

# **1. INTRODUCTION**

## **1.1 INTRODUCTION**

Craft World is a dynamic and user-friendly application developed in Python that aims to provide a convenient platform for craftsman to showcase and sell their handmade craft products while offering customers a diverse selection of unique and quality crafts to purchase with cheap rate. The application facilitates seamless transactions, secure payment processing, and a user-driven community that fosters collaboration and creativity within the crafting ecosystem. This web application developed to overcome the limitations of the existing system by leveraging digital technology and providing a comprehensive online platform.

Craft World is a marketing platform where we can sell and buy craft items by locally. You can find a wide collection of products like craft items, paintings, fashion items etc. in this marketing platform customers can buy products from here and no need to visit the market to find the best deals on pre-owned items there and don't waste time for looking for specific products that can be easily get in 'Craft World' by searching. This is really helpful for craftsman and they can make good income from this software. Craft World allows craftsman to create personalized profiles, where they can showcase their skills, provide materials and tutorial videos, and display a gallery of their craft products. The application features a comprehensive catalog of craft products, organized into various categories for easy navigation and discovery.

Craftsman can effortlessly add new craft products and materials, update existing listings, and manage their inventory through an intuitive interface. Customers can create accounts to access personalized features, including order history, and product reviews. Craft World incorporates powerful search functionalities and filtering options, enabling customers to find specific crafts or refine their search based on preferences.

## **1.2 PROBLEM STATEMENT**

The goal of problem statement is the recognition of basic problems of elements as indicated by customer. The basic purpose of problem statement is to obtain a thorough understanding of the needs of customer and user what exactly is desired from the software is the constraints on

solution. Crafts man often struggle to reach a wide customer base due to limited marketing channels and lack of direct connections with customers. This results in reduced market opportunities and limited income potential.

The traditional craft product selling process involves time-consuming and inefficient methods, such as physical market visits or phone calls, leading to delays, miscommunications, and inconvenience for both sellers and consumers. Customers face challenges in ensuring the quality of products they purchase. Without reliable information and transparency, there is a lack of trust and confidence in the product's origin and handling. By traditionally selling of products sellers cannot get the actual profit for their work. Because sellers first handover their products to third party persons. They would sell it for higher price and crafts man get only a cheap income, never get a good profit for their hard work. By selling product through this application they can earn good income and feedback for their product.

The 'Craft World' system aims to overcome these challenges by developing a digital platform that connects sellers directly with customers, streamlines the transaction process, ensures quality assurance, promotes price transparency, and enhances market accessibility for all stakeholders.

### **1.3 SCOPE AND RELEVANCE OF THE PROJECT**

The scope of 'Craft World' project encompasses the development and implementation of a comprehensive solution that addresses the challenges faced by craftsman and customers. The project includes the design and development of a digital platform, such as a website, to facilitate quality product to customers. It also involves the integration of features for product listings, search functionality, secure payment processing, and user reviews and ratings. The project scope extends to marketing and user acquisition strategies to promote the platform and attract a significant user base. Additionally, considerations for scalability, security, and usability are vital for the successful implementation of the project.

The 'Craft World' project holds significant relevance and benefits for various stakeholders, including customers and sellers.

## **1.4 OBJECTIVES**

The main objective of the system is to sell and buy craft product. It is a perfect online market place to sell and buy locally. You can find a wide collection of products like paintings, pottery and fashion items and many more. In this marketing platform customers can buy products from here and no need to visit the market to find the best deals on pre-owned items there and don't waste time for looking for specific products that can be easily get in 'Craft World' by searching. They can make good income from this software. Sellers can sell their product on this online site, so that everyone can see the products and if the buyers like the products they can buy it from the seller.

The sellers can upload their attractive craft work's images and tutorials and can decide the products cost by their own in this site. Users can watch the tutorials if they want by paying a small amount of money. They can also buy the materials of the products which is uploaded by the seller. Admin controls all of the activities in this project and they also get a profit from this site.

## **2. SYSTEM ANALYSIS**

### **2.1 INTRODUCTION**

System analysis is a procedure or approach that serves to determine the system's performance for a given structure of this system. An example may be a typical student project with a given input data which should be made for a defined system structure. The resulting calculation data characterize system outputs.

System analysis is used in every situation where something is developed. Analysis can also involve a series of components which perform organic functions together, such as system engineering. System engineering is an interdisciplinary field of engineering which focuses on how complex engineering projects should be designed and managed. Practitioners of system analysis are often called up to dissect systems that have grown haphazardly to determine the current components of the system.

While practitioners of system analysis can be called upon to create new systems, they often modify, expand, or document existing systems (processes, procedures, and methods). Researchers and practitioners rely on system analysis. Activity system analysis has been applied to assess various research and practice studies including business management, educational reform, educational technology, etc.

### **2.2 EXISTING SYSTEM**

The existing system of craft product selling and buying field is fine, but there some disadvantages such as craftsman cannot get enough profit or income from their works.

Many of the craftsman approach local boutiques, gift shops, and galleries to see if they are interested in carrying your crafts. This can help you tap into the local market.

But it is more difficult and time consuming process. Also sellers can't get good profit for the products. In today's, there many online market system for showcasing or selling the craft products. But they only provide the products images and short videos. If anyone interested in making the product, Craft World system also provide tutorials for the products.

#### **2.2.1 LIMITATIONS OF THE EXISTING SYSTEM**

- ✧ Never get the exact profit to the sellers.
- ✧ Sellers and users have only limited access to the system.

- ✧ Tutorials videos are not available.
- ✧ Time consuming

## 2.3 PROPOSED SYSTEM

Craft World allow you to set up a online market to showcase and sell your crafted items. These platforms provide a large customer base and tools for managing and processing payments, and shipping. Sellers can upload their works directly in the site with actual rate. They can get the actual price for their works. By uploading tutorial video content showcasing your crafting process, behind-the-scenes looks, and can help you build a loyal following and attract potential customers. Sellers can also upload the work's images. By providing video, users can watch it with low rate and can try to make that product by users if they interested. The materials will also be provided through this site. Craftsman or sellers have much access to the system. They get a good income from their works. Admin is the overall controller of the system. Admin is one who verify the sellers and products.

### 2.3.1 ADVANTAGES OF PROPOSED SYSTEM

- ✧ Provide a good platform for sellers to showcase their products.
- ✧ Craft World applicaton allows you to reach customers not only locally but also globally. Crafts can be discovered by people from different parts of the Kerala, expanding your potential customer base significantly.
- ✧ This Online platforms allows to showcase a larger variety of products without the space limitations of a physical store. This enables you to display your entire range of crafts and attract a broader audience.
- ✧ Updating product listings, adding new items, and making changes to your inventory is relatively simple on this platform.
- ✧ Craft World provide ample space to include detailed product descriptions, high-quality images, materials and even videos. This helps customers make informed purchasing decisions and reduces the need for them to physically inspect the items.
- ✧ This application allows you to directly engage with customers through chat feature. This personalized interaction can build stronger customer relationships and encourage repeat business.

- ✧ Online payment is possible.
- ✧ Sellers and users can update their personal information in the site.

## 2.4 FEASIBILITY STUDY

A feasibility study is an analysis that takes all of a project's relevant factors into account—including economic, technical, legal, and scheduling considerations—to ascertain the likelihood of completing the project successfully. Project managers use feasibility studies to discern the pros and cons of undertaking a project before they invest a lot of time and money into it. Feasibility studies also can provide a company's management with crucial information that could prevent the company from entering blindly into risky businesses.

The goals of feasibility studies are as follows:

- To understand thoroughly all aspects of a project, concept, or plan.
- To become aware of any potential problems that could occur while implementing the project.
- To determine if, after considering all significant factors, the project is viable—that is, worth undertaking.

Need Of Feasibility Study:

Feasibility study is so important stage of Software Project Management Process as after completion of feasibility study it gives a conclusion of whether to go ahead with proposed project as it is practically feasible or to stop proposed project here as it is not right/feasible to develop or to think/analyze about proposed project again.

Along with this Feasibility study helps in identifying risk factors involved in developing and deploying system and planning for risk analysis also narrows the business alternatives and enhance success rate analyzing different parameters associated with proposed project development.

There are three aspects in the feasibility study portion of the preliminary investigation

- ✧ Technical Feasibility
- ✧ Operational Feasibility
- ✧ Economic Feasibility

### **2.4.1 TECHNICAL FEASIBILITY**

A technical issue usually raised during the feasibility stage of the investigation includes the following:

- We have the necessary technology exist to do what is suggested by the customer.
- The Equipment rental system need hardware have the technical capacity to hold the data required to use the new system.
- The proposed system provide adequate response to inquiries regardless of the number of location of users, if users have adequate network connection they can easily access the system.
- The system can update after development as per customer requirement.
- System provide integrity and security to organization data by password protection.
- Earlier no system exists to cater to the needs of the Secure infrastructure implementation system, the current system developed is technically feasible.
- The database's purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles specified. Therefore, it provides the technical guarantee of accuracy reliability and security.

### **2.4.2 OPERATIONAL FEASIBILITY**

In operational feasibility the entire application is checked to analyse whether the system will be used if it is developed and implemented. Also, it is checked whether there will be resistance from users that may undermine the possible application benefits. There is no barrier for implementing the system. The system also helps to access the information immediately as need arises, Thus the system is found to be operationally feasible. Here this system underwent operational feasibility study. The system is checked and it and user-friendly software it can be used by the customers who have limited computer knowledge. The system will help to access the information immediately whenever needed.

### **2.4.3 ECONOMIC FEASIBILITY**

Here an evaluation of development cost weighted against the ultimate income or benefit derived from the developed system. The cost for the development of the project has been evaluated and we want to check that the cost does not exceed. The economic and financial

analysis is used for evaluating the effectiveness of the candidate system. This project also undergone economic feasibility study and found that it is feasible.

For the implementation of the software Craft World system required a computer system, calculate the cost for a system. Time required for developing the web application, server and domain for the deployment of the application, Cost for the server is calculated. So the cost of development, deployment does not exceed its beneficial cost. This brought to the conclusion that the system is economically feasible in this context.

## **2.5 SOFTWARE ENGINEERING PARADIGM APPLIED**

The project follows Agile Process Model. The meaning of Agile is swift or versatile."Agile process model" refers to a software development approach based on iterative development. Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning. The project scope and requirements are laid down at the beginning of the development process. Plans regarding the number of iterations, the duration and the scope of each iteration are clearly defined in advance.

Each iteration is considered as a short time "frame" in the Agile process model, which typically lasts from one to four weeks. The division of the entire project into smaller parts helps to minimize the project risk and to reduce the overall project delivery time requirements. Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client.

### **Phases of Agile Model**

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.

4. Testing: In this phase, the Quality Assurance team examines the product's performance and looks for the bug.

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.

### **3. SYSTEM DESIGN**

#### **3.1 INTRODUCTION**

The system design is the most creative and challenging phase of system development life cycle. It is an approach for the creation of proposed system, in which the logic and details structure of the proposed system is designed, which will help the system coding. The most creative and challenging phase of the system development process is design phase it is a solution, how to approach to the creation of the proposed system. Design is the first step in the development of the engineered product is initiated only after a clear exposition of expected product is available. System Design is vital for efficient database management. It provides the understanding of procedural details necessary for implementing the system .A number of sub-systems is to be identified which constitute the whole system.

#### **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.
- Accuracy and integrity.
- Recovery from failure.

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

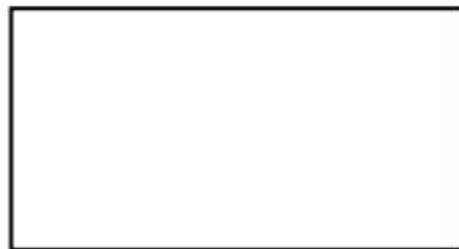
and data has in direct access to stores device. This is the difference between logical and physical data.

### 3.2.1. ENTITY RELATIONSHIP MODEL

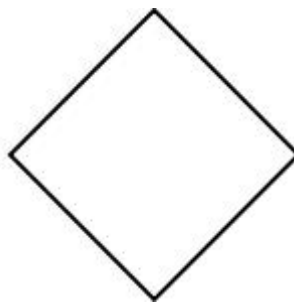
An entity relationship diagram (ERD) is a data modelling technique that graphically illustrates an information system's entities and the relationships between those entities. An ERD is a conceptual and representational model of data used to represent the entity framework infrastructure.

The elements of an ERD are:

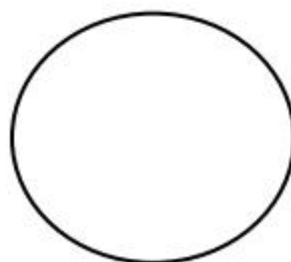
◆ Entities:



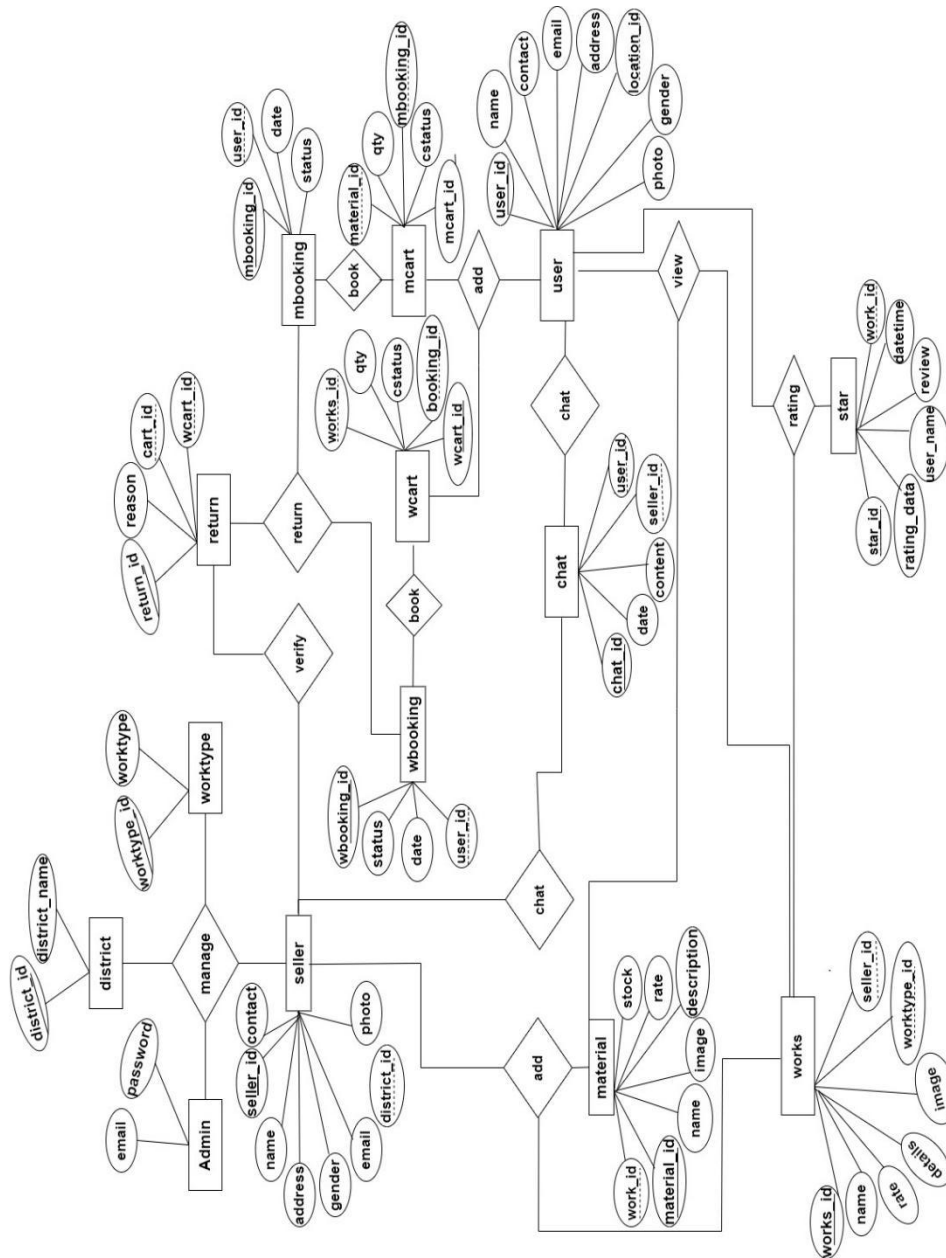
◆ Relationships



◆ Attributes



## E-R DIAGRAM



### 3.2.2. DATA DICTIONARY

A data dictionary is a collection of metadata such as object name, data type, size, classification, and relationships with other data assets. Think of it as a list along with a description of tables, fields, and columns. A data dictionary is used by data administrators, analysts and engineers to understand and trust data assets. It helps in the creation of authentic, transparent and consistent data throughout the organization.

#### TABLES

##### 1. Table name: tbl\_worktype

Description: Used to store the type of works

Primary Key: worktype\_id

Field Name	Data Type	Description
worktype_id	integer	Used to uniquely identify the worktype
worktype	varchar(20)	Type of work

##### 2. Table name: tbl\_district

Description: Used to store district details

Primary Key: district\_id

Field Name	Data Type	Description
district_id	integer	Used to uniquely identify the district
district_name	varchar(50)	Name of district

##### 3. Table name: tbl\_place

Description: Used to store place details

Primary Key: place\_id

Foreign Key: district\_id

Field Name	Data Type	Description
Place_id	integer	To store the id of the place table.
place_name	varchar(20)	To store the name of place.
latitude	varchar(50)	To store the latitude of place.
longitude	varchar(50)	To store the logitude of the place
district_id	integer	To store the id of district

#### 4. Table name: tbl\_location

Description: To store location details

Primary Key: location\_id

Foreign Key: place\_id

Field Name	Data Type	Description
location_id	integer	To store the id of location
name	varchar(20)	To store the name of location
lati	varchar(20)	To store Latitude of location
Longi	varchar(20)	To store longitude of location.
Place_id	integer	To store the id of place

#### 5. Table name: tbl\_user

Description: To store user details

Primary Key: user\_id

Foreign Key: location\_id

Field Name	Data Type	Description
user_id	integer	Used to uniquely identify the user
name	varchar(50)	To store name of user
contact	varchar(50)	To store contact of user
email	varchar(50)	To store email id
address	varchar(50)	To store address of the user
location_id	varchar(50)	To store location id
gender	varchar(50)	To store gender of user
photo	file	To store image of user
password	varchar(50)	To store password of user

6. Table name: tbl\_videopay

Description: To store the video payment details

Primary Key: videopay\_id

Foreign key: seller\_id, user\_id

Field Name	Data Type	Description
videopay_id	integer	To store the videopay table unique id
seller_id	integer	To store the seller id
user_id	integer	To store the user id

## 7. Table name: tbl\_seller

Description: Used to store seller details

Primary Key: seller\_id

Foreign Key: place\_id

Field Name	Data Type	Description
seller_id	integer	Used to uniquely identify the seller
name	varchar(50)	To store name of seller
contact	varchar(50)	To store contact of seller
email	varchar(50)	To store Email id
address	varchar(50)	To store Address of the seller
district	varchar(50)	To store District name
gender	varchar(50)	To store gender of seller
photo	file	To store image of seller
password	varchar(50)	To store password for seller
proof	file	To store proof of seller
status	integer	To store status
place_id	integer	To store place id

## 8. Table name: tbl\_wgallery

Description: To store the gallery images of works

Primary Key: wgallery\_id

Foreign key: work\_id



Field Name	Data Type	Description
wgallery_id	integer	To store the wgallery id
image	file	To store the different view of works
work_id	integer	To store the id of work

9. Table name: tbl\_material

Description: To store the details of materials

Primary Key: material\_id

Foreign key:work\_id

Field Name	Data Type	Description
material_id	integer	Used to uniquely identify material
name	varchar(50)	To store the name of material
image	file	To store the Image of material
rate	integer	To store the rate of material
stock	integer	To store the stock of material
work_id	integer	To store the id of work
description	Varchar(50)	To store the details of material

10. Table name: tbl\_works

Description: To store the details of works

Primary Key: works\_id

Foreign key: seller\_id, worktype\_id

Field Name	Data Type	Description
works_id	integer	Used to uniquely identify the works
name	varchar(50)	Name of work
rate	integer	Rate of work
details	varchar(50)	Details of work
image	file	Image of work
video	file	Making video of work
seller_id	integer	Selled id of the corresponding work
worktype_id	integer	worktype id of the corresponding work

11. Table name: tbl\_chat

Description: To store the chat details

Primary Key: chat\_id

Foreign key: user\_id, seller\_id

Field Name	Data Type	Description
Chat_id	integer	To store the id of chat
Date	date	To store the date of chat
Content	varchar(100)	To store the chat content
From_seller_id	integer	To store the id of seller
From_user_id	integer	To store the id of user
To_seller_id	integer	To store the id of seller
To_user_id	Integer	To store the id of user

12. Table name: tbl\_star

Description: To store the star rating

Primary Key: star\_id

Foreign key:work\_id

Field Name	Data Type	Description
star_id	integer	To store id of star table
rating_data	integer	To store rating data
user_name	varchar(30)	To store the name of user
user_review	varchar(50)	To store the review from user
datetime	datetime	To store the time and date
work_id	integer	To store work id

### 13. Table name: tbl\_complaint

Description: To store the complaints

Primary Key: complaint\_id

Foreign key: user\_id, seller\_id

Field Name	Data Type	Description
complaint_id	integer	Used to uniquely identify complaint id
title	varchar(50)	To store the title of the complaint
content	varchar(50)	To store the content of the complaint
status	integer	To store the status of complaint
reply	varchar(50)	To store the reply from admin
reply_date	date	To store the reply date
date	date	To store the date of complaint
seller_id	integer	To store the Id of Seller
user_id	integer	To store the Id of user

### 14. Table name: tbl\_feedback

Description: To store the feedback details

Primary Key: feedback\_id

Foreign key: seller\_id, user\_id

Field Name	Data Type	Description
feedback_id	integer	To store the id of the feedback
content	varchar(50)	To store the feedback
date	date	To store the date
seller_id	integer	To store the seller id
user_id	integer	To store the user id

#### 15. Table name: tbl\_mbooking

Description: To store material booking details

Primary Key: mbooking\_id

Foreign key: user\_id

Field Name	Data Type	Description
mbooking_id	integer	To store the id of the booking
status	integer	To store the status
date	date	To store the date of booking
User_id	integer	To store the user id

#### 16. Table name: tbl\_mcart

Description: To store the material cart details

Primary Key: mcart\_id

Foreign key: material\_id, mbooking\_id

Field Name	Data Type	Description
mcart_id	integer	Used to store the id of the mcart table
qty	integer	To store the quantity of material
cstatus	integer	To store the status
material_id	integer	To store the material id
mbooking_id	integer	To store the mbooking id

17. Table name: tbl\_wbooking

Description: To store the work booking details

Primary Key: wbooking\_id

Foreign key: user\_id

Field Name	Data Type	Description
wbooking_id	integer	To store id of wbooking table
status	integer	To store the status
date	date	To store the date of booking
user_id	integer	To store the user id

18. Table name: tbl\_wcart

Description: To store the work cart details

Primary Key: wcart\_id

Foreign key: work\_id, booking\_id

Field Name	Data Type	Description
wcart_id	integer	To store the id of the wcart table
qty	integer	To store the quantity of product
cstatus	integer	To store the status
works_id	integer	To store the worksid
booking_id	integer	To store the booking id

19. Table name: tbl\_return

Description: To store the return of material and work details

Primary Key: return\_id

Foreign key: cart\_id, wcart\_id

Field Name	Data Type	Description
return_id	integer	To store the id of return table to identify uniquely
reason	varchar(50)	To store the reason for return
cart_id	integer	To store the material booking id
wcart_id	integer	To store the work booking id

### 3.3 OBJECT ORIENTED DESIGN-UML DIAGRAM

#### 3.3.1 USECASE DIAGRAM

A Use case diagram is a dynamic or behaviour diagram in UML. Use case diagram model the functionality of a system using actors and use cases. Use cases are set of actions, services, and functions that the system needs to perform. In this context, a “system” is something being developed or operated, such as web site. The “actors” are people or entities operating under defined roles within the system.

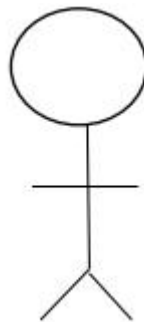
#### Symbols in Use case Diagram

- System

- Use case

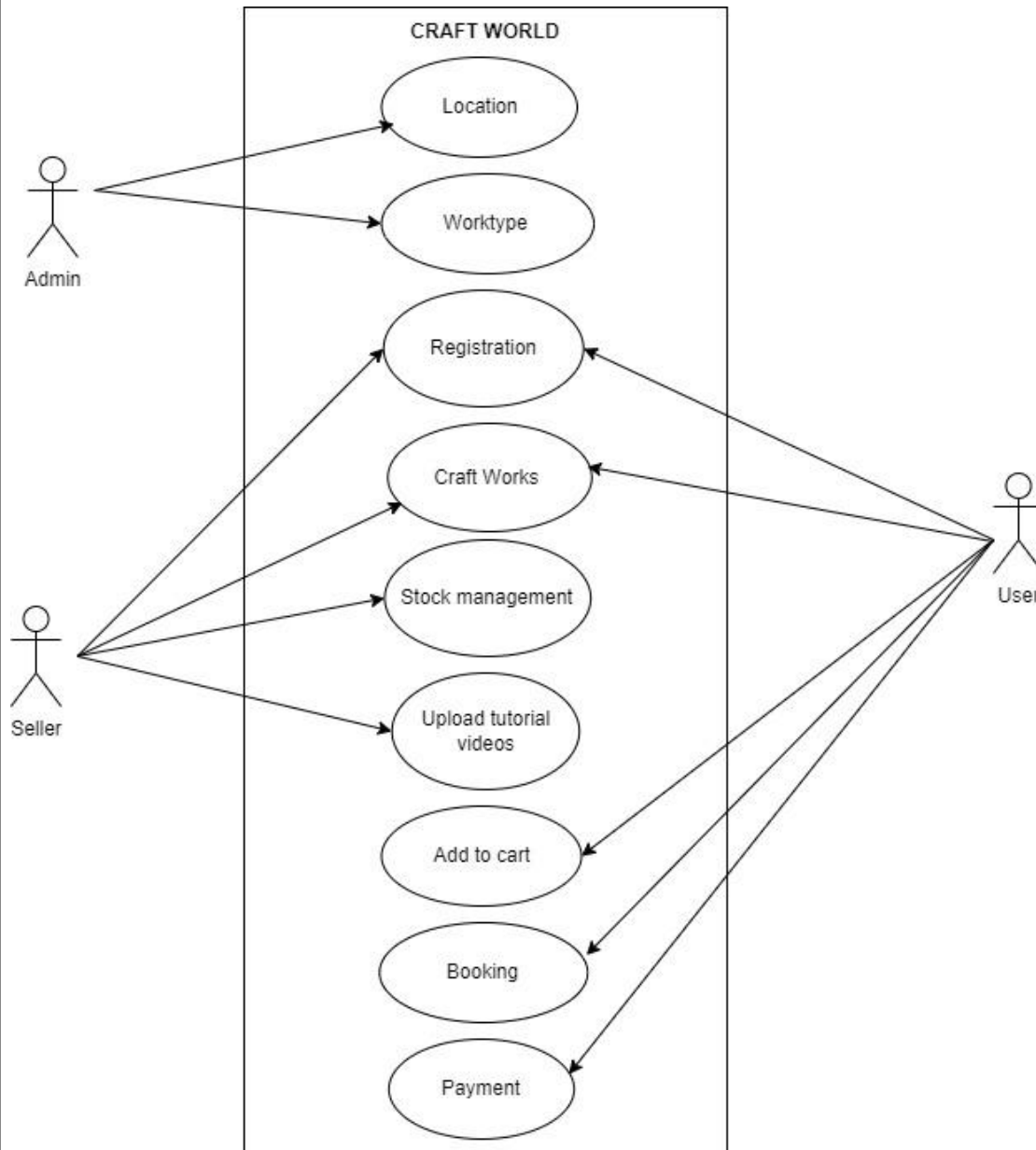


- Actor



- Relationships



**USE CASE DIAGRAM**

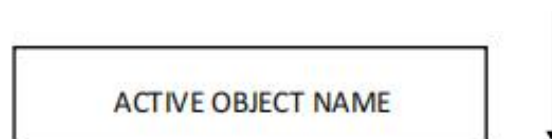


### 3.3.2 SEQUENCE DIAGRAM

A sequence diagram is an interaction that shows how objects operate with one another and in what order. It is a constant of a message sequence chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between them the objects needed to carry out the functionality of the scenario. Sequence diagram are typically associated with the use case realizations in the logical view of the system under development.

#### Symbols in Sequence Diagram

- **Life lines & life line boxes:** Active objects can be an object which is an instance of a class and can be drawn with rectangular box called the life line box with its name is specified within the box. The long dashed line tailing the objects is called life lines.



- **Message:** It is used to illustrate the communication between different active objects. Each message between object is represented with a message expressions and filled arrow solid lines from the calling active objects life line to recipient life lines.

- **Synchronous:** This type of message is used when it is important that a message is received and completed before execution of control flow begins.



- **Return:** Return messages shows that the control flow has return into the calling active objects and the synchronous messages are completed its operations

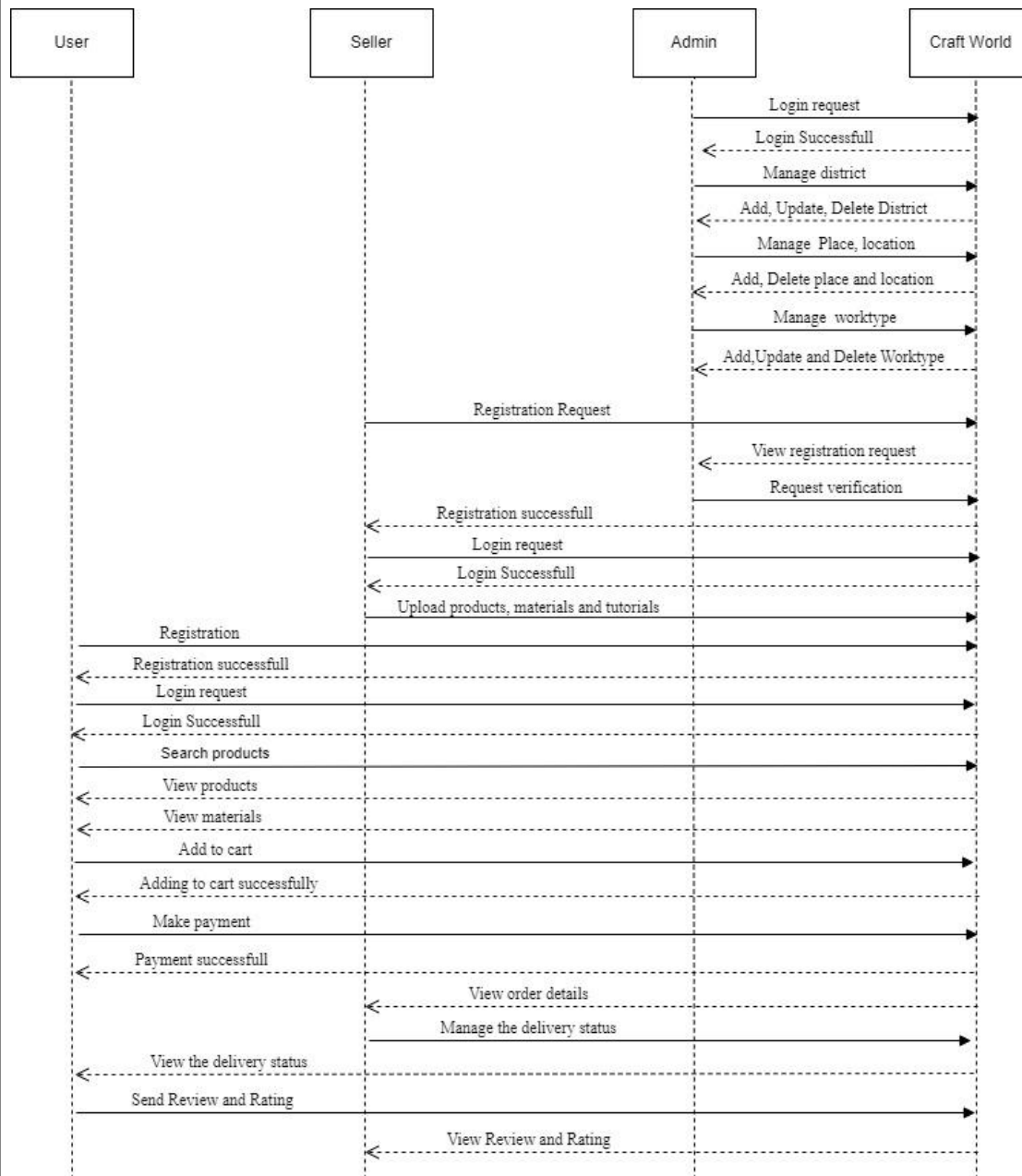


- **Asynchronous:** This type of messages is send from an active object and waits for responds. This type of message is used when control flow doesn't lead to interrupted before the completion of operation.



- **Flat:** This type of message is used when there is no distinction between asynchronous and synchronous messages.

## SEQUENCE DIAGRAM



### 3.3.3 ACTIVITY DIAGRAM

Activity diagrams are graphical representation of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the UML, activity diagrams are intended to model both computational and organizational processes. It shows the overall flow of control.

#### Symbols in Activity Diagram

- **Action State:** It is an automatic state once it started execution. It will come to completion without any interruption at the middle of execution.



- **Transition:** When the action or activity state completes the flow of control passes immediately to the next action or activity state. This is represented by using the solid lines.



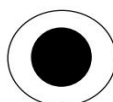
- **Branching:** It specifies the alternative paths that are taken based on some Boolean expressions.



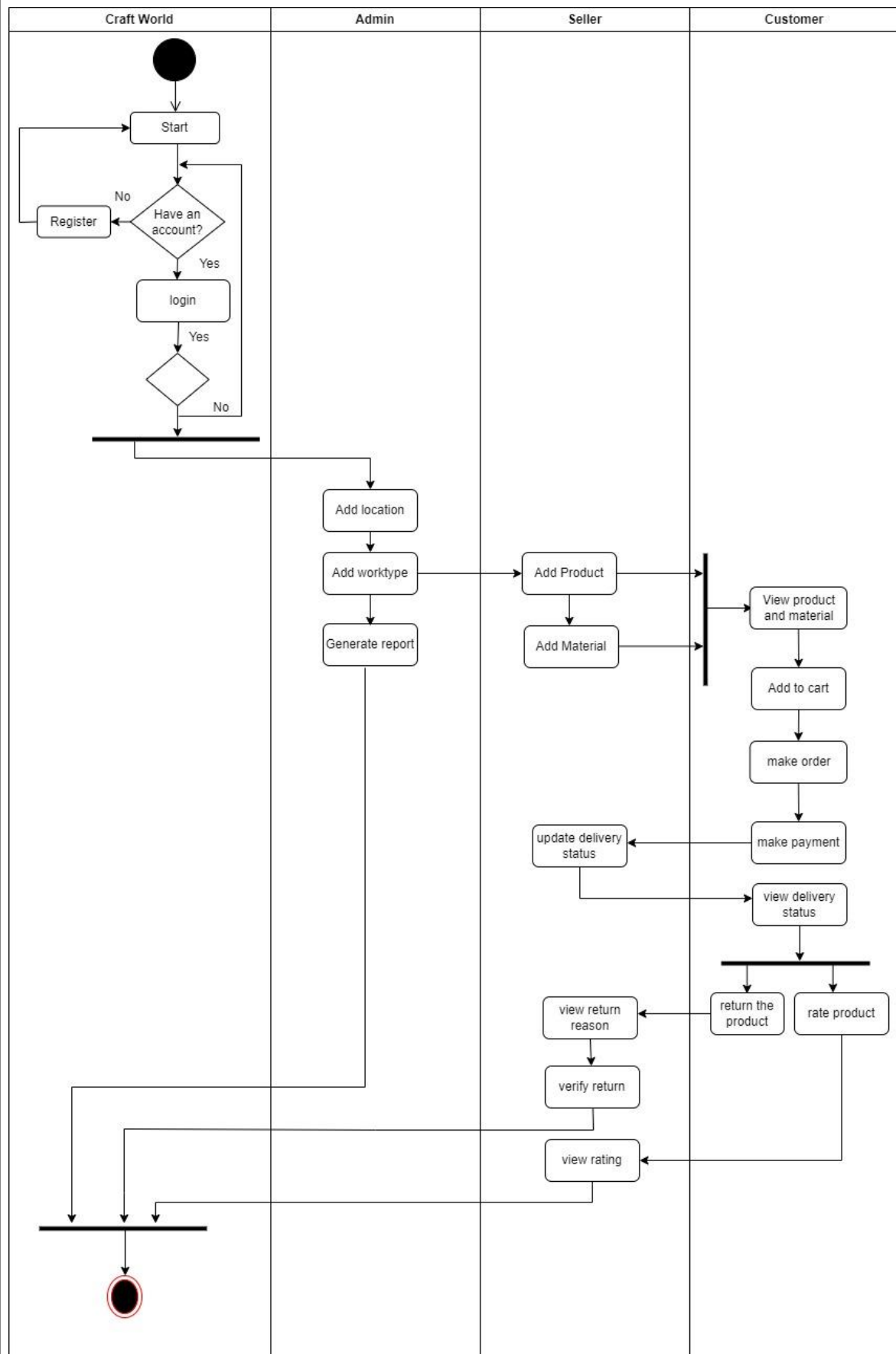
- **Initial node:** The initial node is a control node from where the activity is invoked. An activity may have more than one initial node.



- **Final node:** The final node shows the end of the activity. An activity can have more than one final node but the first one reached will stop the activity.



## ACTIVITY DIAGRAM



### 3.4 MODULAR DESIGN

Modular design is an architecture approach that include breaking down a software system into smaller, self-contained, and interchangeable components known as modules. Each module encapsulates a specific set of functionalities or tasks, and it can be developed, tested, and maintained independently of other modules in the system. This approach is based on the principle of modularity, which aims to improve the software development process by promoting reusability, maintainability, and flexibility.

Key characteristics of a modular description in software engineering include:

- Encapsulation: Each module hides its internal implementation details, exposing only a welldefined interface for interaction with other modules. This encapsulation ensures that changes within a module do not affect the rest of the system, as long as the external interface remains unchanged.
- Abstraction: Modules should provide a clear and abstract interface that allows other parts of the system to use their functionalities without needing to understand their internal complexities. Abstraction helps reduce the complexity of the overall system and improves its comprehensibility.
- Independence: Modules should be independent units that can be developed and tested in isolation. This makes it easier to identify and fix issues within specific modules without affecting the rest of the system. It also enables parallel development, as different teams can work on different modules concurrently.
- Reusability: Modular design encourages the creation of reusable components. Once a module is developed and tested, it can be used in multiple projects, saving time and effort in future developments.
- Maintainability: Since modules are self-contained and have clear interfaces, maintaining and updating the system becomes more manageable. Changes to one module should not impact other parts of the system, reducing the risk of unintended side effects.
- Flexibility: Modularity allows for easier system extension and modification. New features can be added

by developing new modules and integrating them into the existing system, without the need to modify the entire codebase.

The process of modularization involves identifying distinct functionalities in the software, defining the interfaces between modules, and organizing the code base accordingly. The use of standardized interfaces, such as Application Programming Interfaces (APIs), helps ensure smooth communication and integration between different modules.

### **3.4.1 MODULES DESCRIPTION**

The project entitled —THE CRAFT WORLD is a software solution which is a marketing platform for craft items like handicrafts, paintings and jewellery and many more. Admin holds pivotal authority, managing location, sellers and category of products. Seller can upload the products, materials related to craft items and also tutorial videos. Admin and seller can generate reports. Customers can create their user account in this system and can login in to their user accounts. The major modules in this system are:

- User Authentication and Registration
- Product management
- Material management
- Make Orders
- Payment
- Return Products

#### **User Authentication and Registration**

The User Authentication and Registration module is responsible for managing user accounts, ensuring secure access, and facilitating the registration process. Users can register into the system with necessary details such as name, email, and password, mobile number, address and place along with their password. Once a user account is created successfully, they can login to the system using their email and password. Only the registered users with valid login credentials can login to the system.

**Product management**

In this project, Admin adds product category to the system. Seller can upload products to the system along with the product details such as product name, worktype of product, price, description of the product with the image and tutorial videos. The customers can view the uploaded products and when they make an order seller get a request for accepting or rejecting the order. Seller can send mail for accept or reject. After that seller manage the delivery of the product.

**Material management**

Seller can upload craft product making material to the system along with the material details such as material name, price, description of the material with the image. Seller verifies the stock of each and every material in the system. The customers can view the uploaded material and when they make an order seller get a request for accepting or rejecting the order. Seller can send mail for accept or reject. After that seller manage the delivery of the material. Seller verifies the stock of each and every material in the system. When they make purchases, the stock of the material will be automatically reduced according to the quantity of customer order. When the material became out of stock, the customers no longer to access the material if it until its stock is restored.

**Make Orders**

The customers can view various products in 'Craft World'. Customers can login to their user accounts with their email and password and can view various products. They can add items to their shopping cart as their wish and can make orders. They can make orders for any number of products and materials. When they make orders, they can add the quantity of products and material and in case of material if the quantity is greater than the available quantity, they got an alert message indicating the out of stock of material. Once they make orders, the stock of material in the system will be reduced according to the order quantity of material. After the customer makes orders, they can view the status of their orders and they are redirected to make payments.

Craft products quantity is not stored in the system. When customer make order seller get a request of order. Then seller can accept or reject order according to the order details such as quantity.



**Payment**

When the customer makes orders, they are redirected to make payments. Customer can make online payment with their card details. Once their payment is complete, their order status will be changed and the seller can view the paid orders of the customer. Seller will deliver products only after they complete the payment.

**Return Products**

Once the products will be delivered, the customer can return the products within two weeks of delivery. They can return the products back to the seller with valid a reason for return. The products cannot be returned after 2 weeks and when they make a return request, seller verifies the products and the customer will be get the price of that products back. Seller, who have the responsibility of delivery will take the products back and make return payment.

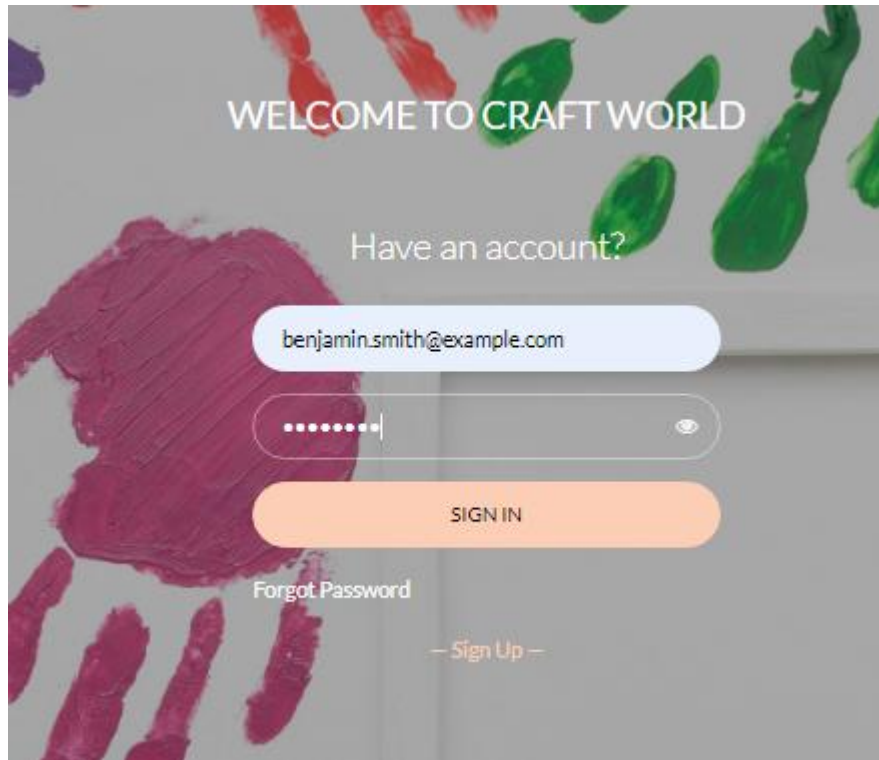
**3.5 INPUT DESIGN**

The user interface design is very important for any application. The interface design describes how the software communicates within itself, to system that 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. They data is validated wherever it requires in the project. This ensures that only the correct data have been incorporated into system. The goal of designing input data is to make the 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 for giving 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 the source document.

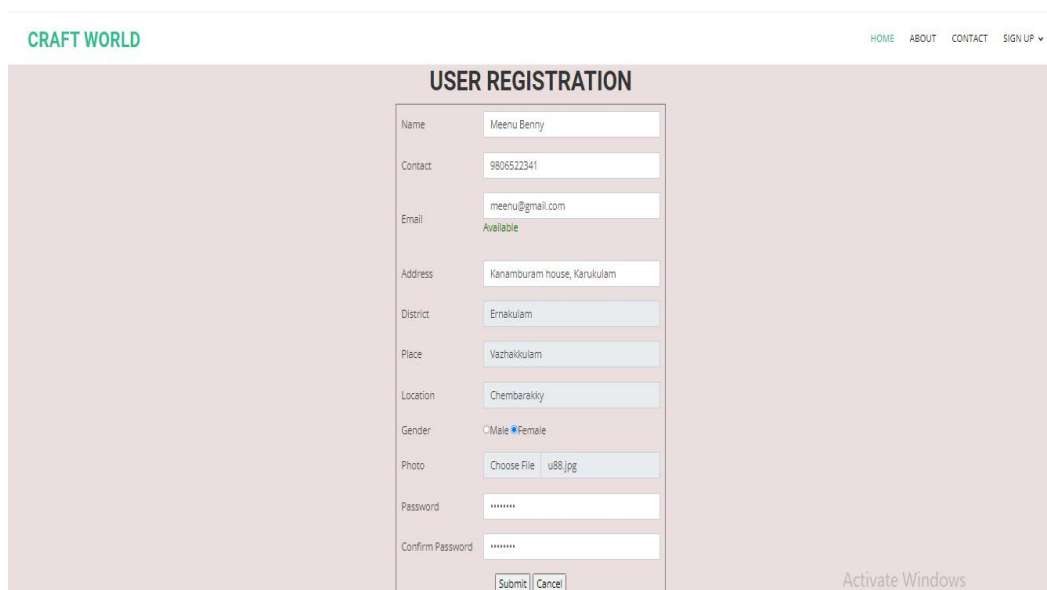
- Prototype online input screens.
- Design the input screen

## LOGIN PAGE



The login page features a background with colorful paint splashes in red, green, and purple. The text "WELCOME TO CRAFT WORLD" is centered at the top. Below it, the text "Have an account?" is displayed. A light blue rounded rectangle contains the email address "benjamin.smith@example.com". Below this, a password field is shown with a series of dots and a toggle icon. An orange rounded rectangle with the text "SIGN IN" is positioned below the password field. To the left of the "SIGN IN" button, the text "Forgot Password" is visible. At the bottom, the text "— Sign Up —" is displayed.

## USER REGISTRATION FORM



The user registration form is titled "USER REGISTRATION" and is set against a light pink background. The form fields are as follows:

- Name: Meenu Benny
- Contact: 9806522341
- Email: meenu@gmail.com (Available)
- Address: Kanamburam house, Karukulam
- District: Ernakulam
- Place: Vazhakkulam
- Location: Chembarakkiy
- Gender: ☐ Male ☒ Female
- Photo: Choose File u88.jpg
- Password: (masked with dots)
- Confirm Password: (masked with dots)


At the bottom of the form are "Submit" and "Cancel" buttons. The "CRAFT WORLD" logo is in the top left, and navigation links (HOME, ABOUT, CONTACT, SIGN UP) are in the top right. An "Activate Windows" watermark is visible in the bottom right corner.

## SELLER REGISTRATION FORM

CRAFT WORLD HOME ABOUT CONTACT SIGN UP

### Seller Registration

Name	<input type="text" value="Rani Bobby"/>
Contact	<input type="text" value="7894561230"/>
Email	<input type="text" value="rani@gmail.com"/> <small>Available</small>
Address	<input type="text" value="Thekkayil house, Karunagappilly"/>
District	<input type="text" value="Kollam"/>
Place	<input type="text" value="Karunagappilly"/>
Gender	<input type="radio"/> Male <input checked="" type="radio"/> Female
Photo	<input type="button" value="Choose File"/> <input type="text" value="u5.jpg"/>
Password	<input type="password" value="*****"/>
Confirm Password	<input type="password" value="*****"/>
Proof	<input type="button" value="Choose File"/> <input type="text" value="images (1).jpg"/>
<input type="button" value="Submit"/> <input type="button" value="Cancel"/>	

Activate Windows 

## ADD WORK FORM

CRAFT WORLD WORKS PRIVACY BOOKINGS COMPLAINT FEEDBACK REPORTS

### Enter your work here!!!

Work Type	<input type="text" value="Painting"/>
Name	<input type="text" value="Acrylic Painting"/>
Rate	<input type="text" value="200"/>
Details	<div><div>Acrylic painting ,canvas painting for decorating house</div></div>
Image	<input type="button" value="Choose File"/> <input type="text" value="paintings.png"/>
Videos	<input type="button" value="Choose File"/> <input type="text" value="Landscape ...s (144p).mp4"/>
<input type="button" value="Save"/> <input type="button" value="Cancel"/>	

## ADD MATERIAL FORM

CRAFT WORLD WORKS PRIVACY BOOKINGS COMPLAINT FEEDBACK REPORTS

### Add Materials

Name	<input type="text" value="Red Beads"/>
Description	<input type="text" value="Craft and embroidery working che"/>
Image	<input type="button" value="Choose File"/> <input type="text" value="red.jpg"/>
Rate	<input type="text" value="100"/>
Stock	<input type="text" value="30"/>
<input type="button" value="Save"/> <input type="button" value="Cancel"/>	

## ADD FEEDBACK FORM

The screenshot shows the 'Give Feedback' form in the CRAFT WORLD application. The form is titled 'Give Feedback' and has a 'Feedback' label. The input field contains the text 'Its a good system and very helpful for buying crafts.' Below the input field are two buttons: 'Submit' and 'Cancel'. The top navigation bar includes the 'CRAFT WORLD' logo and links for 'WORKS', 'PRIVACY', 'BOOKINGS', 'COMPLAINT', 'FEEDBACK', and 'REPORTS'.

## ADD COMPLAINT FORM

The screenshot shows the 'Give Complaints' form in the CRAFT WORLD application. The form is titled 'Give Complaints' and has two input fields: 'Title' and 'Content'. The 'Title' field contains the text 'System Performance' and the 'Content' field contains the text 'System performance is very slow'. Below the input fields are two buttons: 'Submit' and 'Cancel'. The top navigation bar includes the 'CRAFT WORLD' logo and links for 'WORKS', 'PRIVACY', 'BOOKINGS', 'COMPLAINT', 'FEEDBACK', and 'REPORTS'.

## ADD RATING

The screenshot shows the 'Submit Review' modal in the CRAFT WORLD application. The modal is titled 'Submit Review' and has a close button. It features a rating system with five stars, where the first four are yellow and the fifth is grey. Below the stars is a text input field for the reviewer's name, which contains the text 'Manu'. Below the name field is a text area for the review, which contains the text 'Good and attractive'. At the bottom of the modal is a 'Submit' button. The background shows a 'Rating Box' with '0 / 5' and a 'Review' button, and a 'Write Review Here' section with a 'Review' button.

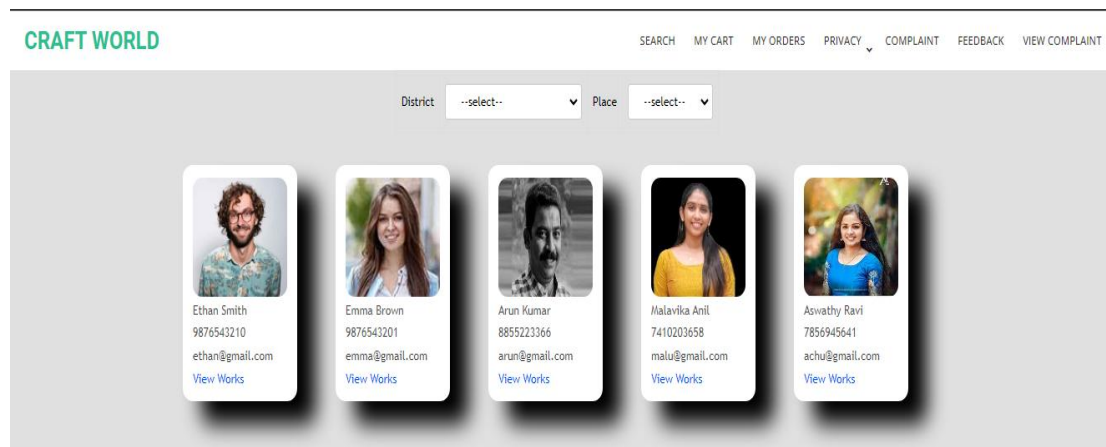
### 3.6 OUTPUT DESIGN

Computer Output is the most important and direct source of information to the user. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship and helps user decision-making.

In the output design it is determined how the implementation is to be played for immediate need and also the hard copy output. A major form of input is a hard copy from the printer. Printouts should be designed around the output requirements of the user. Printers, CRT screen display are the examples for providing computer based output. The output design associated with the system includes the various reports of table generations and query executions.

Output design is one of the, most important features of the information system. The logical design of an information system is analogous to an engineering blue print of an auto mobile. It shows the major features and how they are related to one another. The outputs, inputs and databases are designed in this phase.


### VIEW SELLERS




## VIEW MATERIALS

**CRAFT WORLD** SEARCH MY CART MY ORDERS PRIVACY COMPLAINT FEEDBACK VIEW COMPLAINT


### Materials




Thread  
₹20  
one pack  
[Add to Cart](#)  
[BuyNow](#)



Feather  
₹10  
One pack feather  
[Add to Cart](#)  
[BuyNow](#)



Thread  
₹35  
Bundle of thread  
[Add to Cart](#)  
[BuyNow](#)




Ring  
₹10  
Ring for covering threads  
[Add to Cart](#)  
[BuyNow](#)

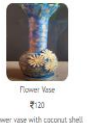
## VIEW WORKS

**CRAFT WORLD** SEARCH MY CART MY ORDERS PRIVACY COMPLAINT FEEDBACK VIEW COMPLAINT


Worktype:




Dream Catcher  
₹200  
Home decoring wall hanging dream catcher  
★★★★★  
[Add to Cart](#) [Gallery](#)  
[Tutorials](#) [Materials](#) [BuyNow](#)



Flower Vase  
₹120  
Flower vase with coconut shell  
★★★★★  
[Add to Cart](#) [Gallery](#)  
[Tutorials](#) [Materials](#) [BuyNow](#)






Beads Bracelet  
₹50  
Red and white beads bracelet  
★★★★★  
[Add to Cart](#) [Gallery](#)  
[Tutorials](#) [Materials](#) [BuyNow](#)



Deer  
₹130  
Handmade paper deer for decorating house  
★★★★★  
[Add to Cart](#) [Gallery](#)  
[Tutorials](#) [Materials](#) [BuyNow](#)

## VIEW WORK BOOKINGS

**CRAFT WORLD** WORKS PRIVACY BOOKINGS COMPLAINT FEEDBACK REPORTS

Sl.No	Product	Details	Rate	Photo	User	Contact Info	Email	Action
1	Dream Catcher	Home decoring wall hanging dream catcher	200		Isabella John	9876543120	isabella@gmail.com	Order is not completed
2	Dream Catcher	Home decoring wall hanging dream catcher	200		Manu Wilson	8520741023	manu@gmail.com	Order Delivered   <a href="#">Chat Now</a>
3	Beads Bracelet	Red and white beads bracelet	50		Manu Wilson	8520741023	manu@gmail.com	<a href="#">Order Confirm</a>   <a href="#">Order Rejected</a>   <a href="#">Chat Now</a>

## WORK BOOKING REPORT




CRAFT WORLD

[WORKS](#) [PRIVACY](#) [BOOKINGS](#) [COMPLAINT](#) [FEEDBACK](#) [REPORTS](#)

### Work Booking Report

From Date  To Date

[Show Result](#)

Sl.No	Booked Date	Product	Details	Rate	Quantity	Photo	User	Contact Info	Email	Total
1	Aug. 5, 2023	Dream Catcher	Home decoring wall hanging dream catcher	200	3		Isabella John	9876543120	isbella@gmail.com	600
2	Aug. 16, 2023	Dream Catcher	Home decoring wall hanging dream catcher	200	1		Manu Wilson	8520741023	manu@gmail.com	200
3	Aug. 19, 2023	Beads Bracelet	Red and white beads bracelet	50	2		Manu Wilson	8520741023	manu@gmail.com	100
Grand Total: 900										

Activate Windows

Go to Settings to activate Windows.

## MATERIAL BOOKING REPORT





CRAFT WORLD

[WORKS](#) [PRIVACY](#) [BOOKINGS](#) [COMPLAINT](#) [FEEDBACK](#) [REPORTS](#)

### Material Booking Report

From Date  To Date

[Show Result](#)

Sl.No	Booked Date	Material	Rate	Quantity	Photo	User	Contact Info	Email
1	Aug. 5, 2023	Thread	20	2		Sophia Marhose	9123456078	elidhosein62@gmail.com
2	Aug. 16, 2023	Feather	10	2		Manu Wilson	8520741023	manu@gmail.com
3	Aug. 16, 2023	Ring	10	1		Manu Wilson	8520741023	manu@gmail.com
4	Aug. 19, 2023	White Beads	50	1		Manu Wilson	8520741023	manu@gmail.com

Activate Windows

Go to Settings to activate Windows.

## **4. SYSTEM ENVIRONMENT**

### **4.1 INTRODUCTION**

The unique technical and operating characteristics of an IT system and its associated environment, including the hardware, software, firmware, communications capability, organization, and physical location. The system environment is primarily the set of variables that define or control certain aspects of process execution. From the system-management point of view, it is important to ensure the user is set up with the correct values at log in. Most of these variables are set during system initialization.

### **4.2 SOFTWARE REQUIREMENT SPECIFICATION**

After the analyst has collected all the required information regarding the software to be developed, and has removed all completeness, inconsistency, and anomalies from the specification, he starts to systematically organize the requirements in the form of an SRS document. The software developers refer to the SRS document to make sure that they develop exactly what the customer requires. The SRS document helps the maintenance engineers to understand the functionality of the system. An SRS document should be clearly specify;

- Functional requirements
- Non -functional requirements
- Goals of implementation

### **FUNCTIONAL REQUIREMENTS**

User Registration and Login: Users (sellers and customers) can register with their basic details and create login credentials for accessing the platform.

Admin Dashboard: Admin will have access to a personalized dashboard to manage sellers and the worktype.

Products details: Sellers can upload their works and materials including item details, quantity, and description.

Customer requirements: Customer can buy and view the details of products and materials. Also they have provision to watch the making video and rating the product.



**Search and Filter:** Customers should be able to search for sellers and products based on worktype and location.

**Booking:** Customers should be able to book products and materials from available craftsman.

**Confirmation:** Seller should receive order requests and have the option to accept or reject the order. Customers should receive email for confirmation or rejection of their bookings.

**Payment Integration:** The application should integrate with a secure payment gateway to enable online payments for products.

**Customer Reviews and Ratings:** Customers should be able to provide ratings, and reviews for products they have received.

**Chat Feature:** Chat option will be available for sellers and customers to communicate directly.

## **NON-FUNCTIONAL REQUIREMENTS**

**Usability :** The application shall have an intuitive and user-friendly interface to enhance user experience. It shall be accessible and usable on different devices and screen sizes.

**Performance :** The system shall be responsive and provide fast response times to handle concurrent user interactions efficiently. Database queries and data retrieval shall be optimized for quick processing.

**Security :** User authentication and data transmission shall be secured using encryption and authentication protocols.

**Access controls** shall be implemented to ensure data privacy and security.

**Scalability :** The application architecture shall be designed to accommodate a growing number of users and products.

**Reliability :** The system shall be robust and reliable, with minimal downtime for maintenance and updates.

**Constraints :** The project must adhere to the budget constraints set for non-profit initiatives. Development must follow ethical guidelines for handling seller and products data.

**Assumptions :** Users will have basic computer literacy and access to a stable internet connection.

Dependencies: The project is dependent on the availability of web hosting services and a database management system

### **Software Interface**

Operating system : Windows

Web Browsers : Google Chrome/Firefox/Internet Explorer

Web Design : HTML, CSS, Java Script

Front End : Python 3.10 Django

Back End : MySQL

## **4.3 HARDWARE REQUIREMENT SPECIFICATION**

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. A hardware requirements list is often accompanied by a hardware compatibility list.

The hardware compatibility lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. Selection of hardware configuration is very important task related to the software development. The processor should be powerful to handle all the operations. The hard disk should have the sufficient capacity to solve the database and the application. The minimum requirement for the implementation of the system is one machine and internet connection.

## **4.4 TOOLS, PLATFORMS**

### **4.4.1 FRONT END TOOL**

#### **PYTHON 3.10**

Python 3.10 is a major release of the Python programming language, offering various new features, optimizations, and improvements over its predecessor, Python 3.9. Released on October 4, 2021, Python 3.10 brought several enhancements that aimed to make the language more powerful, expressive, and developer-friendly.

Some key features of Python 3.10 include:

1. **Pattern Matching (PEP 634):** One of the most significant additions to Python 3.10 is the introduction of pattern matching with the `match` statement. Pattern matching allows developers to write more concise and readable code when dealing with complex conditional expressions.
2. **Parenthesized Context Managers (PEP 634):** This feature allows combining multiple context managers into a single parenthesized expression, improving code readability and organization.
3. **Parenthesized Comprehensions (PEP 612):** Python 3.10 introduced support for using parentheses in list comprehensions and generator expressions, making them more consistent with other language constructs.
4. **Precise Types (PEP 563 and PEP 586):** Python 3.10 provided more precise types for variables and function annotations, enabling better static analysis and improved type checking.
5. **New Syntax Features:** Python 3.10 introduced several syntax enhancements, including better error messages, the use of the `case` keyword in `match` statements, and improved error handling for unpacking.
6. **Performance Improvements:** As with most Python releases, Python 3.10 included performance optimizations that resulted in faster execution for certain code patterns.

Python 3.10 continued to build upon the language's strengths, emphasizing readability, simplicity, and support for various programming paradigms. Developers welcomed these new features, as they provided more powerful tools to express ideas and solve complex problems in a cleaner and more concise manner.

## **DJANGO**

Django is a high-level and open-source web framework written in Python. It was developed to help web developers build web applications rapidly and efficiently by providing a robust set of tools, libraries, and best practices. The framework follows the "don't repeat yourself" (DRY) principle and promotes clean, pragmatic design to encourage code reusability and maintainability.

Key features of Django include:

1. Object-Relational Mapping (ORM): Django's built-in ORM allows developers to interact with the database using Python classes and objects, abstracting away the need to write raw SQL queries. This makes it easier to work with databases and allows for easy switching between different database backends.

2. Admin Interface: Django provides an automatic admin interface that allows developers to manage application data with minimal effort. The admin interface is highly customizable and provides functionalities like adding, editing, and deleting records in the database without the need to build separate admin panels.

3. URL Routing and View Management: Django's URL dispatcher enables developers to map URLs to corresponding views, which are Python functions that handle the logic for processing HTTP requests and returning HTTP responses.

4. Template Engine: Django's template engine allows developers to separate the presentation layer from the business logic, making it easier to create dynamic and reusable HTML templates.

5. Forms Handling: Django simplifies form handling and validation by providing a comprehensive form library that can be easily integrated into web applications.

6. Authentication and Security: Django comes with built-in support for user authentication, permissions, and security features to protect against common web application vulnerabilities.

7. Internationalization and Localization: Django supports multiple languages and allows developers to create localized web applications with ease.

8. Scalability and Flexibility: Django is designed to scale from small projects to large, high-traffic websites. It also provides flexibility for integrating with other technologies and frameworks.

Django has gained widespread popularity in the web development community due to its userfriendly nature, extensive documentation, active community support, and its ability to handle complex web applications. It has been used to build a wide range of projects, from content management systems and social networking sites to e-commerce platforms and data-driven web applications.

By adopting Django, developers can focus more on implementing unique application features and less on repetitive tasks, thus accelerating the development process and promoting the creation of robust and maintainable web applications.

## **HTML**

HTML (Hypertext Mark-up Language) is the most basic building block of the Web. It defines the meaning and structure of web content. Other technologies besides HTML are generally used to describe a web page's appearance/presentation (CSS) or functionality/behaviour (JavaScript).

"Hypertext" refers to links that connect web pages to one another, either within a single website or between websites. Links are a fundamental aspect of the Web. By uploading content to the Internet and linking it to pages created by other people, you become an active participant in the World Wide Web. HTML uses "mark-up" to annotate text, images, and other content for display in a Web browser.

## **CASCADING STYLE SHEETS (CSS)**

CSS, short for Cascading Style Sheets, is a fundamental technology used in web development to control the presentation and layout of HTML documents. It provides a way to separate the content of a web page from its appearance, allowing developers to define styles once and apply them consistently to multiple elements throughout a website.

Key points about CSS:

1. **Styling HTML Elements:** CSS is used to define how HTML elements should be displayed on a web page. It allows developers to control aspects like font size, color, margins, padding, positioning, and more.
2. **Separation of Concerns:** One of the key principles of web development is the separation of concerns. CSS enables this by keeping the design and layout separate from the HTML structure. This separation makes it easier to update styles without affecting the underlying content.
3. **Selectors and Declarations:** CSS uses selectors to target HTML elements and declarations to specify the styles for those elements. Selectors can target elements by tag name, class, ID, or other attributes.

4. Cascading and Specificity: The term "cascading" in CSS refers to the ability of styles to flow from parent elements to their child elements. Additionally, CSS uses a specificity hierarchy to determine which styles should apply when multiple rules target the same element.

5. External Style Sheets: CSS can be written inline within HTML elements, embedded in the head section of an HTML document, or placed in external .css files. External style sheets promote code reusability and ease of maintenance across multiple pages.

6. Media Queries: CSS includes media queries that allow developers to apply different styles based on the characteristics of the user's device, such as screen size, resolution, or orientation. This enables responsive web design, optimizing the layout for different devices like desktops, tablets, and smartphones.

7. CSS Preprocessors: To enhance the capabilities of CSS, developers often use CSS preprocessors like Sass and Less. These preprocessors introduce variables, nesting, and other programming-like constructs to make CSS code more maintainable and organized.

8. Browser Compatibility: While modern web browsers generally support CSS standards, developers often need to consider cross-browser compatibility to ensure consistent rendering across different platforms. CSS plays a crucial role in shaping the visual appearance of websites and web applications. Together with HTML and JavaScript, CSS forms the backbone of modern web development, enabling developers to create appealing, user-friendly, and responsive web experiences. By mastering CSS, web developers can design visually stunning and well-structured websites that cater to diverse user preferences and device types.

## **JAVA SCRIPT**

JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.

JavaScript was first known as LiveScript, but Netscape changed its name to JavaScript, possibly because of the excitement being generated by Java. JavaScript made its first appearance in Netscape 2.0 in 1995 with the name LiveScript. The general-purpose core of the language has been embedded in Netscape, Internet Explorer, and other web browsers.

#### **4.4.2 BACK END TOOL**

##### **MySQL**

MySQL is an open-source relational database management system (RDBMS) in July 2013, it was the world's second most widely used RDBMS, and the most widely used open-source client–server model RDBMS. It is named after co-founder Michael Wideness's. The SQL acronym stands for Structured Query Language. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation for proprietary use, several paid editions are available, and offer additional functionality.

#### **4.4.3 OPERATING SYSTEM**

##### **WINDOWS**

Windows is a graphical operating system developed by Microsoft. It allows users to view and store files, run the software, play games, watch videos, and provides a way to connect to the internet. It was released for both home computing and professional works.

Microsoft introduced the first version as 1.0. It was released for both home computing and professional functions of Windows on 10 November 1983. Later, it was released on many versions of Windows as well as the current version, Windows 10.

In 1993, the first business-oriented version of Windows was released, which is known as Windows NT 3.1. Then it introduced the next versions, Windows 3.5, 4/0, and Windows 2000. When the XP Windows was released by Microsoft in 2001, the company designed its various versions for a personal and business environment. It was designed based on standard x86 hardware, like Intel and AMD processor. Accordingly, it can run on different brands of hardware, such as HP, Dell, and Sony computers, including home-built PCs.

## 5. SYSTEM IMPLEMENTATION

### 5.1 INTRODUCTION

Implementation is the final stage and it's an important phase. It involves the individual programming; system testing, user training and the operational running of developed proposed system that constitutes the application subsystems. One major task of preparing for implementation is education of users, which should really have been taken place much earlier in the project when they were being involved in the investigation and design work.

During the implementation phase system actually takes physical shape. In order to develop a system implemented, planning is very essential. The implementation phase of the software development is concerned with translating design specifications in to source code. The implementation phase ends with an evaluation of the system after placing it into operation for a period of time. Implementation is the third phase of the system process. In order to achieve the objectives and the expected performance the system has been developed in a highly interactive and user-friendly manner.

#### Implementation Plan

The following are the steps involved in the implementation plan:

- Test system with sample data
- Detection and correction of errors
- Make the necessary changes in the system
- Check with the existing system
- Installation of hardware and software utilities
- Training and involvement of user person

#### Training :

To put training program into effect according to definite plan or procedure is called training implementation. Training implementation is the hardest part of the system because one wrong



step can lead to the failure of whole training program. Even the best training program will fail due to one wrong action.

**Conversion required :**

Conversion and installation is the process of upgrading or replacing the existing system with the new system. This includes not only the software and procedures of the new system, but also any changes or improvements to the IT infrastructure. The installation of a new system may require new networks and new hardware. During the conversion and installation process, all the requirements must be taken into account.

**5.2 CODING**

Coding, in simpler terms, is the language used by computers to understand our commands and, therefore, process our requests. It means using the programming language to get the computer to behave as desired. Each line of the code is a set of instructions for the computer. A set of codes form a script, and a set or dozens of sets, form a program.

**5.2.1. CODE STANDARDS**

Different modules specified in the design document are coded in the coding phase according to the module specifications. The main goal of the coding phase is to code from the design document prepared after the design phase through a high-level language and then to unit test the code.

Good software development organizations want their programmers to maintain a well-defined and Standard style of coding called coding standards. They usually make their own coding standards and guidelines depending on what suits their organization the best and based on the types of software they develop. It is very important for the programmers to maintain coding standards, otherwise the code will be rejected during code review.

**Purpose of Having Coding Standards:**

- Coding standards give a uniform appearance to the codes written by different engineers.
- It improves readability and maintainability of the code and it reduces complexity also.
- It helps in code reuse and help to detect error easily.
- It promotes sound programming practices and increases efficiency of the programmers.

**5.2.1 SAMPLE CODES****guest\_views.py**

```

from django.shortcuts import render, redirect

from Admin.models import *

from Guest.models import *

import random

from django.conf import settings #import project settings.py here

from django.core.mail import send_mail #importing send_mail function from djang

# Create your views here.


def user_registration(request):

    districtdata=tbl_district.objects.all()

    placedata=tbl_district.objects.all()

    if request.method=="POST":

        lo=tbl_location.objects.get(id=request.POST.get('loc'))

        tbl_user.objects.create(name=request.POST.get('uname'),

        contact=request.POST.get('contact'),

        email=request.POST.get('email'),

        address=request.POST.get('address'),

        gender=request.POST.get('gender'),

        photo=request.FILES.get('photo'),

        password=request.POST.get('passwd'),location=lo)

        ur=1

        return render(request,'Guest/userregistration.html',{'district':districtdata,'ur':ur})

```

```

else:

    return render(request,'Guest/userregistration.html',{'district':districtdata})

return render(request,"Guest/userregistration.html",{'district':districtdata})

def seller_registration(request):

    districtdata=tbl_district.objects.all()

    placedata=tbl_place.objects.all()

    if request.method=="POST":

        if request.POST.get('pass')==request.POST.get('cpasswd'):

            place=tbl_place.objects.get(id=request.POST.get('place'))

            tbl_seller.objects.create(name=request.POST.get('sname'),

            contact=request.POST.get('contact'),

            email=request.POST.get('email'),

            address=request.POST.get('address'),

            district=request.POST.get('district'),

            gender=request.POST.get('gender'),

            photo=request.FILES.get('photo'),

            password=request.POST.get('pass'),

            proof=request.FILES.get('proof'),place=place)

            sr=1

            return render(request,'Guest/sellerregistration.html',{'district':districtdata,'sr':sr})

        else:

            return render(request,'Guest/sellerregistration.html',{'district':districtdata})

    else:

        return render(request,'Guest/sellerregistration.html',{'district':districtdata})

```

```

return render(request,"Guest/sellerregistration.html",{ 'district':districtdata})

def Ajaxplace(request):

    districtdata=tbl_district.objects.get(id=request.GET.get('disd'))

    placedata=tbl_place.objects.filter(district=districtdata)

    return render(request,"Guest/Ajaxplace.html", {'place':placedata})

def Ajaxloc(request):

    districtdata=tbl_place.objects.get(id=request.GET.get('disd'))

    placedata=tbl_location.objects.filter(place=districtdata)

    return render(request,"Guest/Ajaxlocation.html", {'place':placedata})

def login(request):

    if request.method=="POST":

        usercount=tbl_user.objects.filter(email=request.POST.get('email'),password=

        request.POST.get('passwd')).count()

        sellercount=tbl_seller.objects.filter(email=request.POST.get('email'),password=request.POS

        T.get('passwd'),status=1).count()

        admincount=tbl_adminlogin.objects.filter(email=request.POST.get('email'),password=reque

        st.POST.get('passwd')).count()

        print(usercount)

        if usercount>0:

            userdata=tbl_user.objects.get(email=request.POST.get('email'),password=request.PO

            ST.get('passwd'))

            request.session['uid']=userdata.id

            return redirect("User:userhome")

        elif sellercount>0:

            sellerdata=tbl_seller.objects.get(email=request.POST.get('email'),

```

```

        password=request.POST.get('passwd'),status=1)

    request.session['sid']=sellerdata.id

    return redirect("Seller:sellerhome")

elif admincount>0:
    admindata=tbl_adminlogin.objects.get(email=request.POST.get('email'),password=request.PO
ST.get('passwd'))

    request.session['aid']=admindata.id

    return redirect("Admin:adminhome")

else:

    lo=1

    return render(request,"Guest/Login.html",{ 'lo':lo})

else:

    return render(request,"Guest/Login.html")

def ForgetPassword(request):

    if request.method=="POST":

        otp=random.randint(10000, 999999)

        request.session["otp"]=otp

        request.session["femail"]=request.POST.get('txt_email')

        send_mail(

            'Respected Sir/Madam ',#subject

            "\rYour OTP for Reset Password Is"+str(otp),#body

            settings.EMAIL_HOST_USER,

            [request.POST.get('txt_email')],

            )

```

```

        return redirect("Guest:OtpVerification")

    else:

        return render(request,"Guest/ForgotPassword.html")

def OtpVerification(request):

    if request.method=="POST":

        otp=int(request.session["otp"])

        if int(request.POST.get('txt_otp'))==otp:

            return redirect("Guest:CreateNewPass")

        return render(request,"Guest/ValidateOTP.html")

def CreateNewPass(request):

    if request.method=="POST":

        if request.POST.get('txt_pass')==request.POST.get('txt_confirm'):

            usercount=tbl_user.objects.filter(email=request.session["femail"]).count()

            sellercount=tbl_seller.objects.filter(email=request.session["femail"]).count()

            admincountcount=tbl_adminlogin.objects.filter(email=request.session["femail"]).count()

            ()

            if usercount>0:

                user=tbl_user.objects.get(email=request.session["femail"])

                user.password=request.POST.get('txt_pass')

                user.save()

                p=1

            return render(request,"Guest/CreatePassword.html",{p:p})

            elif sellercount>0:

                doc=tbl_seller.objects.get(email=request.session["femail"])

                doc.password=request.POST.get('txt_pass')

```

```

        doc.save()

        p=1

        return render(request,"Guest/CreatePassword.html",{ 'p':p})

    else:

        adminind=tbl_adminlogin.objects.get(email=request.session["femail"])

        adminind.password=request.POST.get('txt_pass')

        adminind.save()

        p=1

        return render(request,"Guest/CreatePassword.html",{ 'p':p})

    else:

        m=1

        return render(request,"Guest/CreatePassword.html",{ 'm':m})

    else:

        return render(request,"Guest/CreatePassword.html")

def home(request):

    return render(request,"Guest/Home.html")

def contact(request):

    return render(request,"Guest/Contact.html")

def ajaxemail(request):

    usercount=tbl_user.objects.filter(email=request.GET.get("email")).count()

    marketcount=tbl_seller.objects.filter(email=request.GET.get("email")).count()

    farmercount=tbl_adminlogin.objects.filter(email=request.GET.get("email")).count()

    if usercount>0 or marketcount>0 or farmercount>0:

        return render(request,"Guest/Ajaxemail.html",{ 'mess':1})

```

```
else:
```

```
    return render(request,"Guest/Ajaxemail.html")
```

### **seller\_views.py**

```
from django.shortcuts import render,redirect
```

```
from Admin.models import *
```

```
from Guest.models import *
```

```
from Seller.models import *
```

```
from User.models import *
```

```
from django.conf import settings #import project settings.py here
```

```
from django.core.mail import send_mail #importing send_mail function from django
```

```
from datetime import datetime,timedelta
```

```
def seller_home(request):
```

```
    if 'sid' in request.session:
```

```
        return render(request,'Seller/SellerHome.html')
```

```
    else:
```

```
        return redirect("Guest:login")
```

```
def editprofile(request):
```

```
    if 'sid' in request.session:
```

```
        data=tbl_seller.objects.get(id=request.session['sid'])
```

```
        if request.method=="POST":
```

```
            data.name=request.POST.get('name')
```

```
            data.contact=request.POST.get('contact')
```

```
            data.address=request.POST.get('address')
```

```
            data.save()
```



```

        return redirect("Seller:mypr")

        return render(request,'Seller/Editprofile.html',{'data':data})

    else:

        return redirect("Guest:login")

def changepass(request):

    if 'sid' in request.session:

        if request.method=="POST":

sellercount=tbl_seller.objects.filter(id=request.session["sid"],password=request.POST.get('cpa
ss')).count()

        if sellercount>0:

            if request.POST.get('npass')==request.POST.get('pass'):

                sellerdata=tbl_seller.objects.get(id=request.session["sid"])

                sellerdata.password=request.POST.get('npass')

                sellerdata.save()

                return redirect("Seller:sellerhome")

            else:

                return render(request,'Seller/Changepassword.html')

        else:

            return render(request,'Seller/Changepassword.html')

        else:

            return render(request,'Seller/Changepassword.html')

    else:

        return redirect("Guest:login")

def myprofile(request):

```

```
if 'sid' in request.session:

    data=tbl_seller.objects.get(id=request.session['sid'])

    return render(request,'Seller/MyProfile.html',{'data':data})

else:

    return redirect("Guest:login")

def works(request):

    ar=[1,2,3,4,5]

    parry=[]

    avg=0

    if 'sid' in request.session:

        workdata=tbl_worktype.objects.all()

        sellerdata=tbl_seller.objects.get(id=request.session['sid'])

        worksdata=tbl_works.objects.filter(seller=sellerdata)

        for i in worksdata:

            wdata=tbl_works.objects.get(id=i.id)

            tot=0

            ratecount=star.objects.filter(work=wdata).count()

            if ratecount>0:

                ratedata=star.objects.filter(work=wdata)

                for j in ratedata:

                    tot=tot+j.rating_data

                avg=tot//ratecount

                parry.append(avg)

            else:
```

```

        parry.append(0)

    datas=zip(worksdata,parry)

    if request.method=="POST":

        worktypedata=tbl_worktype.objects.get(id=request.POST.get('work'))

        tbl_works.objects.create(seller=sellerdata,worktype=worktypedata,

        name=request.POST.get('name'),

        rate=request.POST.get('rate'),

        details=request.POST.get('details'),

        image=request.FILES.get('image'),

        video=request.FILES.get('video'))

        worksdata=tbl_works.objects.filter(seller=sellerdata)

        for i in worksdata:

            wdata=tbl_works.objects.get(id=i.id)

            tot=0

            ratecount=star.objects.filter(work=wdata).count()

            if ratecount>0:

                ratedata=star.objects.filter(work=wdata)

                for j in ratedata:

                    tot=tot+j.rating_data

                avg=tot//ratecount

                parry.append(avg)

            else:

                parry.append(0)

        datas=zip(worksdata,parry)

```

```

        return render(request,'Seller/Works.html',{'work':workdata,'seller':datas,'ar':ar})

    else:

        return render(request,'Seller/Works.html',{'work':workdata,'seller':datas,'ar':ar})

    else:

        return redirect("Guest:login")

def deletework(request,wid):

    tbl_works.objects.get(id=wid).delete()

    return redirect("Seller:works")

def wgallery(request,did):

    if 'sid' in request.session:

        worksdata=tbl_works.objects.get(id=did)

        if request.method=="POST":

            imagevalues=request.FILES.getlist('gimage')

            for i in imagevalues:

                tbl_wgallery.objects.create(work=worksdata,image=i)

            return redirect("Seller:works")

        else:

            return render(request,'Seller/Workgallery.html')

    else:

        return redirect("Guest:login")

def materials(request,wid):

    if 'sid' in request.session:

        workdata=tbl_works.objects.get(id=wid)

        mdata=tbl_material.objects.filter(work=workdata)

```

```
if request.method=="POST":

    tbl_material.objects.create(work=workdata,

    name=request.POST.get('name'),

    image=request.FILES.get('image'),

    rate=request.POST.get('rate'),

    stock=request.POST.get('stock'),)

    return redirect("Seller:works")

else:

    return render(request,'Seller/Materials.html',{'data':mdata})

else:

    return redirect("Guest:login")

def userbooking(request):

    if 'sid' in request.session:

        slr=tbl_seller.objects.get(id=request.session["sid"])

        data=tbl_wcart.objects.filter(works__seller=slr)

        return render(request,"Seller/Userbooking.html',{'data':data})

    else:

        return redirect("Guest:login")

def materialbooking(request):

    if 'sid' in request.session:

        slr=tbl_seller.objects.get(id=request.session["sid"])

        mdata=tbl_mcart.objects.filter(material__work__seller=slr)

        return render(request,"Seller/Materialbooking.html',{'data':mdata})
```

```
    else:

        return redirect("Guest:login")

def updatestock(request,mid):

    if 'sid' in request.session:

        if request.method=="POST":

            sdata=tbl_material.objects.get(id=mid)

            ustock=request.POST.get('upstock')

            stock=int(sdata.stock)

            newstock=int(stock)+int(ustock)

            sdata.stock=newstock

            sdata.save()

            return redirect("Seller:works")

        else:

            return render(request,"Seller/Stock.html")

    else:

        return redirect("Guest:login")

def rejectorder(request,rid):

    data=tbl_mcart.objects.get(id=rid)

    bid=data.mbooking.id

    mbdata=tbl_mbooking.objects.get(id=bid)

    name=mbdata.user.name

    email=mbdata.user.email

    if request.method=="POST":

        send_mail(
```

```
'Respected Sir/Madam '+name,#subject

request.POST.get("txtreply"),#body

settings.EMAIL_HOST_USER,

[email],

)

data.cstatus=9

data.save()

return redirect("Seller:user_booking")

else:

    return render(request,"Seller/RejectReply.html")

def rejectwork(request,rrid):

    data=tbl_wcart.objects.get(id=rrid)

    bdata=tbl_wbooking.objects.get(id=data.booking.id)

    name=bdata.user.name

    email=bdata.user.email

    if request.method=="POST":

        send_mail(

            'Respected Sir/Madam '+name,#subject

            request.POST.get("txtreply"),#body

            settings.EMAIL_HOST_USER,

            [email],

        )

        data.cstatus=4

        data.save()
```

```
        return redirect("Seller:user_booking")

    else:

        return render(request,"Seller/RejectReply.html")

def acceptorder(request,aid):

    data=tbl_mcart.objects.get(id=aid)

    bid=data.mbooking.id

    mbddata=tbl_mbooking.objects.get(id=bid)

    name=mbddata.user.name

    email=mbddata.user.email

    if request.method=="POST":

        send_mail(

            'Respected Sir/Madam '+name,#subject

            request.POST.get("txtreply"),#body

            settings.EMAIL_HOST_USER,

            [email],

        )

        data.cstatus=1

        data.save()

        return redirect("Seller:user_booking")

    else:

        return render(request,"Seller/AcceptReply.html")

def acceptwork(request,aaid):

    wdata=tbl_wcart.objects.get(id=aaid)

    bdata=tbl_wbooking.objects.get(id=wdata.booking.id)
```



```
name=bdata.user.name

email=bdata.user.email

if request.method=="POST":

    send_mail(

        'Respected Sir/Madam '+name,#subject

        request.POST.get("txtreply"),#body

        settings.EMAIL_HOST_USER,

        [email],

    )

    wdata.cstatus=3

    wdata.save()

    return redirect("Seller:user_booking")

else:

    return render(request,"Seller/AcceptReply.html")

def itempacked(request,pid):

    data=tbl_wcart.objects.get(id=pid)

    data.cstatus=5

    data.save()

    return redirect("Seller:user_booking")

def itemshipped(request,shid):

    data=tbl_wcart.objects.get(id=shid)

    data.cstatus=6

    data.save()

    return redirect("Seller:user_booking")
```

```
def itemdispatched(request,did):

    data=tbl_wcart.objects.get(id=did)

    data.cstatus=7

    data.save()

    return redirect("Seller:user_booking")

def delivered(request,ddid):

    data=tbl_wcart.objects.get(id=ddid)

    data.cstatus=8

    data.save()

    return redirect("Seller:user_booking")

# material

def mitempacked(request,pid):

    data=tbl_mcart.objects.get(id=pid)

    data.cstatus=2

    data.save()

    return redirect("Seller:materialbooking")

def mitemshipped(request,shid):

    data=tbl_mcart.objects.get(id=shid)

    data.cstatus=3

    data.save()

    return redirect("Seller:materialbooking")

def mitemdispatched(request,did):

    data=tbl_mcart.objects.get(id=did)

    data.cstatus=4
```

```

    data.save()

    return redirect("Seller:materialbooking")

def mdelivered(request,ddid):

    data=tbl_mcart.objects.get(id=ddid)

    data.cstatus=5

    data.save()

    return redirect("Seller:materialbooking")

def workreport(request):

    if 'sid' in request.session:

        total=0

        slr=tbl_seller.objects.get(id=request.session["sid"])

        # mdata=tbl_wcart.objects.filter(works__seller=slr)

        if request.method == "POST":

            if request.POST.get('fdate')!="" and request.POST.get('edate')!="":

                data1=tbl_wcart.objects.filter(booking__date__gt=request.POST.get('fdate'),booking__date__lt=request.POST.get('edate'),works__seller=slr)

                for i in data1:

                    total=total+(int(i.qty)*int(i.works.rate))

                return render(request,"Seller/WorkBookingReport.html",{ 'data1':data1,'total':total})

            elif request.POST.get('fdate')!="" and request.POST.get('edate')=="":

                data2=tbl_wcart.objects.filter(booking__date__gt=request.POST.get('fdate'),works__seller=slr)

                for i in data2:

                    total=total+(int(i.qty)*int(i.works.rate))

                return render(request,"Seller/WorkBookingReport.html",{ 'data1':data2,'total':total})

```

```

        elif request.POST.get('fdate') == "" and request.POST.get('edate') != "":
data3=tbl_wcart.objects.filter(booking__date__lt=request.POST.get('edate'),works__seller=slr)

        for i in data3:

            total=total+(int(i.qty)*int(i.works.rate))

        return render(request,"Seller/WorkBookingReport.html",{'data1':data3,'total':total})

    else:

        return render(request,"Seller/WorkBookingReport.html")

    else:

        return render(request,"Seller/WorkBookingReport.html")

    else:

        return redirect("Guest:login")

def materialreport(request):

    if 'sid' in request.session:

        slr=tbl_seller.objects.get(id=request.session["sid"])

        #mdata=tbl_mcart.objects.filter(material__work__seller=slr)

        if request.method == "POST":

            if request.POST.get('fdate') != "" and request.POST.get('edate') != "":

data1=tbl_mcart.objects.filter(mbooking__date__gt=request.POST.get('fdate'),mbooking__date__lt=request.POST.get('edate'),material__work__seller=slr)

                return render(request,"Seller/MaterialBookingReport.html",{'data':data1})

            elif request.POST.get('fdate') != "" and request.POST.get('edate') == "":

data2=tbl_mcart.objects.filter(mbooking__date__gt=request.POST.get('fdate'),material__work__seller=slr)

```

```

        return render(request,"Seller/MaterialBookingReport.html",{ 'data':data2})

        elif request.POST.get('fdate')=="" and request.POST.get('edate')!="":

data3=tbl_mcart.objects.filter(mbooking__date__lt=request.POST.get('edate'),material__work
__seller=slr)

        return render(request,"Seller/MaterialBookingReport.html",{ 'data':data3})

    else:

        return render(request,"Seller/MaterialBookingReport.html")

    else:

        return render(request,"Seller/MaterialBookingReport.html")

    else:

        return redirect("Guest:login")

def viewreason(request,mid):

    data=tbl_mcart.objects.get(id=mid)

    rdata=tbl_return.objects.get(cart=data)

    return render(request,"Seller/ViewReason.html",{ 'data':rdata})

def verifyreason(request,rid):

    data=tbl_return.objects.get(id=rid)

    ddata=tbl_mcart.objects.get(id=data.cart.id)

    ddata.cstatus=8

    ddata.save()

    return redirect("Seller:materialbooking")

    #product return reason

def wviewreason(request,wid):

    data=tbl_wcart.objects.get(id=wid)

```

```

    rdata=tbl_return.objects.get(wcart=data)

    return render(request,"Seller/ViewReason.html",{ 'data':rdata,'ms':1 })

def wverifyreason(request,pid):

    data=tbl_return.objects.get(id=pid)

    ddata=tbl_wcart.objects.get(id=data.wcart.id)

    ddata.cstatus=10

    ddata.save()

    return redirect("Seller:user_booking")

def sellercomplaint(request):

    if 'sid' in request.session:

        slr=tbl_seller.objects.get(id=request.session['sid'])

        #usr=tbl_user.objects.get(id=request.session['uid'])

        if request.method == "POST":

            tbl_complaint.objects.create(seller=slr,

            title=request.POST.get('txttitle'),

            content=request.POST.get('txtcontent'),

            reply="Not at Viewed")

            return redirect("Seller:sellerhome")

        else:

            return render(request,"Seller/Complaints.html")

    else:

        return redirect("Guest:login")

def sellerfeedback(request):

    if 'sid' in request.session:

```

```
slr=tbl_seller.objects.get(id=request.session['sid'])

#usr=tbl_user.objects.get(id=request.session['uid'])

if request.method == "POST":

    tbl_feedback.objects.create(seller=slr,

    content=request.POST.get('txtcontent'))

    return redirect("Seller:sellerhome")

else:

    return render(request,"Seller/Feedbacks.html")

else:

    return redirect("Guest:login")

def logout(request):

    del request.session["sid"]

    return redirect("Guest:login")

def chatuser(request, cid):

    chatobj = tbl_wbooking.objects.get(id=cid)

    if request.method == "POST":

        cied = request.POST.get("cid")

        # print(cied)

        ciedobj = tbl_user.objects.get(id=cied)

        sobj = tbl_seller.objects.get(id=request.session["sid"])

        content = request.POST.get("msg")

        # print(cied)

        # print(content)

        Chat.objects.create(
```

```

        from_seller=sobj,        to_user=ciedobj,        content=content,        from_user=None,
to_seller=None)

        return render(request, 'Seller/Chat.html', {"chatobj": chatobj})

    else:

        return render(request, 'Seller/Chat.html', {"chatobj": chatobj})

def loadchatuser(request):

    cid = request.GET.get("cid")

    request.session["cid"] = cid

    cid1 = request.session["cid"]

    # print(cid1)

    # print(cid)

    shopobj = tbl_user.objects.get(id=cid)

    # print(userobj)

    sid = request.session["sid"]

    # print(sid)

    suserobj = tbl_seller.objects.get(id=request.session["sid"])

    chatobj = Chat.objects.raw(

        "select * from User_chat c inner join Guest_tbl_seller u on (u.id=c.from_seller_id) or
(u.id=c.to_seller_id) WHERE  c.from_user_id=%s or c.to_user_id=%s order by c.date",
params=[(cid1), (cid1)])

    print(chatobj.query)

    return render(request, 'Seller/Load.html', {"obj": chatobj, "sid": sid, "shop": shopobj,
"userobj": suserobj})

def complaint_reply(request):

```



```

if 'sid' in request.session:

    slr=tbl_seller.objects.get(id=request.session['sid'])

    cdata=tbl_complaint.objects.filter(seller=slr)

    return render(request,"Seller/ViewComplaint.html",{ 'data':cdata})

else:

    return redirect("Guest:login")

def videoreport(request):

    sellerdata=tbl_seller.objects.get(id=request.session["sid"])

    data=tbl_videopay.objects.filter(seller=sellerdata)

    return render(request,"Seller/VideopaymentReport.html",{ 'data':data})

```

### Userregistration.html

```

{% extends 'Guest/Head.html' %}

{% load static %}

{% block content %}

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body style="background-color: rgb(234, 222, 222);">

    <center><h1>USER REGISTRATION</h1></center>

```

```

<div id="tab">

  <form action="" method="post" id="userform" enctype="multipart/form-data" data-place-
url="{% url 'Guest:Ajxplace' %}" data-location-url="{% url 'Guest:Ajxloc' %}">

    {% csrf_token %}

    <table border="1" align="center" cellpadding="10" style="border-collapse: collapse;">

      <tr>

        <td>Name</td>

        <td><input type="text" name="uname" id="" autocomplete="off" class="form-
control" required="" onchange="nameval(this)"><span id="name"></span></td>

      </tr>

      <tr>

        <td>Contact</td>

        <td><input type="text" name="contact" id="" autocomplete="off" class="form-
control" required="" onchange="checknum(this)"><span id="contact"></span></td>

      </tr>

      <tr>

        <td>Email</td>

        <td><input type="email" name="email" id="" autocomplete="off" class="form-
control" required="" onchange="emailval(this),chemail(this.value)"><span
id="content"></span><span id="content1"></span></td>

      </tr>

      <tr>

        <td>Address</td>

        <td><input type="text" name="address" id="" autocomplete="off" class="form-
control" required=""></td>

      </tr>

```

```

<tr>

<td>District</td>

<td><select name="district" id="district" required="" class="form-control">

    <option value="">--select district--</option>

    {% for i in district %}

    <option value="{{ i.id }}">{{ i.district_name }}</option>

    {% endfor %}

</select></td>

</tr>

<tr>

<td>Place</td>

<td><select name="place" id="place" required="" class="form-control">

    <option value="">--select place--</option>

    {% for i in place %}

    <option value="{{ i.id }}">{{ i.place_name }}</option>

    {% endfor %}

</select></td>

</tr>

<tr>

<td>Location</td>

<td><select name="loc" id="loc" required="" class="form-control">

    <option value="">--select place--</option>

</select></td>

</tr>

```

```

        <tr>

            <td>Gender</td>

            <td><input type="radio" name="gender" id="" required="">Male

                <input type="radio" name="gender" id="" required="">Female</td>

        </tr>

        <tr>

            <td>Photo</td>

            <td><input type="file" name="photo" id="" autocomplete="off" class="form-
control" required=""></td>

        </tr>

        <tr>

            <td>Password</td>

            <td><input type="password" name="passwd" id="" autocomplete="off"
class="form-control" required=""></td>

        </tr>

        <tr>

            <td>Confirm Password</td>

            <td><input type="password" name="cpasswd" id="" autocomplete="off"
class="form-control" required="" onchange="chkpwd(this,passwd)"><span
id="pass"></span></td>

        </tr>

        <tr>

            <td colspan="2" align="center"><input type="submit" value="Submit"
name="register">

                <input type="reset" value="Cancel"></td>

```

```

        </tr>

    </table>

</form>

</div>

</body>

</html>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<script>

    $("#district").change(function () {

        var did = $(this).val();

        var ur = $("#userform").attr("data-place-url");

        $.ajax({

            url: ur,

            data: { disd: did, },

            success: function (data) {

                $("#place").html(data);

            },

        });

    });

    $("#place").change(function () {

        var did = $(this).val();

        var ur = $("#userform").attr("data-location-url");

        $.ajax({

            url: ur,

```

```
        data: { disd: did, },
        success: function (data) {
            $("#loc").html(data);
        },
    });
});
</script>
<script>

    function chemail(elemvalue)
    {
        //alert(elemvalue);

        $.ajax({
            url: '/ajaxemail/',
            data: { email: elemvalue, },
            success: function (data) {
                $("#content1").html(data);
            },
        });
    }

    function chkpwd(txtrp, tttp)
    {
        if((txtrp.value)!=(tttp.value))
        {
```

```

document.getElementById("pass").innerHTML = "<span style='color: red;'>Passwords
Mismatch</span>";

}

}

function checknum(elem)
{
var numericExpression = /^[0-9]{10,10}$/;

if(elem.value.match(numericExpression))
{
document.getElementById("contact").innerHTML = "";

return true;

}

else

{

document.getElementById("contact").innerHTML = "<span style='color: red;'>Ten digits and
Numbers Only Allowed</span>";

elem.focus();

return false;

}

}

function emailval(elem)
{

var emailexp=/^([a-zA-Z0-9.\_\-])+\@([a-zA-Z0-9.\_\-])+\.[a-zA-Z]{2,4}$/;

if(elem.value.match(emailexp))
{

```

```
document.getElementById("content").innerHTML = "";

return true;

}

else

{

document.getElementById("content").innerHTML = "<span style='color: red;'>Check Email Address Entered</span>";

elem.focus();

return false;

}

}

function nameval(elem)

{

var emailxp=/^([A-Za-z ]*)$/;

if(elem.value.match(emailxp))

{

document.getElementById("name").innerHTML = "";

return true;

}

else

{

document.getElementById("name").innerHTML = "<span style='color: red;'>Alphabets Are Allowed</span>";

elem.focus();

return false;
```



```

}

}

</script>

{% if ur %}

    <script>

        alert("Registered successfully...")

        window.location="/login/"

    </script>

{% endif %}

{% endblock content %}

```

### 5.3. UNIT TESTING

Unit testing involves the testing of each unit or an individual component of the software application. It is the first level of functional testing. The aim behind unit testing is to validate unit components with its performance. A unit is a single testable part of a software system and tested during the development phase of the application software. The purpose of unit testing is to test the correctness of isolated code, to fix bugs early in the development cycle and to save costs, to help the developers to understand the code base and enable them to make changes quickly.

Unit is the smallest testable part of software. Unit testing is used to validate that individual units of source code are working properly. In object-oriented programming, the smallest unit is a method and it contain a base/super class, abstract class or derived/child class but in procedural programming language a unit may be an individual program, function, procedure, etc., while The main advantage of the unit testing is used to improve the quality of code and save the tester's time and effort.

A test plan and test cases are crucial components of the software testing process, helping to ensure that a software product meets the specified requirements and functions as expected. They outline the approach, scope, and details of the testing process, including the test objectives, test scope, test environment, and test procedures.

Test Plan:

A test plan is a comprehensive document that outlines the overall testing strategy for a software project. It acts as a roadmap for the testing activities and provides a systematic approach to ensure that all aspects of the software are thoroughly tested. A typical test plan includes the following components:

1. **Test Objectives:** Clearly defined objectives and goals of the testing effort. It outlines what needs to be achieved through testing, such as verifying specific functionalities or meeting certain quality criteria.
2. **Scope:** The scope defines what will and will not be tested. It includes information about the features or modules that will be covered in the testing process.
3. **Test Strategy:** The test strategy describes the overall approach to testing, including the testing levels (unit testing, integration testing, system testing, etc.), testing types (functional, performance, security, etc.), and any specific methodologies or tools to be used.
4. **Test Environment:** Details about the hardware, software, and network configurations required for testing. It ensures that the testing environment replicates the production environment as closely as possible.
5. **Test Deliverables:** A list of the artifacts or documents that will be produced during testing, such as test cases, test scripts, test reports, and defect reports.
6. **Test Schedule:** A timeline outlining the testing phases and milestones, including the start and end dates of testing activities.
7. **Risks and Contingencies:** Identification of potential risks and their associated mitigation strategies to address any unforeseen issues during testing.

Test Cases:

Test cases are specific detailed steps or scenarios designed to verify that the software functions correctly. Each test case focuses on a particular aspect of the software's functionality and includes preconditions, inputs, expected outcomes, and post conditions. Test cases cover various scenarios, including typical cases, boundary cases, error conditions, and negative scenarios.

Key elements of a test case include:

1. Test Case ID: A unique identifier for each test case.
2. Test Description: A brief description of what the test is intended to validate.
3. Test Steps: Step-by-step instructions on how to execute the test.
4. Test Data: Input data needed to execute the test.
5. Expected Result: The expected outcome or behaviour after executing the test.
6. Actual Result: The actual outcome or behaviour observed during testing.
7. Pass/Fail Status: Indicates whether the test passed or failed.
8. Defects: Any issues or defects discovered during testing.

Test cases are essential for both manual and automated testing, and they serve as a basis for test execution, defect reporting, and regression testing.

Overall, the test plan and test cases together ensure that the software is thoroughly tested, meets the specified requirements, and delivers a high-quality product to end-users.

### 5.3.1 TEST PLAN AND TEST CASES

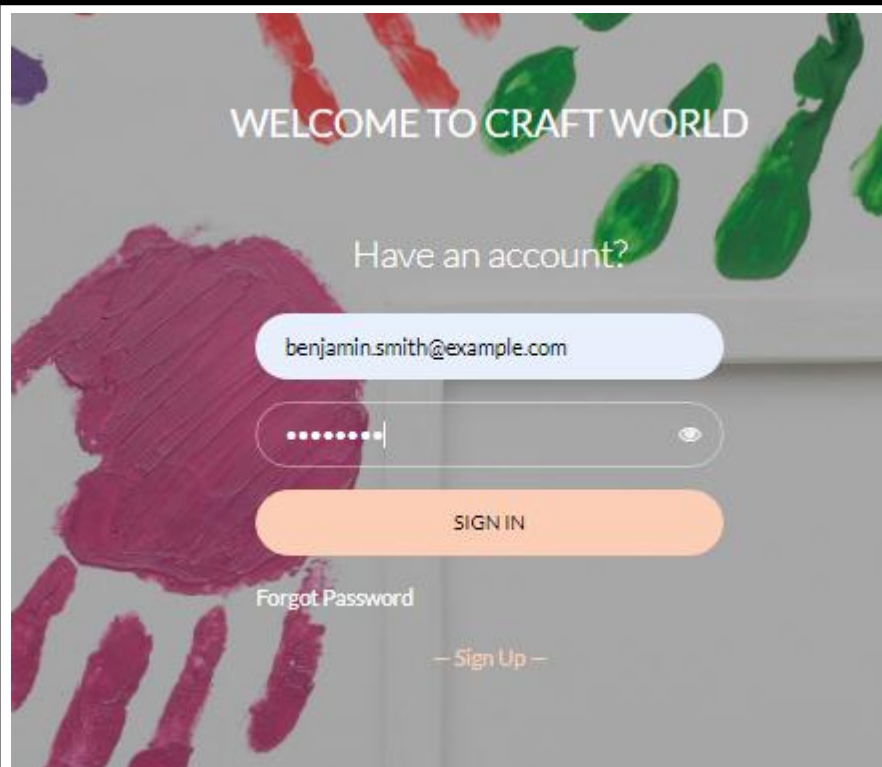
Test Case Id	Test Case	Input Data	Expected Result	Actual Result
TC01	Customer Login with Valid credentials	Email:amal@gmail.com Password: Amal	Customers should be successfully logged into the system.	Pass

TC02	Customer login with Invalid credentials	Email: amal@gmail.com  Password:Aml	The system should display an error message indicating that the entered credentials are incorrect.	Pass
TC0 3	Admin Login with Valid Credentials	Email: admin@gmail.com  Password: Admin123	Admin should be successfully logged into the system.	Pass
TC04	Admin login with Invalid credentials	Email: admin@example.co m  Password: Adm123	The system should display an error message indicating that the entered credentials are incorrect.	Pass
TC05	Seller Login  with Valid credentials	Email: aswanth@gmail.co m  Password: Aswanth	Selllers should be successfully logged into the system.	Pass
TC06	Seller login  With Invalid credentials	Email: aswanth@gmail.co m  Password:Aswanth123	The system should display an error message indicating that the entered credentials are incorrect.	Pass

TC07	Customer successful registration	Name: Seethu Email: seethu@gmail.com Address: Pilappilly myalil Place: Thiruvankulam Phone: 9539468447 Password: Seethu	A customer account should be successfully created, and the platform should redirect to the home page	Pass
TC08	Customer registering with Existing email address	Name: Geethu Email: seethu@gmail.com Address: Valyodil Phone: 9539468447 Password: Geethu Place: Thiruvankulam	The platform should display an error message indicating that the entered phone already exists, prompting the user to provide a different phone number.	Pass

TC09	Customer registration with missing required information	Name: Geethu Email: geethu@gmail.com Address: Valyodil Phone: 9539468447 Password: Geethu Place:	The platform should display an error message indicating that all the required information must be provided, prompting the user to fill in the missing fields	Pass
TC10	Seller successfully adding product	Name: Beads chain Worktype: Jewellery Rate: 200 Description: Simple beads chain	A new work will be successfully added.	Pass
TC11	Seller add new product with missing required information	Name: Beads chain Worktype: Jewellery Price: 200 Description:	The platform should display an error message indicating that all the required information must be provided, prompting the user to fill in the missing fields	Pass

TC12	The customer enters and sends the return details with valid data	Product name:  Reason for return:	The platform should display a message indicating the returning request has been successfully done	Pass
TC13	The customer enters and sends the booking details with valid data	Product Name: Beads chain  quantity: 2	The platform should display a message indicating the booking request has been successfully done	Pass
TC14	The customer enters and sends the booking details missing mandatory fields	Product Name:  quantity:	The platform should display a message indicating that the mandatory fields have to be filled.	Pass



WELCOME TO CRAFT WORLD

Have an account?

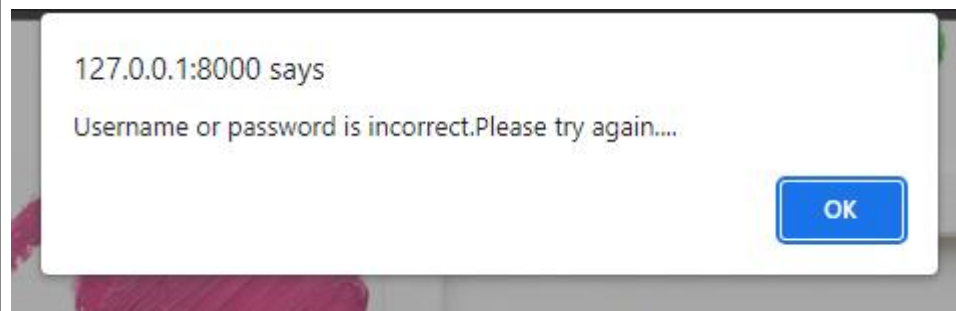
benjamin.smith@example.com

.....

SIGN IN

Forgot Password

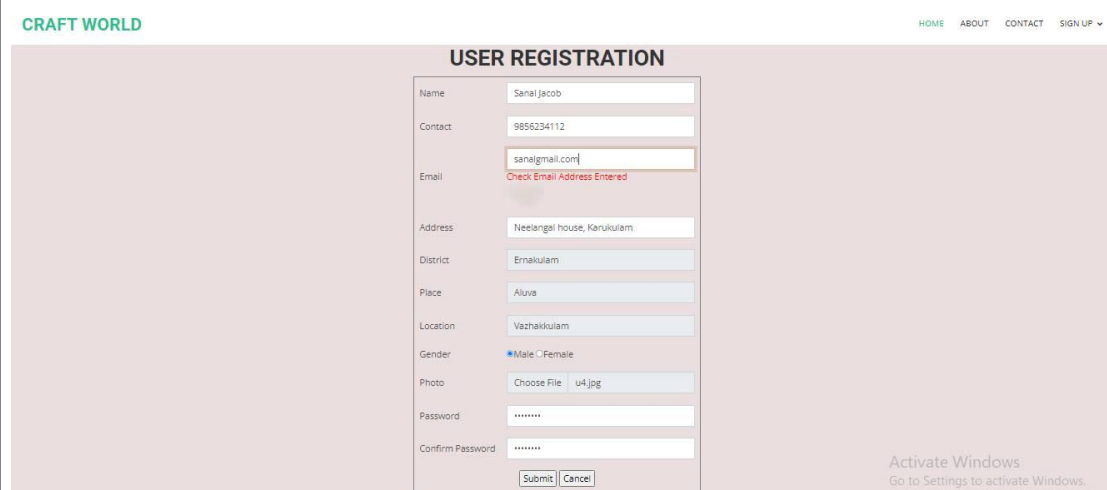
— Sign Up —



127.0.0.1:8000 says

Username or password is incorrect.Please try again....

OK



CRAFT WORLD

HOME ABOUT CONTACT SIGN UP

### USER REGISTRATION

Name	<input type="text" value="Sana Jacob"/>
Contact	<input type="text" value="9856234112"/>
Email	<input type="text" value="sana@gmail.com"/> <small>Check Email Address Entered</small>
Address	<input type="text" value="Neelangal house, Karukulam"/>
District	<input type="text" value="Ernakulam"/>
Place	<input type="text" value="Aluva"/>
Location	<input type="text" value="Vazhakkulam"/>
Gender	<input checked="" type="radio"/> Male <input type="radio"/> Female
Photo	<input type="text" value="Choose File"/> <input type="text" value="u4.jpg"/>
Password	<input type="password" value="....."/>
Confirm Password	<input type="password" value="....."/>

Activate Windows  
Go to Settings to activate Windows.



## 6. SYSTEM TESTING

### 6.1 INTRODUCTION

The purpose of the system testing is to identify and correct errors in the candidate system. Testing is an important element of the software quality assurance and represents the ultimate review of specification, design and coding. The increasing visibility of the software as a system element and the costs associated with a software failure are motivated forces for well-planned through testing. Software testing is a critical element of software quality assurance and represents the ultimate quality review of specifications, design and code generation. Once the source code has been generated, the program should be executed before the customer gets it with the specific intend of finding and removing all errors, test must be designed using disciplined techniques. Testing technique provides the systematic guidance for designing tests. To uncover the errors in the program behaviour function and performance the following steps to be done:

- Execute the integral logic of the software components.
- Execute the input and output domains of the program to uncover errors.
- During testing the system is used experimentally to ensure that the software does not fail, i.e., it will run according to the specification and in the way the user exports.
- Preparation of test data plays n vital rule in the system testing. Different set of test data are generated and the system under study is tested using that data.
- While testing using test data errors are again uncovered and corrected using different testing techniques.

### 6.2 INTEGRATION TESTING

This testing level can be simply defined as integrating and then testing, that is here, many unit tested modules are combined into subsystems, which are then tested. Integration testing aims at whether the modules can be integrated properly. Hence, the emphasis is on testing interfaces between modules. This testing activity can be considered testing the design.

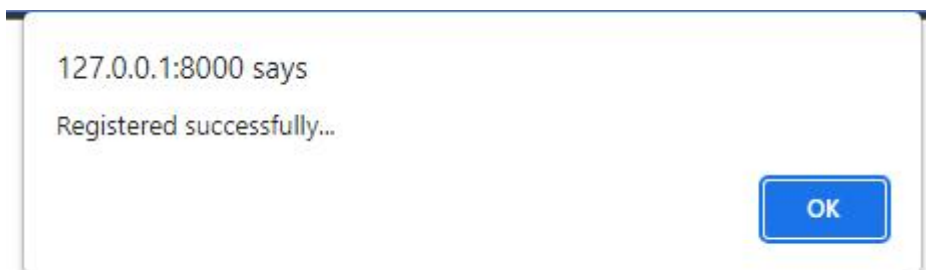
## 6.3 SYSTEM TESTING

System testing is the testing and is used to ensure that by putting the software in different environments it still works. It is done with executing the software system testing the application is working correctly from the point of view of a user. The main purpose of this system testing is to evaluate the system's compliance with the specified requirements. Whole system is tested as per the requirements. Black-box type testing that is related to overall requirements specifications, covers all combined parts of a system.

### 6.3.1 TEST PLAN AND TEST CASES

Test Case: Customer Registration

- Description: To verify that a user Successfully registered
- Input: Name, address, phone, email, password, photo
- Expected output: Successfully registered message displayed and redirected to login page
- Obtained output: Successfully registered message displayed



Test Case: Add to cart

- Description: Add products in to the cart.
- Input: select products
- Expected output: Product successfully added to the cart and redirected to payment page
- Obtained output: Product added to cart message displayed



Test case: Add Work

- Description: To add product details
- Input: Product name, price, quantity, worktype, description, image
- Expected output: Product successfully added and redirected to home page
- Obtained output: Product successfully added message displayed



## 7. SYSTEM MAINTENANCE

### 7.1 INTRODUCTION

System maintenance is an umbrella term that encompasses various forms of computer maintenance needed to keep a system running. The two main components of system maintenance are preventive and corrective maintenance. Preventive maintenance involves taking measures to help keep the system functioning, whereas corrective maintenance involves the replacement or repair of a system or its components after they have already failed.

### 7.2 MAINTENANCE

Maintenance projects can be used for enhancement or defect fixing. Enhancement projects are related to implementing new developments. It is basically modifications of software after it has been delivered in order to correct errors, to improve performance or other properties of IT systems. The key problems of software maintenance in IT systems include both management and technical problems. Important management issues are: adjusting to customer priorities, appropriate system maintenance personnel, cost estimation. The key technical issues are limited understanding, impact analysis, testing, measuring system maintainability, and infrastructure maintenance. It is well known that the guarantee of business survival is its continuous development and improvement of solutions, and thus also the continuous and adequate development of IT systems, combined with maintenance work carried out regularly on all components of the production environment.

#### **Corrective Maintenance:**

This refers to modifications initiated by defects in the software. A defect can result from design errors, logic errors and coding errors. Design errors occur when changes made to software are incorrect, incomplete, wrongly communicated or the change request is misunderstood. Logic errors result from invalid tests and conclusions, incorrect implementation of design specifications, faulty logic flow or incomplete test data.

Coding errors are caused by implementation of detailed logic design and incorrect use of the source code logic. Defects are also caused by data processing errors and system performance errors. In the event of system failure due to an error, actions are taken to restore the operation of the software system. Due to pressure from management, maintenance personnel sometimes

resort to emergency fixes known as “patching”. The adhoc nature of this approach often gives rise to a range of problems that include increased program complexity and unforeseen ripple effects. Unforeseen ripple effects imply that a change to one part of a program may affect other sections in an unpredictable manner, thereby leading to distortion in the logic of the system. This is often due to lack of time to carry out through “impact analysis” before effecting the change.

**Preventive Maintenance:**

Preventive maintenance (PM) is a proactive approach to maintenance that involves regularly scheduled inspections, servicing, and repairs of equipment or systems to prevent potential failures and to ensure they continue to operate efficiently. The main goal of preventive maintenance is to reduce the likelihood of unexpected breakdowns, minimize downtime, and extend the lifespan of the equipment. It is a planned and systematic strategy that focuses on keeping assets in good working condition, thus preventing costly repairs and production disruptions. Key Characteristics of Preventive Maintenance:

- **Scheduled Inspections:** Preventive maintenance involves creating a maintenance schedule based on the manufacturer's recommendations, industry best practices, and historical data. Regular inspections are carried out to identify any signs of wear, damage, or deterioration.
- **Routine Maintenance Tasks:** During preventive maintenance, routine tasks are performed, such as cleaning, lubricating, tightening connections, and replacing worn-out parts. These activities help keep the equipment in optimal working condition.
- **Predictable Costs:** Since preventive maintenance is planned in advance, the costs associated with maintenance activities can be budgeted and accounted for, making it easier to manage expenses.
- **Increased Reliability:** By addressing potential issues before they escalate, preventive maintenance helps improve the overall reliability and performance of the equipment, reducing the risk of unexpected breakdowns.
- **Safety Enhancement:** Preventive maintenance can identify safety concerns and address them promptly, ensuring a safe working environment for employees and users of the equipment.

**Perfective Maintenance:**

Perfective maintenance, also known as "enhancement maintenance" or "adaptive maintenance," is a type of software maintenance that focuses on improving or enhancing the functionality, performance, and usability of a software system. Unlike corrective maintenance, which addresses issues and defects, perfective maintenance aims to make the software better, more efficient, and aligned with changing user needs or business requirements. It involves modifying the software to add new features, optimize performance, and enhance the user experience without changing its original functionality. Key Objectives of Perfective Maintenance:

- Feature Addition: Perfective maintenance involves adding new features, functionalities, or capabilities to the software to enhance its usefulness and extend its capabilities. These additions are based on user feedback, market trends, or business requirements.
- Performance Optimization: The maintenance process may include re-engineering or optimizing the software to improve its speed, efficiency, and resource utilization. This can lead to a more responsive and faster-performing application.
- Usability Improvement: The user interface and overall user experience can be enhanced to make the software more user-friendly and intuitive. This could involve redesigning elements, streamlining workflows, or adding helpful tooltips.

**Adaptive Maintenance:**

Adaptive maintenance is a type of software maintenance that focuses on modifying a software system to keep it functional and up-to-date in response to changes in the computing environment or to ensure its compatibility with evolving hardware, software, or external interfaces. The primary goal of adaptive maintenance is to make the software adapt and remain operational in changing circumstances without altering its original functionality or introducing new features.

Key Objectives of Adaptive Maintenance:

- Compatibility: Adaptive maintenance addresses issues that arise due to changes in the computing environment, such as operating system updates, changes in hardware configurations, or new software dependencies.

- External Interface Conformity: The maintenance process ensures that the software remains compatible with external systems and services it interacts with, such as databases, APIs, or web services.
- Regulatory Compliance: Adaptive maintenance may involve updating the software to meet new industry standards, data protection regulations, or legal requirements.
- Technology Upgrades: Upgrading underlying software libraries, frameworks, or programming languages may be necessary to ensure continued support and security updates.

## **8. SYSTEM SECURITY MEASURES**

### **8.1 INTRODUCTION**

The security of a computer system is a crucial task. It is a process of ensuring the confidentiality and integrity of the OS. Security is one of most important as well as the major task in order to keep all the threats or other malicious tasks or attacks or program away from the computer's software system. A system is said to be secure if its resources are used and accessed as intended under all the circumstances, but no system can guarantee absolute security from several of various malicious threats and unauthorized access.

The security of a system can be threatened via two violations:

- Threat: A program that has the potential to cause serious damage to the system.
- Attack: An attempt to break security and make unauthorized use of an asset.

Security violations affecting the system can be categorized as malicious and accidental threats. Malicious threats, as the name suggests are a kind of harmful computer code or web script designed to create system vulnerabilities leading to back doors and security breaches. Accidental Threats, on the other hand, are comparatively easier to be protected.

### **8.2 DATABASE LEVEL SECURITY**

Database level security refers to the security measures implemented to protect databases and the data they store. Databases often contain sensitive and valuable information, so ensuring their security is crucial.

Here are some important database level security measures:

- Authentication and Authorization: Implement strong authentication mechanisms to verify the identity of users accessing the database. Enforce password policies, use twofactor authentication (2FA), and limit administrative privileges to authorized personnel. Assign appropriate user roles and permissions to control access to database objects and operations.
- Data Encryption: Use encryption techniques to protect sensitive data stored in the database. Encrypting data at rest and in transit helps prevent unauthorized access, even if the underlying



storage or network infrastructure is compromised. Implement transparent data encryption (TDE) or field-level encryption for sensitive data elements.

- **Access Controls:** Implement access controls at the database level to restrict unauthorized access to data. Use role-based access control (RBAC) or attribute-based access control (ABAC) mechanisms to enforce granular permissions and ensure that users can only access the data they are authorized to see.
- **Auditing and Logging:** Enable auditing and logging features provided by the database system to track and monitor database activities. Record user actions, schema changes, data modifications, and access attempts. Regularly review and analyze logs to detect any unusual or suspicious activities.

### 8.3 SYSTEM-LEVEL SECURITY

System level security refers to the security measures implemented at the overall system level to protect the entire computer system, including the hardware, operating system, network components, and software applications.

Here are some important system level security measures:

- **Secure System Configuration:** Ensure that the system is securely configured by following security best practices and vendor recommendations. This includes disabling unnecessary services and features, removing default accounts and passwords, and implementing secure settings for network interfaces, firewalls, and other system components.
- **Regular System Updates and Patching:** Keep the system up to date with the latest security patches and updates provided by the software and hardware vendors. Regularly apply patches to address known vulnerabilities and security weaknesses. Enable automatic updates whenever possible.
- **Malware Protection:** Install reputable antivirus and anti-malware software on the system to detect and remove known malware threats. Keep the software updated with the latest virus definitions to ensure effective protection. Use behavior-based detection and real-time scanning to detect and prevent new and emerging malware threats.

- **Host-Based Firewalls:** Configure host-based firewalls on the system to control inbound and outbound network traffic. Define rules to allow only necessary network connections and block unauthorized access attempts.
- **System Hardening:** Apply system hardening practices to minimize potential vulnerabilities. This includes disabling unnecessary services and protocols, removing or securing unnecessary user accounts, and applying security configurations and access controls to limit system exposure.

## **9. SYSTEM PLANNING AND SCHEDULING**

### **9.1 INTRODUCTION**

Planning and scheduling are distinct but inseparable aspects of managing the successful project. The process of planning primarily deals with selecting the appropriate policies and procedures in order to achieve the objectives of the project. Scheduling converts the project action plans for scope, time cost and quality into an operating timetable. The translating of the project criteria for scope, time, cost, and quality and the requirements for human resources, communications, risk and procurement into workable “machinery” for the project team a critical interface juncture for the project team. Taken together with the project plan and budget, the schedule becomes the major tool for the management of projects. In addition, the integrated cost-time schedule serves as the fundamental basis for monitoring and controlling project activity throughout its life cycle.

This basic level paper addresses the integrated processes of planning and scheduling of multifacet/multidisciplinary programs. The paper presents a working level summary of the major Project Management topics involved in the planning process. The paper also details a systematic process for transforming the Project Plan into the Schedule and the use of the Project Schedule as a model for project control. Intended for the project management novice, the paper concludes with a suggested professional development scheme.

### **9.2 PLANNING A SOFTWARE PROJECT**

Project planning is at the heart of the project life cycle, and tells everyone involved where you’re going and how you’re going to get there. The planning phase is when the project plans are documented, the project deliverables and requirements are defined, and the project schedule is created. It involves creating a set of plans to help guide your team through the implementation and closure phases of the project. The plans created during this phase will help you manage time, cost, quality, changes, risk, and related issues. They will also help you control staff and external suppliers to ensure that you deliver the project on time, within budget, and within schedule.

The purpose of the project planning phase is to:

- Establish business requirements.
- Establish cost, schedule, list of deliverables, and delivery dates.
- Establish resources plans.
- Obtain management approval and proceed to the next phase.

The basic processes of project planning are:

- Scope planning– specifying the in-scope requirements for the project to facilitate creating the work breakdown structure.
- Preparation of the work breakdown structure – spelling out the breakdown of the project into tasks and sub-tasks.
- Project schedule development – listing the entire schedule of the activities and detailing their sequence of implementation.
- Resource planning – indicating who will do what work, at which time, and if any special skills are needed to accomplish the project tasks.
- Budget planning – specifying the budgeted cost to be incurred at the completion of the project.
- Procurement planning– focusing on vendors outside your company and subcontracting.
- Risk management– planning for possible risks and considering optional contingency, plans and mitigation strategies.
- Quality planning – assessing quality criteria to be used for the project

### **9.2.1 STEPS INVOLVED IN PLANNING A SYSTEM**

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process. The following figure is a graphical representation of the various stages of a typical SDLC.

A typical Software Development Life Cycle consists of the following stages:

#### **1. Planning and Requirement Analysis**

2. Defining Requirements
3. Designing the Product Architecture
4. Building or Developing the Product
5. Testing the Product
6. Deployment in the Market and Maintenance

### PROJECT SCHEDULE

TASK	START DATE	END DATE
Study the existing System	15/04/2023	26/04/2023
Identify the problem	26/04/2023	05/05/2023
Defining the system	05/05/2023	30/05/2023
Database Design	30/05/2023	10/06/2023
Input Design	10/06/2023	15/06/2023
Coding	15/06/2023	15/07/2023
Testing	15/07/2023	31/07/2023
Building a prototype	31/07/2023	10/08/2023

## **10. FUTURE ENHANCEMENT AND SCOPE OF FURTHER DEVELOPMENT**

### **10.1 INTRODUCTION**

The application developed is designed in such a way that any further enhancement can be done with ease. The system has the capability for easy integration. New modules can be added to the system with less effort. The website is developed in Python-Django, which makes it more reliable and compatible with other environments. The application proves better extensibility and flexibility for future enhancements.

This software helps to buy the craft products and materials. This has been developed for keeping in mind the problems and limitations faced by the client to search for a good and reliable craft products and materials. The customers can register to the system and buy products and the materials from it. Then the marketing of the products can make more easily.

### **10.2 MERITS OF THE SYSTEM**

The system is newly automated and there are a lot of merits related to this system.

They are:

1. User-friendly system: The system is designed in a manner that anyone could use this system without any training given. They could easily understand the way the system works.
2. Faster performance of the system is a great advantage.
3. Compatibility: Developed using Python-Django, the system boasts compatibility with various platforms and environments, ensuring that it can be seamlessly accessed and used by a wide range of users.
4. Time is money: It greatly saves time. It provides them an option to select from numerous products and materials.
5. Scalability: The system is designed with scalability in mind, allowing for easy integration of new modules and features as the community's needs evolve.

6. User-Friendly Interface: The intuitive user interface enhances the user experience, making it easy for customers, and administrators to navigate the system, add products and materials, return products, facility for chatting.

7. Easy to buy products: The system displays various craft products so that customers can view and add products to their order list.

8. Efficient booking Management: The system streamlines the booking management process, customers can view their bookings and previous booking history.

### **10.3 LIMITATIONS OF THE SYSTEM**

Not every system is perfect and complete. Every system has its own limitations. The website “Craft World” also has some limitations.

- ✧ Internet Connectivity: Users in remote areas with limited internet access could experience difficulties in using the platform consistently, impacting their ability to place orders and interact with the system.
- ✧ Technological Accessibility: Customer, especially those less familiar with technology, might face challenges in adapting to the system, potentially leading to a digital divide within the community.
- ✧ Online Payment Challenges: Online payment processing might face challenges related to transaction failures, payment gateways, and customer reluctance to share sensitive financial information.
- ✧ Return products price will be transferred directly.

### **10.4 FUTURE ENHANCEMENT OF THE SYSTEM**

The system has been designed in such a way that it can be modified with very little effort when such needs arise in the future. New features can be added with slight modifications of software which make it easy to expand the scope of this project. Though the system is working on various assumptions, it can be modified easily to any kind of requirements. Creating a mobile app version of the system would allow users to access the platform conveniently on their smartphones, enhancing user engagement and accessibility.

## 11. ANNEXURE

### 11.1 ORGANIZATION PROFILE

Progressive Cybernetics is a fast developing, well established Information Technology enterprise with highly talented and efficient staff meeting the needs of clients throughout the world. The company is so popular that the clients feed the company with repeat orders, having faith in the capabilities. The staff includes management and engineering personnel, programmers, web developers, graphic designers, business analysts, Technical writers etc, apart from the regular commercial and administrative hands. The motto of this reliable software provider is DO BEST, GET BEST. Progressive Cybernetics committed to provide what they have promised. Progressive Cybernetics strives for the satisfaction of their clients and dedicates themselves for the better services to their customers. Progressive Cybernetics are famous in providing high quality, cost cutting, requirement oriented and value added web and software solutions with remarkable cyber services. Progressive Cybernetics are specialist in developing ever rewarding internet existence for ambitious companies which are in the hangover of swift moving internet facilities. Their aim is to grow with such organizations in a very fruitful manner. Their main concentration is to collect business oriented technological know-how and utilize them for the wellbeing of business concerns. Progressive Cybernetics experts in various technologies are committed to deliver effective and creative service to customers.

Services :

- Software Development
- Web Development
- SEO Services (Search Engine Optimization)
- Consulting Services
- Host Services
- Testing
- SMS Services



**Address:** Progressive Cybernetics Pvt Ltd.

Tech Floor, Koyas Tower

P O Junction, Muvattupuzha

Ernakulam, Kerala - 686 661

### **Why Progressive Cybernetics?**

Provide software development includes Application software, web development, system software, embedded system software development, etc.

➤ Provide software training in various technologies like data science and machine learning, advanced diploma in embedded system, python full stack development, java full stack development, etc.

➤ Progressive Cybernetics has best IT Infrastructure and experienced faculties for providing the course and training.

## **11.2 DOCUMENT GLOSSARY, FIGURES, TABLES**

### **ABBREVIATION**

SI.NO	ABBREVIATION	EXPANSION
1	DBMS	DATABASE MANAGEMENT SYSTEM
2	HTML	HYPERTEXT MARKUP LANGUAGE
3	CSS	CASCADING STYLE SHEETS
4	CPU	CENTRAL PROCESSING UNIT
5	ER	ENTITY RELATIONSHIP

**FIGURES**

<b>FIG.NO</b>	<b>DIAGRAM</b>	<b>PAGE NO</b>
1	ER DIAGRAM	12
2	USE CASE DIAGRAM	24
3	SEQUENCE DIAGRAM	27
4	ACTIVITY DIAGRAM	29

**Tables**

<b>TABLE NO</b>	<b>TABLE NAME</b>	<b>PAGE NO</b>
1	tbl_worktype	13
2	tbl_distict	13
3	tbl_place	13
4	tbl_location	14
5	tbl_user	14
6	tbl_videopay	15
7	tbl_seller	16
8	tbl_wgallery	16

9	tbl_material	17
10	tbl_works	17
11	tbl_chat	18
12	tbl_star	18
13	tbl_complaint	19
14	tbl_feedback	19
15	tbl_mbooking	20
16	tbl_mcart	20
17	tbl_wbooking	21
18	tbl_wcart	21
19	tbl_return	22

## 11.3 REFERENCES

### 11.3.1. BOOKS

1. Software Engineering A Practitioner's Approach, Roger S Pressman, McGrawhill International, Edition, Sixth Edition.
2. Software Engineering, K K Agarwal and Yogesh Singh, New age international, Third Edition.
3. Object Oriented modeling and Design with UML, Michael Blaha, James Rumbaugh, Person, second edition.
4. Craig Larman, "Applying UML and patterns: An introduction to object", Oriented Analysis and Design and Unified Process , 3rd Edition, Pearson Education, 2007.

5. Raghu Ramakrishnan and Johannes Gehrke, Database Management System, Third Edition, McGraw Hill Companies, 2003

6. Programming and Problem Solving with Python, Ashok Namdev Kamthane & Amit Ashok

### **11.3.2 WEBSITES**

- <https://chat.openai.com/>
- <https://tutorialspoint.com/>
- <https://github.com>
- <https://www.w3schools.com/>



