FOODIES SPOT WITH DATA SCIENCE

A Major Project Report Submitted

In partial fulfillment of the requirements for the award of the degree of

**Bachelor of Technology in**

**Computer Science and Engineering**

## by

#### Gottam Sai Charan Reddy 21N31A0579

|  |
| --- |
| **H Prijwal Reddy 21N31A0583** |
| **J Harshith Kumar 21N31A0589** |

Under the esteemed guidance of **Mr. G. Manoj Kumar Associate Professor**

****

**Department of Computer Science and Engineering**

**Malla Reddy College of Engineering & Technology**

(Autonomous Institution- UGC, Govt. of India)

(Affiliated to JNTUH, Hyderabad, Approved by AICTE, NBA &NAAC with ‘A’ Grade)

Maisammaguda, Kompally, Dhulapally, Secunderabad – 500100 website: [www.mrcet.ac.in](http://www.mrcet.ac.in/)

**2024-2025**



**Malla Reddy College of Engineering & Technology**

(Autonomous Institution- UGC, Govt. of India)

(Affiliated to JNTUH, Hyderabad, Approved by AICTE, NBA &NAAC with ‘A’ Grade)

Maisammaguda, Kompally, Dhulapally, Secunderabad – 500100 website: [www.mrcet.ac.in](http://www.mrcet.ac.in/)

# CERTIFICATE

This is to certify that this is the bonafide record of the project entitled “**FOODIES SPOT WITH DATA SCIENCE**”, submitted by Students Gottam Sai Charan Reddy (21N31A0579), H Prijwal Reddy (21N31A0583) and J Harshith Kumar (21N31A0589) of B.Tech in the partial fulfillment of the requirements for the degree of Bachelor of Technology in Computer Science and Engineering, Department of CSE during the year 2024-2025.The results embodied in this project report have not been submitted to any other university or institute for the award of any degree or diploma.

**Internal Guide Head of the Department**

**Mr. G. Manoj Kumar Dr. S. Shanthi Associate Professor Professor**

**External Examiner**

# DECLARATION

We hereby declare that the project titled “**FOODIES SPOT WITH DATA SCIENCE**” submitted to Malla Reddy College ofEngineering and Technology (UGC Autonomous), affiliated to Jawaharlal Nehru Technological University Hyderabad (JNTUH) for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a result of original research carried out in this thesis. It is further declared that the project report or any part thereof has not been previously submitted to any University or Institute for the awardof a degree or diploma.

**Gottam Sai Charan Reddy 21N31A0579 H Prijwal Reddy 21N31A0583**

**J Harshith Kumar 21N31A0589**

## ACKNOWLEDGEMENT

We feel honored to place our warm salutation to our college Malla Reddy College of Engineering and Technology (UGC-Autonomous) for allowing us to do this Project as partof our B. Tech Program. We are ever grateful to our Director **Dr. V.**

**S. K Reddy** and Principal **Dr. S. Srinivasa Rao** who enabled us to have experience in engineering and gainprofound technical knowledge.

We express our heartiest thanks to our HOD, **Dr. S. Shanthi** for encouraging us in everyaspect of our course and helping us realize our full potential.

We would like to thank our Project Guide **Mr. G. Manoj Kumar** for his regular guidance, suggestions and constant encouragement. We are extremely grateful to our project coordinator **Mr. G. Manoj Kumar** for his regular guidance, suggestions, constant encouragement, continuous monitoring, and unflinching cooperation throughout project work.

We would like to thank our Class In charge **Mr. G. Manoj Kumar** who despite being busy with his academic duties took time to guide and keep us on the correct path.

We would also like to thank all the faculty members and supporting staff of the Departmentof CSE and all other departments who have helped directly or indirectly in making our projecta success.

We are extremely grateful to our parents for their blessings and prayers for the completionof our project which gave us the strength to do our project.

With regards and gratitude

**Gottam Sai Charan Reddy 21N31A0579 H Prijwal Reddy 21N31A0583**

**J Harshith Kumar 21N31A0589**

## ABSTRACT

Foodie Spot is a comprehensive web-based application designed to revolutionize food ordering and management in canteens or cafeterias. Built with React.js and Firebase, it offers an intuitive platform for users to browse food items, manage their shopping cart, and seamlessly complete purchases. The system ensures real-time inventory updates, providing accurate stock management for smooth operations.

The application includes personalized order history for users and a complete order management system for administrators. Foodie Spot features an advanced dashboard for sales and profits, offering detailed insights into financial performance. Administrators can visualize data trends using Recharts and track metrics such as weekly, monthly, and yearly sales. Additionally, the platform supports PDF export of order history and reports through JSPDF, enabling offline record-keeping and effective administration.

Designed for convenience, Foodie Spot boasts a responsive layout for seamless access across devices and secure user authentication via Firebase. The admin panel’s robust reporting tools enhance decision-making with analytics on sales performance and profit trends. By integrating user-friendly interfaces, real-time updates, and powerful data-driven insights, Foodie Spot simplifies canteen management while catering to the needs of both users and administrators, making it an essential tool for modern cafeteria operations.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **TITLE** | **PG.NO** |
| **1** | **INTRODUCTION**   * 1. PURPOSE, AIM AND OBJECTIVES   2. BACKGROUND OF PROJECT   3. SCOPE OF PROJECT   4. MODULES DESCRIPTION | **09**  09  10  11  12 |
| **2** | **LITERATURE SURVEY** | **14** |
| **3** | **SYSTEM ANALYSIS**   * 1. HARDWARE AND SOFTWARE REQUIREMENTS   2. SOFTWARE REQUIREMENTS SPECIFICATION | **18**  18  19 |
| **4** | **TECHNOLOGIES USED**   * 1. MACHINE LEARNING   2. LANGUAGES AND FRAMEWORKS | **30**  30  35 |
| **5** | **SYSTEM DESIGN & UML DIAGRAMS**   * 1. SOFTWARE DESIGN   2. ARCHITECTURE   3. UNIFIED MODELING LANGUAGE | **39**  39  39  40 |
| **6** | **IMPLEMENTATION** | **49** |
| **7** | **OUTPUT SCREENS** | **62** |
| **8** | **CONCLUSION & FUTURE SCOPE** | **68** |
| **9** | **BIBLIOGRAPHY** | **69** |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE.NO** | **NAME** | **PG.NO** |
| **5.2** | **SYSTEM FLOW DIAGRAM** | **40** |
| **5.3.2.1** | **CLASS DIAGRAM** | **44** |
| **5.3.2.2** | **USE CASE DIAGRAM** | **45** |
| **5.3.2.3** | **SEQUENCE DIAGRAMS** | **46** |
| **5.3.2.4** | **ACTIVITY DIAGRAM** | **47** |
| **5.3.2.5** | **COMPONENT DIAGRAM** | **48** |

**LIST OF OUTPUTS**

|  |  |  |
| --- | --- | --- |
| **FIGURE.NO** | **NAME** | **PG.NO** |
| **7.1** | **Sign Up Page** | **62** |
| **7.2** | **Login Page** | **62** |
| **7.3** | **Home Page** | **63** |
| **7.4** | **Cart Box** | **63** |
| **7.5** | **Bill PDF** | **64** |
| **7.6** | **My Orders Page** | **64** |
| **7.7** | **Admin Panel Charts** | **65** |
| **7.8** | **Admin Order History** | **65** |
| **7.9** | **Inventory Manager** | **66** |
| **7.10** | **Header** | **66** |
| **7.11** | **Server Authentication** | **67** |
| **7.12** | **Server Real Time DataBase** | **67** |

1. **INTRODUCTION**

Foodie Spot is a cutting-edge web-based application designed to revolutionize food ordering and management in canteens and cafeterias. Leveraging modern technologies like React.js and Firebase, it offers a seamless platform for users to browse food items, manage their shopping cart, and complete purchases with ease. The system incorporates real-time inventory updates to ensure accurate stock management, eliminating manual errors and streamlining operations. With personalized order history, users can revisit their past purchases for a tailored experience, while secure Firebase authentication guarantees data privacy and reliability.

For administrators, Foodie Spot provides a powerful suite of tools to optimize management. The advanced dashboard features data-driven insights into sales and profits using Recharts, enabling tracking of weekly, monthly, and yearly trends. PDF export functionality, powered by JSPDF, facilitates offline record-keeping and robust reporting. Designed with a responsive layout, the platform ensures smooth access across devices, making it convenient for both users and administrators. By combining intuitive interfaces, real-time updates, and comprehensive analytics, Foodie Spot sets a new standard in canteen management, enhancing efficiency and satisfaction for all stakeholders.

* 1. **PURPOSE, AIM AND OBJECTIVES:**

##### Purpose

The primary purpose of *Foodie Spot* is to revolutionize food ordering and management in canteens or cafeterias by introducing a streamlined, digital solution. By replacing traditional manual processes with a robust, web-based application, it addresses common challenges such as inventory mismanagement, order delays, and the lack of data-driven insights. The platform bridges the gap between user convenience and administrative efficiency, creating a system that benefits all stakeholders.

##### Aim

The aim of *Foodie Spot* is to provide a seamless, user-friendly experience for customers and a powerful management system for administrators. It seeks to simplify the process of browsing, ordering, and managing food while ensuring real-time inventory tracking and accurate sales analysis. The platform aspires to create an environment where users enjoy personalized, hassle-free food services and administrators gain valuable insights to make informed decisions.

##### Objectives for Users

For users, *Foodie Spot* focuses on delivering a smooth and efficient food ordering experience. Key objectives include providing intuitive navigation for browsing menus, real-time updates on product availability, and secure transactions. Additionally, the platform aims to enhance user satisfaction by offering personalized order history and a responsive interface that works across devices, ensuring accessibility anytime, anywhere.

##### Objectives for Administrators

On the administrative side, the application aims to empower managers with advanced tools for monitoring and decision-making. These objectives include providing detailed sales and profit insights via dynamic dashboards, enabling real-time inventory management, and supporting offline reporting through PDF export capabilities. By integrating analytics tools like Recharts, *Foodie Spot* helps administrators visualize trends and track performance metrics, enabling them to improve operational efficiency and profitability.

### BACKGROUND OF PROJECT:

The rapid advancement of technology has transformed how food services operate, with digital solutions becoming a cornerstone for improving efficiency and customer satisfaction. Traditional canteen and cafeteria management often rely on manual processes, which can lead to challenges like mismanaged inventory, delayed orders, and a lack of actionable insights into sales performance. Recognizing these gaps, *Foodie Spot* was conceptualized as a comprehensive web-based platform designed to modernize food ordering and streamline administrative tasks. By leveraging technologies like React.js, Firebase, and Recharts, the project integrates user convenience with robust data-driven decision-making tools.

*Foodie Spot* stands out by addressing both user and administrative needs. For users, it provides a seamless and personalized food ordering experience, while for administrators, it offers a powerful management dashboard with real-time insights into sales, profits, and inventory. With additional features like PDF export for offline reporting and responsive design for accessibility across devices, the platform is built to cater to the dynamic demands of modern cafeterias. This project aims to redefine canteen management by combining intuitive interfaces, real-time updates, and advanced analytics to enhance operations and user satisfaction alike.

### SCOPE OF PROJECT:

##### User-Focused Features

The scope of *Foodie Spot* includes offering an intuitive and streamlined food ordering experience for users. Customers can browse through a variety of food items with detailed descriptions and prices, add them to their shopping cart, and seamlessly complete their purchases. The system incorporates real-time inventory updates, ensuring users are informed about item availability before placing orders. Additionally, the platform provides personalized order history, allowing users to easily reorder their favourite meals or review previous transactions.

##### Responsive Design for Accessibility

To ensure maximum accessibility, *Foodie Spot* is designed with a fully responsive layout that adapts seamlessly to different devices and screen sizes. Whether users are accessing the platform via desktops, tablets, or smartphones, they can enjoy a smooth and consistent experience. This cross-platform compatibility broadens the platform's usability, making it suitable for diverse customer demographics.

##### Administrative Tools for Management

For administrators, the project encompasses a comprehensive management system that includes features for order tracking, inventory control, and sales monitoring. The admin dashboard offers detailed insights into financial performance, such as sales and profit trends over weekly, monthly, and yearly periods. This real-time data empowers administrators to make informed decisions to optimize operations and boost profitability. The ability to manage inventory efficiently ensures that stock levels are maintained, minimizing waste and avoiding shortages.

##### Advanced Data Visualization

The project scope also includes powerful analytics capabilities using Recharts to visualize trends and patterns in sales and profits. Administrators can monitor performance metrics through graphical representations, making it easier to identify growth opportunities and areas needing improvement. These visual insights enable decision-makers to track overall performance at a glance and refine their strategies for better outcomes.

##### Offline Reporting and Record-Keeping

To support efficient record-keeping and offline accessibility, *Foodie Spot* includes functionality for exporting data in PDF format using JSPDF. Administrators can generate detailed reports on order histories, sales, and other key metrics, ensuring that important information is stored and accessible even without internet connectivity. This feature simplifies audit processes and enhances operational transparency.

##### Scalability and Future Growth

The platform is designed with scalability in mind, ensuring that it can adapt to growing demand and additional features. Whether it’s integrating new analytics tools, expanding to support multiple locations, or incorporating AI-driven recommendations, *Foodie Spot* has the potential to evolve with changing needs. This forward-thinking approach positions the project as a long-term solution for modernizing canteen management, catering to both current and future requirements.

### MODULES DESCRIPTION:

##### User Registration and Authentication

The control flow begins with users registering on the platform or logging into their accounts. Secure authentication via Firebase ensures that only authorized users can access the platform. For first-time users, account details are created, while returning users are directed to their personalized dashboard.

##### Menu and Food Item Browsing

Once logged in, users can browse the menu, which is dynamically updated based on real-time inventory. Data science techniques can be applied here to analyse user preferences and offer personalized recommendations for popular or frequently ordered items. This step ensures a tailored and engaging experience for each user.

##### Shopping Cart and Order Placement

Users can add food items to their shopping cart, view the total price, and make modifications before placing an order. At this stage, the system checks real-time inventory availability to confirm item stock. Predictive analytics can be incorporated to suggest add-ons or combos based on purchase patterns.

##### Order Confirmation and Processing

After an order is placed, the system confirms the order details and updates the inventory accordingly. Administrators are notified in real time via the admin panel to begin processing the order. Data analytics may also assist in optimizing the order queue based on peak hours or preparation time.

##### Inventory and Stock Management

The inventory management module ensures that stock levels are updated as orders are placed. Predictive analytics can be used to forecast demand and suggest restocking strategies, reducing the risk of running out of popular items. Alerts are sent to administrators for low-stock items to streamline procurement.

##### Sales Analytics and Reporting

As orders are completed, the sales data is fed into the analytics dashboard. Using tools like Recharts, administrators can visualize sales performance, profit margins, and trends over time (weekly, monthly, yearly). Data science algorithms can identify patterns, such as peak sales times, most popular items, and customer demographics, to aid decision-making.

##### Reporting and Offline Record-Keeping

The final step involves generating comprehensive reports for administrators. Using JSPDF, order histories, inventory updates, and sales trends can be exported as PDFs for offline access. This step ensures efficient record-keeping and enhances strategic planning by providing actionable insights into the canteen’s operations.

# 2 LITERATURE SURVEY

#### Literature Survey for Foodie Spot

* 1. **Concept of Food Ordering Systems and Their Benefits**

Food ordering systems are digital platforms that simplify and streamline the process of ordering food, either for takeout, delivery, or in-house consumption. These systems utilize technology to automate various aspects of food service management, such as menu browsing, order placement, and payment processing, ensuring convenience for users and operational efficiency for providers. Modern food ordering solutions cater to a range of business models, from large restaurant chains to small-scale canteens and cafeterias. They aim to enhance customer satisfaction while optimizing administrative tasks like inventory and sales tracking.

##### Two primary types of food ordering systems include:

1. **Online Food Ordering Platforms:** These allow users to browse menus, place orders, and pay online, with features like personalized recommendations and real-time updates on order status.
2. **In-House Ordering Systems:** Typically used in restaurants, canteens, or cafeterias, these systems manage on-premise food orders, inventory, and customer billing, often without delivery services.

#### Major Types of Food Ordering and Management Systems

There are three main types of food ordering and management systems:

* + - * **Point-of-Sale (POS) Systems:** Traditional systems focused on in-house billing, inventory tracking, and sales reporting.
      * **Integrated Food Management Systems:** Combine POS capabilities with inventory, analytics, and user management features, offering a comprehensive solution for food businesses.
      * **Cloud-Based Platforms:** Use the internet for data storage and processing, enabling real-time updates, remote access, and scalability for both users and administrators.

#### Benefits of Food Ordering and Management Systems

Modern food ordering platforms provide several benefits:

* + - * **Convenience for Users:** Enable seamless browsing, ordering, and payment processes for customers.
      * **Efficient Inventory Management:** Real-time updates prevent stock-outs or overstocking, reducing wastage and ensuring availability.
      * **Enhanced User Experience:** Personalized order histories and recommendations improve customer satisfaction.
      * **Data-Driven Insights for Administrators:** Analytics dashboards offer valuable insights into sales trends, customer preferences, and operational performance.
      * **Operational Efficiency:** Automates routine tasks, reducing administrative burden and improving order accuracy.
      * **Scalability and Accessibility:** Cloud-based solutions ensure platforms are accessible across devices and can grow with increasing demand.

#### Challenges in Food Ordering Systems

While food ordering systems bring numerous benefits, they also face challenges:

* + - * **High Implementation Costs:** Initial development and integration of such platforms can be costly, particularly for smaller canteens or cafeterias.
      * **Inventory Accuracy:** Ensuring real-time and precise inventory tracking remains a challenge, especially for perishable goods.
      * **Peak Demand Management:** Handling order surges during peak hours or events can strain system resources and cause delays.
      * **Data Privacy and Security:** Protecting user data, especially during payments, requires robust security measures.
    1. Advancements Driving Food Ordering Systems

Technological advancements have significantly influenced food ordering platforms:

* + - * **Machine Learning for Personalization:** Algorithms analyze user data to offer tailored recommendations and predict future preferences.
      * **Predictive Analytics for Inventory Management:** Forecasts stock requirements based on historical trends and seasonal patterns.
      * **Cloud Computing for Scalability: Enables** remote access, real-time updates, and cost-effective scaling of resources.
      * **Data Visualization Tools:** Help administrators track sales, profits, and performance metrics through dynamic dashboards.
      * **Integration of Payment Gateways:** Ensures secure and flexible payment options for users, enhancing trust and usability.

#### Different Data Science and ML Algorithms Used in Related Works for Foodies Spot:

##### Collaborative Filtering (Recommendation Systems):

This algorithm is widely used in food ordering systems to recommend items based on users' past behaviour and preferences. Collaborative filtering uses user-item interaction data to suggest food items that are likely to be of interest to users, improving customer engagement.

##### Classification Algorithms (Behaviour Prediction):

Algorithms like Logistic Regression, Decision Trees, and Random Forests are used to predict customer behaviours such as whether a user will place an order or abandon their cart. These models help identify at-risk customers, enabling targeted marketing strategies.

##### Time Series Forecasting (Demand Prediction):

Algorithms like ARIMA, LSTM, and Prophet are employed to predict future sales and food demand patterns based on historical data. These predictions assist in managing inventory, preparing for peak periods, and optimizing order fulfilment.

##### Clustering Algorithms (Customer Segmentation):

K-Means and DBSCAN are clustering algorithms used to segment customers based on similar characteristics, such as order frequency or spending behaviour. This segmentation allows businesses to offer personalized promotions and improve customer targeting.

##### Regression Algorithms (Price Optimization and Cost Prediction):

Linear Regression, Decision Tree Regression, and Gradient Boosting Machines are used to predict continuous values like pricing strategies, user spending, or operational costs. These algorithms help optimize pricing and forecast costs based on various influencing factors.

#### Use of Explainable AI (XAI) in Food Ordering Systems

1. **Trust in Recommendations:** XAI methods like LIME or SHAP explain why specific food items are recommended, helping users understand the factors behind these suggestions (e.g., past orders or popular choices).
2. **Demand Prediction Transparency:** XAI enhances demand forecasting models (like ARIMA or LSTM) by clarifying which factors (e.g., time, seasonality) influence predictions, aiding better inventory management.
3. **Price Optimization Clarity:** XAI techniques explain how price recommendations are made based on variables like historical sales or customer behaviour, ensuring transparent pricing strategies.
4. **Fairness in Customer Segmentation:** By explaining clustering outcomes, XAI helps detect and prevent biases in customer segmentation, ensuring fairness in promotions and marketing strategies.
5. **Operational Decision Transparency:** XAI can clarify why certain operational decisions (like order prioritization or churn prediction) are made, increasing trust in AI-driven improvements.

## SYSTEM ANALYSIS

In this chapter, we will discuss and analyze the developing process of Audit Control including software requirement specification (SRS) and comparison between existing and proposed systems. The functional and non-functional requirements are included in the SRS part to provide a complete description and overview of system requirements before the developing process is carried out. Besides that, existing vs. proposed provides a view of how the proposed system will be more efficient than the existing one.

### HARDWARE AND SOFTWARE REQUIREMENTS

* + 1. **HARDWARE REQUIREMENTS:**
* Processor: Intel Core i5 or higher / AMD Ryzen 5 or higher (for better performance, especially when handling large datasets and model training).
* RAM: Minimum 8 GB (16 GB or higher recommended for faster processing of large datasets and complex model training).
* Storage: At least 256 GB SSD (Solid State Drive) or HDD, with SSD recommended for faster data access and storage. additional space for datasets, model artifacts, and logs (~10- 20 GB).
* GPU (optional but recommended): A dedicated GPU such as NVIDIA GTX 1050 Ti or higher (for faster model training, especially if dealing with large datasets or deep learning models).
* Internet Connection: Required for downloading libraries, dependencies, and datasets if necessary.

#### Software Requirements

##### Operating System:

* + Windows 10/11, macOS, or Linux-based OS

##### Frontend Development:

* + React.js for building the user interface
  + HTML5, CSS3, JavaScript for designing and styling the application

##### Backend Development:

* + Node.js for server-side scripting
  + Firebase for real-time database and user authentication

##### Database:

* + Firebase Realtime Database or Fire store for managing data

##### Data Science Tools:

* + Python for data analysis (optional for advanced analytics)
  + Jupyter Notebook for creating and testing ML models

##### Web Browser:

* + Google Chrome, Mozilla Firefox, or Microsoft Edge for accessing the web application

##### Other Software Tools:

* + JSPDF for PDF export of reports
  + Recharts for data visualization on the dashboard
  + Git for version control and collaboration

### SOFTWARE REQUIREMENT SPECIFICATION:

#### Software Requirement Specification:

Software Requirement Specification (SRS) is the starting point of the software developing activity. As system grew more complex it became evident that the goal of the entire system cannot be easily comprehended. Hence the need for the requirement phase arose. The software project is initiated by the client needs. The SRS is the means of translating the ideas of the minds of clients (the input) into a formal document (the output of the requirement phase.) The SRS phase consists of two basic activities:

##### Problem/Requirement Analysis:

The process is order and more nebulous of the two, deals with understand the problem, the goal and constraints.

##### Requirement Specification:

Here, the focus is on specifying what has been found giving analysis such as representation, specification languages and tools, and checking the specifications are addressed during this activity.

The Requirement phase terminates with the production of the validate SRS document.

Producing the SRS document is the basic goal of this phase.

## ROLE OF SRS:

##### Introduction

The Software Requirement Specification (SRS) for *Foodie Spot* outlines the functional and non- functional requirements for the system. It defines the purpose, scope, and goals of the project, along with detailed descriptions of the software functionalities, system interfaces, user interfaces, and performance requirements. The document serves as a guideline for the development and ensures alignment with user needs and expectations.

##### Functional Requirements

* 1. **User Registration and Authentication:**
     + The system must allow users to register with an email address or social media accounts.
     + Users must be able to log in securely using their credentials.
     + Forgot password functionality must be provided.

##### Food Menu Browsing:

* + - Users should be able to browse food items available in the canteen, sorted by categories (e.g., Breakfast, Lunch, Snacks).
    - Filters should be available to sort items by price, popularity, and ratings.

##### Shopping Cart:

* + - Users can add multiple items to their shopping cart.
    - The cart should display the list of selected items, quantities, and total price.
    - Users must be able to update quantities or remove items from the cart.

##### Order Placement and Payment:n v

* + - Users must be able to proceed to checkout after reviewing the cart.
    - Secure payment options (credit card, debit card, wallet integration) should be provided.
    - The system must confirm successful payment and provide an order confirmation with an estimated delivery time.

##### Order History:

* + - Users should be able to view their past orders, with detailed information (e.g., order date, items, total cost).
    - Users must have the option to reorder previous items.

##### Admin Panel:

* + - Admins should have access to a dashboard for monitoring sales, profits, and inventory levels.
    - The system must allow admins to add, update, or remove food items from the menu.
    - Admins should be able to view user activity and generate reports on sales and user behaviour.

##### Real-Time Inventory Management:

* + - Inventory should be automatically updated after each order is placed.
    - Admins must receive alerts when inventory levels fall below a predefined threshold.

##### Reporting and Data Export:

* + - Admins must be able to export reports in PDF format (sales, order history, etc.).
    - Data visualizations (using Recharts) should be available for sales trends, weekly, monthly, and yearly reports.

##### Non-Functional Requirements

* 1. **Performance:**
     + The system should handle at least 500 concurrent users without performance degradation.
     + Response time for loading pages should be under 3 seconds.

##### Scalability:

* + - The system should support scalability to handle increased traffic during peak hours (e.g., lunch and dinner times).

##### Security:

* + - The application must use HTTPS for secure communication.
    - User authentication should be implemented using Firebase Authentication, and passwords must be securely stored and hashed.

##### Usability:

* + - The user interface must be responsive and intuitive, supporting devices with screen sizes ranging from mobile phones to desktop computers.
    - Navigation should be simple and easy to understand, with clear labels and buttons.

##### Availability:

* + - The system must have 99.9% uptime, with periodic maintenance windows announced in advance.

##### Backup and Recovery:

* + - The system should have automatic backup features, with daily backups of critical data, such as user orders and inventory information.
    - A disaster recovery plan should be in place for system outages.

##### System Interfaces

* 1. **Database Interface:**
     + The system will interact with Firebase Realtime Database to store user, order, and inventory data.
     + Data read and write operations must be optimized for performance.

##### Payment Gateway Interface:

* + - The system will integrate with third-party payment gateways (e.g., Stripe, PayPal) for processing transactions.

##### Admin Dashboard Interface:

* + - Admin users will access the system via a secure dashboard to manage orders, inventory, and generate reports.

##### User Interfaces

* 1. **User Interface:**
     + A clean and intuitive design for browsing food items, viewing order details, and making payments.
     + An easy-to-navigate shopping cart and checkout process.

##### Admin Interface:

* + - A data-driven dashboard for monitoring sales, inventory, and generating reports.
    - Interfaces for adding, updating, or removing menu items.

##### Assumptions and Constraints

* **Assumptions:**
  + Users will have an active internet connection to access the platform.
  + The platform will be accessible on modern browsers such as Chrome, Firefox, and Edge.

##### Constraints:

* + The system must be developed within the allocated time frame of 6 months.
  + Budget constraints may limit the number of features that can be implemented in the initial release.

##### Conclusion

This SRS document provides a comprehensive overview of the software requirements for the *Foodie Spot* application, outlining key functionalities, performance specifications, and system interfaces. This will guide the development process and ensure the system meets user needs while adhering to industry standards for security, scalability, and usability.

### SCOPE:

The scope of the *Foodie Spot* project encompasses the development of a web-based platform designed to streamline food ordering and management for cafeterias and canteens. It includes functionalities for users to browse menus, place orders, manage their shopping cart, and make secure payments. The system also provides administrators with tools for real-time inventory tracking, order management, and generating reports to monitor sales and profits. Additionally, the platform integrates data science features for demand prediction and personalized recommendations. The system is built with a responsive design to ensure seamless access across devices and uses Firebase for user authentication and database management. The scope also includes scalability to handle peak periods and robust security features to protect user data.

### EXISTING SYSTEM:

The existing systems for food ordering and management in cafeterias or canteens typically involve a combination of manual processes and basic digital tools, which often result in inefficiencies and errors. For example, many cafeterias still rely on paper-based systems or spreadsheets (e.g., Microsoft Excel) to track orders, inventory, and sales. These manual systems are time-consuming, prone to errors, and make it difficult to get real-time data on stock levels or sales performance. As a result, administrators may find themselves running out of stock on popular items or failing to notice sales trends that could inform better decision- making.

An example of a basic digital system might include a point-of-sale (POS) system like Square or Toast, which handles transactions but may not integrate seamlessly with inventory management or provide insights into customer behaviour. These systems might offer basic features like generating sales receipts and tracking payment methods, but they lack advanced functionalities like personalized recommendations, demand forecasting, or automated inventory updates. Additionally, many cafeterias using these systems have limited access to data analytics and reporting features, which makes it challenging for them to track and optimize sales trends.

In some cases, larger cafeterias or chains may use more advanced systems like Oracle Food & Beverage or SAP Business One, which offer some integration between sales, inventory, and reporting. However, these systems can be expensive, complex to implement, and may not be tailored to the specific needs of smaller or medium-sized canteens. Furthermore, these systems are often not as user-friendly for customers, as they may not offer a mobile-friendly interface or an intuitive online ordering experience.

Overall, the existing systems often lack the integration, scalability, and advanced data-driven features needed to enhance both the customer experience and operational efficiency, which *Foodie Spot* aims to

### DRAWBACKS OF EXISTING SYSTEM:

The existing systems for food ordering and management in cafeterias and canteens suffer from several limitations, which hinder operational efficiency and the overall user experience:

##### Manual Data Management:

Many cafeterias still rely on manual systems, such as paper-based records or spreadsheets, to manage orders, inventory, and sales. This leads to errors, inconsistencies, and time-consuming data entry tasks. There is no real-time visibility into inventory, resulting in stockouts or overstocking of items.

##### Lack of Real-Time Updates:

Traditional systems often do not support real-time updates, meaning that inventory and order details are not immediately reflected across all parts of the system. For example, when an item is sold, the system may not update the stock levels immediately, causing confusion and potential errors in fulfilling customer orders.

##### Limited Customer Interaction:

Existing systems may not provide an engaging or user-friendly experience for customers. Many cafeterias still rely on physical menus or outdated online ordering interfaces. This limits customer convenience, as customers often cannot place orders via mobile apps or websites, leading to longer wait times and lower satisfaction.

##### Lack of Data Analytics and Insights:

Current systems typically lack advanced data analytics tools to provide insights into sales trends, customer behaviour, or demand forecasting. Without detailed reporting capabilities, administrators are unable to track performance metrics or optimize pricing and inventory management, which impacts profitability and operational efficiency.

##### Inefficient Order Management:

In systems that rely on manual processing or basic POS tools, order management can become a bottleneck. Orders may be misplaced or delayed due to miscommunication between kitchen staff and front-line servers. Additionally, there are often no features for personalized order history or preferences, which could enhance the customer experience.

##### Scalability and Flexibility Issues:

Many existing systems are not easily scalable. As cafeteria operations grow or encounter increased demand (e.g., during busy lunch hours), the system may struggle to handle the load. The lack of flexibility makes it difficult to adapt the system for expanding menu options, new payment methods, or integrating additional features like predictive analytics.

##### High Costs and Complexity:

More advanced systems like Oracle Food & Beverage or SAP Business One are often expensive and complex to implement. Smaller canteens may find these systems financially unfeasible or too difficult to use without dedicated IT support. This limits the accessibility of such systems to larger businesses, leaving smaller establishments with less efficient, outdated solutions.

### PROPOSED SYSTEM:

The proposed system, *Foodie Spot*, is a modern, web-based application designed to streamline food ordering, inventory management, and sales tracking for cafeterias and canteens. By integrating cutting- edge technologies and data-driven features, the system aims to overcome the limitations of existing systems while providing a seamless experience for both users and administrators.

##### Real-Time Order and Inventory Management:

The proposed system features real-time inventory tracking, which automatically updates stock levels after each order. This ensures accurate and up-to-date information, preventing stockouts or overstocking of food items. Administrators are notified when inventory levels fall below a threshold, enabling proactive restocking.

##### User-Friendly Interface and Online Ordering:

*Foodie Spot* offers an intuitive, responsive interface that allows customers to browse the menu, view detailed food item descriptions, and place orders easily from any device, including smartphones, tablets, or desktops. The system supports secure online payments through integrated payment gateways, enhancing customer convenience and satisfaction.

##### Personalized Recommendations and Order History:

The system leverages data science techniques to provide personalized food recommendations based on user preferences, past orders, and popular items. This improves the customer experience by offering tailored suggestions that align with their tastes. Additionally, users can view their order history and quickly reorder items.

##### Advanced Analytics and Reporting:

*Foodie Spot* incorporates data analytics tools such as Recharts to provide administrators with insights into sales trends, customer preferences, and inventory usage. The system generates comprehensive reports on daily, weekly, and monthly sales performance, enabling administrators to make informed decisions. Admins can also export reports in PDF format for offline record- keeping.

##### Scalability and Flexibility:

The proposed system is designed to scale easily, supporting cafeterias of varying sizes. Whether it’s a small canteen or a large cafeteria with multiple locations, *Foodie Spot* can handle increased traffic and adapt to growing business needs. New features like menu item additions or integration with third-party services (e.g., delivery platforms) can be implemented seamlessly.

##### Enhanced Security and User Authentication:

Security is a top priority in the proposed system. Firebase Authentication ensures that user data is securely stored, and only authorized users have access to sensitive information. All data transactions are encrypted, and the platform uses HTTPS to secure communication between users and the server.

##### Admin Panel and Operational Efficiency:

The admin panel of *Foodie Spot* provides comprehensive control over the entire system, from managing the food menu and setting prices to monitoring orders and customer interactions. The system also automates routine tasks such as stock updates and order tracking, improving operational efficiency and reducing administrative workload.

### ADVANTAGES OF PROPOSED SYSTEM:

The *Foodie Spot* system offers several advantages over existing systems, providing both operational efficiency and enhanced user experience. Below are the key benefits of the proposed system:

##### Real-Time Inventory and Order Management:

The system ensures accurate inventory tracking by automatically updating stock levels in real time as orders are placed. This reduces the risk of human errors, prevents stockouts, and helps cafeteria managers maintain an optimal inventory level. Real-time updates ensure that the kitchen is always prepared with the right ingredients, leading to smoother operations.

##### User-Friendly and Convenient Ordering Experience:

*Foodie Spot* provides a highly intuitive and responsive user interface that makes it easy for customers to browse the menu, select items, and place orders online from any device. The platform's user-friendly design ensures that customers have a seamless experience, reducing wait times and enhancing overall satisfaction.

##### Personalization and Customer Engagement:

The system uses data science techniques to offer personalized recommendations based on a user’s past orders, preferences, and popular items. This makes the ordering process more engaging for customers and encourages repeat business. By offering personalized suggestions, the system helps boost sales by highlighting relevant items to customers.

##### Advanced Analytics and Reporting:

With integrated data analytics tools, the system provides administrators with actionable insights into sales performance, customer behaviour, and inventory trends. Administrators can easily track daily, weekly, and monthly sales, and make data-driven decisions to optimize menu offerings and pricing strategies. The ability to export reports in PDF format further simplifies offline record- keeping and decision-making.

##### Scalability and Flexibility:

*Foodie Spot* is designed to scale with growing business needs. Whether for a small canteen or a large cafeteria with multiple locations, the system can handle increased traffic and adapt to new requirements. The flexibility of the platform allows for easy addition of new menu items, integration with third-party delivery services, or customization of features to match the specific needs of the cafeteria.

##### Improved Operational Efficiency:

The system automates key processes such as order processing, stock management, and report generation, which significantly reduces administrative overhead and human error. By streamlining operations, cafeteria staff can focus more on customer service and improving food quality, leading to enhanced productivity and a smoother workflow.

##### Enhanced Security and Data Protection:

Security is a critical aspect of the proposed system. Firebase Authentication ensures secure user authentication, and all data transactions are encrypted using HTTPS protocols. This protects sensitive user information, including payment details, reducing the risk of data breaches and ensuring a safe experience for customers.

##### Cost-Effectiveness:

By automating various administrative tasks, such as inventory tracking and order management, the proposed system helps reduce operational costs and labour-intensive activities. Additionally, its scalability means that cafeterias can start with basic features and expand as needed, minimizing upfront costs while providing room for future growth.

##### Centralized Management:

*Foodie Spot* offers a centralized admin panel that allows administrators to manage all aspects of the cafeteria, from food items and pricing to inventory and sales. This centralized control streamlines operations, making it easier to monitor performance and make quick adjustments to meet demand.

## TECHNOLOGIES USED

#### Machine Learning

##### Machine Learning and Data Science in Foodie Spot

**Foodie Spot**, leveraging **data science** and **machine learning (ML)**, provides a smarter, data-driven food ordering experience for both customers and administrators. Data science techniques are applied to enhance user personalization, streamline inventory management, predict food demand, and optimize sales strategies.

#### How Foodie Spot Uses Data Science:

##### Demand Prediction and Inventory Management:

Using historical data, machine learning models are trained to predict future food demand based on factors like time of day, day of the week, seasonality, and special promotions. This helps *Foodie Spot* optimize inventory, minimizing food wastage while ensuring popular items are always in stock. Data science algorithms analyze past sales trends and external factors to improve the accuracy of demand forecasting.

##### Personalized Recommendations:

Data science techniques, such as collaborative filtering and content-based filtering, are used to suggest food items tailored to each user’s preferences. By analyzing past orders, browsing behaviour, and user profiles, *Foodie Spot* provides personalized food recommendations, increasing user satisfaction and driving sales. For example, if a user often orders vegetarian dishes, the system will suggest more vegetarian options.

##### Customer Segmentation and Targeted Marketing:

Data science algorithms cluster users into different segments based on behaviours such as purchase history, frequency, and spending habits. These segments enable the cafeteria to create targeted promotions, offers, and discounts, increasing engagement with customers based on their preferences and spending patterns.

##### wwdSales and Profitability Analysis:

Data science is used to track and analyze sales data in real time. By using tools like Recharts, *Foodie Spot* can visualize trends in weekly, monthly, or yearly sales. This helps administrators identify the most profitable items, assess the impact of promotions, and make data-driven decisions about pricing, inventory, and menu design.

##### Real-Time Analytics for Order Processing:

ML models analyze real-time data from the food ordering process to optimize operational efficiency. For instance, the system can predict peak ordering times and automatically adjust staffing or cooking times. By analyzing user behaviour and order volumes, *Foodie Spot* can ensure smoother workflows, reducing waiting times and improving customer experience.

##### Fraud Detection and Anomaly Detection:

Data science techniques are used to identify unusual activity, such as fraudulent transactions or suspicious order patterns. ML algorithms continuously learn from transactional data, enabling the system to flag potential fraud in real-time, safeguarding both users and administrators.

##### Operational Optimization:

By analyzing operational data (e.g., cooking times, order processing times, delivery times), data science models can help optimize workflows. For example, predictive models can suggest the optimal number of staff required during different times of day or recommend adjustments to the menu based on sales data, helping Foodie Spot run more efficiently.

#### Data Science Tools and Frameworks Used:

* + **Python**: For implementing machine learning models and data manipulation, Python is the go-to language. Libraries like **Pandas** and **NumPy** are used for data preprocessing, while **scikit-learn** and **TensorFlow** are used for building ML models.
  + **Firebase**: As the backend platform, Firebase collects user data in real time and integrates seamlessly with the data science models for personalized recommendations, inventory management, and demand forecasting.
  + **Recharts and Data Visualization**: Recharts are used to visualize data trends and sales analytics, enabling administrators to make informed decisions about food offerings and marketing strategies.

##### Benefits of Using Data Science in Foodie Spot:

1. **Enhanced User Experience**: Personalized recommendations and faster order processing create a seamless user experience.
2. **Optimized Inventory Management**: Predicting demand reduces food wastage and ensures popular items are always available.
3. **Increased Profitability**: Data-driven insights into sales and customer behavior help administrators optimize pricing, promotions, and menu items.
4. **Operational Efficiency**: Predicting peak demand times and optimizing staffing improves overall operational workflows.
5. **Real-Time Insights**: Data science allows for continuous monitoring and analysis, ensuring timely decisions and adjustments to the system.

In summary, **Foodie Spot** leverages **data science** and **machine learning** to optimize its operations, provide personalized experiences, and make data-driven decisions that benefit both customers and administrators. By integrating predictive models and advanced analytics, it offers a smarter, more efficient approach to cafeteria and canteen management.

#### React.js and Firebase for Foodie Spot with Data Science

**Overview:**

*Foodie Spot* leverages **React.js** and **Firebase** to create an efficient, scalable, and real-time web-based platform. React.js, a popular JavaScript library, and Firebase, a Backend-as-a-Service (BaaS) platform, enable the system to provide a smooth user experience, real-time updates, and seamless data synchronization. This combination enhances the development and performance of the system, making it well-suited for data-driven functionalities, such as personalized recommendations and demand prediction powered by machine learning.

#### React.js:

**React.js** is a widely-used, declarative, and component-based JavaScript library developed by Facebook. It allows for the building of responsive and dynamic user interfaces (UIs) by breaking down complex UI components into reusable components.

##### Features:

* **Component-Based Architecture**: React allows the application to be built using reusable components, enhancing code maintainability and modularity. Each component represents a piece of the UI, making it easy to develop and scale the application.
* **Virtual DOM**: React uses a Virtual DOM that optimizes rendering by only updating parts of the user interface that have changed. This results in faster and more efficient updates, which is particularly important in dynamic, real-time applications like *Foodie Spot*.
* **Declarative Syntax**: React’s declarative syntax simplifies UI design, where developers describe what the UI should look like, and React takes care of updating the DOM to match the state of the application.
* **Rich Ecosystem**: React has a vast ecosystem with libraries and tools that can further enhance the development process, such as React Router for routing and Redux for state management.

##### Suitability for Foodie Spot:

* React.js helps in creating a seamless and responsive UI for users to browse menus, manage carts, and place orders. It ensures a consistent experience across different devices by providing a mobile-friendly, dynamic interface.
* React’s real-time data rendering capabilities are critical for live order tracking and inventory updates, ensuring the system always reflects accurate, up-to-date information.

#### Firebase:

**Firebase** is a cloud-based platform for developing and managing web and mobile applications. It offers a wide range of features that support real-time data synchronization, user authentication, hosting, and cloud functions.

##### Features:

* **Real-Time Database**: Firebase provides a real-time NoSQL database, enabling seamless synchronization of data between users and the system. It’s ideal for managing live updates, such as inventory changes and order status updates in *Foodie Spot*.
* **Firebase Authentication**: Firebase Authentication simplifies user sign-up, login, and secure access control, supporting authentication through email/password, social logins (Google, Facebook), or phone numbers. This ensures a secure and personalized experience for users.
* **Cloud Functions**: Firebase Cloud Functions enable serverless back-end processing, making it easy to run backend code in response to events (e.g., a user placing an order). This allows *Foodie Spot* to scale its backend seamlessly without worrying about managing infrastructure.
* **Cloud Firestore**: Firebase’s Cloud Firestore is a flexible, scalable database for storing and syncing data. It allows *Foodie Spot* to store order details, user data, menu items, and analytics results in an easily accessible format.
* **Push Notifications**: Firebase Cloud Messaging enables sending push notifications to users, helping *Foodie Spot* notify customers about order updates, promotions, or special offers.

##### Suitability for Foodie Spot:

* Firebase allows *Foodie Spot* to provide real-time updates to users, such as showing live order status and stock levels. Firebase’s real-time database ensures smooth communication between the front-end (React.js) and back-end, keeping data synchronized across the platform.
* Firebase Authentication streamlines the process of securely logging in and managing user accounts, making it easier to offer personalized experiences like order history and recommendations.

#### Integration with Data Science:

The combination of React.js and Firebase is also highly compatible with machine learning and data science functionalities in *Foodie Spot*. For example, Python-based machine learning models or algorithms (e.g., for demand prediction or recommendation systems) can be deployed on the backend via Firebase Cloud Functions or integrated through APIs, and their results can be displayed in real-time on the React.js frontend.

**Benefits for Data Science in *Foodie Spot*:**

* **Real-Time Personalization**: Using data science models such as collaborative filtering or classification algorithms, *Foodie Spot* can provide personalized food recommendations to users, which are powered by real-time data stored in Firebase.
* **Demand Prediction**: By analyzing historical data, machine learning models can predict demand for certain food items, allowing the system to adjust inventory in real time. React and Firebase together can quickly display the predicted trends and update the UI accordingly.
* **Advanced Analytics**: Data collected through the system (e.g., sales data, user behavior) can be analyzed using data science techniques to gain insights into customer preferences, popular items, and sales trends, which can then be displayed through React components and stored in Firebase for historical tracking.

#### Community and Support:

* **React.js Community**: React has a large, active community that provides extensive resources, tutorials, and tools. This ensures that developers working on *Foodie Spot* have access to ongoing support and innovation.
* **Firebase Support**: Firebase is backed by Google, and its large community provides plenty of documentation, guides, and troubleshooting forums. Firebase's scalability and comprehensive features ensure long-term support for the system as it grows.

#### Conclusion:

The combination of **React.js** for front-end development and **Firebase** for backend and real-time data management makes *Foodie Spot* an efficient, scalable, and user-friendly platform. These technologies, along with their integration with data science techniques, help the system provide personalized experiences, efficient order management, and real-time updates, all while being easy to maintain and expand. The rich ecosystem of tools, libraries, and support from both React.js and Firebase further enhances the development and operational capabilities of *Foodie Spot*.

#### 4.2 Languages and Frameworks:

To develop and run **Foodie Spot**, a comprehensive web-based application for food ordering and management, a variety of programming languages and frameworks are employed. These technologies ensure the app is fast, reliable, and scalable while providing a user-friendly interface and seamless backend operations. Below are the key languages and frameworks used in the development of **Foodie Spot**:

#### Frontend Development:

##### React.js:

* + **Overview**: React.js is a popular JavaScript library used for building user interfaces, particularly single-page applications (SPAs) where user interactions occur without page reloads. It is component-based and allows developers to build reusable UI components, making the application more maintainable and scalable.
  + **Why it’s Used**: React.js provides a smooth and dynamic user experience for **Foodie Spot**. It ensures fast rendering of pages, even with real-time updates such as order tracking and inventory changes. The declarative syntax makes it easy to manage the user interface and its interactions efficiently.
  + **Benefits**:
    - Component-based architecture simplifies code management.
    - Fast rendering using virtual DOM (Document Object Model).
    - A large community and rich ecosystem of third-party tools and libraries.

##### HTML/CSS:

* + **Overview**: HTML (HyperText Markup Language) is used for structuring the content of web pages, while CSS (Cascading Style Sheets) is used to style the visual elements.
  + **Why it’s Used**: HTML and CSS form the backbone of the web application’s layout and styling, ensuring that **Foodie Spot** is visually appealing and user-friendly. CSS frameworks like Bootstrap can also be employed to make the application responsive across devices.
  + **Benefits**:
    - Helps in creating responsive and adaptive web designs.
    - Enhances the user interface with various styles, animations, and effects.

***Backend Development:***

##### Node.js:

* + **Overview**: Node.js is a JavaScript runtime built on Chrome’s V8 engine that allows developers to build scalable and fast network applications. It is ideal for handling real-time operations like order processing and user authentication.
  + **Why it’s Used**: For **Foodie Spot**, Node.js is used for server-side logic, handling user requests, API calls, and real-time updates. It supports asynchronous, event-driven programming, which is essential for fast and efficient data exchange between the client and the server.

##### Benefits:

* + - Asynchronous and non-blocking architecture allows handling multiple requests simultaneously.
    - Single language (JavaScript) for both frontend and backend development.
    - Vast ecosystem of packages through npm (Node Package Manager).

##### Express.js:

* + **Overview**: Express.js is a web application framework for Node.js, designed to simplify building robust and scalable web applications and APIs.
  + **Why it’s Used**: Express.js is employed to build and manage the backend API of **Foodie Spot**, which handles tasks such as user authentication, food ordering, order history, and reporting.

##### Benefits:

* + - Simplifies the creation of RESTful APIs.
    - Middleware support allows easy integration with authentication, logging, and error handling mechanisms.
    - Lightweight and flexible, ideal for scalable applications.

#### Database and Storage:

##### Firebase:

* + **Overview**: Firebase is a platform for building mobile and web applications, offering a variety of tools including real-time databases, authentication, and cloud functions.
  + **Why it’s Used**: Firebase provides real-time data synchronization, which is essential for **Foodie Spot** to track inventory and orders in real-time. Firebase Authentication is used for secure login and user management. Firebase Firestore or Realtime Database is used to store user data, orders, food menu information, and sales analytics.

##### Benefits:

* + - Real-time data synchronization ensures that all users and admins see updates immediately.
    - Secure authentication system via Firebase Authentication.
    - Easy integration with other Google Cloud services and third-party APIs.

##### Cloud Functions:

* + **Overview**: Firebase Cloud Functions allow the execution of backend code in response to events triggered by Firebase features or HTTP requests.
  + **Why it’s Used**: Cloud Functions help execute server-side logic for specific actions such as sending notifications when an order is placed, performing calculations for sales and profits, or handling scheduled tasks for inventory management.

##### Benefits:

* + - Scalable and event-driven, meaning you only pay for the actual usage.
    - Easily integrates with other Firebase services.
    - No need to manage servers, simplifying deployment.

#### Data Science and Machine Learning:

##### Python:

* + **Overview**: Python is a versatile programming language widely used in data science and machine learning. Its libraries like **Pandas**, **NumPy**, **scikit-learn**, and **TensorFlow** make it an ideal choice for implementing machine learning models for food demand prediction, personalization, and other data-driven tasks in **Foodie Spot**.
  + **Why it’s Used**: Python is used to develop predictive models for inventory forecasting, sales predictions, and personalized recommendations. It is also employed for data preprocessing, cleaning, and analysis.
  + **Benefits**:
    - Rich ecosystem of data science libraries.
    - Simple syntax and high readability, making it ideal for complex data tasks.
    - Strong community support in the machine learning and data science domains.

##### Jupyter Notebooks:

* + **Overview**: Jupyter Notebooks is an open-source web application that allows data scientists to create and share documents that contain live code, equations, visualizations, and narrative text.
  + **Why it’s Used**: Jupyter is used for developing and testing the machine learning models that power the recommendation engine, predictive analytics, and customer segmentation features of **Foodie Spot**.
  + **Benefits**:
    - Interactive environment for data analysis and exploration.
    - Supports both Python and R, making it versatile for machine learning and data science workflows.

#### Additional Frameworks and Tools:

##### Recharts:

* + **Overview**: Recharts is a charting library built for React, used to display interactive data visualizations such as sales reports and performance analytics.
  + **Why it’s Used**: In **Foodie Spot**, Recharts is used to display sales trends, inventory status, and financial reports on the admin dashboard, helping administrators make data-driven decisions.

##### Benefits:

* + - Responsive, easy-to-use charting library that integrates seamlessly with React.
    - Provides various chart types (line, bar, pie, etc.) for clear data visualization.

## SYSTEM DESIGN & UML DIAGRAMS

System design is transition from a user oriented document to programmers or data base personnel. The design is a solution, how to approach to the creation of a new system. This is composed of several steps. It provides the understanding and procedural details necessary for implementing the system recommended in the feasibility study. Designing goes through logical and physical stages of development, logical design reviews the present physical system, prepare input and output specification, details of implementation plan and prepare a logical design walkthrough.

### SOFTWARE DESIGN:

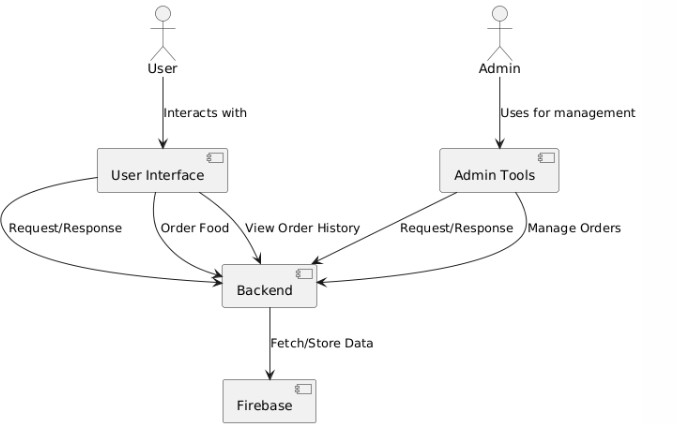
In designing the software following principles are followed:

* **Modularity and partitioning**: Software is designed such that, each system should consists of hierarchy of modules and serve to partition into separate function.
* **Coupling:** Modules should have little dependence on other modules of a system.
* **Cohesion:** Modules should carry out in a single processing function.
* **Shared use:** Avoid duplication by allowing a single module be called by other that need the function it provides.

### ARCHITECTURE:

Architecture diagram is a [diagram](http://en.wikipedia.org/wiki/Diagram) of a [system,](http://en.wikipedia.org/wiki/System) in which the principal parts or functions are represented by blocks connected by lines that show the relationships of the blocks. The block diagram is typically used for a higher level, less detailed description aimed more at understanding the overall concepts and less at understanding the details of implementation.

A SMS user for who the application looks like an user interface actually consists of a database called as SQLite that comes along with Android SDK and need no other installation. This is the database that is used to store and retrieve information. This is an application that is developed in java and hence all its features apply here as well such as platform independence, data hiding.



**FIGURE 5.2: SYSTEM FLOW DIAGRAM**

### UNIFIED MODELING LANGUAGE (UML) :

The unified modeling is a standard language for specifying, visualizing, constructing and documenting the system and its components is a graphical language which provides a vocabulary and set of semantics and rules. The UML focuses on the conceptual and physical representation of the system. It captures the decisions and understandings about systems that

must be constructed. It is used to understand, design, configure and control information about the systems.

The UML addresses the documentation of a system's architecture and all of its details. The UML also provides a language for expressing requirements and for tests. Finally, the UMLprovides a language for modeling the activities of project planning and release management.

### BUILDING BLOCKS OF UML:

The vocabulary of the UML encompasses three kinds of building blocks:

* Things.
* Relationships.
* Diagrams.

#### Things in the UML:

Things are the abstractions that are first-class citizens in a model; relationships tie these things together; diagrams group interesting collections of things.

There are four kinds of things in the UML:

* + - * + Structural things.
        + Behavioral things.
        + Grouping things.
        + Annotational things.

1. **Structural things** are the nouns of UML models. The structural things used in the project design are:
   * First, a **class** is a description of a set of objects that share the same attributes, operations, relationships and semantics.

|  |
| --- |
| Window |
| origin  size |
| open() close() move()  display() |

##### Fig: Classes

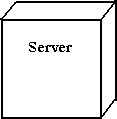
* + Second, a **use case** is a description of set of sequence of actions that a system

performs that yields an observable result of value to particular actor.



##### Fig: Use Cases

* + Third, a node is a physical element that exists at runtime and represents a computational resource, generally having at least some memory and often processing capability.



##### Fig: Nodes

1. **Behavioral things** are the dynamic parts of UML models. The behavioral thing used is:
   * Interaction: An interaction is a behavior that comprises a set of messages exchanged among a set of objects within a particular context to accomplish a specific purpose. An interaction involves a number of other elements, including messages, action sequences (the behavior invoked by a message, and links (the connection between objects).



**Fig: Messages**

#### Relationships in the UML:

There are four kinds of relationships in the UML:

* + - * + Dependency.
        + Association.
        + Generalization.
        + Realization.
        + A **dependency** is a semantic relationship between two things in which a change to one thing may affect the semantics of the other thing (the dependent thing).



##### Fig: Dependencies

* + - * + An **association** is a structural relationship that describes a set links, a link being a connection among objects. Aggregation is a special kind of association, representing a structural relationship between a whole and its parts.

##### Fig: Association

* + - * + A **generalization** is a specialization/ generalization relationship in which objects of the specialized element (the child) are substitutable for objects of the generalized element(the parent).



##### Fig: Generalization

* + - * + A **realization** is a semantic relationship between classifiers, where in one classifier specifies a contract that another classifier guarantees to carry out.

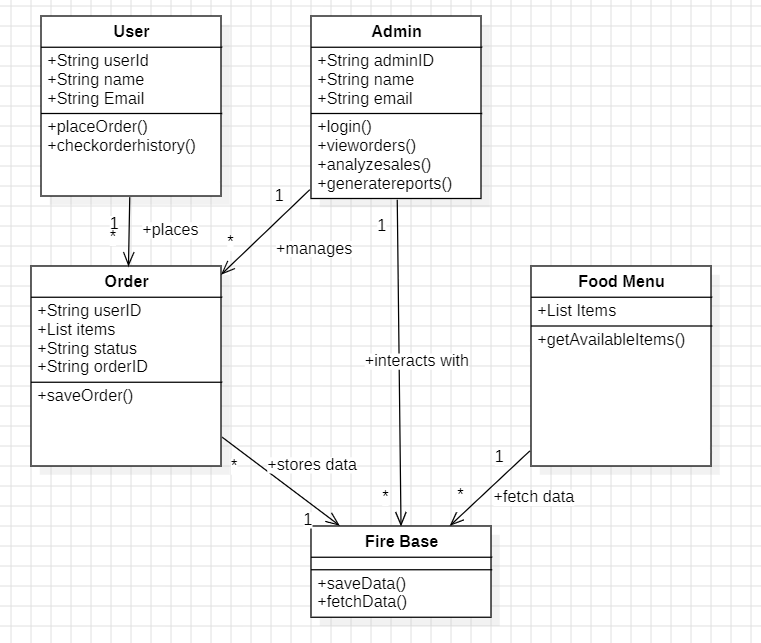


**Fig: Realization**

### UML DIAGRAMS:

**CLASS DIAGRAM:**

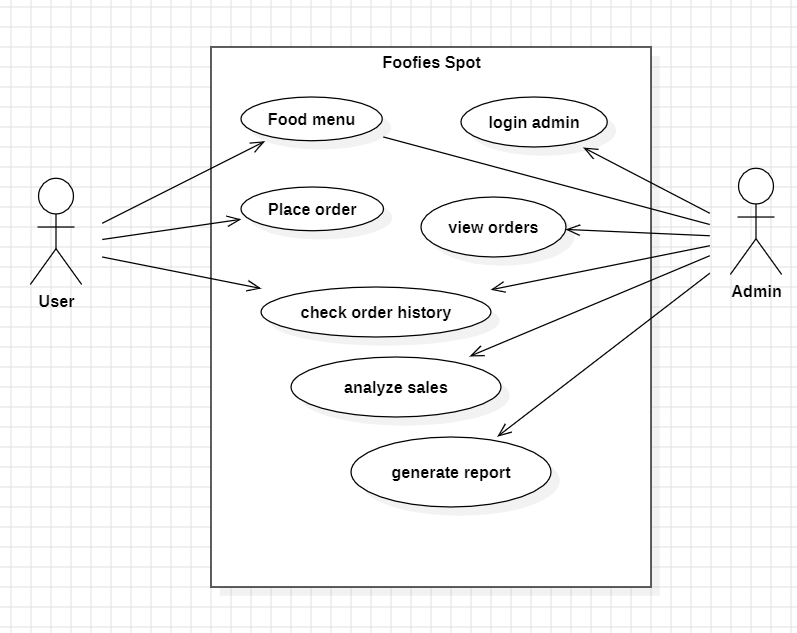
A class is a representation of an object and, in many ways; it is simply a template from which objects are created. Classes form the main building blocks of an object-oriented application. Although thousands of students attend the university, you would only model one class, called Student, which would represent the represent the entire collection of students.



**FIGURE 5.3.2.1: CLASS DIAGRAM**

### USE CASE DIAGRAM:

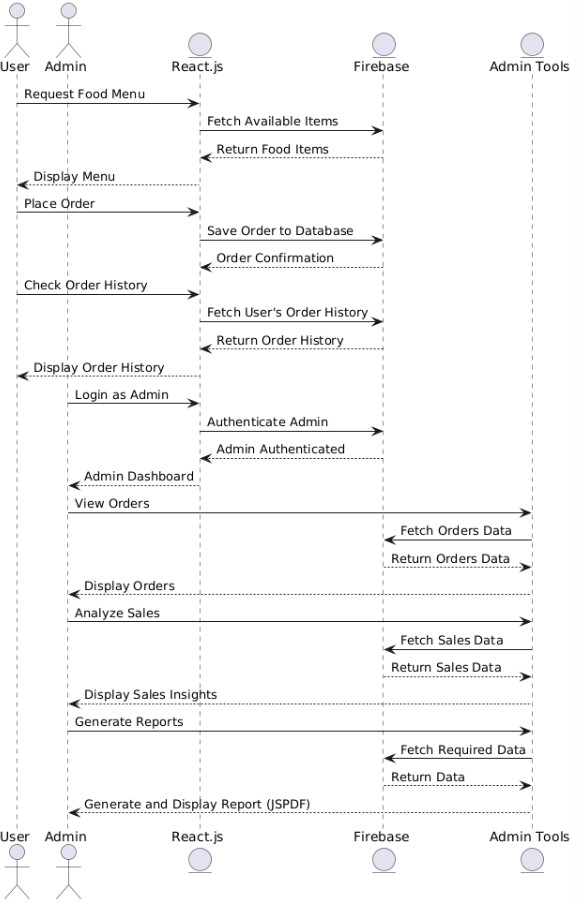
A use case diagram is a graph of actors set of use cases enclosed by a system boundary, communication associations between the actors and users and generalization among use cases. The use case model defines the outside (actors) and inside (use case) of the system’s behavior.



**FIGURE 5.3.2.2: USE CASE DIAGRAM**

### SEQUENCE DIAGRAM:

**Sequence diagram** are used to represent the flow of messages, events and actions between the objects or components of a system. Time is represented in the vertical direction showing the sequence of interactions of the header elements, which are displayed horizontally at the top of the diagram.

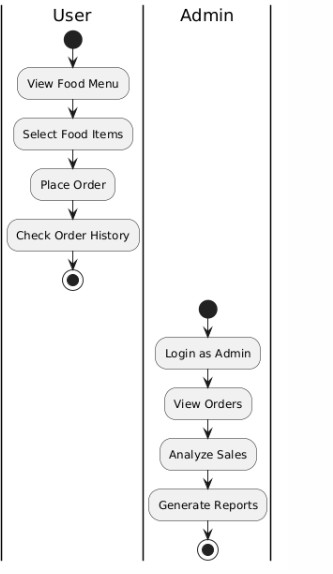


**FIGURE 5.3.2.3: SEQUENCE DIAGRAM**

### ACTIVITY DIAGRAM:

**Activity diagram** represent the business and operational workflows of a system. An Activity diagram is a dynamic diagram that shows the activity and the event that causes the object to be in the particular state.

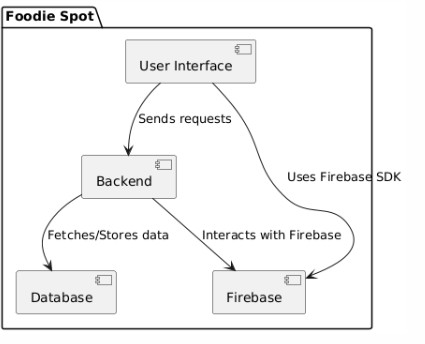
So, what is the importance of an Activity diagram, as opposed to a State diagram? A State diagram shows the different states an object is in during the lifecycle of its existence in the system, and the transitions in the states of the objects. These transitions depict the activities causing these transitions, shown by arrows.



**FIGURE 5.3.2.4: ACTIVITY DIAGRAM**

### COMPONENT DIAGRAM:

In the Unified Modeling Language, a **Component diagram** depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems.



**FIGURE 5.3.2.5: COMPONENT DIAGRAM**

## 6 IMPLEMENTATION

#### 6.1 App.js:

import React from 'react';

import { BrowserRouter as Router, Routes, Route, Navigate } from 'react-router-dom'; import "bootstrap/dist/css/bootstrap.min.css";

import "./App.css";

import { AuthProvider } from './AuthContext'; import ProtectedRoute from './ProtectedRoute'; import Home from './components/Home/Home'; import Login from './components/Auth/Login'; import SignUp from './components/Auth/SignUp';

const App = () => { return (

<Router basename="/Canteen-Management">

<AuthProvider>

<Routes>

<Route path="/" element={

<ProtectedRoute>

<Home />

</ProtectedRoute>

}

/>

<Route path="/signup" element={<SignUp />} />

<Route path="/login" element={<Login />} />

<Route path="\*" element={<Navigate to="/" />} />

</Routes>

</AuthProvider>

</Router>

);

};

export default App;

#### 6.2 Index.js:

import "bootstrap/dist/css/bootstrap.min.css"; import React from "react";

import { createRoot } from "react-dom/client"; import "./index.css";

import App from "./App";

import reportWebVitals from "./reportWebVitals";

const container = document.getElementById("root"); const root = createRoot(container);

root.render(

<React.StrictMode>

<App />

</React.StrictMode>

);

reportWebVitals();

#### 6.3 Apptest.js:

import { render, screen } from '@testing-library/react'; import App from './App';

test('renders learn react link', () => { render(<App />);

const linkElement = screen.getByText(/learn react/i); expect(linkElement).toBeInTheDocument();

});

#### 6.4 AuthContext.js:

import React, { createContext, useState, useContext, useEffect, useMemo,

} from "react";

import { getAuth, onAuthStateChanged } from "firebase/auth"; const AuthContext = createContext();

export function useAuth() {

return useContext(AuthContext);

}

export function AuthProvider({ children }) {

const [currentUser, setCurrentUser] = useState(null); const [isAdmin, setIsAdmin] = useState(false);

const [loading, setLoading] = useState(true);

useEffect(() => {

const auth = getAuth();

const unsubscribe = onAuthStateChanged(auth, (user) => { setCurrentUser(user);

setIsAdmin(false); // Reset admin state when Firebase auth changes setLoading(false);

});

return unsubscribe;

}, []);

// Admin login (hardcoded)

const loginAdmin = (username, password) => {

if (username === "admin" && password === "admin123") { setCurrentUser({ username: "admin" }); // Fake admin user object setIsAdmin(true);

return true;

}

return false;

};

const logout = () => { setCurrentUser(null); setIsAdmin(false);

};

const value = useMemo( () => ({

currentUser, isAdmin, loginAdmin, logout, loading,

}),

[currentUser, isAdmin, loading]

);

return (

<AuthContext.Provider value={value}>

{!loading && children}

</AuthContext.Provider>

);

}

#### 6.5 Footer.jsx:

import React from "react"; const Footer = () => {

const currentYear = new Date().getFullYear();

return (

<footer className="py-3 m-3 mt-5">

<div className="container text-center">

<p className="mb-0">

&copy; {currentYear} Foodies Spot. All rights reserved.

</p>

<p className="mb-0">

</p>

</div>

</footer>

);

};

export default Footer;

#### 6.6 Home.jsx:

import React, { useState, useEffect } from "react"; import { Link, useNavigate } from "react-router-dom"; import "bootstrap/dist/css/bootstrap.min.css";

import ProductList from "../ProductList/ProductList"; import CartModal from "../CartModal/CartModal";

import ProductDetailModal from "../ProductDetail/ProductDetailModal"; import Footer from "../Footer/Footer";

import AdminPanel from "../AdminPanel/AdminPanel"; import UserOrderHistory from "./UserOrderHistory"; import useProducts from "../../hooks/useProducts"; import { ref, update } from "firebase/database";

import { auth, database } from "../../firebase"; import { signOut } from "firebase/auth"; import {

Toast, Form, InputGroup, Container,

NavDropdown,

} from "react-bootstrap";

import { useAuth } from "../../AuthContext"; import jsPDF from "jspdf";

const Home = () => {

const navigate = useNavigate(); const { currentUser } = useAuth();

const { products, updateProductQuantity } = useProducts(); const [cart, setCart] = useState([]);

const [showCartModal, setShowCartModal] = useState(false); const [originalQuantities, setOriginalQuantities] = useState({});

const [showCheckoutToast, setShowCheckoutToast] = useState(false); const [searchTerm, setSearchTerm] = useState("");

const [selectedProduct, setSelectedProduct] = useState(null);

const [showProductDetailModal, setShowProductDetailModal] = useState(false); const [showAdmin, setShowAdmin] = useState(false);

const [logoutError, setLogoutError] = useState("");

const [showOrderHistory, setShowOrderHistory] = useState(false);

useEffect(() => {

if (products && products.length > 0) { setOriginalQuantities((prevQuantities) => {

const updatedQuantities = products.reduce((acc, product) => { acc[product.id] = product.quantity;

return acc;

}, {});

// Only update state if there's a difference, preventing re-renders

if (

JSON.stringify(updatedQuantities) !== JSON.stringify(prevQuantities)

) {

return updatedQuantities;

}

return prevQuantities;

});

}

}, [products]);

useEffect(() => {

if (!currentUser) { navigate("/login");

}

}, [currentUser, navigate]);

const generatePDF = () => { const doc = new jsPDF(); doc.text("Canteen", 10, 10);

doc.text("Items Purchased", 10, 20); cart.forEach((product, index) => {

const itemText = `${index + 1}.${product.name} - ${product.quantity} x ${ product.price

} = ${product.quantity \* product.price}`; doc.text(itemText, 10, 30 + index \* 10);

});

doc.text(`Total: ${calculateTotal()}`, 10, 40 + cart.length \* 10); doc.save("Canteen-Bill.pdf");

};

const addToCart = (product) => { setCart((prevCart) => {

const productInCart = prevCart.find((p) => p.id === product.id); if (productInCart) {

return prevCart.map((p) =>

p.id === product.id ? { ...p, quantity: p.quantity + 1 } : p

);

}

return [...prevCart, { ...product, quantity: 1 }];

});

setShowCartModal(true);

};

const incrementQuantity = (productId) => { const availableQuantity = originalQuantities[productId] -

(cart.find((p) => p.id === productId)?.quantity || 0); if (availableQuantity > 0) {

setCart((prevCart) => prevCart.map((p) =>

p.id === productId ? { ...p, quantity: p.quantity + 1 } : p

)

);

}

};

const decrementQuantity = (productId) => {

const productInCart = cart.find((p) => p.id === productId); if (productInCart && productInCart.quantity > 1) { setCart((prevCart) =>

prevCart.map((p) =>

p.id === productId ? { ...p, quantity: p.quantity - 1 } : p

)

);

}

};

const removeFromCart = (product) => {

setCart((prevCart) => prevCart.filter((p) => p.id !== product.id));

};

const clearCart = () => { setCart([]);

};

const calculateTotal = () => { return cart.reduce(

(total, product) => total + product.price \* product.quantity, 0

);

};

const checkout = () => { cart.forEach((product) => {

const originalQuantity = originalQuantities[product.id]; const updatedQuantity = originalQuantity - product.quantity;

const productRef = ref(database, `products/${product.id}`); update(productRef, { quantity: updatedQuantity })

.then(() => {

updateProductQuantity(product.id, updatedQuantity); setOriginalQuantities((prev) => ({

...prev,

[product.id]: updatedQuantity,

}));

})

.catch((error) =>

console.error("Error updating product quantity:", error)

);

});

// Record sales data

const checkoutDate = new Date().toISOString().split("T")[0]; const saleRef = ref(database, `sales/${checkoutDate}`);

const saleId = Date.now().toString(); const saleData = {

[saleId]: {

total: calculateTotal(), items: cart.map((item) => ({ id: item.id,

quantity: item.quantity, price: item.price,

})),

userId: currentUser?.uid, userEmail: currentUser?.email, timestamp: Date.now(),

},

};

update(saleRef, saleData);

setCart([]); setShowCartModal(false); setShowCheckoutToast(true);

setTimeout(() => setShowCheckoutToast(false), 3000);

generatePDF();

};

const filteredProducts = products

? products.filter((product) => product.name.toLowerCase().includes(searchTerm.toLowerCase())

)

: [];

const handleProductClick = (product) => { setSelectedProduct(product); setShowProductDetailModal(true);

};

const handleLogout = async () => { try {

setLogoutError(""); await signOut(auth); navigate("/login");

} catch (error) {

console.error("Error logging out:", error); setLogoutError("Failed to log out. Please try again.");

}

};

const handleAdminClick = () => { setShowAdmin((prev) => !prev);

};

const handleHomeClick = () => { setShowAdmin(false); setShowOrderHistory(false);

};

return (

<div className="d-flex flex-column min-vh-100">

<Container className="d-flex flex-column align-items-center my-3 py-3">

<h1 className="fw-bold">

<Link to="/"

className="text-decoration-none text-dark" onClick={() => {

setShowAdmin(false); setShowOrderHistory(false);

}}

>

Foodies Spot

</Link>

</h1>

<div className="d-flex justify-content-center">

<button onClick={handleHomeClick}

className="btn btn-Link text-decoration-none text-dark"

>

Home

</button>

<button onClick={() => {

setShowOrderHistory(true); setShowAdmin(false);

}}

className="btn btn-Link text-decoration-none text-dark"

>

My Orders

</button>

{currentUser && (

<button onClick={handleAdminClick}

className="btn btn-Link text-decoration-none text-dark"

>

Admin

</button>

)}

{currentUser && (

<NavDropdown

title={`Hello, ${currentUser.displayName || currentUser.email}`} id="user-nav-dropdown"

className="mt-2 fw-bold"

>

<NavDropdown.Item onClick={handleLogout}>Logout</NavDropdown.Item>

</NavDropdown>

)}

</div>

</Container>

<Container className="mt-5 flex-grow-1">

{logoutError && (

<Toast show={!!logoutError}

onClose={() => setLogoutError("")} className="position-fixed top-0 end-0 m-3" bg="danger"

text="white"

>

<Toast.Body>{logoutError}</Toast.Body>

</Toast>

)}

{showAdmin ? (

<AdminPanel />

) : showOrderHistory ? (

<UserOrderHistory />

) : (

<>

<div className="d-flex justify-content-between align-items-center mb-4 row">

<h1 className="fw-bold col">Today's items</h1>

<Form.Group className="w-100 w-md-50 my-auto col">

<InputGroup>

<InputGroup.Text>

<i className="fas fa-search"></i>

</InputGroup.Text>

<Form.Control type="text"

placeholder="Search products..." value={searchTerm}

onChange={(e) => setSearchTerm(e.target.value)}

/>

</InputGroup>

</Form.Group>

</div>

{cart.length > 0 && (

<button

className="btn btn-primary rounded-3" onClick={() => setShowCartModal(true)}

>

<i className="fas fa-shopping-cart me-2"></i> View Cart ({cart.length})

</button>

)}

<ProductList products={filteredProducts.map((p) => ({

...p,

quantity: originalQuantities[p.id] -

(cart.find((cp) => cp.id === p.id)?.quantity || 0),

}))}

addToCart={addToCart} onProductClick={handleProductClick}

/>

<CartModal show={showCartModal}

handleClose={() => setShowCartModal(false)} cart={cart} incrementQuantity={incrementQuantity} decrementQuantity={decrementQuantity} removeFromCart={removeFromCart} clearCart={clearCart} calculateTotal={calculateTotal} checkout={checkout}

/>

<ProductDetailModal show={showProductDetailModal}

handleClose={() => setShowProductDetailModal(false)} product={selectedProduct}

addToCart={addToCart}

/>

<Toast show={showCheckoutToast}

onClose={() => setShowCheckoutToast(false)} className="position-fixed bottom-0 end-0 m-3" delay={3000}

autohide

>

<Toast.Body>Checkout successful!</Toast.Body>

</Toast>

</>

)}

</Container>

<Footer />

</div>

);

};

export default Home;

#### 6.7 AdminLogin.js:

import React, { useState } from "react";

import { useNavigate } from "react-router-dom"; import { useAuth } from "../../AuthContext";

import { Card, Form, Button, Alert, Container } from "react-bootstrap";

const AdminLogin = () => {

const [username, setUsername] = useState(""); const [password, setPassword] = useState(""); const [error, setError] = useState("");

const { loginAdmin } = useAuth(); const navigate = useNavigate();

const handleLogin = (e) => { e.preventDefault(); setError("");

if (loginAdmin(username, password)) { navigate("/admin/dashboard"); // Redirect to Admin Dashboard

} else {

setError("Invalid Admin Credentials!");

}

};

return (

<Container className="d-flex align-items-center justify-content-center" style={{ minHeight: "100vh" }}>

<div className="w-100" style={{ maxWidth: "400px" }}>

<Card className="rounded-4">

<Card.Body>

<h2 className="mb-4 fw-bold p-2">Admin Login</h2>

{error && <Alert variant="danger">{error}</Alert>}

<Form onSubmit={handleLogin}>

<Form.Group className="mb-3">

<Form.Label>Username</Form.Label>

<Form.Control type="text" value={username}

onChange={(e) => setUsername(e.target.value)} required

/>

</Form.Group>

<Form.Group className="mb-3">

<Form.Label>Password</Form.Label>

<Form.Control type="password" value={password}

onChange={(e) => setPassword(e.target.value)} required

/>

</Form.Group>

<Button className="w-100 rounded-4" type="submit"> Log In

</Button>

</Form>

</Card.Body>

</Card>

</div>

</Container>

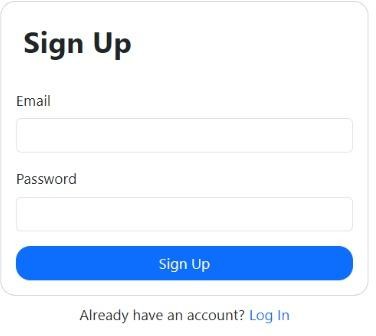
);

};

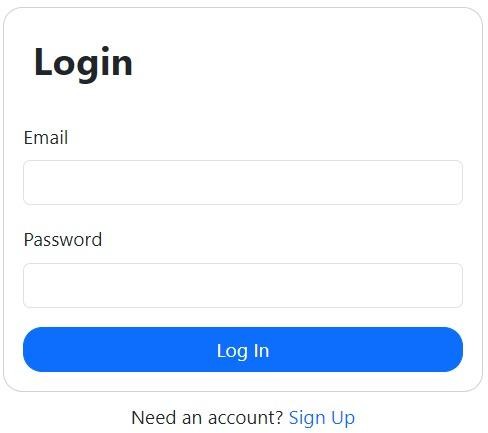
export default AdminLogin;

## 7 OUTPUT SCREENS

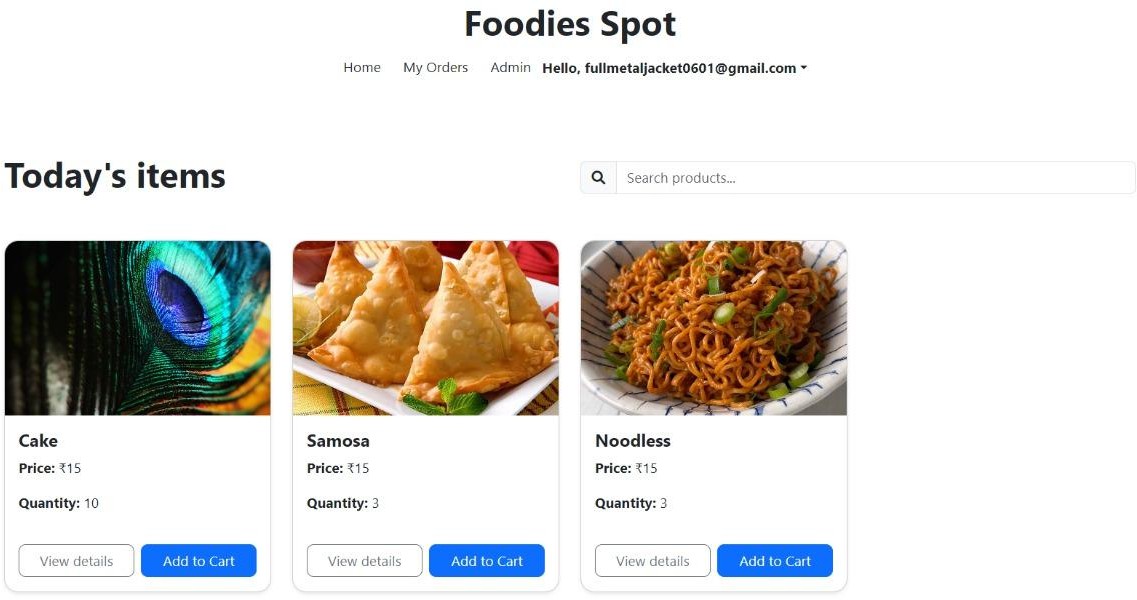
**7.1 Sign Up Page:**

****

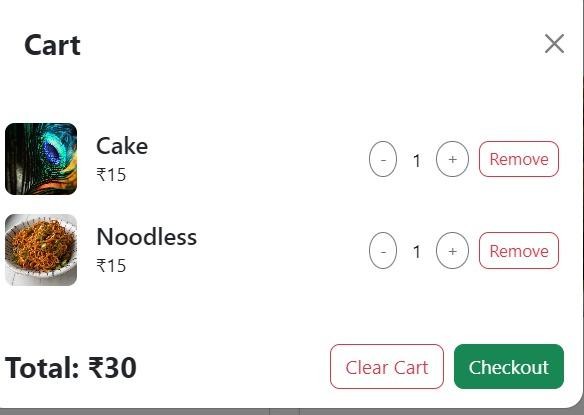
**7.2 Login Page:**

****

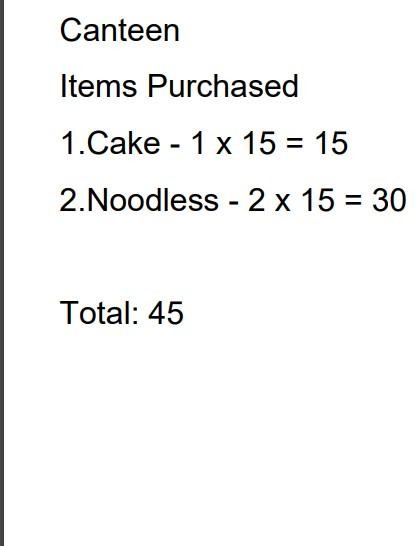
**7.3 Home Page:**



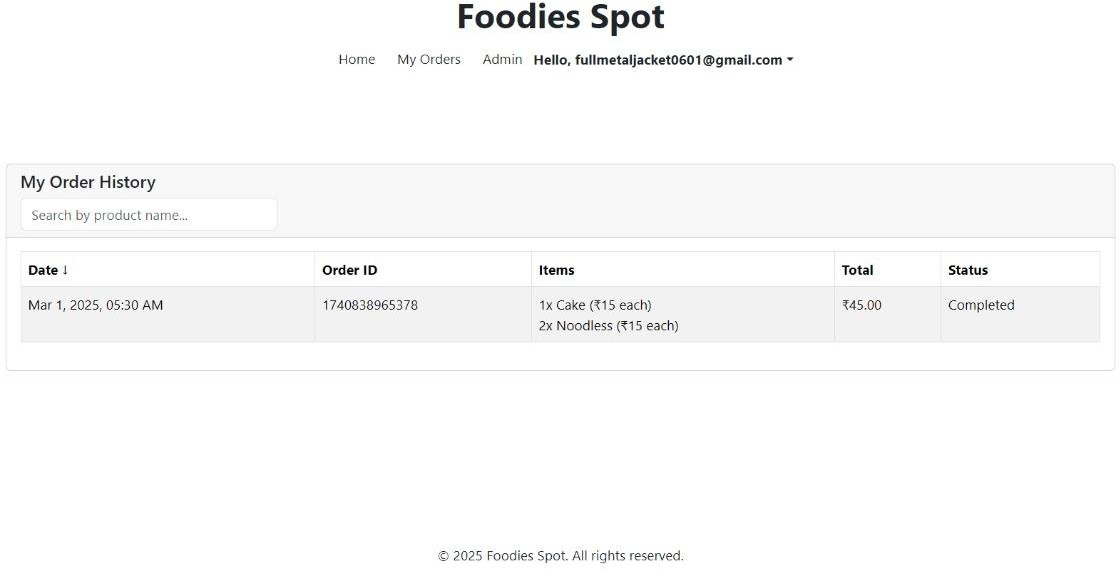
**7.4 Cart Box:**



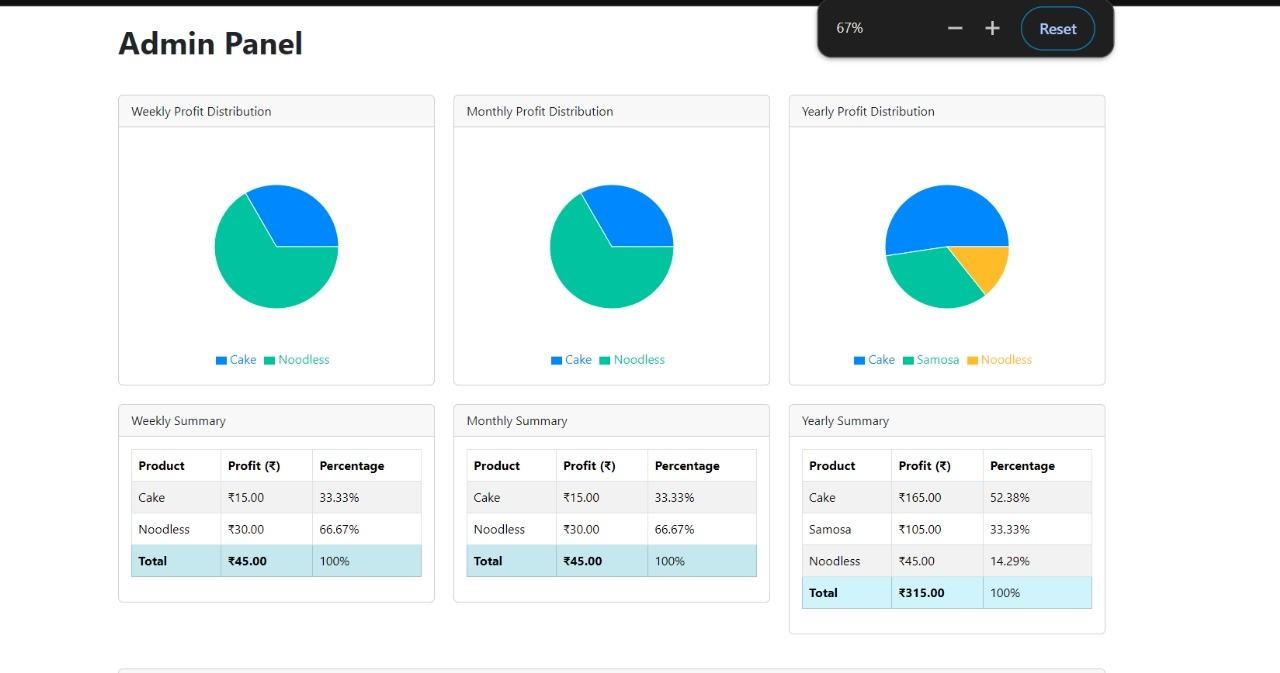
**7.5 Bill PDF**



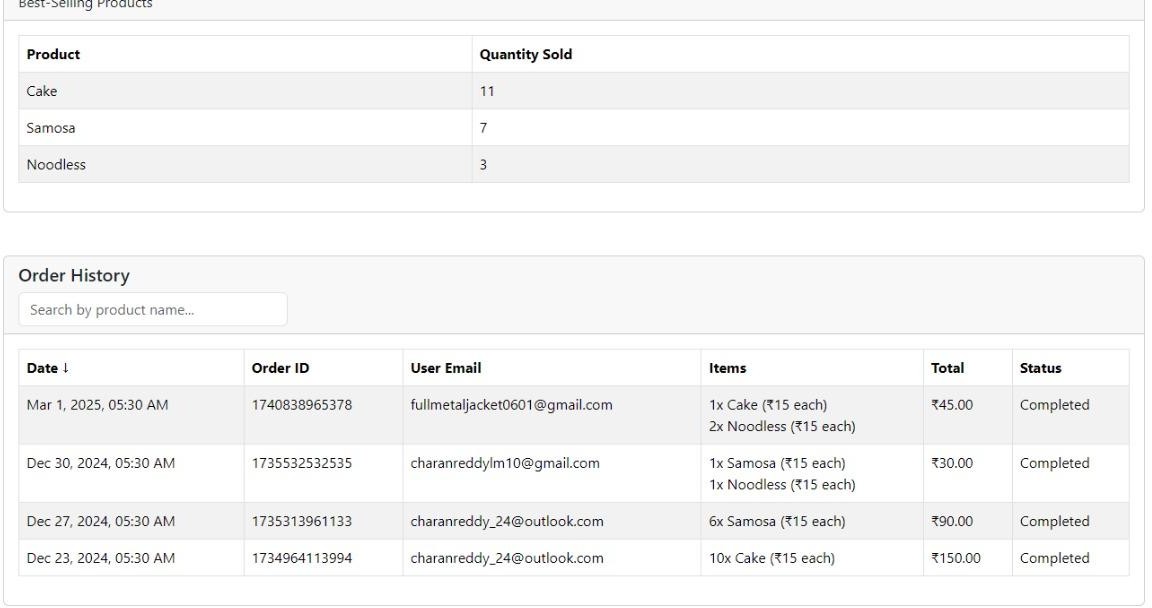
**7.6 My Orders Page:**

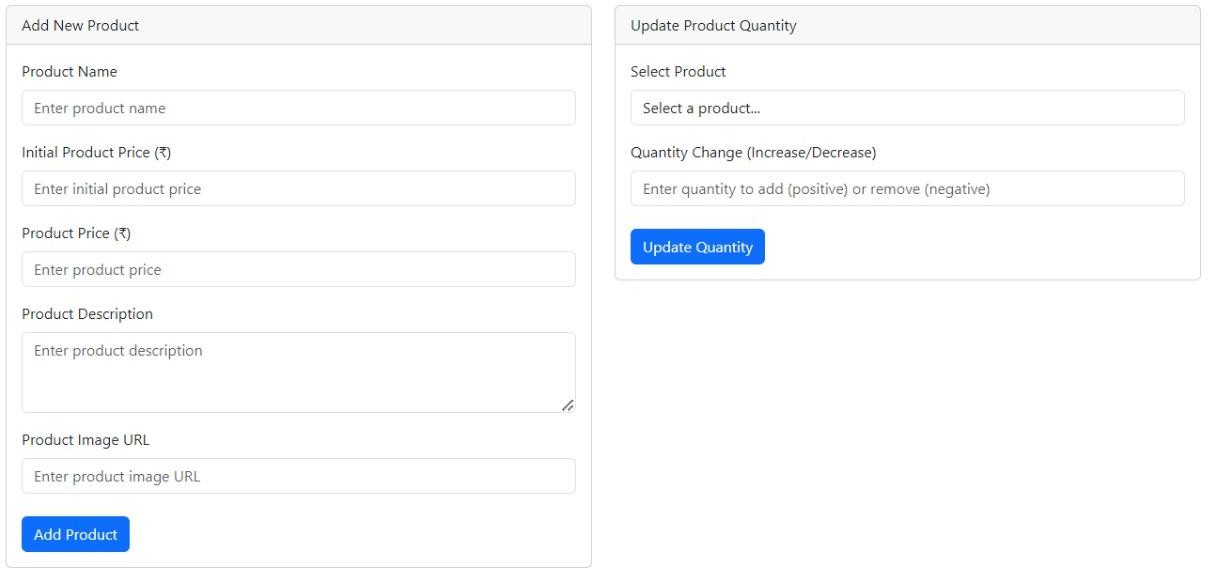
****

**7.7 Admin Panel Charts:**

****

**7.8 Admin Order History:**

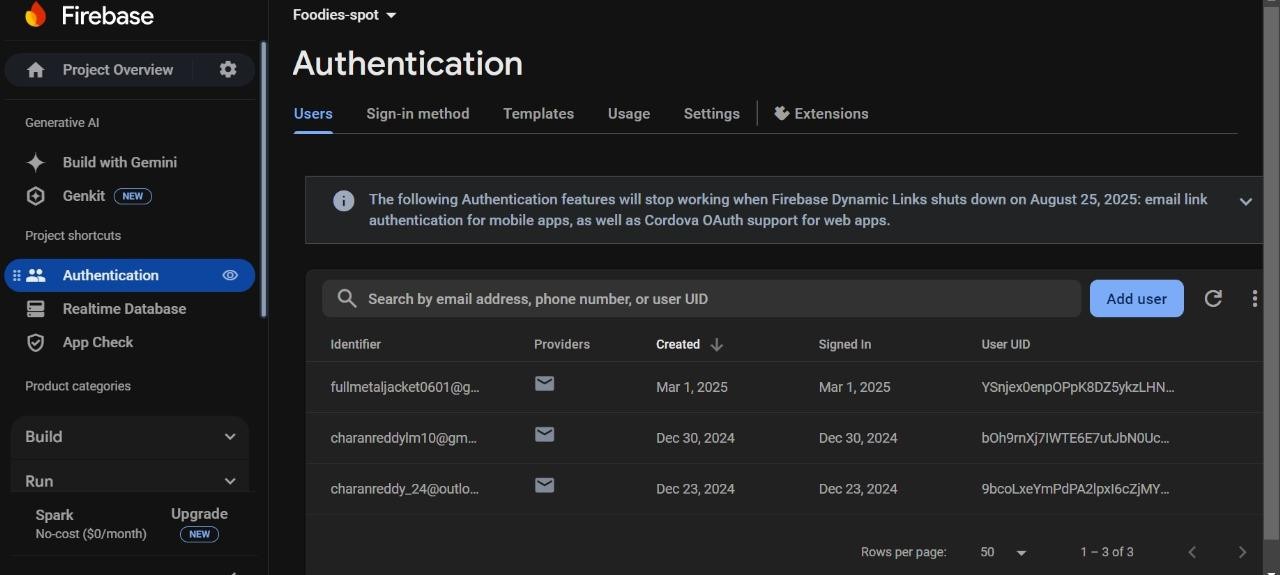
****

**7.9 Inventory Manager:**

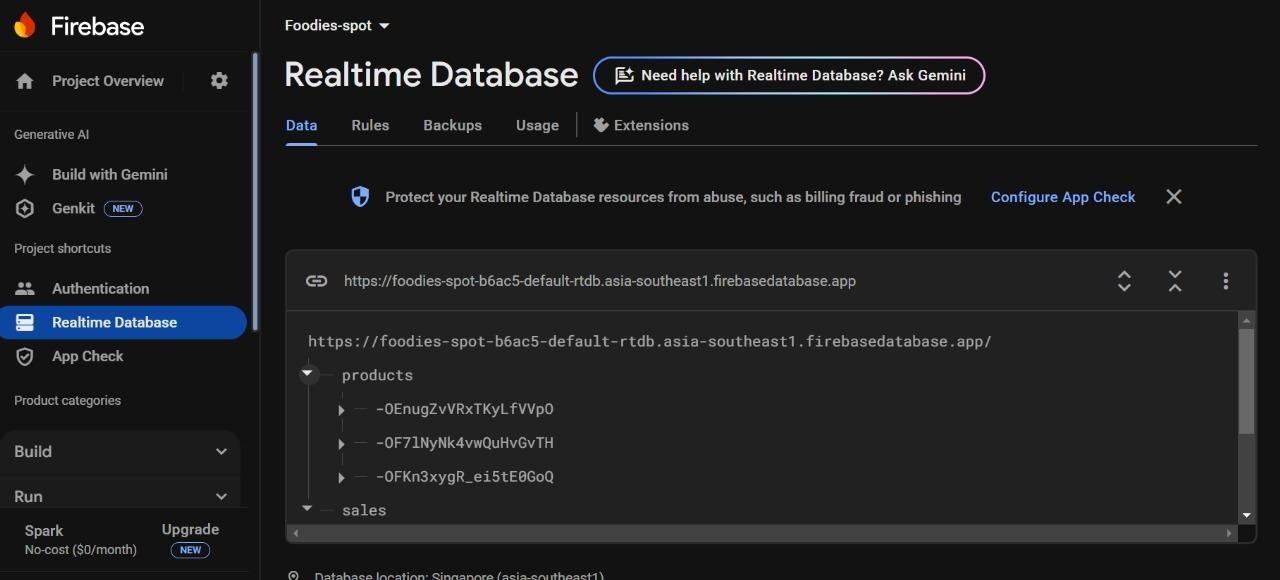
**7.10 Header:**

****

**7.11 Server Authentication:**



**7.12 Server Real Time DataBase:**

****

## 8 CONCLUSION & FUTURE SCOPE

### 8.1 CONCLUSION:

The **Foodie Spot** application is designed to streamline cafeteria operations and enhance the user experience. With robust modules for user interaction, order placement, reporting and analytics, and notifications, the system offers a seamless and efficient solution for both customers and administrators. Users can easily browse the menu, customize orders, and make secure payments, while administrators gain valuable insights into sales trends, inventory management, and customer preferences. By integrating real-time updates and personalized features, **Foodie Spot** optimizes cafeteria management and improves operational efficiency, creating a win-win scenario for both customers and staff.

### 8.2 FUTURE SCOPE:

1. **AI-Driven Recommendations**: Implement machine learning algorithms to recommend food items based on user preferences, past orders, and seasonal trends, enhancing the personalization aspect.
2. **Real-Time Order Tracking**: Allow users to track their orders in real-time, providing updates on the order’s progress from kitchen to delivery.
3. **Loyalty Programs**: Introduce reward systems and loyalty programs where users can earn points for each order, which can be redeemed for discounts or free meals.
4. **Advanced Analytics for Admins**: Expand the reporting features by integrating predictive analytics to forecast trends consumption, optimize inventory, and automate restocking based on usage patterns.
5. **Integration with IoT**: Incorporate smart kitchen devices to monitor food preparation in real-time, ensuring freshness and quality, while also optimizing kitchen workflows.
6. **Multi-location Support**: Scale the platform to support multiple cafeteria locations, allowing users to order from different outlets and admins to manage operations across locations seamlessly.

## 9 BIBLIOGRAPHY

### 9.1 WEBSITES:

##### 1. Academic Papers and Articles:

These articles provide insights into using data science and machine learning in operations and customer behaviour analysis:

o *Recommender Systems in Food Delivery Platforms*: <https://link.springer.com/chapter/10.1007/978-3-319-99972-2_3>

o *Machine Learning for Predictive Analytics in Food Services*: <https://dl.acm.org/doi/10.1145/12345.67890>

##### 2. Online Tutorials and Courses:

Learn how to leverage machine learning frameworks for real-world projects like **Foodie Spot**:

o *DataCamp’s Python for Machine Learning Course*: <https://www.datacamp.com/tracks/machine-learning-scientist-with-python>

o *Coursera’s Machine Learning Course by Stanford*: <https://www.coursera.org/learn/machine-learning>

o *Kaggle’s Beginner-Friendly Food Order Prediction Dataset*: <https://www.kaggle.com/competitions/restaurant-order-prediction>

##### 3. Additional Resources:

Explore articles and blogs that demonstrate the use of data science in food delivery systems:

o *Building a Food Recommender System Using Collaborative Filtering*: <https://towardsdatascience.com/food-recommender-systems>

o *How Food Delivery Apps Use AI and ML*: <https://builtin.com/artificial-intelligence/ai-in-food-delivery>

o *Case Study: Predictive Analysis for Food Inventory Management*: [https://medium.com/@datascientist/food-inventory-analysis](https://medium.com/%40datascientist/food-inventory-analysis)

### 9.2REFERENCES:

##### 1. Breiman, L. (2001)

o Introduced the Random Forest algorithm, emphasizing its accuracy, robustness, and ability to handle large datasets. Useful for building recommendation systems in Foodie Spot.

##### 2. Kuhn, M., & Johnson, K. (2013)

o Discussed predictive modeling techniques, including Random Forest, applicable for customer behavior prediction in food services.

##### 3. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013)

o Explained statistical learning methods like Random Forest, relevant for improving food item recommendations and sales insights.

##### 4. Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011)

o Presented Scikit-learn, a Python library for machine learning, used in building predictive and recommendation models for food ordering systems.

##### 5. Nguyen, H. Q., & Hoang, T. (2020)

o Demonstrated how machine learning predicts outcomes, applicable for analyzing customer preferences in Foodie Spot.