High-Level Components

1. **Core**:
   - Contains the application's main entry point and handles startup/shutdown logic.
2. **MarketData**:
   - Fetches and caches market data (historical and real-time).
3. **TradingStrategies**:
   - Encapsulates various trading strategies.
4. **TradingEngine**:
   - Evaluates strategies and sends trade signals.
5. **OrderManagement**:
   - Places, modifies, and cancels orders.
6. **RiskManagement**:
   - Monitors risk constraints like maximum drawdown, position limits, etc.
7. **DataStorage**:
   - Handles interactions with the database (e.g., Postgres).
8. **Logging**:
   - Centralized logging for debugging and analytics.

Class Design

1. Core

- **`Program.cs`**
  - Entry point of the application.
  - Initializes all components and starts the trading loop.

2. MarketData

- **`IMarketDataService`** (Interface)
  - Methods:
    - `GetHistoricalData(string symbol, DateTime startDate, DateTime endDate): IEnumerable<PriceData>`
    - `GetRealtimeData(string symbol): PriceData`
- **`MarketDataService`** (Implementation)
  - Integrates with IBClient to fetch and cache data.
  - Acts as a bridge between the API and other components.

---

3. TradingStrategies

- **`ITradingStrategy`** (Interface)
  - Methods:
    - `Evaluate(MarketContext context): TradeSignal`
    - `GetName(): string`
- **`MovingAverageCrossoverStrategy`**
  - A sample strategy using moving averages.
- **`RSIStrategy`**
  - A sample strategy using Relative Strength Index.

---

4. TradingEngine

- **`TradingEngine`**
  - Dependencies:
    - `IMarketDataService`
    - `ITradingStrategy` (supports multiple strategies)
    - `IOrderService`
    - `IRiskManagementService`
  - Logic:
    - Evaluates market data using registered strategies.
    - Sends trade signals to the order management system.

---

5. OrderManagement

- **`IOrderService`** (Interface)
  - Methods:
    - `PlaceOrder(Order order): OrderResponse`
    - `CancelOrder(string orderId): bool`
- **`OrderService`** (Implementation)
  - Integrates with IBClient for order placement and tracking.

---

6. RiskManagement

- **`IRiskManagementService`** (Interface)
  - Methods:
    - `EvaluateTrade(TradeSignal signal): bool`
- **`RiskManagementService`**
  - Implements risk constraints like position sizing, leverage limits, etc.

---

7. DataStorage

- **IDataStorage** (Interface)
  - Methods:
    - InsertHistoricalData(IEnumerable<PriceData> data)
    - GetHistoricalData(string symbol, DateTime start, DateTime end): IEnumerable<PriceData>
- **PostgresDataStorage**
  - Implementation for Postgres database using raw SQL.

---

8. Logging

- **ILogger** (Interface)
  - Methods:
    - LogInfo(string message)
    - LogError(string message, Exception ex)
- **ConsoleLogger**
  - Simple implementation that logs to the console.
- **FileLogger**
  - Writes logs to a file.