



FULLSTACK WEB PROJECT

Waterlily Labs Assessment

Submitted By: GGG Wathsala

wathsala.g@waterlilylabs.com

<https://github.com/wathsalaggg/product-dashboard.git>

Contents

Introduction	3
Project Objectives	3
Project Overview	3
Technology Stack	4
Backend Technology Stack (.NET)	4
Frontend Technology Stack (Dual UI Approach)	4
Entity Relationship Diagram (ERD)	6
Architecture Patterns	6
Key Integration Points	7
Development Workflow	7
Implemented Features	7
Core Requirements Met	7
Technical Implementation	8
Responsiveness Implementation	8
Challenges Faced & Solutions	8
Backend Challenges	8
Frontend Challenges	9
React-Specific Challenges	9
Error Handling	10
Development & Build Tools	10
Performance Optimizations	10
Assessment Requirement Validation	11
Getting Started	11
Future Enhancements	12
Screenshots	13

Introduction

This document provides comprehensive technical documentation for the **Product Dashboard and Shopping Cart system**, a hybrid web application leveraging both ASP.NET MVC with Razor Views and React.js for the user interface. This solution represents a modern approach to web development, combining the strengths of server-side rendering with client-side interactivity.

The application serves as a product management dashboard with integrated e-commerce functionality, allowing users to browse, search, and filter products while maintaining a shopping cart across sessions. The implementation follows industry best practices for scalability, maintainability, and performance.

Project Objectives

Primary Business Objectives

1. **Create an intuitive product management dashboard** that allows users to efficiently browse, search, and manage product inventory
2. **Implement a seamless shopping experience** with persistent cart functionality across sessions
3. **Deliver a responsive application** that provides consistent user experience across desktop, tablet, and mobile devices
4. **Increase conversion rates** through improved product discovery and streamlined checkout process

Technical Objectives

1. **Implement hybrid architecture** leveraging both ASP.NET MVC Razor Views and React.js for optimal performance and developer experience
2. **Establish clear separation of concerns** between presentation logic (React/Razor) and business logic (.NET controllers/services)
3. **Create reusable component library** for consistent UI patterns and reduced

Project Overview

- Hybrid Architecture
 - Utilizes both server-rendered Razor Views and client-side React components
- Responsive Design

- Fully functional across desktop, tablet, and mobile devices
- Dynamic Functionality
 - Implements real-time search, filtering, and cart management
- Modern Development Practices
 - Incorporates reusable components, AJAX communication, and clean separation of concerns

Technology Stack

Backend Technology Stack (.NET)

Core Framework:

- .NET 8.0
- ASP.NET Core MVC
- Entity Framework Core 8.x

Database:

- MySQL 8.0+ (via Pomelo.EntityFrameworkCore.MySql)
- Database migrations with EF Core

API & Services:

- RESTful API Controllers
- JSON serialization with System.Text.Json
- Swagger/OpenAPI documentation

Session Management:

- Distributed memory cache
- Session state management

Development Tools:

- Swagger UI for API testing
- EF Core migrations for database management

Frontend Technology Stack (Dual UI Approach)

Primary: Razor Views

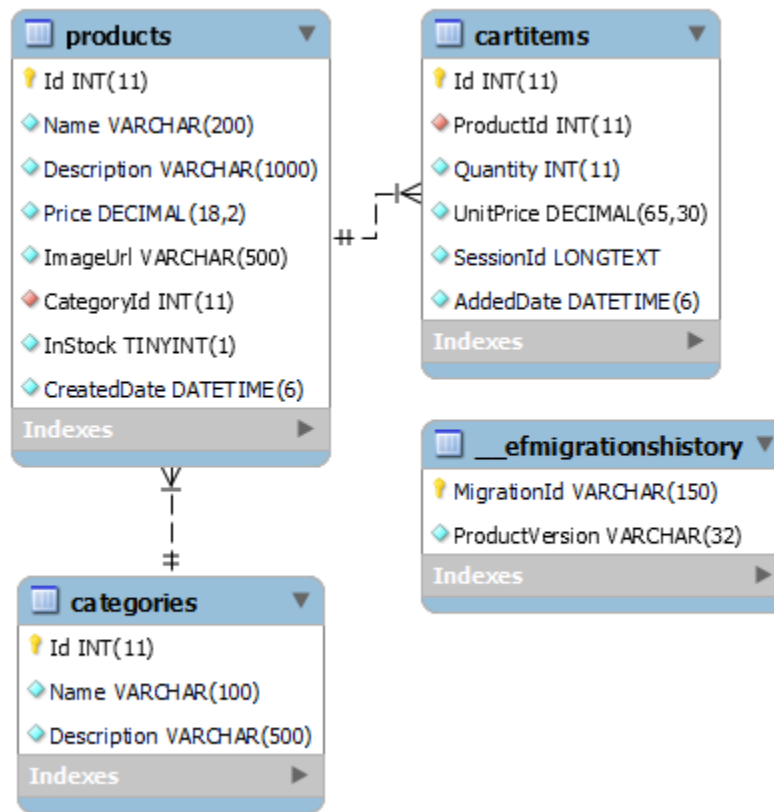
- ASP.NET Core Razor Pages
- Server-side rendering

- jQuery (implied by assessment requirements)
- Bootstrap or custom CSS (implied by responsive requirements)

Optional: React SPA

- React 19.1.1
- TypeScript
- Vite (build tool)
- Tailwind CSS 4.x
- Headless UI components
- Heroicons
- Framer Motion (animations)
- Zustand (state management)
- TanStack React Query (data fetching)
- React Router DOM 7.x
- Axios (HTTP client)

Entity Relationship Diagram (ERD)



Architecture Patterns

Backend

- MVC Pattern (Controllers, Views)
- Service Layer (ProductService)
- Repository Pattern (EF Core)
- Dependency Injection

Frontend

- **Razor Views**: Traditional server-rendered pages with jQuery AJAX
- **React**: Component-based SPA (optional alternative)
- Shared API endpoints serving both UIs

Key Integration Points

1. Razor Views as Primary UI
 - Main application uses server-rendered Razor views
2. React as Optional SPA
 - Alternative UI available but not required
3. Shared API Backend
 - Both UIs consume the same .NET API endpoints
4. Session Management
 - Shopping cart stored in server sessions for both UIs
5. Database Integration
 - EF Core with MySQL for data persistence

Development Workflow

1. **Primary Development:** Razor Views with jQuery
2. **Optional React Path:** Parallel SPA development
3. **API-First Approach:** Backend serves both UI approaches
4. **Database Management:** EF Core migrations with MySQL

Implemented Features

Core Requirements Met

- **Product List:** Responsive grid with pagination (>10 items) using partial views
- **Search Functionality:** Real-time filtering with text highlighting
- **Category Filtering:** AJAX-based filtering without page refresh
- **Shopping Cart:** Add/remove items, quantity management, total price calculation
- **Reusable JS Functions:** Generic modal and AJAX helpers
- **Responsive Layout:** Mobile, tablet, and desktop compatible
- **Details Preview:** Modal-based product details with image download

Technical Implementation

- MVC separation with proper controller/view structure
- jQuery AJAX for dynamic content loading
- Partial views for pagination and product rendering
- Session-based cart management
- React alternative UI (optional)

Responsiveness Implementation

Breakpoints Handled:

- **Mobile:** < 768px (vertical layout, hamburger menu)
- **Tablet:** 768px - 1024px (adjusted grid, compact navigation)
- **Desktop:** > 1024px (full grid, sidebar filters)

Responsive Techniques:

- CSS Flexbox/Grid for product layout
- Media queries for adaptive styling
- Touch-friendly interfaces for mobile
- Conditional rendering based on screen size

Challenges Faced & Solutions

Backend Challenges

1. Database Integration

- Challenge: MySQL compatibility with EF Core
- Solution: Used Pomelo.EntityFrameworkCore.MySql provider

2. Session Management

- Challenge: Maintaining cart state across requests
- Solution: Implemented distributed memory cache with session

3. API Design

- Challenge: Creating endpoints for both Razor and React
- Solution: RESTful design with consistent JSON responses

Frontend Challenges

1. Dual UI Approach

- Challenge: Maintaining two frontend implementations
- Solution: Shared API endpoints with UI-specific adaptations

2. Real-time Search

- Challenge: Performance with large product catalogs
- Solution: Debounced API calls with server-side filtering

3. Cart Synchronization

- Challenge: Keeping UI in sync with server state
- Solution: jQuery DOM manipulation after AJAX calls

4. Modal Management

- Challenge: Reusable modal system
- Solution: Created generic JavaScript modal controller

React-Specific Challenges

1. State Management

- Challenge: Sharing state between components
- Solution: Implemented Zustand for global state

2. Data Fetching

- Challenge: Efficient API calls with caching
- Solution: Used TanStack React Query

Error Handling

Client-Side Errors:

- AJAX error handling with user feedback
- Form validation with clear error messages
- Fallback UI for failed API calls

Server-Side Errors:

- Global exception handling middleware
- Structured error responses
- Logging implementation

Development & Build Tools

Backend

- .NET CLI
- Entity Framework Core CLI

Frontend

- Vite (for React build)
- TypeScript compiler
- ESLint + Prettier
- Tailwind CSS

Performance Optimizations

Implemented

- Pagination to limit data transfer
- Debounced search to reduce API calls
- EF Core query optimization
- Bundle splitting for React build

Future Improvements

- Response caching for static data
- CDN for static assets
- Database indexing optimization

Assessment Requirement Validation

Requirement	Status	Implementation Details
MVC Structure	Done	Proper separation with controllers, views, models
jQuery Usage	Done	AJAX calls, DOM manipulation, event handling
Responsiveness	Done	Mobile-first design with breakpoints
UI/UX Design	Done	Clean, professional interface
Code Quality	Done	Reusable functions, comments, standards
Functionality	Done	All required features implemented

Getting Started

Prerequisites:

- .NET 8.0 SDK
- Node.js 18+ (for React build)
- MySQL 8.0+
- Git

Installation:

1. Clone repository: `git clone https://github.com/wathsalaggg/product-dashboard.git`
2. Database setup: `dotnet ef database update`
3. Seed data: `dotnet run SeedData`
4. Run application: `dotnet run`
5. Or use `.net run` option

To see the frontend

1. In the terminal run “npm install”
2. Run -> dotnet run --environment Development --project ProductDashboardBackend.csproj
3. Go to <http://localhost:49783/swagger/index.html>
4. Run “npm run dev” in vs code

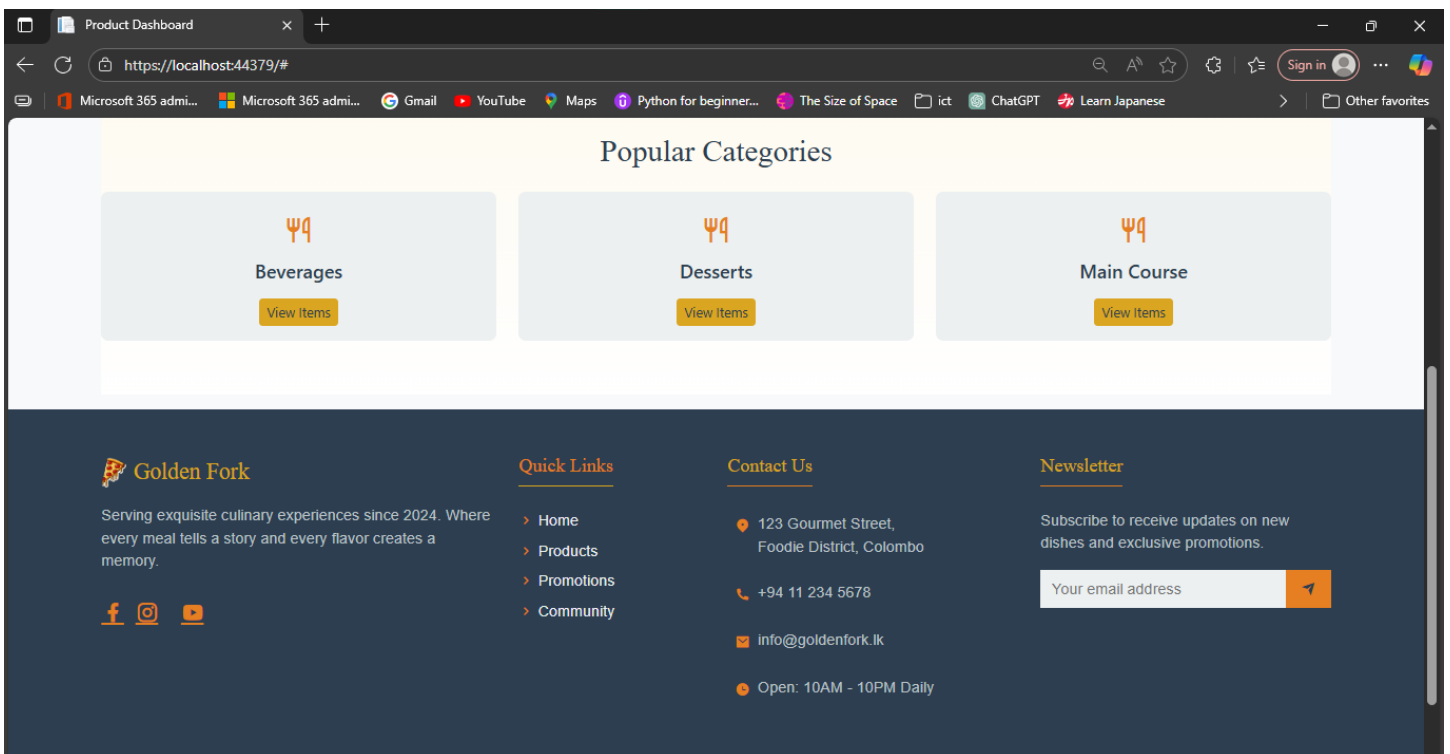
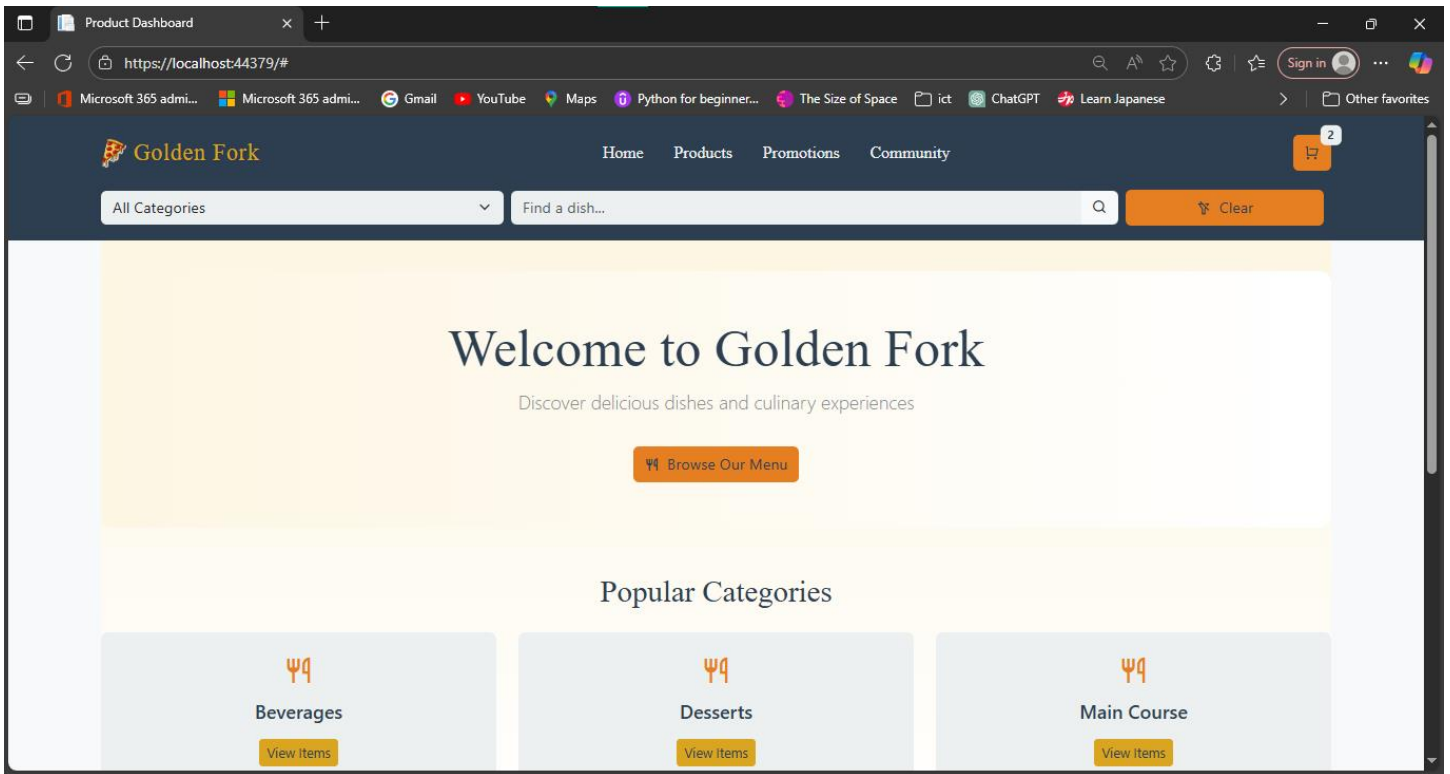
Future Enhancements

Planned Features:

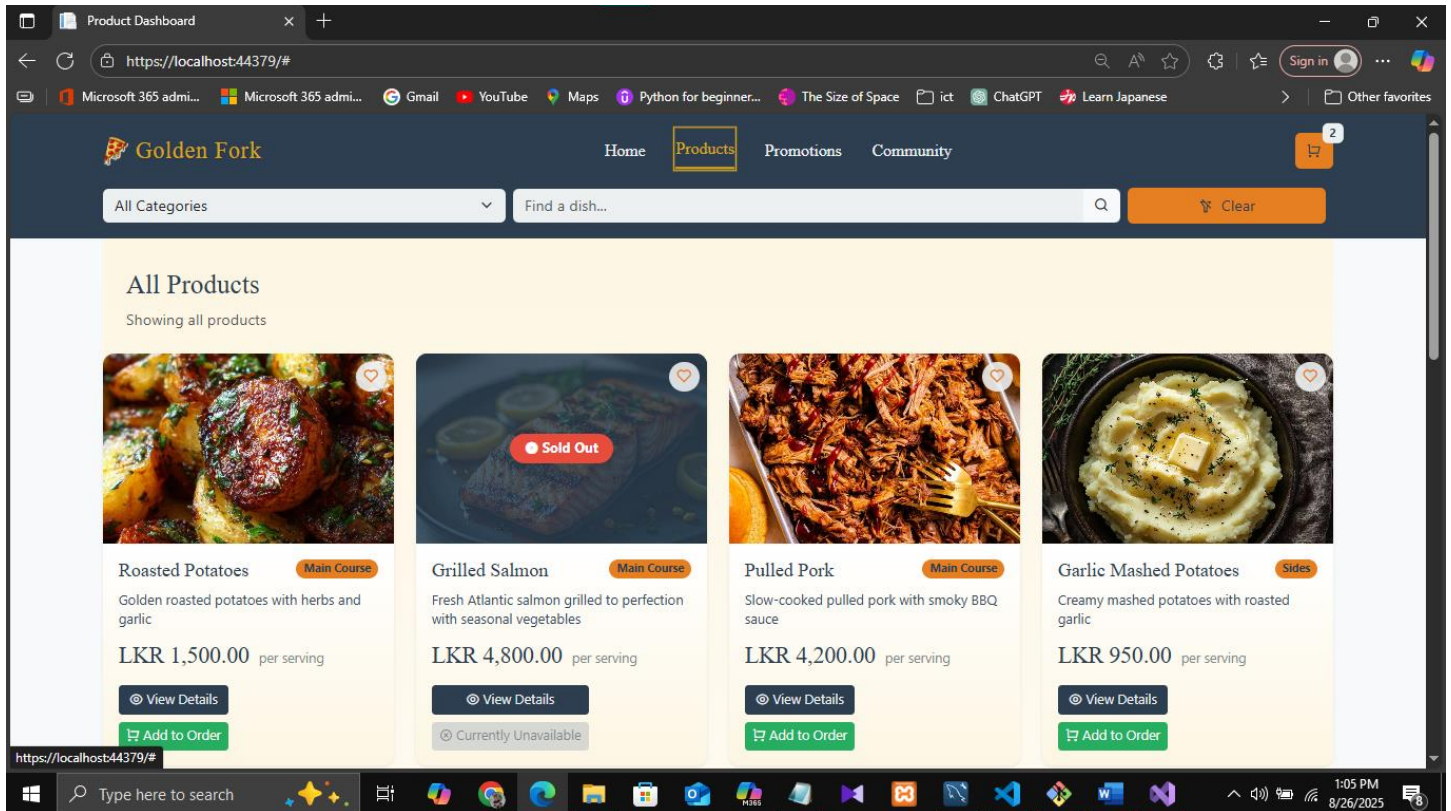
- User authentication and profiles
- Payment integration
- Order history and tracking
- Product reviews and ratings
- Admin dashboard for product management

Screenshots

Home page




Products page



Modal

✕ Product Details

✕



⌵

Spaghetti Carbonara

Main Course

Creamy pasta with crispy bacon, eggs, and parmesan cheese

LKR 3,500.00

per serving

✔ In Stock - Ready to order!

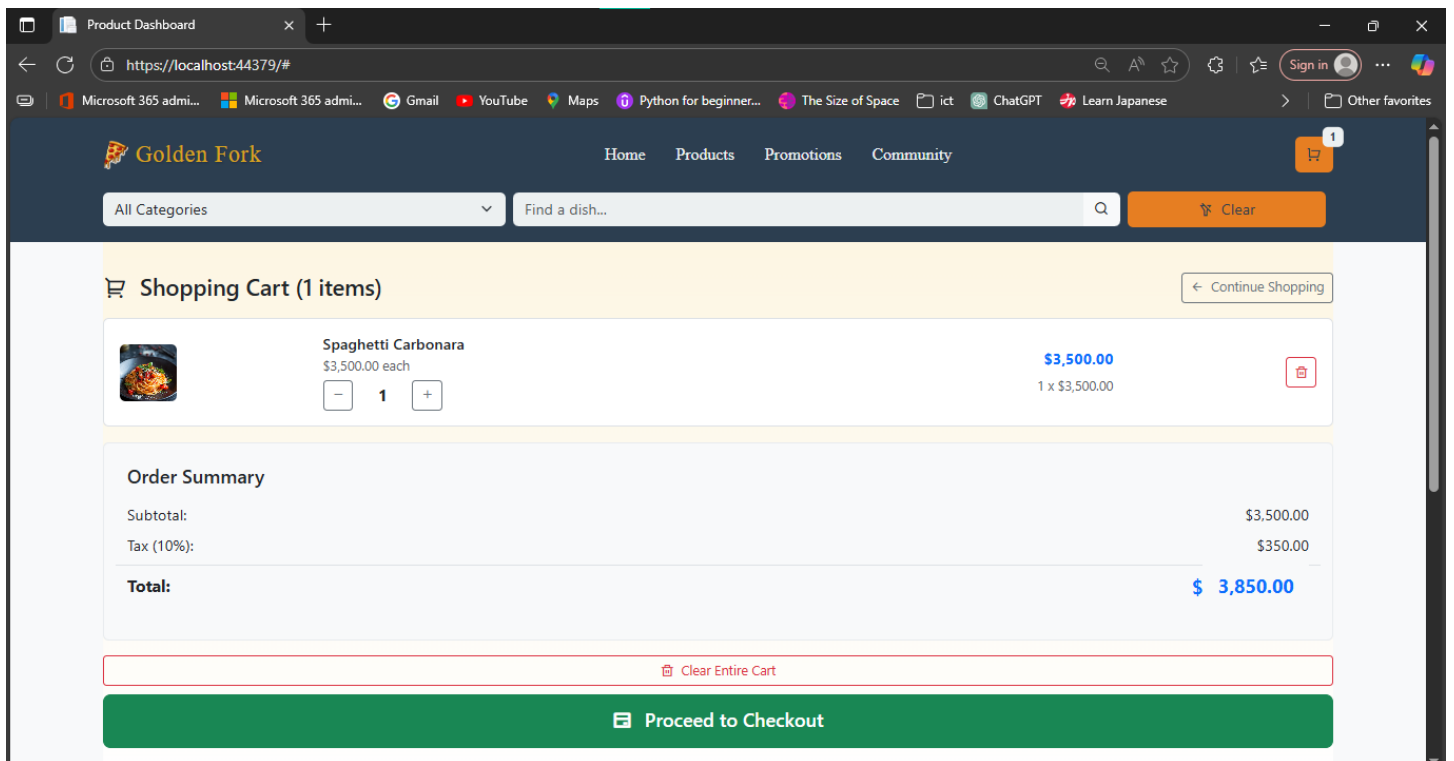
Add to Order

✕ Close

Cart


wll-mid-assignment

PAGE 15



Filters




[Home](#)
[Products](#)
[Promotions](#)
[Community](#)

1

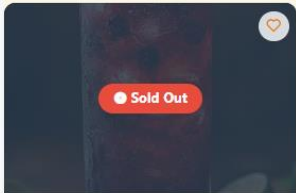
Beverages

Find a dish...

Clear

Search Results

• Category: Beverages




Iced Tea Beverages

Refreshing iced tea with lemon and mint

LKR 700.00 per serving

[View Details](#)

Currently Unavailable




Mango Smoothie Beverages

Creamy smoothie with fresh ripe mangoes

LKR 1,200.00 per serving

[View Details](#)

[Add to Order](#)




Lemonade Beverages

Refreshing homemade lemonade with a zesty twist

LKR 650.00 per serving

[View Details](#)

[Add to Order](#)




Fresh Orange Juice Beverages

Freshly squeezed orange juice, served chilled

LKR 750.00 per serving

[View Details](#)

[Add to Order](#)


[Home](#)
[Products](#)
[Promotions](#)
[Community](#)

1


Starters

Find a dish...

Clear

Search Results

• Category: Starters




Coleslaw Starters

Creamy shredded cabbage and carrot slaw

LKR 850.00 per serving

[View Details](#)

[Add to Order](#)




Chicken Satay Starters

Grilled skewers of marinated chicken with peanut sauce

LKR 1,550.00 per serving

[View Details](#)

[Add to Order](#)




Mozzarella Sticks Starters

Crispy fried mozzarella served with marinara sauce

LKR 1,200.00 per serving

[View Details](#)

[Add to Order](#)



Stuffed Mushrooms Starters

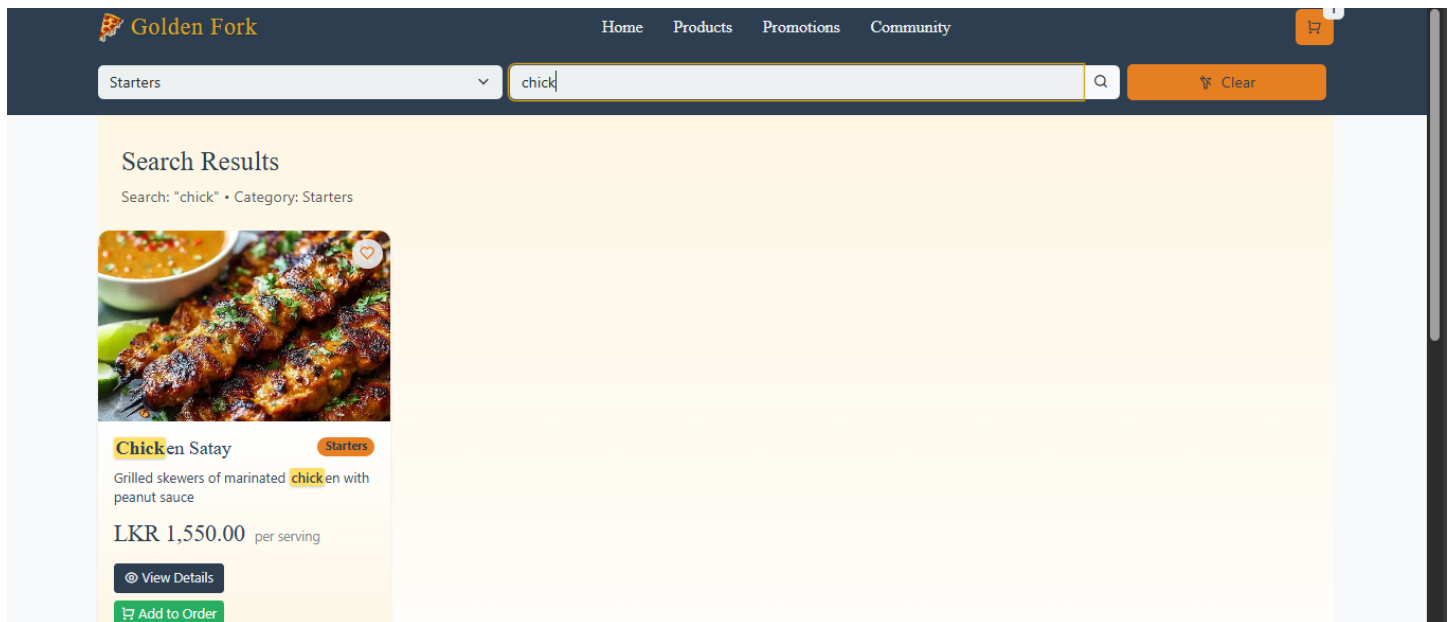
Mushrooms filled with cheese and herbs, baked golden

LKR 1,350.00 per serving

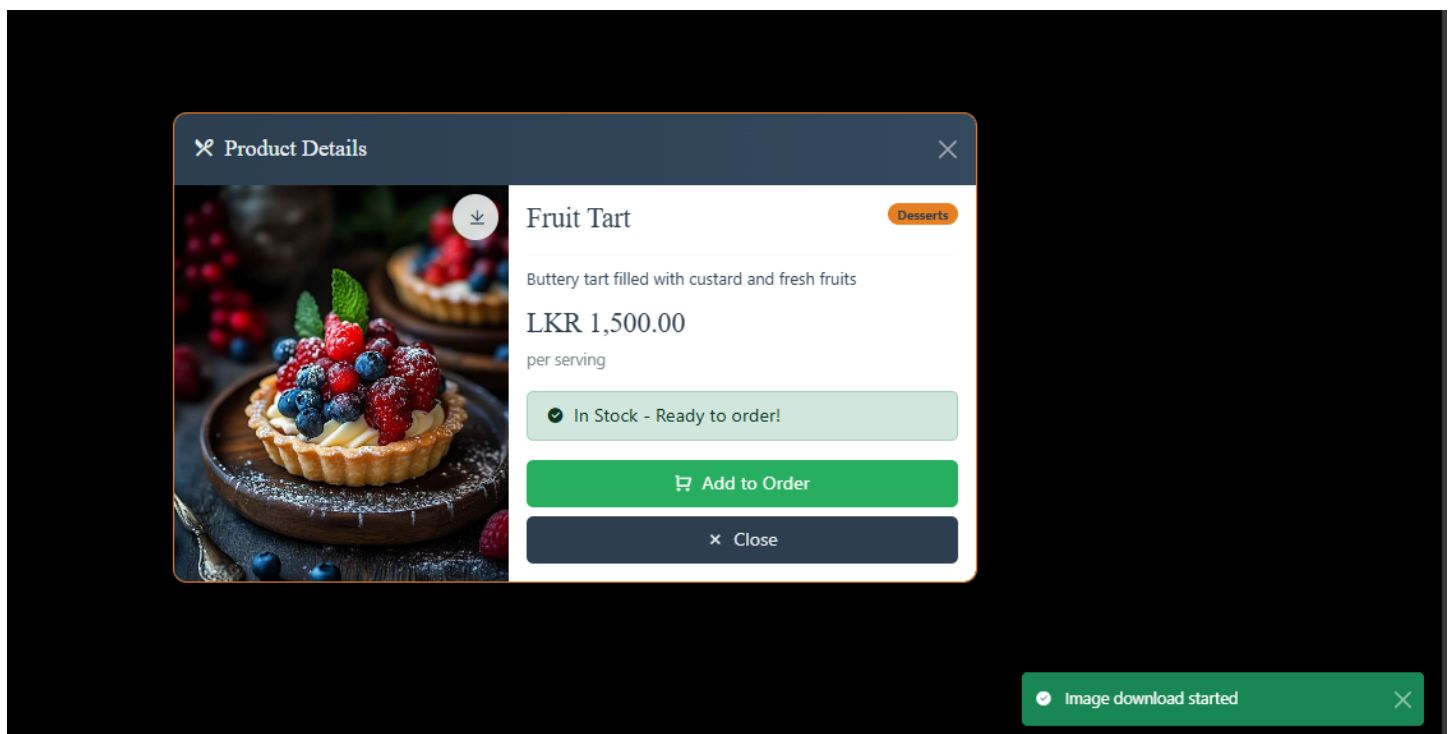
[View Details](#)

[Add to Order](#)

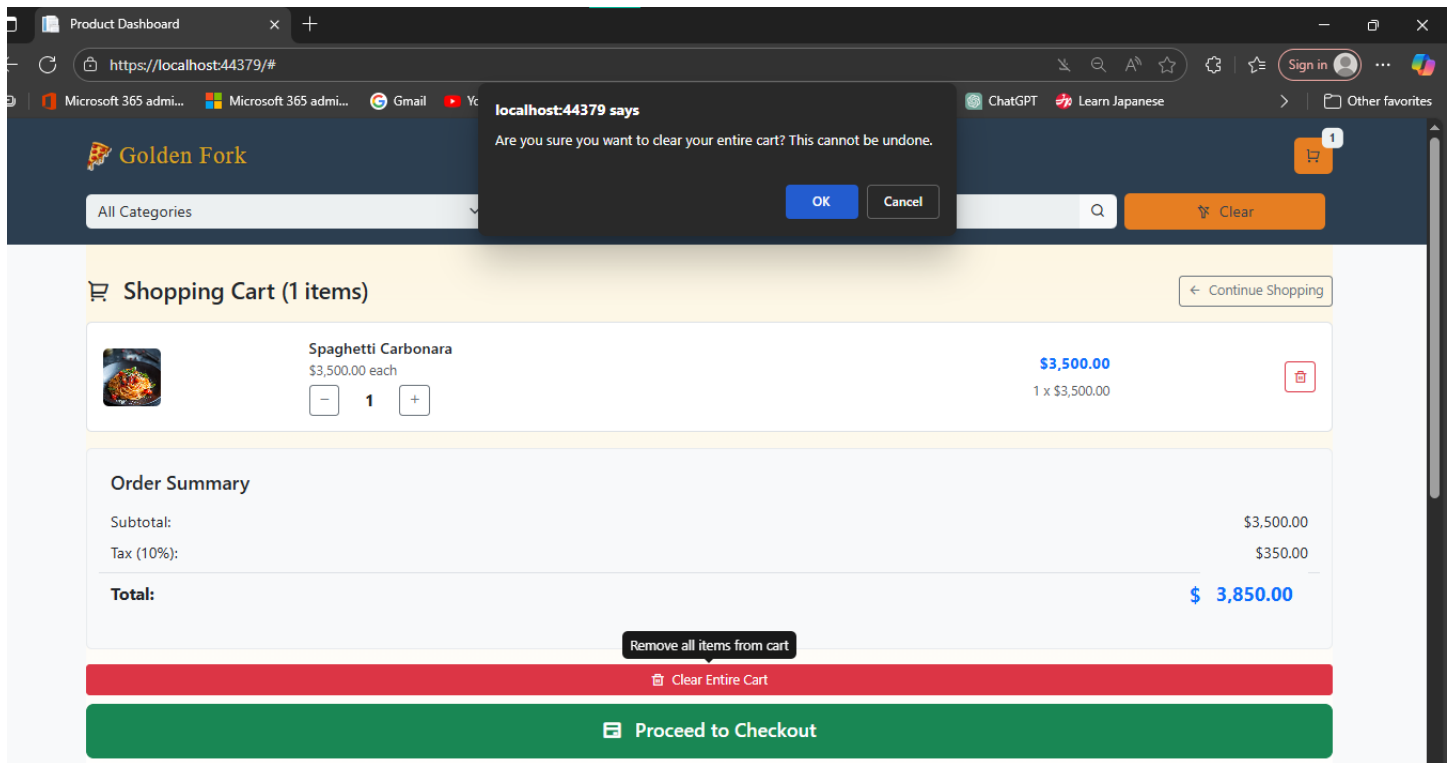
Highlight the keyword



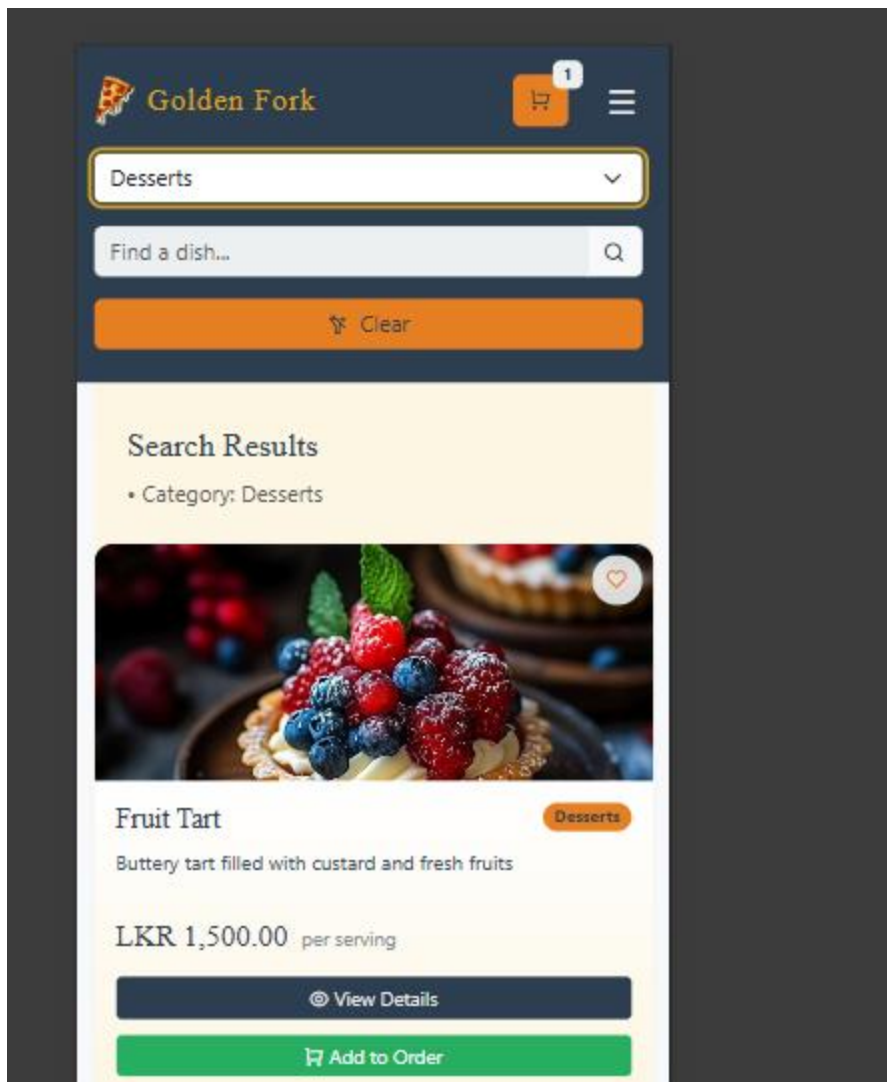
Download product image



Added tooltips



Mobile ui





Cheesecake

Desserts

Creamy New York style cheesecake with berry compote

LKR 1,600.00 per serving

View Details

Add to Order



Chocolate Mousse

Desserts

Light and airy chocolate dessert with whipped cream

LKR 1,350.00 per serving



Serving exquisite culinary experiences since 2024.
Where every meal tells a story and every flavor creates a memory.



Quick Links

- > [Home](#)
- > [Products](#)
- > [Promotions](#)
- > [Community](#)

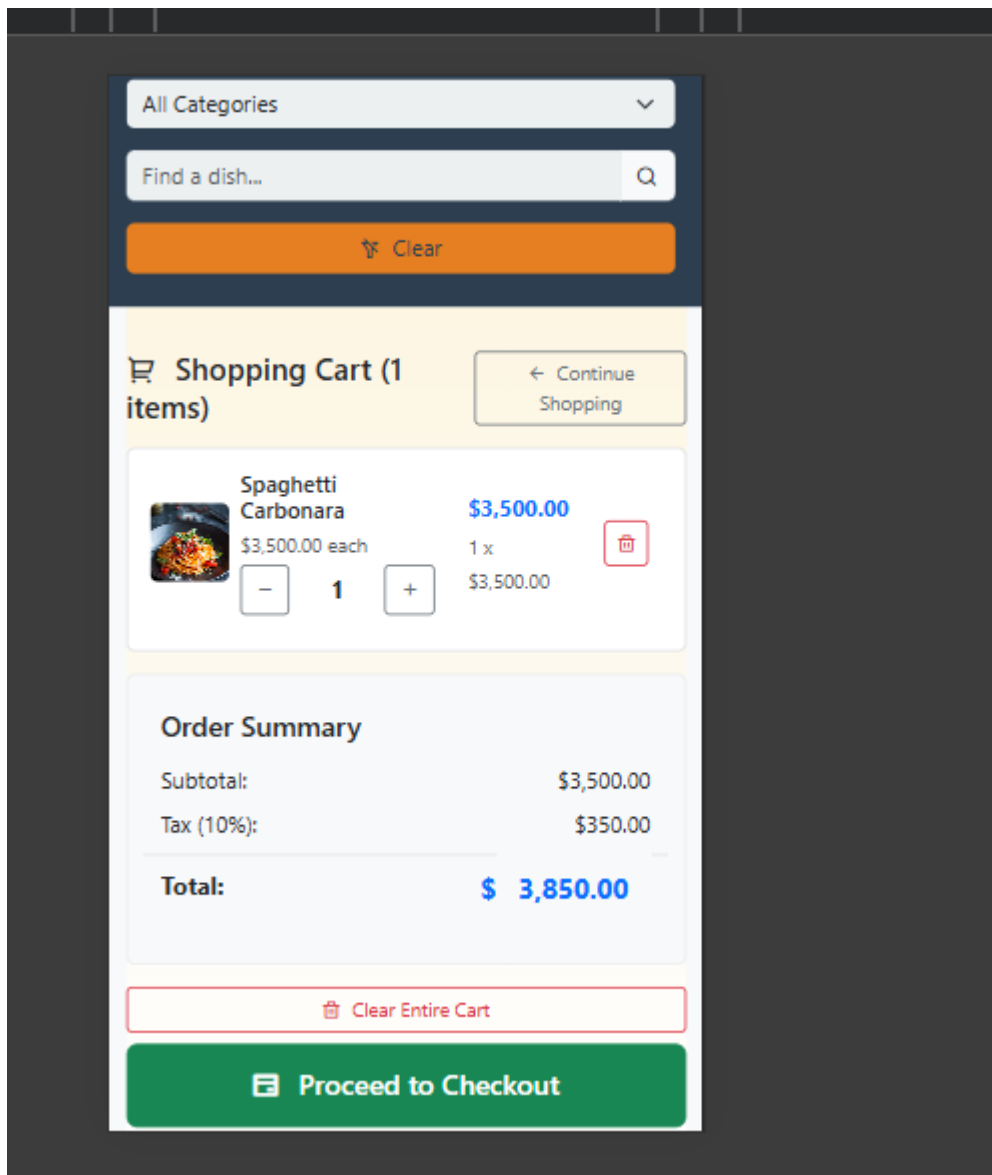
Contact Us

📍 123 Gourmet Street,
Foodie District, Colombo

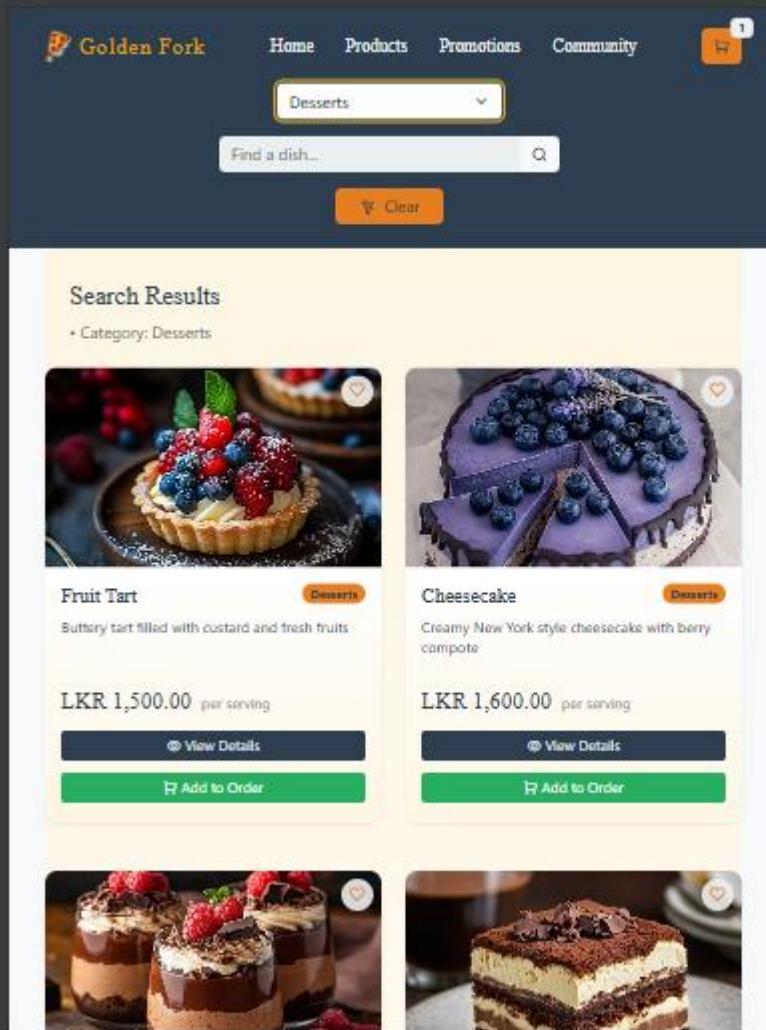
☎ +94 11 234 5678

✉ info@goldenfork.lk

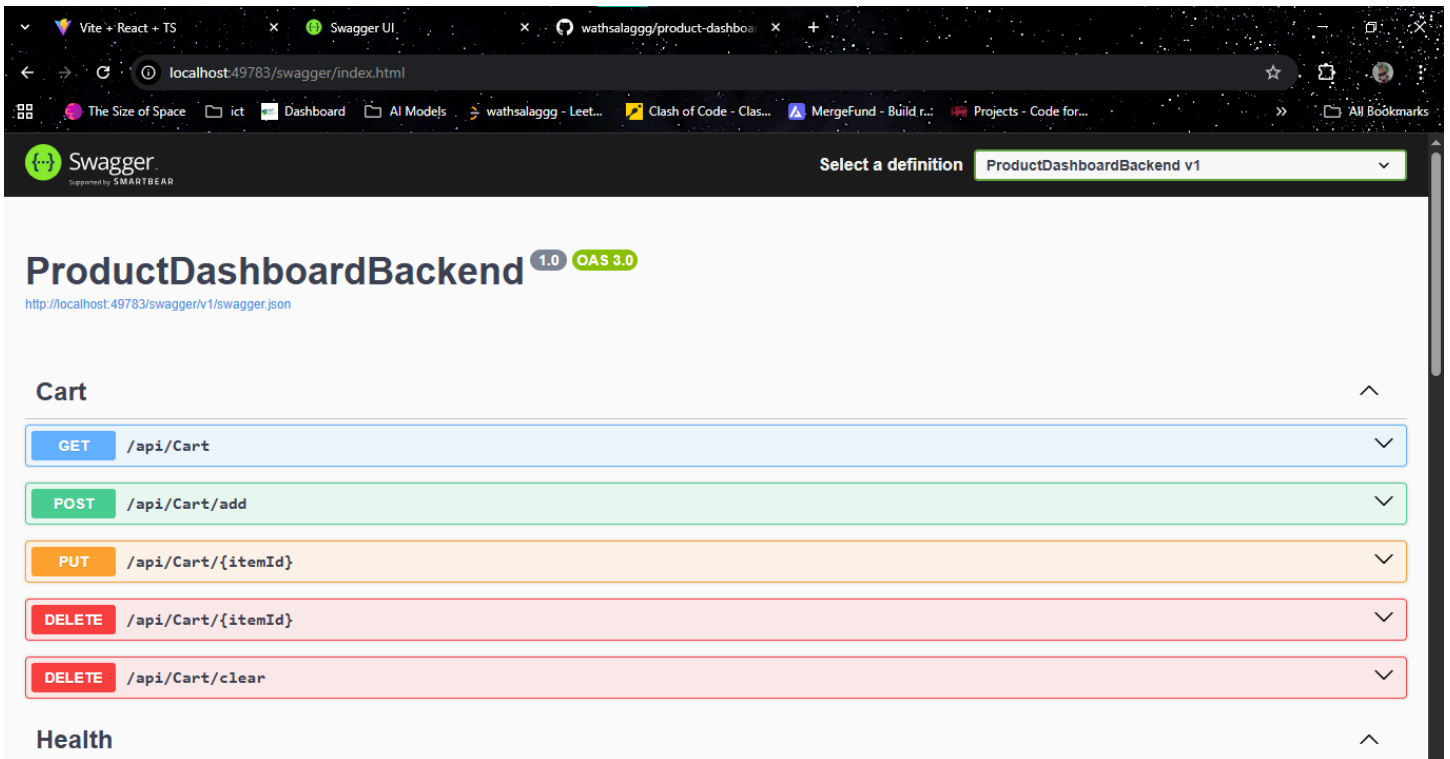
🕒 Open: 10AM - 10PM Daily



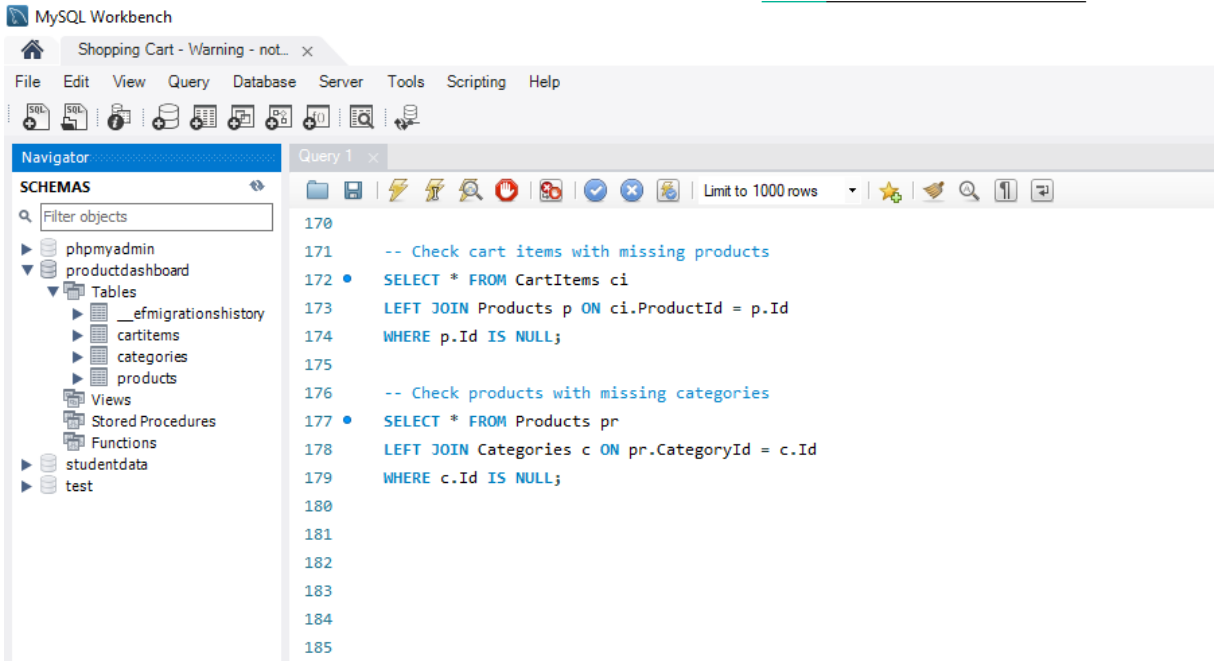
Tab ui



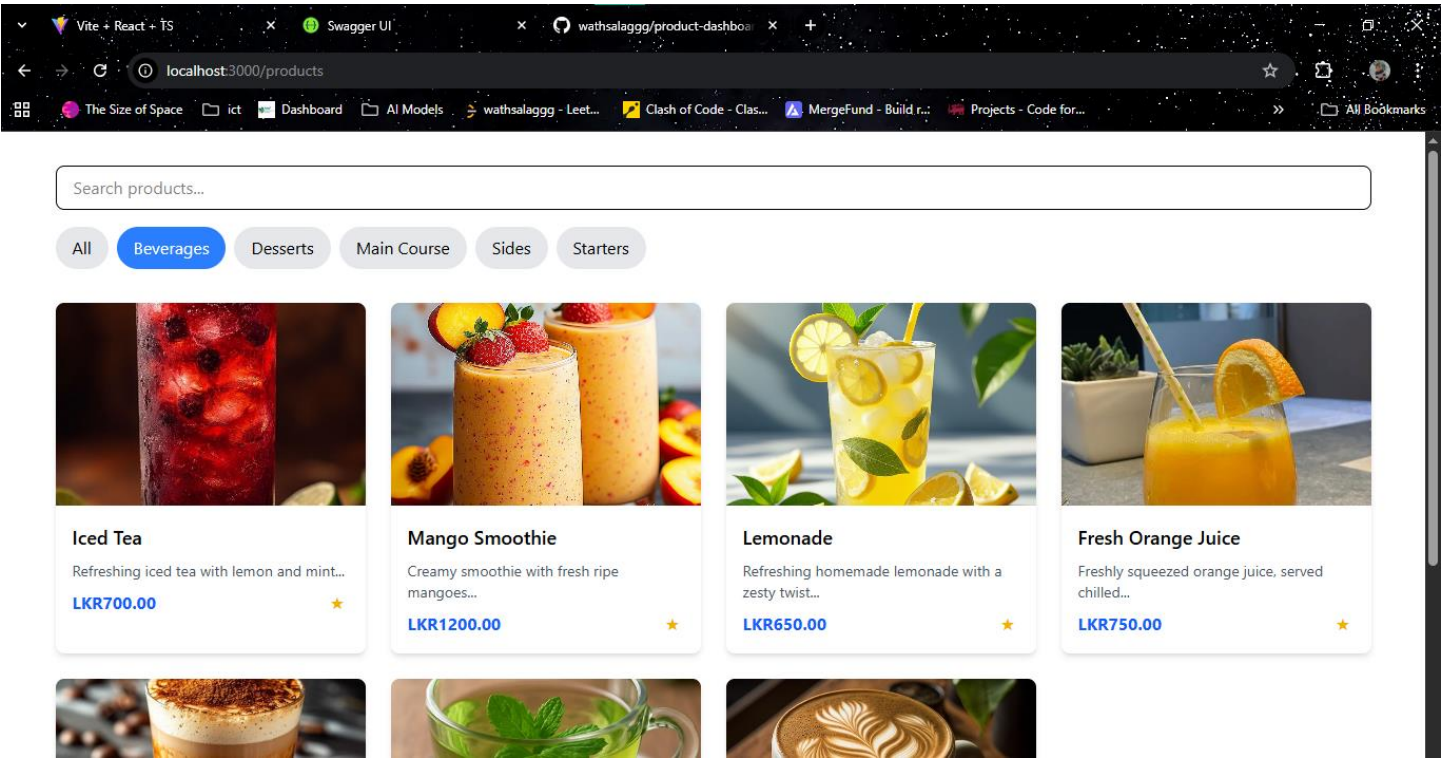
Swagger ui



DB



React ui – didn't have enough time to complete this



Github work ss

