**SYNOPSIS FOR**

**CerealSphere**

**IN PARTIAL FULFILLMENT OF MASTER OF COMPUTER APPLICATION SEM-IV**

**BY**

**Pooja Kothawade**

**MCA-SY SEM-IV DIV-C ROLL. NO. 22332**

**2023-2024**

**Project Guide: Prof. Rupali Taware**

**SUBMITTED TO**

**SAVITRIBAI PHULE PUNE UNIVERSITY**

# INTRODUCTION

In this project, we aim to develop a feature-rich cereal buisness platform using the Python Django framework. Our goal is to create a robust and scalable solution that offers a seamless shopping experience to users.

User Authentication and Profiles: Users can create accounts, manage their profiles, and track their order history.

Cereals Catalog: A comprehensive catalog of cereals with detailed descriptions, images, and pricing information.

Shopping Cart: Users can add products to their cart, update quantities, and proceed to checkout. Order Management: Admins can manage orders, process payments, and update order statuses.

Payment Gateway Integration: Integration with popular payment gateways for secure and seamless payment processing.

Responsive Design: A responsive design to ensure a consistent user experience across devices.

# OBJECTIVES AND SCOPES

1. **Build a Robust Cereal buisness Platform**: Develop a fully functional e-commerce platform using Python Django that meets industry standards for performance, security, and scalability.
2. **Provide a Seamless online Experience**: Create a user-friendly interface that allows customers to easily browse products, add them to cart, and complete transactions with minimal friction.
3. **Integrate Secure Payment Processing**: Implement integration with a reliable payment gateway to ensure that transactions are secure and meet compliance standards.
4. **Offer a Diverse Cereal Catalog**: Populate the platform with a diverse range of products across various categories to cater to a wide audience.
5. **Enable User Account Management**: Allow users to create accounts, manage their profiles, track order history, and receive personalized recommendations.
6. **Ensure Mobile Responsiveness**: Ensure that the platform is responsive and optimized for mobile devices to provide a consistent experience across all devices

# EXISTING SYSTEM

Django-Oscar is a popular open-source e-commerce framework for Django. It provides a solid foundation for building custom e-commerce applications with features such as product management, order processing, and payment integration.

1. **Product Management:** Django-Oscar allows you to create and manage product catalogs with support for multiple product types, categories, and attributes.
2. **Order Processing:** The framework includes features for managing orders, including order tracking, fulfillment, and refunds.
3. **Payment Integration:** Django-Oscar supports integration with various payment gateways, allowing you to securely process payments online.
4. **User Accounts:** The framework provides user authentication and account management features, allowing customers to create accounts, manage their profiles, and track their orders.
5. **Customizable Templates:** Django-Oscar comes with a set of customizable templates that you can use to design your storefront.
6. **Security:** The framework includes built-in security features to protect customer data and prevent fraud.

# SYSTEM REQUIREMENT

## SERVER HARDWARE REQUIREMENTS:

**PROCESSOR:** INTEL PENTIUM OR AMD

**RAM:** 1GB+ DDR

**HDD:** 40GB

## SERVER SOFTWARE REQUIREMENTS:

**OPERATING SYSTEM:** Windows/MacOS/Linux **WEB BROWSER:** Mozilla Firefox, Google Chrome etc. **FRONT END:** HTML, CSS, JAVASCRIPT

**BACK END:** Python Django for backend development

**DATABASE**: PostgreSQL

## CLIENT HARDWARE REQUIREMENTS:

**PROCESSOR:** INTEL PENTIUM OR AMD

**RAM:** 1GB+ DDR

**HDD:** 40GB

## CLIENT SOFTWARE REQUIREMENTS:

**OPERATING SYSTEM:** Windows/MacOS/Linux/Android

**WEB BROWSER:** Mozilla Firefox, Google Chrome etc.

# MODULES

### Profile Module:

#### Admin Profile:

* + - * **Register/login:** Admin Register and Login.
      * **View Profile**: Admin can view profile.
      * **Edit Profile**: Admin can edit profile.
      * **View/Manage Customer**: Admin can view and manage Customer.

#### Customer Profile:

* + - * **Register/login:** Customer Register and Login.
      * **View Profile**: Customer can view profile.
      * **Edit Profile**: Customer can edit profile information.

### Cereal Module:

#### Admin:

* + - * **View Product**: Admin can view Product Details.
      * **Accept Product Request**: Admin can accept Product request from Customer.
      * **Manage Product Details**: Admin can manage all product details.

#### Customer:

* + - * **View Product**: Customer can view Product Details.
      * **Select Product**: Customer can select product.
      * **Buy Product:** Customer can order the product.

### Order Module:

#### Admin:

* + - * **View Bookings**: Admin can view order details.
      * **Manage Order:** Admin can manage order**.**
      * **Manage Order Status:** Admin can change order status.
    - **Customer:**

### Payment Module:

#### Admin:

* **Make Booking**: Customer can make order of the product.
* **View Order Status**: Customer can view order status.
* **View Previous Orders**: Customer can view previous orders.
  + **View Payment**: Admin can view payment details and payment status.

#### Customer:

* + **Make Payment**: Customer can make payment through COD, UPI or debit/credit card.

# PROPOSED SYSTEM

The proposed system is an e-commerce platform built using Python Django, aiming to provide a seamless shopping experience for users and a robust set of tools for merchants to manage their online stores.

#### Key Features:

1. **User Authentication and Profiles:** Users can create accounts, manage their profiles, and track their order history.
2. **Cereal Catalog:** A comprehensive catalog of products with detailed descriptions, images, and pricing information.
3. **Shopping Cart:** Users can add products to their cart, update quantities, and proceed to checkout.
4. **Order Management:** Admins can manage orders, process payments, and update order statuses.
5. **Payment Gateway Integration:** Integration with popular payment gateways for secure and seamless payment processing.
6. **Responsive Design:** A responsive design to ensure a consistent user experience across devices.

# LIMITATIONS

1. **Scalability Concerns:** While Django is known for its scalability, large-scale e-commerce platforms may encounter performance issues as the user base and transaction volume increase. Additional optimization and infrastructure scaling may be required to handle high traffic loads.
2. **Customization Complexity:** While Django offers a high degree of customization, implementing complex features or customizations may require significant development effort and expertise.
3. **Limited Built-in Features:** While Django provides a solid foundation, some advanced e- commerce features may need to be implemented from scratch or using third-party packages, adding complexity to the development process.
4. **Dependency on Third-Party Packages:** Relying on third-party packages for essential e- commerce features such as payment gateways or shipping integrations can introduce dependencies and potential compatibility issues.
5. **Security Vulnerabilities:** As with any web application, Django-based e-commerce platforms are susceptible to security vulnerabilities if not properly secured and maintained. Regular security audits and updates are essential to mitigate these risks.
6. **Learning Curve:** Developing and maintaining a Django-based e-commerce platform requires a certain level of proficiency in Python and Django, which may be a barrier for developers new to the framework.
7. **Maintenance Overhead:** Keeping the platform up to date with the latest Django and third- party package releases, as well as ensuring compatibility with evolving web standards, can require ongoing maintenance effort.