**System Architecture**

Main Components

1. **User Management Subsystem**
2. **Authentication Service Subsystem**
3. **Asset Management Subsystem**
4. **Goal Tracking Subsystem**
5. **Banking Transaction Subsystem**
6. **Zakat Calculation Subsystem**

Class Documentation

1. User Management

User Class

* **Purpose**: Represents a system user and their financial data
* **Properties**:
  + Name: User's name
  + Password: User's password
  + Email: User's email address
  + Is\_Logged\_in: Login status flag
  + BAcount: Associated bank account
  + ass\_coll: Collection of user assets
* **Methods**:
  + CreateAccount(): Links a bank account to the user
  + Get/Set methods for all properties
  + Addass(): Adds a new asset to the collection

UpdateProfile Class

* **Purpose**: Handles user profile updates
* **Methods**:
  + Update\_Profile(): Provides menu for updating name, email, or password

2. Authentication Service

Auth\_Service Class

* **Purpose**: Handles user registration and login
* **Methods**:
  + Sign\_Up(): Creates new user accounts
  + Log\_in(): Authenticates existing users

CheckUsers Class

* **Purpose**: Validates user existence
* **Methods**:
  + FoundCheck(): Checks if user already exists

3. Asset Management

AssetColletion Class

* **Purpose**: Manages collection of user assets
* **Methods**:
  + addAsset(): Adds new asset
  + editAsset(): Modifies existing asset
  + removeAsset(): Deletes asset
  + printAsset(): Displays all assets
  + GetTotalAssetValue(): Calculates total value of all assets

Asset Class

* **Purpose**: Represents a financial asset
* **Properties**:
  + AssetName: Name of asset
  + PurchaseName: Purchase details
  + ValueOfAsset: Monetary value
  + RiskValue: Risk assessment
  + AssetGoal: Associated financial goal
* **Methods**:
  + AddGoal(): Links a goal to the asset

4. Goal Tracking

UserGoals Class

* **Purpose**: Tracks financial goals
* **Properties**:
  + goalName: Goal name
  + description: Goal description
  + targetAmount: Target value
  + currentAmount: Current saved amount
* **Methods**:
  + AddProgress(): Updates progress toward goal
  + IsAchieved(): Checks if goal is met
  + GetProgressPercentage(): Calculates completion percentage

Supporting Classes:

* GoalInputHandler: Handles goal creation input
* GoalViewer: Displays goal information
* IGoalObserver/Notify: Implements observer pattern for goal notifications

5. Banking Transactions

BankAccount Class

* **Purpose**: Represents a bank account
* **Properties**:
  + bankName: Bank institution name
  + accountNumber: Account identifier
  + Amount: Account balance
  + t: List of transactions
* **Methods**:
  + AddTransaction(): Processes new transactions

Transaction Classes:

* ITransactionStrategy: Interface for transaction strategies
* DepositStrategy, WithdrawalStrategy, TransferStrategy, PaymentStrategy: Concrete transaction implementations
* Transaction: Represents a financial transaction
* TransacionHistory: Manages transaction records

6. Zakat Calculation

calculateZakat Class

* **Purpose**: Calculates zakat obligations
* **Methods**:
  + CalculateZakat(): Computes 2.5% of total asset value

7. Main Menu System

main\_menu Class

* **Purpose**: Provides user interface and navigation
* **Methods**:
  + start(): Main menu loop
  + managAsset(): Asset management submenu
  + managGoals(): Goal management submenu

Usage Flow

1. **Authentication**:
   * New users sign up with name, email, and password
   * Existing users log in with credentials
2. **Main Menu Options**:
   * Manage assets (add, view, edit, remove)
   * Set and track financial goals
   * Calculate zakat obligations
   * Link bank accounts
   * Perform transactions (deposits, withdrawals, etc.)
   * View transaction history
   * Update profile information
3. **Asset Management**:
   * Track various financial assets with details
   * Associate goals with specific assets
   * Monitor progress toward goals
4. **Banking**:
   * Link external bank accounts
   * Record financial transactions
   * View complete transaction history

Design Patterns Used

1. **Strategy Pattern**:
   * Implemented in transaction processing (ITransactionStrategy and concrete implementations)
2. **Observer Pattern**:
   * Used for goal tracking notifications (IGoalObserver and Notify)
3. **Separation of Concerns**:
   * Clear division between authentication, asset management, goal tracking, and banking subsystems

Error Handling

The system includes basic input validation for:

* Numeric values (asset values, amounts)
* User authentication
* Menu selection boundaries
* Transaction type validation

Limitations

1. No persistent data storage (all data is in-memory)
2. No password encryption/hashing
3. Basic error handling
4. No multi-user concurrency support

Future Enhancements

1. Database integration for persistence
2. Enhanced security (password hashing)
3. More sophisticated reporting
4. Budget tracking features
5. Investment performance tracking