**Food Order & Delivery App - Technical Documentation**

**Architecture Overview**

The application follows the MVVM (Model-View-ViewModel) architecture pattern with Room database, ensuring separation of concerns and maintainable codebase.

**Project Structure**

**1. Java Source (java/com.example.foodorderapp/)**

**Activities (activities/)**

Contains the app's main screens and entry points:

* LoginActivity.java: User authentication screen
* RegisterActivity.java: New user registration
* MainActivity.java: Customer's main interface
* AdminActivity.java: Administrator's dashboard
* DeliveryActivity.java: Delivery agent's interface
* SplashActivity.java: App launch screen

**Adapters (adapters/)**

RecyclerView adapters for list displays:

* CartAdapter.java: Shopping cart items display
* MenuAdapter.java: Food menu items list
* OrderAdapter.java: Customer's order list
* ManageOrderAdapter.java: Admin's order management
* DeliveryOrderAdapter.java: Delivery agent's order list

**Database (database/)**

Room database implementation with three sub-packages:

**DAO (database/DAO/)**

Data Access Objects for database operations:

* UserDAO.java: User account operations
* MenuItemDAO.java: Food menu CRUD operations
* CartDAO.java: Shopping cart management
* OrderDAO.java: Order processing and tracking
* DeliveryAgentDAO.java: Delivery agent management
* AddressDAO.java: Customer address management

**Entities (database/entities/)**

Database table definitions:

* UserEntity.java: User account information
* MenuItemEntity.java: Food item details
* CartItemEntity.java: Shopping cart items
* OrderEntity.java: Order information
* OrderItemEntity.java: Individual order items
* DeliveryAgentEntity.java: Delivery agent details
* AddressEntity.java: Customer addresses

**Repositories (database/repositories/)**

Business logic and data operations:

* UserRepository.java: User authentication and management
* MenuRepository.java: Menu items management
* CartRepository.java: Cart operations
* OrderRepository.java: Order processing
* DeliveryAgentRepository.java: Delivery management
* AddressRepository.java: Address handling

**Fragments (fragments/)**

UI components for different sections:

* MenuFragment.java: Food menu display
* CartFragment.java: Shopping cart interface
* OrdersFragment.java: Order history and tracking
* ManageMenuFragment.java: Admin menu management
* ManageOrdersFragment.java: Admin order handling
* ManageUsersFragment.java: User management

**Models (models/)**

Data models for business logic:

* MenuItem.java: Food item model
* Other helper classes

**Utils (utils/)**

Helper classes and utilities:

* OrderStatus.java: Order status constants
* OrderNotificationService.java: Push notifications
* Other utility classes

**2. Resources (res/)**

**Layouts (layout/)**

XML layouts for UI:

* Activities layouts (activity\_\*.xml)
* Fragment layouts (fragment\_\*.xml)
* List item layouts (item\_\*.xml)
* Dialog layouts (dialog\_\*.xml)

**Menu (menu/)**

Navigation and option menus:

* bottom\_navigation\_menu.xml: Customer navigation
* admin\_bottom\_navigation.xml: Admin navigation
* options\_menu.xml: App options menu

**Drawable (drawable/)**

Icons and images:

* ic\_\*.xml: Vector icons
* Other image resources

**Data Flow**

**Authentication Flow**

1. User inputs credentials
2. LoginActivity validates input
3. UserRepository checks database
4. Role-based redirection to appropriate activity

**Order Processing Flow**

1. Customer adds items to cart
2. Cart data stored in local database
3. Order creation triggers notification
4. Admin receives and processes order
5. Delivery agent assignment
6. Real-time status updates

**Database Schema**

sql

*// Key Tables Structure*

users (id, name, email, password, phone, role)

menu\_items (id, name, description, price, category)

orders (id, userId, totalAmount, status, deliveryAgentId)

cart\_items (id, menuItemId, quantity, price)

delivery\_agents (id, name, email, phone, isAvailable)

**Key Features Implementation**

**Real-time Order Tracking**

* Status updates through OrderRepository
* Notification service for status changes
* LiveData observers for UI updates

**Cart Management**

* Local database storage
* Real-time total calculation
* Automatic sync with menu changes

**Delivery System**

* Agent availability tracking
* Order assignment system
* Status update mechanism

**Technical Dependencies**

* Room Persistence Library
* Android Architecture Components
* Material Design Components
* ViewModels and LiveData
* RecyclerView for lists
* SQLite database

**Performance Considerations**

* Asynchronous database operations
* Efficient list rendering
* Memory management
* Background task handling

**Security Measures**

* Password encryption
* Role-based access control
* Input validation
* Secure data storage

**Testing Strategy**

* Unit tests for repositories
* Integration tests for database
* UI tests for critical flows
* Manual testing checklist

**Future Enhancements**

1. Real-time location tracking
2. Payment gateway integration
3. Chat system
4. Analytics dashboard
5. Multi-language support

This documentation provides a comprehensive overview of the application's architecture and technical implementation. For specific implementation details, refer to the individual source files and inline documentation.