UNIFIED MENTOR

YOUR SKILL, SUCCESS & JOURNEY

A

PROJECT REPORT

ON

Catering Reservation and Ordering System

By

SATVIK SUNIL BODKE

Internship ID: UMIP279303

satvikbodke@gmail.com

Internship Duration : February 25, 2025 - August 25, 2025

Company: Unified Mentor

Domain: Web Development

Submission Date: August 25, 2025

DECLARATION

I, Satvik Sunil Bodke, do hereby solemnly declare and affirm that this project report, titled “Catering Reservation and Ordering System,” is a complete and authentic record of the original work I have personally and independently undertaken. This project was completed in a remote, online internship environment provided by Unified Mentor, and it is submitted in partial fulfillment of the program's requirements.
The entire architecture, codebase, and system design documented herein are the product of my own effort, developed using the online study resources and guidance provided. The work is free from plagiarism, and all external libraries or code snippets have been appropriately used and acknowledged. I certify that this submission is a true representation of my capabilities and the practical application of my learning in a self-directed setting.

ACKNOWLEDGEMENTS

I wish to express my profound and sincere gratitude to Unified Mentor for curating an exceptional online internship program that provided me with the opportunity and resources to develop this project. The well-structured online materials and clear objectives were instrumental in guiding my development process in a remote setting.
I am particularly grateful for the self-paced learning model, which fostered an environment of independence and deep learning. This experience was crucial in building not only my technical proficiency in full-stack web development with JavaScript and Firebase but also in strengthening my skills in self-management, problem-solving, and independent research. Navigating the challenges of building a complete e-commerce application from the ground up on my own has been a tremendous learning experience. I am indebted to the organization for providing a platform that truly encourages and respects independent effort and practical application of knowledge.

ABSTRACT

This report provides a comprehensive architectural and functional overview of the “Catering Reservation and Ordering System,” also known as "The Golden Spoon," a modern e-commerce web application. The project was conceived to provide a digital platform for caterers, particularly those in rural areas, to showcase their products and reach a broader market. The core objective was to engineer a secure, two-role system that serves the distinct needs of an administrator for product management and customers for a seamless ordering experience, thereby promoting local culinary skills and traditional culture.
The system is architected as a modern client-server application, leveraging a Backend-as-a-Service (BaaS) model for scalability and rapid development. The front-end was meticulously crafted with HTML5, CSS3, and vanilla JavaScript, featuring a persistent client-side shopping cart managed with localStorage. The entire backend is powered by Google Firebase, utilizing Firebase Authentication for secure user login and Firestore Database as the real-time data store for products and orders.
The application’s functionality is segregated into two primary roles: an Administrator dashboard for uploading new catering products and viewing all customer orders, and a Customer portal that allows for user registration, browsing the menu, managing a shopping cart, placing orders, and viewing personal order history. The final deployed system is a robust, functional e-commerce platform that successfully fulfills its mission, serving as a powerful demonstration of building a complete web application in a self-directed project environment.

INDEX

1. Introduction

The rise of the digital economy has created unprecedented opportunities for businesses of all sizes. However, small-scale entrepreneurs and local artisans, such as independent caterers, often lack the technical resources to establish a strong online presence. This project, the "Catering Reservation and Ordering System," was undertaken as a capstone effort during a self-driven online internship to address this digital divide. It aims to provide a simple yet powerful e-commerce solution that empowers caterers to market their services effectively. By creating an accessible and intuitive online portal, this system not only facilitates business growth but also helps promote local and traditional cuisine. This report documents the comprehensive process of building this application, from conceptual design and technological architecture to final implementation and deployment.

2. Problem Statement

The primary challenge facing many small and rural caterers is market access. Their reliance on traditional, word-of-mouth marketing and manual ordering processes (via phone or in-person visits) severely limits their customer base and operational efficiency. They lack a dedicated, secure, and user-friendly platform to display their menu, manage orders, and process customer requests. This results in missed business opportunities, administrative burdens, and an inability to compete in the expanding digital marketplace. The core problem is the absence of a low-cost, easy-to-manage technological solution that allows these caterers to transition their business online, manage product information, and streamline the entire customer ordering journey.

3. Project Objectives

To deliver a robust solution, the project was guided by several key technical and functional objectives:

- **To Develop a Secure, Two-Role System:** The primary objective was to engineer an application with distinct access control and functionalities for two roles:
    - **Administrator:** Responsible for all back-end product management and order oversight.
    - **Customer:** The end-user who browses the menu and places orders.
- **To Implement a Persistent Client-Side Cart:** A crucial goal was to create a seamless shopping experience by using localStorage to manage the user's shopping cart, ensuring items are retained even if the browser is refreshed.
- **To Create a Full E-commerce Workflow:** The system was designed to provide a complete customer journey, from user registration and product browsing to placing an order and viewing subsequent order history.
- **To Leverage a Scalable and Serverless Backend:** A core technical goal was to build the application on Google Firebase, utilizing its services to handle authentication, data storage, and hosting without the need for traditional server management.

4. Scope and Limitations

Scope

The scope of this project is to provide a fully functional, albeit streamlined, e-commerce platform for a catering business.

- **Admin Scope:** The system provides a secure dashboard for a single, hardcoded administrator (admin@catering.com). This user has the ability to perform **Create** operations (upload new products with a name, description, and price) and **Read** operations (view a comprehensive list of all orders placed by all customers).
- **Customer Scope:** Any registered user can log in to browse the menu, add or remove items from their personal shopping cart, place an order, and view a history of their own past orders.

Limitations

The project was intentionally scoped to be a foundational e-commerce platform and has several limitations.

- **No Payment Gateway Integration:** The "Place Order" functionality records the order in the database but does not process any real financial transactions.
- **Single Administrator:** The system is designed for a single administrator whose email is hardcoded. There is no functionality for creating multiple admin or staff accounts.

- **No Inventory Management:** The system does not track product stock levels. Any item uploaded by the admin is considered infinitely available.
- **No Real-Time Order Status Updates:** The order status is fixed to "Placed" upon creation. There is no admin functionality to update an order's status (e.g., to "Preparing" or "Delivered").

5. Existing System Analysis

The online food ordering market is dominated by large-scale aggregators (like Zomato or Uber Eats) and complex e-commerce platforms (like Shopify).

- **Food Aggregator Platforms:** While these platforms offer immense market reach, they often charge high commission fees and provide limited brand control, which can be prohibitive for small caterers.
- **E-commerce Platform Builders:** Services like Shopify or WooCommerce are powerful but can be complex and costly to set up and maintain for a business with simple needs.
- **The Proposed System's Niche:** This project carves a niche by providing a custom, lightweight, and commission-free solution. It offers the core functionalities needed by a small caterer without the overhead, complexity, or cost of larger platforms, making it an ideal entry point for businesses looking to establish their first online presence.

6. Requirements Analysis

6.1 Functional Requirements

A detailed breakdown of the functional requirements for each module:

- **Admin Module:**
  - The system shall authenticate a user as an admin if their email matches admin@catering.com.
  - The system shall provide a form for the admin to upload a new product, including its name, description, and price.
  - The system shall display a list of all orders from all customers, showing the order date, items, total amount, and user ID
- **Customer Module:**
  - The system shall allow new users to register and existing users to log in.
  - The system shall display all available products from the database on the main menu page.
  - The system shall allow users to add any product to a client-side shopping cart.
  - The system shall allow users to remove items from their cart.
  - The system shall persist the cart's contents using local Storage
  - The system shall allow logged-in users to place an order, which records their cart contents and user ID in the database.
  - The system shall display a history of a user's own past orders.

6.2 Non-functional Requirements

- **Security:** User authentication must be secure. Customers should only be able to view their own orders.
- **Performance:** The client-side cart implementation ensures that adding/removing items is instantaneous. Database reads for products and orders should be efficient.
- **Usability:** The interface must be clean, modern, and intuitive for both the admin and customers.
- **Data Integrity:** The system must accurately record the items and total price at the time of order placement.

.

6.3 Hardware/Software Requirements

- A computer or mobile device with a modern web browser (e.g., Google Chrome, Mozilla Firefox).
- No specific hardware requirements are needed beyond the ability to run a web browser.

7. System Design

7.1 Architecture

The application is built on a serverless architecture, which minimizes maintenance overhead and enhances scalability.

- **Frontend (Client):** The client-side is responsible for rendering the UI and managing all user interactions and application state. It uses HTML, CSS, and modular JavaScript to create a dynamic experience. A key architectural choice was the use of localStorage for shopping cart management, making the cart state fast and resilient to page reloads.
- **Backend (Firebase):** Google Firebase provides all backend services. Firebase Authentication handles user identity, while Firestore, a NoSQL database, stores the product catalog and all customer orders. Security is managed through a combination of client-side authentication checks and Firestore Security Rules (in a production scenario).
- **Data Flow Example (Placing an Order):**
    1. A customer adds items to their cart; the data is written to the browser's localStorage.
    2. The customer navigates to the cart page, which reads from localStorage to display the items.
    3. Upon clicking "Place Order," the application verifies the user is logged in via Firebase Auth.
    4. A new document is created in the orders collection in Firestore, containing the userId, a copy of the cart items, the total amount, and a server-generated timestamp.
    5. Finally, the client-side localStorage cart is cleared.

7.2 Database Design

The Firestore schema is simple and effective, using two main collections:

- **products**: A top-level collection where each document represents a catering item. Fields include name (string), description (string), and price (number).
- **orders**: A collection where each document is a customer order. Fields include userId (string, linking to the user in Firebase Auth), items (an array of map objects, each with item details), totalAmount (number), status (string), and orderDate (timestamp).

8. Implementation

8.1 Technology Stack

- **Frontend Development:** HTML5, CSS3, JavaScript (ES6)
- **Backend Services:** Google Firebase (Firestore Database, Firebase Authentication)
- **Client-Side Storage:** Browser localStorage API
- **Deployment and Hosting:** Firebase Hosting

8.2 Code Structure

The application's logic is cleanly separated into different JavaScript files.

- **auth.js**: This module handles both registration (createUserWithEmailAndPassword) and login (signInWithEmailAndPassword). Its most critical feature is the post-login redirect logic, which checks if the logged-in user's email is admin@catering.com and routes them to the admin dashboard accordingly.
- **main.js**: This script is for the customer-facing menu. It fetches all documents from the products collection in Firestore and dynamically renders them on the page. It also contains the event listeners for the "Add to Cart" buttons, which manage the localStorage object.
- **cart.js:** This module is responsible for the shopping cart page. It reads the cart data from localStorage to render the items and calculate the total. It also contains the "Place Order" logic, which constructs the order object and writes it to the Firestore orders collection using the addDoc function.
- **admin/dashboard.js and admin/view-orders.js:** These scripts power the admin interface. dashboard.js handles the form for uploading new products to the products collection. view-orders.js performs a global query on the orders collection to fetch and display every order placed on the platform.

9. Testing

9.1 Testing Methods

To ensure a high-quality, bug-free application, a comprehensive testing strategy was employed, covering all core functionalities for both the administrator and customer roles. The tests were designed to validate the end-to-end user workflow and data integrity.

9.2 Test Cases

Test Case ID	Action	Expected Result	Actual Result
TC-01	A user logs in with the email admin@catering.com.	The user is successfully authenticated and redirected to the /admin/dashboard.html page.	Pass
TC-02	A user logs in with a non-admin email.	The user is successfully authenticated and redirected to the main menu page (/index.html).	Pass
TC-03	Admin fills out the "Upload Product" form and submits it.	A new document is created in the products collection in Firestore, and the product appears on the customer menu.	Pass
TC-04	A customer adds three different items to their cart.	The localStorage "cart" object should contain an array of three unique items with their quantities.	Pass
TC-05	The customer adds an item that is already in the cart.	The quantity of the existing item in the localStorage cart should increment by one. The total price updates correctly.	Pass
TC-06	The customer refreshes the browser window while on the cart page.	The cart page reloads and displays the same items and total, as the data is read from localStorage.	Pass
TC-07	A logged-in customer clicks the "Place Order" button with items in their cart.	A new document is created in the orders collection in Firestore. The customer's localStorage cart is cleared.	Pass
TC-08	A customer navigates to the "My Orders" page.	The page displays only the orders placed by that specific customer, matching their userId.	Pass
TC-09	The Admin navigates to the "All Customer Orders" page.	The page displays a list of every order from every customer stored in the orders collection.	Pass
TC-10	A user who is not logged in tries to place an order.	An alert message is shown prompting the user to log in first. The order is not placed.	Pass

10. Results

The project successfully achieved its goal of creating a functional and deployed Catering Reservation and Ordering System. The application performs all its core functions as designed, providing a seamless experience for both administrators and customers. The two-role architecture effectively separates concerns, and the use of Firebase as a backend provides a scalable and reliable foundation. The client-side cart managed with localStorage proved to be a highly responsive and user-friendly solution. The final deployed application is a robust and complete product that effectively solves the initial problem statement.

11. Conclusion

Summary of Achievement This project successfully transformed a concept into a fully realized e-commerce web application. The "Catering Reservation and Ordering System" addresses the critical need for small-scale caterers to have a digital presence, providing a platform that is both powerful in function and simple in its execution. The system's design showcases a strong understanding of modern web development, including front-end state management, serverless architecture, and database design.

Personal Learning and Reflection This project was a significant endeavor undertaken in a self-directed online internship. The process was a practical deep-dive into full-stack development, demanding not just coding skills but also discipline in planning, architecture, and problem-solving. Implementing the entire workflow, from user authentication to client-side state management and database interaction, provided invaluable hands-on experience. This project stands as a major achievement in my learning journey, demonstrating my capability to independently build and deploy a complete, real-world application from start to finish.

12. Future Enhancements

While the current application is a complete product, several features could be added in the future to enhance its commercial viability:

- **Payment Gateway Integration:** Integrating a service like Stripe or PayPal to process real credit card payments at the time of order.
- **Real-Time Order Status Management:** Adding functionality for the admin to update an order's status (e.g., "Confirmed," "Preparing," "Out for Delivery"), which would then be reflected in the customer's "My Orders" page in real-time.
- **Inventory Management:** Implementing a system to track the stock levels of products, automatically marking items as "sold out" when they are no longer available.
- **Multi-Admin/Staff Roles:** Expanding the authentication system to allow for multiple administrator or staff accounts with varying levels of permissions.

13. References

During the development of this project, the following industry-standard resources were consulted for official documentation, technical guidance, and best practices:

- Firebase Official Documentation: firebase.google.com/docs
- MDN Web Docs (Mozilla Developer Network): developer.mozilla.org
- W3Schools: w3schools.com
- Stack Overflow Community: stackoverflow.com