COMPX576 Project Summary

Gallery

Project Description

Gallery is a unique interactive online space designed specifically for painters to showcase not only their finished works but also the entire creative journey behind each piece. Unlike traditional art platforms that primarily display completed artworks, The Gallery Platform enables painters to upload multiple stages of a single artwork, including initial sketches, draft versions, coloring, and final touches. This allows viewers to follow and appreciate the evolution of the work.

The platform's chronological display offers a deeper understanding of the painters' methods, techniques, and creative decisions, fostering a more intimate connection between viewers and the artwork. By organizing works in a timeline view, users can record and reflect on their artistic progress over time. Each stage of the creation process is not just a visual presentation but also an opportunity for users to share their thoughts, challenges, and breakthroughs, encouraging a dynamic form of storytelling. In this way, the journey of creating a piece of art becomes as compelling and important as the final product itself.

For viewers, experiencing artworks on Gallery transforms into an immersive journey through evolving narratives. They can follow an painters' work from conception to completion, gaining richer insights into the creative process. This chronological presentation deepens their appreciation of the painters' methods and decisions, making the process as meaningful as the final result.

Moreover, The Gallery Platform integrates social features designed to cultivate a vibrant community. With real-time communication tools such as integrated chat, users can connect, exchange feedback, and engage in collaborative discussions. This interaction elevates the platform from a static gallery to a dynamic creative hub, where users can support one another, share ideas, and inspire each other in real time.

Completed Functionalities

Authentication and User Management

The Gallery platform utilizes Django's built-in authentication framework to manage user authentication and registration securely. This system includes features such as password hashing to enhance account security. Non-logged-in users can search and view artwork, but full access to functionalities like uploading art, managing profiles, and engaging in social features requires authentication. Additionally, users can log out easily via the "Logout" option in the navigation menu, promoting a seamless and secure user experience.

Artwork Management

The Gallery platform features a robust art management system that allows user to showcase their work flexibly. User can upload multiple images illustrating various stages of their creations, such as sketches and final products. Users can easily adjust the display order of images through drag-and-drop functionality, enhancing the visual presentation of their work. The platform also automatically sets the most recently uploaded image as the portfolio cover for optimal display.

Additionally, users have the ability to rename or delete entire albums, granting them complete control over their content. A key advantage is the automatic saving of all changes, ensuring a smooth user experience without the need for manual saving. The platform supports image enlargement, allowing users and visitors to view detailed versions of artworks, enhancing the overall viewing experience. This comprehensive suite of art management tools, along with real-time updates and intuitive workflows, empowers user to efficiently manage and showcase their creative processes.

Search

The Gallery platform features an efficient and user-friendly search system that enables users to quickly locate artworks and users of interest. Users can search using keywords, allowing them to find specific artworks or users profiles by scanning album titles and usernames. The search results include thumbnail previews of the artworks, making it easier for users to visually identify the pieces they seek. This streamlined search functionality significantly enhances the user experience by facilitating efficient navigation through the platform’s extensive collection of artworks and user profiles.

Public Chat

The Gallery platform incorporates a real-time public chat system using WebSocket, facilitating seamless communication among users. This feature fosters an interactive environment where users can exchange ideas, provide feedback, and engage in discussions about art in real time. Additionally, the system includes an online user list that displays all participants in the public chat room, offering quick access to each user’s profile. By clicking on a username, users can easily navigate to the painters' personal homepage to view their work, enhancing collaboration and community engagement within the platform.

Private Chat

The platform features a one-on-one private chat room functionality, enabling users to engage in personal conversations with other users. The private chat utilizes a polling mechanism for periodic updates, enhancing performance and reducing server load during individual interactions. All conversation records are stored in the database, allowing users to access their complete chat history at any time. The chat interface displays the entire conversation history, enabling users to seamlessly resume discussions. Additionally, users can delete conversations, providing them with control over their chats and ensuring their personal space and privacy are maintained.

AWS Could

The project is entirely deployed in the cloud utilizing AWS services, ensuring both scalability and reliability. The MySQL database is managed through Amazon RDS, which streamlines setup, maintenance, and scaling processes. Both front-end and back-end components are hosted on an EC2 Windows instance, offering a flexible environment for server configuration and application logic execution, leading to efficient request handling and data processing. Additionally, user-uploaded images are securely stored in an S3 bucket, providing a reliable solution for image management.

Unimplemented Functionalities

User Profile Update

In my initial plan, users would have the ability to change their username and update their password through their profile settings. However, due to a lack of clear definition for this feature during the early stages of development, I inadvertently overlooked it in the later phases of the project. As a result, this functionality remains unstarted and would require further development to be included in future updates.

Key Steps to Use the Application

Most of the Gallery features require the user to log in. If the user is a new user, the system will guide them through the account registration process; if the user already has an account, they can log in directly.

After logging in, the user can perform multiple operations. First, the user can use the search function to find other users' art albums and browse their homepages. If the user is interested in an author, the user can also communicate with the author one-on-one through the private chat function.

The system also provides public chat rooms where users can have open discussions with other users. Users can choose to further browse other users' personal homepages and have the opportunity to establish deeper connections through private chats.

In addition, users can manage their own art albums. The system supports users to create new albums and add or edit artworks. If the user does not need to create a new album, the system also allows them to manage existing albums and perform operations such as modifying, deleting, and reordering.

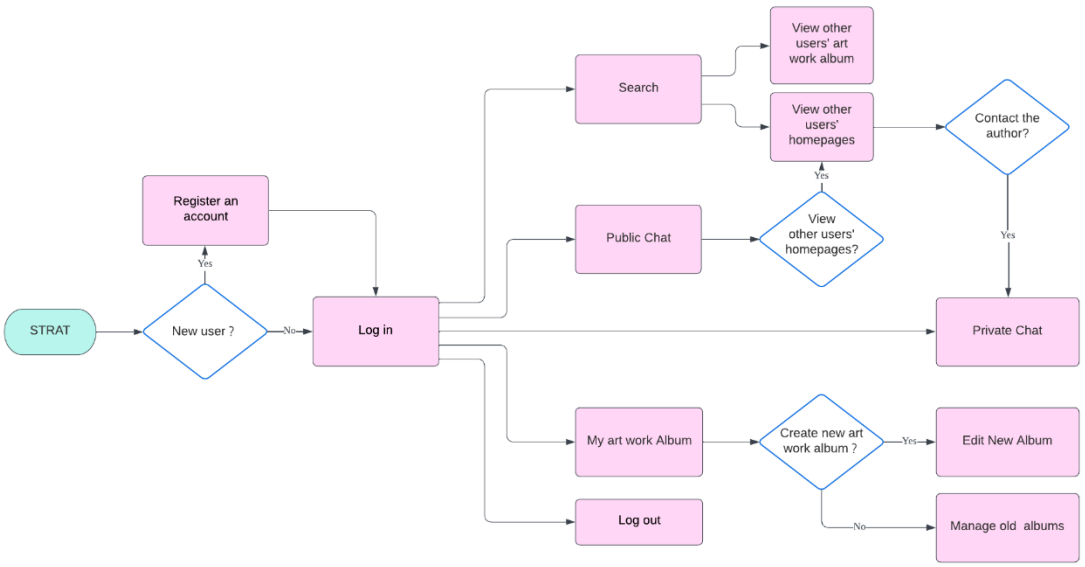

Figure 1 User Flow Diagram

Technologies & Architecture

AWS Could

The main technologies in the gallery's cloud architecture include:

- **Amazon EC2**: As a web server, it handles requests from users. EC2 instances interact with RDS databases through TCP/IP connections to perform data queries and write operations (Amazon Web Services, n.d.).
- **RDS MySQL**: Provides relational database services for storing and managing application data, supports high availability and scalability, and ensures data security and consistency.
- **Amazon S3**: Used to store image data. EC2 instances communicate with S3 through HTTP/HTTPS APIs and are mainly responsible for uploading and managing image files (Amazon Web Services, n.d.).

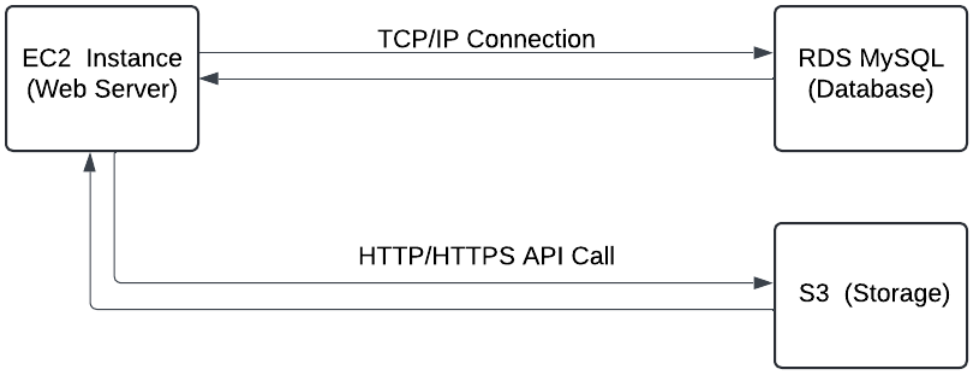

Figure 2 Could Architecture

Front End & Back End

In the architecture of the gallery system, the front-end and back-end separation mode is adopted. The main technologies include:

- **React:** React used to build dynamic and highly interactive user interfaces, using component-based development so that the interface can respond quickly to user operations. API requests are initiated through the Axios library to communicate efficiently with the back-end (Axios, n.d.).
- **Django**: Django provides powerful functions and a rich ecosystem, supporting built-in routing and authentication functions. Using Django's ORM can simplify database operations and improve code readability and maintainability (Khatri, 2024).

- **MySQL**: As a persistent data storage, it is responsible for saving user information and business data, supporting large amounts of data and high-concurrency requests with high performance and reliability. Combined with Django's ORM, it ensures data consistency and integrity.

Figure 3 Front End & Back End

Real-time chat

In the architecture of the real-time chat system, a variety of advanced technologies are used to ensure the efficiency and scalability of the system.

- **WebSocket:** A communication protocol that enables full-duplex, two-way communication between browsers and servers (Dudik, 2024).
- **Daphne ASGI server**: An ASGI (Asynchronous Server Gateway Interface) server that handles WebSocket connections and HTTP requests asynchronously (ASGI, 2018).
- **Channel layer:** A middleware component that facilitates message routing and distribution.
- **Redis server:** Acts as the backend of the channel layer, providing publish/subscribe functionality and temporary data storage (Redis, n.d.).
- **WebSocket consumer:** A server-side component that handles incoming WebSocket messages and manages chat logic.

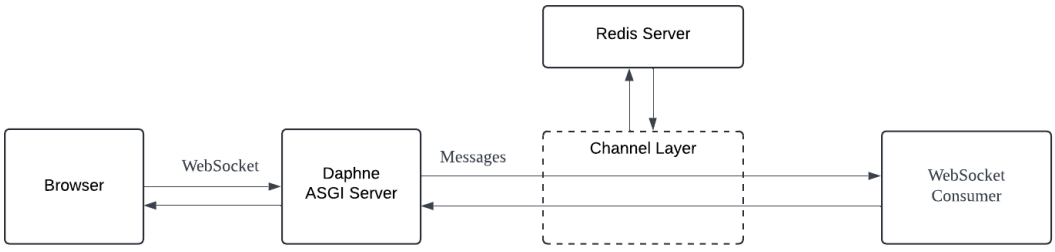

Figure 4 Real-time Chat System

Challenges & Solutions

New technology

One of the main challenges I faced during development was a knowledge gap. I was unfamiliar with the new technology I was using. In particular, it was not clear to me at first how the various components of the live chat system worked together.

To address this issue, I spent time reviewing technical documentation related to the technology. This included reading guides, tutorials, and API references that explained how different components interacted.

Lack of experience

Due to a lack of experience, I encountered several issues during development. Many small bugs occurred, which affected the overall progress of the project. These bugs often required extensive debugging and testing, taking away valuable time from developing new features.

To overcome this challenge, I searched online for development logs and project documentation from others who had worked on similar projects. By reviewing their experiences and potential solutions, I was able to identify best practices and troubleshooting techniques.

Future work

The gallery still needs more features to improve the user experience. I plan to add more features in the future, such as:

- **Like:** Implement a "Like" feature for illustrations. When an illustration receives more Likes, it increases its visibility to other users. This social proof can encourage more interaction and help highlight popular works within the community.
- **Favorites:** Introduce a feature that allows users to save their Favorite works. This will enable users to easily access and find their preferred illustrations later. A Favorites section can enhance user experience and promote engagement with content they enjoy.
- **Business Opportunities:** Create a trading channel for users that facilitates various business interactions, such as rewarding painters, purchasing copyrights, and handling commissions. This feature can provide a monetization avenue for painters and encourage collaborations, benefiting both creators and buyers.

References

Amazon Web Services. (n.d.*). Automatically connecting an EC2 instance and a DB instance*. Retrieved from https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/ec2-rds-connect.html

Amazon Web Services. (n.d.). *Use Amazon S3 with Amazon EC2 instances*. Retrieved from https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AmazonS3.html

ASGI. (2018). *ASGI (Asynchronous Server Gateway Interface) specification*. Retrieved from https://asgi.readthedocs.io/en/latest/specs/main.html

Axios. (n.d.). *Axios*. Retrieved from https://github.com/axios/axios

Dudik, S. (2024). *Mastering real-time communication: A comprehensive WebSocket tutorial*. Retrieved from https://medium.com/@sergey.dudik/mastering-real-time-communication-a-comprehensive-websocket-tutorial-0f6cf384d1e8#:~:text=WebSocket%20is%20a%20communication%20protocol,for%20real%2Dtime%20data%20transfer.

Khatri, S. (2024). *Python Django framework: An engineering leader's guide*. Retrieved from https://www.index.dev/blog/python-django-maximizing-efficiency-across-engineering-product-and-hiring

Redis. (n.d.). *Redis Pub/Sub*. Retrieved from https://redis.io/docs/latest/develop/interact/pubsub/