**Architecture**

**1. Conceptual Architecture**

This view provides a high-level overview of the system. It should outline the major components and their relationships without going into implementation details. Use diagrams to illustrate the structure.

**Components:**

* **Frontend Layer**: Thymeleaf templates for user interaction.
* **Business Logic Layer**: Implements application logic and uses services like EventService and LocationService.
* **Data Layer**: Repositories for persisting and retrieving data (EventRepository, LocationRepository).
* **External Data Source**: Integration with the URL providing stock market data.
* **Pipe and Filter Components**: Process and filter stock market data.
* **Authentication and Authorization**: Ensure secure access for users.

**Diagram:**

Create a conceptual diagram showing:

* Users interacting with the system.
* The separation of concerns into frontend, backend, and data layers.
* External integrations and their flow into the system.

**2. Execution Architecture**

This focuses on how the application will be deployed and executed. Include details about distributed architecture, microservices, and containerization.

**Components:**

* **Frontend Application**: Deployed on a web server.
* **Backend Services**:
  + Microservices for event management, location management, and stock market data processing.
  + Each service containerized using Docker.
* **Database**: A centralized or distributed database (e.g., PostgreSQL).
* **Data Processing Pipelines**: Use microservices with Pipe and Filter architecture to process stock market data.
* **Deployment Platform**: Use Docker Compose or Kubernetes for managing containers.

**Diagram:**

* Illustrate containerized components (e.g., Docker containers for services).
* Show how services communicate using REST APIs.
* Highlight distributed architecture using microservices.

**3. Implementation Architecture**

This view provides technical details about how the components will be implemented and integrated.

**Details:**

* **Frontend**:
  + Thymeleaf templates for rendering views like listEvents.html and bookingConfirmation.html.
  + Integration with Spring Controllers (e.g., EventController).
* **Backend**:
  + Spring Boot application for handling business logic.
  + REST APIs for microservices communication.
* **Data Layer**:
  + Repositories for interacting with the database.
  + Integration of Pipe and Filter components to process external stock market data.
* **Security**:
  + Implement authentication/authorization with Spring Security.
* **Containerization**:
  + Dockerfiles for each microservice.
  + Use a docker-compose.yml file to manage containers.

**Diagram:**

* Class diagrams for major backend components (e.g., **EventService**, **EventController**, **EventRepository**).
* Flow diagrams showing integration of frontend, backend, and external services.
* Sequence diagrams for critical use cases like event booking or data processing.