VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELAGAVI-590018

A DBMS Mini Project Report

on

" Quickcart "

Submitted in partial fulfillment of the requirements for the V semester and award of the degree of Bachelor of Engineering in Computer Science and Engineering of Visvesvaraya Technological University, Belagavi

Submitted by:

Nishanth C	*1RN20CS092*
Pramod Ganapati Naik	*1RN20CS101*

Under the Guidance of:

Mrs. Mamatha S Jajur
Assistant Professor
Dept. of CSE

Department of Computer Science and Engineering
Accredited by NBA upto June 2025
RNS Institute of Technology
Channasandra, Dr.Vishnuvardhan Road, Bengaluru-560 098

2022-2023

RNS Institute of Technology

Channasandra, Dr.Vishnuvardhan Road, Bengaluru-560098

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

(NBA Accredited for academic years 2022-25)

CERTIFICATE

Certified that the mini project work entitled **" Quickcart "** has been successfully carried out by **"Nishanth C"** bearing USN **"1RN20CS092"** and **"Pramod Ganapati Naik"** bearing USN **"1RN20CS101"**, bonafide students of **"RNS Institute of Technology"** in partial fulfillment of the requirements for the 5th semester of **"Bachelor of Engineering in Computer Science and Engineering of Visvesvaraya Technological University"**, Belagavi, during academic year 2022-2023. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the DBMS laboratory requirements of 5th semester BE, CSE.

Signature of the Guide	Signature of the HoD	Signature of the Principal
Mrs. Mamatha S Jajur	**Dr. Kiran P**	**Dr. M K Venkatesha**
Assistant Professor	Professor, HoD	Principal
Dept. of CSE	Dept. of CSE	

External Viva:

Name of the Examiners	Signature with Date
1.	
2.	

Acknowledgement

Any achievement, be it scholastic or otherwise does not depend solely on the individual efforts but on the guidance, encouragement and cooperation of intellectuals, elders and friends. A number of personalities, in their own capacities have helped us in carrying out this project work. We would like to take this opportunity to thank them all.

We are grateful to Management and **Dr. M K Venkatesha**, Principal, RNSIT, Bangalore, for his support towards completing this mini project.

We would like to thank **Dr. Kiran P**, Professor & Head, Department of Computer Science & Engineering, RNSIT, Bangalore, for his valuable suggestions and expert advice.

We deeply express our sincere gratitude to our guide **Mrs. Mamatha S Jajur** , Assistant Professor, Department of CSE, RNSIT, Bangalore, for her able guidance, regular source of encouragement and assistance throughout this project.

We would like to thank all the teaching and non-teaching staff of Department of Computer Science & Engineering, RNSIT, Bengaluru-98 for their constant support and encouragement.

Abstract

An ecommerce website is an online platform that allows businesses to sell products or services directly to consumers. These websites are often designed to be easy to use and navigate, with features like shopping carts, product images, and online payment options. Ecommerce websites can be customized to meet the specific needs of a business, and can include features such as inventory management, customer relationship management, and analytics.

One of the key benefits of an ecommerce website is that it allows businesses to reach a global audience. Customers can purchase products or services from anywhere in the world, at any time of the day. Additionally, ecommerce websites can often be optimized for search engines, making them more visible to potential customers. This can help to drive more traffic to the site, and ultimately increase sales.

Another advantage of ecommerce websites is that they can save time and money for both the business and the customer. Customers can browse and purchase products or services without ever having to leave their homes, and businesses can automate many of the processes involved in selling products or services. This can help to reduce costs and increase efficiency.

Contents

List of Figures

Chapter 1

Introduction

1.1 Database Technologies

The essential feature of database technology is that it provides an internal representation (model) of the external world of interest. Examples are, the representation of a particular date/time/flight/aircraft in an airline reservation or of the item code/item description/quantity on hand/reorder level/reorder quantity in a stock control system.

The technology involved is concerned primarily with maintaining the internal representation consistent with external reality; this involves the results of extensive RD over the past 30 years in areas such as user requirements analysis, data modelling, process modelling, data integrity, concurrency, transactions, file organisation, indexing, rollback and recovery, persistent programming, object-orientation, logic programming, deductive database systems, active database systems... and in all these (and other) areas there remains much more to be done. The essential point is that database technology is a CORE TECHNOLOGY which has links to:

- Information management / processing
- Data analysis / statistics
- Data visualization / presentation
- Multimedia and hypermedia
- Office and document systems
- Business processes, workflow, CSCW (computer-supported cooperative work)

Relational DBMS is the modern base technology for many business applications. It offers flexibility and easy-to-use tools at the expense of ultimate performance. More recently relational systems

have started extending their facilities in directions like information retrieval, objectorientation and deductive/active systems which lead to the so-called 'Extended Relational Systems'.

Information Retrieval Systems began with handling library catalogues and then extended to full free-text by utilizing inverted index technology with a lexicon or thesaurus. Modern systems utilize some KBS (knowledge-based systems) techniques to improve the retrieval. Object-Oriented DBMS started for engineering applications in which objects are complex, have versions and need to be treated as a complete entity. OODBMSs share many of the OOPL features such as identity, inheritance, late binding, overloading and overriding. OODBMSs have found favours in engineering and office systems but haven't been successful yet in traditional application areas.

Deductive / Active DBMS has evolved over the last 20 years and combines logic programming technology with database technology. This allows the database itself to react to the external events and also to maintain its integrity dynamically with respect to the real world.

1.2 Characteristics of Database Approach

Traditional form included organising the data in file format. DBMS was a new concept then, and all kinds of research was done to make it overcome the deficiencies in traditional style of data management. A modern DBMS has the following characteristics

- Real-world entity A modern DBMS is more realistic and uses real-world entities to design its architecture. It uses behaviour and attribute too. For example, a school database may use students as an entity and their age as an attribute.

- Relation-based tables DBMS allows entities and relations to form tables. A user can understand the architecture of a database by just looking at the table names.

- Isolation of data and application A database system is entirely different than its data. A database is an active entity, whereas data is said to be passive, on which the database works and organizes. DBMS also stores metadata, which is data about data, to ease its own process.

- Less redundancy DBMS follows the rules of normalization, which splits a relation when any of its attributes has redundancy in its values. Normalization is a mathematically rich and scientific process that will reduces the data redundancy.

- Consistency Consistency is a state where every relation in a database remains consistent. There exists methods and techniques, that can detect an attempt of leaving database in an inconsistent

state. DBMS can provide greater consistency as compared to earlier forms of data storing applications like file-processing systems.

- Query Language DBMS is equipped with query language, which makes it more efficient to retrieve and manipulate data. A user can apply as many and the filtering options as required to retrieve a set of data. Traditionally it was not possible where file-processing system was used.

- ACID Properties DBMS follows the concepts of Atomicity, Consistency, Isolation, and Durability (normally shortened as ACID). These concepts are applied on transactions, which manipulate data in a database. ACID properties help the database to stay healthy in multi-transactional environments and also in case of failure.

- Multiuser and Concurrent Access DBMS supports multi-user environment and allows them to access and manipulate data in parallel. Though there are restrictions on transactions when users attempt to handle the same data item, but users are always unaware of them.

- Multiple views DBMS offers multiple views for different users. A user in the Sales department will have a different view of the database from the person working in the Production department. This feature enables the users to have a concentrate view of the database according to their requirements.

- Security Features like multiple views offer security to certain extent when users are unable to access the data of other users and departments. DBMS offers methods to impose constraints while entering data into the database and retrieving the same at a later stage. DBMS offers many different levels of security features, which enables multiple users to have different views with different features. For example, a user in the Sales department cannot see the data that belongs to the Purchase department. It can also be helpful in deciding how much data of the Sales department should be displayed to the user. Since a DBMS is not saved on the disk as traditional file systems, it is very hard for miscreants to break the code.

1.3 Applications of DBMS

Applications of Database Management Systems :

- Telecom: There is a database to keeps track of the information regarding the calls made, network usage, customer details etc. Without the database system it is hard to maintain such huge amounts of data which gets updated every millisecond.

- Industry: Whether it is a manufacturing unit, a warehouse or a distribution centre, each one needs a database to keep the records of the ins and outs. For example, a distribution centre should keep a track of the product units that were supplied to the centre as well as the products that got delivered from the distribution centre on each day; this is where DBMS comes into picture.

- Banking System: For storing information regarding a customer, keeping a track of his/her day to day credit and debit transactions, generating bank statements etc is done with through Database management systems.

- Education sector: Database systems are frequently used in schools and colleges to store and retrieve the data regarding the student , staff details, course details, exam details, payroll data, attendance details, fees details etc. There is lots of inter-related data that needs to be stored and retrieved in an efficient manner.

- Online shopping: You must be aware of the online shopping websites such as Amazon, Flip kart etc. These sites store the product information, your addresses and preferences, credit details and provide you the relevant list of products based on your query. All this involves a Database management system.

1.4 Problem Description/Statement

An ecommerce database project is a complex undertaking that involves designing, building and maintaining a database to support the operations of an online store. The primary goal of the project is to ensure that the ecommerce website can handle a large volume of data, such as customer information, product details, and sales transactions, in an efficient and reliable manner.

One of the main challenges in an ecommerce database project is dealing with the large volume of data that is constantly being added, updated and accessed.The database must also be designed to handle the high level of complexity that comes with an ecommerce website, such as product catalogs, inventory management, and order processing.

Security is also a critical aspect of ecommerce database projects. The database must be designed to protect sensitive customer information, such as credit card numbers and personal details, from unauthorized access. This requires the use of advanced security measures, such as encryption and authentication, to protect the database from potential cyber-attacks.

Chapter 2

Resource Requirements

2.1 Hardware Analysis

The Hardware requirements are very minimal and the program can be run on most of the machines.

Processor : i5 processor

Processor Speed : 1.2 GHz

RAM : 1 GB

Storage Space : 40 GB

Monitor Resolution : 1024*768 or 1336*768 or 1280*1024

2.2 Software Requirements

System specifications and software required are,

Operating System used: Windows 10

Technologies used: HTML, CSS, Django, Python, JavaScript

Server: MySQL, Django-Admin, Mysql.connector

IDE used: Visual Studio Code, PyCharm

Browser that supports HTML

2.3 End User Requirements

The technical requirements for the project are mentioned below:

1.Django 2.MySQL 3.HTML 4.CSS 5.JavaScript

2.3.1 Django

Django is a high-level Python web framework that enables the rapid development of secure and maintainable websites and web applications. It follows the Model-View-Controller (MVC) architectural pattern and is built on top of Python's standard library, making it easy to use and understand for developers familiar with Python.

One of the key features of Django is its built-in support for models, which represent the data structures of a web application. Models are defined using Python classes, and they can be easily created, modified, and deleted using Django's Object-Relational Mapping (ORM) system. This allows developers to focus on the logic of their application, rather than writing complex SQL queries to interact with the database.

Another important aspect of Django is its views, which handle the logic of a web application. Views are Python functions that handle incoming HTTP requests and return appropriate HTTP responses. They can be used to perform actions such as fetching data from the database, processing form submissions, and rendering templates. Django's built-in URL dispatcher makes it easy to map URLs to specific views, allowing developers to define the routing of their application.

Django also includes a powerful template engine that allows developers to separate the presentation logic of their application from the business logic. Templates are written using a simple language and can include variables, loops, and conditional statements, making it easy to create dynamic, reusable HTML templates.

Additionally, Django provides a built-in administration interface that allows developers to easily manage the data of their application. This feature is particularly useful for creating applications that need to be maintained by non-technical users. The interface can be customized and extended, and allows the management of users, groups, permissions and more.

Django also provides built-in support for security features such as Cross-Site Request Forgery (CSRF) protection and SQL injection prevention. This helps developers to create secure web applications without having to spend a lot of time and effort on implementing these features themselves.

Overall, Django is a powerful and flexible web framework that makes it easy to create and maintain websites and web applications. Its built-in support for models, views, templates, and security features, along with its focus on maintainability and reusability, make it a great choice for developers looking to build robust and scalable web applications

2.3.2 MySQL

MySQL is a popular open-source relational database management system (RDBMS) that is widely used for web applications and data warehousing. It is developed, distributed, and supported by Oracle Corporation. MySQL uses the SQL (Structured Query Language) to manage and manipulate the data stored in its databases.

MySQL is known for its speed, reliability, and ease of use. It is also highly customizable and can be easily integrated with other software and programming languages such as PHP, Python, and Java. MySQL supports various storage engines, which allow users to choose the storage method that best suits their needs. Some of the most commonly used storage engines are InnoDB, MyISAM, and Memory.

MySQL is also highly scalable and can handle large amounts of data and concurrent connections. This makes it well suited for use in high-traffic websites, online transaction processing (OLTP) systems, and data warehousing. MySQL also supports advanced features such as replication and partitioning, which help users to improve the performance and availability of their databases.

Additionally, MySQL provides a wide range of tools for managing and maintaining databases, including the MySQL Command Line Client, the MySQL Workbench, and the MySQL Administrator. These tools enable users to create and modify databases, manage users and permissions, and perform backups and restores.

Overall, MySQL is a powerful and widely used RDBMS that is well suited for a wide range of applications. Its open-source nature, scalability, and support for advanced features make it a popular choice among developers and system administrators.

MySQL Connector module of Python is used to connect MySQL databases with the Python programs, it does that using the Python Database API Specification v2.0 (PEP 249). It uses the Python standard library and has no dependencies.To create a connection between the MySQL database and the python application, the connect() method of mysql.connector module is used.

2.3.3 HTML and CSS

HTML (Hypertext Markup Language) and CSS (Cascading Style Sheets) are two essential technologies used for creating and designing websites. HTML is used to structure and organize the content of a website, while CSS is used to control the layout and appearance of that content.

HTML is a markup language that uses tags to define the structure of a web page. These tags include headings, paragraphs, lists, links, images, and more. By using these tags, developers can create a hierarchical structure for their content, making it easy for both humans and machines to

understand the organization of the page.

CSS is used to apply styles to the HTML elements on a web page. Styles include things like color, font, size, spacing, and positioning. With CSS, developers can create a consistent look and feel across all pages of a website, and can also easily make changes to the layout and design without having to alter the HTML code.

CSS also allows for the separation of presentation and content, which makes it easier to maintain and update a website over time. It also enables responsive design, which allows the layout and design of a website to adapt to the size of the device or screen on which it is viewed. This is becoming increasingly important as more and more people access the internet on mobile devices.

CSS also has some advanced features like CSS Grid and Flexbox which enables the developers to create complex and responsive layouts easily. With the help of these features, developers can create grid-based layouts and align elements on a web page in a flexible and responsive way.

In conclusion, HTML and CSS are essential building blocks of the modern web. HTML provides the structure and organization of a web page, while CSS provides the layout and design. Together, they make it possible to create beautiful and functional websites that are easy to navigate and maintain.

2.3.4 JavaScript

JavaScript often abbreviated as JS, is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS. As of 2022, 98 of websites use JavaScript on the client side for webpage behavior, often incorporating third-party libraries. All major web browsers have a dedicated JavaScript engine to execute the code on users devices.

JavaScript is a high-level, often just-in-time compiled language that conforms to the ECMAScript standard.It has dynamic typing, prototype-based object-orientation, and first-class functions. It is multi-paradigm, supporting event-driven, functional, and imperative programming styles. It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures, and the Document Object Model (DOM).

Chapter 3

Database Design

3.1 Entities, Attributes and Relations

- Customer

  Customer Email - Primary key

  Name

  Password

  Mobile Number

  Address

- Category

  Category ID - primary key

  Name

- Product

  Product ID primary key

  Name

  Image

  Description

  Brand

  Price

  Reviews

  Category ID - foreign key references to Category (Category ID)

- Cart

  Customer ID - foreign key references to Customer (Customer Email)

  Product ID - foreign key references to Product (Product ID)

  Quantity

  Primary key(Customer ID, Product ID)

- Orders

  Order ID

  Customer ID - foreign key reference to Customer (Customer ID)

  Order Date

  Total Amount

  Address

  Primary key(Order ID, Customer ID)

3.2 ER Diagram

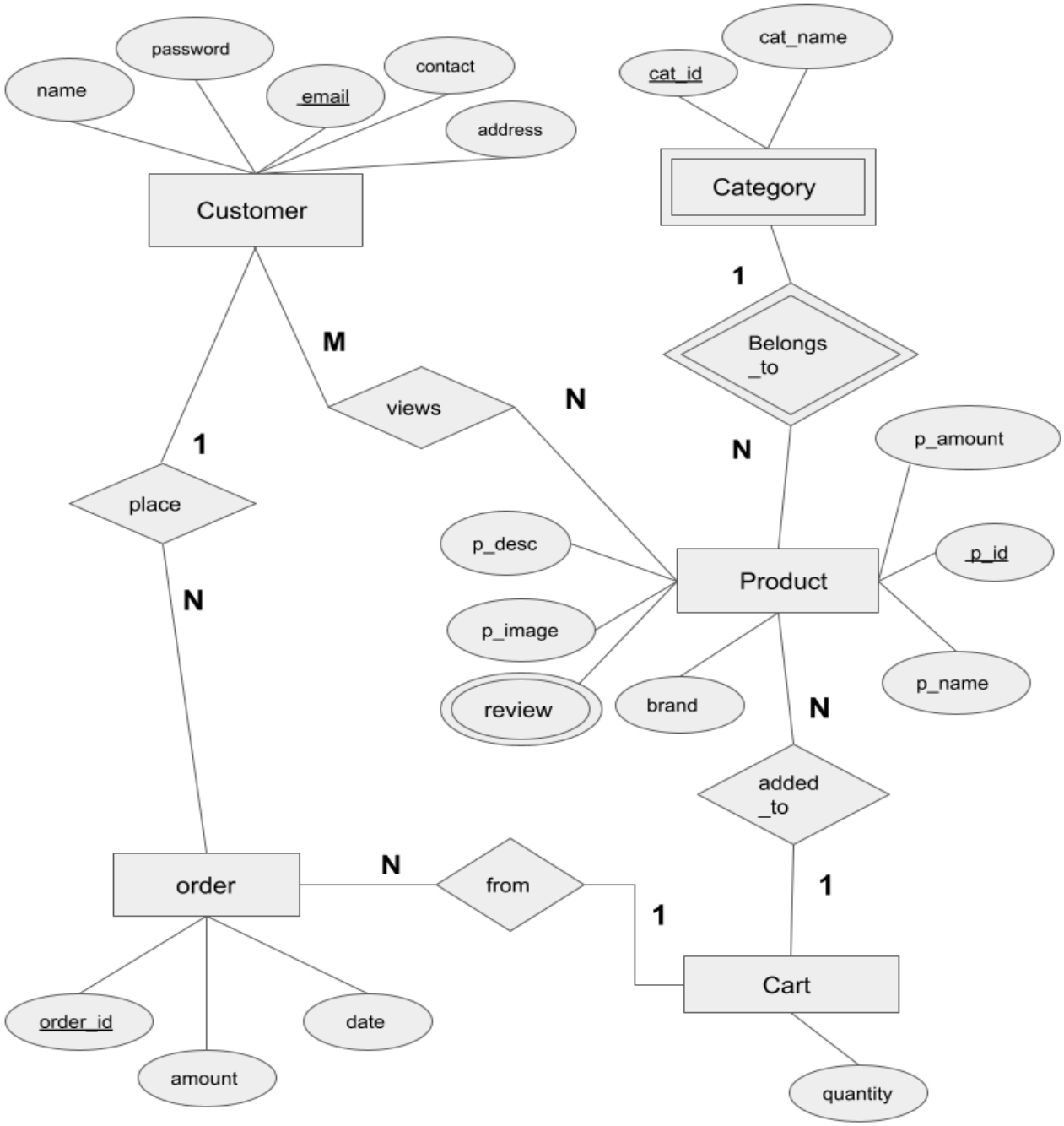

Figure 3.1: ER Diagram

3.3 Schema Diagram

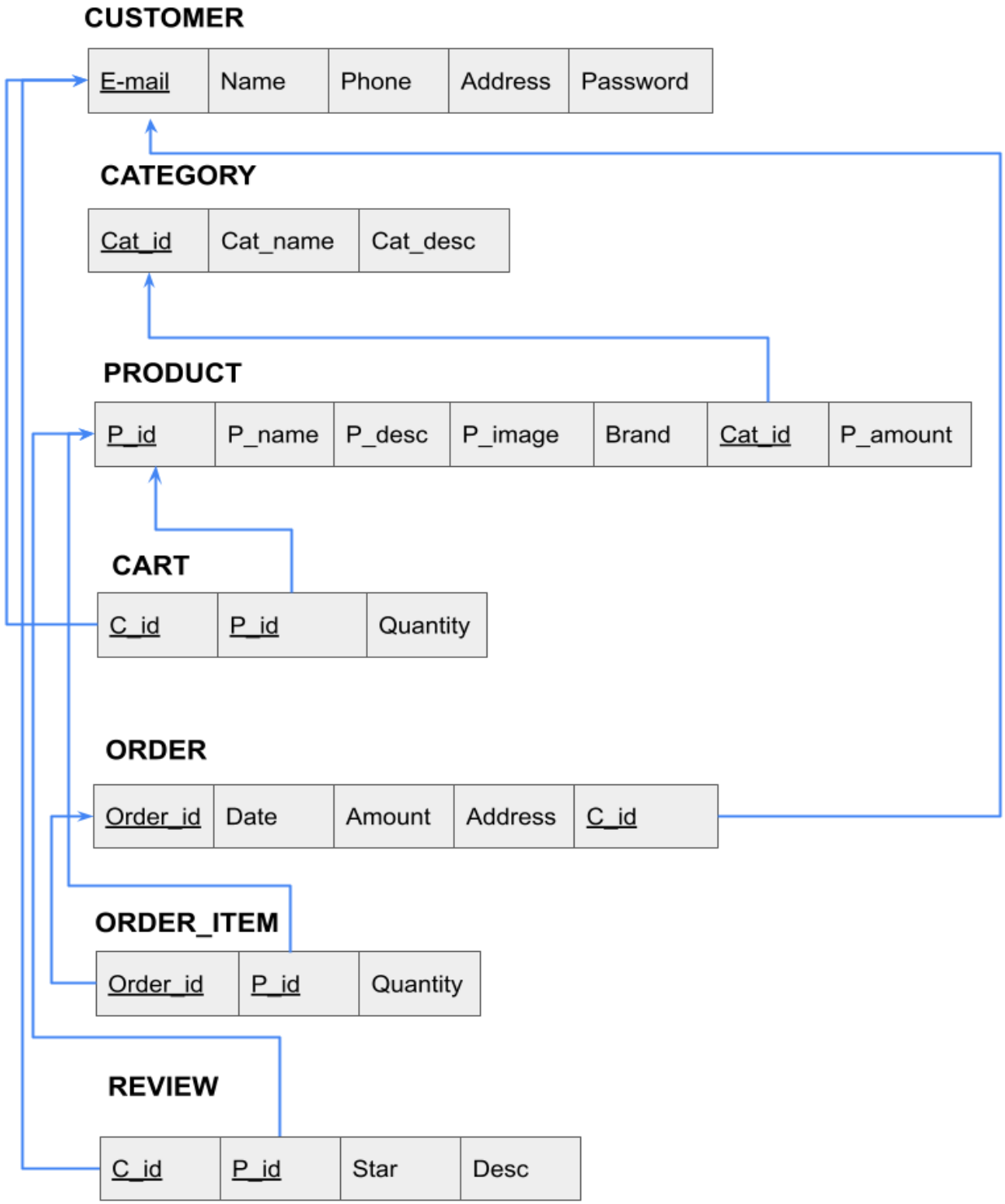

Figure 3.2: Schema Diagram

Chapter 4

Implementation

4.1 Database Connectivity

```
import mysql.connector as sql

m = sql.connect(host="localhost", user="root", passwd="Password",
database="database_name")

cursor = m.cursor()
c = "select * from Customer where Name ='{}' and Password ='{}'"
    .format(name, pwd)
cursor.execute(c)
users = tuple(cursor.fetchall())
c = "insert into product values('{}','{}','{}','{}','{}','{}','{}')"
    .format(P_id, P_name.upper(), P_image, P_desc,P_brand, P_price, P_catid)
cursor.execute(c)
m.commit()
```

4.2 Pseudo code for Major Functionalities

4.2.1 Home Page

```
def index(request):
    m = sql.connect(host="localhost", user="root", passwd="Pramod@12",
        database="register")
    cursor = m.cursor()
    c = "create trigger IF NOT EXISTS order_backup "
        "before delete on orders "
        "for each row "
        "begin "
        "insert into order_history "
        "values(old.ord_id,old.cust_id,old.C_name,sysdate(),old.C_phone,"
        "old.city,old.state,old.total,old.address,old.pincode); "
        "end"
    cursor.execute(c)
    m.commit()
    c = "create procedure IF NOT EXISTS remove_from_cart
        (in cust_id varchar(200),in P_id int)"
        " begin"
        " delete from cart where cart.cust_id = cust_id and
        cart.P_id = P_id; "
        "end"
    cursor.execute(c)
    c = "create procedure IF NOT EXISTS delete_cart(in cust_id varchar(30))
        begin delete from orders  where orders.cust_id  = cust_id ;
        delete from cart where cart.cust_id = cust_id ;end"
    cursor.execute(c)
    return render(request, 'home.html')
```

4.2.2 Login

```
def login_check(request):
 if request.method == "POST":
     name = request.POST.get('name', '')
     pwd = request.POST.get('password', '')
     m = sql.connect(host="localhost", user="root", passwd="Pramod@12",
     database="register")
     cursor = m.cursor()
     c = "select * from users where Name ='{}' and  Password = '{}'
     ".format(name, pwd)
     cursor.execute(c)
     t = tuple(cursor.fetchall())
     us = authenticate(username=name, password=pwd)
     if us is not None:
         login(request, us)
         messages.success(request, "Successfully logged in")
         return redirect('shop')
     else:
         messages.error(request, "Wrong password or user_id")
         return redirect('/shop/login_check/')
 return render(request, 'login.html')
```

4.2.3 Registration

```
def register(request):
    if request.method == "POST":
        m = sql.connect(host="localhost", user="root", passwd="Pramod@12",
        database="register")
        cursor = m.cursor()
        name = request.POST.get('name', '')
        password = request.POST.get('password', '')
        email = request.POST.get('email', '')
        phone = request.POST.get('phone', '')
        adrs = request.POST.get('address', '')
```

```
        city = request.POST.get('city', '')
        state = request.POST.get('state', '')
        pincode = request.POST.get('pincode', '')
        user = User.objects.create_user(name, email, password)
        user.save()
        l = "select * from users where Phone='{}' or
        Email='{}'".format(phone, email)
        cursor.execute(l)
        p = tuple(cursor.fetchall())
        if p != ():
            messages.error(request, 'Email already exists')
            return redirect('/shop/register')
        c = "insert into users values (
        '{}','{}','{}','{}','{}','{}','{}','{}' )".format(name, password,
        email, adrs,city,state, phone, pincode)
        cursor.execute(c)
        m.commit()
        messages.success(request, 'Registration successful')
        return redirect('/shop/login_check')
    else:
        return render(request, 'register.html')
```

4.2.4 Cart

```
def cart(request, myid):
    if myid != 9481:
        m = sql.connect(host="localhost", user="root", passwd="Pramod@12",
        database="register")
        cursor = m.cursor()
        if request.user.is_authenticated:
            cust = request.user.email
            c = "select * from product where P_id='{}'".format(myid)
            cursor.execute(c)
            t = tuple(cursor.fetchall())
```

```
            d = {}
            c = "insert into cart values ('{}','{}','{}')".format(cust,
            t[0][0], 1)
            cursor.execute(c)
            m.commit()
            return redirect('/shop')
        else:
            messages.error(request, "Login to add the items to cart")
            return redirect('/shop')
    m = sql.connect(host="localhost", user="root", passwd="Pramod@12",
    database="register")
    cursor = m.cursor()
    n = 0
    d = []
    total = 0
    if request.user.is_authenticated:
        cust = request.user.email
        c = "select count(*) from cart where cust_id ='{}'".format(cust)
        cursor.execute(c)
        n = tuple(cursor.fetchall())
        if n != ():
            n = n[0][0]
        else:
            n = 0
        c = "select * from product p JOIN cart c on p.P_id=c.P_id and
        cust_id ='{}' ".format(cust)
        cursor.execute(c)
        t = tuple(cursor.fetchall())
        d = []
        total = 0
        if t != ():
            for i in range(0, len(t)):
                params = {'Pid': t[i][0], 'Pname': t[i][1], 'P_image': t[i][2],
                          'P_desc': t[i][3], 'P_brand': t[i][4], 'P_price': t[i][5],
```

```
                        'P_quantity': t[i][9]
                        }
                total += (t[i][9] * t[i][5])
                d.append(params)
    if total > 1000:
        tot = total - 100
        dis = 100
    else:
        tot = total
        dis = 0
    return render(request, 'cart.html', {"len": n, "dict": d, "total":
    total, "tot": tot, "dis": dis})
```

4.2.5 Orders

```
def order(request):
    m = sql.connect(host="localhost", user="root", passwd="Pramod@12",
    database="register")
    cursor = m.cursor()
    cust = request.user.email
    c = "select * from users where Email ='{}'".format(cust)
    cursor.execute(c)
    us1 = tuple(cursor.fetchall())
    c = "select * from orders where cust_id='{}'".format(cust)
    cursor.execute(c)
    us = tuple(cursor.fetchall())
    print(us1)
    if us == ():
        c = "insert into orders(cust_id,C_name,ord_date,C_phone,
            city,state,address,pincode)" values('{}','{}',sysdate(),
            '{}','{}','{}','{}','{}')".format(cust,us1[0][0],
            us1[0][6], us1[0][4],us1[0][5],us1[0][3], us1[0][7])
        cursor.execute(c)
        m.commit()
```

```
    c = "select * from orders where cust_id='{}'".format(cust)
    cursor.execute(c)
    us = tuple(cursor.fetchall())
c = "select * from product p JOIN cart c on p.P_id=c.P_id and
    cust_id ='{}' ".format(cust)
cursor.execute(c)
t = tuple(cursor.fetchall())
d = []
cus = us[0]
customer = {'C_name': cus[2].capitalize(), 'C_Email': cus[1],
'C_Addr1': cus[8], 'C_city': cus[5],'C_state': cus[6],
'C_phone': cus[4], 'ord_id': cus[0]}
total = 0
if t != ():
    for i in range(0, len(t)):
        params = {'Pid': t[i][0], 'Pname': t[i][1].capitalize(),
        'P_image': t[i][2],'P_desc': t[i][3], 'P_brand': t[i][4],
        'P_price': t[i][5], 'P_quantity': t[i][9]}
        total += (t[i][5] * t[i][9])
        d.append(params)
if total > 1000:
    dis = 100
    total = total - 100
else:
    dis = 0
c = "update orders set total='{}'where cust_id='{}'".
    format(total, cust)
cursor.execute(c)
m.commit()
return render(request, 'order.html', {'dit': d, 'total': total,
        'dis': dis, 'cust': customer})
```

Chapter 5

Results and Snapshots

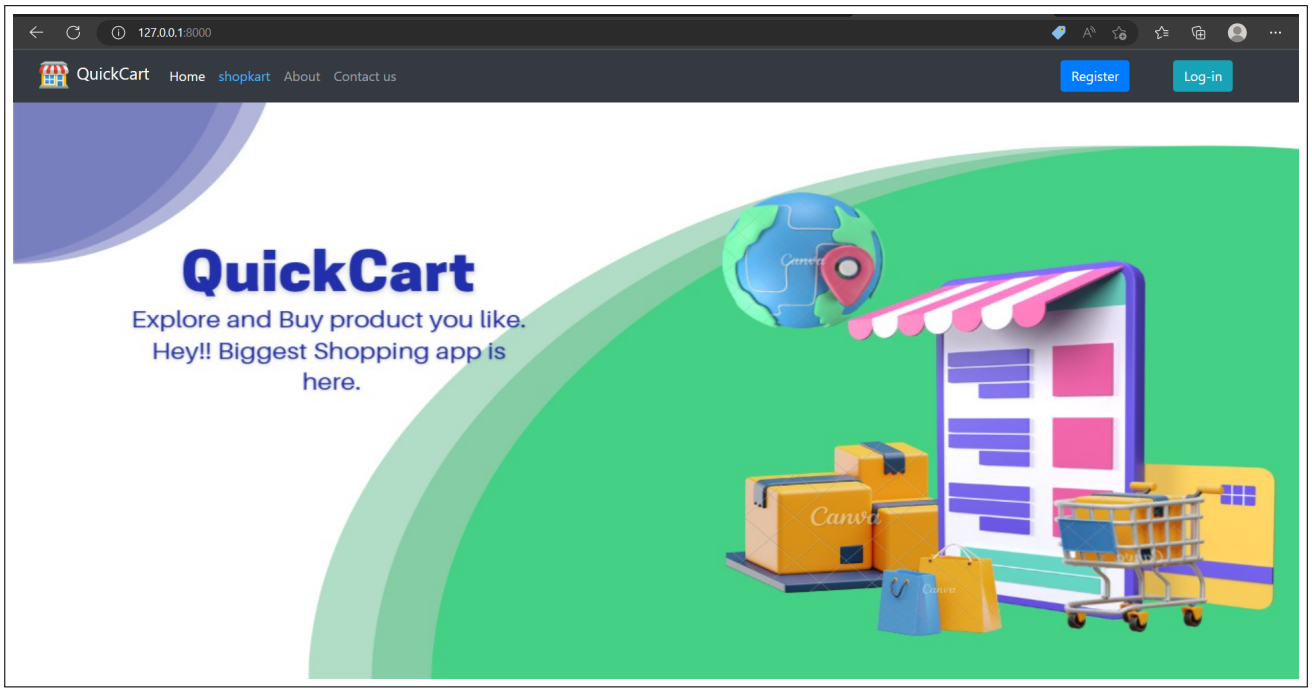

Figure 5.1: Home

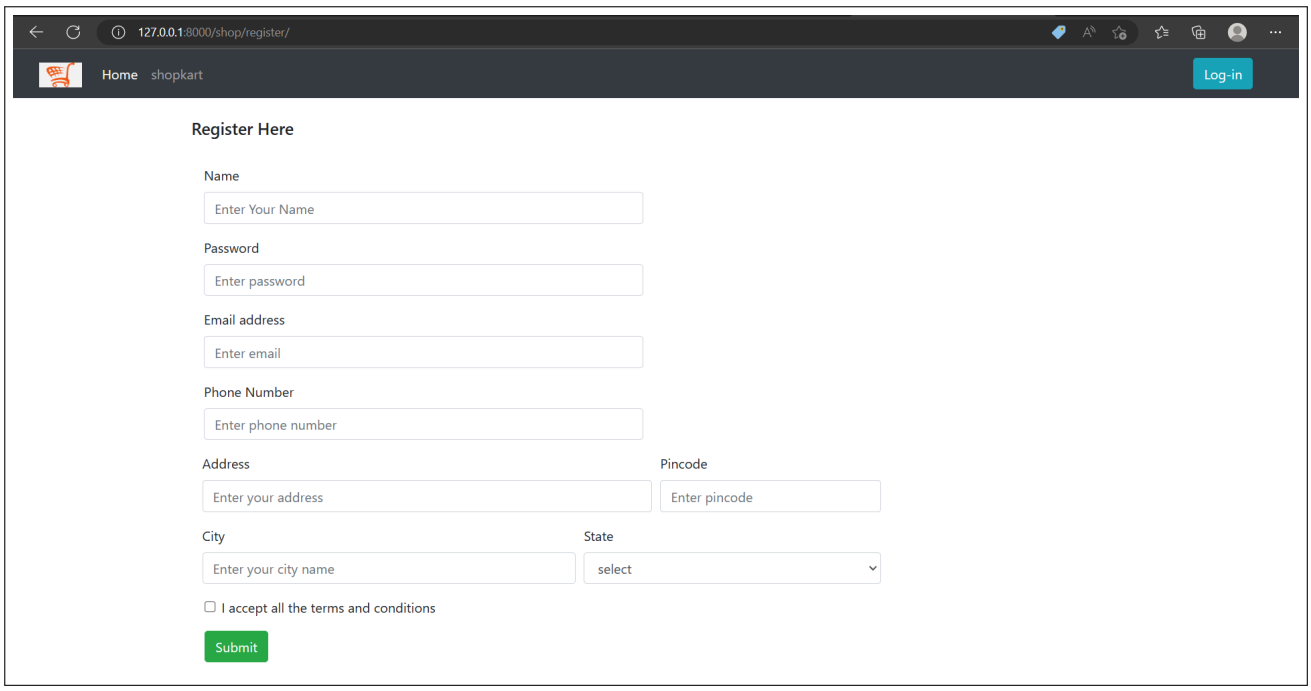

Figure 5.2: Registration

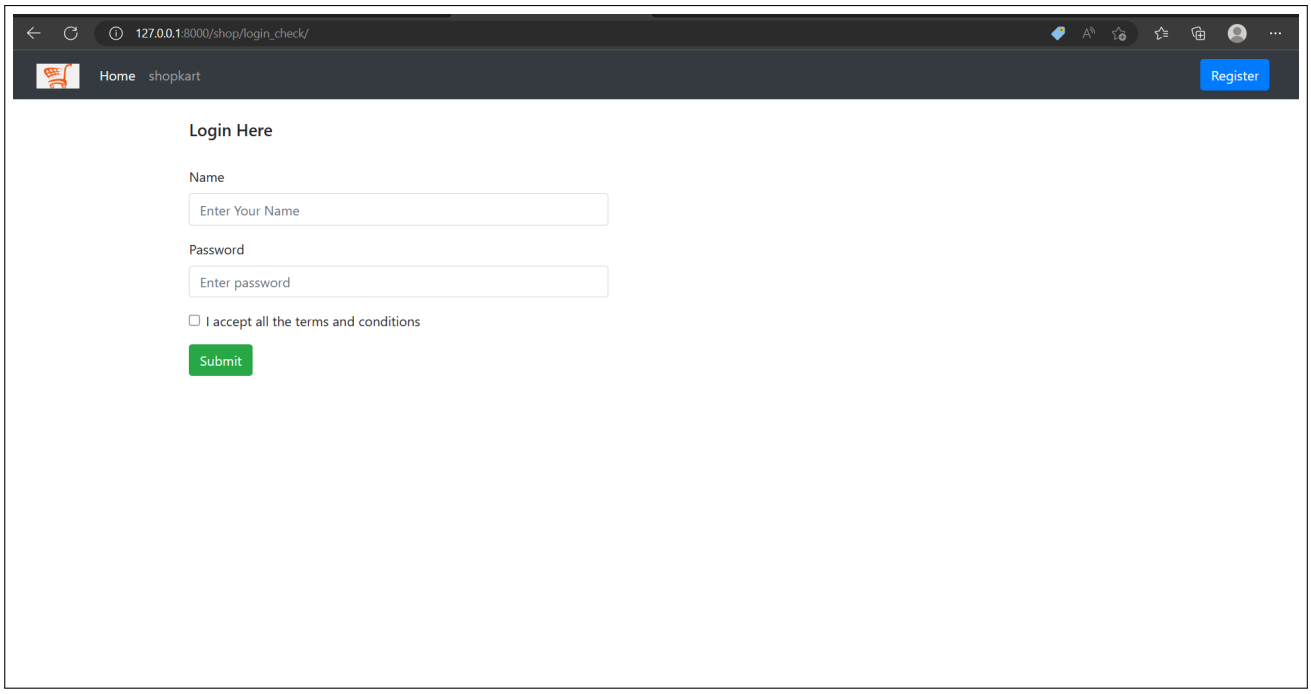

Figure 5.3: Login

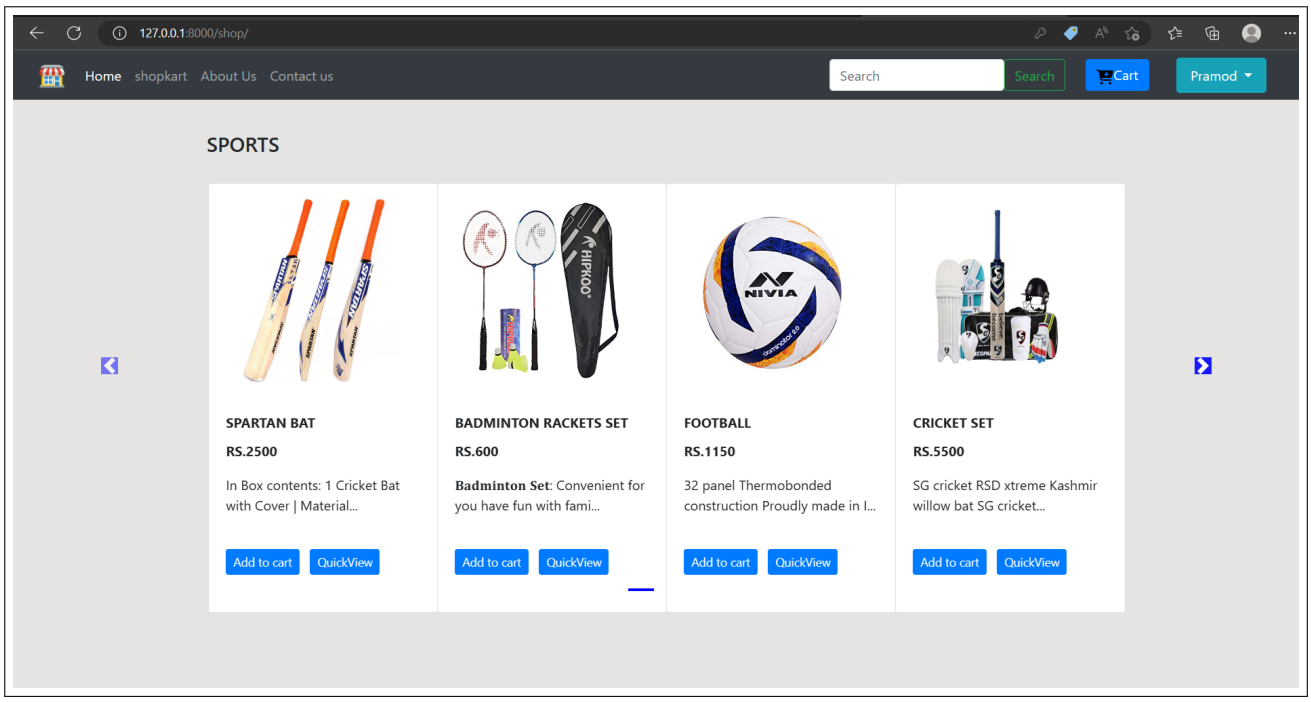

Figure 5.4: Products

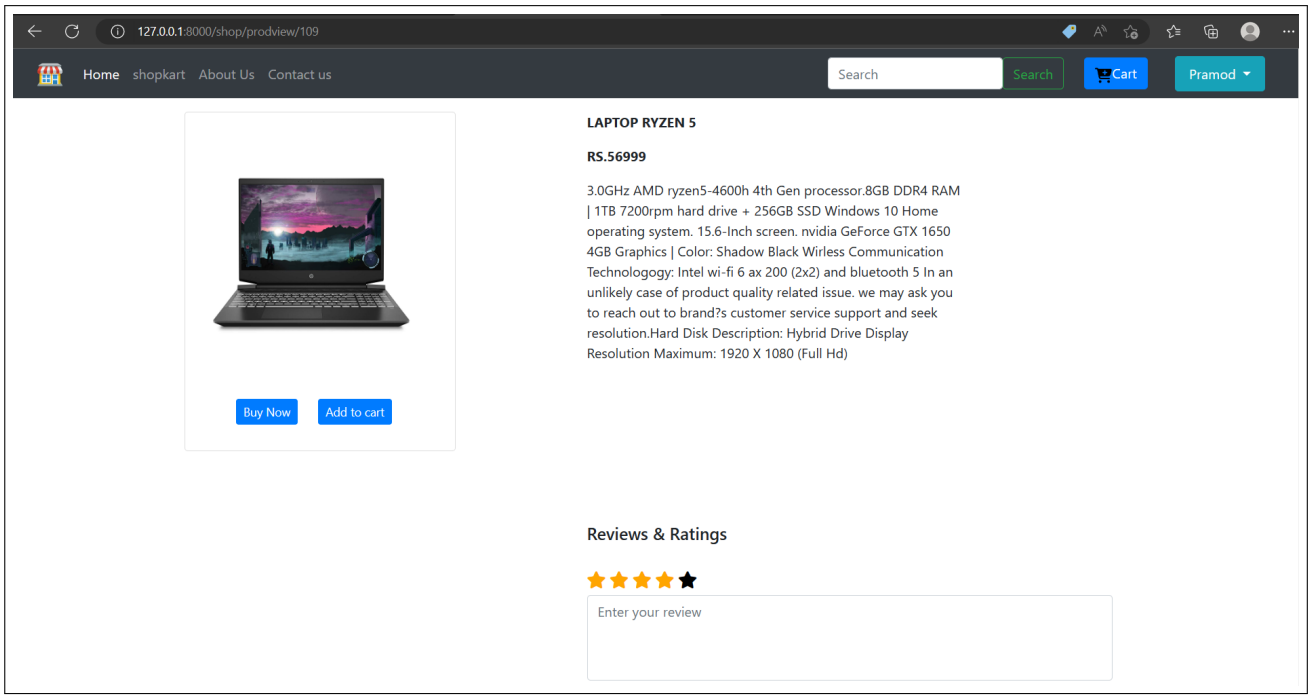

Figure 5.5: Product view

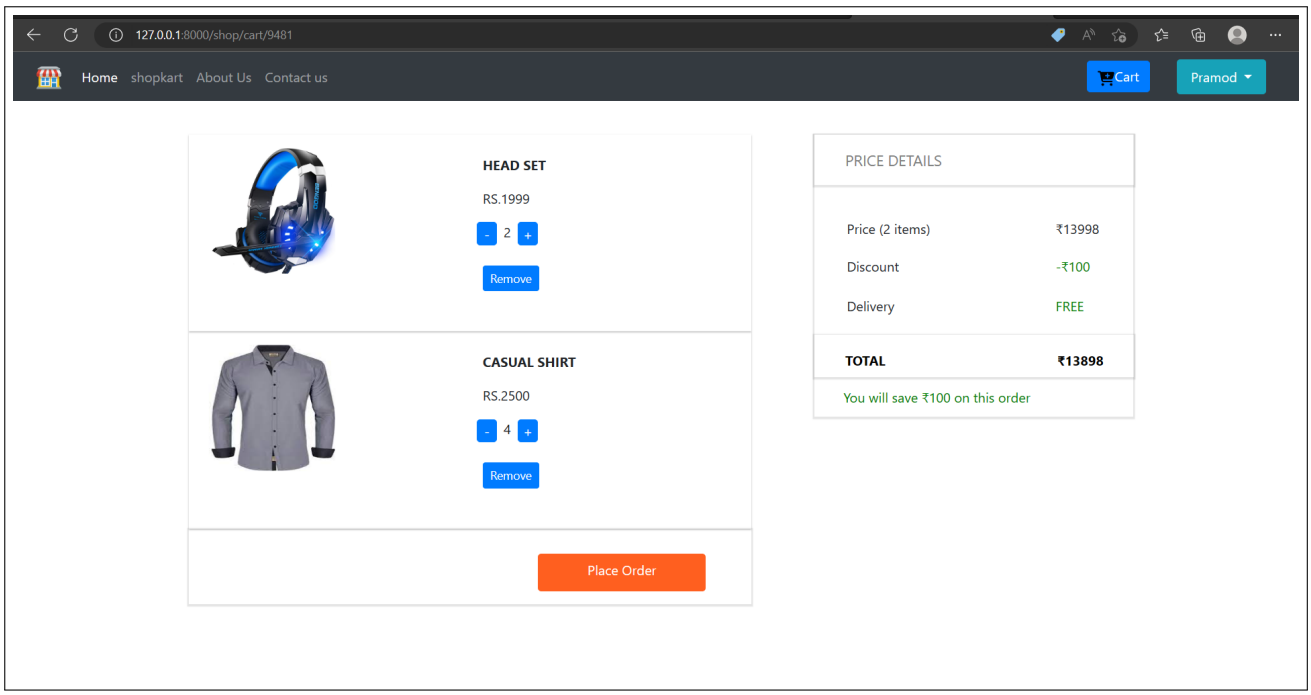

Figure 5.6: Cart

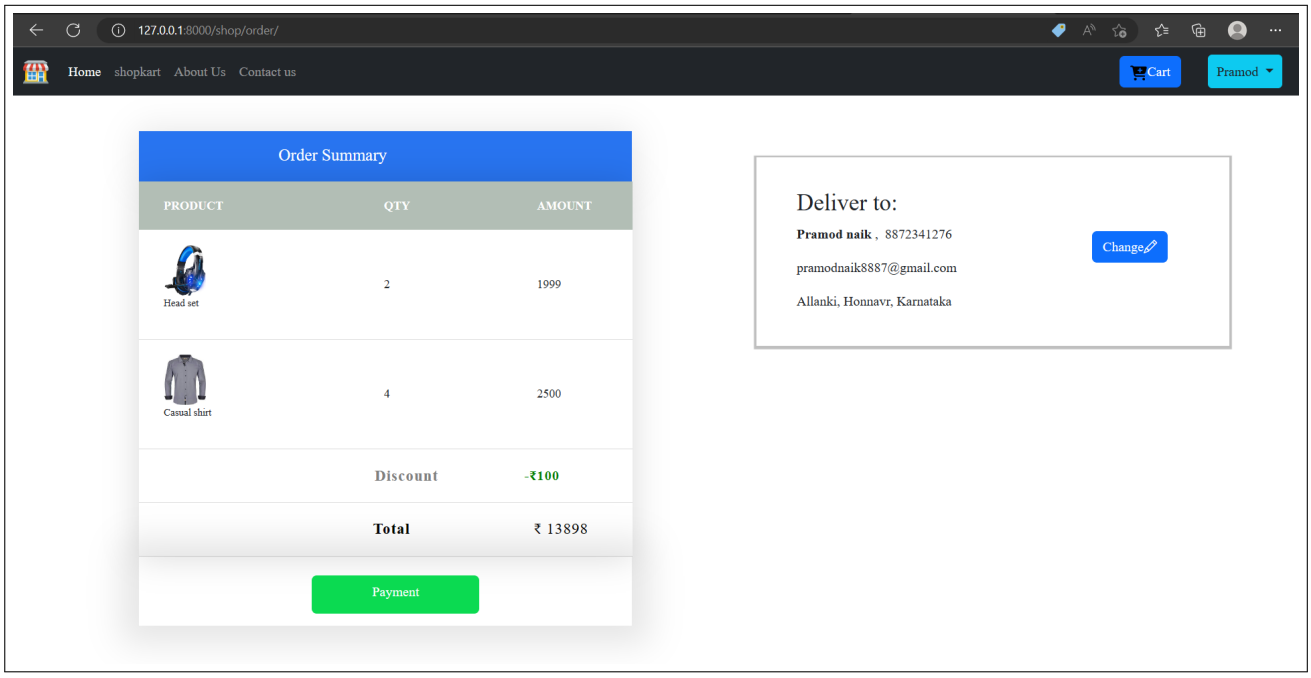

Figure 5.7: Order

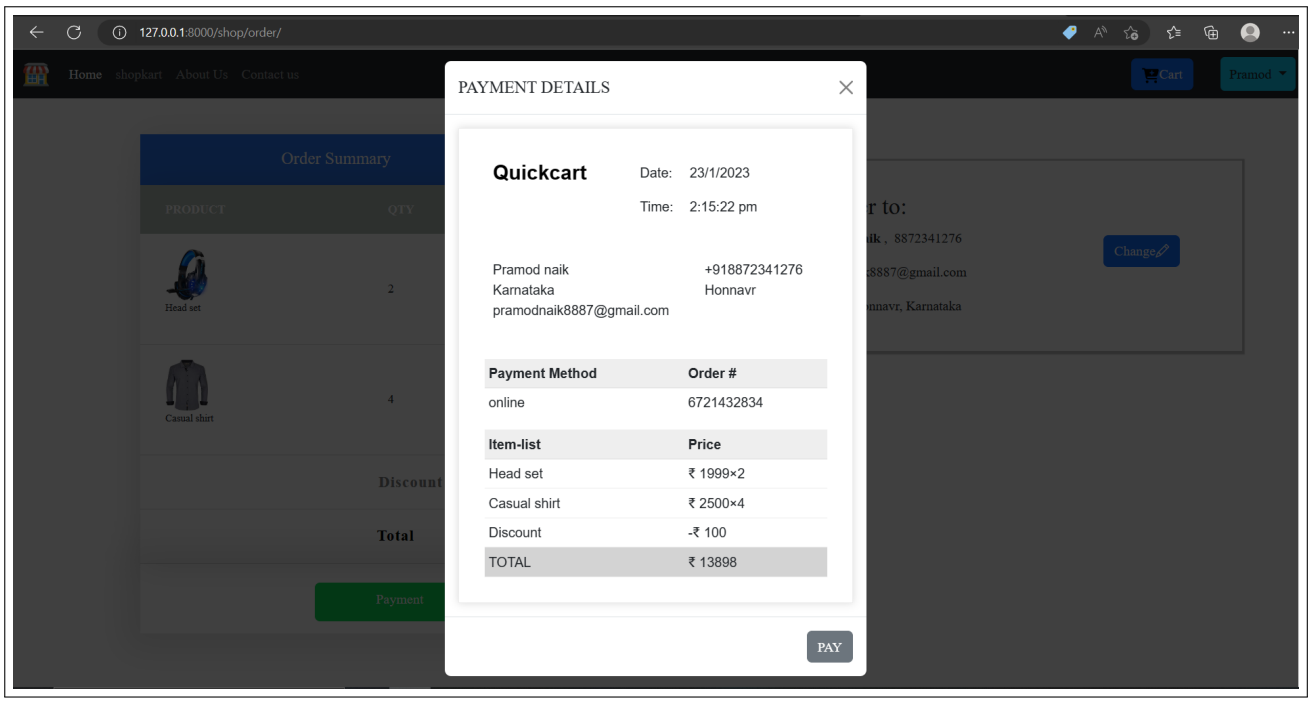

Figure 5.8: Payment

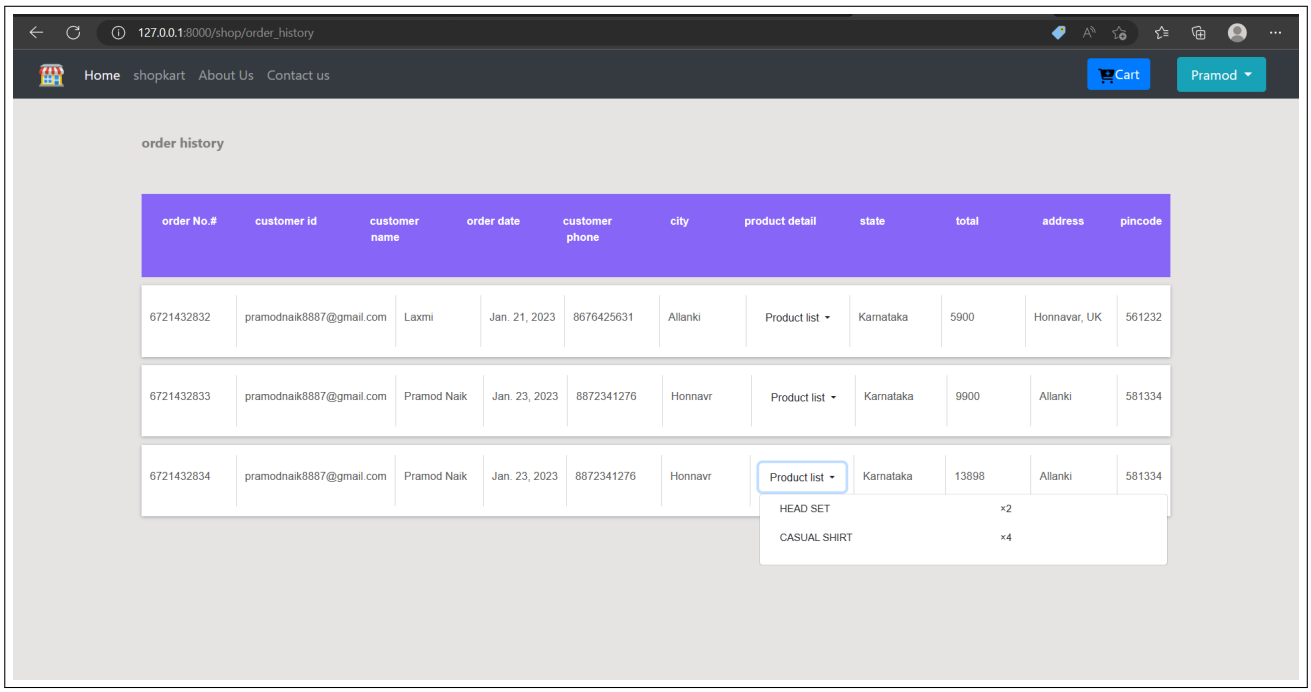

Figure 5.9: Order history

Chapter 6

Conclusion & Future Enhancements

6.1 Conclusion

In conclusion, an e-commerce website is a vital tool for businesses looking to expand their reach and increase sales. It allows businesses to sell products and services to customers all over the world, 24/7. Ecommerce websites can be designed to be user-friendly and visually appealing, with features such as product catalogs, shopping carts, and secure checkout to make the buying process as smooth as possible for customers. Additionally, ecommerce websites can be optimized for search engines, making it easier for customers to find products and services they are looking for.

Another important aspect of ecommerce websites is the use of data and analytics to track customer behavior and preferences. This information can be used to improve the customer experience and increase sales. For example, businesses can use data to personalize their marketing efforts, recommend products, and offer targeted discounts.

However, it is important to note that creating a successful e-commerce website requires careful planning, design and development, including the selection of the appropriate platform, payment gateways and security measures. It also requires ongoing maintenance and updates to ensure that it remains secure and user-friendly.

Overall, an e-commerce website can help businesses reach new customers, increase sales, and improve the overall customer experience. With the right strategy and implementation, an e-commerce website can be an extremely valuable asset for any business.

6.2 Future Enhancements

E-commerce websites are constantly evolving to improve the customer experience and streamline the buying process. In the future, we can expect to see the following enhancements:

Payment: Payment using credit card/debit card is one of most common mode of electronic payment and other options such as Apple Pay and Google Wallet, ecommerce websites will integrate these options to make checkout faster and more secure.

Mobile Optimization: With more and more customers shopping on their mobile devices, ecommerce websites will continue to optimize their mobile experience to make it easy for customers to shop on the go.

Personalization: Websites will use artificial intelligence and machine learning to personalize the shopping experience for each customer. This can include personalized product recommendations, customized search results, and tailored promotions.

Faster Delivery: With the growth of drone and autonomous vehicle technology, e-commerce websites will be able to offer faster delivery options, such as same-day or even same-hour delivery.

Overall, e-commerce websites will continue to use technology to improve the customer experience and make the buying process faster and more convenient.

References

- Django Documentation : docs.djangoproject.com/en/4.1/
- Bootstrap : getbootstrap.com
- GeeksforGeeks : www.geeksforgeeks.org/web-technology/html-css/
- w3schools : www.w3schools.com/html/html-css.asp