

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**

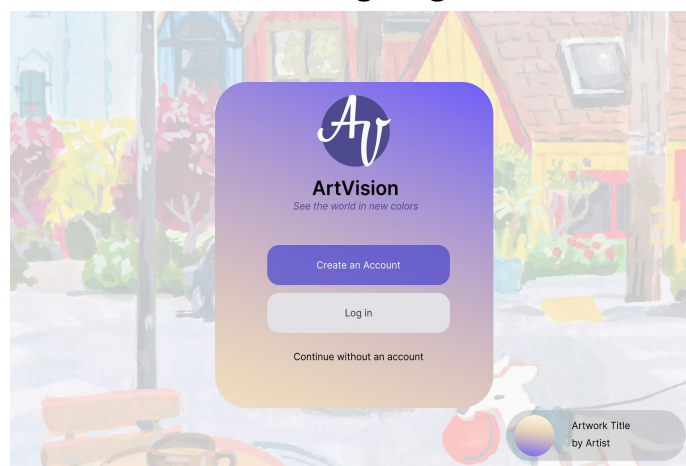
- **There are no admin users at this time** but we do have a page to access important site data.
- User Management
 - View all ArtVision users and their account information
- Artwork Management
 - View all artwork submitted to the site
- View all transaction information

4. Website Mockup

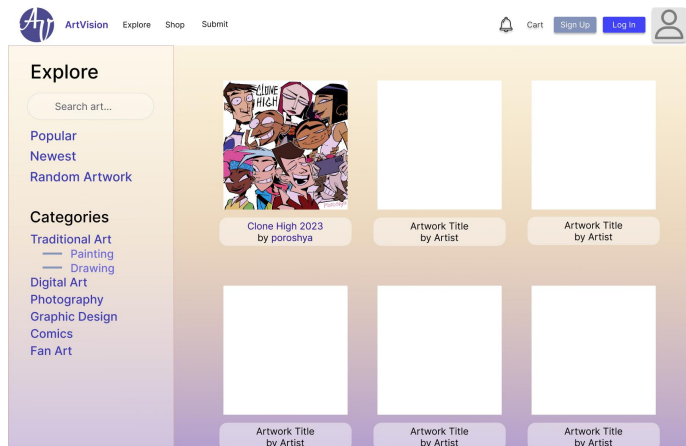
To view designs for each of ArtVision's (initially planned) pages, visit the Figma board: <https://www.figma.com/file/jYv5lE4s6qmNwuADLbjczi/ArtVision?type=design&node-id=0-1&mode=design>

a. Selected Pages:

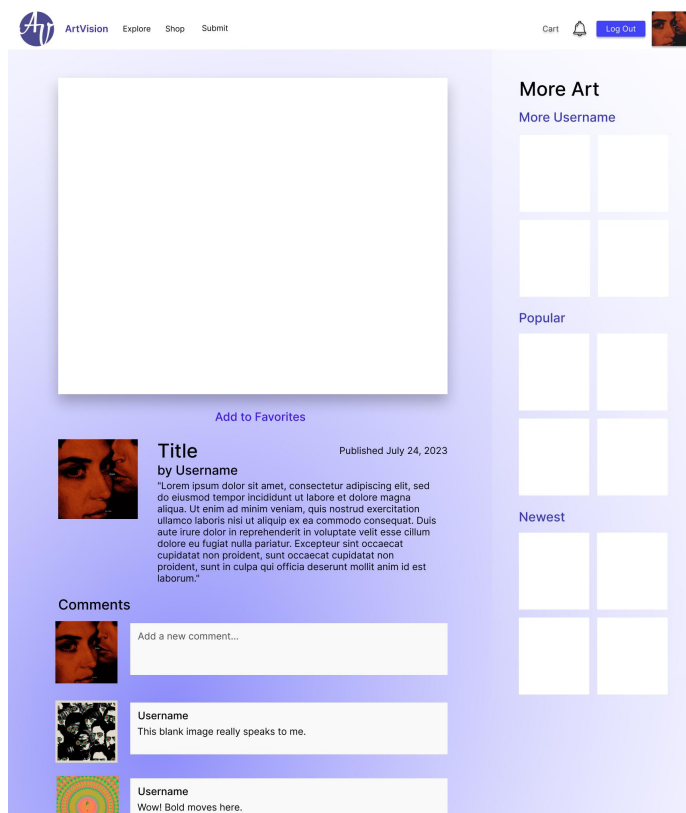
Landing Page



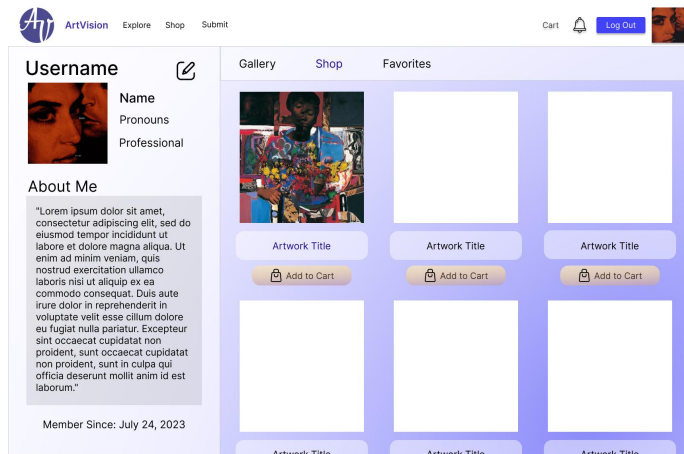
Explore Page



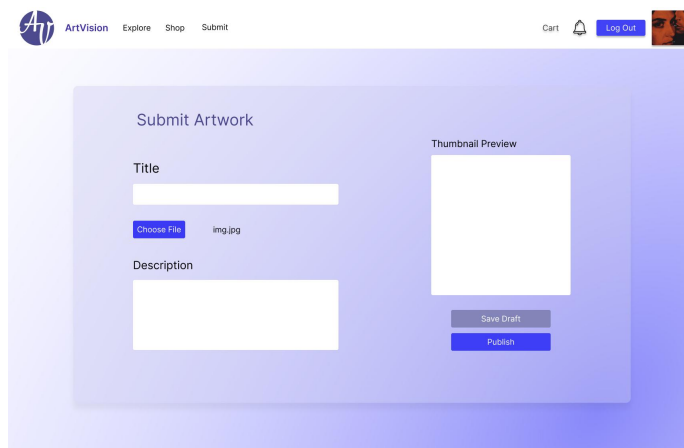
Artwork Detail Page



User Profile Page - Shop Tab



Submit Art Page



Shop Checkout Page

ArtVision Explore Shop Submit

Cart Sign Up Log in

Checkout

Contact Information

E-mail

Phone

Shipping Address

First name Last name

Address

City Country

State Postal Code

Payment Details

Card number

Name on card

Expiration date Security code

Summary

Name \$ \$ Delete

1

Name \$ \$ Delete

1

TOTAL USD \$ \$

Pay

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:**

```

1 import logging
2 from logging.config import fileConfig
3
4 from flask import current_app
5
6 from alembic import context
7
8 # this is the Alembic Config object, which provides
9 # access to the values within the .ini file in use.
10 config = context.config
11
12 # Interpret the config file for Python logging.
13 # This line sets up loggers basically.
14 fileConfig(config.config_file_name)
15 logger = logging.getLogger('alembic.env')
16
17
18 def get_engine():
19     try:
20         # this works with Flask-SQLAlchemy<3 and Alchemical
21         return current_app.extensions['migrate'].db.get_engine()
22     except TypeError:
23         # this works with Flask-SQLAlchemy>=3
24         return current_app.extensions['migrate'].db.engine
25
26
27 def get_engine_url():
28     try:
29         return get_engine().url.render_as_string(hide_password=False).replace(
30             '%', '%%')
31     except AttributeError:
32         return str(get_engine().url).replace('%', '%%')
33
34
35 # add your model's MetaData object here
36 # for 'autogenerate' support
37 # from myapp import mymodel
38 # target_metadata = mymodel.Base.metadata
39 config.set_main_option('sqlalchemy.url', get_engine_url())
40 target_db = current_app.extensions['migrate'].db
41
42 # other values from the config, defined by the needs of env.py,
43 # can be acquired:
44 # my_important_option = config.get_main_option("my_important_option")
45 # ... etc.
46
47
48 def get_metadata():
49     if hasattr(target_db, 'metadatas'):
50         return target_db.metadatas[None]
51     return target_db.metadata
52
53
54 def run_migrations_offline():
55     """Run migrations in 'offline' mode.
56
57     This configures the context with just a URL
58     and not an Engine, though an Engine is acceptable
59     here as well. By skipping the Engine creation
60     we don't even need a DBAPI to be available.
61
62     Calls to context.execute() here emit the given string to the
63     script output.
64
65     """
66     url = config.get_main_option("sqlalchemy.url")
67     context.configure(
68         url=url, target_metadata=get_metadata(), literal_binds=True
69     )
70
71     with context.begin_transaction():
72         context.run_migrations()
73
74
75 def run_migrations_online():
76     """Run migrations in 'online' mode.
77
78     In this scenario we need to create an Engine
79     and associate a connection with the context.
80
81     """
82
83     # this callback is used to prevent an auto-migration from being generated
84     # when there are no changes to the schema
85     # reference: http://alembic.zzzcomputing.com/en/latest/cookbook.html
86     def process_revision_directives(context, revision, directives):
87         if getattr(config.cmd_opts, 'autogenerate', False):
88             script = directives[0]
89             if script.upgrade_ops.is_empty():
90                 directives[:] = []
91                 logger.info('No changes in schema detected.')

```

```

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:

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>{{artwork.title}} by {{artist.userName}}</title>
7   <!-- Add your CSS file link here -->
8   <link rel="stylesheet" href="{{url_for('static', filename='css/styles.css')}}" />
9 </head>
10 <body class="details-page gradient-bg">
11 <!-- Header (You can create a separate header file and include it on all pages) -->
12 <header>
13   <!-- extends "base.html" %>
14   <div>
15     <!-- block content %>
16
17     <!-- Artwork Gallery Section -->
18     <div class="artwork">
19
20       <!-- Display the user's artworks here -->
21       <!-- You can use a loop to generate the artwork thumbnails -->
22       <!-- Example of a single artwork thumbnail -->
23       <div class="full-img">
24         <a href="{{artwork.url}}">
25           
26         </a>
27       </div>
28
29       <div class="info">
30         <a href="{{url_for('userProfile', user_id=artwork.user_id)}}"></a>
31         <div class="detail">
32           <a href="{{artwork.title}}>
33             <p>by <a href="{{url_for('userProfile', user_id=artwork.user_id)}}">{{artist.userName}}</a></p>
34             <p>{{artwork.description}}</p>
35             <p>Published {{artwork.uploadDate.strftime("%B %d, %Y")}}</p>
36           </div>
37         </div>
38
39         <div>
40           <!-- % if artwork.shop_item %>
41           <div>Buy a print</div>
42           <div>{{artwork.price}}</div>
43           <div>{{artwork.status}}</div>
44           <form action="{{url_for('addCart', artwork_id=artwork.id)}}" method="POST"
45             enctype="multipart/form-data">
46             <div>Label for "Quantity" Quantity: <input type="text" value="1" min="1">
47             <div><input type="submit" value="Add to cart">
48           </form>
49         </div>
50       </div>
51     </div>
52

```

```

54 <!-- Comment Section -->
55 <!-- form class="ui form" action="/artwork/{{artwork.id}}" method="post" enctype="multipart/form-data">
56 <div>
57   <input type="text" name="text" placeholder="Add a new comment...">
58   <button class="ui blue button custom-button" type="submit">Comment</button>
59 </form>
60
61 <div comment-section>
62   <!-- % for comment in comments %>
63   <div>
64     <a href="{{url_for('userProfile', user_id=comment.author_id)}}"></a>
65     <p><a href="{{url_for('userProfile', user_id=comment.author_id)}}">{{comment.author}}</a> says: {{comment.text}}</p>
66     <!-- % if user.userName == comment.author %>
67     <a href="{{url_for('deleteComment', comment_id=comment.id, artwork_id=artwork.id)}}" class="ui blue button custom-button">Delete</a>
68     <!-- % endif %>
69   </div>
70 </div>
71
72 <div>
73   <div>Comments</div>
74
75   <form class="ui form" action="/artwork/{{artwork.id}}" method="post" enctype="multipart/form-data">
76     <div>
77       <input type="text" name="text" placeholder="Add a new comment...">
78       <button class="ui blue button custom-button" type="submit">Comment</button>
79     </div>
80
81     <div>
82       <div>
83         <div>
84           <input type="text" name="text" placeholder="Log in to comment...">
85           <a href="{{url_for('loginPage')}}" class="ui blue button custom-button">Log in</a>
86         </div>
87       </div>
88     </div>
89
90   </div>
91 </div>
92
93 <div comment-section>
94   <!-- % for comment in comments %>
95   <div>
96     <a href="{{url_for('userProfile', user_id=comment.author_id)}}"></a>
97     <p><a href="{{url_for('userProfile', user_id=comment.author_id)}}">{{comment.author}}</a> says: {{comment.text}}
98     <!-- % if user and user.userName == comment.author %>
99     <a href="{{url_for('deleteComment', comment_id=comment.id, artwork_id=artwork.id)}}" class="ui blue button custom-button">Delete</a>
100   </div>
101 </div>
102
103 <!-- Footer -->
104
105 <!-- Add your JavaScript file link here if needed -->
106 <script src="{{url_for('static', filename='js/scripts.js')}}">
107 </script>

```

- Base.html:


```

40 <a class="nav-link" href="{url_for(register)}" >>Sign in</a>
41 </li>
42 <li>
43 {% if user %}
44 </li>
45 </li>
46 </li>
47 </li>
48 </li>
49 </li>
50 </li>
51 </li>
52 <li>
53 {% if session.logged_in %}
54 <a class="nav-link" href="{url_for('current_page', width='16' height='16', fill='currentColor' class='bi bi-white-viewbox')}>
55 </a>
56 </li>
57 </li>
58 </li>
59 </li>
60 </li>
61 </li>
62 </li>
63 </li>
64 </li>
65 </li>
66 </li>
67 </li>
68 </li>
69 </li>
70 </li>
71 </li>
72 </li>
73 </li>
74 </li>
75 </li>
76 </li>
77 </li>
78 </li>
79 </li>
80 </li>
81 </li>
82 </li>
83 </li>
84 </li>

```

[illegible]

```

40 <a href="{url_for('galleryPage', user_id=user.id)}" > Sell Your Art! </a> <br>
41 <a href="{url_for('galleryPage', user_id=user.id)}" > Edit Account </a>
42 { % if message %}
43 { % if message %} </p>
44 { % endif %}
45 </div>
46 { % endif %}
47 </div>
48
49 <!-- Artwork Gallery Section -->
50 <div class="artwork-gallery">
51 <!-- Form action="delete"/>{{user.id}} method="POST" enctype="multipart/form-data"
52 <div class="artwork-grid">
53 <!-- Form action="delete"/>{{user.id}} method="POST" enctype="multipart/form-data"
54 { % for artwork in artworks %}
55 <!-- Display the user's artworks here -->
56 <!-- You can use a loop to generate the artwork thumbnails -->
57 <!-- Example of a single artwork thumbnail -->
58 <div class="artwork-thumbnail">
59 <input type="checkbox" name="artworkToDelete" value="{{artwork.id}}">
60 <a href="{url_for('artworkDetail', artwork_id=artwork.id)}">
61 
62 </a>
63 <div class="art-title">{{artwork.title}}</div>
64 </div>
65 { % endfor %}
66 </div>
67 <button class="ui blue button custom-button" type="submit">Delete Selected Artwork</button>
68 </form>
69 </div>
70 </div>
71
72 <!-- Footer -->
73 <div>
74 <!-- Your footer content here -->
75 </div>
76
77 <script src="{url_for('static', filename='js/user_data_fetch.js')}"></script>
78 { % endblock %}
79 </body>
80 </html>
81

```

■ Error404.html:

```

1 <!DOCTYPE html>
2 <html lang="en" dir="ltr">
3 <head>
4 <meta charset="utf-8">
5 <title>Error Page</title>
6 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js" integrity="sha384-gaf790kctDZqQuwQQuL3Wf9OCFMDKxwz131pb//6851/3ed0k1" crossorigin="anonymous"></script>
7 <link rel="stylesheet" href="{url_for('static', filename='css/style.css')}">
8 <div class="jumbotron">
9 <h1>Sorry, we couldn't find the page you were looking for.</h1>
10 </div>
11 </html>

```

■ Homepage.html:

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="utf-8">
5 <title>ArtVision</title>
6 <link rel="stylesheet" href="{url_for('static', filename='css/registrationstyles.css')}" />
7 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.0/jquery.min.js"></script>
8 <script src="{url_for('static', filename='js/scripts.js')}"></script>
9 </script>
10 <script>
11 var all_art = {{all_art|tojson}}
12 var all_art_objects = {{all_art_objects|tojson}}
13 var all_user_objects = {{all_user_objects|tojson}}
14 </script>
15 </head>
16
17 <body id="background" style="background-image: url('{{all_art[0]}}');">
18 <div class="ui-container">
19 
20 <h1>ArtVision</h1>
21 <p>See the world in new colors</p>
22 <div class="home-buttons">
23 <button type="button" class="btn btn-primary" onclick="window.location.href='{url_for('register')}'">Create an Account</button>
24 <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 -->
25 </div>
26 <p><a href="{url_for('explore', sort='random')}">Continue without an Account</a></p>
27 </div>
28
29 <div class="artwork-info">
30 
31 <div class="artist-detail">
32 <p id="artwork_title">Artwork title</p>
33 <p by id="artwork_artist">Artist</p>
34 </div>
35 </div>
36 <script src="{url_for('static', filename='js/scripts.js')}"></script>
37 </body>
38 </html>

```

■ loginPage.html:

```

1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="utf-8">
5     <title>ArtVision | Login</title>
6     <link rel="stylesheet" href="{{ url_for('static', filename='css/registrationstyles.css') }}" />
7     <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.0/jquery.min.js"></script>
8     <script>
9       var all_art = {{ all_art|tojson }}
10      var all_art_objects = {{ all_art_objects|tojson }}
11      var all_user_objects = {{ all_user_objects|tojson }}
12    </script>
13  </head>
14
15  <body id="background" style="background-image: url('{{ all_art[0] }}');">
16    <div class="id-container">
17      <a href="{{ url_for('index') }}"></a>
18      <h1>Login to Your Account</h1>
19      <form method="POST">
20        {{ form.hidden_tag() }}
21        <br>
22        {{ form.userIdentity(placeholder="Email address or username") }}
23        <br>
24        {{ form.password(placeholder="Password", type="password") }}
25        <br>
26        {{ form.submit(class="custom-button") }}
27      </form>
28      {% if error %}
29      <p>{{ error }}</p>
30      {% endif %}
31      <p>Not a member yet? <a class="createAccount" href="{{ url_for('register') }}">Create an Account</a></p>
32    </div>
33
34    <div class="artwork-info">
35      
36      <div class="artist-details">
37        <p id="artwork_title">Artwork title</p>
38        <p by <span id="artwork_artist">Artist</span></p>
39      </div>
40    </div>
41
42    <script src="../../static/js/scripts.js"></script>
43  </body>
44 </html>

```

■ Purchase.html:

```

1 <!--[if IE]-->
2 <html lang="en">
3
4 <meta charset="UTF-8" />
5 <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6 <link rel="stylesheet" href="{ url_for('static', filename='css/style.css') }}" />
7 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0/dist/css/bootstrap.min.css" rel="stylesheet"
8 integrity="sha384-U98Zt/FknK9ZjXw4j6T5W9p43yF4n5W43v2R56075p43e4884m22ce88001a4f0000" crossorigin="anonymous">
9 <title Purchase /title>
10
11 <body class="purchase-page gradient-bg">
12 <header>
13 <!-- extends "base.html" -->
14 </header>
15 <!-- block content -->
16
17 <main class="main-content">
18 <div class="wrapper">
19 <!-- Checkout /h1 -->
20 <div class="form-container">
21 <div class="checkout-summary">
22 <!-- Summary /h2 -->
23 <!-- If session["cart"] -->
24 <!-- for item, art in cart.items() -->
25 <div class="card">
26 <!-- href="{ url_for('artworkDetails', artwork_id=item) }" -->
27 <div class="card-img" style="width: 100px; height: 100px; background-color: #d3d3d3">
28 
29 </div>
30 </div>
31 <div class="card-details">
32 <div class="card-title">
33 <div class="card-price">
34 <div class="quantity">
35 <div class="decrease">
36 <div class="increase">
37 </div>
38 </div>
39 </div>
40 </div>
41 </div>
42 </div>
43 </div>
44 </div>
45 </div>
46 </div>
47 <form method="POST" enctype="multipart/form-data" action="{ url_for('thankyou', user_id=user_id) }">
48 <h2>Shipping Address /h2>
49 <fieldset>
50 <div>
51 <label for="name">Name /label>
52 <input type="text" id="name" name="name" />
53 </div>
54 <div>
55 <label for="address">Address /label>
56 <input type="text" id="address" name="address" />
57 </div>
58 <div>
59 <label for="city">City /label>
60 <input type="text" id="city" name="city" />
61 </div>
62 <div>
63 <label for="state">State /label>
64 <input type="text" id="state" name="state" />
65 </div>
66 <div>
67 <label for="country">Countries /label>
68 <select id="country" name="country">
69 <option value="USA">United States /option>
70 <option value="CAN">Canada /option>
71 <option value="GBR">United Kingdom /option>
72 </select>
73 </div>
74 <div>
75 <label for="zip-code">Zip Code /label>
76 <input type="text" id="zip-code" name="zip-code" />
77 </div>
78 <input type="submit" value="Submit" />
79 </fieldset>
80
81 <h2>Payment Details /h2>
82 <fieldset>
83 <div>
84 <label for="card-number">Card Number /label>
85 <input type="text" id="card-number" name="card-number" />
86 </div>
87 <div>
88 <label for="c-name">Name on Card /label>
89 <input type="text" id="c-name" name="c-name" />

```

```

68     <select id="country" name="country" >
69         <option value="USA">United States</option>
70         <option value="CAN">Canada</option>
71         <option value="GBR">United Kingdom</option>
72     </select>
73 </div>
74 <div>
75     <label for="zip-code">Zip Code</label>
76     <input type="text" id="zip-code" name="zip-code" />
77 </div>
78 <!-- <input type="submit" value="Submit" /> -->
79 </fieldset>
80
81 <h2>Payment Details</h2>
82 <fieldset>
83     <div>
84         <label for="card-number">Card Number</label><br>
85         <input type="text" id="card-number" name="card-number" />
86     </div>
87     <div>
88         <label for="c-name">Name on Card</label><br>
89         <input type="text" id="c-name" name="c-name" />
90     </div>
91     <div>
92         <label for="date">Expiration Date</label>
93         <input
94             type="text"
95             id="date"
96             name="date"
97             placeholder="MM/YYYY"
98         />
99     </div>
100 </fieldset>
101 <div>
102     <label for="cvv">CVV</label>
103     <input type="text" id="cvv" name="cvv" placeholder="****" />
104 </div>
105 </fieldset>
106 <button class="btn custom-button" type="submit">Submit</button>
107 </form>
108 </div>
109 </div>
110 </main>
111 <script src="{ url_for('static', filename='js/update_cart.js') }" ></script>
112 {% endblock %}
113 </body>
114 </html>
115

```

■ Register.html

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>ArtVision | Register
8 </title>
9   <link rel="stylesheet" href="{{ url_for('static', filename='css/registrationstyles.css') }}">
10  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.0/jquery.min.js"></script>
11  <script>
12    var all_art = {{ all_art|tojson }}
13    var all_art_objects = {{ all_art_objects|tojson }}
14    var all_user_objects = {{ all_user_objects|tojson }}
15  </script>
16 </head>
17
18 <body id="background" style="background-image: url('{{all_art[0]}}');">
19   <div class="register fd-container">
20     <a href="{{url_for('index')}}"></a>
21     <h1>Create an Account</h1>
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>

```

```

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     </div>
52
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     
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

```

1 <DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8" />
5   <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6   <title>Selling</title>
7   <link rel="stylesheet" href="{{ url_for('static', filename='css/styles.css') }}" />
8   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet"
9     integrity="sha384-9ndCyUaIbzAi1fNpb3IZj0jqCy8PxWSxF40Jpe31ZFkwLHu4Ni2PK15Mn8Ku8XQFfd" crossorigin="anonymous">
10 </head>
11 <body class="selling-page gradient-bg">
12   <header>
13     {% extends "base.html" %}
14   </header>
15   {% block content %}
16     <main class="main-content">
17       <div class="wrapper">
18         <h1>New listing for {{ user.name }} </h1>
19         <div class="form-container">
20           <form method="POST" enctype="multipart/form-data" action="{{ url_for('addArt', user_id=user_id) }}">
21             <div>
22               <label>Title</label>
23               <br />
24               <input type="text" name="title" />
25             </div>
26             <div>
27               <input type="file" name="file" class="fileInput" />
28             </div>
29             <div class="submit-category">
30               <p><label for="category">Select artwork category:</label>
31                 <select id="category" name="category" required>
32                   <option value="Traditional Art">Traditional Art</option>
33                   <option value="Digital Art">Digital Art</option>
34                   <option value="Mixed Media">Mixed Media</option>
35                   <option value="Graphic Design">Graphic Design</option>
36                   <option value="Photography">Photography</option>
37                   <option value="Comics">Comics</option>
38                   <option value="Fan Art">Fan Art</option>
39                   <option value="Other">Other</option>
40                 </select>
41               </p>
42             </div>
43             <div>
44               <label>Description</label> <br />
45               <textarea id="description" name="description" rows="5" cols="40"></textarea>
46             </div>
47             <div>
48               <label>Price</label>
49               <br />
50               <input class="price-input" type="number" step="0.01" min="0.01" name="price" placeholder="$" />
51             </div>
52             <p class="estimate">Estimated earnings:</p>
53           </div>
54           <button class="btn custom-button" type="submit">Submit</button>

```



```

25     </div>
26     <div>
27       <input type="file" name="file" class="fileInput" />
28     </div>
29     <div class="submit-category">
30       <p><label for="category">Select artwork category:</label>
31         <select id="category" name="category" required>
32           <option value="Traditional Art">Traditional Art</option>
33           <option value="Digital Art">Digital Art</option>
34           <option value="Mixed Media">Mixed Media</option>
35           <option value="Graphic Design">Graphic Design</option>
36           <option value="Photography">Photography</option>
37           <option value="Comics">Comics</option>
38           <option value="Fan Art">Fan Art</option>
39           <option value="Other">Other</option>
40         </select>
41       </p>
42     </div>
43     <div>
44       <label>Description</label> <br />
45       <textarea id="description" name="description" rows="5" cols="40"></textarea>
46     </div>
47     <div>
48       <label>Price</label>
49       <br />
50       <input class="price-input" type="number" step="0.01" min="0.01" name="price" placeholder="$" />
51     </div>
52     <p class="estimate">Estimated earnings:</p>
53   </div>
54   <button class="btn custom-button" type="submit">Submit</button>
55 </form>
56
57 <div class="preview">
58   Preview
59   <div class="image-preview"></div>
60 </div>
61 </div>
62 <div>
63 </main>
64 <script>
65   const estimatedEarnings = document.querySelector('.estimate');
66   document.querySelector('.price-input').addEventListener('input', function(){
67     let inputPrice = parseFloat(this.value)
68     let total;
69     if (inputPrice >= 0){
70       total = inputPrice * 0.8;
71       estimatedEarnings.textContent = `Estimated earnings: ${total.toFixed(2)}`
72     }
73   });
74 </script>
75 <script src="{{ url_for('static', filename='js/file_upload_preview.js') }}"></script>
76 {% endblock %}
77 </body>

```

■ Shop.html

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Shope ArtVision</title>
7   <!-- Add your CSS file link here -->
8   <link rel="stylesheet" href="{{ url_for('static', filename='css/styles.css') }}" />
9 </head>
10 <body>
11   <!-- Navbar (You can create a separate header file and include it on all pages) -->
12   <header>
13     {% extends "base.html" %}
14   </header>
15   {% block content %}
16
17     <!-- Artwork Gallery Section -->
18     <div class="artwork-gallery">
19       <h2>ArtVision Gallery</h2>
20       <div class="artwork-grid">
21         {% for artwork in artworks %}
22           <!-- Display the site-wide artworks here -->
23           <!-- Draw random/popular artwork from within a certain time period -->
24           <!-- Example of a single artwork thumbnail -->
25           <div class="artwork-thumbnail">
26             <a href="{{ url_for('artworkDetails', artwork_id=artwork.id) }}">
27               
28             </a>
29             <div class="artwork-details">
30               <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>
31             </div>
32             <p>${"{:.2f}".format(artwork.price)}</p>
33           </div>
34         {% endfor %}
35       </div>
36     </div>
37
38     <!-- Footer -->
39     <footer>
40       <!-- Your footer content here -->
41     </footer>
42
43     <!-- Add your JavaScript file link here if needed -->
44     <script src="{{ url_for('static', filename='scripts.js') }}"></script>
45   {% endblock %}
46 </body>
47 </html>
48

```

■ Thankyou.html

```

1 <!DOCTYPE html>
2 <html lang="en" dir="ltr">
3
4   <head>
5     <meta charset="utf-8">
6     <title>Thank You</title>
7     <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js" integrity="sha384-gWwVafroWreM2corkqlo+PeW82K7qdvvfxptglentjeGajyhF9AjlHoo4b4pY+lJzer/cgh/" crossorigin="anonymous"></script>
8     <link rel="stylesheet" href="{{url_for('static', filename='css/style.css')}}">
9   </html>
10
11   <body>
12     <header>
13       {% extends "base.html" %}
14     </header>
15     {% block content %}
16     <div class="text-center">
17       <h1>{{message}}</h1>
18     </div>
19     {% endblock %}
20   </body>

```

■ Upload.html

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8" />
5   <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6   <title>Upload</title>
7   <link rel="stylesheet" href="{{ url_for('static', filename='css/styles.css') }}" />
8   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet"
9     integrity="sha384-9ndCyUa1bzL12FUXJ10C3mCapSm075npJef0486ghLnuZ2cdeRh082luK6FUUVM" crossorigin="anonymous">
10 </head>
11
12 <body class="upload-page gradient-bg">
13 <header>
14   {% extends "base.html" %}
15 </header>
16 {% block content %}
17 <main class="main-content">
18   <div class="wrapper">
19     <h1>Hi {{ user.name }}, submit your artwork</h1>
20     <div class="form-container">
21       <form class="ui form" action="{{ url_for('addArt', user_id=user_id) }}" method="POST"
22         enctype="multipart/form-data">
23         <div>
24           <label>Title</label>
25           <br />
26           <input type="text" name="title" />
27         </div>
28         <div>
29           <input type="file" name="file" class="fileInput" />
30         </div>
31         <div class="submit-category">
32           <p><label for="category">Select artwork category:</label>
33             <select id="category" name="category" required>
34               <option value="Traditional Art">Traditional Art</option>
35               <option value="Digital Art">Digital Art</option>
36               <option value="Mixed Media">Mixed Media</option>
37               <option value="Graphic Design">Graphic Design</option>
38               <option value="Photography">Photography</option>
39               <option value="Comics">Comics</option>
40               <option value="Fan Art">Fan Art</option>
41               <option value="Other">Other</option>
42             </select>
43           </p>
44         </div>
45         <div>
46           <label>Description</label> <br />
47           <textarea id="description" name="description" rows="5" cols="40"></textarea>
48         </div>
49         <!-- <div>
50           <label>price</label>
51           <br />
52           <input
53             type="number"
54             min="0" max="1000000" />
55         </div>
56       </form>
57     </div>
58   </div>
59 </main>
60 </div>
61 </div>
62 </div>
63 </div>
64 </div>
65 </div>
66 </div>
67 </div>
68 </div>
69 </div>
70 </div>
71 </div>
72 </div>
73 </div>
74 </div>
75 </div>
76 </div>
77 </div>
78 </div>
79 </div>
80 </div>
81 </div>
82 </div>
83 </div>
84 </div>
85 </div>
86 </div>
87 </div>
88 </div>
89 </div>
90 </div>
91 </div>
92 </div>
93 </div>
94 </div>
95 </div>
96 </div>
97 </div>
98 </div>
99 </div>
100 </div>

```

```

29     <input type="file" name="file" class="fileInput" />
30 </div>
31 <div class="submit-category">
32     <p><label for="category">Select artwork category:</label>
33     <select id="category" name="category" required>
34         <option value="Traditional Art">Traditional Art</option>
35         <option value="Digital Art">Digital Art</option>
36         <option value="Mixed Media">Mixed Media</option>
37         <option value="Graphic Design">Graphic Design</option>
38         <option value="Photography">Photography</option>
39         <option value="Comics">Comics</option>
40         <option value="Fan Art">Fan Art</option>
41         <option value="Other">Other</option>
42     </select>
43 </p>
44 </div>
45 <div>
46     <label>Description</label> <br />
47     <textarea id="description" name="description" rows="5" cols="40"></textarea>
48 </div>
49 <!-- <div>
50     <label>price</label>
51     <br />
52     <input
53         type="number"
54         step="0.01"
55         min="0.01"
56         name="price"
57         placeholder="price"
58     />
59 </div> -->
60
61 <!-- <div>
62     <label>Status</label>
63     <br />
64     <input type="text" name="status" placeholder="status" />
65 </div> -->
66 <button class="btn custom-button" type="submit">Submit</button>
67 </form>
68
69 <div class="preview">
70     Preview
71     <div class="image-preview"></div>
72 </div>
73 </div>
74 <div>
75 </main>
76 <script src="{{ url_for('static', filename='js/file_upload_preview.js') }}"></script>
77 {% endblock %}
78 </body>
79 </html>

```

■ userProfile.html

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>{{user.userName}} | ArtVision</title>
7     <!-- Add your CSS file link here -->
8     <link rel="stylesheet" href="{{ url_for('static', filename='css/profile.css') }}" />
9
10 </head>
11 <body class="user-profile">
12     <!-- Navbar (You can create a separate header file and include it on all pages) -->
13     {% extends "base.html" %}
14     <!-- User Profile Section -->
15     {% block content %}
16         <div class="profile">
17             <!-- Profile personal details -->
18             <div class="profile-details">
19                 <h1 class="user-name">{{ user.userName }} </h1>
20                 <div class="user-intro">
21                     <div class="user-img-container">
22                         
23                     </div>
24                     <div>
25                         <p>{{ user.name }}</p>
26                         <p>{{ user.pronouns }}</p>
27                         <p>{{ user.title }}</p>
28                     </div>
29                 </div>
30                 <p><b>About Me</b></p>
31                 <div class="bio">
32                     <p>{{ user.bio }}</p>
33                 </div>
34                 <p>Member Since: {{ (user.registrationDate).strftime("%B %d, %Y") }}</p>
35
36                 <!-- options if is users profile -->
37                 {% if isUsersProfile %}
38                 <div>
39                     <p><a href="{{ url_for('logout') }}" class="btn btn-primary">Log Out</a></p>
40                     <!--<p><a href="{{ url_for('uploadPage', user_id=user.id ) }}" class="btn btn-primary">Submit Art</a></p>-->
41                     <!--<p><a href="{{ url_for('sellingPage', user_id=user.id ) }}" class="btn btn-primary">Sell Your Art</a></p>-->
42                     <p><a href="{{ url_for('deletePage', user_id=user.id ) }}" class="btn btn-primary">Manage Gallery</a></p>
43                     <p><a href="{{ url_for('editPage', user_id=user.id ) }}" class="btn btn-primary">Edit Account</a></p>
44                     {% if message %}
45                     <p>{{ message }}</p>
46                     {% endif %}
47                 </div>
48                 {% endif %}
49             </div>
50
51             {% if isUsersProfile %}
52             <div>
53                 <p><a href="{{ url_for('logout') }}" class="btn btn-primary">Log Out</a></p>
54                 <!--<p><a href="{{ url_for('uploadPage', user_id=user.id ) }}" class="btn btn-primary">Submit Art</a></p>-->
55                 <!--<p><a href="{{ url_for('sellingPage', user_id=user.id ) }}" class="btn btn-primary">Sell Your Art</a></p>-->
56                 <p><a href="{{ url_for('deletePage', user_id=user.id ) }}" class="btn btn-primary">Manage Gallery</a></p>
57                 <p><a href="{{ url_for('editPage', user_id=user.id ) }}" class="btn btn-primary">Edit Account</a></p>
58                 {% if message %}
59                 <p>{{ message }}</p>
60                 {% endif %}
61             </div>
62             {% endif %}
63
64             <!-- Artwork Gallery Section -->
65             <div class="artwork-gallery">
66                 <nav>
67                     <a href="#" class="user-data" onclick="loadContent('gallery')" value="{{user.id}}">Gallery</a>
68                     <a href="#" class="user-data" onclick="loadContent('shop')" value="{{user.id}}">Shop</a>
69                     <a href="#" class="user-data" onclick="loadContent('favorites')" value="{{user.id}}">Favorites</a>
70                 </nav>
71                 <div class="artwork-grid">
72                     <!-- Artworks will populate here when explore is visited -->
73                     {% for artwork in artworks %}
74                     {% if not artwork.shop_item %}
75                     <!-- Display the user's artworks here -->
76                     <!-- You can use a loop to generate the artwork thumbnails -->
77                     <!-- Example of a single artwork thumbnail -->
78                     <div class="artwork-thumbnail">
79                         <a href="{{ url_for('artworkDetails', artwork_id=artwork.id) }}">
80                             
81                         </a>
82                         <div class="art-title"><a href="{{ url_for('artworkDetails', artwork_id=artwork.id) }}">{{ artwork.title }}</a></div>
83                     </div>
84                     {% endif %}
85                     {% endfor %}
86                 </div>
87             </div>
88
89             <!-- Footer -->
90             <footer>
91                 <!-- Your footer content here -->
92             </footer>
93
94             <!-- Add your JavaScript file link here if needed -->
95             <script src="{{ url_for('static', filename='js/user_data_fetch.js') }}"></script>
96             {% endblock %}
97         </body>
98     </html>

```

○ Crud.py:

```

1  from main import app, db, User, Artwork
2  import os
3  import boto3
4
5  # ###
6  # GRAB ACCESS_KEY and SECRET_KEY FROM GITHUB. DO NOT COMMIT TO GITHUB WITH ACCESS KEYS IN CODE
7
8  ACCESS_KEY = os.getenv("ACCESS_KEY")
9  SECRET_KEY = os.getenv("SECRET_KEY")
10 AWS_REGION = os.getenv("AWS_REGION")
11
12
13 client = boto3.client(
14     's3',
15     aws_access_key_id=ACCESS_KEY,
16     aws_secret_access_key=SECRET_KEY,
17     region_name=AWS_REGION
18 )
19
20 # s3 = boto3.resource(
21 #     's3',
22 #     aws_access_key_id=ACCESS_KEY,
23 #     aws_secret_access_key=SECRET_KEY,
24 #     region_name=AWS_REGION
25 # )
26
27 # Currently not working!!!
28
29 def delete_photo_from_s3(photo_url):
30     if not photo_url:
31         return
32     url_parts = photo_url.split("/")
33
34     # Get the elements we need from the URL parts
35     userName = url_parts[4]
36     file_name = url_parts[-1]
37     file_name_parts = file_name.split("?")
38     fileName = file_name_parts[0]
39
40     # print("User name:", userName)
41     # print("File name:", fileName)
42
43     # Create a Boto3 client for S3
44     client = boto3.client(
45         's3',
46         aws_access_key_id=ACCESS_KEY,
47         aws_secret_access_key=SECRET_KEY,
48         region_name=AWS_REGION
49     )

```

```

# 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 NECESSARY

# 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_user)
# 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

```
1 import os
2 import re
3 import datetime
4 import random
5 from flask import Flask, render_template, session, redirect, url_for, request, jsonify
6 from flask_wtf import FlaskForm
7 from wtforms import (StringField, SubmitField, BooleanField, DateTimeField,
8                      RadioField, SelectField, TextAreaField, DecimalField)
9 from flask_wtf.file import FileField, FileAllowed, FileRequired
10 from wtforms.validators import DataRequired, Length
11 from flask_sqlalchemy import SQLAlchemy
12 from flask_migrate import Migrate
13 from sqlalchemy import LargeBinary, func, desc
14 import boto3
15 from werkzeug.utils import secure_filename
16 import random
17 from dotenv import load_dotenv
18 import requests # need to pip install requests
19 import json
20 from json import dumps
21 from sqlalchemy.orm import class_mapper
22
23 load_dotenv()
24
25 # ###
26 # GRAB ACCESS_KEY and SECRET_KEY FROM DISCORD. DO NOT COMMIT TO GITHUB WITH ACCESS KEYS IN CODE
27 ACCESS_KEY = os.getenv("ACCESS_KEY")
28 SECRET_KEY = os.getenv("SECRET_KEY")
29 AWS_REGION = os.getenv("AWS_REGION")
30
31 #artvisionbucket
32
33 basedir = os.path.abspath(os.path.dirname(__file__))
34 # __file__ refers to main.py
35 # abspath -> absolute path -> provides the full directory path
```

```
33
34 basedir = os.path.abspath(os.path.dirname(__file__))
35 # __file__ refers to main.py
36 # abspath -> absolute path -> provides the full directory path
37
38 # We render templates by importing the render_template function from flask and
39 # returning an .html file from our view function
40 app = Flask(__name__)
41
42 app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///'+os.path.join(basedir, 'data.sqlite')
43 app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
44
45 app.config['SECRET_KEY'] = 'oursecretkey'
46
47 db = SQLAlchemy(app)
48 Migrate(app, db)
49
50 client = boto3.client(
51     's3',
52     aws_access_key_id = ACCESS_KEY,
53     aws_secret_access_key = SECRET_KEY,
54     region_name=AWS_REGION
55 )
56
57
58 def passwordValidation(PWD):
59     regexCapLetter = r'[A-Z]'
60     regexLowLetter = r'[a-z]'
61     regexEndNumber = r'[0-9]$',
62     regexList = [regexCapLetter, regexLowLetter, regexEndNumber]
63     count = 0
64     for regex in range(0,3):
65         match = re.search(regexList[regex], PWD)
66         if match:
67             count+=1
68     if count == 3:
```

```

57
58 def passwordValidation(PWD):
59     regexCapLetter = r'[A-Z]'
60     regexLowLetter = r'[a-z]'
61     regexEndNumber = r'[0-9]$$'
62     regexList = [regexCapLetter, regexLowLetter, regexEndNumber]
63     count = 0
64     for regex in range(0,3):
65         match = re.search(regexList[regex],PWD)
66         if match:
67             count+=1
68     if count == 3:
69         return True
70     else:
71         return False
72
73 def delete_photo_from_s3(photo_url):
74     if not photo_url:
75         return
76     url_parts = photo_url.split("/")
77
78     # Get the elements we need from the URL parts
79     userName = url_parts[4]
80     file_name = url_parts[-1]
81     file_name_parts = file_name.split("?")
82     fileName = file_name_parts[0]
83
84     # print("User name:", userName)
85     # print("File name:", fileName)
86
87     # Create a Boto3 client for S3
88     # client = boto3.client(
89     #     's3',
90     #     aws_access_key_id=ACCESS_KEY,
91     #     aws_secret_access_key=SECRET_KEY,

```

```

72
73 def delete_photo_from_s3(photo_url):
74     if not photo_url:
75         return
76     url_parts = photo_url.split("/")
77
78     # Get the elements we need from the URL parts
79     userName = url_parts[4]
80     file_name = url_parts[-1]
81     file_name_parts = file_name.split("?")
82     fileName = file_name_parts[0]
83
84     # print("User name:", userName)
85     # print("File name:", fileName)
86
87     # Create a Boto3 client for S3
88     # client = boto3.client(
89     #     's3',
90     #     aws_access_key_id=ACCESS_KEY,
91     #     aws_secret_access_key=SECRET_KEY,
92     #     region_name=AWS_REGION
93     # )
94
95     # s3 = boto3.resource(
96     #     's3',
97     #     aws_access_key_id=ACCESS_KEY,
98     #     aws_secret_access_key=SECRET_KEY,
99     #     region_name=AWS_REGION
100     # )
101     # Currently not working!!!
102     try:
103         client.delete_object(Bucket="artvisionbucket", Key="profilephoto/" + userName + "/" + fileName)
104         # s3.Object("artvisionbucket", "profilephoto/" + fileName).delete()
105         print(f"Photo {fileName} deleted from S3 bucket")
106     except Exception as e:
107         print(f"Error deleting photo {fileName} from S3 bucket: {e}")

```

```

101 # Currently not working!!!
102 try:
103     client.delete_object(Bucket="artvisionbucket", Key="profilephoto/" + userName + "/" + fileName)
104     # s3.Object("artvisionbucket", "profilephoto/" + filename).delete()
105     print(f"Photo {fileName} deleted from S3 bucket")
106 except Exception as e:
107     print(f"Error deleting photo {fileName} from S3 bucket: {e}")
108
109
110 def delete_artwork_from_s3(artwork_url):
111     if not artwork_url:
112         return
113
114     url_parts = artwork_url.split("/")
115
116     # Get the elements we need from the URL parts
117     userName = url_parts[4]
118     file_name = url_parts[-1]
119     file_name_parts = file_name.split("?")
120     fileName = file_name_parts[0]
121     # Create a Boto3 client for S3
122     # client = boto3.client(
123     #     's3',
124     #     aws_access_key_id=ACCESS_KEY,
125     #     aws_secret_access_key=SECRET_KEY,
126     #     region_name=AWS_REGION
127     # )
128
129     try:
130         client.delete_object(Bucket="artvisionbucket", Key="artgallery/" + userName + "/" + fileName)
131         print(f"Artwork {fileName} deleted from S3 bucket")
132     except Exception as e:
133         print(f"Error deleting artwork {fileName} from S3 bucket: {e}")
134

```

```

135 class LoginForm(FlaskForm):
136     useridentity = StringField(validators = [DataRequired()])
137     password = StringField(validators = [DataRequired()])
138     submit = SubmitField('Login')
139
140 class RegistrationForm(FlaskForm):
141     name = StringField(validators = [DataRequired()])
142     email = StringField(validators = [DataRequired()])
143     password = StringField(validators = [DataRequired()])
144     confirmPassword = StringField(validators = [DataRequired()])
145     userName = StringField(validators = [DataRequired()])
146     #profilePhoto = FileField('Profile Photo', validators=[FileAllowed(['jpg', 'jpeg', 'png']), FileSize(max_size=2 * 1024 * 1024, message='No photos larger than 2MB.')]
147     bio = TextAreaField()
148     pronouns = SelectField('Choose your pronouns', choices=[('option1', 'she/her'), ('option2', 'he/him'), ('option3', 'they/them'), ('option4', 'she/they'), ('option5',
149     title = SelectField('Choose a title', choices=[('title1', 'Professional'), ('title2', 'Student'), ('title3', 'Hobbyist')])
150     submit = SubmitField('Sign Up')
151
152 #Upload File Form
153 class UploadForm(FlaskForm):
154     title = StringField('Title', validators = [DataRequired(), Length(max=150)])
155     fileInput = FileField('Upload File', validators = [FileRequired(), FileAllowed(['png', 'jpg', 'jpeg', 'gif'])])
156     description = TextAreaField('Artwork Description', validators = [DataRequired(), Length(max=400)])
157     category = SelectField('Select artwork category', validators = [DataRequired()]), choices=[('option1', 'Traditional Art'), ('option2', 'Digital Art'), ('option3', 'Mix
158     saveDraft = SubmitField('Save Draft')
159     submit = SubmitField('Submit')
160
161
162 #Selling Form inherits from UploadForm
163 class SellingForm(UploadForm):
164     price = DecimalField('Price', validators=[DataRequired()])
165

```

```

166
167 class CommentForm(FlaskForm):
168     text = StringField()
169     submit = SubmitField("Comment")
170
171
172 def serialize(model):
173     columns = [c.key for c in class_mapper(model.__class__).columns]
174     return dict((c, getattr(model, c)) for c in columns)
175
176 class User(db.Model):
177     __tablename__ = "users"
178
179     id = db.Column(db.Integer, primary_key = True)
180     name = db.Column(db.Text)
181     email = db.Column(db.Text)
182     userName = db.Column(db.Text)
183     password = db.Column(db.Text)
184     bio = db.Column(db.Text)
185     profilePhotoLink = db.Column(db.Text)
186     pronouns = db.Column(db.Text)
187     title = db.Column(db.Text)
188     registrationDate = db.Column(db.DateTime)
189
190     def __init__(self, name, email, userName, password, bio, profilePhotoLink, pronouns, title, registrationDate):
191         self.name = name
192         self.email = email
193         self.userName = userName
194         self.password = password
195         self.bio = bio
196         self.profilePhotoLink = profilePhotoLink
197         self.pronouns = pronouns
198         self.title = title
199         self.registrationDate = registrationDate

```

```

204
205 class Artwork(db.Model):
206     __tablename__ = "artworks"
207
208     id = db.Column(db.Integer, primary_key = True)
209     title = db.Column(db.String(80), nullable = False)
210     description = db.Column(db.String(120), nullable = False)
211     category = db.Column(db.String(32))
212     price = db.Column(db.Float, nullable = True)
213     status = db.Column(db.String(20), nullable = True)
214     url = db.Column(db.Text)
215     artist = db.Column(db.String(32))
216     user_id = db.Column(db.Integer, db.ForeignKey('users.id'), nullable = False)
217     uploadDate = db.Column(db.DateTime)
218     shop_item = db.Column(db.Boolean, default=False)
219
220
221     def __init__(self, title, description, category, price, status, url, artist, user_id, uploadDate, shop_item):
222         self.title = title
223         self.description = description
224         self.category = category
225         self.price = price
226         self.status = status
227         self.url = url
228         self.artist = artist
229         self.user_id = user_id
230         self.uploadDate = uploadDate
231         self.shop_item = shop_item
232
233     def __repr__(self):
234         return f"<Artwork {self.title}>"
235

```

```

236
237 class Comment(db.Model):
238     __tablename__ = "comments"
239     id = db.Column(db.Integer, primary_key=True)
240     artwork_id = db.Column(db.Integer)
241     text = db.Column(db.String(140))
242     profile_pic = db.Column(db.Text)
243     author = db.Column(db.String(32))
244     author_id = db.Column(db.Text)
245     timestamp = db.Column(db.DateTime(), index=True)
246
247     def __init__(self, artwork_id, text, profile_pic, author, author_id, timestamp):
248         self.artwork_id = artwork_id
249         self.text = text
250         self.profile_pic = profile_pic
251         self.author = author
252         self.author_id = author_id
253         self.timestamp = timestamp
254
255     def __repr__(self):
256         return f"<Comment {self.text}>"
257
258 with app.app_context():
259     # Create the tables (if not already created)
260     db.create_all()
261
262 # success = False # Login variable
263

```

```

284 @app.route('/loginPage', methods = ['GET', 'POST'])
285 def loginPage():
286     form = LoginForm()
287     if form.validate_on_submit():
288         userIdentity = form.userIdentity.data
289         passwordInput = form.password.data
290
291         user = User.query.filter((User.userName==userIdentity) | (User.email==userIdentity)).first()
292         if user and user.password==passwordInput:
293             # global success
294             # success = True
295             session['logged_in'] = True
296             session['user_id'] = user.id
297             session['cart'] = {}
298             return redirect(url_for('userProfile', user_id=user.id)) #User's home page
299         else:
300             error = "Incorrect Login Info"
301
302     all_art = []
303
304     all_art_objects = []
305     all_user_objects = []
306
307     for art in Artwork.query.all():
308         all_art.append(art.url)
309         all_art_objects.append(serialize(art))
310
311     random.shuffle(all_art)
312
313     for user in User.query.all():
314         all_user_objects.append(serialize(user))
315
316     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=

```

```

318 @app.route("/register")
319 def register():
320     all_art = []
321
322     all_art_objects = []
323     all_user_objects = []
324
325     for art in Artwork.query.all():
326         all_art.append(art.url)
327         all_art_objects.append(serialize(art))
328
329     for user in User.query.all():
330         all_user_objects.append(serialize(user))
331
332     random.shuffle(all_art)
333
334     return render_template("register.html", all_art=all_art, all_art_objects=all_art_objects, all_user_objects=all_user_objects)
335
336 # Adds
337 @app.route("/add", methods = ["POST"])
338 def add():
339     all_art = []
340
341     all_art_objects = []
342     all_user_objects = []
343
344
345     for art in Artwork.query.all():
346         all_art.append(art.url)
347         all_art_objects.append(serialize(art))
348
349     for user in User.query.all():
350         all_user_objects.append(serialize(user))
351
352     random.shuffle(all_art)
353
354     genericPhotoLink = 'image/profile_photo.jpeg'
355

```

```

354 genericPhotoLink = 'image/profile_photo.jpeg'
355
356 name = request.form.get("name")
357 email = request.form.get("email")
358 username = request.form.get("username")
359 password = request.form.get("password")
360 confirmPassword = request.form.get("confirmpassword")
361 bio = request.form.get("bio")
362 f = request.files["file"]
363 pronouns = request.form.get("pronouns")
364 title = request.form.get("title")
365 # filename = f.filename.split("\")[1]
366 filename = f"{username}profilepicture.{f.filename.split('.')[1]}"
367
368 emailCheck = User.query.filter_by(email=email).first()
369 if emailCheck: # If email already exists in database
370     error = "The email you entered is already taken."
371     return render_template('register.html', error=error, all_art=all_art, all_art_objects=all_art_objects, all_user_objects=all_user_objects)
372
373 userNameCheck = User.query.filter_by(userName=username).first()
374 if userNameCheck: # If email already exists in database
375     error = "The username you entered is already taken."
376     return render_template('register.html', error=error, all_art=all_art, all_art_objects=all_art_objects, all_user_objects=all_user_objects)
377
378 if not passwordValidation(password):
379     error = "Password must contain at least one capital letter, one lowercase letter, and end with a number."
380     return render_template('register.html', error=error, all_art=all_art, all_art_objects=all_art_objects, all_user_objects=all_user_objects)

```

```

372     userNameCheck = User.query.filter_by(userName=username).first()
373     if userNameCheck: # If email already exists in database
374         error = "The username you entered is already taken."
375         return render_template('register.html', error=error, all_art=all_art, all_art_objects=all_art_objects, all_user_objects=all_user_objects)
376
377     if not passwordValidation(password):
378         error = "Password must contain at least one capital letter, one lowercase letter, and end with a number."
379         return render_template('register.html', error=error, all_art=all_art, all_art_objects=all_art_objects, all_user_objects=all_user_objects)
380
381     if password != confirmpassword:
382         error = "Passwords do not match."
383         return render_template('register.html', error=error, all_art=all_art, all_art_objects=all_art_objects, all_user_objects=all_user_objects)
384
385     # if ' ' in filename:
386     #     error = "photo name cannot contain a space."
387     #     return render_template('register.html', error=error)
388
389     f.save(secure_filename(filename))
390
391     # client = boto3.client(
392     #     's3',
393     #     aws_access_key_id = ACCESS_KEY,
394     #     aws_secret_access_key = SECRET_KEY ,
395     #     region_name=AWS_REGION
396     # )
397

```

```

382     if password != confirmpassword:
383         error = "Passwords do not match."
384         return render_template('register.html', error=error, all_art=all_art, all_art_objects=all_art_objects, all_user_objects=all_user_objects)
385
386     # if ' ' in filename:
387     #     error = "photo name cannot contain a space."
388     #     return render_template('register.html', error=error)
389
390     f.save(secure_filename(filename))
391
392     # client = boto3.client(
393     #     's3',
394     #     aws_access_key_id = ACCESS_KEY,
395     #     aws_secret_access_key = SECRET_KEY ,
396     #     region_name=AWS_REGION
397     # )
398     client.upload_file(filename, "artvisionbucket", "profilephoto/" + username + "/" + filename)
399     # presigned_url = client.generate_presigned_url('get_object',
400     #     Params={
401     #         "Bucket": "artvisionbucket",
402     #         "Key": "profilephoto/" + username + "/" + filename
403     #     },
404     # )
405     bucket_name = "artvisionbucket"
406     s3_key = "profilephoto/" + username + "/" + filename
407     url = f"https://{bucket_name}.s3.{AWS_REGION}.amazonaws.com/{s3_key}"
408     os.remove(filename)
409
410     newUser = User(name=name, email=email,
411         userName=username, password=password, bio=bio, profilePhotoLink=url, pronouns=pronouns, title=title, registrationDate=datetime.date.today())
412
413     db.session.add(newUser)
414     db.session.commit()
415     return redirect('loginPage')
416     # return redirect('/')
417
418
419 @app.route('/user/cint:user_id')
420 def userProfile(user_id):
421     if session['logged_in'] == True and 'user_id' in session and session['user_id'] == user_id: # 1st person profile visit
422         user = User.query.get(user_id)
423         artworks = Artwork.query.filter_by(user_id=user_id).all()
424
425         return render_template('userProfile.html', user=user, currentUser=user, isUsersProfile=True, artworks=artworks)
426

```

```

419 @app.route('/user/cint:user_id')
420 def userProfile(user_id):
421     if session['logged_in'] == True and 'user_id' in session and session['user_id'] == user_id: # 1st person profile visit
422         user = User.query.get(user_id)
423         artworks = Artwork.query.filter_by(user_id=user_id).all()
424
425         return render_template('userProfile.html', user=user, currentUser=user, isUsersProfile=True, artworks=artworks)
426
427     elif session['logged_in'] == True and 'user_id' in session and session['user_id'] != user_id:
428         # 3rd person profile visit
429         currentUser = User.query.get(session['user_id'])
430         user = User.query.get(user_id)
431         artworks = Artwork.query.filter_by(user_id=user_id).all()
432
433         return render_template('userProfile.html', user=user, currentUser=currentUser, isUsersProfile=False, artworks=artworks)
434
435     else:
436         currentUser=None
437         user = User.query.get(user_id)
438         artworks = Artwork.query.filter_by(user_id=user_id).all()
439         return render_template('userProfile.html', user=user, currentUser=currentUser, isUsersProfile=False, artworks=artworks)
440
441     # else:
442     #     return redirect(url_for('error404'))
443
444
445 @app.route('/explore', methods=["GET", "POST"])
446 def explore():
447     artworks = Artwork.query.all()
448     random.shuffle(artworks)
449
450     randomArtwork = random.choice(Artwork.query.all())
451
452     if 'user_id' in session and session['logged_in'] == True:
453         currentUser = User.query.get(session['user_id'])
454         user = User.query.get(session['user_id'])
455
456         if request.method == "POST":
457             search = request.form.get("search_term")
458             print(search)
459             artworks = Artwork.query.filter((Artwork.title.like("%"+search+"%")) | (Artwork.artist.like("%"+search+"%")) | (Artwork.description.like("%"+search+"%"))).all()
460
461             random.shuffle(artworks)
462             return render_template('explore.html', artworks=artworks, currentUser=currentUser, randomArtwork=randomArtwork, user=user, userLoggedIn=True)
463

```

```

445 @app.route('/explore', methods=["GET", "POST"])
446 def explore():
447     artworks = Artwork.query.all()
448     random.shuffle(artworks)
449
450     randomArtwork = random.choice(Artwork.query.all())
451
452     if 'user_id' in session and session['logged_in'] == True:
453         currentUser = User.query.get(session['user_id'])
454         user = User.query.get(session['user_id'])
455
456         if request.method == "POST":
457             search = request.form.get("search_term")
458             print(search)
459             artworks = Artwork.query.filter((Artwork.title.like("%"+search+"%")) | (Artwork.artist.like("%"+search+"%")) | (Artwork.description.like("%"+search+"%"))).all()
460
461             random.shuffle(artworks)
462             return render_template('explore.html', artworks=artworks, currentUser=currentUser, randomArtwork=randomArtwork, user=user, userLoggedIn = True)
463
464         else:
465             return render_template('explore.html', artworks=artworks, currentUser=currentUser, randomArtwork=randomArtwork, user=user, userLoggedIn = True)
466
467     else:
468         if request.method == "POST":
469             search = request.form.get("search_term")
470             random.shuffle(artworks)
471             return render_template('explore.html', artworks=artworks, randomArtwork=randomArtwork, userLoggedIn = False)
472
473         else:
474             return render_template('explore.html', artworks=artworks, randomArtwork=randomArtwork, userLoggedIn = False)

```

```

474 #shop page
475 @app.route('/shop')
476 def shop():
477     artworks = Artwork.query.filter_by(shop_item=True).all()
478     random.shuffle(artworks)
479
480     if 'user_id' in session and session['logged_in'] == True:
481         currentUser = User.query.get(session['user_id'])
482         user = User.query.get(session['user_id'])
483         return render_template('shop.html', artworks=artworks, user=user, userLoggedIn = True, currentUser=currentUser)
484
485 @app.route('/artwork/<int:artwork_id>', methods=["GET", "POST"])
486 def artworkDetails(artwork_id):
487     form=CommentForm()
488     artwork = Artwork.query.get(artwork_id)
489     print(artwork_id)
490     artist = User.query.get(artwork.user_id)
491
492     if 'user_id' not in session: # add this line
493         # handle the case when the user is not logged in
494         user = None # or whatever is appropriate in your case
495         userLoggedIn = False # add this line
496
497     else:
498         user = User.query.get(session['user_id'])
499         userLoggedIn = True # add this line
500
501     comments = Comment.query.filter_by(artwork_id = artwork_id).all()
502     commentsCount = db.session.execute(Comment.query.filter_by(artwork_id=artwork.id).statement.with_only_columns(func.count()).order_by(None)).scalar()
503
504     if not artwork:
505         return render_template('error.html', message='Artwork not found.')
506
507     if request.method == "POST":
508         if users: # add this line
509             text = request.form.get("text")
510             newComment = Comment(artwork_id=artwork_id, text=text, author=user.userName, profile_pic=user.profilePhotoLink, author_id=user.id, timestamp=datetime.date.today())
511
512             db.session.add(newComment)
513             db.session.commit()
514             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))
515
516     return render_template('artworkDetails.html', artwork=artwork, user=user, currentUser=user, artist=artist, form=form, comments=comments, commentsCount=commentsCount, userLoggedIn=userLoggedIn)

```

```

474 #shop page
475 @app.route('/shop')
476 def shop():
477     artworks = Artwork.query.filter_by(shop_item=True).all()
478     random.shuffle(artworks)
479     if 'user_id' in session and session['logged_in'] == True:
480         currentUser = User.query.get(session['user_id'])
481         user = User.query.get(session['user_id'])
482         return render_template('shop.html', artworks=artworks, user=user, userLoggedIn = True, currentUser=currentUser)
483
484 @app.route('/artwork/<int:artwork_id>', methods=['GET', 'POST'])
485 def artworkDetails(artwork_id):
486     form=CommentForm()
487     artwork = Artwork.query.get(artwork_id)
488     print(artwork_id)
489     artist = User.query.get(artwork.user_id)
490
491     if 'user_id' not in session: # add this line
492         # handle the case when the user is not logged in
493         user = None # or whatever is appropriate in your case
494         userLoggedIn = False # add this line
495     else:
496         user = User.query.get(session['user_id'])
497         userLoggedIn = True # add this line
498
499     comments = Comment.query.filter_by(artwork_id = artwork_id).all()
500     commentsCount = db.session.execute(Comment.query.filter_by(artwork_id=artwork_id).statement.with_only_columns(func.count()).order_by(None)).scalar()
501
502     if not artwork:
503         return render_template('error.html', message='Artwork not found.')
504
505     if request.method == 'POST':
506         if user: # add this line
507             text = request.form.get('text')
508             newComment = Comment(artwork_id=artwork_id, text=text, author=user.userName, profile_pic=user.profilePhotoLink, author_id=user.id, timestamp=datetime.date.today())
509             db.session.add(newComment)
510             db.session.commit()
511             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))
512
513     return render_template('artworkDetails.html', artwork=artwork, user=user, currentUser=user, artist=artist, form=form, comments=comments, commentsCount=commentsCount, userLoggedIn=userLoggedIn)
514
515
516
517 #upload File Page
518 @app.route('/uploadPage/<int:user_id>', methods = ['GET', 'POST'])
519 def uploadPage(user_id):
520     user = User.query.get(user_id)
521     return render_template('upload.html', user=user, user_id=user_id, currentUser=user)
522
523
524 #Upload Comission Page
525 @app.route('/sellingPage/<int:user_id>', methods = ['GET', 'POST'])
526 def sellingPage(user_id):
527     user = User.query.get(user_id)
528     return render_template('selling.html', user=user, user_id=user_id, currentUser=user)
529
530
531 #Checkout Page
532 @app.route('/purchasePage/<int:user_id>', methods = ['GET', 'POST'])
533 def purchasePage(user_id):
534     user = User.query.get(user_id)
535     total = calculateCartTotal(session['cart'])
536     return render_template('purchase.html', user=user, user_id=user_id, currentUser=user, cart=session['cart'], total=total)
537
538
539 @app.route('/thankyou', methods = ['POST'])
540 def thankyou():
541     user_id = session['user_id']
542     if 'cart' in session and session['cart']:
543         user = User.query.get(user_id)
544         session['cart'] = {} #empty cart
545         message = "Thank you for your order"
546         return render_template('thankyou.html', message=message, currentUser=user, user=user)
547
548     return redirect(url_for('purchasePage', user_id=user_id))
549
550
551
552 @app.route('/addArt/<int:user_id>', methods = ['POST'])
553 def addArt(user_id):
554     user = User.query.get(user_id)
555     artist = user.userName
556
557     title = request.form.get("title")
558     description = request.form.get("description")
559     category = request.form.get("category")
560     price = request.form.get("price")
561     status = request.form.get("status")
562     shop_item = "price" in request.form
563
564     f = request.files["file"]
565     filename = f.filename.split("\\\\")[1]
566     f.save(secure_filename(filename))
567
568     # client = boto3.client(
569     #     's3',
570     #     aws_access_key_id= ACCESS_KEY,
571     #     aws_secret_access_key= SECRET_KEY,
572     #     region_name=AWS_REGION
573     # )

```



```

547
548 @app.route('/addArt/cint:user_id', methods = ['POST'])
549 def addArt(user_id):
550     user = User.query.get(user_id)
551     artist = user.userName
552
553     title = request.form.get('title')
554     description = request.form.get('description')
555     category = request.form.get('category')
556     price = request.form.get('price')
557     status = request.form.get('status')
558     shop_item = 'price' in request.form
559
560     f = request.files['file']
561     filename = f.filename.split('\\')[-1]
562     f.save(secure_filename(filename))
563
564     # client = boto3.client(
565     #     's3',
566     #     aws_access_key_id= ACCESS_KEY,
567     #     aws_secret_access_key= SECRET_KEY,
568     #     region_name=AWS_REGION
569     # )
570
571     # Upload the file to S3 bucket
572     client.upload_file(filename, 'artvisionbucket', 'artgallery/' + str(user.userName) + "/" + filename)
573     # presigned_url = client.generate_presigned_url('get_object',
574     #     Params={
575     #         'Bucket': 'artvisionbucket',
576     #         'Key': 'artgallery/' + str(user.userName) + "/" + filename
577     #     },
578     # )
579
580     bucket_name = 'artvisionbucket'
581     s3_key = ('artgallery/' + user.userName) + '/' + filename
582     url = "https://(bucket_name).s3.(AWS_REGION).amazonaws.com/(s3_key)"
583
584     os.remove(filename)
585     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)
586     db.session.add(newArt)
587     db.session.commit()
588     return redirect(url_for('userProfile', user_id=user_id))
589
590

```

```

591
592 #Add artwork to cart
593 @app.route('/addCart/cint:artwork_id', methods = ['POST'])
594 def addCart(artwork_id):
595     artwork = Artwork.query.get(artwork_id)
596     quantity = int(request.form.get('quantity'))
597     if not artwork:
598         return render_template('error.html', message='Artwork not found.')
599     cart = session['cart']
600
601     if str(artwork_id) not in cart:
602         cart[str(artwork_id)] = {'quantity': 0, 'price': artwork.price, 'image_url': artwork.url, 'title': artwork.title, 'id': artwork.id}
603
604     cart[str(artwork_id)]['quantity'] += quantity
605     session['cart'] = cart
606     return redirect(url_for('artworkDetails', artwork_id=artwork_id))
607
608 #Update cart quantity
609 @app.route('/updateCart', methods=['POST'])
610 def updateCart():
611     data = request.get_json()
612     artwork_id = data.get('artwork_id')
613     quantity = int(data.get('quantity', 0))
614     cart = session['cart']
615     cart[artwork_id]['quantity'] = quantity
616     session['cart'] = cart
617     return jsonify(success=True)
618
619 def calculateCartTotal(cart):
620     total = 0.00
621     for attr in cart.values():
622         total += attr['quantity'] * attr['price']
623     return total
624
625 #Delete item from cart
626 @app.route('/deleteCartItem/cint:artwork_id')
627 def deleteCartItem(artwork_id):
628     cart = session.get('cart', {})
629     if str(artwork_id) in cart:
630         del cart[str(artwork_id)]
631     session['cart'] = cart
632     return redirect(url_for('purchasePage', user_id=session['user_id']))
633
634
635
636
637 #Upload File Page
638 @app.route('/deletePage/cint:user_id', methods = ['GET', 'POST'])
639 def deletePage(user_id):
640     user = User.query.get(user_id)
641     artworks = Artwork.query.filter_by(user_id=user_id).all()
642     return render_template('deleteArt.html', user=user, user_id=user_id, artworks=artworks, currentUser=user)
643

```

```

637 #Upload File Page
638 @app.route('/deletePage/cint:user_id', methods = ['GET', 'POST'])
639 def deletePage(user_id):
640     user = User.query.get(user_id)
641     artworks = Artwork.query.filter_by(user_id=user_id).all()
642     return render_template('deleteArt.html', user=user, user_id=user_id, artworks=artworks, currentUser=user)
643
644 @app.route('/deleteArt/cint:user_id', methods = ['GET', 'POST'])
645 def deleteArt(user_id):
646     user = User.query.get(user_id)
647     artworksSelected = request.form.getlist("artworkToDelete")
648
649     if request.method == 'POST':
650         for artwork_id in artworksSelected:
651             artwork = Artwork.query.get(artwork_id)
652             filename = (artwork.url).partition(user.userName)[2]
653             if artwork:
654                 client = boto3.client(
655                     's3',
656                     aws_access_key_id= ACCESS_KEY,
657                     aws_secret_access_key= SECRET_KEY,
658                     region_name=AWS_REGION
659                 )
660                 client.delete_object(Bucket="artvisionbucket", Key="artgallery/" + user.userName + "/" + filename)
661                 db.session.delete(artwork)
662             db.session.commit()
663             return redirect(url_for('userProfile', user_id=user_id, currentUser=user))
664         else:
665             return (redirect(url_for('explore'))))
666     # Need query to find and delete art from database
667
668
669
670
671 @app.route('/error404')
672 def error404():
673     return render_template ('error404.html')
674
675 @app.route('/logout')
676 def logout():
677     # session.pop('logged_in', None)
678     session['logged_in'] = False
679     session.pop('user_id', None)
680     return redirect(url_for('index'))

```

```

681
682 @app.route('/deleteAccount')
683 def deleteAccount():
684     user_id = session['user_id']
685     user = User.query.get(user_id)
686     artworks = Artwork.query.filter_by(user_id=user_id).all()
687     comments = Comment.query.filter_by(author_id=user_id).all()
688
689     for artwork in artworks:
690         delete_artwork_from_s3(artwork.url)
691         db.session.delete(artwork)
692
693     for comment in comments:
694         db.session.delete(comment)
695
696     db.session.delete(user)
697     db.session.commit()
698
699     session['logged_in'] = False
700     session.pop('user_id', None)
701
702     return redirect(url_for('explore'))
703
704
705 @app.route('/deleteComment/<int:comment_id>/<int:artwork_id>')
706 def deleteComment(comment_id, artwork_id):
707     form=CommentForm()
708     artwork = Artwork.query.get(artwork_id)
709     artist = User.query.get(artwork.user_id)
710     user = User.query.get(session['user_id'])
711     Comment.query.filter_by(id=comment_id).delete()
712     comments = Comment.query.filter(artwork_id=artwork_id).all()
713
714     db.session.commit()
715     return redirect(url_for('artworkDetails', artwork=artwork, user=user, currentUser=user, artist=artist, form=form, comments=comments, artwork_id=artwork_id))
716
717 #Routes for user galleries
718 @app.route('/options/<int:user_id>')
719 def gallery(option, user_id):
720     user = User.query.get(user_id)
721     if option == 'gallery':
722         artworks = Artwork.query.filter_by(user_id=user_id, shop_item=False).all()
723     elif option == 'shop':
724         artworks = Artwork.query.filter_by(user_id=user_id, shop_item=True).all()
725     serialized_artworks = (artwork.id : "id": artwork.id, "title": artwork.title, "price": artwork.price, "url": artwork.url) for artwork in artworks)
726     return jsonify(serialized_artworks)
727
728
729 #Routes for explore
730 @app.route('/category/<category>')
731 def exploreCategories(category):
732     artworks = Artwork.query.filter_by(category=category).all()
733     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)
734     return jsonify(serialized_artworks)
735
736
737
738 @app.route('/editPage/<int:user_id>', methods = ['GET', 'POST'])
739 def editPage(user_id):
740     user = User.query.get(user_id)
741     artworks = Artwork.query.filter_by(user_id=user_id).all()
742     return render_template('editAccount.html', user=user, user_id=user_id, artworks=artworks, currentUser=user)
743
744 @app.route('/editAccount/<int:user_id>', methods = ["GET", "POST"])
745 def editAccount(user_id):
746     user = User.query.get(user_id)
747     artworks = Artwork.query.filter_by(user_id=user_id).all()
748     update = False
749
750     if request.method == "POST":
751         ##UPDATE USER PROFILE INFORMATION
752         if request.form.get('profile-change') == "Update":
753             newUsername = request.form.get("newUsername")
754             newDisplayName = request.form.get("newDisplayName")
755             newPronouns = request.form.get("newPronouns")
756             newTitle = request.form.get("newTitle")
757             newBio = request.form.get("newBio")
758
759             if newDisplayName:
760                 user.name = newDisplayName
761                 update = True
762
763             if newPronouns:
764                 user.pronouns = newPronouns
765                 update = True
766
767             if newTitle:
768                 user.title = newTitle
769                 update = True
770
771             if newBio:
772                 user.bio = newBio
773                 update = True
774
775             userNameCheck = User.query.filter_by(userName=newUsername).first()
776             # print(userNameCheck)
777             currentUsername = user.userName
778             if newUsername and userNameCheck is None:
779                 user.userName = newUsername
780
781             # setting up client
782             # client = boto3.client(
783             #     's3',
784             #     aws_access_key_id = ACCESS_KEY,
785             #     aws_secret_access_key = SECRET_KEY ,
786             #     region_name=AWS_REGION
787             # )
788
789             # transferring profile photo to new userName
790
791             # Saving profile photo
792             response = requests.get(user.profilePhotoLink)
793             url_parts = user.profilePhotoLink.split("/")
794             filename = url_parts[-1]
795

```

```

782 # aws_secret_access_key = SECRET_AZ1 +
786 # region_name=AWS_REGION
787 # )
788
789 # transferring profile photo to new username
790
791 # saving profile photo
792 response = requests.get(user_profilePhotoLink)
793 url_parts = user_profilePhotoLink.split("/")
794 filename = url_parts[-1]
795
796 with open(filename, 'wb') as f:
797     f.write(response.content)
798
799 # uploading profile photo
800 client.upload_file(filename,"artvisionbucket", "profilephoto/" + user.userName + "/" + filename)
801
802 bucket_name = "artvisionbucket"
803 s3_key = "profilephoto/" + user.userName + "/" + filename
804 url = "https://"+bucket_name+s3[AWS_REGION].amazonaws.com/s3_key"
805 user_profilePhotoLink = url
806 client.delete_object(Bucket="artvisionbucket", Key="profilephoto/" + currentUsername + "/" + filename)
807 os.remove(filename)
808
809 for artwork in artworks:
810     url = artwork.url
811     url_parts = url.split("/")
812
813     url_parts[4] = newUsername
814     newUrl = "/".join(url_parts)
815     artwork.url = newUrl
816
817     response = requests.get(url)
818     filename = url_parts[-1]
819
820     with open(filename, 'wb') as f:
821         f.write(response.content)
822
823     client.upload_file(filename,"artvisionbucket", "artgallery/" + user.userName + "/" + filename)
824     bucket_name = "artvisionbucket"
825     s3_key = "artgallery/" + user.userName + "/" + filename
826     url = "https://"+bucket_name+s3[AWS_REGION].amazonaws.com/s3_key"
827     client.delete_object(Bucket="artvisionbucket", Key="artgallery/" + currentUsername + "/" + filename)
828     os.remove(filename)
829     update = True
830 db.session.commit()
831
832 ##UPDATE ACCOUNT INFORMATION
833 if request.form.get('account-change') == "Update":
834     ##Verification
835     user = User.query.filter_by(id=user_id).first()
836     passwordInDatabase = user.password
837     emailInDatabase = user.email

```

```

831
832 ##UPDATE ACCOUNT INFORMATION
833 if request.form.get('account-change') == "Update":
834     ##Verification
835     user = User.query.filter_by(id=user_id).first()
836     passwordInDatabase = user.password
837     emailInDatabase = user.email
838
839     currentEmail = request.form.get("currentemail")
840     currentPassword = request.form.get("currentpassword")
841
842     ##Proposed changes
843     newEmail = request.form.get("newemail")
844     confirmNewEmail = request.form.get("confirmnewemail")
845
846     newPassword = request.form.get("newpassword")
847     confirmNewPassword = request.form.get("confirmnewpassword")
848
849     ##Error checking
850     errors = []
851
852     #check for inputs for current email and current password
853     if currentEmail == "":
854         error = "Please enter your current email."
855         errors.append(error)
856
857     if currentPassword == "":
858         error = "Please enter your current password."
859         errors.append(error)
860
861     # checks database for email matching current user.
862     if currentEmail and currentEmail != emailInDatabase:
863         error = "Incorrect email! Please try again."
864         errors.append(error)
865
866     # checks database for password matching current user.
867     if currentPassword and currentPassword != passwordInDatabase:
868         error = "Incorrect password! Please try again."
869         errors.append(error)
870
871     # makes sure new email matches.
872     if newEmail and newEmail != confirmNewEmail:
873         error = "Your new email inputs do not match. Please try again."
874         errors.append(error)
875
876     # makes sure new password matches and meet requirements.
877     if newPassword and newPassword != confirmNewPassword and passwordValidation(newPassword)== False:
878         error = "Your new password inputs do not match and/or do not meet password requirements."
879         errors.append(error)
880
881     if (len(errors) > 0):
882         return render_template('editAccount.html', user=user, user_id=user_id, artworks=artworks, currentUser=user, errors=errors)
883

```

```

850
851 #error checking
852 errors = []
853 #check for inputs for current email and current password
854 if currentEmail == "":
855     error = "Please enter your current email."
856     errors.append(error)
857
858 if currentPassword == "":
859     error = "Please enter your current password."
860     errors.append(error)
861
862 # checks database for email matching current user.
863 if currentEmail and currentEmail != emailInDatabase:
864     error = "Incorrect email! Please try again."
865     errors.append(error)
866
867 # checks database for password matching current user
868 if currentPassword and currentPassword != passwordInDatabase:
869     error = "Incorrect password! Please try again."
870     errors.append(error)
871
872 # makes sure new email matches.
873 if newEmail and newEmail != confirmNewEmail:
874     error = "Your new email inputs do not match. Please try again."
875     errors.append(error)
876
877 # makes sure new password matches and meet requirements.
878 if newPassword and newPassword != confirmNewPassword and passwordValidation(newPassword)!= False:
879     error = "Your new password inputs do not match and/or do not meet password requirements."
880     errors.append(error)
881
882 if (len(errors) > 0):
883     return render_template('editAccount.html', user=user, user_id=user_id, artworks=artworks, currentUser=user, errors=errors)
884
885 else:
886     if newEmail:
887         user.email = newEmail
888         update = True
889
890     if newPassword:
891         user.password = newPassword
892         update = True
893
894     db.session.commit()
895
896 if update == True:
897     message = "Information updated"
898 if update == False:
899     message = "Information unchanged"
900 db.session.commit()
901 return redirect(url_for('userProfile', user_id=user_id, message=message))
902

```

```

903
904 # makes sure new password matches and meet requirements.
905 if newPassword and newPassword != confirmNewPassword and passwordValidation(newPassword)!= False:
906     error = "Your new password inputs do not match and/or do not meet password requirements."
907     errors.append(error)
908
909 if (len(errors) > 0):
910     return render_template('editAccount.html', user=user, user_id=user_id, artworks=artworks, currentUser=user, errors=errors)
911
912 else:
913     if newEmail:
914         user.email = newEmail
915         update = True
916
917     if newPassword:
918         user.password = newPassword
919         update = True
920
921     db.session.commit()
922
923 if update == True:
924     message = "Information updated"
925 if update == False:
926     message = "Information unchanged"
927 db.session.commit()
928 return redirect(url_for('userProfile', user_id=user_id, message=message))
929
930 if __name__ == '__main__':
931     app.run(host='0.0.0.0', debug=True)
932
933 # app.run(debug=True)
934

```

○ Requirements.txt:

```
≡ requirements.txt
1  alembic==1.11.1
2  blinker==1.6.2
3  boto3==1.28.12
4  botocore==1.31.12
5  certifi==2023.7.22
6  charset-normalizer==3.2.0
7  click==8.1.6
8  docopt==0.6.2
9  Flask==2.3.2
10 Flask-Migrate==4.0.4
11 Flask-SQLAlchemy==3.0.5
12 Flask-WTF==1.1.1
13 idna==3.4
14 importlib-metadata==6.8.0
15 importlib-resources==6.0.0
16 itsdangerous==2.1.2
17 Jinja2==3.1.2
18 jmespath==1.0.1
19 Mako==1.2.4
20 MarkupSafe==2.1.3
21 pipreqs==0.4.13
22 python-dateutil==2.8.2
23 python-dotenv==1.0.0
24 requests==2.31.0
25 s3transfer==0.6.1
26 six==1.16.0
27 SQLAlchemy==2.0.19
28 typing_extensions==4.7.1
29 urllib3==1.26.16
30 Werkzeug==2.3.6
31 WTForms==3.0.1
32 yarg==0.1.9
33 zipp==3.16.2
34
```

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.