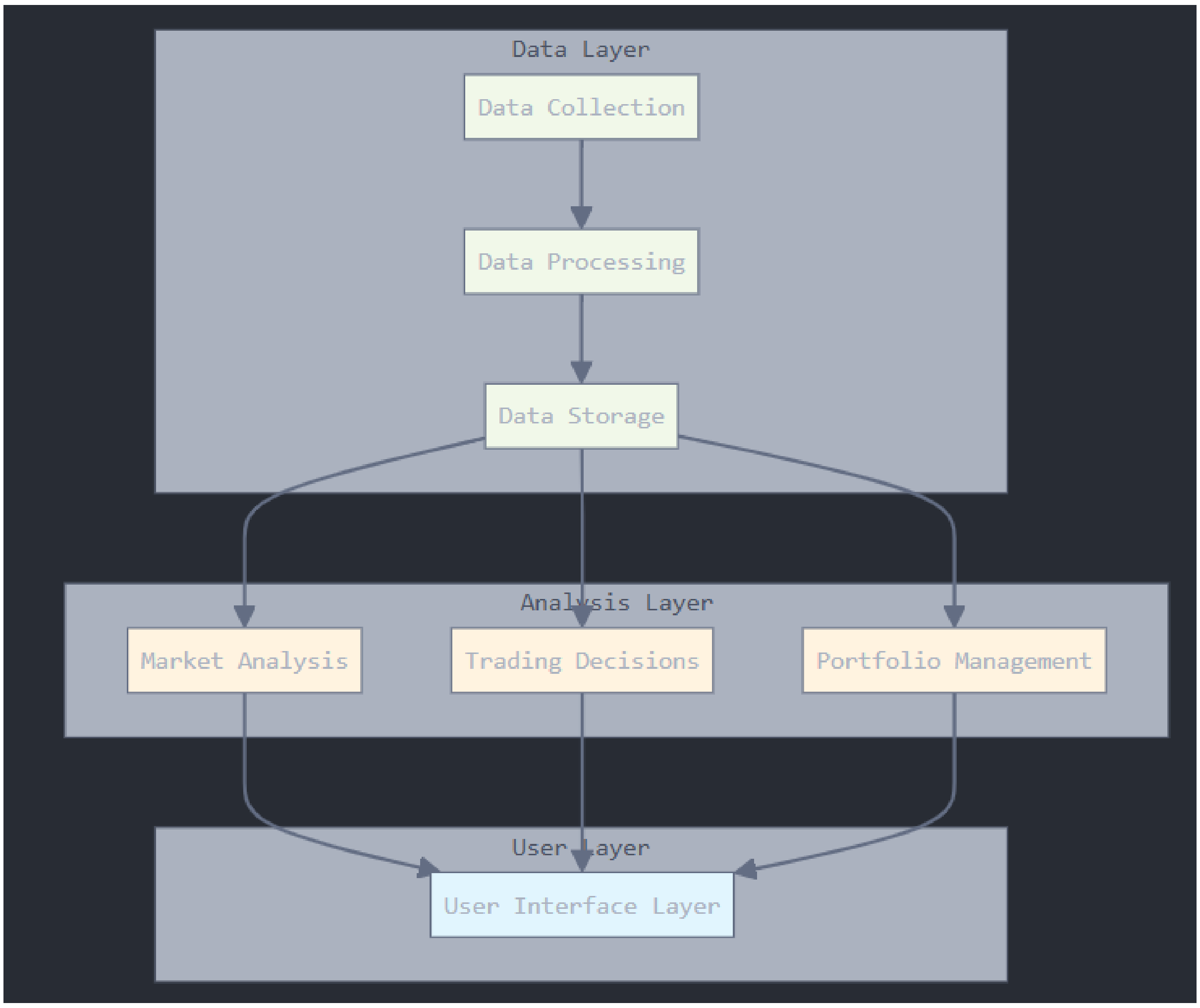
1. Conceptual Architecture Documentation



1. User Layer

- Purpose: Provide intuitive access to market data and analysis
- Components: Dashboard, reports, visualizations
- Capabilities: View market data, receive recommendations, manage portfolio

2. Analysis Layer

- Market Analysis
  - Technical indicators processing
  - Fundamental data analysis
  - Pattern recognition
  - Market trend identification
- Trading Decisions
  - Buy/sell recommendations
  - Risk assessment
  - Market opportunity identification
  - Confidence scoring
- Portfolio Management

    - Portfolio tracking
    - Performance monitoring
    - Investment distribution
    - Risk balancing

3. Data Layer

- Data Collection
    - Market data acquisition
    - Company reports gathering
    - Historical data compilation
    - Real-time updates
- Data Processing
    - Data cleaning and validation
    - Format standardization
    - Data enrichment
    - Quality assurance
- Data Storage
    - Historical price archives
    - Processed market data
    - Analysis results
    - Trading recommendations

4. Key Data Flows

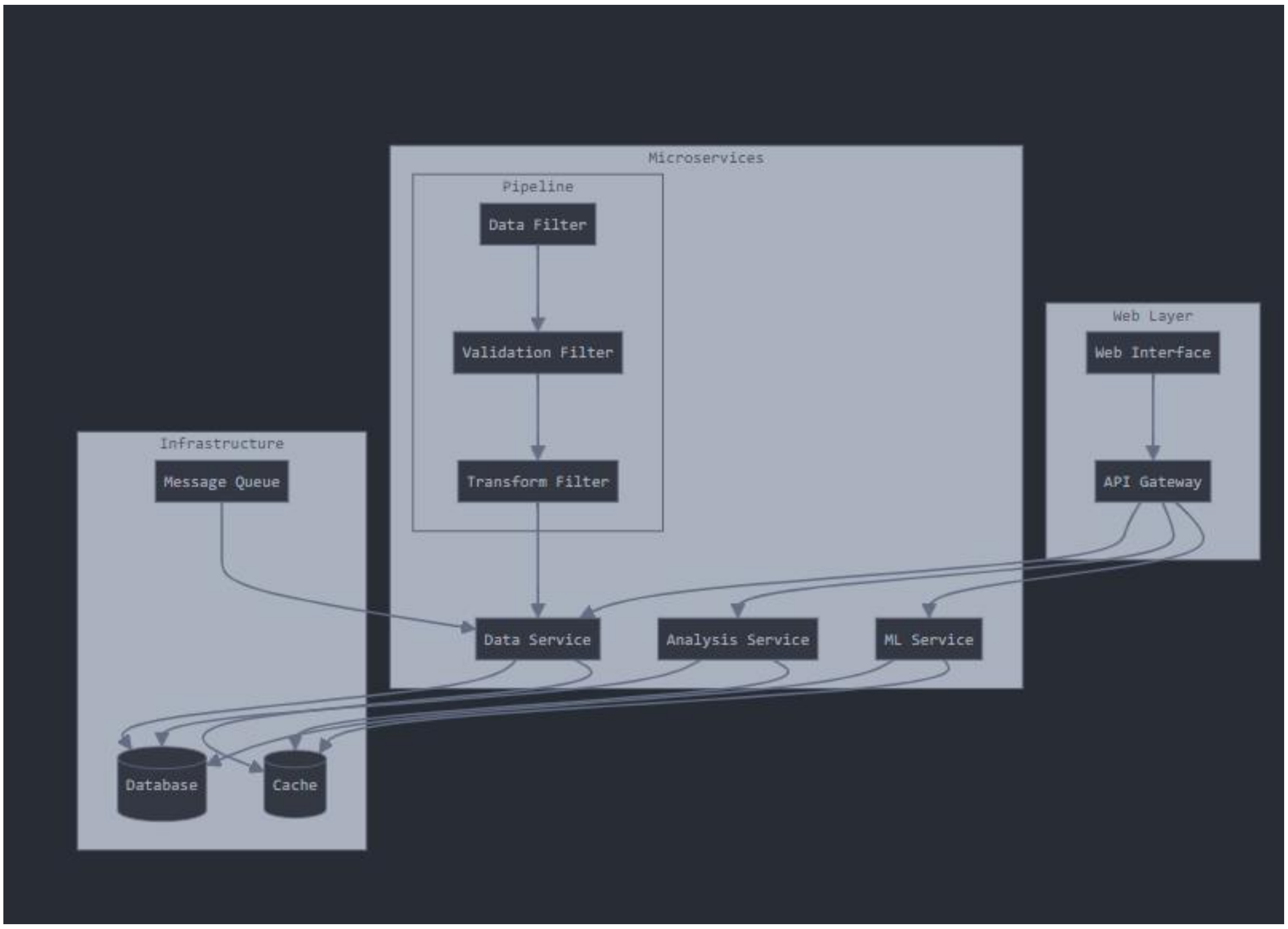
- Raw data → Processed data → Analysis
- Analysis results → Trading decisions
- Trading decisions → User recommendations
- Market updates → Real-time processing → Live display

5. Business Rules

- Data validation requirements
- Trading recommendation criteria
- Risk management constraints
- Portfolio diversification rules
- Update frequency requirements

This architecture emphasizes business functionality while remaining technology-agnostic, making it accessible to all stakeholders while providing a clear foundation for technical implementation.

2. Execution Architecture Documentation:



Docker Environment:

- Application container: Django backend, Redis cache, ETL pipeline, Analysis/ML modules
- Database container: PostgreSQL
- Nginx container: Reverse proxy, load balancing

System Components:

1. Frontend: Browser-based UI with Chart.js visualizations
2. Backend: Django application handling business logic
3. Database: PostgreSQL for data persistence
4. Cache: Redis for performance optimization
5. ETL: Data pipeline for processing market data
6. Analysis: Technical/fundamental analysis engine
7. ML: Prediction and recommendation system

Data Flow:

- External data sources → ETL pipeline → Database
- User requests → Nginx → Django → Cache/Database

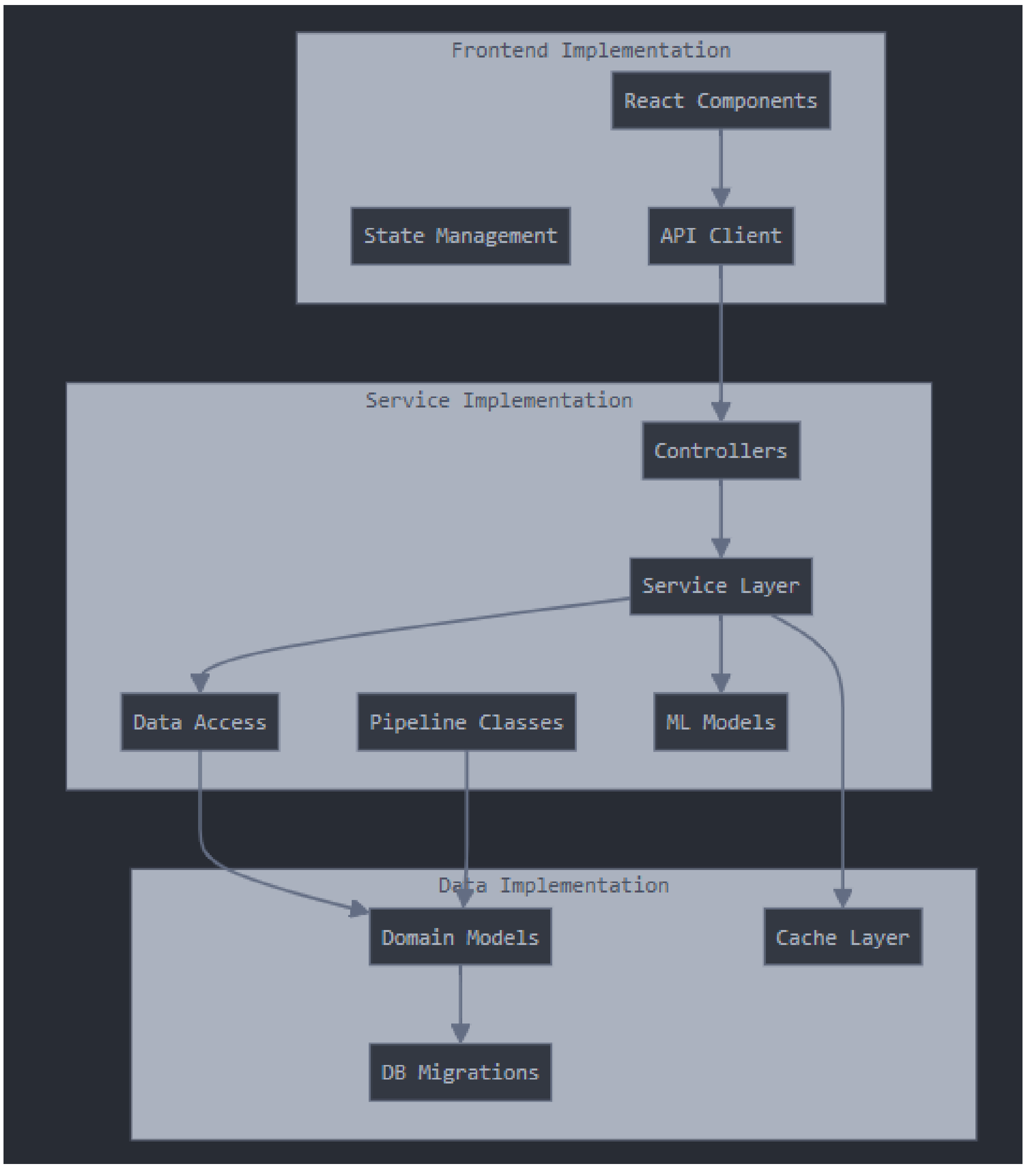
- Analysis requests → ML module → Database
- Results → Django → Frontend

Deployment:

- Docker Compose for orchestration
- Container networking
- Volume persistence
- Environment configuration

This architecture ensures scalability, maintainability, and isolation of components while facilitating efficient data processing and analysis.

3. Implementation Architecture Documentation:



Component Implementation:

1. Frontend

- React components for UI
- Redux for state management
- Chart.js for visualizations

2. Backend

- Django MVT architecture
- REST API endpoints
- Pipeline processing system
- ML model integration

3. Data Layer

- PostgreSQL models and migrations
- Redis caching implementation
- ETL scripts and schedulers

Technologies:

- Frontend: React, Redux, Chart.js
- Backend: Django, Django REST
- Database: PostgreSQL, Redis
- ML: Scikit-learn/TensorFlow
- Infrastructure: Docker, Nginx

This architecture details exact implementation patterns, directory structures, and technology choices for development.