29 MARCH 2022

CSN - 254

DESIGN DOCUMENT

MEMBERS:

BINDU	20114024
DEEPANSHU	20114030
DEEPANSHU MEENA	20114031
PRIYA SINGH	20114077
SONALI GUPTA	20114094
YOUWAN SONI	20114107

SUMMARY

This design document has been developed to elucidate the design and
workflow implementation of key features of Arthub. The primary goals this

design document will accomplish are to increase operational efficiency and
to provide a sustainable base and framework around which the coding phase
can be structured. This document provides detailed recommended
implementation for multiple features like login-signup, result declaration, buying an art, giveaways and managing the sponsors. We will also be looking at the possibility of incorporating a bidding system property. It details both the logical and physical design considerations related to all infrastructure components including storage,
encryption and registration. This document identifies the points of contact of
the project, lists implementation requirements, provides a brief description of
all the deliverables and provides an overview of the implementation process
for the project. The scope of this document is specific to the implementation of the participation and buying and bidding algorithms. The workflow of the user interaction use - cases and the deliverables involved have been elucidated intricately for the programmer by making use of
well-developed use case diagrams. The user workflow and interaction with
the application frontend and its response by the application backend has been

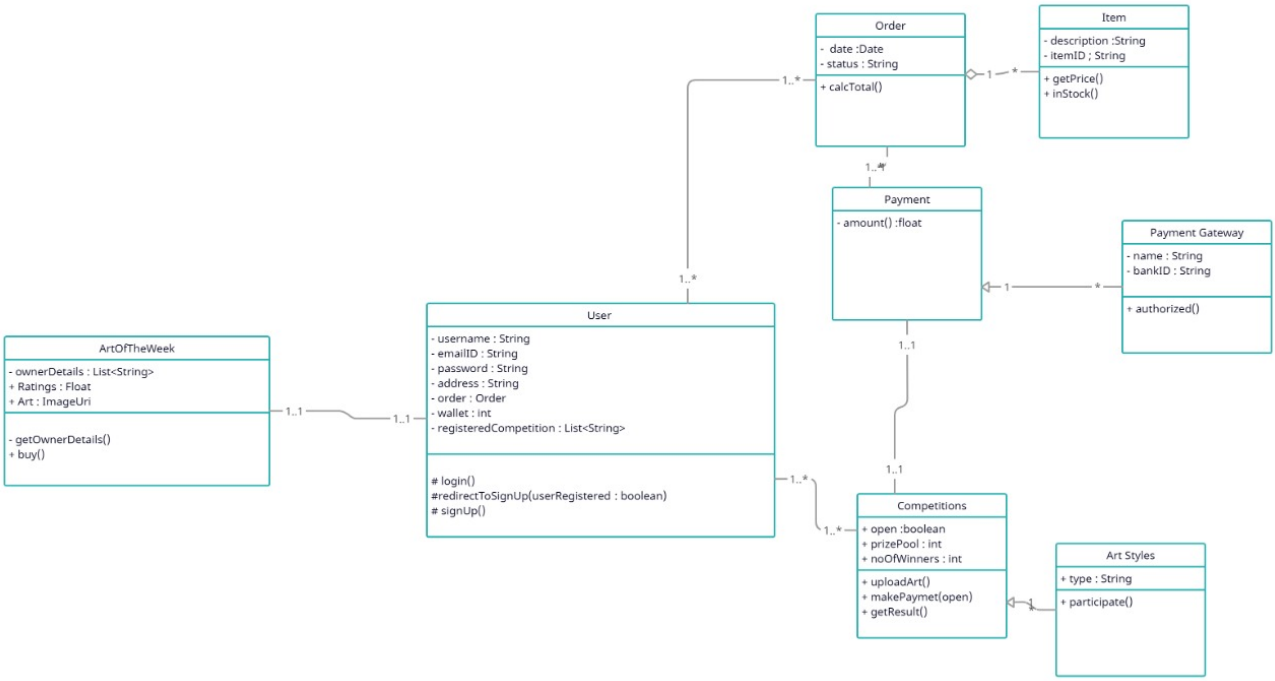
detailed further by employing a flow chart diagram, which will help in
tackling any logical issues that arise with the working of the application in the
coding phase. The developer can also find a well - documented class diagram
in this design document, which will prove to be very handy while coding up
the entire project and if referenced properly at the right intervals, the documentation of the class diagram can provide useful insights.

The design specifications include the following things:

1. **High Level Design**
   a. Use Case Diagram
   b. Class Diagram
   c. Sequential Diagrams
2. **Low Level Design**
   a. User Authentication algorithms.
   b. Participation in competitions and buying algorithms.

DIAGRAMS

CLASS DIAGRAM



- **User:**
  - Attributes:
    - email(private): stores the email id entered by the user
    - username(private): stores the username entered by user.
    - password(private): stores the password set by the user.
    - address(private): stores the address of the user
    - order(private): an object of the Order class which stores information about the orders of the users if any.
    - wallet(private): stores the info about inBuilt wallet of the user.
    - registeredCompetition(private): stores the info about the competitions of the user.
  - Methods:

        - Login(): lets users to login into the system after the details are authenticated.
        - SignUp(): lets the user signUp into the system by entering the details and verifying if those already exist or not.
- **Competitions:**
    - Attributes:
        - open(public): stored information of the competitions being free or not.
        - prizePool(public): stores information about the total prize pool of the competitions and also the prize distribution.
        - noOfWinners(public): stores information about the number of winners among whom the prize has to be distributed.
    - Methods:
        - uploadArt(): allows the participant user to upload his artwork for the competition.
        - makePayment(): allows the participant user to make payment for the competition.
        - getResult(): this method allows the user to see the result of the competition they have taken part in.
- **Art of the week:**
    - Attributes:
        - ownerDetails(private): stores the information about the user details of the use
        - ratings(public): stores the rating of the painting as goven bu the judges.
        - Art(public): stores the image of the art to be displayed to the public.

- Methods:
  - getOwnerDetails(): allows the admin to access the user detail of the owner of the painting as and when required.
  - buy(): allows the user to to buy that painting through the bidding system.

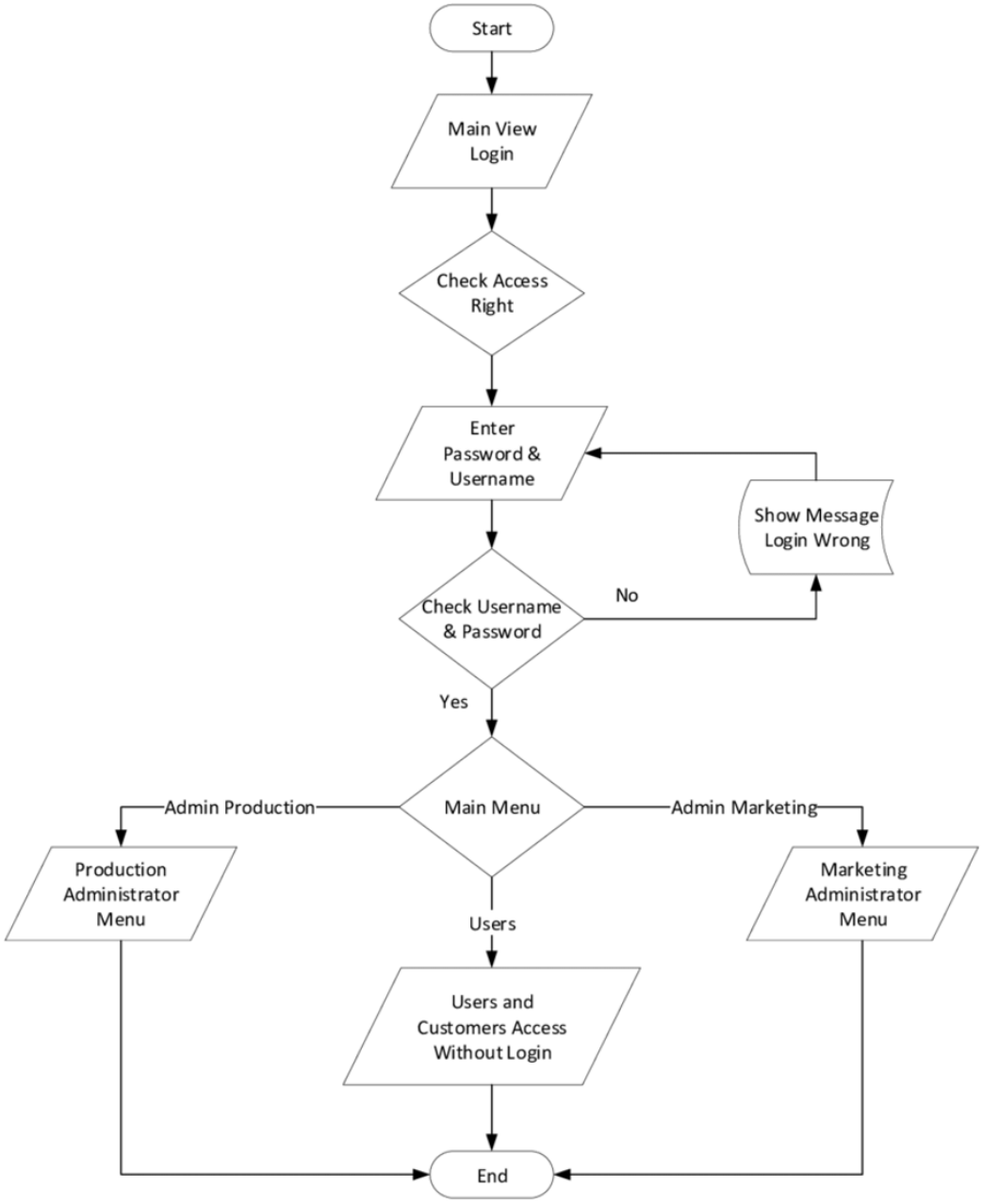
- **Order:**
  - Attributes:
    - description(private): stores the description of the art.
    - itemID(private): stores the item ID of the art.
    - date(private): stores the date of the order.
    - status(private): stores the status of the order being processed.
  - Methods:
    - getPrice(): gets the price of the item for calculating the total amount.
    - inStock(): helps to know if the item is in the stock or not.
    - calcTotal(): calculates the total amount of item including the texes.

USER AUTHENTICATION

1. **User Registration:**
    a. Get the details from the form.
    b. If email already exists in the database:
        Raise an error that displays "email-id already exists".
        Redirect to the signup form.
    c. If the username already exists in the database
        Raise an error that displays "username can't be taken or already exists".
        Redirect to the signup form.
    d. Create an object of user class with the details filled by the user.
    e. Save the user object in the database.
    f. Post a message "account created successfully" and redirect to the home screen.
2. **User Login:**
    a. Get the details from the form.
    b. Verify the username and password entered by the user.
    c. If details are correct authenticate the user and redirect to the home page.
    d. Else raise an error that displays "Invalid Credentials!". Redirect to the login page.
3. **Change Password:**
    a. Get the details from the form.
    b. Verify the old password.
    c. If the old password does not match raise an error that displays "Incorrect Old Password!".

d. Else post a message "Password changed successfully! Please login with a new password." and redirect to the login page.

DATA FLOW DIAGRAM:



Participation in competition:

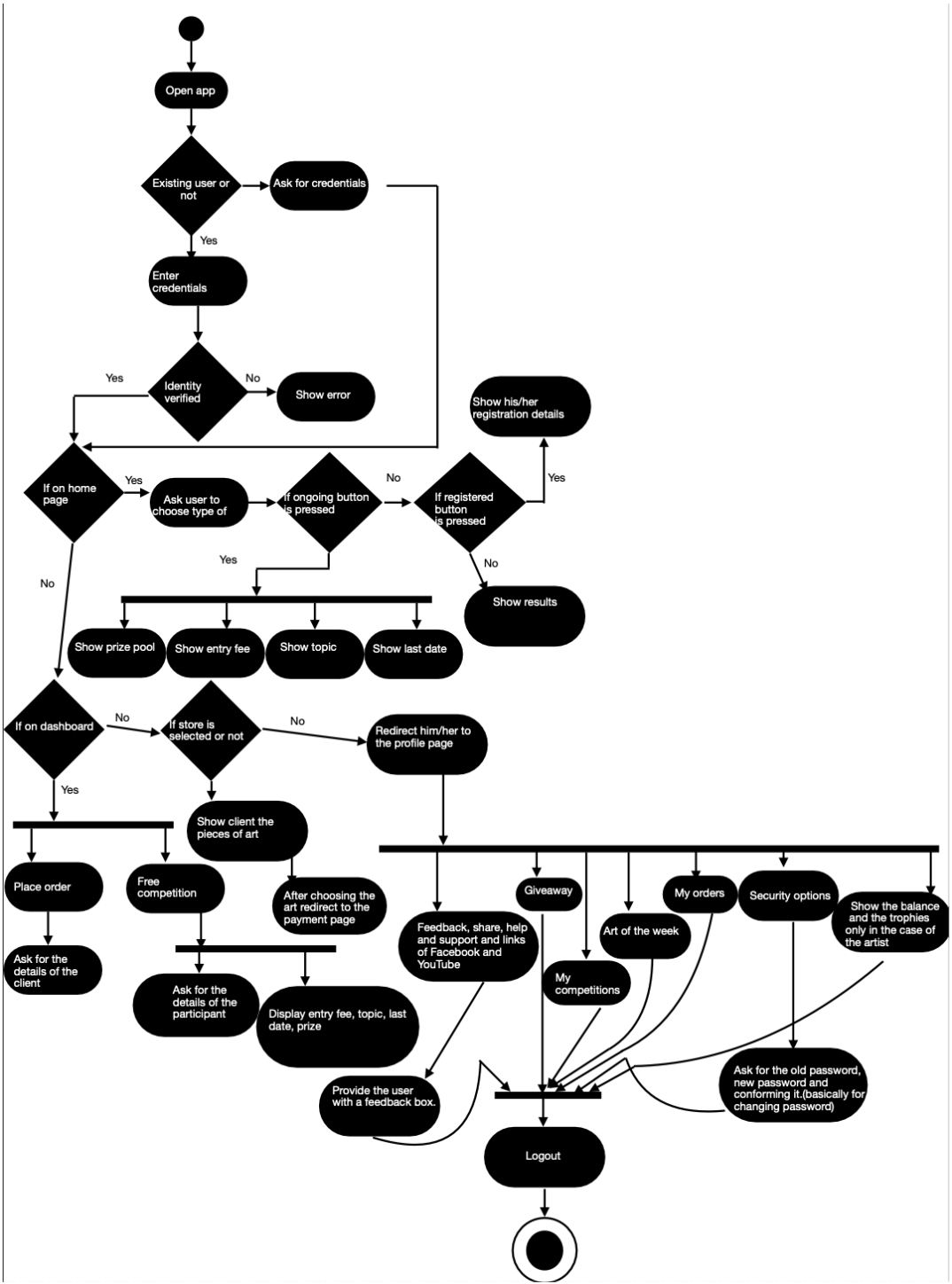
- Get the details of the competitions from the backend from the competition chosen by the user.
- Get the user registration details from the form.
- Check if the user has already registered.
  - If registered show a popup message - "Already registered."
- Save the uploaded art image in the database
- If the competition is not free, redirect the user to the payment gateway.
  - If the payment is successful, show a message "Registration successful"
  - Else show a message- "Registration could not be completed. Payment failed"
- Compile all the registered users and their art, and give ratings to all the painting
- Declare the result of the competition and notify the registered users using app notifications and emails.
- Transfer the winners' money to the inBuilt wallet of the winners.

Buying of paintings:

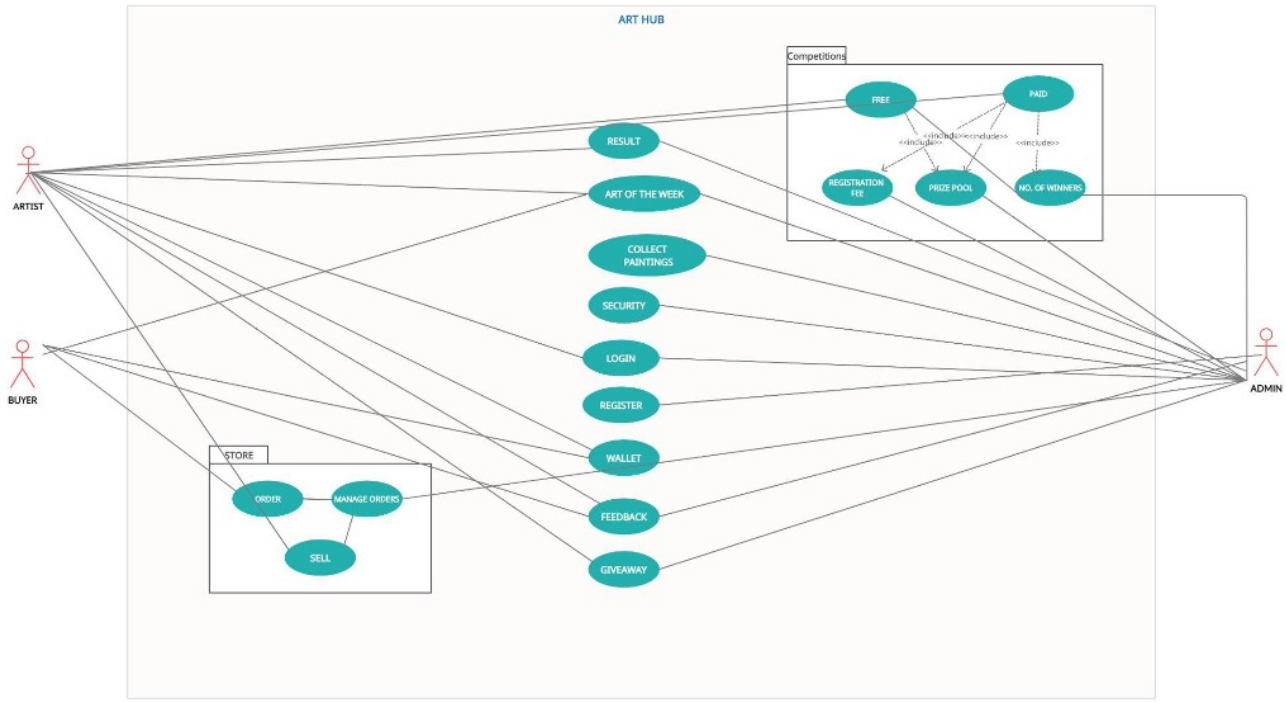
- Get the details of the user and the painting he is interested in buying.
- Get the details of the buyer address and details about the decoratives of the paintings like canvas and frame type.

- The user is then directed to the payment gateway where he enters bank details which are fetched to authenticate the payment.
  - If the payment is successful, send the order details to user at the email address and display a message- "Order placed successfully"
  - Else a message is displayed- "Can't place the order - payment unsuccessful"

ACTIVITY DIAGRAM:



USE CASE DIAGRAM:



COMMENTS:

We have implemented here is an app for all the artists and the people interested in collecting the art. It gives motivation to the artists to work in that field and provides better competition for them. Our app conducts competitions that can be free and paid, and art lovers can contact directly to good artists according to their interests. This document is quite helpful in the coding phase for the app developers and for the users also to completely understand the functionalities on a higher level.