FOOD DELIVERY SYSTEM



Submitted by:

Sheramie Bigwas

Athena Khamille Quinit

Submitted to:

Mr. Anthonie Momo

**1.Planning**

The purpose of a food delivery is to provide a convenient platform that connects customers with restaurants or vendors, enabling them to order and receive food or items quickly and efficiently. Here are the key purposes:

**For customers**:

* Convenience: Allows customers to order food anytime, anywhere without visiting a restaurant.
* Flexible Payment Options: Accepts various payment methods such as cash on delivery, PayPal, PayMaya, visa and g-cash.
* Time-Saving: Ensures fast delivery through efficient order processing and routing.
* Real-Time Updates: Keeps customers informed on their order preparation, dispatch, and delivery status.

**For Riders/Delivery Personnel:**

* Job Opportunities: Provides flexible employment opportunities, especially for part-time or gig workers.
* Flexible Working Hours: Allows delivery personnel to set their own schedules and work as much or as little as they want.
* Provides the option to choose preferred working hours, such as evenings, weekends, or peak meal times.
* Safety and Security: Some platforms offer safety equipment like helmets, gloves, or protective gear for drivers.
* Community Engagement: Encourages delivery personnel to support local businesses and contribute to their community by transporting food.
* Skill Development: Provides a platform to learn time management, customer service, and route planning skills.
* Earnings Potential: Increases income based on the number of deliveries completed, with incentives like tips, bonuses, or commissions.
* Allows delivery personnel to earn additional money during busy hours (e.g., lunch and dinner rushes).

**For Administrators:**

* Customer Trust and Satisfaction: Builds trust by showcasing genuine feedback from users.
* Helps customers make informed choices by providing visible ratings.
* Customer Satisfaction: Prevents customers from ordering unavailable items, avoiding disappointment or frustration.
* Content Management: Update website content regularly, such as menus, delivery information, and promotions.
* User Account Management: Monitor and manage user accounts (customers and riders).

**Scope:**

A food delivery system has a wide-ranging scope that benefits multiple stakeholders, including customers, restaurants, delivery personnel, and administrators. It revolutionizes how food services are provided and consumed, adapting to modern needs and leveraging technology to ensure convenience, efficiency, and growth.

**For Customers:**

* Convenience: Customers can order food from the comfort of their homes or workplaces at any time.
* Access to Variety:
* Offers access to a wide range of cuisines and restaurants that may not be physically near them.
* Real-Time Updates:
* Provides estimated delivery and notifications.
* Payment Options: Supports multiple payment methods, including online payments such as PayMaya, cash on delivery, visa, PayPal and g-cash.

**For Riders/ Delivery Personnel:**

* Job Opportunities: Provides flexible job roles for delivery drivers with the potential to earn based on orders delivered.
* Increased Efficiency: Allows multiple orders to be managed simultaneously, enhancing delivery speed and productivity.

**For Administrators:**

* Platform Management: Enables smooth operation of the system in user account management and delivery assignments.
* Quality Control: Monitors customer and delivery personnel performance, ensuring high standards.

**Feasibility:**

The feasibility of a food delivery system in our project appears promising, as it leverages modern web technologies like Laravel for backend development, ensuring efficient data management and secure payment processing. By incorporating key features such as role-based user access (admin, user, and rider), the system can streamline the ordering, delivery, and management processes. With a clear focus on ease of use, mobile-friendly design, and robust order tracking, your system aligns well with the growing demand for convenient, online food delivery services, offering a scalable and user-centric solution.

**2. System Analysis**

* **Existing System Evaluation:**
  + If the food delivery system already exists in some form (e.g., a basic delivery platform or manual system), the analysis should examine the current state:
* **Current System Workflow:**
  + How are orders placed? (e.g., phone calls, website)
  + How are deliveries managed? (e.g., manually assigned, status?)
  + How is payment handled? (e.g., gcash, paymaya, paypal, cash on delivery and visa)
* **Performance Evaluation:** Speed of order placement and processing
  + Time taken for delivery.
  + System downtime, if any.
  + Customer feedback and vendor satisfaction.
  + Functional Requirements:
* **Customer Side:**
  + User Registration & Authentication: Users must be able to register and log in.
  + Browse Restaurants and Menus: Customers can view available menus and dishes.
  + Order Placement: Customers select items, place orders, and choose payment methods.
  + Real-Time Order Tracking: Customers can track the status of their orders and see delivery progress.
  + Ratings and Reviews: After delivery, customers can rate both food and delivery experience.
* **Delivery Personnel Side:**
  + Delivery Staff Registration & Assignment: Delivery personnel should register, receive order details, and track deliveries.
  + Status Updates: Delivery personnel must update order status (e.g., , on delivery, delivered)
* **Admin Side:**
  + User and Vendor Management: Admin should be able to manage customers and delivery personnel.
  + System Monitoring and Maintenance: Ensure smooth functioning, handle issues, and maintain the database.
  + Report Generation: Admin should generate reports on sales, deliveries, and customer feedback.

The food delivery project reveals a well-structured architecture that efficiently manages the flow of orders from customers to riders. By utilizing Laravel for backend development, the system ensures seamless interactions between users, admins, and riders, with clear role-based access control for better security and functionality. The integration of features such as product management, order tracking, payment handling, and user authentication enables smooth operations. Additionally, the system is designed to be scalable, adaptable to increasing user demand, and responsive to the evolving needs of food delivery services.

**3. Data Flow and System Design:**

* **Customer Data Flow:**
  + Registers and logs in.
  + Browses restaurants, selects food items, and places an order.
  + Makes payment and tracks order.
  + Provides feedback on food and delivery service.
* **Delivery Data Flow:**
  + Registers and logs in.
  + Receives order details from the system, including pickup and delivery locations.
  + Marks the order as on delivery and delivered.
* **Admin Data Flow:**
  + Manages customer and delivery personnel accounts.
  + Oversees all orders, ensuring smooth operations.
  + Generates reports and handles any issues (disputes, complaints).
  + System Architecture Design:
  + Customer App: Web and mobile versions for ordering food.
  + Delivery App: Interface for delivery personnel to receive and manage deliveries.
  + Admin Dashboard: For system administrators to monitor, manage, and generate reports.
  + Backend:
  + Order Processing System: Manages incoming orders, payment validation, and status updates.
  + Payment Gateway Integration: Secure processing of customer payments (cash on delivery, visa, PayMaya, Visa, PayPal and g-cash).
  + Real-Time Tracking System: Provides order status updates to customers and tracks delivery progress.

**4. Development**

The development of a food delivery system involves several stages, from planning and design to implementation and deployment. Each phase focuses on specific aspects of the system to ensure functionality, scalability, security, and usability.

**Planning and Requirements Gathering:**

Define the project scope, identify stakeholders, and gather detailed requirements from customers, delivery personnel, and administrators.

**Tasks:**

* Conduct stakeholder interviews (customers, and delivery personnel).
* Define the system’s functional and non-functional requirements.
* Create a project plan, setting timelines for each phase of development.

**System Architecture Design:**

Design a scalable and reliable architecture that can support the operations of the food delivery system.

**Tasks:**

**Frontend Design:**

* Design customer, rider, and admin interfaces (using wireframes and mockups).
* Choose the frontend technologies (Laravel version 9).

Backend Design:

* Define API endpoints and server-side logic (using Laravel for backend services).
* Design the database schema to store deliveries, orders, users, payments, products, carts and ratings (using MySQL databases).

**Frontend Development:**

Build the user-facing components (mobile apps) for customers, riders and administrators.

**Tasks:**

**Customer Interface:**

* Design pages for menu browsing, order placement, payment, and status.
* Implement features for user login, registration, and profile management.
* Implement a responsive UI using CSS frameworks (e.g., Tailwind CSS) or libraries (e.g., Bootstrap).

**Delivery Personnel Interface:**

* Build features for delivery staff to receive and accept orders, navigate routes, and update order statuses.

**Admin Interface:**

* Create a dashboard to monitor and manage users, orders and system settings.

**5. Testing**

* User Registration and Authentication:
* Admin: Can register, login, and access admin pages.
* Customer: Can register, login, and place an order.
* Rider: Can register, login, and view assigned orders.

**Product Management (Admin):**

**Test cases:**

* Admin can add, update, and delete food items.
* Admin can categorize products.
* Admin can view a list of products.

**Order Placement (Customer):**

**Test cases:**

* Customer can browse products.
* Customer can add products to the cart and proceed to checkout.
* Customer can select a payment method (g-cash, PayPal, cash on delivery, visa and PayMaya).
* Customer can track the status of their order (pending, dispatched, delivered).

**Order Management (Admin & Rider):**

* Riders can accept orders and update delivery status.
* Payment Gateway Integration:

**Test cases:**

* Validate that the payment amount matches the order total.
* Check for correct payment processing via integrated payment systems like Stripe or Paypal.

**Delivery Tracking (Rider):**

**Test cases:**

* Test real-time tracking of rider's location (if implemented).
* Test that the rider's status updates (e.g., "Out for Delivery", "Delivered").

**Feedback System:**

**Test cases:**

* Customers can submit feedback after receiving their orders.
* Admin can view only feedback but can't reply to the customer's feedback.

**6. Implementation**

1. composer create-project laravel/laravel:^9.0 delivery-app
2. cd delivery-app
3. composer require laravel/breeze --dev
4. php artisan breeze:install
5. go to .env write the database name in DB\_DATABASE=delivery-app
6. php artisan migrate

**7. Maintenance**

1. Monitoring and Performance Optimization

* Regular Monitoring: Use monitoring tools to track server performance, API response times, and database load.
* Scalability: Ensure the system scales with the increase in users or orders by optimizing backend services and databases.
* Bug Fixing: Continuously monitor for errors and fix them promptly to avoid system downtime.

2.Security Updates

* Patch Vulnerabilities: Regularly apply security patches to the system, including frameworks, libraries, and third-party integrations.
* Data Encryption: Ensure all data, especially payment and personal details, is encrypted during storage and transmission.
* Access Control: Regularly review and update user roles and permissions to maintain secure access.

3.Database Management

* Regular Backups: Schedule automatic backups to prevent data loss.
* Optimization: Optimize database queries and indexes for better performance.
* Cleanup: Remove outdated or unnecessary data to maintain database efficiency.

4.User Feedback and Feature Updates

* Feedback Integration: Collect and analyze user feedback to identify areas for improvement.
* Feature Enhancements: Roll out updates to improve user experience, such as improved UI/UX or new functionality like real-time order tracking.

5. System Updates

* Version Upgrades: Update the system dependencies, libraries, or frameworks to their latest stable versions.
* Compatibility Checks: Ensure the system remains compatible with updated devices, browsers, or payment gateways.

6. Incident Management

* Support Team: Establish a dedicated support team for resolving user complaints or technical issues.
* Incident Logs: Maintain detailed logs of incidents to identify patterns and prevent future occurrences.

7. Training and Documentation

* Staff Training: Provide regular training to staff managing the system on new features or updates.
* Comprehensive Documentation: Maintain updated documentation for developers and stakeholders.

8. Legal Compliance

* Data Protection Laws: Ensure compliance with local regulations like GDPR or CCPA.
* Payment Compliance: Adhere to payment standards like PCI DSS.

9. Periodic Testing

* Load Testing: Simulate peak usage scenarios to check system robustness.
* Regression Testing: Conduct tests to ensure new updates do not introduce bugs.