

A Project Report on  
**“E-COMMERCE WEBSITE”**

*Submitted in partial fulfilment  
for the award of the Degree in*

**BACHELOR OF COMPUTER APPLICATION**

*By*

**“SUBRATA KUMAR GHOSH”**

**AJU/190206/BCA/021**

Under the guidance of  
**MR. DEBANJAN GHOSH**



**ARKA JAIN UNIVERSITY**

**JAMSHEDPUR, JHARKHAND**

**DEPARTMENT OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY  
2019-22**



**ARKA JAIN UNIVERSITY**

A Project Report on

# **“E-COMMERCE WEBSITE”**

*Submitted in partial fulfilment  
for the award of the Degree in*

**BACHELOR OF COMPUTER APPLICATION**

**GUIDED BY:**

**MR. DEBANJAN GHOSH**

**PRESENTED BY:**

**SUBRATA KUMAR GHOSH**

**Enrollment Number: AJU/190206**

**ARKA JAIN UNIVERSITY**

**JAMSHEDPUR, JHARKHAND**

**DEPARTMENT OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY**

**2019-22**

**A Project Report on**  
**“E-COMMERCE WEBSITE”**

*Submitted in partial fulfilment  
for the award of the Degree in*

**BACHELOR OF COMPUTER APPLICATION**

*By*

**“SUBRATA GHOSH”**

**AJU/190206/BCA/021**

**Under the guidance of  
MR. DEBANJAN GHOSH**



**ARKA JAIN UNIVERSITY**

**JAMSHEDPUR, JHARKHAND**

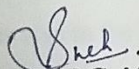
**DEPARTMENT OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY  
2019-22**

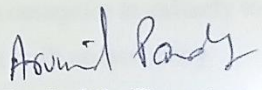


**ARKA JAIN UNIVERSITY**  
**JAMSHEDPUR, JHARKHAND**  
**DEPARTMENT OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY**

**CERTIFICATE**

This is to certify that the project entitled “E-COMMERCE WEBSITE”, is bona-fide work of **SUBRATA KUMAR GHOSH** bearing Registration Number: **AJU/19206** submitted in partial fulfillment of the requirement for the award of degree in **BACHELOR OF COMPUTER APPLICATION** from **ARKA JAIN UNIVERSITY, JHARKHAND**.

  
**Internal Guide**

  
**Head of the Department**

**Date:** 26/05/2022



# ABSTRACT

Online Shopping play a great importance in the modern business environment. Dream gate has opened the door of opportunity and advantage to the firms. This paper analysed the different issue of online shopping. The project aims to provide theoretical contribution in understanding the present status of online shopping. The Study Discuss the consumers' online shopping behaviours. Paper also identify the problems face by the consumers when they want to accept internet shopping. Present paper is an expressive study based on the detailed review of earlier pertinent studies related to the various concepts of online shopping to discover the concept of online shopping. Solitude and safety risk emerge regularly as a reason for being cautious about internet shopping.

Shopping convenience, information seeking, social contact, and diversity affect the consumer attitude towards online shopping. The impossibility of product testing, problems with complaints, product return and misuse of personal data are the main doubts regarding on-line shopping. Keywords E-Commerce is now seen as a reality for many businesses and a normal part of a business plan. The immediate benefits, in terms of cost savings, efficiencies and enhanced profitability are clear at every stage in the supply chain. Adopting e-business is no longer a competitive advantage, but a normal business process, without which an enterprise is unlikely to survive in the New economy . Year 2000 saw many Dot-com companies built up and many companies going into E-commerce however now it is a different story, more and more companies are failing, and investors are becoming cautious to invest money into Internet ventures.

# ACKNOWLEDGEMENT

I would like to express my sincere gratitude to several individuals and organization for supporting me throughout the completion of my project.

First, I wish to express my sincere gratitude to my mentor Mr. DEBANJAN GHOSH for her enthusiasm patience, insightful comments, helpful information, practical advices and unceasing ideas that have helped me tremendously at all times in my Project and writing of this thesis. her immense knowledge, profound experience and professional expertise in Backend has enabled me to complete this project successfully. Without her support and guidance, this project would not have been possible.

I am also thankful to our respected H.O.D and all faculty members for loving inspiration and timely guidance. I also wish to express my sincere thanks to the Department of Computer science & Information technology of ARKA JAIN UNIVERSITY for accepting this project.

Thanks for all your encouragement!



## INTERNSHIP APPRECIATION LETTER

**Certificate No:** WON-SKG-0421-002

**Date:** 03/05/2021

### TO WHOMSOEVER IT MAY CONCERN

This is to certify that **Mr. Subrata Kumar Ghosh** has successfully completed a 30 Days Internship Program with our organization in **Android App Development with Java**. During the span of training with us, the candidate has been actively and diligently involved in the projects and tasks assigned to him. The candidate has exclusively worked on the project which implicated the practical execution of the courseware and the skills the student has learned with us as a part of E-Cell, VNIT-Nagpur & SKILLiGENCE Internship Program during **02/04/2021** to **02/05/2021**.

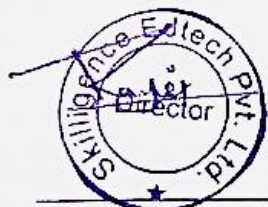
So, hereby with this letter, we acknowledge that the student has worked on the following projects:

### **CHAT APPLICATION WITH REAL TIME DATE USING FIREBASE DATA**

We found him sincere, hardworking, technically sound, and result-oriented. He has also shown an impressive skill towards being a diligent task handler as well as collaborating with the team and being a seamless part of the team during his tenure. He has shown the ability to incorporate the ideologies and concepts of the company and being a strong team player and contributing towards successful result-oriented goals.

We take this opportunity to thank him for being part of our organization and wish him all the best for his future endeavors.

Best Regards,



**Mr. Ranjit Kumar**  
Director, DIN 08943960  
SKILLiGENCE EdTech Pvt. Ltd.

---

**SKILLiGENCE EDTECH PVT. LTD.**

## DECLARATION

We hereby declare that the Project entitled "E-COMMERCE WEBSITE" done at

ARKA JAIN UNIVERSITY has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university.

This project is done in partial fulfilment of the requirement for the award of degree of BACHELOR OF COMPUTER APPLICATION to be submitted as final semester project as part of our curriculum.

  
Signature

**Subrata Kumar Ghosh**



# Table of Contents

Certificate	<b>Page No:</b>
Abstract	
Acknowledgement	
Declaration	
Chapter 1: Introduction	11-12
1.1 Objective	
1.2 Purpose & Scope	
Chapter 2: System Analysis	13-15
2.1 Existing System	
2.2 Purpose System	
2.3 Requirement Analysis	
2.4 Hardware Requirement & Software Requirement	
2.5 Justification of Selection of Technology	
2.6 Database normalization & DFD	
Chapter 3: System Design	16-27
3.1 Module Division	
3.2 Data Dictionary	
3.3 ER Diagram	
Chapter 4 : Coding	28-69
Chapter 5 : implementation and testing	70-71
5.1 Testing Approach	
5.2 Unit Testing	
5.3 Integration Testing	
Chapter 6: Result & Discussion	72-74
Chapter 7: Conclusion	75
Chapter 8: Reference	76

# Chapter 1

## Introduction

An E-commerce website requires appropriate strategy of successful design and implementation. Everything is required to plan from scratch to end of website. The e-commerce sector is seen the exponential growth thus a new option will easily part of this regatta of commercial website. The e-commerce website will feature the online shopping facility of various fashion products under a single web space. The proposed web application will allow business personnel to make their total business using it and increase their reachability thousands of times more than today they have, over the internet. It will allow multiple shopping vendors to sale their products online. The product management in the system will be done in the form of categories. The safety of information is the main requirement of the system and will be handling according to that. To formulate this project first task is to do is cost estimation. For probabilistic assessment of the project cost estimation is required. Cost estimation covers the accurate estimations of cost and effort required for the project.

As a project manager and developer as well, it's estimates are defined to early stage in the project. Cost estimation in application development project includes the set of procedures and techniques that will be utilized, required to produce by organisation for development. The available resources of a company are also affecting the cost estimation. It will be very complex project. To demonstrate knowledge learnt in class, tech communities and online materials, The environment variants depend on the further requirements of the ecommerce web application.

.

### Objective

Online Shopping play a great importance in the modern business environment. Dream gate has opened the door of opportunity and advantage to the firms. This paper analysed the different issue of online shopping. The project aims to provide theoretical contribution in understanding the present status of online shopping. The Study Discuss the consumers' online shopping behaviours. Paper also identify the problems face by the consumers when they want to accept internet shopping. Present paper is an expressive study based on the detailed review of earlier pertinent studies related to the various concepts of online shopping to discover the concept of online shopping. Solitude and safety risk emerge regularly as a reason for being cautious about internet shopping. Shopping convenience, information seeking, social contact, and diversity affect the consumer attitude towards online shopping. The impossibility of product testing, problems with complaints, product return and missus of personal data are the main doubts regarding on-line shopping.

**Keywords :** E-Commerce is now seen as a reality for many businesses and a normal part of a business plan. The immediate benefits, in terms of cost savings, efficiencies and enhanced profitability are clear at every stage in the supply chain. Adopting e-business is no longer a competitive advantage, but a normal business process, without which an enterprise is unlikely to survive in the New economy . Year 2000 saw many Dot-com companies built up and many companies going into E-commerce however now it is a different story, more and more companies are failing, and investors are becoming cautious to invest money into Internet ventures.

## **Purpose & Scope**

### **Purpose**

Traditionally, customers are used to buying the products at the real, in other words, factual shops or supermarkets. It needs the customers to show up in the shops in person, and walk around different shopping shelves, and it also needs the owners of shops to stock, exhibit, and transfer the products required by customers. It takes labour, time and space to process these operations. Furthermore, the spread of the Covid-19 pandemic has caused a lot of changes in our lifestyle, people fearing to get outside their homes, transportation almost shut down and social distancing becoming all the more important. Big to small scale business that relied on the traditional incur a lot of consequence due to the lockdown issues. Some tend to move towards using social media platforms like Facebook to sell their product. However, the social media platforms have been beneficial for marketing purposes alone but leaves the whole task of customer and massive order management via direct messaging .

### **Scope**

#### **Dream gate**

system provides a solution to reduce and optimize these expenses. Authorized Customers do not need to go to the factual shops to choose, and bring the products they need by hands. They simply browse their Personal computers or cellphones to access shops, and evaluate the products description, pictures on the screen to choose products. In addition, the owners of the shop do not need to arrange or exhibit their stock products. They just input the description, prices of products, and upload their pictures. Simply, both customers and shop owners do not need to touch the real products in the whole process of shopping, and management. In the end the logistic centre will distribute the products required by customers, or products ordered by shop owners to their locations. The customers are able to track the status of their orders until delivery, after which they can leave a review of the type of service they received.

# Chapter 2

## System Analysis

As far as the project is developed the functionality is simple, the objective of the proposal is to strengthen the functioning of Audit Status Monitoring and make them effective and better. The entire scope has been classified into five streams known as Coordinator Level, management Level, Auditor Level, User Level and State Web Coordinator Level. The proposed software will cover the information needs with respect to each request of the user group viz. accepting the request, providing vulnerability document report and the current status of the audit.

### Existing System

In the existing system, each task is carried out manually and processing is also a tedious job. In previous system travelers were maintaining time table details manually in pen and paper, which was time taking and costly. The travelers are not able to achieve their need in time and also the results may not be accurate. Because of the manual maintenance there are a number of difficulties and drawbacks exist in the system. Some of them are

Drawbacks of the Existing System:

- Increased transaction leads to increased source document and hence maintenance becomes difficult.
- If any admin, user entry is wrongly made then the maintenance becomes very difficult.

### Proposed System

Reports said that customers can take enjoy online shopping for 24 hour per day. Consumers can purchase any goods and services anytime at everywhere. Online shopping is user friendly compared to in store shopping because consumers can just complete his requirements just with a click of mouse without leaving their home.

- Online shopping has some advantages like below  
Save the Time of the consumers.

- They can purchase any time anywhere



- They can compare the price with the others retailers very easily.
- Compare the advertising price and actual price
- They can easily track their product

- They can use cash back policy
- They can purchase the product from the foreign marketers.

## **Requirement Analysis**

### **Hardware Requirements:**

- PIV 2.8 GHz Processor and Above
- RAM 512MB and Above
- HDD 40 GB Hard Disk Space and Above

### **Software Requirements:**

- WINDOWS OS (XP / 2000 / 200 Server / 2003 Server)
- Microsoft visual studio 2017
- Web server software
- Firebase
- MERN STACK

## **Justification of Selection of Technology**

The following is the desired functionality of the new system. The proposed project would cover:

### **Customer Module**

- Customer can view/search products without login.
- Customer can also add/remove product to cart without login (if customer try to add same product in cart. It will add only one)
- When customer try to purchase product, then he/she must login to system.
- After creating account and login to system, he/she can place order.
- \*If customer click on pay button, then their payment will be successful and their order will be placed.
- Customer can check their ordered details by clicking on orders button.
- Customer can see the order status (Pending, Confirmed, Delivered) for each order
- Customer can Download their order invoice for each order
- Customer can send feedback to admin (without login)

### **Admin Module**

- Admin can provide username, email, password and your admin account will be created.
- After login, there is a dashboard where admin can see how many customers is registered, how many products are there for sale, how many orders placed.
- Admin can add/delete/view/edit the products.

# Chapter 3

## System Design

### Module Division

The module division contains the following things:

- Login Page: The login page contains the username and the password.
- Sign up Page: If the user is new to the application, sign up page will open for them.
- Forget Password: After sign up, suppose you forget your password that get, password recovery page will open.
- Loading frame: After Sign in, a loading frame will pop-up on the screen.
- Dashboard: Main frame of the Project will open – It is basically the dashboard from where the whole project will be processed.
- Payment: There will be a payment gateway in order to make the payment of the Booking.

### Data Dictionary

A Data Dictionary is a collection of names, definitions, and attributes about data elements that are being used or captured in a database, information system, or part of a research project. Data dictionary, or metadata repository, as defined in the IBM Dictionary of Computing, is a "centralized repository of information about data such as meaning, relationships to other data, origin, usage, and format". Oracle defines it as a collection of tables with metadata.

## Tables:

### product table:

Field_Name	Data_Type	Field_length	Constraints	Description
Id	int	20	not null	Name of the user for e-commerce account.
Name	Varchar	20	not null	Name of the user
category	Varchar	20	not null	Product category
Price	decimal	50	not null	Price of the product
discount_id	int	30	not null	Discount on the product

### user table:

Field_Name	Data_Type	Field_length	Constraints	Description
Id	Int	30	not null	Name of the user
username	Varchar	30	not null	Name of the user
password	text	10	not null	Password for account
first_name	varchar	10	not null	First name of user
last_name	Varchar	10	not null	Last name of user
address	Varchar	20	not null	Address of the user
phone	Int	20	not null	Phone no of the user
created_at	Varchar	20	primary key	Created id on site
modified_at	Varchar	20	not null	Modified details in id
deleted_at	Varchar	20	not null	Deleted id on site



**payment table:**

Field_Name	Data_Type	Field_length	Constraints	Description
Id	Int	30	not null	Name of the user
Order_id	Int	30	not null	Order id of the product
amount	Int	10	not null	Amount of the product
provider	Varchar	30	not null	Provider of the product
status	Varchar	30	not null	Status of the product
created_at	Varchar	20	not null	Created id on site
modified_at	Varchar	20	not null	Modified details in id

**cart table:**

Field_Name	Data_Type	Field_length	Constraints	Description
Id	Int	30	not null	Name of the user for
session_id	Int	40	not null	Id for user's identity
product_id	Int	40	Primary key	Id number of user's product
quantity	Varchar	50	not null	Quantity of the product
created_at	Varchar	20	not null	Created list of order
modified_at	Varchar	20	not null	Modified list of order

**shopping table:**

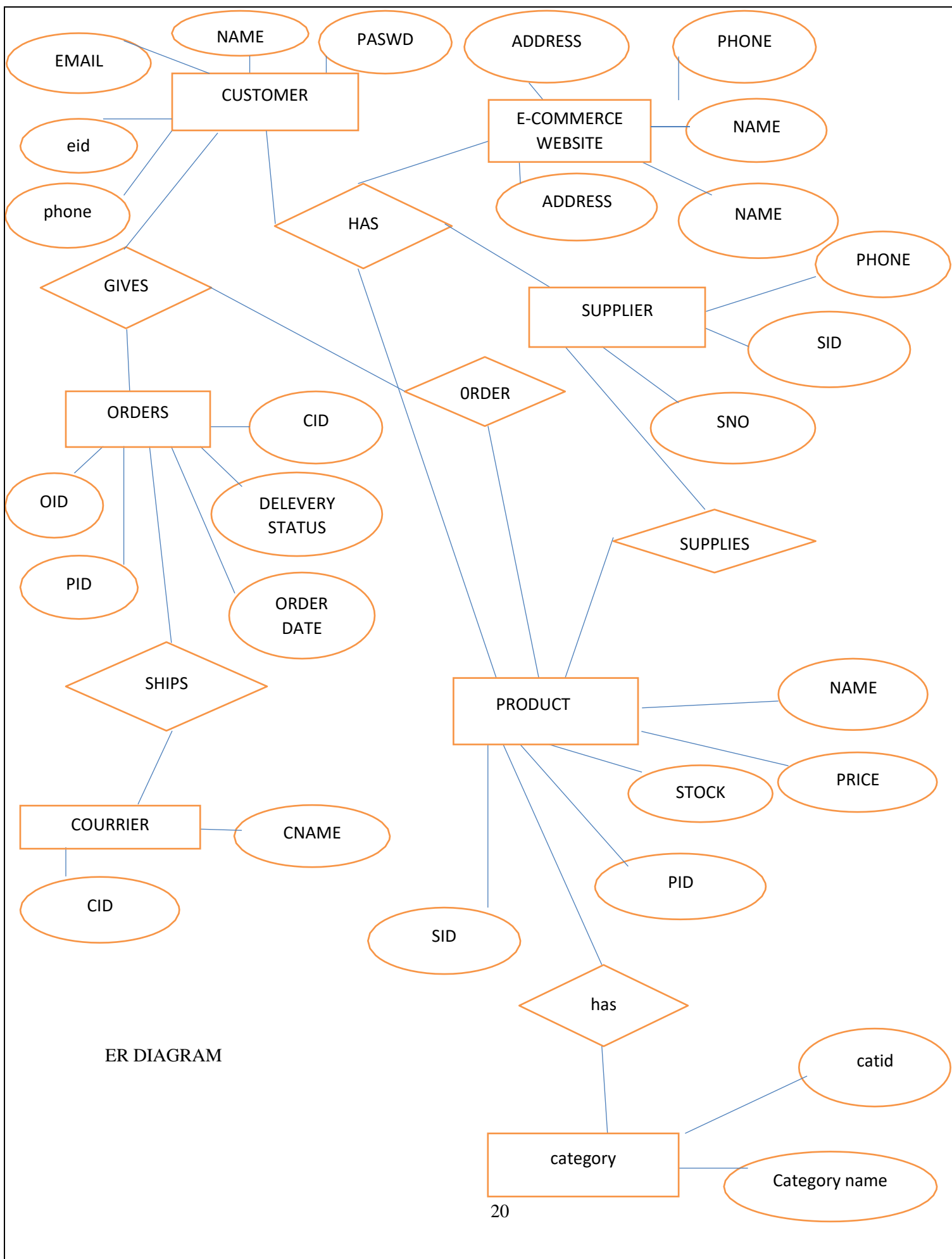
Field_Name	Data_Type	Field_length	Constraints	Description
Id	Int	40	not null	Name of the user For site
user_id	Varchar	25	not null	User for identity
Tota l	decimal	20	not null	Total of the product
created_at	Varchar	20	not null	Created list of product

**Order details table:**

Field_Name	Data_Type	Field_length	Constraints	Description
Id	Int	40	not null	Name of the user For site
user_id	Int	25	not null	User for identity
Tota l	decimal	20	not null	Total of the product
created_at	Varchar	20	not null	Created list of product
payment_id	Int	20	not null	Payment for product

**Oder items table:**

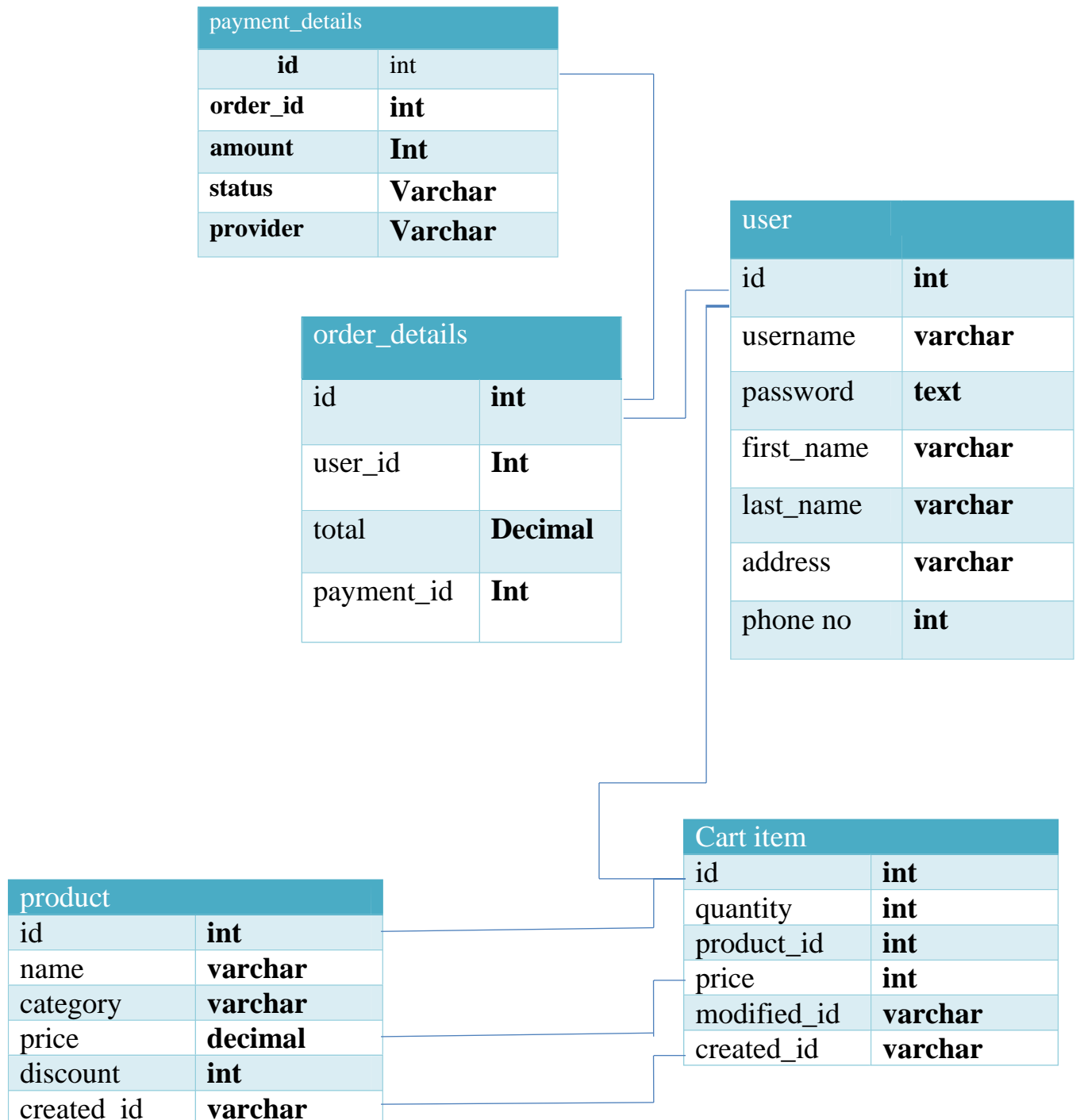
Field_Name	Data_Type	Field_length	Constraints	Description
Id	Int	40	not null	Id of the user
order_id	Int	25	not null	Order id of the user
product_id	Int	20	not null	Product id of the user





## Database Normalization

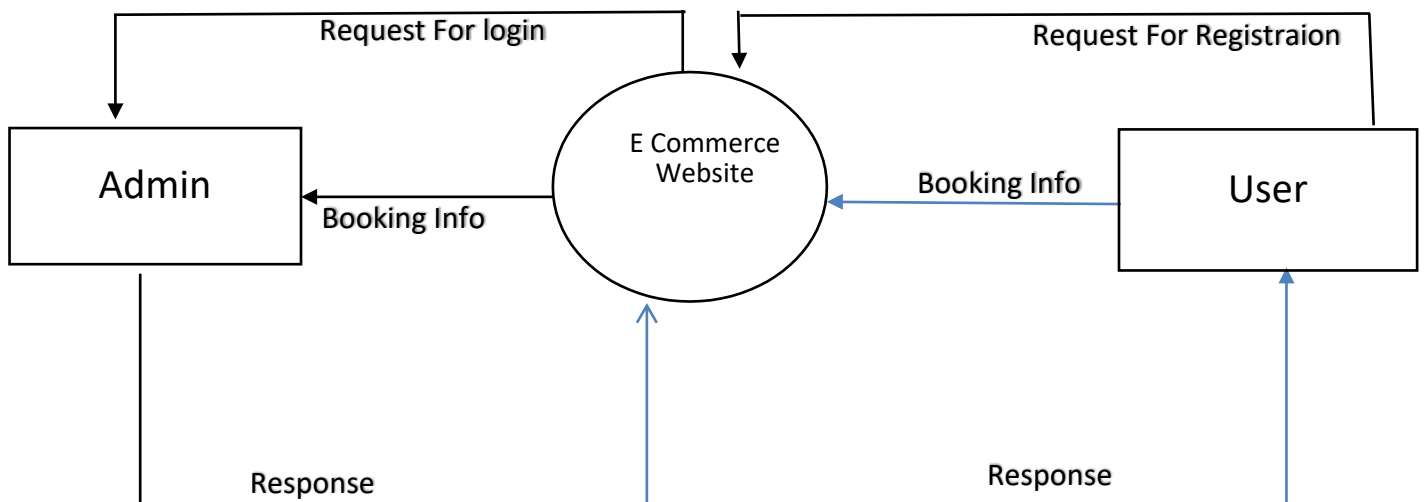
**Normalization** is a database design technique that reduces data redundancy and eliminates undesirable characteristics like insertion, update and Deletion Anomalies. Normalization rule divides larger tables into smaller tables and link them using relationships. The purpose of Normalization in SQL is to eliminate redundant (repetitive) data and ensure data is stored logically.



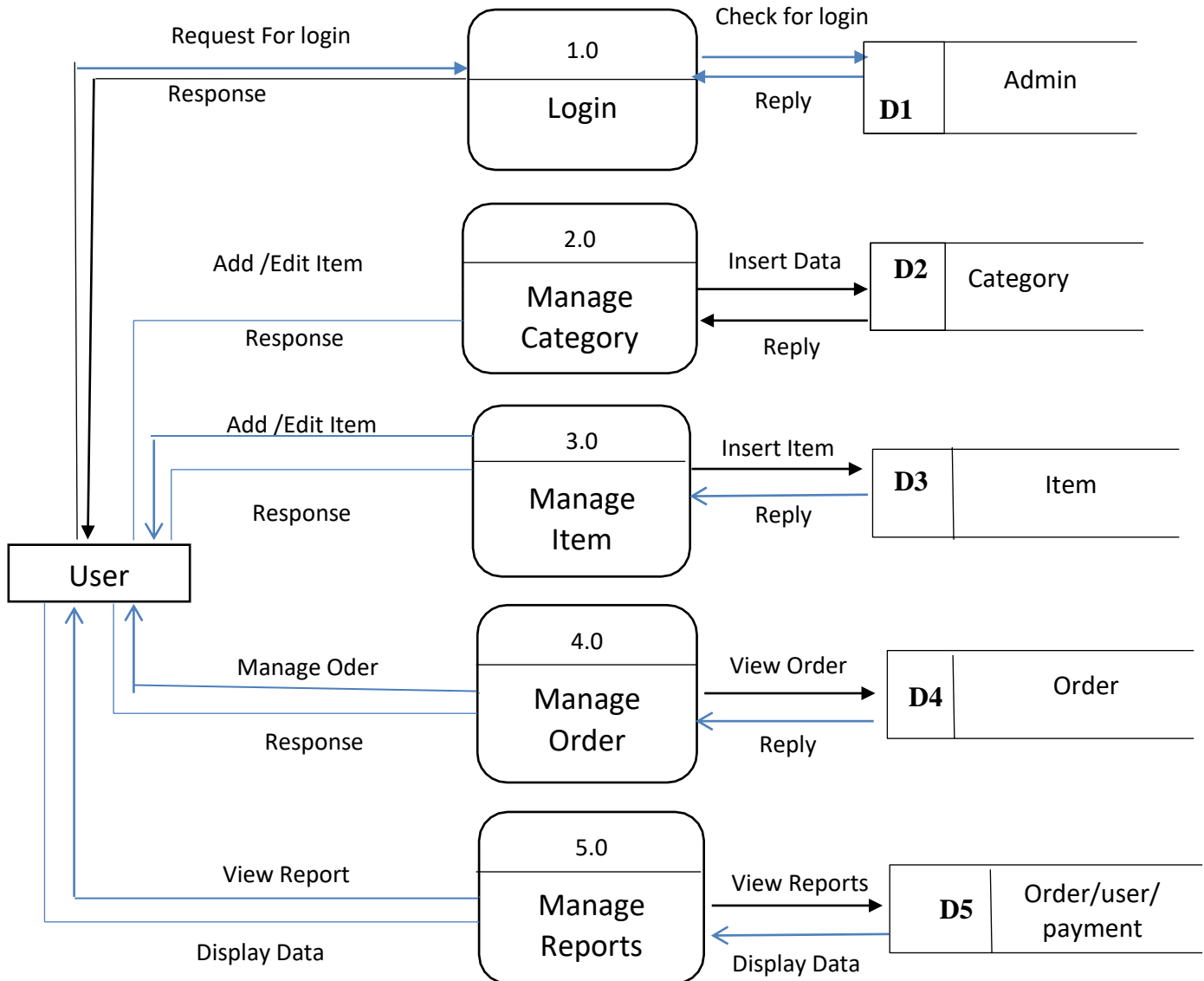
## Data Flow Diagram (DFD)

A data flow diagram is graphical tool used to describe and analyze movement of data through a system. These are the central tool and the basis from which the other components are developed. The transformation of data from input to output, through processed, may be described logically and independently of physical components associated with the system. These are known as the logical data flow diagrams. The physical data flow diagrams show the actual implements and movement of data between people, departments and workstations.

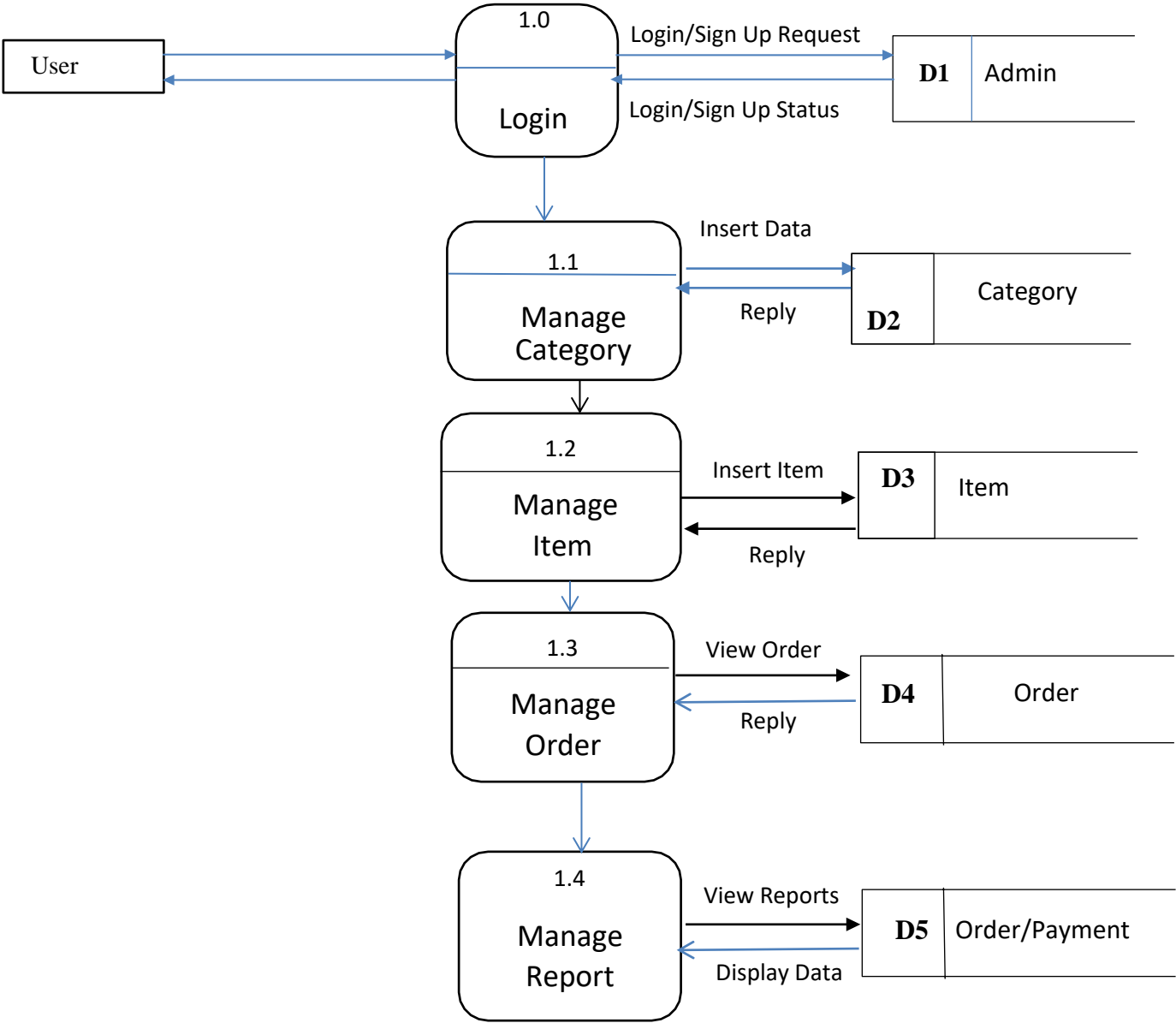
### DFD Level 0



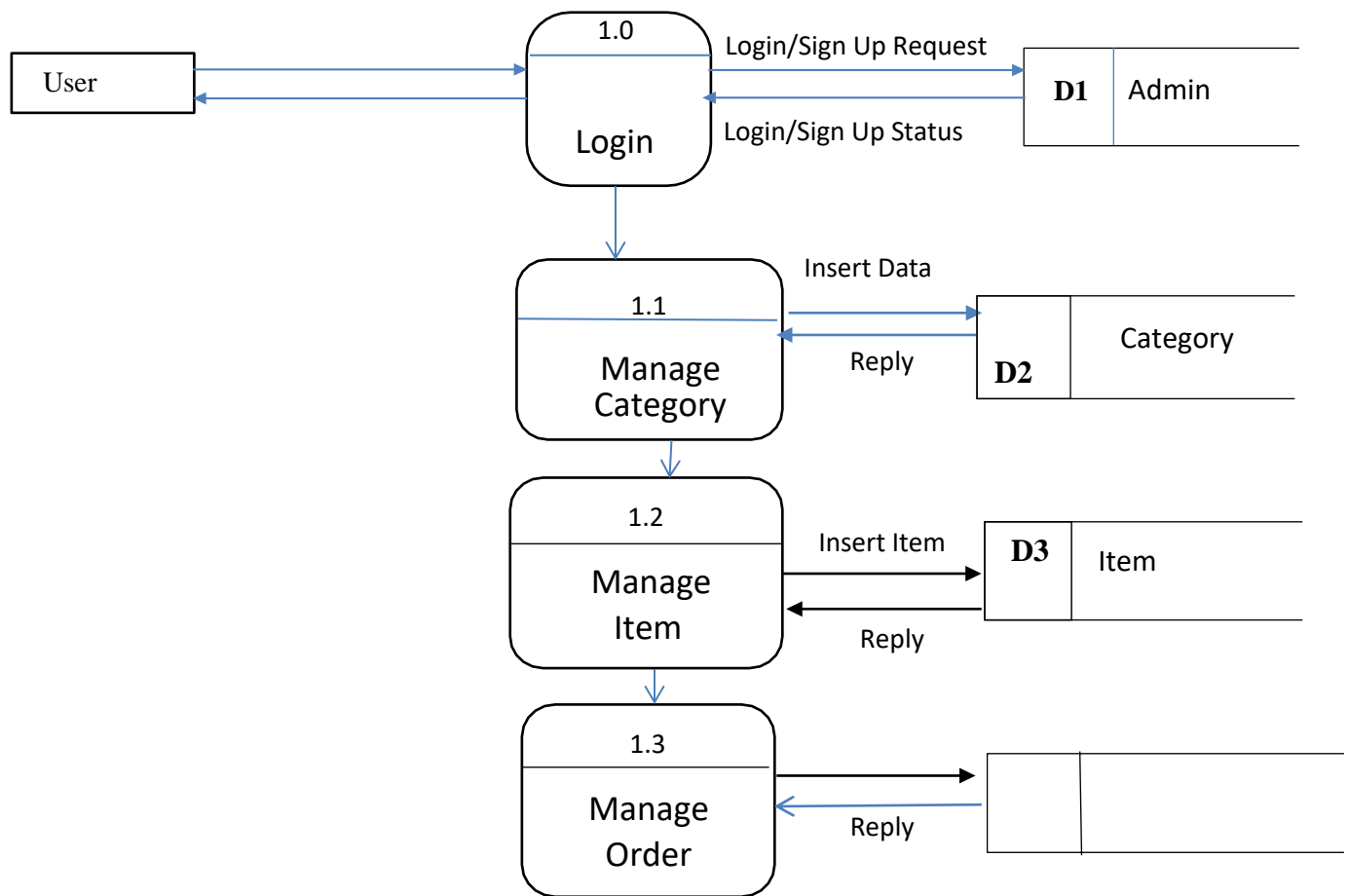
## DFD Level 1



**DFD Level 2**



### DFD Level 2 Continued:





# Chapter 4

## CODING

### Home.js

```
import React from "react";
import "./Home.css";
import Product from "./Product";

function Home() {
  return (
    <div className="home">
      <div className="home__container">
        

        <div className="home__row">
          <Product
            id="12321341"
            title="The Lean Startup: How Constant Innovation Creates Radically Successful Businesses Paperback"
          />
        </div>
      </div>
    </div>
  );
}
```

price={ 11.96}

```

        rating={5}

        image="https://images-na.ssl-images-
amazon.com/images/I/51Zymoq7UnL._SX325_BO1,204,203,200_.jpg"

    />

    <Product

        id="49538094"

        title="Kenwood kMix Stand Mixer for Baking, Stylish Kitchen Mixer with K-beater,
Dough Hook and Whisk, 5 Litre Glass Bowl"

        price={239.0}

        rating={4}

        image="https://images-na.ssl-images-
amazon.com/images/I/81O%2BGNdkzKL._AC_SX450_.jpg"

    />

</div>

<div className="home__row">

    <Product

        id="4903850"

        title="Samsung LC49RG90SSUXEN 49' Curved LED Gaming Monitor"

        price={199.99}

        rating={3}

        image="https://images-na.ssl-images-
amazon.com/images/I/71Swqqe7XAL._AC_SX466_.jpg"

    />

    <Product

        id="23445930"

        title="Amazon Echo (3rd generation) | Smart speaker with Alexa, Charcoal Fabric"

        price={98.99}

        rating={5}

```

image="https://media.very.co.uk/i/very/P6LTG\_SQ1\_0000000071\_CHARCOAL\_SLf?\$300x400\_retinamobilex2\$"

/>

<Product

id="3254354345"

title="New Apple iPad Pro (12.9-inch, Wi-Fi, 128GB) - Silver (4th Generation)"

price={ 598.99 }

rating={ 4 }

image="https://images-na.ssl-images-amazon.com/images/I/816ctt5WV5L.\_AC\_SX385\_.jpg"

/>

</div>

<div className="home\_\_row">

<Product

id="90829332"

title="Samsung LC49RG90SSUXEN 49' Curved LED Gaming Monitor - Super Ultra Wide Dual WQHD 5120 x 1440"

price={ 1094.98 }

rating={ 4 }

image="https://images-na.ssl-images-amazon.com/images/I/6125mFrzr6L.\_AC\_SX355\_.jpg"

/>

</div>

</div>

</div>

);

}

```
export default Home;
```

### **Home.css**

```
home {  
  display: flex;  
  justify-content: center;  
  margin-left: auto;  
  margin-right: auto;  
  max-width: 1500px;  
}
```

```
.home_row {  
  display: flex;  
  z-index: 1;  
  margin-left: 5px;  
  margin-right: 5px;  
}
```

```
.home_image {  
  width: 100%;  
  z-index: -1;  
  margin-bottom: -150px;  
  mask-image: linear-gradient(to bottom, rgba(0, 0, 0, 1), rgba(0, 0, 0, 0));  
}
```

## **Index.js**

```
import React from "react";  
import ReactDOM from "react-dom";  
import "./index.css";  
import App from "./App";  
import * as serviceWorker from "./serviceWorker";  
import reducer, { initialState } from "./reducer";  
import { StateProvider } from "./StateProvider";
```

```
ReactDOM.render(  
  <React.StrictMode>  
    <StateProvider initialState={initialState} reducer={reducer}>  
      <App />  
    </StateProvider>  
  </React.StrictMode>,  
  document.getElementById("root")  
);
```

```
// If you want your app to work offline and load faster, you can change  
// unregister() to register() below. Note this comes with some pitfalls.  
// Learn more about service workers: https://bit.ly/CRA-PWA  
serviceWorker.unregister();
```

## **Index.css**

```
* {  
    margin: 0;  
}  
  
body {  
    background-color: rgb(234, 237, 237);  
    margin: 0;  
    font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", "Roboto", "Oxygen",  
        "Ubuntu", "Cantarell", "Fira Sans", "Droid Sans", "Helvetica Neue",  
        sans-serif;  
    -webkit-font-smoothing: antialiased;  
    -moz-osx-font-smoothing: grayscale;  
}  
  
code {  
    font-family: source-code-pro, Menlo, Monaco, Consolas, "Courier New",  
        monospace;  
}
```



## **Login.js**

```
import React, { useState } from 'react';
import './Login.css'
import { Link, useHistory } from "react-router-dom";
import { auth } from "../firebase";

function Login() {
  const history = useHistory();
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");

  const signIn = e => {
    e.preventDefault();

    auth
      .signInWithEmailAndPassword(email, password)
      .then(auth => {
        history.push('/')
      })
      .catch(error => alert(error.message))
  }

  const register = e => {
    e.preventDefault();
```

```

    auth
      .createUserWithEmailAndPassword(email, password)
      .then((auth) => {
        // it successfully created a new user with email and password
        if (auth) {
          history.push('/')
        }
      })
      .catch(error => alert(error.message))
  }

  return (
    <div className='login'>
      <Link to='/'>
        <img
          className="login_logo"

src='https://upload.wikimedia.org/wikipedia/commons/thumb/a/a9/Amazon_logo.svg/1024px-Amazon_logo.svg.png'

        />
      </Link>

      <div className='login_container'>
        <h1>Sign-in</h1>

        <form>
          <h5>E-mail</h5>
          <input type='text' value={email} onChange={e => setEmail(e.target.value)} />

```

```

        <h5>Password</h5>

        <input      type='password'      value={password}      onChange={e      =>
setPassword(e.target.value)} />

        <button type='submit' onClick={signIn} className='login__signInButton'>Sign
In</button>

    </form>

    <p>

        By signing-in you agree to the AMAZON FAKE CLONE Conditions of Use & Sale.

Please

        see our Privacy Notice, our Cookies Notice and our Interest-Based Ads Notice.

    </p>

    <button onClick={register} className='login_registerButton'>Create your Amazon
Account</button>

    </div>

</div>

)

}

export default Login

```

## **Login.css**

```
.login {  
    background-color: rgba(90, 76, 89, 0.007);  
    height: 100vh;  
    display: flex;  
    flex-direction: column;  
    align-items: center;  
}
```

```
.login_logo {  
    margin-top: 20px;  
    margin-bottom: 20px;  
    object-fit: contain;  
    width: 100px;  
    margin-right: auto;  
    margin-left: auto;  
}
```

```
.login_container {  
    width: 300px;  
    height: fit-content;  
    display: flex;  
    flex-direction: column;  
    border-radius: 5px;  
    border: 1px solid rgb(151, 117, 5);
```

```
padding: 20px;  
}
```

```
.login_container > h1 {  
  font-weight: 500;  
  margin-bottom: 20px;  
}
```

```
.login_container > form > h5 {  
  margin-bottom: 5px;  
}
```

```
.login_container > form > input {  
  height: 30px;  
  margin-bottom: 10px;  
  background-color: rgb(235, 224, 224);  
  width: 98%;  
}
```

```
.login_container > p {  
  margin-top: 15px;  
  font-size: 12px;  
}
```

```
.login_signInButton {  
  background: #6fb3eb;  
  border-radius: 2px;  
  width: 100%;
```

```
height: 30px;  
border: 1px solid;  
margin-top: 10px;  
border-color: #1b0ce9 }
```

```
.login_registerButton {  
border-radius: 2px;  
width: 100%;  
height: 30px;  
border: 1px solid;  
margin-top: 10px;  
border-color: rgb(133, 197, 14);  
}
```

## **Header.css**

```
.header {  
    height: 60px;  
    display: flex;  
    align-items: center;  
    background-color: #bd11a6;  
    position: sticky;  
    top: 0;  
    z-index: 100;  
}
```

```
.header_logo {  
    width: 160px;  
    object-fit: contain;  
    margin: 0 20px;  
    margin-top: 5px;  
}
```

```
.header_Location{  
    width: 20px;  
    object-fit: contain;  
    margin: 0 20px;  
    margin-top: 5px;  
}
```

```
.header__search {
```



```
display: flex;
flex: 1;
align-items: center;
border-radius: 24px;
}
```

```
.header_searchInput {
height: 12px;
padding: 10px;
border: none;
width: 100%;
}
```

```
.header_searchIcon {
padding: 5px;
height: 22px !important;
background-color: #ddf169;
}
```

```
.header_optionLineOne {
font-size: 10px;
}
```

```
.header_optionLineTwo {
font-size: 13px;
font-weight: 800;
}
```

```
.header_optionBasket {  
  display: flex;  
  align-items: center;  
  color: white;  
}
```

```
.header_basketCount {  
  margin-left: 10px;  
  margin-right: 10px;  
}
```

```
.header_nav {  
  display: flex;  
  justify-content: space-evenly;  
}
```

```
.header_option {  
  display: flex;  
  flex-direction: column;  
  margin-left: 10px;  
  margin-right: 10px;  
  color: white;  
}
```

## **Header.js**

```
import React from "react";
import "./Header.css";
import SearchIcon from "@material-ui/icons/Search";
import ShoppingBasketIcon from "@material-ui/icons/ShoppingBasket";
import { Link } from "react-router-dom";
import { useStateValue } from "../StateProvider";
import { auth } from "../firebase";

function Header() {
  const [{ basket, user }, dispatch] = useStateValue();

  const handleAuthentication = () => {
    if (user) {
      auth.signOut();
    }
  }

  return (
    <div className="header">
      <Link to="/">
        

```

```
  
</Link>
```

```
<div className="header__search">  
  <input className="header__searchInput" type="text" />  
  <SearchIcon className="header__searchIcon" />  
</div>
```

```
<div className="header__nav">  
  <Link to={!user && '/login'}>  
    <div onClick={handleAuthenticaton} className="header__option">  
      <span className="header__optionLineOne">Hello {!user ? 'Guest' : user.email}</span>  
      <span className="header__optionLineTwo">{user ? 'Sign Out' : 'Sign In'}</span>  
    </div>  
  </Link>
```

```
<Link to="/orders">  
  <div className="header__option">  
    <span className="header__optionLineOne">Returns</span>  
    <span className="header__optionLineTwo">& Orders</span>  
  </div>  
</Link>
```

```
<div className="header__option">
```

```

    <span className="header__optionLineOne"></span>
    <span className="header__optionLineTwo">offer%</span>
  </div>

  <Link to="/checkout">
    <div className="header__optionBasket">
      <ShoppingBasketIcon />
      <span className="header__optionLineTwo header__basketCount">
        {basket?.length}
      </span>
    </div>
  </Link>
</div>
</div>
);
}

export default Header

```

## **Firebase.js**

```
import firebase from "firebase";

const firebaseConfig = {
  apiKey: "AIzaSyCcPSKlYtpdzBoAC8soeSmIARMzVKzrf5I",
  authDomain: "challenge-4b2b2.firebaseio.com",
  databaseURL: "https://challenge-4b2b2.firebaseio.com",
  projectId: "challenge-4b2b2",
  storageBucket: "challenge-4b2b2.appspot.com",
  messagingSenderId: "962418448875",
  appId: "1:962418448875:web:f6cce5eeaf819481f661ae",
};

const firebaseApp = firebase.initializeApp(firebaseConfig);

const db = firebaseApp.firestore();
const auth = firebase.auth();

export { db, auth };
```

### **checkoutProduct.css**

```
.checkoutProduct {  
    display: flex;  
    margin-top: 20px;  
    margin-bottom: 20px;  
}
```

```
.checkoutProduct_info {  
    padding-left: 20px  
}
```

```
.checkoutProduct_info > button {  
    background: #f0c14b;  
    border: 1px solid;  
    margin-top: 10px;  
    border-color: #a88734 #9c7e31 #846a29;  
    color: #111;  
}
```

```
.checkoutProduct_image {  
    object-fit: contain;  
    width: 180px;  
    height: 180px;  
}
```

```
.checkoutProduct_rating {  
  display: flex;  
}
```

```
.checkoutProduct_title {  
  font-size: 17px;  
  font-weight: 800;  
}
```



## **Checkoutproduct.js**

```
import React from "react";
import "./Checkout.css";
import Subtotal from "./Subtotal";
import { useStateValue } from "./StateProvider";
import CheckoutProduct from "./CheckoutProduct";

function Checkout() {
  const [{ basket, user }, dispatch] = useStateValue();

  return (
    <div className="checkout">
      <div className="checkout_left">
        

        <div>
          <h3>Hello, {user?.email}</h3>
          <h2 className="checkout_title">Your shopping Basket</h2>

          {basket.map(item => (
```

```
      <CheckoutProduct
        id={item.id}
        title={item.title}
        image={item.image}
        price={item.price}
        rating={item.rating}
      />
    )))}

  </div>
</div>

<div className="checkout_right">
  <Subtotal />
</div>
</div>
);
}

export default Checkout;
```

### **checkout.js**

```
import React from "react";
import "./Checkout.css";
import Subtotal from "./Subtotal";
import { useStateValue } from "./StateProvider";
import CheckoutProduct from "./CheckoutProduct";

function Checkout() {
  const [{ basket, user }, dispatch] = useStateValue();

  return (
    <div className="checkout">
      <div className="checkout_left">
        

        <div>
          <h3>Hello, {user?.email}</h3>
          <h2 className="checkout_title">Your shopping Basket</h2>

          {basket.map(item => (
```

```

        <CheckoutProduct
          id={item.id}
          title={item.title}
          image={item.image}
          price={item.price}
          rating={item.rating}
        />
      ))}

    </div>
  </div>

  <div className="checkout_right">
    <Subtotal />
  </div>
</div>
);
}

export default Checkout;

```

### **Checkout.css**

```
.checkout {  
  display: flex;  
  padding: 20px;  
  background-color: white;  
  height: max-content;  
}
```

```
.checkout_ad {  
  width: 100%;  
  margin-bottom: 10px;  
}
```

```
.checkout_title {  
  margin-right: 10px;  
  padding: 10px;  
  border-bottom: 1px solid lightgray;  
}
```

## **App.js**

```
import React, { useEffect } from "react";
import "./App.css";
import Header from "./Header";
import Home from "./Home";
import { BrowserRouter as Router, Switch, Route } from "react-router-dom";
import Checkout from "./Checkout";
import Login from "./Login";
import Payment from "./Payment";
import Orders from "./Orders";
import { auth } from "./firebase";
import { useStateValue } from "./StateProvider";
import { loadStripe } from "@stripe/stripe-js";
import { Elements } from "@stripe/react-stripe-js";

const promise = loadStripe(

"pk_test_51HPvU9DFg5koCdLGJJbNo60QAU99BejacsvnKvT8xnCu1wFLCuQP3WBArscK3
RvSQmSIB3N0Pbsc7TtbQiJ1vaOi00X9sIbazL"

);

function App() {
  const [{}, dispatch] = useStateValue();

  useEffect(() => {
    // will only run once when the app component loads...
```

```

auth.onAuthStateChanged((authUser) => {
  console.log("THE USER IS >>> ", authUser);

  if (authUser) {
    // the user just logged in / the user was logged in

    dispatch({
      type: "SET_USER",
      user: authUser,
    });
  } else {
    // the user is logged out
    dispatch({
      type: "SET_USER",
      user: null,
    });
  }
});
}, []);

return (
  <Router>
    <div className="app">
      <Switch>
        <Route path="/orders">
          <Header />
          <Orders />

```

```
    </Route>
    <Route path="/login">
      <Login />
    </Route>
    <Route path="/checkout">
      <Header />
      <Checkout />
    </Route>
    <Route path="/payment">
      <Header />
      <Elements stripe={promise}>
        <Payment />
      </Elements>
    </Route>
    <Route path="/">
      <Header />
      <Home />
    </Route>
  </Switch>
</div>
</Router>
);
}

export default App;
```



## ServiceWorker.js

```
// This optional code is used to register a service worker.  
// register() is not called by default.
```

```
// This lets the app load faster on subsequent visits in production, and gives  
// it offline capabilities. However, it also means that developers (and users)  
// will only see deployed updates on subsequent visits to a page, after all the  
// existing tabs open on the page have been closed, since previously cached  
// resources are updated in the background.
```

```
// To learn more about the benefits of this model and instructions on how to  
// opt-in, read https://bit.ly/CRA-PWA
```

```
const isLocalhost = Boolean(  
  window.location.hostname === 'localhost' ||  
    // [::1] is the IPv6 localhost address.  
    window.location.hostname === '[::1]' ||  
    // 127.0.0.0/8 are considered localhost for IPv4.  
    window.location.hostname.match(  
      /^127(?:\.(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)){3}$/  
    )  
);
```

```
export function register(config) {  
  if (process.env.NODE_ENV === 'production' && 'serviceWorker' in navigator) {  
    // The URL constructor is available in all browsers that support SW.  
    const publicUrl = new URL(process.env.PUBLIC_URL, window.location.href);  
    if (publicUrl.origin !== window.location.origin) {  
      // Our service worker won't work if PUBLIC_URL is on a different origin  
      // from what our page is served on. This might happen if a CDN is used to  
      // serve assets; see https://github.com/facebook/create-react-app/issues/2374  
      return;  
    }  
  }
```

```
  window.addEventListener('load', () => {  
    const swUrl = `${process.env.PUBLIC_URL}/service-worker.js`;  
  
    if (isLocalhost) {  
      // This is running on localhost. Let's check if a service worker still exists or not.  
      checkValidServiceWorker(swUrl, config);  
  
      // Add some additional logging to localhost, pointing developers to the  
      // service worker/PWA documentation.  
      navigator.serviceWorker.ready.then(() => {  
        console.log(  
          'This web app is being served cache-first by a service ' +
```

```

        'worker. To learn more, visit https://bit.ly/CRA-PWA'
    );
});
} else {
    // Is not localhost. Just register service worker
    registerValidSW(swUrl, config);
}
});
}
}

function registerValidSW(swUrl, config) {
    navigator.serviceWorker
        .register(swUrl)
        .then(registration => {
            registration.onupdatefound = () => {
                const installingWorker = registration.installing;
                if (installingWorker == null) {
                    return;
                }
                installingWorker.onstatechange = () => {
                    if (installingWorker.state === 'installed') {
                        if (navigator.serviceWorker.controller) {
                            // At this point, the updated precached content has been fetched,
                            // but the previous service worker will still serve the older
                            // content until all client tabs are closed.
                            console.log(
                                'New content is available and will be used when all ' +
                                'tabs for this page are closed. See https://bit.ly/CRA-PWA.'
                            );

                            // Execute callback
                            if (config && config.onUpdate) {
                                config.onUpdate(registration);
                            }
                        } else {
                            // At this point, everything has been precached.
                            // It's the perfect time to display a
                            // "Content is cached for offline use." message.
                            console.log('Content is cached for offline use.');
```

```

    };
  };
})
.catch(error => {
  console.error('Error during service worker registration:', error);
});
}

function checkValidServiceWorker(swUrl, config) {
  // Check if the service worker can be found. If it can't reload the page.
  fetch(swUrl, {
    headers: { 'Service-Worker': 'script' },
  })
  .then(response => {
    // Ensure service worker exists, and that we really are getting a JS file.
    const contentType = response.headers.get('content-type');
    if (
      response.status === 404 ||
      (contentType !== null && contentType.indexOf('javascript') === -1)
    ) {
      // No service worker found. Probably a different app. Reload the page.
      navigator.serviceWorker.ready.then(registration => {
        registration.unregister().then(() => {
          window.location.reload();
        });
      });
    } else {
      // Service worker found. Proceed as normal.
      registerValidSW(swUrl, config);
    }
  })
  .catch(() => {
    console.log(
      'No internet connection found. App is running in offline mode.'
    );
  });
}

export function unregister() {
  if ('serviceWorker' in navigator) {
    navigator.serviceWorker.ready
      .then(registration => {
        registration.unregister();
      })
      .catch(error => {
        console.error(error.message);
      });
  }
}

```

```
}  
}
```

### **Payment.js**

```
import React, { useState, useEffect } from 'react';  
import './Payment.css';  
import { useStateValue } from './stateProvider';  
import CheckoutProduct from './CheckoutProduct';  
import { Link, useHistory } from 'react-router-dom';  
import { CardElement, useStripe, useElements } from '@stripe/react-stripe-js';  
import CurrencyFormat from 'react-currency-format';  
import { getBasketTotal } from './reducer';  
import axios from './axios';  
import { db } from './firebase';
```

```
function Payment() {  
  const [{ basket, user }, dispatch] = useStateValue();  
  const history = useHistory();  
  
  const stripe = useStripe();  
  const elements = useElements();  
  
  const [succeeded, setSucceeded] = useState(false);  
  const [processing, setProcessing] = useState("");  
  const [error, setError] = useState(null);  
  const [disabled, setDisabled] = useState(true);  
  const [clientSecret, setClientSecret] = useState(true);  
  
  useEffect(() => {  
    // generate the special stripe secret which allows us to charge a customer  
    const getClientSecret = async () => {  
      const response = await axios({  
        method: 'post',  
        // Stripe expects the total in a currencies subunits  
        url: `/payments/create?total=${getBasketTotal(basket) * 100}`  
      });  
      setClientSecret(response.data.clientSecret)  
    }  
  
    getClientSecret();  
  });  
}
```

```

}, [basket])

console.log('THE SECRET IS >>>', clientSecret)
console.log('😊', user)

const handleSubmit = async (event) => {
  // do all the fancy stripe stuff...
  event.preventDefault();
  setProcessing(true);

  const payload = await stripe.confirmCardPayment(clientSecret, {
    payment_method: {
      card: elements.getElement(CardElement)
    }
  }).then(({ paymentIntent }) => {
    // paymentIntent = payment confirmation

    db
      .collection('users')
      .doc(user?.uid)
      .collection('orders')
      .doc(paymentIntent.id)
      .set({
        basket: basket,
        amount: paymentIntent.amount,
        created: paymentIntent.created
      })

    setSucceeded(true);
    setError(null)
    setProcessing(false)

    dispatch({
      type: 'EMPTY_BASKET'
    })

    history.replace('/orders')
  })
}

const handleChange = event => {
  // Listen for changes in the CardElement
  // and display any errors as the customer types their card details
  setDisabled(event.empty);
  setError(event.error ? event.error.message : "");
}

```

```

return (
  <div className='payment'>
    <div className='payment__container'>
      <h1>
        Checkout (
          <Link to="/checkout">{basket?.length} items</Link>
        )
      </h1>

      { /* Payment section - delivery address */ }
      <div className='payment__section'>
        <div className='payment__title'>
          <h3>Delivery Address</h3>
        </div>
        <div className='payment__address'>
          <p>{user?.email}</p>
          <p>123 React Lane</p>
          <p>Los Angeles, CA</p>
        </div>
      </div>

      { /* Payment section - Review Items */ }
      <div className='payment__section'>
        <div className='payment__title'>
          <h3>Review items and delivery</h3>
        </div>
        <div className='payment__items'>
          {basket.map(item => (
            <CheckoutProduct
              id={item.id}
              title={item.title}
              image={item.image}
              price={item.price}
              rating={item.rating}
            />
          ))}
        </div>
      </div>

      { /* Payment section - Payment method */ }
      <div className='payment__section'>
        <div className='payment__title'>
          <h3>Payment Method</h3>
        </div>

```

```

    <div className="payment__details">
      { /* Stripe magic will go */ }

      <form onSubmit={handleSubmit}>
        <CardElement onChange={handleChange}/>

        <div className='payment__priceContainer'>
          <CurrencyFormat
            renderText={ (value) => (
              <h3>Order Total: {value}</h3>
            ) }
            decimalScale={2}
            value={getBasketTotal(basket)}
            displayType="text"
            thousandSeparator={true}
            prefix={"$"}
          />
          <button disabled={processing || disabled || succeeded}>
            <span>{processing ? <p>Processing</p> : "Buy Now"}</span>
          </button>
        </div>

        { /* Errors */ }
        {error && <div>{error}</div>}
      </form>
    </div>
  </div>
</div>
</div>
)
}

```

export default Payment

### **Reducer.js**

```

export const initialState = {
  basket: [],
  user: null
};

// Selector
export const getBasketTotal = (basket) =>
  basket?.reduce((amount, item) => item.price + amount, 0);

const reducer = (state, action) => {

```

```

console.log(action);
switch (action.type) {
  case "ADD_TO_BASKET":
    return {
      ...state,
      basket: [...state.basket, action.item],
    };

  case 'EMPTY_BASKET':
    return {
      ...state,
      basket: []
    }

  case "REMOVE_FROM_BASKET":
    const index = state.basket.findIndex(
      (basketItem) => basketItem.id === action.id
    );
    let newBasket = [...state.basket];

    if (index >= 0) {
      newBasket.splice(index, 1);
    } else {
      console.warn(
        `Cant remove product (id: ${action.id}) as its not in basket!`
      )
    }

    return {
      ...state,
      basket: newBasket
    }

  case "SET_USER":
    return {
      ...state,
      user: action.user
    }

  default:
    return state;
}
};

export default reducer;

```



### Order.js

```
import React, { useState, useEffect } from 'react';
import { db } from "../firebase";
import './Orders.css'
import { useStateValue } from "../StateProvider";
import Order from './Order'

function Orders() {
const [{ basket, user }, dispatch] = useStateValue();
const [orders, setOrders] = useState([]);

    useEffect(() => {
        if(user) {
            db
                .collection('users')
                .doc(user?.uid)
                .collection('orders')
                .orderBy('created', 'desc')
                .onSnapshot(snapshot => (
setOrders(snapshot.docs.map(doc => ({
                    id: doc.id,
                    data: doc.data()
                }))))
                ))
            } else {
                setOrders([])
            }

        }, [user])

        return (
            <div className='orders'>
                <h1>Your Orders</h1>

                <div className='orders__order'>
                    {orders?.map(order => (
                        <Order order={order} />
                    ))}
                </div>
            </div>
        )
    }
}
```

```
export default Orders
```

### **Subtotal.js**

```
import React from "react";
import "../Subtotal.css";
import CurrencyFormat from "react-currency-format";
import { useStateValue } from "../stateProvider";
import { getBasketTotal } from "../reducer";
import { useHistory } from "react-router-dom";

function Subtotal() {
  const history = useHistory();
  const [{ basket }, dispatch] = useStateValue();

  return (
    <div className="subtotal">
      <CurrencyFormat
        renderText={(value) => (
          <>
            <p>
              { /* Part of the homework */ }
              Subtotal ({basket.length} items): <strong>{ value }</strong>
            </p>
            <small className="subtotal__gift">
              <input type="checkbox" /> This order contains a gift
            </small>
          </>
        )}
        decimalScale={2}
        value={getBasketTotal(basket)} // Part of the homework
        displayType="text"
        thousandSeparator={true}
        prefix={"$"}
      />

      <button onClick={e => history.push('/payment')}>Proceed to Checkout</button>
    </div>
  );
}

export default Subtotal;
```

## **StateProvider.js**

```
import React, { createContext, useContext, useReducer } from "react";

// Prepares the dataLayer
export const StateContext = createContext();

// Wrap our app and provide the Data layer
export const StateProvider = ({ reducer, initialState, children }) => (
  <StateContext.Provider value={useReducer(reducer, initialState)}>
    {children}
  </StateContext.Provider>
);

// Pull information from the data layer
export const useStateValue = () => useContext(StateContext);
```

## **Orders.js**

```
import React from 'react'
import './Order.css'
import moment from "moment";
import CheckoutProduct from "./CheckoutProduct";
import CurrencyFormat from "react-currency-format";

function Order({ order }) {
  return (
    <div className='order'>
      <h2>Order</h2>
      <p>{moment.unix(order.data.created).format("MMMM Do YYYY, h:mm")}</p>
      <p className="order__id">
        <small>{order.id}</small>
      </p>
      {order.data.basket?.map(item => (
        <CheckoutProduct
          id={item.id}

```

```

        title={item.title}
        image={item.image}
        price={item.price}
        rating={item.rating}
        hideButton
      />
    )))
    <CurrencyFormat
      renderText={(value) => (
        <h3 className="order__total">Order Total: {value}</h3>
      )}
      decimalScale={2}
      value={order.data.amount / 100}
      displayType={"text"}
      thousandSeparator={true}
      prefix={"$"}
    />
  </div>
)
}

```

export default Order

# **Chapter 5**

## **Implementation & Testing**

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. In fact, testing is the one step in the software engineering process that could be viewed as destructive rather than constructive.

### **UNIT TESTING**

Unit testing focuses verification effort on the smallest unit of software design, the module. The unit testing we have is white box oriented and some modules the steps are conducted in parallel.

### **WHITE BOX TESTING**

- \*. This type of testing ensures that
- \* All independent paths have been exercised at least once
- \* All logical decisions have been exercised on their true and false sides

### **DATA FLOW TESTING**

This type of testing selects the path of the program according to the location of definition and use of variables. This kind of testing was used only when some local variable were declared. The definition-use chain method was used in this type of testing. These were particularly useful in nested statements.

### **LOOP TESTING**

- \* In this type of testing all the loops are tested to all the limits possible. The following exercise was adopted for all loops:
- \*. All the loops were tested at their limits, just above them and just below them.
- \*. All the loops were skipped at least once.
- \*. For nested loops test the inner most loop first and then work outwards

## INTEGRATION TESTING

Established technique of flow graph with Cyclomatic complexity was used to derive test cases for all the functions. The main steps in deriving test cases were:

Use the design of the code and draw correspondent flow graph.

Determine the Cyclomatic complexity of resultant flow graph, using formula:

$$V(G)=E-N+2 \text{ or}$$

$$V(G)=P+1 \text{ or}$$

$$V(G)=\text{Number Of Regions}$$

Where  $V(G)$  is Cyclomatic complexity

$E$  is the number of edges,

$N$  is the number of flow graph nodes,

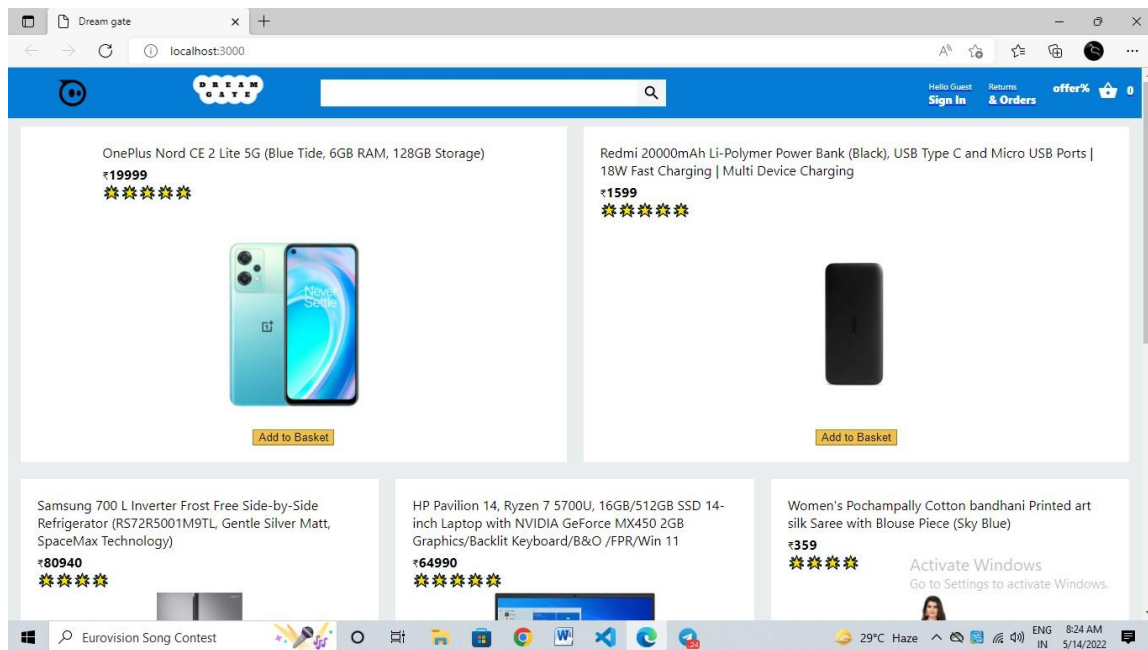
$P$  is the number of predicate nodes.

Determine the basis of set of linearly independent paths.

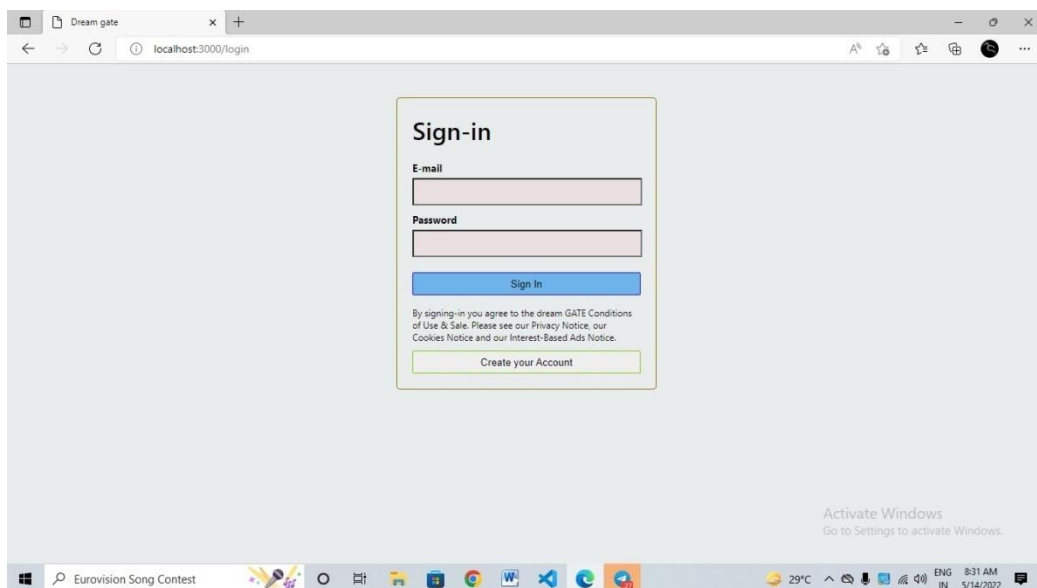
# Chapter 6

## Result & Discussion

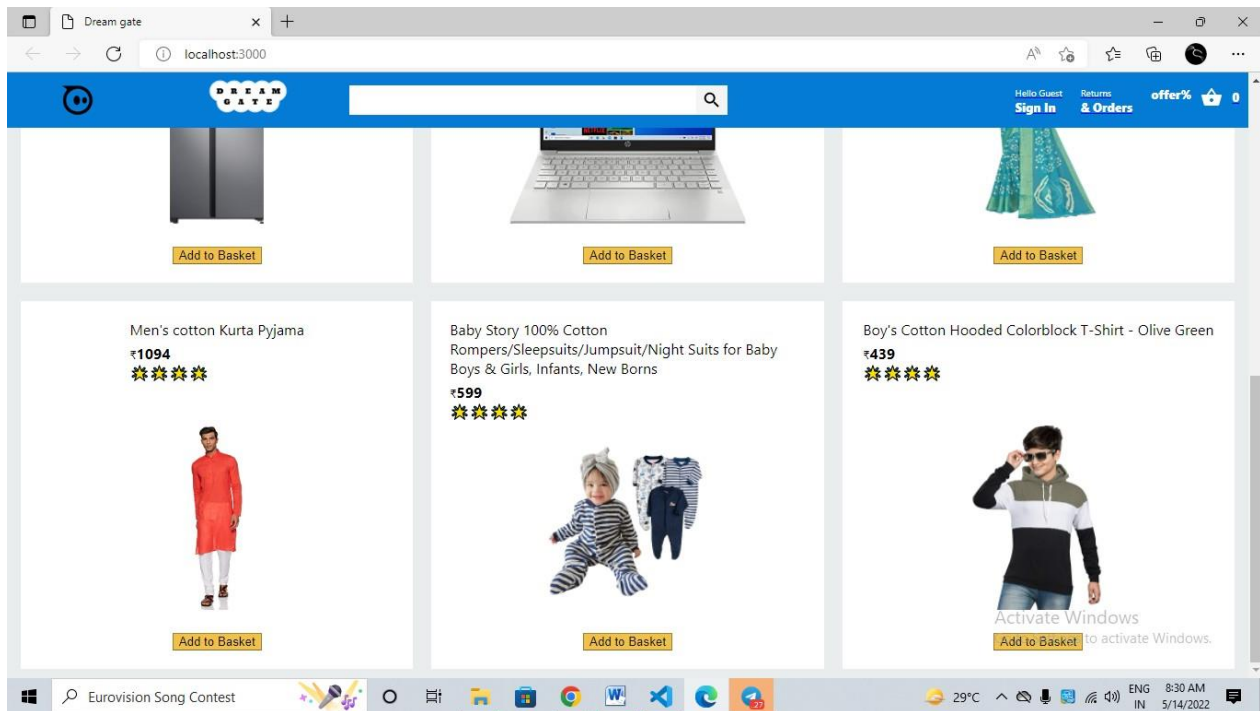
**Login Page:** This allow the user to login to the Travel Management System.



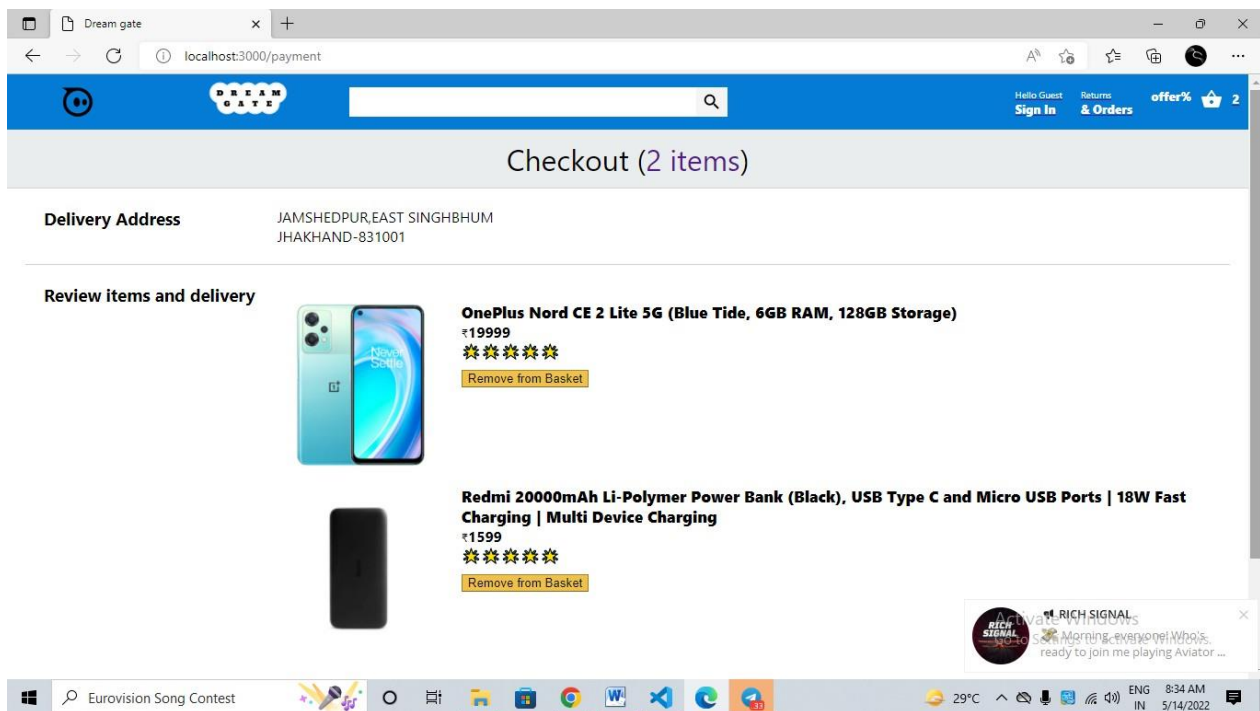
**Signup Page:** This allow the user to make a new registration to the Travel Management System.



**Dashboard:** This allow the user to enter to the main page of the E-commerce website

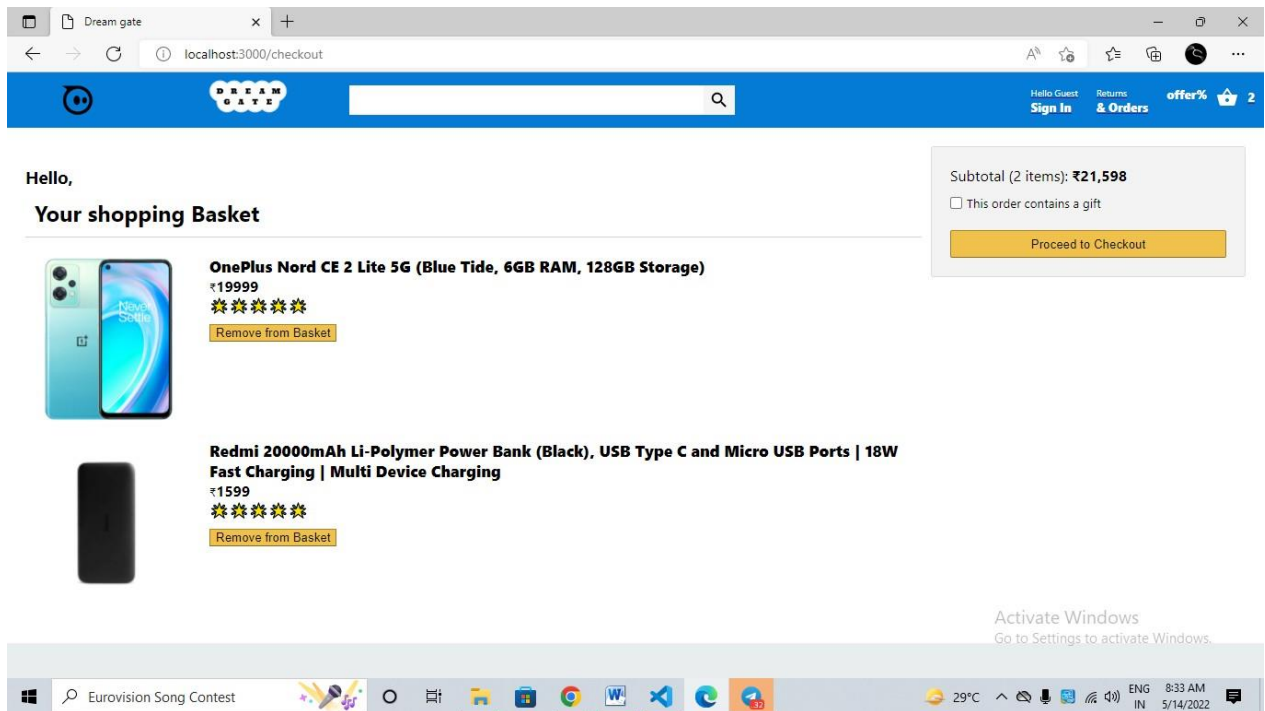


**Checkout details** The details of the product entered by the user.

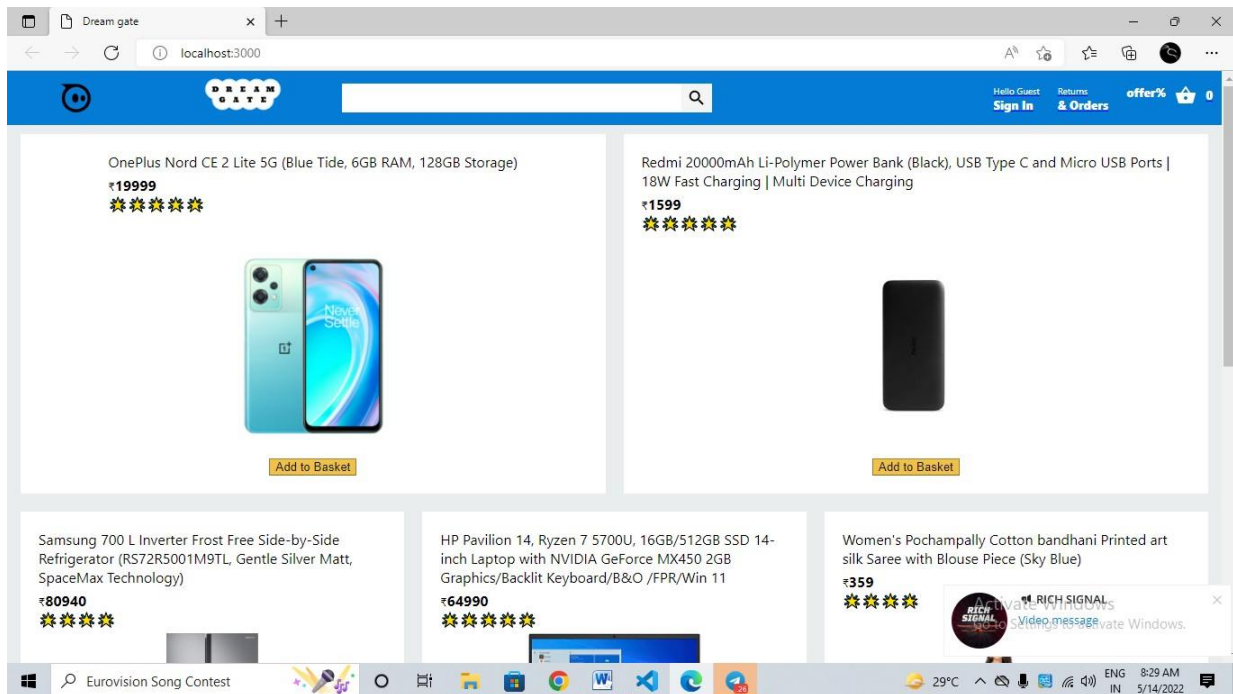




**Total Details:** The total of the products to be updated.



**product Details:** The details of the customer to be viewed.



# **Chapter 7**

## **CONCLUSION**

The project entitled dream gate (e commerce website) was completed successfully. The system has been developed with much care and free of errors and at the same time it is efficient and less time consuming. The purpose of this project was to develop a web application for purchasing items from a fashion shop. This project enabled me gain valuable information and practical knowledge on several topics like designing web pages using html & CSS, usage of responsive templates, designing of full stack Django application, and management of database using SQLite. The entire system is secured. Also, the project helped me understanding about the development phases of a project and software development life cycle. I learned how to test different features of a project. This project has given me great satisfaction in having designed an application which can be implemented to any nearby shops or branded shops selling various kinds of products by simple modifications. 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.

# CHAPTER 8

## REFERENCE

1. [www.msdn.microsoft.com](http://www.msdn.microsoft.com)
2. [www.phpcity.com](http://www.phpcity.com)
3. [ww.phpgurukul.com](http://ww.phpgurukul.com)
4. [www.myproject.com](http://www.myproject.com)
5. [www.way3html.com](http://www.way3html.com)
6. [w3schools.com](http://w3schools.com)