RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

ONLINE GRAMENT SHOPPING

CS23332 - Database Management Systems

Submitted by

Mugunthan S(231001125/IT-B)
Lingesh C(231001100/IT-B)

Mini Project
November 2024

Department of Information Technology
Rajalakshmi Engineering College
Thandalam, Chennai - 602105

BONAFIDE CERTIFICATE

This is to certify that the Mini project work titled **"ONLINE GRAMENT SHOPPING"** done by Mugunthan S(231001125) ,Lingesh C(231001100)a record of bonafide work carried out under my supervision as a part of the Mini Project for the course CS23332 – Database Management Systems, Department of Information Technology, REC.

Supervisor:

Ms.T.Sangeetha

Assistant Professor ,

Department of Information Technology,

Rajalakshmi Engineering College, Thandalam,

Chennai-602105

Head/IT:

Dr.P.Valarmathie

Professor and HOD,

Department of Information Technology,

Rajalakshmi Engineering College,Thandalam,

Chennai - 602105

DATE:23.11.2024

ABSTRACT

Online garment shopping has transformed the fashion industry, offering consumers an accessible, convenient way to purchase clothing from a wide range of brands and retailers. The growth of e-commerce platforms has been propelled by advancements in technology, the ubiquity of smartphones, and the increasing consumer preference for shopping from home. With a variety of online stores, shoppers have the ability to browse through an extensive selection of garments without geographical constraints, making it easier to access global fashion trends and niche brands.

One of the key advantages of online garment shopping is the personalized experience it offers. Through artificial intelligence and data analytics, platforms provide tailored recommendations based on consumer preferences and past behavior. Virtual fitting rooms, size guides, and customer reviews also enhance the shopping experience by helping consumers make more informed decisions. Moreover, the ease of comparing prices across different websites allows for more price-conscious shopping.

However, the growth of online garment shopping is not without its challenges. Common issues include sizing discrepancies, inability to feel or try on products, and concerns over product quality. Return policies, shipping delays, and the environmental impact of fast fashion are also significant concerns for consumers. To address these issues, many retailers are adopting augmented reality tools, implementing more accurate sizing algorithms, and offering easier return processes.

As online garment shopping continues to evolve, future trends are expected to focus on sustainability, faster delivery options, and greater integration of online and offline shopping experiences, making the process more seamless, personalized, and efficient for consumers.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1 Problem Statement

Online garment shopping faces challenges related to sizing discrepancies, the inability to physically try on items, and the environmental impact of returns and packaging waste. Consumers often struggle with product quality assessments, leading to dissatisfaction and increased returns. Logistical issues, such as shipping delays and delivery costs, also complicate the experience. Retailers face difficulties in inventory management and competition in the global market. This research aims to explore these challenges and propose solutions to improve the online garment shopping experience.

1.2 Objective of Project (Online Garment Shopping)

1. **User Account Management**: Develop a secure and easy-to-use platform that allows customers to create an account, log in securely, update their profiles, and manage their shopping preferences and order history.
2. **Product Browsing and Personalization**: Implement a user-friendly interface that allows customers to browse and search for garments, with personalized recommendations based on browsing history, preferences, and past purchases.
3. **Secure Payment System:** Integrate a secure and reliable payment gateway to ensure safe transactions, supporting multiple payment methods while safeguarding user financial data.
4. **Sizing and Fit Assistance**: Provide an effective size guide, virtual fitting room, or AI-driven fit prediction tools to help customers select the right size, reducing returns and improving satisfaction.
5. **Order Tracking and Delivery Management:** Enable customers to track their orders in real-time, view delivery status, and receive notifications for shipping and delivery updates.
6. **Scalability and Performance:** Design the system to handle high volumes of traffic and transactions, ensuring seamless performance even during peak shopping periods such as sales and holidays.

CHAPTER 2

SYSTEM DESIGN

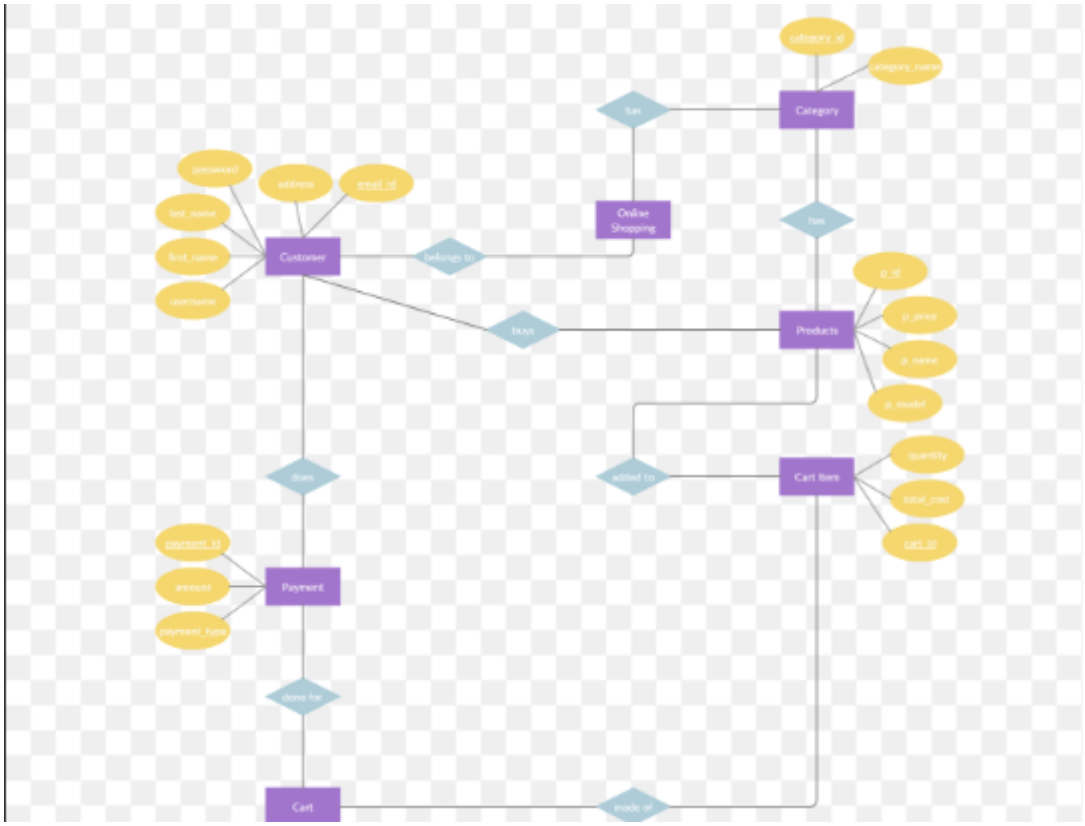
2.1 System Architecture

The Online Garment Store system follows a client-server architecture with a modular design, consisting of two main components: the client-side (front-end) and the server-side (back-end).

1. **Client-Side:**
   Users interact with the system through a responsive web interface built with HTML, CSS, and JavaScript. It includes pages for browsing garments, product details, shopping cart, order placement, user registration, and login.

2. **Server-Side:**
   The back-end is built with Flask, handling user authentication, order management, and product data storage. MongoDB is used for database management. The server also integrates features such as order tracking, payment processing, and customer support via email.

2.2 ER Diagram



2.3 System Specification

Software Requirements:

The User Identification System relies on a range of software tools to ensure functionality and ease of development. Flask, a lightweight Python framework, powers the backend, handling routing, user authentication, and request processing, while Python facilitates seamless interaction between components. MongoDB, a NoSQL database, manages user data, notes, and system information with scalability and flexibility. The frontend is built using HTML5, CSS3, and JavaScript (ES6) for a user-friendly, visually appealing interface. For security, Werkzeug ensures secure password hashing and authentication, with SSL/TLS for encrypted communication in online deployments.

Development is supported by IDEs like PyCharm or VS Code, Git for version control, and Postman for API testing. Optional tools, including Docker for containerization and Nginx for load balancing, improve scalability and deployment. The system is designed to function across modern

browsers, such as Chrome, Firefox, Safari, and Edge, ensuring broad compatibility and a seamless user experience.

Hardware Requirements:

Client-Side

- **Device**: Desktop, laptop, tablet, or smartphone.
- **Processor**: Dual-core processor (2 GHz or higher).
- **RAM**: 4 GB (8 GB recommended).
- **Storage**: At least 2 GB of free space.
- **Internet**: Stable broadband connection (minimum 5 Mbps).
- **Browser**: Latest version of Chrome, Firefox, Safari, or Edge.

Server Side:

- **Processor**: Quad-core CPU (2.5 GHz or higher).
- **RAM**: 8 GB (16 GB recommended).
- **Storage**: SSD with at least 256 GB.
- **Network**: High-speed internet connection.

Tools/Platforms:

Tools/Platforms for Online Garment Shopping System (Java & SQL)

Backend Development

- **Java**: Primary language for backend logic and system functionality.
- **Spring Boot**: Java framework for creating stand-alone, production-grade applications.

- **Hibernate**: ORM framework to map Java objects to SQL database tables.

Database Management

- **MySQL** or **PostgreSQL**: Relational databases for structured data storage and management.
- **JDBC (Java Database Connectivity)**: For connecting and interacting with the SQL database.

Frontend Development

- **HTML5**, **CSS3**, **JavaScript**: For building the user interface and interactivity.
- **Angular** or **React**: JavaScript frameworks for dynamic, responsive front-end.

Version Control

- **Git**: Version control for managing code changes.
- **GitHub** or **GitLab**: Platforms for repository hosting and collaboration.

Testing

- **JUnit**: For backend unit testing in Java.
- **Selenium**: For automated frontend testing.

Deployment and Hosting

- **AWS**, **Google Cloud**, **Heroku**: Cloud platforms for hosting applications.
- **Apache Tomcat**: Server for deploying Java-based web applications.

Security

- **Spring Security**: Framework for authentication and authorization in Java.
- **SSL/TLS Certificates**: For securing data transmission.
- **Cloudflare**: For DDoS protection.

CHAPTER 3

IMPLEMENTATION

3.1 Sign-up Page

3.2 Program:

```
package lf;


import javax.swing.*;
import javax.swing.border.*;
import java.awt.*;
import com.mongodb.client.MongoCollection;
import org.bson.Document;
import com.mongodb.client.model.Filters;


public class RegisterPage extends JFrame {
    private JTextField emailField;
    private JPasswordField passwordField, confirmPasswordField;
    private JButton registerButton, loginButton;
    private JLabel messageLabel;
    private Color primaryColor = new Color(30, 136, 229); // Brighter blue
    private Color backgroundColor = Color.WHITE;
    private Font mainFont = new Font("Segoe UI", Font.PLAIN, 14);
    private Font titleFont = new Font("Segoe UI", Font.BOLD, 32);
    private Font subtitleFont = new Font("Segoe UI", Font.BOLD, 24);


    public RegisterPage() {
```

```
// Basic frame setup
setTitle("Lost and Found - Rajalakshmi");
setSize(400, 700);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setLocationRelativeTo(null);
setResizable(false);

// Main panel with padding
JPanel mainPanel = new JPanel();
mainPanel.setLayout(new BoxLayout(mainPanel, BoxLayout.Y_AXIS));
mainPanel.setBackground(backgroundColor);
mainPanel.setBorder(BorderFactory.createEmptyBorder(40, 40, 40, 40));

// App name
JLabel appNameLabel = new JLabel("Lost and Found");
appNameLabel.setFont(titleFont);
appNameLabel.setForeground(primaryColor);
appNameLabel.setAlignmentX(Component.CENTER_ALIGNMENT);

// Welcome text
JLabel titleLabel = new JLabel("Create Account");
titleLabel.setFont(subtitleFont);
```

```
        titleLabel.setForeground(new Color(33, 33, 33));
        titleLabel.setAlignmentX(Component.CENTER_ALIGNMENT);

        // Subtitle
        JLabel subtitleLabel = new JLabel("Sign up to get started");
        subtitleLabel.setFont(mainFont);
        subtitleLabel.setForeground(new Color(117, 117, 117));
        subtitleLabel.setAlignmentX(Component.CENTER_ALIGNMENT);

        // Form container
        JPanel formContainer = new JPanel();
        formContainer.setLayout(new BoxLayout(formContainer, BoxLay-
out.Y_AXIS));
        formContainer.setBackground(backgroundColor);
        formContainer.setAlignmentX(Component.CENTER_ALIGNMENT);
        formContainer.setBorder(BorderFactory.createEmptyBorder(30, 0, 30, 0));

        // Email field
        JLabel emailLabel = new JLabel("Email Address");
        emailLabel.setFont(mainFont);
        emailLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
        emailField = new JTextField(20);
```

```
        styleTextField(emailField);

        // Password field
        JLabel passwordLabel = new JLabel("Password");
        passwordLabel.setFont(mainFont);
        passwordLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
        passwordField = new JPasswordField(20);
        styleTextField(passwordField);

        // Confirm Password field
        JLabel confirmPasswordLabel = new JLabel("Confirm Password");
        confirmPasswordLabel.setFont(mainFont);
        confirmPasswordLabel.setAlignmentX(Component.CENTER_ALIGN-
MENT);
        confirmPasswordField = new JPasswordField(20);
        styleTextField(confirmPasswordField);

        // Register button
        registerButton = new JButton("Sign Up");
        styleButton(registerButton);
        registerButton.addActionListener(e -> registerUser());
```

```
// Login button
loginButton = new JButton("Already have an account? Log in");
loginButton.setFont(mainFont);
loginButton.setForeground(primaryColor);
loginButton.setBorderPainted(false);
loginButton.setContentAreaFilled(false);
loginButton.setFocusPainted(false);
loginButton.setAlignmentX(Component.CENTER_ALIGNMENT);
loginButton.setCursor(new Cursor(Cursor.HAND_CURSOR));
loginButton.addActionListener(e -> {
    new LoginPage();
    dispose();
});

// Message label
messageLabel = new JLabel();
messageLabel.setFont(mainFont);
messageLabel.setForeground(Color.RED);
messageLabel.setAlignmentX(Component.CENTER_ALIGNMENT);

// Add components with proper spacing
mainPanel.add(Box.createVerticalGlue());
```

```
mainPanel.add(appNameLabel);
mainPanel.add(Box.createVerticalStrut(20));
mainPanel.add(titleLabel);
mainPanel.add(Box.createVerticalStrut(10));
mainPanel.add(subtitleLabel);
mainPanel.add(Box.createVerticalStrut(40));

// Add form elements
formContainer.add(emailLabel);
formContainer.add(Box.createVerticalStrut(10));
formContainer.add(emailField);
formContainer.add(Box.createVerticalStrut(20));
formContainer.add(passwordLabel);
formContainer.add(Box.createVerticalStrut(10));
formContainer.add(passwordField);
formContainer.add(Box.createVerticalStrut(20));
formContainer.add(confirmPasswordLabel);
formContainer.add(Box.createVerticalStrut(10));
formContainer.add(confirmPasswordField);
formContainer.add(Box.createVerticalStrut(30));
formContainer.add(registerButton);
formContainer.add(Box.createVerticalStrut(15));
```

```
        formContainer.add(messageLabel);
        formContainer.add(Box.createVerticalStrut(20));
        formContainer.add(loginButton);

        mainPanel.add(formContainer);
        mainPanel.add(Box.createVerticalGlue());

        // Add main panel to frame
        add(mainPanel);
        setVisible(true);
    }

    private void styleTextField(JTextField field) {
        field.setFont(mainFont);
        field.setMaximumSize(new Dimension(300, 35));
        field.setPreferredSize(new Dimension(300, 35));
        field.setAlignmentX(Component.CENTER_ALIGNMENT);
        field.setBorder(BorderFactory.createCompoundBorder(
            BorderFactory.createLineBorder(new Color(189, 189, 189)),
            BorderFactory.createEmptyBorder(5, 10, 5, 10)
        ));
    }
```

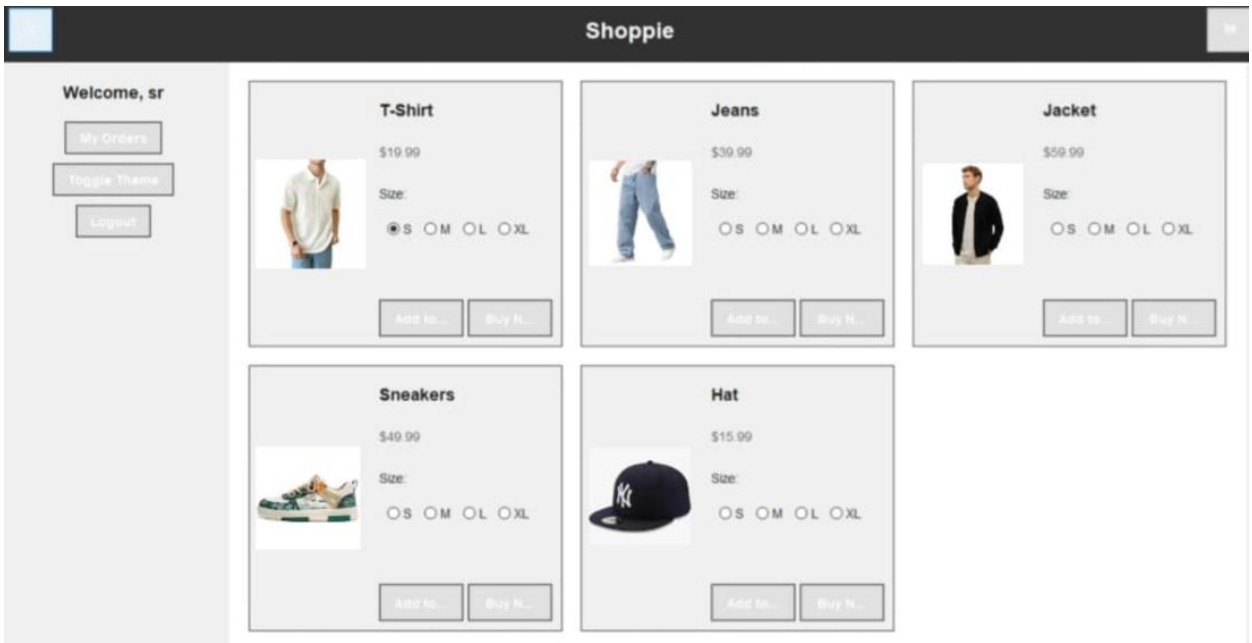
```
private void styleButton(JButton button) {
    button.setFont(new Font("Segoe UI", Font.BOLD, 14));
    button.setForeground(Color.WHITE);
    button.setBackground(primaryColor);
    button.setMaximumSize(new Dimension(300, 40));
    button.setPreferredSize(new Dimension(300, 40));
    button.setAlignmentX(Component.CENTER_ALIGNMENT);
    button.setFocusPainted(false);
    button.setBorderPainted(false);
    button.setOpaque(true);
    button.setCursor(new Cursor(Cursor.HAND_CURSOR));
    button.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseEntered(java.awt.event.MouseEvent evt) {
            button.setBackground(primaryColor.darker());
        }
        public void mouseExited(java.awt.event.MouseEvent evt) {
            button.setBackground(primaryColor);
        }
    });
}
```

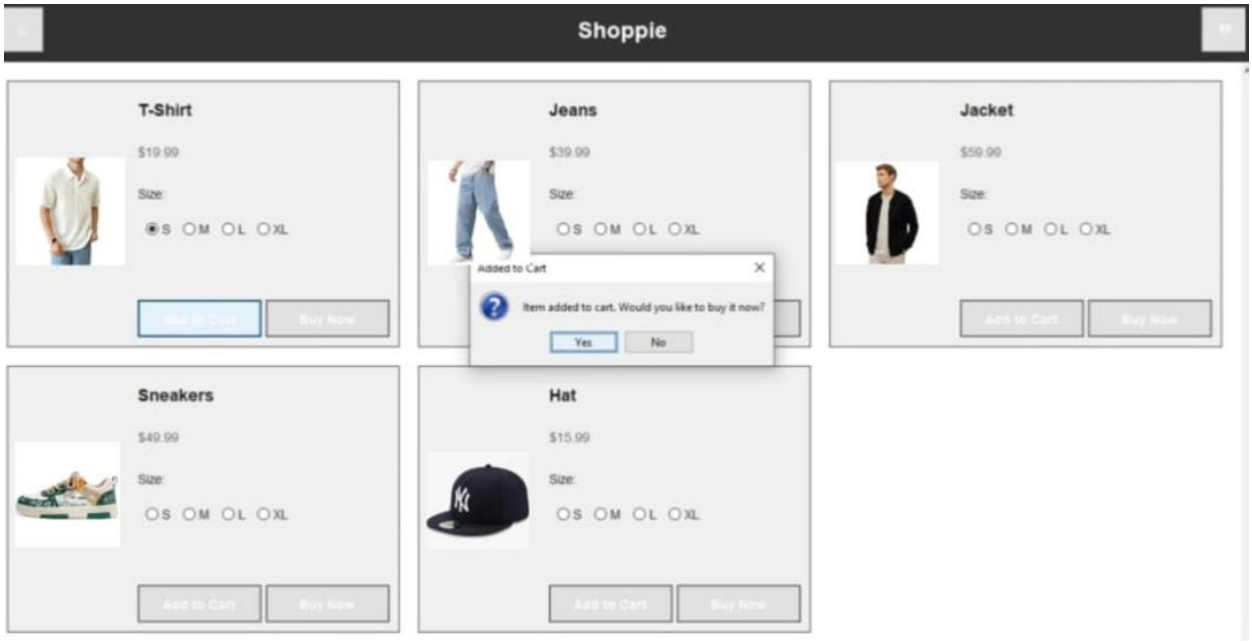
```
    private void registerUser() {
        String email = emailField.getText();
        String password = new String(passwordField.getPassword());
        String confirmPassword = new String(confirmPasswordField.getPassword());

        // Validate email domain
        if (!User.isValidDomain(email)) {
            showError("Invalid email domain. Only @rajalaksh.mi.edu.in allowed.");
            return;
        }

        // Validate password match
        if (!password.equals(confirmPassword)) {
            showError("Passwords don't match.");
            return;
        }

        // Connect to MongoDB and check if email already exists
        MongoCollection<Document> usersCollection = MongoDBUtil.getDatabase().getCollection("users");
```

```
        Document existingUser = usersCollection.find(Filters.eq("email",
email)).first();
        if (existingUser != null) {
            showError("Email already registered.");
            return;
        }

        // Insert the new user
        Document userDoc = new Document("email", email)
                .append("password", password);

        usersCollection.insertOne(userDoc);
        showSuccess("Registration successful.");
    }

    private void showError(String message) {
        messageLabel.setForeground(new Color(198, 40, 40));
        messageLabel.setText(message);
    }

    private void showSuccess(String message) {
        messageLabel.setForeground(new Color(46, 125, 50));
```

```
        messageLabel.setText(message);
    }

    public static void main(String[] args) {
        try {
            UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClass-
Name());
        } catch (Exception e) {
            e.printStackTrace();
        }
        SwingUtilities.invokeLater(() -> new RegisterPage());
    }
}
```

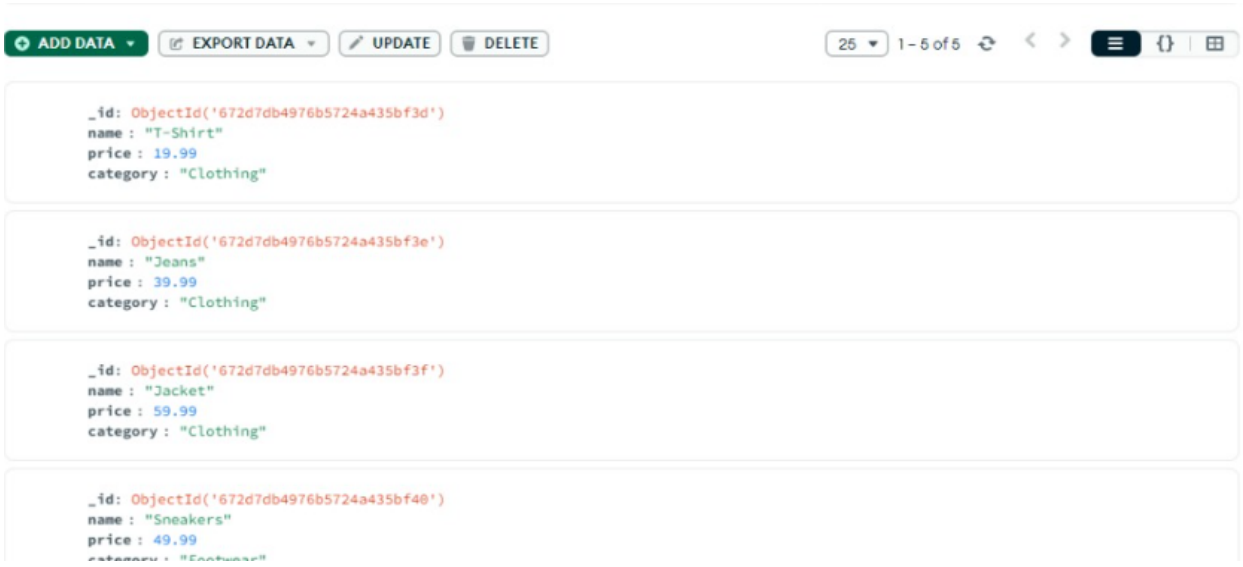
3.3 Cart Page:

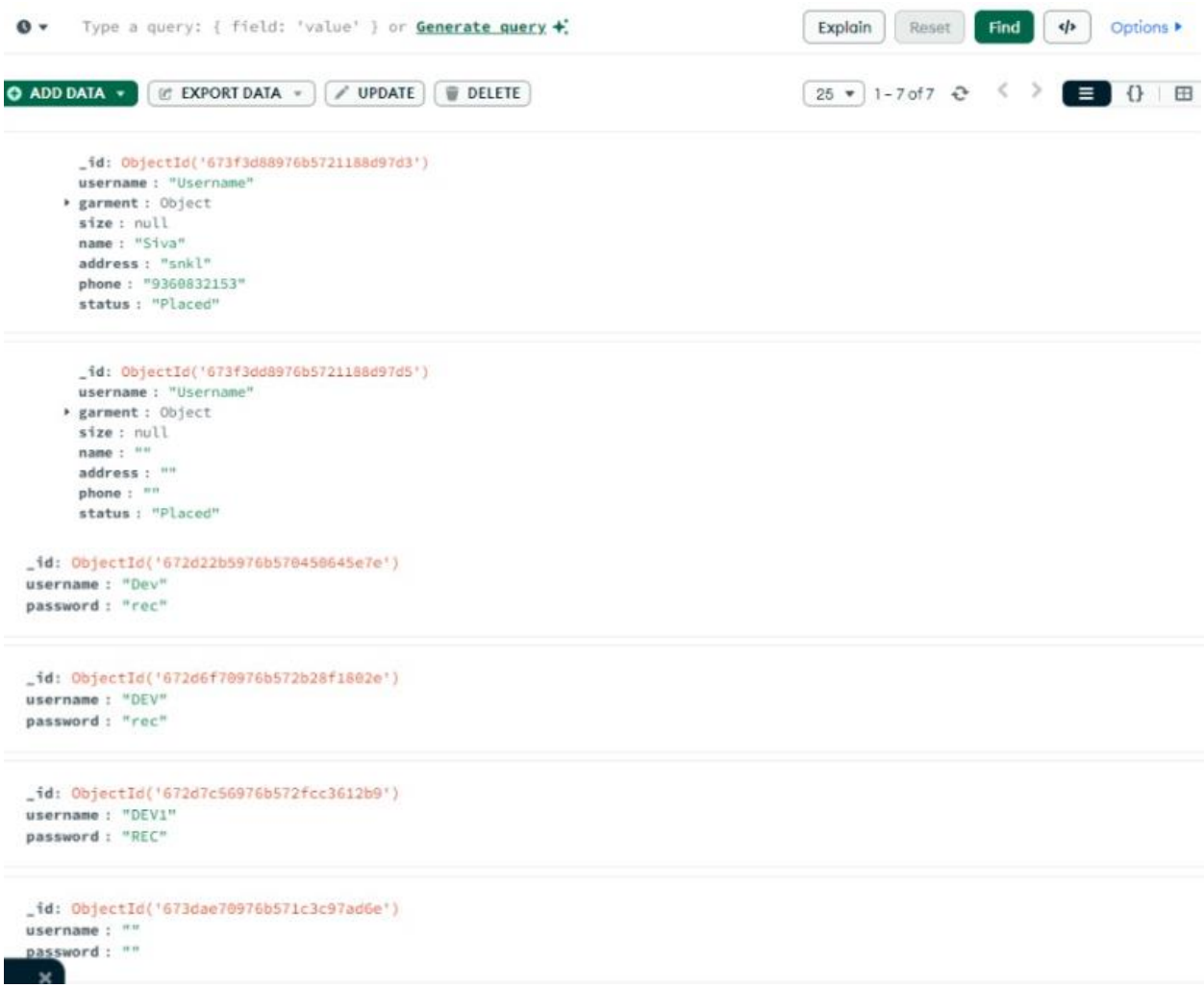


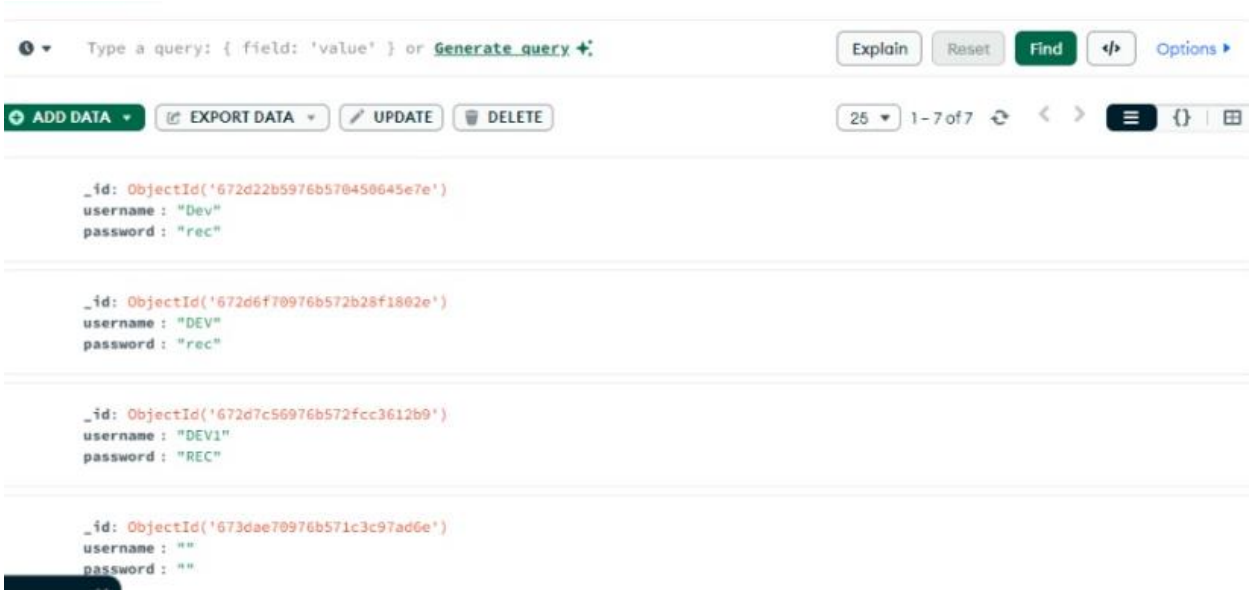

Shoppie
T-Shirt
$19.99
Size:
S M L XL
Jeans
$39.99
Size:
S M L XL
Added to Cart
Item added to cart. Would you like to buy it now?
Yes
No
Jacket
$59.99
Size:
S M L XL
Sneakers
$49.99
Size:
S M L XL
Hat
$15.99
Size:
S M L XL

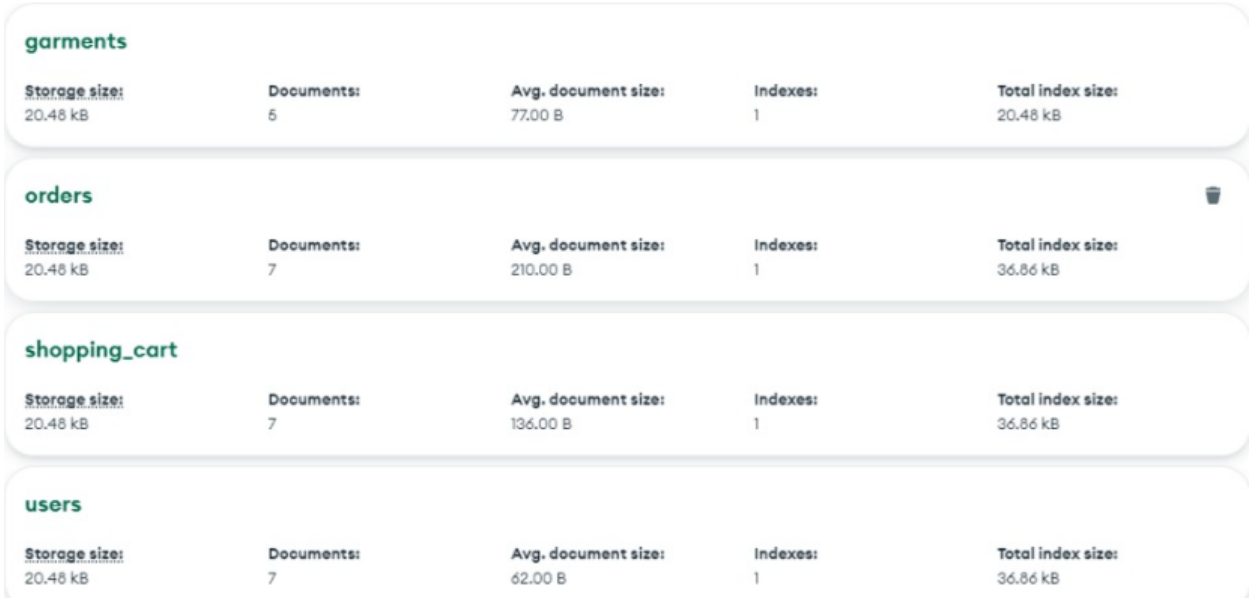

Back
Your Cart
T-Shirt
$19.99
Size: S
Buy Now
Remove
Jeans
$39.99
Size: M
Buy Now
Remove
Jacket
$59.99
Buy Now
Remove
Total: $195.95
Checkout

3.4 Sample Databases:










```
# Update user profile in the database
    users_collection.update_one(
        {"_id": ObjectId(user_id)},
        {
            "$set": {
                "first_name": first_name,
                "last_name": last_name,
                "email": email,
                "mobile": mobile,"dob": dob,
                "age": age, "address": address,
                "profile_photo": prfile_photo_data
            }
        }
```

```
# MongoDB connection
client = MongoClient('mongodb://localhost:27017/')
db = client['user_identification_system']
users_collection = db['users']
notes_collection = db['notes']
```

CHAPTER-4

KEY FEATURES

User Registration

- **Endpoint**: /signup
- Users register with personal details such as name, email, mobile number, date of birth, address, and an optional profile picture.
- Passwords are securely hashed using werkzeug.security.generate_password_hash.
- Profile pictures (if provided) are encoded in Base64 and stored in the database.

User Login

- **Endpoint**: /login
- Users authenticate with email and password. Upon successful login, users are redirected to the home page, and their user ID is stored in a session.
- Passwords are verified using check_password_hash.

Profile Completion Calculation

- Tracks and calculates the percentage of completed profile fields, including name, email, mobile, date of birth, and address.
- Displays this profile completion percentage to users on their profile or notes page.

Notes Feature

- **Endpoint**: /notes
- Authenticated users can create, save, retrieve, and edit notes.
- Notes are stored in the MongoDB collection and displayed along with profile completion status.

Profile Update

- **Endpoint**: /update_profile
- Users can update profile details, including uploading a new profile picture.

- Age is recalculated based on the updated date of birth.

QR Code Login

- **Endpoint**: /generate_qr_code/<user_id>
- Generates a unique QR code for users, which can be scanned for login without entering credentials.
- QR code redirects to /qr_login for session validation and access.

Password Reset

- **Endpoints**: /forgot_password, /reset_password/<token>
- Users can request a password reset link via email.
- The link includes a token for verification and allows users to reset their password securely.

Documentation Page

- **Endpoint**: /documentation
- Serves as a static page that provides instructions or an overview of how to use the application.

CHAPTER-5

CONCLUSION

The Online Garment Shopping System offers a well-rounded, secure, and user-friendly platform for seamless shopping experiences. It includes essential features such as user registration, login, and profile management, ensuring that users can easily create accounts, authenticate, and maintain their personal information. Security is a key focus, with hashed passwords for safe storage and QR code login options for quick access without needing to enter credentials.

The system tracks profile completion, providing users with real-time feedback on how much of their profile is filled, and encourages them to keep their information up-to-date. The notes feature allows users to store important information, while profile updates enable them to upload a new profile picture and update personal details. Users can also reset their passwords securely using a token-based system, providing an added layer of security and convenience.

Additionally, the documentation page offers users a detailed guide to navigate and maximize the platform's features. With its scalable backend, secure password management, and intuitive interface, the Online Garment Shopping System ensures a smooth and personalized shopping experience. The system’s robust security and thoughtful features make it a reliable and effective solution for users seeking an efficient and secure online shopping experience.

CHAPTER- 6

REFRENCES

1. **Flask Documentation**

   Flask is a lightweight Python web framework that powers the backend of the online garment shopping system. Its official documentation provides comprehensive details on routing, handling HTTP requests, and building RESTful APIs.
   Flask Documentation

2. **WerkzeugDocumentation**

   Werkzeug is a utility library for Python that handles secure password hashing, among other tasks. This library is crucial for ensuring that passwords are stored safely in the system.
   Werkzeug Documentation

3. **MongoDB Documentation**

   MongoDB is the NoSQL database used to store user data, notes, and other system-related information. Its flexible structure supports scalability and efficient data management.
   MongoDB Documentation

4. **SpringBootDocumentation**(Javaalternative)

   For those using Java, Spring Boot offers an alternative framework to Flask for building backend services. It simplifies application development and configuration.
   Spring Boot Documentation

5. **ReactDocumentation**

   React is used for the frontend development of the system, enabling dynamic and interactive user interfaces. Its component-based structure makes it ideal for building responsive websites.
   React Documentation

6. **PostgreSQL Documentation**

PostgreSQL is an alternative relational database management system to MongoDB. Its documentation provides valuable guidance for setting up and managing relational databases.
PostgreSQL Documentation

7. **JWT Authentication in Flask**

This tutorial explains how to implement JSON Web Tokens (JWT) for token-based user authentication in a Flask application, ensuring secure login and session management.
JWT Authentication in Flask

8. **SSL/TLS Best Practices**

For securing data transmission, SSL/TLS certificates are used to encrypt communication between users and the server. This reference offers best practices for implementation.
SSL/TLS Best Practices