CHAPTER 5: IMPLEMENTATION AND TESTING

# 5.1 INTRODUCTION

This chapter provides a comprehensive overview of the implementation and testing processes carried out for the Electronic Shelf Label (ESL) system integrated with an inventory management platform. The system includes ESP32-based devices with OLED displays, connected via Wi-Fi and controlled through a web interface. Inventory data is managed and authenticated using Supabase.

# 5.2 CODING AND CONSTRUCTION

The ESL system was implemented using Arduino C++ for the ESP32 microcontroller. The ESPAsyncWebServer library enables a dynamic HTTP server to be hosted directly on the ESP32. OLED displays are controlled via the Adafruit SSD1306 library. Inventory authentication and data synchronization are handled through Supabase, leveraging its RESTful API and user management features.

The inventory system was implemented useing Taiwind CSS (front end), python (back end) and Supabase (database and authentication.

# 5.3 TESTING

The following functional tests were conducted:  
- Wi-Fi Connection Test: Ensured ESP32 connected to hotspot within 15 seconds.  
- Display Test: Verified correct rendering of product name, price, and expiry on OLED.  
- Form Input Test: Submitted data via web interface and confirmed updates on display.  
- Supabase Auth Test: Verified token-based secure access to inventory records.  
- Failover Simulation: Tested OLED and serial outputs during no Wi-Fi scenario.

# 5.4 SECURITY

Security mechanisms include:  
- WPA2 Wi-Fi encryption for network access.  
- Supabase authentication via secure JWT tokens.  
- Input sanitation in the ESP32 web handler.  
- Restricted database rules for inventory updates.

# 5.5 INSTALLATION

Installation steps involved uploading firmware to ESP32 using Arduino IDE, provisioning Wi-Fi credentials, and deploying the Supabase project. OLED modules were soldered and mounted, and test product data was loaded.

# 5.6 TRAINING

Training was offered on:  
- Updating labels through browser interface.  
- Viewing logs via serial monitor.  
- Administering Supabase users and access roles.

# 5.7 MAINTENANCE

Maintenance guidelines:  
- Monitor Wi-Fi signal strength.  
- Refresh ESP32 firmware periodically.  
- Review Supabase logs for unauthorized attempts.  
- Clean OLED displays to prevent ghosting.

# 5.8 SYSTEM EVALUATION

System evaluation showed high performance and responsiveness. The asynchronous web server and lightweight form handling resulted in smooth updates with minimal overhead. OLED visuals were crisp and readable.

# 5.9 FILE CONVERSION AND SYSTEM CHANGEOVER

Inventory CSV data was converted and inserted into Supabase via its spreadsheet-like interface. Legacy manual labels were phased out gradually as ESLs were deployed in segments.

# 5.10 SYSTEM REVIEW

Periodic reviews verified that the solution met user expectations for ease of updates, display clarity, and secure data access. Issues such as hotspot instability were identified for future refinement.

# 5.11 RECOMMENDATIONS

- Upgrade to e-ink displays for lower power draw.  
- Use SSL for web server communication.  
- Add barcode/NFC integration for in-store updates.  
- Deploy battery monitoring feature on ESP32.

# 5.12 CONCLUSION

The system successfully demonstrates the feasibility of digital shelf labeling using IoT components. Integration with Supabase enhances inventory visibility, and the project sets a foundation for scalable deployment.

APPENDIX: USER MANUAL

The manual includes steps for connecting to Wi-Fi, submitting product updates, interpreting display output, and accessing Supabase logs.

APPENDIX A

1. Sample Interview Questions  
- What challenges do you face updating paper labels?  
2. Sample Observation Sheets  
- Note latency during label update cycle.  
3. Sample Questionnaire Questions  
- Rate the clarity of OLED display from 1–5.

APPENDIX B

1. Sample Program Code

#include <WiFi.h>  
#include <AsyncTCP.h>  
#include <ESPAsyncWebServer.h>  
#include <Adafruit\_GFX.h>  
#include <Adafruit\_SSD1306.h>  
#define SCREEN\_WIDTH 128  
#define SCREEN\_HEIGHT 64  
#define OLED\_RESET -1  
Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, OLED\_RESET);  
  
const char\* ssid = "YourHotspotName";  
const char\* password = "YourHotspotPassword";  
AsyncWebServer server(80);  
String productName = "Product";  
String price = "$0.00";  
String expiry = "01/01/2099";  
  
void updateDisplay() {  
 display.clearDisplay();  
 display.setTextSize(2);  
 display.setCursor((SCREEN\_WIDTH - 6\*productName.length())/2, 0);  
 display.println(productName);  
 display.setTextSize(3);  
 display.setCursor((SCREEN\_WIDTH - 6\*price.length())/2, 25);  
 display.println(price);  
 display.setTextSize(1);  
 display.setCursor((SCREEN\_WIDTH - 6\*expiry.length())/2, 50);  
 display.println("EXP: " + expiry);  
 display.display();  
}  
  
void setup() {  
 Serial.begin(115200);  
 if (!display.begin(SSD1306\_SWITCHCAPVCC, 0x3C)) while (true);  
 WiFi.begin(ssid, password);  
 while (WiFi.status() != WL\_CONNECTED) delay(500);  
 server.on("/", HTTP\_GET, [](AsyncWebServerRequest \*request) {  
 request->send(200, "text/html", "<form action='/update'><input name='product'><input name='price'><input name='expiry'><input type='submit'></form>");  
 });  
 server.on("/update", HTTP\_GET, [](AsyncWebServerRequest \*request) {  
 if (request->hasParam("product")) productName = request->getParam("product")->value();  
 if (request->hasParam("price")) price = request->getParam("price")->value();  
 if (request->hasParam("expiry")) expiry = request->getParam("expiry")->value();  
 updateDisplay(); request->redirect("/");  
 });  
 server.begin(); updateDisplay();  
}  
  
void loop() {}

REFERENCES

[1] Supabase Documentation, Available: https://supabase.com/docs

[2] Adafruit, “SSD1306 OLED Display Guide,” Available: https://learn.adafruit.com/monochrome-oled-breakouts

[3] AsyncWebServer Library GitHub, Available: https://github.com/me-no-dev/ESPAsyncWebServer

[4] Arduino ESP32 Core Documentation, Available: https://docs.espressif.com/projects/arduino-esp32