Karshak - Farmers Market Place: Revolutionizing Agricultural Trade for Farmer Prosperity and Good Food for Good Life

A Project Report Submitted
in the partial fulfillment of the requirements
for the award of the degree of

Master of Computer Applications
In

Department of Computer Science and Applications
By

BANDI NAVANEETHA
(2201600151)

Under the supervision of

DR. R. D. SATHIYA

Professor

Department of Computer Science and Applications

K L E F, Green Fields,

Vaddeswaram, Guntur, Andhra Pradesh, India- 522502.

2023- 24

KONERU LAKSHMAIAH EDUCATION FOUNDATION

DEPARTMENT OF COMPUTER SCIENCE AND APPLICATIONS

DECLARATION

The Project Report entitled "**Karshak-Farmers Market Place: Revolutionizing Agricultural Trade for Farmer Prosperity and Good Food for Good Life"** is a record of bonafide work of **BANDI NAVANEETHA,** submitted in partial fulfillment for the award of **Master of Computer Applications** in **Computer Science and Applications** of the K L University. The results embodied in this report have not been copied from any other departments/University/Institute.

Signature of the Student

Bandi Navaneetha

(2201600151)

KONERU LAKSHMAIAH EDUCATION FOUNDATION

DEPARTMENT OF COMPUTER SCIENCE AND APPLICATIONS

CERTIFICATE

This is to certify that the Project Report entitled "**Karshak-Farmers Market Place: Revolutionizing Agricultural Trade for Farmer Prosperity and Good Food for Good Life"** is being submitted by **BANDI NAVANEETHA**, in partial fulfillment of the requirements for the award of **Master of Computer Applications** in **Computer Science and Applications** to the K L Education Foundation is a record of bonafide work carried out under our guidance and supervision. The results embodied in this report have not been copied from any other departments/ University/ Institute.

Signature of the Supervisor	**Signature of the HOD**
(Dr.R.D.Sathiya)	**(Dr. G. Krishana Mohan)**
Professor	**Professor & HOD**

Signature of the Examiner

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people who made it possible and whose constant guidance and encouragement crown all the efforts with success.

I am very thankful to our project guide **Dr.R.D.Sathiya,** Professor for his continuous support and encouragement in completing the Project work. Without his help the project couldn't have been completed.

I express our heartfelt gratitude to **Dr. G. Krishna Mohan,** Head of the Department of Computer Science and Applications for providing us with adequate facilities, ways and means by which we are able to complete this project.

I express our heartfelt gratitude to **Dr. Subrahmanyam,** Professor and Principal College of Sciences, for providing us with adequate facilities, ways and means by which we can complete this project.

Last but not the least, we thank all Teaching and Non-Teaching Staff of our department and especially my classmates and my friends for their support in the completion of our project.

Signature of the Student

Bandi Navaneetha

(2201600151)

INDEX

ABSTRACT:

Karshak-Farmers Market Place embodies a transformative vision, blending technological innovation with a steadfast commitment to farmer prosperity and the promotion of good food for a good life. Leveraging the power of Django, HTML, CSS, Bootstrap, JavaScript, and AJAX, this platform seeks to reshape the traditional agricultural marketplace, particularly in the context of India.

By establishing a direct avenue for farmers to connect with consumers, Karshak-Market eliminates the reliance on intermediaries like brokers and dealers, ensuring fair prices and maximizing profits for farmers. Through an intuitive user interface and dynamic features driven by JavaScript and AJAX, the platform facilitates seamless access to high-quality, certified organic produce, promoting a healthier lifestyle and sustainable farming practices.

Karshak-Market's mission, encapsulated in "Good Food for Good Life," underscores its dedication to enhancing well-being and environmental stewardship. Stringent certification processes uphold the authenticity of organic products, instilling confidence in consumers and driving demand for sustainable agriculture.

Furthermore, Karshak-Market serves as a catalyst for empowerment within Indian agricultural communities. By providing farmers with access to valuable resources such as market insights, agricultural best practices, and financial assistance, the platform equips them to make informed decisions and thrive in their endeavors.

In summary, Karshak-Market epitomizes the transformative potential of technology in revolutionizing agricultural trade, ensuring prosperity for farmers and enriching lives through access to nutritious, organic food. As India's backbone, agriculture stands as a pivotal sector, and Karshak-Market's innovative approach holds promise in shaping a more sustainable and equitable food system for generations to come.

Additionally, recognizing the importance of government support, this paper also introduces the concept of an Android-based mobile application aimed at providing timely and relevant information on various government schemes for farmers across India. By addressing the shortcomings of existing applications, such as outdated information and security risks, this initiative aims to further empower farmers and enhance their access to crucial support mechanisms.

Keywords: - Model Creation, Django, Python-based Web Framework, Django Library, Web Server, HTML, CSS, AJAX, JavaScript, Bootstrap, Organic Products, Direct Sales, Sustainable Agriculture, Farmer-Customer Interaction, Online Marketplace, E-commerce, Fair Trade, Agricultural Innovation

OBJECTIVE:

The primary aim of this project is to establish a web application using Django and associated technologies to facilitate direct transactions between farmers and customers, enabling the sale of organic products and promoting sustainable agriculture practices.

CHAPTER 1

INTRODUCTION

In an era characterized by growing awareness of environmental sustainability and conscious consumerism, the direct exchange of organic products between farmers and consumers emerges as a pivotal solution. This application serves as a catalyst in this evolving landscape, leveraging technological advancements and the comprehensive capabilities of Django to redefine the accessibility and distribution of organic goods.

At its heart, this application serves as a vital bridge connecting farmers dedicated to cultivating organic produce with consumers who prioritize health and environmental responsibility. By harnessing Django's robust framework alongside HTML, CSS, AJAX, JavaScript, and Bootstrap, this platform orchestrates seamless transactions within an intuitive online marketplace.

The crux of this application lies in its transformative potential to empower farmers, offering them a direct platform to exhibit and sell their products, thereby circumventing conventional intermediaries. Beyond ensuring equitable remuneration for their labor, this approach nurtures a sustainable agricultural ecosystem by championing organic farming methodologies.

For consumers, the platform offers a portal to a diverse array of organic offerings, each embodying the values of ethical production and environmental stewardship. Through an immersive user interface complemented by dynamic functionalities powered by AJAX and JavaScript, customers can navigate, procure, and advocate for sustainable agriculture effortlessly.

In essence, this application transcends mere technological innovation; it embodies a vision of fostering a more equitable and environmentally conscious food system. As we embark on this journey, we embrace the profound potential of technology to forge genuine connections between farmers and consumers, heralding a future where every purchase contributes to a brighter and healthier world for all.

1.1 Purpose:

The primary objective of this project paper is to provide a comprehensive examination of the profound impact of utilizing Django and its associated technologies in crafting an innovative online platform. This platform acts as a pivotal conduit, facilitating direct engagements between farmers and consumers within the realm of organic food transactions. The focus of our inquiry extends beyond the mere technological framework; we aim to delve into the intricate development process, elucidating the functionalities and implications inherent in this transformative endeavor.

By meticulously detailing the evolution of this application, we aspire to highlight its pivotal role in reshaping the dynamics of agricultural trade. Emphasizing the significance of sustainable farming practices, we underscore how this platform serves as a catalyst for promoting ethical production methods and fostering environmental stewardship.

Central to our exploration is the profound impact of this platform on farmer empowerment. Through direct access to consumers, farmers are liberated from the constraints imposed by traditional intermediaries, ensuring fair compensation for their labor and reinforcing economic sustainability within agricultural communities.

Moreover, our inquiry extends to the broader implications for consumers, who are provided with unprecedented access to high-quality, certified organic food options. We illuminate how this platform enhances consumer choice, fosters a deeper connection to the food they consume, and promotes a culture of health-consciousness and environmental responsibility.

Through rigorous analysis and comprehensive exploration, we endeavor to shed light on the transformative potential of technology in fostering a more equitable and environmentally conscious food system. Ultimately, our aim is to contribute to the overarching goal of cultivating a healthier and more sustainable future for all stakeholders involved in the agricultural ecosystem.

1.2 Scope

This paper delves into the multifaceted scope of leveraging Django and associated technologies to establish an online platform facilitating direct transactions between farmers and consumers, particularly focusing on organic products. The scope encompasses an in-depth exploration of variables to include in the predictive model.

Feature engineering plays a critical role in enhancing the model's predictive capabilities.Once features are chosen and engineered, the project moves forward to developing predictive models utilizing suitable algorithms. This step involves employing various machine learning techniques to train models capable of accurately predicting smartphone addiction based on the collected data.

After model development, rigorous evaluation is necessary to assess its performance effectively. This evaluation phase helps determine the model's accuracy, reliability, and generalizability, ensuring its effectiveness in real-world applications.

Ethical considerations are paramount throughout the project, with careful attention paid to privacy, fairness, and bias mitigation. Ethical guidelines and standards must be adhered to, safeguarding the rights and well-being of individuals involved in the data collection and model deployment processes.

Upon successful development and evaluation, the final step involves deploying the model into practical applications. This deployment phase involves integrating the predictive model into relevant systems or platforms, enabling its use for early detection of smartphone addiction and facilitating timely interventions.

Documentation of the entire process is essential for transparency, reproducibility, and knowledge dissemination. Detailed documentation provides insights into the project's methodologies, findings, and implications, contributing to ongoing research efforts and informing future endeavors in the field of predictive analytics for public health outcomes improvement.

CHAPTER 2

LITERATURE SURVEY

[1] Joshi, N. C., Chayapathi, A. R., Nair, A. A., Ajay, S., & Suhail, M. S. (2021). Kisan Soch – A Mobile App for Farmers. International Journal of Creative Research Thoughts (IJCRT), 9(7), 346-351.

In their paper titled "Kisan Soch – A Mobile App for Farmers," Joshi et al. (2021) highlight the importance of agriculture in India and the need for a mobile application to provide farmers with information about government schemes. They address the limitations of existing applications, such as outdated information and language barriers, by developing a multilingual mobile application using React Native and Neural Machine Translation (NMT) techniques.

The authors discuss the system architecture, which is based on a Client-Server Architecture with three tiers: Presentation Layer, Business Layer, and Data Layer. They emphasize the use of React Native for cross-platform compatibility and Google Firebase for data storage. The application utilizes NMT with an attention mechanism to translate government scheme information from English to the user's selected language.

In the literature survey, the authors explore various research papers on NMT and transformer models to improve translation quality. They discuss the challenges associated with NMT and highlight the effectiveness of transformer models with attention mechanisms in achieving accurate translations.

The methodology section outlines the development process, including programming the application, data collection, data pre-processing, and deploying the machine learning model to the cloud. The authors emphasize the importance of multilingualism in catering to the diverse language preferences of Indian farmers.

The results section showcases snapshots of the application in different languages and discusses its features, including crop calendar, loan EMI calculator, weather forecast, and government schemes search. The conclusion emphasizes the goal of providing a convenient tool for farmers to access information in their desired language.

In terms of future work, the authors propose enhancements such as providing additional regional language options, UI improvements, integrating GPS for weather prediction, and adding schemes from different states.

Reference:

Joshi, N. C., Chayapathi, A. R., Nair, A. A., Ajay, S., & Suhail, M. S. (2021). Kisan Soch – A Mobile App for Farmers. International Journal of Creative Research Thoughts (IJCRT), 9(7), 346-351.

Additionally, the authors reference a YouTube video (https://youtu.be/qwFBXuEeg1U?si=xCrfdMj4uFcBYbHe) in their paper, which might provide further context or background information relevant to their research.

2.1 SUMMARY:

The literature survey for the "Karshak" project succinctly summarizes existing research and developments in agricultural mobile applications and machine learning techniques. It underscores the significance of technology in supporting farmers and advancing agricultural practices. The survey examines various mobile applications tailored to assist farmers in accessing information, managing crops, and accessing markets, emphasizing the importance of user-friendly interfaces and multilingual support. It explores the role of machine learning algorithms in agriculture, particularly in predicting crop yield, detecting diseases, and analyzing soil composition. Additionally, the survey identifies challenges faced by farmers, such as limited access to information and market volatility, while highlighting opportunities presented by technology, such as data-driven decision-making and precision agriculture. It also suggests areas for future research, including improving the scalability and accessibility of agricultural applications and integrating IOT devices for real-time monitoring. Overall, the literature survey provides a comprehensive overview of the current landscape of agricultural technology, informing the development of the "Karshak" application to address the needs of farmers and enhance agricultural productivity.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

The existing system, as described in the reference paper, refers to the current state or setup before the proposed mobile application is introduced. In the context of the paper, the existing system likely encompasses traditional methods or platforms used by farmers to access information about government schemes, agricultural practices, and market trends.

3.2 DISADVANTAGES

In the reference paper, one notable potential disadvantage is the absence of direct connectivity with farmers and customers. While the paper focuses on the development of a mobile application aimed at providing information on government schemes to farmers, it does not explicitly mention features or mechanisms for direct interaction between farmers and customers. This lack of direct connectivity could limit the app's effectiveness in facilitating communication, feedback, or transactions between farmers and their customers.

Direct connectivity could include features such as:

- **Communication Channels:** Providing chat or messaging functionality within the app for farmers to interact with customers, agricultural experts, or government representatives.
- **Marketplace Integration:** Facilitating direct sales or transactions between farmers and customers through an integrated marketplace feature within the app.
- **Feedback Mechanisms:** Allowing farmers to receive feedback from customers regarding their produce or services, enabling continuous improvement and customer satisfaction.
- **Customer Engagement:** Implementing features to engage customers with farmers' stories, agricultural practices, or sustainability efforts, fostering a sense of connection and support for local farmers.

Addressing this limitation could foster stronger relationships between farmers and their customers, potentially improving market access and agricultural livelihoods.Furthermore, another potential disadvantage highlighted in the context of the reference app is that customers may not directly benefit from the products or services offered through the mobile application. While the primary focus of the app is to provide information about government schemes to farmers, it may overlook the needs and preferences of end consumers who purchase agricultural products.

3.3 Proposed System

To overcome this limitation, the app could incorporate features such as:

Based on the information provided in the abstract of your paper "Karshak with Add-On Advantages," your project appears to address several of the challenges present in the existing system described in the reference paper. Here's how your project may overcome those challenges.

3.4 Advantages:

Direct Connectivity with Farmers and Customers: Your project likely provides features that facilitate direct connectivity between farmers and customers. This could include communication channels within the application, such as chat or messaging functionality, allowing farmers to interact with customers, agricultural experts, or government representatives.

Enhanced Market Access: By facilitating direct sales or transactions between farmers and customers through an integrated marketplace feature within the app, your project improves market access for farmers. This enables farmers to showcase their products and engage with potential buyers, overcoming the challenge of limited market access described in the existing system.

Improved Information Access: Your project may provide comprehensive information within the app about government schemes, agricultural practices, and market trends, addressing the challenge of limited access to up-to-date information faced by farmers in the existing system. Additionally, if your project supports multiple languages, it can overcome the language barrier and ensure that information is accessible to farmers regardless of their native language.

Feedback Mechanisms: By incorporating feedback mechanisms within the app, your project enables farmers to receive input from consumers or agricultural experts regarding their products and farming techniques. This addresses the challenge of limited feedback mechanisms described in the existing system, allowing farmers to tailor their offerings to better meet consumer preferences.

Customer Benefits: Your project likely focuses on providing direct benefits to customers by offering them access to high-quality agricultural products, information about farming practices, and opportunities to support local farmers. This ensures that customers directly benefit from the products or services offered through the app, overcoming the potential disadvantage mentioned in the reference paper.

Overall, your project "Karshak with Add-On Advantages" appears to provide solutions that overcome many of the challenges present in the existing system, thereby improving the efficiency, accessibility, and effectiveness of agricultural practices and market interactions for farmers and customers alike.

CHAPTER 4
SYSTEM REQUIREMENTS SPECIFICATION

4.1 HARDWARE CONFIGURATION:

- Processor - Intel(R) Core(TM) i3-1005G1 CPU @ 1.20GHz 1.19 GHz
- Hard Disk -160 GB
- RAM - 8 GB

4.2 SOFTWARE CONFIGURATION:

- Operating System : Windows 7/8/10 .
- Server side Script : HTML, CSS ,JAVASCRIPT, AJAX,BOOTSTRAP.
- IDE : VSCODE.
- Libraries Used : OS, Django (Python)
- Technology : Python 3.6+.

4.3 FUNCTIONAL AND NONFUNCTIONAL REQUIREMENTS:

Requirement's analysis is a very critical process that enables the success of a system or software project to be assessed. Requirements are generally split into two types: Functional and non-functional requirements.

Functional Requirements:

User Registration and Authentication:

- Users should be able to register an account.
- Users should be able to log in securely with their credentials.

Multilingual Support:

- The application should support multiple languages to cater to users from diverse linguistic backgrounds.

Search Functionality:

- Users should be able to search for specific government schemes based on keywords or categories.

Direct Connectivity:

- Implement features for direct communication between farmers and customers.
- Facilitate transactions or feedback mechanisms between farmers and customers.

User Feedback:

- Allow users to provide feedback on the application's usability and the information provided about government schemes.

Non-Functional Requirements:

Security:

- Ensure robust security measures to protect user data and authentication credentials.
- Implement encryption for sensitive information transmission.

Performance:

- The application should load quickly and respond promptly to user interactions.
- Optimize data retrieval and processing to minimize loading times.

Reliability:

- The application should be stable and reliable, with minimal downtime or crashes.
- Implement error handling mechanisms to gracefully handle unexpected situations.

Scalability:

- Design the application architecture to accommodate potential future growth in user base and data volume.
- Ensure scalability of the backend infrastructure to handle increased traffic and workload.

Accessibility:

- Ensure that the application is accessible to users with disabilities, following relevant accessibility guidelines and standards.

Usability:

- Design the user interface to be intuitive and user-friendly, catering to users with varying levels of technical expertise.
- Conduct usability testing to gather feedback and make improvements based on user experience.

These requirements serve as a foundation for designing, developing, and testing the "Karshak" project, ensuring that it meets the needs of its target users effectively and efficiently.

Methodology:

Methodology for Developing the "Karshak" Application:

A. Development Environment Setup:

Utilize Visual Studio Code (VSCODE) as the integrated development environment (IDE) for coding.

Set up the development environment with Python 3.6 or higher as the primary programming

language.

Install necessary libraries and frameworks, including Django for backend development, and utilize OS library for system-level operations.

B. Frontend Development:

Utilize HTML, CSS, JavaScript, and Bootstrap for frontend development to create a responsive and visually appealing user interface.

Implement dynamic frontend features using JavaScript to enhance user interaction and experience.

Leverage Bootstrap framework for efficient styling and layout of frontend components.

C. Backend Development:

Develop the backend logic using Django, a high-level Python web framework, to handle server-side operations and data processing.

Utilize Django's built-in features for user authentication, data modeling, and routing to streamline backend development.

Implement AJAX (Asynchronous JavaScript and XML) for asynchronous communication between the frontend and backend, enabling seamless data exchange without page reloads.

D. Data Collection and Preprocessing:

Gather relevant datasets containing agricultural information, government schemes, and other pertinent data for the application.

Preprocess the collected data using Python scripts to clean, filter, and organize it for efficient storage and retrieval.

E. Integration and Deployment:

Integrate the frontend and backend components to create a cohesive and functional application.

Deploy the application on a hosting platform compatible with Python, such as Heroku or PythonAnywhere, for accessibility and scalability.

Ensure smooth deployment by testing the application thoroughly in the production environment.

F. Testing and Quality Assurance:

Conduct rigorous testing of the application to identify and address any bugs, errors, or usability issues.

Perform user acceptance testing (UAT) to ensure that the application meets the requirements and expectations of its target users.

Implement continuous integration and continuous deployment (CI/CD) practices to automate testing and deployment processes for efficiency and reliability.

By following this methodology, the "Karshak" application aims to provide a robust and user-friendly platform for farmers to access essential agricultural information and resources effectively.

CHAPTER 5

SYSTEM ARCHITECTURE

ARCHITECTURE

```
[User Interface] <---> [Application Logic] <---> [Data Access Layer]
                         |
                         V
               [External Services]
```

5.1 ALGORITHM:

User Authentication:

- Prompt the user to log in with their credentials (username and password).
- Verify the user's credentials against the database.

Main Menu Navigation:

- Once logged in, display the main menu options to the user.
- Allow the user to choose from various features like viewing government schemes, accessing market information, etc.

View Government Schemes:

- Provide an option for users to view various government schemes available for farmers.
- Retrieve schema information from the database and display it to the user.

Access Market Information:

- Allow users to access market information such as crop prices, weather forecasts, etc.
- Integrate with external APIs or services to fetch real-time market data.

Product Information:

- Include a section where users can access information about different agricultural products available in the market.
- Display details such as product name, description, cultivation methods, nutritional value, etc.

Health Benefits:

- Provide information about the health benefits of various agricultural products.
- Highlight nutritional content, medicinal properties, and dietary advantages to encourage users to consume these products.

Update Profile:

- Enable users to update their profile information such as contact details, farm location, etc.
- Validate and save the updated information to the database.

Feedback Mechanism:

- Implement a feedback mechanism where users can provide feedback or suggestions.
- Store feedback in the database for further analysis and improvement of the application.

Logout:

- Provide an option for users to log out securely from their accounts.

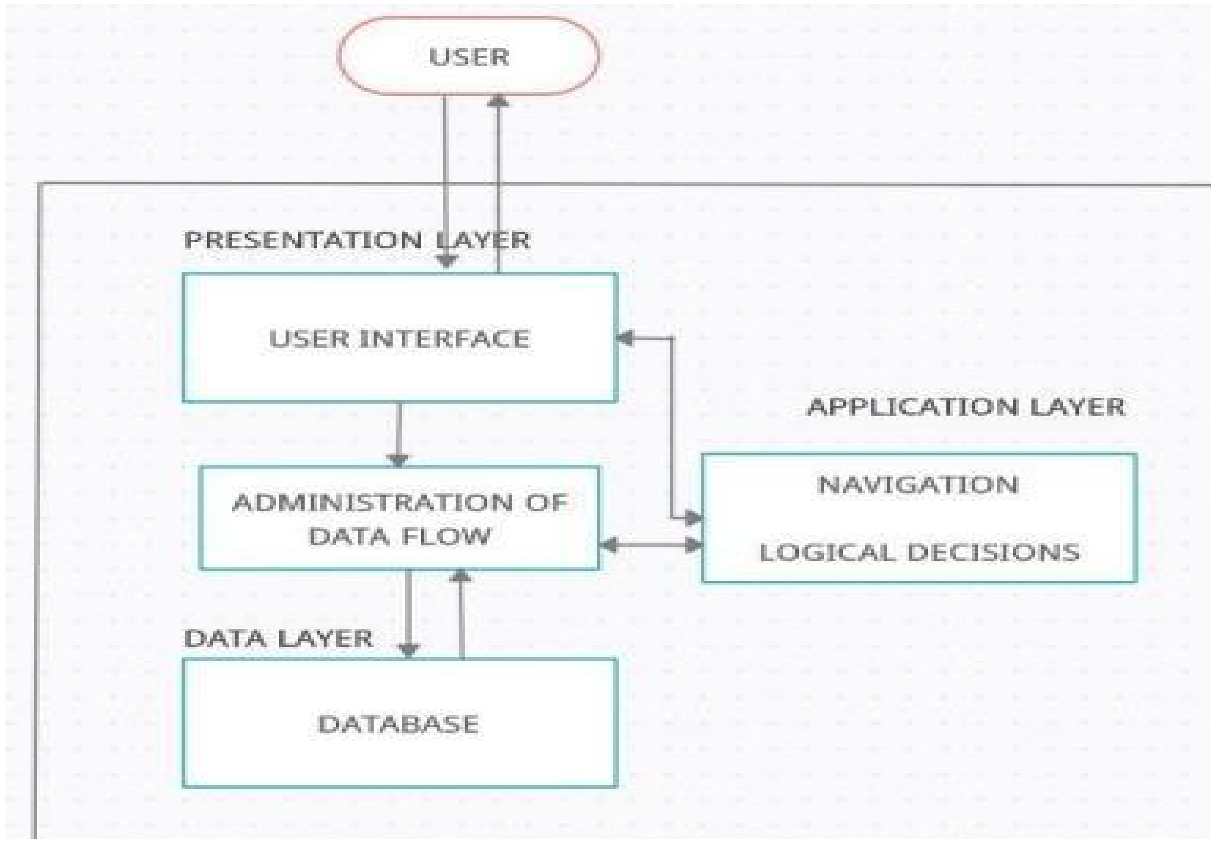


Fig. 5.1 Architecture diagram

CHAPTER 6

SYSTEM DESIGN

Block Diagram:

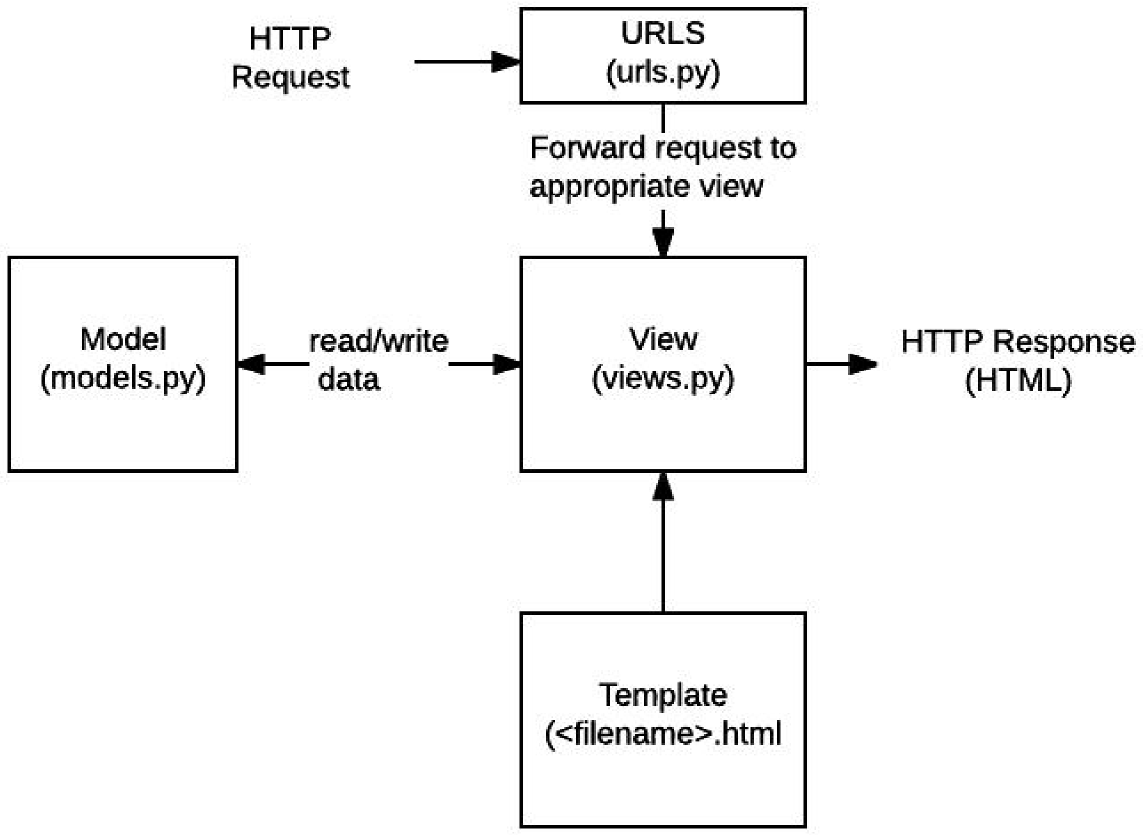


Fig 6. Block Diagram of Proposed System

6.1 UML DIAGRAMS:

The UML (Unified Modeling Language) diagram for the "Karshak" application provides a visual representation of its architectural components and their interactions. Here's some context for the UML diagram:

Overall Architecture:

- The diagram depicts the high-level architecture of the "Karshak" application, including its major components and how they interact with each other.

Component Breakdown:

- It illustrates the key components of the application, such as the user interface, application logic, data access layer, external services, security layer, and any other relevant modules.

Component Interaction:

- The diagram shows how these components communicate and interact with each other to fulfill various functionalities of the application. This includes the flow of data and control between different parts of the system.

Deployment Configuration:

- It may include details about how the application is deployed across different nodes or computing environments, such as servers, databases, cloud platforms, or client devices.

Technology Stack:

- The diagram may also indicate the technologies and frameworks used for implementing each component, such as React for the front-end, Node.js for the back-end, and Firebase for data storage.

Security and Infrastructure:

- If applicable, the diagram might include components related to security measures, such as authentication and authorization mechanisms, as well as infrastructure components like load balancers or caching servers.

Overall, the UML diagram for the "Karshak" application serves as a blueprint for understanding its architecture and helps stakeholders visualize the system's design and implementation. It provides valuable insights for developers, architects, and project managers involved in building and maintaining the application.

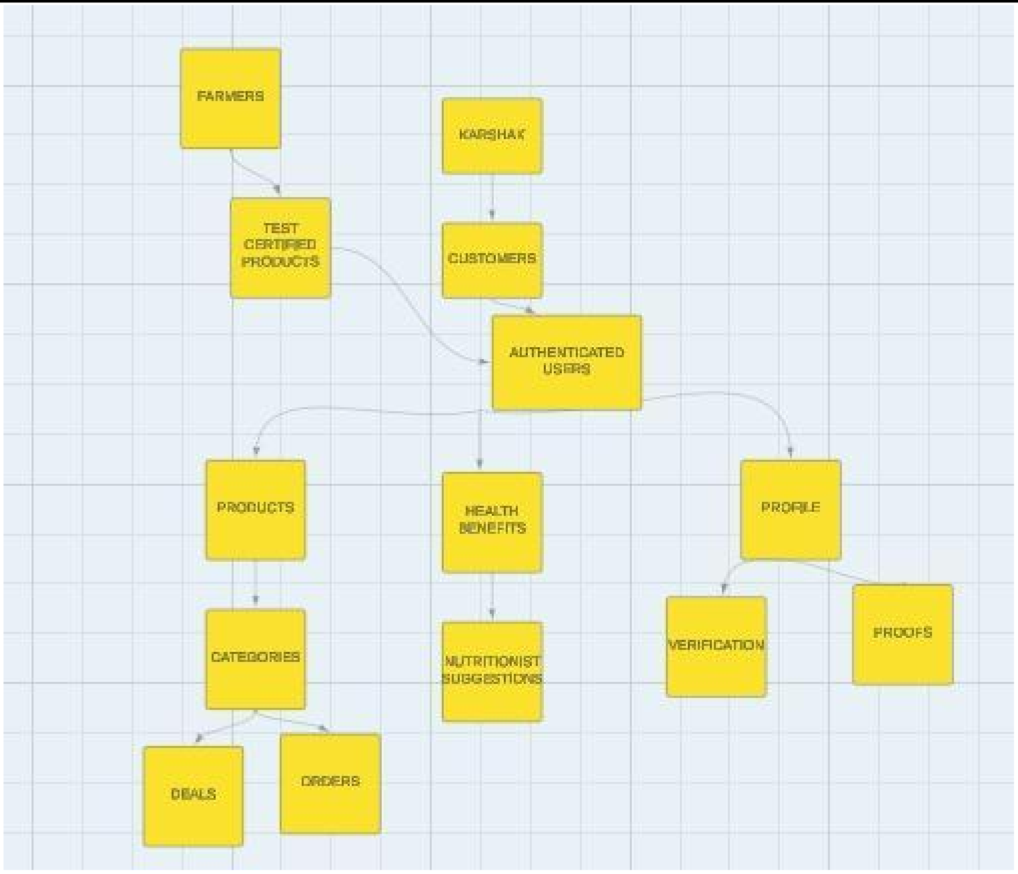


Fig 6.1

6.2 GOAL:

The Primary goals in the design of the UML are as follows:

- The primary goal of the UML diagram for the "Karshak" application is to visually represent the system's architecture, components, and interactions in a clear and organized manner. This diagram serves as a blueprint for developers, stakeholders, and other involved parties to understand the structure and functionality of the application. It helps in communicating design decisions, identifying potential issues, and ensuring alignment with project objectives. Ultimately, the UML diagram supports the development process by providing a comprehensive overview of the "Karshak" application's design and implementation.

6.3 USE CASE DIAGRAM:

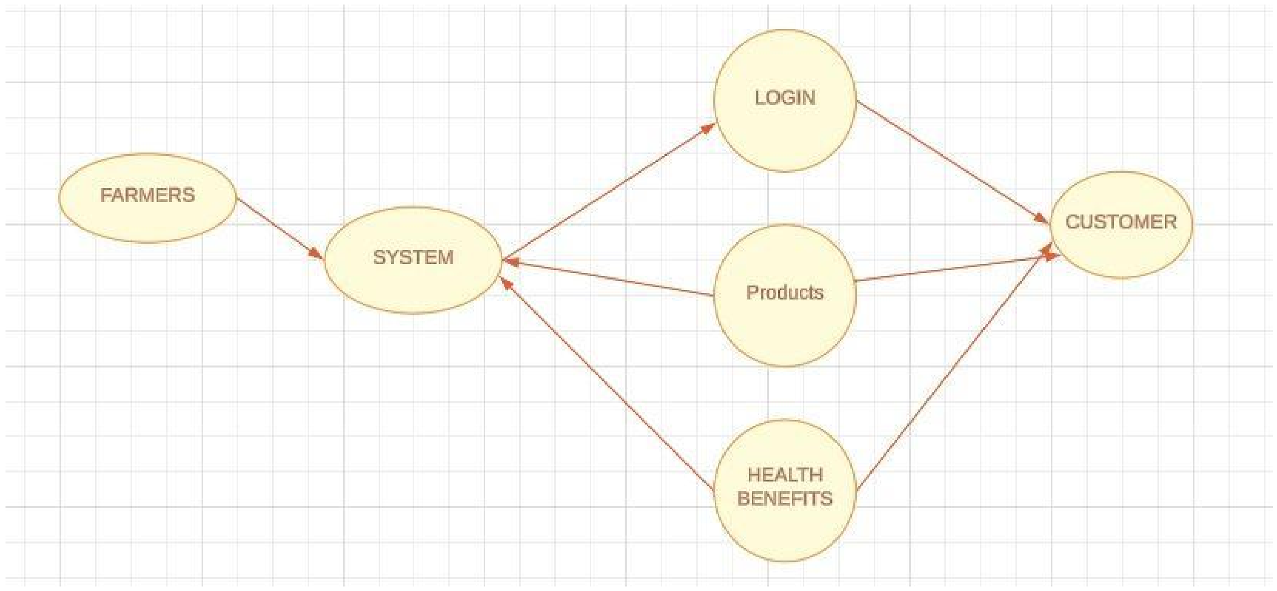


Fig6.3

► A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis.

► Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.

► The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

6.4 DEPLOYMENT DIAGRAM:

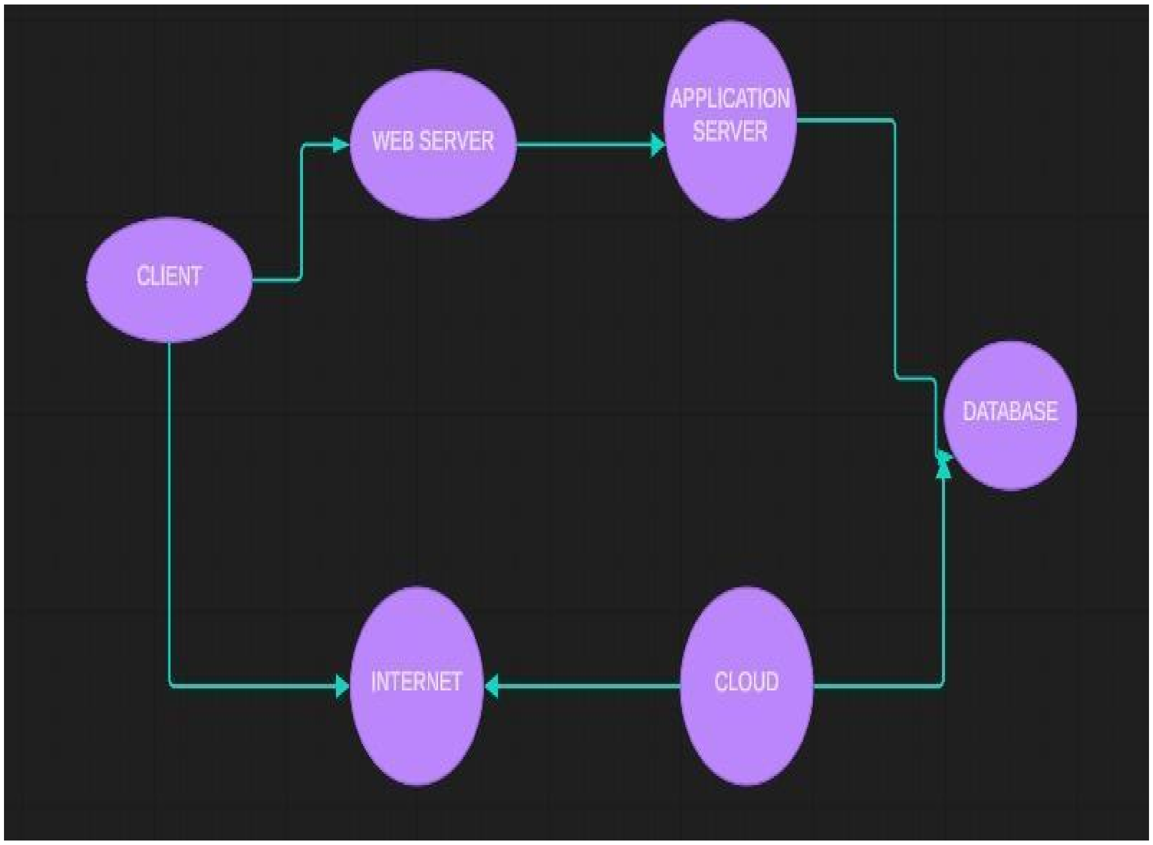


Fig 6.4. Deployment diagram

A deployment diagram for the "Karshak" application illustrates the distribution of software components across different nodes or hardware devices in a networked environment. Here's how it could be structured:

Client Devices:

- **Represents various client devices such as smartphones, tablets, or desktop computers used by farmers and other stakeholders to access the "Karshak" application.**

Web Server Node:

- **Hosts the web server software responsible for serving the frontend components of the application to client devices.**
- **Utilizes technologies like Django (Python) for backend logic and routing.**

Application Server Node:

- **Hosts the application server software responsible for executing the backend logic of the "Karshak" application.**
- **Runs the Django framework to handle requests from client devices, process data, and communicate with external services.**
- **May include components for user authentication, data processing, and business logic implementation.**

Database Server:

- **Manages the storage and retrieval of data used by the "Karshak" application.**
- **Utilizes a database management system (DBMS) such as PostgreSQL or SQLite to store information related to users, agricultural data, government schemes, etc.**
- **Ensures data integrity, security, and scalability.**

External Services:

- Represents any external services or APIs accessed by the application for additional functionality or data integration.
- Examples include weather APIs for fetching real-time weather data, translation APIs for multilingual support, or government APIs for accessing agricultural schemes.

Cloud Hosting Platform:

- Provides the infrastructure and resources necessary for hosting and deploying the "Karshak" application.
- Utilizes platforms like Heroku, AWS (Amazon Web Services), or Azure for cloud-based deployment.
- Offers scalability, reliability, and accessibility for the application.

Internet Connection:

- Represents the network connectivity required for communication between client devices and the backend servers.
- Ensures data transmission and exchange between the application components and external services.

This deployment diagram illustrates how the "Karshak" application is distributed across different nodes and components within a networked environment, enabling farmers and stakeholders to access essential agricultural information and resources seamlessly.

6.5 ACTIVITY DIAGRAM:

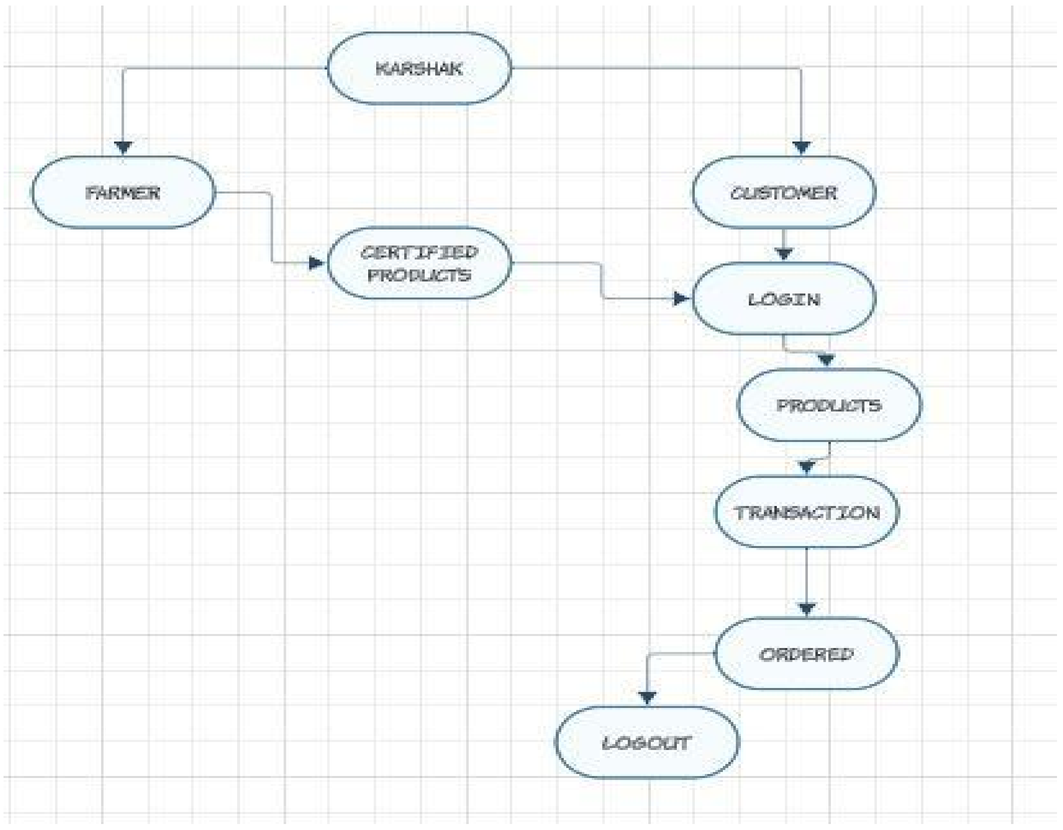


Fig6.5. Activity diagrams

An activity diagram for the "Karshak" application could depict the flow of activities and interactions within the system. Here's an outline of how it could be structured:

User Registration and Login:

- Start: User accesses the registration or login screen.
- Action: User enters registration details (if new user) or login credentials (if existing user).
- Decision: System checks if the user is registered or not.
- If new user:
    - Action: User submits registration form.
    - Action: System validates registration details.
    - Action: System creates a new user account.
- If existing user:
    - Action: User submits login credentials.
    - Action: System verifies login information.
    - Decision: If login successful, user proceeds to the dashboard.
    - If login fails, user is prompted to re-enter credentials or register.

Dashboard Navigation:

- Start: User accesses the dashboard after successful login.
- Action: User views available options on the dashboard.
- Decision: User selects a specific feature or functionality.
- Action: System navigates the user to the selected feature.
- Loop: Users can perform multiple actions within the dashboard.

Accessing Agricultural Information:

- Start: User selects the "Agricultural Information" option from the dashboard.
- Action: System retrieves relevant agricultural data from the database.
- Action: User views the information displayed on the screen.
- Loop: Users can navigate through different categories of agricultural information.
- End: User exits the agricultural information section.

Interacting with Government Schemes:

- Start: User selects the "Government Schemes" option from the dashboard.
- Action: System fetches a list of available government schemes.
- Action: User selects a specific scheme to view details.
- Action: System displays information about the selected scheme.
- Decision: Users can choose to apply for the scheme or learn more details.
- End: User completes interaction with government schemes.

Language Translation Feature:

- Start: User accesses the language translation feature within the application.
- Action: User selects the source and target languages.
- Action: System translates the text from the source language to the target language.
- Action: Translated text is displayed to the user.
- End: User completes translation activity.

Logout:

- Start: User selects the "Logout" option from the dashboard.
- Action: System logs out the user and clears session data.
- End: User is redirected to the login screen.

This activity diagram provides a visual representation of the various activities and interactions that users can perform within the "Karshak" application, guiding them through different features and functionalities available in the system.

6.6 COMPONENT DIAGRAM:

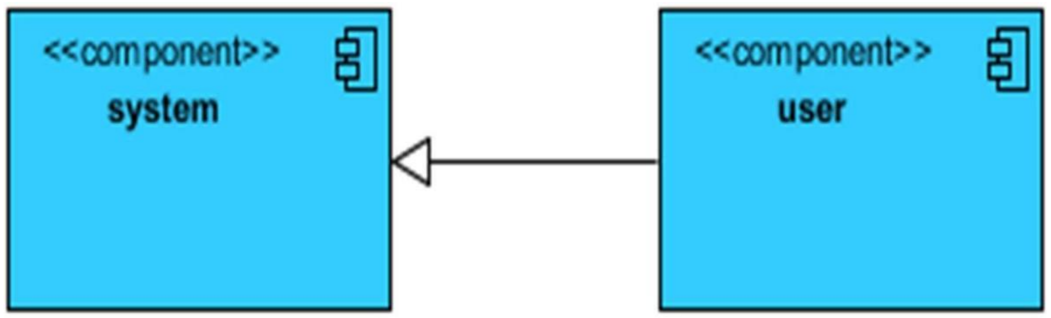


Fig6.6. Component diagram

A component diagram, also known as a UML component diagram, describes the organization and wiring of the physical **c**omponents in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required function is covered by planned development.

6.7 ER DIAGRAM:

An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of the E-R model are: entity set and relationship set.

An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in a database, so by showing relationships among tables and their attributes, ER diagrams show the complete logical structure of a database. Let's have a look at a simple ER diagram to understand this concept.

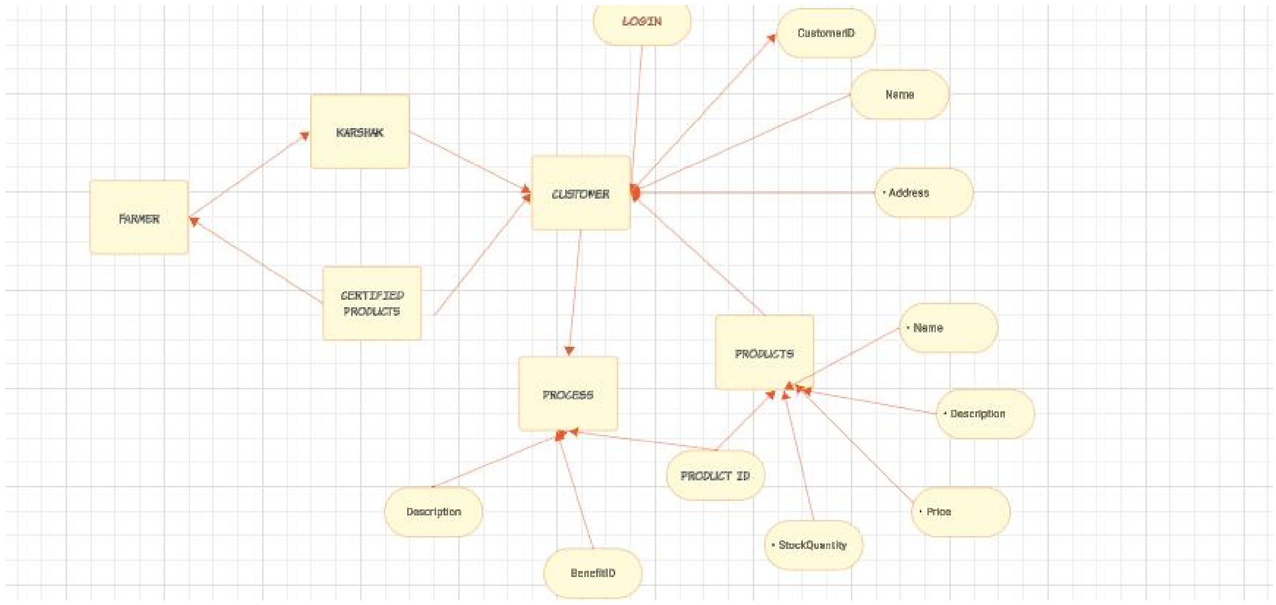


Fig6.7. ER Diagram

Entities:

Farmers

Customer

Product

Health Benefits

Relationships:

Customer - Product (Many-to-Many):

- Each customer can purchase multiple products.
- Each product can be purchased by multiple customers.

Product - Health Benefits (One-to-Many):

- Each product can have multiple health benefits.
- Each health benefit is associated with one product.

Attributes:

Customer:

- CustomerID (Primary Key)
- Name
- Email
- Address

Product:

- ProductID (Primary Key)
- Name
- Description
- Price
- StockQuantity

Health Benefits:

- BenefitID (Primary Key)
- Description
- ProductID (Foreign Key)

This simplified ER diagram illustrates the relationships between customers, products, and health benefits in the "Karshak" application, focusing specifically on customer interactions

6.8 DFD DIAGRAM:

Data flow diagram (DFD) for the customer side of the "Karshak" application:

- Customer Interface: This is the interface through which customers interact with the application.
- Customer Operations: This component includes all the actions and operations that a customer can perform within the application, such as browsing products, making purchases, viewing health benefits, etc.
- Karshak Application Logic / Backend: This represents the backend logic of the Karshak application, including processing customer requests, handling business logic, and interacting with the database.
- Database / Data Storage System: This component stores all the data related to customers, products, health benefits, orders, etc. It manages the persistent storage of information required by the application.

This diagram outlines the flow of data from the customer interface through various components of the Karshak application, ultimately leading to data storage and retrieval from the database.

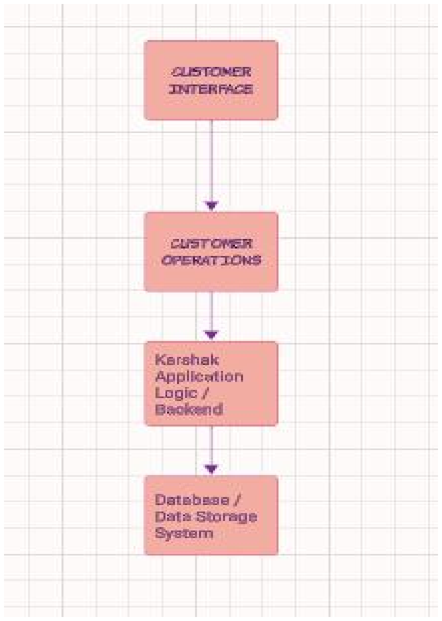

CUSTOMER INTERFACE
CUSTOMER OPERATIONS
Karshak Application Logic / Backend
Database / Data Storage System

Fig 6.8

CHAPTER 7

CODE, RESULTS AND TEST CASE

7.1 CODE:

Base.html:

```
<!DOCTYPE html>
{% load static %}
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet"
integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
crossorigin="anonymous">
  <link rel="stylesheet" href="{% static 'karshakapp/css/owl.carousel.min.css' %}" />
  <link rel="stylesheet" href="{% static 'karshakapp/css/all.min.css' %}" />
  <link rel="stylesheet" href="{% static 'karshakapp/css/style.css' %}" />
  <script src="https://checkout.razorpay.com/v1/checkout.js"></script>
  <title>farmers | {% block title %} {% endblock title %}</title>
</head>
<body>
 <nav class="navbar navbar-expand-lg navbar-light bg-info  sticky-top ">
    <div class="container-fluid mb-15">
      <a class="navbar-brand" href="#">
      <img src=" {% static "karshakapp/images/logo1.png" %} "  width=90px height=60px />
      <img src= "{% static "karshakapp/images/karshaklogo1.png" %}" width=140px
height=70px/></a>
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-
label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarSupportedContent">
        <ul class="navbar-nav me-auto mb-2 mb-lg-0">
          {% if request.user.is_authenticated %}
          <li class="nav-item">
            <a class="nav-link text-white" href="{% url 'home' %}" tabindex="-1" aria-
disabled="true">Home</a>
          </li>
          <li class="nav-item dropdown">
          <a class="nav-link dropdown-toggle text-white" href="#" id="navbarDropdown" role="button"
data-bs-toggle="dropdown" aria-expanded="false">
           Products
          </a>
          <ul class="dropdown-menu" aria-labelledby="navbarDropdown">
           <li><a class="dropdown-item" href="{% url 'category' 'CR' %}">Crops</a></li>
           <li><a class="dropdown-item" href="{% url 'category' 'VE' %}">Vegetables</a></li>
           <li><a class="dropdown-item" href="{% url 'category' 'IN' %}">Ingredients</a></li>
           <li><a class="dropdown-item" href="{% url 'category' 'OI' %}">Oils</a></li>
           <li><a class="dropdown-item" href="{% url 'category' 'FL' %}">Flours</a></li>
           <li><a class="dropdown-item" href="{% url 'category' 'NU' %}">Nuts</a></li>
           <li><a class="dropdown-item" href="{% url 'category' 'GR' %}">Grains</a></li>
           <li><a class="dropdown-item" href="{% url 'category' 'HO' %}">Honeys</a></li>
           <li><a class="dropdown-item" href="{% url 'category' 'PU' %}">Pulses</a></li>
           <li><a class="dropdown-item" href="{% url 'category' 'FR' %}">Fruits</a></li>
           <li><a class="dropdown-item" href="{% url 'category' 'NG' %}">Non veg</a></li>
           <li><a class="dropdown-item" href="{% url 'category' 'GH' %}">Ghees</a></li>
          </ul>
         </li>
         <li class="nav-item">
          <a class="nav-link text-white" href="{% url 'about' %}" tabindex="-1" aria-
disabled="true">About Us</a>
         </li>
        </li>
       <li class="nav-item">
       <a class="nav-link  text-white" href="{% url 'contact' %}"  tabindex="-1" aria-
disabled="true">Contact Us</a>
       </li>

      </div>
      <div>
        <form class="d-flex" role="search" action="/search">
         <input class="form-control me-2" type="search" placeholder="Search"  name="search" aria-
label="Search">

         <button class="btn btn-outline-light" type="submit">Search</button>
        </form>
       </div>
       <div>
        <ul class="navbar-nav me-auto mb-2 mb-lg-0">
       <li class="nav-item dropdown">
       <a class="nav-link dropdown-toggle text-white" href="#" id="profileDropdown" role="button"
```

```
data-bs-toggle="dropdown" aria-expanded="false">
            {{ request.user }}
          </a>
          <ul class="dropdown-menu " aria-labelledby="profileDropdown">
            <li><a class="dropdown-item text-dark" href="{% url 'profile' %}">Profile</a></li>
            <li><a class="dropdown-item text-dark" href="{% url 'orders' %}">Orders</a></li>
            <li><a class="dropdown-item text-dark" href="{% url 'passwordchange' %}">Change
Password</a></li>
            <li><a class="dropdown-item text-dark" href="{% url 'logout' %}">Logout</a></li>



         </ul>
        </li>
        <li class="nav-item ex-2">
          <a href="{% url 'showcart' %}" class="nav-link text-white"><span class="badge bg-success">{% if
totalitem > 0 %}{{totalitem}}{% endif %}</span>Cart</a>
        </li>
        <li class="nav-item ex-2">
          <a href="{% url 'wishlist' %}" class="nav-link text-white"><span class="badge bg-warning">{% if
wishitem > 0 %}{{wishitem}}{% endif %}</span>  <i class="fas fa-heart fa-lg"></i></a>
        </li>
        {% else %}
        <li class="nav-item ex-2">
          <a href="{% url 'login' %}" class="nav-link text-white">Login</a>
        </li>
        <li class="nav-item ex-2">
          <a href="{% url 'customerregistration' %}" class="nav-link text-white">Registrations</a>
        </li>
        {% endif %}
      </ul>
        </div>
      </div>
      </div>
      </nav>
      {% block banner_slider  %}{% endblock banner_slider %}
      {% block information %}{% endblock information %}
      {% block main_content %}{% endblock main_content %}
      {% block payment-gateway %}{% endblock payment-gateway %}

      <div class="container-fluid d-flex flex-row justify-content-center bg-info pb-0 text-center text-white
fixed-bottom">
        <div>
      <img src=" {% static "karshakapp/images/logo1.png" %} "  width=90px height=60px />
      </div>
      <div>
        <h6 class="pt-1">karshak.com</h6>
        <p >
        India
        </p>
       </div>
    </div>

    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>


        <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.8/dist/umd/popper.min.js"
integrity="sha384-I7E8VVD/ismYTF4hNIPjVp/Zjvgyol6VFvRkX/vR+Vc4jQkC+hVqc2pM8ODewa9r"
crossorigin="anonymous"></script>
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.min.js"
integrity="sha384-0pUGZvbkm6XF6gxjEnlmuGrJXVbNuzT9qBBavbLwCsOGabYfZo0T0to5eqruptLy"
crossorigin="anonymous"></script>
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
      <script src="{% static 'karshakapp/js/owl.carousel.min.js' %}"></script>
      <script src="{% static 'karshakapp/js/all.min.js' %}"></script>
      <script src="{% static 'karshakapp/js/myscript.js' %}"></script>
</body>
</html>
```

Home.html:

```
{% extends 'karshakapp/base.html' %}
{% load static %}
{% block title %}Home{% endblock title %}

{% block banner_slider %}
<div id="carouselExampleInterval" class="carousel slide" data-bs-ride="carousel">
  <div class="carousel-inner">
    <div class="carousel-item active" data-bs-interval="10000">
      <img class="d-block w-100" src="{% static "karshakapp/images/banner/b.jpg" %}" alt="F1 slide"
style="height: 600px;">
  </div>
    <div class="carousel-item" data-bs-interval="2000">
      <img class="d-block w-100" src="{% static "karshakapp/images/banner/b1.png" %}" alt="Second
slide" style="height: 600px;">
  </div>
```

```
     <div class="carousel-item">
       <img class="d-block w-100" src="{% static "karshakapp/images/banner/b2.jpg" %}" alt="Second
slide" style="height: 600px;">
    </div>
    <div class="carousel-item">
       <img class="d-block w-100" src="{% static "karshakapp/images/banner/b3.jpg" %}" alt="Second
slide" style="height: 600px;">
    </div>
    <div class="carousel-item">
       <img class="d-block w-100" src="{% static "karshakapp/images/banner/b4.jpg" %}" alt="Second
slide" style="height: 600px;">
    </div>
    <div class="carousel-item">
       <img class="d-block w-100" src="{% static "karshakapp/images/banner/b5.jpg" %}" alt="Second
slide" style="height: 600px;">
    </div>
    <div class="carousel-item">
       <img class="d-block w-100" src="{% static "karshakapp/images/banner/b6.png" %}" alt="Second
slide" style="height: 600px;">
    </div>
    <div class="carousel-item">
       <img class="d-block w-100" src="{% static "karshakapp/images/banner/b7.jpg" %}" alt="Second
slide" style="height: 600px;">
    </div>
    </div>
    <button class="carousel-control-prev" type="button" data-bs-target="#carouselExampleInterval" data-
bs-slide="prev">
     <span class="carousel-control-prev-icon" aria-hidden="true"></span>
     <span class="visually-hidden">Previous</span>
    </button>
    <button class="carousel-control-next" type="button" data-bs-target="#carouselExampleInterval" data-
bs-slide="next">
     <span class="carousel-control-next-icon" aria-hidden="true"></span>
     <span class="visually-hidden">Next</span>
    </button>
  </div>
{% endblock banner_slider %}
{% block information %}
<div class="container">
    <div class="row bg-info text-center p-5 text-white border-bottom shadow">
    <h1 class="text-center mb-4" style="margin: 0 auto;"> Fresh, Healthy, Organic, Delicious
Products</h1>
```

```html
    <p class="fs-5">
    Say no to harmful chemicals and go fully organic with our range of fresh fruits and veggies. Pamper your body and your senses with the true and unadulterated gifts from mother nature. with the true and unadulterated gifts from mother nature.

    The Fresh Food group provides fresh-cut fruits and vegetables directly picked from our partner farms and farm houses so that you always get them tree to plate.</p>
    </div>
    </div>
    {% endblock information %}
{% block main_content %}
<div class="explore-menu-section pt-5 pb-5" id="exploreMenuSection">
    <div class="container">
        <div class="row">
            <div class="col-12">
            </div>
            <div class="col-12 col-md-6 col-lg-3">
                <div class="shadow menu-item-card p-3 mb-3">
                    <img src="{% static "karshakapp/images/products/p1.jpg" %}" class="menu-item-image w-100" />
                    <h1 class="menu-card-title">Crops</h1>
                    <a href="{% url 'category' 'CR' %}" class="menu-item-link">
                        View All
                        <svg width="16px" height="16px" viewBox="0 0 16 16" class="bi bi-arrow-right-short" fill="#d0b200" xmlns="http://www.w3.org/2000/svg">
                            <path fill-rule="evenodd" d="M4 8a.5.5 0 0 1 .5-.5h5.793L8.146 5.354a.5.5 0 1 1 .708-.708l3 3a.5.5 0 0 1 0 .708l-3 3a.5.5 0 0 1-.708-.708L10.293 8.5H4.5A.5.5 0 0 1 4 8z" />
                        </svg>
                    </a>
                </div>
            </div>
            <div class="col-12 col-md-6 col-lg-3">
                <div class="shadow menu-item-card p-3 mb-3">
                    <img src="{% static "karshakapp/images/products/p2.jpg" %}" class="menu-item-image w-100" />
                    <h1 class="menu-card-title">Vegetables</h1>
                    <a href="{% url 'category' 'VE' %}" class="menu-item-link">
                        View All
                        <svg width="16px" height="16px" viewBox="0 0 16 16" class="bi bi-arrow-right" fill="#d0b200" xmlns="http://www.w3.org/2000/svg">
                            <path fill-rule="evenodd" d="M4 8a.5.5 0 0 1 .5-.5h5.793L8.146 5.354a.5.5 0 1 1 .708-.708l3 3a.5.5 0 0 1 0 .708l-3 3a.5.5 0 0 1-.708-.708L10.293 8.5H4.5A.5.5 0 0 1 4 8z" />
            </svg>
          </a>
        </div>
      </div>
      <div class="col-12 col-md-6 col-lg-3">
        <div class="menu-item-card shadow p-3 mb-3">
          <img src="{% static "karshakapp/images/products/p4.jpeg" %}" class="menu-item-image w-
100" />
          <h1 class="menu-card-title">Ingredients</h1>
          <a href="{% url 'category' 'IN' %}" class="menu-item-link">
            View All
            <svg width="16px" height="16px" viewBox="0 0 16 16" class="bi bi-arrow-right"
fill="#d0b200" xmlns="http://www.w3.org/2000/svg">
              <path fill-rule="evenodd" d="M4 8a.5.5 0 0 1 .5-.5h5.793L8.146 5.354a.5.5 0 1 1 .708-
.708l3 3a.5.5 0 0 1 0 .708l-3 3a.5.5 0 0 1-.708-.708L10.293 8.5H4.5A.5.5 0 0 1 4 8z" />
            </svg>
          </a>
        </div>
      </div>
      <div class="col-12 col-md-6 col-lg-3">
        <div class="menu-item-card shadow p-3 mb-3">
          <img src="{% static "karshakapp/images/products/oils.jpg" %}" class="menu-item-image w-
100" />
          <h1 class="menu-card-title">Oils</h1>
          <a href="{% url 'category' 'OI' %}" class="menu-item-link">
            View All
            <svg width="16px" height="16px" viewBox="0 0 16 16" class="bi bi-arrow-right"
fill="#d0b200" xmlns="http://www.w3.org/2000/svg">
              <path fill-rule="evenodd" d="M4 8a.5.5 0 0 1 .5-.5h5.793L8.146 5.354a.5.5 0 1 1 .708-
.708l3 3a.5.5 0 0 1 0 .708l-3 3a.5.5 0 0 1-.708-.708L10.293 8.5H4.5A.5.5 0 0 1 4 8z" />
            </svg>
          </a>
        </div>
      </div>
      <div class="col-12 col-md-6 col-lg-3">
        <div class="menu-item-card shadow p-3 mb-3">
          <img src="{% static "karshakapp/images/products/p6.jpg" %}" class="menu-item-image w-
100" />
          <h1 class="menu-card-title">Flours</h1>
          <a href="{% url 'category' 'FL' %}" class="menu-item-link">
            View All
            <svg width="16px" height="16px" viewBox="0 0 16 16" class="bi bi-arrow-right"
```

```
fill="#d0b200" xmlns="http://www.w3.org/2000/svg">
              <path fill-rule="evenodd" d="M4 8a.5.5 0 0 1 .5-.5h5.793L8.146 5.354a.5.5 0 1 1 .708-
.708l3 3a.5.5 0 0 1 0 .708l-3 3a.5.5 0 0 1-.708-.708L10.293 8.5H4.5A.5.5 0 0 1 4 8z" />
            </svg>
          </a>
        </div>
      </div>
      <div class="col-12 col-md-6 col-lg-3">
        <div class="menu-item-card shadow p-3 mb-3">
          <img src="{% static "karshakapp/images/products/p7.webp" %}" class="menu-item-image
w-100" />
          <h1 class="menu-card-title">Nuts</h1>
          <a href="{% url 'category' 'NU' %}" class="menu-item-link">
            View All
            <svg width="16px" height="16px" viewBox="0 0 16 16" class="bi bi-arrow-right"
fill="#d0b200" xmlns="http://www.w3.org/2000/svg">
              <path fill-rule="evenodd" d="M4 8a.5.5 0 0 1 .5-.5h5.793L8.146 5.354a.5.5 0 1 1 .708-
.708l3 3a.5.5 0 0 1 0 .708l-3 3a.5.5 0 0 1-.708-.708L10.293 8.5H4.5A.5.5 0 0 1 4 8z" />
            </svg>
          </a>
        </div>
      </div>
      <div class="col-12 col-md-6 col-lg-3">
        <div class="menu-item-card shadow p-3 mb-3">
          <img src="{% static "karshakapp/images/products/grains.jpg" %}" class="menu-item-image
w-100" />
          <h1 class="menu-card-title">Grains</h1>
          <a href="{% url 'category' 'GR' %}" class="menu-item-link">
            View All
            <svg width="16px" height="16px" viewBox="0 0 16 16" class="bi bi-arrow-right"
fill="#d0b200" xmlns="http://www.w3.org/2000/svg">
              <path fill-rule="evenodd" d="M4 8a.5.5 0 0 1 .5-.5h5.793L8.146 5.354a.5.5 0 1 1 .708-
.708l3 3a.5.5 0 0 1 0 .708l-3 3a.5.5 0 0 1-.708-.708L10.293 8.5H4.5A.5.5 0 0 1 4 8z" />
            </svg>
          </a>
        </div>
      </div>
      <div class="col-12 col-md-6 col-lg-3">
        <div class="menu-item-card shadow p-3 mb-3">
          <img src="{% static "karshakapp/images/products/honey.jpeg" %}" class="menu-item-image
w-100" />
          <h1 class="menu-card-title">Honeys</h1>
```

```html
                <a href="{% url 'category' 'HO' %}" class="menu-item-link">
                    View All
                    <svg width="16px" height="16px" viewBox="0 0 16 16" class="bi bi-arrow-right" fill="#d0b200" xmlns="http://www.w3.org/2000/svg">
                        <path fill-rule="evenodd" d="M4 8a.5.5 0 0 1 .5-.5h5.793L8.146 5.354a.5.5 0 1 1 .708-.708l3 3a.5.5 0 0 1 0 .708l-3 3a.5.5 0 0 1-.708-.708L10.293 8.5H4.5A.5.5 0 0 1 4 8z" />
                    </svg>
                </a>
            </div>
        </div>

        <div class="col-12 col-md-6 col-lg-3">
            <div class="menu-item-card shadow p-3 mb-3">
                <img src="{% static "karshakapp/images/products/Pulses.webp" %}" class="menu-item-image w-100" />
                <h1 class="menu-card-title">Pulses</h1>
                <a href="{% url 'category' 'PU' %}" class="menu-item-link">
                    View All
                    <svg width="16px" height="16px" viewBox="0 0 16 16" class="bi bi-arrow-right" fill="#d0b200" xmlns="http://www.w3.org/2000/svg">
                        <path fill-rule="evenodd" d="M4 8a.5.5 0 0 1 .5-.5h5.793L8.146 5.354a.5.5 0 1 1 .708-.708l3 3a.5.5 0 0 1 0 .708l-3 3a.5.5 0 0 1-.708-.708L10.293 8.5H4.5A.5.5 0 0 1 4 8z" />
                    </svg>
                </a>
            </div>
        </div>

        <div class="col-12 col-md-6 col-lg-3">
            <div class="menu-item-card shadow p-3 mb-3">
                <img src="{% static "karshakapp/images/products/p3.jpeg" %}" class="menu-item-image w-100" />
                <h1 class="menu-card-title">Fruits</h1>
                <a href="{% url 'category' 'FR' %}" class="menu-item-link">
                    View All
                    <svg width="16px" height="16px" viewBox="0 0 16 16" class="bi bi-arrow-right" fill="#d0b200" xmlns="http://www.w3.org/2000/svg">
                        <path fill-rule="evenodd" d="M4 8a.5.5 0 0 1 .5-.5h5.793L8.146 5.354a.5.5 0 1 1 .708-.708l3 3a.5.5 0 0 1 0 .708l-3 3a.5.5 0 0 1-.708-.708L10.293 8.5H4.5A.5.5 0 0 1 4 8z" />
                    </svg>
                </a>
            </div>
        </div>
```

```
         <div class="col-12 col-md-6 col-lg-3">
            <div class="menu-item-card shadow p-3 mb-3">
               <img src="{% static "karshakapp/images/products/p11.webp" %}" class="menu-item-image
w-100" />
               <h1 class="menu-card-title">Eggs</h1>
               <a href="{% url 'category' 'EG' %}" class="menu-item-link">
                  View All
                  <svg width="16px" height="16px" viewBox="0 0 16 16" class="bi bi-arrow-right"
fill="#d0b200" xmlns="http://www.w3.org/2000/svg">
                     <path fill-rule="evenodd" d="M4 8a.5.5 0 0 1 .5-.5h5.793L8.146 5.354a.5.5 0 1 1 .708-
.708l3 3a.5.5 0 0 1 0 .708l-3 3a.5.5 0 0 1-.708-.708L10.293 8.5H4.5A.5.5 0 0 1 4 8z" />
                  </svg>
               </a>
            </div>
         </div>

         <div class="col-12 col-md-6 col-lg-3">
            <div class="menu-item-card shadow p-3 mb-3">
               <img src="{% static "karshakapp/images/products/p10.jpeg" %}" class="menu-item-image
w-100" />
               <h1 class="menu-card-title">Ghees</h1>
               <a href="{% url 'category' 'GH' %}" class="menu-item-link">
                  View All
                  <svg width="16px" height="16px" viewBox="0 0 16 16" class="bi bi-arrow-right"
fill="#d0b200" xmlns="http://www.w3.org/2000/svg">
                     <path fill-rule="evenodd" d="M4 8a.5.5 0 0 1 .5-.5h5.793L8.146 5.354a.5.5 0 1 1 .708-
.708l3 3a.5.5 0 0 1 0 .708l-3 3a.5.5 0 0 1-.708-.708L10.293 8.5H4.5A.5.5 0 0 1 4 8z" />
                  </svg>
               </a>
            </div>
         </div>
      </div>
   </div>
</div>
{% endblock main_content %}
```

About.html:

```
{% extends 'karshakapp/base.html' %}
{% load static %}
{% block title %}About Us{% endblock title %}

{% block banner_slider %}
<div id="carouselExampleInterval" class="carousel slide" data-bs-ride="carousel">
   <div class="carousel-inner">
     <div class="carousel-item active" data-bs-interval="10000">
       <img class="d-block w-100" src="{% static "karshakapp/images/banner/b.jpg" %}" alt="F1
slide" style="height: 600px;">
   </div>
     <div class="carousel-item" data-bs-interval="2000">
       <img class="d-block w-100" src="{% static "karshakapp/images/banner/b1.png" %}"
alt="Second slide" style="height: 600px;">
   </div>
     <div class="carousel-item">
       <img class="d-block w-100" src="{% static "karshakapp/images/banner/b2.jpg" %}"
alt="Second slide" style="height: 600px;">
   </div>
   <div class="carousel-item">
       <img class="d-block w-100" src="{% static "karshakapp/images/banner/b3.jpg" %}"
alt="Second slide" style="height: 600px;">
   </div>
   <div class="carousel-item">
       <img class="d-block w-100" src="{% static "karshakapp/images/banner/b4.jpg" %}"
alt="Second slide" style="height: 600px;">
   </div>
   <div class="carousel-item">
       <img class="d-block w-100" src="{% static "karshakapp/images/banner/b5.jpg" %}"
alt="Second slide" style="height: 600px;">
   </div>
   <div class="carousel-item">
       <img class="d-block w-100" src="{% static "karshakapp/images/banner/b6.png" %}"
alt="Second slide" style="height: 600px;">
   </div>
   <div class="carousel-item">
       <img class="d-block w-100" src="{% static "karshakapp/images/banner/b7.jpg" %}"
alt="Second slide" style="height: 600px;">
   </div>
   </div>
   <button class="carousel-control-prev" type="button" data-bs-target="#carouselExampleInterval"
data-bs-slide="prev">
```

```
    <span class="carousel-control-prev-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Previous</span>
  </button>
  <button class="carousel-control-next" type="button" data-bs-target="#carouselExampleInterval"
data-bs-slide="next">
    <span class="carousel-control-next-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Next</span>
  </button>
 </div>
{% endblock banner_slider %}
{% block information %}
<div class="container">
  <div class="row bg-light text-center  p-5 text-info mt-5 border-bottom shadow">
  <h1 class="text-center mb-4" style="margin: 0 auto;"> Fresh, Healthy, Organic, Delicious
Products</h1>
  <p class="fs-5">
  Say no to harmful chemicals and go fully organic with our range of fresh fruits and veggies. Pamper
your body and your senses with the true and unadulterated gifts from mother nature. with the true and
unadulterated gifts from mother nature.

  The Fresh Food group provides fresh-cut fruits and vegetables directly picked from our partner farms
and farm houses so that you always get them tree to plate.</p>
  <h1 class="text-center mb-4 mt-5" style="margin: 0 auto;"> Mission</h1>
  <p>To embrace the right technology to delight our Customers Empowering the farmer community
through our unique 'Relationship Farming' Model To anticipate, understand and respond to our
Customers' needs by creating high-quality products and making them available through innovative and
convenient channels</p>
  </div>

  <div class="wcu-section pt-5 pb-5" id="wcuSection">
    <div class="container">
      <div class="row">
        <div class="col-12">
          <h1 class="wcu-section-heading">Why Choose Us?</h1>
          <p class="wcu-section-description">
            Organic products directly from farmers to your lovely home.
          </p>
        </div>
        <div class="col-12 col-md-4">
          <div class="wcu-card p-3 mb-3">
            <img src="https://d1tgh8fmlzexmh.cloudfront.net/ccbp-responsive-website/food-
```

```
serve.png" class="wcu-card-image" />
            <h1 class="wcu-card-title mt-3">Item Service</h1>
            <p class="wcu-card-description">
              Experience fine dining at the comfort of your home. All our
              orders are carefully packed and arranged to give you the nothing less than perfect.
            </p>
          </div>
        </div>
        <div class="col-12 col-md-4">
          <div class="wcu-card p-3 mb-3">
            <img src="https://d1tgh8fmlzexmh.cloudfront.net/ccbp-responsive-website/fruits-
img.png" class="wcu-card-image" />
            <h1 class="wcu-card-title mt-3">Fresh Items</h1>
            <p class="wcu-card-description">
              The Fresh Food group provides fresh-cut fruits and vegetables
              directly picked from our partner farms and farm houses so that
              you always get them tree to plate.
            </p>
          </div>
        </div>
        <div class="col-12 col-md-4">
          <div class="wcu-card p-3 mb-3">
            <img src="https://d1tgh8fmlzexmh.cloudfront.net/ccbp-responsive-website/offers-
img.png" class="wcu-card-image" />
            <h1 class="wcu-card-title mt-3">SPecial Offers</h1>
            <p class="wcu-card-description">
              Food Coupons & Offers upto
              <span class="offers">50% OFF</span>
              and Exclusive Promo Codes on All Online Food Orders.
            </p>
          </div>
        </div>
      </div>
    </div>
  </div>
  </div>


  {% endblock information %}
```

Contact.html:

```
{% extends 'karshakapp/base.html' %}
{% load static %}
{% block title %}Contact Us{% endblock title %}

{% block banner_slider %}
<div id="carouselExampleInterval" class="carousel slide" data-bs-ride="carousel">
  <div class="carousel-inner">
    <div class="carousel-item active" data-bs-interval="10000">
      <img class="d-block w-100" src="{% static "karshakapp/images/banner/b.jpg" %}" alt="F1
slide" style="height: 600px;">
  </div>
    <div class="carousel-item" data-bs-interval="2000">
      <img class="d-block w-100" src="{% static "karshakapp/images/banner/b1.png" %}"
alt="Second slide" style="height: 600px;">
  </div>
    <div class="carousel-item">
      <img class="d-block w-100" src="{% static "karshakapp/images/banner/b2.jpg" %}"
alt="Second slide" style="height: 600px;">
  </div>
  <div class="carousel-item">
      <img class="d-block w-100" src="{% static "karshakapp/images/banner/b3.jpg" %}"
alt="Second slide" style="height: 600px;">
  </div>
  <div class="carousel-item">
      <img class="d-block w-100" src="{% static "karshakapp/images/banner/b4.jpg" %}"
alt="Second slide" style="height: 600px;">
  </div>
  <div class="carousel-item">
      <img class="d-block w-100" src="{% static "karshakapp/images/banner/b5.jpg" %}"
alt="Second slide" style="height: 600px;">
  </div>
  <div class="carousel-item">
      <img class="d-block w-100" src="{% static "karshakapp/images/banner/b6.png" %}"
alt="Second slide" style="height: 600px;">
  </div>
  <div class="carousel-item">
      <img class="d-block w-100" src="{% static "karshakapp/images/banner/b7.jpg" %}"
alt="Second slide" style="height: 600px;">
  </div>
  </div>
  <button class="carousel-control-prev" type="button" data-bs-target="#carouselExampleInterval"
```

```
data-bs-slide="prev">
    <span class="carousel-control-prev-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Previous</span>
  </button>
  <button class="carousel-control-next" type="button" data-bs-target="#carouselExampleInterval"
data-bs-slide="next">
    <span class="carousel-control-next-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Next</span>
  </button>
 </div>
{% endblock banner_slider %}
{% block information %}
<div class="container">
  <div class="row bg-info text-center p-5 text-white border-bottom shadow">
  <h1 class="text-center mb-4" style="margin: 0 auto;"> Fresh, Healthy, Organic, Delicious
Products</h1>
  <p class="fs-5">
  Say no to harmful chemicals and go fully organic with our range of fresh fruits and veggies. Pamper
your body and your senses with the true and unadulterated gifts from mother nature. with the true and
unadulterated gifts from mother nature.

  The Fresh Food group provides fresh-cut fruits and vegetables directly picked from our partner farms
and farm houses so that you always get them tree to plate.</p>
  </div>
  </div>
  {% endblock information %}

  {% block main_content %}

    <div class="container mt-3">
      <div class="text-center">
        <div class="row container">

        <div class="col-md-12">
                <div class="shadow menu-item-card card-center p-3 mb-3">
                    <img src=" {% static "karshakapp/images/logo1.png" %} "  width=90px height=60px
/>
                    <h3>Karshak is farmers market place here we directly purchase products from
Customers.</h3>
                    <h5>Address: Karshak Farmers Market Place, Vaddeswaram,
                        Guntur District, A.P., INDIA.
                    Pincode : 522 302.</h5>
                    <h5>Contact NO.: 1234567890</h5>
                    <h5>Email:karshak@gmail.com</h5>
                  </div>



            <div class="col-md-12">
            <div>
              <h1 class="follow-us-section-heading">Follow Us</h1>
            </div>
            <div>
              <div class="d-flex flex-row justify-content-center">
                <div class="p-3">
                <img src=" {% static "karshakapp/images/banner/fb.webp" %} "  width=90px
height=60px />
                </div>
                <div class="p-3">
                  <img src=" {% static "karshakapp/images/banner/Insta.png" %} "  width=90px
height=60px />
                </div>
                <div class="p-3">
                  <img src=" {% static "karshakapp/images/banner/tw.ico" %} "  width=90px
height=60px />
                </div>
              </div>
            </div>
            </div>
          </div>
        </div>



  <br><br><br><br>

  </div>{% endblock main_content %}
```

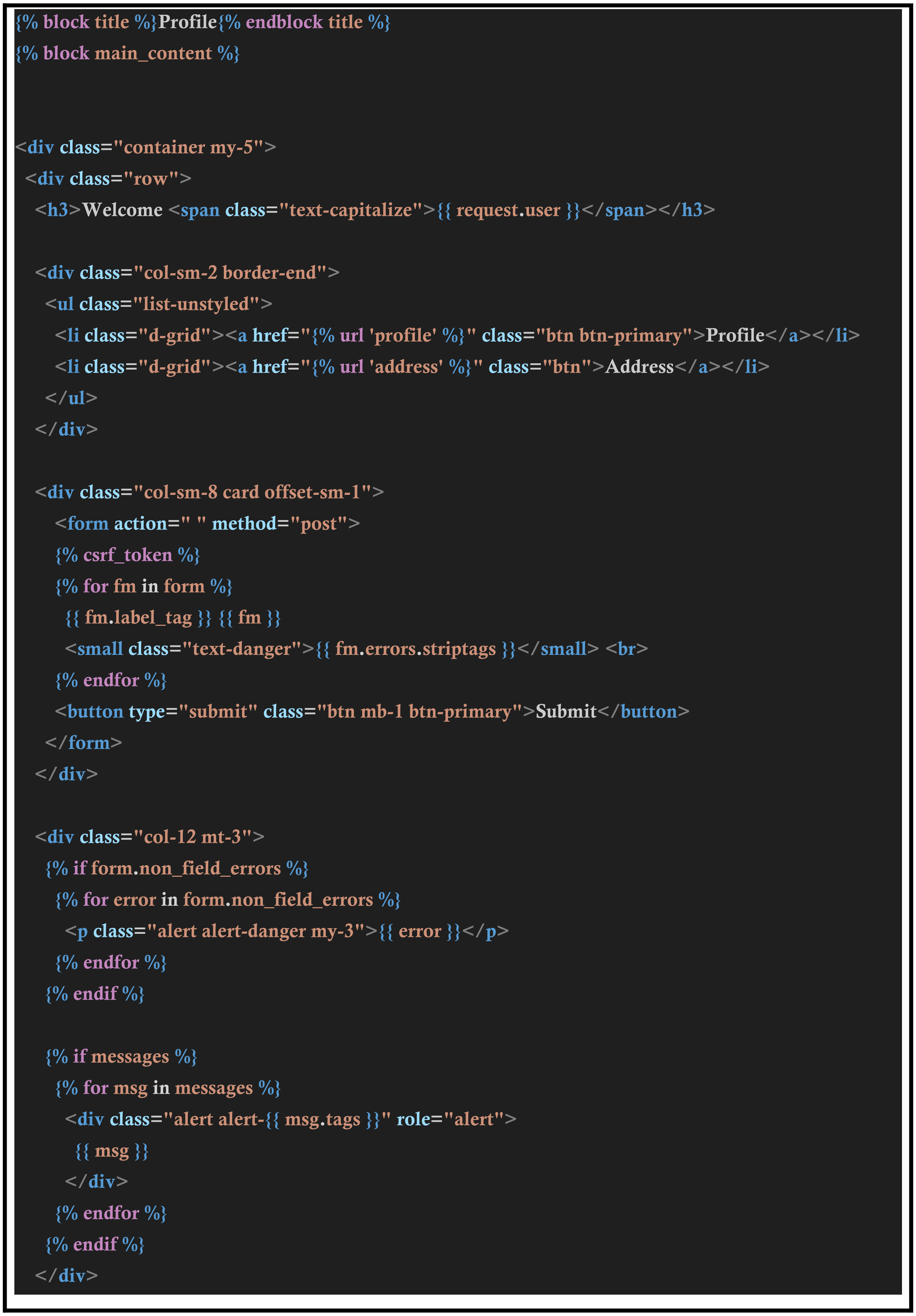
Products.html:

```
{% extends 'karshakapp/base.html' %}
{% load static %}
{% block title %}productdetail{% endblock title %}

{% block main_content %}
```

```
  <div class="container">
    <div class="row flex justify-content-between">
      <div class="img2 col-lg-5 mt-5">
        <img src="{{ product.product_image.url }}" class="image col-xs-6 col-sm-12 col-lg-12 mt-3
text-sm-center w-100 h-50" alt="">
      </div>
      <div class="productdetail col-lg-5 shadow menu-item-card p-4 mb-5">
        <h1 style="font-family: Georgia; font-size: 50px;">{{ product.title }}</h1>
        <h3 style="font-family: Georgia;">{{ product.description }} </h3>
        <h5>Rs.{{ product.discounted_price }}/-</h5>
        <small class="text-decoration-line-through text-muted fs-
5"><del>Rs.{{ product.selling_price }}/-</del></small>
        <br><br><h4>Product Features</h4>
        <ul class="fs-5" style="font-family: Rajdhani;">
          <li>{{ product.prodapp }}</li>
          <li>{{ product.composition }}</li>
        </ul>
        <form action="{% url 'add-to-cart' %}" method="post" class="d-inline">
          {% csrf_token %}
          <input type="hidden" name="prod_id" value="{{ product.id }}">
          <button type="submit" class="btn1 btn mb-2 btn-warning shadow px-5 py-2">Add to
Cart</button>
        </form>
        <a href="{% url 'showcart' %}" class="btn mb-2 btn-success px-5 py-2 ms-4">Buy Now</a>
        {% if wishlist %}
        <a pid="{{ product.id }}" class="minus-wishlist button-center btn btn-success shadow px-5
py-2 ms-4"><i class="fas fa-heart fa-lg"></i></a>
        {% else %}
        <a pid="{{ product.id }}" class="plus-wishlist btn btn-danger text-center shadow px-5 py-2
ms-4"><i class="fas fa-heart fa-lg"></i></a>
        {% endif %}

    </div>
  </div>

  <br><br><br>
{% endblock main_content %}
```

Profile.html:

```
{% extends 'karshakapp/base.html' %}
{% load static %}
```



```
{% block title %}Profile{% endblock title %}
{% block main_content %}


<div class="container my-5">
 <div class="row">
  <h3>Welcome <span class="text-capitalize">{{ request.user }}</span></h3>

  <div class="col-sm-2 border-end">
   <ul class="list-unstyled">
    <li class="d-grid"><a href="{% url 'profile' %}" class="btn btn-primary">Profile</a></li>
    <li class="d-grid"><a href="{% url 'address' %}" class="btn">Address</a></li>
   </ul>
  </div>

  <div class="col-sm-8 card offset-sm-1">
    <form action=" " method="post">
    {% csrf_token %}
    {% for fm in form %}
     {{ fm.label_tag }} {{ fm }}
     <small class="text-danger">{{ fm.errors.striptags }}</small> <br>
    {% endfor %}
    <button type="submit" class="btn mb-1 btn-primary">Submit</button>
   </form>
  </div>

  <div class="col-12 mt-3">
   {% if form.non_field_errors %}
    {% for error in form.non_field_errors %}
     <p class="alert alert-danger my-3">{{ error }}</p>
    {% endfor %}
   {% endif %}

   {% if messages %}
    {% for msg in messages %}
     <div class="alert alert-{{ msg.tags }}" role="alert">
      {{ msg }}
     </div>
    {% endfor %}
   {% endif %}
  </div>
 </div>
</div>
<br><br><br>
{% endblock main_content %}
```

Address.html:

```
{% extends 'karshakapp/base.html' %}
{% load static %}
{% block title %}Address{% endblock title %}
{% block main_content %}

<div class="container my-5">
   <div class="row">
      <h3 class="text-color">Welcome <span class="text-capitalize">{{ request.user }}</span></h3>

      <div class="col-sm-2 border-end">
         <ul class="list-unstyled">
            <li class="d-grid"><a href="{% url 'profile' %}" class="btn ">Profile</a></li>
            <li class="d-grid"><a href="{% url 'address' %}" class="btn btn-
primary">Address</a></li>
         </ul>
      </div>

      <div class="col-sm-8  offset-sm-1">
         <div class="row">
            {% for address in addresses %}
               <div class="col-sm-6">
                  <div class="card m-2 bg-light">
                     <div class="card-body">
                        <h3 class="text-color">Address {{ forloop.counter }}</h3>
                        <p>Name: {{ address.name }}</p>
                        <p>Locality: {{ address.locality }}</p>
                        <p>Mobile: {{ address.mobile }}</p>
                        <p>City: {{ address.city }}</p>
                        <p>State: {{ address.state }}</p>
                        <p>Pincode: {{ address.zipcode }}</p>
                        <p class="text-center">
                           <a href="{% url 'updateAddress' pk=address.id %}" class="btn btn-primary"
role="button">Update Address</a>
```

```
                </p>


              </div>
            </div>
          </div>
        {% endfor %}
      </div>
    </div>
  </div>
</div>

{% endblock main_content %}
{% extends 'karshakapp/base.html' %}
{% load static %}
{% block title %}Update Address{% endblock title %}
{% block main_content %}


<div class="container my-5">
 <div class="row">
  <h3>Welcome <span class="text-capitalize">{{ request.user }}</span></h3>

  <div class="col-sm-8 card offset-sm-1">
    <form action=" " method="post">
    {% csrf_token %}
    {% for fm in form %}
     {{ fm.label_tag }} {{ fm }}
     <small class="text-danger">{{ fm.errors.striptags }}</small> <br>
    {% endfor %}
    <div class="col-12 mt-3 mb-1">
      <button type="submit" class="btn btn-primary">Update</button>
      <a href="{% url 'address' %}" class="btn btn-danger" role="button">Back</a>
    </div>
    {% if form.non_field_errors %}
    {% for error in form.non_field_errors %}
     <p class="alert alert-danger my-3">{{ error }}</p>
    {% endfor %}
   {% endif %}
   </form>
  </div>






  </div>
 </div>
</div>
<br><br><br>
{% endblock main_content %}
```

Addtocart.html:

```
{% extends 'karshakapp/base.html' %}
{% load static %}

{% block title %}
  Add to Cart
{% endblock title %}

{% block main_content %}
<div class="container my-5">
  <div class="row">
    {% if cart %}
  <h1 class="text-center mb-5">Shopping Cart</h1>

        <div class="col-sm-8">
          <div class="card">
            <div class="card-body">
              {% for item in cart %}
                <div class="row">
                  <div class="col-sm-3 text-center align-self-center">
                    <img src="{{ item.product.product_image.url }}" alt="" srcset="" class="img-
fluid img-thumbnail shadow-sm" height="150" width="150">
                  </div>
                  <div class="col-sm-9">
                    <div>
                      <h5>{{ item.product.title }}</h5>
                      <p class="mb-2 text-muted small">{{ item.product.description }}</p>
                      <div class="my-3">
                        <label for="quantity">Quantity:</label>
                        <a class="minus-cart btn" pid="{{ item.product.id }}"><i class="fas fa-
```

```
minus-square fa-lg"></i></a>
                                            <span id="quantity">{{ item.quantity }}</span>
                                            <a class="plus-cart btn" pid="{{ item.product.id }}"><i class="fas fa-plus-
square fa-lg"></i></a>
                                        </div>
                                        <div class="d-flex justify-content-between">
                                            <a href="#" class="remove-cart btn btn-sm btn-secondary mr-3"
pid="{{ item.product.id }}">Remove item</a>
                                            <p class="mb-0"><span><strong>Rs.
{{ item.product.discounted_price }}</strong></span></p>
                                        </div>
                                    </div>
                                </div>
                            </div>
                            <hr class="text-muted">
                        {% endfor %}
                    </div>
                </div>
            </div>
            <div class="col-sm-4">
                <div class="card">
                    <div class="card-body">
                    <h3>The Total Amount of </h3>
                    <ul class="list-group">
                        <li class="list-group-item d-flex justify-content-between align-items-center border-0 px-
0 pb-0">Amount<span id="amount">Rs. {{ amount }}</span></li>
                        <li class="list-group-item d-flex justify-content-between align-items-center px-
0">Shipping<span>Rs. 40.00</span></li>
                        <li class="list-group-item d-flex justify-content-between align-items-center border-0 px-
0 mb-3">
                            <div>
                            <strong>Total</strong> <small>(including GST)</small>
                            </div>
                            <span id="totalamount"><strong>Rs. {{ totalamount }}</strong></span>
                        </li>
                    </ul>
                    <div class="d-grid">
                        <a href="{% url 'checkout' %}" class="btn btn-primary">Place Order</a>
                    </div>
                </div>
            </div>
```

```
        {% else %}
            <h1 class="text-center mb-5">Cart is Empty</h1>
        {% endif %}
    </div>
</div>

{% endblock main_content %}
```

wishlist.html:

```
{% extends 'karshakapp/base.html' %}
{% load static %}

{% block title %}
    Wishlist
{% endblock title %}

{% block main_content %}
<div class="container my-5">
    <div class="row">
        {% if wishlist %}
            <h1 class="text-center mb-5">Wishlist</h1>
            <div class="col-sm-8">
                <div class="card">
                    <div class="card-body">
                        {% for item in wishlist %}
                            <div class="row">
                                <div class="col-sm-3 text-center align-self-center">
                                    <img src="{{ item.product.product_image.url }}" alt="" srcset="" class="img-
fluid img-thumbnail shadow-sm" height="150" width="150">
                                </div>
                                <div class="col-sm-9">
                                    <div>
                                        <h5>{{ item.product.title }}</h5>
                                        <p class="mb-2 text-muted small">{{ item.product.description }}</p>
                                        <div class="my-3">
                                            <label for="quantity">Quantity:</label>
                                            <span id="quantity">{{ item.quantity }}</span>
                                        </div>

                                    </div>
                                </div>
                            </div>
                        {% endfor %}
                    </div>
                </div>
            </div>
        {% else %}
            <div class="col">
                <p>Your wishlist is empty.</p>
            </div>
        {% endif %}
    </div>
</div>
{% endblock main_content %}
```

Ajax Code:

```
from django.contrib.auth.models import Group
from django.contrib import admin
from django.urls import reverse
from .models import OrderPlaced, Payment, Product,Customer,Cart, Wishlist
from django.utils.html import format_html


@admin.register(Product)
class ProductModelAdmin(admin.ModelAdmin):
    list_display = ['id', 'title', 'discounted_price', 'get_category', 'product_image']

    def get_category(self, obj):
        return obj.category
    get_category.short_description = 'Category'

@admin.register(Customer)
class CustomerModelAdmin(admin.ModelAdmin):
    list_display = ['id', 'name','mobile', 'locality', 'city',  'zipcode', 'state']

@admin.register(Cart)
class CartModelAdmin(admin.ModelAdmin):
   list_display = ['id', 'user', 'product','quantity']
   def products(self, obj):
       link = reverse("admin:app_product_change", args=[obj.product.pk])
       return format_html('<a href="{}">{}</a>', link, obj.product.title)
```

```
@admin.register(Payment)
class PaymentModelAdmin(admin.ModelAdmin):
   list_display = ['id', 'user', 'amount', 'razorpay_order_id', 'razorpay_payment_status',
'razorpay_payment_id', 'paid']

@admin.register(OrderPlaced)
class OrderPlacedModelAdmin(admin.ModelAdmin):
   list_display = ['id', 'user', 'customers', 'products', 'quantity', 'ordered_date', 'status', 'payments']

   def customers(self, obj):
      link = reverse('admin:app_customer_change', args=[obj.customer.pk])
      return format_html('<a href="{}">{}</a>', link, obj.customer.name)

   def products(self, obj):
      link = reverse('admin:app_product_change', args=[obj.product.pk])
      return format_html('<a href="{}">{}</a>', link, obj.product.title)

   def payments(self, obj):
      link = reverse('admin:app_payment_change', args=[obj.payment.pk])
      return format_html('<a href="{}">{}</a>', link, obj.payment.razorpay_payment_id)
@admin.register(Wishlist)
class WishlistModelAdmin(admin.ModelAdmin):
   list_display = ['id','user','product']
   def products(self, obj):
      link = reverse("admin:app_product_change", args=[obj.product.pk])
      return format_html('<a href="{}">{}</a>', link, obj.product.title)

admin.site.unregister(Group)
```

CSS Code:

```
.food-munch-logo {
   width: 80px;
   height: 70px;
}

#navItem1 {
   color:#323f4b;
   font-family: "Roboto";
}

#navItem2 {
   color: #323f4b;
```

```
  font-family: "Roboto";
}

#navItem3 {
  color: #323f4b;
  font-family: "Roboto";
}

#navItem4 {
  color: #323f4b;
  font-family: "Roboto";
}

.banner-section-bg-container {
  background-image: url("https://d1tgh8fmlzexmh.cloudfront.net/ccbp-responsive-
website/foodmunch-banner-bg.png");
  height: 100vh;
  background-size: cover;
}

.banner-heading {
  color: white;
  font-family: "Roboto";
  font-size: 45px;
  font-weight: 300;
}

.banner-caption {
  color: #f5f7fa;
  font-family: "Roboto";
  font-size: 24px;
  font-weight: 300;
}


/*!
 * Font Awesome Free 5.15.1 by @fontawesome - https://fontawesome.com
 * License - https://fontawesome.com/license/free (Icons: CC BY 4.0, Fonts: SIL OFL 1.1, Code:
MIT License)
 */
.fa,.fab,.fad,.fal,.far,.fas{-moz-osx-font-smoothing:grayscale;-webkit-font-
```

```
smoothing:antialiased;display:inline-block;font-style:normal;font-variant:normal;text-
rendering:auto;line-height:1}.fa-lg{font-size:1.33333em;line-height:.75em;vertical-align:-.0667em}.fa-
xs{font-size:.75em}.fa-sm{font-size:.875em}.fa-1x{font-size:1em}.fa-2x{font-size:2em}.fa-3x{font-
size:3em}.fa-4x{font-size:4em}.fa-5x{font-size:5em}.fa-6x{font-size:6em}.fa-7x{font-size:7em}.fa-
8x{font-size:8em}.fa-9x{font-size:9em}.fa-10x{font-size:10em}.fa-fw{text-
align:center;width:1.25em}.fa-ul{list-style-type:none;margin-left:2.5em;padding-left:0}.fa-
ul>li{position:relative}.fa-li{left:-2em;position:absolute;text-align:center;width:2em;line-
height:inherit}.fa-border{border:.08em solid #eee;border-radius:.1em;padding:.2em .25em .15em}.fa-
pull-left{float:left}.fa-pull-right{float:right}.fa.fa-pull-left,.fab.fa-pull-left,.fal.fa-pull-left,.far.fa-pull-
left,.fas.fa-pull-left{margin-right:.3em}.fa.fa-pull-right,.fab.fa-pull-right,.fal.fa-pull-right,.far.fa-pull-
right,.fas.fa-pull-right{margin-left:.3em}.fa-spin{-webkit-animation:fa-spin 2s linear
infinite;animation:fa-spin 2s linear infinite}.fa-pulse{-webkit-animation:fa-spin 1s steps(8)
infinite;animation:fa-spin 1s steps(8) infinite}@-webkit-keyframes fa-spin{0%{-webkit-
transform:rotate(0deg);transform:rotate(0deg)}to{-webkit-
transform:rotate(1turn);transform:rotate(1turn)}}@keyframes fa
```

Views.py

```python
from django.conf import settings

from django.db.models import Count

from django.http import JsonResponse

from django.shortcuts import get_object_or_404, render,redirect

from django.views.generic.base import View

import razorpay

from requests import get

from . models import Customer, OrderPlaced, Product,Cart, Wishlist

from . forms import CustomerRegistrationForm ,CustomerProfileForm,MyPasswordChangeForm

from django.contrib import messages

from django.contrib.auth.decorators import login_required

from django.utils.decorators import method_decorator


@login_required

def home(request):

    totalitem = 0

    wishitem = 0
    if request.user.is_authenticated:
        totalitem = len(Cart.objects.filter(user=request.user))
        wishitem = len(Wishlist.objects.filter(user=request.user))
    return render(request,"karshakapp/home.html",locals())
@login_required
def about(request):
    totalitem = 0
    wishitem = 0
    if request.user.is_authenticated:
        totalitem = len(Cart.objects.filter(user=request.user))
        wishitem = len(Wishlist.objects.filter(user=request.user))
    return render(request,"karshakapp/about.html",locals())
@login_required
def contact(request):
    totalitem = 0
    wishitem = 0
    if request.user.is_authenticated:
        totalitem = len(Cart.objects.filter(user=request.user))
        wishitem = len(Wishlist.objects.filter(user=request.user))
    return render(request,"karshakapp/contact.html",locals())



@method_decorator(login_required, name='dispatch')
class CategoryView(View):
    def get(self, request,val):
        product = Product.objects.filter(category=val)
        title = Product.objects.filter(category=val).values('title').values('title')
        totalitem = 0
        wishitem = 0
        if request.user.is_authenticated:
            totalitem = len(Cart.objects.filter(user=request.user))
            wishitem = len(Wishlist.objects.filter(user=request.user))
        return render(request, "karshakapp/category.html",locals())


@method_decorator(login_required, name='dispatch')
class CategoryTitle(View):
    def get(self,request,val):
        product = Product.objects.filter(title=val)
        title = Product.objects.filter(category=product[0].category).values('title')
        totalitem = 0
        wishitem = 0
        if request.user.is_authenticated:
            totalitem = len(Cart.objects.filter(user=request.user))
            wishitem = len(Wishlist.objects.filter(user=request.user))
        return render(request,"karshakapp/category.html",locals())


@method_decorator(login_required, name='dispatch')
class ProductDetail(View):
        def get(self,request,pk):
            product = Product.objects.get(pk=pk)
            wishlist = Wishlist.objects.filter(Q(product=product)&Q(user=request.user))
            totalitem = 0
            wishitem = 0
            if request.user.is_authenticated:
                totalitem = len(Cart.objects.filter(user=request.user))
                wishitem = len(Wishlist.objects.filter(user=request.user))
            return render(request,"karshakapp/productdetail.html",locals())
```

```python
class CustomerRegistrationView(View):
    def get(self, request):
        form = CustomerRegistrationForm()
        return render(request, "karshakapp/customerregistration.html", locals())
    def post(self, request):
        form = CustomerRegistrationForm(request.POST)
        if form.is_valid():
            form.save()
            messages.success(request, "Congratulations! User Registered Successfully")
        else:
            messages.warning(request, "Invalid Input Data")
        return render(request, 'karshakapp/customerregistration.html', locals())


@method_decorator(login_required, name='dispatch')
class ProfileView(View):
    def get(self, request):
        form = CustomerProfileForm()
        totalitem = 0
        wishitem = 0
        if request.user.is_authenticated:
            totalitem = len(Cart.objects.filter(user=request.user))
            wishitem = len(Wishlist.objects.filter(user=request.user))
        return render(request, 'karshakapp/profile.html', locals())
    def post(self, request):
        form = CustomerProfileForm(request.POST)
        if form.is_valid():
            user = request.user
            name = form.cleaned_data['name']
            locality = form.cleaned_data['locality']

            city = form.cleaned_data['city']

            mobile = form.cleaned_data['mobile']

            state = form.cleaned_data['state']

            zipcode = form.cleaned_data['zipcode']

            reg = Customer(user=user, name=name, locality=locality, mobile=mobile, city=city, state=state, zipcode=zipcode)

            reg.save()

            messages.success(request, "Congratulations! Your profile has been saved successfully.")

            return redirect('profile')  # Redirect to the profile page

        else:

            messages.warning(request, "Invalid input data. Please correct the errors.")

        return render(request, 'karshakapp/profile.html', locals())

@login_required

def addresses(request):

    addresses = Customer.objects.filter(user=request.user)

    totalitem = 0

    wishitem = 0

    if request.user.is_authenticated:

        totalitem = len(Cart.objects.filter(user=request.user))

        wishitem = len(Wishlist.objects.filter(user=request.user))

    return render(request, 'karshakapp/address.html', locals())


@method_decorator(login_required, name='dispatch')

class UpdateAddress(View):

    def get(self, request, pk):

        add = Customer.objects.get(pk=pk)

        form = CustomerProfileForm(instance=add)

        totalitem = 0

        wishitem = 0
        if request.user.is_authenticated:
            totalitem = len(Cart.objects.filter(user=request.user))
            wishitem = len(Wishlist.objects.filter(user=request.user))
        return render(request, 'karshakapp/updateAddress.html', locals())
    def post(self, request, pk):
        form = CustomerProfileForm(request.POST)
        if form.is_valid():
            add = Customer.objects.get(pk=pk)
            add.name = form.cleaned_data['name']
            add.locality = form.cleaned_data['locality']
            add.city = form.cleaned_data['city']
            add.mobile = form.cleaned_data['mobile']
            add.state = form.cleaned_data['state']
            add.zipcode = form.cleaned_data['zipcode']
            add.save()
            messages.success(request, "Congratulations! Your profile has been saved successfully.")
        else:
            messages.warning(request, "Invalid input data. Please correct the errors.")
        return redirect("address")



@login_required
def add_to_cart(request):
    if request.method == 'POST':
        user = request.user
        product_id = request.POST.get('prod_id')
        product = get_object_or_404(Product, id=product_id)
        Cart.objects.create(user=user, product=product)
        totalitem = 0
        wishitem = 0
        if request.user.is_authenticated:
            totalitem = len(Cart.objects.filter(user=request.user))
            wishitem = len(Wishlist.objects.filter(user=request.user))
        return redirect("/cart")
    else:
        # Redirect GET requests to the home page or any other appropriate page
        return redirect("/")
@login_required
def show_cart(request):
    user = request.user
    cart = Cart.objects.filter(user=user)
    amount = 0
    for p in cart:
        value = p.quantity * p.product.discounted_price
        amount += value
    totalamount = amount + 40  # Assuming 40 is some additional fee
    totalitem = 0
    wishitem = 0
    if request.user.is_authenticated:
        totalitem = len(Cart.objects.filter(user=request.user))
        wishitem = len(Wishlist.objects.filter(user=request.user))
    return render(request, 'karshakapp/addtocart.html',  locals())

from django.http import JsonResponse
from django.db.models import Q
from .models import Cart
@login_required
def plus_cart(request):
    if request.method == 'GET':
        prod_id = request.GET.get('prod_id')
        cart_items = Cart.objects.filter(product=prod_id, user=request.user)
        if cart_items.exists():
            c = cart_items.first()
            c.quantity += 1
            c.save()
            cart = Cart.objects.filter(user=request.user)
            amount = sum(p.quantity * p.product.discounted_price for p in cart)
            totalamount = amount + 40
            data = {
                "quantity": c.quantity,
                "amount": amount,
                "totalamount": totalamount
            }
            return JsonResponse(data)


@login_required
def minus_cart(request):
    if request.method == 'GET':
        prod_id = request.GET.get('prod_id')
        cart_items = Cart.objects.filter(product=prod_id, user=request.user)
        if cart_items.exists():
            c = cart_items.first()
            c.quantity -= 1
            c.save()
            cart = Cart.objects.filter(user=request.user)
            amount = sum(p.quantity * p.product.discounted_price for p in cart)
            totalamount = amount + 40
```

```python
            data = {
                "quantity": c.quantity,
                "amount": amount,
                "totalamount": totalamount
            }
            return JsonResponse(data)
@login_required
def remove_cart(request):
    if request.method == 'GET':
        prod_id = request.GET.get('prod_id')
        cart_items = Cart.objects.filter(product=prod_id, user=request.user)
        if cart_items.exists():
            cart_items.delete()
            cart = Cart.objects.filter(user=request.user)
            amount = sum(p.quantity * p.product.discounted_price for p in cart)
            totalamount = amount + 40

            data = {
                "amount": amount,
                "totalamount": totalamount
            }
            return JsonResponse(data)

@method_decorator(login_required, name='dispatch')
class checkout(View):
    def get(self, request):
        totalitem = 0
        wishitem = 0
        if request.user.is_authenticated:
            totalitem = len(Cart.objects.filter(user=request.user))
            wishitem = len(Wishlist.objects.filter(user=request.user))
        user = request.user
        addresses = Customer.objects.filter(user=user)
        cart_items = Cart.objects.filter(user=user)
        famount = 0
        for p in cart_items:
            value = p.quantity * p.product.discounted_price
            famount += value
        totalamount = famount + 40
        razoramount = int(totalamount * 100)
        client = razorpay.Client(auth=(settings.RAZOR_KEY_ID,settings.RAZOR_KEY_SECRET))
        data = { "amount": razoramount, "currency": "INR", "receipt": "order_rcptid_12"}
        payment_response = client.order.create(data=data)
        #{'id': 'order_NuwG8N39x6Ikxk', 'entity': 'order', 'amount': 16000, 'amount_paid': 0,
'amount_due': 16000, 'currency': 'INR', 'receipt': 'order_rcptid_12', 'offer_id': None, 'status': 'created',
'attempts': 0, 'notes': [], 'created_at': 1712319606}
        print(payment_response)
        order_id = payment_response['id']
        order_status = payment_response['status']
        if order_status == "created":
            payment=Payment(
                user=user,
                amount=totalamount,
                razorpay_order_id = order_id,
                razorpay_payment_status = order_status
            )
            payment.save()
    return render(request, 'karshakapp/checkout.html', locals())


from django.shortcuts import redirect

from .models import Payment, Customer, Cart


from django.shortcuts import redirect

from .models import Customer, Payment, OrderPlaced

from .models import Cart


from django.shortcuts import redirect

from django.contrib.auth.decorators import login_required

from .models import Payment, Customer, Cart


@login_required

def payment_done(request):

  # Extract parameters from the request

  order_id = request.GET.get('order_id')

  payment_id = request.GET.get('payment_id')

  cust_id = request.GET.get('cust_id')


  # Retrieve the current user

  user = request.user


  # Retrieve the customer based on cust_id

  customer = Customer.objects.get(id=cust_id)


  # Update payment status and payment ID

  payment = Payment.objects.get(razorpay_order_id=order_id)

  payment.paid = True
```

```
    payment.razorpay_payment_id = payment_id
    payment.save()

    # Save order details
    cart = Cart.objects.filter(user=user)
    for c in cart:
        OrderPlaced.objects.create(
            user=user,
            customer=customer,
            product=c.product,
            quantity=c.quantity,
            payment=payment
        )
        c.delete()

    # Redirect to orders page after payment
    totalitem = 0
    wishitem = 0
    if request.user.is_authenticated:
        totalitem = len(Cart.objects.filter(user=request.user))
        wishitem = len(Wishlist.objects.filter(user=request.user))
    return redirect("orders")
@login_required
def orders(request):
    order_placed=OrderPlaced.objects.filter(user=request.user)
    totalitem = 0
    wishitem = 0
    if request.user.is_authenticated:
        totalitem = len(Cart.objects.filter(user=request.user))
        wishitem = len(Wishlist.objects.filter(user=request.user))
    return render(request, 'karshakapp/orders.html',locals())


@login_required
def plus_wishlist(request):
    if request.method == 'GET':
        prod_id = request.GET.get('prod_id')
        product = get_object_or_404(Product, id=prod_id)
        user = request.user
        wishlist_item, created = Wishlist.objects.get_or_create(user=user, product=product)
        # If the item was not already in the wishlist, create it
        if created:
            wishlist_item.save()  # Save the newly created wishlist item
            data = {'message': 'Product added to wishlist successfully.'}
        else:
            data = {'message': 'Product is already in the wishlist.'}
        return JsonResponse(data)


@login_required
def minus_wishlist(request):
    if request.method == 'GET':
        prod_id = request.GET.get('prod_id')
        product = get_object_or_404(Product, id=prod_id)
        user = request.user
        wishlist_item = Wishlist.objects.filter(user=user, product=product).first()
        if wishlist_item:
            wishlist_item.delete()
            data = {'message': 'Product removed from wishlist successfully.'}
        else:
            data = {'message': 'Product is not in the wishlist.'}

        return JsonResponse(data)


@login_required

def search(request):

    query = request.GET['search']

    totalitem = 0

    wishitem=0

    if request.user.is_authenticated:

        totalitem = len(Cart.objects.filter(user=request.user))

        wishitem = len(Wishlist.objects.filter(user=request.user))

    product = Product.objects.filter(Q(title__icontains=query))

    return render(request, "karshakapp/search.html",locals())


from django.shortcuts import render

from .models import Wishlist


def wishlist(request):

    wishlist_items = Wishlist.objects.filter(user=request.user)

    return render(request, 'karshakapp/wishlist.html', {'wishlist': wishlist_items})
```

Models.py

```
from django.db import models

from django.contrib.auth.models import User

# Create your models here.


STATE_CHOICES = [

    ('Andhra Pradesh', 'Andhra Pradesh'),

    ('Arunachal Pradesh', 'Arunachal Pradesh'),

    ('Assam', 'Assam'),
    ('Bihar', 'Bihar'),

    ('Chhattisgarh', 'Chhattisgarh'),

    ('Goa', 'Goa'),

    ('Gujarat', 'Gujarat'),

    ('Haryana', 'Haryana'),

    ('Himachal Pradesh', 'Himachal Pradesh'),

    ('Jammu and Kashmir', 'Jammu and Kashmir'),

    ('Jharkhand', 'Jharkhand'),

    ('Karnataka', 'Karnataka'),

    ('Kerala', 'Kerala'),

    ('Madhya Pradesh', 'Madhya Pradesh'),

    ('Maharashtra', 'Maharashtra'),

    ('Manipur', 'Manipur'),

    ('Meghalaya', 'Meghalaya'),

    ('Mizoram', 'Mizoram'),

    ('Nagaland', 'Nagaland'),

    ('Odisha', 'Odisha'),

    ('Punjab', 'Punjab'),

    ('Rajasthan', 'Rajasthan'),

    ('Sikkim', 'Sikkim'),

    ('Tamil Nadu', 'Tamil Nadu'),

    ('Telangana', 'Telangana'),

    ('Tripura', 'Tripura'),

    ('Uttarakhand', 'Uttarakhand'),

    ('Uttar Pradesh', 'Uttar Pradesh'),

    ('West Bengal', 'West Bengal'),

    ('Andaman and Nicobar Islands', 'Andaman and Nicobar Islands'),

    ('Chandigarh', 'Chandigarh'),

    ('Dadra and Nagar Haveli and Daman and Diu', 'Dadra and Nagar Haveli and Daman and Diu'),
    ('Delhi', 'Delhi'),

    ('Lakshadweep', 'Lakshadweep'),

    ('Puducherry', 'Puducherry'),

]



CATEGORY_CHOICES=(

    ('CR','Crops'),

    ('VE','Vegetables'),

    ('IN','Ingredients'),

    ('OI','Oils'),

    ('FL','Flours'),

    ('NU','Nuts'),

    ('GR','Grains'),

    ('HO','Honeys'),

    ('PU','Pulses'),

    ('FR','Fruits'),

    ('NG','Non veg'),

    ('GH','Ghees'),


)


STATUS_CHOICES = (

    ('Accepted', 'Accepted'),

    ('Packed', 'Packed'),

    ('On The Way', 'On The Way'),

    ('Delivered', 'Delivered'),

    ('Cancel', 'Cancel'),

    ('Pending', 'Pending'),
)


class Product(models.Model):
    title = models.CharField(max_length=100)
    selling_price = models.FloatField()
    quantity = models.FloatField(default=0)
    discounted_price = models.FloatField()
    description = models.TextField()
    composition = models.TextField(default='')
    prodapp = models.TextField(default='')
    category = models.CharField(choices=CATEGORY_CHOICES, max_length=50)
    product_image = models.ImageField(upload_to='product')

    def __str__(self):
        return self.title


class Customer(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    name = models.CharField(max_length=200)
    locality = models.CharField(max_length=200)
    city = models.CharField(max_length=50)
    mobile = models.IntegerField(default=0)
    zipcode = models.IntegerField()
    state = models.CharField(choices=STATE_CHOICES, max_length=100)

    def __str__(self):
        return self.name


class Cart(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)

    product = models.ForeignKey(Product, on_delete=models.CASCADE)

    quantity = models.PositiveIntegerField(default=1)


    @property

    def total_cost(self):

        return self.quantity * self.product.discounted_price


class Payment(models.Model):

    user = models.ForeignKey(User, on_delete=models.CASCADE)

    amount = models.FloatField()

    razorpay_order_id = models.CharField(max_length=100, blank=True, null=True)

    razorpay_payment_status = models.CharField(max_length=100, blank=True, null=True)

    razorpay_payment_id = models.CharField(max_length=100, blank=True, null=True)

    paid = models.BooleanField(default=False)



class OrderPlaced(models.Model):

    user = models.ForeignKey(User, on_delete=models.CASCADE)

    customer = models.ForeignKey(Customer, on_delete=models.CASCADE)

    product = models.ForeignKey(Product, on_delete=models.CASCADE)

    quantity = models.PositiveIntegerField(default=1)

    ordered_date = models.DateTimeField(auto_now_add=True)

    status = models.CharField(max_length=50, choices=STATUS_CHOICES, default='Pending')

    payment = models.ForeignKey(Payment, on_delete=models.CASCADE, default="")


    @property

    def total_cost(self):

        return self.quantity * self.product.discounted_price
```

```
class Wishlist(models.Model):  # Corrected 'Nodel' to 'Model'
    user = models.ForeignKey(User, on_delete=models.CASCADE)  # Corrected 'ForeignKey' to
'ForeignKey', 'on_delete-models' to 'on_delete=models'
    product = models.ForeignKey(Product, on_delete=models.CASCADE)  # Corrected 'Foreignkey' to
'ForeignKey', 'on_delete-models' to 'on_delete=models'
```

Forms.py

```
from django import forms
from django.contrib.auth.forms import UserCreationForm,AuthenticationForm, UsernameField,
PasswordResetForm ,PasswordChangeForm,SetPasswordForm
from django.contrib.auth.models import User
from .models import Customer


class LoginForm(AuthenticationForm):
    username = UsernameField(widget=forms.TextInput(attrs={'autofocus ':'True','class':'form-
control'}))
    password=forms.CharField( widget=forms.PasswordInput(attrs={'autocomplete':'current-
password','class':'form-control'}))


class CustomerRegistrationForm(UserCreationForm):
    username = forms.CharField(widget=forms.TextInput(attrs={'autofocus ':'True','class':'form-
control'}))
    email = forms.EmailField(widget=forms.EmailInput(attrs={'class':'form-control'}))
    password1=forms.CharField(label="Password", widget=forms.PasswordInput(attrs={'class':'form-
control'}))
    password2=forms.CharField(label="Confirm Password",
widget=forms.PasswordInput(attrs={'class':'form-control'}))
    class Meta:
        model = User
        fields = ['username', 'email', 'password1','password2']
```

```
# forms.py

class MyPasswordChangeForm(PasswordChangeForm):
    old_password = forms.CharField(label='Old Password',
        widget=forms.PasswordInput(attrs={'autofocus': 'True', 'autocomplete': 'current-password',
'class': 'form-control'}))
    new_password1 = forms.CharField(label='New Password',
        widget=forms.PasswordInput(attrs={'autocomplete': 'current-password', 'class': 'form-control'}))
    new_password2 = forms.CharField(label='Confirm Password',
        widget=forms.PasswordInput(attrs={'autocomplete': 'current-password', 'class': 'form-control'}))

class MyPasswordResetForm(PasswordResetForm):
    email = forms.EmailField(widget=forms.EmailInput(attrs={'class': 'form-control'}))

class MySetPasswordForm(SetPasswordForm):
    new_password1 = forms.CharField(label='New
Password',widget=forms.PasswordInput(attrs={'autocomplete': 'current-password', 'class': 'form-
control'}))
    new_password2 = forms.CharField(label='Confirm New
Password',widget=forms.PasswordInput(attrs={'autocomplete': 'current-password', 'class': 'form-
control'}))

class CustomerProfileForm(forms.ModelForm):
    class Meta:
        model = Customer
        fields = ['name', 'mobile',  'locality', 'city','zipcode', 'state']
        widgets = {
            'name': forms.TextInput(attrs={'class': 'form-control'}),
            'mobile': forms.NumberInput(attrs={'class': 'form-control'}),

            'locality': forms.TextInput(attrs={'class': 'form-control'}),

            'city': forms.TextInput(attrs={'class': 'form-control'}),

            'zipcode': forms.NumberInput(attrs={'class': 'form-control'}),

            'state': forms.Select(attrs={'class': 'form-select'}),

        }
```

Urls.py

```
from django.urls import path


from django.contrib.auth.admin import admin

from . import views

from django.conf import settings

from django.conf.urls.static import static

from django.contrib.auth import views as auth_view

from . forms import LoginForm,  MyPasswordResetForm,
MyPasswordChangeForm,MySetPasswordForm




urlpatterns = [

    path("", views.home,name="home"),

    path("about/", views.about,name="about"),

    path("contact/", views.contact,name="contact"),

    path("category/<slug:val>", views.CategoryView.as_view(), name="category"),

    path("category-title/<val>", views.CategoryTitle.as_view(),name="category-title"),

    path("product-detail/<int:pk>", views.ProductDetail.as_view(),name="product-detail"),

    path("profile/", views.ProfileView.as_view(), name="profile"),

    path("address/", views.addresses, name="address"),

    path("updateAddress/<int:pk>", views.UpdateAddress.as_view(), name="updateAddress"),

    path('add-to-cart/', views.add_to_cart, name='add-to-cart'),
    path('cart/', views.show_cart, name="showcart"),

    path('checkout/', views.checkout.as_view(), name="checkout"),

    path('paymentdone/', views.payment_done, name="paymentdone"),

    path("orders", views.orders,name="orders"),

    path("search/", views.search,name="search"),

    path('pluscart/', views.plus_cart),

    path('minuscart/', views.minus_cart),

    path('removecart/', views.remove_cart),

    path('pluswishlist/', views.plus_wishlist),

    path('minuswishlist/', views.minus_wishlist),

    path("wishlist/", views.wishlist,name="wishlist"),

    #login requirments

    path("registration/", views.CustomerRegistrationView.as_view(),name="customerregistration"),

    path('accounts/login/', auth_view.LoginView.as_view(template_name='karshakapp/login.html', authentication_form=LoginForm), name='login'),


    path('passwordchange/', auth_view.PasswordChangeView.as_view(template_name='karshakapp/changepassword.html',

    form_class=MyPasswordChangeForm,success_url='/passwordchangedone/'), name='passwordchange'),

    path('passwordchangedone/', auth_view.PasswordChangeDoneView.as_view(template_name='karshakapp/passwordchangedone.html'), name='passwordchangedone'),


    path('logout/', auth_view.LogoutView.as_view(next_page='login'),name='logout'),


    path('password-reset/', auth_view.PasswordResetView.as_view(template_name='karshakapp/password_reset.html', form_class=MyPasswordResetForm), name='password_reset'),

    path('password-reset/done/', auth_view.PasswordResetDoneView.as_view(template_name='karshakapp/password_reset_done.html'), name='password_reset_done'),
    path('password-reset-confirm/<uidb64>/<token>/',
auth_view.PasswordResetConfirmView.as_view(template_name='karshakapp/password_reset_confir
m.html',form_class=MySetPasswordForm), name='password_reset_confirm'),

    path('password-reset-complete/',
auth_view.PasswordResetCompleteView.as_view(template_name='karshakapp/password_reset_comp
lete.html'), name='password_reset_complete'),


]+static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)


from django.contrib import admin


admin.site.site_header = "Karshak"  # Set the site header

admin.site.site_title = "Karshak"  # Set the site title

admin.site.site_index_title = "Welcome to Karshak"  # Set the site index title
```

Settings.py

```
"""
Django settings for karshak project.


Generated by 'django-admin startproject' using Django 4.2.11.


For more information on this file, see
https://docs.djangoproject.com/en/4.2/topics/settings/


For the full list of settings and their values, see
https://docs.djangoproject.com/en/4.2/ref/settings/
"""


import os
```

```
from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = "django-insecure-wt1^ygx$sk_0c^)%%dp8($z)w1ho^r-s#usfdh_0fo61d-&_hq"

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    "django.contrib.admin",
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",
    "karshakapp",
]
```

```
MIDDLEWARE = [
    "django.middleware.security.SecurityMiddleware",
    "django.contrib.sessions.middleware.SessionMiddleware",
    "django.middleware.common.CommonMiddleware",
    "django.middleware.csrf.CsrfViewMiddleware",
    "django.contrib.auth.middleware.AuthenticationMiddleware",
    "django.contrib.messages.middleware.MessageMiddleware",
    "django.middleware.clickjacking.XFrameOptionsMiddleware",
]

ROOT_URLCONF = "karshak.urls"

TEMPLATES = [
    {
        "BACKEND": "django.template.backends.django.DjangoTemplates",
        "DIRS": [],
        "APP_DIRS": True,
        "OPTIONS": {
            "context_processors": [
                "django.template.context_processors.debug",
                "django.template.context_processors.request",
                "django.contrib.auth.context_processors.auth",
                "django.contrib.messages.context_processors.messages",
            ],
        },
    },
]
```

```
WSGI_APPLICATION = "karshak.wsgi.application"



# Database
# https://docs.djangoproject.com/en/4.2/ref/settings/#databases

DATABASES = {
    "default": {
        "ENGINE": "django.db.backends.sqlite3",
        "NAME": BASE_DIR / "db.sqlite3",
    }
}



# Password validation
# https://docs.djangoproject.com/en/4.2/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        "NAME": "django.contrib.auth.password_validation.UserAttributeSimilarityValidator",
    },
    {
        "NAME": "django.contrib.auth.password_validation.MinimumLengthValidator",
    },
    {
        "NAME": "django.contrib.auth.password_validation.CommonPasswordValidator",
    },
    {
        "NAME": "django.contrib.auth.password_validation.NumericPasswordValidator",
    },
]



# Internationalization
# https://docs.djangoproject.com/en/4.2/topics/i18n/

LANGUAGE_CODE = "en-us"

TIME_ZONE = "UTC"

USE_I18N = True

USE_TZ = True



# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.2/howto/static-files/




import os
STATIC_URL = '/static/'

MEDIA_URL = '/media/'
MEDIA_ROOT = BASE_DIR/'media'
LOGIN_REDIRECT_URL = "/profile/"

BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
```

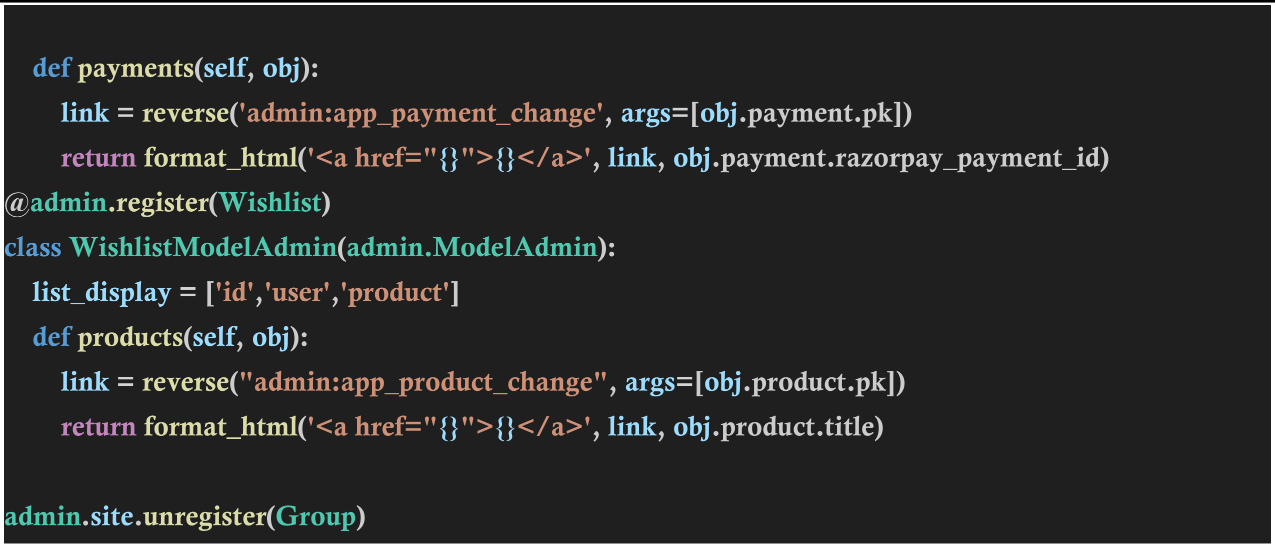
```
# Static files (CSS, JavaScript, Images)
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, 'static'),
]

# Define the directory where collected static files will be stored
STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')

# Default primary key field type
# https://docs.djangoproject.com/en/4.2/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = "django.db.models.BigAutoField"

EMAIL_BACKEND = 'django.core.mail.backends.console.EmailBackend'

RAZOR_KEY_ID = "rzp_test_Lzr0UZN4v5RRAX"
RAZOR_KEY_SECRET = "rypq4C2hmSdKUamkmJY26U89"
```

Urls.py(Karshak):

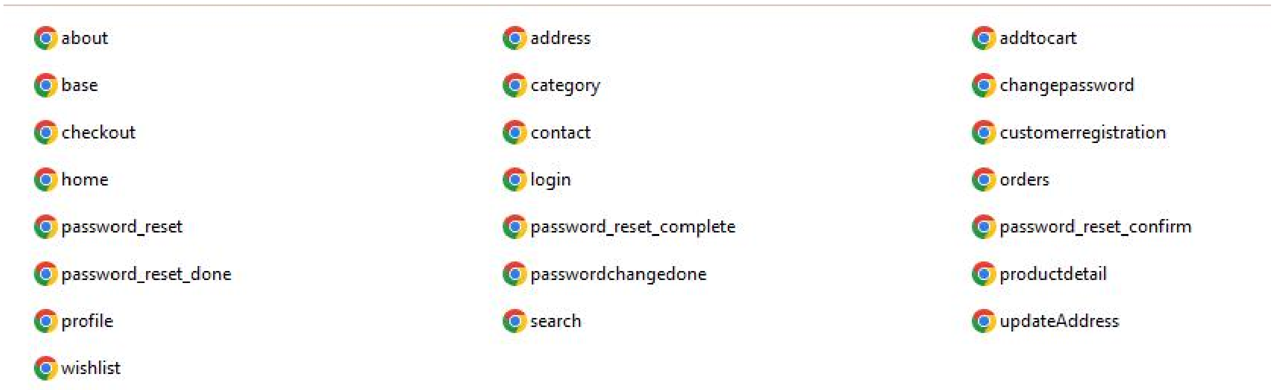
```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path("admin/", admin.site.urls),
    path("", include("karshakapp.urls")),
]
```

Admin.py

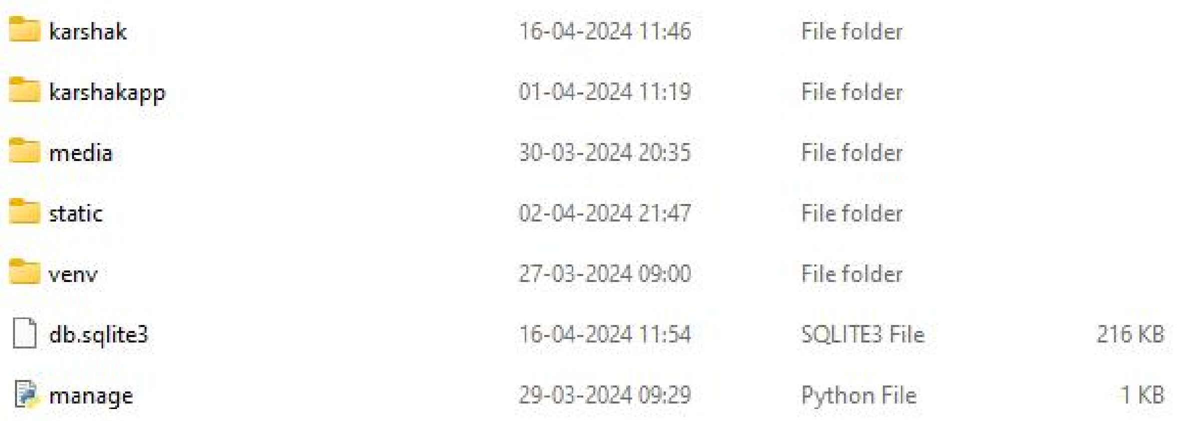
```python
from django.contrib.auth.models import Group
from django.contrib import admin
from django.urls import reverse
from .models import OrderPlaced, Payment, Product,Customer,Cart, Wishlist
from django.utils.html import format_html


@admin.register(Product)
class ProductModelAdmin(admin.ModelAdmin):
    list_display = ['id', 'title', 'discounted_price', 'get_category', 'product_image']

    def get_category(self, obj):
        return obj.category
    get_category.short_description = 'Category'

@admin.register(Customer)
class CustomerModelAdmin(admin.ModelAdmin):
    list_display = ['id', 'name','mobile', 'locality', 'city',  'zipcode', 'state']

@admin.register(Cart)
class CartModelAdmin(admin.ModelAdmin):
   list_display = ['id', 'user', 'product','quantity']
   def products(self, obj):
        link = reverse("admin:app_product_change", args=[obj.product.pk])
        return format_html('<a href="{}">{}</a>', link, obj.product.title)
@admin.register(Payment)
class PaymentModelAdmin(admin.ModelAdmin):
    list_display = ['id', 'user', 'amount', 'razorpay_order_id', 'razorpay_payment_status',
'razorpay_payment_id', 'paid']

@admin.register(OrderPlaced)
class OrderPlacedModelAdmin(admin.ModelAdmin):
    list_display = ['id', 'user', 'customers', 'products', 'quantity', 'ordered_date', 'status', 'payments']

    def customers(self, obj):
        link = reverse('admin:app_customer_change', args=[obj.customer.pk])
        return format_html('<a href="{}">{}</a>', link, obj.customer.name)

    def products(self, obj):
        link = reverse('admin:app_product_change', args=[obj.product.pk])
        return format_html('<a href="{}">{}</a>', link, obj.product.title)
```



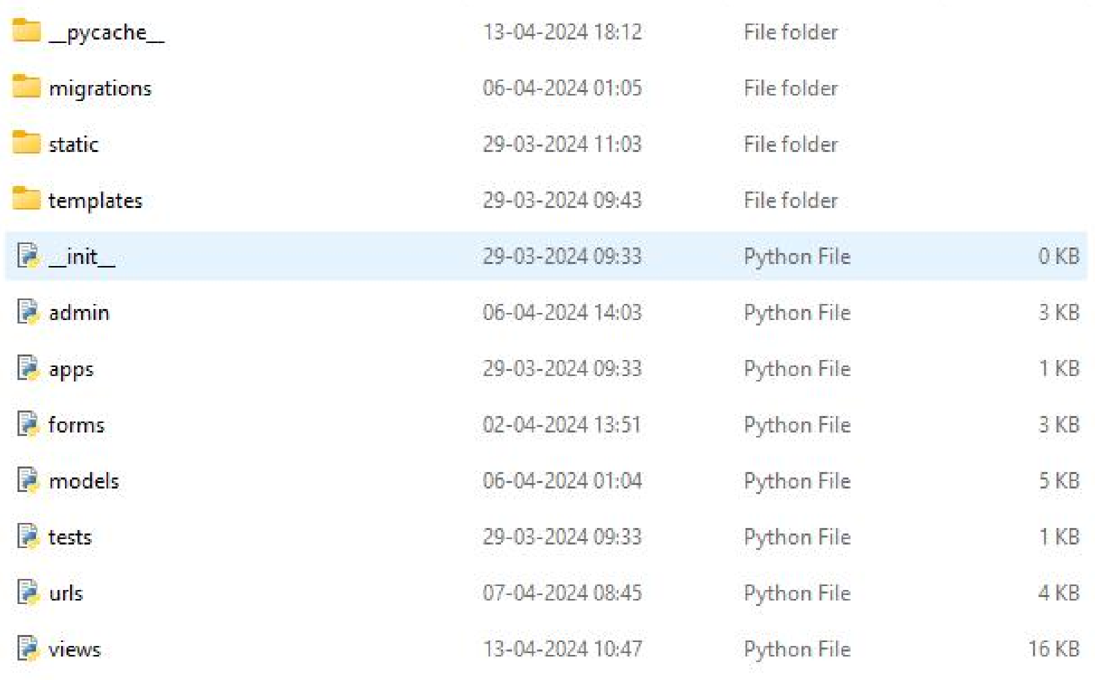
```
  def payments(self, obj):
    link = reverse('admin:app_payment_change', args=[obj.payment.pk])
    return format_html('<a href="{}">{}</a>', link, obj.payment.razorpay_payment_id)
@admin.register(Wishlist)
class WishlistModelAdmin(admin.ModelAdmin):
  list_display = ['id','user','product']
  def products(self, obj):
    link = reverse("admin:app_product_change", args=[obj.product.pk])
    return format_html('<a href="{}">{}</a>', link, obj.product.title)

admin.site.unregister(Group)
```



about	address	addtocart
base	category	changepassword
checkout	contact	customerregistration
home	login	orders
password_reset	password_reset_complete	password_reset_confirm
password_reset_done	passwordchangedone	productdetail
profile	search	updateAddress
wishlist		



karshak	16-04-2024 11:46	File folder	
karshakapp	01-04-2024 11:19	File folder	
media	30-03-2024 20:35	File folder	
static	02-04-2024 21:47	File folder	
venv	27-03-2024 09:00	File folder	
db.sqlite3	16-04-2024 11:54	SQLITE3 File	216 KB
manage	29-03-2024 09:29	Python File	1 KB



__pycache__	13-04-2024 18:12	File folder	
migrations	06-04-2024 01:05	File folder	
static	29-03-2024 11:03	File folder	
templates	29-03-2024 09:43	File folder	
__init__	29-03-2024 09:33	Python File	0 KB
admin	06-04-2024 14:03	Python File	3 KB
apps	29-03-2024 09:33	Python File	1 KB
forms	02-04-2024 13:51	Python File	3 KB
models	06-04-2024 01:04	Python File	5 KB
tests	29-03-2024 09:33	Python File	1 KB
urls	07-04-2024 08:45	Python File	4 KB
views	13-04-2024 10:47	Python File	16 KB

7.2 Modules:

User:

The user is a customer can order products

Register:

Users can register for the Mobile web application and any Pc.

Login:

After registering, the user can access his portal.

View Data:

Products,Orders,Transactions

Input :

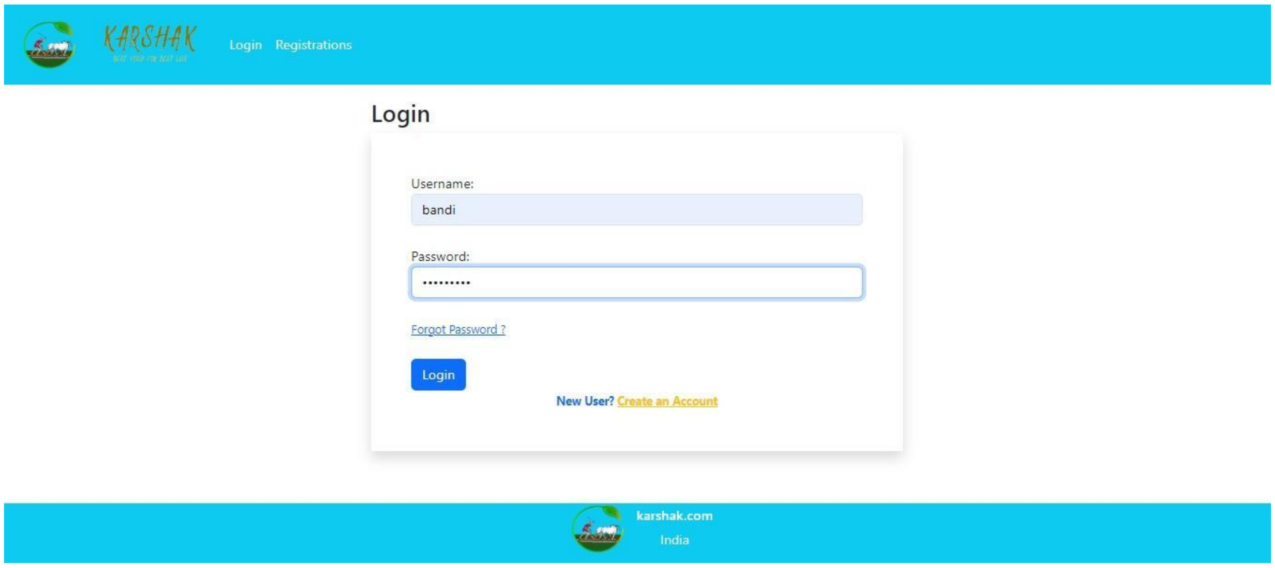
User will give the input values.

Result History:

So Many Orders Placed and delivery

7.3 Results:

Home Page: Users can view the Home page.



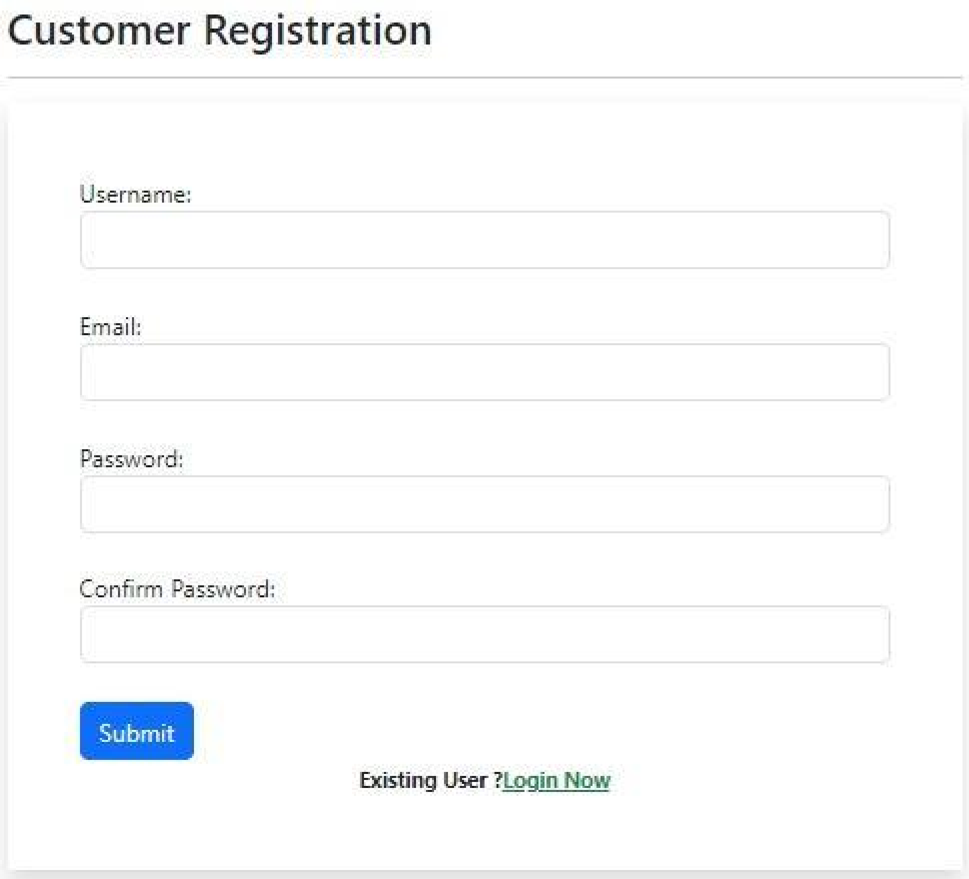
Login

Username:
bandi

Password:
•••••••••

New User? Create an Account

User Registration page: User can register with required details.



Customer Registration

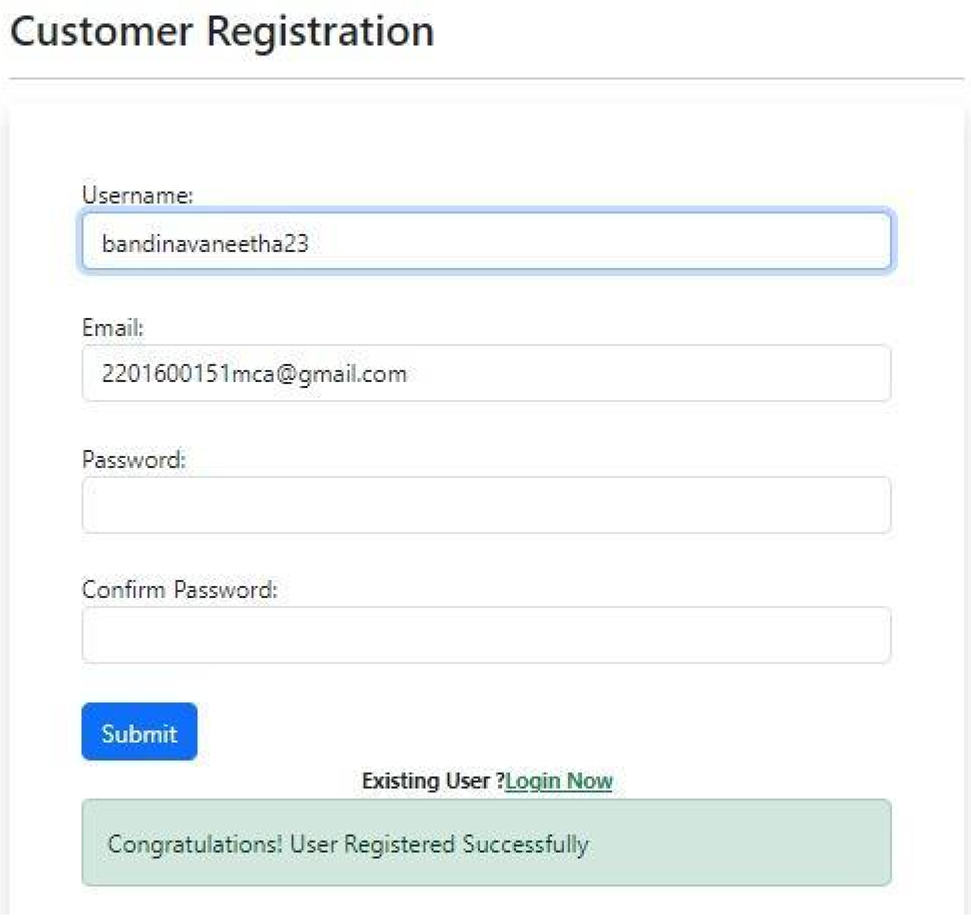
Username:

Email:

Password:

Confirm Password:

Existing User ?Login Now



Customer Registration

Username:
bandinavaneetha23

Email:
2201600151mca@gmail.com

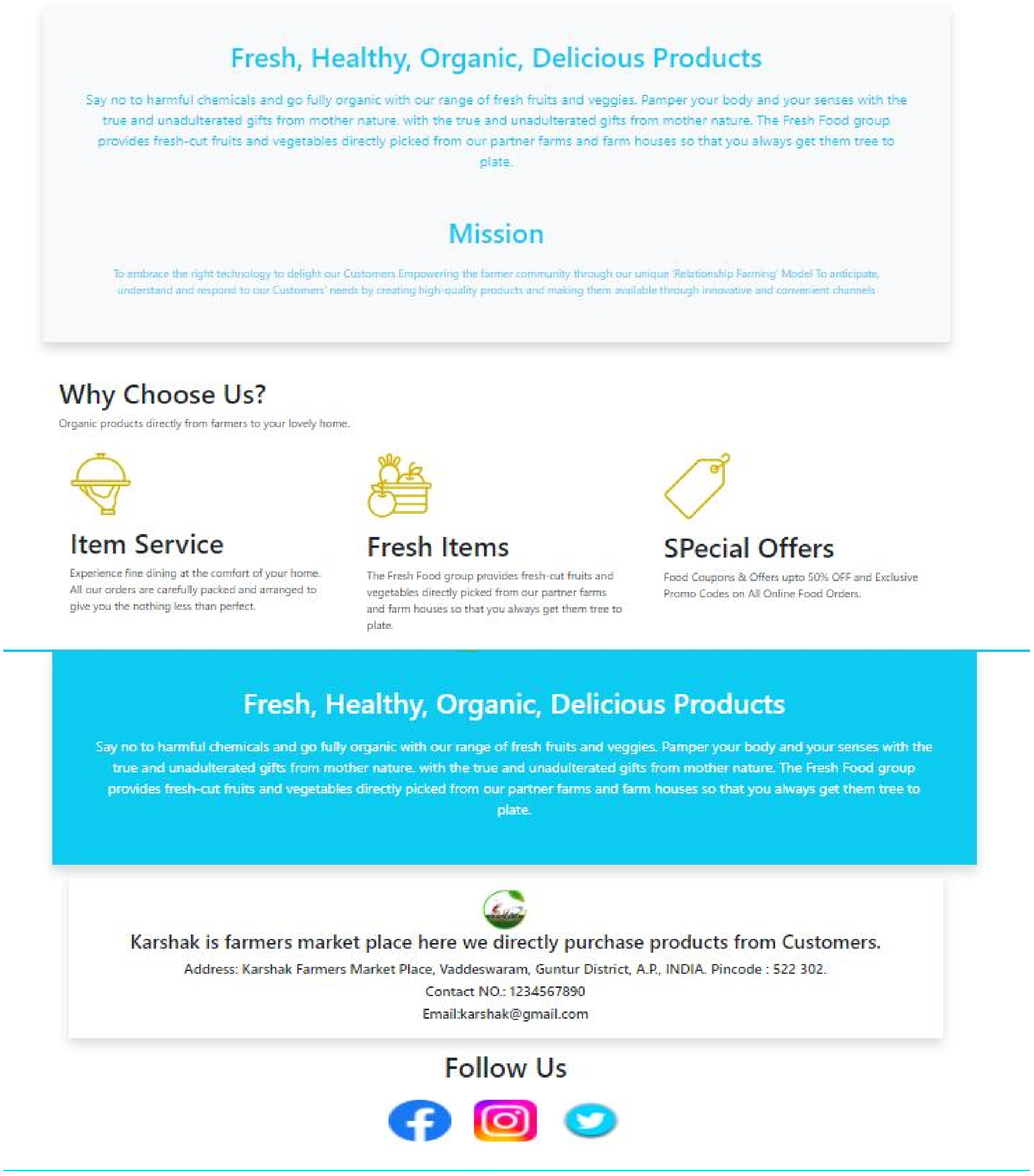
Password:

Confirm Password:

Existing User ?Login Now

Congratulations! User Registered Successfully

About: This is The small information about project



Fresh, Healthy, Organic, Delicious Products

Say no to harmful chemicals and go fully organic with our range of fresh fruits and veggies. Pamper your body and your senses with the true and unadulterated gifts from mother nature. with the true and unadulterated gifts from mother nature. The Fresh Food group provides fresh-cut fruits and vegetables directly picked from our partner farms and farm houses so that you always get them tree to plate.

Mission

To embrace the right technology to delight our Customers Empowering the farmer community through our unique 'Relationship Farming' Model To anticipate, understand and respond to our Customers' needs by creating high quality products and making them available through innovative and convenient channels

Why Choose Us?

Organic products directly from farmers to your lovely home.

Item Service

Experience fine dining at the comfort of your home. All our orders are carefully packed and arranged to give you the nothing less than perfect.

Fresh Items

The Fresh Food group provides fresh-cut fruits and vegetables directly picked from our partner farms and farm houses so that you always get them tree to plate.

SPecial Offers

Food Coupons & Offers upto 50% OFF and Exclusive Promo Codes on All Online Food Orders.

Fresh, Healthy, Organic, Delicious Products

Say no to harmful chemicals and go fully organic with our range of fresh fruits and veggies. Pamper your body and your senses with the true and unadulterated gifts from mother nature. with the true and unadulterated gifts from mother nature. The Fresh Food group provides fresh-cut fruits and vegetables directly picked from our partner farms and farm houses so that you always get them tree to plate.

Karshak is farmers market place here we directly purchase products from Customers.

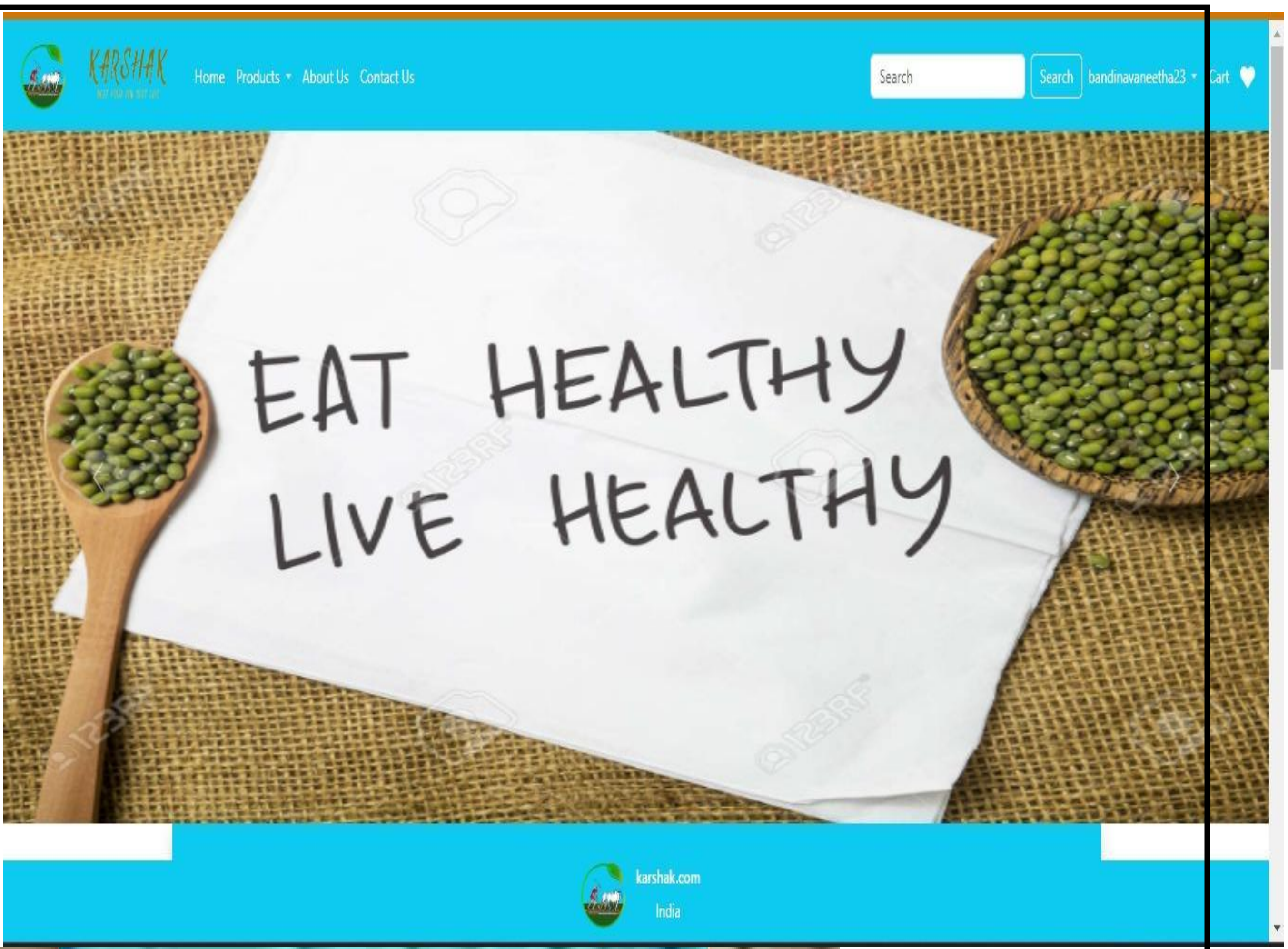
Address: Karshak Farmers Market Place, Vaddeswaram, Guntur District, A.P., INDIA. Pincode : 522 302.

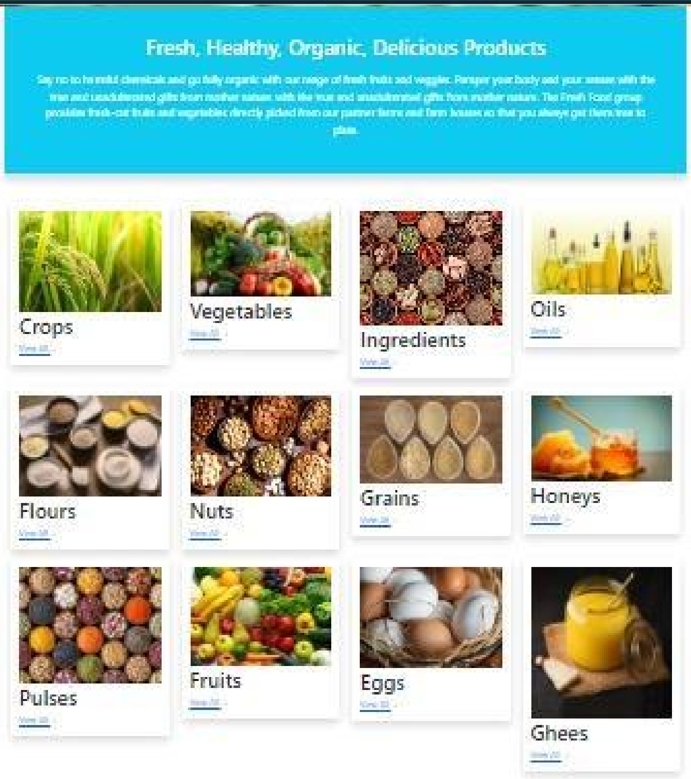
Contact NO.: 1234567890

Email:karshak@gmail.com

Follow Us

Home Page: Authenticated Users can view the Home page.

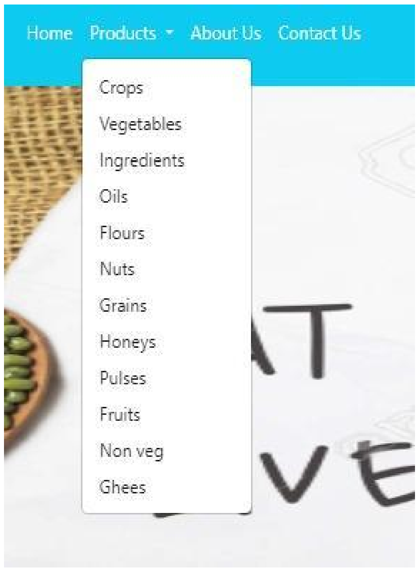




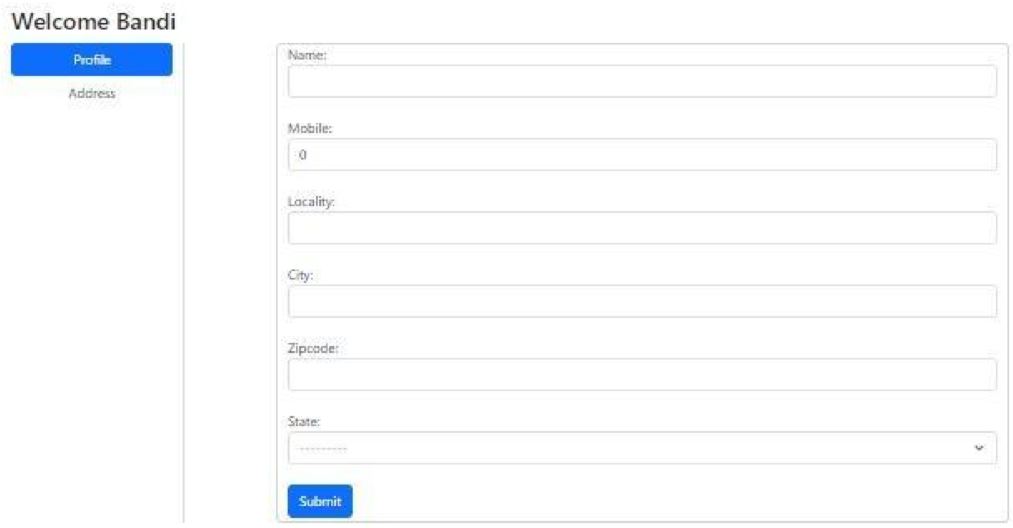
Fresh, Healthy, Organic, Delicious Products

Say no to harmful chemicals and go fully organic with our range of fresh fruits and veggies. Pamper your body and your senses with the true and unadulterated gifts from mother nature. with the true and unadulterated gifts from mother nature. The Fresh Food group provides fresh-cut fruits and vegetables directly picked from our partner farms and farm houses so that you always get them true to plate.

Products:



Profile:


Welcome Bandi

Name:

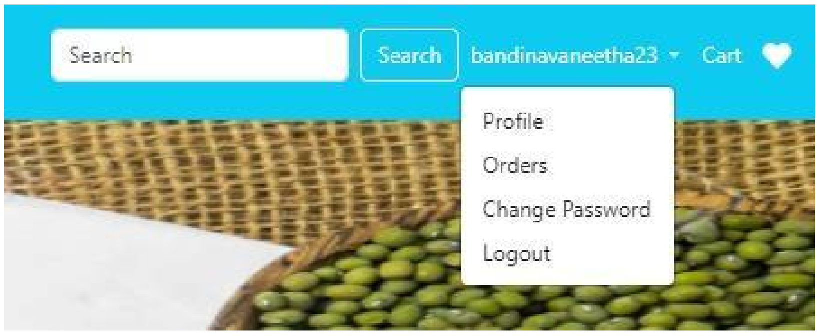
Mobile: 0

Locality:

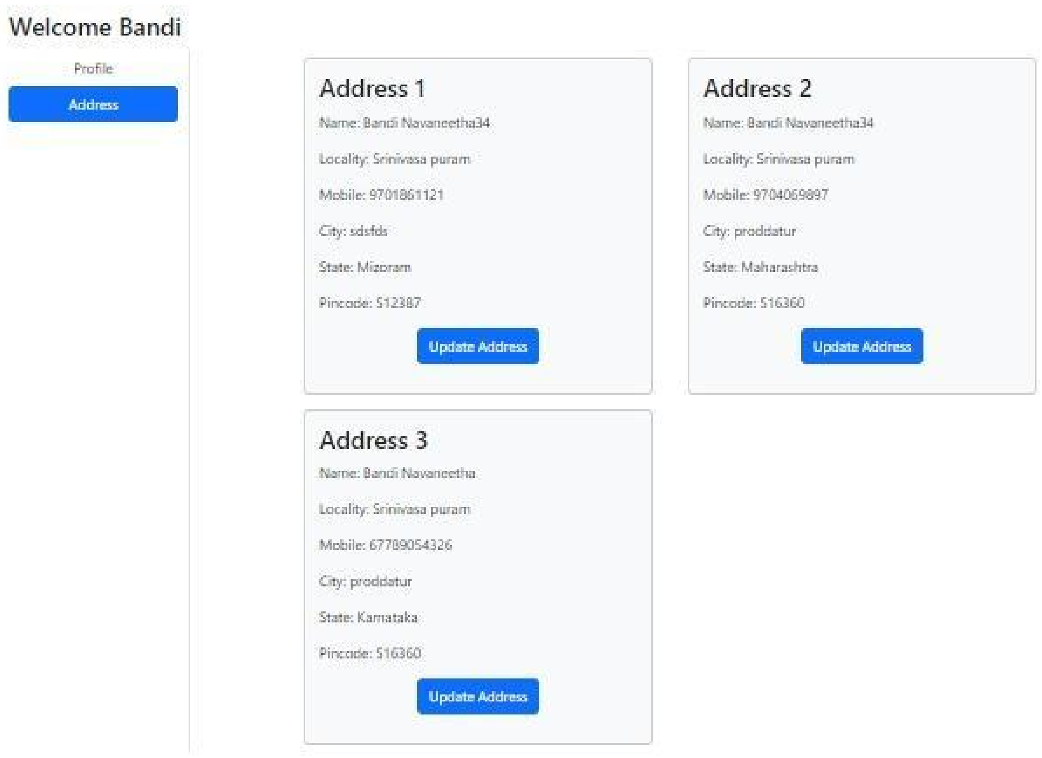
City:

Zipcode:

State: ---------



Address Update:



Welcome Bandi

Address 1

Name: Bandi Navaneetha34

Locality: Srinivasa puram

Mobile: 9701861121

City: sdsfds

State: Mizoram

Pincode: 512387

Address 2

Name: Bandi Navaneetha34

Locality: Srinivasa puram

Mobile: 9704069897

City: proddatur

State: Maharashtra

Pincode: 516360

Address 3

Name: Bandi Navaneetha

Locality: Srinivasa puram

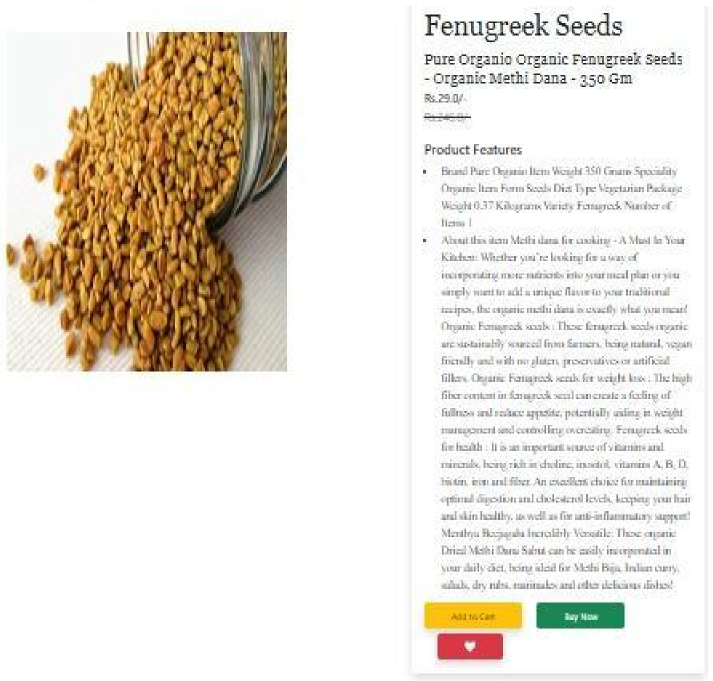
Mobile: 67789054326

City: proddatur

State: Karnataka

Pincode: 516360

Add to cart:



Fenugreek Seeds

Pure Organio Organic Fenugreek Seeds - Organic Methi Dana - 350 Gm

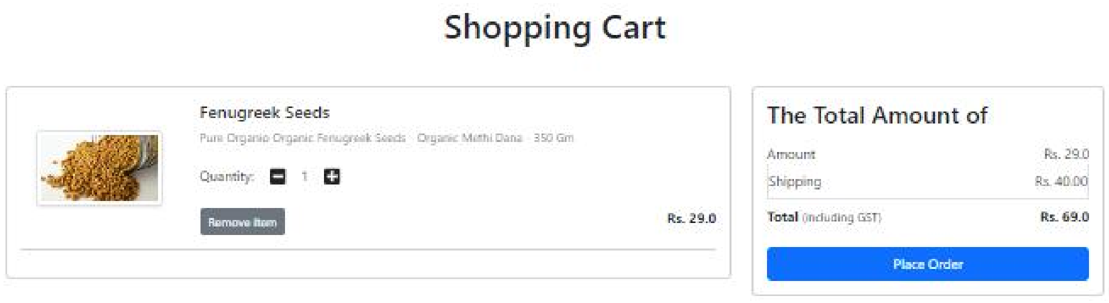
Rs.29.0/-

~~Rs.240.0/-~~

Product Features

- Brand Pure Organio Item Weight 350 Grams Speciality Organic Item Form Seeds Diet Type Vegetarian Package Weight 0.37 Kilograms Variety Fenugreek Number of Items 1
- About this item Methi dana for cooking - A Must In Your Kitchen: Whether you're looking for a way of incorporating more nutrients into your meal plan or you simply want to add a unique flavor to your traditional recipes, the organic methi dana is exactly what you need! Organic Fenugreek seeds : These fenugreek seeds organic are sustainably sourced from farmers, being natural, vegan friendly and with no gluten, preservatives or artificial fillers. Organic Fenugreek seeds for weight loss : The high fiber content in fenugreek seed can create a feeling of fullness and reduce appetite, potentially aiding in weight management and controlling overeating. Fenugreek seeds for health : It is an important source of vitamins and minerals, being rich in choline, inositol, vitamins A, B, D, biotin, iron and fiber. An excellent choice for maintaining optimal digestion and cholesterol levels, keeping your hair and skin healthy, as well as for anti-inflammatory support! Menthya Beejagalu Incredibly Versatile: These organic Dried Methi Dana Sabut can be easily incorporated in your daily diet, being ideal for Methi Bija, Indian curry, salads, dry rubs, marinades and other delicious dishes!

Order:



Shopping Cart

Fenugreek Seeds

Pure Organic Organic Fenugreek Seeds · Organic Methi Dana · 350 Gm

Quantity: 1

Remove Item

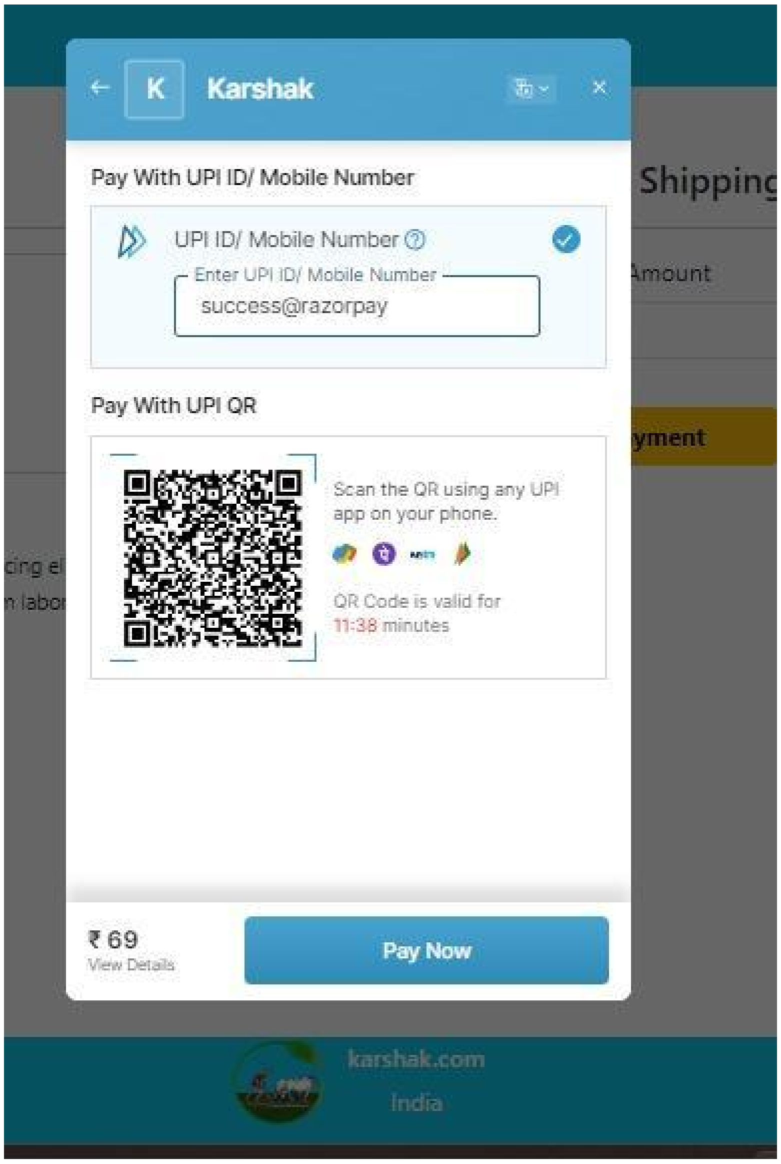
Rs. 29.0

The Total Amount of

Amount	Rs. 29.0
Shipping	Rs. 40.00
Total (including GST)	Rs. 69.0

Place Order

Transaction:



Karshak

Pay With UPI ID/ Mobile Number

UPI ID/ Mobile Number

Enter UPI ID/ Mobile Number

success@razorpay

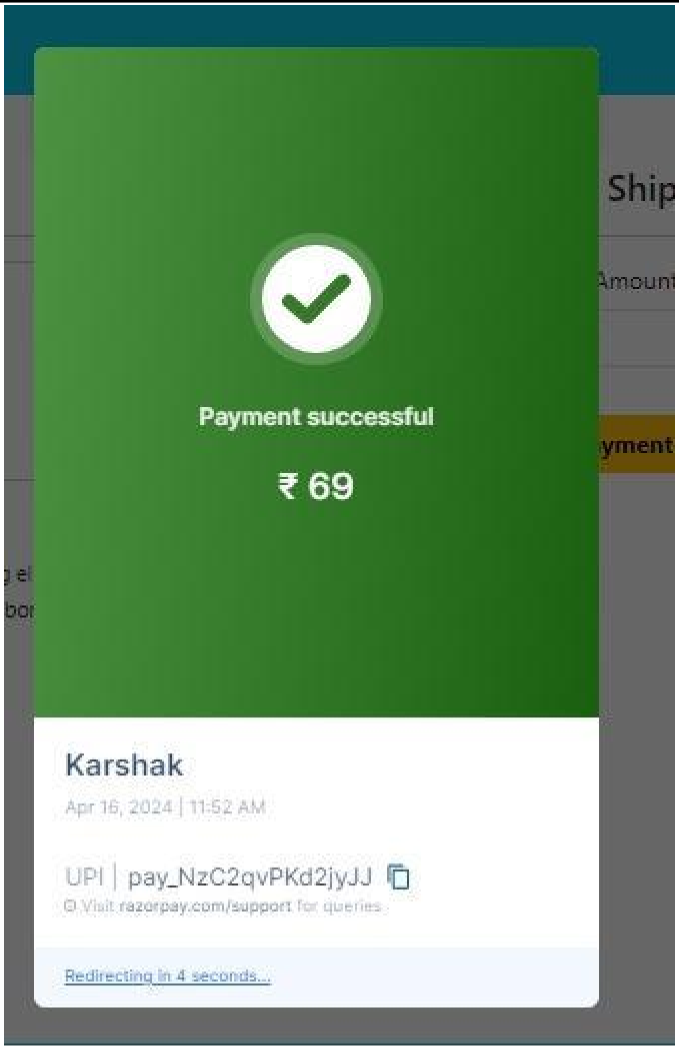
Pay With UPI QR

Scan the QR using any UPI app on your phone.

QR Code is valid for 11:38 minutes

₹ 69
View Details

Pay Now


Payment successful

₹ 69

Karshak

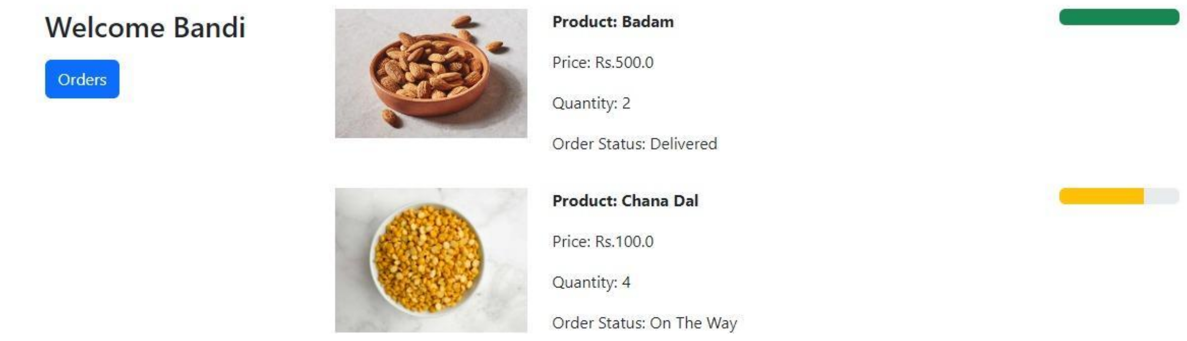
Apr 16, 2024 | 11:52 AM

UPI | pay_NzC2qvPKd2jyJJ

Visit razorpay.com/support for queries

Redirecting in 4 seconds...

Order Status(RESULT):


Welcome Bandi

Product: Badam

Price: Rs.500.0

Quantity: 2

Order Status: Delivered

Product: Chana Dal

Price: Rs.100.0

Quantity: 4

Order Status: On The Way

7.4 Test Cases:

Input	Output	Result
Input	Tested for different model given by user on the different model.	Success
Orders	Tested for different input given by the user on different models are created using the different algorithms and data.	Success
Transactions	Transactions will be performed using the different models built from the algorithms.	Success

7.5 Test cases Model building:

S.NO	Test cases	I/O	Expected O/T	Actual O/T	P/F
1	Register	Enter, email, password, mobile number, gender, address	Registration successful	Registratio n successfully completed	P
2	Register	Enter email, password, mobile number, gender, address	Registration successful	User Email already existed	F
3	Login	Enter valid email	Login to the page Userhome successfully	Login to the page Userhome successfully	P
4	Login	Enter invalid email or OTP or secret key	LoginUserhome successfully	Invalid Email	F
5	User can give a Input	Proper Input.	Algorithm can Predict a required input and generate result on Mobile Addictions	Result Generate d Successfully	P
6	User can give a Input	In proper Input	Algorithm cannot Predict a specific input	Invalid Input	F

CHAPTER 8

SYSTEM TESTING

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

8.1 TYPES OF TESTS:

System Testing for the Karshak Application:

User Authentication Testing:

- Verify that users can successfully register an account using their email ID and password.
- Test the login functionality to ensure users can access their accounts securely.
- Validate the password recovery process to ensure users can reset their passwords if forgotten.

Multilingual Support Testing:

- Test the application's ability to switch between languages (e.g., English, Hindi, Kannada) seamlessly.
- Verify that all text elements, including menus, buttons, and alerts, are correctly translated into the selected language.

Market Access Testing:

- Test the functionality to connect farmers with potential buyers or markets for their agricultural products.
- Verify that farmers can list their products with relevant details such as type, quantity, and price.

Communication Channels Testing:

- Validate the chat or messaging feature to ensure farmers can communicate with agricultural experts or government representatives.
- Test the feedback mechanism to allow users to provide input on the application's features or report issues.

Performance Testing:

- Test the application's performance under different network conditions (3G, 4G, Wi-Fi) to ensure smooth functionality.
- Validate the application's responsiveness and loading times for various screens and features.

Security Testing:

- Conduct security testing to identify and address vulnerabilities such as data breaches or unauthorized access.
- Test user authentication mechanisms to ensure they are robust and resistant to attacks like brute force or SQL injection.

Compatibility Testing:

- Test the application on different devices (smartphones, tablets) and operating systems (Android, IOS) to ensure compatibility.
- Validate the application's responsiveness and layout across various screen sizes and resolutions.

Localization Testing:

- Test the application's compatibility with regional settings, including date formats, currency symbols, and number formats.
- Verify that all location-based features, such as weather forecasts or market access, function correctly based on the user's location.

By conducting comprehensive system testing across these areas, we can ensure that the Karshak application meets its functional requirements, performs reliably, and delivers a seamless user experience to farmers and other stakeholders.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

CHAPTER 9

CONCLUSION

In conclusion, our project "Karshak" has achieved its objectives of creating a comprehensive mobile application aimed at empowering farmers and enhancing agricultural practices. Through the utilization of advanced technologies such as React Native, Node JS, and Firebase, we have developed a user-friendly platform that provides valuable resources and support to farmers across India.

By focusing on features such as government scheme updates, market access tools, and multilingual support, we have addressed key challenges faced by farmers, including access to information and communication barriers. Our application facilitates seamless interaction between farmers, experts, and stakeholders, enabling knowledge exchange, collaboration, and support within the agricultural community.

Through extensive data collection, preprocessing, and machine learning techniques such as the Transformer model, we have ensured the accuracy and reliability of information provided through our application. By leveraging cloud deployment and multilingual capabilities, we have enhanced accessibility and usability for farmers of diverse backgrounds and regions.

Overall, the "Karshak" project represents a significant advancement in agricultural technology, with the potential to drive positive change and impact the livelihoods of millions of farmers in India. By promoting sustainable practices, improving market access, and fostering education and awareness, our application contributes to the overall development and prosperity of rural communities. Moving forward, we aim to continue refining and expanding the capabilities of "Karshak" to further empower farmers and support the growth of the agricultural sector.

FUTURE WORKS:

Native Language

Farmers Interface

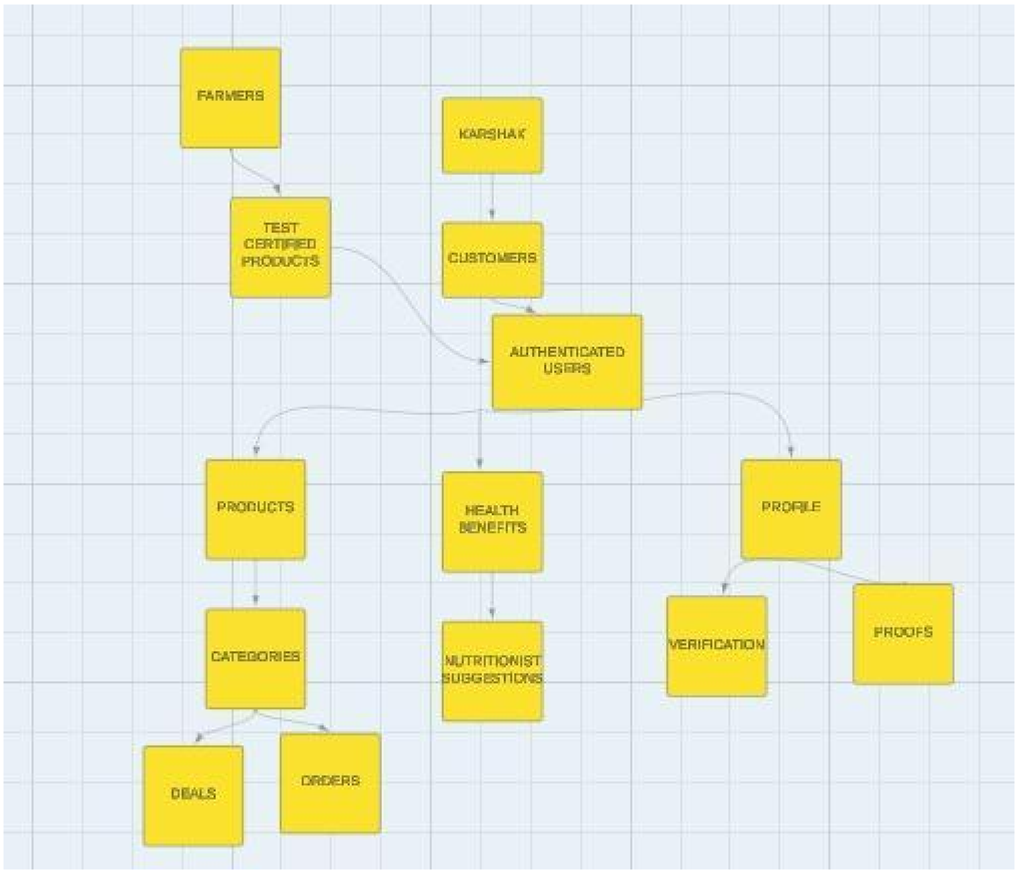
Providing different regional language options for the application.

UI enhancement

Adding GPS to automatically predict weather

Adding schemes from different states

UML DIAGRAM FOR FUTURE WORKS:



UML DIAGRAM

CHAPTER 10
REFERENCES

[1] Yuki Kawara, Chenhui Chu and Yuki Arase, “Preordering Encoding on Transformer for Translation,”.

[2] Sahinur Rahman Laskar, Abinash Dutta, Partha Pakray and CSivaji Bandyopadhyay, “Neural Machine Translation: English to Hindi,”.

[3] Sufeng Duan1, Hai Zhao1, Junru Zhou1 and Rui Wang2, “Syntax aware Transformer Encoder for Neural Machine Translation,”.

[4] J. Li, D. Xiong, Z. Tu, M. Zhu, M. Zhang, and G. Zhou, “Modeling source syntax for neural machine translation,” in ACL, 2017, pp. 688–697

[5] Kyunghyun Cho, Bart van Merrenboer, Dzmitry Bahdanau and Yoshua Bengio, “On the Properties of Neural Machine Translation: Encoder–Decoder Approaches,”.

[6] https://reactnative.dev/docs/getting-started.

[7] https://firebase.google.com/.

[8] https://agricoop.nic.in/en.

[9] https://openweathermap.org/api.

[10] https://nfsm.gov.in/nfmis/rpt/calenderreport.asp

[11] Joshi, N. C., Chayapathi, A. R., Nair, A. A., Ajay, S., & Suhail, M. S. (2021). Kisan Soch – A Mobile App for Farmers. International Journal of Creative Research Thoughts (IJCRT), 9(7), 346-351.

[12] Reference a YouTube video (https://youtu.be/qwFBXuEeg1U?si=xCrfdMj4uFcBYbHe)