Abdelrahman Abdelwahab Ahwa owner paragraph.

This report presents the design and implementation of the Smart Ahwa Manager application in Flutter, built to streamline operations in a Cairo coffee shop. The app allows an ahwa owner to manage customer orders, track popular drinks, and generate daily sales reports. More importantly, the design implementation was used in each part.
Firstly the presentation pattern followed to design and implement this project was as follows.

```
UI Layer → Use Case → Repository Interface → Repository
Implementation → Data Source → SQLite Database
```

First the UI layer composed of custom widgets and pages to display content and analytics.
Second layer Usecase contains business rules and logic.
Third Implement data access logic, coordinate between use cases and data sources.
Fourth layer defines contracts for data operations.
Fifth layer Handle direct database operations, raw data access.

Secondly The implementation of SOLID and OOP principles.
This application implements all OOP principles and have few examples of SOLID principles.
For example:
1- Abstraction, Abstract repository hides implementation details and UI layer doesn't know about SQLite - only uses abstraction.

```
abstract class ItemRepository {
 Future<List<ItemModel>> getAllItems();
 Future<void> addItem(ItemModel item);
}

//
class ItemsPage extends StatefulWidget {
 final ItemUsecase itemService;
}
```

2- Inheritance, Abstract repository contracts and Concrete implementation inherits behavior

```
abstract class OrderRepository {
 Future<List<OrderModel>> getAllOrders();
 Future<void> addOrder(OrderModel order);
}

class OrderRepositorySQLite implements OrderRepository {
 @override
 Future<List<OrderModel>> getAllOrders() => _db.getAllOrders();
}
```

3- Encapsulation, AppDatasource - Private instance and database access, private database instance, controlled access and private constructor

```
class AppDatasource {
 static final AppDatasource _instance = AppDatasource._internal();
 Database? _db;

 factory AppDatasource() => _instance;
 AppDatasource._internal();   }
```

4-Polymorphism, as for this one, there is NavigationServicve, which is basically used for navigation.

SOLID principles applied:
1-SRP, in OrderModel it only handles order data structure and in OrderUsecase it only handles business logic

```
class OrderModel {
 final String orderId;
 final ItemModel items;
 final OrderStatus orderStatus;
}

class OrderUsecase {
 Future<void> addOrder(OrderModel order) => repository.addOrder(order);
}
```

2-OCP. IN OrderRepository we can add new repository implementations without changing existing code.

```
abstract class OrderRepository {
 Future<List<OrderModel>> getAllOrders();
}
```

3-LSP, in OrderUsecase any OrderRepository implementation can replace another

```
class OrderUsecase {
 final OrderRepository repository;  // Works with any implementation

 OrderUsecase(this.repository);
}
// In main.dart - can switch implementations easily
final orderService = OrderUsecase(orderRepo);
```

4-ISP, Separate interfaces for different responsibilities so the child class could implement only what they need.

```dart
abstract class DailyReportsRepository {
 double totalSales();
 int totalItemsSold();
}

abstract class TopSellingReportsRepository {
 Map<String, int> topSellingItems();
}

class DailyReportsRepositoryImp implements DailyReportsRepository {
 // Only implements daily reports methods
}
```

5-DIP, the High-level modules depend on abstractions also `UI depends on use cases, not repositories directly and Dependency injection in main.dart`

```dart
class OrderUsecase {
 final OrderRepository repository;  // Abstract interface, not concrete class
 OrderUsecase(this.repository);
}


class DashboardPage extends StatefulWidget {
 final OrderUsecase orderService;  // Abstract use case, not concrete repo
}


final orderRepo = OrderRepositorySQLite();  // Concrete implementation
final orderService = OrderUsecase(orderRepo);  // Injected dependency
```

Note: All full solutions will be provided in problems folder in the repository.
Repo: https://github.com/Garfend/week1_assignments.git