Architecture Document for Expenditure Data Analysis

Jaideep Jaiswal and Kushagra Taneja

October 31, 2024

Contents

1 Executive Summary

This architecture document provides an in-depth view of the "Expenditure Data Analysis" project, designed for Physics Wallah to deliver robust financial analytics capabilities. The project includes data sourcing, transformation, analysis, visualization, and insights into expenditure metrics.

2 Introduction

2.1 Background and Importance

Effective data analysis helps organizations make informed decisions. The "Expenditure Data Analysis" project aims to create a comprehensive dashboard for tracking and understanding financial data to improve operational efficiency.

2.2 Objectives and Goals

- Deliver a dashboard that visualizes key financial data and insights.
- Provide interactive features for detailed data filtering and trend analysis.
- Enhance the decision-making process through clear data representation.

3 System Architecture Overview

The architecture comprises five main layers:

1. **Data Sources**: Includes ERP systems, financial databases, CSV uploads, and cloud-based data.
2. **ETL Process**: Extract, Transform, and Load procedures using Apache NiFi and custom Python scripts.
3. **Data Storage**: Utilizes a SQL-based data warehouse for structured storage and Amazon S3 for raw data.
4. **Data Processing and Analytics**: Python libraries like Pandas for data manipulation and Matplotlib for visualization.
5. **User Interface**: Tableau and Power BI for presenting data insights with interactive filters.

4 Detailed Architecture Components

4.1 Data Sources

- **Internal Data**: Financial records and ERP exports.
- **External Sources**: Public financial data and economic indicators.
- **Input Formats**: CSV, Excel sheets, and database connections.

4.2 ETL Process

- **Extraction**: Automated data fetching from the sources using scheduled tasks.
- **Transformation**: Cleaning and structuring data using Python (Pandas).
- **Loading**: Loading data into a SQL-based warehouse.
- **Scheduling**: Cron jobs for periodic data updates.

4.3 Data Storage

- **Primary Database**: MySQL for relational data storage.
- **Backup Storage**: Amazon S3 for backup and archiving.
- **Storage Structure**: Data organized into tables by categories like revenue, expenses, and budget forecasts.

4.4 Data Processing and Analytics

- **Data Preparation**: Data cleaning and feature engineering.
- **Analytics Frameworks**: Scikit-learn for predictive modeling and NumPy for computational efficiency.
- **Visualization**: Use of libraries such as Matplotlib and Seaborn.

4.5 User Interface and Dashboard

- **Visualization Tools**: Tableau and Power BI for creating dynamic dashboards.
- **Dashboard Features**:
  - Interactive graphs and charts.
  - Filtering by regions, dates, and financial metrics.
  - Drill-down capabilities to explore granular data.

5 Technical Stack and Justification

- **Programming Languages**: Python for data manipulation, JavaScript for web components.
- **Frameworks**: Flask for backend API services, Tableau for visual dashboards.
- **Data Storage**: MySQL for structured data and Amazon S3 for unstructured backups.
- **Why This Stack?** Python's rich data libraries and integration with visualization tools make it ideal for this project. Tableau offers superior visualization capabilities for end-users.

6 Data Flow and Interaction Diagrams

6.1 Data Flow

- **Input**: Data is ingested through Python scripts from CSV files and API calls.
- **Processing**: ETL operations standardize data before it's moved to MySQL.
- **Output**: Final processed data is sent to Tableau for visualization.

6.2 Interaction Diagram

Depict the relationship between users, data processing tools, and the visualization dashboard.

7 Component Descriptions

7.1 Data Sources and Ingestion

- **Automated Fetching**: Uses Python and Pandas to import CSV and database data.
- **Error Handling**: Real-time checks to ensure data integrity.

7.2 Transformation Logic and Tools

- **Data Normalization**: Standardizing formats across various input sources.
- **Feature Engineering**: Creating derived features such as percentage change over periods.

7.3 Storage Solutions

- **SQL Tables**: Segregated by expenditure categories.
- **S3 Buckets**: Organized by fiscal year and data type for easy access.

7.4 Data Processing Frameworks

- **Batch Processing**: Regular batch jobs for updates using Apache Airflow.
- **Real-Time Processing**: Flask-based microservices for quick data retrieval.

8 Security and Compliance

8.1 Data Protection

- **Encryption**: SSL/TLS for data in transit, AES for data at rest.
- **Access Controls**: Role-based access for dashboard users.

- **Compliance**: GDPR and relevant financial data regulations.

8.2 User Authentication

Integrated OAuth for secure access and user verification.

9 Performance and Optimization Strategies

9.1 Load Management

- **Load Balancing**: Distribute data processing across multiple servers.
- **Indexing**: SQL indexing for fast retrieval.

9.2 Caching Mechanisms

- **Redis**: Used for caching frequently accessed data to improve performance.

10 Scalability and Future Enhancements

10.1 Vertical and Horizontal Scaling

- **Cloud Services**: AWS autoscaling groups for server management.

10.2 Planned Enhancements

- Integration with AI for predictive expenditure analysis.
- Real-time notifications for anomalies in spending.

11 User Experience and Interface Design

11.1 Layout and Navigation

- Simple, clean UI with an emphasis on accessibility.
- Logical grouping of filters and display panels.

11.2 Accessibility Considerations

- **Color Contrast**: High contrast mode for visually impaired users.
- **Keyboard Navigation**: Ensures the dashboard is operable without a mouse.

12 Challenges Faced and Solutions Implemented

12.1 Challenge: Data Consistency

Solution: Implementation of validation scripts during ETL.

12.2 Challenge: Load Times

Solution: Use of data prefetching and indexing.

13 Testing, Validation, and Quality Assurance

13.1 Testing Frameworks

- **PyTest**: For unit testing ETL scripts.
- **Selenium**: For UI testing in the dashboard.

13.2 Validation Checks

- **Data Consistency**: Comparison against benchmark datasets.
- **User Testing**: Pilot testing with real financial analysts.

14 Deployment Strategy and CI/CD Pipeline

14.1 Deployment Pipeline

- **Tools**: Jenkins and GitHub Actions for automated builds and deployments.
- **Staging Environment**: AWS EC2 for pre-deployment testing.

15 Monitoring, Maintenance, and Support

15.1 Monitoring Tools

- **Prometheus and Grafana**: For tracking system health and data flow.

15.2 Maintenance Strategy

Scheduled database optimizations and software updates.

16 Conclusion

The "Expenditure Data Analysis" project architecture provides a comprehensive framework for financial insights at Physics Wallah, ensuring data integrity, efficiency, and ease of use for decision-making.