Coffee Ordering System Proposal

Prepared for: Choeng Rayu

Date: May 30, 2025

1. Project Objective

To build a dual-platform Coffee Ordering System—one via Telegram and one via a responsive web application—that enables users to register, customize and place orders, receive a unique QR code upon completion, and verify pickup at the shop, thereby reducing waste from unclaimed orders and automating order management.

2. Scope of Work

2.1 Telegram Bot

- Framework & Hosting
  - Node.js with Telegraf.js
  - Hosted on Render.com via Webhook
- Database
  - MongoDB (Mongoose ORM)
- Key Features
  1. Menu Browsing: Hot, Iced, Frappes categories
  2. Customization: Size, sugar/ice level, add-ons
  3. Order Confirmation: Summary, price, “Confirm Order” button
  4. QR Code Generation: Unique code upon confirmed order
  5. Pickup Verification: User shows QR at shop; staff scans to complete
  6. No-Show Management: Strike system, mandatory deposit option

2.2 Web Application

- Front-End
  - React (JSX)
  - Styling via CSS Modules only
  - HTTP client: Axios
  - QR code generation: react-qr-code or equivalent
- Back-End
  - Node.js with Express
  - REST API endpoints

  - MongoDB (Mongoose Atlas)
- Hosting
  - API & Front-End: Render.com
  - Database: MongoDB Atlas

3. Detailed Requirements

3.1 User Registration & Authentication

- Registration Form: name, email, username, password
- Login Form: email or username + password
- Stay Signed In: Save login state in browser or session
- No password encryption or token security used

3.2 Ordering Flow (Web)

1. Login/Register
2. Browse Menu: images, descriptions, prices
3. Customize & Add to Cart
4. Review Cart: edit or confirm items
5. Place Order → system generates unique QR code
6. Order History: list of past orders & statuses

3.3 QR Code Generation & Verification

- On order completion, generate a UUID-based token and render as QR code
- Display QR code in-app and allow download/save
- Shop Side: simple web page or mobile view where staff scan QR
  - Endpoint /api/verify-qr receives token, checks for pending order
  - If valid → marks order completed & returns success; else → error

4. Technology Stack

- Front-End: React (JSX), CSS Modules, Axios
- Back-End: Node.js, Express
- Bot: Node.js, Telegraf.js
- Database: MongoDB with Mongoose
- QR Library: react-qr-code (web) / qr-image or uuid (bot)

- Hosting: Render.com (web, API, bot)

5. Database Schema

User

```
{
  _id: ObjectId,
  username: String,
  email: String,
  password: String,
  createdAt: Date
}
```

Order

```
{
  _id: ObjectId,
  userId: ObjectId,
  items: [
    { productId: ObjectId, name: String, options: Object, price: Number }
  ],
  total: Number,
  status: "pending" | "completed" | "no-show",
  qrToken: String,
  createdAt: Date
}
```

6. API Endpoints

- POST /api/register: Create new user account
- POST /api/login: Authenticate user and start session
- GET /api/menu: Retrieve coffee menu
- POST /api/orders: Place new order, returns order ID & QR
- GET /api/orders/:id/qr: Return QR code for given order ID
- POST /api/verify-qr: Validate QR token & mark order completed
- GET /api/user/orders: List orders for authenticated user

7. No-Show & Waste Reduction Strategies

1. Mandatory Deposit Option: small prepayment via ABA QR
2. Unique, One-Time QR: prevents reuse or fraudulent claims
3. Strike System: warnings → prepayment → temporary block
4. Auto-Cancel: orders unclaimed after 30 minutes marked “no-show”
5. Loyalty Rewards: encourage good behavior (e.g., 5 pickups → 1 free)

8. Deployment Plan

1. Repository Setup: Git monorepo with /client, /server, /bot
2. CI/CD: GitHub Actions to build & deploy each service to Render.com
3. Database: Provision MongoDB Atlas cluster, connect via Mongoose
4. Environment Management: use .env for secrets (DB URI)

10. Conclusion

This proposal outlines a robust, scalable Coffee Ordering System that leverages modern JavaScript stacks to deliver seamless ordering via both Telegram and web. By enforcing QR-based pickup and no-show mitigation, the solution minimizes waste and maximizes operational efficiency.