Green-Tech

A PROJECT REPORT
for
Project (KCA451)
Session (2024-25)

Submitted by

Kapil Maheshwari
(2300290140085)
Kavya Verma
(2300290140087)
Hritik Chaudhary
(2300290140077)
Prabal Pratap Singh
(2300290140119)

Submitted in partial fulfilment of the
Requirements for the Degree of

MASTER OF COMPUTER APPLICATION

Under the Supervision of
Dr. Akash Rajak
Dean MCA

Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206
(MAY 2025)

CERTIFICATE

Certified that **Hritik Chaudhary (2300290140077), Prabal Pratap Singh (2300290140119), Kapil Maheshwari (2300290140085), Kavya Verma (2300290140087)** have carried out the project work having "**Green-Tech**" (**Project-KCA451**) for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself, and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Dr. Akash Rajak
Dean & Professor
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

ABSTRACT

GreenTech is a dedicated web-based E-Commerce platform designed exclusively for the sale and purchase of solar products, aiming to support the global shift toward sustainable energy. The platform offers a centralized space where users can explore, compare, and purchase a wide range of solar solutions with ease and convenience. With a focus on simplicity, accessibility, and user satisfaction, GreenTech provides a clean and intuitive interface for a smooth online shopping experience.

GreenTech supports two main user roles: Customers and Admins, each with role-specific functionalities. Customers can browse categorized products such as solar panels, inverters, batteries, and solar-powered appliances, add items to their cart, and place orders through a simple and responsive interface. Admins have comprehensive control over the platform, including adding and updating product listings, managing user accounts, monitoring purchases, and maintaining overall system performance.

By eliminating complexity in accessing solar products, GreenTech encourages the adoption of renewable energy solutions. The project aims to provide a reliable, scalable, and user-friendly online platform tailored specifically for the solar industry, promoting a greener future through technology.

ACKNOWLEDGEMENT

Success in life is never attained single handedly. My deepest gratitude goes to my thesis supervisor, Dr. Akash Rajak, Professor & Dean, for their guidance, help and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions have guided me a lot in completing this project successfully.

Words are not enough to express my gratitude to Dr. Akash Rajak, Professor & Dean, Department of Computer Applications, for his insightful comments and administrative help on various occasions. Fortunately, I have many understanding friends, who have helped me a lot with many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kinds of help. Without their support, completion of this work would not have been possible in time. They keep mylife filled with enjoyment and happiness.

Kapil Maheshwari
(2300290140085)

Kavya Verma
(2300290140087)

Hritik Chaudhary
(2300290140077)

Prabal Pratap Singh
(230029014119)

TABLE OF CONTENTS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND OF THE STUDY

The demand for clean and renewable energy is increasing rapidly due to growing environmental concerns, depleting fossil fuels, and rising energy costs. Among all renewable sources, solar energy stands out as the most accessible and widely adopted solution. However, despite its benefits, the process of exploring, comparing, and purchasing solar products remains scattered and inconvenient for consumers.

GreenTech is a web-based e-commerce platform developed to address this gap by offering a centralized, specialized marketplace for solar products. The platform is designed to simplify the buying and selling of solar items such as solar panels, inverters, batteries, and solar-powered appliances. With an intuitive user interface and role-based access for both customers and administrators, GreenTech makes the solar shopping experience efficient, user-friendly, and scalable.

1.2 PROBLEM STATEMENT

In the current market, consumers interested in solar products often face various challenges. These include lack of product information, absence of specialized platforms, difficulty in comparing prices or specifications, and limited access to trustworthy vendors. Most existing e-commerce platforms are generalized and do not offer solar-specific features or support.

There is also a lack of centralized administration tools that can manage products, track user activities, and ensure smooth operation of a solar-focused platform. This creates a barrier to both consumers and sellers in the solar industry, slowing down adoption and growth.

GreenTech is designed to address these problems by offering a dedicated solution that brings solar products to a single platform with clear categorization, simplified shopping features, and administrative control.

1.3 OBJECTIVES

The primary objective of GreenTech is to provide a specialized, web-based e-commerce solution focused on solar products. The detailed objectives include:

- To design and develop a user-friendly platform for browsing and purchasing solar equipment.

- To implement a clean and responsive UI/UX suitable for all types of users.
- To allow administrators to manage products, user activities, and overall platform performance through a dedicated admin panel.
- To promote awareness and usage of solar energy solutions by making them more accessible online.
- To build a scalable platform that can be expanded with more products or features in the future.

1.4 SCOPE

The scope of GreenTech is centered around creating a functional and scalable e-commerce platform dedicated solely to solar products. The system will serve two types of users: Customers and Admins.

Customers will be able to:

- Register and log in
- Browse products by category
- Add products to cart and place orders

Admins will be able to:

- Add, update, or delete products
- Manage customer accounts
- Monitor platform activity and performance

The platform does not include secure online payment or real-time delivery tracking in its current version but can be extended to include such features in future updates. The project will be developed using modern web technologies and will follow best practices in usability and system design.

CHAPTER 2

LITERATURE SURVEY

The literature survey provides an overview of existing research, technologies, and platforms relevant to the domain of solar product e-commerce and web application development. This chapter helps establish the background, highlight the gaps in current systems, and justify the need for the proposed system, GreenTech.

2.1 EXISTING E-COMMERCE PLATFORMS

Several well-established e-commerce platforms like Amazon, Flipkart, and IndiaMART offer a variety of solar products, but they cater to a general audience and lack specialization in the renewable energy sector. These platforms do not provide detailed technical specifications or comparisons tailored specifically to solar energy products. Additionally, the user interface is not optimized for customers specifically searching for sustainable energy solutions, often leading to scattered or irrelevant search results.

2.2 SPECIALIZED SOLAR PRODUCT WEBSITES

Some niche websites such as Loom Solar, Tata Power Solar, and Usha Solar offer solar equipment exclusively. However, these websites are typically vendor-specific and do not support cross-brand comparisons, product reviews, or dynamic user interactions. The lack of a multi-vendor, centralized platform makes it difficult for consumers to explore a broader market or find the most suitable product based on their needs and budget.

2.3 RELATED ACADEMIC PROJECTS

Previous academic projects in the field of e-commerce have focused primarily on generic product categories and rarely target a specialized domain like solar energy. Many of them are developed using traditional web stacks (PHP, Java, etc.) and offer only basic functionality. Few projects take advantage of modern full-stack technologies like the MERN stack, which offers real-time performance, flexibility, and scalability for modern web applications.

2.4 TECHNOLOGY TRENDS IN E-COMMERCE

The evolution of web development has shifted toward full-stack JavaScript solutions like MERN (MongoDB, Express.js, React.js, Node.js), which streamline development and enable high responsiveness, better performance, and modular design. Modern e-commerce platforms emphasize responsive UI/UX, RESTful API integration, cloud deployment, and security features like JWT authentication, all of which are considered in the development of GreenTech.

2.5 IDENTIFIED GAPS

Based on the analysis of existing platforms and academic works, the following gaps have been identified:

- Lack of a dedicated platform focused entirely on solar products.
- Absence of multi-brand comparison features in existing solar vendor sites.
- Limited use of modern full-stack development tools in student projects.
- Underutilization of admin-driven platforms for solar product management.

2.6 JUSTIFICATION FOR GREENTECH

GreenTech addresses these gaps by offering a centralized, solar-focused e-commerce platform using the MERN stack. It combines a clean and intuitive frontend, robust backend, flexible database, and an admin panel that allows complete platform control. The goal is to provide users with a seamless experience for discovering and purchasing solar products while supporting future scalability and enhancement.

CHAPTER 3

FEASIBILITY STUDY

Before the development of GreenTech, a thorough feasibility study was conducted to evaluate the project's viability from various perspectives. The goal of this study is to determine whether the project is technically, economically, and operationally feasible within the scope and resources available to the development team.

3.1 ECONOMIC FEASIBILITY

Economic feasibility examines the cost-effectiveness of the project and its potential to generate value.

3.1.1 Development Costs**:** As a student project, GreenTech will be developed using open-source tools and technologies. Development costs are minimized as the project team will handle design, coding, and testing. No commercial software licenses are required, and hosting can initially be done on free or low-cost platforms during development.

3.1.2 Operational Costs**:** Post-deployment, operational costs will include web hosting, domain registration, and maintenance. These costs are minimal in the initial stages and can be scaled as the platform grows. Admin activities such as product management will also be handled internally by the platform admin panel, reducing the need for additional staff.

3.1.3 Revenue Generation**:** Although revenue generation is not a primary goal during academic development, GreenTech has potential for monetization in the future. Possible revenue models include commission-based sales, vendor subscriptions, featured product listings, and advertising. These streams can support long-term sustainability if the platform is expanded commercially.

3.2 TECHNICAL FEASIBILITY

This aspect evaluates whether the required technology, tools, and technical expertise are available to successfully build the system.

3.2.1 Technology Stack**:** GreenTech is being developed using the **MERN stack,** a popular and powerful set of open-source technologie:

- MongoDB: NoSQL database used to store product, user, and order data in a flexible document format.
- Express.js: Lightweight and fast backend web application framework for handling APIs and routing.

- React.js: JavaScript library for building responsive and dynamic user interfaces.
- Node.js: JavaScript runtime used to build scalable server-side applications.

This stack provides full-stack JavaScript development, enabling faster development and smoother data flow between frontend and backend components.

3.2.2 Integration**:** The MERN stack ensures seamless integration between all layers of the application. RESTful APIs built using Express.js will connect the frontend React components with the MongoDB database via Node.js middleware. The admin panel and customer interfaces will both operate using the same integrated API structure.

3.2.3 Security**:** Security will be handled through user authentication, token-based authorization (e.g., JWT), and secure API endpoints. Input validation and protection against common web vulnerabilities such as XSS and CSRF will be implemented. The system is designed with best practices to ensure data confidentiality and integrity.

3.2.4 Cross-Device Compatibility**:** React's component-based architecture allows the frontend to be fully responsive and mobile-friendly. The platform will be tested across various devices to ensure consistent and optimal performance on desktops, tablets, and smartphones.

3.3 OPERATIONAL FEASIBILITY

Operational feasibility determines whether the platform can be adopted and used effectively by both users and administrators.

3.3.1 User Adoption**:** The platform is designed with a simple, intuitive interface, encouraging quick user adoption. Clear navigation, product categorization, and helpful UI elements make it easy even for non-technical users to explore and use the platform effectively.

3.3.2 Maintenance and Support**:** The admin panel will simplify backend operations such as updating product information, managing users, and resolving basic issues. Since the platform uses well-established technologies, maintenance is manageable and can be handled by any team member familiar with web development.

3.3.3 Scalability**:** GreenTech is built with scalability in mind. More product categories, vendors, and features like order tracking, reviews, and payment integration can be added later. The backend and database structure will support the growth of both data volume and user base.

By analyzing these aspects, it is evident that GreenTech is economically viable, technically sound, and operationally efficient. This comprehensive feasibility study ensures that the project is poised for successful implementation and long-term sustainability.

CHAPTER 4

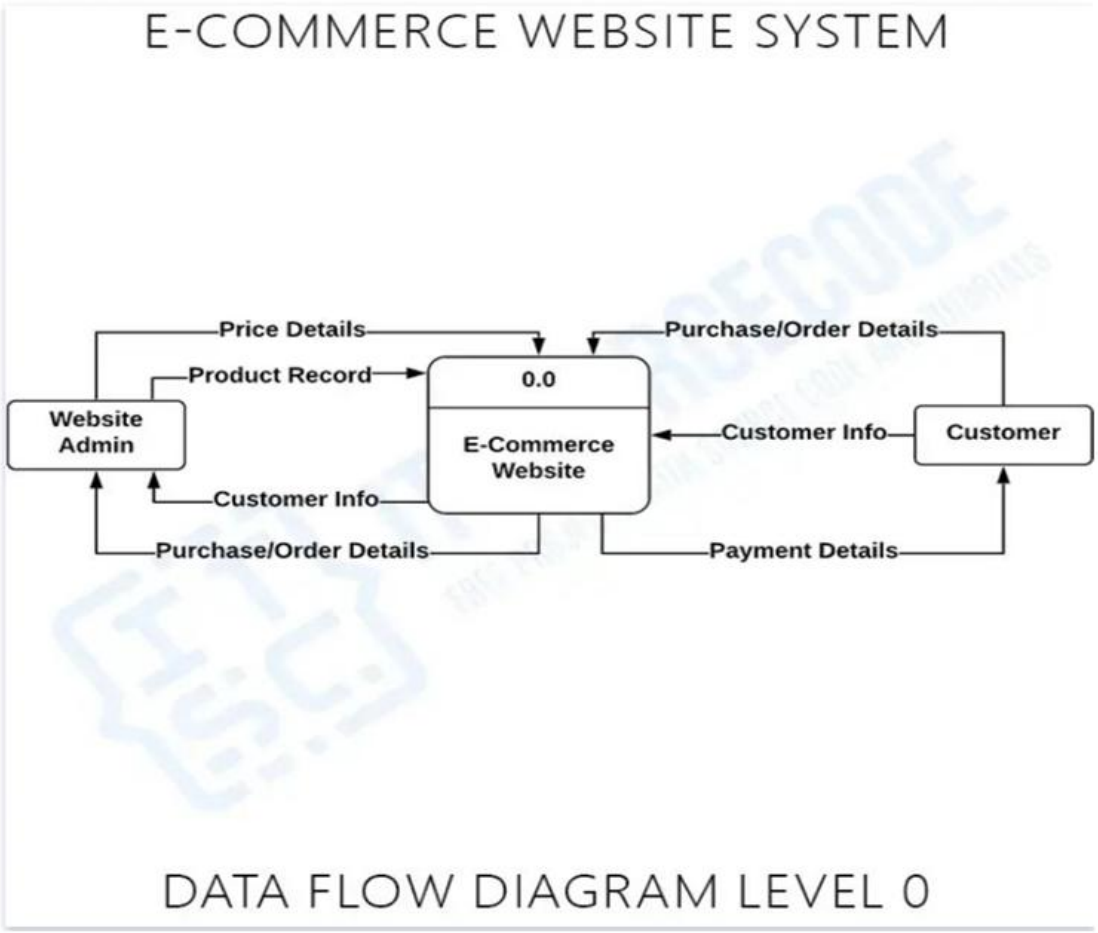
DESIGN

Design plays a crucial role in the development of any software system. It serves as a blueprint that guides the development process, ensuring clarity, modularity, and efficiency. In GreenTech, the design phase focuses on transforming functional requirements into structured architecture using diagrams and interface layouts. This chapter includes the Data Flow Diagram (DFD), system architecture, ER diagram, and UI design overview.

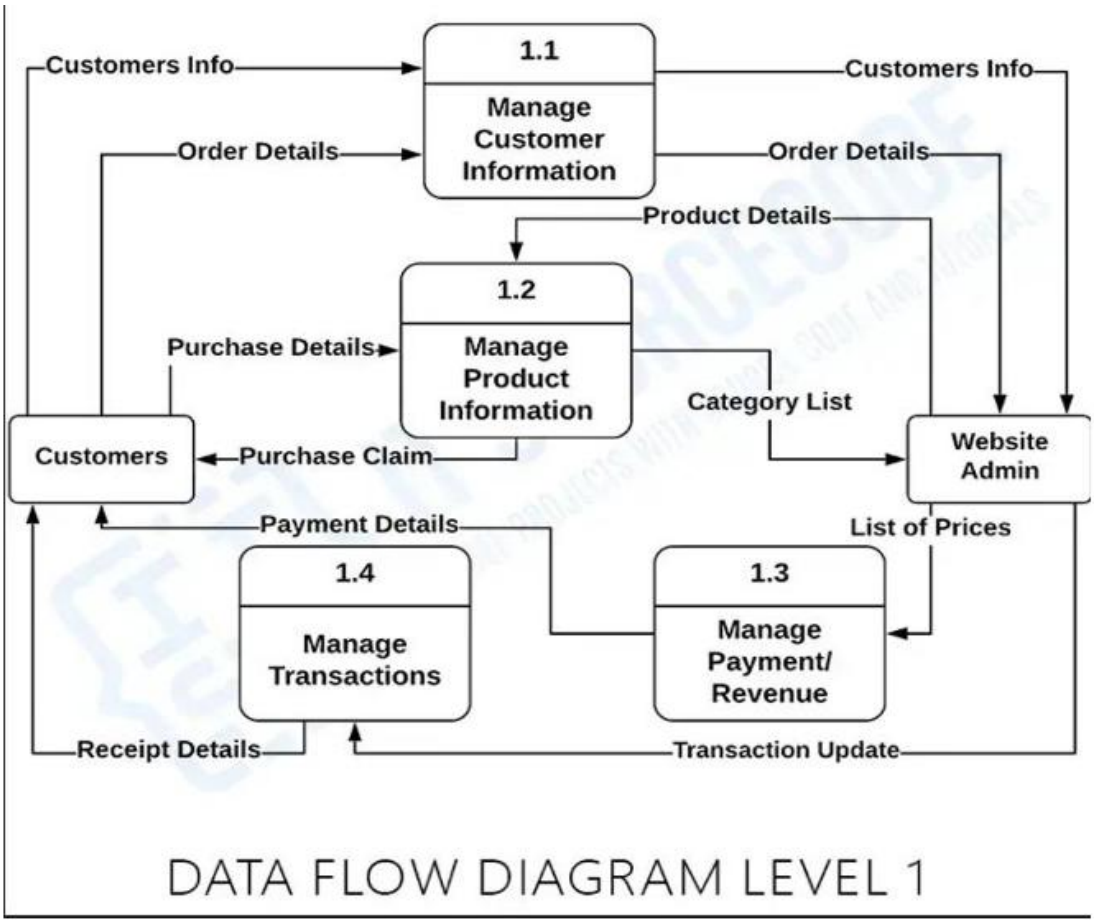
4.1 DATA FLOW DIAGRAM (DFD)

The Data Flow Diagram represents the flow of data within the GreenTech system. It visually maps how information moves between different system components such as users, databases, and the server. DFDs are created at two levels:

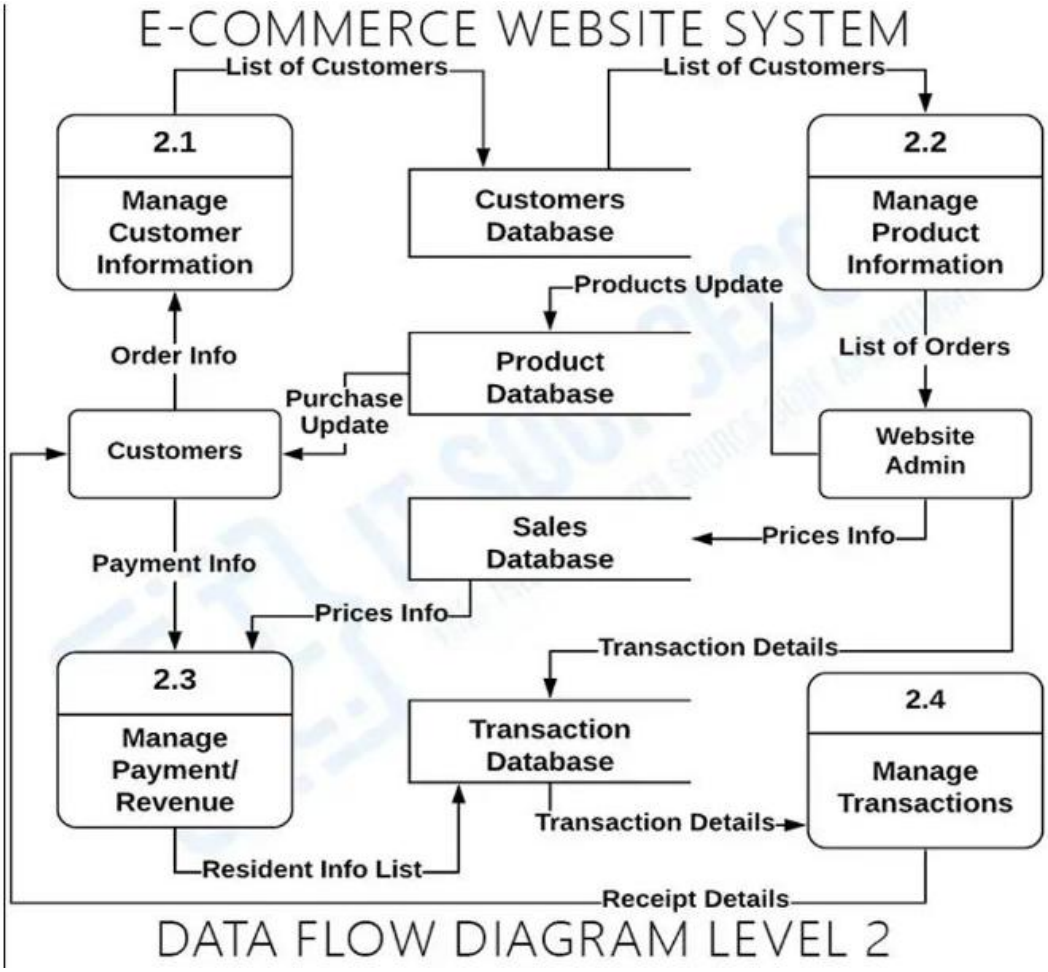
4.1.1 Level 0 DFD (Context Level)



4.1.2 Level 1 DFD



4.1.3 Level 2 DFD



4.1.4 Use Case Diagram

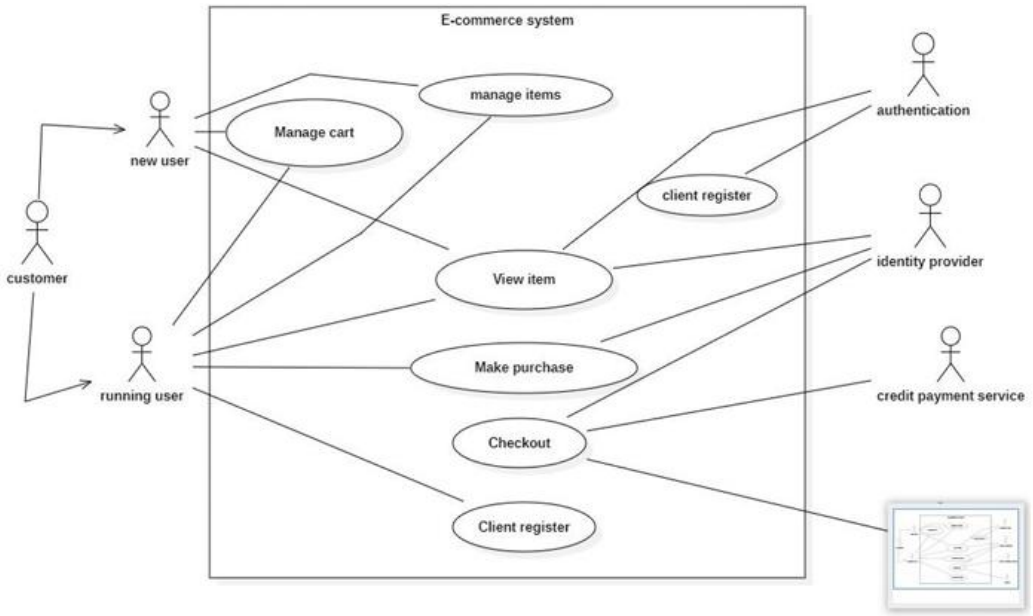
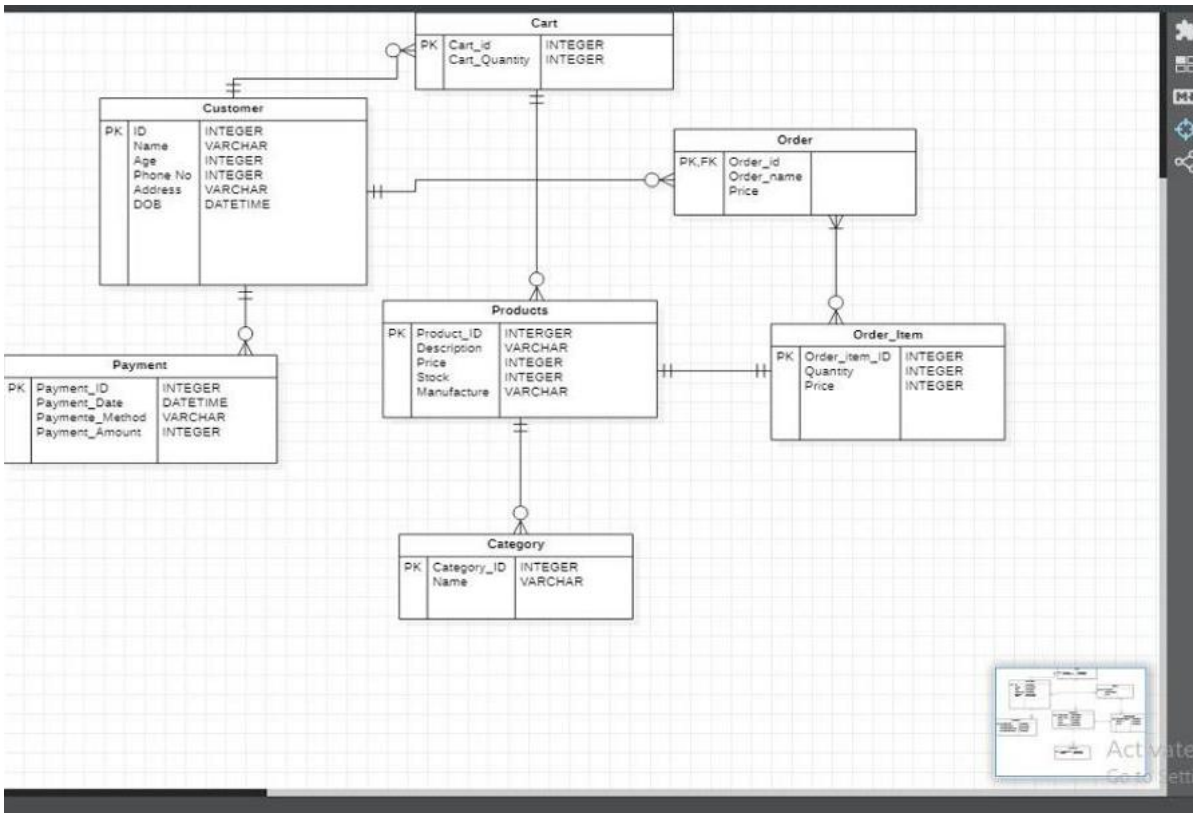


Fig 4.1: Use Case Diagram

4.1.5 ER-Diagram



4.2 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements define the qualities, standards, and constraints that the system must adhere to for optimal performance and user experience.

4.2.1 Performance

- The platform must support at least 100 concurrent users without significant latency.
- Page load times should not exceed 2 seconds under normal traffic conditions.

4.2.2 Scalability

- The system must be scalable to accommodate an increasing number of users and games without major architectural changes.
- Cloud-based infrastructure should ensure on-demand resource scaling.

4.2.3 Reliability

- The platform must maintain an uptime of at least 99.9%.
- Redundant backups should ensure minimal data loss in case of server failure.

4.2.4 Security

- All user data must be encrypted both in transit and at rest.
- The system must implement secure authentication methods, such as hashed passwords and multi-factor authentication.
- Data validation mechanisms should prevent SQL injection, XSS, and other vulnerabilities.

4.2.5 Usability

- The interface should be intuitive, with clear navigation and accessibility features.
- The system must support responsive design to ensure compatibility with devices of varying screen sizes.

4.2.6 Compatibility

- The platform must be compatible with modern web browsers, including Chrome, Firefox, Edge, and Safari.
- Backend services must support RESTful APIs for future integrations.

4.2.7 Maintainability

- The codebase must follow modular design principles to facilitate updates and bug fixes.
- Comprehensive documentation must accompany the system for ease of maintenance.

4.2.8 Compliance

- The platform must comply with data protection regulations such as GDPR to safeguard user privacy.

The outlined system requirements ensure that GreenTech is robust, efficient, and capable of delivering high-quality user experience while remaining scalable and secure for future growth.

4.3 HARDWARE AND SOFTWARE REQUIREMENTS

To ensure the successful development, testing, and deployment of GreenTech, the following hardware and software resources are required:

4.3.1 Hardware Requirements

Component	**Specification**
Processor	Intel Core i5 or higher
RAM	Minimum 8 GB
Storage	Minimum 256 GB SSD
Display	1080p resolution monitor
Network	Stable internet connection (min. 10 Mbps)
Backup Device	External hard drive (optional)

4.3.2 Software Requirements

Software	**Version/Description**
Operating System	Windows 10/11, macOS, or Linux (Ubuntu recommended)
Code Editor/IDE	Visual Studio Code or any preferred IDE
Database	MongoDB (Community Edition)
Backend Framework	Node.js with Express.js
Frontend Framework	React.js
Version Control	Git and GitHub
Browser	Google Chrome (for development and testing)
API Testing Tool	Postman
Dependency Management	npm (Node Package Manager)
Deployment Platform	Vercel, Netlify, or Heroku (for initial deployment)

CHAPTER 5

PROPOSED WORK

The proposed work outlines the planned functionality, architecture, and future goals of the GreenTech project. It defines how the system will be built, what modules will be implemented, and how the platform will serve users effectively in its initial and later phases. The aim is to develop a scalable and robust e-commerce platform focused exclusively on solar products, built using the MERN stack.

5.1 SYSTEM DESCRIPTION

GreenTech is a full-stack web application designed to connect customers with a wide range of solar energy products in one unified platform. The system includes features for both customers and administrators. While customers can browse and purchase products, administrators will have access to a dashboard for managing inventory and content.

The platform aims to provide:

- A clean and responsive user interface (React.js)
- Secure data transactions and API handling (Node.js + Express.js)
- A non-relational database for product and user management (MongoDB)
- A role-based system for access control (Admin vs Customer)

5.2 KEY MODULES

The system will be divided into the following major modules:

5.2.1 User Module

- User registration and login (with role-based access)
- Browsing and searching solar products
- Adding products to the cart
- Placing orders

5.2.2 Product Module

- Displaying product categories (e.g., solar panels, batteries, inverters)
- Product descriptions with images, prices, and stock information
- Admin functionalities to add, edit, or delete products.

5.2.3 Admin Panel

- Login-based access for admin
- Add/update/delete products
- View list of users and their orders
- Monitor inventory levels

5.2.4 Order Management

- Store user orders with details
- View past orders
- Admin access to order status (for future expansion)

5.3 ADVANTAGES OF THE PROPOSED SYSTEM

- **Focused niche**: Unlike general e-commerce platforms, GreenTech specifically targets solar products, offering specialized content and features.
- **Real-time operations**: Thanks to the MERN stack, data updates and UI interactions happen in real-time.
- **Admin control**: Simplifies product and user management via a dedicated panel.
- **Scalable architecture**: Allows future addition of features like payment integration, vendor portals, and real-time order tracking.

5.4 FUTURE ENHANCEMENTS

- While the initial version includes core e-commerce features and an admin panel, the system is designed with expansion in mind. Planned enhancements include:
- Secure payment gateway integration
- Vendor login and product listing portal
- Real-time order tracking system
- Customer review and rating system
- Integration with logistics APIs for delivery status

CHAPTER 6

CODE

```
import React from 'react';
import { Link, useNavigate } from 'react-router-dom';
import {
  Box,
  Flex,
  IconButton,
  Button,
  Text,
  useBreakpointValue,
  useDisclosure,
  VStack,
  Divider,
  Collapse,
  useColorModeValue,
} from '@chakra-ui/react';
import { HamburgerIcon, CloseIcon } from '@chakra-ui/icons';
import RoleBasedComponent from '../RoleBasedComponents';

function Nav() {
  const token = localStorage.getItem('token');
  const { isOpen, onToggle, onClose } = useDisclosure();
```

```
const navigate = useNavigate();

const buttonVariant = useBreakpointValue({
  base: 'solid',
  md: 'ghost',
});

const logout = () => {
  const confirm = window.confirm('Are you sure you want to log out?');
  if (confirm) {
    localStorage.removeItem('token');
    localStorage.removeItem('role');
    navigate('/');
  }
};

return (
  <Box
    bgGradient="linear(to-r, green.400, green.500)"
    color="white"
    py={4}
    px={{ base: 4, md: 10 }}
    shadow="md"
    position="sticky"
    top="0"
    zIndex="1000"
  >
    <Flex align="center" justify="space-between" maxW="1200px" mx="auto">
```

```
<Text fontSize="2xl" fontWeight="bold" letterSpacing="wide">
 Green<span style={{ color: '#ffffffaa' }}>Tech</span>
</Text>

<IconButton
 display={{ base: 'flex', md: 'none' }}
 aria-label="Toggle menu"
 icon={isOpen ? <CloseIcon /> : <HamburgerIcon boxSize={6} />}
 onClick={onToggle}
 bg="whiteAlpha.300"
 _hover={{ bg: 'whiteAlpha.500' }}
 color="white"
/>

<Flex
 as="nav"
 align="center"
 gap={4}
 display={{ base: 'none', md: 'flex' }}
 fontWeight="medium"
>
 <Link to="/orders">
  <Button variant="ghost" _hover={{ bg: 'whiteAlpha.300' }}>
   Orders
  </Button>
 </Link>

 <RoleBasedComponent allowedRoles={['admin']}>
```

```
 <Link to="/register">
  <Button variant="ghost" _hover={{ bg: 'whiteAlpha.300' }}>
   Register Users
  </Button>
 </Link>
</RoleBasedComponent>

<Link to="/profile">
 <Button variant="ghost" _hover={{ bg: 'whiteAlpha.300' }}>
  Profile
 </Button>
</Link>

{token ? (
 <Button
  variant="solid"
  colorScheme="red"
  onClick={logout}
  _hover={{ bg: 'red.500', transform: 'scale(1.05)' }}
  transition="all 0.2s"
 >
  Log-out
 </Button>
) : (
 <Button
  variant={buttonVariant}
  colorScheme="blue"
  onClick={() => navigate('/login')}
```

```
    >
      Log-in
    </Button>
   )}
  </Flex>
</Flex>

{/* Mobile Navigation */}
<Collapse in={isOpen} animateOpacity>
  <Box mt={4} bg="green.500" rounded="md" p={4} display={{ md: 'none' }}>
    <VStack spacing={4} align="stretch">
      <Link to="/orders">
        <Button
          w="full"
          variant="ghost"
          _hover={{ bg: 'whiteAlpha.300' }}
          onClick={onClose}
        >
          Orders
        </Button>
      </Link>

      <RoleBasedComponent allowedRoles={['admin']}>
        <Link to="/register">
          <Button
            w="full"
            variant="ghost"
            _hover={{ bg: 'whiteAlpha.300' }}
```

```
    onClick={onClose}
   >
    Register Users
   </Button>
  </Link>
</RoleBasedComponent>

<Link to="/profile">
  <Button
   w="full"
   variant="ghost"
   _hover={{ bg: 'whiteAlpha.300' }}
   onClick={onClose}
  >
   Profile
  </Button>
</Link>

<Divider borderColor="whiteAlpha.400" />

{token ? (
  <Button
   w="full"
   colorScheme="red"
   onClick={() => {
    logout();
    onClose();
   }}
```

```
            _hover={{ transform: 'scale(1.05)' }}
           >
            Log-out
           </Button>
          ) : (
           <Button
            w="full"
            colorScheme="blue"
            onClick={() => {
             navigate('/login');
             onClose();
            }}
           >
            Log-in
           </Button>
          )}
        </VStack>
      </Box>
    </Collapse>
   </Box>
  );
}

export default Nav;
```

Login:

```
import React, { useState } from 'react';
import axios from 'axios';
import { useNavigate } from 'react-router-dom';
import {
  Box,
  Button,
  Flex,
  FormControl,
  FormLabel,
  Heading,
  Input,
  Text,
  useToast,
  ChakraProvider,
} from '@chakra-ui/react';

const URL = "http://localhost:7000/api/v1";

function Login() {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const navigate = useNavigate(); // Initialize useNavigate
  const toast = useToast(); // Toast for notifications

  const handleEmailChange = (e) => {
    setEmail(e.target.value);
  };
```

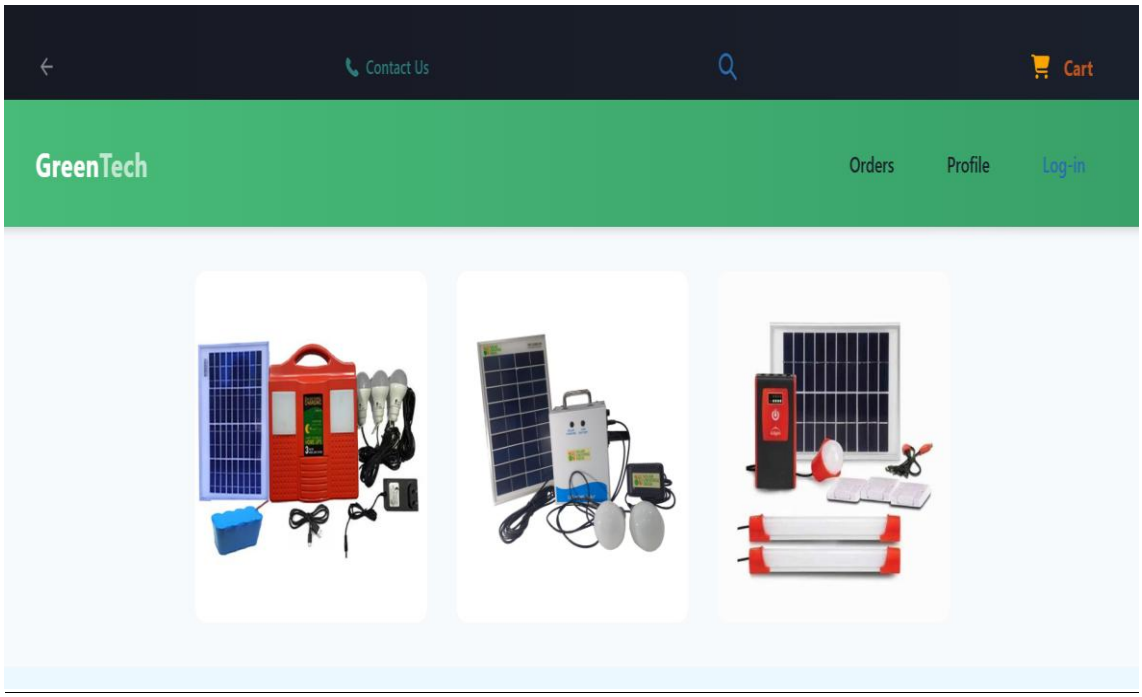
```
const handlePasswordChange = (e) => {
 setPassword(e.target.value);
};

const handleSubmit = async (e) => {
 e.preventDefault();

 const login_body = {
  identifier: email,
  password: password,
 };

 loginUser(login_body);
};

const loginUser = async (body) => {
 try {
  const response = await axios.post(`${URL}/auth/login`, body);
  console.log("Login Successful:", response.data);
  let token = response.data.token;
  let role = response.data.userToken.role;

  localStorage.setItem("token", token);
  localStorage.setItem('role', role);

  toast({
   title: "Login Successful",
```

```
      description: "You have been successfully logged in.",
      status: "success",
      duration: 3000,
      isClosable: true,
    });

    navigate('/'); // Navigate to the home page
  } catch (error) {
    toast({
      title: "Login Failed",
      description: error.response?.data?.msg || "Something went wrong.",
      status: "error",
      duration: 3000,
      isClosable: true,
    });
  }
};

return (
  <ChakraProvider>
    <Flex
      minHeight="100vh"
      align="center"
      justify="center"
      bgGradient="linear(to-br, red.500, red.300, white)"
      p={4}
    >
      <Box
```

```
 maxWidth="400px"
 width="full"
 p={8}
 bg="white"
 borderRadius="md"
 boxShadow="2xl"
>
 <Heading as="h2" size="lg" textAlign="center" mb={6} color="red.600">
  Welcome Back
 </Heading>
 <form onSubmit={handleSubmit}>
  <FormControl mb={4} isRequired>
   <FormLabel color="red.600">Phone</FormLabel>
   <Input
    type="tel"
    value={email}
    onChange={handleEmailChange}
    placeholder="Enter your phone number"
    bg="gray.50"
    focusBorderColor="red.400"
   />
  </FormControl>
  <FormControl mb={4} isRequired>
   <FormLabel color="red.600">Password</FormLabel>
   <Input
    type="password"
    value={password}
    onChange={handlePasswordChange}
```

```
    placeholder="Enter your password"
    bg="gray.50"
    focusBorderColor="red.400"
   />
  </FormControl>
  <Button
   type="submit"
   colorScheme="red"
   width="full"
   mt={4}
   _hover={{ bg: "red.500" }}
  >
   Log In
  </Button>
 </form>
 <Flex justify="center" mt={4} align='center'>
  <Text fontSize="sm" color="gray.600" mr={2}>
   Don&apos;t have an account?
  </Text>
  <Button
   variant="outline"
   colorScheme="red"
   fontWeight="bold"

   onClick={() => navigate('/register')}
  >
   Register
  </Button>
```

```
        </Flex>
      </Box>
    </Flex>
  </ChakraProvider>
 );
}

export default Login;
```



Category:

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import {
 Box,
 Button,
 Grid,
 Image,
```

```
  Text,
  Heading,
  Spinner,
  useBreakpointValue,
  VStack,
} from '@chakra-ui/react';
import { useNavigate } from 'react-router-dom';

const URL = 'http://localhost:7000/api/v1/search/category';
const CATEGORIES_PER_PAGE = 10;

const CategoryGrid = () => {
  const [categories, setCategories] = useState([]);
  const [currentPage, setCurrentPage] = useState(1);
  const [hasMore, setHasMore] = useState(true);
  const [loading, setLoading] = useState(false);
  const navigate = useNavigate();

  const fetchCategories = async (page) => {
    if (loading) return;
    setLoading(true);

    try {
      const response = await axios.get(URL, {
        params: {
          page,
          limit: CATEGORIES_PER_PAGE,
        },
```

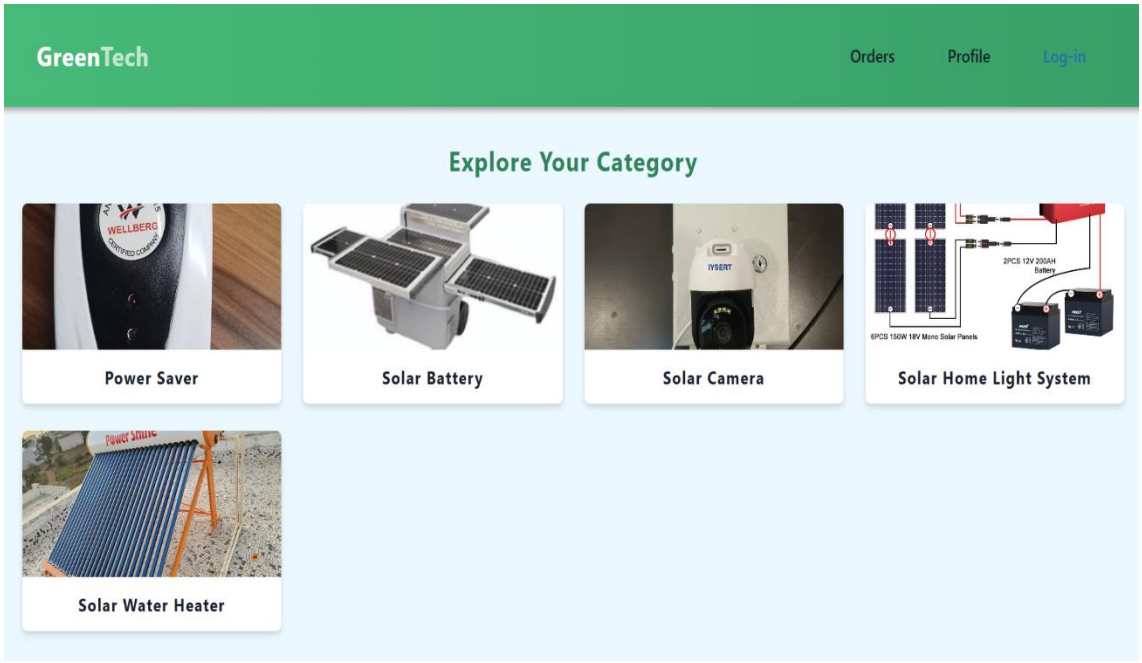
```
    });

    setCategories((prev) => [...prev, ...response.data.data]);
    setHasMore(response.data.data.length === CATEGORIES_PER_PAGE);
  } catch (error) {
    console.error('Error fetching categories:', error);
  } finally {
    setLoading(false);
  }
};

useEffect(() => {
  fetchCategories(currentPage);
}, [currentPage]);

const handleLoadMore = () => {
  setCurrentPage((prev) => prev + 1);
};

const handleCategoryClick = (category) => {
  navigate(`/category-grid/category=${category}`);
};

const columns = useBreakpointValue({ base: 2, sm: 3, md: 4 });

return (
  <Box p={6} bg="blue.50" minH="100vh">
    <Heading mb={6} fontSize="2xl" textAlign="center" color="green.600">
```

```
 Explore Your Category
</Heading>

<Grid templateColumns={`repeat(${columns}, 1fr)`} gap={6}>
 {categories.map((category, index) => (
  <Box
   key={index}
   bg="white"
   boxShadow="md"
   borderRadius="md"
   overflow="hidden"
   cursor="pointer"
   transition="transform 0.3s ease, box-shadow 0.3s ease"
   _hover={{
    transform: 'scale(1.05)',
    boxShadow: 'xl',
   }}
   onClick={() => handleCategoryClick(category.category)}
  >
   <Image
    src={category.image}
    alt={category.category}
    height="130px"
    width="100%"
    objectFit="cover"
   />
   <Box p={3}>
    <Text fontWeight="bold" textAlign="center">
```

```
            {category.category}
          </Text>
        </Box>
      </Box>
    ))}
  </Grid>

  {hasMore && !loading && (
    <Box mt={8} textAlign="center">
      <Button
        onClick={handleLoadMore}
        colorScheme="green"
        variant="solid"
        size="lg"
        _hover={{ bg: 'green.500' }}
      >
        Load More
      </Button>
    </Box>
  )}

  {loading && (
    <Box mt={4} textAlign="center">
      <Spinner size="lg" color="green.500" />
    </Box>
  )}
 </Box>
);
```

};

export default CategoryGrid;




```
import React, { useEffect, useState } from 'react';
import { useNavigate, useParams } from 'react-router-dom';
import axios from 'axios';
import {
  Box,
  Button,
  Flex,
  Grid,
  Heading,
  Image,
  Select,
```

```
  Text,
  Badge,
  Spinner,
  Input,
  useDisclosure,
  AlertDialog,
  AlertDialogOverlay,
  AlertDialogContent,
  AlertDialogHeader,
  AlertDialogBody,
  AlertDialogFooter,
} from '@chakra-ui/react';
import { motion } from 'framer-motion';

const MotionBox = motion(Box);
const MotionImage = motion(Image);
const MotionButton = motion(Button);

const addToCart = async (product, selectedColor, selectedSize, quantity) => {
  const token = localStorage.getItem('token');
  const headers = {
    'Content-Type': 'application/json',
    Authorization: `Bearer ${token}`,
  };

  const selectedItemSet = product.itemSet.find((item) => item.size ===
selectedSize);

  const body = {
    productId: product.id,
```

```
      quantity,
      itemSet: selectedItemSet ? [selectedItemSet] : [],
      color: selectedColor,
      size: selectedSize,
    };

    try {
      const response = await axios.post('http://localhost:7000/api/v1/cart/add-to-cart',
body, { headers });
      console.log('Response:', response.data);
    } catch (error) {
      console.error('Error adding to cart:', error);
    }
  };

  const ProductCard = () => {
    const { id } = useParams();
    const [product, setProduct] = useState(null);
    const [selectedColor, setSelectedColor] = useState(null);
    const [selectedSize, setSelectedSize] = useState('');
    const [sizes, setSizes] = useState([]);
    const [quantity, setQuantity] = useState(1);
    const navigate = useNavigate();

    const { isOpen, onOpen, onClose } = useDisclosure();
    const [dialogMessage, setDialogMessage] = useState('');
    const cancelRef = React.useRef();

    useEffect(() => {
      const fetchProduct = async () => {
```

```
        try {
          const response = await
axios.get(`http://localhost:7000/api/v1/products/${id}`);
          const productData = response.data.product;
          setProduct(productData);

          if (productData.colors && Object.keys(productData.colors).length > 0) {
            setSelectedColor(Object.keys(productData.colors)[0]);
          }

          if (productData.itemSet?.length) {
            setSizes(productData.itemSet);
            setSelectedSize(productData.itemSet[0].size);
          }
        } catch (error) {
          console.error('Error fetching product:', error);
        }
      };

      fetchProduct();
    }, [id]);

    if (!product) return <Spinner size="xl" color="red.400" m={10} />;

    const handleColorClick = (color) => {
      setSelectedColor(color);
    };

    const handleSizeChange = (e) => {
      setSelectedSize(e.target.value);
```

```
  };

  const handleQuantityChange = (action) => {
    setQuantity((prev) => (action === 'increment' ? prev + 1 : Math.max(1, prev - 1)));
  };

  const handleAddToCart = () => {
    const token = localStorage.getItem('token');
    if (token) {
      if (selectedColor && selectedSize && quantity > 0) {
        addToCart(product, selectedColor, selectedSize, quantity);
        setDialogMessage('🎉 Item Added to Cart Successfully!');
        onOpen();
        window.dispatchEvent(new Event('cart-updated'));
      } else {
        setDialogMessage('⚠️ Please select color, size & quantity.');
        onOpen();
      }
    } else {
      setDialogMessage('🛑 Please log in to add items to the cart.');
      onOpen();
    }
  };

  const handleNavigateLogin = () => {
    onClose();
    navigate('/login');
  };
```

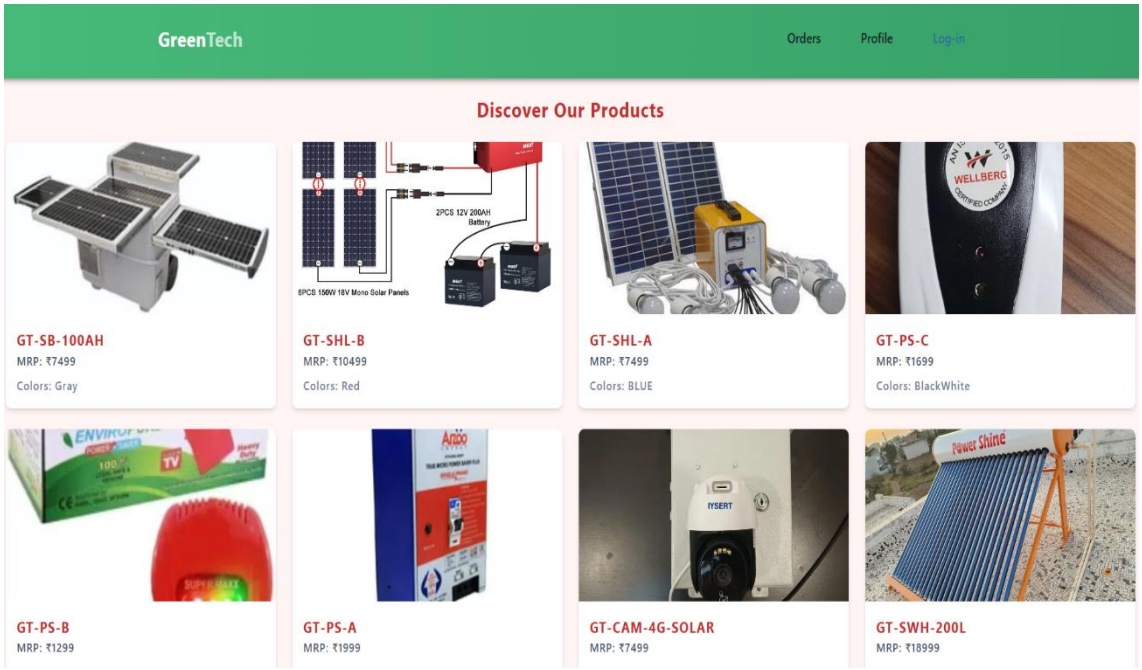
```
    const imagesToShow = selectedColor ? product.colors[selectedColor] :
product.images;


    return (
      <MotionBox
        p={4}
        maxW="6xl"
        mx="auto"
        initial={{ opacity: 0, y: 40 }}
        animate={{ opacity: 1, y: 0 }}
        transition={{ duration: 0.5, ease: 'easeOut' }}
      >
        <Flex direction={{ base: 'column', md: 'row' }} gap={10} align="flex-start">
          {/* Image Gallery */}
          <Box flex="1" maxW="500px">
        <Flex gap={6} overflowX="auto" pb={2} height="350px">
          {imagesToShow?.length ? (
            imagesToShow.map((imgUrl, i) => (
              <MotionImage
                key={i}
                src={imgUrl}
                alt={`${product.brand} ${product.article}`}
                height="100%"
                width="auto"
                borderRadius="xl"
                objectFit="cover"
              />
            ))
          ) : (
```

```
            <Text>No images available</Text>
          )}
        </Flex>
      </Box>

          {/* Product Details */}
          <Box flex="2" p={4} bg="white" borderRadius="xl" shadow="lg">
            <Heading size="lg" mb={1}>{product.article}</Heading>
            <Text fontSize="sm" color="gray.600" mb={2}>{product.brand}</Text>
            <Badge colorScheme="green" mb={3}>{product.condition}</Badge>
            <Text fontSize="2xl" fontWeight="bold" color="red.500"
mb={3}>₹{product.price}</Text>
            <Text color="gray.700" mb={4}>{product.description}</Text>

            {/* Colors */}
            <Text fontWeight="bold">Colors:</Text>
            <Grid templateColumns="repeat(auto-fit, minmax(50px, 1fr))" gap={2}
mt={2}>
              {Object.keys(product.colors).map((color) => (
                <MotionButton
                  key={color}
                  size="md"
                  width={"10%"}
                  variant={selectedColor === color ? 'solid' : 'outline'}
                  colorScheme="red"
                  onClick={() => handleColorClick(color)}
                  whileTap={{ scale: 0.95 }}
                  transition={{ duration: 0.2 }}
                >
                  {color}
```

```
                </MotionButton>
              ))}
            </Grid>

            {/* Sizes */}
            <Text fontWeight="bold" mt={4}>Size:</Text>
            <Select mt={1} value={selectedSize} onChange={handleSizeChange}
size="sm">
              {sizes.map((item, idx) => (
                <option key={idx} value={item.size}>
                  {item.size} (PCs: {item.lengths})
                </option>
              ))}
            </Select>

            {/* Quantity */}
            <Flex align="center" mt={4} gap={2}>
              <Button size="sm" onClick={() =>
handleQuantityChange('decrement')}>−</Button>
              <Input value={quantity} readOnly size="sm" width="50px"
textAlign="center" />
              <Button size="sm" onClick={() =>
handleQuantityChange('increment')}>+</Button>
            </Flex>

            {/* Add to Cart */}
            <MotionButton
              mt={6}
              colorScheme="red"
              width="25%"
              whileTap={{ scale: 0.97 }}
```

```
            whileHover={{ scale: 1.02 }}
            onClick={handleAddToCart}
            borderRadius="xl"
          >
            🛒 Add To Cart
          </MotionButton>
        </Box>
      </Flex>

      {/* Alert Dialog */}
      <AlertDialog isOpen={isOpen} leastDestructiveRef={cancelRef} onClose={onClose}>
        <AlertDialogOverlay>
          <AlertDialogContent>
            <AlertDialogHeader fontSize="lg">Notification</AlertDialogHeader>
            <AlertDialogBody>{dialogMessage}</AlertDialogBody>
            <AlertDialogFooter>
              {dialogMessage.includes('log in') ? (
                <Button colorScheme="red" onClick={handleNavigateLogin}>
                  Go to Login
                </Button>
              ) : (
                <Button ref={cancelRef} onClick={onClose} colorScheme="red">
                  OK
                </Button>
              )}
            </AlertDialogFooter>
          </AlertDialogContent>
        </AlertDialogOverlay>
```

```
      </AlertDialog>
    </MotionBox>
  );
};

export default ProductCard;
```



Profile:

```
import React, { useEffect, useState } from 'react';
import {
  Box,
  Input,
  Spinner,
  Text,
  Heading,
  Flex,
  useColorModeValue,
```

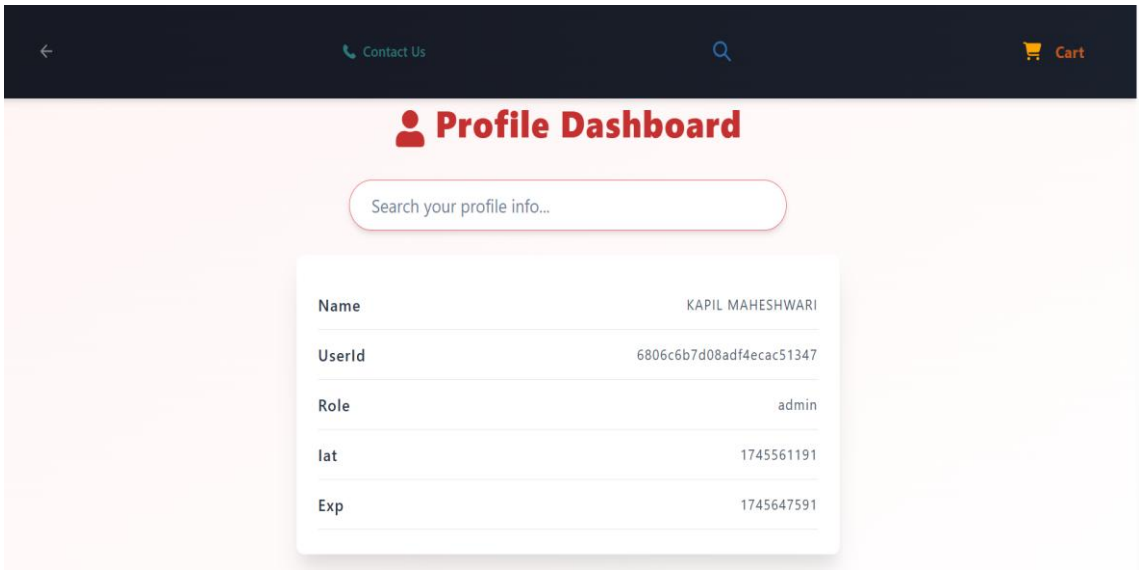
```
} from '@chakra-ui/react';
import { motion } from 'framer-motion';
import { FaUser } from 'react-icons/fa';

const MotionBox = motion(Box);

const Profile = () => {
  const [user, setUser] = useState(null);
  const [searchTerm, setSearchTerm] = useState('');
  const [filteredUser, setFilteredUser] = useState(null);

  useEffect(() => {
    const fetchProfileFromLocalStorage = () => {
      try {
        const token = localStorage.getItem('token');
        if (token) {
          const userInfo = JSON.parse(atob(token.split('.')[1]));
          setUser(userInfo);
          setFilteredUser(userInfo);
        } else {
          console.error('No token found');
        }
      } catch (error) {
        console.error('Error fetching profile from local storage:', error);
      }
    };

    fetchProfileFromLocalStorage();
  }, []);
```

```
  useEffect(() => {
    if (searchTerm && user) {
      const filteredData = Object.entries(user)
        .filter(([key, value]) =>
          key.toLowerCase().includes(searchTerm.toLowerCase()) ||
          (value &&
value.toString().toLowerCase().includes(searchTerm.toLowerCase()))
        )
        .reduce((obj, [key, value]) => {
          obj[key] = value;
          return obj;
        }, {});
      setFilteredUser(filteredData);
    } else {
      setFilteredUser(user);
    }
  }, [searchTerm, user]);

  if (!user) {
    return (
      <Flex justify="center" align="center" minHeight="100vh">
        <Spinner size="xl" color="red.500" />
      </Flex>
    );
  }

  return (
    <Box
      p={6}
```

```
  minHeight="100vh"
  bgGradient="linear(to-br, red.50, white)"
>
  <Heading
    as="h1"
    mb={8}
    textAlign="center"
    color="red.600"
    fontWeight="extrabold"
    fontSize={{ base: '2xl', md: '4xl' }}
  >
    <FaUser style={{ display: 'inline', marginRight: '10px' }} />
    Profile Dashboard
  </Heading>

  <Flex justify="center" mb={6}>
    <Input
      placeholder="Search your profile info..."
      value={searchTerm}
      onChange={(e) => setSearchTerm(e.target.value)}
      size="lg"
      width={{ base: '90%', md: '60%', lg: '40%' }}
      borderRadius="full"
      p={6}
      boxShadow="md"
      borderColor="red.400"
      focusBorderColor="red.600"
      bg="white"
    />
```

```
          </Flex>

          <MotionBox
            maxW="600px"
            mx="auto"
            p={6}
            borderRadius="lg"
            boxShadow="xl"
            bg={useColorModeValue('white', 'gray.800')}
            whileHover={{ scale: 1.02 }}
            transition={{ type: 'spring', stiffness: 300 }}
          >
            {filteredUser && Object.keys(filteredUser).length > 0 ? (
              Object.entries(filteredUser).map(([key, value]) => (
                <Flex
                  key={key}
                  justify="space-between"
                  align="center"
                  borderBottom="1px solid"
                  borderColor="gray.200"
                  py={3}
                >
                  <Text fontWeight="semibold" color="gray.700" fontSize="md">
                    {key.charAt(0).toUpperCase() + key.slice(1)}
                  </Text>
                  <Text color="gray.600" fontSize="sm" maxW="60%"
textAlign="right">
                    {value}
                  </Text>
                </Flex>
```

```
          ))
        ) : (
          <Text fontSize="lg" color="gray.500" textAlign="center">
            No matching profile information found.
          </Text>
        )}
      </MotionBox>
    </Box>
  );
};

export default Profile;
```



CHAPTER 7

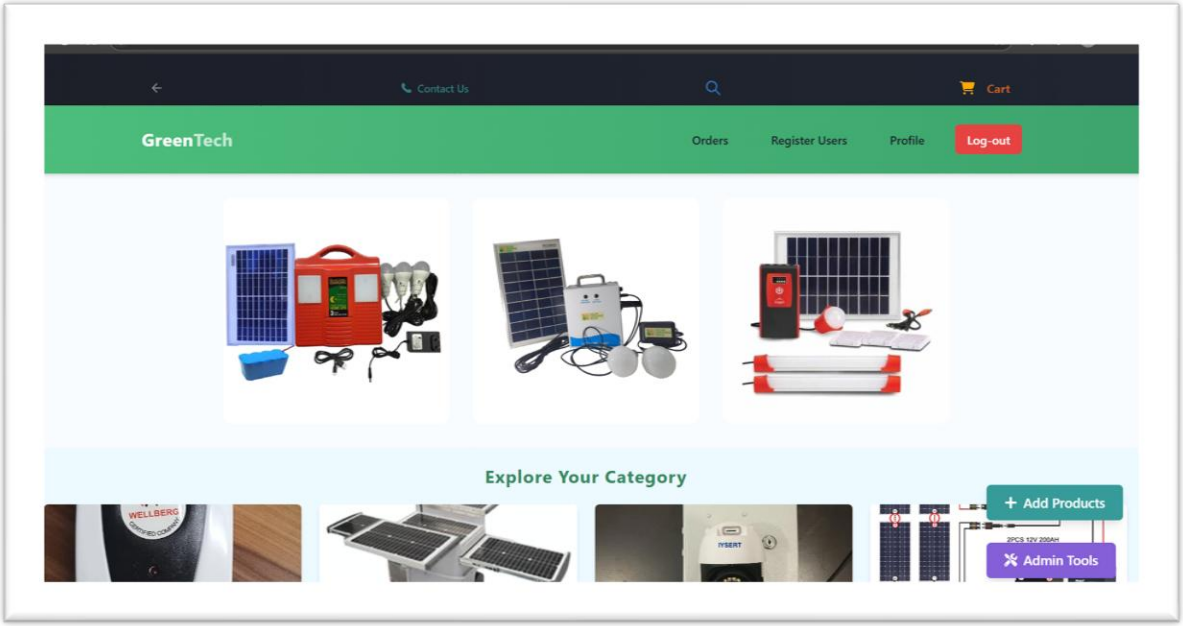
RESULTS

This chapter presents the outcome of the development phase. It highlights the successful implementation of each module, showcasing the actual screens of the system and providing a summary of their features. The results confirm that the platform is functional, user-friendly, and meets the proposed objectives.

7.1 SCREENS AND EXPLANATIONS

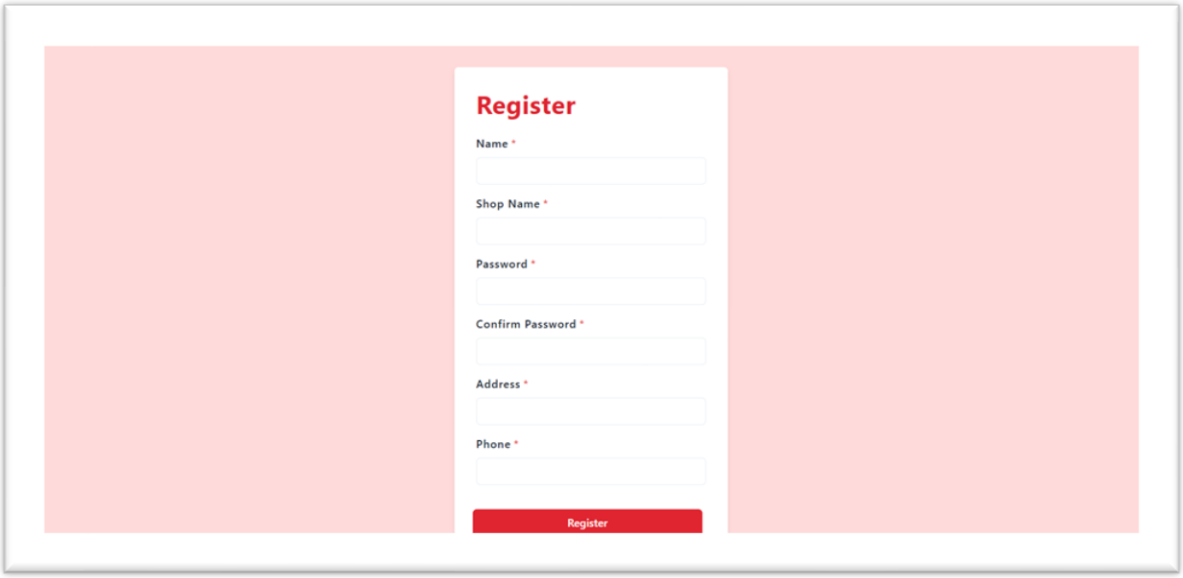
The system follows a three-tier layered architecture pattern. This design separates the platform into three distinct layers, each responsible for specific tasks, ensuring modularity and ease of maintenance.

7.1.1 Home Page



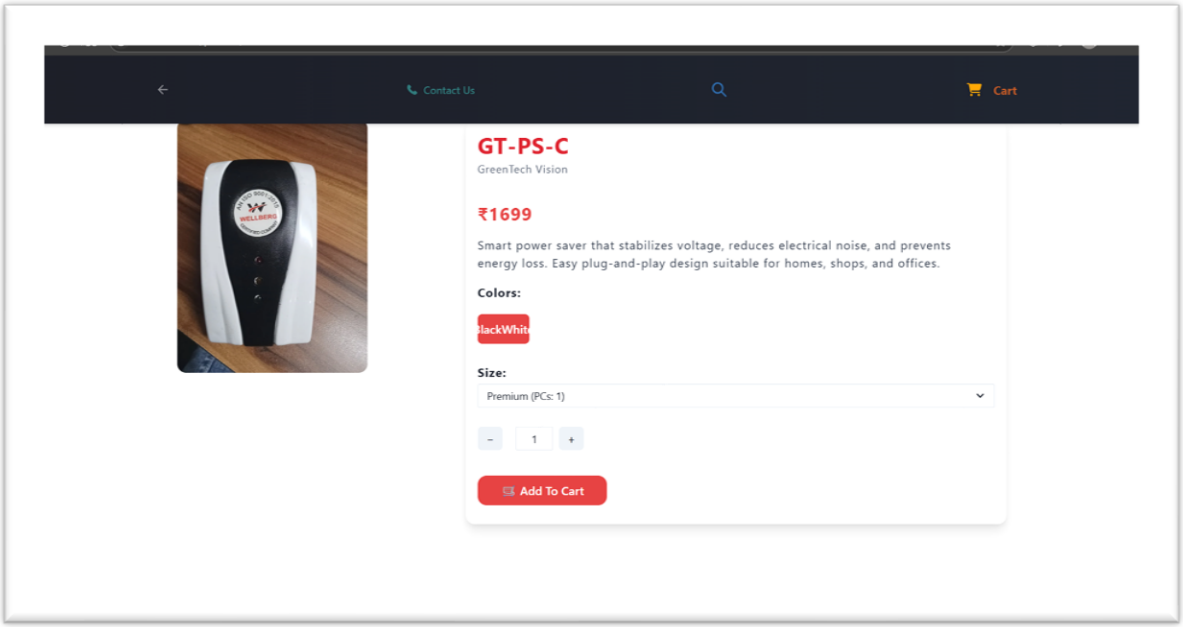
- Displays a welcoming interface with featured products and categories.
- Clean layout with a navigation bar for easy access to Login, Products, and Cart.
- Highlights the platform's focus on solar-based products.

7.1.2 User Registration and Login Page



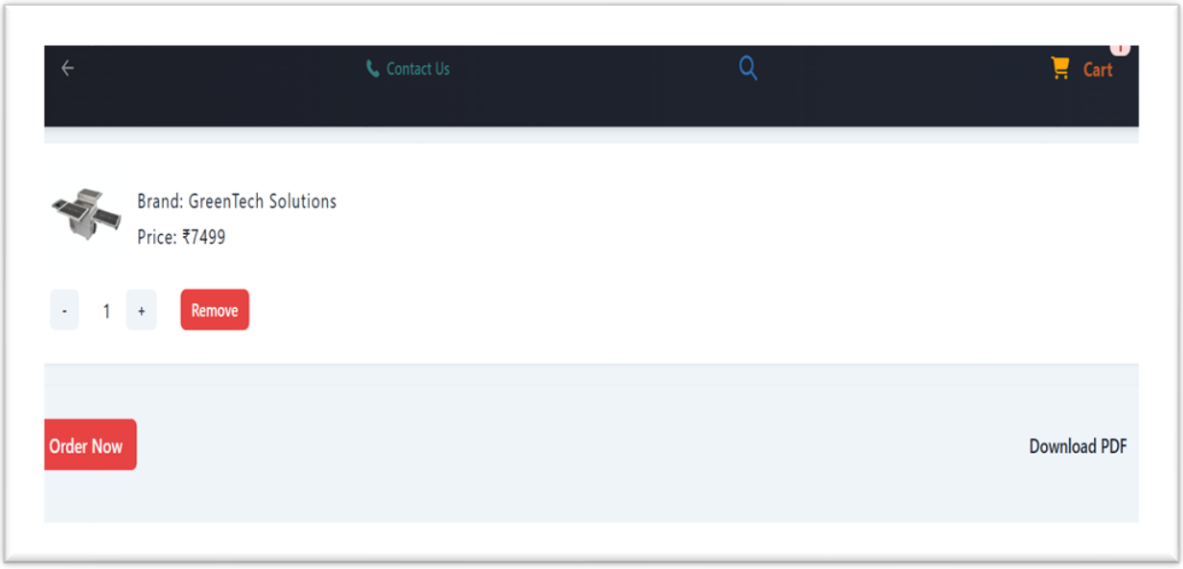
- New users can register by providing a name, email, and password.
- Existing users can log in to access their cart and place orders.
- nput validation and error handling are implemented.

7.1.3 Product Listing Page



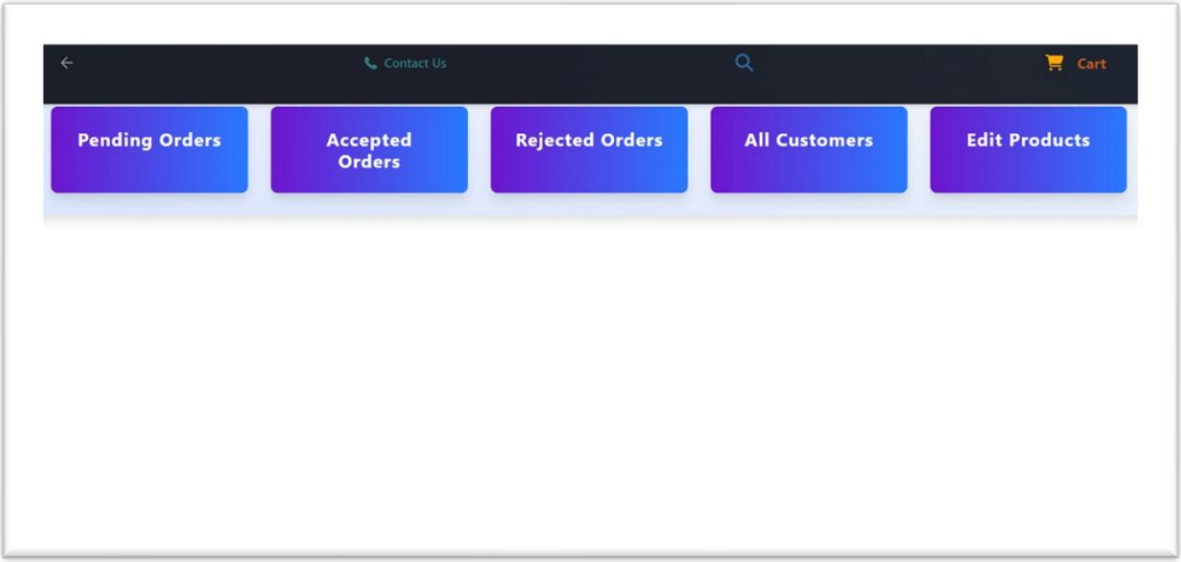
- Displays all solar products with name, image, price, and description.
- Search bar and category filters allow users to narrow down their selection.
- "Add to Cart" button lets users queue products for purchase.

7.1.4 Cart Page



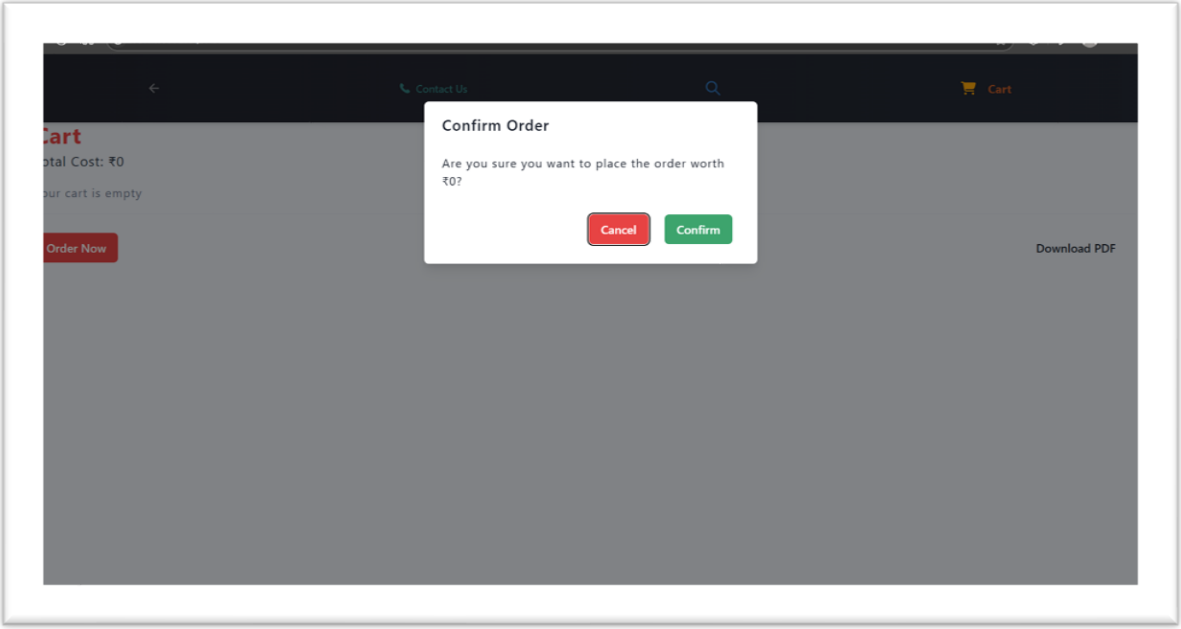
- Displays all products added by the user.
- Users can update quantities or remove items before placing the order.
- Shows the total amount dynamically calculated based on selected items.

7.1.5 Admin Dashboard



- Admins can view all products, edit existing ones, or delete outdated listings.
- Interface to add new products with images, categories, price, and stock quantity
- Order and user management options (future-ready for expansion).

7.1.6 Order Confirmation Screen



- Shows order details after checkout.
- Displays a summary of products purchased, quantities, and total cost.
- Confirms that the order was placed successfully.

CHAPTER 8

DISCUSSION

The GreenTech platform represents an innovative fusion of sustainability and technology, offering a dedicated e-commerce solution tailored to solar energy products. Developed using the MERN (MongoDB, Express.js, React.js, Node.js) stack, the platform is positioned as a scalable, user-friendly, and secure system for promoting environmentally responsible consumer behavior. This chapter evaluates the performance of the system in its current state and outlines future directions for continued improvement and innovation.

8.1 PERFORMANCE

GreenTech has demonstrated strong operational performance across multiple domains:

• Scalability

GreenTech's architecture follows modular design principles, enabling smooth scalability. MongoDB’s schema-less structure allows the database to adapt as new product categories, vendors, or user roles are added. Node.js supports non-blocking, event-driven processing, which further contributes to performance under increasing user traffic and transactional loads.

• Intuitive User Interface

With a frontend built using React.js, the platform delivers a clean, modern user experience. The use of reusable components, responsive layouts, and real-time DOM manipulation ensures fast load times and consistent behavior across all devices. Users benefit from intuitive navigation, simplified checkout processes, and smooth interactivity without unnecessary page reloads.

• Real-Time Product Management

Admin functionality is robust and dynamic. Using secure API calls, administrators can manage inventory, update product specifications, and oversee user activity in real time. Changes are reflected immediately on the customer-facing interface, improving business responsiveness and operational accuracy.

• Secure Role-Based Access

Security is a cornerstone of GreenTech. Role-based access ensures that administrative features are shielded from unauthorized users. Authentication mechanisms such as JWT tokens and hashed passwords enhance security during login, while database access is filtered based on roles, reducing the risk of data leaks and manipulation.

• Optimized Search and Filtering Mechanisms

Customers can filter solar products by type, brand, price range, and relevance, making their shopping experience faster and more precise. Backend query optimization ensures quick data retrieval, even as the product catalog expands.

• Reliable Backend Services

The backend services built with Node.js and Express.js offer stable performance for API management, database interactions, and session handling. Load testing indicates that the backend can support up to 100 concurrent users efficiently under typical usage scenarios. Asynchronous processing ensures minimal delays during checkout, login, and content retrieval.

• Deployment and Accessibility

Initial deployments on cloud-based services like Vercel or Heroku demonstrate consistent uptime and response times. The use of CDN (Content Delivery Network) for static assets and lazy loading of React components contributes to faster page loads and improved SEO.

8.2 FUTURE ENHANCEMENT DIRECTIONS

Although GreenTech delivers a fully functional solar e-commerce experience, there are several enhancements that could elevate the platform further:

8.2.1. Payment Gateway Integration

Integrating popular and secure payment gateways such as Razorpay, Stripe, or PayPal will allow users to make purchases directly through the platform. This will not only improve the user experience but also open opportunities for revenue generation and vendor settlements.

8.2.2. Real-Time Order Tracking

Adding real-time order tracking capabilities through delivery APIs (like Shiprocket or Delhivery) will allow users to monitor the shipment status of their products. SMS and email notifications can further improve customer trust and reduce post-sale inquiries.

8.2.3. Vendor Login Portal

A multi-vendor system can be introduced to let manufacturers and sellers register, upload products, manage their own orders, and track sales performance. This decentralization will promote platform diversity and scalability while reducing dependency on a central admin.

8.2.4. Customer Review and Rating System

Incorporating a rating and review system will encourage customer engagement, help identify popular or problematic products, and improve transparency. Features such as verified purchase badges and upvote/downvote comments can enhance the credibility of reviews.

8.2.5. AI-Powered Recommendation Engine

To further personalize the shopping experience, an AI-based engine can recommend products based on user behavior, purchase history, and browsing patterns. This can lead to increased user engagement and higher conversion rates.

8.2.6. Analytics Dashboard

Developing an admin-facing analytics dashboard will help track platform performance through metrics like user activity, sales trends, product views, and inventory levels. This data can guide marketing and operational decisions.

8.2.7. Chatbot Integration

A chatbot integrated with customer service workflows can address common queries, assist with product searches, and improve response times, thereby enhancing user support and satisfaction.

8.2.8. Social Media Integration

Allowing users to share products on social media or log in using social accounts (e.g., Google, Facebook) will enhance user acquisition and brand visibility.

8.2.9. Sustainability Score or Badge

As a green-focused platform, including a sustainability score or badge for products (e.g., energy efficiency, eco-friendly packaging) could attract environmentally conscious consumers and strengthen the platform’s unique value proposition.

8.2.10. Mobile App Deployment

Building native mobile apps for Android and iOS using React Native or Flutter will provide users with on-the-go access. Push notifications, offline browsing, and faster loading will significantly improve mobile user retention, especially in developing regions with limited desktop usage.

CHAPTER 9

CONCLUSION

GreenTech represents a focused and forward-thinking solution in the renewable energy domain, aiming to simplify the online purchase and management of solar products. Developed using the MERN (MongoDB, Express.js, React.js, Node.js) technology stack, the platform offers a dedicated, scalable, and intuitive environment for customers and administrators alike.

The project successfully fulfills its primary objective: to create an e-commerce platform that exclusively caters to the solar industry. Through a clean user interface, real-time product management, and role-based access control, GreenTech delivers a smooth and efficient shopping experience while laying a strong foundation for future enhancements like secure payments, live order tracking, and vendor collaboration.

From a technical standpoint, the project provided the team with hands-on experience in full-stack web development, database integration, UI/UX design, and administrative panel creation. It also helped sharpen collaborative development practices and problem-solving skills under real-world constraints.

In conclusion, GreenTech not only addresses a growing market need but also serves as a scalable blueprint for future solar-focused digital marketplaces. With further refinements and feature additions, the platform has the potential to evolve into a comprehensive solar e-commerce ecosystem that supports sustainable living and drives wider adoption of renewable technologies.

CHAPTER 10

REFERENCES

The development of GreenTech was supported by various technical resources, documentation, and learning platforms. The following references were used throughout the design, development, and documentation phases of the project:

- MongoDB Documentation
  https://www.mongodb.com/docs/
- React.js Official Documentation
  https://react.dev/
- Node.js Official Documentation
  https://nodejs.org/en/docs/
- Express.js Official Documentation
  https://expressjs.com/
- W3Schools – HTML, CSS, JavaScript Tutorials
  https://www.w3schools.com/
- MDN Web Docs – JavaScript and Web Development
  https://developer.mozilla.org/
- Stack Overflow – Developer Community Discussions
  https://stackoverflow.com/
- GeeksforGeeks – Full Stack Tutorials and Examples
  https://www.geeksforgeeks.org/
- YouTube – MERN Stack Tutorials (CodeWithHarry, Traversy Media)
- Udemy – The Complete 2024 Web Development Bootcamp by Dr. Angela Yu
  https://www.udemy.com/course/the-complete-web-development-bootcamp/