Industrial Internship Report on

”Food Delivery System”

Prepared by

Gone Sreshta

Executive Summary
This report provides details of the Industrial Internship provided by upskill Campus and The IoT Academy in collaboration with Industrial Partner UniConverge Technologies Pvt Ltd (UCT). This internship was focused on a project/problem statement provided by UCT. We had to finish the project including the report in 6 weeks’ time. My project was Food Delivery System. Food delivery is a courier service in which a restaurant, store, or independent food-delivery company delivers food to a customer. This internship gave me a very good opportunity to get exposure to Industrial problems and design/implement solution for that. It was an overall great experience to have this internship.

TABLE OF CONTENTS

1 Preface

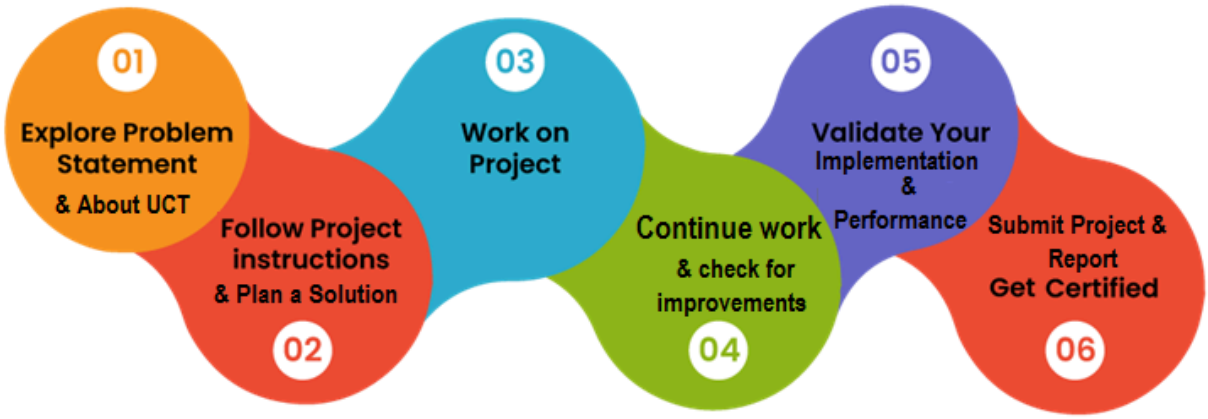
It was a good experience learning from this internship.

It is very important to have relevant Internship in career development and to explore the things deeper.

My project was Food Delivery System. Food delivery is a courier service in which a restaurant, store, or independent food-delivery company delivers food to a customer.

Opportunity given by USC/UCT.

How Program was planned



I have done internship on Full stack Development and learnt a lot from it and also built an application using python.

Thanks to all especially Ujjwal Mishra sir, and also who have helped you directly or indirectly.

My suggestion for juniors : it is the best way to learn new technologies and gain hands on experience.

2 Introduction

2.1 About UniConverge Technologies Pvt Ltd

A company established in 2013 and working in Digital Transformation domain and providing Industrial solutions with prime focus on sustainability and RoI.

For developing its products and solutions it is leveraging various **Cutting Edge Technologies e.g. Internet of Things (IoT), Cyber Security, Cloud computing (AWS, Azure), Machine Learning, Communication Technologies (4G/5G/LoRaWAN), Java Full Stack, Python, Front end** etc.

i. UCT IoT Platform (Insight

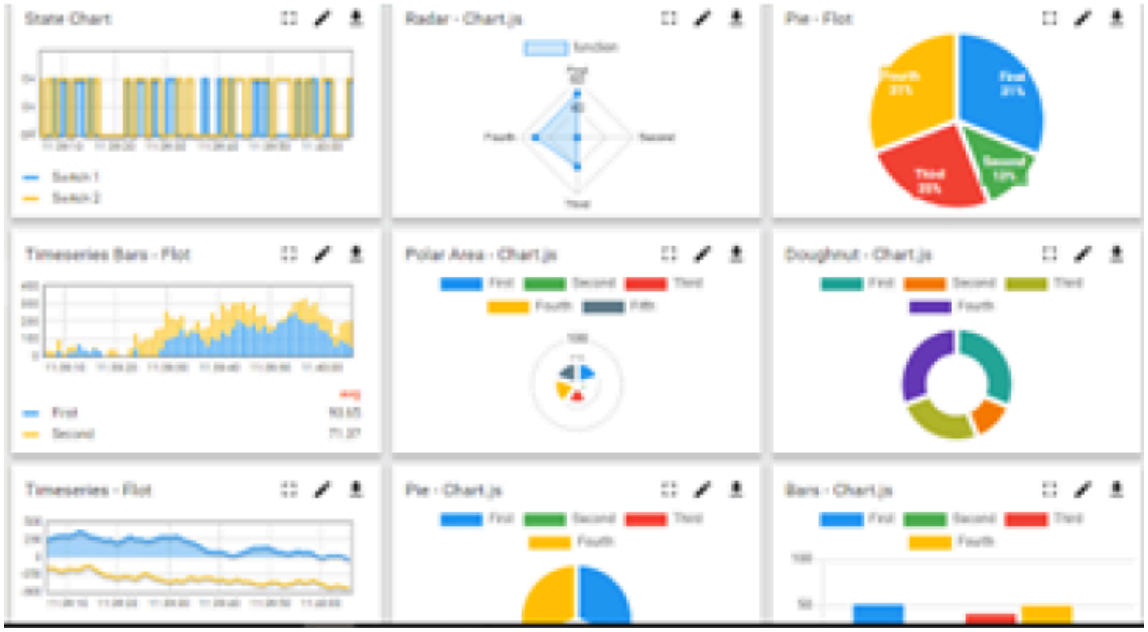
UCT Insight is an IOT platform designed for quick deployment of IOT applications on the same time providing valuable “insight” for your process/business. It has been built in Java for backend and ReactJS for Front end. It has support for MySQL and various NoSql Databases.

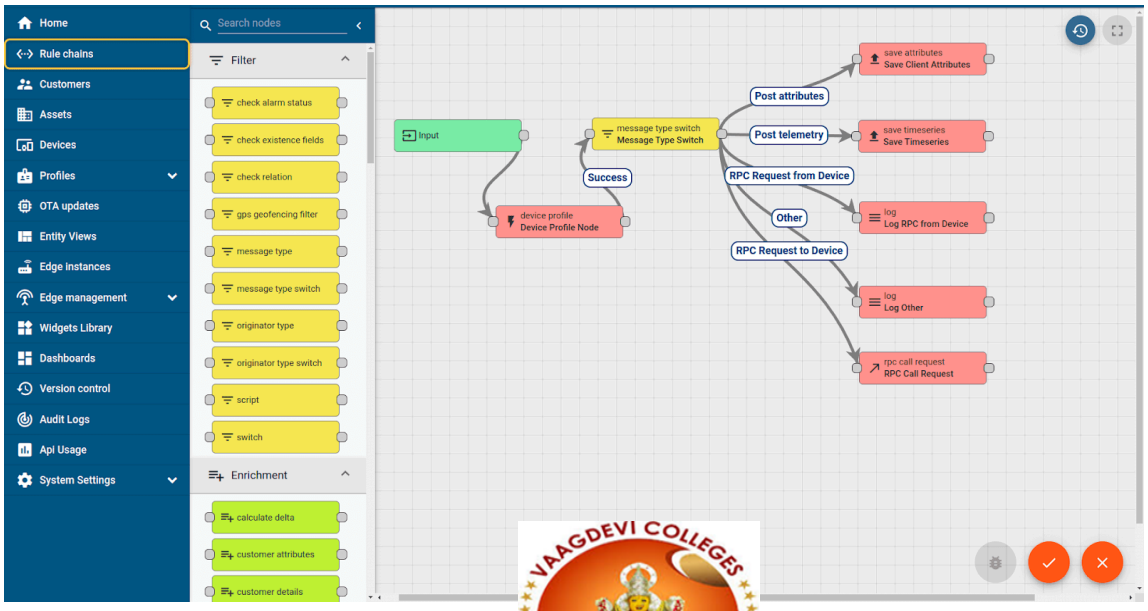
- It enables device connectivity via industry standard IoT protocols - MQTT, CoAP, HTTP, Modbus TCP, OPC UA

- It supports both cloud and on-premises deployments.

It has features to

- Build Your own dashboard
- Analytics and Reporting
- Alert and Notification
- Integration with third party application(Power BI, SAP, ERP)
- Rule Engine





ii. **Smart Factory Platform (** FACTORY WATCH **)**

Factory watch is a platform for smart factory needs.

It provides Users/ Factory

- with a scalable solution for their Production and asset monitoring
- OEE and predictive maintenance solution scaling up to digital twin for your assets.
- to unleased the true potential of the data that their machines are generating and helps to identify the KPIs and also improve them.
- A modular architecture that allows users to choose the service that they what to start and then can scale to more complex solutions as per their demands.

Its unique SaaS model helps users to save time, cost and money.

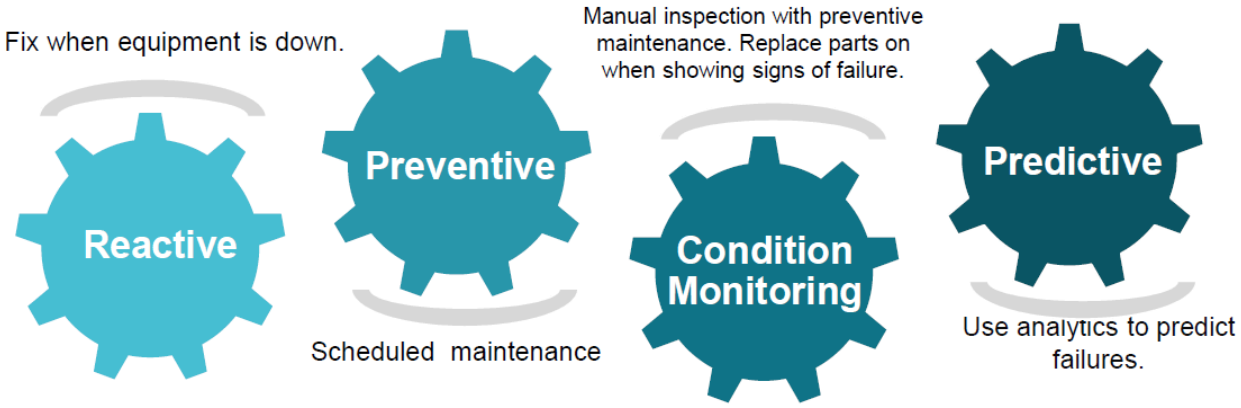
Machine	Operator	Work Order ID	Job ID	Job Performance	Job Progress		Output		Rejection	Time (mins)				Job Status	End Customer
					Start Time	End Time	Planned	Actual		Setup	Pred	Downtime	Idle		
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i

iii. LoRaWAN based Solution

UCT is one of the early adopters of LoRAWAN teschnology and providing solution in Agritech, Smart cities, Industrial Monitoring, Smart Street Light, Smart Water/ Gas/ Electricity metering solutions etc.

iv. Predictive Maintenance

UCT is providing Industrial Machine health monitoring and Predictive maintenance solution leveraging Embedded system, Industrial IoT and Machine Learning Technologies by finding Remaining useful life time of various Machines used in production process.

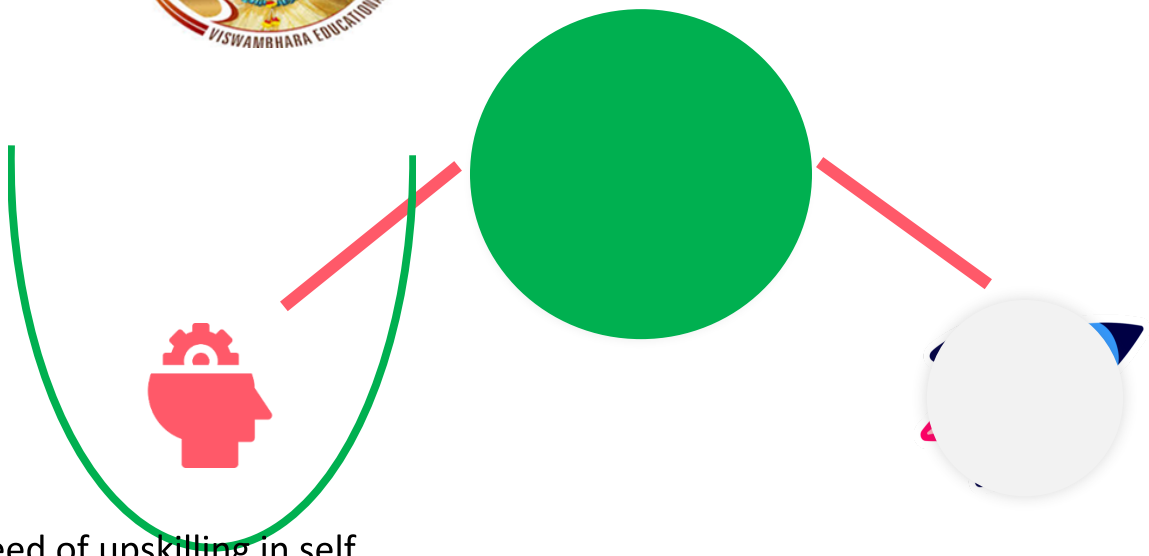


2.2 About upskill Campus (USC)

upskill Campus along with The IoT Academy and in association with Uniconverge technologies has facilitated the smooth execution of the complete internship process.

USC is a career development platform that delivers **personalized executive coaching** in a more affordable, scalable and measurable way.

College Logo]



Seeing need of upskilling in self paced manner along-with additional support services e.g. Internship, projects, interaction with Industry experts, Career growth Services

upSkill Campus aiming to upskill 1 million learners in next 5 year

https://www.upskillcampus.com/

2.3 The IoT Academy

The IoT academy is EdTech Division of UCT that is running long executive certification programs in collaboration with EICT Academy, IITK, IITR and IITG in multiple domains.

2.4 Objectives of this Internship program

The objective for this internship program was to

- ☛ get practical experience of working in the industry.
- ☛ to solve real world problems.
- ☛ to have improved job prospects.
- ☛ to have Improved understanding of our field and its applications.
- ☛ to have Personal growth like better communication and problem solving.

2.5 Reference

[1] Reference1

[2] Reference2

[3] Reference3

2.6 Glossary

Terms	Acronym
P2C	Platform to Customer
R2C	Restaurant-to-Consumer
GPC	Global Positioning System
UI/UX	User Interface/User Experience
POS	Point of Sale

3 Problem Statement

In the assigned problem statement

Developing a food delivery system involves addressing challenges that ensure seamless interactions between customers, restaurants, and delivery partners. Below is a concise problem statement:

"Design a comprehensive and scalable food delivery system that connects customers with a wide variety of restaurants, facilitates efficient order placement and real-time tracking, and ensures timely delivery while maintaining food quality and minimizing operational costs."

Key Challenges to Address:

1. **User Experience (UX):** Creating an intuitive platform for users to browse menus, customize orders, and make payments.
2. **Delivery Logistics:** Optimizing delivery routes and managing delays due to traffic or weather conditions.
3. **Restaurant Integration:** Ensuring smooth integration with restaurant systems for inventory, order management, and menu updates.
4. **Real-Time Tracking:** Providing accurate GPS tracking for orders to enhance transparency.
5. **Scalability:** Building a system that can handle a growing user base and fluctuating demand during peak hours.
6. **Payment Security:** Implementing secure and diverse payment methods to build customer trust.
7. **Customer Support:** Offering prompt assistance to address complaints and queries efficiently.

This problem statement and its components can serve as a foundation for creating a functional and user-friendly food delivery system. Let me know if you'd like examples or references for similar systems!

4 Existing and Proposed solution

Existing Solutions for Food Delivery Systems

Aggregator Platforms (e.g., Uber Eats, DoorDash, Grubhub):

- How they work: These platforms act as intermediaries between customers and restaurants. Customers can browse a variety of menus from different restaurants and place their orders through a central app or website. The platform handles order placement, payment processing, and delivery logistics via third-party drivers.
- Challenges: Issues include high delivery fees, quality inconsistency, dependency on third-party delivery drivers, and late deliveries. Additionally, restaurants often have to pay a commission fee, which can impact their profitability.

Proposed /Optimized Delivery Logistics Using AI and Automation:

- **How it works**: AI and machine learning can be used to optimize delivery routes, predict busy times, and even automatically dispatch the best-suited delivery drivers. This would help reduce delays and increase the efficiency of food deliveries.
- **Proposed Benefits**: Faster deliveries, reduced delivery costs, and improved customer satisfaction.
- **Example**: AI-powered systems like those being implemented by Uber Eats and DoorDash to predict demand and streamline delivery .

Hybrid Delivery Combination of Aggregators and In-House Delivery:

- **How it works**: Restaurants could partner with platforms for marketing and order aggregation while also using their own delivery fleet. This hybrid system would allow restaurants to maintain better control over delivery times and customer service while still benefiting from the reach of platforms.

4.1 Code submission link

4.2 Report submission link

4.2

5 Proposed Design/ Model

The proposed food delivery system incorporates various components to provide an efficient, scalable, and user-friendly service. Below is a breakdown of the design structure, which includes both high-level architecture and technical components.

System Architecture Overview

The system can be broken down into four primary components:

1. **User Interface (Front-End)**:
    - Customers interact with the platform through a **web application** or **mobile app** (iOS/Android).
    - **Restaurant dashboard** for restaurant owners to manage orders, update menus, and track deliveries.
    - **Admin panel** to manage users, restaurant listings, and overall system operations.
2. **Back-End (Server-Side)**:
    - The back-end handles the **business logic**, user requests, order processing, and API endpoints.
    - **Database** stores user, order, restaurant, and payment information.
    - **Payment Gateway** integrates with third-party services to handle transactions securely.
    - **Order Management System (OMS)** manages order status and coordinates between restaurants, customers, and delivery drivers.
3. **Delivery Logistics**:
    - **AI-powered route optimization** ensures drivers take the most efficient paths.
    - **Driver Tracking** provides real-time GPS updates to customers and admins.
    - **Green Logistics**: Electric vehicles or bicycles used for deliveries to reduce carbon footprint.
4. **External Integrations**:

- **Payment Gateways**: Supp ds, digital wallets, and even cryptocurrency.

5.1 High Level Diagram (if applicable)

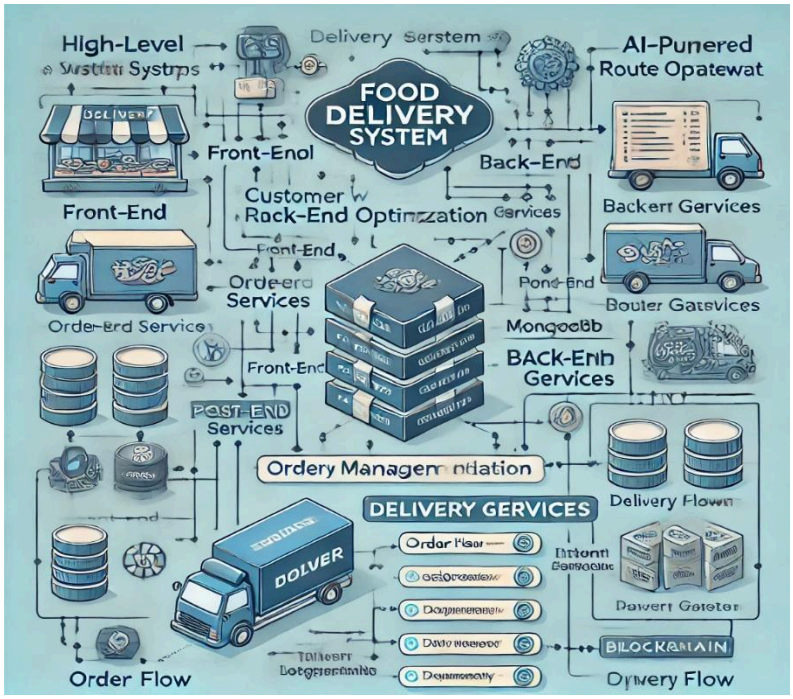
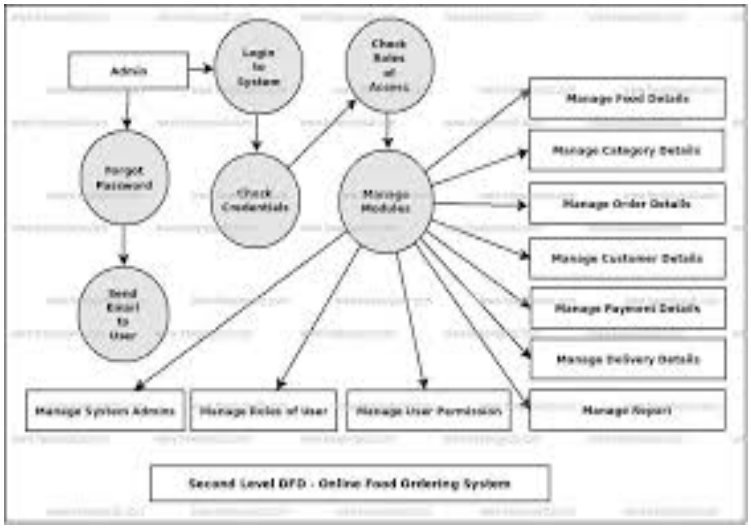


Figure 1: HIGH LEVEL DIAGRAM OF THE SYSTEM

5.2 Low level Diagram (if applicable)



5.3 Interfaces (if applicable)

A food delivery app typically has several interfaces that cater to both customers and service providers (drivers, restaurants). Here are the key interfaces and their features:

- **1. User (Customer) Interface:**
- **Home Screen:**
    - Search bar for restaurants, cuisines, or dishes.

- Categories (e.g., fast food, fine dining, desserts, etc.).
- Featured or recommended restaurants and promotions.

- **Restaurant Listings:**
  - Restaurant name, image, and ratings.
  - Filter options (price range, delivery time, ratings)
  - Sorting options (distance, p

- **Restaurant (Merchant) Interface:**
- **Dashboard:**

Overview of recent orders.

- Current order status (pending, in progress, delivered).

- **Order Management:**
  - Accept/reject orders.
  - Real-time updates for the kitchen (order received, preparing, ready to deliver).
- **Menu Management:**
  - Add, edit, or remove dishes.
  - Set pricing and availability (e.g., special items or daily deals).
- **Reports and Analytics:**
  - Sales data and performance metrics.
  - Customer feedback and ratings.
- **Driver (Delivery Partner) Interface:**
- **Dashboard:**
  - Available orders nearby.
  - Active orders with pickup and delivery locations.

- **Navigation:**
    - In-app GPS navigation to restaurants and delivery addresses.
    - Traffic and route optimization.

6 Performance Test

This is very important part and defines why this work is meant of Real industries, instead of being just academic project.

Here we need to first find the constraints.

Load testing assesses how the system performs under expected traffic volumes. For a food delivery system, the typical traffic includes:

- **Peak Load**: A large number of users browsing restaurants, placing orders, and making payments simultaneously (e.g., during meal times).
- **Concurrent Users**: Test how many users can interact with the system concurrently without degradation in performance.

Tools for Load Testing:

- **Apache JMeter**: Can simulate multiple users accessing the platform, browsing, ordering, and making payments.
- **Gatling**: Another powerful tool for simulating user interactions and analyzing system performance under load.

Test Scenarios:

- Simulating 100,000 users browsing the restaurant menu at once.
- Simulating 50,000 users placing orders in a short period.

Key Metrics to Measure:

- Response time (time to load the web page or complete a transaction).
- Throughput (requests per second).
- Resource utilization (CPU, memory, disk, etc.).

6.3 Test Plan/ Test Cases

1. **Test Case 1: User Registration**
    - **Description**: Verify that customers and restaurant admins can register successfully.
    - **Steps**:
        1. Open the registration page.
        2. Enter valid user details.
        3. Submit the form.
    - **Expected Result**: User account is created successfully, and a confirmation message is shown.
2. **Test Case 2: Browse Restaurants**
    - **Description**: Ensure that customers can view available restaurants.
    - **Steps**:
        1. Open the app/website.
        2. Search or browse through restaurants.
    - **Expected Result**: Restaurants appear with valid details like name, ratings, menu items.
3. **Test Case 3: Place an Order**
    - **Description**: Verify customers can place an order.
    - **Steps**:
        1. Add items to the cart.
        2. Proceed to checkout.
        3. Select delivery method and payment option.
        4. Complete the order.

- **Expected Result**: Order and a confirmation message is displayed.

4. **Test Case 4: Order Status Update**
    - **Description**: Verify that the restaurant can update the status of an order (e.g., Preparing, Out for Delivery).
    - **Steps**:
        1. The restaurant logs into the system.
        2. Select the order and update the status.
    - **Expected Result**: The order status is updated and visible to the customer.
5. **Test Case 5: Payment Processing**
    - **Description**: Validate the system handles payments correctly.
    - **Steps**:
        1. Customer places an order.
        2. Selects a payment method (e.g., credit card, PayPal).
        3. Completes the payment process.
    - **Expected Result**: Payment is processed successfully, and the transaction is recorded.

---

6. **Test Case 6: Load Testing (Concurrent Users)**
    - **Description**: Test system performance when handling multiple users.
    - **Steps**:
        1. Simulate 1000, 5000, and 10000 concurrent users browsing and placing orders.
    - **Expected Result**: The system should maintain acceptable performance (response time under 3 seconds per page).

6.4 Test Procedure

The test procedure outlines the systematic steps to be followed when testing the food delivery system. It covers functional, performance, security, and usability tests, ensuring the system works as intended and meets the specified requirements. Here is a detailed test procedure for various testing types:

- **1. Preparation Phase**

1.1 Test Environment Setup:

- **Install and configure required tools** (e.g., JMeter, Selenium, Postman).
- **Set up the test servers** (production, staging, or testing environments) where the system components will be deployed.
- **Install the application** (web/mobile client, server-side, database).
- **Configure databases** (MySQL/PostgreSQL for relational data, MongoDB for unstructured data).
- **Prepare test data** (sample user profiles, restaurant menus, orders, and payment details).
- **Set up monitoring tools** (e.g., New Relic, Prometheus, or Grafana) to observe system performance.

1.2 Test Case Identification:

- Identify the test cases to be executed based on the requirements and test plan (functional, performance, security, and database tests).

- **2. Functional Testing Procedure**

2.1 User Registration Testing:

- **Objective**: Ensure the user can successfully register in the system.
- **Steps**:
    1. Navigate to the registration page on the website or app.
    2. Input valid user details (name, email, password, etc.).
    3. Submit the registration form.

4. Verify that the system sends a confirmation email or message to the user.

- **Expected Result**: Registration is successful, and the user is redirected to the login screen.

2.2 Order Placement Testing:

- **Objective**: Ensure that the order process works correctly.
- **Steps**:
    1. Log in as a customer.
    2. Browse the menu and select items.
    3. Add items to the cart and proceed to checkout.
    4. Select the delivery method (pickup or delivery).
    5. Enter payment details and complete the purchase.
- **Expected Result**: The order is placed successfully, and the customer receives an order confirmation with an estimated delivery time.

2.3 Payment Gateway Integration Testing:

- **Objective**: Test the integration of the payment gateway (e.g., Stripe, PayPal).
- **Steps**:
    1. During the order placement, select a payment option.
    2. Enter payment details (credit card, debit card, PayPal).
    3. Complete the payment.
    4. Verify that the payment is processed and recorded.
- **Expected Result**: Payment is processed securely, and the transaction is recorded in the system.

- **3. Performance Testing Procedure**

3.1 Load Testing:

- **Objective**: Evaluate system performance under expected traffic.

- **Steps**:
    1. Use **JMeter** or **Gatling** to simulate a number of concurrent users (e.g., 1000 users).
    2. Simulate users browsing the menu, placing orders, and completing payments.
    3. Monitor server performance (response time, CPU, memory usage).
    4. Gradually increase the number of users to simulate peak traffic.
- **Expected Result**: The system should be able to handle the load without significant performance degradation.

6.5 Performance Outcome

The **performance outcome** of a food delivery system is a key indicator of its reliability, scalability, and efficiency. Performance outcomes are measured across various domains, including **response time**, **throughput**, **scalability**, **resource usage**, and **error rates**. Here's an outline of what the expected performance outcomes should be for a food delivery system:

- **1. Response Time**
- **Expected Outcome**: The system should respond within acceptable time frames even under load.
    - **Key metrics**:
        - **Page Load Time**: Less than 2-3 seconds for menu browsing, restaurant details, and checkout pages.
        - **Order Placement**: Order confirmation should be received within 5 seconds after submission.
        - **Payment Processing**: Payment confirmation should be completed within 5-10 seconds.
    - **Impact**: Longer response times can lead to a negative user experience, potentially abandoning carts or orders.

- **2. Throughput**
- **Expected Outcome**: The system should handle a high volume of requests per second without performance degradation.
    - **Key metrics**:
        - **Orders per Minute**: The system should process hundreds or thousands of orders per minute, depending on peak traffic.
        - **Simultaneous Users**: The system should handle thousands of concurrent users browsing menus, placing orders, and making payments without any slowdown or errors.
    - **Impact**: If throughput is too low, the system may become unresponsive during peak traffic times, such as lunch or dinner hours.

- **3. Scalability**
- **Expected Outcome**: The system should scale horizontally or vertically to handle increased load.
    - **Key metrics**:
        - **Horizontal Scaling**: Ability to add more web servers or services without significant performance loss.
        - **Vertical Scaling**: Ability to upgrade server resources (CPU, memory) without significant disruptions to service.
    - **Impact**: The system must scale effectively during high-demand periods (e.g., holidays, sales) to maintain service levels.

- **4. Resource Utilization**
- **Expected Outcome**: The system should use resources efficiently under normal and peak loads.
    - **Key metrics**:

- **CPU Usage**: Should stay under 80% under normal load; higher during peak traffic but should not reach 100%.
- **Memory Usage**: Sl ith minimal memory leaks. Memory usage should not g requests increase.

7. My learnings

Over the course of working with the food delivery system project, I have gained insights into several critical areas of software development, performance optimization, and system design, all of which are pivotal for my career growth. Here is a summary of key takeaways and how they will shape my professional development:

Software Development & Full-Stack Design

I've learned how to structure and build a food delivery system from both the client and server sides. This includes:

- **Front-end technologies**: Building intuitive user interfaces for customers (HTML, CSS, JavaScript).
- **Back-end technologies**: Designing robust back-end systems using Python (Flask/Django), handling requests, payments, and data storage (MySQL, MongoDB).
- **APIs**: Integrating third-party services, such as payment gateways and map services, to enhance functionality and user experience.

Career Impact:

This full-stack development experience is invaluable for positions such as **Full Stack Developer**, **Back End Engineer**, and **Software Architect**, where building integrated systems is essential. The ability to handle both client-facing and server-side concerns will make me a versatile asset in tech teams.

8. Future work scope

Here are some ideas that could not be implemented due to time limitations but can be explored and incorporated in the future development of the food delivery system:

- **1. Advanced AI for Personalized Recommendations**
- **Idea**: Integrate a machine learning algorithm to suggest personalized food options based on users' order history, preferences, dietary restrictions, and even mood (based on time of day or weather).
- **Future Benefits**: This would improve user experience by tailoring the platform to individual tastes and increase user retention by providing relevant suggestions.

- **2. Multi-Vendor Marketplace**
- **Idea**: Develop a multi-vendor feature that allows customers to order from multiple restaurants in a single checkout process. This would be particularly useful for groups who want different types of food (e.g., pizza from one restaurant and burgers from another).
- **Future Benefits**: Increased convenience for users, higher order volumes, and a broader range of customer needs being met in one place.

- **3. Real-Time Delivery Tracking with Augmented Reality (AR)**
- **Idea**: Use augmented reality to display the real-time location of the delivery in the user's app, overlaying the delivery route and estimated time of arrival (ETA) on a map or even within the real-world view through the phone's camera.
- **Future Benefits**: This would provide a more immersive and engaging customer experience, improving trust in delivery times and adding a cutting-edge feature to the platform.