ABSTRACT

In today's fast changing business environment, it's extremely important to be able to respond to client needs in the most effective and timely manner. If your customers wish to see your business online and have instant access to your products or services.

Full Stack Ecommerce Web App is an online web application, which is also a Progressive Web App (PWA), built with React js, Node js, Express, and MongoDB where the user can buy items from the web app. Each user will be able to buy things such as home appliances, electronics, furniture, devices, fashion products etc. Each user will have a specific account in the application. Users will be only able to make a purchase if he/she is logged in. Every common person can walk through and buy items easily with the simple User Interface. The user can search the products, find the required item, add the item to cart and proceed with the deal with payment gateway, or pay later by cash on delivery method. The web app will be very user-friendly. As it is also a PWA app it can be used like a native app in your device for easier access and use irrespective of any operating system.

This document will discuss each of the underlying technologies to create and implement a full stack ecommerce web app.

INTRODUCTION

1.1 Overview of project

This project is entitled with Full Stack Ecommerce web app. This application developed using MERN Stack which uses React js as frontend MongoDB as server and Node Js Express as backend language.
E-commerce is a fast-gaining ground as an accepted and used business paradigm. More and more business houses are implementing web and native applications providing functionality for performing commercial transactions over the web. It is reasonable to say that the process of shopping on the web is becoming commonplace.
The object of this project is to develop a general-purpose e-commerce store where various categories of products like devices, home appliances, fashion products etc. can be brought from the comfort of home through the internet. This application will provide you with a wide range of choices.

An online store is a virtual store on the internet where customers can browse the catalog and select products of interest. The selected items may be collected in a shopping cart. At checkout time, the items in the shopping cart will be presented as an order. At that time, more information will be needed to complete the transaction. Usually, the customer will be asked to fill or select a billing address, a shipping address, a shipping option, and payment information such as UPI or credit/debit card number.

Overall Description

- ➢ Any member can register and view available products.
- ➢ Only registered member can purchase multiple products regardless of quality.
- ➢ Contact page is available to contact Admin for queries.
- ➢ There are three roles available Visitor, User and Admin
  - Visitor can view available products
  - User can view and purchase products
  - An admin has extra privilege including all privileges of visitor and user
    - ✓ Admin can add/edit products information and add/remove product
    - ✓ Admin can add user, edit user information and can remove user.
    - ✓ Admin can ship order to user based on order placed by notifications

1.2 Modules Description

The major objective of this project is to develop an application that is designed for the People and provide user friendly environment for the user. All the options needed for the users are available in this application.

It has been modularized into following modules.

- Administrator Module
- Customer Module
- Visitor Module

1.2.1 Administrator Module:

Used for managing the users, products and orders. Admin module allows system administrators to set up the back-end of the system and perform basic system configuration. In this project, admin can add/edit the products, manage the users and process the orders

1.2.3 Customer Module:

Used for purchasing the products. Customer module allows the customer to register and login to the web app and buy products online. Customers can search the products, find the desired product, add to cart, and process the payment with the payment gateway.

1.2.2 Visitor Module:

Visitors of the web app are able to view the products and search the products. If the visitor likes to purchase an item he needs to register and login to the web app.

SOFTWARE REQUIRMENT SPECIFICATION

2.1 About the Program Language

React is an open-source, front end, JavaScript library for building user interfaces or UI components. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications.

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that executes JavaScript code outside a web browser.

Express.js is a back-end web application framework for Node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs. It has been called the de facto standard server framework for Node.js.

MongoDB is a cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas. MongoDB is developed by MongoDB Inc. and licensed under the Server-Side Public License.

2.2 Software Requirements

System requirements are expressed in a software requirement document. The software requirement document (also known as software requirement specification or SRS) is the official statement of what is required of the system developers.

• Software	:	Visual Studio Code.
• Operating System	:	Windows 7 or higher, MacOS, Linux.
• Browser	:	Google Chrome, Safari , MS Edge etc.
• Front End	:	React js
• Back End	:	Node js
• Server	:	MongoDB

- **MongoDB (DBMS) -** MongoDB is an open source, non-relational database management system (DBMS) that uses flexible documents instead of tables and rows to process and store various forms of data. Mongo DB is a cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas. MongoDB is developed by MongoDB Inc. and licensed under the Server-Side Public License (SSPL).
- **Visual Studio Code** -Visual Studio Code is a free source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.
- **Node js -** Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that executes JavaScript code outside a web browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm,[6] unifying web-application development around a single programming language, rather than different languages for server-side and client-side scripts.
- **React js -**React is an open-source, front end, JavaScript library for building user interfaces or UI components. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications.
- **Quick -** React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes.

SYSTEM ANALYSIS

3.1 Existing system

Suppose we went to a supermarket or a shop to purchase some products. We need to search for the required products by walking around only. We need to pick the products, wait in the queue for the billing and we need to transport the products to home. The whole process consumes a sufficient amount of time and effort. Sometimes we will be out of cash and we want to pay through card but some shops will not accept cards.

Advantages:

- Customers can look and pick the products physically.
- No need to wait for delivery

Disadvantages

- Time consuming
- Transportation cost
- Sometimes cards might not be acceptable.
- Lack of social distancing during this COVID-19 spreading time.

3.2 Proposed system

The tool is developed using MERN stack. Customers can register online and they can search and browse products online that are arranged in different categories systematically. Each customer can create their own account to see all products. And each user has the right to purchase any products using the integrated payment gateway and track the status of orders.

Advantages

- Anyone with a smartphone, computer or laptop can buy products.
- Easy to buy products.
- You can buy with cash on delivery or online payment.
- Products are delivered to the address provided.
- It can be modified further.
- Reliable and user friendly.

Disadvantages

- Customers cannot physically look and pick the products.
- Delay of delivery.

SYSTEM DESIGN& FEASIBILITY STUDY

4.1 Database Design

The main motive of database is to help manage the complete data records in a proper manner. MongoDB is a cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas including:

User/Admin Table: -

Column Name	Data Type	Description
_id	Object_Id	Primary Key
isAdmin	Boolean	Defines Admin or User
Name	String	Name of Admin/User
Email	String	Email of Admin/User
Password	String	Password for Admin/User
createdAt	Date	Date account created
updatedAt	Date	Date account updated

Products Table: -

Column Name	Data Type	Description
_id	Object_Id	Primary Key
Rating	Mixed	Rating of product
numReviews	Int32	Number of Reviews
Price	Int32	Price of products
countInStock	Int32	Stock of products
Name	String	Name of Product
Image	String	Image of Product
description	String	Description about Product
Brand	String	Brand of Product
category	String	Category of Product
reviews	Array	Reviews of Users
createdAt	Date	Date Product created
updatedAt	Date	Date Product updated

Orders Table: -

Column Name	Data Type	Description
_id	Object_Id	Primary Key
itemsPrice	Mixed	Price of Items
shippingPrice	Int32	Cost for Shipping
isPaid	Boolean	Payment Completed or Not
isDelevered	Boolean	Product Delivered or Not
User	Object_Id	User Ordered the Product
orderItems	Array	Items Ordered
shippingAddress.address	Object	Address for Delivery
shippingAddress.city	String	City of Address for Delivery
shippingAddress.pincode	String	Pin code of Address for Delivery
shippingAddress.state	String	State of Address for Delivery
paymentMethod	String	Method used for Payment
createdAt	Date	Order Placed
updatedAt	Date	Updated Status of Order
paidAt	Date	Payment Date
paymentResult.id	Object	Payment Transaction Id
paymentResult.status	String	Payment Status
paymentResult.update_time	String	Update of Payment
paymentResult.email_address	String	Email Id Used for Payment
deleveredAt	Date	Order Delivered Date

4.1.1 Maintaining relationship between data in the database

Ensuring that data stored correctly and that the rules defining data relationships are not violated. Recover all data a point of known consistency in case of system failures.

4.1.2 NoSQL database

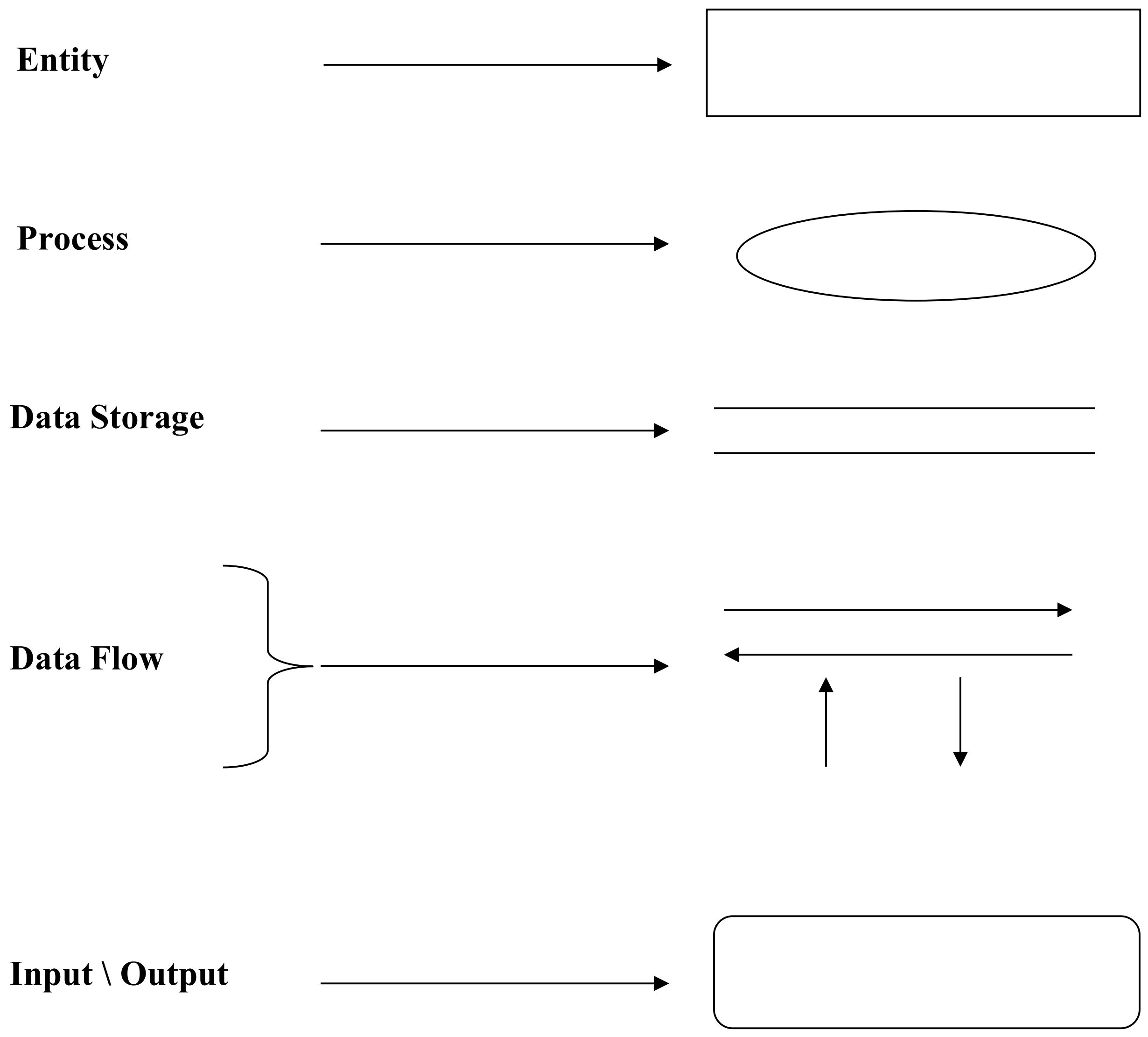
NoSQL, which stands for “not only SQL,” is an approach to database design that provides flexible schemas for the storage and retrieval of data beyond the traditional table structures found in relational databases.

4.2 Data Flow Diagram

4.2.1 Introduction

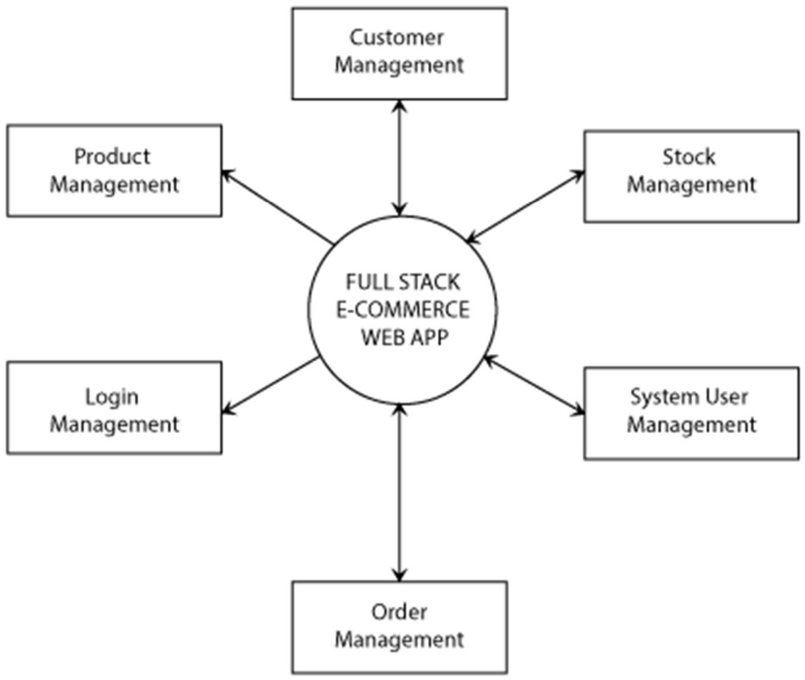
Goal of SDLC is to deliver a system in line with the users requirement analysis is the heart of the process. Data flow diagram illustrates how data flows through a system and how output is derived from the input through a sequence of functional transformations.

Data Flow Diagrams are a useful and innovative way of describing a system. They are normally understandable without special training, especially if control information is executed. They show end to end processing that is, the flow of processing from when data enters the system to when it leaves the system can be traced. Data Flow design is an integral part of a number of design methods and more CASE tools support data flow design creation. Alternatively, it could be implanted as a number of communicating tasks.

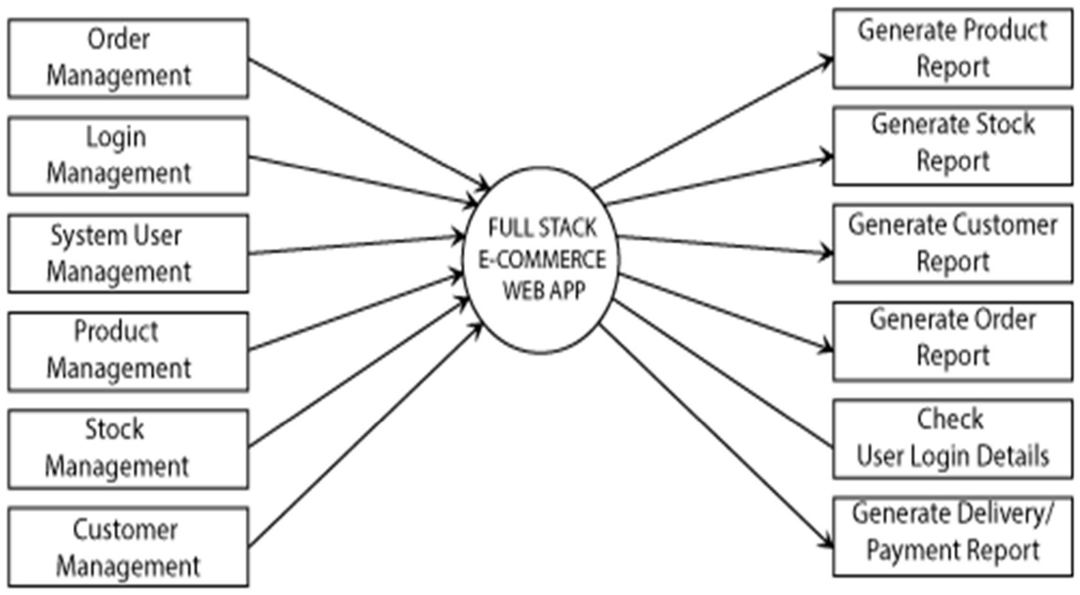

Entity
Process
Data Storage
Data Flow
Input \ Output

DATA FLOW DIAGRAM

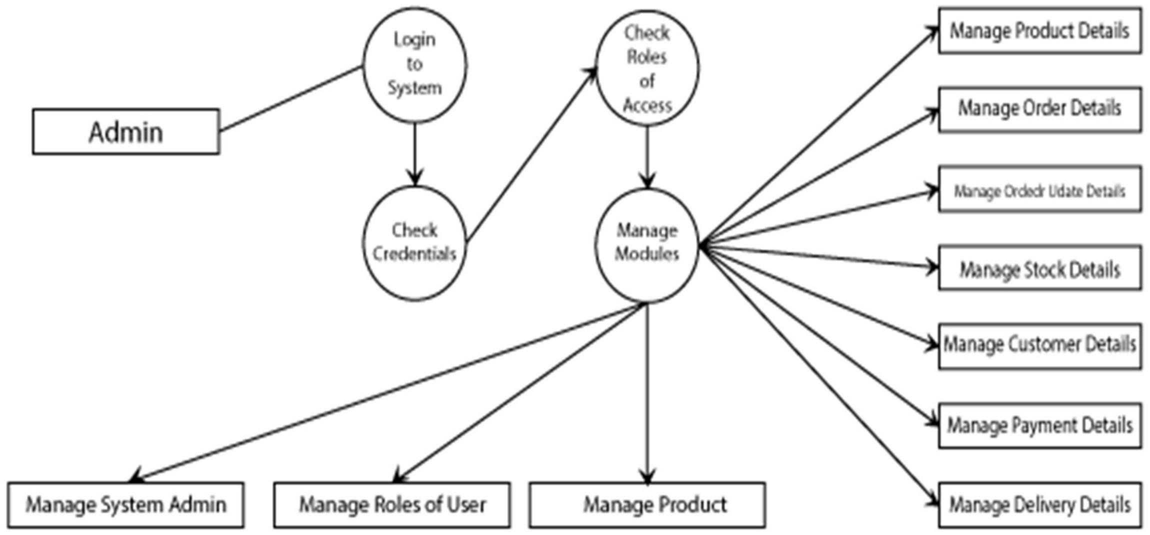
Level 0



Level 1


Order Management
Login Management
System User Management
Product Management
Stock Management
Customer Management
FULL STACK E-COMMERCE WEB APP
Generate Product Report
Generate Stock Report
Generate Customer Report
Generate Order Report
Check User Login Details
Generate Delivery/ Payment Report

Level 2


Admin
Login to System
Check Credentials
Check Roles of Access
Manage Modules
Manage Product Details
Manage Order Details
Manage Orcledr Udate Details
Manage Stock Details
Manage Customer Details
Manage Payment Details
Manage Delivery Details
Manage System Admin
Manage Roles of User
Manage Product

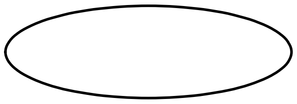
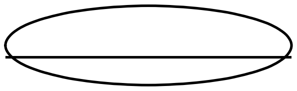
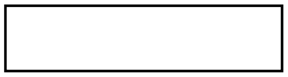
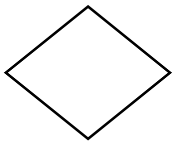
4.3 ER – Diagram

An entity – relationship (ER) diagram is a specialized graphic that illustrate the interrelationship between in a database. ER diagram often use symbols to represents three different types of information. Boxes are commonly used to represent entities. Diamonds are normally used to represents relationship diagrams don't show single entities or single instance of relations. Rather, they show entity sets and relationship sets.

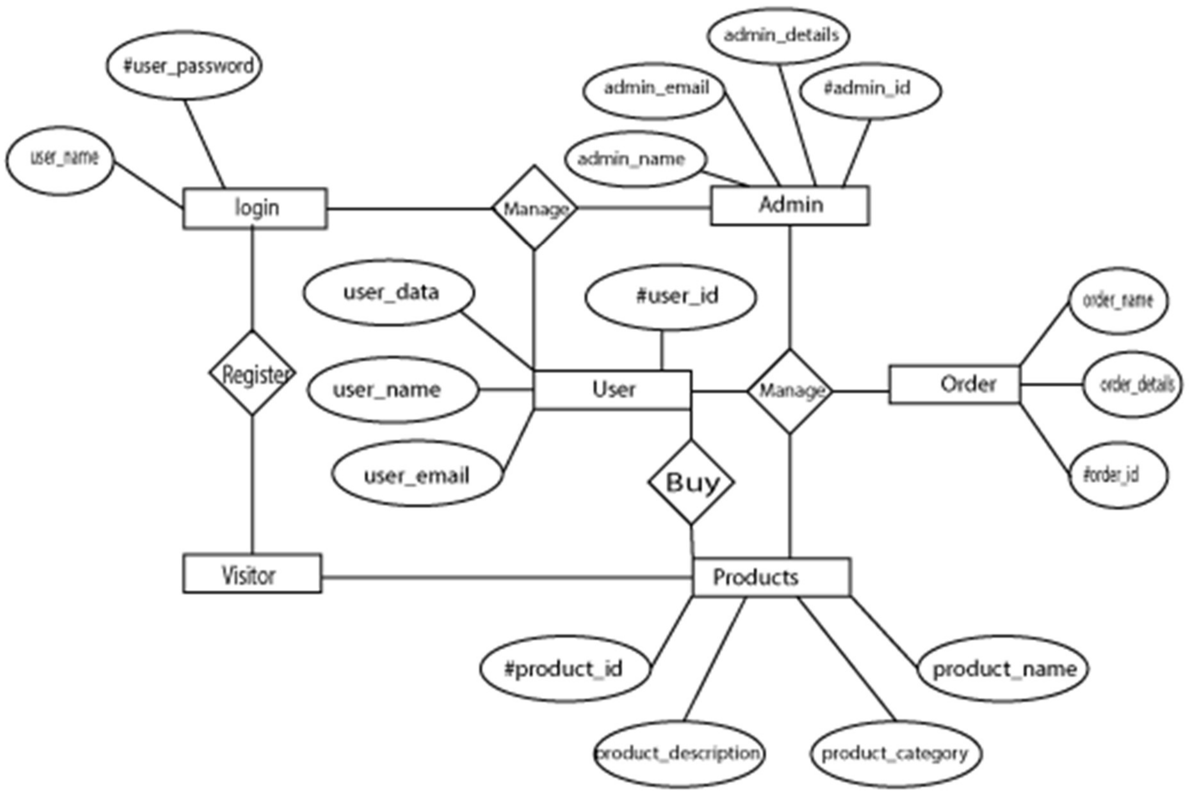
Lines are drawn between entity sets and the relationship sets they are involved in. if all entities in an entity sets must participates in the relationship set, a thick or double line is drawn. This is called participation constraints. If each entity set can participate in at most one relationship set, an arrow is drawn from the entity set to the relationship set.

Associative entity is used to solve the problem of the two entities with a many-to-many relationship.

Definition of symbols:

Attribute	
Key Attribute	
Entity	
Entity relations	

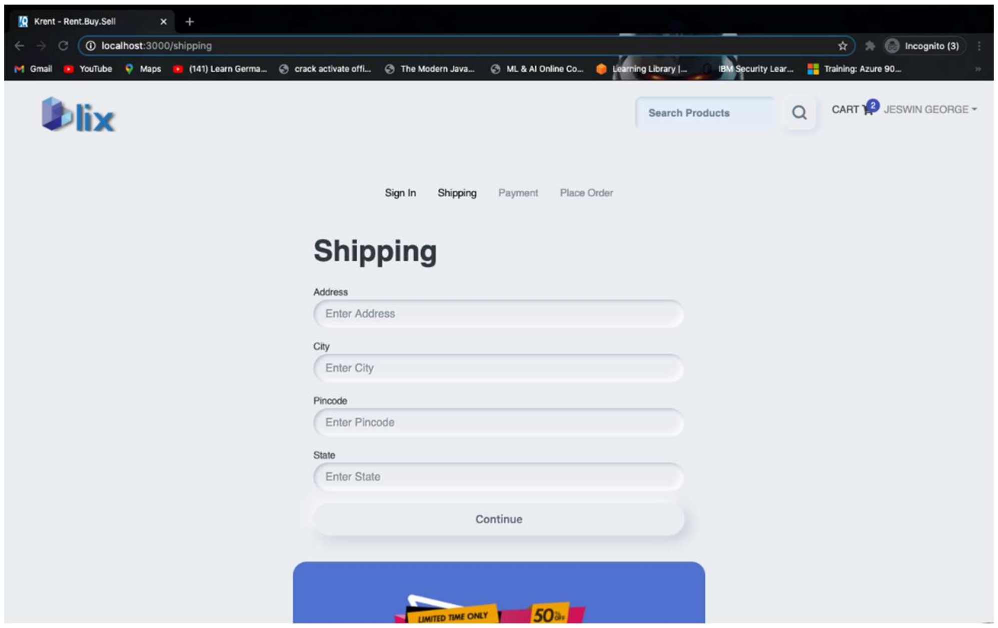
ER- DIAGRAM


admin_details
#user_password
admin_email
#admin_id
user_name
admin_name
login
Manage
Admin
user_data
#user_id
order_name
Register
user_name
User
Manage
Order
order_details
user_email
Buy
#order_id
Visitor
Products
#product_id
product_name
product_description
product_category

4.4 Input Design

Input design is the method by which valid data are accepted from the user. The valid data turn is stored as operational data in the database. Incorrect input data are the most common cause of errors in the data processing. The input design is carried out in such a way that the input screens are user friendly. The goals of designing input design are to make input data entry as easy and error free. Input screen takes care to filter the valid data from becoming an operational data at the data entry phase.

Input design is the part of the overall system design that receives careful attention and is the most expensive phase. It is the point of most contact for the users with the system and so it is prone to errors.



Shipping

Address

Enter Address

City

Enter City

Pincode

Enter Pincode

State

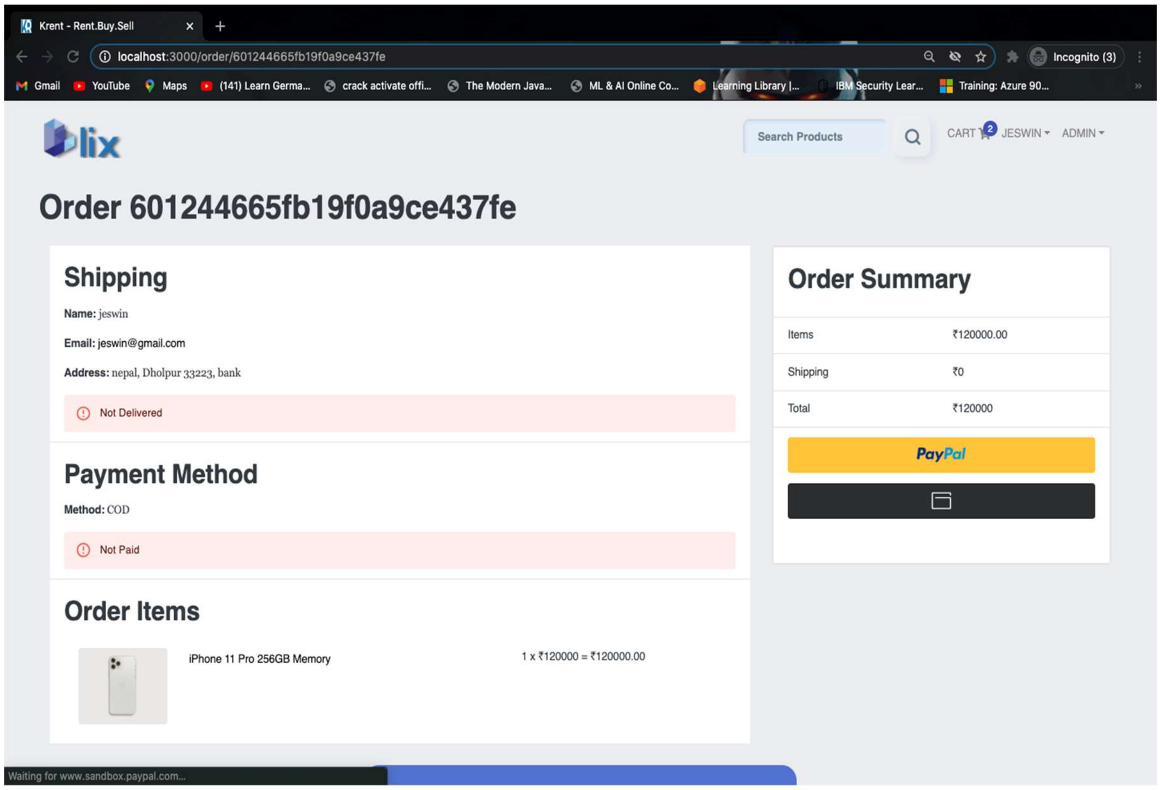
Enter State

Continue

4.5 Output Design

The output design defines the output required and the format in which it is to be produced. Care must be given to present the right information.

The output is the most important and direct source of information to the user. Efficient, output design should improve the systems relationship with the user and helps in decision making. A major form of output is a hard copy from the printer. Printouts should be designed around the output requirement to the user. The standard that is maintained for output design is clear. Output provides a permanent copy of the results for the later consultation.



Order 601244665fb19f0a9ce437fe

Shipping

Name: jeswin

Email: jeswin@gmail.com

Address: nepal, Dholpur 33223, bank

Not Delivered

Payment Method

Method: COD

Not Paid

Order Items

iPhone 11 Pro 256GB Memory 1 x ₹120000 = ₹120000.00

Order Summary

Items	₹120000.00
Shipping	₹0
Total	₹120000

4.6 Feasibility Study

The feasibility study investigates the problem and the information needs of the stakeholders. It seeks to determine the resources required to provide an information systems solution, the cost and benefits of such a solution, and the feasibility of such a solution. The analyst conducting the study gathers information using a variety of methods, the most popular of which are:

- Interviewing users, employees, managers, and customers.
- Developing and administering questionnaires to interested stakeholders, such as potential users of the information system.
- Observing or monitoring users of the current system to determine their needs as Well as their satisfaction and dissatisfaction with the current system.
- Collecting, examining, and analyzing documents, reports, layouts, procedures, manuals, and any other documentation relating to the operations of the current system.
- Modeling, observing, and simulating the work activities of the current system.

The goal of the feasibility study is to consider alternative information systems solutions, evaluate their feasibility, and propose the alternative most suitable to the organization. The feasibility of a proposed solution is evaluated in terms of its components. These components are:

CODING

5.1 Webpage design

LOGIN_FORM

```
import React, { useState, useEffect } from "react";
import { Link } from "react-router-dom";
import { Form, Button, Row, Col } from "react-bootstrap";
import { useDispatch, useSelector } from "react-redux";
import Message from "../components/Message";
import Loader from "../components/Loader";
import { login } from "../actions/userActions";
import FormContainer from "../components/FormContainer";
const LoginScreen = ({ location, history }) => {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");

  const dispatch = useDispatch();
  const userLogin = useSelector((state) => state.userLogin);

  const { loading, error, userInfo } = userLogin;

  const redirect = location.search ? location.search.split("=")[1] : "/";
  useEffect(() => {
    if (userInfo) {
      history.push(redirect);
    }
  }, [history, userInfo, redirect]);
  const submitHandler = (e) => {
    dispatch(login(email, password));
    e.preventDefault();
  };

  return (
    <FormContainer className="neu__form mt-5">
```

```
<h1 className="neu__form-h1">Sign In</h1>
{error && <Message severity="error">{error}</Message>}
{loading ? (
  <Loader />
) : (
  <Form onSubmit={submitHandler} className="neu__form">
    <Form.Group controlId="email">
      {/* <Form.Label>Email Address</Form.Label> */}
      <Form.Control
        className="neu__form-input mb-4"
        type="email"
        placeholder="Enter email"
        value={email}
        onChange={(e) => setEmail(e.target.value)}
      ></Form.Control>
    </Form.Group>
    <Form.Group controlId="password">
      {/* <Form.Label>Password</Form.Label> */}
      <Form.Control
        className="neu__form-input"
        type="password"
        placeholder="Enter password"
        value={password}
        onChange={(e) => setPassword(e.target.value)}
      ></Form.Control>
    </Form.Group>
    <Button type="submit" varient="primary" className="neu__button mt-4">
      <i className="fas fa-lock"></i> &ensp; Sign In
    </Button>
  </Form>
)}

<Row className="py-3">
  <Col>
    New Customer?{" "}
    <Link to={redirect ? `/register?redirect=${redirect}` : "/register"}>
      Register
    </Link>
```

```
        </Col>
      </Row>
    </FormContainer>
  );
};

export default LoginScreen;
```

REGSTER_SCREEN

```
import React, { useState, useEffect } from "react";
import { useDispatch, useSelector } from "react-redux";
import { Link } from "react-router-dom";

import { Row, Col, Form, Button } from "react-bootstrap";

import FormContainer from "../components/FormContainer";

import Loader from "../components/Loader";

import Message from "../components/Message";

import { register } from "../actions/userActions";


const RegisterScreen = ({ history, location }) => {

  const [name, setName] = useState("");

  const [email, setEmail] = useState("");

  const [password, setPassword] = useState("");

  const [confirmPassword, setConfirmPassword] = useState("");

  const [message, setMessage] = useState(null);
const dispatch = useDispatch();
const userRegister = useSelector((state) => state.userRegister);

const { loading, error, userInfo } = userRegister;

const redirect = location.search ? location.search.split("=")[1] : "/";
useEffect(() => {
  if (userInfo) {
    history.push(redirect);
  }
}, [history, userInfo, redirect]);
const submitHandler = (e) => {
  e.preventDefault();
  if (password !== confirmPassword) {
    setMessage("Passwords Not Match");
  } else {
    dispatch(register(name, email, password));
  }
};
return (
  <FormContainer className="neu__form mt-5">
    <h1 className="neu__form-h1">Sign Up</h1>
    {message && <Message severity="error">{message}</Message>}
```

```
{error && <Message severity="error">{error}</Message>}
{loading ? (
 <Loader />
) : (
 <Form onSubmit={submitHandler} className="neu__form">
  <Form.Group controlId="name">
   {/* <Form.Label>Name</Form.Label> */}
   <Form.Control
    type="name"
    className="neu__form-input mb-4"
    placeholder="Enter Name"
    value={name}
    onChange={(e) => setName(e.target.value)}
   ></Form.Control>
  </Form.Group>
  <Form.Group controlId="email">
   {/* <Form.Label>Email Address</Form.Label> */}
   <Form.Control
    className="neu__form-input mb-4"
    type="email"
    placeholder="Enter email"
    value={email}
    onChange={(e) => setEmail(e.target.value)}
   ></Form.Control>
```

```
</Form.Group>
<Form.Group controlId="password">
  {/* <Form.Label>Password</Form.Label> */}
  <Form.Control
    className="neu__form-input mb-4"
    type="password"
    placeholder="Enter password"
    value={password}
    onChange={(e) => setPassword(e.target.value)}
  ></Form.Control>
</Form.Group>
<Form.Group controlId="confirmPassword">
  {/* <Form.Label>Confirm Password</Form.Label> */}
  <Form.Control
    className="neu__form-input mb-4"
    type="password"
    placeholder="Confirm password"
    value={confirmPassword}
    onChange={(e) => setConfirmPassword(e.target.value)}
  ></Form.Control>
</Form.Group>
<Button className="neu__button mt-4" type="submit" varient="primary">
  <i className="fas fa-user-plus"></i> &ensp; Register
</Button>
```

```
        </Form>
      )}

      <Row className="py-3">
        <Col>
          Have and Account?{" "}
          <Link to={redirect ? `/login?redirect=${redirect}` : "/login"}>
            Login
          </Link>
        </Col>
      </Row>
    </FormContainer>
  );
};

export default RegisterScreen;
```

CONNECT_DB

```
?> import mongoose from 'mongoose';
const connectDB = async ()=>{
    try {
        const conn = await mongoose.connect(process.env.MONGO_URI, {
            useUnifiedTopology: true,
            useNewUrlParser: true,
            useCreateIndex:true
        })
        console.log(`Mongodb Connected: ${conn.connection.host}`.blue.underline)
    } catch (error) {

        console.log(`Erorr:${error.message}`.red.underline.bold)
        process.exit(1)
    }
}

export default connectDB;
```

SERVER

```
import express from 'express's

import dotenv from 'dotenv'

import colors from 'colors'

import morgan from 'morgan';

import path from 'path'

import connectDB from './config/db.js'

import productRoutes from './routes/productRoutes.js'

import userRoutes from './routes/userRoutes.js'

import orderRoutes from './routes/orderRoutes.js'

import uploadRoutes from './routes/uploadRoutes.js'
import {errorHandler,notFound} from './middleware/errorMiddleware.js'
dotenv.config()


connectDB();
const app = express()


app.use(express.json())
```

```
if(process.env.NODE_ENV === 'development'){
  app.use(morgan('dev'))
}

app.get('/',(req,res)=>{
   res.send('API running..')
})

// routes
app.use('/api/products',productRoutes)
app.use('/api/users',userRoutes)
app.use('/api/orders',orderRoutes)
app.use('/api/upload',uploadRoutes)

app.get('/api/config/paypal', (req,res) => res.send(process.env.PAYPAL_CLIENT_ID) )

const __dirname = path.resolve()
app.use('/uploads', express.static(path.join(__dirname, '/uploads')))

// error handler for 404
app.use(notFound)
// error handler middleware
app.use(errorHandler)

const PORT = process.env.PORT || 5000

app.listen(PORT,()=>{
    console.log(`server running in ${process.env.NODE_ENV} mode in port
${PORT}`.yellow.bold)
})
```

TESTING & IMPLIMENTATION

6.1 Need for Testing

Testing is the major quality control measure and during software development. Its basic is to detect errors in the software. The system is tested to its accuracy, speed data entry and response system requires a test plan, which specifies conditions, different units to be tested and the manner in which modules to be integrated together. After a test plan has been developed, system testing begins with testing program modules separately followed by testing bundles modules as a unit.

6.2 Black Box Testing

The black box testing takes an external perspective of the test objective to derive the test case; this test can be functional or non-functional, though usually functional. The test designer selects valid and invalid input and determines the correct output. There is no knowledge of the test objects internal structure. This method of test design applicable to all levels of software testing; unit, integration, functional testing, system and acceptance. The higher the level, and hence bigger and more complex the box, the more one is forced to use black box testing to simplify. While this method can uncover unimplemented parts of the specification, one cannot be sure that all existing paths are tested. Black box testing is not a type of testing; instead, is a testing strategy, which does not need any knowledge of internal design or code etc. As the same "black box" suggests, no knowledge of internal logic or code structure is required.

6.3 White Box Testing

White box testing (A.K. A. Clear box testing, glass box testing or structural testing) uses an internal perspective of the system to design test cases based on internal structure. It requires programming skill to identify all through the software. The tester chooses test case input to exercise the path through the code and determines the appropriate output. In this type of testing inside the programmer himself identifies the control and flow condition testing inside the program and decides the test data so that it will pass through all the statements. By passing the data to the system will identify pick out as logical errors and conditional errors.

6.4 Unit Testing

Unit testing is the best method to verify the logic and coding, commutation/ functionality and error handing of each module. It involves testing the entire software, unit by unit as the online voting system is developed as different modules. Here white box testing has been adopted, which assures the full coverage of the code. The white box testing is rigorous activity. It is a case design method that uses the control structure of the procedural design to derive test cases.

6.5 Integration Testing

In this phase of software testing in which individual software modules are combined and tested as a group. If follow unit testing and precedes system testing. Integration testing takes as its inputs modules that have checked out by unit testing groups them larger aggregates, and delivers as its output the integrated system ready for system testing. The modules that have passed the unit testing like admin and membership detail view are combined together to check in information module are combined together to check if the prepared result is correct.

6.6 System testing

This provides the final assurance that the software meets all the functional, behavioral and performance requirements. The software is completely assembled as a package validation succeeds when the software function is a manner in which user wishes validation refer to the process of using software in live environment in order to find errors. During the course of validation, the system failure may occur and sometimes the coding as to be changed according to the requirement. Thus, the feedback from the validation phase generally produces changes in the software. Once the application was made all logical and interface errors, inputting a dummy date ensured that the software developed satisfied all the requirements of the user. This dummy date is known as a test case.

In this the system as a whole is tested against the design specifications; -

- system testing will be responsibility of testing engineer.
- system and integration testing shall be performed to ensure that the

  system work as a whole.
- system testing will be performed jointly by testing engineer.
- A system test plan will be prepared by the test manager.
- system testing will be performed in the test environment located on the server.
- system testing will include volume testing (with a high number of transactions processed) and stress testing (with transactions processed at height frequency)
- Result of tests will be recorded and where system components do not perform as expected, a test problem report will be raised.

6.7 Validation Testing

It involves checking processes which makes sure that software conforms to its specification and meets the needs of the customers. It starts requirements review, continues through design and code review to product testing. They are concerned with the analysis and checking of system representations such as the requirements documents, design diagram and the program code. They even involve executing an implantation of the software with test data and examining the outputs of the software to check that it is performing as required.

6.8 INTEGRATION TESTING

Integration testing is a systematic technique for constructing tests to uncover errors associated within the interface. In the project, all the modules. are combined and then the entire programmer is tested as a whole. In the integration-testing step, all the error uncovered is corrected for the next testing steps.

6.9 Output testing

After performing the validation testing, the next step is output testing of the proposed system. Since, no system would be termed as a useful unit if it does not produce the requested output in the specified format. Here the output format is considered in two ways.

One is the screen and the other is the printer. The output forms on the screen are found to be correct since the format was designed according to user needs. For hard copy also the output comes out as the specified requirements of the work. Here the output testing does not result in any correction in the system.

- User acceptance testing

The system has met the user's requirement in the following case.

- Data entry
- Error handling
- Reporting and correcting
- System output
- Implementation

Implementation involves:

- Software installation

In order to run the package "BLIX" the software's like Node and MongoDB server must be installed.

- User testing

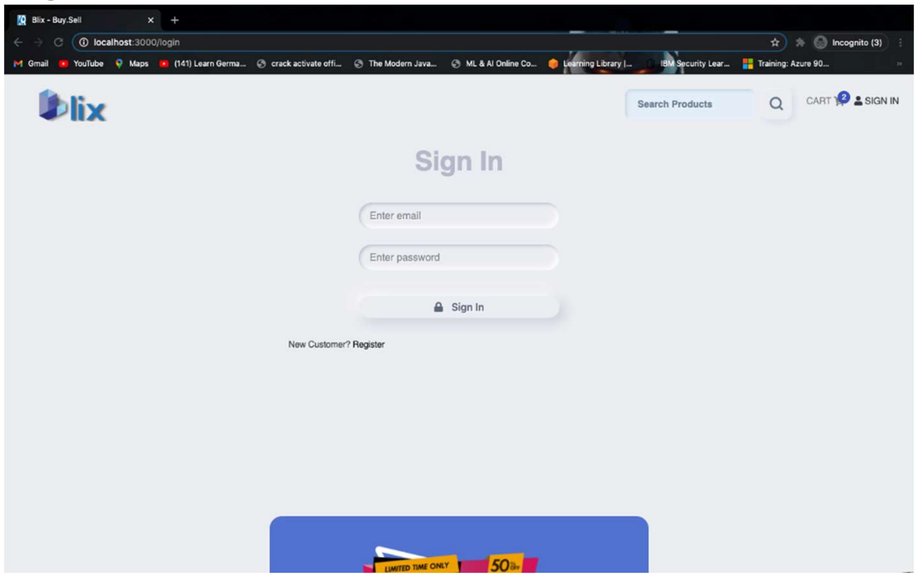
After installing the package, the user must be explained about the module functionalities and screen working.

- The tools used for implementation are Windows 10 operating system.
- MongoDB server as back-end interface.
- React as the format end interface.
- Triggers for various operations.
- Report generation using web browser data reporting designer.

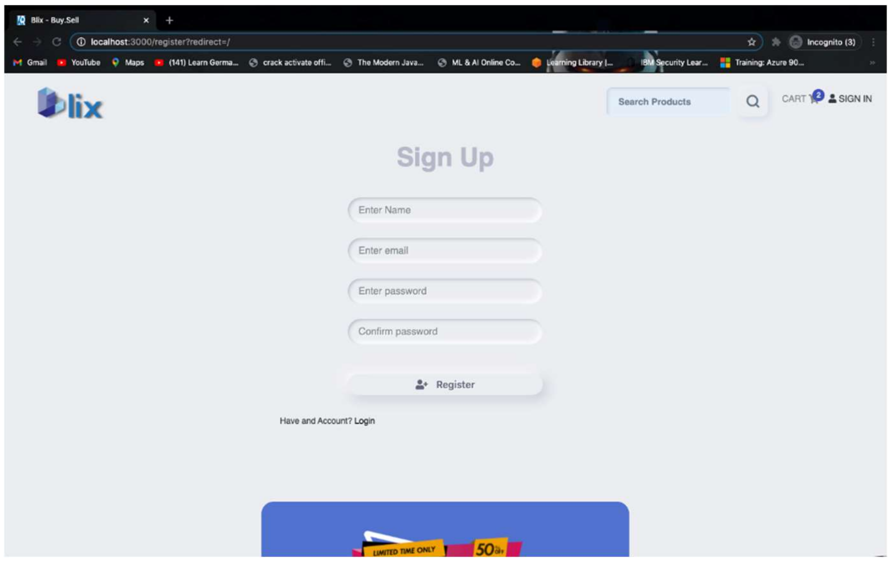
SCREENSHOTS

In this chapter, we are going to display some of the output screens of Art Gallery Management System to give the idea of how the software looks like in the execution part.

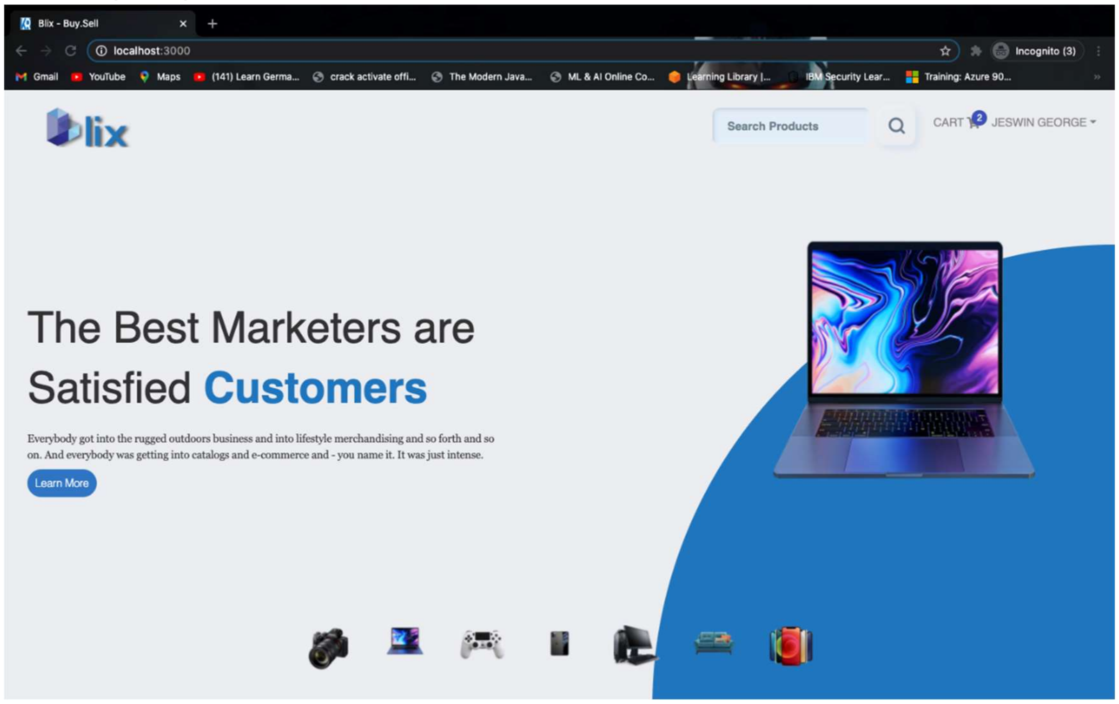
7.1 Signin Page:



7.2 Signup page:



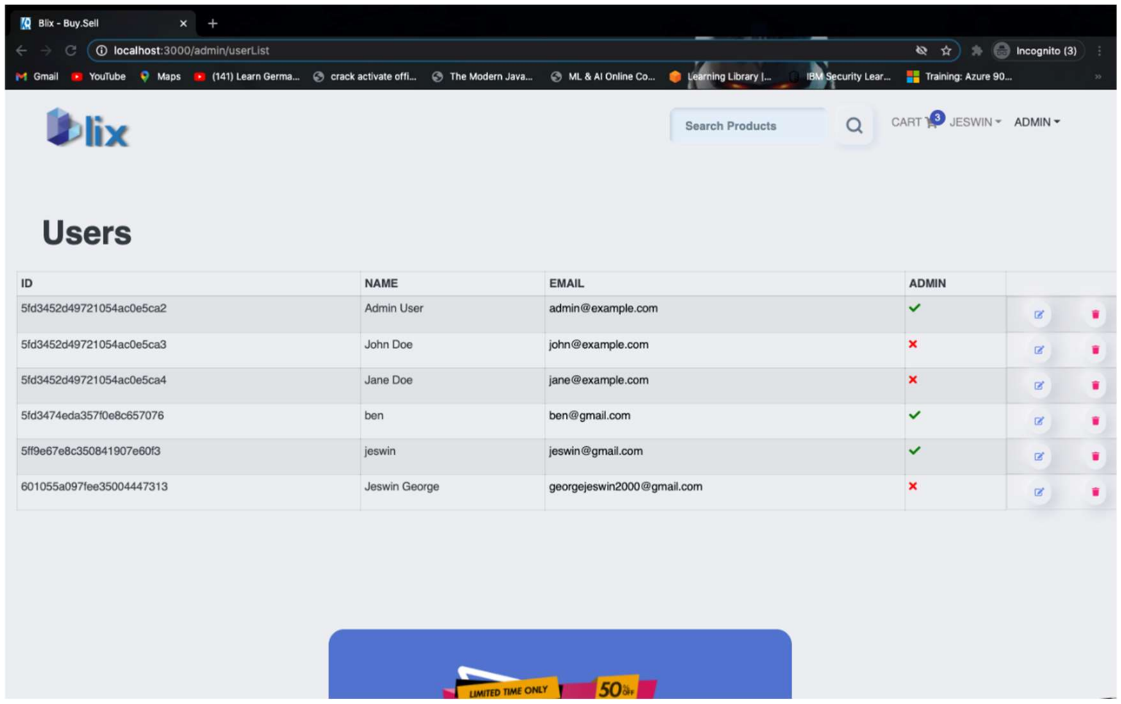
7.3 Landing Page :



The Best Marketers are Satisfied **Customers**

Everybody got into the rugged outdoors business and into lifestyle merchandising and so forth and so on. And everybody was getting into catalogs and e-commerce and - you name it. It was just intense.

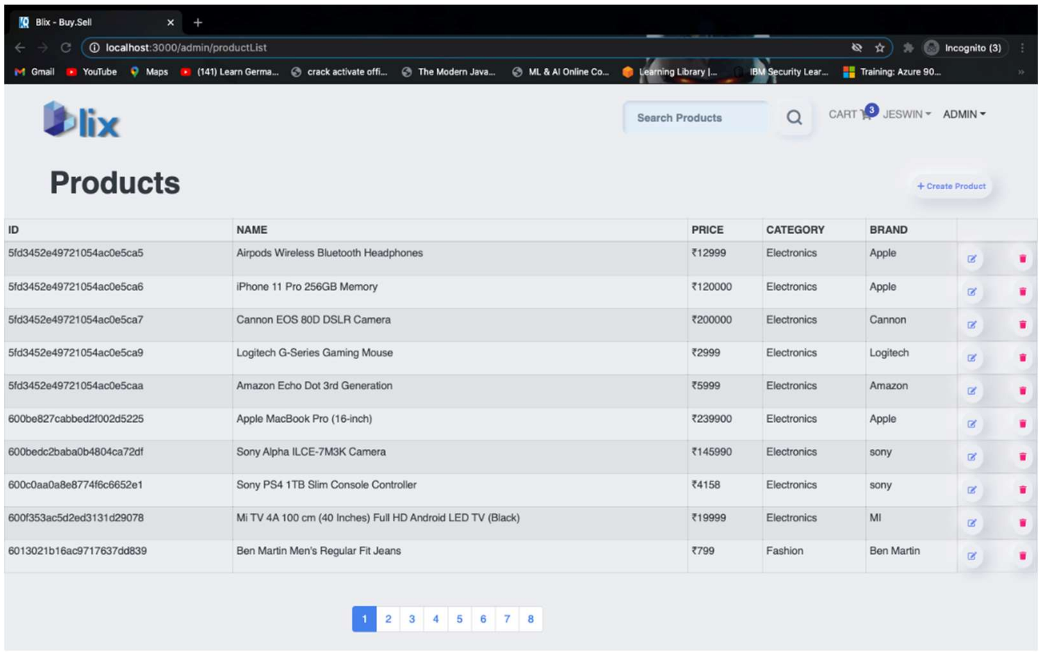
7.4 View Users Info:



Users

ID	NAME	EMAIL	ADMIN
5fd3452d49721054ac0e5ca2	Admin User	admin@example.com	✓
5fd3452d49721054ac0e5ca3	John Doe	john@example.com	✗
5fd3452d49721054ac0e5ca4	Jane Doe	jane@example.com	✗
5fd3474eda357f0e8c657076	ben	ben@gmail.com	✓
5ff9e67e8c350841907e60f3	jeswin	jeswin@gmail.com	✓
601055a097fee35004447313	Jeswin George	georgejeswin2000@gmail.com	✗

7.5 Admin Product List :

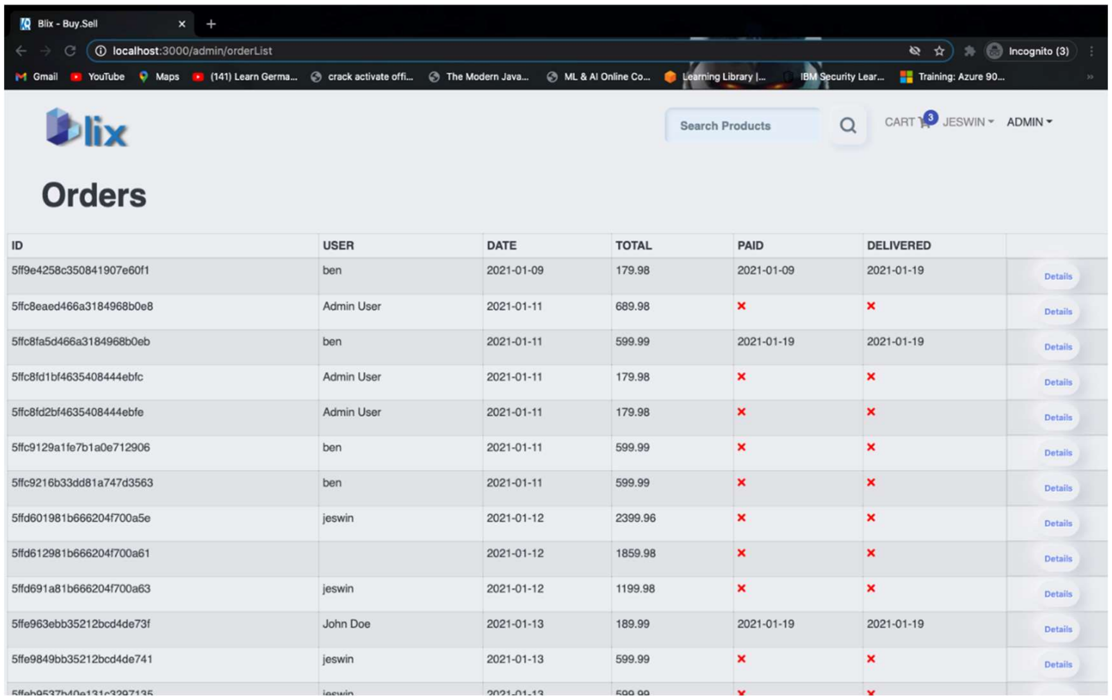


Products

ID	NAME	PRICE	CATEGORY	BRAND
5fd3452e49721054ac0e5ca5	Airpods Wireless Bluetooth Headphones	₹12999	Electronics	Apple
5fd3452e49721054ac0e5ca6	iPhone 11 Pro 256GB Memory	₹120000	Electronics	Apple
5fd3452e49721054ac0e5ca7	Cannon EOS 80D DSLR Camera	₹200000	Electronics	Cannon
5fd3452e49721054ac0e5ca9	Logitech G-Series Gaming Mouse	₹2999	Electronics	Logitech
5fd3452e49721054ac0e5caa	Amazon Echo Dot 3rd Generation	₹5999	Electronics	Amazon
600be827cabbed2f002d5225	Apple MacBook Pro (16-inch)	₹239900	Electronics	Apple
600bedc2baba0b4804ca72df	Sony Alpha ILCE-7M3K Camera	₹145990	Electronics	sony
600c0aa0a8e8774f6c6652e1	Sony PS4 1TB Slim Console Controller	₹4158	Electronics	sony
600f353ac5d2ed3131d29078	Mi TV 4A 100 cm (40 Inches) Full HD Android LED TV (Black)	₹19999	Electronics	MI
6013021b16ac9717637dd839	Ben Martin Men's Regular Fit Jeans	₹799	Fashion	Ben Martin

1 2 3 4 5 6 7 8

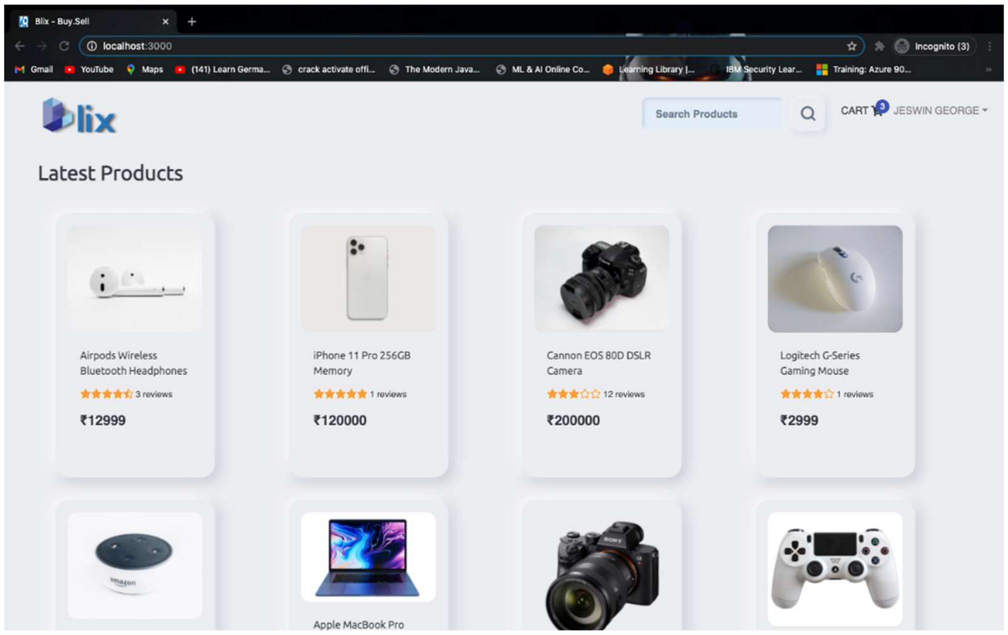
7.6 Admin Order List:



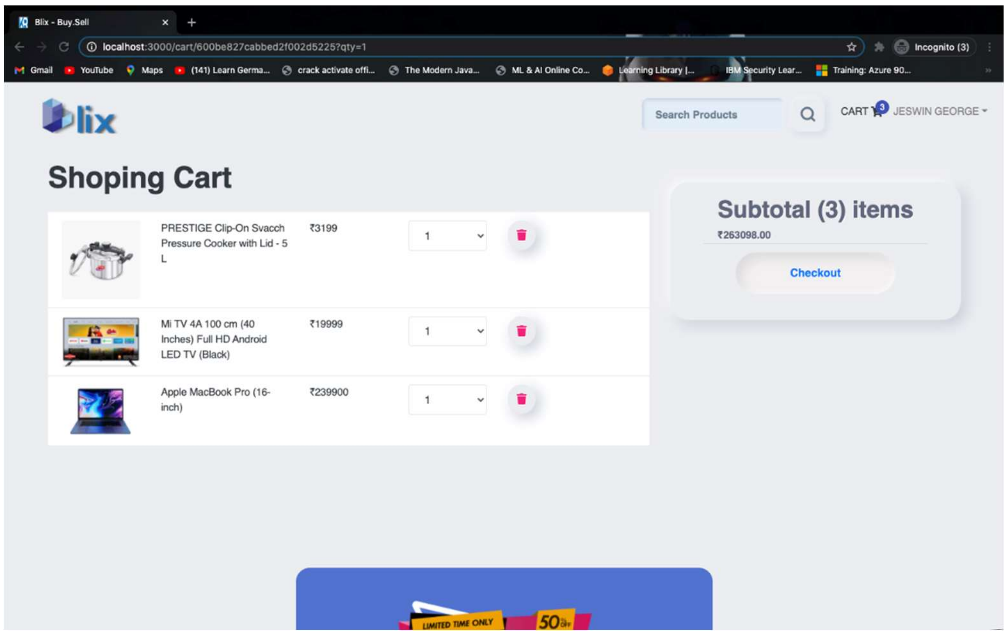
Orders

ID	USER	DATE	TOTAL	PAID	DELIVERED	
5ff9e4258c350841907e60f1	ben	2021-01-09	179.98	2021-01-09	2021-01-19	Details
5ffc8eaed466a3184968b0e8	Admin User	2021-01-11	689.98	✗	✗	Details
5ffc8fa5d466a3184968b0eb	ben	2021-01-11	599.99	2021-01-19	2021-01-19	Details
5ffc8fd1bf4635408444ebfc	Admin User	2021-01-11	179.98	✗	✗	Details
5ffc8fd2bf4635408444ebfe	Admin User	2021-01-11	179.98	✗	✗	Details
5ffc9129a1fe7b1a0e712906	ben	2021-01-11	599.99	✗	✗	Details
5ffc9216b33dd81a747d3563	ben	2021-01-11	599.99	✗	✗	Details
5ffd601981b666204f700a5e	jeswin	2021-01-12	2399.96	✗	✗	Details
5ffd612981b666204f700a61		2021-01-12	1859.98	✗	✗	Details
5ffd691a81b666204f700a63	jeswin	2021-01-12	1199.98	✗	✗	Details
5ffe963ebb35212bcd4de73f	John Doe	2021-01-13	189.99	2021-01-19	2021-01-19	Details
5ffe9849bb35212bcd4de741	jeswin	2021-01-13	599.99	✗	✗	Details
5ffeb9537b40e131c3297135	jeswin	2021-01-13	599.99	✗	✗	

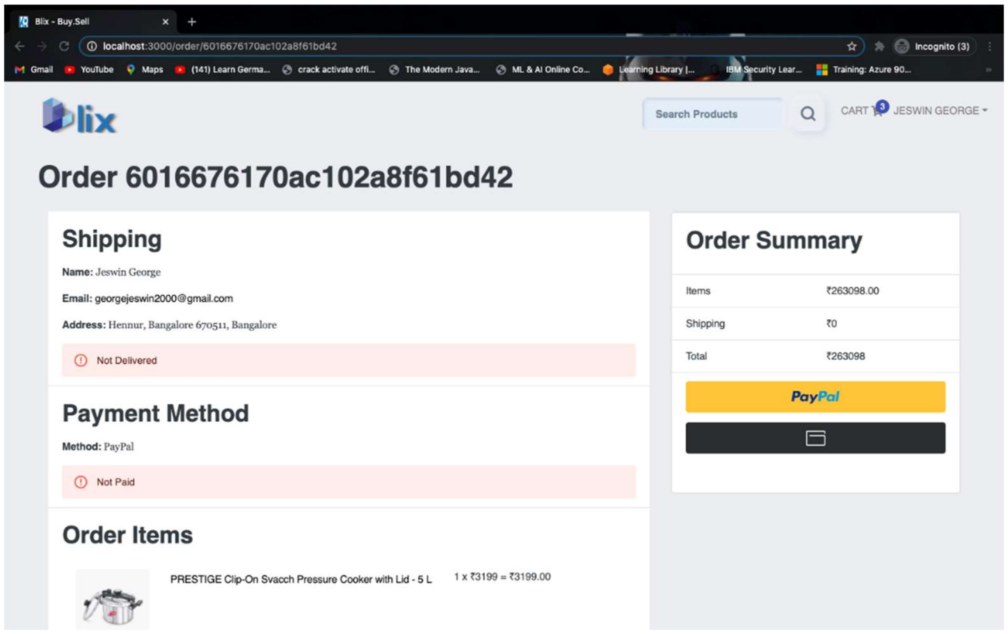
7.7 User Product List :



7.8 User Shopping Cart:


Blix - Buy.Sell
localhost:3000/cart/600be827cabbed2f002d5225?qty=1
Search Products
CART 3 JESWIN GEORGE
Shoping Cart
PRESTIGE Clip-On Svacch Pressure Cooker with Lid - 5 L
₹3199
1
Mi TV 4A 100 cm (40 Inches) Full HD Android LED TV (Black)
₹19999
1
Apple MacBook Pro (16-inch)
₹239900
1
Subtotal (3) items
₹263098.00
Checkout
LIMITED TIME ONLY
50

7.9 User Order Payment:



Order 6016676170ac102a8f61bd42

Shipping

Name: Jeswin George

Email: georgejeswin2000@gmail.com

Address: Hennur, Bangalore 670511, Bangalore

Not Delivered

Payment Method

Method: PayPal

Not Paid

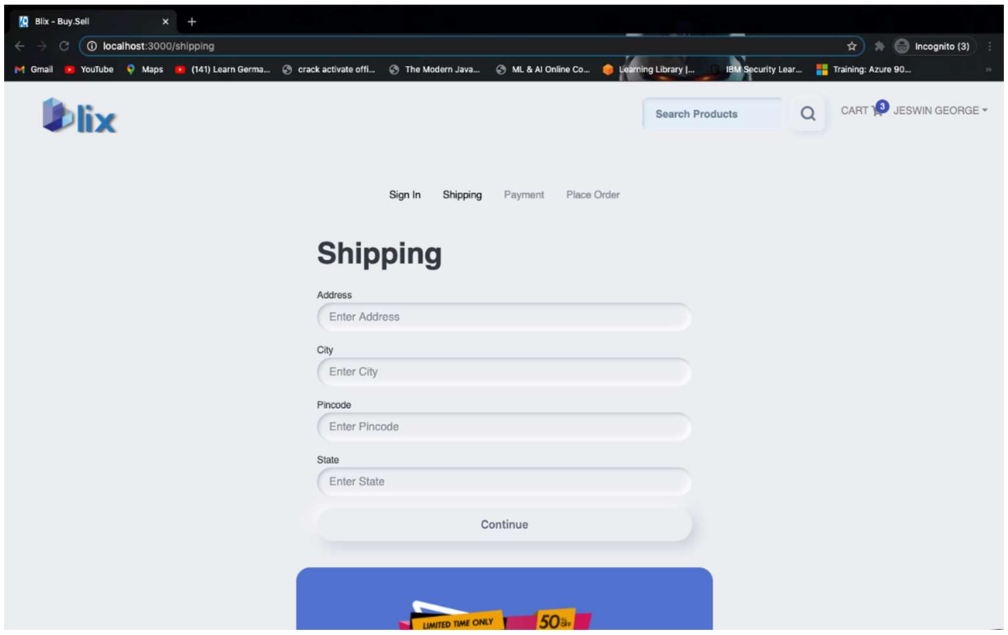
Order Items

PRESTIGE Clip-On Svacch Pressure Cooker with Lid - 5 L 1 x ₹3199 = ₹3199.00

Order Summary

Items	₹263098.00
Shipping	₹0
Total	₹263098

7.9 User Shipping Address:



Shipping

Address: Enter Address

City: Enter City

Pincode: Enter Pincode

State: Enter State

Continue

CONCLUSION

The internet has become a major resource in modern business, thus electronic shopping has gained significance not only from the entrepreneur's but also from the customer's point of view. For the entrepreneur electronic shopping generates new business opportunities and for the customer, it makes comparative shopping possible.

As per the survey, most consumers of online stores are impulsive and usually decide to stay on a site within the first few seconds. Website design is like a shop interior. If the shop looks like hundreds of other shops the customer is most likely to skip to the other sites. Hence, we have designed the project to provide the user with easy navigation, retrieval of data and necessary feedback as much as possible. In this project, the user is provided with an e-commerce web application that can be used to buy various categories of things like devices, electronics, fashion products etc. To implement this as a web application we used MERN stack as the technology. MERN stack as a full stack has several advantages such as enhanced performance, scalability, built in security, and simplicity. It also provides knowledge about the latest technology used in developing web enabled application and client server technology that will be in great demand in future. This will provide better opportunities and guidance in future in developing projects independently

BIBLIOGRAPHY

Eloquent JavaScript	**:**	**Marijn Haverbeke**
Full Stack React	**:**	**Anthony Accomazzo, Nate Murray**
Beginning Node	**:**	**Basarat Ali Sye**
Scaling MongoDB	**:**	**Kristina Chodorow**

WEB SITES:

YouTube, Udemy, Coursera, Google, Stack Overflow, GitHub

For React, Node js Express and mongoDB installation:

https://reactjs.org

http://nodejs.org

http://expressjs.com

https://www.mongodb.com/2