Techathon Preli Case: Bistro 92 Ordering System

# Objective

The objective of this project is to design a smart ordering system for Bistro 92 that enhances customer satisfaction and operational efficiency. The system is intended to reduce wait times, minimize ordering errors, and streamline restaurant operations by integrating a cloud-based system with smart devices at the dining tables.

# Key Points for Solving

1. Menu Navigation: Customers can browse the menu, select items, and adjust quantities using a smart device placed on each table.

2. Order Placement: Orders are placed directly from the table, reducing wait times and improving order accuracy.

3. Cloud Integration: Orders are sent to a cloud-based system, enabling real-time updates for kitchen staff and management.

4. User-Friendly Interface: The system is designed with simple push buttons and an OLED/LCD display for ease of use by all customers.

# Coding

The core of this system is implemented using ESP32, which communicates with a cloud-based backend. The following is a simple outline of the code logic used for menu navigation, item selection, and order placement.

#include <Wire.h>

#include <Adafruit\_GFX.h>

#include <Adafruit\_SSD1306.h> // OLED display size

#define SCREEN\_WIDTH 128

#define SCREEN\_HEIGHT 64 Adafruit\_SSD1306

**display**(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, -**1**); // Button pins

#define BUTTON\_MENU 12

#define BUTTON\_SELECT 14

#define BUTTON\_UP 27

#define BUTTON\_DOWN 26 // Menu states**enum** State { MENU, QUANTITY };

State currentState = MENU;

**int** currentMenuIndex = **0**;**int** quantity = **1**;

String menuItems[] = { "Burger", "Pizza", "Soda", "Fries" };**int** totalMenuItems = **sizeof**(menuItems) / **sizeof**(menuItems[**0**]);

// Order cart**struct** Item {

String name;

**int** quantity;

};

Item cart[**10**];**int** cartIndex = **0**;

**unsigned** **long** lastPressTime = **0**;**const** **unsigned** **long** longPressDuration = **2000**; // 2 seconds

**void** **setup**() {

Serial.begin(**115200**);

pinMode(BUTTON\_MENU, INPUT\_PULLUP);

pinMode(BUTTON\_SELECT, INPUT\_PULLUP);

pinMode(BUTTON\_UP, INPUT\_PULLUP);

pinMode(BUTTON\_DOWN, INPUT\_PULLUP);

**if** (!display.begin(SSD1306\_SWITCHCAPVCC, **0x3C**)) {

Serial.println("OLED failed");

**while** (true);

}

display.clearDisplay();

display.setTextColor(SSD1306\_WHITE);

showMenu();

}

**void** **loop**() {

**if** (digitalRead(BUTTON\_MENU) == LOW) {

delay(**200**); // debounce

resetOrder();

**return**;

}

**if** (currentState == MENU) {

**if** (digitalRead(BUTTON\_UP) == LOW) {

currentMenuIndex = (currentMenuIndex - **1** + totalMenuItems) % totalMenuItems;

delay(**200**);

} **else** **if** (digitalRead(BUTTON\_DOWN) == LOW) {

currentMenuIndex = (currentMenuIndex + **1**) % totalMenuItems;

delay(**200**);

} **else** **if** (digitalRead(BUTTON\_SELECT) == LOW) {

lastPressTime = millis();

**while** (digitalRead(BUTTON\_SELECT) == LOW) {

**if** (millis() - lastPressTime > longPressDuration) {

finalizeOrder();

**return**;

}

}

currentState = QUANTITY;

}

showMenu();

}

**else** **if** (currentState == QUANTITY) {

**if** (digitalRead(BUTTON\_UP) == LOW) {

quantity++;

delay(**200**);

} **else** **if** (digitalRead(BUTTON\_DOWN) == LOW) {

quantity = max(**1**, quantity - **1**);

delay(**200**);

} **else** **if** (digitalRead(BUTTON\_SELECT) == LOW) {

cart[cartIndex++] = { menuItems[currentMenuIndex], quantity };

quantity = **1**;

currentState = MENU;

}

showQuantityDialog();

}

}

// --- Display Functions ---

**void** **showMenu**() {

display.clearDisplay();

display.setCursor(**0**, **0**);

display.setTextSize(**1**);

**for** (**int** i = **0**; i < totalMenuItems; i++) {

**if** (i == currentMenuIndex) display.print("> ");

**else** display.print(" ");

display.println(menuItems[i]);

}

display.display();

}

**void** **showQuantityDialog**() {

display.clearDisplay();

display.setCursor(**0**, **0**);

display.setTextSize(**1**);

display.println("Set Quantity:");

display.println(menuItems[currentMenuIndex]);

display.print("Qty: ");

display.println(quantity);

display.display();

}

**void** **finalizeOrder**() {

display.clearDisplay();

display.setCursor(**0**, **0**);

display.setTextSize(**1**);

display.println("Order Summary:");

**for** (**int** i = **0**; i < cartIndex; i++) {

display.print(cart[i].name);

display.print(" x");

display.println(cart[i].quantity);

}

display.println("Order Sent!");

display.display();

delay(**3000**);

resetOrder();

}

**void** **resetOrder**() {

cartIndex = **0**;

quantity = **1**;

currentMenuIndex = **0**;

currentState = MENU;

showMenu();

}

# Circuit Connection

The hardware setup includes the following components:

1. ESP32: Central processing unit controlling the system.

2. OLED/LCD Display: For showing the menu and order status.

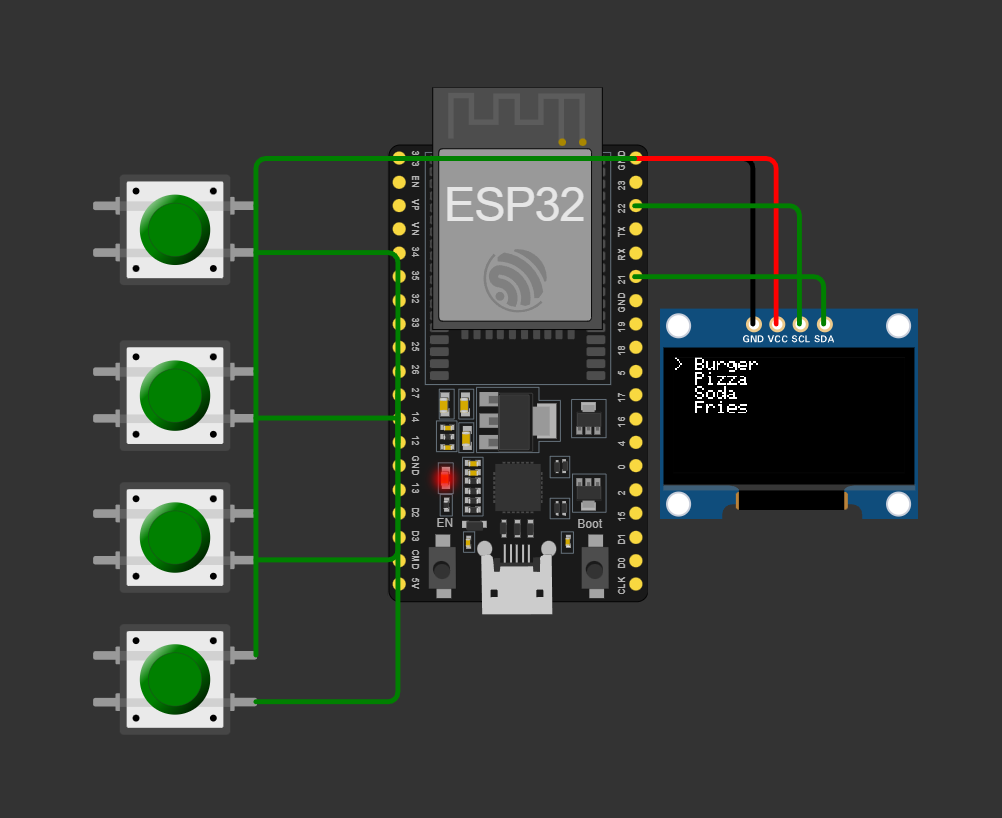
3. Push Buttons: For navigating the menu, selecting items, adjusting quantity, and placing orders.

The connections between components are as follows:

1. Push buttons are connected to GPIO pins on the ESP32.

2. The OLED/LCD display is connected via I2C protocol for simple communication.

3. The ESP32 sends order data to the cloud-based server using WiFi.



# Simulation File

<https://wokwi.com/projects/429328812549701633>

**Quick Fixes for Bistro 92**

**Q1: Three Essential Features for Customer Satisfaction and Efficient Order Processing**

1. **Real-Time Order Updates:**  
   Customers and kitchen staff must instantly see order confirmations and status updates to avoid delays and confusion.
2. **Easy and Fast Menu Navigation:**  
   Customers should be able to browse and select items quickly using intuitive button controls, ensuring smooth flow even for tech novices.
3. **Error-Free Order Submission:**  
   The system must prevent incomplete or wrong orders with clear quantity confirmation steps and simple options for canceling/resetting an order.

**Q2: Two Design Principles for an Intuitive Smart Pad Interface**

1. **Simplicity and Minimal Steps:**  
   Minimize the number of actions needed to browse, select, and order items. Only show essential information at each step to avoid overwhelming the user.
2. **Consistent Button Behavior:**  
   Keep button functions uniform across screens (e.g., Button 2 always confirms, Button 3 always scrolls up, Button 4 always scrolls down) so users can quickly learn by intuition.

**Q3: Three Potential Security Vulnerabilities and Solutions**

| **Vulnerability** | **Solution** |
| --- | --- |
| **Order Tampering** | Use encrypted communication (e.g., HTTPS/MQTT with SSL) between the smart pad and cloud server. |
| **Device Theft or Misuse** | Implement device authentication and tracking, like device ID checks before accepting orders. |
| **Unauthorized System Access** | Require admin login for sensitive areas like dashboard settings; add timeout-based auto-logout. |

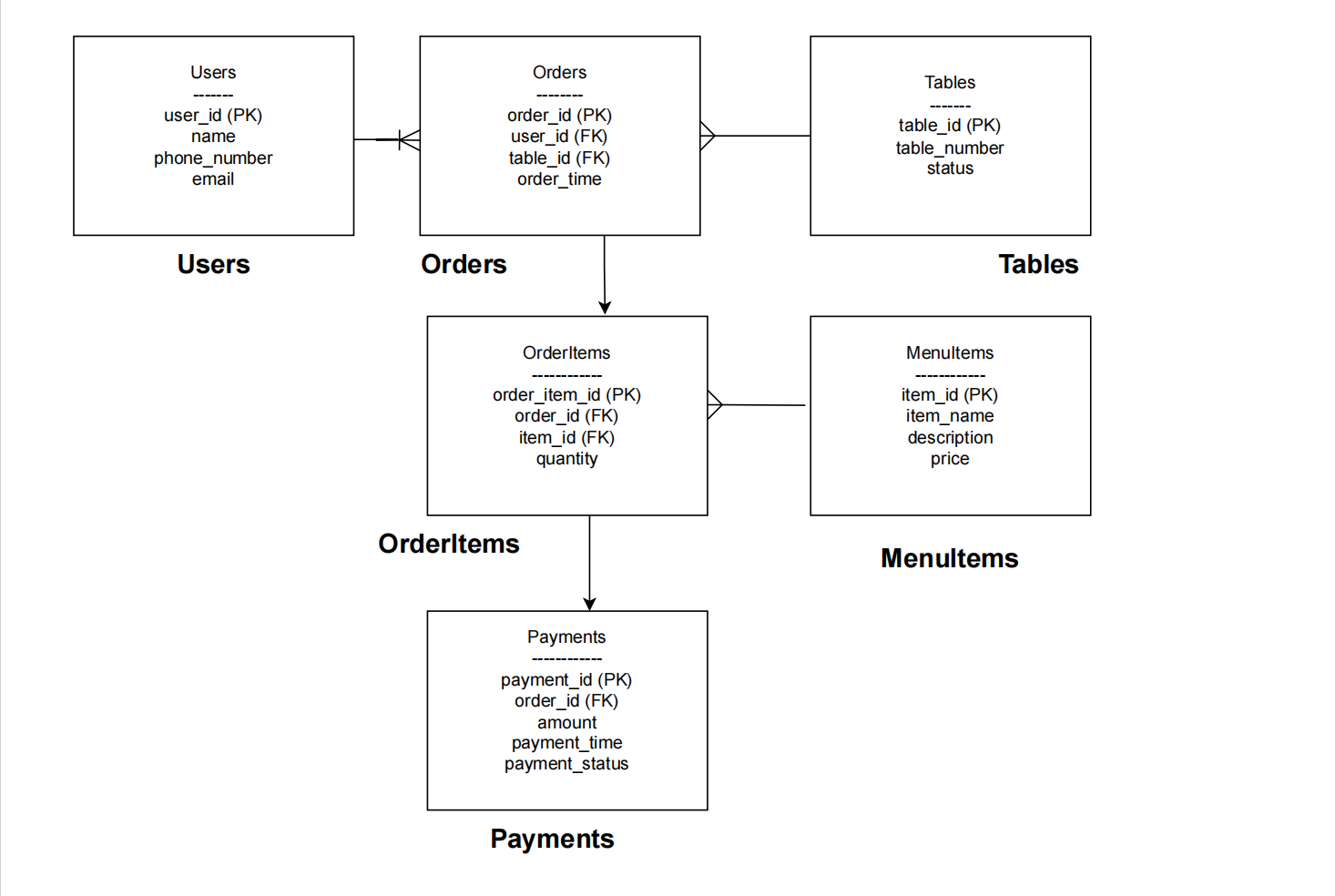
**Q4: Two Strategies to Keep System Responsive and Stable During Peak Hours**

1. **Load Balancing:**  
   Distribute incoming orders and dashboard requests across multiple cloud servers or serverless functions to avoid overload on a single system.
2. **Efficient Database Indexing:**  
   Use indexed queries (especially on fields like order time, table number) to retrieve and update records swiftly without database slowdowns.

**Q5 : One Method to Integrate the Existing Inventory System with the New System Without Disrupting Operations**

* **Use an API Bridge Approach:**  
  Develop a middleware/API layer that reads and updates inventory stock in real-time without touching the old system directly. This ensures the new order system fetches item availability live while the old inventory software continues operating independently in the background. Gradually, full migration can happen with no downtime.

**DataBase:**



**Firebase SDK:**

// Import the functions you need from the SDKs you need

import { initializeApp } from "firebase/app";

import { getAnalytics } from "firebase/analytics";

// TODO: Add SDKs for Firebase products that you want to use

// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration

// For Firebase JS SDK v7.20.0 and later, measurementId is optional

const firebaseConfig = {

apiKey: "AIzaSyCcCLXcgYyxceqmHPlvMeUDLksWe4JxtMw",

authDomain: "restaurant-ordering-e4bfd.firebaseapp.com",

projectId: "restaurant-ordering-e4bfd",

storageBucket: "restaurant-ordering-e4bfd.firebasestorage.app",

messagingSenderId: "920226852042",

appId: "1:920226852042:web:245fb994e850fe3e279833",

measurementId: "G-81ZJ34NCEF"

};

// Initialize Firebase

const app = initializeApp(firebaseConfig);

const analytics = getAnalytics(app);

# Conclusion

This solution provides an innovative approach to improving the dining experience at Bistro 92. By incorporating a smart ordering system, the restaurant can enhance customer satisfaction, streamline order processing, and improve overall efficiency. The system is easy to use, cost-effective, and scalable, offering a high potential for real-world application.