CSCI-GA.2433-001 Database Systems

Part IV Final Report

Food Delivery Platform

Date: 12/20/24

Authors:

Weiyi Li - wl3307@nyu.edu
Zitong Luo - zl5847@nyu.edu
Beisong Liu - bl1986@nyu.edu

GitHub:

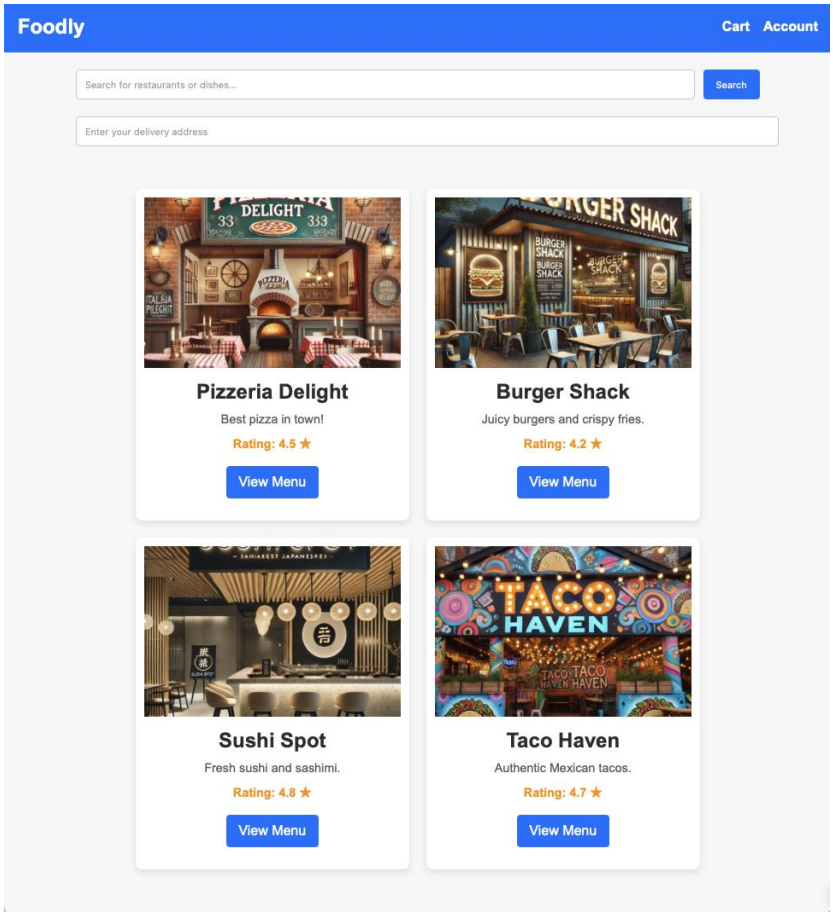
https://github.com/luozt20/2433-Database-System---Final-Project

Use Case

Our food delivery platform aims to connect customers with restaurants, enabling a seamless ordering, delivery, and feedback process. Below are the key business use cases for this data-driven, workflow-based platform:

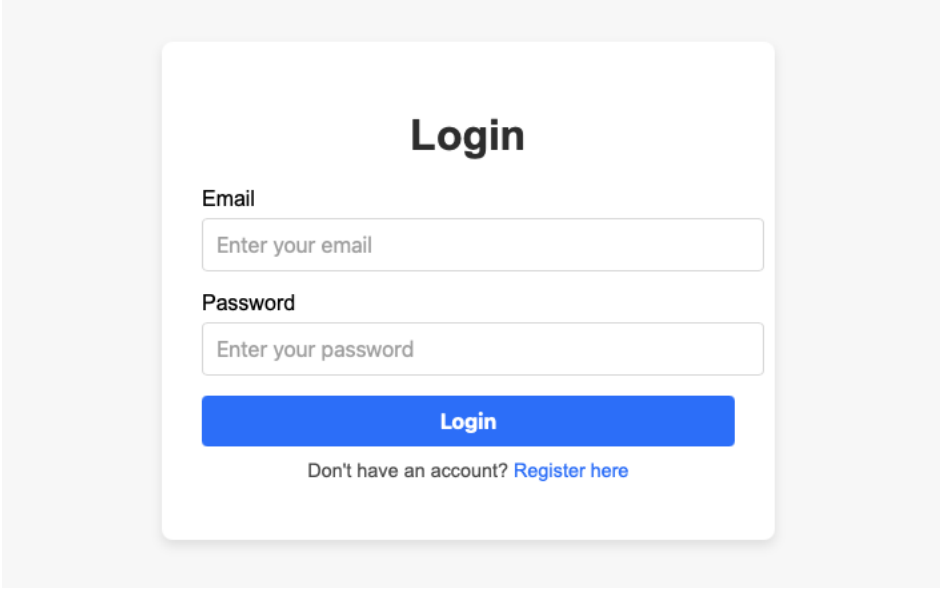
1. **Home page**

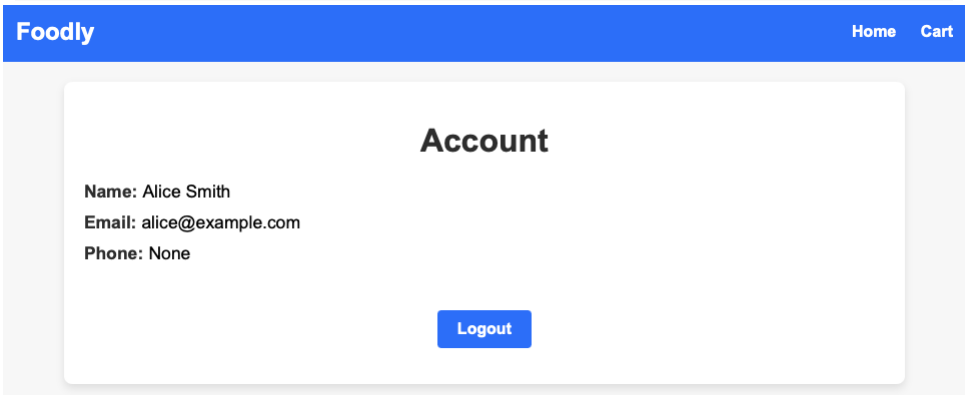
    - Customers can enter their delivery address to see restaurants nearby, search restaurants and view their cart and account information
    - The name, ratings, brief introduction of restaurants are listed



2. **Customer Registration and Login**

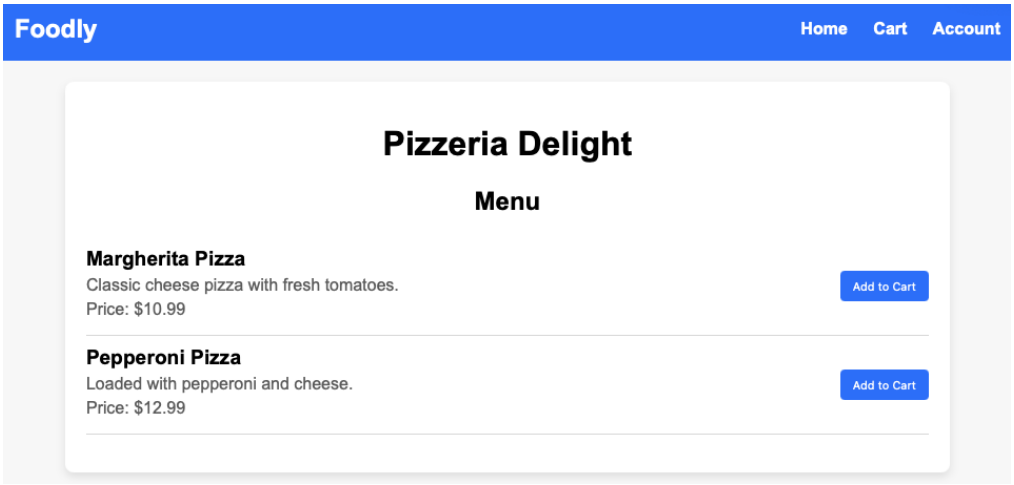
    - Customers can register and log in to the platform, and view their account information in “Account”.
    - Customer database is updated on a real-time basis to make sure data consistency, accuracy and security.





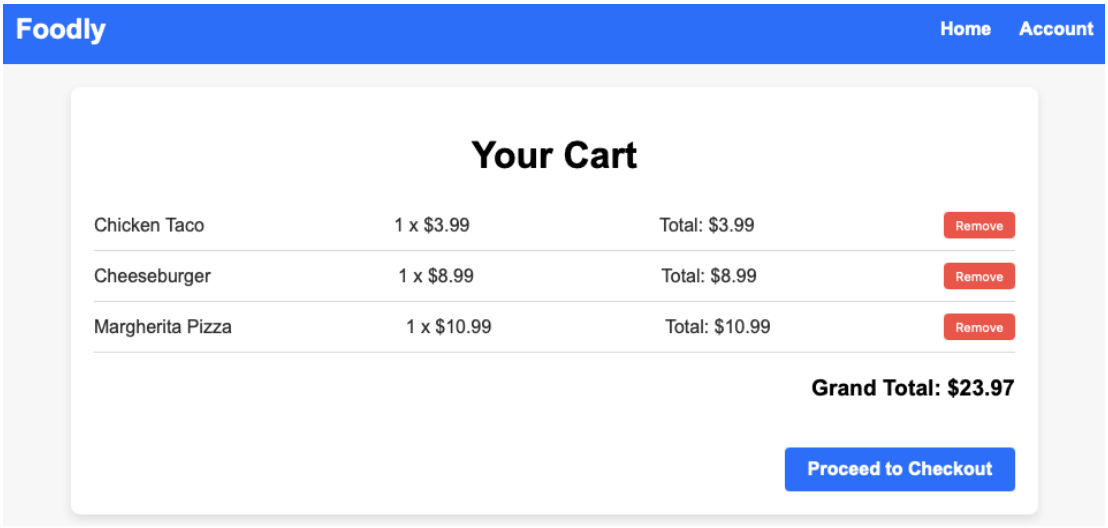
3. Restaurant and Menu Browsing

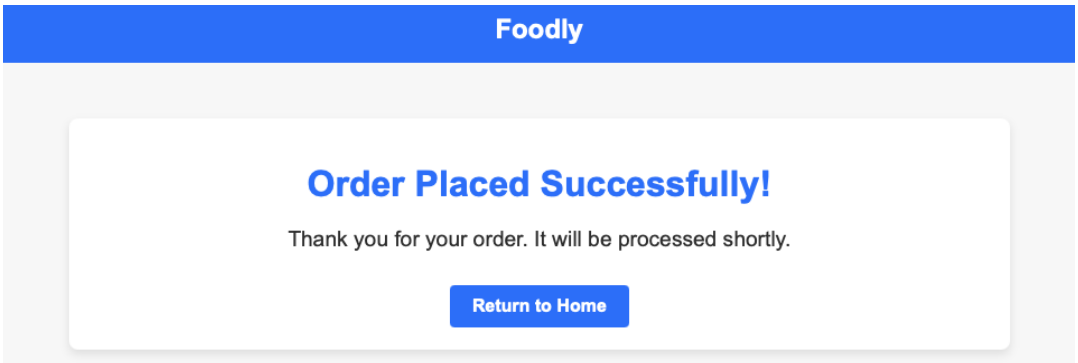
- Customers can browse restaurant listings, view menus and add items to the cart “Cart”
- At the “Cart” page, customers can proceed to checkout or edit the list of items.

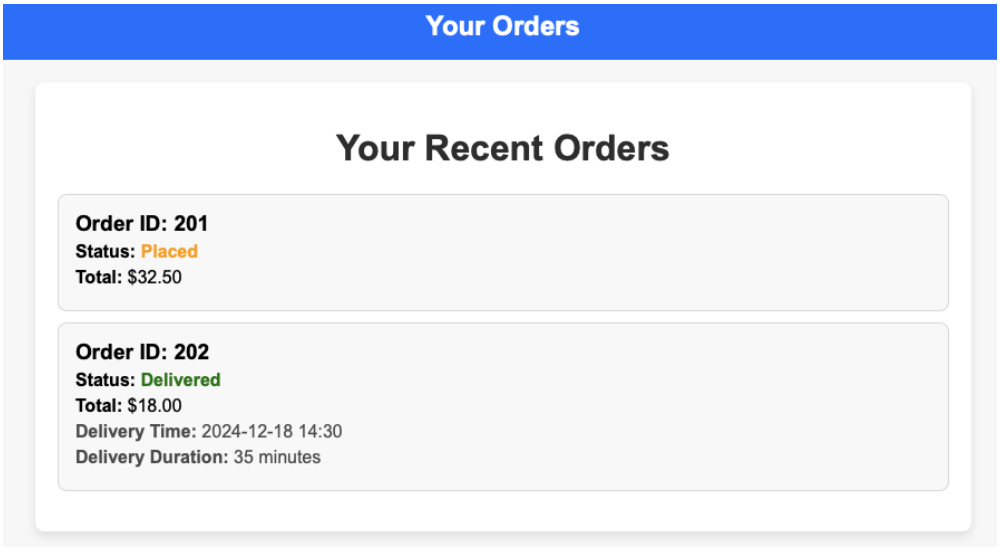


4. Order Placement

- Customers can place orders, track order status, and manage past orders.
- The delivery time of each order is recorded for future analysis and business decision-making

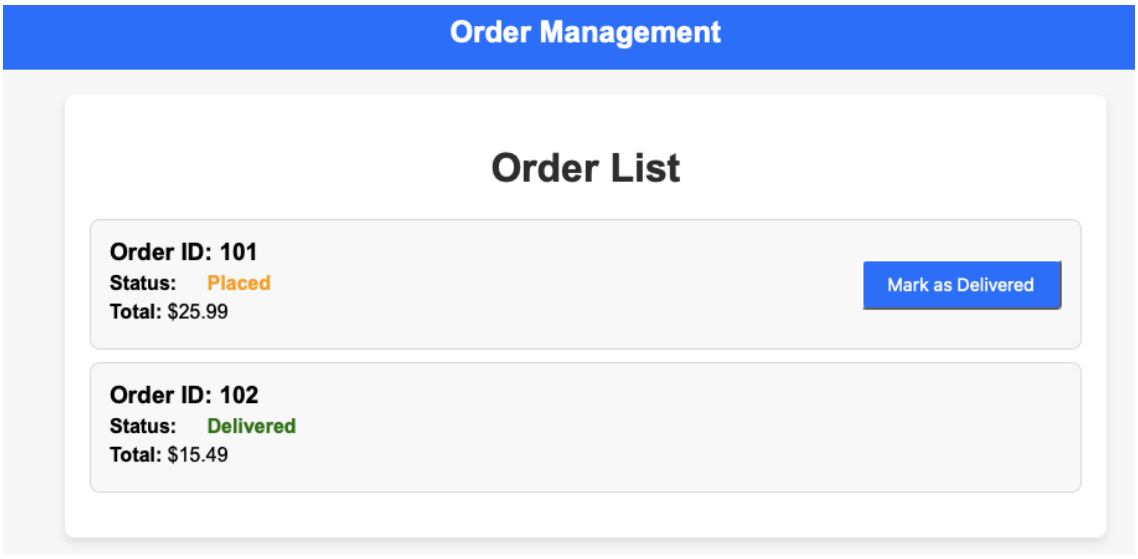






5. Delivery Management

- Delivery personnel can view and update delivery statuses. They click the “Mark as Delivered” when an order is delivered.



6. Promotion and Discounts

- Restaurants can offer promotions, and customers can apply discount codes at checkout.

Implementation

We developed the food delivery platform using Flask for the backend and HTML/CSS for the frontend. The platform integrates with an SQLite database to manage users, restaurants, menus, orders, and cart items. Below is a breakdown of the architecture and functionality:

Backend Implementation

1. **Flask Framework**:
    - Flask was used to handle HTTP requests and responses. The platform consists of routes that correspond to various pages such as home, login, cart, checkout, and restaurant menus.
    - Flask's session management system is used to maintain the state of logged-in users. Users are required to log in to place an order or view their cart.
2. **Database**:
    - The app interacts with an SQLite database to interact with the database, allowing for operations such as fetching restaurant details, adding items to the cart, and placing orders.
    - Tables include Restaurant, MenuItem, Order, and OrderItem, among others.
3. **Key Features**:
    - **User Authentication**: Users log in using their email and password, and their session is maintained across requests. If they are not logged in, they are redirected to the login page.
    - **Order Management**: Users can add items to their cart, remove them, and proceed to checkout. Once an order is confirmed, it is stored in the database, and the order status is updated.
4. **Routes and Views**:
    - /home: Displays a list of restaurants fetched from the database. Users can browse restaurant details and view their menus.
    - /restaurant/<restaurant_id>: Displays the menu of a selected restaurant. Users can add items to their cart.
    - /cart: Displays the items currently in the user's cart. Users can remove items or proceed to checkout.
    - /checkout: Displays a summary of the cart with the option to confirm the order.

- /order-success: Displays a success message after an order is placed.

Frontend Implementation

The frontend is built using HTML and CSS, and it renders dynamic content using Flask's templating engine. Below are the key templates:

1. home.html:
    - Displays a list of restaurants available for delivery. Each restaurant is shown with its name, description, rating, and an image.
    - A search bar allows users to search for restaurants or dishes.
2. restaurant_menu.html:
    - Displays the menu for a specific restaurant, showing the items along with their names, descriptions, and prices. Users can add items to their cart from this page.
3. cart.html:
    - Shows the items currently in the user's cart with details such as quantity and total price. Users can remove items from their cart or proceed to checkout.
4. checkout.html:
    - Displays a summary of the cart, showing the items and their total price. Users can confirm the order by submitting the form.
5. checkout_success.html and order_success.html:
    - These pages display a confirmation message once an order is successfully placed, thanking the user for their purchase and offering a link back to the home page.
6. login.html:
    - Displays the login form for users to sign in with their email and password. Upon successful login, users are redirected to the home page or the page they were trying to access.
7. account.html:
    - Shows the user's account details (name, email, phone number) and provides an option to log out.

Flow of Operation

1. User Login:
    - Users must log in to access their account and place orders. If they are not logged in, they are redirected to the login page.

2. Restaurant and Menu Browsing:
    - After logging in, users can browse available restaurants on the home page, view their menus, and select items to add to their cart.
3. Cart Management:
    - Users can view, add, or remove items from their cart. The cart displays the items along with the total price.
4. Checkout and Order Placement:
    - After reviewing the cart, users can confirm their order by proceeding to checkout. Once confirmed, the order is placed and stored in the database, and the user receives a success message.
5. Order Confirmation:
    - After successfully placing an order, the user is shown a confirmation page with a thank-you message and a link to return to the homepage.

Physical Database Design

1. **Database System Configuration:**
    - We set the basic block size to 8KB which is the fundamental unit of data storage
    - We allocate 4GB for the database cache to keep frequently accessed data in memory
    - We configure total memory allocation of 8GB for overall database operations

2. **Tablespace Organization:**
    - We create separate tablespaces for different types of data:
        - User data gets 100MB initial space with ability to grow to 1GB
        - Restaurant data starts at 200MB and can grow to 2GB
        - Order data begins with 500MB and can expand to 5GB
    - Each tablespace automatically manages its space to avoid fragmentation

3. **Table Storage Design:**
    - For frequently accessed tables like IndividualUser:
        - We allocate 64MB initial space
        - We allow it to grow in 64MB increments
        - We reserve 10% free space in each block for future updates
        - We keep these tables in memory for faster access

4. **Table Partitioning:**
    - Large tables are divided into smaller, manageable pieces:
        - Order table is partitioned by year (2023, 2024, future)
        - Restaurant table is partitioned by region (north, south, west)
        - This makes data retrieval and maintenance more efficient

5. **Index Organization:**
    - We create indexes for frequently searched columns
    - Each index is placed in the same tablespace as its table
    - Indexes are created for:
        - Email addresses for quick user lookup
        - Order status for tracking
        - Payment status for monitoring

6. **Memory Management:**
    - Frequently accessed tables are kept in memory
    - We use different memory areas:
        - Keep pool for important data
        - Default pool for regular data
        - Each has specific size allocations

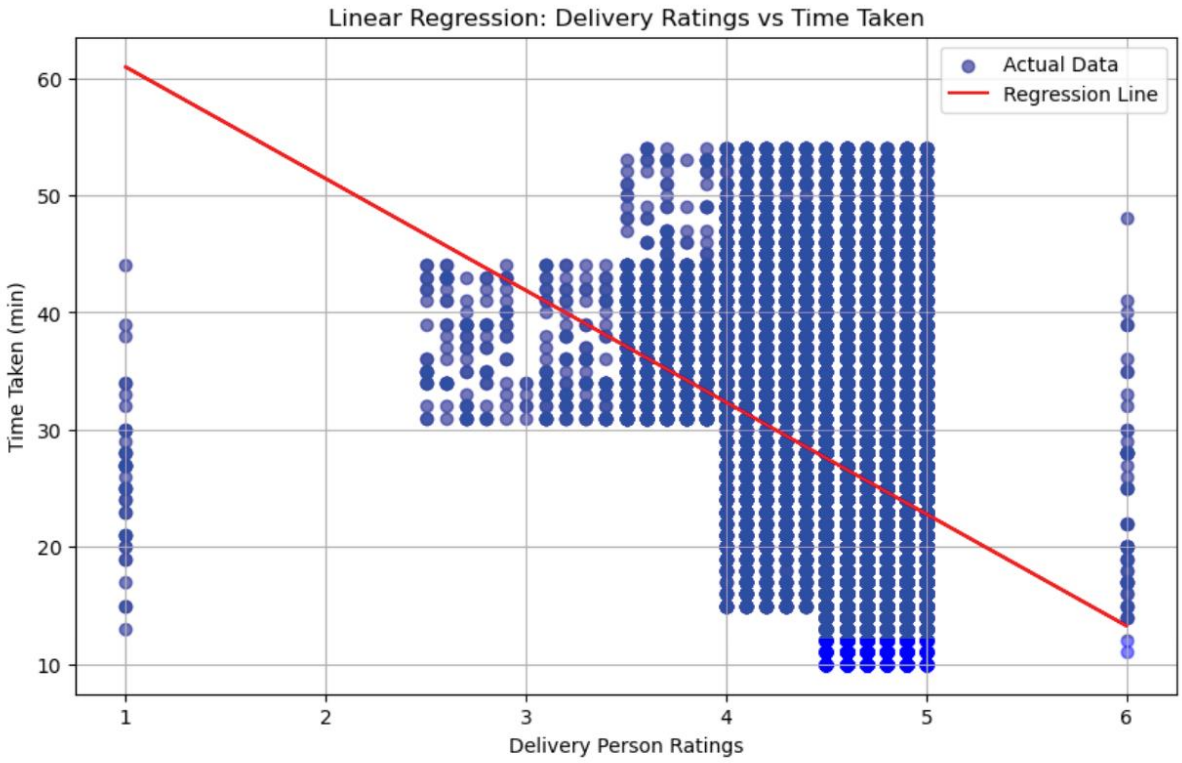
7. **Performance Features:**
    - We implement statistics gathering for query optimization
    - We set up daily maintenance windows for upkeep
    - We configure backup and recovery parameters

Machine Learning to Provide Business Insights

Machine learning (ML) provides actionable business insights for our food delivery platform, helping to optimize operations, enhance customer satisfaction, and drive efficiency.

1. Customer Satisfaction Analysis

- Faster deliveries correlate with higher ratings, emphasizing the need for logistical efficiency.
- **Business Implications**:
    - Invest in route optimization and faster vehicles.
    - Highlight estimated delivery times during order placement.

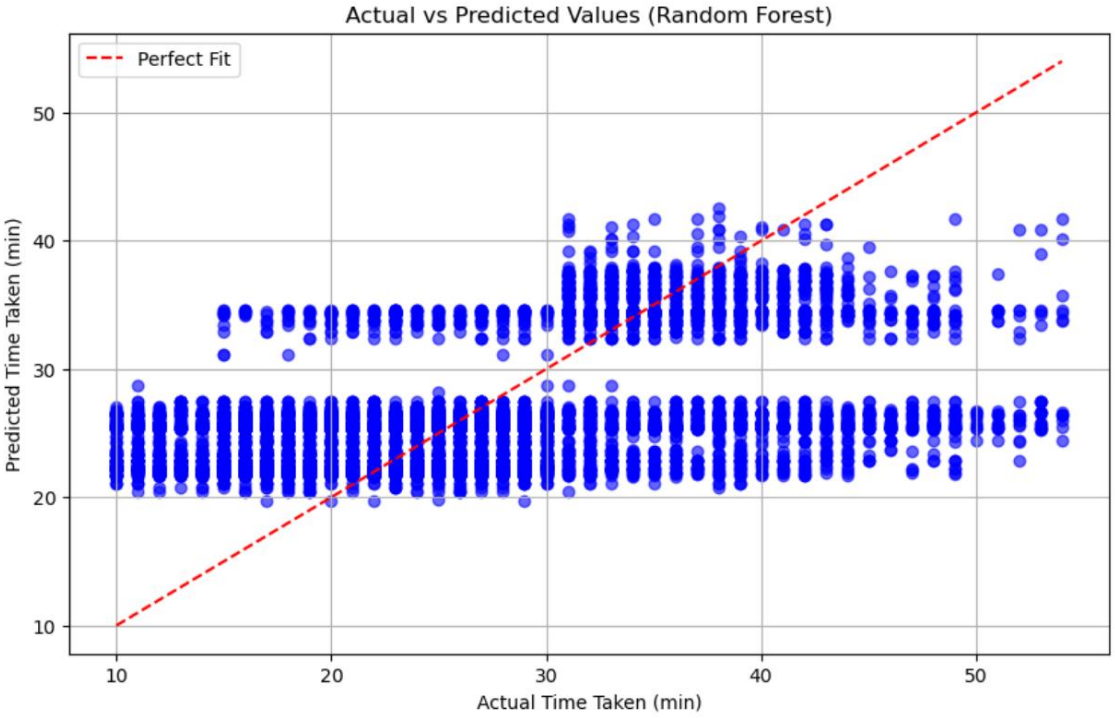



```
Mean Squared Error (MSE): 78.52949712994987
R² Score: 0.10434345368386344
```

2. Delivery Time Prediction

- **Factors**: Delivery person ratings, order type, and vehicle type significantly influence delivery times which are recorded and collected from customer orders.
- **Business Implications**:

- Assign time-sensitive orders to highly rated personnel.
- Use faster vehicles like motorcycles for urban deliveries.
- Set realistic customer expectations based on predictions.



3. Operational Recommendations

- Train delivery personnel on route optimization to improve efficiency and ratings.
- Forecast demand patterns to allocate resources effectively.

4. Feature Enhancements

- **Real-Time Predictions**: Incorporate live traffic data for dynamic delivery estimates.
- **Advanced Models**: Use deep learning for improved demand forecasting and efficiency.
- **Expanded Features**: Factor in weather and regional data for better accuracy.

Data Governance

1. **Data Ownership**
    - Customer Data: Chief Customer Officer
    - Restaurant Data: Restaurant Partnership Director
    - Order & Payment Data: Operations/Finance Directors Added Details:
    - Each owner responsible for data accuracy and access approvals
    - Monthly data quality reports to owners
    - Regular review of access permissions

2. **Data Quality**
    - Set quality standards (email validity, address accuracy)
    - Regular data audits
    - Automated validation checks
    - Data completeness check: 98% minimum target
    - Weekly data cleansing routines
    - Monthly quality score reporting

3. **Security**
    - Data Classification:
        - High Risk: Payment data, passwords
        - Medium Risk: Order history, addresses
        - Low Risk: Public menus, ratings
    - Access Control Levels based on job roles
    - Encryption for all high-risk data
    - 90-day password rotation policy
    - Access logs maintained for 1 year

4. **Compliance**
    - Data Retention: Customer (3 years), Orders (5 years)
    - Regular compliance audits
    - Payment security standards (PCI DSS)
    - Quarterly compliance reviews
    - Automated retention policy enforcement
    - Annual external audit

5. Monitoring

- System performance tracking
- Data quality metrics
- Regular reporting schedule
- Daily performance dashboards
- Alert system for data anomalies
- Monthly trend analysis

6. Recovery Plan

- Regular backups
- Disaster recovery procedures
- Business continuity planning
- 15-minute recovery point objective
- Monthly recovery testing
- 24/7 support team availability

Future Improvements

1. Scalability:

- Deploy application to Azure for higher availability.
- Introduce load balancing for API endpoints.

2. Advanced Analytics:

- Use real-time analytics for demand forecasting.
- Enhance ML models with deep learning techniques for better predictions.

3. Feature Expansion:

- Add support for multiple delivery zones.
- Enable personalized recommendations systems based on user preferences.