# SAMPLE CODING

**ARDUINO Program**

#include <ESP8266WiFi.h>

#include <ESP8266HTTPClient.h>

#include <ArduinoJson.h>

#include <SPI.h>

#include <MFRC522.h>

#define RST\_PIN D3

#define SS\_PIN D4

#define TRIG\_PIN D5

#define ECHO\_PIN D6

#define BUZZER\_PIN D7

const char\* ssid = "YOUR\_SSID";

const char\* password = "YOUR\_WIFI\_PASSWORD";

const char\* supabaseUrl = "https://your-project.supabase.co/rest/v1";

const char\* supabaseKey = "YOUR\_SUPABASE\_ANON\_OR\_SERVICE\_ROLE\_KEY";

// Your table names

const char\* inventoryTable = "inventory";

const char\* rfidLogsTable = "rfid\_logs";

MFRC522 mfrc522(SS\_PIN, RST\_PIN);

int inventory = 5;

const int maxStock = 5;

bool alerted = false;

bool restocking = false;

long distance;

unsigned long lastPenTime = 0;

bool firstPenTaken = false;

WiFiClientSecure client; // for HTTPS requests

void setup() {

Serial.begin(115200);

WiFi.begin(ssid, password);

Serial.print("Connecting to WiFi");

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("\nWiFi connected");

// Setup pins

pinMode(TRIG\_PIN, OUTPUT);

pinMode(ECHO\_PIN, INPUT);

pinMode(BUZZER\_PIN, OUTPUT);

digitalWrite(BUZZER\_PIN, LOW);

SPI.begin();

mfrc522.PCD\_Init();

client.setInsecure(); // Disable certificate verification (not secure for prod!)

uploadInventory(inventory); // Initial upload

}

void loop() {

// Ultrasonic sensor reading

digitalWrite(TRIG\_PIN, LOW);

delayMicroseconds(2);

digitalWrite(TRIG\_PIN, HIGH);

delayMicroseconds(10);

digitalWrite(TRIG\_PIN, LOW);

distance = pulseIn(ECHO\_PIN, HIGH) \* 0.034 / 2;

if (distance < 20 && !restocking) {

unsigned long currentMillis = millis();

if (!firstPenTaken) {

delay(4000);

inventory--;

Serial.println("Pen taken");

firstPenTaken = true;

lastPenTime = currentMillis;

uploadInventory(inventory);

} else if (currentMillis - lastPenTime >= 5000) {

if (inventory > 0) {

inventory--;

Serial.println("Pen taken");

lastPenTime = currentMillis;

uploadInventory(inventory);

}

}

if (inventory == 1 && !alerted) {

beepBuzzer(5000);

alerted = true;

}

if (inventory == 0) {

beepBuzzer(5000);

restocking = true;

alerted = false;

}

delay(1000);

}

if (restocking) {

inventory = maxStock;

Serial.println("Restocked");

uploadInventory(inventory);

restocking = false;

firstPenTaken = false;

}

checkRFID();

}

void checkRFID() {

if (mfrc522.PICC\_IsNewCardPresent() && mfrc522.PICC\_ReadCardSerial()) {

String uid = "";

for (byte i = 0; i < mfrc522.uid.size; i++) {

if (mfrc522.uid.uidByte[i] < 0x10) uid += "0"; // pad leading 0

uid += String(mfrc522.uid.uidByte[i], HEX);

}

uid.toUpperCase();

Serial.print("RFID UID: ");

Serial.println(uid);

logRFID(uid);

mfrc522.PICC\_HaltA();

}

}

void uploadInventory(int count) {

if (WiFi.status() == WL\_CONNECTED) {

HTTPClient https;

String url = String(supabaseUrl) + "/" + inventoryTable + "?id=eq.1";

// Build JSON body for PATCH (update where id=1)

String jsonBody = "{\"count\":";

jsonBody += String(count);

jsonBody += "}";

https.begin(client, url);

https.addHeader("apikey", supabaseKey);

https.addHeader("Authorization", "Bearer " + String(supabaseKey));

https.addHeader("Content-Type", "application/json");

int httpCode = https.PATCH(jsonBody);

if (httpCode > 0) {

String payload = https.getString();

Serial.print("Inventory updated: ");

Serial.println(payload);

} else {

Serial.print("Error updating inventory: ");

Serial.println(httpCode);

}

https.end();

}

}

void logRFID(String uid) {

if (WiFi.status() == WL\_CONNECTED) {

HTTPClient https;

String url = String(supabaseUrl) + "/" + rfidLogsTable;

// Build JSON body for POST (insert)

String jsonBody = "{\"uid\":\"" + uid + "\", \"timestamp\": \"now()\"}";

https.begin(client, url);

https.addHeader("apikey", supabaseKey);

https.addHeader("Authorization", "Bearer " + String(supabaseKey));

https.addHeader("Content-Type", "application/json");

int httpCode = https.POST(jsonBody);

if (httpCode > 0) {

String payload = https.getString();

Serial.print("RFID logged: ");

Serial.println(payload);

} else {

Serial.print("Error logging RFID: ");

Serial.println(httpCode);

}

https.end();

}

}

void beepBuzzer(int durationMs) {

digitalWrite(BUZZER\_PIN, HIGH);

delay(durationMs);

digitalWrite(BUZZER\_PIN, LOW);

}

**Web Application**

# App.tsx

import { Toaster } from "@/components/ui/toaster";

import { Toaster as Sonner } from "@/components/ui/sonner";

import { TooltipProvider } from "@/components/ui/tooltip";

import { QueryClient, QueryClientProvider } from "@tanstack/react-query";

import { BrowserRouter, Routes, Route } from "react-router-dom";

import { InventoryProvider } from "./context/InventoryContext";

import Layout from "./components/layout/Layout";

import Dashboard from "./pages/Dashboard";

import Products from "./pages/Products";

import AddProduct from "./pages/AddProduct";

import EditProduct from "./pages/EditProduct";

import Scanner from "./pages/Scanner";

import Bills from "./pages/Bills";

import BillDetail from "./pages/BillDetail";

import NotFound from "./pages/NotFound";

const queryClient = new QueryClient();

const App = () => (

<QueryClientProvider client={queryClient}>

<TooltipProvider>

<InventoryProvider>

<Toaster />

<Sonner />

<BrowserRouter>

<Layout>

<Routes>

<Route path="/" element={<Dashboard />} />

<Route path="/products" element={<Products />} />

<Route path="/products/add" element={<AddProduct />} />

<Route path="/products/edit/:id" element={<EditProduct />} />

<Route path="/scanner" element={<Scanner />} />

<Route path="/bills" element={<Bills />} />

<Route path="/bills/:id" element={<BillDetail />} />

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

</Routes>

</Layout>

</BrowserRouter>

</InventoryProvider>

</TooltipProvider>

</QueryClientProvider>

);

export default App;

# Dashboard.jsx

import React from 'react';

import { useNavigate } from 'react-router-dom';

import { useInventory } from '@/context/InventoryContext';

import { Card, CardContent, CardDescription, CardHeader, CardTitle } from '@/components/ui/card';

import { Button } from '@/components/ui/button';

import { BarChart, Bar, XAxis, YAxis, CartesianGrid, Tooltip, ResponsiveContainer } from 'recharts';

import { AlertCircle, PackagePlus, FileText, Package, ArrowUp, ArrowDown } from 'lucide-react';

const Dashboard = () => {

const { products, alerts, stockMovements, bills } = useInventory();

const navigate = useNavigate();

// Calculate stats

const totalProducts = products.length;

const totalValue = products.reduce((sum, product) => sum + (product.price \* product.quantity), 0);

const lowStockCount = products.filter(p => p.quantity <= p.threshold).length;

const outOfStockCount = products.filter(p => p.quantity === 0).length;

const recentSales = bills.slice(0, 5);

// Prepare chart data

const stockData = products

.filter(p => p.quantity > 0)

.slice(0, 6)

.map(p => ({

name: p.name.length > 12 ? p.name.substring(0, 12) + '...' : p.name,

quantity: p.quantity,

threshold: p.threshold,

}));

// Recent stock movement stats

const incomingStock = stockMovements

.filter(m => m.type === 'in')

.reduce((sum, m) => sum + m.quantity, 0);

const outgoingStock = stockMovements

.filter(m => m.type === 'out')

.reduce((sum, m) => sum + m.quantity, 0);

return (

<div className="space-y-8 animate-fade-in">

<div className="flex justify-between items-center">

<h1 className="text-3xl font-bold text-gray-800">Dashboard</h1>

<div className="space-x-2">

<Button onClick={() => navigate('/products/add')} className="bg-inventory-blue hover:bg-inventory-darkBlue">

<PackagePlus className="h-4 w-4 mr-2" /> Add Product

</Button>

<Button onClick={() => navigate('/scanner')} variant="outline">

Scan Products

</Button>

</div>

</div>

{/\* Stats Overview \*/}

<div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-6">

<Card className="bg-white">

<CardHeader className="pb-2">

<CardTitle className="text-lg">Total Products</CardTitle>

</CardHeader>

<CardContent>

<div className="flex justify-between items-center">

<span className="text-3xl font-bold">{totalProducts}</span>

<Package className="h-8 w-8 text-inventory-blue" />

</div>

</CardContent>

</Card>

<Card className="bg-white">

<CardHeader className="pb-2">

<CardTitle className="text-lg">Inventory Value</CardTitle>

</CardHeader>

<CardContent>

<div className="flex justify-between items-center">

<span className="text-3xl font-bold">₹{totalValue.toFixed(2)}</span>

<FileText className="h-8 w-8 text-inventory-green" />

</div>

</CardContent>

</Card>

<Card className="bg-white">

<CardHeader className="pb-2">

<CardTitle className="text-lg">Low Stock Items</CardTitle>

</CardHeader>

<CardContent>

<div className="flex justify-between items-center">

<span className="text-3xl font-bold">{lowStockCount}</span>

<AlertCircle className="h-8 w-8 text-inventory-orange" />

</div>

<p className="text-sm mt-2 text-gray-500">

Including {outOfStockCount} out-of-stock items

</p>

</CardContent>

</Card>

<Card className="bg-white">

<CardHeader className="pb-2">

<CardTitle className="text-lg">Stock Movement</CardTitle>

</CardHeader>

<CardContent>

<div className="grid grid-cols-2 gap-4">

<div className="flex items-center">

<ArrowUp className="h-5 w-5 mr-2 text-inventory-green" />

<div>

<p className="text-sm text-gray-500">In</p>

<p className="font-semibold">{incomingStock}</p>

</div>

</div>

<div className="flex items-center">

<ArrowDown className="h-5 w-5 mr-2 text-inventory-red" />

<div>

<p className="text-sm text-gray-500">Out</p>

<p className="font-semibold">{outgoingStock}</p>

</div>

</div>

</div>

</CardContent>

</Card>

</div>

{/\* Stock Level Chart \*/}

<div className="grid grid-cols-1 lg:grid-cols-2 gap-6">

<Card className="bg-white">

<CardHeader>

<CardTitle>Current Stock Levels</CardTitle>

<CardDescription>Stock quantity vs threshold for top products</CardDescription>

</CardHeader>

<CardContent>

<div className="h-[300px]">

<ResponsiveContainer width="100%" height="100%">

<BarChart

data={stockData}

margin={{ top: 10, right: 10, left: 0, bottom: 40 }}

barSize={20}

>

<CartesianGrid strokeDasharray="3 3" />

<XAxis

dataKey="name"

angle={-45}

textAnchor="end"

height={70}

/>

<YAxis />

<Tooltip />

<Bar dataKey="quantity" fill="#0EA5E9" name="Current Stock" />

<Bar dataKey="threshold" fill="#DC2626" name="Threshold" />

</BarChart>

</ResponsiveContainer>

</div>

</CardContent>

</Card>

{/\* Recent Alerts \*/}

<Card className="bg-white">

<CardHeader className="border-b">

<CardTitle>Recent Alerts</CardTitle>

<CardDescription>Latest inventory alerts and notifications</CardDescription>

</CardHeader>

<CardContent className="divide-y">

{alerts.length > 0 ? (

alerts.slice(0, 5).map((alert) => (

<div key={alert.id} className={`py-3 ${!alert.read ? 'bg-blue-50' : ''}`}>

<div className="flex justify-between">

<div className="flex items-center">

{alert.type === 'low-stock' && (

<div className="h-2.5 w-2.5 rounded-full bg-yellow-500 mr-2" />

)}

{alert.type === 'out-of-stock' && (

<div className="h-2.5 w-2.5 rounded-full bg-red-500 mr-2" />

)}

{alert.type === 'restock' && (

<div className="h-2.5 w-2.5 rounded-full bg-green-500 mr-2" />

)}

<span className="text-sm">{alert.message}</span>

</div>

</div>

<p className="text-xs text-gray-500 mt-1">

{new Date(alert.timestamp).toLocaleString()}

</p>

</div>

))

) : (

<div className="py-4 text-center text-gray-500">No recent alerts</div>

)}

{alerts.length > 5 && (

<div className="pt-3">

<Button

variant="ghost"

className="w-full text-inventory-blue hover:text-inventory-darkBlue"

onClick={() => navigate('/alerts')}

>

View all alerts

</Button>

</div>

)}

</CardContent>

</Card>

</div>

{/\* Recent Activity \*/}

<Card className="bg-white">

<CardHeader className="border-b">

<CardTitle>Recent Sales</CardTitle>

<CardDescription>Latest transactions and bill generation</CardDescription>

</CardHeader>

<CardContent>

<div className="divide-y">

{recentSales.length > 0 ? (

recentSales.map((bill) => (

<div key={bill.id} className="py-4">

<div className="flex justify-between items-start">

<div>

<p className="font-medium">

Bill #{bill.id.substring(0, 8)}

{bill.customerName && ` - ₹{bill.customerName}`}

</p>

<p className="text-sm text-gray-500">

{new Date(bill.timestamp).toLocaleString()}

</p>

<p className="text-sm mt-1">

{bill.items.length} {bill.items.length === 1 ? 'item' : 'items'}

</p>

</div>

<div className="text-right">

<p className="font-bold">₹{bill.total.toFixed(2)}</p>

<Button

variant="outline"

size="sm"

className="mt-1"

onClick={() => navigate(`/bills/₹{bill.id}`)}

>

View Bill

</Button>

</div>

</div>

</div>

))

) : (

<div className="py-4 text-center text-gray-500">No recent sales</div>

)}

</div>

{bills.length > 5 && (

<div className="mt-4">

<Button

variant="outline"

className="w-full"

onClick={() => navigate('/bills')}

>

View All Bills

</Button>

</div>

)}

</CardContent>

</Card>

</div>

);

};

export default Dashboard;