|  |  |
| --- | --- |
| School of Electronic Engineering and Computer Science    Final Year  Undergraduate Project 2020/21  QMLogo | **Interim/Final Report**  **Programme of study:**  Computer Science BSc  **Project Title:**  **ArtNote**  **Supervisor:**  George Fazekas  **Student Name:**  Duy Huu Nguyen  Date: 20/11/2020 |

Abstract

Art demands artists to be creative and requires a lot of focus and dedication to an artwork, thus rumination occurs and potentially causes depression, fortunately, with the use of social media, this problem can be resolved.

However, the problem is that painting session takes a lot of time and focus, thus putting artists in a position where they could hardly find gaps to relieve stress by surfing social media platform. In addition, artists ’s mental health are affected by many elements, such as the people who judges their art or even themselves. The reason for this is the fact that different people see the world differently, thus making art subjective, meaning each artists ‘s ability to paint or their perspective on the world will be different from their audience. The crucial element about this is that artists tend to judge their ability based on their audience feedback, which are subjective, meaning that their artwork might be beautiful in ones perspective but others might see it differently. All of these aspects contribute the build up of anxiety in artists and is harmful to their mental health, potentially causing depression. The goal of the report is to explain the solution to tackle this problem by building a platform that encourages socialization that artist could easily adapt to as discussed later in the report, socialization is a good method for fighting against depression.

Lastly, the report will also be capturing the functional and non functional requirements for this social platform as well as the high level and low level design and implementation of this social platform. In addition, potential risks that may occur during development process will also be mentioned to prevent the development cycle from failing.

**Table of Contents**

**[Chapter 1: Introduction 2](#_Toc14811)**

[1.1 Problem 3](#_Toc25989)

[1.2 Objectives 3](#_Toc20663)

[1.3 Scope 4](#_Toc16573)

**[Chapter 2: Research 7](#_Toc10056)**

[2.1 Depression 7](#_Toc11653)

[2.1.1 What is depression? 7](#_Toc5157)

[2.1.2 Artist and depression 7](#_Toc16233)

[2.1.3 Treatment 8](#_Toc32416)

[2.2 Digital art 8](#_Toc8872)

[2.2.1 What is digital art? 9](#_Toc25469)

[2.2.2 Tools 9](#_Toc25703)

[2.3 Django. 10](#_Toc30940)

[2.3.1 What is Django? 10](#_Toc4499)

[2.3.2 Usage 10](#_Toc12827)

[2.3.3 What is web server? 10](#_Toc2477)

[2.3.4 What ASGI & WSGI? 10](#_Toc11240)

[2.3.5 ASGI & Web socket 12](#_Toc3347)

[2.3.6 Django channels 14](#_Toc2539)

[2.4 Real time application. 14](#_Toc2698)

[2.4.1 Example of real time application and its implementation 14](#_Toc23208)

[2.5 Application critics. 15](#_Toc3599)

[2.5.1 Existing application with similar requirements 15](#_Toc31625)

[2.5 Conclusion. 16](#_Toc826)

[2.6 Summary. 17](#_Toc1593)

**[Chapter 3: Requirement specifications 17](#_Toc20411)**

[3.1 Functional requirement. 17](#_Toc22951)

[3.2 Non-functional requirement. 20](#_Toc11960)

[3.3 Use case examples. 20](#_Toc9337)

[3.4 Summary. 24](#_Toc15375)

**[Chapter 4: Design and Implementation 25](#_Toc17328)**

[4.1 Design 25](#_Toc22227)

[4.1.1 Overview 25](#_Toc8137)

[4.1.2 Design pattern 25](#_Toc31864)

[4.1.3 Framework 26](#_Toc17322)

[4.2 High level design 27](#_Toc2692)

[4.2.1 Model 27](#_Toc1041)

[4.2.2 View 28](#_Toc717)

[4.2.3 Consumer 33](#_Toc865)

[4.2.4 Interface 34](#_Toc3711)

[4.2.4 Database 39](#_Toc32005)

[4.3 Low level design 40](#_Toc7812)

[4.3.1 Model 40](#_Toc23746)

[4.3.2 URL patterns and views 44](#_Toc15368)

[4.3.3 Consumer and Route: 51](#_Toc2471)

[4.3.4 Interface 60](#_Toc29222)

[4.4 Minimum vial product (MVP) 64](#_Toc18821)

[4.5. Evaluation 66](#_Toc16904)

[4.5.1 Evaluating the project ideas 67](#_Toc2317)

[4.5.2 Summary 73](#_Toc23808)

**[Chapter 5: Risk management 74](#_Toc30223)**

**[Chapter 6: Progress summary (To be finished later) 78](#_Toc31840)**

**[Reference 79](#_Toc1125)**

**[Appendices 81](#_Toc11548)**

# **Chapter 1: Introduction**

In the first chapter, I am going discuss the about the origin of the project, indicating the purpose of the project, what problem is required to be solved

## 1.1 Problem

Creativity is a crucial part of an artist’s life as it is the engine that keeps them motivated and focused during long painting sessions. However, even though creativity might be seen as a powerful tool that lead to success, it also might become a wall that kills the artist. Based on an article by Tanner(2013),in order to be creative, rumination is required, and this is agreed by many psychologists and psychiatrists to be the source of depression in artists. In addition, Tanner also states that rumination a process that demands logical/connective thinking in order to sculpt/combine small details and “Creatives naturally tend to think more and think about their every thought too”. Due to this, it is stated that those who ruminate tends to be impacted by pain and suffering more often, because artists is a group that gravitate towards stressful events from time to time. Fortunately, there are methods for alleviate depression, one of it is socializing. According to Nuacht UCD(2012),the more socializing ones become the more likely the person’s self-esteem is rebuilt, which possibly implies the growth in having healthy and positive relationships.

Artists tend to have deep thoughts while working on artworks. Nonetheless, this could lead self-isolation due to lack of socializing because of long hours of work. In addition, this is an act of rumination, which is agreed by psychologists and psychiatrists to be the cause of depression because not only do artists have to loop through an experience again and again due to deep thinking, they also might not be able to share to those experiences to anyone because of self-isolation.

## 1.2 Objectives

This project aims to aid artist in easing the looping cycle of thoughts in artists due to rumination and creating a platform where artists can socialize with each other while still be able to work simultaneously. Because of this, the goal of the project to is to create a platform where artists can interact with each other through socializing and painting, the project can be a great source of inspiration and entertainment for artist as this can act as a meeting for artist to come to paint and have conversation with each other.

Below are the main objectives of the project:

* Design an application for sharing artworks
* Design a gallery system for displaying artworks that can be viewed by users(artists)
* Design a chatting platform for artists to come for chatting and painting
* The design allows artists to interact with the canvas in real time
* Design canvas and brush systems with UI and functionality that artists are familiar with
* Research on similar products, evaluate their pros and cons
* Do literature review on work relate to Depression, socialization for artist
* Derive from the research the methods, algorithms and techniques need to be applied to and what can be improved on the project based on other similar projects
* Decide on what tool/languages/framework to use for implementation
* Decide which DBMS to use for implementation

## 1.3 Scope

* DELIVERABLE

The project is a web application that allows user to create/join rooms with other users to communicate through voice/text and paint on the same canvas. In addition, each users will have their own gallery which allows them to share artworks and interact with other users on the platform, while having their mood monitored by the system which will then provide suitable artworks from other users that help improve mental health .

* TIMELINE

# **Chapter 2: Research**

## 2.1 Depression

In this section, I will be discussing about the definition of depression and the effects it has on a person. Moreover, it is also important to research about what impact these effects has on artists and the treatments that minimize or help people with depression to recover from it.

### 2.1.1 What is depression?

Based on an article from NHS(2018), depression is a type of symptom disorder that has impact on ones action, feeling and appetite. Several form of depression are:

* Persistent depressive disorder: the person with this form will have various experiences of major depression together with less serious symptoms.
* Psychotic depression: is a form of depression ,in which the person experiencing the symptoms is also affected by psychosis such as delusion, find it is unpleasant in things other people don’t(hallucination)…
* The article stated that a person with depression may suffer from some of these symptom:
  + Anxious or empty mood
  + Hopelessness
  + Loose interest in activities
  + Struggle to focus and concentration
  + Suicide thoughts
  + From these signs we could conclude that depression not only pessimistically alters a person’s mental health but also might have possibly to give a person suicide thoughts.

### 2.1.2 Artist and depression

Ever since the old days of art, artists and mental health symptom, specifically depression, have always been believed to have a connection with each other. According to Dodgson(2018), there has always existed a connection between popular artists in the past and hallucination/delusion, a typical example is the famous Vincent Van Goth, who servers his own ears due to a fight with his friends Paul Gauguin and committed suicide 2 years later. It was described by the artist that: ”I am unable to describe exactly what is the matter with me ”,” Now and then there are horrible fits of anxiety, apparently without cause, or otherwise a feeling of emptiness and fatigue in the head … At times I have attacks of melancholy”. With this, it could be concluded that without self-awareness or being too dedicated, a person will be affected or even absorbed by anxiety and hollow, which could potentially leads to death. In addition, it was also mentioned in the article, people who work an artistic job are more likely to die by suicide as reported from a study the Office of National Statistic. This once again, strengthen the connection between the artists and the deadly symptoms, depression.

Furthermore, an article by Terranova(2020), mentioned that they need loneliness in order to be able to create arts. Even though self isolation helps with being focus and creative, it is hard to create connection with other all the time and make ones feel lonely. It was also mentioned artist tends to experience loneliness and this is due to not knowing other artists or even worst, not having a chance to socialize with other people with the same interest. Terranova also said that, sometime ones have to fit into a “clique” in order to be accepted, which is the problem with todays social media platform, if you want to be on the trending tab, you need to follow what the algorithm tells you do.

### 2.1.3 Treatment

Fortunately, there many ways to ease the effects of depression and mental issues, one of those is socialization. Based on an article by Bulfut(2019), psychologist find that an individual’s health and mental is improved tremendously when they are surrounded by people who can understand them, share similar challenges, these elements create a feeling of warmth and mutual understanding, thus making the individual feel cheerful and comfortable. It is also mentioned, unorganised and informal meeting/conversation, face-to-face interaction could positively leave an impact on a person’s lifestyle and mental state.

Another type of treatment is art therapy, in which a person will use activities such as sketching, journaling, painting,… in order to artistically express themselves ,thus relieve stress, written in article from rtor.org(2018).It also mentioned that activities done together as a group assist the connection between people and people, which is likely to minimize the effect of depression. In addition, it is also stated that during the process of art therapy, one does not only have a chance for self-exploration but also for connecting with others, hence overcome emotional difficulties.

It was also mentioned by Terrranove(2020) that one of the way artist tackle loneliness is by finding each other to connect and to talk about what they have in common. This allows for opportunities to meet new people and socialization, fortunately, with the help of internet, it is easier than ever to find people with the same interest to talk to. Because of this, I decided that the project will be a web application that artist could easily access and find the right type of people to socialize with as the platform’ s main audience is the art community. Furthermore, the application will tries to adapt to what artists are most familiar with, which is painting.

## 2.2 Digital art

### 2.2.1 What is digital art?

Traditional art is form a of painting using traditional medium such as brushes, canvases, watercolor, gauche,…. On the other hand, digital painting uses technology/devices to paint on machine such as a computer or tablet. According to an article on Concept Art Empire (https://conceptartempire.com/digital-painting/),the usage of technology, for example: painting software, tablet,... has become more popular as artists prefer the long-term benefit in terms of investment and this form of art is also required in a variety of industry including animation, illustration, Game,.… Overall, digital art might has become the standard for working in industry, which possibly means that people has become more familiar with this format and vice versa, this format has become more accessible to more people.

### 2.2.2 Tools

In order to get to start with digital art, a number of tools are required. According to the Adobe’s beginner’s guide to digital drawing tools, the below tools are required:

1. Digital stylus/Smart pen

To start off, a type of digital pen is required, and it is mentioned that there are 2 types of pen: Smart pens and digital stylus. Firstly, the former trails the lines and points the user make when drawing on a special paper, take their data and transfer it via the built-in blue tooth of the pen to a computer in real time. Because of this, user’s drawing will appear on the screen simultaneously as they drag the pen across the paper. However, the later gives a more natural and traditional feel to the process as it allows user to draw directly on a surface of a device such as a graphic tablet.

1. Device

Base on the type of stylus/pen recommend above by adobe, we could conclude that, some of the devices required might be a computer or a tablet with stylus functionality. In addition , it might be possible to that the choice of which type of those devices is depend on ones work-flow. The reason for this is while a computer might give the user powerful tools, a broad view of their current project due to seeing only the artwork on the computer’s screen while working, it is also a stationary device which means portability is a problem due to the size of the device. In contrast to this, a small tablet provides portability ,user could utilize this to comfortably do their job where they want to, for example, when it would be impossible to sketch outside with a tool such as a computer, however, it would likely be capable with a tablet, while still providing the essential features a computer has. In conclusion, each device has its own benefits and drawbacks, which means each of them could bring its own unique profit to different businesses.

1. Drawing software

Lastly, a type of software is compulsory, this aids the artist in various way one of them is correcting mistakes during painting. Since the medium is a software ,it is feasible to erase/undo any errors, making the work flow more enjoyable.

## 2.3 Django.

### 2.3.1 What is Django?

It was mentioned in the Django homepage,it is a python framework that allows rapid, fast development while providing re-usability saving time and effort from implementing existing code. In addition, it also aids the user with modules that help with authentication, administration and many other tasks. Lastly, Django protects developers by providing strong security preventing SQL injection, cross-site scripting, cross-site request forgery, clickjacking.

### 2.3.2 Usage

Based on Kalpit(2018),Django could be used to develop these applications:

* Web app or API for the back-end
* Quick/rapid development
* Applications that demand strong security
* The code-base encapsulates the web app and the API back-end.
* Django does not require directly making database queries, instead it provides its own interface for database transactions.

### 2.3.3 What is web server?

As written on developer.mozilla.org (2021),a web server can be either a software or hardware, in terms of hardware ,this a storage for website’s resources, for example: HTML, CSS, images, videos,.... and in terms of software, it is an Http server that interprets URL and HTTP, this type of server can be accessed using domain name which then response to the client with the requested resources.On the basis, in order to access a website, client must send an HTTP request to the web server hosting the files that will return those files after validation.

### 2.3.4 What ASGI & WSGI?

Based on an article by Positive Stud(2020):

**WSGI(Web server gateway interface)**: is an interface used to communicate between web server and the python application or frameworks such as Django. For example, once a web server has received requests from many clients, it requires interaction with the python application and for this to happen, WSGI needs to take part in the communication. The negotiation is carried out as follow:

The web server is able to send request to the WSGI container and callable objects with specific functionalities provided by the python application can be called by the WSGI container corresponding to the requests it received. The figure below demonstrates how the process is delivered:

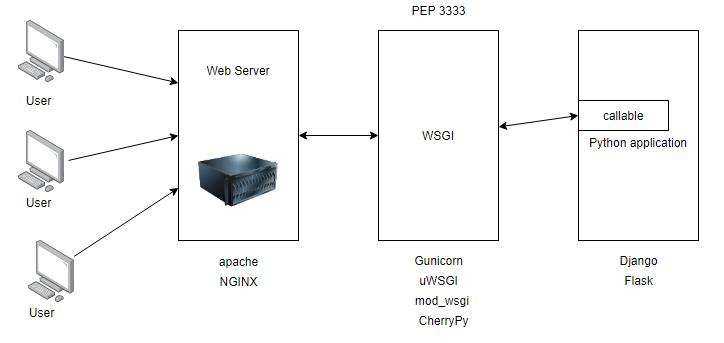


Figure 3. Communication between user, web server, WSGI, and Python application(https://medium.com/)

**ASGI (Asynchronous Server Gateway Interface)**: is derived from WSGI and allows Asynchronous interaction between Python application, frameworks and web server (https://asgi.readthedocs.io/).It was also mentioned that ASGI is constructed using 3 elements:

**scope**: a dictionary containing information about a specific connection. Each connection from a client to the ASGI application is a call to the application callable that handles the connection and “scope” store the life span and information about each connection. For example, for Web socket protocol, the live span of the connection is the time it has connection to the python application and scope stores the connection web socket’s path

**send**: a callable for sending event messages to the client

**receive**: a dictionary containing key-value pair of many event messages to the application.

**Events**: are divided by ASGI into 2:events that the python application receive and event responses that the application send. Each event is a dictionary with a key “type”, describing the type of the event’s message that be freely invented by the user.

|  |  |
| --- | --- |
| **Asynchronous request** | **Synchronous request** |
| Allows clients to send multiple requests and still be able to continue execution while waiting for the response | The client must wait and block until the response is received in order to continue execution |

### 2.3.5 ASGI & Web socket

**Web socket protocol**: according to geeksforgeeks.org, the protocol is a stateful protocol that uses client-server communication, thus making the life span of the connection last until one of the member disconnect. The process is as follow:

1/Client-server agree on a connection by making a handshake and this connection is kept alive as long as both members are still connected

2/While alive, messages in the communication is bidirectional, which means the client and server can both send and receive messages from each other.

3/If either the client or server dies or disconnect, the communication ends, and the connection is closed

**Web socket protocol relationship with asgi**

There are many components for web socket connection through asgi:

**Scope**:

Web socket scope is stored in the scope dictionary of the callable, this contain meta data about the connection and must contain at least a “type” key describing the type of the connection, for example: http or WebSocket. The keys from scope include:

* type: Describing the type of the connection, for example, http or web socket
* asgi[“version”]:version asgi spec
* asgi[“spec\_version”]:the version of asgi that this server interpret
* http\_version:version of http
* scheme:url scheme,such as:ws,http,wss
* path:the url of the request removing the all query string after it
* raw\_path:unmodified http path
* query\_string:portion of the url after “?”
* root\_path:the root path of the application
* headers:a dictionary containing the request’s headers
* client:a key-value pair storing the client’s IP address and listening port
* server:a key-value pair storing the server’s IP address and listening port
* subprotocols:Sub protocols the client advertised

**Connect - receive event**:

This event happens when the server receive requests from the client when the connection is opened and about to be closed. These requests must be responded with an Accept or Close message before being passed. These messages(Accept or Close) must be sent during the handshake process and the handshake only ends if the server receive a reply. The keys from receive include:

* type: websocket.connect

**Accept - send event:**

Sent by a client to accept a connection. The keys include:

* + type:websocket.accept
  + subprotocol:the subprotocol to be to accepted
  + headers:a dictionary storing headers of the message.

**Receive - receive event**:

Messages for the server sent by the client. The keys include:

* + type: websocket.receive
  + bytes:content sent in binary form
  + Text:content sent in text form

**Send- send event:**

A request from the application to send a message to the client. The keys include:

* + type: websocket.send
  + bytes:content sent in binary form
  + Text:content sent in text form

**Disconnect- receive event:**

The application receive this if the connection is either lost due to the client losing connection, the server closing connection or socket loss. The keys include:

* + type: websocket.disconnect
  + code:close code for websocket

**Close- send event:**

Sent by the application requesting the server to close the connection. The keys include:

* + type: websocket.close
  + code:close code for websocket

### 2.3.6 Django channels

Allows handling other protocols apart from http ,such as ,web socket, chat protocol,... and is based on ASGI. In addition, channel also provide authentication, session system and more.

Since channels is based on ASGI, it also split the connection into sections: connection scope and events.

* **Scope**: Web socket scope last as long as the connection is kept alive. The scope is a dictionary storing information about the connections as discussed in the section before.
* **Event**: These events occur while the scope is kept alive and describe user such as sending a web socket frame. For each scope created ,channel will be instantiated to handle the events specific to that scope.

Sequentially, channel have a concept called Consumer, which takes event as input, then using a routing table direct the request to the corresponding consumer that can be specified as asynchronous or synchronous to handle the requests.

## 2.4 Real time application.

Having done the research about WSGI,ASGI and web socket, i am able to have a idea about the basic about the architecture of a web socket based application and that means the application will include the use of Web server ,ASGI server, storing scope and handling event requests sent by the ASGI server to the backend application which will then be “consumed” or processed by the consumer that corresponds to the path the request is sent to. In addition, i believe that in order to understand the implementation better, it might be a good idea to take a look at an example of a real time application that utilizes Django’s channels and a front end framework to develop responsive website.

### 2.4.1 Example of real time application and its implementation

Using reactjs and Django stack , an example of a real time application could be a chat box as provided by Ruben(2018), the implementation is as follow:

**Requirements:**

* Fronted: React yarn(or node.js)
* Apophony 3,pip,redis server, PostgreSQL

**Set up:**

* **Back-end**

1. Creating and migrating Message model for storing messages from users
2. Setting up Channel and point to Redis server location
3. Define the routing table for channels, this directs all web socket requests to the consumer corresponding to the path in the routing table, which will handle the requests.

* **Front-end:**

In App.js of the react app, create a function called “create”, which will add callbacks that are invoked when web socket opens, receives messages closes or error.

## 2.5 Application critics.

Now that the foundation of the implementation has been established as above, we need to examine and evaluate existing application with similar features that the project desires.

### 2.5.1 Existing application with similar requirements

As a result of researching, I found that the availability of real time drawing app is limited, even though there are a variety of visualization application that allows similar features to live drawing, some of them are: Miro ,which is a web application for collaboration that provide tools for visualization such as an infinite canvas along with widgets and the ability to communicate through video calls as well as screen sharing. However, this type of application’s goal is to aid project and meeting management whereas my project is focused on providing chat room and drawing tools that artists are familiar with in art industry. Nonetheless, I was able to find an application called Magma studio ,which is an application that has ideas of chatting through artistic visualization. The website allows users to create their own or join another user’s room to draw and communicate through text or voice chat as well as providing a real time experience, which means that every user in the same room has access to drawing on the same canvas, possibly resulting in relaxing and comfortable socialization. I also wanted the project to allow users to have a platform for sharing the canvas belong to their room and their own artwork in order to interact with other user and express themselves. Despite sharing important features to my application Magma studio has not had a platform for sharing at the moment, because of this , a type of social media app needs to be examine. In terms of sharing pictures/images, i have found several website that specifically aiming at artist: Artstation and pixiv,both of which satisfy the needs for the project. Below is the comparison with other similar application:

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirements** | Magma studio | Artstation | Pixiv |
| **Communication and live drawing** | | | |
| **Room chat box** | Yes |  |  |
| **Live drawing** | Yes |  |  |
| **Room creation** | Yes |  |  |
| **Individual chat box** |  | Yes |  |
| **Canvas** | Yes |  | Yes |
| **Digital tools for painting** | Yes |  |  |
| **Live streaming room activities** |  |  | Yes |
| **Have private or public room** |  |  |  |
| **Send request to join a room** | Yes |  |  |
| **Hosting room** | Yes |  |  |
| **Voice chat** | Yes |  |  |
| **Auto saving canvas of a room** | Yes |  |  |
| **Sharing platform** | | | |
| **Each user has a gallery** |  | Yes | Yes |
| **Posting images** |  | Yes | Yes |
| **Like/Comment on posts** |  | Yes | Yes |
| **Real time notification** |  | Yes | Yes |
| **Post canvas from a room** |  |  |  |
| **Simple and accessible UI** | Yes | Yes | Yes |
| **Profile for each user** |  | Yes | Yes |
| **Discovery page** |  | Yes | Yes |

## 2.5 Conclusion.

As a result of the research, it is possible to conclude that an application focusing on communicating through posting and drawing/chatting is not widely available and the available applications lack the needs for interaction between users as they only a sub part of features in terms of communications, for example ,Magma studio provides the needs to on-line communications through drawing and chatting, however user can only interact with people they know as there is no platform for communicating with other people such similar to Artstation.The project focus on these aspects, which form its attraction points. The project will have to have a simple to use but artistic design as the targeted stakeholders are artists, this also means there need to have a lot of core tools popular in the digital art industry so that user can easily become familiar with the application quickly.

## 2.6 Summary.

In this chapter , I have discussed about the problem needed to be solved, which is depression as well as gave an introduction about digital art and the required tools to start digital art.In addition, Django and its uses/operations has also been explained, this allows for further understanding of how the concept of WSGI and ASGI have been applied into Django channels ,thus allows for sending asynchronous requests making the application works in real time. Finally, I have compared the differences between the project and the different existing software to improve the quality of the project.

# **Chapter 3: Requirement specifications**

Requirement is an essential aspect for a project because it helps form the foundation and the features the project must have in order to achieve the desired result. In this chapter , we will go through the requirements for the projects which consist of : Functional requirements and Non-functional requirements. These requirements should be able to identify the functionality the project must have as well as the performance of the app as it performs these these operations.

## 3.1 Functional requirement.

|  |
| --- |
| **Requirements** |
| **General** |
| System must allow many users to create a drawing room |
| System must allow many users to draw on the same canvas |
| System must provide a list of brushes and lasso functionalities for users to use |
| System must let user create an account that has its own profile |
| Each profile has a gallery storing artworks posted by the profile owner |
| System must allow users to interact with other users and posts |
| System must allow user to switch from gallery mode to drawing room mode |
| System must allow user to invite people to room |
| System must allow user to join a room |
| **User** |
| User must be allowed to create an account |
| User must be authenticated using tokens |
| User must be allowed to create a drawing room |
| User must be allowed to join a room |
| User must be allowed to be a member of many rooms |
| User must be allowed to switch from Drawing room mode to Gallery mode |
| User must be able to login/sign in |
| User must be able to like post |
| User must be able to post |
| User must be able to delete post |
| User must able to receive notification in real time |
| User has these attributes: firstname, lastname, email, password, occupation, working, location, interest’s list, about me section |
| **Drawing mode (DM)** |
| DM must have a collection of room that the user created |
| DM must have a collection of room that the user is a member of |
| DM must allow user to change to Gallery mode |
| DM must display information of each room including: no.member,host’s name,on line status, group’s name |
| DM must allow user to search for a group they are hosting or joined using its name |
| DM must allow user to enter a room number and join it |
| **Drawing room(DR)** |
| Every room must have only 1 single host |
| DR must have a canvas for many users to draw on |
| DR must provide a list of tool for users to use on the canvas |
| DR must provide a layer system for user to use |
| DR must provide a Color Picler for user to use |
| DR must allow user to invite other account to a group. |
| **Creating a room** |
| User must first enter: name of room/search and click on the room to join |
| Host can choose to invite a person by search for their name and invite them |
| A room must have these information: ID,room name, background image |
| **Canvas** |
| User must be allowed to select a portion of the canvas to draw in using the tool system |
| Canvas must store these attributes: ID,room |
| **Layer** |
| System must have a panel for storing and modifying the layers |
| User must be able to create a layer for a canvas |
| User must be able to delete a layer for a canvas |
| A layer must have these properties: mode and opacity |
| A layer can be toggle off/on |
| Layer must store these attribute:ID,Canvas,Image |
| **Tool** |
| List of tools must include: brush, airbrush, eraser, |
| **Brush** |
| User choose the brush tool to use the brush |
| A brush has many tips and has 5 tips by default, default is a round tip |
| A brush has these properties:size,density,opacity,hardness,color |
| User can import and upload their own brush tips |
| User can apply pressure to these properties:size,density,opacity,pressure for size, pressure for opacity |
| **Gallery mode** |
| **Sign up** |
| User is asked for their first name,last name,email,password,password confirmation, occupation, where they are working, other information they would like to provide to their bio. |
| User is asked to choose their interest from a list |
| User is asked to write an About me summary and provide 3 pictures of their artwork |
| System must save into database the user’s firstname, lastname, email, password, occupation, working location, interest’s list, about me section and its 3 pictures and assign an ID number to the user |
| **Login** |
| User must enter their user email ,password |
| System must validate those information with records from database |
| System must provide a message confirming the user that they have logged in successfully |
| System must provide a message confirming the user that they have logged in unsuccessfully |
| User must be able to logout |
| **Post** |
| A post must have these attribute: number of like,currentInteraction(updated every hour),lastInteraction(updated with currentInteraction every 15 minutes), comments, the author. These must be stored in the database and assigned with the author’s ID |
| For a post, a user must be able to like/comment on that post |
| Owner must have option to edit/delete the post |
| Owner while editing the post can delete/add images and change the summary, once the images are changed, the post mood is updated.These information is then used to update the record in the database |
| **Posting a post** |
| User must choose an image before they can post |
| The image along with its owner id is sent to the server to create a new post in the database. |
| **Discovery** |
| The tab must show popular/latest posts |
| Each post on the page can be clicked on to direct the user to the post page, where user can like/comment on the post |
| **User homepage** |
| User homepage has Gallery for storing a user’s posts |
| User homepage must display information about a user such as: first name, last name, occupation, location, about me |

## 3.2 Non-functional requirement.

* **Scalability**: The project is a social and communication application which stores a large number of data that need to be extended once the limit is exceeded, if this happens, the increase in the number of node for storing this data is critical. In order to do this it is important to choose a DBMS that allows scalability, for example, a type of database that allows the increase in the number of nodes like mongo dB.
* **Reliability**: The application is real time performance dependent, this demands the system to be running with minimal crashing chance of 1-5% and alive time of the server must be above 90% of the time together with real time updates of user data and notifications.
* **Latency**: The runs on a large database of users ,which mean there will be a large number of transactions that need to deliver in time so that the user feel satisfies. In order for this to happen, transactions might have to take advantage of indexes in Misusing algorithm from the back-end for sorting data into a specific format in the shortest time possible or sending data in batches so that there is more time for sorting. For the front end, lazy loading could allow some time for the back end to sort data as this only requires the back-end to sequentially send small amount of data.
* **Security**: The system stores and make transactions with user’s data which need to be save and inaccessible to other, this requires a secure way of transferring as well as preventing Injections, Cross-site scripting and other form of hijacking. The application must also protect the privacy of user from different chat room by allowing only the host of the room to be able to accept/decline request from users.

## 3.3 Use case examples.

The application has 2 main pages, draw&chat page and gallery page, in this section, I am going to discuss 2 core use cases for each of them.

##### Draw & Chat

**Use case 1**:Creating a room

Prerequisite: User have created an account on the website

1/Initially, when the user enter page , they will need to provide their user name and password to login

2/Once logged in, if the user has rooms that they are hosting or they have already joined more than 1 room, they will be directed their room’s list(list of rooms they have joined or are hosting).

2.1/User click on the “Create room” button to create a room

2.2/User enter room’s information

2.3/User click submit

2.4/Once the room is created, the user is directed to that room and provided with a canvas belong to that room along with the tool UI

3/If the user is a new account, meaning they have not joined or are hosting a room yet, they are directed to the room create form

3.1/User create room by entering the room info

3.2/User click submit

3.3/User is directed to the created room

**Use case 2**: Draw

Prerequisite: User have created an account on the website

1/Initially, when the user enter page, they will need to provide their user name and password to login

2/if the user has created or joined more than 1 room, they will be presented with a list of those rooms:

2.1/The user click on one of the rooms they want to start drawing in

2.2/The user enter the room and presented with a canvas and tool for drawing

3/If the user has not created a room yet, they have an optional choice to create a room or join a room once login

3.1/User create or join room by following the create/join room use case

3.2/Once one of those use cases is done, they will be directed to the room they have chosen or created

3.3/User start drawing in the room

Below is the Use case diagram for these 2 use cases:

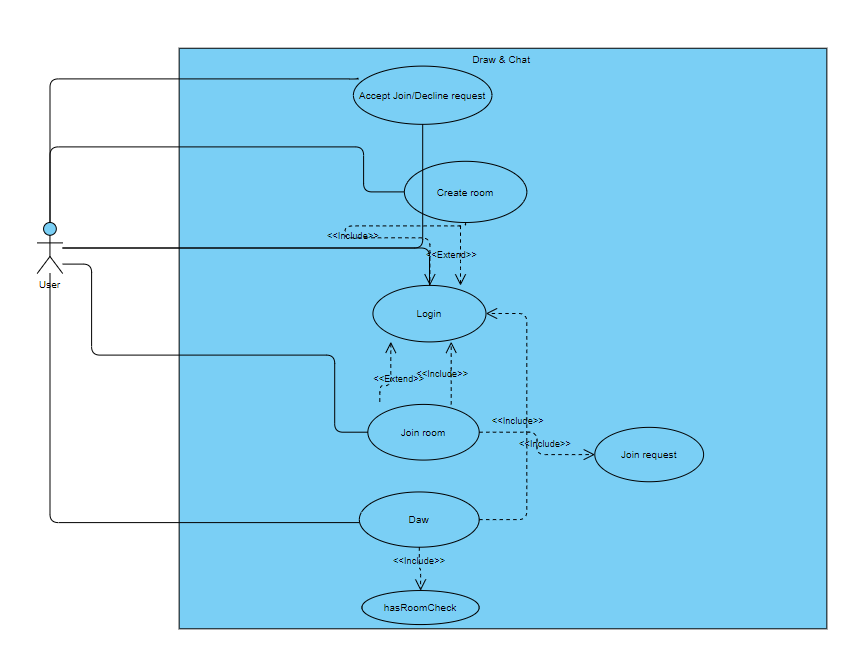


Figure 4. Use Case diagram for draw use case and create room use case

##### Gallery

**Use case 1**: Posting image

Prerequisite: User have created an account on the website

1/Initially, when the user enter page, they will need to provide their username and password to login

2/User is directed is checked whether they have created or joined a room yet, they can optionally answer this and is allowed skip to the room’s list page

3/Once in the room’s list, on the top left corner, user click on a button to change to gallery mode

4/Once in gallery mode, user will already have their own profile created and can start posting.

5/User can start posting by clicking on the “+” symbol on the top right

6/User is directed to the Post submission page.

7/User enter the information for the post

8/User click submit

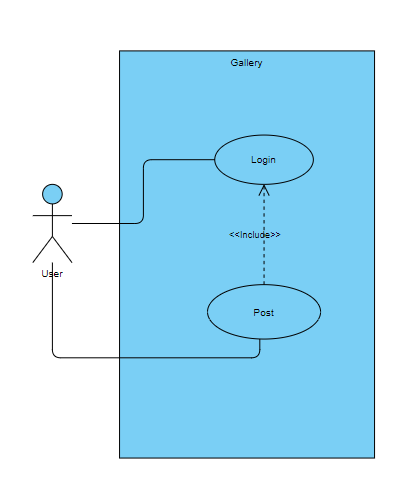


Figure 5. Use case diagram for posting use case

**Use case 2**: Posting canvas from room

Prerequisite: User have created an account on the website and already have joined or created at least 1 room

1/Initially, when the user enter page, they will need to provide their user name and password to login

2/User is directed is checked whether they have created or joined a room yet, they can optionally answer this and is allowed skip to the room’s list page

3/In the room’s list page, user choose a room containing the canvas they want to post

4/User is directed to the room , where user click on the button to post the current state of the canvas to their profile

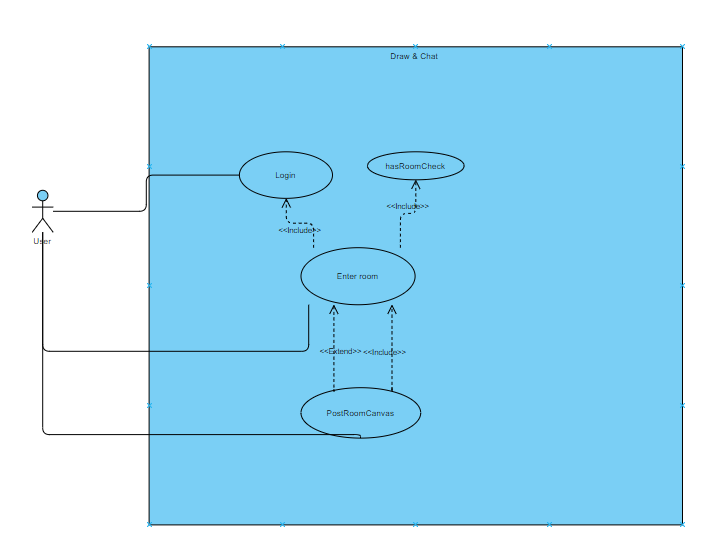


Figure 6. Use case diagram for posting canvases from a room

## 3.4 Summary.

This chapter has identified the functional requirements while mentioning the non-functional requirements which the system needs to satisfy that the application requires to form the its structure. Furthermore, This helps capture the design choices that have been chosen as solution to create a sociable and interactive environment between user and user on the application platform. Lastly, 4 essential use cases of the application has also been discussed, this explores in more details the system’s purpose and give general ideas about what are the critical functionalities of application and how they are used.

# **Chapter 4: Design and Implementation**

In chapter 4, we are going to discuss about the design patterns that form the structure of the application, this give a broad view about the project’s architecture and a general idea about the entities participating in the application, their relationships, what they are for. Furthermore, through the progression of the chapter I will also be narrowing down the topic from general to detail by explaining the implementation of each main components and give clarification to the design choices.

## 4.1 Design

### 4.1.1 Overview

The application is web-based and developed using JavaScript for the front-end and python for the back-end. In addition, for the database, SQLite has been chosen as it is integrated as part of the python framework that will be discussed later. The database provides a REST API which is useful as this allows the development to focus only on the front-end/back-end integration and not data transaction.

### 4.1.2 Design pattern

The design pattern i have chosen to form the architecture of the application is Model View Template(MVT). The framework separates logic, interface and data into sections managed by Django framework, this allows encapsulation across different component ,thus making the testing process more approachable.

* M - Model stores data and relationship between class-based entities, which are then migrated to the built-in SQLite of Django framework. Model in Django has ORM mechanism for interpreting relationship type: 1 - 1, 1 - many, many - many.
* V - View receives request and data from URL dispatcher and client , then performs logic depending on the request and response to the client.
* T - Template can be html files that receives context and is rendered by view. Using context and JavaScript, the template displays to the client the desired information using the passed context with bootstrap or CSS.

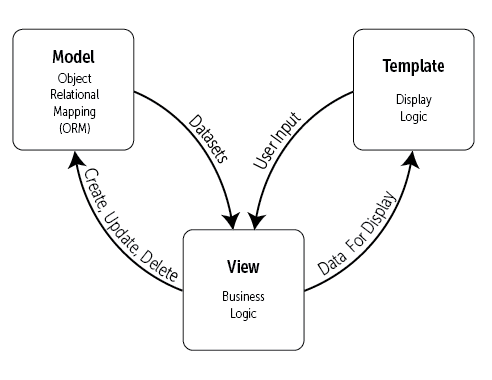


Figure 7. MVT model (Figure 3.1 https://djangobook.com/mdj2-django-structure/)

### 4.1.3 Framework

The project is divided into 3 parts: front-end, back-end and database

* **Front-end**: Instead of using template for displaying information, the design uses Reactjs for a responsive design, this allows modularity as Reactjs website is built using combination of components along with the a wide range of module, thus allows re-usability of code.
* **Back-end**: Implemented using Django framework, this allows for past pace development for quick result, provide ORM which can be modified if needed and minimize the use of SQL query as the framework has API for translating python code to Sql queries and receive as query jQuery.
* **Database**: Django provides built-in database, SQLite. Moreover, Django has REST API for SQLite database, this gives easy access and manipulation to database records.

## 4.2 High level design

### 4.2.1 Model

Model contains classes representing entities and their relationships, which are handled by Django or can be manually defined. In addition, beside class diagram, since SQLite is being used, ERD can be used for designing relational tables and use normalization to minimize redundancy for better performance. The classes are defined as below:

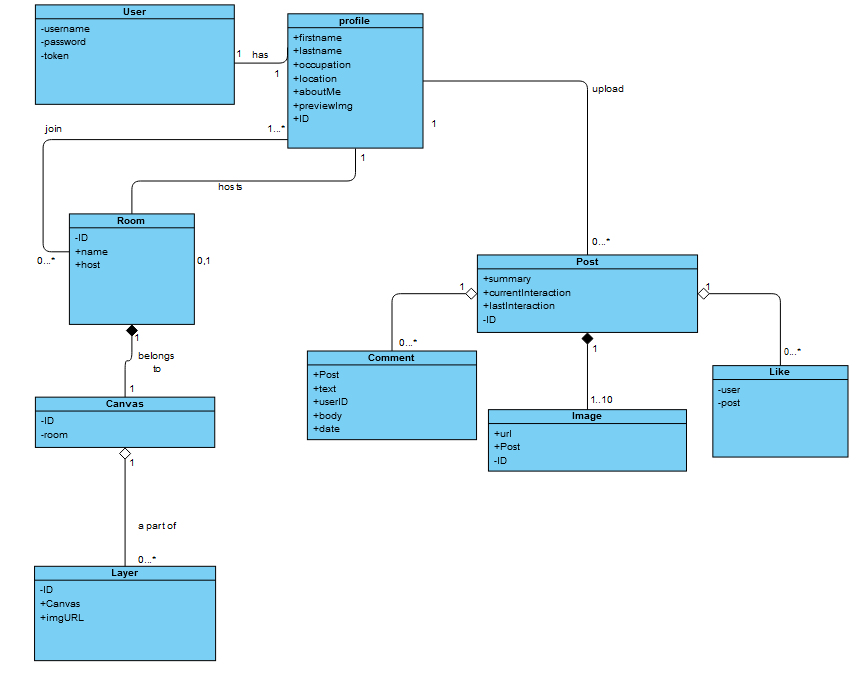


Figure 8.Class diagram describing the entities in the project

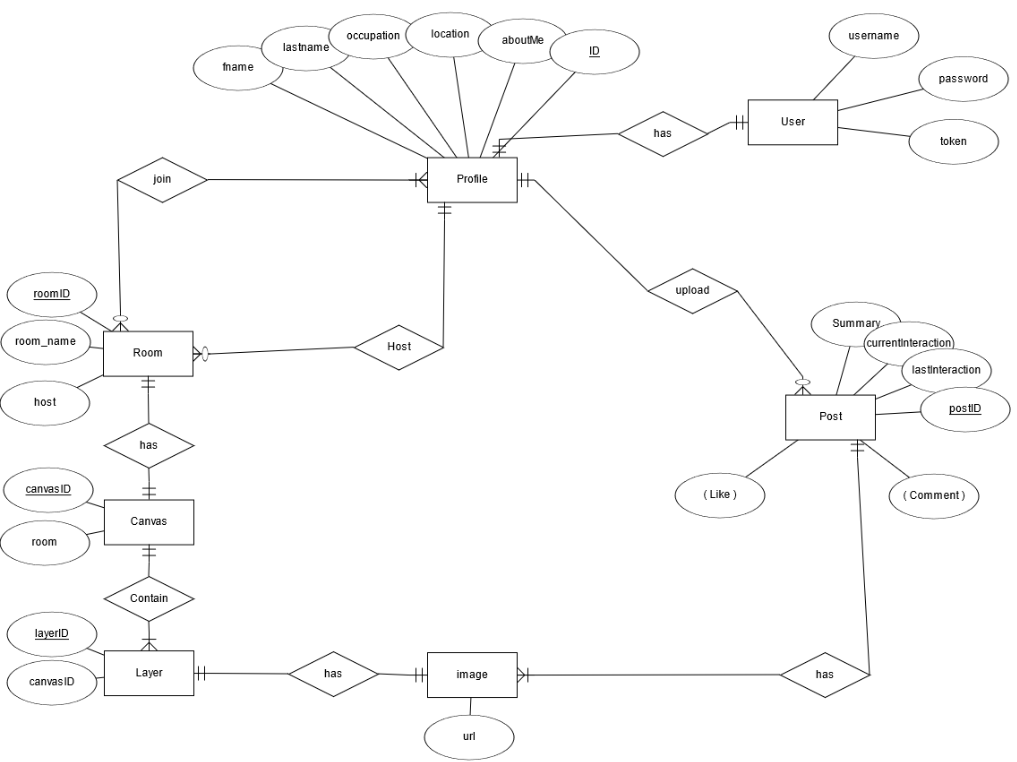


Figure 9.Entity relationship diagram describing the models required

### 4.2.2 View

View contains logic for handling different requests from the client. When the client sends a request to the back-end handled by Django, the request initially arrives at the URL dispatcher, which is then forward to the view corresponding to the URL that the request is sent to in the URL table. Sequentially, the called view function will take as input, the request, which is a dictionary containing the sent data that can be manipulated by the view. Finally, once the data has been processed, the response will be parse into json data and delivered to the client. In conclusion, View acts as a handler for different requests from the client and each of them is called based on the URL corresponding to them in the URL dispatcher. On a high level view, the model contains views for handling:

#### 4.2.2.1 Room lobbies and canvases

* **Room view-set**: Each time a client send a GET/POST/DELETE request for a room, those requests are handled by the Room view-set that will authenticate the user, perform the request and send a feedback about the request to the client

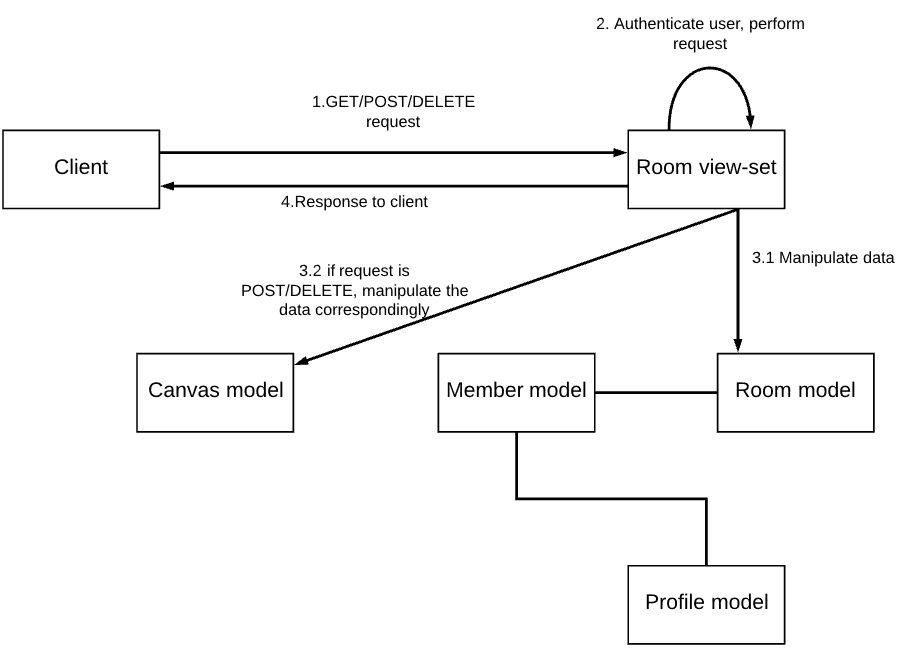


Figure 10. Room viewset high level

* **Member view-set**: is an intermediate view-set between room model and user model. Every time a join request(POST) or a leave request(Delete) is received, the member view-set authenticates the client and add them into the room as a member or delete their entry from the room.

POST request:

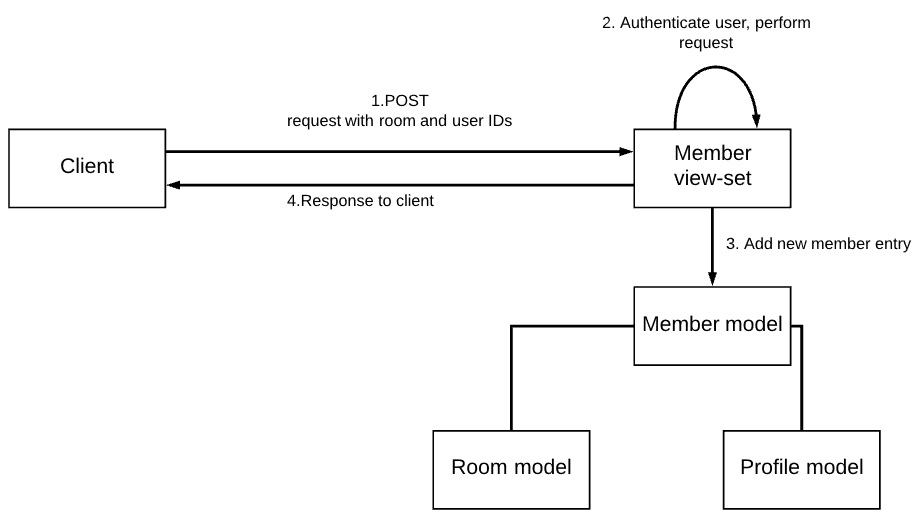


Figure 11. member viewset post high level

DELETE request:

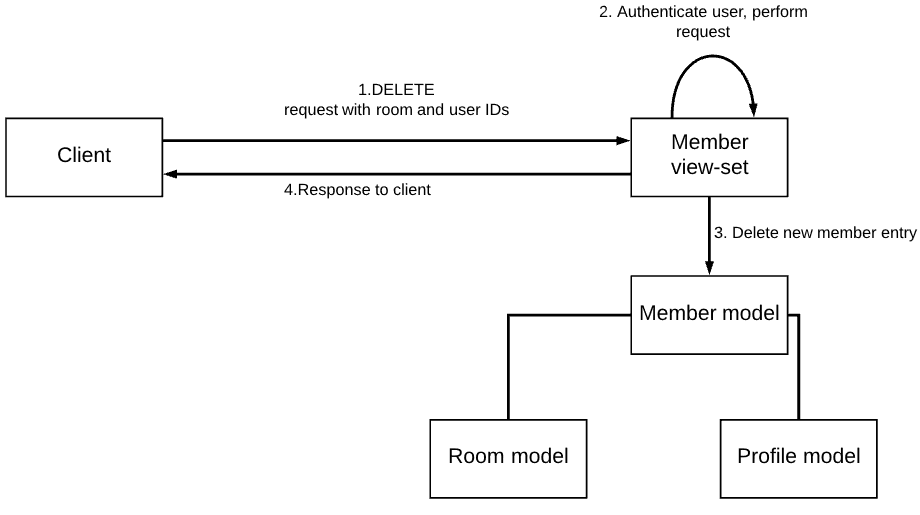


Figure 12. member viewset delete high level

#### 4.2.2.2 User authentication

* **User view-set:** Every time a user register, the view set create a new user entry and authenticate that user using sessions.

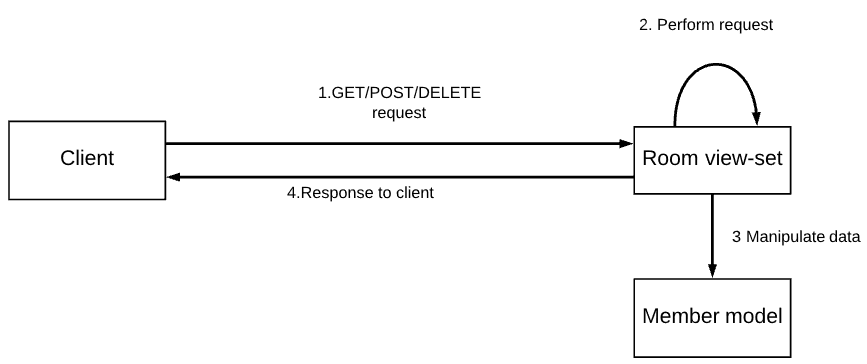


Figure 13. user viewset high level

* **Profile view-set:** Each user is assigned with a unique profile object that stores their information besides username and password. If a user object is deleted, the view-set deletes the corresponding profile object. If a user object is created, the views-set ask the user for the information of the corresponding profile object and assign it to the new user entry

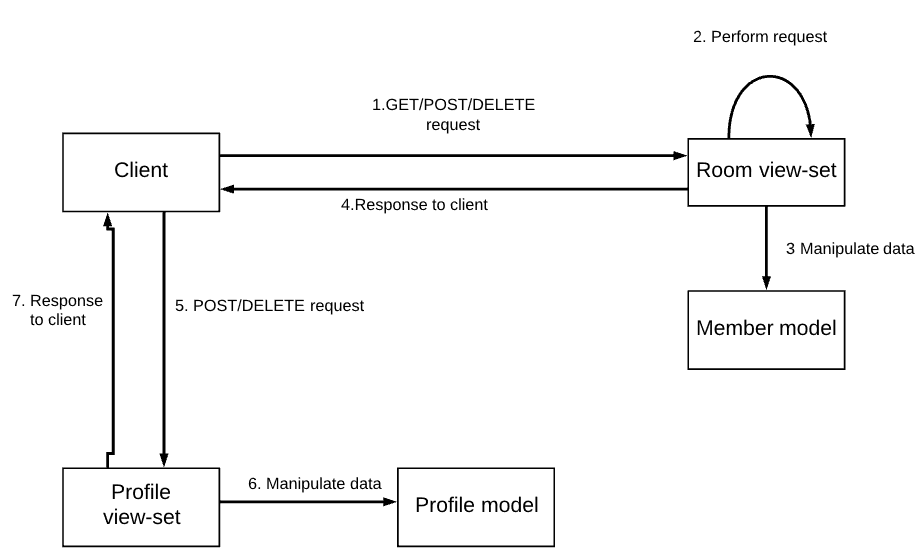


Figure 14. Room view-set post high level

#### 4.2.2.3 Gallery

* **Post view-set**: is used for handling user’s posts. Every time a GET/ POST/ DELETE request is received, the sender is authenticated and the request is performed.

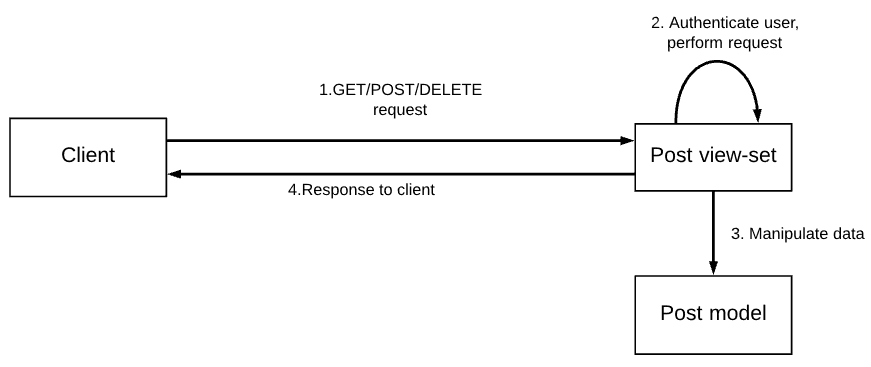


Figure 15. Post view-set

* **Comment view-set:** Each post object has 0 to N comment objects.For each POST/ DELETE comment request received, the view-set must authenticate the sender and assign it to a post object or delete the request entry from the database.

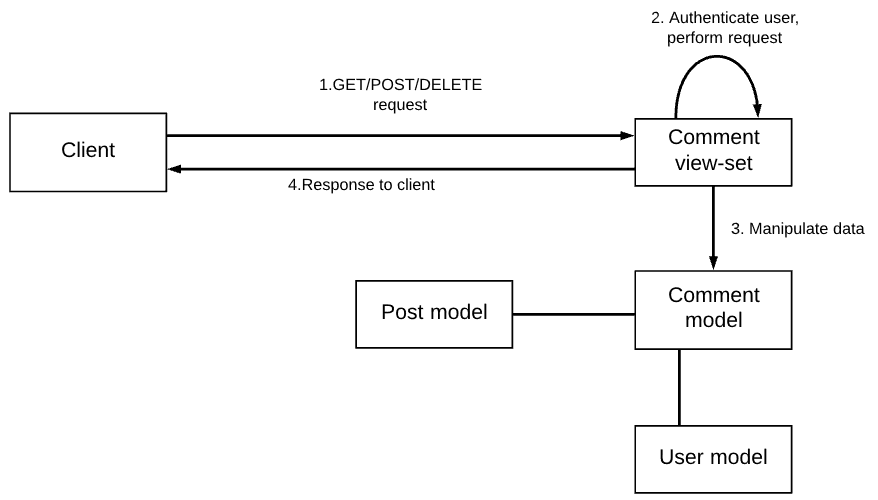


Figure 16. comment view-set high level

* **Like view-set:** Each post object has 0 to N like object. For each POST/ DELETE comment request received, the view-set must authenticate the sender and assign it to a post object or delete the request entry from the database.

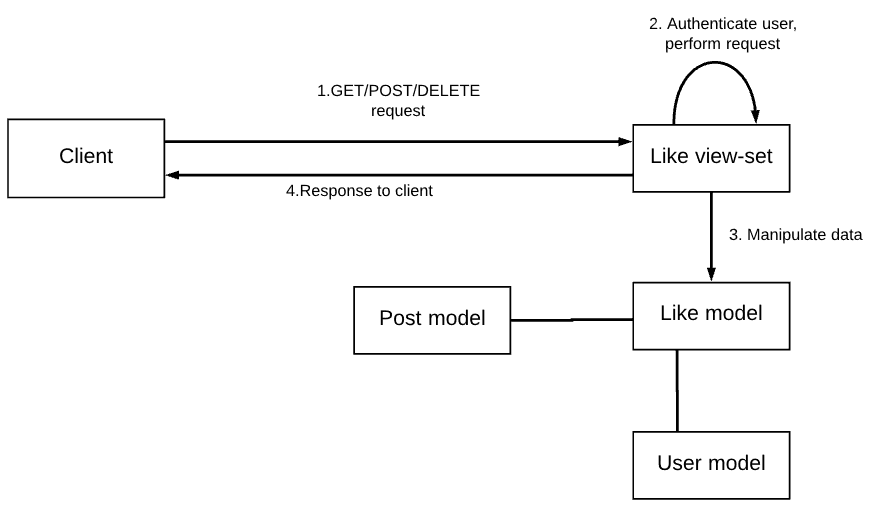


Figure 17. like view-set high level

### 4.2.3 Consumer

Django channel API allows web-socket communication handling using route and consumer, which is similar to the concept of url patterns and views, I will be explaining and discussing about the concept of route and consumer in this section:

**Route**: The route in channel API checks the path of the web-socket request from the client and create a stateful connection between the client and the server.

**Consumer**: The consumer is an interface that allows for establishment of client-server connection and handles message events from clients. Once a client open a web-socket connection using a certain path, the path’ s corresponding consumer and that client will perform a handshake and establish a stateful connection that is kept alive as long as each side is still connected. In addition, in Django Channel API,on connection, the consumer could also create and add the client to a channel group, where any channel in the channel group can send a message to the group, which will then be sent to all individual channels in the channel group.

As discussed above, consumer and route greatly contribute to the real time communication between different users and this is applied to the design of the application, the high level view of the consumer includes: ChatConsumer and Canvas Consumer:

* **ChatConsumer**: the consumer is responsible for handling text messages from users in the same drawing room. It receives messages of type “chat\_message” from 1 channel and pass it to all other channels, who then pass their received message to their client

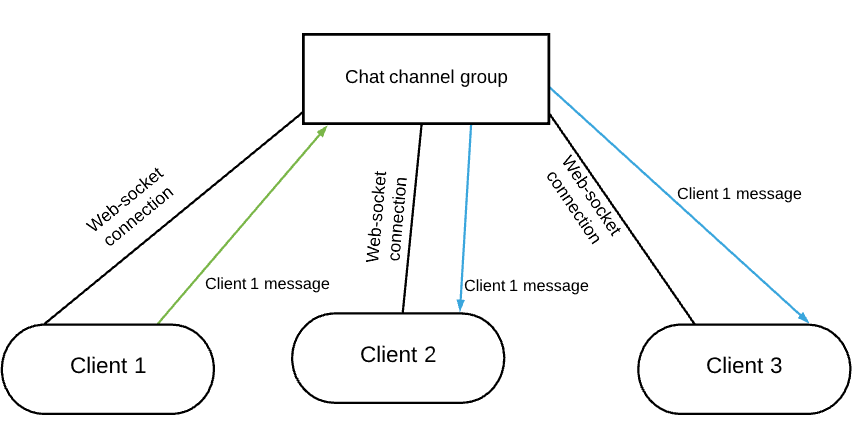


Figure 18. Chat Consumer high level

* **CanvasConsumer:** the consumer is responsible for handling: brush strokes, eraser strokes and canvas layering. Firstly, as mention before, each canvas will have 0 to N number of layers, for each layer, a web-socket connection is established and a channel group is created for that layer. Secondly, for brush stroke message event of type “stroke” or eraser stroke message event of type ”eraser”, the sever receives message event with brush or eraser stroke data and send it to all client except for the sender client for rendering on canvas. Lastly, each layer’ s channel group has many channels, but only 1 channel has permission to send data to canvas layer’ s channel, the reason for this is to protect the privacy of each client during drawing sessions so that ones drawing can not be interfered by another client.

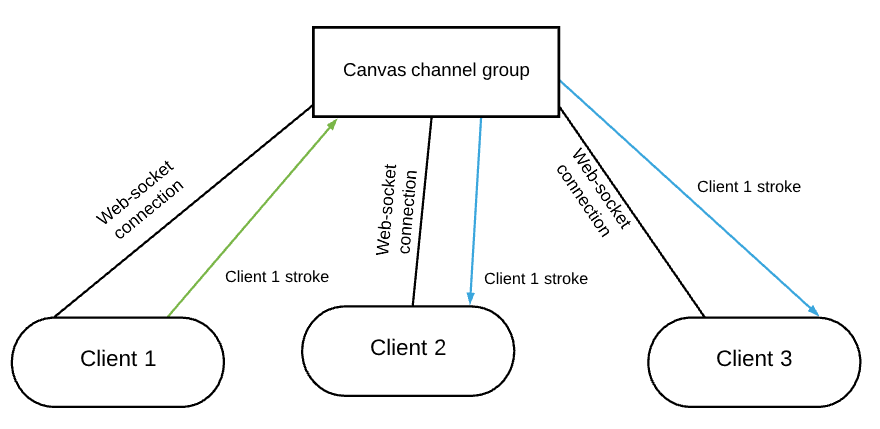


Figure 19. canvas consumer high level

### 4.2.4 Interface

The front end uses tools and modules provided by Reactjs. The UI preserve the concept of modularity as it can be built using combination of components that is specified by the developer. The goal of the front end is to be a interface to take inputs from the client, format it so that it could be interpreted by Django. Moreover, it also open web-socket connections to the server to interact with other users in real time. In a high level point of view, the UI contains the following components:

* **Introduction guide and Registration/Login**: interface for new user to register and login as well as to introduce the application and guide on how to use it.

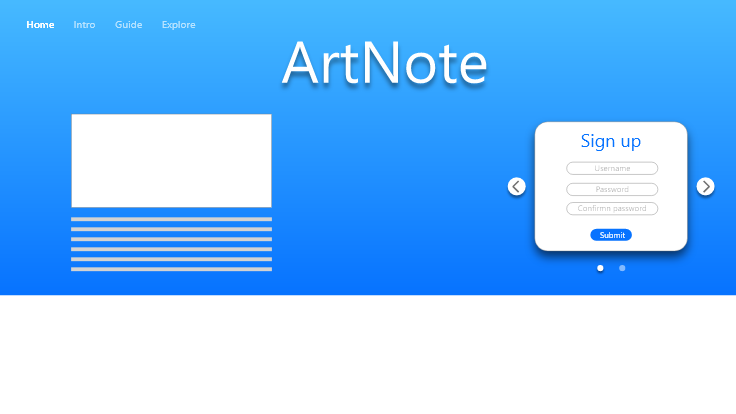


Image 1. Welcome page-login section

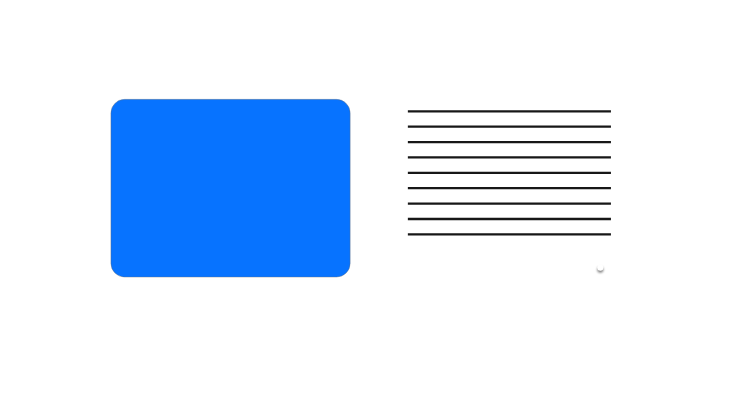


Image 2. Welcome page- greeting section

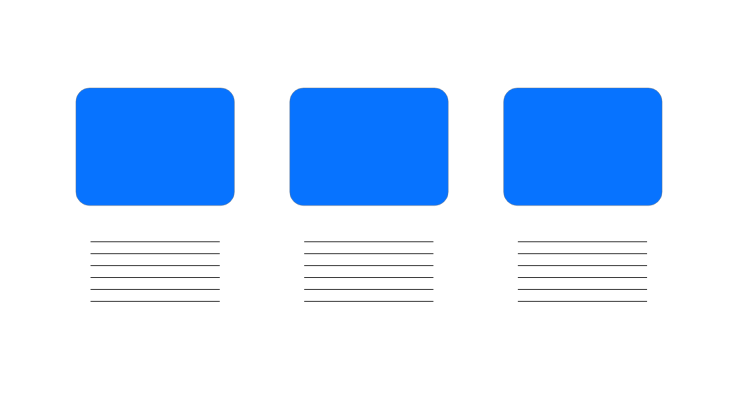


Image 3. Welcome page-feature introduction

* **Lobby of rooms**: interface for creating rooms to join as well as displaying room of type “Joined rooms” or “Popular rooms”

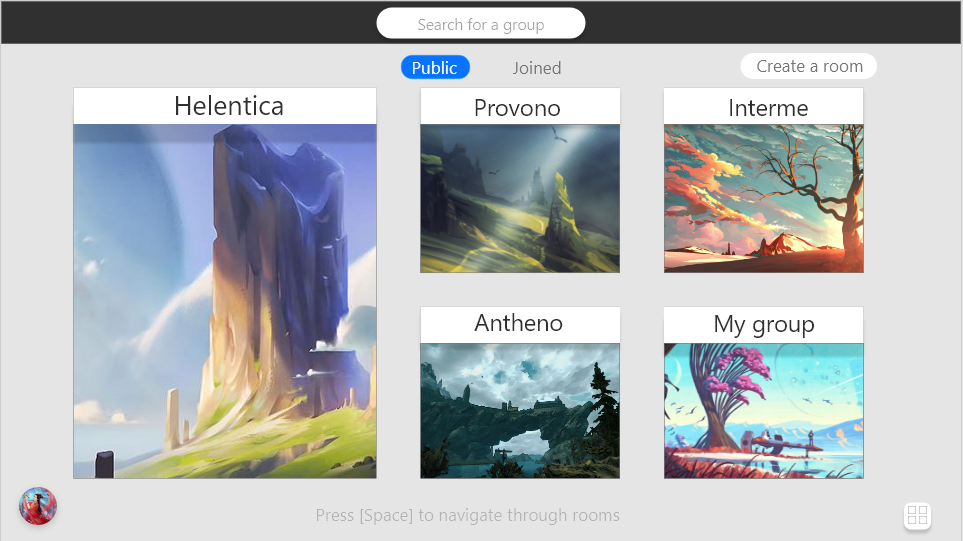


Image 4. room lobby

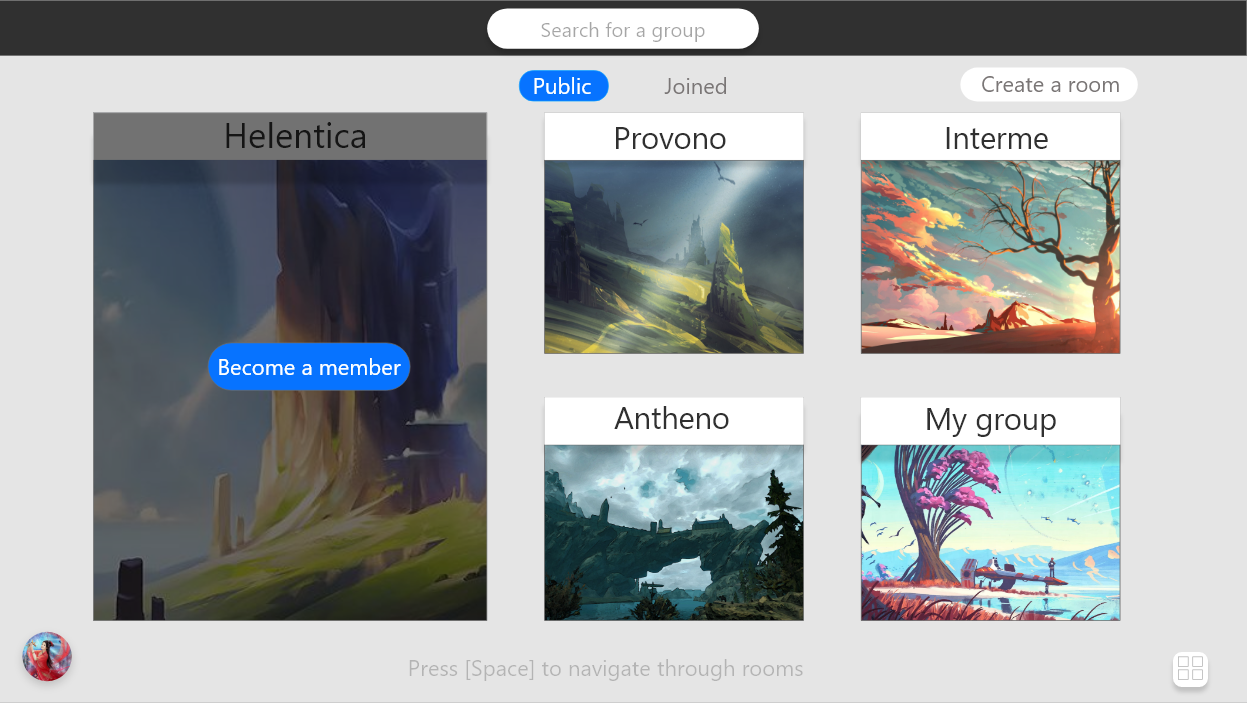


Image 5. room lobby- join room option



Image 6. joined room lobby- options

* **Drawing rooms**: interface for displaying a room canvas and its layers along with tools such as brush, eraser, eye dropper and canvas manipulation options such as, save or publish current canvas
* **Chat-box**: interface for sending/receiving and displaying text messages from different clients in a room.
* **On-line status**: interface for notifying every member in a room who is in the current drawing session

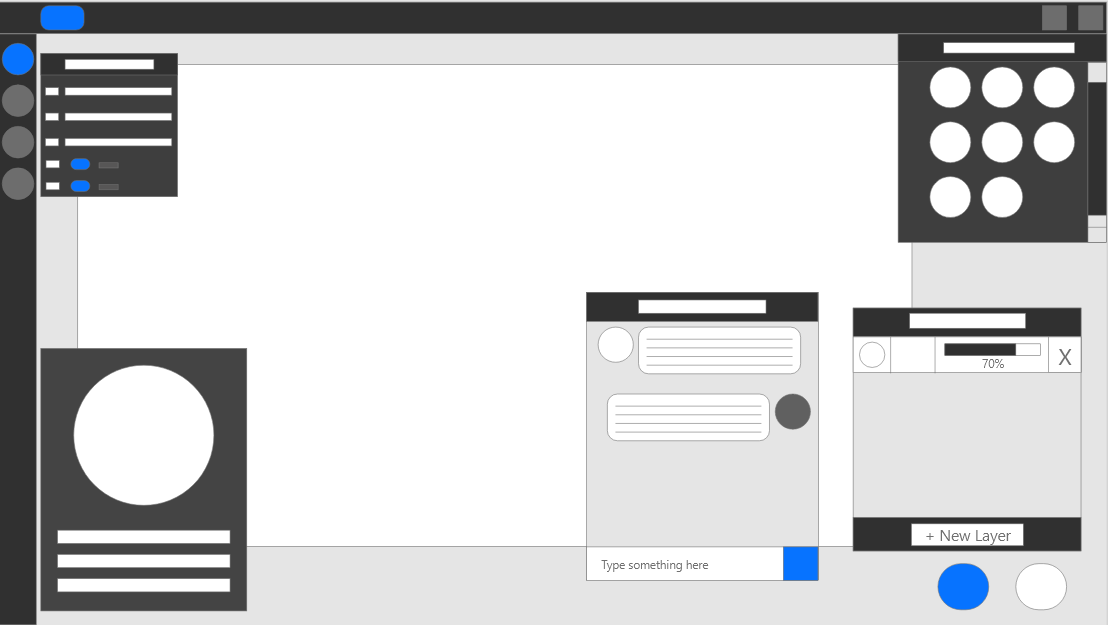


Image 7. Room canvas

* **Gallery**: a customizable gallery page for each user to post their artwork

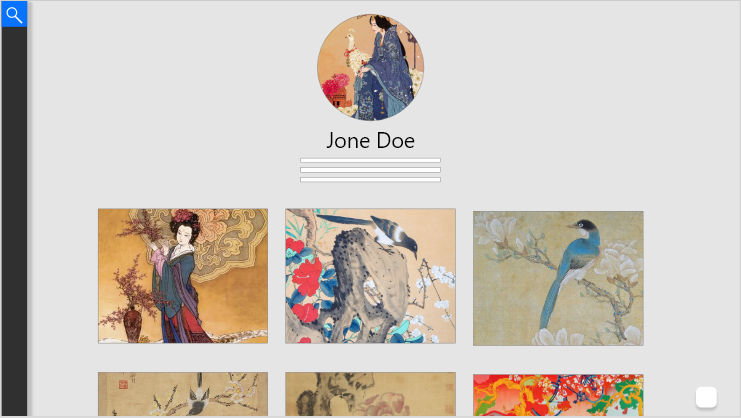


Image 8. user gallery

* **Explore page**: interface for displaying user posts and search for user

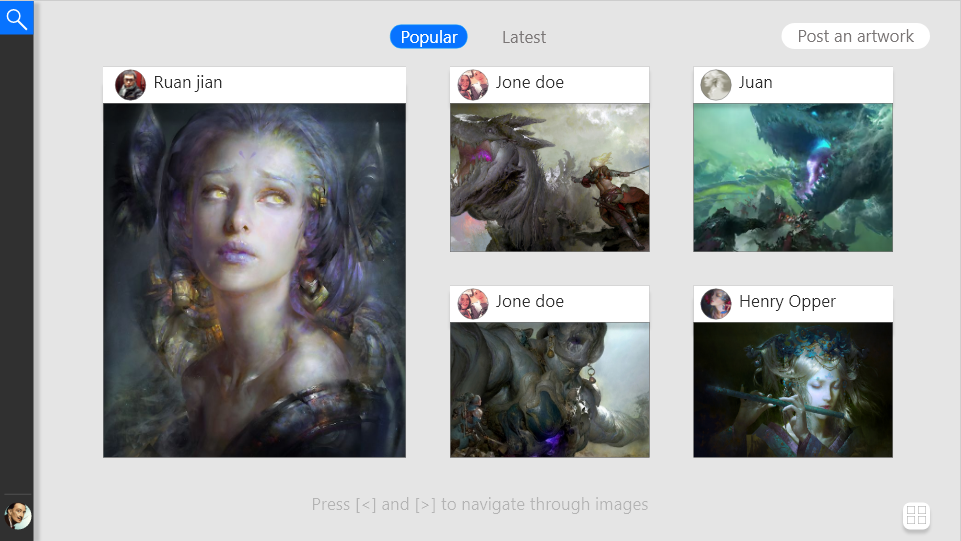


Image 9. Explore page

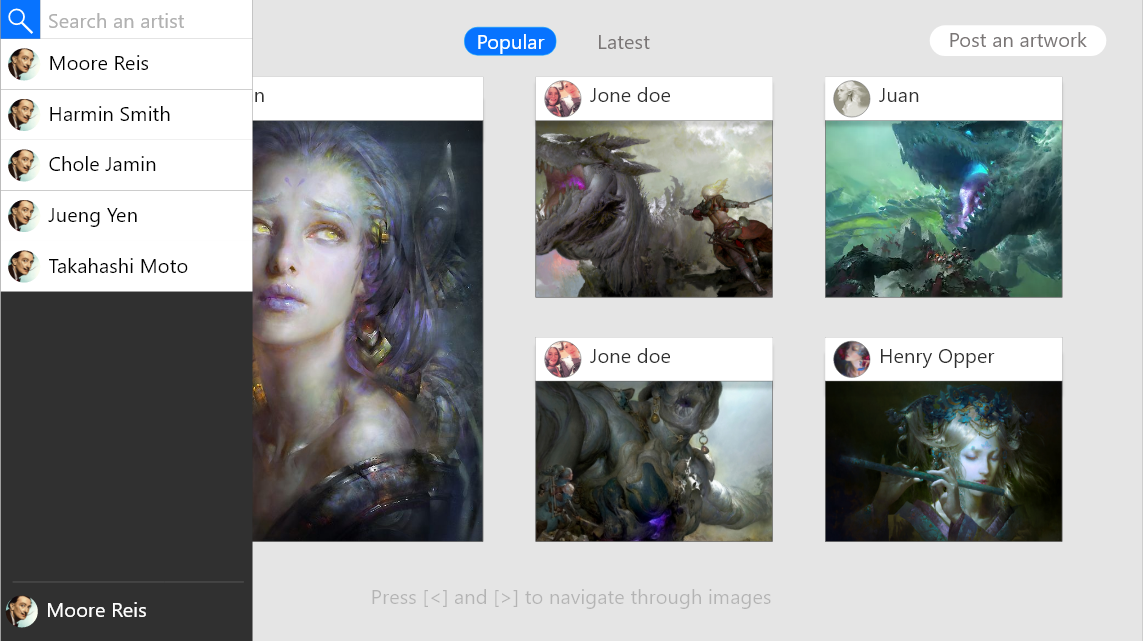


Image 10. explore page- search

### 4.2.4 Database

The project is using Django built in SQLite. The reason is that the framework has an API for the database, this allows for quick data insertion/delete for testing and debugging. Even though, MySQL is more suitable in terms of scalability, security, but since this a small social application, SQLite is much quicker to use as it does not require authentication while still using sql syntax. Moreover, Django framework also provide ORM mechanism and this can prevent many web application attack, such as injection attacks.

## 4.3 Low level design

This section will be discussing the detail implementation of each aspect: Model, View, Consumer, Interface and the reasons why these implementation methods have been chosen.

### 4.3.1 Model

The models are used for storing fields and states of entities participating in the application. In addition, these models represent relationships between these entities, this allows for encapsulation of related data and giving dependency on each entities so access to these entities become more secure:

#### Entities:

* **User**: Django provides a built-in User model that provides its own security functionality. Each user model contain 2 main attributes: username and password. The benefit of this is that Django’s User model passwords are hashed and salted and this make it more difficult to compute the password using different methods such as exhaustive search or dictionary attack. Lastly, Django’s User model allows easy and secure authorization using tokens, this will be explained more consistently in the next section about Views and Serializers. User model’s fields: username, password
* **Profile**: This model can be seen as a custom User model that has its own fields beside username and password. For every user object created, a profile object is also created and assigned to that and only that user object, if the user object is deleted, this profile object is also deleted. Profile model’s fields: user(user object unique to a profile object), profile\_pic(profile picture) ,fname(first name), lname(last name), occupation, location, aboutMe



code listing 1. profile model

* **Room**: This model is used for storing rooms information. Each room is created a by user and that user becomes the host of that. Each room can only have 1 host and if that host(user) object is deleted, the room is deleted as well. Room model’s fields:room name(name of the room), roomBackground(background of the room) and host(user that created the room)

****

code listing 2. room model

* **Layer:** This model is used for storing different layers of a canvas that belong to a room. Each canvas is made of many layers stack on top of each other and each layer is an image that is drawn by a users in the same room. Layer model’s fields: canvas(canvas that a layer belong to), image(the image of that canvas), index(what number is that layer, for example, layer 1, 2, 3,…)

****

code listing 3. layer model

* **Member:** This is a intermediate model between a member(user) and a room. Each time a user joins a room, a new Member record is added for that member and the room they join. Member model’s fields: room(room the member joined) and member(the member of the room)

****code listing 4. member model

* **Post:** This model is used for storing information about posts published by a user. Each post created will essentially be storing an image the user publish and the user who publish that image. Post model’s fields: user (user that own that post), image (post’s image), interaction (number of interaction it has), currentInteraction (number of current interaction it has), lastInteraction (number of last it had before updating), last\_update\_date ( the last time the interaction that post is updated) and published\_date (the date the post is uploaded). Further information about interaction will be discussed later in View and Serializers section.

****

code listing 5. post model

* **Like:** This is an intermediate model for storing the “like” interaction between a user and a post. Each time a user like a post, a new Like record will be created storing the post and the person who like the post. Like model’s fields: post(the that is liked) and user(user who like that post)

****

code listing 6. like model

* **Comment:** This is an intermediate model for storing the “comment” interaction between a user and a post. Each time a user comment a post, a new comment record will be created storing the post, the person who comment on the post and the comment text. Comment model’s fields: post(the that is commented on), user(user who comment on that post) and comment(the content of the comment)

****code listing 7. comment model

#### Relationships:

**One-to-One:**

* User-Profile: Each user model created must have 1 and only 1 profile object assigned to it
* User-Token: Each user, each time login or register will have a unique token assign to them for authorization.
* Room-Canvas: Each room object created must have 1 and only 1 canvas object assigned to it

**One-to-Many:**

* Canvas-Layer: 1 canvas object can have 0 to many layer objects
* Post-User: 1 user object can have many post objects

**Many-to-Many:**

* User-Like-Post: every time a user like a post a Like record is created to link that user and the post they like
* Comment:every time a user comment on a post a Comment record is created to link that user and the post they commented on.

### 4.3.2 URL patterns and views

#### URL pattern

There are 2 main URL dispatchers:

* First one stores all URL patterns end points for authorization and handling **general** API requests.

code listing 7. root url patterns

* Second one is a collection of router and URL patterns for handling **specific** API requests.

code listing 8. api url patterns

In this section, only the main URL patterns will be discussed as they have important roles in the application. The patterns are the following:

**GET request:**

* “**api/user/**”: this pattern is for creating and authorizing users who want to register or login.
* “**api/profile\_get/<int:user\_id>/**”: this pattern is for retrieving a user’s profile object using “user\_id” query.
* “**api/room/<int:room\_id>/**”: this pattern is for retrieving a room object using “room\_id” query
* “**api/post\_get/<int:post\_id>/**”: this pattern is for retrieving a post object using “post\_id” query
* “**api/post\_popular\_get/<int:index>/**”: this pattern is for retrieving 5 popular posts, starting with 5 most popular posts and after each fetch, the popularity of 5 posts are decreased.
* “**api/post\_latest\_get/<int:index>/**”: this pattern is for retrieving 5 most recent posts,

Starting with 5 most latest posts and after each fetch, the published dates for each 5 posts is older for every fetch.

**POST request:**

* “**api/user/**”: this pattern is for creating a record for a user registering.
* “**auth/**”: this pattern is for authorizing a token that is sent a by user login in to create a session between a client and the server.
* “**api/profile/**”: this pattern is for creating a profile and assign it to a user.
* “**api/room/**”: this pattern is for creating a room.
* “**api/post\_click\_interaction/**” and ”**api/post\_like\_comment\_update\_interaction/**”: these patterns are for updating the popularity of a post based on the type of interaction it receives. Further detail about how this is implemented is discussed in the View section.

#### Views

This section will be discussing about the main Views that associated with the URL pattern above:

**GET request:**

****code listing 9. user viewset

* “api/user/”- **UserViewSet**: this view is a inheritance of Django’s ModelViewSet that is able to handle **GET, POST, PUT, DELETE** requests. Since the ViewSet’s aim is to register users, no authorization is needed to access this view.

code listing 10. profile get view

* “api/profile\_get/<int:user\_id>/”-**profile\_get** : this view is used for retrieving a specific profile of a user along with all post objects and user object associated with that profile using a that user id. Another way of retrieving a profile is using that profile is by using a ProfileViewSet, however, this requires the id of that profile to be available but only the id of user who own the profile is available. Hence, a custom view is needed to retrieve a profile using its user id. Since this view have access to user’s critical information, it needs to authorize a user before allowing for access using a custom decorator function **Custom\_ token\_ authorization** function.

Before moving on, the Custom \_ token \_ authorization will be discussed more thoroughly as it play an important roles in most Views.

The Custom\_token\_authorization function is a decorator that take a function as input and return a new function. Moreover, this new function extends the functionality of the function it take as input without internal modification of that function. In this case, this decorator, will first authorize the user who make a request to a view, if the user authorizes successfully, they have access to the view, otherwise, they are denied from accessing the view.

 code listing 10. profile get view

* “api/room/<int:room\_id>”-**RoomViewSet**: this view is a inheritance of Django’s ModelViewSet that is able to handle **GET, POST, PUT, DELETE** requests. This view specifically is used for retrieving a certain room object using that room’s id. The view also requires authorization of the user before they are allowed to retrieve a room.

code listing 11. post get view

* “api/post\_get/<int:post\_id>” -post\_get : this view is used for retrieving a specific post using that post id. There are other ways to achieve this such as using PostViewSet, however, this only give the fields of that post without the information about the person who create the post. Hence a custom view need to be implemented. Since this view have access to user and post critical information, if an attacker, provided that they don’t have access to any authorization token, some how able to retrieve the id of a post then they can easily send, for example, a DELETE request to the PostViewSet and remove the post without permission from the owner. Due to this, this view is wrapped by the Custom \_ token \_ authorization to allow only authorised user to send request.

Before going over the popular and latest post views, the protocol for updating the popularity of a post will be explained, from this point on, this protocol will be refereed as **popularity protocol.**

##### Popularity protocol

Each post object will have 4 critical fields:

* Current interaction: This is a number denoting the number of interaction a post is receiving, which is updated each time a user like, comment or click to view a post. This number will be reset to 0 if the time difference between the current date time and the last reset date time is larger or equal to 15 minute. Once reset, the current interaction number = 0
* Last interaction: This is a number denoting the number of interaction( current interaction number above)a post had before reset. Whenever, the current interaction number is reset, its value will be assigned to the last interaction field, this represent the number of interaction a post had before it is reset.
* Last updated date: Every time the current interaction field is reset, the last updated date is updated to the date and time that reset happen, this is to keep track when the interaction of the post need to reset.
* Interaction: This is a number denoting the number of interaction a post is currently having, it is used to identify how popular a post is. Every time the reset time is reached, the following cycle is carried out:

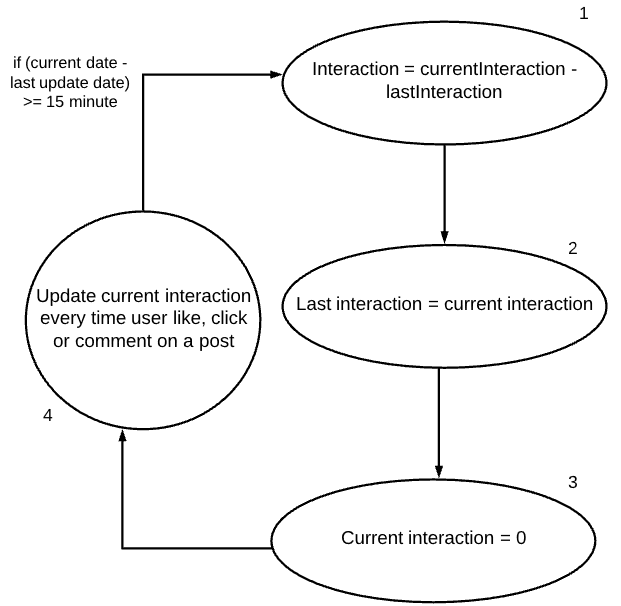


Figure 20. popular protocol

This let us identify how popular a post is dynamically, as the number of interaction a post has every 15 minute is different> The idea behind this protocol is that, If in the first 15 minutes, post A has 10 interactions and post B has 5 interactions then post A is more popular than post B. However,if before the next reset time occurs, post B receives 15 more interaction and post A receive only 2 interactions, once the reset time occurs and the reset happens, post B have = 15-5 = 10 interactions and post A will have 2-10 = -8 interaction, this make post B more popular than post A for the second 15 minute as post B receive more interaction in 15 minute than post A.

* “api/post\_popular\_get/<int:index>/”-**post\_popular-batch\_get**: this view is used for retrieving post based on the popularity. Before each fetch, all data will be checked if they need to be reset to update the interaction number.
* “api/post\_latest\_get/<int:index>/”-**post\_latest\_batch\_get**: this view is used for retrieving post based on the published date.Before each fetch, all data will be checked if they need to be reset to update the interaction number.

**POST request:**

* “api/user/”-**UserViewSet**: this view is a inheritance of Django’s ModelViewSet that is able to handle **GET, POST, PUT, DELETE** requests. The view is used for registering a user and create and a user object corresponding to that user.
* “auth/”-**CustomObtainAuthToken**: This view takes a user’s username and password to obtain a session token unique to that user object, this is used to authorize and restrict that user’s action in the back-end and front-end.
* “api/profile/”-**ProfileViewSet**: this view is a inheritance of Django’s ModelViewSet that is able to handle **GET, POST, PUT, DELETE** requests. This view can be used to create a profile project with POST request.
* “api/room/”-**RoomViewSet**:this view is a inheritance of Django’s ModelViewSet that is able to handle **GET, POST, PUT, DELETE** requests. This view can be used to create a room object with POST request.
* “api/post \_ click \_ interaction /” - **click\_update\_Interaction** and ”api/post \_ like \_ comment \_ update \_ interaction/” - **like\_comment\_update\_interaction**: these views are used for updating the current interaction of a post.

### 4.3.3 Consumer and Route:

This section will be discussing about the URL routes and consumers for web-socket connections. However, before this, I will be discussing about the canvas and layering system of a drawing room.

##### **Canvas**

As mentioned before, each canvas will have 0 to many layers, each layer of a client will open a web-socket connection to the server, this mean that canvas layer 1 of client 1 will be linked to canvas layer 1 of client 2 and canvas layer 1 of client 3, hence client 1 and 2, 3 can both send and receive messages from other. This allows for the sending of data of a stroke from from 1 canvas layer of a client to all other client’s canvas layer. When a client leave the room, all of their layers web-socket connections are closed and will receive any more messages.

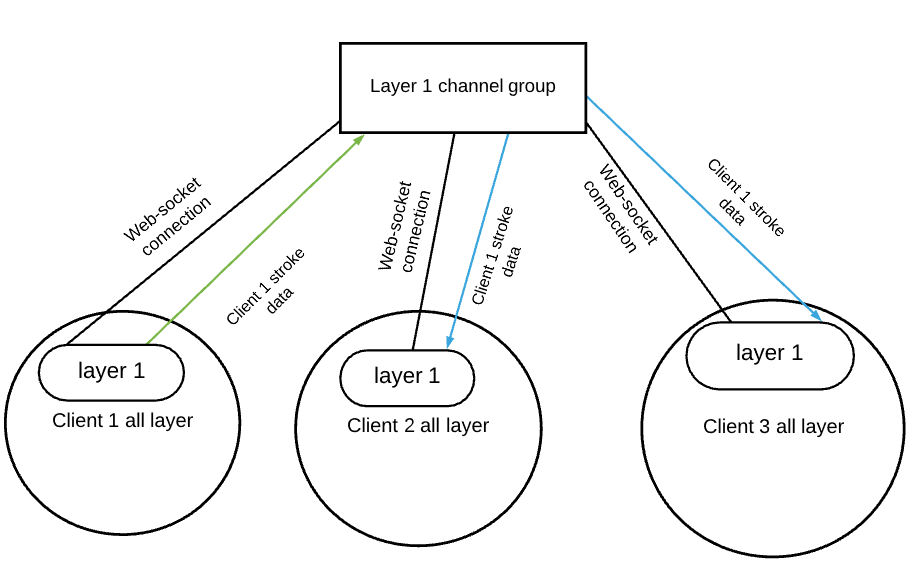


Figure 21. stroke message

##### **Layer**

A canvas layer is a just an html canvas and stacking many layers on top of each other create a canvas for a room, in other words, a canvas of room is just just a combination of html canvases stack on top of each other. Each user must choose which layer they want to draw on before they can draw on a canvas and initially when a user first enter a room, the default layer they are assigned to is layer 1 of the room canvas. In order to manage the choosing of layers, a layering system has been implemented and this allows user to choose which layer they want to draw on.

**Problem:**

However, there are some problems that need to be discussed. Firstly, since everyone is drawing on the same canvas, one user can interfere with another user drawing by choosing the same canvas as them, we don’t want this and also want to allow users to able to interact with each other. Secondly, each html canvas has a 2Dcontext object that user can use to render strokes, however, if many users are drawing on the same html canvas, which mean they are using the same 2Dcontext, one user’s stroke may be interfere by another user stroke, due to various reasons, for example, slow internet connection, packet loss and retransmission of packets. In order to fix this, each layer of canvas will give permission to draw to only 1 client. Moreover, if there are not enough canvases to draw on, a user can create a new layer to draw on and if there a client want to get permission to draw on already taken layer, they can click on that layer in the layer management system and ask for permission and the user who has permission can reply with “Yes” or “No”.

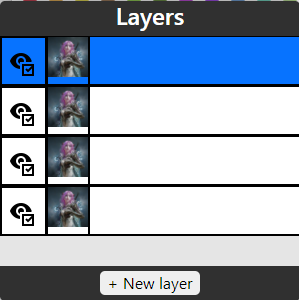


Figure 22.Layering system

Each rectangle box is 1 layer and the highlighted rectangle is the layer the current client is selecting(does not mean he has permission to draw on that layer).

##### **Route**

Each route is to direct a connection request to the right consumer that handle requests from many clients:

* “**Room/?P<room\_name>/**”: this route is for directing connection request to a room chat channel.
* “**Canvas/?P<canvas\_id>/**”: this route is for directing connection request to a room’s canvas layer channel.
* “**DrawingRoom/?P<room\_name>/**”: this route is for directing connection request to a room layer management system channel.

##### **Consumer**

Each route mentioned above has a consumer assigned to it:

* “Room/?P<room\_name>/” - **ChatConsumer**
* “Canvas/?P<canvas\_id>/”-**CanvasConsumer**
* “DrawingRoom/?P<room\_name>/”-**LayerConsumer**

For each consumer, a channel group is created, a channel group can be used to store many channels, channels in the same channel group can send and receive message from each other. Furthermore, there many type of messages that can be defined, using this property, different protocols can be established, such as the layering permission protocol mentioned above. From now on, I will be discussing about each Consumer and the message types defined to establish the mentioned protocols.

**Chat consumer**

Message type:

* chat\_message: this type will deliver a message in the chat from one client to all other client.

The way this message type works is that: a client will send a message to the server, the sever receive the message as an event and deliver the message to all other client in the same channel group chat. After receiving message delivered by the server, each client will the message on the front end.

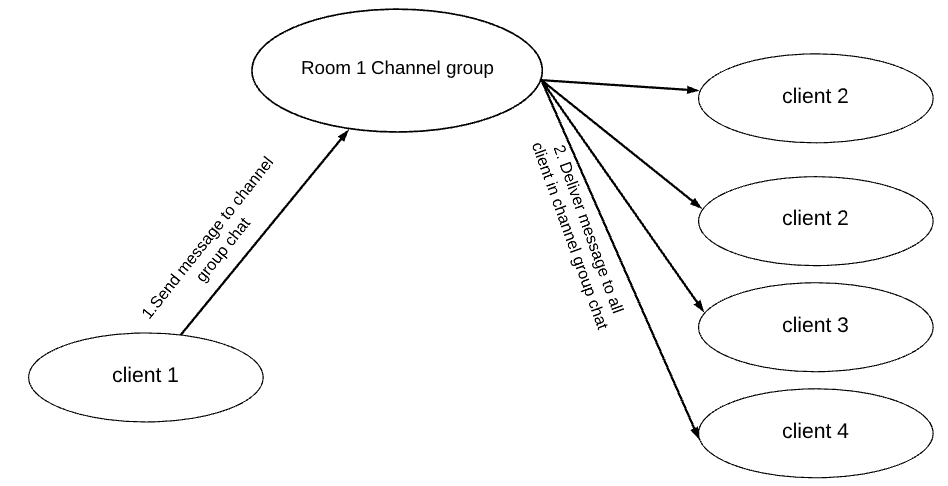


Figure 23. Chat consumer

**Canvas Consumer**

Message type:

* Stroke: this type deliver a stroke data message from one client to all client except the sender
* Eraser: this type deliver an eraser stroke data message from one client to all client except the sender

Before going through the next 4 types, I will be explaining the permission protocol to handle layering of a canvas.

Initially, when a client enter a room, if they are the first client to join the room channel group, they will have permission to draw on all layer. After that, any user who enter a room must initially ask for permission for layer 1 as it is the default layer a client will choose to draw on when they first enter a room. Sequentially, if the client who have permission of layer 1 reply with “Yes” to hand over permission, the client asking for permission will have permission to draw on layer 1. In conclusion, if a user wish to draw a canvas that they can not draw on because they don’t have permission, they need to send an ask permission message to all channel that have permission on that layer and handle based their reply. Below describe this process:

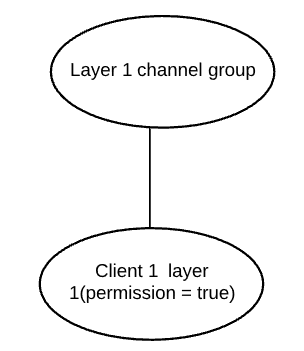


Figure 24. Only client in a room

1. Initially, the room only have client 1, who enter the room first of all client and permission to draw on all layer(for now, we are just focusing on layer 1).

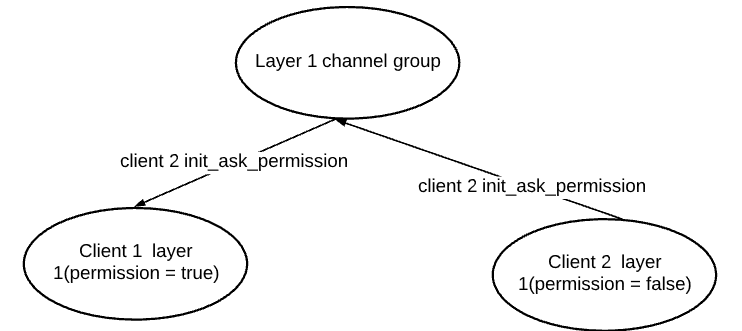


Figure 25. Client 2 join the room

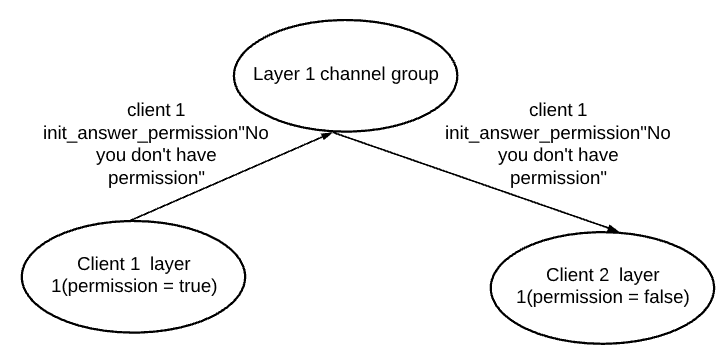


Figure 26. client 2 handles permission management based on client 1 reply

1. Then client 2 enter the room and must send a init\_ask permission message to all clients in Layer 1 channel group to see if anyone has already have permission to draw on layer 1 and client 1, being the only person who has permission, will reply with “No, you can not have permission over this layer 1 because I already have permission on layer 1”, denying client 2 from drawing on layer 1.

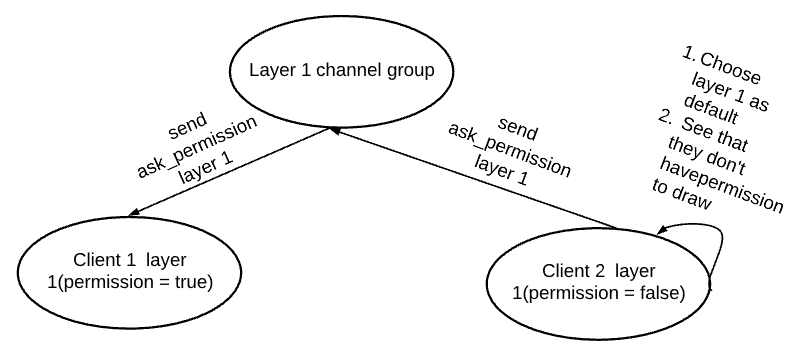


Figure 27. Send ask permission message to client 1

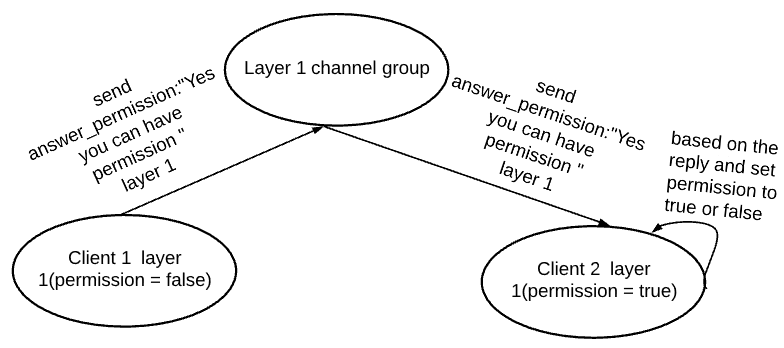


Figure 28. Client agrees to give permission

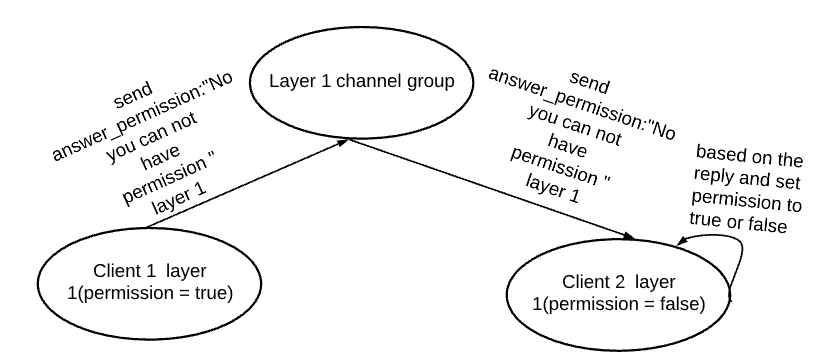


Figure 29. Client 2 does not agree to give permission

1. However, since a user must choose a layer to draw on, client 2 will send an ask\_permission message to any people who has permission to draw on layer 1 and since only client 1 has permission, they will have to answer whether to give permission or not. Client 1 reply will be delivered to client 2 and based on the reply, client 2 will either choose has permission or not. Later on, if client 2 wish to have permission on any other layer they can’t draw on because they don’t have permission, they to click on that layer to send an ask\_permission request to people with permission and wait for reply.

* init\_ask\_permission: This type of message is delivered when a client enter a room to ask if anyone has permission over a certain layer
* init\_answer\_permission: This type of message is delivered as reply to init\_ask to notify the person who is asking for permission that there is people having permission over the layer they are asking for and they don’t have permission on that layer
* ask\_permission:This type of message is delivered when a client choose a layer they want to have permission on and ask if anyone is willing to give that permission
* answer\_permission:This type of message is delivered as reply to ask\_permission to notify the person who is asking for permission that there is people having permission over the layer they are asking for and will give them a reply if they are allowed to have permission or not.

**Drawing room consumer**

Message type:

* add\_canvas\_layer: this type of message is deliver to all channel in a channel to notify that they should create a new layer on their client side as some client has created a new layer.

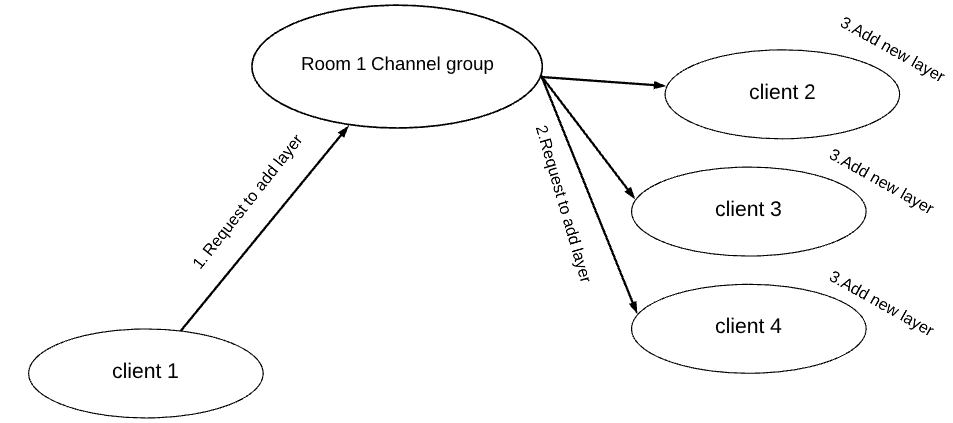


Figure 30. add\_canvas\_layer message

Furthermore, another feature that is important is to allow each client to keep track of who is currently in the current drawing session and we do that by providing each room with a list of people who is on-line and is in the same drawing session. The following section will be discussing how to decide who and how many people are on-line.

This task requires 3 message types:

* “is\_online”:each client when they first enter a room, they will have to send this message to the server, which will be delivered to all other clients. Sequentially, every client who receives the message will then add the client who send the message to their online list and the client who send the message will add themselves to their own on-line list when they first join the room.

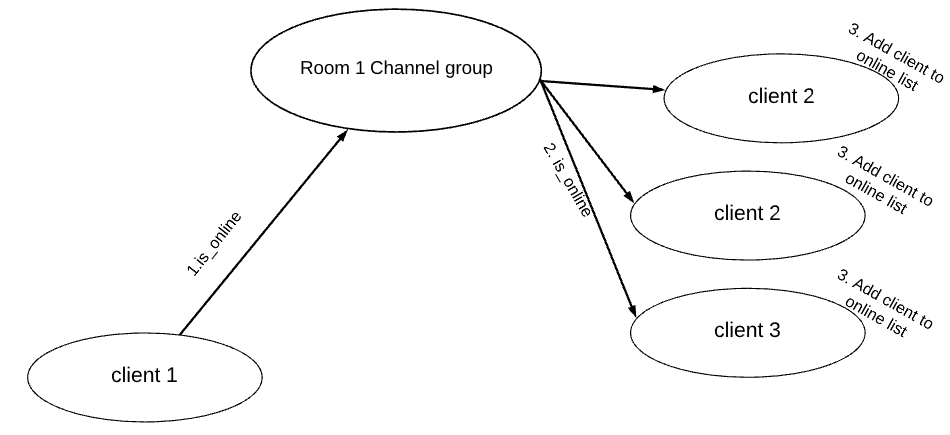


Figure 31. is\_online message

* “not\_online”: each client leaving the room must send this message to the server, which will then be delivered to all other clients. Once received the message, all clients must remove the client who sent the message from their on-line list.

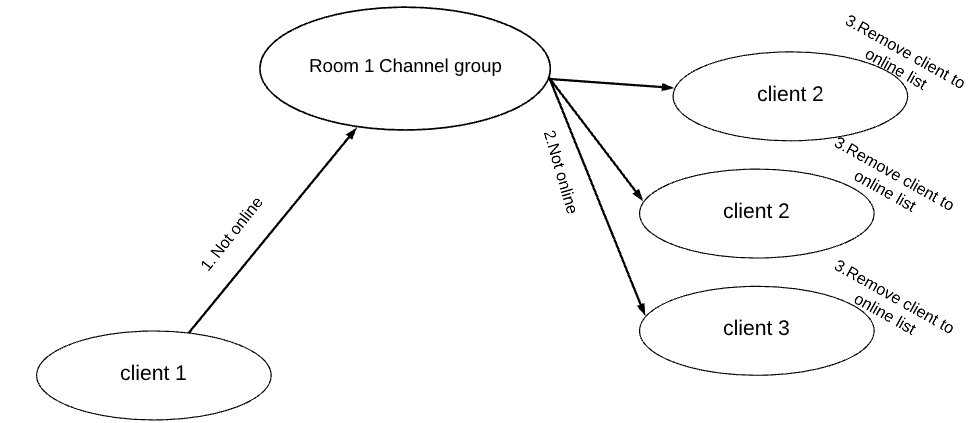


Figure 32.not\_online message

* “who\_is\_online”: each client entering a room, after sending “is\_online” message, must send a ”who\_is\_online” message to all other clients except themselves. Once received the message, each client must reply back to the client who sent the message with a “is\_online” message, for each “is\_online” received, the asking client must add each client, who replied, to their on-line list.

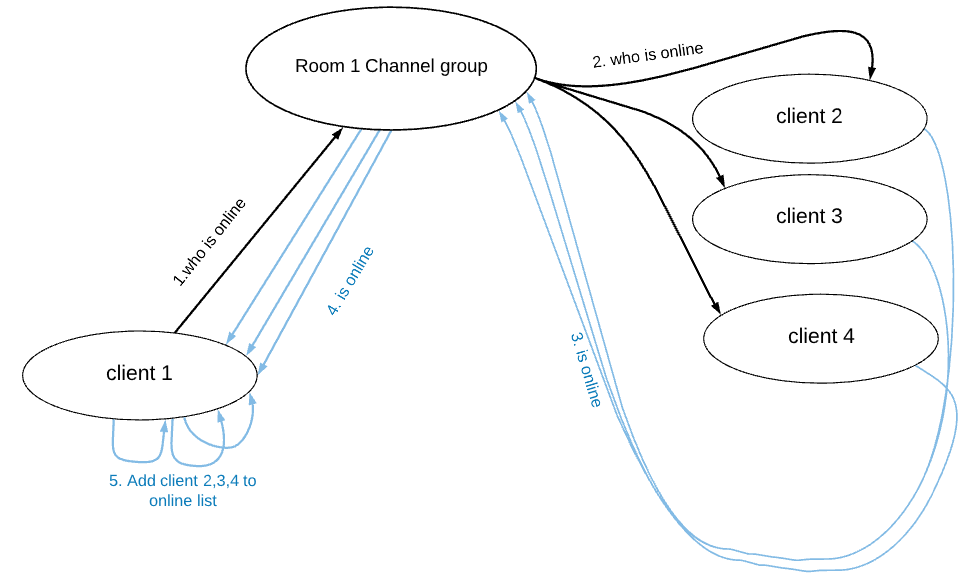


Figure 33.who\_is\_online message

### 4.3.4 Interface

In this section, I will be discussing about the main interfaces that clients will interact with and will be explaining their functionalities, how they work and implemented. Furthermore, these interfaces are responsible for sending http requests and handling web-socket messages from server, thus some details about consumers and algorithm will be mentioned as well. However, further information about algorithm used will be more detailed in the algorithm chapter.

This section will be separated into the following parts:

* Login/Register/authentication
* Explore page
* Room lobby
* Room canvas

#### LOGIN/REGISTER/AUTHENTICATION

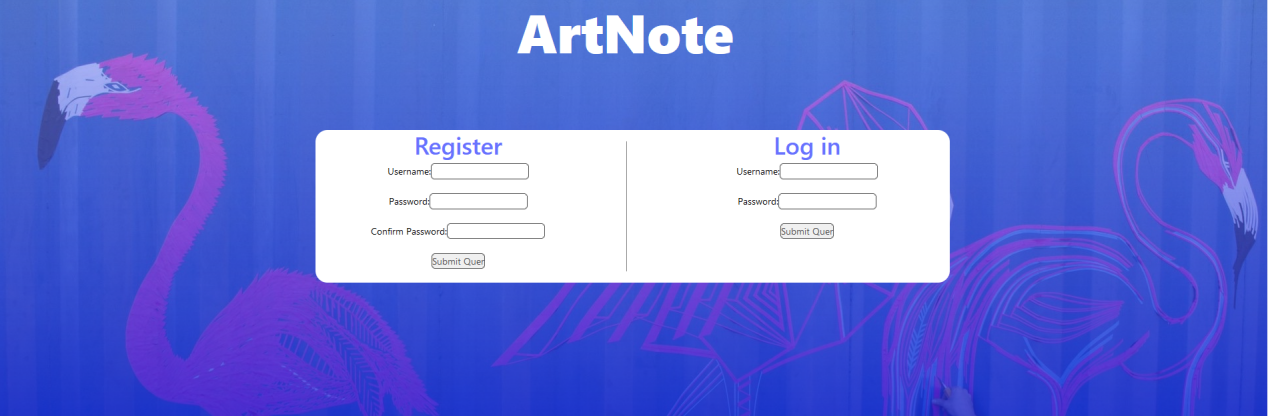


Figure 34. login area

Each user will have to create an account or login when they first enter the page, this is to check whether a user is valid to perform some certain request and if they are a member of the application as only member of the application is allowed to interact with the page and other members. If a user, who have not logged in yet, tries to access a page that requires pre-authentication, that user will be sent back to the log in page.

Furthermore, for security purposes, each user logging in by user-name and password alone is not enough to authenticate a user as any other clients, assuming that they know the end points of a certain request, they can send that request to the server to carry out an operation, for example, POST or DELETE, which are dangerous. Due to this, each user, once logged in, they will create a session with the server using authentication token. This allows for a more secure way of authenticating users as now, every time a client send a request to the server, that client must attach to their request’s header a token that only a logged in member of application has, so no other outsider can perform the request except for logged in users.

In summary, the purpose of the log in page is to authenticate a user before they can send certain requests to the server, it also helps the back-end to be sure that the requests it receives is actually from where it is expecting, which is the front-end. In addition, logging in is also used to uniquely identify users as each user has their own gallery and information that can be shared with other users, once logged in, the user information is stored on the front-end to help reduce the number of request needed and increase load time.

#### EXPLORE PAGE

As shown in *Image 9. Explore page*, this page is for displaying different user’s posts. Not only that, posts can be filtered based popularity or published date and how the popularity of a post is decided will be discuss in the next section.

A post’s popularity is a number representing the number of interaction it has, the higher the number, the more popular the post is. This number is modified by 3 different type of interactions:

* Click: this interaction happens when a user click on a post in the explore page or a post in a user gallery page. Every click interaction increase the interaction number by 1 unit.
* Like: this interaction happens when a user like a post on a *post page*. Every Like interaction increment the interaction number by 2 unit.
* Comment: this interaction happens when a user comment on a post on a *post page*. Every comment interaction increment the interaction number by 2 unit.

The increment of interaction number is done by sending a request to the back-end ’s interaction view to increment the “current interaction” number, after a certain amount of time, the interaction number is updated using the “current interaction ”.(For further details about how the back-end handle this, check the *Popularity protocol*)

Another important feature of the front-end is how new posts are retrieved and displayed for each filter type: popular and latest. Firstly, posts are fetch in batches of 5, each batch contain posts different from the batch, this is the same for both filter types. Secondly, the front-end store an index “i”, this indicates the position of the first post of a batch, once the position is defined, a request with the index is sent to the back-end. Sequentially, the back-end receives the request, fetch all records of the post table, order them in popularity in descending order, and find the “index” position of in the record and fetch 5 post starting from position “index” to “index” + 5. Finally, the front-end receive the respond, render the 5 posts and increment the index by 5 unit. Every time the user press the space bar, a request for new batch is sent to the server and this process is repeated.

#### ROOM LOBBY

Room lobby is a page for creating and displaying rooms. The purpose of this page is to store drawing session so that user can create and save drawing sessions as well as joining them easily. Furthermore, a user could easily find all drawing room as they are displayed publicly and they find the room they joined easily as they are displayed as well. Lastly, user could also easily and quickly create a room and join it, each time a user create a room, that user become the host of that room and only that user can delete that room.

#### ROOM CANVAS

Each room will contain a canvas and several tools for interacting with the canvas. In this section, we will be discussing about the important components that management user interactions. The components include:

* Canvas
* Layering system
* Color picker
* Tool bar
* Online status
* Chatbox

##### CANVAS

Each room will have 1 canvas assigned to it, a canvas is a combination of html canvases stacking on top of each other, these are call layers. A user must choose a layer they want to drawing on, then, provided that they have permission to draw on that layer, they can start on drawing. However, if multiple client try to draw on the same layer, different brush stroke will interfered with each other. In order to fix this, only 1 user have permission to draw on a layer, if other client want to gain permission, they have to ask for permission using the layer system component. If a user leave a room, all the layers they have permission to draw will be locked, meaning no body in the currently drawing session can gain permission, but if another user joins the session after that user leaves, that newly joined user will have permission to draw on all layer the left user owned.

LAYERING SYSTEM

* Layer permission:This component is used to manage selection and permission of layers. Initially, when the first user enters a room, that user gains permission of all layers and can draw on all layers. Later, if other layer enter they won’t have permission to any layer as all layers are already owned by the first user who entered the room. In addition, If a user wishes to gain permission for a layer, they will have to click on the layer they are targeting, a web-socket message will sent to all client who have permission for that layer(in this case there is only 1 person) and that user will reply with yes or no. Based on their reply, the asking client will gain permission or be declined to gain permission.
* Show or hide layer: each user can display or hide a layer by click on the “eye” icon, this allows a user to show a layer if they want to see or interact with it, or to hide a layer because they want to focus on their drawing or just to display only their drawing on their layer.
* Select a layer: each user can select a layer by clicking on it on the layering system, however, selecting a layer does not allow that user to draw on it as they still need to wait for the respond from the user who has permission on that layer.

##### COLOR PICKER

This is a color wheel where user can select the color for their brush. There are many options for user choose color: they can manually, click on a color region to pick a color or manually enter an rgb value of hex value to get a color they want.

##### TOOL BAR

There 3 tools that user can use: brush, eraser and eyedropper.

* Brush: a brush for drawing lines with many options that user can change such as: width and opacity.
* Eraser: a brush for eraser lines with many options that user can change such as: width and opacity.
* Eyedropper: a tool that let the user pick a pixel color on the canvas.

##### ONLINE STATUS



Image 11. online status

This is a component that display who is currently in drawing session. Each user, when enter a drawing will send a “is\_online” message to all clients in the same drawing session. After that, they will need to send a “who\_is\_online” message to all client and receive all the “is\_online” message from those clients, finally adding all those user to their online list. When a client leaves a session, they will need to send a “not\_online” message to all client in the drawing session and they will remove that user from the online list.

##### CHATBOX

Each user will have a chatbox that store text messages from all user and each user can send their own messages using the chatbox. Initially, when a user finishes typing their messages and press enter, a web-socket message is sent to the server and delivered to all clients, client who receive this message, will check if the message is from them or from other, if it is from them, style it differently from messages received from other clients.

SAVING

After every stroke on a layer, each layer is converted to dataURL and sent to the back-end to store as an image. Next time the user enter the room, after loading the layers, messages will be sent to the server to retrieve the images for all layers and draw those images on the layers. But this saving feature is only done by the client, who does the stroke, not client who receive and render the stroke from web-socket messages. This prevent race condition from modification from many users. Moreover, this auto-saving feature means user does not need to save their work before leaving the session.

## 4.4 Minimum vial product (MVP)

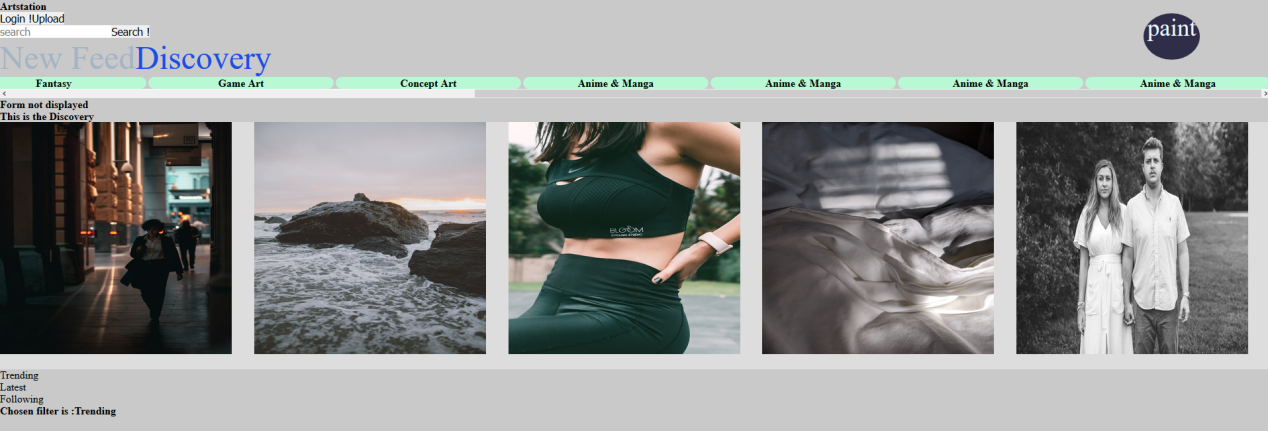


Figure 35. initial idea of explore page

This is the explore page, the components of the page include:

* The main area for displaying posts from users that the account is following.
* A tab containing all the type of post: All, illustration, character design..., which can be clicked on by the user to filter all the post currently in the post area.
* A tab to filter trending or latest posts.
* Each post can be clicked on and then the user will be led to the post page displaying information about the post and its owner’s account.
* A search bar for searching posts by keyword.
* The paint button will direct the user to the drawing page that will be documented later below.

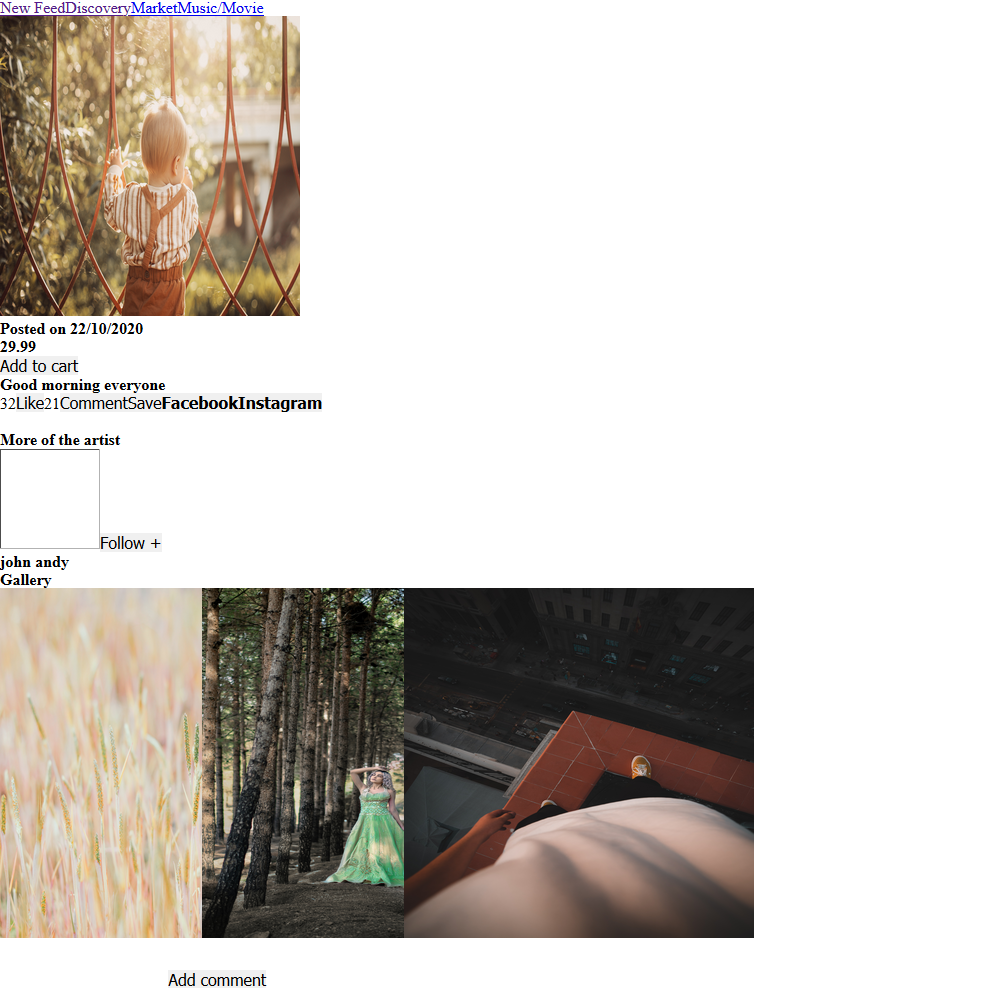


Figure 36. Initial idea of user gallery

This is the post page for displaying a post the user click on, this page include:

* A comment sections
* The post’s image
* Like/comments number
* Post date
* Follow/Like/Save button
* User’s gallery

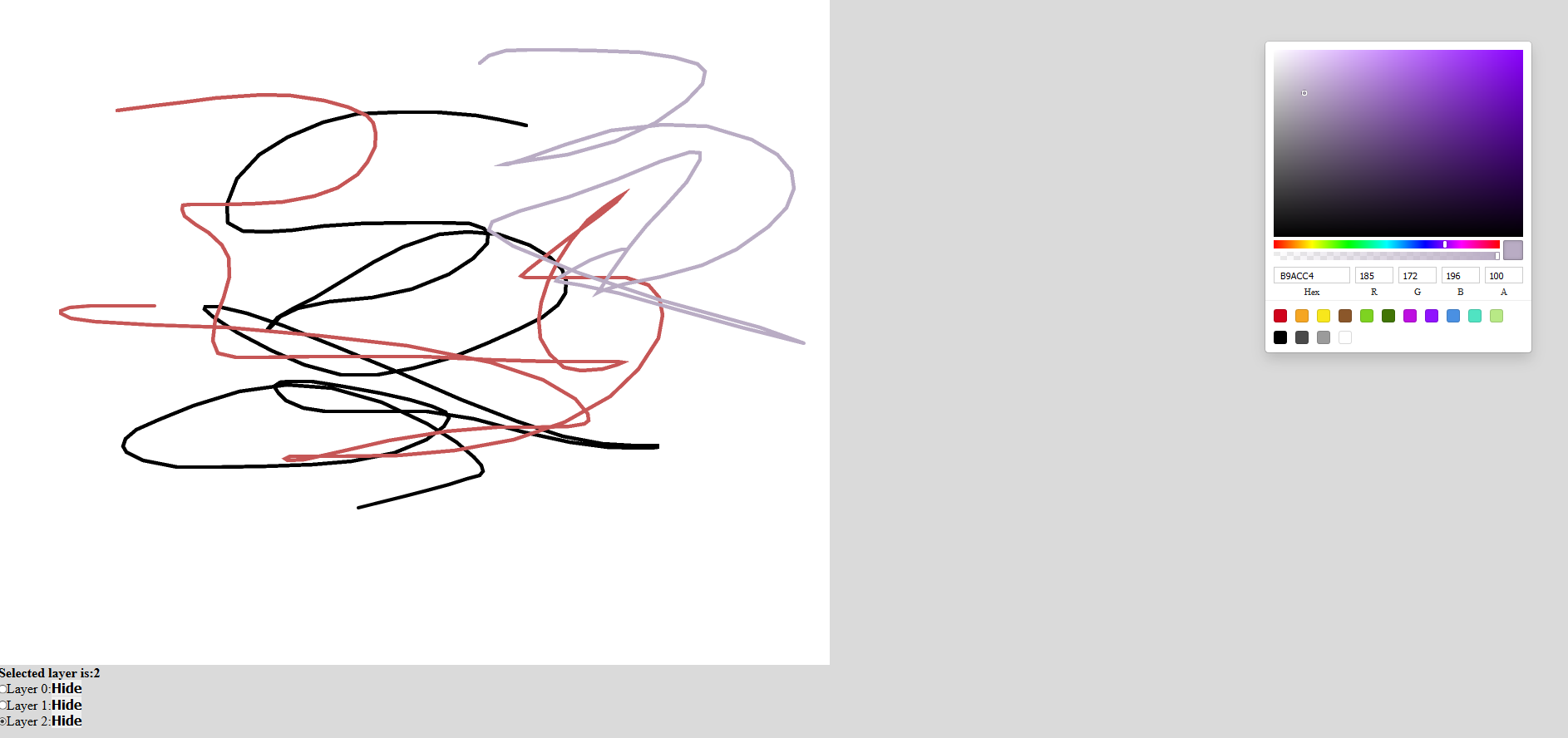


Figure 37. Room canvas

This is the Drawing room , later in the development there will be a list of room and each room will have its own canvas as above, and each canvas will have many layers the user can draw on. This page’s components include:

* A canvas with a list layers that can be turn on and off and user can decide on which layer to draw on
* A color board for choosing color
* Each layer be turn on/off in terms of visibility and interaction and user to check one and only one of the layers that they want to draw on.

## 4.5. Evaluation

For evaluation, I have chosen to utilize Google survey to get feed back from my primary stakeholder, artists. In addition, I have separated the the evaluation process into 2 stages: evaluate the project idea and evaluate the project user’s experience. At the moment, I have only been able to conduct the first stage and have gotten several feedbacks, leading some changes to the design and ideas of the project. Furthermore, I have finished the core functionalities of the application and is proceeding to the second stage of the evaluation by posting a survey consisting of a video demonstrating the core interfaces and functionalities of the application as well as some questions about the video to get the feedback from the different users about the project. In summary, I have finished the first stage of evaluation and is moving onto the second stage, the next section will be the discussing about the first stage and its questions.

### 4.5.1 Evaluating the project ideas

Firstly, this evaluation’s process uses on-line survey to get feedback from stakeholders, below are the questions and their results as well as reasoning why those question had been asked.

**Question 1:**

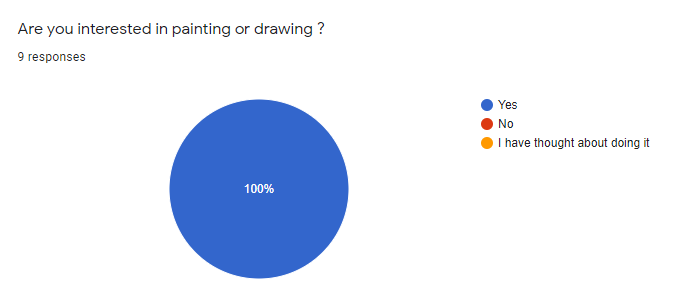


Figure 38. Question 1

This question is for filtering the correct stakeholders, which are artists

**Question 2:**

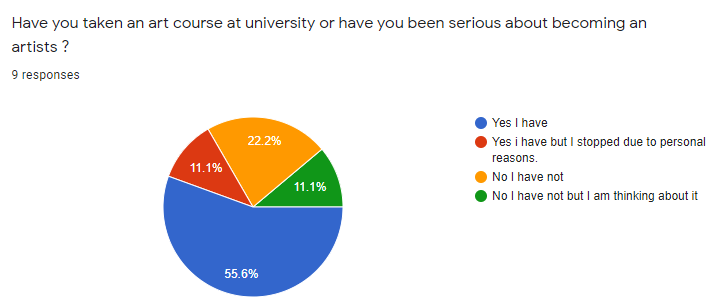


Figure 39. Question 2

This question is to clarify how interested each candidate is in drawing

**Question 3:**

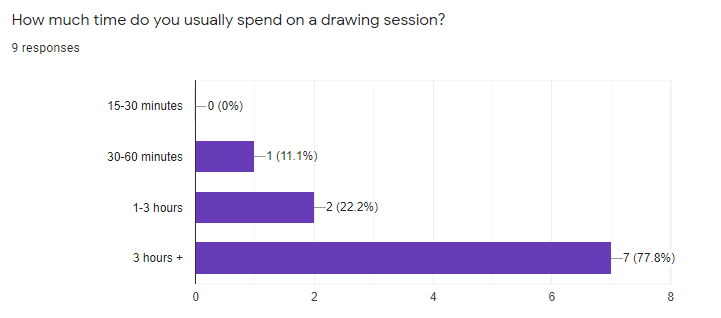


Figure 40. Question 3

This question is to monitor how much to time users tend to spend on drawing sessions to maintain the performance of the application.

**Question 4:**

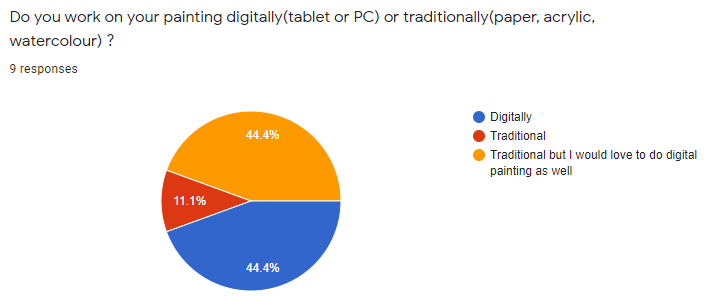


Figure 41. Question 4

This question is to check if the project’s way to approach painting is suitable for the candidate, if they prefer drawing digitally or traditionally.

**Question 5:**

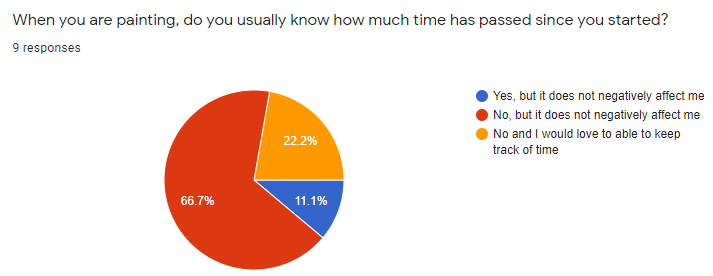


Figure 42. Question 5

This question is to check how concious each candidate is when they are drawing to monitor how much their mental health are affected.

**Question 6:**

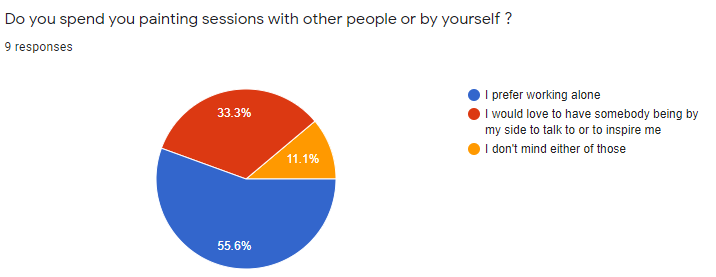


Figure 43. Question 6

This question is to check if the candidates are interested in drawing as a group.

**Question 7:**

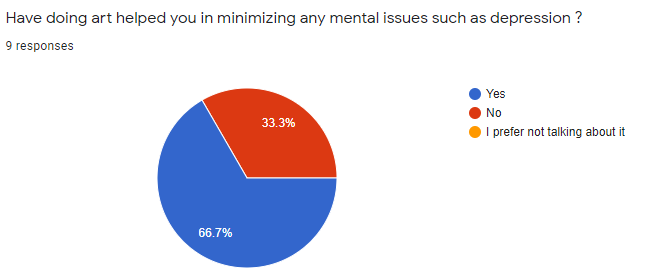


Figure 44. Question 7

This question is to check the effect of drawing has impact on ones mental health.

**Question 8:**

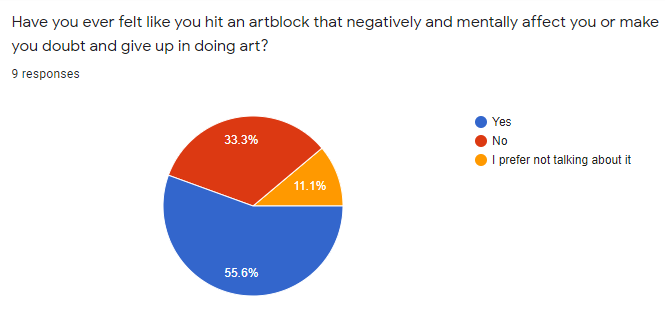


Figure 45. Question 8

This question is to prepare for the next question

**Question 9:**

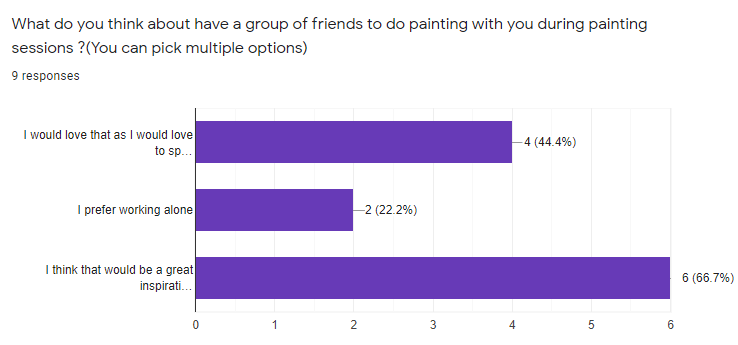


Figure 46. Question 9

This question helps evaluate if the application can help the problem from the last question.

**Question 10:**

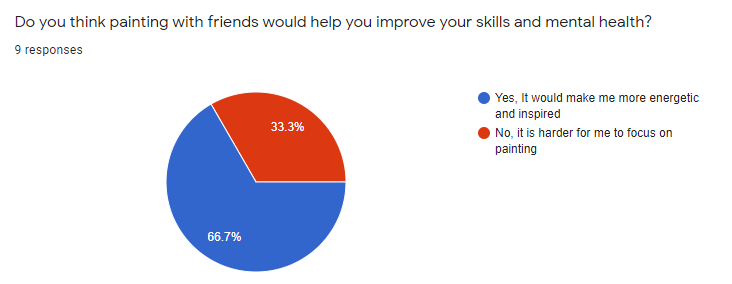


Figure 47. Question 10

This question helps strengthening the project’s solution to the mental health problem.

**Question 11:**

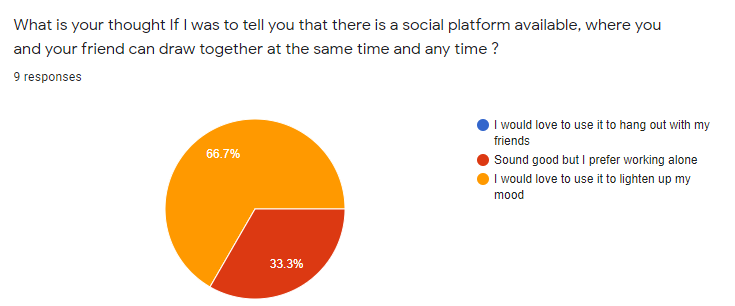


Figure 48. Question 11

This question helps evaluate how enthusiastic people are in using the application

**Question 12:**

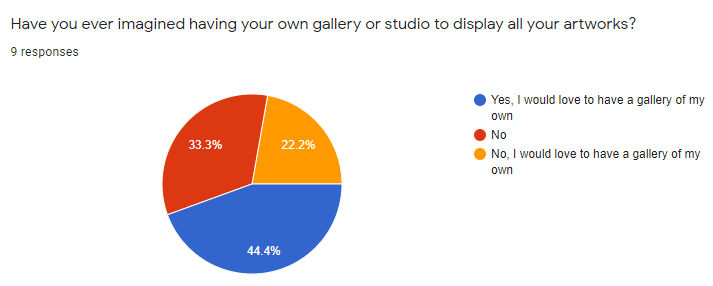


Figure 49. Question 12

Since this application is a community application, sharing is an essential element of it, this question helps capture if the idea of having a personal art gallery for sharing is positively accepted.

**Question 13:**

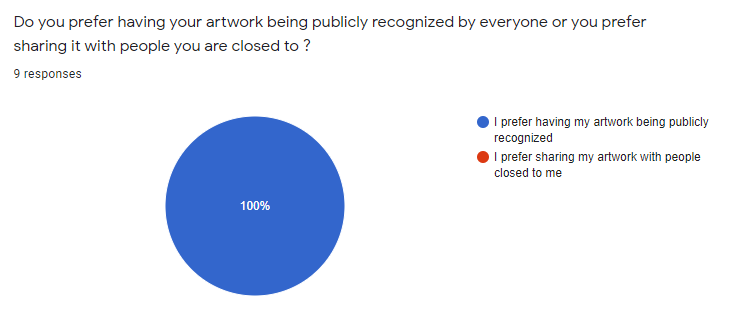


Figure 50. Question 13

Since this application is a community application, sharing is an essential element of it, this question helps capture if the idea of sharing art is positively accepted and different point of views in sharing artworks

**Question 14:**

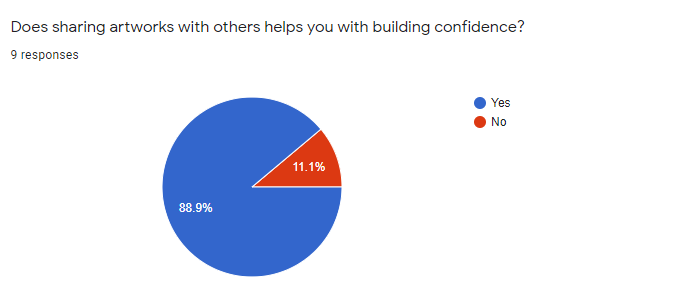


Figure 51. Question 14

This question helps in evaluating how sharing artworks impacts the mental health.

**Question 15:**



Figure 52. Question 15

This question is see each candidate’ s individual thoughts on the idea of being able to share their artwork anywhere and any time to influence the design of the application

### 4.5.2 Summary

In summary, after receiving the feedbacks from stage 1, it could be firmly assumed that the idea is well received and has potential in solving the mental health problem discussed in the other sections of the report. Furthermore, this also helps guide and update the design of the application to be more suitable for the users, this mean the high level design and low level design must be changed as the requirements are now different from the original assumptions, the changes are generally as follows:

* The concept of creating groups of people for drawing still remain the same as the feedback are positive about the idea.
* The original idea of simply having a place to display artworks publicly are has been changed to a design that allow user to share their artwork at ease while still being able to display them to the application’ s community publicly.
* There is no longer the concept of public and private rooms as users prefer having their drawing to be publicly viewed.

# **Chapter 5: Risk management**

# **Chapter 6: Progress summary (To be finished later)**

Firstly, the core functionalities has been completed, they include:

* Front end:
  + Design and implementation of interfaces for creating rooms
  + Design and implementation of interfaces for displaying/joining/leaving rooms
  + Separate interfaces for displaying user’ s joined rooms and all public rooms
  + Front end authentication for user registration and log in using tokens
  + Design and implementation of interfaces for canvas and canvas’s layers
  + Implementation of tools for user to use on canvas
  + Implementation of client web-socket connection
  + User can now establish web-socket connections for live painting and texting while in a group painting session.
  + Improved performance for sending data through web-socket.
* Back end:
  + Build 2 consumers for handling web-socket messages: chat box consumer for handling chat-box messages and client’ s brush stroke consumer to send brush stroke to all channels.
  + Build all database’s table required
  + Build View Sets for easy handling of http requests and authentication for different clients using tokens to have more security

Secondly, as I have mentioned in the evaluation section, there are changes that have been made so the current implementation is currently being changed to fit the new design. Lastly, I am currently preparing the questionnaire and videos for the next stage of evaluation and is planning on publishing it on google survey in the next week.

# Reference

1. Tanner Christensen(July 2013). https://creativesomething.net [Online]

<https://creativesomething.net/post/55508909341/the-link-between-depression-and-creativity-and>

1. Celine Terranova(Januray 2020).theparttimeartist.com[Online]

<https://theparttimeartist.com/loneliness-and-being-an-artist/>

1. Nuacht UCD(June 2012), www.ucd.ie[Online] <https://www.ucd.ie/news/2012/06JUN12/130612-Socialising-helps-to-alleviate-symptoms-of-depression.html>
2. NHS[Online]www.nimh.nih.gov

<https://www.nimh.nih.gov/health/topics/depression/index.shtml>

1. Lindsay Dodgson(July 2018)[Online]

<https://www.insider.com/the-link-between-creativity-and-mental-health-2018-7>

1. Bulfut(2019)” Socialization Helps The Treatment Of Depression In Modern Life ”
2. rtor.org(July 2018)[Online]

<https://www.rtor.org/2018/07/10/benefits-of-art-therapy/>

1. Concept Art Empire[Online]

<https://conceptartempire.com/digital-painting/>

1. Adobe. Adobe’s beginer’s guide to digital drawing tools[Online]

<https://www.adobe.com/uk/creativecloud/illustration/discover/digital-pens-digital-tools.html>

1. Django. Django.com [Online]

<https://www.djangoproject.com/start/overview/>

1. Kalpit(Aug 2018). When To Use(And When not to).medium.com[Online]

<https://medium.com/crowdbotics/when-to-use-django-and-when-not-to-9f62f55f693b>

1. developer.mozilla.org[Online]

<https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server>

1. Positive Stud(September 2020).What is WSGI(Web Server Gatewat Interface).www.medium.com[Online]

<https://medium.com/analytics-vidhya/what-is-wsgi-web-server-gateway-interface-ed2d290449e>

1. ASGI(March 2019). ASGI(Asynchronous Server Gateway Interface)[Online]

<https://asgi.readthedocs.io/en/latest/specs/main.html>

1. Geeksforgeeks(Dec 2019)[Online]

<https://www.geeksforgeeks.org/what-is-web-socket-and-how-it-is-different-from-the-http/>

1. Ruben Dinis(2018). A simple Real Time Chat with Django Channels and React[Online]

<https://revs.runtime-revolution.com/a-simple-real-time-chat-with-django-channels-and-react-b73edc3a79f2>

1. Djangobook.Django’s structure-a heretic’s eye view, Figure 3.1[Online]

<https://djangobook.com/mdj2-django-structure/>

# **Appendices**

This section includes risk management, Gantt chart, project schedule time line and the supporting material for the applications

**Supporting material:**

The source code of the application is inside the “application” folder, inside, the front-end and back-end are separated into 2 folders, due to lack of time, I was not able to deploy the application, so currently it can only run on local-host, but all the functionalities are there. In order to run the application, there are several required tasks to be done:

* Download and install Docker
* Download and install conda(Miniconda)
* Download and install node.js
* Go the “Application” folder where there are 2 folders:”frontend” amd “backend” and do the following tasks:
  + Open a terminal in this “Application” folder
  + Using the terminal, enter these commands in this order:

conda create --name FinalYear python=**3.7**.**9**

conda activate FinalYear

conda install pip

python -m pip install Django

python -m pip install -U channels

python -m pip install channels\_redis

conda install -c anaconda pillow

python -m pip install django-cors-headers

conda install -c conda-forge djangorestframework

**Project schedule:**

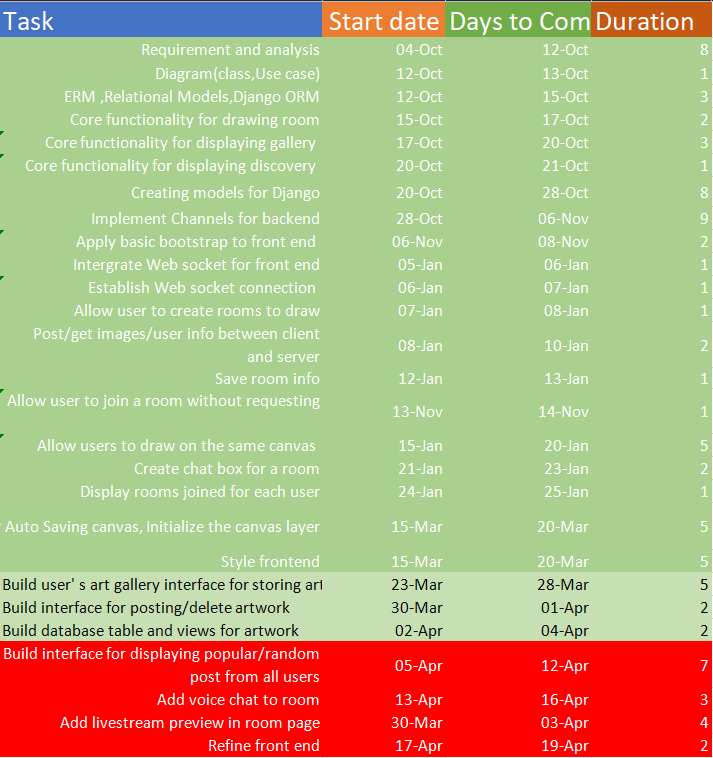


Figure 1. Project’s schedule

**Gantt chart:**

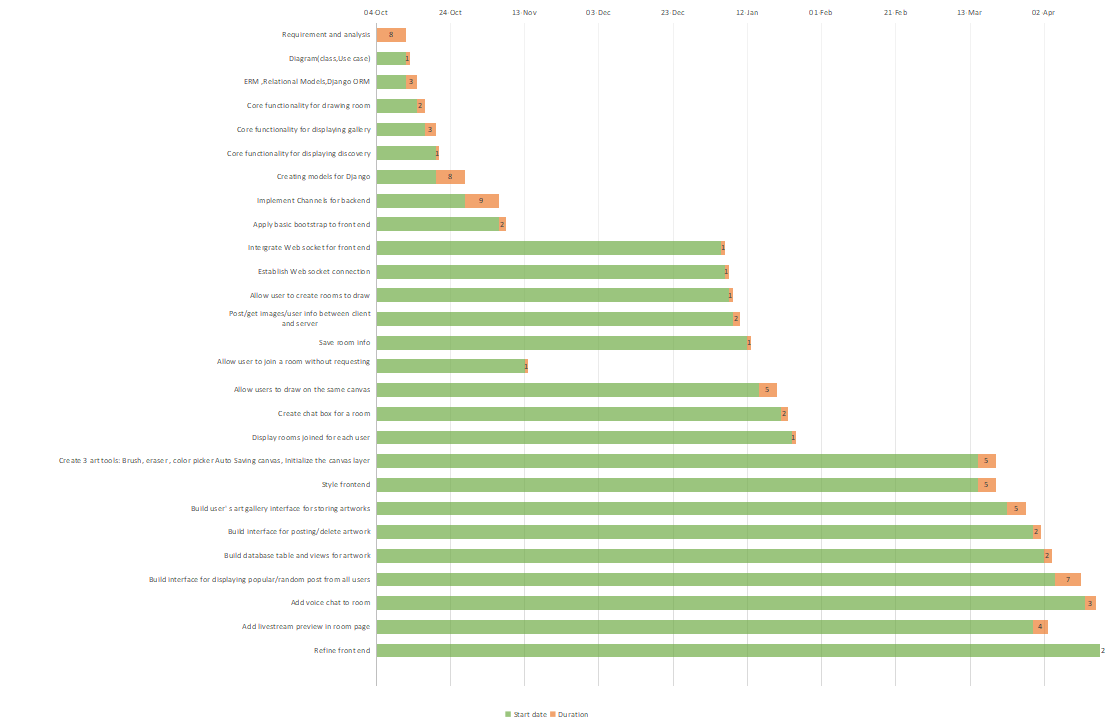


Figure 2. Gantt chart

**Risk management:**

Initially, this project is mostly different from the current version of it. At the start, my idea was to create a platform for people to share their artworks and to open their own on-line art store, but later on I realized the problem I was trying to solve, which is the idea that many artists have a hard time with socialization, was too broad so it needs to be more narrow. Because of this, I decided to look further into the connection between artists and mental health issue, how this connection is established and how dangerous it is, thus changing the idea of the project. The hardest part for me in the beginning was to clearly define a problem and the next problem was to define a solution to that problem as mental issues affect people differently and each person has their own way of dealing with their problems. Even though, later in semester 1, my problem was more consistent defined, I was a lot more busy with other modules and didn’t have much time to work on the implementation, which was suppose to take a lot of time, so I had to scale down the project to fit to the schedule.In summary, the initial problem I was trying to solve was too broad and that make it even harder to develop a solution for it, due to this, I was not able to move on to analysing the foundation structure of the project and had to postpone and reschedule my task.

Secondly, as I have mentioned before in the implementation chapter, I am using Django framework as my sever and I have never worked with any other framework but Reactjs, so it took a considerable amount of time to be familiar with the new framework. Furthermore, I was not familiar with python at the time so I had to spent more time learning from different application and examine how they combine different frameworks together. Later on, based on my research, in order to have a real time application, it is required to use other application protocol from http and web-socket protocol was a candidate. At this point, I discovered that I need study about web-socket protocol and how they work and this forced me to push back my schedule again. After learning all the required frameworks and protocols, I was able to continue to work on the implementation, but then I found a problem that is when multiple attempt to use html canvas context2D to perform a brush stroke happens at the same time with interfere with each other and create a race condition, so I had to find a solution to this. This was the hardest part of the implementation process as I couldn’t find much resource about html canvas applications, fortunately, I was able to come up with a solution, which is inspired by a web application called magma studio and idea is limiting the number of people that can draw on the canvas and at this point, the only problem left was to implement it. Overall, in terms of implementation, the hardest part was to come up with and implement the permission protocol to limit people from using the canvas.

Finally, the last problem I had was related to the performance of the application, where each brush stroke of one client was not consistently rendered on other clients. The source of the problem was that, I allowed the brush to have an opacity level, where the higher the level is, the less transparent each brush stroke become. Due this, each stroke’s opacity is rendered randomly and make the stroke less consistent with some part of it being more dense then the others. This took me a long time to fix as I have said before, there wasn’t much resource about html canvas.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Implementation | | | | |
| **Name** | **Description** | **Frequency** | **Impact** | **Solution** |
| **Developer experience** | This risk happens during development when the developer has implementation related problems such as: not knowing how to solve a problem with using frameworks/ programming languages, how to integrate back-end with front-end, understanding how web socket protocol works, ... | High | High impact | **Avoidance**: Do research about the tools and framework/modules needed for the development before implementation  **Solution**: Scale the project down to suitable needs while still providing the initial requirements |
| **Version control** | The code is stored on the disk which means if the disk was to be broken or damaged, its data will be lost , thus losing all the project’s resource | High | High impact | **Avoidance**: Use of version control such as git to store resources on a server so that we have a copy that can be pulled from or push to. This allows the code to be safely preserved for most of the time  **Solution**: Scale down the project and find tools most suitable for fast development. |
| **Lack of resource preparation** | This problems occurs when at the start of the project, after specifying requirements, a lists of tools need to be evaluated and selected to be used for development. However ,later in the project ‘s life-cycle ,an event such as requirement changes or an external frameworks/tools that have not been included initially are needed, this requires the integration of those tools into the application but several obstacles might appear including: developer is not familiar with the tools ,putting more time on planning the project leading to delaying. | Low | Medium impact | **Avoidance**: None  **Solution**: Evaluate if the tools will take a long time to adapt , if not, proceed to apply plan and apply to the project, otherwise scale the use of the tool for the project down so that only the core elements are implemented. |
| **Requirement** | | | | |
| **Name** | **Description** | **Frequency** | **Impact** | **Solution** |
| **False collection** | The foundation of a project is collecting the required elements from the client and knowing the aim of the application, if not prepared probably could leads to developing a product not in need or not satisfying a target group of stakeholder | Medium | High impact | **Avoidance**: Focus on the targeted requirement and stakeholder of the application and implements core features instead extra features that contribute little to the product and compare the aim of the end product with the requirements noted. |
| **Requirement change** | Application is usually develop based a number of predefined requirements, however ,their might be changes need to be made due to the need of the customer has changed or the scale of the projects, thus demanding evaluation and adjustment to the code | Medium | Medium impact | **Avoidance**: None  **Solution**: Analyse and apply framework/tools the take the shortest to adapt to satisfy the changes |
| **Requirement difficulty** | Requirements form the foundation of project, however if the requirements exceed the capability of the developer, this could be a problem as this requires knowledge of different methods to satisfy the requirements | High | Medium impact | **Avoidance**: Understand the fundamentals required to handle to the project, such as , frame works and protocol.  **Solution**: Analyse choose the tool that provide the quickest time to learn while providing the fast pace development to complete the requirement in the most effective way |
| **Time management** | | | | |
| **Name** | **Description** | **Frequency** | **Impact** | **Solution** |
| **Planning** | Once the analysis process is done., next step is to create the time-line and milestone for the project. This is important as the project must meet the deadline and satisfy the requirement, if the planning and time-line is not planning equally in terms of work effort and length and not stating the core milestone for the core features, it would be confusing or might leave out some core aspect of the application | High | High impact | **Avoidance**: Give more time than needed for each planned task to ensure that there is extra time to handle time consuming problems that possibly occur later  **Solution**: Identify the big milestones for project and using those milestones to establish the timeline plan. |
| **Delay from changes** | In the life cycle of the project, many problem could occur in term or changes: requirement changes, lack resources leading to extra needed for adapting to new tools,....As a result , the plan deadline couldn’t be met. | High | High impact | **Avoidance**: None  **Solution**: Scale down the project to be suitable for the timeline. |
| **Personal** | | | | |
| **Name** | **Description** | **Frequency** | **Impact** | **Solution** |
| **Personal health** | During development, it is unpredictable what cold or disease a person might catch, if this happens project delay will happen | Low | High impact | **Avoidance**: Spread the time so that there is rest time between milestone while still meeting the deadline and taking care personal health |
| **Environment** | A project from a group or individual can have member from anywhere ,whether it is a different country or has a different time zone or the place the member living in is unideal, for example lack of internet | Low | Medium impact | **Avoidance**: Divide the work equally depending on the environment condition of member |
| **Other projects** | There are other projects a member might have, this also contribute to the ability to work punctually and meeting the deadlines. This is because, for example, if ones has to do a side project along with this project and still has to follow the planned timeline, the result might be not being able to finish his/her part at the right time or the task is done poorly | Low | Medium impact | **Avoidance**: Divide the work equally so that there is time for other projects and task  **Solution**: Divide the work equally so that there is time for other projects and task |
| **Copyright** | | | | |
| **Name** | **Description** | **Frequency** | **Impact** | **Solution** |
| **Plagiarism** | Using resources on the from other existing software is a way to identify the flaw of the project however, it is important that the project does not copy the implementation of the others | Medium | High impact | **Avoidance**: Only compare the project and the others existing ones to identity and understanding the implementation then think of a way to implement. |
| **User plagiarism** | The end product might be plagiarized or stolen by other individual who is not a part of the project. | Medium | High impact | **Avoidance**: Avoid exposing or leaking project’s resources before publishing and have strong security to protect source code. |