NeoPOS – Restaurant Management System

Project Report & Technical Explanation

1. Introduction

NeoPOS is a **cross-platform Point of Sale (POS) management system** developed using **Flutter** with **Firebase Firestore** as the backend database and **BLoC (Business Logic Component)** as the state-management architecture.
The application is designed to manage **restaurant operations** efficiently, including **product management, table management, order handling, user roles, and sales analytics**, with support for **both web and mobile devices**.

The system follows **modular architecture**, clean separation of concerns, and scalable patterns suitable for real-world production applications.

2. Technology Stack

Layer	Technology
Frontend	Flutter (Web + Android + iOS)
State Management	BLoC + Equatable
Backend	Firebase Firestore
Dependency Injection	GetIt
Localization	Flutter gen_l10n (ARB based)
Charts & Analytics	Syncfusion Charts, Pie Chart
Image Handling	CachedNetworkImage
Architecture	Feature-based modular architecture

3. Application Architecture

NeoPOS follows a **layered and modular architecture**:

UI (Screens & Widgets)
  ↓
BLoC Layer (Events → Business Logic → States)
  ↓
Data Layer (Firebase Firestore)

Key Architectural Principles

- **Unidirectional Data Flow** using BLoC
- **Loose coupling** using dependency injection (GetIt)
- **Feature-wise separation** (Products, Tables, Users, Sales)
- **Single source of truth** (Firestore)
- **Reactive UI updates** via BlocBuilder / BlocConsumer

4. Core Modules Explanation

4.1 Localization Module

The app supports **multi-language UI (English & Hindi)** using Flutter's **official gen_l10n system**.

- Language preference stored using SharedPreferences
- LocalizationBloc controls language switching
- ARB files (app_en.arb, app_hi.arb) define all UI strings
- Dynamic language switching without app restart

Benefits

- Global-ready application
- Easy future language expansion
- Clean separation of UI text from logic

4.2 Product Management Module

This module handles **CRUD operations for products**.

Features

- Add, update, delete, and read products
- Product attributes:
    - Name
    - Description
    - Category
    - Price
    - Availability
    - Veg / Non-Veg type
    - Image (URL)
- Responsive UI for desktop and mobile

Architecture

- ReadProductsBloc fetches products from Firestore
- Product data mapped using ProductModel
- UI reacts via ReadDataLoadedState
- Dialog-based edit/delete operations

State Management

Event → ReadInitialEvent
State → Loading → Loaded / Error

4.3 Table Management Module

Used to manage restaurant seating tables.

Features

- Create tables with capacity
- Update table name and capacity
- Delete tables with admin credential verification
- Live table synchronization using `live_table` collection

Security

- Admin role verification before delete/update
- Credential validation using Firestore users collection

Key Blocs

- `TableBloc` → Read tables
- `CreateTableBloc` → Create table
- `TableDeletionBloc` → Secure deletion
- `TableUpdateBloc` → Update table details

4.4 User Management Module

Handles **staff accounts** (Admin / Waiter).

Features

- Create users with role-based access
- Update user details
- Delete users (Admin-only action)
- Role validation before sensitive actions

User Model Includes

- User ID
- First Name / Last Name
- Password
- Role
- Created & Updated timestamps

Security Logic

- Admin credentials required for delete
- Firestore query validation
- Bloc-controlled authentication flow

4.5 Sales Dashboard & Analytics Module

This is the **analytics heart** of NeoPOS.

Dashboard Features

- Daily, weekly, and monthly revenue
- Pie chart of category-wise sales
- Top 5 selling products (Daily / Weekly / Monthly)
- Sales trend graph (month-wise)

- Recent order history

Data Processing

- Raw Firestore order data
- Date-based aggregation using `intl`
- Week number calculation logic
- Dynamic graph filtering using `GraphDashboardBloc`

Visualization

- Column charts for daily sales
- Pie charts for category contribution
- Carousel UI for mobile dashboard

4.6 Order Handling Module

- Product selection with quantity
- Add items to table orders
- Order persistence in Firestore
- Used by both desktop and mobile flows

5. Routing & Navigation

- Centralized routing via `AppRouter`
- Named routes using `RoutePaths`
- Supports:
    - Splash screen

    - Login
    - Dashboard
    - Order menu (Web & Mobile)

Advantages

- Clean navigation control
- Role-based routing possible
- Easy future expansion

6. Dependency Injection

Firebase Firestore is injected using **GetIt**:

```
locator.registerLazySingleton<FirebaseFirestore>(
  () => FirebaseFirestore.instance
);
```

Benefits

- Testability
- No tight coupling
- Centralized service management

7. Key Strengths of the Project

- ✅ Production-ready architecture
- ✅ Clean separation of UI & business logic
- ✅ Scalable Firebase schema
- ✅ Role-based access control

- ✅ Multi-language support
- ✅ Responsive Web + Mobile UI
- ✅ Real-time analytics
- ✅ Enterprise-grade state management

8. Conclusion

NeoPOS is a **complete, scalable, and real-world POS application** built using modern Flutter best practices.
The project demonstrates strong understanding of:

- **State management with BLoC**
- **Firebase integration**
- **Clean architecture**
- **Role-based security**
- **Data analytics & visualization**
- **Localization & responsiveness**

Data Flow Diagrams (DFD) – NeoPOS

1️⃣DFD Level 0 – Context Diagram

Entities

- Admin
- Waiter
- Firebase Firestore

Actor	Data Sent	Data Received
Admin	Login, Manage Products, Tables, Users	Dashboard, Reports
Waiter	Orders, View Products	Order Status
NeoPOS	CRUD requests	Stored data / Analytics
Firestore	Data	Query Results

Explanation

- Admin & Waiter interact with **NeoPOS UI**
- NeoPOS processes logic via **BLoC**
- Data is stored/retrieved from **Firebase Firestore**

2DFD Level 1 – System Decomposition

Here, the system is broken into **major modules**.

Processes

1. User Management
2. Product Management

3. Table Management
4. Order Processing
5. Sales & Analytics
6. Localization

DFD Process	Flutter Module
User Management	`CreateUserBloc`, `ReadUserBloc`, `UpdateUserBloc`
Product Management	`ReadProductsBloc`, `ProductDeletionBloc`
Table Management	`TableBloc`, `CreateTableBloc`, `TableDeletionBloc`
Order Processing	`OrderContentBloc`
Sales Analytics	`SalesDashboardBloc`, `GraphDashboardBloc`, `Top5Bloc`
Localization	`LocalizationBloc`

3️⃣ DFD Level 2 – Product Management (Detailed)

Admin
↓
Product UI (Flutter Screen)

↓
ReadProductsBloc / ProductDeletionBloc
↓
Firebase Firestore (products)
↓
Updated Product List

Explanation

- UI triggers **BLoC events**
- BLoC validates logic
- Firestore stores product data
- UI updates reactively via states

4️⃣ DFD Level 2 – Table Management

Admin
↓
Table UI
↓
CreateTableBloc / UpdateTableBloc / TableDeletionBloc
↓
Firestore (table + live_table)

Security

Delete/update requires Admin credentials

Firestore is queried for role verification

5️⃣ DFD Level 2 – User Management

Admin

↓

User UI

↓

CreateUserBloc / UpdateUserBloc / UserDeletionBloc

↓

Firestore (users)

6️⃣ DFD Level 2 – Order & Sales Analytics

This directly matches your **SalesDashboardBloc** logic.

Flow

Orders
↓
order_history (Firestore)
↓
SalesDashboardBloc
↓
• Daily Revenue
• Weekly Revenue
• Monthly Revenue
• Pie Chart
• Top 5 Products
• Sales Graph

Processing

Date grouping

Week calculation

Category aggregation

Graph filtering by month