

A Project Report On

“Online Art Gallery”

Submitted to the

Department of Computer Science

In partial fulfillment of the

MASTER OF COMPUTER APPLICATIONS

Under the guidance of

Mr. Haarrish Sabu

Project Done by

SARATH SIBY (Reg No: 223242210932)

KARTHIKA SANTHOSH (Reg No: 223242210928)



SANTHIGIRI
COLLEGE OF COMPUTER SCIENCES

Affiliated to MG University and Approved by AICTE

DEPARTMENT OF COMPUTER SCIENCE

DECEMBER 2023



SANTHIGIRI

COLLEGE OF COMPUTER SCIENCES

Affiliated to MG University and Approved by AICTE

BONAFIDE CERTIFICATE

Certified that the Project Work entitled

“Online Art Gallery”

is a bonafide work done by

Sarath Siby (Reg No: 223242210932)

Karthika Santhosh (Reg No: 223242210928)

In partial fulfillment of the requirement for the Award of

MASTER OF COMPUTER APPLICATIONS

Degree From

Mahatma Gandhi

University, Kottayam

(2022-2024)

Head of Department

Project Guide

Submitted for the Viva-Voice Examination held on

External Examiner1
(Name & Signature)

External Examiner2
(Name & Signature)



SANTHIGIRI

COLLEGE OF COMPUTER SCIENCES

Affiliated to MG University and Approved by AICTE

CERTIFICATE

This is to certify that the project entitled “**ONLINE ART GALLERY**” has been successfully carried out by **SARATH SIBY** (Reg. No: **223242210932**) and **KARTHIKA SANTHOSH** (Reg. No: **223242210928**) in partial fulfillment of the Course **Master of Computer Applications**.

Date:

INTERNAL GUIDE

HEAD OF THE DEPARTMENT



SANTHIGIRI

COLLEGE OF COMPUTER SCIENCES

Affiliated to MG University and Approved by AICTE

CERTIFICATE

This is to certify that the project entitled “**ONLINE ART GALLERY**” has been successfully carried out by **SARATH SIBY (Reg.No: 223242210932)** and **KARTHIKA SANTHOSH (Reg.No: 223242210928)** in partial fulfillment of the course **Master of Computer Applications** under my guidance.

Date

Mr. Haarrish Sabu
Internal Guide



SANTHIGIRI

COLLEGE OF COMPUTER SCIENCES

Affiliated to MG University and Approved by AICTE

DECLARATION

We, **SARATH SIBY** and **KARTHIKA SANTHOSH**, hereby declare that the project work entitled “**ONLINE ART GALLERY**” is an authenticated work carried out by us at **SANTHISOFT TECHNOLOGIES**, under the guidance of **Ms. ROSLINTHOMAS** for the partial fulfillment of the course **MASTER OF COMPUTER APPLICATIONS**. This work has not been submitted for similar purpose anywhere else except to **SANTHIGIRI COLLEGE OF COMPUTER SCIENCES**.

We understand that detection of any such copying is liable to be punished in any way the college deems fit.

SARATH SIBY(Reg. No: 223242210932)

Signature

KARTHIKA SANTHOSH (Reg. No: 223242210928)

Signature

Date :

Place:

ACKNOWLEDGEMENT

A project is not complete if one fails to acknowledge all who have been instrumental in the successful completion of the project. If words were to be the symbol of undiluted feelings and token of gratitude, then let the words play the heralding role of expressing my gratitude.

First of all, we thank the “**God Almighty**” for his immense grace and blessings in our life and at each stage of this project.

We express our sincere and profound gratitude to **Rev. Fr. Prof. Dr. Baby Joseph CMI**, Principal, Santhigiri College of Computer Sciences, Vazhithala for providing all the facilities during the period of the project.

We extend our gratitude to **Mr. Gibin George**, Head of the Department of Computer Science, Department of Computer Science, who is a constant source of inspiration and whose advice helped us to complete this project successfully.

We express our deep sense of gratitude to our project guide, **Mr. Haarrish Sabu**, Assistant Professor, Department of Computer Science, for her profound guidance for the successful completion of this project.

With great enthusiasm we express our gratitude to all the faculty members of Department of Computer Science for their timely help and support. Finally, we express our deep appreciation to all our friends and family members for the moral support and encouragement they have given to complete this project successfully.

TABLE OF CONTENTS

1. Introduction	1
1.1 Introduction	1
1.2 Problem Statement	2
1.3 Scope and Relevance of the Project	2
1.4 Objectives.....	3
2. System Analysis	4
2.1 Introduction	4
2.2 Existing System	5
2.2.1 Limitation of Existing System	5
2.3 Proposed System.....	5
2.3.1 Advantages of the Proposed System	6
2.4 Feasibility Study	7
2.4.1 Technical Feasibility	7
2.4.2 Operational Feasibility	8
2.4.3 Economic Feasibility	8
2.5 Software Engineering Paradigm Applied.....	9
3. System Design	12
3.1 Introduction	12
3.2 Database Design	12
3.2.1 Entity Relationship Model.....	20
3.3 Process Design Dataflow Diagrams.....	23
3.4 Object Oriented Design – UML Diagrams	24
3.4.1 Activity Diagram.....	24
3.4.2 Sequence Diagram	27
3.4.3 Use Case Diagram.....	29
3.5 Input Design	29
3.6 Output Design.....	30
4. System Environment	32
4.1 Introduction	32

4.2 Software Requirement Specification	32
4.3 Hardware Requirement Specification	33
4.4 Tools, Platforms.....	34
4.4.1 Front End Tool.....	34
4.4.1.1 Angular	34
4.4.2 Back End Tool	35
4.4.2.1 Node.js	35
4.4.2.2 MySQL	36
4.4.3 Visual Studio Code.....	38
4.4.4 Operating System.....	39
5. System Implementation	41
5.1 Introduction	41
5.2 Coding	42
5.2.1 Sample Codes.....	42
5.2.2 Code Validation & Optimization	47
6. System Testing	52
6.1 Introduction	52
6.2 Unit Testing.....	52
6.3 Integration Testing	53
6.4 System Testing.....	53
6.4.1 Test Plan &Test Case	53
7. System Maintenance	56
7.1 Introduction	56
7.2 Maintenance	56
8. Future Enhancement and Further Development	57
8.1 Introduction	57
8.2 Merits of the System	57
8.3 Limitation of the System	57
8.4 Future Enhancement of the System	57
9. Conclusion	59
10. Bibliography.....	60

11. Appendix	61
11.1 Coding	61
11.2 Screenshots	74
12. Glossary	85

ABSTRACT

An online art gallery is a platform that allows artists to showcase their artwork and sell it to collectors and art enthusiasts worldwide. It is a virtual space that provides a wide range of contemporary and modern art, including abstract paintings, sculptures, photographs, and more. Online art galleries have become increasingly popular in recent years due to their accessibility and convenience. They offer a unique opportunity for artists to reach a global audience and for collectors to discover new and exciting works of art from the comfort of their own homes. Online Art Gallery contains a database which stores buyer, gallery information. This website allows the artist and buyer the ability to browse for artwork by categories, subjects, artists, tags and prices. This website allows advanced searching where a potential buyer can search galleries based on specifics such as theme, author, date, size, price, or color. The buyer will also be sent to a booking form and payment process to finalize transactions. This website will also incorporate a user survey and comment form.

Online art galleries have revolutionized the way we view and purchase art. They provide a unique opportunity for artists to showcase their work to a global audience and for collectors to discover new and exciting works of art from the comfort of their own homes.

CHAPTER 1 INTRODUCTION

1.1 INTRODUCTION

Online art gallery is kind of web page or website that you can put your art work images for viewing. Usually, nowadays everything is available on online shopping market so why not art works. Every kind of painting either they are traditional medium based or digital art based you can get it online. On some online art gallery website the galleries also conduct art auctions. The purpose of online art gallery is to reach art works every corner of world by using technology. When everywhere things are available on online then why art works leave behind. So online art gallery is a system or a program where the art collector can view, select and purchase art work within their comfort zone. Online art gallery is a user based website in which artist can upload their art images and buyer can view these images, choose and purchase. This concept helps people who are not available to visit the place on particular time due to some constraints. They open the site, go through it and choose what they like. Through this digitization we can stay connected to every corner of the world. And this connectivity of art exhibitions also become online in that the viewership increased and selling of art works also increased. These art galleries had started these virtual art galleries concepts. Art works is communicated by the combination form of the display/exhibition; it is not just an architectural space. Now days, the internet has become a global language of communication this medium offers new and very unique opportunity for artist to grow themselves by selling their artworks online.

Another advantage of online galleries is that the area of people and choice had increased by the internet. Those who don't have the knowledge of art is also trying to understand the art works and get the wide range of art works that they view in the comfort at home.

1.2 PROBLEM STATEMENT

An online art gallery might have limited reach and visibility compared to a physical art gallery. This can be due to a lack of marketing, poor website design, or a lack of awareness among potential customers. Customers might be hesitant to purchase art online due to concerns about authenticity, quality, and shipping. This can be addressed by providing detailed information about the artwork, offering a return policy, and using secure payment methods. Maintaining an online art gallery requires technical expertise in areas such as website design, e-commerce, and digital marketing. This can be addressed by hiring professionals or outsourcing these tasks. Online art galleries face stiff competition from other online galleries as well as physical galleries. This can be addressed by offering unique artwork, providing excellent customer service, and building a strong brand identity. Online art galleries need to store and ship artwork safely and efficiently. This can be addressed by using high-quality packaging materials, partnering with reliable shipping companies, and insuring artwork during transit.

1.3 SCOPE AND RELEVANCE OF THE PROJECT

Online art galleries are platforms that allow artists to showcase their work to a wider audience and sell their art online. They provide a space for artists to display their work without the need for a physical gallery, which can be expensive and difficult to maintain. Online art galleries also offer a range of benefits to art collectors, such as the ability to browse and purchase art from the comfort of their own home, access to a wider range of artists and styles, and the ability to purchase art at more affordable prices.

The scope of an online art gallery project can vary depending on the goals of the project. Some online art galleries may focus on showcasing the work of emerging artists, while others may focus on established artists. Some online art galleries may specialize in a particular style or medium, while others may offer a more diverse range of works.

The relevance of an online art gallery project is significant in today's digital age. With more people turning to the internet for their shopping needs, online art galleries provide a convenient and accessible way for people to purchase art. They also provide a platform for artists to reach a wider audience and gain exposure for their work.

1.4 OBJECTIVES

The main objective of an online art gallery is to provide a platform for artists to showcase their work and for art enthusiasts to view and purchase art from the comfort of their homes. Here are some objectives that an online art gallery project could have:

- **To provide a platform for artists:** The online art gallery should provide a platform for to showcase their work and reach a wider audience.
- **To provide a platform for art enthusiasts:** The online art gallery should provide a platform for art enthusiasts to view and purchase art from the comfort of their homes.
- **To provide a secure payment gateway:** The online art gallery should have a secure payment gateway that ensures that transactions are safe and secure.
- **To provide an easy-to-use interface:** The online art gallery should have an easy-to-use interface that allows users to browse through the collection of art with ease.
- **To provide a search functionality:** The online art gallery should have a search functionality that allows users to search for specific pieces of art based on various criteria such as artist name, medium, style, etc.
- **To provide detailed information about the artwork:** The online art gallery should provide detailed information about each piece of artwork, including the artist's name, the medium used, the dimensions, and any other relevant information.

CHAPTER 2 SYSTEM ANALYSIS

2.1 INTRODUCTION

Software Engineering is the analysis, design, construction, verification and management of technical or social entities. To engineer software accurately, a software engineering process must be defined. System analysis is a detailed study of the various operations performed by the system and their relationship within and module of the system. It is a structured method for solving the problems related to the development of a new system. The detailed investigation of the present system is the focal point of system analysis. This phase involves the study of the parent system and the identification of system objectives. Information has to be collected from all people who are affected by or who use the system. During analysis, data are collected on the variable files, decision points and transactions handled by the present system. The main aim of the system is to provide efficient and user-friendly automation. So, the system analysis process should be performed with extreme precision so that an accurate picture of the existing system, its disadvantages, and the requirements of the new system can be obtained. System analysis involves gathering the necessary information and using the structured tool for analysis. This includes the studying existing system and its drawback, designing a new system, and conducting a cost-benefit analysis. System analysis is a problem-solving activity that requires intensive communication between the system users and system developers. The system is studied to the minute detail and analyzed. The system is viewed as a whole and the inputs to the system are identified. The outputs from the organization are traced through various phases of processing of inputs.

2.2 EXISTING SYSTEM

Suppose we went to the museum or art gallery there we found something which we really want to purchase so for that we need to contact to the manager and talk to him to get aware about the procedure of purchasing, but at the last moment, we come to know that someone has already booked that painting or art there we feel helpless because that painting was on sale for three days and you were not able to purchase it just because of lack of time and the distance you need to cover from your home to art gallery. This thing generally happens in art galleries sometimes we are out of cash and want to pay through credit card but this mode of payment is not allowed in some art galleries.

Here are some of the limitations of the existing system:

- This process is so time-consuming.
- Existing System is not user friendly.
- There is a threat to the record of the customer so it might be the case that one painting can be sold twice.
- There is no proper way of getting new paintings to record customers need to search on their own.
- The data are recorded in register book so duplication of data may occurred.
- By contacting the person one by one is very time consuming.

2.3 PROPOSED SYSTEM

The main objective of the proposed system is to eliminate the limitations of the existing manual system. Most of the limitations of the existing system can be overcome by the proposed system. we manually purchase the products we need more time, Speed and accuracy are the main advantages of proposed system. There is no redundancy of data in the proposed system which we ensure all the. Since all the details are stored in computer searching time paintings can be reduced. The information can be more secure because the computer systems are more secure. The proposed system eliminates the drawbacks of the existing system to a great extent and it provides security of data. The system manages Using this system, users can register on their own details if they are interested to purchase paintings. To register in the system, they have to enter their contact information like address mobile number, email, and password these details are added by the Administrator.

The admin has many facilities like adding various paintings, updating various paintings available, and viewing various customers who have registered into the system. When a person wants to purchase paintings, he has to register to the system. Customer registration is very easy to get registered to the system he has to fill up the registration form. After submitting the registration form, he can create a username and password. The customer has to give information like ,contact details etc. A feedback form is also given into the system if any feedback wants to give to art gallery about this system.

The proposed system consists of two stakeholders. They are:

1. Administrator
2. Customer

2.3.1 Advantages of the Proposed System

Here are some of the advantages of the proposed system:

- The system avoids redundancy by the use of several type of validation that is the system is enhanced
- Quick access and processing are the main advantage that forces as to implement the proposed system.
- The main alteration between the existing system and the new automated system lies in the specialty which reduces the time consumption in an appropriate manner.
- The customer can check new paintings which have been uploaded by the admin on the application.
- Our proposed system promote a person to purchase paintings.
- The system will reduce the amount of paper work require.
- It handles the Painting stock details.
- It saves human efforts and resources.
- Easy to handle all functions.
- Save time.
- User friendly.
- Reduce errors.

2.4 FEASIBILITY STUDY

Online Art Gallery is efficiently handled all the requirements for service providers and service seekers in one system. It allows freeing up wasted time, information is easy to find anytime, anywhere. Use of one system is more secure than using lot of applications. Using this system tasks are difficult to handle the existing system are easily performed.

And for the System to be act as worth-while it should pass through some test that examine that it should proceed further or not. This series of test is commonly known as feasibility study on the system and it plays a very vital role for every system projects. Feasibility studies undergo three major analyses to predict the system to be success and they are as follows: -

- Technical Feasibility
- Operational Feasibility
- Economic Feasibility

2.4.1 Technical Feasibility

We can strongly agree that the website Online Art Gallery is designed in such a way that it is an online application system for finding different types of paintings in easy way. It is technically feasible because the user does not need any additional technical knowledge in order to interacting with the system. The user can view the details according to the role. The administrator can see their Stock availability, view customer details. This web application is technically feasible because it satisfying all the basic requirements. Angular and Nodejs are used for the development of this system. The basic requirement to use the application is a computer with internet connection. The application is technically feasible because all the basic requirements are available.

Recommending the Hardware Part:-

Table 2.1 Hardware parts

Slno.	Hardware used	Specification
1.	Monitor	LCD 15" screen (HP)
2.	Keyboard	Intex Wired
3.	Mouse	Intex Wired
4.	Hard drive	40GB (gigabyte) hard drive
5.	Ram	512 MB (megabyte)
6.	Processor	Pentium 3,665MHZ (megahertz)
7.	Graphics:	On board graphics card, 8MB (Megabyte of memory)
8.	System type	1GHZ (gigahertz) 32-bit (x86)

2.4.2 Operational Feasibility

By providing the basic knowledge about the system, the users (Admin, Customer) in the project can efficiently operate the system. Every user can login to their home page by just providing the credentials. And they are provided with a user-friendly interface in order to view the details according to their roles. Operational feasibility determines whether the system is operating effectively once it is developed. It ensures that the management should support the proposed system and its working feasible in the current organizational environment. The stakeholders or users have a basic knowledge to operate the system. The application is operationally feasible because all the users are able to work with the application without any training.

2.4.3 Economic Feasibility

Economic analysis is the most frequently used technique for evaluating the effectiveness of a proposed system. More commonly known as Cost / Benefit analysis, the procedure is to determine the benefits and savings that are expected from a proposed system and compare them with costs.

Economic feasibility determines whether the required software is capable of generating financial gains for an organization. It involves the cost incurred on the software development team, estimated cost of hardware and software, cost of performing feasibility study, and so on. For this, it is essential to consider expenses made on purchases (such as hardware purchase) and activities required to carry out software development. In addition, it is necessary to consider the benefits that can be achieved by developing the software. Software is said to be economically feasible if it focuses on the issues listed below.

- Cost incurred on software development to produce long-term gains for an organization.
- Cost required to conduct full software investigation (such as requirements elicitation and requirements analysis).
- Cost of hardware, software, development team, and training.

It is estimated that my project is economically feasible because it is a student project and also we are included only a less number of data bases, which result in reduced maintenance and hosting charges for the software. The overall maintenance charge is comparatively low while comparing to the expenditure of the existing system so we can proudly say that my system is economically feasible.

The project is developed as an academic project so there is no need to pay extra amount to the developers. The colleges have enough basic requirements to install the system. The additional charge is only for hosting so the application is economically feasible.

2.5 SOFTWARE ENGINEERING PARADIGM APPLIED

One of the basic notions of the software development process is SDLC models which stand for Software Development Life Cycle models. SDLC – is a continuous process, which starts from the moment, when it's made a decision to launch the project, and it ends at the moment of its full removal from the exploitation. Software development life-cycle (SDLC) is a framework that defines the steps involved in the development of software. It covers the detailed plan for building, deploying, and maintaining the software. SDLC defines the complete cycle of development i.e., all the tasks involved in gathering a requirement for the maintenance of a Product. Some of the common SDLC models are Waterfall Model, V- Shaped Model, Prototype Model, Spiral Model, Iterative Incremental Model, Big Bang Model, and Agile Model. We used

Agile Model for our Project. Agile Model is a combination of the Iterative and incremental model. This model focuses more on flexibility while developing a product rather than on the requirement. In the agile methodology after every development iteration, the client is able to see the result and understand if he is satisfied with it or he is not. Extreme programming is one of the practical uses of the agile model. The basis of this model consists of short meetings where we can review our project. In Agile, a product is broken into small incremental builds. It is not developed as a complete product in one go. At the end of each sprint the project guide verifies the product and after his approval, it is finalized. Client feedback is taken for improvement and his suggestions and enhancement are worked on in the next sprint. Testing is done in each sprint to minimize the risk of any failures.

Phases of Agile Model

are the phases in the Agile model are as follows:

- 1.Requirements gathering
- 2.Design the requirements
- 3.Construction/ iteration
- 4.Testing/ Quality assurance
- 5.Deployment
- 6.Feedback

1.Requirements gathering: In this phase, you must define the requirements. You should explain business opportunities and plan the time and effort needed to build the project. Based on this information, you can evaluate technical and economic feasibility.

2.Design the requirements: When you have identified the project, work with stakeholders to define requirements. You can use the user flow diagram or the high-level UML diagram to show the work of new features and show how it will apply to your existing system.

3.Construction/ iteration: When the team defines the requirements, the work begins. Designers and developers start working on their project, which aims to deploy a working product. The product will undergo various stages of improvement, so it includes simple, minimal functionality.

5.Deployment: In this phase, the team issues a product for the user's work environment.

6.Feedback: After releasing the product, the last step is feedback. In this, the team receives feedback about the product and works through the feedback.

Advantages of Agile Model:

- It allows more flexibility to adapt to the changes.
- The new feature can be added easily.
- Customer satisfaction as the feedback and suggestions are taken at every stage.
- Risks are minimized thanks to the flexible change process.

Disadvantages:

- Lack of documentation.
- If a customer is not clear about how exactly they want the product to be, then the project would fail.
- With all the corrections and changes there is possibility that the project will exceed expected time.

CHAPTER 3 SYSTEM DESIGN

3.1 INTRODUCTION

A system architecture or systems architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures of the system. System architecture can comprise system components, the externally visible properties of those components, the relationships (e.g. the behavior) between them. It can provide a plan from which products can be procured, and systems developed, that will work together to implement the overall system. There have been efforts to formalize languages to describe system architecture; collectively these are called architecture description languages (ADLs). The system architecture can best be thought of as a set of representations of an existing(or to be created) system. It is used to convey the informational content of the elements comprising a system, the relationships among those elements, and the rules governing those relationships. The architectural components and set of relationships between these components that architecture describes may consist of hardware, software, documentation, facilities, manual procedures, or roles played by organizations or people. The system architecture is primarily concerned with the internal interfaces among the system's components or subsystems, and the interface between the system and its external environment, especially the user. The structural design reduces complexity, facilitates change, and result in easier implementation byencouraging parallel development of different parts of the system. The procedural design transforms structural elements of program architecture into a procedural description of software components. The architectural design considers architecture as the most important functional requirement. The system is based on the three-tier architecture. The first level is the user interface(presentation logic), which displays controls, receives, and validates user input. The second level is the business layer (business logic) where the application-specific logic takes place. The third level is the data layer where the application information is stored in files or databases. It contains logic about retrieving and updating data. The important feature about the three- tier design is that information only travels from one level to an adjacent level.

3.2 DATABASE DESIGN

A database is a collection of interrelated data stored with minimum redundancy to serve many users quickly and efficiently. The general objective is to make information access easy, quick,

inexpensive and flexible for the users. The general theme behind a database is to integrate all information. Database design is recognized as a standard of management information system and is available virtually for every computer system.

In database design several specific objectives are considered:

- Ease of learning and use
- Controlled redundancy
- Data independence
- More information at low cost
- Accuracy and integrity

A database is an integrated collection of data and provides centralized access to the data. Usually, the centralized data managing the software is called RDBMS. The main significant difference between RDBMS and other DBMS is the separation of data as seen by the program and data has in direct access to stores device. This is the difference between logical and physical data.

Table Structure

Table is a collection of complete details about a particular subject. These data are saved in rows and Columns. Hence, rows are called RECORDS and Columns of each row are called FIELDS. Data is stored in tables, which is available in the back-end.

The items and data, which are entered in the input, form id directly stored in this table using linking of database. We can link more than one table to input forms. We can collect the details from the different tables to display on the output.

There are 08 tables in our project. They are,

- tbl_login
- tbl_district
- tbl_place
- tbl_category
- tbl_artwork
- tbl_user
- tbl_booking
- tbl_payment

Table No: 1**Table Name:** tbl_login**Description:** This table is used to store login details for identify the user.**Primary Key:** login_id**Foreign key:** Nill

Table 3.1 tbl_login

SI. No	FIELDNAME	DATATYPE	DESCRIPTION
1.	login_id	int	Used as a primary key to uniquely identify column and it is an auto increment column
2.	username	varchar(20)	This field for storing the state name or sample string
3.	password	varchar(20)	Used for storing password
4.	role	varchar(20)	Used to store role of user

Table No: 2**Table Name:** tbl_district**Description:** This table stores the District Details.**Primary Key:** district_id**Foreign key:** nill

Table 3.2 tbl_district

Sl. No	FIELD NAME	DATA TYPE	DESCRIPTION
1.	district_id	Int	Used as key column and it is an auto increment column
2.	districtname	varchar(20)	it is used to store the district name as string.

Table No: 3**Table Name:** tbl_place**Description:** This table used to store the Place Details.**Primary Key:** place_id**Foreign key:** district_id

Table 3.3 tbl_place

Sl. No	FIELD NAME	DATATYPE	DESCRIPTION
1.	place_id	Int	Used as key column and it is an auto increment column
2.	placename	varchar(30)	it is used to store the place name as string.
3.	district_id	Int	Used to generate relation between place table and district table

Table No: 4**Table Name:** tbl_category**Description:** This table used to add category.**Primary Key:** category_id**Foreign key:** Nill

Table 3.4 tbl_category

Sl. No	FIELD NAME	DATATYPE	DESCRIPTION
1.	category_id	Int	Used as key column and it is an auto increment column
2.	categoryname	varchar(30)	This field for storing the category name orsamplestring
3.	categoryimage	varchar	This field is used to store category image

Table No: 5**Table Name:** tbl_artwork**Description:** This table is used to store the artwork details.**Primary Key:** art_id**Foreign key:** category_id

Table 3.5 tbl_artwork

Sl. No	FIELD NAME	DATATYPE	DESCRIPTION
1.	artwork_id	int	Used as key column and it is n auto increment column
2.	artname	varchar(30)	This field for storing the art name or sample string.
3.	artrate	varchar(30)	It is used to store the art price.
4.	artimage	varchar(30)	It is used to store the art image.
5.	category_id	int	It is used to store the category id as foreign key

Table No: 6

Table Name: tbl_user

Description: This table is used to store the User details.

Primary Key: user_id **Foreign key:** place_id

Table 3.6 tbl_user

SI. No	FIELD NAME	DATA TYPE	DESCRIPTION
1.	user_id	Int	Used as a primary key to uniquely identify column and it is an auto increment column.
2.	fullname	varchar(30)	This field for storing the full name or sample string.
3.	contact	Int	This field is used to store the contact details
4.	email	Varchar(30)	This field is used for storing email id of the user.
5.	housetname	Varchar(30)	It is used to store the house name of the user.
6.	gender	Varchar(30)	It is used to store the gender of the user.
7.	Place_id	int	Used to store the relation between this table and place table.
8.	username	Varchar(30)	Used to store username.
9.	password	Varchar(30)	Used to store password.

Table No: 7**Table Name:** tbl_booking**Description:** This table used to store the booking details of art.**Primary Key:** booking_id

Foreign key: user_id, art_id

Table 3.7 tbl_booking

Sl. No	FIELD NAME	DATATYPE	DESCRIPTION
1.	booking_id	Int	Used as key column and it is an auto increment column.
2.	user_id	Int	Used to generating relation between user table and this table.
3.	Art_id	int	Used to generating relation between art table and this table.
4.	quantity	int	Used to store art quantity.
5.	bookingdate	date	Store the booking date of art by user.
6.	requiredate	date	Used to store require date.
7.	totalamount	int	Used to store art total amount of art.
8.	bookingstatus	Varchar(30)	Used to store the status of booking.
9.	remark	Varchar(30)	Used to store remark.

Table No: 8

Table Name: tbl_payment**Description:** This table is used to store the payment details.**Primary Key:** payment_id**Foreign key:** booking_id

Sl. No	FIELDNAME	DATA TYPE	DESCRIPTION
1.	payment_id	Int	Used as key column and it is an auto increment column
2.	paymentdate	date	it is used to store the payment date
3.	amount	int	Used to store the amount of art.
4.	booking_id	int	It is used to store the relationship between booking table and payment table.
5.	paymentstatus	Varchar(30)	Used to store payment status.

Table 3.8 tbl_payment

3.2.1 ENTITY-RELATIONSHIP MODEL

ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system. It develops a conceptual design for the database. It also develops a very simple and easy to design view of data. In ER modelling, the database structure is portrayed as a diagram called an entity relationship diagram. An entity-relationship model (or ER model) describes interrelated things of interest in a specific domain of knowledge. A basic ER model is composed of entity types and

specifies relationships that can exist between entities. In software engineering, an ER model is commonly formed to represent things a business needs to remember in order to perform business processes. Consequently, the ER model becomes an abstract data model, that defines a data or information structure which can be implemented in a database, typically a relational database.

Admin

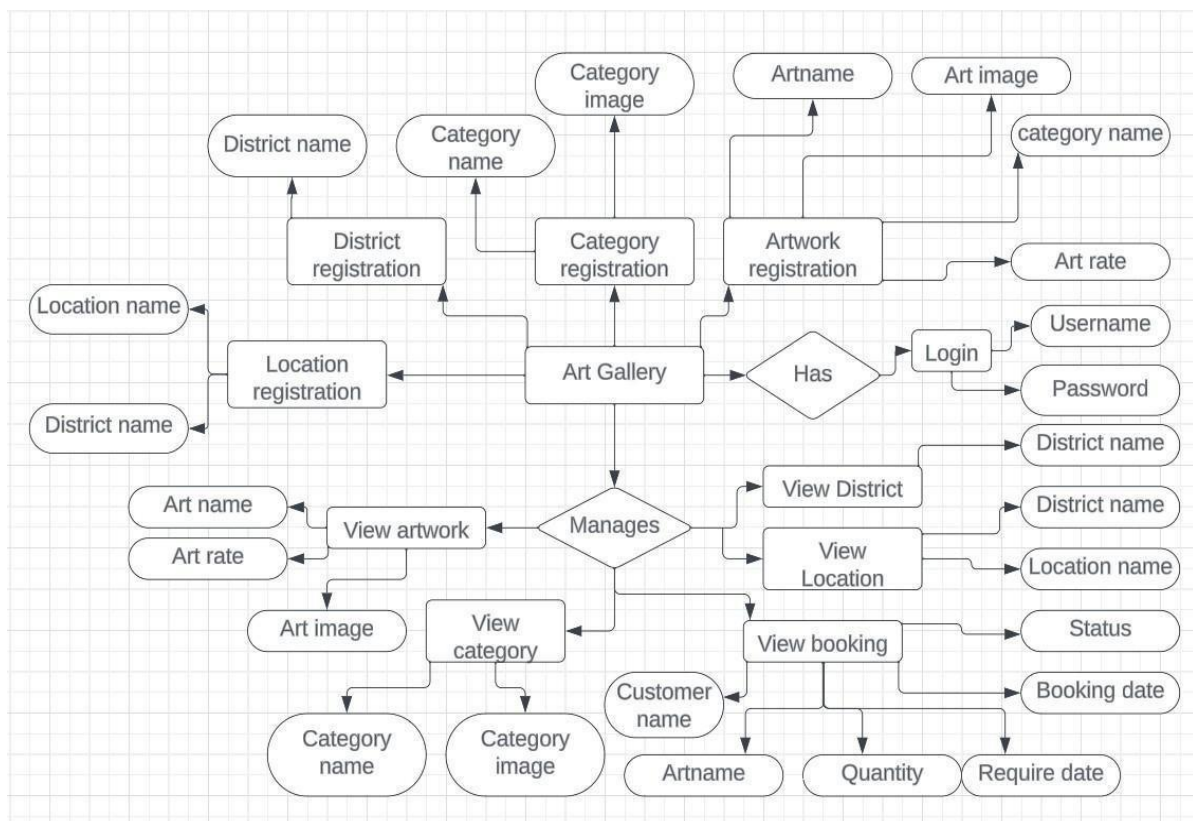


Fig 3.1 ER diagram for Admin

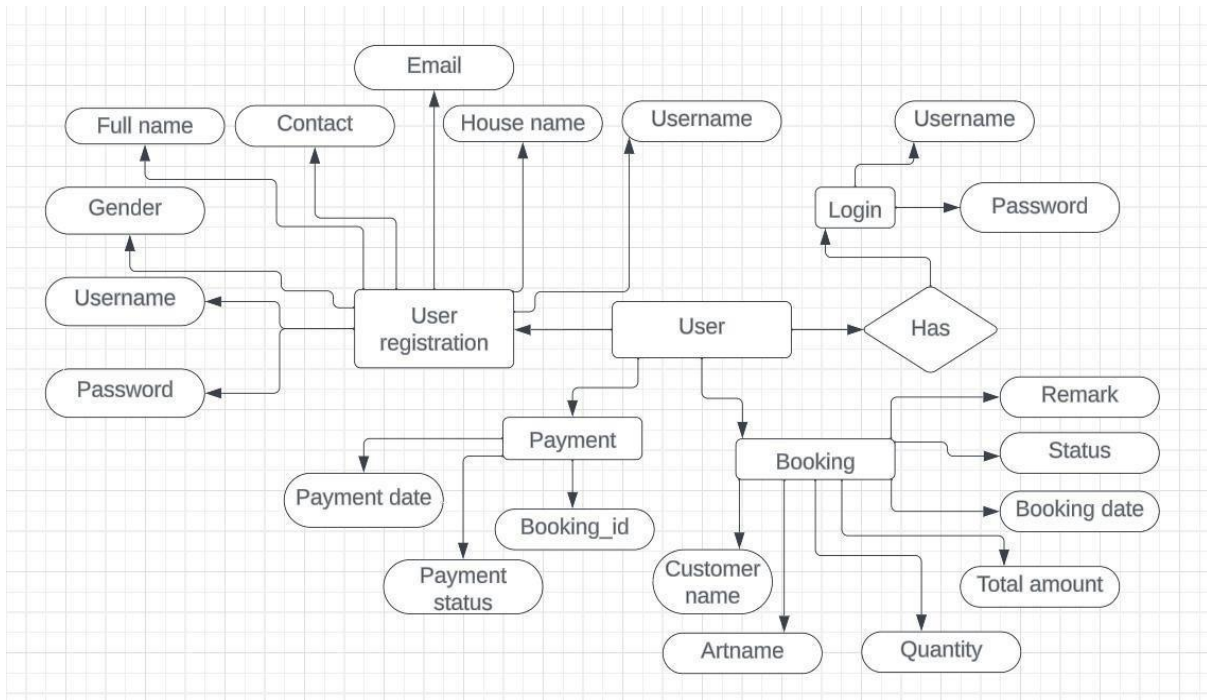
User

Fig 3.2 ER diagram for User

3.3 PROCESS DESIGN –DATAFLOW DIAGRAMS

Data Flow Diagram is a network that describes the flow of data and processes that change, or transforms, data throughout the system. This network is constructed by using a set of symbols that do not imply a physical implementation. It is a graphical tool for structured analysis of the system requirements. DFD models a system by using external entities from which data flows to a process, which transforms the data and creates, output-data-flows which go to other processes or external entities or files. Data in files may also flow to processes as inputs.

There are various symbols used in a DFD. Bubbles represent the processes. Named arrows indicate the data flow. External entities are represented by rectangles. Entities supplying data are known as sources and those that consume data are called sinks. Data are stored in a data store by a process in the system. Each component in a DFD is labeled with a descriptive name. Process names are further identified with a number. The Data Flow Diagram shows the logical flow of a system and defines the boundaries of the system. For a candidate system, it describes the input (source), outputs (destination), database (files), and procedures (data flow), all in a format that meets the user's requirements. The main merit of DFD is that it can provide an overview of system requirements, what data a system would process, what transformations of data are done, what files are used, and where the results flow. This network is constructed by using a set of symbols that do not imply a physical implementation. It is a graphical tool for structured analysis of the system requirements. DFD models a system by using external entities from which data flows to a process, which transforms the data and creates, output- data-flows which go to other processes or external entities or files. External entities are represented by rectangles. Entities supplying data are known as sources and those that consume data are called sinks. Data are stored in a data store by a process in the system. It is a graphical tool for structured analysis of the system requirements. DFD models a system by using external entities from which data flows to a process, which transforms the data and creates, output-data-flows which go to other processes or external entities or files. Data in files may also flow to processes as inputs.

1. Rules for constructing a Data Flow Diagram
2. Arrows should not cross each other
3. Squares, circles and files must bear names.
4. Decomposed data flow squares and circles can have same time
5. Choose meaningful names for data flow

6. Draw all data flows around the outside of the diagram.

Each component in a DFD is labeled with a descriptive name. Process names are further identified with a number. Context level DFD is drawn first. Then the process is decomposed into several elementary levels and is represented in the order of importance. A DFD describes what data flow (logical) rather than how they are processed, so it does not depend on hardware, software, data structure, or file organization. A DFD methodology is quite effective; especially when the required design.

3.4 OBJECT ORIENTED DESIGN – UML DIAGRAMS

3.4.1 Activity Diagram

Activity Diagrams describe how activities are coordinated to provide a service that can be at different levels of abstraction. Typically, an event needs to be achieved by some operations, particularly where the operation is intended to achieve a number of different things that require coordination, or how the events in a single-use case relate to one another, in particular, use cases where activities may overlap and require coordination. It is also suitable for modeling a collection of use cases coordinates to represent business workflows.

Activity diagram of a Admin: -

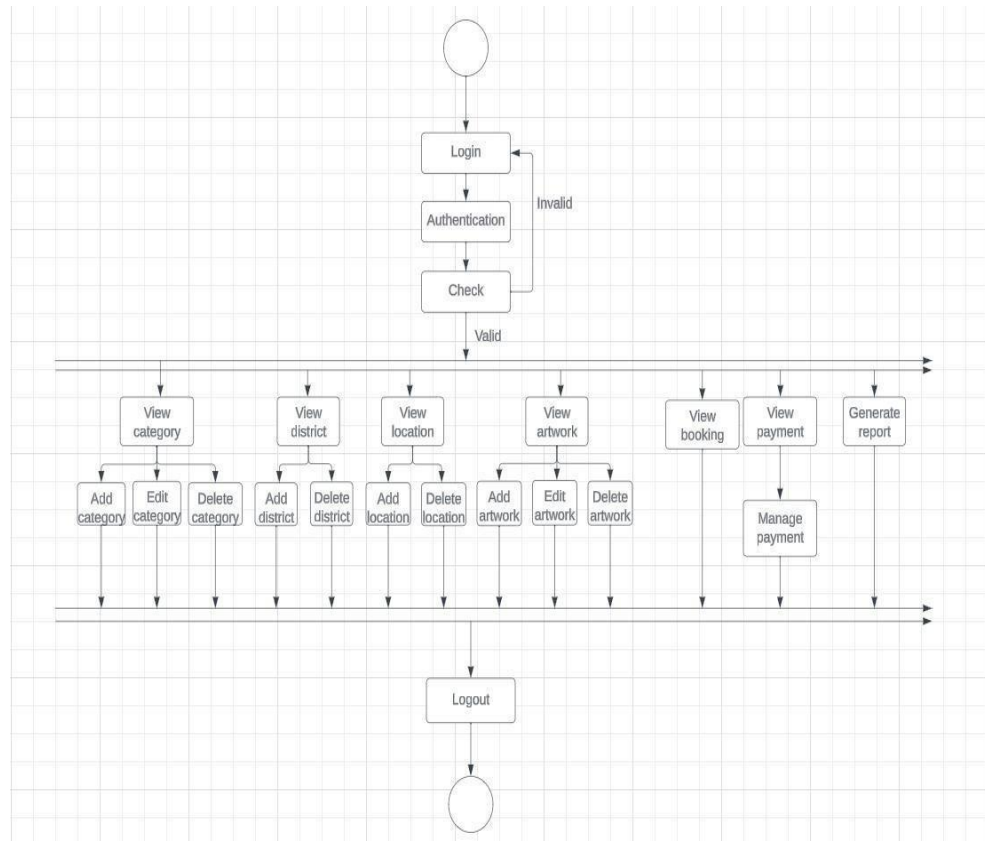


Fig 3.3 Activity Diagram for Admin

Activity diagram of User: -

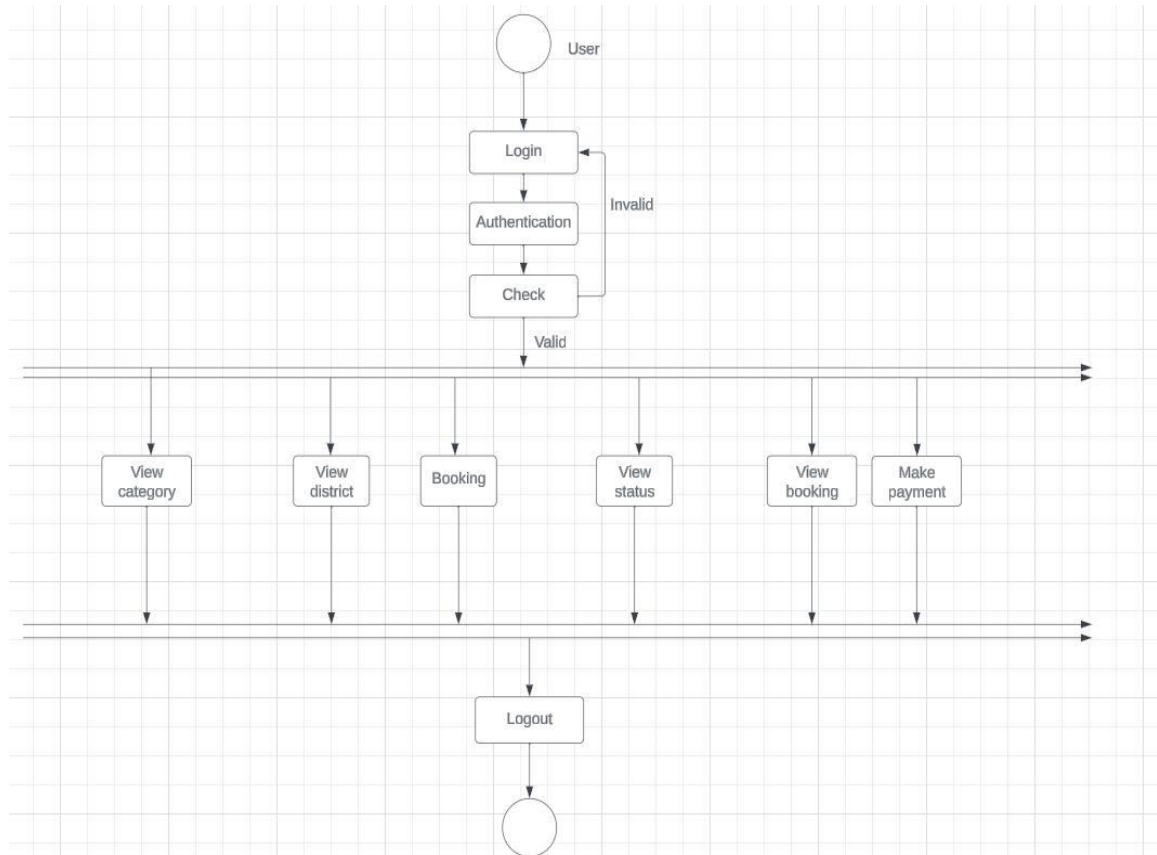


Fig 3.4 Activity Diagram for User

3.4.2 Sequence Diagram

Sequence diagrams, commonly used by developers, model the interactions between objects in a single-use case. They illustrate how the different parts of a system interact with each other to carry out a function, and the order in which the interactions occur when a particular use case is executed. In simpler words, a sequence diagram shows different parts of a system working in a 'sequence' to get something done. The following nodes and edges are typically drawn in a UML sequence diagram: lifeline, execution specification, message, combined fragment, interaction use, state invariant, continuation, and destruction occurrence. Major elements of the sequence diagram are shown below:

Sequence diagram for Admin:

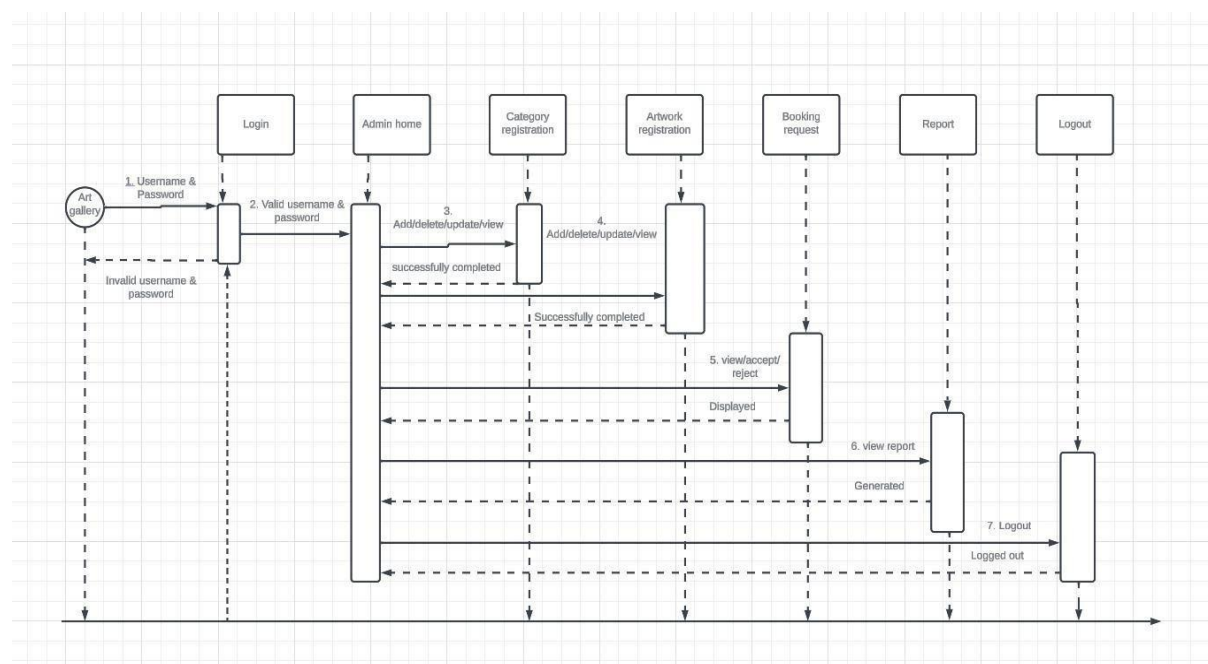


Fig 3.5 Sequence Diagram for Admin

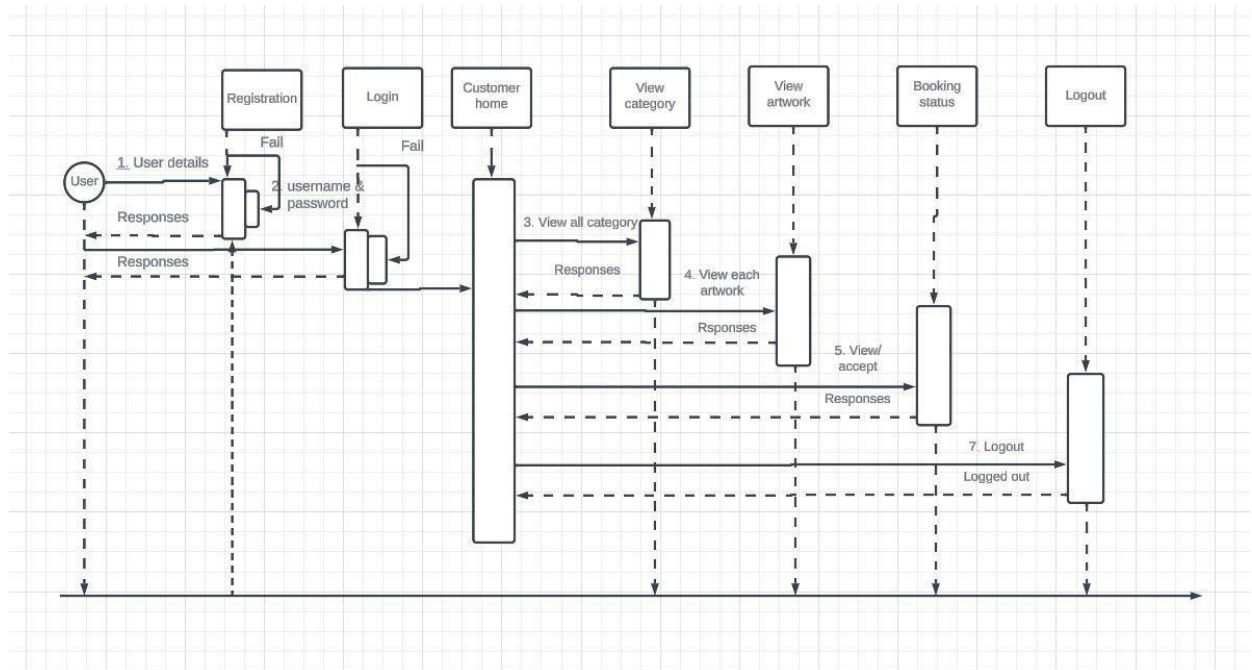
Sequence diagram for User:

Fig 3.6 Sequence Diagram for User

3.4.3 Use Case Diagram

The following shows the overall use case diagram of Online Art Gallery:

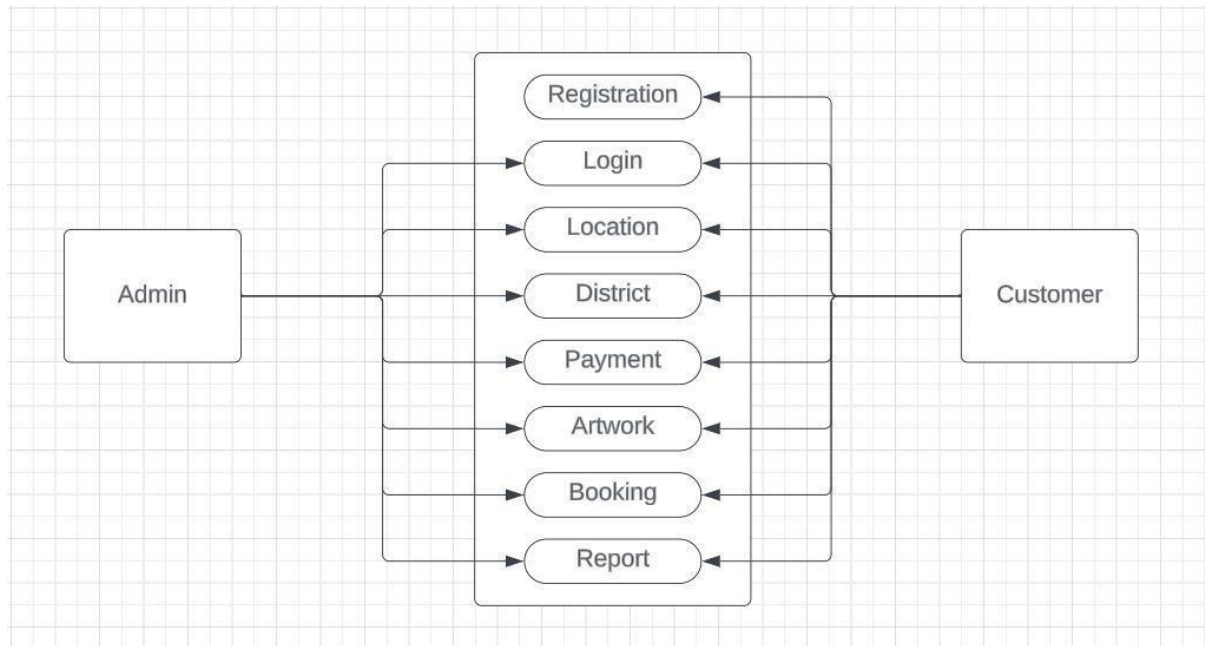


Fig 3.7 Use case Diagram for Online Art Gallery

3.5 INPUT DESIGN

The user interface design is very important for any application. The interface design describes how the software communicates within itself, to the system that is interpreted with it and with humans who use it. The input design is the process of converting the user-oriented inputs into the computer-based format. The data is fed into the system using simple inactive forms. The forms have been supplied with messages so that the user can enter data without facing any difficulty. The data are validated wherever requires in the project. This ensures that only the correct data have been incorporated into the system. The goal of designing input data is to make automation as easy and free from errors as possible. For providing a good input design for the application easy data input and selection features are adopted. The input design requirements such as user-friendliness, consistent format, and interactive dialogue forgiving the right messages and help for the user at right are also considered for development for this project. Input Design is a part of the overall design. The input methods can be broadly classified into batch and online. Internal controls must be established for monitoring the number of inputs and for ensuring that the data are valid. The basic steps involved in input design are:

- Review input requirements.

- Decide how the input data flow will be implemented.
- Decide on the source document.
- Prototype on line input screens.
- Design the input screens.

The quality of the system input determines the quality of the system output. Input specifications describe the manner in which data enter the system for processing. Input design features can ensure the reliability of the system and produce results from accurate data. The input design also determines whether the user can interact efficiently with the system. In this system several forms are used as input screens for collecting data from the users. Forms contain text-box, drop-down, button etc. For entering values text box is used. For single selection drop-down, drop-down is used. For multiple selections, multiple drop-downs are used. Validation checking is done for all mandatory fields. Already Exist Validation is also provided in some forms to avoid repeated entries. If an attempt to reenter same values, an error message will be displayed.

3.6 OUTPUT DESIGN

Quality output is one, which meets the requirements of the end-user and presents the information clearly. In any system results of processing are communicated to the user and to the other system through outputs. In the output design, it is determined how the information is to be displayed for immediate need. It is the most important and direct source of information is to the user. Efficient and intelligent output design improves the system's relationships with the user and helps in decision-making. The objective of the output design is to convey the information of all the past activities, current status and to emphasize important events. The output generally refers to the results and information that is generated from the system. Outputs from computers are required primarily to communicate the results of processing to the users. Output also provides a means of storage by copying the results for later reference in consultation. There is a chance that some of the end-users will not actually operate the input data or information through workstations, but will see the output from the system. Two phases of the output design are:

1. Output Definition
2. Output Specification

Output Definition takes into account the type of output contents, its frequency, and its volume, the appropriate output media is determined for output. Once the media is chosen, the detailed specification of output documents is carried out. The nature of output required from the

proposed system is determined during the logical design stage. It takes the outline of the output from the logical design and produces output as specified during the logical design phase. In a project, when designing the output, the system analyst must accomplish the following:

- Determine the information to present.
- Decide whether to display, print, speak the information and select the output medium.
- Arrange the information in acceptable format.
- Decide how to distribute the output to the intended receipt.

Thus, by following the above specifications, a high-quality output can be generated. In this system, an output or response is given to the user after submission of each form. Output screens are well designed. In most of the form's tables are used to display the output information to the user. Reports values are displayed in a table format.

CHAPTER 4 SYSTEM ENVIRONMENT

4.1 INTRODUCTION

A software development environment (SDE) is the collection of hardware and software tools a system developer uses to build software systems. When you are developing software, you probably don't want your users to see every messy part of your application creation process. In order to make sure you control what people see and when they have access to it, development teams use environments to create "stages" of the app which they consider good for releasing. Each environment has its own unique purpose. There are different standards of environments that are used in the industry, although almost every process starts at the 'development' stage and ends with 'production'. Different organizations all have their own purposes and policies which dictate when and how each environment is used.

4.2 SOFTWARE REQUIREMENT SPECIFICATION

In Online Art Gallery, there are 2 main Stakeholders. They are:

1. Admin
2. User

Stakeholder: Admin

- The proposed site should provide the provision for login by the owner with username and password
- The proposed system Admin should have the provision for add update/delete district, location, category, artwork
- The site should have a home page for Administrator, with the options to accept or reject booking
- The site should have a home page for Admin, with the options to update the category.
- The site should have a home page for owner, with the options to get the view the various categories and related artworks.
- The site should provide the provision to add new category and artworks.
- The site should provide the facility to generate various report.
- The proposed system Admin should have the provision for viewing the feedback from the user.

Stakeholder: User

- The proposed site should provide the provision for login by the user with mail id and password
- The proposed site should provide the provision to view the available location at the time of registration
- The site should have a home page for user, and view available categories.
- The site should have a home page for user, and view available artworks
- The site should provide the facility of view the viewmore page.
- The proposed system user should have the provision for booking.
- The site should provide the facility to view booking status

4.3 HARDWARE REQUIREMENT SPECIFICATION

The selection of hardware configuring is a very task related to the software development, particularly inefficient RAM may affect adversely on the speed and corresponding on the efficiency of the entire system. The processor should be powerful to handle all the operations. The hard disk should have sufficient to solve the database and the application. Hardware used for development:

CPU : Intel Core i7 Processor

Memory : 8 GB

Cache : 6 MB

Hard Disk : 1 TB

Monitor : 15.6" Monitor

Keyboard : Standard PS/2 Keyboard

Mouse : Optical Mouse

CPU : Pentium IV Processor

Memory : 256 MB Above

Cache : 512 KB Above

Hard Disk : 20 GB Above

Monitor : Any

Keyboard : Any

Mouse : Any

4.4 TOOLS, PLATFORMS

This project is built upon the latest technology

software.Frontend : Angular

Back end : Node.js

Database : My SQL

Web Server : WAMP server

Development Tools : Visual StudioCode

Operating System : Windows 10

4.4.1 Front End Tool

4.4.1.1 Angular

Angular is a platform and framework for building single-page client applications using HTML and TypeScript. Angular is written in TypeScript. The architecture of an Angular application relies on certain fundamental concepts. The basic building blocks of the Angular framework are Angular components that are organized into Ng Modules. Ng Modules collect related code into functional sets; an Angular application is defined by a set of Ng Modules. An application always has at least a root module that enables bootstrapping, and typically has many more feature modules.

Modules

An Ng Module declares a compilation context for a set of components that is dedicated to an application domain, a workflow, or a closely related set of capabilities. An Ng Module can associate its components with related code, such as services, to form functional units. Every Angular application has a root module, conventionally named AppModule, which provides the

bootstrap mechanism that launches the application. An application typically contains many functional modules.

Components

Every Angular application has at least one component, the root component that connects a component hierarchy with the page document object model (DOM). Each component defines a class that contains application data and logic and is associated with an HTML template that defines a view to be displayed in a target environment.

4.4.2 Back End Tool

4.4.2.1 Node.js

Node.js tutorial provides basic and advanced concepts of Node.js. Our Node.js tutorial is designed for beginners and professionals both. Node.js is a cross-platform environment and library for running JavaScript applications which is used to create networking and server-side applications. Our Node.js tutorial includes all topics of Node.js such as Node.js installation on windows and Linux, REPL, package manager, callbacks, event loop, os, path, query string, cryptography, debugger, URL, DNS, Net, UDP, process, child processes, buffers, streams, file systems, global objects, web modules etc. There are also given Node.js interview questions to help you better understand the Node.js technology. Node.js is a cross-platform runtime environment and library for running JavaScript applications outside the browser. It is used for creating server-side and networking web applications. It is open source and free to use. Many of the basic modules of Node.js are written in JavaScript. Node.js is mostly used to run real-time server applications.

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices. Node.js also provides a rich library of various JavaScript modules to simplify the development of web applications.

Node.js Features

Extremely fast: Node.js is built on Google Chrome's V8 JavaScript Engine, so its library is very fast in code execution.

I/O is Asynchronous and Event Driven: All APIs of Node.js library is asynchronous i.e., non-blocking. So, a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call. It is also a reason that it is very fast.

Single threaded: Node.js follows a single threaded model with event looping.

Highly Scalable: Node.js is highly scalable because event mechanism helps the server to respond in a non-blocking way.

No buffering: Node.js cuts down the overall processing time while uploading audio and video files. Node.js applications never buffer any data. These applications simply output the data in chunks.

Open source: Node.js has an open-source community which has produced many excellent modules to add additional capabilities to Node.js applications.

License: Node.js is released under the MIT license.

4.4.2.2 MySQL

MySQL is the world's most popular open-source database software, with over 100 million copies of its software downloaded or distributed throughout its history. With its superior speed, reliability, and ease of use, MySQL has become the preferred choice for Web, Web 2.0, SaaS, ISV, Telecom companies and forward-thinking corporate IT Managers because it eliminates the major problems associated with downtime, maintenance and administration for modern, online applications. Many of the world's largest and fastest-growing organizations use MySQL to save time and money powering their high-volume Web sites, critical business systems, and packaged software including industry leaders such as Yahoo!, Alcatel-Lucent, Google, Nokia, YouTube, Wikipedia, and Booking.com. The flagship MySQL offering is MySQL Enterprise, a comprehensive set of production-tested software, proactive monitoring tools, and premium support services available in an affordable annual subscription. MySQL is a key part of LAMP (Linux, Apache, MySQL, PHP / Perl / Python), the fast-growing open-source enterprise software stack. More and more companies are using.

LAMP as an alternative to expensive proprietary software stacks because of its lower cost and freedom from platform lock-in. MySQL was originally founded and developed in Sweden by two Swedes and a Finn David Axmark, Allan Larsson and Michael "Monty" Widenius, who

had worked together since the 1980's. MySQL, the most popular Open- Source SQL database management system, is developed, distributed, and supported by Oracle Corporation. MySQL is a database management system. A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access, and process data stored in a computer database, you need a database management system such as MySQL Server. Since computers are very good at handling large amounts of data, database management systems play a central role in computing, as standalone utilities, or as parts of other applications. MySQL databases are relational. A relational database stores data in separate tables rather than putting all the data in one big storeroom. The database structures are organized into physical files optimized for speed. The logical model, with objects such as databases, tables, views, rows, and columns, offers a flexible programming environment. You set up rules governing the relationships between different data fields, such as one-to-one, one-to-many, unique, required or optional, and pointers between different tables. The database enforces these rules, so that with a well-designed database, your application never sees inconsistent, duplicate, orphan, out-of-date, or missing data. The SQL part of MySQL stands for Structured Query Language. SQL is the most common standardized language used to access databases. Depending on your programming environment, you might enter SQL directly (for example, to generate reports), embed SQL statements into code written in another language, or use a language-specific API that hides the SQL syntax. SQL is defined by the ANSI/ISO SQL Standard. The SQL standard has been evolving since 1986 and several versions exist. In this manual, SQL-92 refers to the standard released in 1992, SQL:1999 refers to the standard released in 1999, and SQL:2003 refers to the current version of the standard. We use the phrase the SQL standard to mean the current version of the SQL Standard at any time. MySQL software is Open Source. Open Source means that it is possible for anyone to use and modify the software. Anybody can download the MySQL software from the Internet and use it without paying anything. If you wish, you may study the source code and change it to suit your needs. The MySQL software uses the GPL (GNU General Public License), to define what you may and may not do with the software in different situations. If you feel uncomfortable with the GPL need to embed MySQL code into a commercial application, you can buy a commercially licensed version from us. The MySQL Database Server is very fast, reliable, scalable, and easy to use. If that is what you are looking for, you should give it a try. MySQL Server can run comfortably on a desktop or laptop, alongside your other applications, web servers, and so on, requiring little or no attention. If you dedicate an entire machine to MySQL, you can adjust the settings to take advantage of all the

memory, CPU power, and I/O capacity available. MySQL can also scale up to clusters of machines, networked together. MySQL Server was originally developed to handle large databases much faster than existing solutions and has been successfully used in highly demanding production environments for several years. Although under constant development, MySQL Server today offers a rich and useful set of functions. Its connectivity, speed, and security make MySQL Server highly suited for accessing databases on the Internet. MySQL Server works in client/server or embedded systems. The MySQL Database Software is a client/server system that consists of a multithreaded SQL server that supports different backends, several different client programs and libraries, administrative tools, and a wide range of application programming interfaces (APIs). We also provide MySQL Server as an embedded multithreaded library that you can link into your application to get a smaller, faster, easier-to-manage standalone product. A large amount of contributed MySQL software is available. MySQL Server has a practical set of features developed in close cooperation with our users. It is very likely that your favorite application or language supports the MySQL Database Server.

4.4.3 Visual Studio Code

Visual Studio Code, also commonly referred to as VS Code, is a source-code editor made by Microsoft with the Electron Framework, for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality. In the Stack Overflow 2021 Developer Survey, Visual Studio Code was ranked the most popular developer environment tool among 82,000 respondents, with 70% reporting that they use it. Visual studio Code is a source-code editor that can be used with a variety of programming languages, including C, C#, C++, Fortran, Go, Java, JavaScript, Node.js, Python, Rust. It is based on the Electron framework, which is used to develop Node.js web applications that run on the Blink layout engine. Visual Studio Code employs the same editor component (codenamed "Monaco") used in Azure DevOps (formerly called Visual Studio Online and Visual Studio Team Services). Out of the box, Visual Studio Code includes basic support for most common programming languages. This basic support includes syntax highlighting, bracket matching, code folding, and configurable snippets. Visual Studio Code also ships with IntelliSense for JavaScript, TypeScript, JSON, CSS, and HTML, as well as debugging support for Node.js. Support for additional languages can be provided by freely available extensions on the VS Code Marketplace. An orange version of the Visual Studio Code logo for the insider's version of

Visual Studio Code Visual Studio Code Logo Instead of a project system, it allows users to open one or more directories, which can then be saved in workspaces for future reuse. This allows it to operate as a language-agnostic code editor for any language. It supports many programming languages and a set of features that differs per language. Unwanted files and folders can be excluded from the project tree via the settings. Many Visual Studio Code features are not exposed through menus or the user interface but can be accessed via the command palette. Visual Studio Code can be extended via extensions, available through a central repository. This includes additions to the editor and language support. A notable feature is the ability to create extensions that add support for new languages, themes, debuggers, time travel debuggers, perform static code analysis, and add code linters using the Language Server Protocol. Source control is a built-in feature of Visual Studio Code. It has a dedicated tab inside of the menu bar where users can access version control settings and view changes made to the current project. To use the feature, Visual Studio Code must be linked to any supported version control system (Git, Apache Subversion, Perforce, etc.). This allows users to create repositories as well as to make push and pull requests directly from the Visual Studio Code program. Visual Studio Code includes multiple extensions for FTP, allowing the software to be used as a free alternative for web development. Code can be synced between the editor and the server, without downloading any extra software. Visual Studio Code allows users to set the code page in which the active document is saved, the newline character, and the programming language of the active document. This allows it to be used on any platform, in any locale, and for any given programming language. Visual Studio Code collects usage data and sends it to Microsoft, although this can be disabled. Due to the open-source nature of the application, the telemetry code is accessible to the public, who can see exactly what is collected.

4.4.4 Operating System

Windows 10 also introduced the Microsoft Edge web browser, a virtual desktop system, a window and desktop management Windows 10 is a series of personal computer operating systems produced by Microsoft as part of its Windows NT family of operating systems. It is the successor to Windows 8.1 and was released to manufacturing on July 15, 2015, and broadly released for retail sale on July 29, 2015. Windows 10 receives new builds on an ongoing basis, which are available at no additional cost to users, in addition to additional testbuilds of Windows 10 which are available to Windows Insiders. Devices in enterprise environments can receive these updates at a slower pace, or use long-term support milestones that only receive critical updates, such as security patches, over their ten-year lifespan of extended

support. One of Windows 10's most notable features is support for universal apps, an expansion of the Metrostyle apps first introduced in Windows 8. Universal apps can be designed to run across multiple Microsoft product families with nearly identical code—including PCs, tablets, smartphones, embedded systems, Xbox One, Surface Hub and Mixed Reality. The Windows user interface was revised to handle transitions between a mouse-oriented interface and a touchscreen- 31 Department of Computer Science optimized interface based on available input devices particularly on 2-in-1 PCs, both interfaces include an updated Start menu which incorporates elements of Windows 7's traditional Start menu with the tiles of a feature called Task View, support for fingerprint and face recognition login, new security features for enterprise environments, and DirectX 12. Windows 10 received mostly positive reviews upon its original release in July 2015. Critics praised Microsoft's decision to provide a desktop-oriented interface in line with previous versions of Windows, contrasting the tablet-oriented approach of 8, although Windows 10's touch- oriented user interface mode was criticized for containing regressions upon the touch- oriented interface of Windows 8. Critics also praised the improvements to Windows 10's bundled software over Windows 8.1, Xbox Live integration, as well as the functionality and capabilities of the Cortana personal assistant and the replacement of Internet Explorer with Edge. However, media outlets have been critical of changes to operating system behaviors,

including mandatory update installation, privacy concerns over data collection performed by the OS for Microsoft and its partners and the adware-like tactics used to promote the operating system on its release. Although Microsoft's goal to have Windows 10 installed on over a billion devices within three years of its release had failed, it still had an estimated usage share of 60% of all the Windows versions on traditional PCs, and thus 47% of traditional PCs were running Windows 10 by September 2019. Across all platforms (PC, mobile, tablet and console), 35% of devices run some kind of Windows, Windows 10 or older.

CHAPTER 5 SYSTEM IMPLEMENTATION

5.1 INTRODUCTION

The “Online Art Gallery” is tested as it starts to move into the implementation phase. Ideally, the system should be completed and fully tested implementation gets underway but unless a package is being installed this seldom happens. Normally what happens is that parts of the system which are required for file set-up are completed first and this process gets underway. Conversion programs may also have to be available which allows data from another system to be used in setting up the files. Once this data is set up it must keep up to date and thus the first use is made of the new system. This may be followed by a period of parallel running and then a decision is made to drop the old system. Implementation involved placing the completed and tested system of hardware and software into the actual work environment of the users. When systems personnel check out and put new equipment into use, train user personnel, install the new application, and construct any files of data needed to use it, we say it is implemented. There are both technical and people-oriented activities during this stage. Examples of technical activities include converting datafiles, replacing old programs with new ones, and scheduling computer operations. Examples of people-oriented activities include orientation, training, and support. Implementation includes all those activities that take place to convert from the old system to the new one. The new system may be totally new, replacing an external system manual or automated system or it may be a modification to an external system. The process of putting the developed system in actual use is called system implementation. This includes all those activities that take place to convert the old system to a new one. The most crucial stages are achieving a new successful system and giving confidence in the new system that it will work efficiently and effectively. The system is implemented only after thorough checking is done and if it is found working according to the specifications.

5.2 CODING

5.2.1 Sample Codes

Here is the sample code of Customer registration:

```
<!DOCTYPE html>

<html>

<head>

</head>

<body>

<div class="container login-container">

<div class="row">

<div class="col-md-6 login-form-2">

<h3 style="text-align: center;"> Customer Registration </h3>

<form [formGroup]='RegisterFormGroup'>

<div class="form-group

<input type="text" formControlName='fullname' class="form-control"

placeholder="Full name">

</div>

<div *ngIf="RegisterFormGroup.controls['fullname'].invalid &&

(RegisterFormGroup.controls['fullname'].dirty ||

RegisterFormGroup.controls['fullname'].touched) || validationstatus">

<div *ngIf="RegisterFormGroup.controls['fullname'].errors['required']"

style="color:red;">PleaseEnter Your Name </div>

<div *ngIf="RegisterFormGroup.controls['fullname'].errors['pattern']" style="color:

red;">Characters only Allowed </div>

</div><br>
```

```
<div class="form-group">

<input type="text" formControlName='contact' class="form-control"placeholder="Contact">

</div>

<div *ngIf="RegisterFormGroup.controls['contact'].invalid    &&
(RegisterFormGroup.controls['contact'].dirty || RegisterFormGroup.controls['contact'].touched)
|| validationstatus">

<div *ngIf="RegisterFormGroup.controls['contact'].errors['required']" style="color:red;">Please
Enter Your Phone Number </div>

<div *ngIf="RegisterFormGroup.controls['contact'].errors['pattern']" style="color:red;">Please
check your format of Contact number </div>

</div>

<br>

<div class="form-group">

<input type="text" formControlName='email' class="form-control"placeholder="Email">

</div>

<div *ngIf="RegisterFormGroup.controls['email'].invalid &&
(RegisterFormGroup.controls['email'].dirty
||RegisterFormGroup.controls['email'].touched) || validationstatus">

<div *ngIf="RegisterFormGroup.controls['email'].errors['required']" style="color: red;">Please
Enter Your Email Address </div>

<div *ngIf="RegisterFormGroup.controls['email'].errors['email']" style="color: red;">Please
check your format of Email </div>

</div>

<br>

<div class="form-group">

<input type="text" formControlName='houasename' class="form-control"placeholder="House
Name">
```

```
</div>

<div *ngIf="RegisterFormGroup.controls['houseName'].invalid &&
(RegisterFormGroup.controls['houseName'].dirty ||
RegisterFormGroup.controls['houseName'].touched) || validationstatus">

<div *ngIf="RegisterFormGroup.controls['houseName'].errors['required']" style="color:
red;">Please Enter Your House Name </div>

<div *ngIf="RegisterFormGroup.controls['houseName'].errors['pattern']" style="color: red;">Characters
only Allowed </div>

</div> <br> <br>

<div class="form-group">Gender

<br>

Male<input type="radio" formControlName='gender' name="gender" value="male"
[checked]="true" />

Female<input type="radio" formControlName='gender' name="gender" value="female" />

</div>

<div *ngIf="RegisterFormGroup.controls['gender'].invalid &&
(RegisterFormGroup.controls['gender'].dirty
||RegisterFormGroup.controls['gender'].touched) || validationstatus">

<div *ngIf="RegisterFormGroup.controls['gender'].errors['required']" style="color:red;">Please
Select the Gender</div>

</div>

<br>

<br>

<div class="form-group">Select your location

<select formControlName='location_id' class="form-control">

<option value="">--Select--</option>
```

```
<option *ngFor="let row of locationdata" [value]="row.location_id">

  {{row.locationname}}

</option>

</select>

</div><br>

<div *ngIf="RegisterFormGroup.controls['location_id'].invalid &&
(RegisterFormGroup.controls['location_id'].dirty ||
RegisterFormGroup.controls['location_id'].touched) || validationstatus">

<div *ngIf="RegisterFormGroup.controls['location_id'].errors['required']"style="color:
red;">Please Enter Your Location</div>

</div> <br>

<div class="form-group">

<input type="text"    formControlName='username'          class="form-
control"placeholder="Username">

</div>

<div *ngIf="RegisterFormGroup.controls['username'].invalid &&
(RegisterFormGroup.controls['username'].dirty ||
RegisterFormGroup.controls['username'].touched) || validationstatus">

<div *ngIf="RegisterFormGroup.controls['username'].errors['required']"
style="color:red;">PleaseEnter Your Location</div>

</div><br>

<div>

<input type="password"    formControlName='password'          class="form-
control"placeholder="YourPassword"></div>

<div *ngIf="RegisterFormGroup.controls['password'].invalid &&
(RegisterFormGroup.controls['password'].dirty
||RegisterFormGroup.controls['password'].touched) || validationstatus">
```

```
<div *ngIf="RegisterFormGroup.controls['password'].errors['required']"
style="color:red;">PleaseEnter Your Password</div>

</div><br><div>

<button type="button" type="submit" (click)="onSubmit()">Register Now</button>

</div>

<br>

</form></div></div></div></body>

<!-- This templates was made by Colorlib (https://colorlib.com) -->

</html>-
```

Here is the screenshot of above display page:

The screenshot displays the 'Customer Registration' form within the ARTGALLERY web application. The header features the ARTGALLERY logo, navigation links (Home, Login, Customer Register), and contact information (+91 9876543210, artgallery@example.com). The form includes input fields for Name (placeholder: e.g John Mathew), Phone Number (placeholder: 1234567890), Email (placeholder: e.g.john@your-domain.com), and House Name. It also has a Gender selection (Male/Female) and a location dropdown (placeholder: --Select--). The bottom section contains Username and Password fields, followed by a 'Register Now' button.

Fig 5.1 Form for Customer Registration

5.2.2 Code Validation & Optimization

Software/Code validation is often considered to be overwhelming for some organizations. With all the requirements and guidance specified in the standards and regulations, it appears to be a monumental task. However, it is not as complex as some may think. This tangled web can be simplified so that it is more easily understood and not just meet regulatory requirements but serve as a useful business tool as well. While very few software systems are developed in-house anymore, many of the systems used are configurable to meet your business needs. The regulations state, that these configured systems must be validated for their intended use. Depending on the risk and complexity of the software, different levels of validation rigor should be performed.

Optimization is a program transformation technique, which tries to improve the code by making it consume fewer resources (i.e. CPU, Memory) and deliver high speed. In optimization, high-level general programming constructs are replaced by very efficient low-level programming codes. A code optimizing process must follow the three rules given below:

1. The output code must not, in any way, change the meaning of the program.
2. Optimization should increase the speed of the program and if possible, the program should demand a smaller number of resources.
3. Optimization should itself be fast and should not delay the overall compiling process.

Admin Login HTML:

```
<form [formGroup]="LoginFormGroup">

<div class="login-wrap">

<div class="login-html">

<input id="tab-1" type="radio" name="tab" class="sign-in" checked><label for="tab-1"
class="tab">Sign In</label>

<input id="tab-2" type="radio" name="tab" class="sign-up"><label for="tab-2"
class="tab">Sign Up</label>

<div class="login-form">

<div class="sign-in-html">
```

```
<div class="group">

<label for="user" class="label">Username</label>

<input id="user" type="text" formControlName="username" class="input">

</div>

<div class="group">

<label for="pass" class="label">Password</label>

<input id="pass" type="password" class="input" formControlName="password" data
type="password">

</div>

<div class="group">

<input id="check" type="checkbox" class="check" checked>

<label for="check"><span class="icon"></span> Keep me Signed in</label>

</div>

<div class="group">

<input type="submit"(click)="Onsubmit()" class="button" value="LOGIN">

</div>

<div class="hr"></div>

<div class="foot-lnk">

<a href="#forgot">Forgot Password?</a>

</div>

</div>

<!-- <div class="sign-up-html">

<div class="group">

<label for="user" class="label">Username</label>

<input id="user" type="text" class="input">
```

```
</div>

<div class="group">

<label for="pass" class="label">Password</label>

<input id="pass" type="password" class="input" data-type="password">

</div>

<div class="group">

<label for="pass" class="label">Repeat Password</label>

<input id="pass" type="password" class="input" data-type="password">

</div>

<div class="group">

<label for="pass" class="label">Email Address</label>

<input id="pass" type="text" class="input">

</div>

<div class="group">

<input type="submit" class="button" value="Sign Up">

</div>

<div class="hr"></div>

<div class="foot-lnk">

<label for="tab-1">Already Member?</a>

</div>

</div>

</div>

</div>

</div>

</form>
```

Admin Login ts:

```
import { Component, OnInit } from '@angular/core'; import { FormBuilder } from
'@angular/forms'; import

{ Router } from '@angular/router';

import { DbuserService } from 'src/app/dbuserService.service';@Component({

selector: 'app-login',

templateUrl: './login.component.html',styleUrls: ['./login.component.scss']

})

export class LoginComponent implements OnInit

{public LoginArray:any[]=[];dbservice: any;

constructor(private fb: FormBuilder, private router: Router,private dbservice:

DbuserService)

{ }

LoginFormGroup = this.fb.group({username: [""],password: [""]

})

Onsubmit(){

//console.log(this.LoginFormGroup.value)

this.dbuserService.login(this.LoginFormGroup.value).then((data:any)=>{this.LoginArray=data;

console.log(data)

if(data==""){

alert('invalid username and password')

this.router.navigate(['/guestmaster/login'])return

}

else{

// console.log(this.LoginArray) varrole = this.LoginArray[0].role;
```

```
// console.log(role)

localStorage.setItem("login_id",this.LoginArray[0].login_id);
localStorage.setItem("username",this.LoginArray[0].username)

var status= this.LoginArray[0].status;

// console.log(status)if(role == "Admin")

{

this.router.navigate(['/Adminmaster/Adminhome'])

}

else

{

this.router.navigate(['/customermaster/customerhome'])

}

}

})

}

// alert('success') ngOnInit(): void {

}

}
```

CHAPTER 6 SYSTEM TESTING

6.1 INTRODUCTION

The objective of system testing is to ensure that all individual programs are working as expected, that the programs link together to meet the requirements specified and to ensure that the computer system and the associated clerical and other procedures work together. The initial phase of system testing is the responsibility of the analyst who determines what conditions are to be tested, generates test data, produced a schedule of expected results, runs the tests, and compares the computer-produced results with the expected results. The analyst may also be involved in procedures testing. When the analyst is satisfied that the system is working properly, he hands it over to the users for testing. The importance of system testing by the user must be stressed. Ultimately it is the user must verify the system and give the go- ahead. During testing, the system is used experimentally to ensure that the software does not fail, i.e., that it will run according to its specifications and in the way, users expect it to. Special test data is input for processing (test plan) and the results are examined to locate unexpected results. A limited number of users may also be allowed to use the system so analysts can see whether they try to use it in unexpected ways. It is preferable to find these surprises before the organization implements the system and depends on it. In many organizations, testing is performed by persons other than those who write the original programs.

6.2 UNIT TESTING

In computer programming, unit testing is a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures are tested to determine if they are fit for use. In this testing, we test each module individually and integrated the overall system. Unit testing focuses verification efforts on the smaller unit of software design in the module. This is also known as module testing. The modules of the system are tested separately. The testing is carried out during the programming stage itself. In this testing step each module is found to work satisfactorily as regards the expected output from the module. There are some validation checks for verifying the data input given by the user which both the formal and validity of the entered. It is very easy to find errors debug the system. We have continued Unit Testing from the starting of the coding phase itself. Whenever we complete done small submodule, some amount

of testing was done based on the requirements to see if the functionality is aligned to the gathered requirements.

6.3 INTEGRATION TESTING

Integration testing (sometimes called integration and testing, abbreviated I&T) is the phase in software testing in which individual software modules are combined and tested as a group. Software components may be integrated in an iterative way or together ("big bang"). Normally the former is considered a better practice since it allows interface issues to be located more quickly and fixed. Data can be lost across an interface; one module can have an adverse effect on the other sub-functions when combined by, may not produce the desired major functions. Integrated testing is the systematic testing for constructing the uncover errors within the interface. This testing was done with sample data. The developed system has run successful for this sample data. The need for an integrated test is to find the overall system performance. Integration testing is a logical extension of unit testing. In its simplest form, two units that have already been tested are combined into a component and the interface between them is tested. A component, in this sense, refers to an integrated aggregate of more than one unit. Integration testing identifies problems that occur when units are combined. By using a test plan that requires you to test each unit and ensure the viability of each before combining units, you know that any errors discovered when combining units are likely related to the interface between units. This method reduces the number of possibilities to a far simpler level of analysis. Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system.

6.4 SYSTEM TESTING

6.4.1 Test Plan & Test Cases

A test case is a set of conditions or variables under which a tester will determine whether a system under test satisfies requirements or works correctly. The process of developing test cases can also help find problems in the requirements or design on an application.

A test plan is a detailed document that describes the test strategy, objectives, schedule, estimation, deliverable, and resources required to perform testing for a software product. Test-plan helps us determine the effort needed to validate the quality of the application under test.

The test plan serves as a blueprint to conduct software testing activities as a defined process, which is minutely monitored and controlled by the test manager.

Table 6.1: test plan

Test Level	Project Team	External Party
Unit Testing	T	
Integration Testing	T	
Validation Testing	T	T
System Testing	T	T

Table 6.2: Admin Login Test Case Report

Function	Description	% Test case executed	% Test passed
Login	Open the application	100%	97%
Login	Open the application	100%	97%
Login	Enter valid username and password	100%	97%
Login	Enter valid username and password	100%	97%

Table 6.3:test case

TEST CASE FOR ONLINE ART GALLERY LOGIN SCREEN			
Test Steps	Expected Result	Actual Result	P a s s / F a i l
Run the application and navigate to loginscreen.	Login screen Contains two fields for entering user name & password and login button should be present.	For entering username & password together with a login button is available.	Pass
Enter a valid username & password and press the login button.	Shop owner must successfully login to the admin dashboard.	Login successful and navigate to cake shop dashboard.	Pass
Enter a valid username & invalid password and press the login button.	A message should be displayed that invalid password.	An asterisk(*) is displayed if no password is given.	Pass
Enter a valid user name and leave password field and press login button.	An asterisk(*) should be displayed that please fill out this field.	A message has been stating that invalid username & password.	Pass
Press login button without entering username & password.	A message should be displayed that required field validation status.	A message has been stating that required field validation status.	Pass
Enter invalid data in username & password field and press login button.	A message should be displayed that regular expression validation status.	A message has been stating that regular expression validation status.	Pass

CHAPTER 7 SYSTEM MAINTENANCE

7.1 INTRODUCTION

Software Maintenance is the process of modifying a software product after it has been delivered to the customer. The main purpose of software maintenance is to modify and update software applications after delivery to correct faults and to improve performance.

Need for Maintenance Software Maintenance must be performed in order to:

- Correct faults.
- Improve the design.
- Implement enhancements.
- Interface with other systems.
- Accommodate programs so that different hardware, software, system features, and telecommunications facilities can be used.
- Migrate legacy software.
- Retire software.

7.2 MAINTENANCE

The definition of software maintenance can be given by describing four activities that are undertaken after the program is released for use. The first maintenance activity occurs since it is unreasonable to assume that software testing will uncover all errors in a large software system. The process of including the diagnosis and correction of one or more errors is called corrective maintenance. The second activity that contributes to a definition of maintenance occurs since rapid change is encountered in every aspect of computing. Therefore, adaptive maintenance modifies to properly interface with a changing environment . The third activity involves recommendations for new capabilities, modification to the exiting function, and general enhancement when the software is used. To satisfy requests, prefecture maintenance is performed. The fourth maintenance activity occurs when software is changed to improve future maintainability or reliability. This is called preventive maintenance.

CHAPTER 8 FUTURE ENHANCEMENT AND FURTHER DEVELOPMENT

8.1 INTRODUCTION

Any system which has been in use for a number of years gradually decays and becomes less effective because of changes in the environment to which it has to be adapted. For the time being it is possible to overcome problems by amendments and minor modifications to acknowledge the need for fundamental changes.

8.2 MERITS OF THE SYSTEM

The merits of Online Art Gallery are as follows:

- It is a very flexible and user-friendly system.
- It reduces the effort of maintaining records in this system.
- It reduces time and human effort which exists in the system.
- The backup facility is available in this proposed system. Hence it prevents data loss.
- It is easy to maintain records.
- It provides 24/7 service availability.
- Users can easily search and Request their desired artwork more efficiently.

8.3 LIMITATION OF THE SYSTEM

Our Proposed system is for Single Art Gallery, it is not possible implement in multiple Art Gallery if it is meant for multiple Art Gallery this website can make huge profit and beneficial for users's of the system and increases the availability of artworks more efficiently.

8.4 FUTURE ENHANCEMENT OF THE SYSTEM

The future enhancement overcomes the limitations of the proposed system. The “Online Art Gallery” can be enhanced in a way such that in future by giving customization of finding the

artwork by chat bot. In future the proposed system can implement for multiple artworks. Chatbot system enable to interact with the art Gallery via online. So, they could get 24/7 interactive support and clarify their inquiries and feedback.

CHAPTER 9 CONCLUSION

The project “Online Art Gallery” is a project which aims in developing a computerized system to maintain all the actions of a Art Gallery. All the requirements’ specifications were followed as for as possible and few additional features were added that can make the application more user friendly and less complicated. The project was successfully completed within the time span allotted. All the modules are tested separately and put together to form the main system. Finally, the system is tested with real data and it worked successfully. Thus, the system has fulfilled the entire objective defined.

CHAPTER 10 BIBLIOGRAPHY

- [1] Waman S Jawadekar – —Software Engineering principles & Practicel- 2nd EditionTata,Mc- GrawHilll Publishing Co. Ltd.
- [2] <https://www.wisdomjobs.com/e-university/uml-tutorial-175/domain-model-13285.html>
- [3] [https://en.wikipedia.org/wiki/C_Sharp_\(programming_language\)](https://en.wikipedia.org/wiki/C_Sharp_(programming_language))
- [4] Grady Booch, James Rumbaugh, Ivar Jacobson, —The Unified ModelingLanguage UserGuidel, 2nd Edition, Pearson Education, 2007.
- [5] <https://online.visual-paradigm.com/app/diagrams/>

CHAPTER 11 APPENDIX (CODING & SCREENSHOTS)

11.1 CODING

Art Work-registration.component.html

```
<div class="container">

<div class="row"> <div class="col-md-6">

<br>

<br>

<br>

<h3 style="text-align: center;">Art Work</h3>

<form [formGroup]='artworkregistration'>

<div class="form-group">

<input type="text" formControlName="artname" class="form-control" placeholder="Enter
theArtName" />

</div><br>

<div class="form-group">

<input type="text" formControlName="artrate" class="form-control" placeholder="Enter
theArtRate" />

</div><br>

<div class="field">

<label class="label_field">Image</label>

<input type="file"

formControlName="artimage" (change)="selectFile($event)"placeholder="Image" />

</div>

<div class="field">
```

```
<label class="label_field">ChooseCategory</label>

<select class="custom-select" FormControlName="category_id">

<!-- <option selected>Choose Category</option> -->

<option*ngFor="let row of categorydata"
value="{{row.category_id}}">{{row.categoryname}}</option>

</select>

</div>

<div class="form-group">

<br>

<button type="submit" (click)="Onsubmit()" class="btn btn-danger" rounded="true">Save
Data</button>

</div>

</form>
```

Art Work-registration.component.ts

```
import { Component, OnInit } from '@angular/core';

import { FormBuilder } from '@angular/forms';

import { DbuserService } from 'src/app/dbuserService.service';

import { Observable } from 'rxjs';

import { Router } from '@angular/router';

@Component({

selector: 'app-artwork',

templateUrl: './artwork.component.html',

styleUrls: ['./artwork.component.scss']

})

export class ArtworkComponent implements OnInit
```



```
{public categorydata:any[]=[];

selectedFiles?: FileList;

currentFile?: any;

message= "";

fileInfos?: Observable<any>;http: any;

constructor(private fb:FormBuilder,private dbservice:DbuserService,private router:Router){

}

artworkregistration=this.fb.group( {artname:[""],artrate:[""],

artimage:[""],

category_id:[""]

})

ngOnInit(): void

{ this.dbservice.viewcategory().then((data:any)=>{this.categorydata=data;console.log(data)})

}

selectFile(event: any): void

{ this.selectedFiles= event.target.files;

}

Onsubmit(){

if(this.selectedFiles) {

// console.log(this.selectedFiles);

const file: File | null = this.selectedFiles.item(0);if (file)

{

this.currentFile = file; this.dbservice.upload(this.currentFile).subscribe((event: any) => {

this.message = event.body.message; }) ;

}
```

```
}
```

```
this.artworkregistration.value.artimage=this.currentFile.name;
```

dbservice.service.ts

```
import { Injectable } from '@angular/core';
```

```
import { HttpClient, HttpEvent, HttpRequest } from '@angular/common/http'; import {  
Observable } from 'rxjs';
```

```
@Injectable({ providedIn: 'root'
```

```
})
```

```
export class DbuserService {
```

```
  booking(value: any) {
```

```
    throw new Error('Method not implemented.');
```

```
  }
```

```
  constructor(private http: HttpClient) { }
```

```
  //District Register insertdistrictdata(data: any) {
```

```
    this.http.post("http://localhost:3000/districtregister", data).toPromise()
```

```
  }
```

```
  //View District ViewDistrictDetails() {
```

```
    return this.http.get("http://localhost:3000/viewdistrict").toPromise()
```

```
  }
```

```
  getdistrictbaseid(data: any) {
```

```
    // console.log(districtid);
```

```
    return this.http.post<any>("http://localhost:3000/getdistrictdetails", data).toPromise()
```

```
  }
```

```
  //Edit District updatedistrictdata(districtid: any) {
```

```
    return this.http.post("http://localhost:3000/editdistrictdetails", districtid).toPromise()
```

//Location Register

```
insertlocationtdata(data: any) {  
  
this.http.post("http://localhost:3000/locationregister", data).toPromise().then(result =>{  
  
})  
  
}
```

//View Location ViewLocationDetails() {

```
return this.http.get("http://localhost:3000/viewlocation").toPromise()  
  
}
```

//View Categoryviewcategory() {

```
return this.http.get("http://localhost:3000/viewcategory").toPromise()  
  
}
```

//View Artworkviewartwork() {

```
return this.http.get("http://localhost:3000/viewartwork").toPromise()  
  
}
```

//Customerartview viewartworkdetails(data:any) {

```
return this.http.post("http://localhost:3000/customerartview", data).toPromise()  
  
}
```

//Viewcustomer viewmorepageviewartmoredetails(data:any) {

```
return this.http.post("http://localhost:3000/viewmore", data).toPromise()  
  
}
```

//Delete Didtrict Deletedistrictdata(data: any){

```
return this.http.post("http://localhost:3000/Deleteddistrict",data).toPromise()  
  
}
```

//Delete Location

```
Deletelocationdata(data: any){  
  
return this.http.post("http://localhost:3000/Deletelocation",data).toPromise()  
  
}  
  
//Delete Category  
  
Deletecategorydata(data: any){  
  
return this.http.post("http://localhost:3000/Deletecategory",data).toPromise()  
  
}  
  
//Delete Artwork  
  
Deleteartworkdata(data: any){  
  
return this.http.post("http://localhost:3000/deleteartwork",data).toPromise()  
  
}  
  
//Login  
  
login(data:any) {  
  
return this.http.post("http://localhost:3000/login", data).toPromise()  
  
}  
  
//View Customer  
  
viewcustomerdata(data: any){  
  
return this.http.post("http://localhost:3000/customerdetails", data).toPromise()  
  
}  
  
//View Artwork  
  
viewartdata(data: any){  
  
return this.http.post("http://localhost:3000/artdetails", data).toPromise()  
  
}  
  
upload(file: File): Observable<HttpEvent<any>>
```

```
{const formData: FormData = new FormData();

formData.append('file', file);

const req = new HttpRequest('POST', `http://localhost:3000/upload`,
formData,

{ reportProgress: true,responseType: 'json'

});

return this.http.request(req);

}

//Cateogory Register

insertcategorydata(data: any) {

return this.http.post("http://localhost:3000/categoryregister", data).toPromise()

}

//Artwork Register

insertartworkdata(data: any) {

return this.http.post("http://localhost:3000/artwork", data).toPromise()

}

//Customer Register

insertcustomerdata(data: any) {

this.http.post("http://localhost:3000/customerregister", data).toPromise()

}

//Booking

insertbookingdata(data: any) {

return this.http.post("http://localhost:3000/booking", data).toPromise()

}
```

//Accept Booking

```
updatestatusdetails(booking_id:any) {  
  
return this.http.post("http://localhost:3000/acceptbooking",booking_id).toPromise()  
  
}
```

//Reject Booking

```
updatestatusdetailsreject(booking_id:any) {  
  
return this.http.post("http://localhost:3000/bookingdetails",booking_id).toPromise()  
  
}
```

//Viewbooking

```
ViewBookingDetails() {  
  
return this.http.get("http://localhost:3000/viewbookingdetails").toPromise()  
  
}
```

//Confirmation View

```
viewstatus(data: any){  
  
return this.http.post("http://localhost:3000/viewstatus",data).toPromise()  
  
}
```

//Deletebooking

```
from adminviewrejectdetails(data: any){  
  
return this.http.post("http://localhost:3000/deletebookingview",data).toPromise()  
  
}
```

//View Payment

```
paymentdetailsview(data:any){  
  
return this.http.post("http://localhost:3000/paymentdetails",data).toPromise()  
  
}
```

//Payment

```
payment(data:any){  
  
return this.http.post("http://localhost:3000/payment",data).toPromise()  
  
}
```

//Datewise Report

```
report(data:any){  
  
return this.http.post("http://localhost:3000/report",data).toPromise()  
  
}
```

//Excel Report

```
productview() {  
  
return this.http.get("http://localhost:3000/productview").toPromise();  
  
}}
```

User Registration Form

```
<!DOCTYPE html>  
  
<html>  
  
<head></head>  
  
<body>  
  
<div class="container login-container">  
  
<div class="row">  
  
<div class="col-md-6 login-form-2">  
  
<h3 style=" text-align: center;"> Customer Registration </h3>  
  
<form [formGroup]='RegisterFormGroup'>  
  
<div class="form-group  
  
<input type="text"    formControlName='fullname' class="form-control"  
placeholder="Fullname">
```

```
</div>

<div *ngIf="RegisterFormGroup.controls['fullname'].invalid    &&
(RegisterFormGroup.controls['fullname'].dirty    ||
RegisterFormGroup.controls['fullname'].touched)

|| validationstatus">

<div *ngIf="RegisterFormGroup.controls['fullname'].errors['required']"
style="color:red;">PleaseEnter Your Name </div>

<div *ngIf="RegisterFormGroup.controls['fullname'].errors['pattern']"    style="color:
red;">Characters only Allowed </div>

</div><br>

<div class="form-group">

<input type="text" formControlName='contact' class="form-control" placeholder="Contact">

</div>

<div *ngIf="RegisterFormGroup.controls['contact'].invalid    &&
(RegisterFormGroup.controls['contact'].dirty ||
RegisterFormGroup.controls['contact'].touched) ||validationstatus">

<div *ngIf="RegisterFormGroup.controls['contact'].errors['required']" style="color:red;">Please
Enter Your Phone Number </div>

<div *ngIf="RegisterFormGroup.controls['contact'].errors['pattern']"    style="color:
red;">Pleasecheck your format of Contact number </div>

</div><br>

<div class="form-group">

<input type="text" formControlName='email' class="form-control" placeholder="Email">

</div>

<div *ngIf="RegisterFormGroup.controls['email'].invalid &&
(RegisterFormGroup.controls['email'].dirty ||RegisterFormGroup.controls['email'].touched) ||
validationstatus">
```



```
<div *ngIf="RegisterFormGroup.controls['email'].errors['required']" style="color: red;">Please
Enter Your Email Address </div>
```

```
<div *ngIf="RegisterFormGroup.controls['email'].errors['email']" style="color:
red;">Please check your format of Email </div>
```

```
</div><br>
```

```
<div class="form-group">
```

```
<input type="text" formControlName='houseName' class="form-control" placeholder="House
Name">
```

```
</div>
```

```
div *ngIf="RegisterFormGroup.controls['houseName'].invalid &&
(RegisterFormGroup.controls['houseName'].dirty ||
RegisterFormGroup.controls['houseName'].touched) || validationstatus">
```

```
<div *ngIf="RegisterFormGroup.controls['houseName'].errors['required']" style="color:
red;">Please Enter Your House Name </div>
```

```
<div *ngIf="RegisterFormGroup.controls['houseName'].errors['pattern']" style="color:
red;">Characters only Allowed </div>
```

```
</div><br><br>
```

```
<div class="form-group">Gender<br>
```

```
Male<input type="radio" formControlName='gender' name="gender" value="male"
[checked]="true" />
```

```
Female<input type="radio" formControlName='gender' name="gender" value="female" />
```

```
</div>
```

```
<div *ngIf="RegisterFormGroup.controls['gender'].invalid &&
(RegisterFormGroup.controls['gender'].dirty || RegisterFormGroup.controls['gender'].touched) ||
validationstatus">
```

```
<div *ngIf="RegisterFormGroup.controls['gender'].errors['required']" style="color:
red;">Please Select the Gender</div>
```

```
</div><br><br>

<div class="form-group">Select your location

<select formControlName='location_id' class="form-control">

<option value="">--Select--</option>

<option *ngFor="let row of locationdata" [value]="row.location_id">

  {{row.locationname}}

</option>

</select>

</div><br>

<div *ngIf="RegisterFormGroup.controls['location_id'].invalid &&
(RegisterFormGroup.controls['location_id'].dirty ||
RegisterFormGroup.controls['location_id'].touched) || validationstatus">

<div *ngIf="RegisterFormGroup.controls['location_id'].errors['required']" style="color:
red;">Please Enter Your Location</div>

</div><br>

<div class="form-group">

<input type="text" formControlName='username' class="form-control"
placeholder="Username">

</div>

<div *ngIf="RegisterFormGroup.controls['username'].invalid &&
(RegisterFormGroup.controls['username'].dirty ||
RegisterFormGroup.controls['username'].touched)
|| validationstatus">

<div *ngIf="RegisterFormGroup.controls['username'].errors['required']"
style="color:red;">Please Enter Your Location</div></div><br>

<div>
```

```
<input type="password"      formControlName='password' class="form-control"
placeholder="YourPassword"></div>

<div  *ngIf="RegisterFormGroup.controls['password'].invalid    &&
(RegisterFormGroup.controls['password'].dirty
||RegisterFormGroup.controls['password'].touched) || validationstatus">

<div  *ngIf="RegisterFormGroup.controls['password'].errors['required']"
style="color:red;">PleaseEnter Your Password</div>

</div><br><div>

<button type="button" type="submit" (click)="onSubmit()">Register Now</button>

</div>

<br>

</form></div></div></div></body>

<!-- This templates was made by Colorlib (https://colorlib.com) -->

</html>
```

11.2 SCREENSHOTS

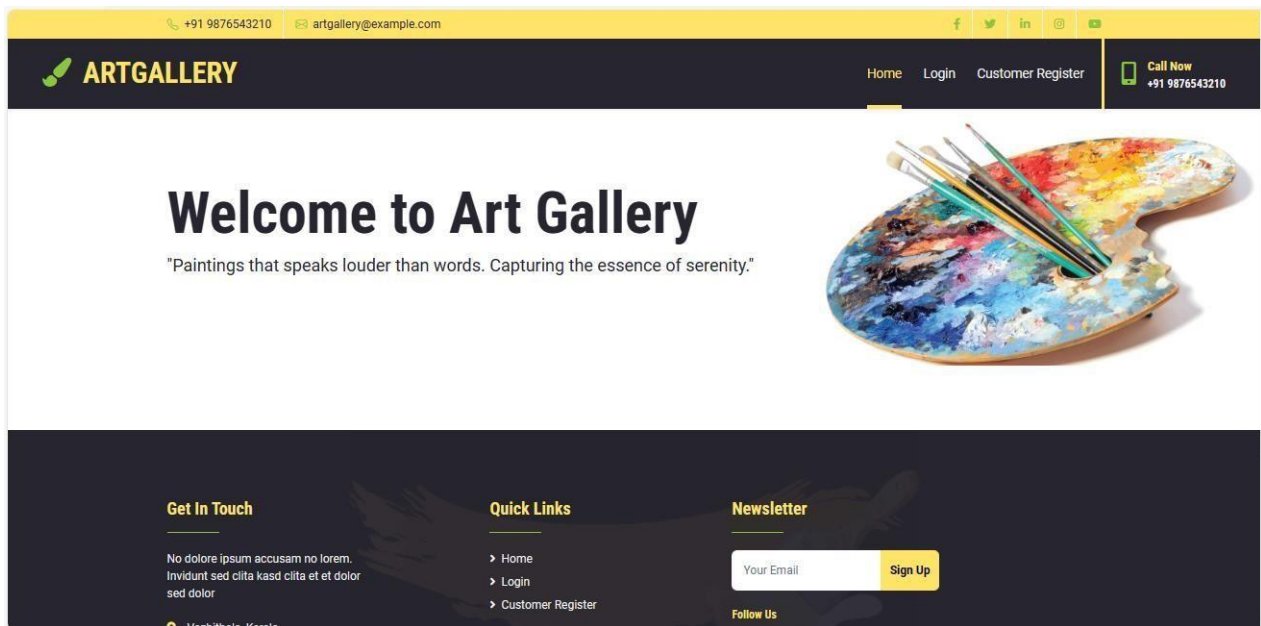


Fig 11.1 Guest Home

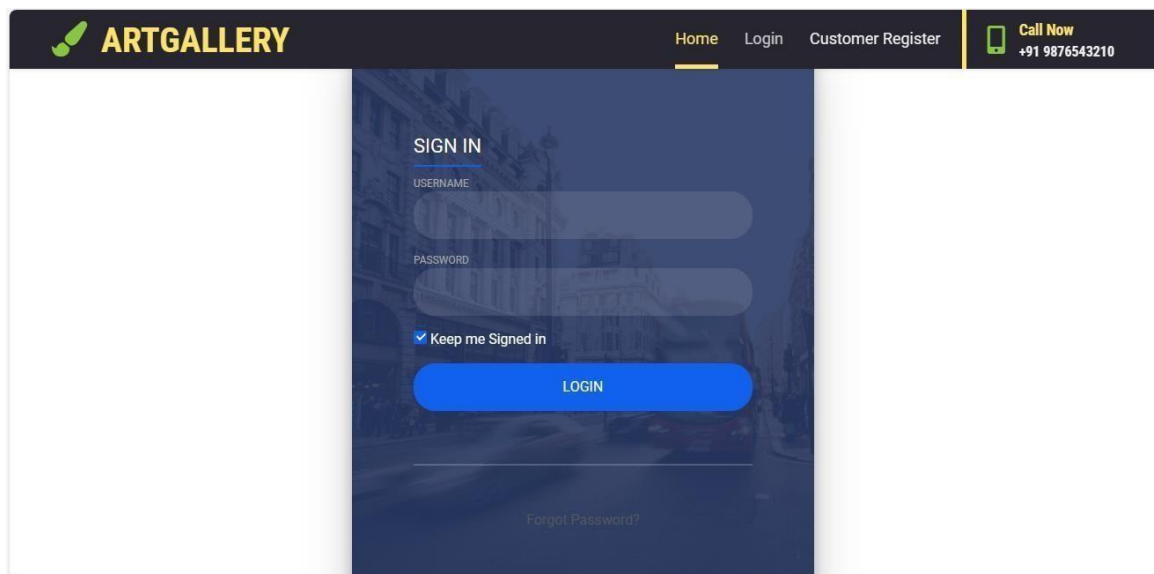


Fig 11.2 Login

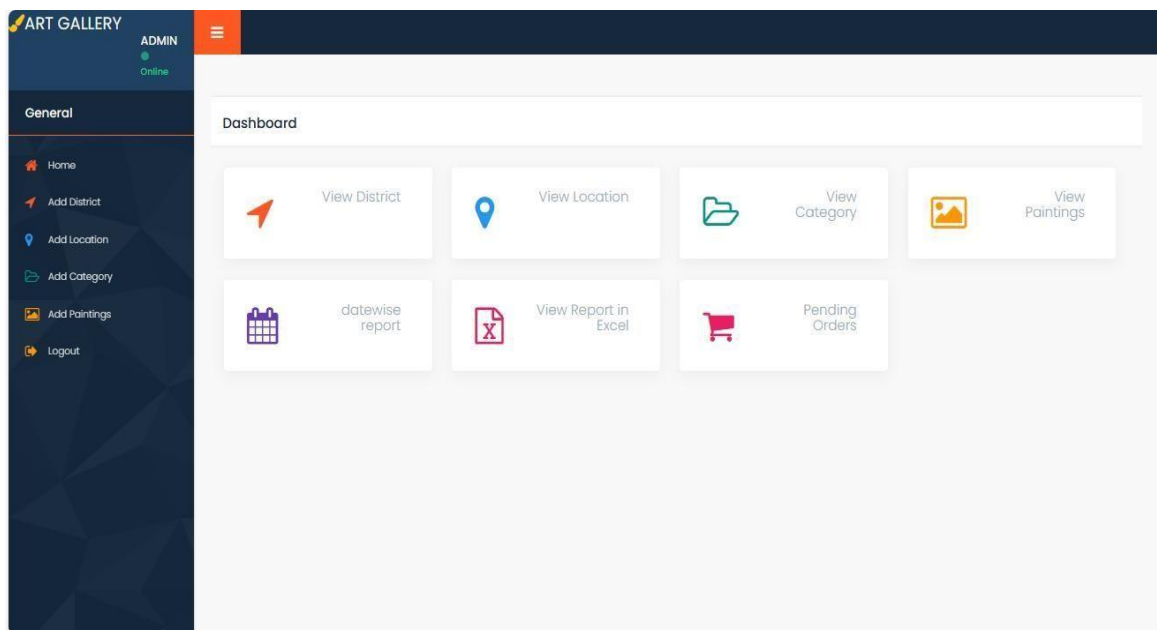


Fig 11.3 Admin Home

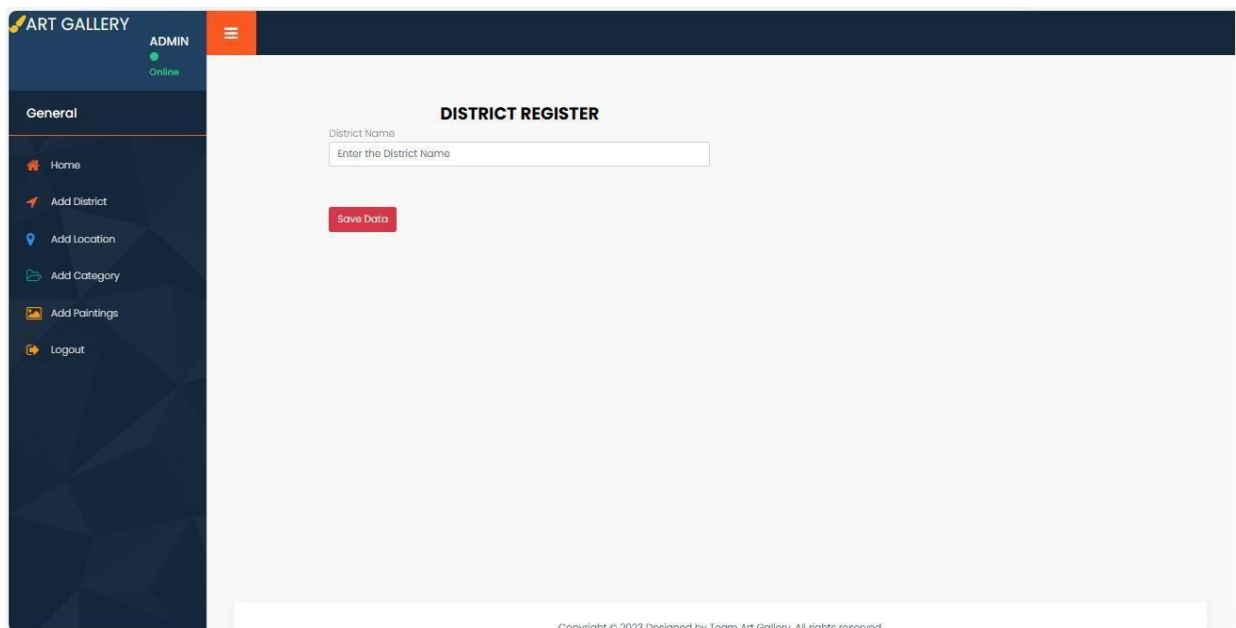


Fig 11.4 District Register

The screenshot shows the 'LOCATION REGISTER' form in the Online Art Gallery admin interface. The left sidebar contains the 'ART GALLERY' logo, 'ADMIN' status, and a menu with 'General', 'Home', 'Add District', 'Add Location', 'Add Category', 'Add Paintings', and 'Logout'. The main content area has a title 'LOCATION REGISTER' and two input fields: 'Select your District' (a dropdown menu) and 'Location' (a text input with placeholder 'Enter the Location Name'). A red 'Save Data' button is positioned below the inputs.

Fig 11.5 Location Register

The screenshot shows the 'Category Register' form in the Online Art Gallery admin interface. The left sidebar is identical to the previous figure. The main content area has a title 'Category Register' and two input fields: 'Categoryname' (a text input with placeholder 'Category Name') and 'Image' (a file upload button labeled 'Choose File' with the text 'No file chosen' next to it). A green 'Submit' button is positioned below the inputs. At the bottom of the form, there is a copyright notice: 'Copyright © 2023 Designed by Team Art Gallery. All rights reserved.'

Fig 11.6 Category Register

The screenshot shows an admin dashboard for an 'ART GALLERY'. The left sidebar contains a 'General' section with links: Home, Add District, Add Location, Add Category, Add Paintings, and Logout. The main content area is titled 'ART WORK' and contains the following form fields:

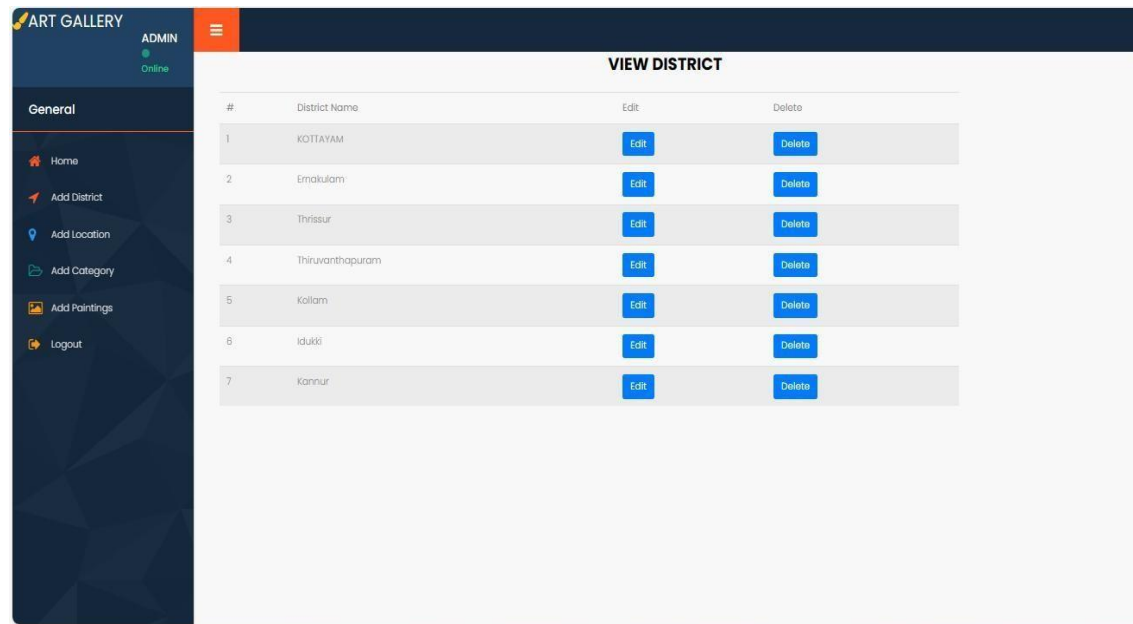
- Enter the Art Name:
- Enter the Art Rate:
- Image: No file chosen
- ChooseCategory:
-

Fig 11.7 Artwork Register

The screenshot shows the 'Customer Registration' form on the website. The header includes the 'ARTGALLERY' logo, navigation links (Home, Login, Customer Register), and contact information (+91 9876543210, artgallery@example.com). The form fields are:

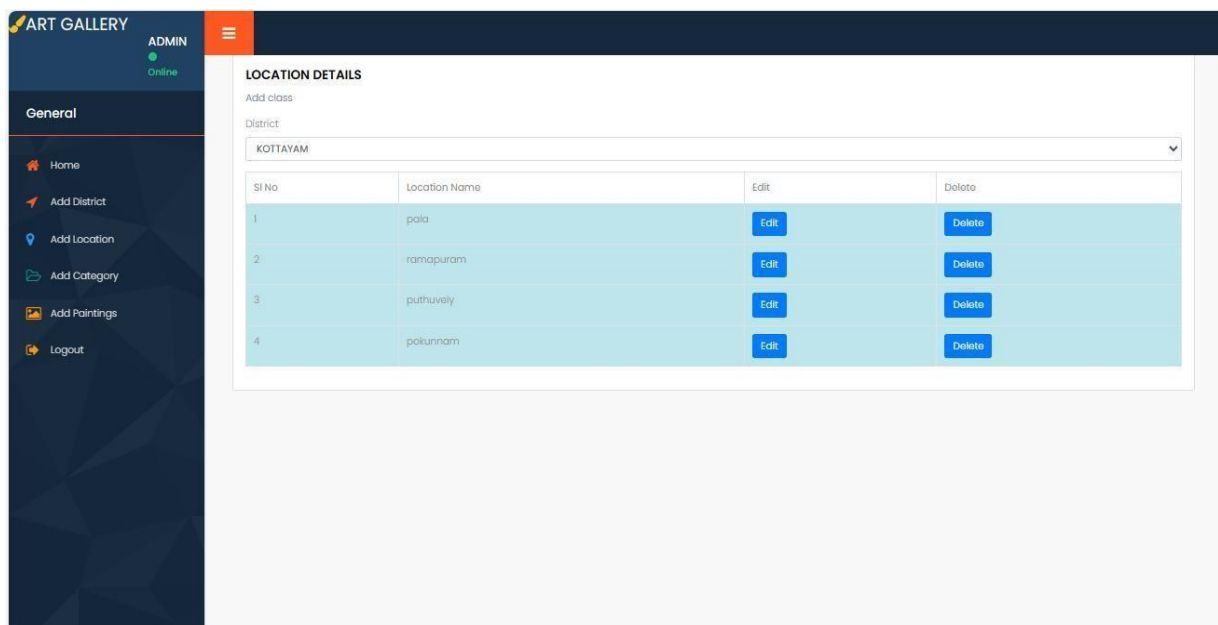
-
-
-
-
- Gender: Male ☐ Female ☐
- Select your location:
-
-
-

Fig 11.8 User Register



#	District Name	Edit	Delete
1	KOTTAYAM	Edit	Delete
2	Ernakulam	Edit	Delete
3	Thrissur	Edit	Delete
4	Thiruvananthapuram	Edit	Delete
5	Kollam	Edit	Delete
6	Idukki	Edit	Delete
7	Kannur	Edit	Delete

Fig 11.9 View District



LOCATION DETAILS

Add class:

District: KOTTAYAM

Sl No	Location Name	Edit	Delete
1	pola	Edit	Delete
2	ramapuram	Edit	Delete
3	puthuvely	Edit	Delete
4	pokunnam	Edit	Delete

Fig 11.10 View Location

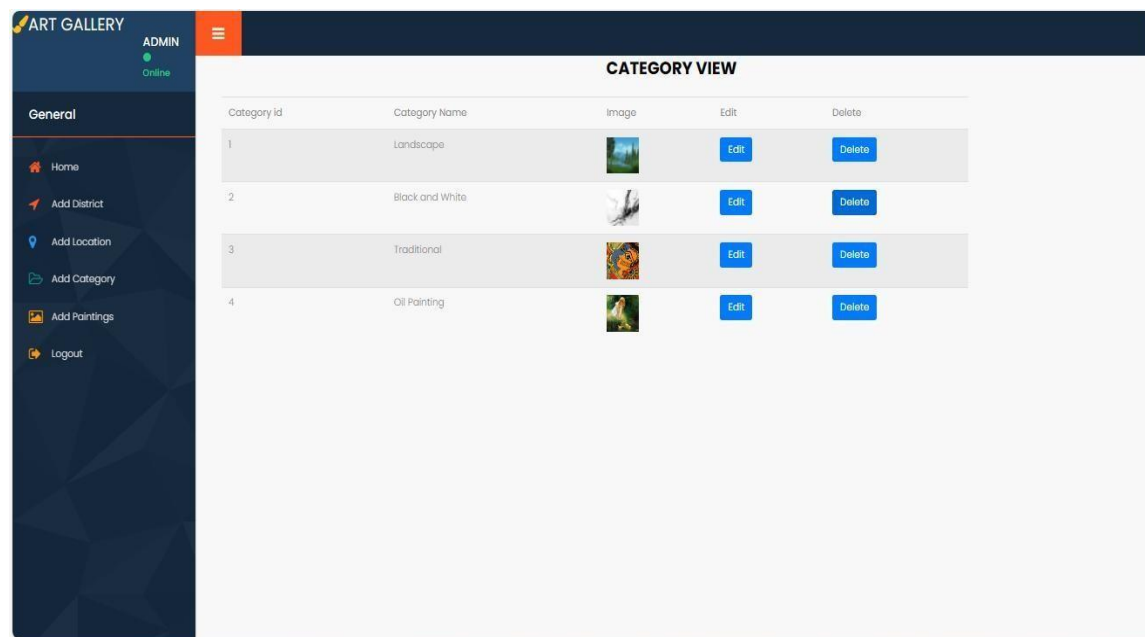


Fig 11.11 View Category

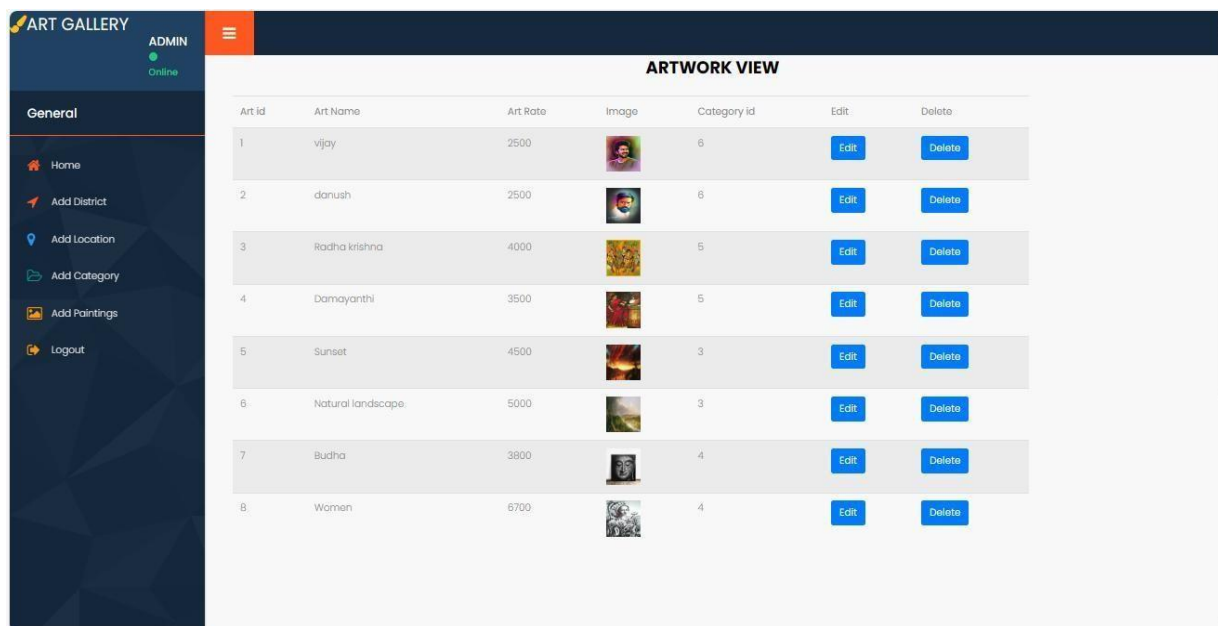


Fig 11.12 View Artwork

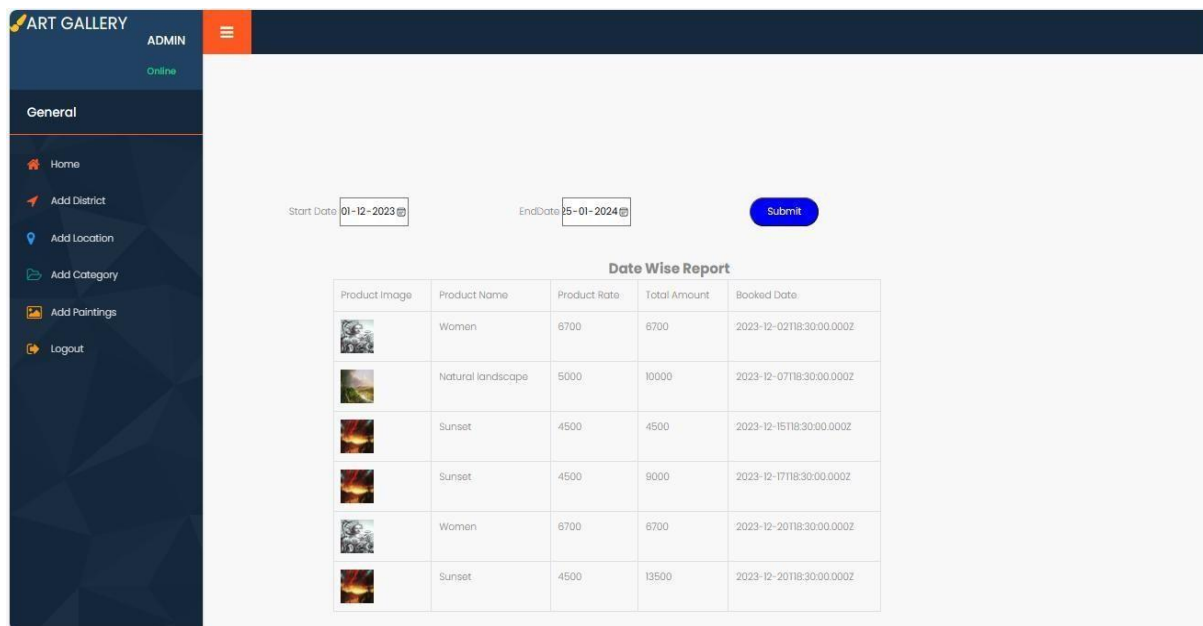


Fig 11.13 Datewise Report

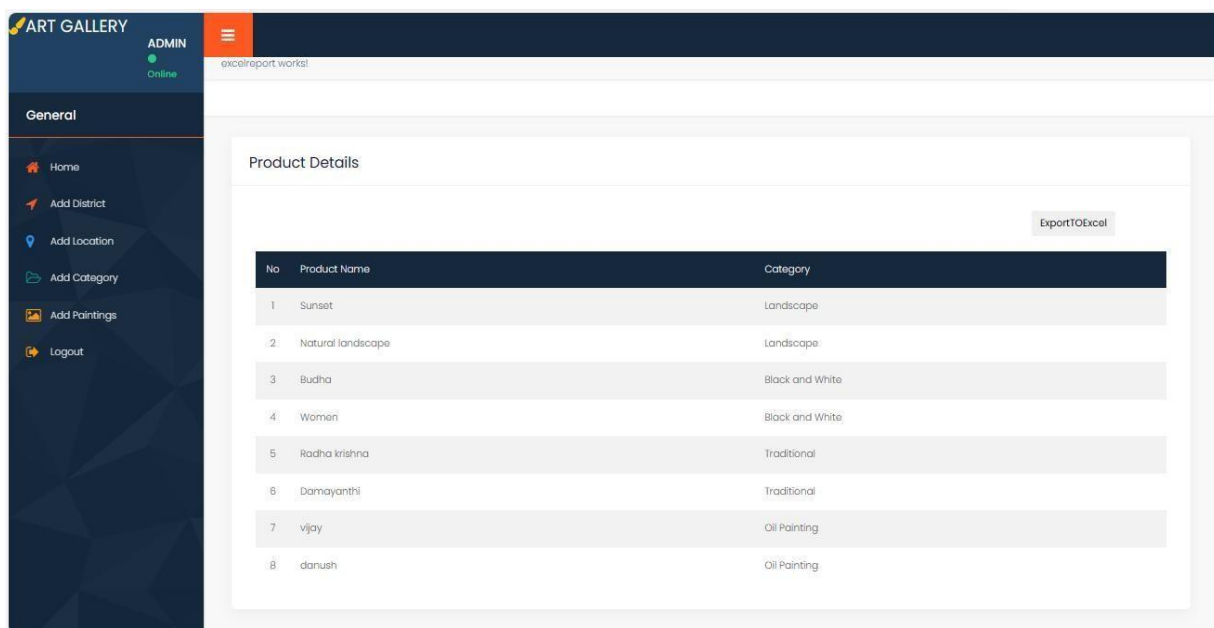


Fig 11.14 Excel Report

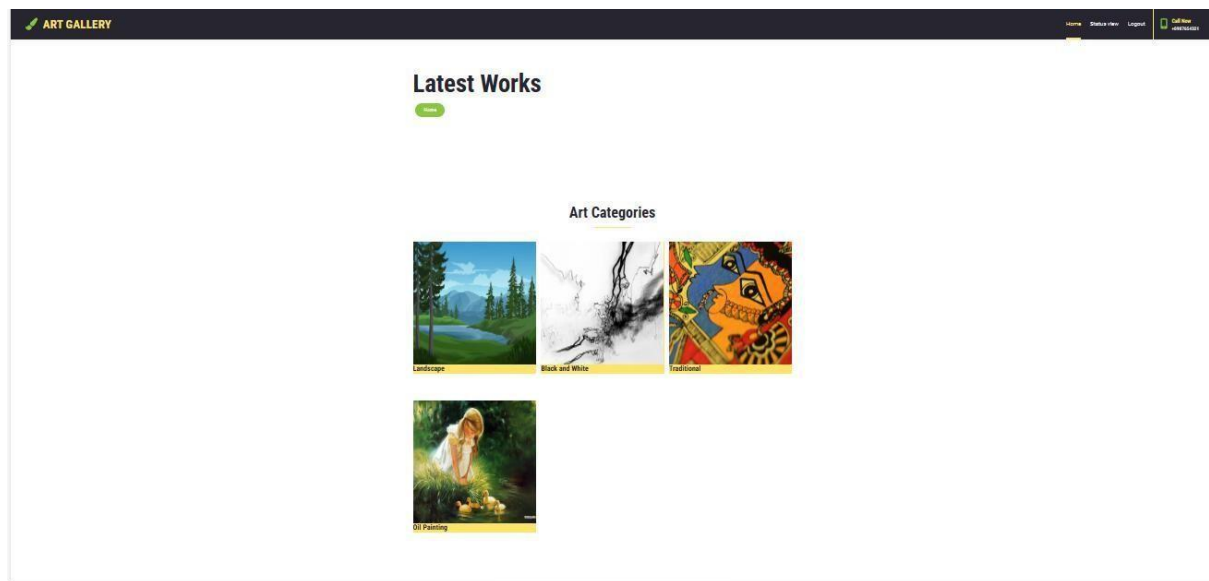


Fig 11.15 Customerhome

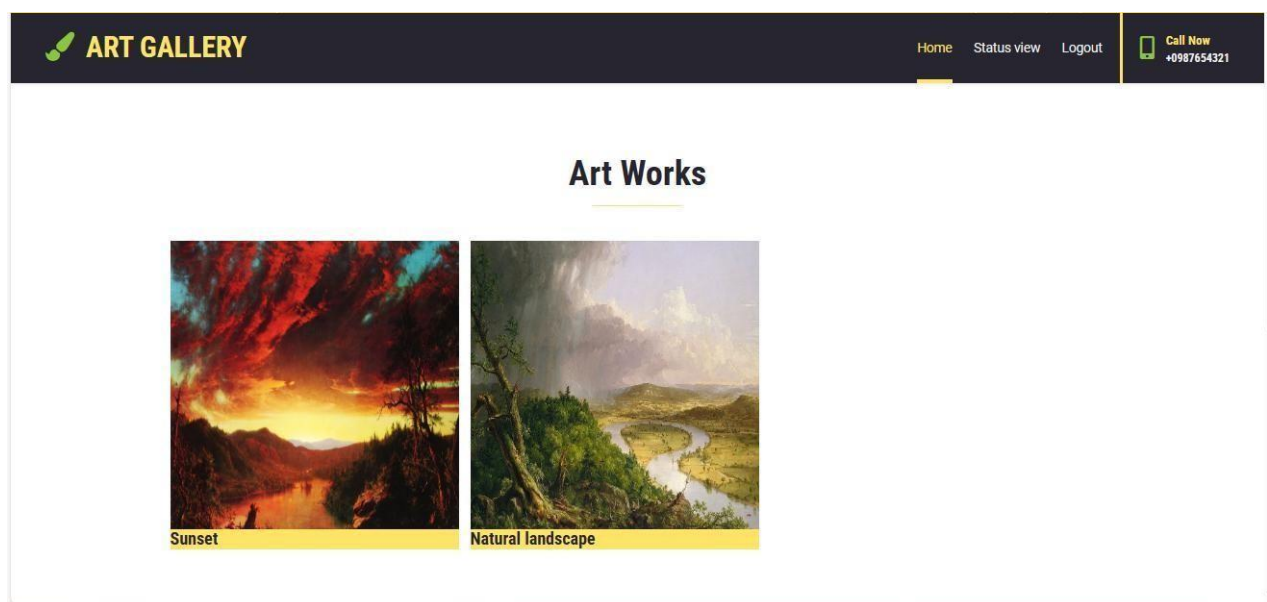


Fig 11.16 Viewcustomer Artwork

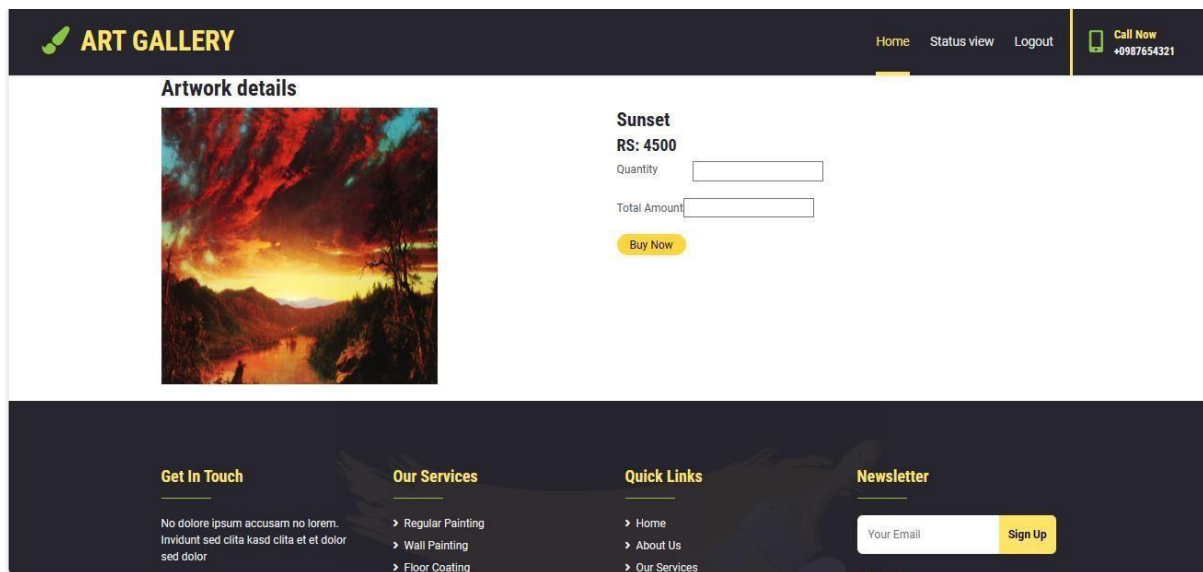


Fig 11.17 viewmore

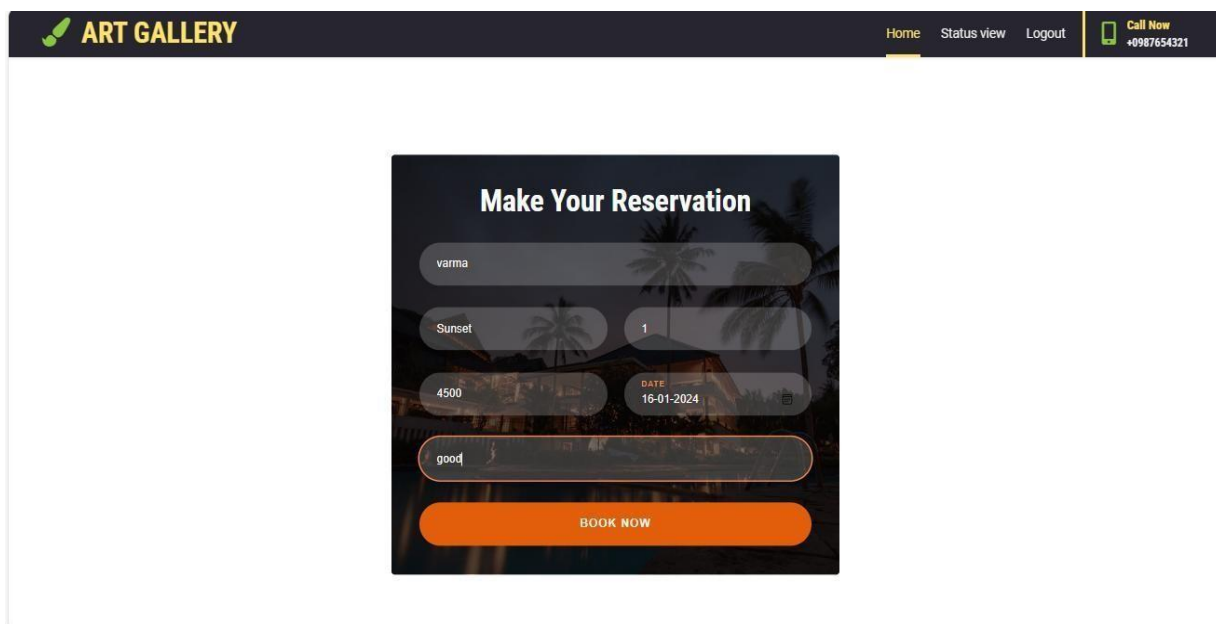


Fig 11.18 Booking

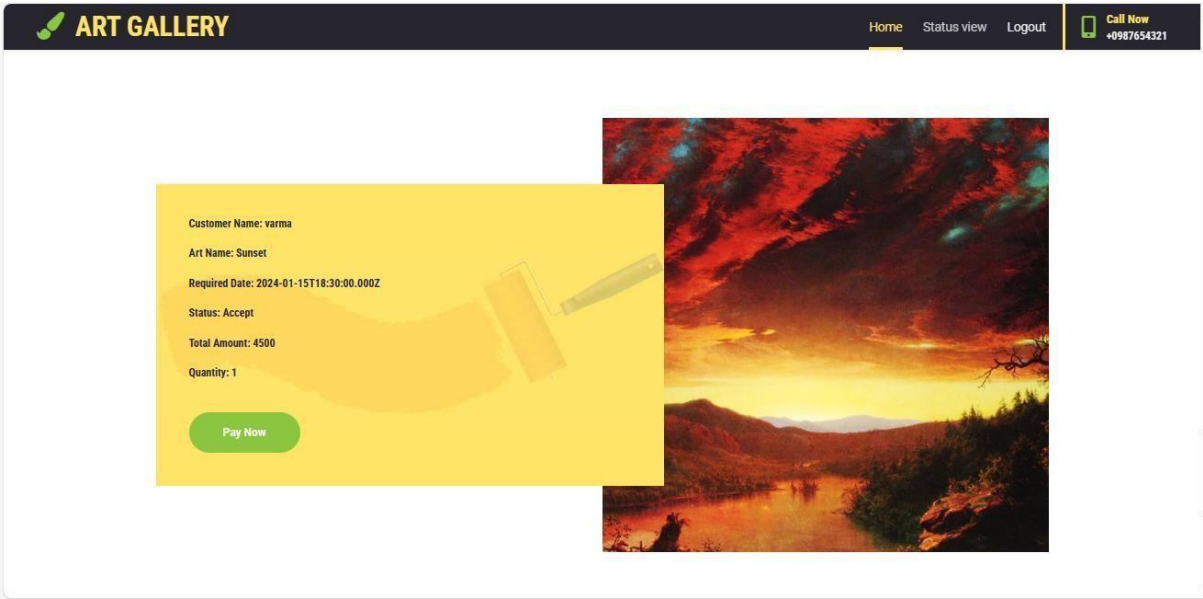


Fig 11.19 ViewStatus

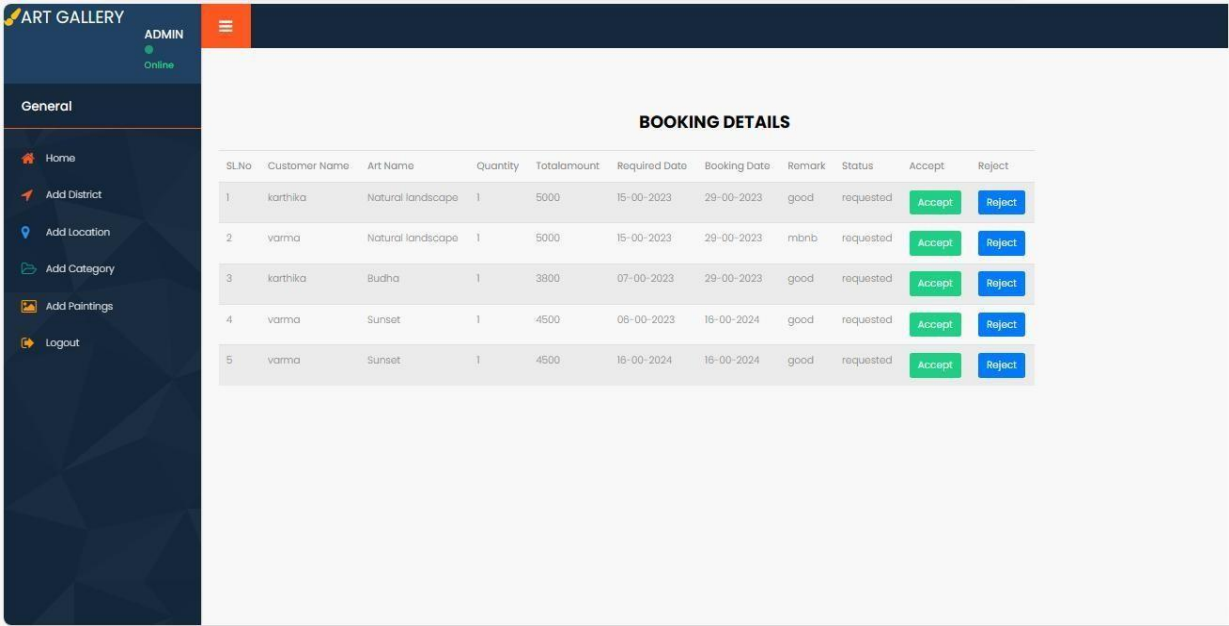


Fig 11.20 Booking View

ART GALLERY

[Home](#) [Status view](#) [Logout](#) [Call Now](#)
+0987654321

Payment

Order Summary

Customer Name:	varma
Art Name:	Sunset
Quantity:	1
Total Amount:	4500

Make Payment

CreditCardName

Credit number

6011 0000 0000 0000

CVV

Pay Now

Fig 11.21 Payment

CHAPTER 12 GLOSSARY**List of Figures**

Fig. No.	Figure Name	Page No.
Fig 3.1	ER diagram for Admin	21
Fig 3.2	ER diagram for User	22
Fig 3.3	Activity diagram of Admin	25
Fig 3.4	Activity diagram of User	26
Fig 3.5	Sequence diagram for Admin	27
Fig 3.6	Sequence diagram for User	28
Fig 3.7	Use case diagram	29
Fig 5.1	Form for User Registration	46
Fig 11.1	Guest home page	74
Fig 11.2	Login	74
Fig 11.3	Admin home	75
Fig 11.4	District registration	75
Fig 11.5	Location Register	76
Fig 11.6	Category Register	76
Fig 11.7	Artwork Register	77

Fig 11.8	User registration	77
Fig 11.9	View District	78
Fig 11.10	View Location	78
Fig 11.11	View Category	79
Fig 11.12	View Artwork	79
Fig 11.13	View Datewise Report	80
Fig 11.14	View Excel Report	80
Fig 11.15	Customer Home	81
Fig 11.16	Viewcustomer Artwork	81
Fig 11.17	Viewmore	82
Fig 11.18	Booking	82
Fig 11.19	View Status	83
Fig 11.20	Booking View	83
Fig 11.21	Payment	84

List of Tables

TableNo.	Table Name	Page No.
table 2.1	Hardware part specification Table	8
table 3.1	tbl_login	15
table 3.2	tbl_district	16
table 3.3	tbl_place	16
table 3.4	tbl_category	17
table 3.5	tbl_artwork	17
table 3.6	tbl_user	18
table 3.7	tbl_booking	19
table 3.8	tbl_payment	20
table 6.1	Test plan	57
table 6.2	Admin Login test case Report	57
table 6.3	Test case	58

List of Abbreviations

PK	: Primary Key
FK	: Foreign Key
ER	: Entity Relationship
SQL	: Structured Query Language
OS	: Operating System
DBMS	: Data Base Management System
DFD	: Data Flow Diagram
NF	: Normal Forms
CPU	: Central Processing Unit
IDE	: Integrated Development Environment
ER	: Entity Relationship
VS	: Visual Studio
UML	: Unified Modeling Language
SRS	: Software Requirement Specification
SDE	: Software Development Environment
SDLC	: Software Development Life Cycle
WAMP	: Windows, Apache, MySQL and PHP