POST'])
def uploadPage(user_id):
    user = User.query.get(user_id)
    return render_template('upload.html', user=user, user_id=user_id, currentUser=user)

#Upload Comission Page
@app.route('/sellingPage/<int:user_id>', methods = ['GET', 'POST'])
def sellingPage(user_id):
    user = User.query.get(user_id)
    return render_template('selling.html', user=user, user_id=user_id, currentUser=user)

#Checkout Page
@app.route('/purchasePage/<int:user_id>', methods = ['GET', 'POST'])
def purchasePage(user_id):
    user = User.query.get(user_id)
    total = calculateCartTotal(session['cart'])
    return render_template('purchase.html', user=user, user_id=user_id, currentUser=user, cart=session['cart'], total=total)

@app.route ('/thankyou', methods = ["POST"])
def thankyou():
    user_id = session['user_id']
    if 'cart' in session and session['cart']:
        user = User.query.get(user_id)
        session['cart'] = {} #empty cart
        message = "Thank you for your order"
        return render_template('thankyou.html', message=message, currentUser=user, user=user)

    return redirect(url_for('purchasePage', user_id=user_id))


@app.route("/addArt/<int:user_id>", methods = ["POST"])
def addArt(user_id):
    user = User.query.get(user_id)
    artist = user.userName

    title = request.form.get("title")
    description = request.form.get("description")
    category = request.form.get("category")
    price = request.form.get("price")
    status = request.form.get("status")
    shop_item = "price" in request.form

    f = request.files["file"]
    filename = f.filename.split("\\")[-1]
    f.save(secure_filename(filename))

    # client = boto3.client(
    #     's3',
    #     aws_access_key_id= ACCESS_KEY,
    #     aws_secret_access_key= SECRET_KEY,
    #     region_name=AWS_REGION
    # )
```

```python
@app.route("/addArt/<int:user_id>", methods = ["POST"])
def addArt(user_id):
    user = User.query.get(user_id)
    artist = user.userName

    title = request.form.get("title")
    description = request.form.get("description")
    category = request.form.get("category")
    price = request.form.get("price")
    status = request.form.get("status")
    shop_item = "price" in request.form

    f = request.files["file"]
    filename = f.filename.split("\\")[-1]
    f.save(secure_filename(filename))

    # client = boto3.client(
    #     's3',
    #     aws_access_key_id= ACCESS_KEY,
    #     aws_secret_access_key= SECRET_KEY,
    #     region_name=AWS_REGION
    # )

    # Upload the file to S3 bucket
    client.upload_file(filename, "artvisionbucket", "artgallery/" +str(user.userName)+"/"+ filename)
    # presigned_url = client.generate_presigned_url("get_object",
    #     Params={
    #         "Bucket":"artvisionbucket",
    #         "Key":"artgallery/" +str(user.userName)+"/"+ filename
    #     },
    #     )

    bucket_name = "artvisionbucket"
    s3_key = f"artgallery/{user.userName}/{filename}"
    url = f"https://{bucket_name}.s3.{AWS_REGION}.amazonaws.com/{s3_key}"

    os.remove(filename)
    newArt = Artwork(title =title, description=description, category=category, price=price, status=status, url=url, user_id=user_id, artist=artist, uploadDate=datetime.date.today(), shop_item=shop_item)
    db.session.add(newArt)
    db.session.commit()
    return redirect(url_for('userProfile', user_id=user_id))
```

```python
#add artwork to cart
@app.route("/addCart/<int:artwork_id>", methods = ["POST"])
def addCart(artwork_id):
    artwork = Artwork.query.get(artwork_id)
    quantity = int(request.form.get('quantity'))
    if not artwork:
        return render_template('error.html', message='Artwork not found.')
    cart = session['cart']

    if str(artwork.id) not in cart:
        cart[str(artwork.id)] = {'quantity': 0,'price': artwork.price,'image_url': artwork.url,'title' : artwork.title,'id' : artwork.id}

    cart[str(artwork.id)]['quantity'] += quantity
    session['cart'] = cart
    return redirect(url_for('artworkDetails', artwork_id=artwork_id))

#update cart quantity
@app.route('/updateCart', methods=['POST'])
def updateCart():
    data = request.get_json()
    artwork_id = data.get('artwork_id')
    quantity = int(data.get('quantity', 0))
    cart = session['cart']
    cart[artwork_id]['quantity'] = quantity
    session['cart'] = cart
    return jsonify(success=True)

def calculateCartTotal(cart):
    total = 0.00
    for attr in cart.values():
        total += attr['quantity'] * attr['price']
    return total


#delete item from cart
@app.route('/deleteCartItem/<int:artwork_id>')
def deleteCartItem(artwork_id):
    cart = session.get('cart', {})
    if str(artwork_id) in cart:
        del cart[str(artwork_id)]
        session['cart'] = cart
    return redirect(url_for('purchasePage', user_id=session['user_id']))




#Upload File Page
@app.route('/deletePage/<int:user_id>', methods = ['GET', 'POST'])
def deletePage(user_id):
    user = User.query.get(user_id)
    artworks = Artwork.query.filter_by(user_id=user_id).all()
    return render_template('deleteArt.html', user=user, user_id=user_id, artworks=artworks, currentUser=user)
```

```python
#Upload File Page
@app.route('/deletePage/<int:user_id>', methods = ['GET', 'POST'])
def deletePage(user_id):
    user = User.query.get(user_id)
    artworks = Artwork.query.filter_by(user_id=user_id).all()
    return render_template('deleteArt.html', user=user, user_id=user_id, artworks=artworks, currentUser=user)

@app.route("/deleteArt/<int:user_id>", methods = ["GET","POST"])
def deleteArt(user_id):
    user = User.query.get(user_id)
    artworksSelected = request.form.getlist("artworkToDelete")


    if request.method == 'POST':
        for artwork_id in artworksSelected:
            artwork = Artwork.query.get(artwork_id)
            filename = (artwork.url).partition(user.userName)[2]
            if artwork:
                client = boto3.client(
                    's3',
                    aws_access_key_id= ACCESS_KEY,
                    aws_secret_access_key= SECRET_KEY,
                    region_name=AWS_REGION
                )
                client.delete_object(Bucket="artvisionbucket", Key="artgallery/" + user.userName +"/"+ filename)
                db.session.delete(artwork)

                db.session.commit()
        return redirect(url_for('userProfile', user_id=user_id, currentUser=user))
    else:
        return (redirect(url_for('explore')))
    # Need query to find and delete art from database


@app.route('/error404')
def error404():
    return render_template ('error404.html')

@app.route('/logout')
def logout():
    # session.pop('logged_in', None)
    session['logged_in'] = False
    session.pop('user_id', None)
    return redirect(url_for('index'))
```

```python

@app.route('/deleteAccount')
def deleteAccount():
    user_id = session['user_id']
    user = User.query.get(user_id)
    artworks = Artwork.query.filter_by(user_id=user_id).all()
    comments = Comment.query.filter_by(author_id=user.id).all()

    for artwork in artworks:
        delete_artwork_from_s3(artwork.url)
        db.session.delete(artwork)

    for comment in comments:
        db.session.delete(comment)

    db.session.delete(user)
    db.session.commit()

    session['logged_in'] = False
    session.pop('user_id', None)

    return redirect(url_for('explore'))


@app.route('/deleteComment/<int:comment_id>/<int:artwork_id>')
def deleteComment(comment_id, artwork_id):
    form=CommentForm()
    artwork = Artwork.query.get(artwork_id)
    artist = User.query.get(artwork.user_id)
    user = User.query.get(session['user_id'])
    Comment.query.filter_by(id=comment_id).delete()
    comments = Comment.query.filter(artwork_id==artwork.id).all()

    db.session.commit()
    return redirect(url_for('artworkDetails', artwork=artwork, user=user, currentUser=user, artist=artist, form=form, comments=comments, artwork_id=artwork_id))

#Routes for user galleries
@app.route('/<option>/<int:user_id>')
def gallery(option, user_id):
    user = User.query.get(user_id)
    if option == 'gallery':
        artworks = Artwork.query.filter_by(user_id=user_id, shop_item=False).all()
    elif option == 'shop':
        artworks = Artwork.query.filter_by(user_id=user_id, shop_item=True).all()
    serialized_artworks = {artwork.id: {"id": artwork.id, "title": artwork.title, "price": artwork.price, "url": artwork.url} for artwork in artworks}
    return jsonify(serialized_artworks)
```

```python
#Routes for explore
@app.route('/category/<category>')
def exploreCategories(category):
    artworks = Artwork.query.filter_by(category=category).all()
    serialized_artworks = {artwork.id: {"id": artwork.id, "title": artwork.title, "price": artwork.price, "url": artwork.url, "user_id" : artwork.user_id, "artist" : artwork.artist} for artwork in artworks}
    return jsonify(serialized_artworks)



@app.route('/editPage/<int:user_id>', methods = ['GET', 'POST'])
def editPage(user_id):
    user = User.query.get(user_id)
    artworks = Artwork.query.filter_by(user_id=user_id).all()
    return render_template('editAccount.html', user=user, user_id=user_id, artworks=artworks, currentUser=user)

@app.route("/editAccount/<int:user_id>", methods = ["GET","POST"])
def editAccount(user_id):
    user = User.query.get(user_id)
    artworks = Artwork.query.filter_by(user_id=user_id).all()
    update = False

    if request.method == "POST":
        ##UPDATE USER PROFILE INFORMATION
        if request.form.get('profile-change') == "Update":
            newUsername = request.form.get("newUsername")
            newDisplayName = request.form.get("newDisplayName")
            newPronouns = request.form.get("newPronouns")
            newTitle = request.form.get("newTitle")
            newBio = request.form.get("newBio")

            if newDisplayName:
                user.name = newDisplayName
                update = True

            if newPronouns:
                user.pronouns = newPronouns
                update = True

            if newTitle:
                user.title = newTitle
                update = True

            if newBio:
                user.bio = newBio
                update = True

            userNameCheck = User.query.filter_by(userName=newUsername).first()
            # print(userNameCheck)
            currentUsername = user.userName
            if newUsername and userNameCheck is None:
                user.userName = newUsername
```

```python
            userNameCheck = User.query.filter_by(userName=newUsername).first()
            # print(userNameCheck)
            currentUsername = user.userName
            if newUsername and userNameCheck is None:
                user.userName = newUsername

                # setting up client
                # client = boto3.client(
                #     's3',
                #     aws_access_key_id = ACCESS_KEY,
                #     aws_secret_access_key = SECRET_KEY ,
                #     region_name=AWS_REGION
                #     )

                # transferring profile photo to new userName

                # Saving profile photo
                response = requests.get(user.profilePhotoLink)
                url_parts = user.profilePhotoLink.split("/")
                filename = url_parts[-1]
```

```
            #     aws_secret_access_key = SECRET_KEY ,
            #     region_name=AWS_REGION
            #     )

            # transferring profile photo to new userName

            # Saving profile photo
            response = requests.get(user.profilePhotoLink)
            url_parts = user.profilePhotoLink.split("/")
            filename = url_parts[-1]

            with open(filename, 'wb') as f:
                f.write(response.content)

            # uploading profile photo
            client.upload_file(filename,"artvisionbucket", "profilephoto/" + user.userName + "/" + filename)

            bucket_name = "artvisionbucket"
            s3_key = "profilephoto/" + user.userName + "/" + filename
            url = f"https://{bucket_name}.s3.{AWS_REGION}.amazonaws.com/{s3_key}"
            user.profilePhotoLink = url
            client.delete_object(Bucket="artvisionbucket", Key="profilephoto/" + currentUsername +"/"+ filename)
            os.remove(filename)

            for artwork in artworks:
                url = artwork.url
                url_parts = url.split("/")

                url_parts[4] = newUsername
                newUrl = "/".join(url_parts)
                artwork.url = newUrl

                response = requests.get(url)
                filename = url_parts[-1]

                with open(filename, 'wb') as f:
                    f.write(response.content)

                client.upload_file(filename,"artvisionbucket", "artgallery/" + user.userName + "/" + filename)
                bucket_name = "artvisionbucket"
                s3_key = "artgallery/" + user.userName + "/" + filename
                url = f"https://{bucket_name}.s3.{AWS_REGION}.amazonaws.com/{s3_key}"
                client.delete_object(Bucket="artvisionbucket", Key="artgallery/" + currentUsername +"/"+ filename)
                os.remove(filename)
                update = True
        db.session.commit()

    ##UPDATE ACCOUNT INFORMATION
    if request.form.get('account-change') == "Update":
        ##Verification
        user = User.query.filter_by(id=user_id).first()
        passwordInDatabase = user.password
        emailInDatabase = user.email
```

```
    ##UPDATE ACCOUNT INFORMATION
    if request.form.get('account-change') == "Update":
        ##Verification
        user = User.query.filter_by(id=user_id).first()
        passwordInDatabase = user.password
        emailInDatabase = user.email

        currentEmail = request.form.get("currentemail")
        currentPassword = request.form.get("currentpassword")

        ##Proposed changes
        newEmail = request.form.get("newemail")
        confirmNewEmail = request.form.get("confirmnewemail")

        newPassword = request.form.get("newpassword")
        confirmNewPassword = request.form.get("confirmnewpassword")


        ##Error checking
        errors = []
        #check for inputs for current email and current password
        if currentEmail == "":
            error = "Please enter your current email."
            errors.append(error)

        if currentPassword == "":
            error = "Please enter your current password."
            errors.append(error)

        # checks database for email matching current user.
        if currentEmail and currentEmail != emailInDatabase:
            error = "Incorrect email! Please try again."
            errors.append(error)

        # checks database for password matching current user.
        if currentPassword and currentPassword != passwordInDatabase:
            error = "Incorrect password! Please try again."
            errors.append(error)

        # makes sure new email matches.
        if newEmail and newEmail != confirmNewEmail:
            error = "Your new email inputs do not match. Please try again."
            errors.append(error)

        # makes sure new password matches and meet requirements.
        if newPassword and newPassword != confirmNewPassword and passwordValidation(newPassword)== False:
            error = "Your new password inputs do not match and/or do not meet password requirements."
            errors.append(error)

        if (len(errors) > 0):
            return render_template('editAccount.html', user=user, user_id=user_id, artworks=artworks, currentUser=user, errors=errors)
```

```
            ##Error checking
            errors = []
            #check for inputs for current email and current password
            if currentEmail == "":
                error = "Please enter your current email."
                errors.append(error)

            if currentPassword == "":
                error = "Please enter your current password."
                errors.append(error)

            # checks database for email matching current user.
            if currentEmail and currentEmail != emailInDatabase:
                error = "Incorrect email! Please try again."
                errors.append(error)

            # checks database for password matching current user.
            if currentPassword and currentPassword != passwordInDatabase:
                error = "Incorrect password! Please try again."
                errors.append(error)

            # makes sure new email matches.
            if newEmail and newEmail != confirmNewEmail:
                error = "Your new email inputs do not match. Please try again."
                errors.append(error)

            # makes sure new password matches and meet requirements.
            if newPassword and newPassword != confirmNewPassword and passwordValidation(newPassword)== False:
                error = "Your new password inputs do not match and/or do not meet password requirements."
                errors.append(error)

            if (len(errors) > 0):
                return render_template('editAccount.html', user=user, user_id=user_id, artworks=artworks, currentUser=user, errors=errors)

            else:
                if newEmail:
                    user.email = newEmail
                    update = True

                if newPassword:
                    user.password = newPassword
                    update = True

                db.session.commit()

    if update == True:
        message = "Information updated"
    if update == False:
        message = "Information unchanged"
    db.session.commit()
    return redirect(url_for('userProfile', user_id=user_id, message=message))
```

```
            # makes sure new password matches and meet requirements.
            if newPassword and newPassword != confirmNewPassword and passwordValidation(newPassword)== False:
                error = "Your new password inputs do not match and/or do not meet password requirements."
                errors.append(error)

            if (len(errors) > 0):
                return render_template('editAccount.html', user=user, user_id=user_id, artworks=artworks, currentUser=user, errors=errors)

            else:
                if newEmail:
                    user.email = newEmail
                    update = True

                if newPassword:
                    user.password = newPassword
                    update = True

                db.session.commit()

    if update == True:
        message = "Information updated"
    if update == False:
        message = "Information unchanged"
    db.session.commit()
    return redirect(url_for('userProfile', user_id=user_id, message=message))


if __name__ == '__main__':
    app.run(host='0.0.0.0',debug=True)
    # app.run(debug=True)
```

- **Requirements.txt:**

8. Discussion / Future Expansion

Although some minor site features were not implemented before our development deadline, we believe that we have achieved all of our major goals for the development of ArtVision. Users can submit their artwork to their galleries and/or shops, manage their accounts, provide feedback on other users' artwork, purchase prints, and navigate the site's database of artwork through various means.

Ideas for future expansion of ArtVision include giving users the ability to send friend requests and direct messages, creating a comment section or guestbook feature for users' profiles, sending users notifications when they receive comments or favorites on their artwork, bolstering security features (e.g. password hashing, email confirmation, etc.), and exploring more opportunities to utilize AWS and enhance both performance and scalability of our website.

9. Conclusion

ArtVision's successful development is a testament to our team's tenacity and highly collaborative efforts that merged a surprisingly diverse array of skill sets. We have built a practical yet beautiful application with features that truly support our goal of creating a space for art enthusiasts to share and further their craft with like-minded individuals. However, with the understanding that there is always room for improvement, we anticipate adding new features such as those discussed in the previous section to support ArtVision's growth as a vibrant hub of creativity.

10. References

- **W3Schools | HTML <input type="file">**
    - This resource was instrumental to understanding how users can upload files on a website, a crucial functionality for a platform where artists submit artworks.
    - Source: https://www.w3schools.com/tags/att_input_type_file.asp#:~:text=The%20%3Cinput%20type%3D%22file,tag%20for%20best%20accessibility%20practices

- **Stack Overflow | Flask HTML Loops**
    - The guidance from this resource was beneficial for handling data display in our Flask application.
    - Source: https://stackoverflow.com/questions/45167508/flask-template-for-loop-iteration-keyvalue

- **Amazon Web Services | AWS Architecture Center:**

    - This resource was essential in helping us understand and implement cloud-based solutions for our platform, offering numerous well-documented patterns, designs, and best practices.
    - Source: https://aws.amazon.com/architecture/?cards-all.sort-by=item.additionalFields.sortDate&cards-all.sort-order=desc&awsf.content-type=*all&awsf.methodology=*all&awsf.tech-category=*all&awsf.industries=*all&awsf.business-category=*all

- **Pallets Projects**
    - **Flask-SQLAlchemy documentation**
        - This extensive resource was pivotal in our understanding and utilization of the Flask-SQLAlchemy extension, which greatly simplified interaction between Flask and our database.
        - Source: https://flask-sqlalchemy.palletsprojects.com/en/3.0.x/
    - **JavaScript Fetch and JSON**
        - Source: https://flask.palletsprojects.com/en/2.3.x/patterns/javascript/
    - **Jinja Documentation**
        - Source: https://jinja.palletsprojects.com/en/3.1.x/templates/

- **Miguel Grinberg | Implementing User Comments with SQLAlchemy**
    - We chose to implement the basic comment solution (no reply function) illustrated in this resource.
    - Source: https://blog.miguelgrinberg.com/post/implementing-user-comments-with-sqlalchemy

- **Stack Overflow | Multiple submit buttons in form**
    - The answer provided here was used to handle actions for the two update buttons on the Edit Account page.
    - Source: https://stackoverflow.com/questions/65938462/multiple-submit-buttons-in-flask
- **Stack Overflow | SQLAlchemy database models to JSON**
    - This resource was used to understand how to serialize our database models for conversion to JSON. This was necessary to display user-submitted artwork and the corresponding details in the landing page, registration page, and login page's slideshow backgrounds.
    - Source: https://stackoverflow.com/questions/14920080/how-to-create-sqlalchemy-to-json

- **SQLite Viewer**

    - We used this to easily view information in our SQL database.
    - Source: https://inloop.github.io/sqlite-viewer/

- **Python Docs | datetime**
    - Source: https://docs.python.org/3/library/datetime.html

- **ChatGPT**
    - This resource was primarily used to generate ideas for technologies we could use for development of this project as well as quick debugging tips.
    - Our use of this resource meets the Guidelines for Artificial Intelligence Use at UAB that was shared by President Ray Watts.
    - All of our project's code was written directly by our team members. Absolutely no portion of our code was copied from ChatGPT output.