Work Breakdown Structure (WBS): A2B Restaurant Billing Automation System

Project: A2B Restaurant Billing Automation System

1. Requirements Analysis

Objective: Understand what the billing system must do.

- **1.1 Define project scope and objectives**
- **1.2 Identify users and roles** (Admin, Cashier, Branch Manager)
- **1.3 Gather feature requirements**
    - Menu sync
    - Offline billing
    - Central dashboard
    - Reporting
- **1.4 Select tools and tech stack**
    - Python 3.x, Django, MySQL, REST APIs, React, Power BI
- **1.5 Define deployment model** (offline-first, cloud sync)

---

2. System Design Phase

Objective: Define system architecture and flow.

2.1 Define project folder & module structure

- menu.py, order.py, sync_service.py, etc.

2.2 Design MySQL database schema

- Tables: menu_items, orders, users, branches

2.3 Draft flowcharts for core modules

- Sync engine
- POS workflow

2.4 Plan offline-first architecture

2.5 Design REST API endpoints for sync and dashboard

2.6 Define data sync logic and triggers

---

3. Development Phase

Objective: Implement core functional modules in Python/Django.

3.1 Menu Management (menu.py)

- 3.1.1 Create MenuManager class
- 3.1.2 CRUD operations (add, edit, delete)
- 3.1.3 Sync with central server
- 3.1.4 Integrate with Django admin

3.2 Order & Billing System (order.py, billing.py)

- 3.2.1 Create OrderManager class
- 3.2.2 Accept item input via GUI or CLI
- 3.2.3 Calculate subtotal, tax, discounts
- 3.2.4 Generate and print bills
- 3.2.5 Save orders in local MySQL

3.3 Offline Capability

- 3.3.1 Local DB setup and connection
- 3.3.2 Log failed syncs locally
- 3.3.3 Retry logic for data syncing

3.4 Sync Engine (sync_service.py, scheduler.py)

- 3.4.1 Push unsynced orders to server
- 3.4.2 Pull updated menu
- 3.4.3 Handle data conflicts
- 3.4.4 Create sync status tracker

3.5 Admin Dashboard (React + Django APIs)

- 3.5.1 Display real-time sales data
- 3.5.2 View and filter historical orders
- 3.5.3 Menu & user management UI
- 3.5.4 Export data (CSV, PDF)

3.6 User and Role Management (auth.py, roles.py)

- 3.6.1 Implement login/logout
- 3.6.2 Session and password management
- 3.6.3 Role-based access control

---

4. Testing Phase

Objective: Verify every module and integration point.

- **4.1 Unit test each module**
    - menu.py, order.py, sync_service.py
- **4.2 Integration test across modules**
- **4.3 Test offline & online sync transitions**
- **4.4 Validate role-based access**
- **4.5 Perform edge case and stress testing**

---

5. Documentation

Objective: Provide project clarity for devs and users.

- **5.1 README with setup & usage instructions**
- **5.2 Technical Design Document**

- **5.3 API Documentation (Swagger/Postman)**
- **5.4 User Manual & SOPs**
- **5.5 Training materials for A2B staff**

---

6. Packaging & Deployment

Objective: Prepare the system for installation and scale.

- **6.1 Package backend/frontend separately**
- **6.2 Create environment setup scripts**
- **6.3 Create install/config guide for branches**
- **6.4 Perform dry run on test machine**
- **6.5 Prepare deployment package for new branches**

---

7. Maintenance & Support

Objective: Ensure long-term success and stability.

- **7.1 Define support SLA and escalation matrix**
- **7.2 Bug and feature request tracker**
- **7.3 Plan for monthly patch updates**
- **7.4 Data backup and recovery strategy**