Data Analysis Management System

Ghazi Lassoud Mohamed Daoud Skander Challouf Dhia Zrelli

Tunis Business School

January 26th, 2025

Agenda

- Project Overview
- Functional Non-Functional Requirements
- System Architecture
- Class Structure Relationships (UML)
- Technology Stack
- Live Demo (Postman)
- Code Highlights
- Challenges Solutions
- Project Results
- Future Plans
- Q&A

Project Overview

- Goal: monitor projects progress, their budgets, status.
- Stakeholders: Project Manager, Project Officers
- Uses: Java, Spring Boot, PostgreSQL, OOP principles.
- Provides: User-friendly API for data management, analysis, and visualization.

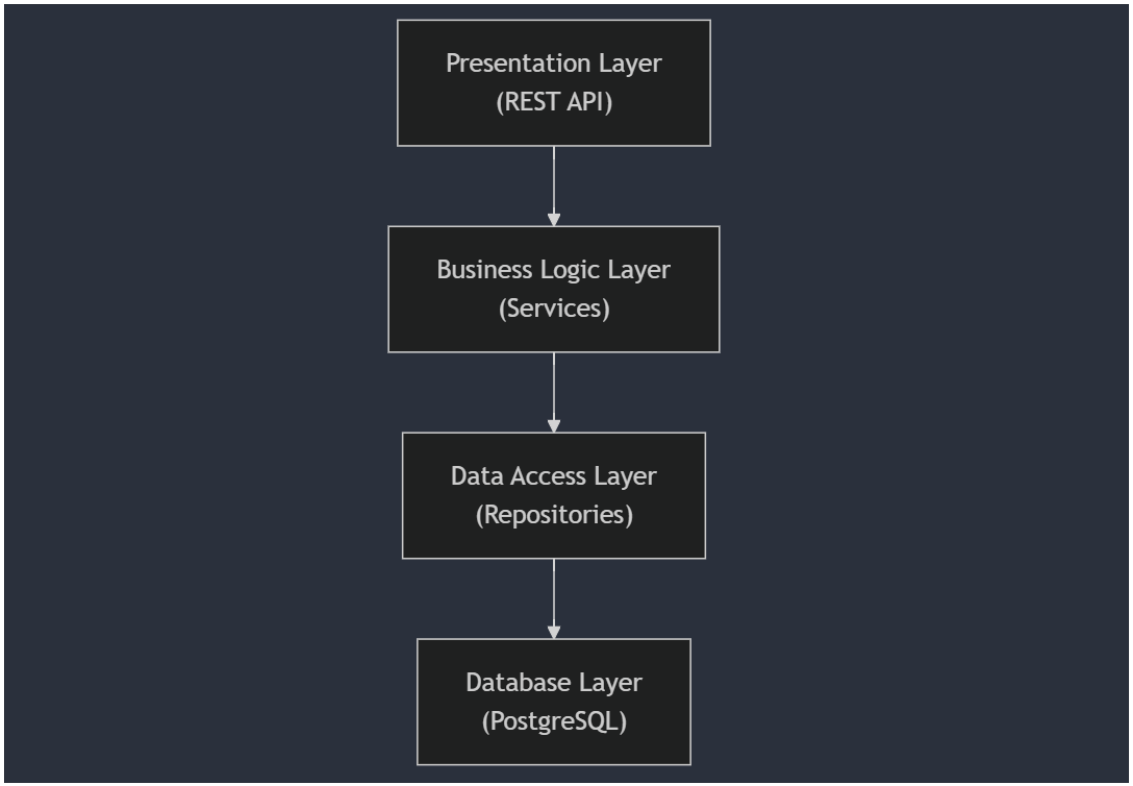
Functional Requirements

- **Data Ingestion**
  - CSV file import (OpenCSV)
  - Data validation
- **Data Storage**
  - PostgreSQL database
  - Spring Data JPA
- **Data Analysis**
  - Mean budget calculation
- **Data Visualization**
  - Bar charts (JFreeChart)
- **System Management**
  - RESTful API endpoints

Non-Functional Requirements

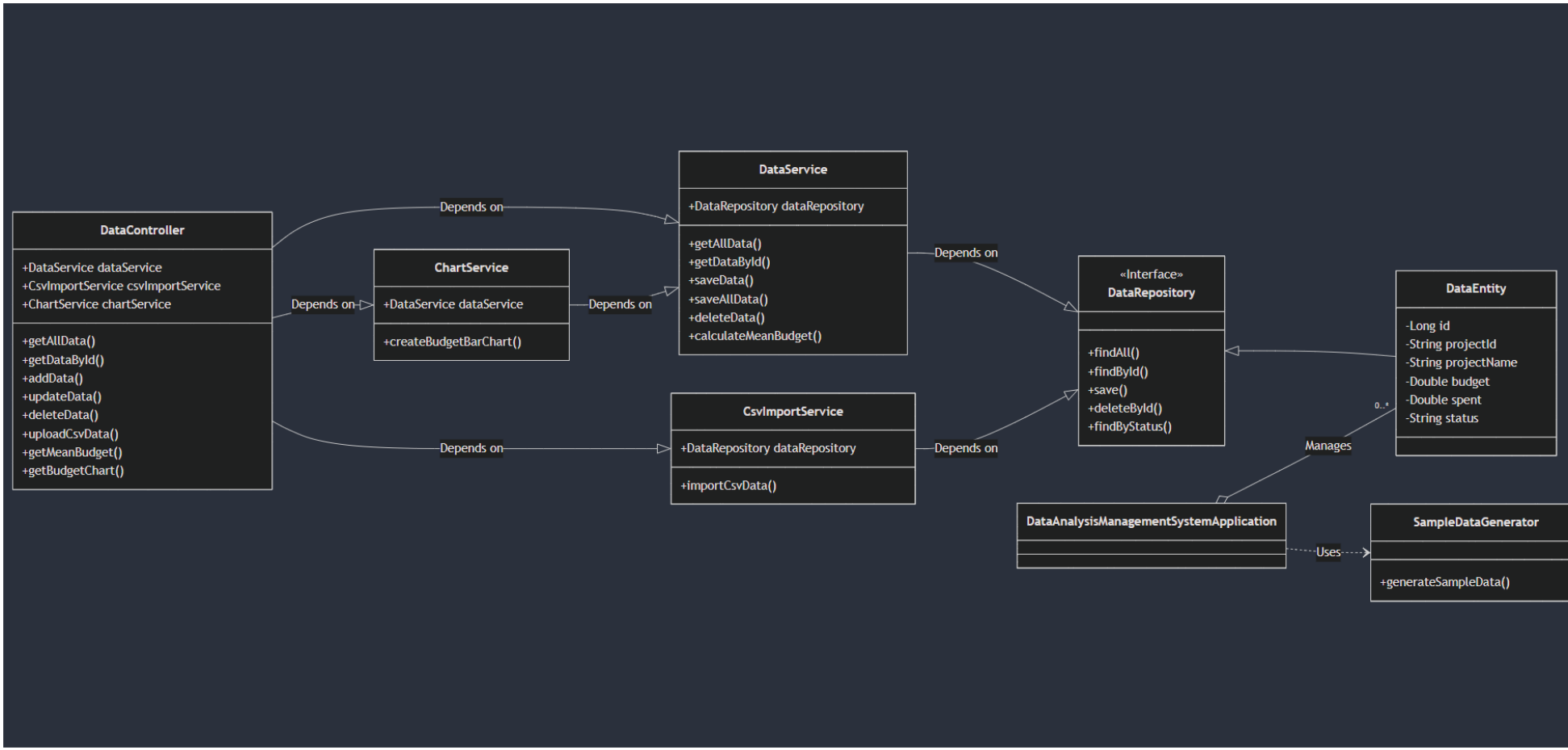
Requirement	**Implementation Status**
User-friendliness	REST API with Postman documentation
Scalability	Layered architecture
Security	Basic input validation
Maintainability	OOP principles applied

System Architecture



- **Presentation Layer:** REST API (Controllers)
- **Business Logic Layer:** Services (DataService, CsvImportService, ChartService)
- **Data Access Layer:** Repositories (DataRepository)
- **Database Layer:** PostgreSQL

Class Structure & Relationships (UML)



Class Specifications

- **DataEntity (Model)**: Represents data records (id, projectId, projectName, budget, spent, status).
- **DataRepository (Interface)**: Spring Data JPA repository for CRUD operations.
- **DataService**: Handles business logic, validation, and calculations.
- **CsvImportService**: Parses CSV files, transforms data, and inserts into the database.
- **ChartService**: Generates data visualizations using JFreeChart.
- **DataController**: Manages REST API endpoints and handles HTTP requests.

Object-Oriented Programming (OOP) Principles

- **Encapsulation**: Data and methods are encapsulated within classes (e.g., `DataEntity`).
- **Inheritance**: `DataRepository` inherits from `JpaRepository`.
- **Polymorphism**: Spring Data's method name parsing, `JpaRepository` implementation.
- **Abstraction**: Repository interface hides database implementation; service classes abstract business logic.

Technology Stack

Layer	**Technologies**
Presentation	Spring Boot Web, REST
Business Logic	Java 17, JFreeChart, Java Faker
Data Access	Spring Data JPA, Hibernate
Database	PostgreSQL
Build Tool	Maven

Live Demo (Postman)

- Demonstrate API endpoints using Postman.
- Show how to:
  - Add data (POST request)
  - Retrieve data (GET requests)
  - Generate the chart
  - Calculate the mean budget
- **Open Postman and switch to the application.**

Code Highlight: DataEntity

Listing 1: DataEntity.java

```java
@Entity
@Table(name = "data_entries")
public class DataEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    // ... other attributes (projectId, projectName, etc.)

    // Getters and setters (encapsulation)
    // ...
}
```

Code Highlight: DataRepository

Listing 2: DataRepository.java

```
@Repository
public interface DataRepository extends JpaRepository<
    DataEntity, Long> {
    List<DataEntity> findByStatus(String status);
}
```

Code Highlight: DataService (Validation)

Listing 3: DataService.java

```
public DataEntity saveData(DataEntity dataEntity) {
    if (dataEntity.getProjectName() == null || dataEntity.
        getProjectName().trim().isEmpty()) {
        throw new IllegalArgumentException("Project Name
            cannot be empty");
    }
    // ... other validation rules
    return dataRepository.save(dataEntity);
}
```

Code Highlight: ChartService

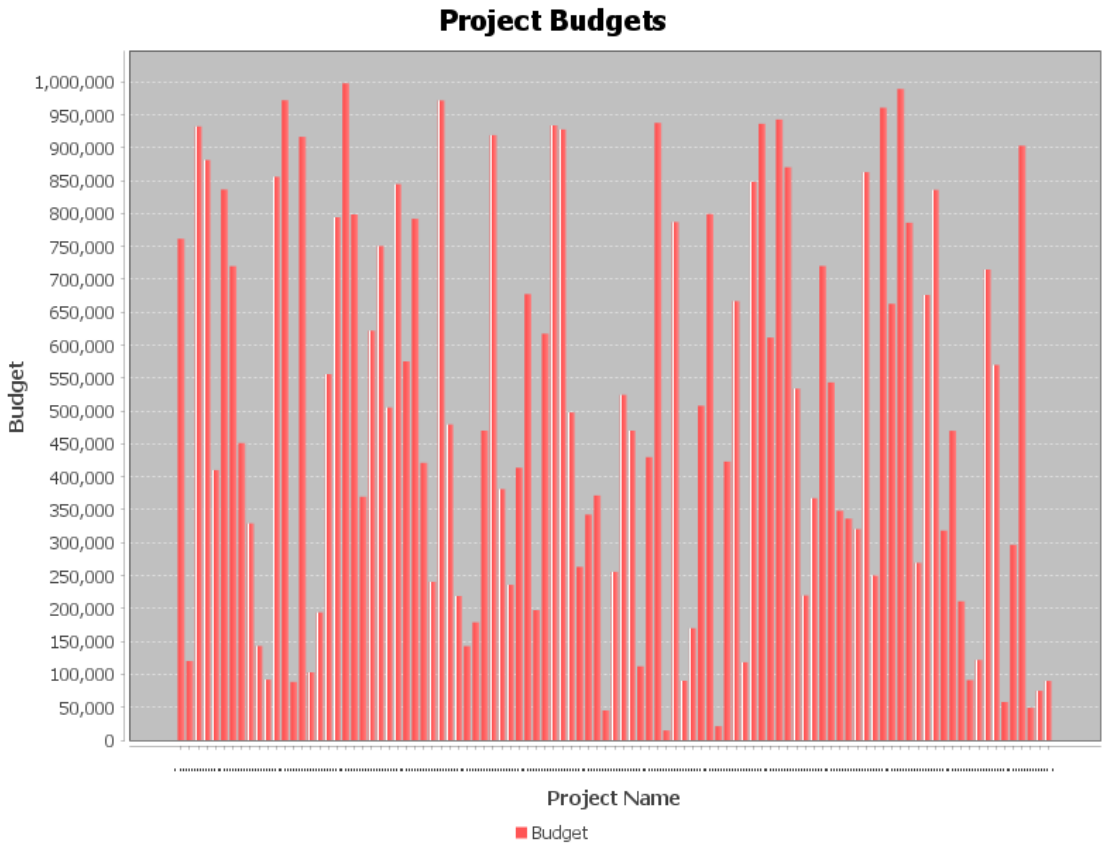
Listing 4: ChartService.java

```java
@Service
public class ChartService {
    // ...
    public JFreeChart createBudgetBarChart() {
        List<DataEntity> data = dataService.getAllData();
        DefaultCategoryDataset dataset = new
            DefaultCategoryDataset();

        for (DataEntity entity : data) {
            dataset.addValue(entity.getBudget(), "Budget",
                entity.getProjectName());
        }

        return ChartFactory.createBarChart( /* ... */ );
    }
}
```

Challenges & Solutions

- **Challenge 1**: Setting up the development environment with multiple technologies.
- **Solution**: Used detailed documentation and online resources (e.g., Spring Boot tutorials, Stack Overflow).
- **Challenge 2**: Understanding and applying Spring Data JPA.
- **Solution**: Experimented with different repository methods and practiced with examples.
- **Challenge 3**: Implementing data validation logic.
- **Solution**: Added custom validation in the `DataService` class.

Project Results



- These visual representations effectively demonstrated how the system successfully transformed raw data into actionable insights and comprehensive visualizations, showcasing its efficiency and reliability in achieving the project's objectives.

Future Plans

- **Enhanced Data Analysis**:
  - Add median/mode calculations.
  - Implement correlation analysis.
- **Security Features**:
  - Add Spring Security.
  - Implement JWT authentication.
- **Advanced Visualization**:
  - Time series charts.
  - Interactive dashboards.
- **Machine Learning**:
  - Implement prediction models.

Q&A

Questions?

Thank You!