PROJECT REPORT ON

E-COMMERCE WEBSITE

PROJECT TITLE	:	**TWRIL AND TREND : AN ONLINE CLOTHING STORE DEVELOPED WITH DJANGO AND MONGODB**
SUBMITTED BY	:	**HARSHINE S S**
REGISTER NO	:	**TWHOO23**
SUBMITTED TO	:	**T4TEQ SOFTWARE SOLUTIONS**
DATE OF SUBMISSION	:	**06/06/2025**
DURATION	:	**6 WEEKS**
DOMAIN	:	**FULL STACK WEB DEVELOPMENT USING PYTHON**

TABLE OF CONTENTS

ABSTRACT

Twirl and Trend is a dynamic e-commerce web application developed using Django and MongoDB, designed to provide a seamless shopping experience for users looking to purchase clothing for men, women, and children. The platform supports essential online shopping features such as user registration, product browsing, cart management, and secure checkout, all within a clean and responsive interface.

The use of MongoDB as the back-end allows for flexible, scalable data storage, ideal for handling various product categories and user interactions. The system is modular in design, with separate components for user management, cart system, checkout process, and admin controls, ensuring better maintainability and extensibility.

This project serves as a practical example of integrating modern web technologies to develop a complete and efficient e-commerce solution, meeting the needs of both customers and administrators.

CHAPTER 1

INTRODUCTION

In today's digital era, online shopping has become an essential part of people's lives. With the increasing reliance on the internet and smartphones, e-commerce platforms have seen rapid growth and have become the preferred mode of shopping for many customers. Clothing and fashion, in particular, are among the most in-demand sectors in online retail.

Twirl and Trend is an e-commerce web application built using Django and MongoDB, designed to simplify and enhance the shopping experience for users. It offers a variety of clothing collections for men, women, and children, providing a responsive and organized platform for browsing and purchasing fashion products.

This project demonstrates the integration of frontend design, backend logic, and database management to create a functional, user-friendly online store.

1.1. BACKGROUND AND OBJECTIVES

With the rise of online shopping in fashion, Twirl and Trend offers a simple, responsive platform for selling clothes. Built using Django and MongoDB, it allows users to browse products, manage carts, and simulate purchases, while providing admin tools for product management.

1.2. SCOPE OF THE PROJECT

This project focuses on creating a basic yet functional clothing store website. It includes core features like user registration, product browsing by category, cart handling, and a checkout system. An admin panel allows for managing products and user orders. Advanced features like payment gateways, shipping integration, and reviews are not included in this version but can be added later

1.3. OUTLINE OF THE PROJECT

This project is organized into key sections detailing its design and functionality. It begins with an overview of the system and main modules such as user, cart, checkout, and admin. The report also covers the database structure, implementation with Django and MongoDB, testing, screenshots, limitations, and future enhancements.

CHAPTER 2

PROBLEM STATEMENT

2.1. CURRENT ISSUE

Many existing e-commerce platforms are either too complex or require significant computing resources, making them inaccessible for small businesses or individuals learning to build such systems. These platforms often involve steep learning curves or high costs, which discourage new entrants. Furthermore, traditional relational databases used by many solutions lack the flexibility to handle rapidly changing product information, which is crucial in dynamic markets like fashion. As a result, scalability and quick updates become challenging. This gap highlights the need for a simpler, more adaptable platform that supports easy customization and fast deployment.

2.2. LIMITATIONS OF EXISTING SOLUTIONS

Current e-commerce platforms often adopt a generic approach that may not fit the unique requirements of niche markets, such as fashion clothing, where frequent product changes and style variations are common. Many platforms are slow due to heavy back-end processes or complicated interfaces that frustrate users. In addition, several lack smooth multi-item cart functionality, which is essential for a seamless shopping experience. The absence of strong admin tools to efficiently manage product catalogs further restricts their usability. These shortcomings motivate the development of a more specialized and user-friendly solution.

2.3. PROJECT GOALS

This project is designed to address these challenges by building a lightweight and easy-to-use e-commerce platform specifically tailored for fashion products. It aims to provide a smooth and intuitive shopping experience with reliable cart and checkout processes that handle multiple items effortlessly. Admin functionalities will allow for straightforward product management, making the system practical for small businesses. Leveraging MongoDB ensures flexible and scalable data storage, accommodating frequent product updates without disrupting operations. Overall, the project seeks to combine usability, flexibility, and efficiency in one cohesive system.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1. FUNCTIONAL REQUIREMENTS

- User Signup and Login functionality for account creation and authentication.
- Display products with essential details like name, price, and image.
- Add to Cart feature to store selected items.
- View, update, or remove items from the cart.
- Checkout system to simulate purchase and save order data.
- Display Purchase History for users.
- Admin functionality to add, edit, and delete products.

3.2. NON-FUNCTIONAL REQUIREMENTS

- **Responsive UI:** Fast-loading pages and mobile-friendly layout.
- **User-Friendly Navigation:** Simple and intuitive interface for all user actions.
- **Security:** Input validations and secure handling of login credentials.
- **Scalability:** Ability to handle increasing data and user load.
- **Reliability:** The system should run consistently without crashing.

3.3. SOFTWARE REQUIREMENTS

- **Programming Language:** Python 3.x
- **Web Framework:** Django 4.x
- **Database:** MongoDB (Atlas or Local)
- **Database Driver:** PyMongo
- **Frontend:** HTML, CSS, Bootstrap for UI design

3.4. HARDWARE REQUIREMENTS

- A computer or laptop with internet access
- Minimum **4 GB RAM** (Recommended for smooth development/testing)
- Any modern web browser (Chrome, Firefox, etc.
- MongoDB environment (either cloud-based Atlas or local server)

CHAPTER 4
SYSTEM ARCHITECTURE

4.1 . ARCHITECTURE DIAGRAM

The system architecture follows a multi-tier client-server design. The user interacts with the frontend interface, which communicates with Django-based backend controllers. These controllers handle business logic and interact with MongoDB to retrieve or store data. This architecture supports modularity, scalability, and separation of concerns.

4.2. DESCRIPTION

User requests such as logging in, adding items to the cart, or viewing products are sent from the frontend to Django views. The views process these requests, querying MongoDB collections for product data or user information, then prepare and send responses back to the client. This flow ensures efficient data handling and smooth user interaction.

CHAPTER 5
MODULE DESCRIPTION

5.1. USER MODULE

This module handles user registration and login. It uses MongoDB to store and retrieve user credentials securely. The global variable current_user is used to track login status instead of session management. Upon successful signup or signin, users are redirected to the homepage, and unauthorized access attempts to cart or purchase pages redirect users to the signin page.

5.2. PRODUCT MODULE

The product module includes category-specific views for Men, Women, and Kids. Each view handles product display with form-based input for size and quantity. When a product is added to the cart, its details (name, price, size, quantity) are pushed to the user's `cart` array in MongoDB. These templates and views offer tailored shopping experiences per category.

5.3. CART MODULE

The cart view retrieves products added by the user and allows dynamic updates such as quantity adjustments and removal. You also included logic for calculating total cost. This module supports two paths: (1) remove item from cart and (2) trigger checkout. It uses the `cart` array under the user's document to reflect live changes.

5.4. CHECKOUT MODULE

Checkout consolidates all items in the cart, moves them to the product field (representing purchased products), and clears the cart. The project includes both checkout (bulk purchase) and buy_now (single item purchase) features. Each completed purchase includes purchase time and item details.

5.5. PURCHASE HISTORY MODULE

This module displays a list of all previously purchased products. It queries the product array in the user's document and renders this data in a dedicated purchased.html template. Each item includes details like name, price, size, quantity, and optionally, purchase time if recorded.

5.6. CONTACT MODULE

This module allows logged-in users to submit contact details such as name, email, subject, and message. These are stored under the user's contact array in MongoDB. If the user is not logged in, the contact form submission redirects them to the signin page, ensuring secure feedback collection.

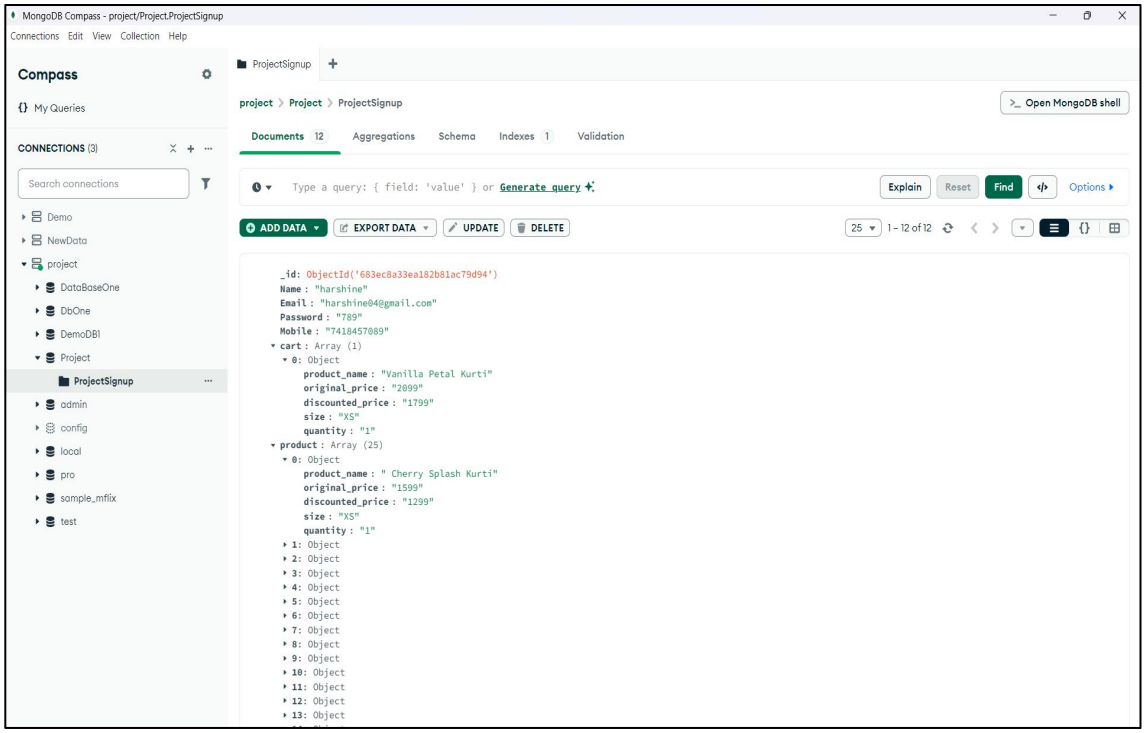
CHAPTER 6

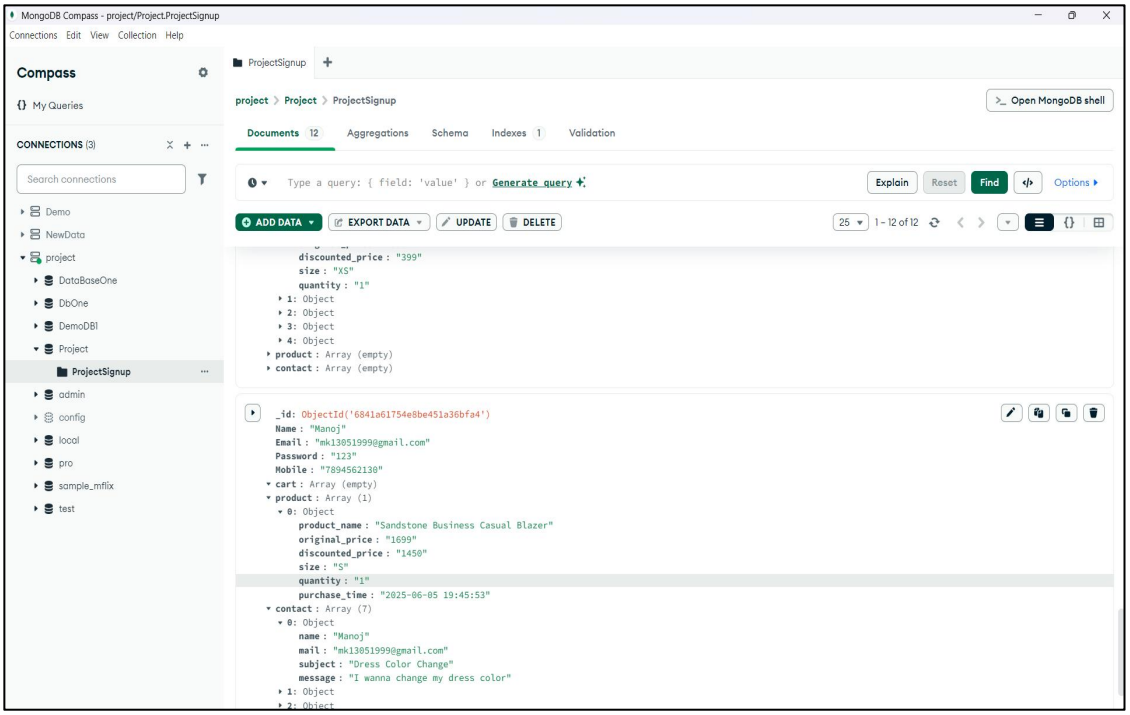
DATABASE DESIGN

The project uses **MongoDB**, a NoSQL document-oriented database, known for its flexibility and scalability. It stores data in JSON-like documents, allowing easy integration of nested objects and arrays — ideal for user carts, product details, and purchase history.The project mainly uses a single collection named ProjectSignup, where all user-related data is stored. Each document includes fields such as:

- Name, Email, Password, Mobile
- cart: list of cart items
- product: list of purchased items
- contact: contact form submissions

Each field stores structured data that MongoDB can query efficiently.




MongoDB Compass - project/Project.ProjectSignup
Connections Edit View Collection Help
Compass
My Queries
CONNECTIONS (3)
Search connections
Demo
NewData
project
DataBaseOne
DbOne
DemoDB1
Project
ProjectSignup
admin
config
local
pro
sample_mflix
test
ProjectSignup
project > Project > ProjectSignup
Open MongoDB shell
Documents 12
Aggregations
Schema
Indexes 1
Validation
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
UPDATE
DELETE
25
1 - 12 of 12
discounted_price : "399"
size : "XS"
quantity : "1"
1: Object
2: Object
3: Object
4: Object
product : Array (empty)
contact : Array (empty)
_id: ObjectId('6841a61754e8be451a36bfa4')
Name : "Manoj"
Email : "mk13051999@gmail.com"
Password : "123"
Mobile : "7894562130"
cart : Array (empty)
product : Array (1)
0: Object
product_name : "Sandstone Business Casual Blazer"
original_price : "1699"
discounted_price : "1450"
size : "S"
quantity : "1"
purchase_time : "2025-06-05 19:45:53"
contact : Array (7)
0: Object
name : "Manoj"
mail : "mk13051999@gmail.com"
subject : "Dress Color Change"
message : "I wanna change my dress color"
1: Object
2: Object

CHAPTER 7

IMPLEMENTATION

7.1. TECHNOLOGIES USED

The project utilizes several core technologies to deliver a functional and responsive e-commerce platform:

- **Django (Python Framework):** Powers the backend logic, URL routing, and view handling. It provides a clean and structured way to build dynamic web applications.
- **HTML, CSS, Bootstrap:** These are used to design the frontend interface. HTML forms the layout, CSS handles styling, and Bootstrap ensures responsiveness across various devices.
- **PyMongo:** Acts as a bridge between Django and MongoDB. It is a Python library that allows for CRUD operations (Create, Read, Update, Delete) directly on MongoDB documents.
- **MongoDB Atlas:** A cloud-hosted database that stores user details, cart items, purchases, and contact information in a NoSQL format.

7.2. PROJECT DIRECTORY STRUCTURE

The directory structure is organized logically to separate concerns and maintain clarity:

- **Templates Folder:** Contains HTML files such as signup.html, signin.html, index.html, womens.html, etc. These templates define the structure and content displayed to users.
- **Static Folder:** Stores all static assets like CSS stylesheets, JavaScript files, and product images to enhance the visual design and interactivity.
- **Views (Python Files):** The views.py file handles request processing, logic for sign in/signup, cart operations, and checkout functionality.
- **URL Configuration:** URLs are mapped to appropriate view functions using Django's URL dispatcher, maintaining navigation flow across the application.
- **MongoDB Integration:** Connections and queries are written inside the views using PyMongo, directly interacting with the MongoDB collection.

7.3. IMPORTANT CODE SNIPPETS

Key functional areas of the project are managed through important code snippets:

- **User Authentication:** The signup() and signin() functions validate user credentials, create new accounts, and allow login with proper verification.
- **Cart Management:** Functions like addtocart(), womenspage(), menspage(), and kidspage() allow users to add or remove products from their cart, including dynamic quantity updates.
- **Checkout Logic:** The checkout() and buy_now() views finalize purchases by moving items from the cart to the purchase history and clearing the cart afterward.
- **Contact Form:** The contactpage() function saves submitted user queries to the database under the contact field of the logged-in user document.

These implementations together make the project a functional, interactive shopping platform with backend integration and database operations.

CHAPTER 8

TESTING

8.1. MANUAL TESTING

Manual testing plays a key role in verifying that the application functions as expected from a user's perspective. Each module, such as signup, login, cart management, and checkout, is tested by simulating real user interactions. This includes entering invalid and valid inputs, navigating between pages, submitting forms, and triggering different functionalities. The system's behavior is observed to ensure appropriate responses are returned — like error messages for invalid login attempts or success alerts upon purchases. Browser compatibility and responsiveness are also checked on different screen sizes.

Manual Testing Covers:

- Signup and login validations (duplicate usernames, empty fields)
- Cart operations (add, update, remove items
- Checkout and buy-now functionality
- Form submissions in the contact page
- Purchase history retrieval

8.2. TEST CASES

Test cases are written to ensure that each feature works correctly in isolation and within the overall system flow. These test cases include a test case ID, the feature/module under test, a brief description, inputs used, expected output, and the actual result. This structured approach helps identify and fix bugs, and ensures that modifications don't break existing functionalities.

Test Case ID	Module	Description	Input	Expected Output	Actual Output
TC_01	Signup	Register with unique username	Valid user info	Redirect to index.html	Passed
TC_02	Signup	Register with duplicate username	Existing username	Showerror: "Username already exists"	Passed

TC_03	Login	Login with valid credentials	Correct username and password	Redirect to homepage	Passed
TC_04	Cart	Add product to cart	Click "Add to Cart" on product	Item added and visible in cart	Passed
TC_05	Checkout	Complete checkout	Click checkout with items in cart	Purchase successful message	Passed
TC_06	Contact Form	Submit contact details	Name, email, message	Show thank-you message	Passed

CHAPTER 9
CONCLUSION

9.1.CONCLUSION

The proposed e-commerce web application, *Twirl and Trend*, offers a simplified and user-friendly shopping experience for fashion products across men, women, and kids categories. It integrates Django with MongoDB to deliver seamless features like user signup/login, dynamic cart management, secure checkout, and purchase history tracking. The system's design prioritizes clarity and ease of use, making it accessible for both end-users and developers seeking lightweight, scalable solutions.

9.2. LIMITATION OF PROPOSED SYSTEM:

Although the system covers core e-commerce functionalities, it has some limitations. Currently, it lacks advanced features such as payment gateway integration, session-based login management, product search suggestions, and real-time inventory updates. Admin functionalities are limited and require further enhancement for product control and user management. Additionally, since MongoDB is used without a relational structure, handling complex data relationships may require additional effort.

9.3. FUTURE SCOPE:

Future improvements can include implementing session-based authentication for better security, integrating secure payment gateways, and adding a dedicated admin dashboard for managing products and user queries. Features like product ratings, reviews, wishlists, and order tracking could be added to enrich user experience. Additionally, incorporating analytics to monitor sales trends and user behavior will help scale the system for commercial use.

APPENDICES

A.1. SOURCE CODE

// views.py

```
from django.shortcuts import render, redirect

from datetime import datetime

# Create your views here.

import pymongo

from pymongo import MongoClient

from bson.objectid import ObjectId

url = "mongodb+srv://harshine:10152004@clusterfirst.xkuitu2.mongodb.net/"

client = MongoClient(url)

#  database and collection

db = client['Project']  # Database Name

collection = db['ProjectSignup']  # Collection Name

current_user = None

def signup(request):

    global current_user

    if request.method == 'POST' and request.POST.get('btn') == 'signup':

        uname = request.POST.get('uname')

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

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

        mob = request.POST.get('phnum')

        cart = []

        product_purchased = []

        contact = []
```

```
        existing_user = collection.find_one({'Name': uname})
        if existing_user:
            return render(request, 'signup.html', {'error': 'Username already exists'})
userdetails = {
            "Name": uname,
            "Email": email,
            "Password": password,
            "Mobile": mob,
            "cart": cart,
            "product": product_purchased,
            "contact": contact,
        }
        collection.insert_one(userdetails)
        current_user = uname
        return render(request, 'index.html')
    return render(request, 'signup.html')


def signin(request):
    global current_user
    if request.method == 'POST' and request.POST.get('btn') == 'signin':
        uname = request.POST.get('uname')
        password = request.POST.get('pass')
        user = collection.find_one({'Name': uname, 'Password': password})
        if user:
            current_user = uname
            return render(request, 'index.html')
```

```
        else:
            return render(request, 'signin.html', {'error': 'Login Failed, Invalid username or password'})
    return render(request, 'signin.html')
def indexpage(request):
    return render(request, 'index.html')
def womenspage(request):
    global current_user
    if request.method == 'POST':
        if not current_user:
            return render(request, 'signin.html', {'error': 'Please login first'})
        pn = request.POST['addtocart']
        product_details = {
            'product_name': request.POST.get('product_name'),
            'original_price': request.POST.get('original_price'),
            'discounted_price': request.POST.get('discounted_price'),
            'size': request.POST.get('size'),
            'quantity': request.POST.get('quantity'),
        }
        collection.update_one({'Name': current_user},{'$push': {'cart': product_details}})
        return render(request, 'womens.html', {'message': 'Product added to cart!'})
    return render(request, 'womens.html')
def menspage(request):
    global current_user
    if request.method == 'POST':
        if not current_user:
            return render(request, 'signin.html', {'error': 'Please login first'})
```

```
        pn = request.POST['addtocart']
        product_details = {
            'product_name': request.POST.get('product_name'),
            'original_price': request.POST.get('original_price'),
            'discounted_price': request.POST.get('discounted_price'),
            'size': request.POST.get('size'),
            'quantity': request.POST.get('quantity'),
        }
        collection.update_one({'Name': current_user},{'$push': {'cart': product_details}} )
        return render(request, 'mens.html', {'message': 'Product added to cart!'})
    return render(request, 'mens.html')
def kidspage(request):
    global current_user
    if request.method == 'POST':
        if not current_user:
            return render(request, 'signin.html', {'error': 'Please login first'})
        pn = request.POST['addtocart']
        product_details = {
            'product_name': request.POST.get('product_name'),
            'original_price': request.POST.get('original_price'),
            'discounted_price': request.POST.get('discounted_price'),
            'size': request.POST.get('size'),
            'quantity': request.POST.get('quantity'),
        }
        collection.update_one({'Name': current_user},{'$push': {'cart': product_details}} )
        return render(request, 'kids.html', {'message': 'Product added to cart!'})
```

```
    return render(request, 'kids.html')
def addtocart(request):
    global current_user
    if not current_user:
        return render(request, 'signin.html', {'error': 'Please login first'})
    user = collection.find_one({'Name': current_user})
    cart_items = user.get('cart', [])
    message = ''
    total = sum(
        int(item['discounted_price']) * int(item['quantity'])
        for item in cart_items
        if item.get('discounted_price') and item.get('quantity')
    )
   if request.method == 'POST':
        if 'remove_index' in request.POST:
            try:
                remove_index = int(request.POST.get('remove_index'))
                if 0 <= remove_index < len(cart_items):
                    cart_items.pop(remove_index)
                    collection.update_one({'Name': current_user},{'$set': {'cart': cart_items}} )
                    message = 'Item removed successfully.'
                    total = sum(int(item['discounted_price']) * int(item['quantity'])
                      for item in cart_items
                    )
            except Exception as e:
                message = f'Error removing item: {str(e)}'
```

```
        elif 'buy_now' in request.POST:
            if cart_items:
                collection.update_one({'Name': current_user},{  '$push': {'product': {'$each':
                cart_items}},'$set': {'cart': []}})
                return render(request, 'purchase.html', {'purchased_items': cart_items,
                    'message': 'Thank you for your purchase!'
                })
            else:
                message = 'Cart is empty. Nothing to purchase.'
    return render(request, 'addtocart.html', {
        'cart_items': cart_items,
        'total': total,
        'message': message
    })
def checkout(request):
    global current_user
    if not current_user:
        return render(request, 'signin.html', {'error': 'Please login first'})
    user = collection.find_one({'Name': current_user})
    cart_items = user.get('cart', [])
    if cart_items:
        collection.update_one(
            {'Name': current_user},{ '$push': {'product': {'$each': cart_items}},'$set': {'cart': []}})
        message = "Thank You for Your Purchase! Your order has been successfully placed."
    else:
        message = "Your cart is empty."
```

```
    return render(request, 'addtocart.html', {
        'cart_items': [],
        'total': 0,
        'message': message
    })
def buy_now(request):
    global current_user
    if not current_user:
        return render(request, 'signin.html', {'error': 'Please login first'})
   user = collection.find_one({'Name': current_user})
cart_items = user.get('cart', [])
message = ''
   if request.method == 'POST':
        index_str = request.POST.get('buy_index')
        if index_str is not None and index_str.isdigit():
            buy_index = int(index_str)
            if 0 <= buy_index < len(cart_items):
                item_to_buy = cart_items.pop(buy_index)
                item_to_buy['purchase_time'] = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
                collection.update_one({'Name': current_user},{ '$push': {'product': item_to_buy}, '$set':
            {'cart': cart_items}})
                message = "Thank You for your purchase! Your order has been placed successfully."
            else:
                message = "Invalid item index."
        else:
            message = "No item selected to buy."
user = collection.find_one({'Name': current_user})
```

```
    cart_items = user.get('cart', [])
    total = sum(int(item['discounted_price']) * int(item['quantity']) for item in cart_items)
   return render(request, 'addtocart.html', {
        'cart_items': cart_items,
        'total': total,
        'message': message
    })
def purchased_products(request):
    global current_user
    if not current_user:
        return render(request, 'signin.html', {'error': 'Please login first'})
 user = collection.find_one({'Name': current_user})
    purchased_items = user.get('product', [])
    return render(request, 'purchased.html', {
        'purchased_items': purchased_items
    })


def aboutpage(request):
    return render(request, 'about.html')
def contactpage(request):
    global current_user
    if request.method == 'POST' and request.POST.get('contactus') == 'submit':
        if not current_user:
            return render(request, 'signin.html', {'error': 'Please login first'})
        contactdetails = {
            'name': request.POST.get('name'),
```

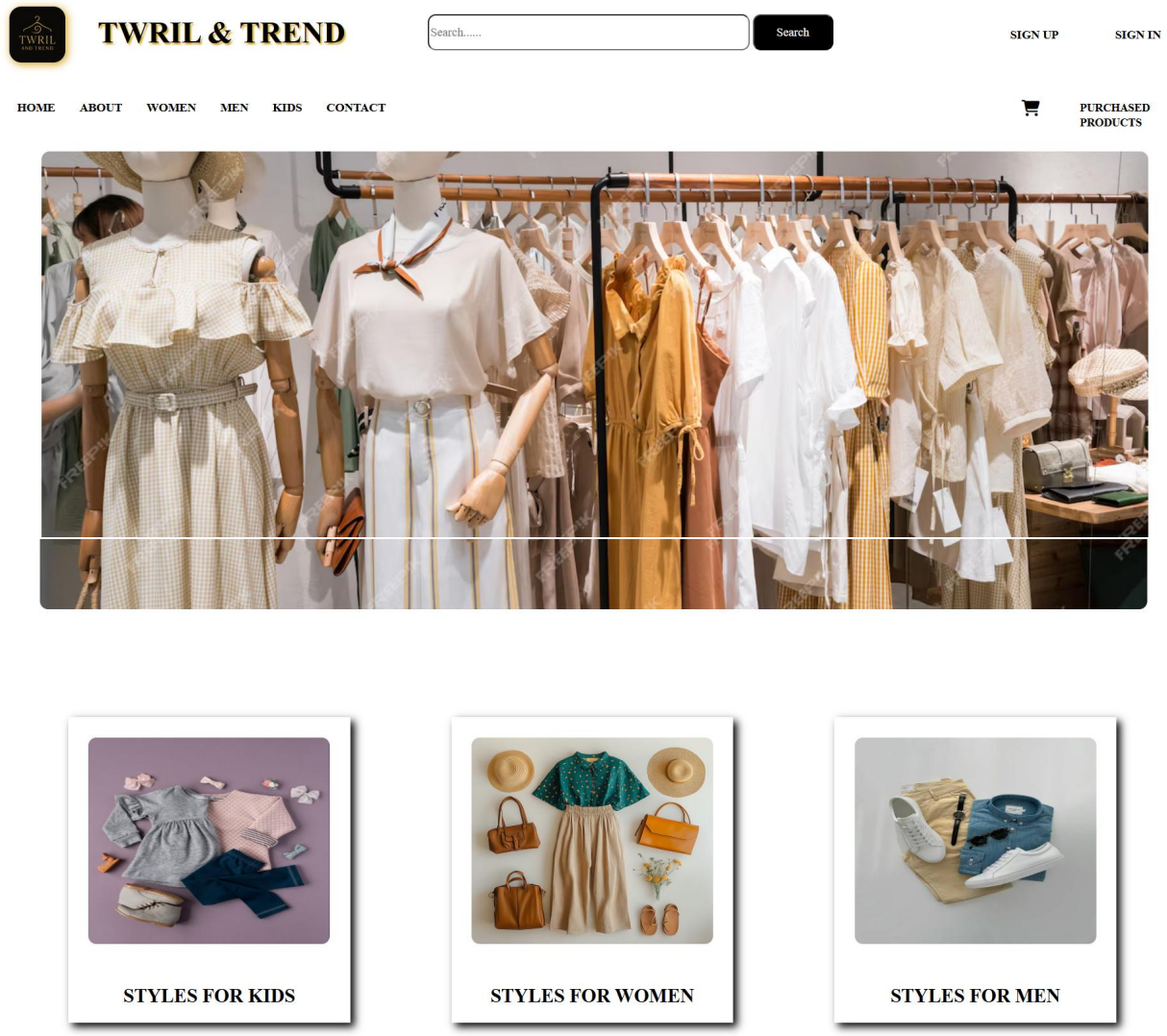
```
            'mail': request.POST.get('email'),
            'subject': request.POST.get('subject'),
            'message': request.POST.get('message')
        }
        collection.update_one({'Name': current_user},{'$push': {'contact': contactdetails}} )
        print("Form submitted successfully")
        return render(request, 'contact.html', {'message': 'Your contact details have been successfully added! Stay in touch!'})
    return render(request, 'contact.html')
```

// urlss.py

```
from django.contrib import admin
from django.urls import path
from . import views
fromAppOne.viewsimportindexpage,aboutpage,womenspage,menspage,kidspage,contactpage,signup,signin,addtocart
urlpatterns = [ path('signup/',views.signup, name='signup'),
   path('signin/',views.signin, name='signin'),
   path('indexpage/',views.indexpage,name='indexpage'),
   path('aboutpage/',views.aboutpage,name='aboutpage'),
   path('womenspage/',views.womenspage,name='womenspage'),
   path('menspage/',views.menspage,name='menspage'),
   path('kidspage/',views.kidspage,name='kidspage'),
   path('contactpage/',views.contactpage,name='contactpage'),
   path('addtocart/',views.addtocart, name='addtocart'),
   path('checkout/', views.checkout, name='checkout'),
   path('buy-now/', views.buy_now, name='buy_now'),
   path('purchased/', views.purchased_products, name='purchased_products'),]
```

A.2. SCREENSHOTS

//indexpage.html



//aboutpage.html

TWRIL & TREND

Search..... Search

SIGN UP SIGN IN

HOME ABOUT WOMEN MEN KIDS CONTACT

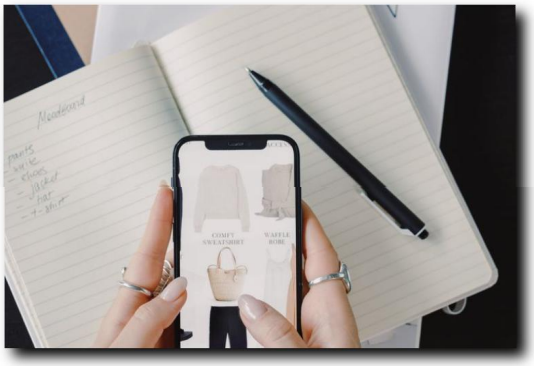
PURCHASED PRODUCTS

ABOUT US

Welcome to Twril and Trend, where fashion meets passion and every outfit tells your unique story. We believe that style is more than just clothes — it's an expression of your personality, confidence, and creativity. At Twril and Trend, we curate bold, beautiful, and trendsetting pieces that help you twirl with confidence and stand out in every crowd. Our team of fashion lovers is dedicated to bringing you the latest trends combined with timeless elegance. Whether you're dressing for everyday moments or special occasions, Twril and Trend is your partner in style.

OUR STORY

Twril and Trend was born out of a desire to empower individuals through fashion. What started as a small idea grew into a vibrant community that celebrates self-expression and individuality. We believe fashion is for everyone, and we strive to create styles that inspire you to be your authentic self — wherever you go, whatever you do.



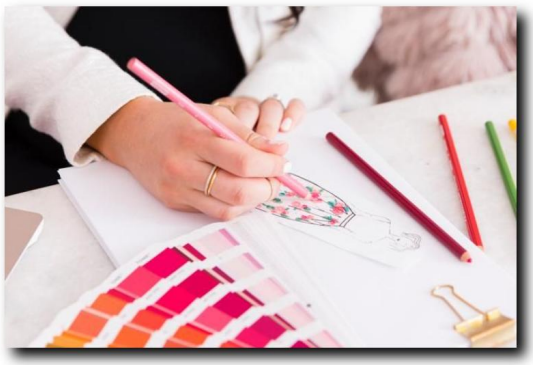
VISION AND MISSION

Our Vision: To inspire confidence and creativity in every person through accessible, stylish, and sustainable fashion.
Our Mission: To empower individuals to express their unique identity by offering trend-forward, high-quality clothing and accessories that combine style, comfort, and sustainability. Together, we aim to create a world where fashion is inclusive, inspiring, and a true reflection of who you are.

WHAT WE OFFER

At Twril and Trend, you'll find a curated selection of:

- Trend-forward apparel for every occasion
- Statement accessories to complete your look
- Seasonal collections that keep you ahead of the curve
- Style tips and inspiration to fuel your creativity
- We carefully select each piece to ensure quality, comfort, and timeless appeal.

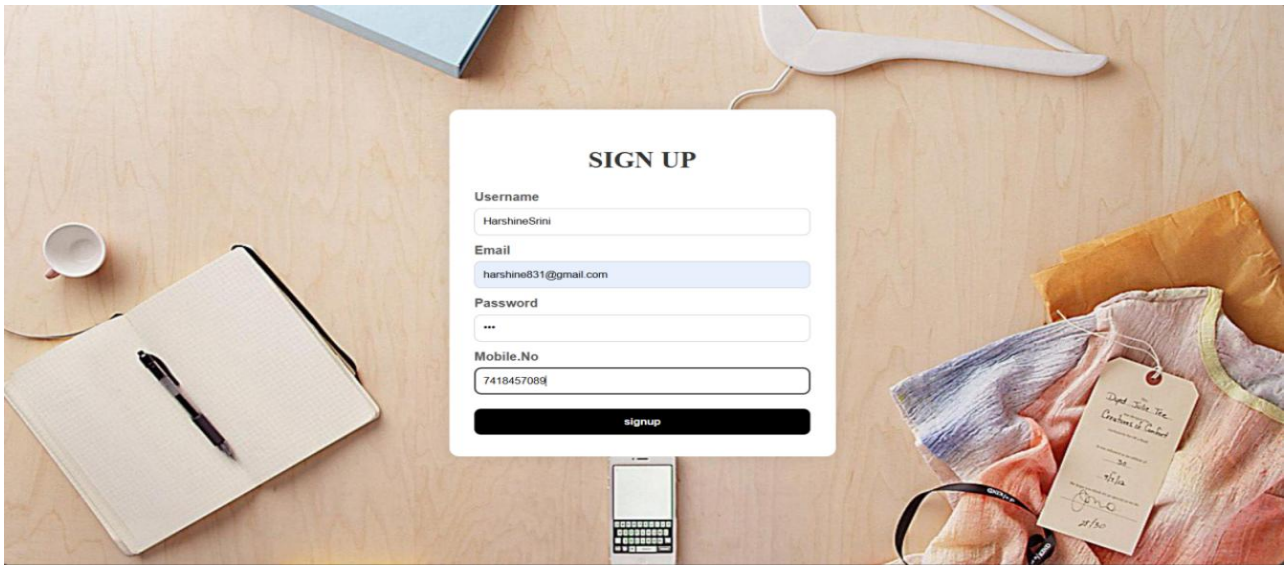


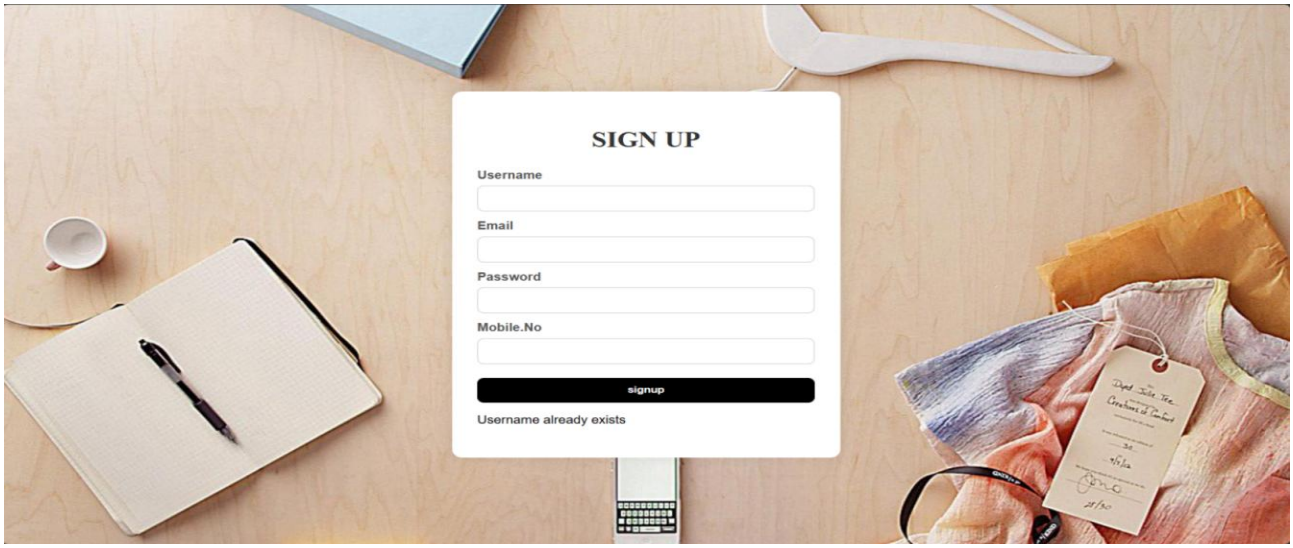
JOIN OUR COMMUNITY

More than just a brand, Twril and Trend is a vibrant community of fashion enthusiasts who love to share, inspire, and support each other. Follow us on social media, tag your looks with #TwrilAndTrend, and be part of a movement that celebrates diversity, confidence, and bold style.

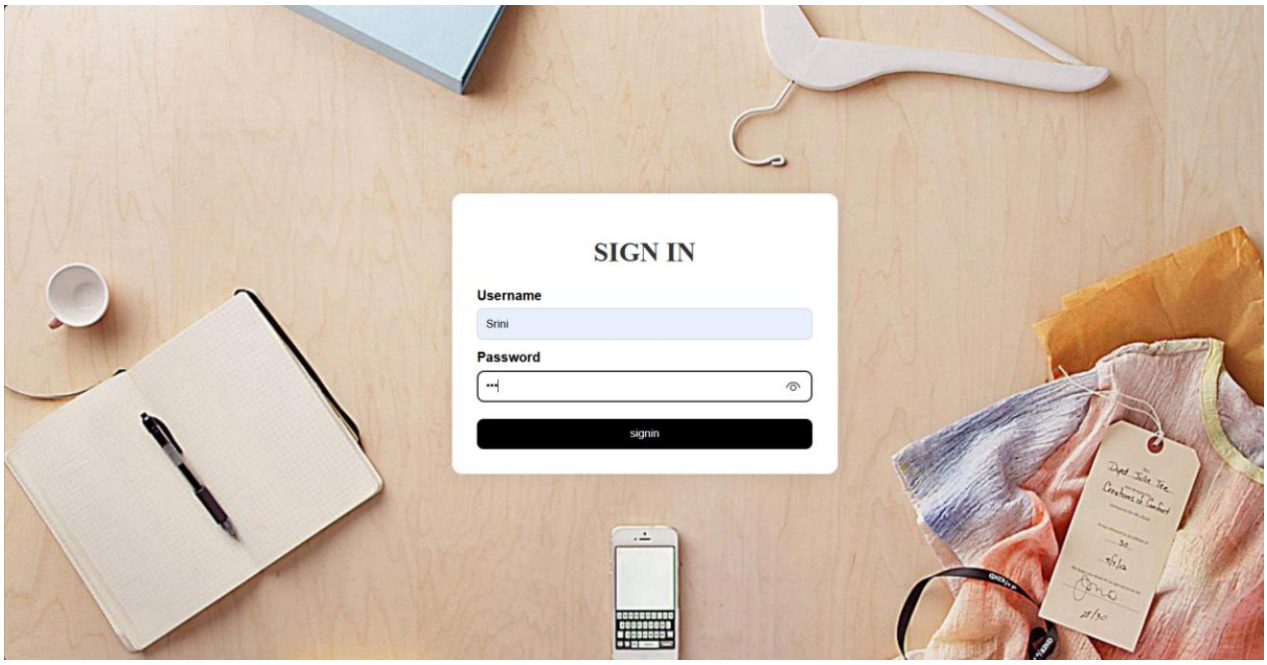
@ 2025 , Twril and Trend

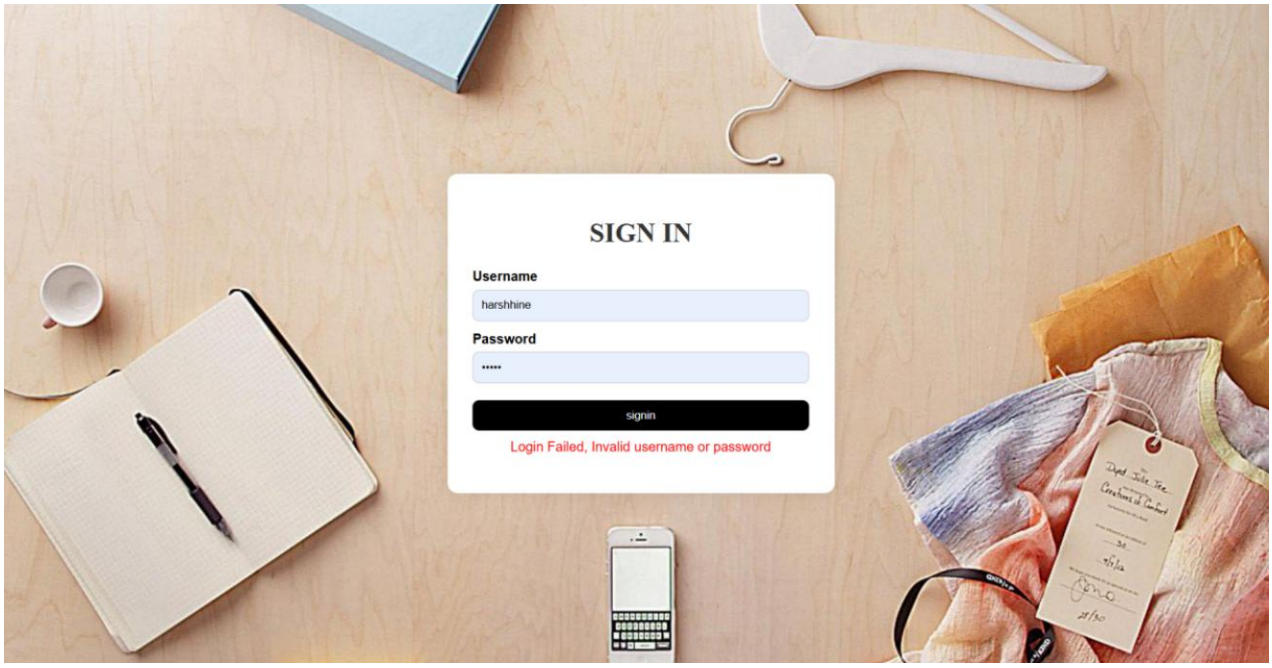
//signup.html





//signin.html


SIGN IN
Username
Srini
Password
signin


SIGN IN
Username
harshhine
Password
signin
Login Failed, Invalid username or password

//womens.html

KURTAS & SETS

SAREES

JEANS & TROUSERS

ETHNIC WEAR

LEHANGAS

TOPS & SHIRTS

JUMPSUITS

SKIRTS

FROKS & MAXIES

MATERNITY WEAR

CATEGORIES

- KURTAS & SETS
- SAREES
- JEANS & TROUSERS
- ETHNIC WEAR
- LEHANGAS
- TOPS & SHIRTS
- JUMPSUITS
- SKIRTS
- FROKS & MAXIES
- MATERNITY WEAR

KURTAS & SETS

Sunflower Dream Kurti

Yellow cotton kurti with floral prints, cheerful and summery charm.

Price : ~~Rs.1799/-~~ **Rs.1499/-**

Size : Select

Quantity : 1

ADD TO CART

Vanilla Petal Kurti

Soft white kurti with delicate design, ideal for sunny days.

Price : ~~Rs.2099/-~~ **Rs.1799/-**

Size : Select

Quantity : 1

ADD TO CART

Crimson Trail Kurti

White base kurti with red accents, bold and stylish fit.

Price : ~~Rs.1899/-~~ **Rs.1599/-**

Size : Select

Quantity : 1

ADD TO CART

Cherry Splash Kurti

Sleeveless red floral kurti, flowy and perfect for brunches.

Price : ~~Rs.1599/-~~ **Rs.1299/-**

Size : Select

Quantity : 1

ADD TO CART

//mens.html

CASUALS

FORMALS

JEANS & PANTS

SHORTS & TROUSERS

HOODIES

JACKETS

KURTA SETS

BLAZERS

DHOTIS & SHIRTS

T-SHIRTS

CATEGORIES

- CASUALS
- FORMALS
- JEANS & PANTS
- SHORTS & TROUSERS
- HOODIES
- JACKETS
- KURTAS
- BLAZERS
- DHOTIS & SHIRTS
- T-SHIRTS

CASUALS

Mint Corduroy Casual Shirt

Light green corduroy shirt perfect for cool everyday styling.

Price : ~~Rs.1799/-~~ **Rs.1499/-**

Size : Select

Quantity : 1

ADD TO CART

Skywashed Denim Look Shirt

Faded blue shirt with denim vibes and full-sleeve design.

Price : ~~Rs.1299/-~~ **Rs.1099/-**

Size : Select

Quantity : 1

ADD TO CART

Autumn Checks Wool Shirt

Cozy woolen shirt with brown checkered pattern, warm and stylish.

Price : ~~Rs.1099/-~~ **Rs.799/-**

Size : Select

Quantity : 1

ADD TO CART

Mocha Smart Buttoned Shirt

Brown collared shirt with chest pockets, ideal for winter wear.

Price : ~~Rs.1399/-~~ **Rs.1199/-**

Size : Select

Quantity : 1

ADD TO CART

//kids.html

BOYS T-SHIRTS

BOYS ETHNIC WEAR

BOYS SHIRTS

BOYS SHORTS

BOYS NIGHT DRESS

GIRLS TSHIRTS & TOPS

GIRLS ETHNIC WEAR

GIRLS DRESSES

GIRLS SKIRTS

GIRLS NIGHT DRESS

CATEGORIES

- BOYS TSHIRTS
- BOYS ETHNIC WEAR
- BOYS SHIRTS
- BOYS SHORTS
- BOYS NIGHT DRESS
- GIRLS TSHIRTS
- GIRLS ETHNIC WEAR
- GIRLS SKIRTS
- GIRLS DRESSES
- GIRLS NIGHT DRESS

BOYS TSHIRTS

Sage Splash Graphic Tee

Light green tee with modern splash art for relaxed wear.

Price : ~~Rs.599/-~~ **Rs.399/-**

Size : Select

Quantity : 1

ADD TO CART

Tropical Blaze Split Tee

Half black, half white with orange flamingo jungle vibes.

Price : ~~Rs.499/-~~ **Rs.399/-**

Size : Select

Quantity : 1

ADD TO CART

Coastal Summer Print Tee

Cream and blue beach elements make it perfect for holidays.

Price : ~~Rs.799/-~~ **Rs.699/-**

Size : Select

Quantity : 1

ADD TO CART

Abstract Earthy Street Tee

Beige with bold black abstract art for urban statement.

Price : ~~Rs.560/-~~ **Rs.450/-**

Size : Select

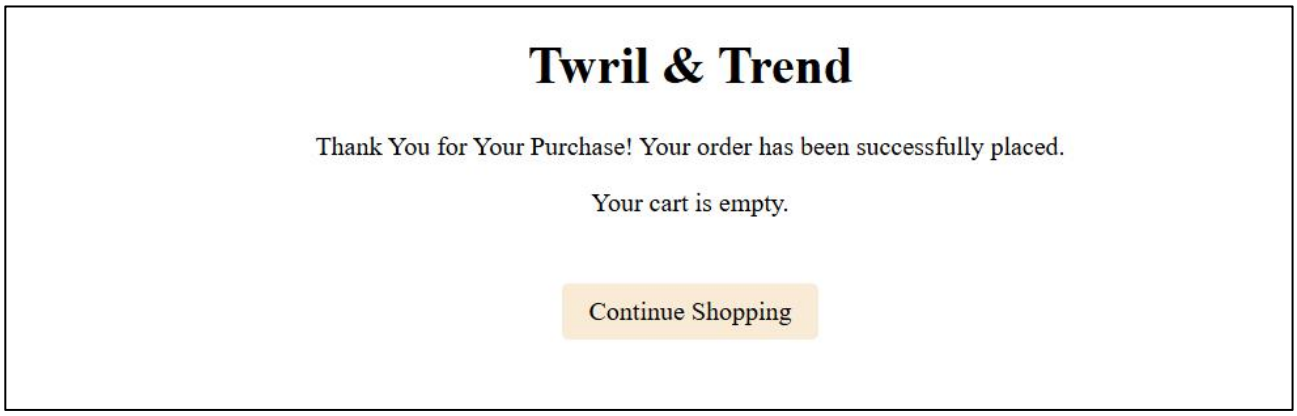
Quantity : 1

Coastal Summer Print Tee

ADD TO CART

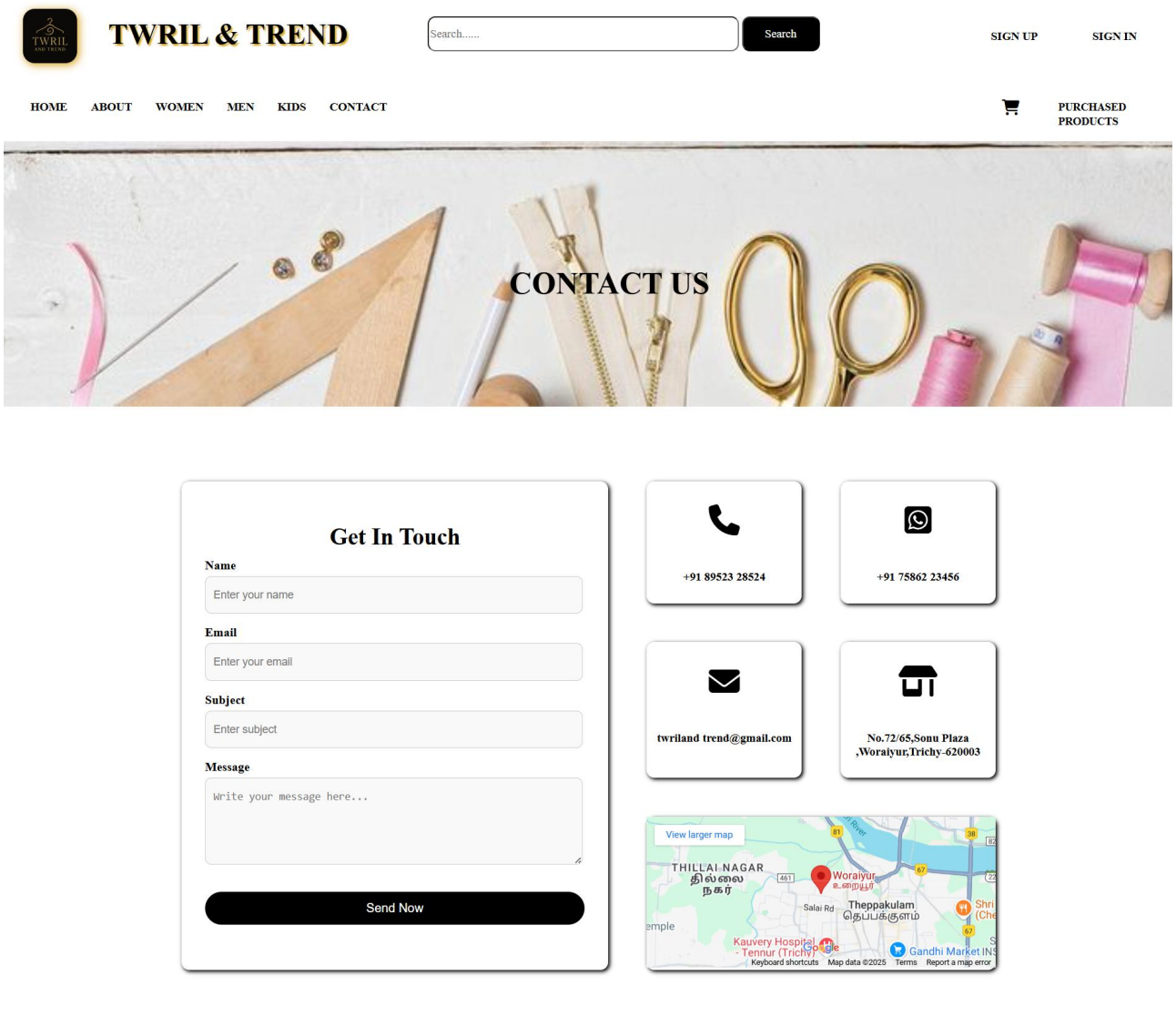
//addtocart.html

//checkout



//purchased.html

//contacts.html



@ 2025 , Twril and Trend