Banking System Architecture - Layered Pattern with MVC

Presentation Layer

LoginUI   Customer Dashboard UI   Admin Dashboard UI

↓ *Business Operations & Validation*

Business Logic Layer

Customer   Admin   Account   Transaction   Loan   Transfer

↓ *CRUD Operations*

Data Access Layer

DB Connection
(Singleton)

MySQL Database

Utility Components

PDF Generator   Charts

→ Report Generation

→ Data Visualization

Design Patterns Applied:

- **Singleton:** DBConnection ensures single database connection instance
- **MVC:** Clear separation between UI, business logic, and data
- **Facade:** Dashboard classes provide simplified interfaces
- **Observer:** For real-time notifications and updates
- **Factory:** For creating different types of accounts

Architecture Justification:

The layered architecture provides:

- **Separation of Concerns:** Each layer has distinct responsibilities
- **Maintainability:** Changes in one layer don't affect others
- **Scalability:** Individual layers can be scaled independently
- **Testability:** Each layer can be tested in isolation
- **Clear Dependencies:** Top-down dependency flow

Data Flow: Presentation Layer → Business Logic Layer → Data Access Layer → Database
Cross-cutting: Utility components serve multiple layers for specialized functions