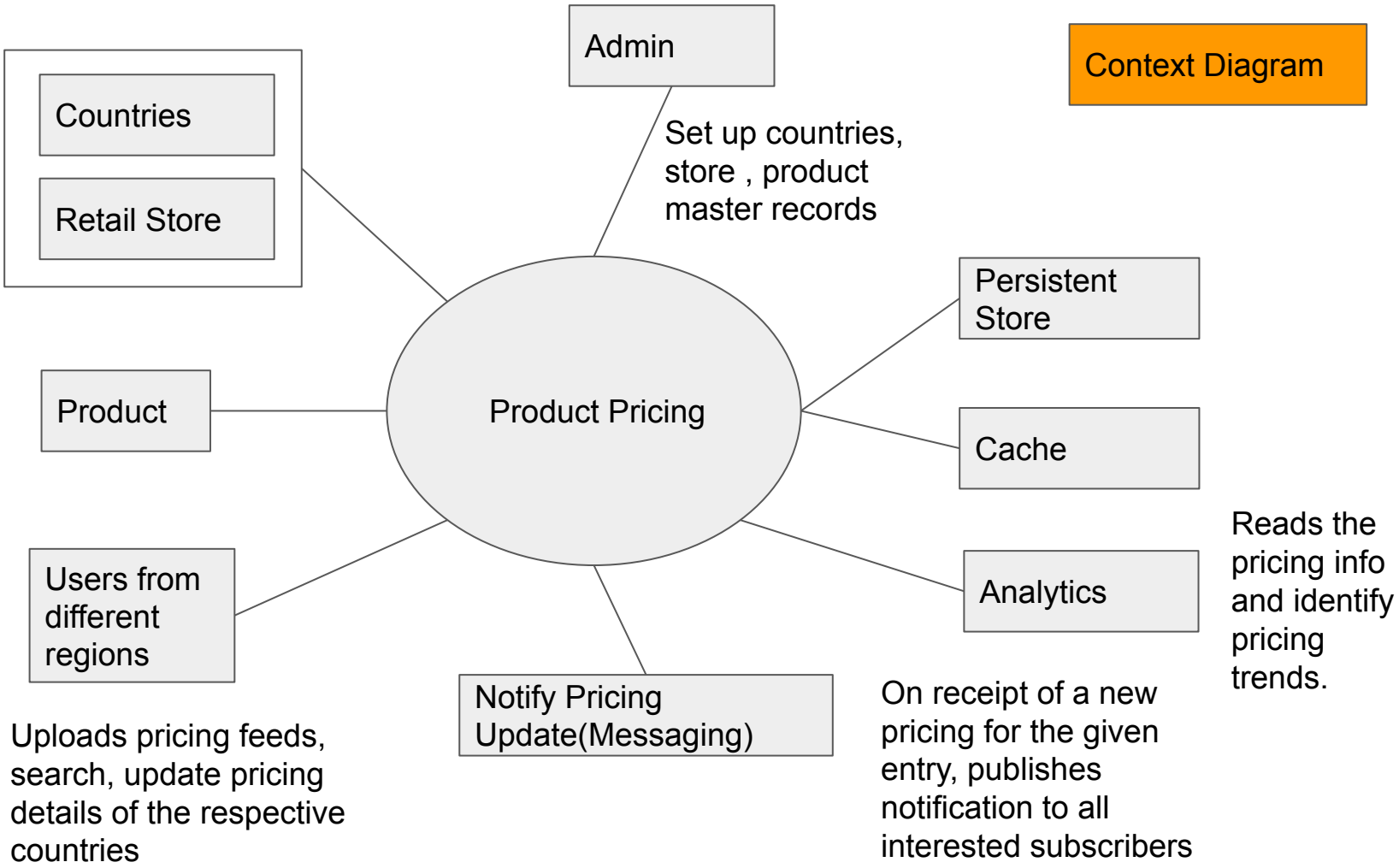
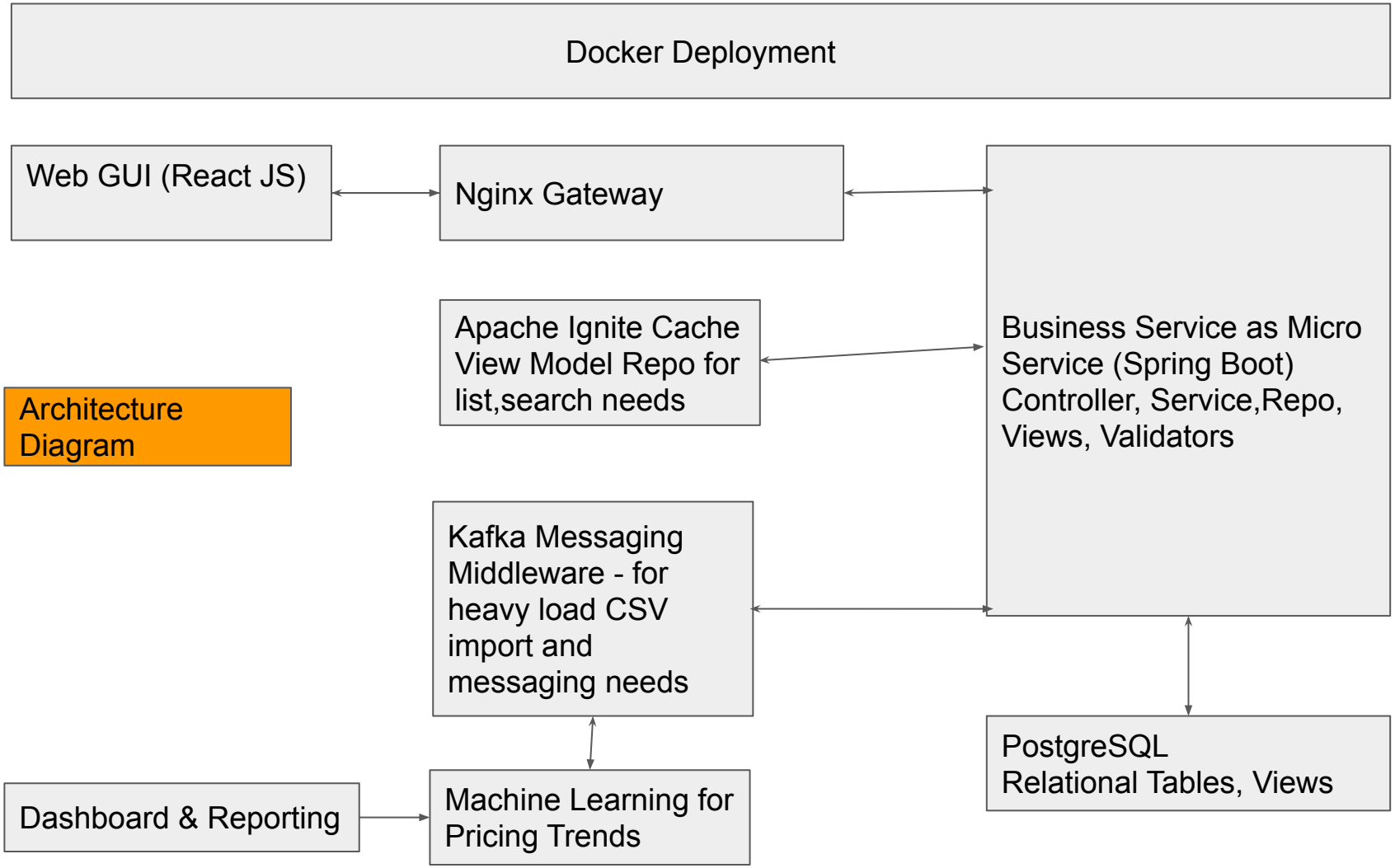
Case Study on Product Pricing

Govindan Neelagiri Subramanian


Context Diagram
Admin
Set up countries, store , product master records
Countries
Retail Store
Persistent Store
Product
Product Pricing
Cache
Reads the pricing info and identify pricing trends.
Users from different regions
Analytics
Notify Pricing Update(Messaging)
Uploads pricing feeds, search, update pricing details of the respective countries
On receipt of a new pricing for the given entry, publishes notification to all interested subscribers


Docker Deployment
Web GUI (React JS)
Nginx Gateway
Apache Ignite Cache
View Model Repo for
list,search needs
Business Service as Micro
Service (Spring Boot)
Controller, Service,Repo,
Views, Validators
Architecture
Diagram
Kafka Messaging
Middleware - for
heavy load CSV
import and
messaging needs
PostgreSQL
Relational Tables, Views
Dashboard & Reporting
Machine Learning for
Pricing Trends

Design Decisions

- Web GUI as the frontend of choice
  - From the ground up built without any UI toolkit, for better long term support.
- Response Web GUI layout
- i18n support to cater different locales(language, date,etc)
- Ad-hoc querying support to cater any future filtering needs
  - For example price range
- For better user experience/ quick turnaround time, view model cache is introduced
- The list will display only one row per product for that store, running pricing feeds will be discarded and wont persisted, it will be forwarded to kafka for analytics purpose
- User will be attached to a country and will NOT be able to view other country pricing

Non functional requirements considered

- Usability
  - Web GUI is the mostly used frontend for enterprise grade software that provide better experience
- Security
  - React JS avoids XSS attacks,
  - All inputs sanitized by service before store.
  - Nginx could be configured to set different content security policies.
  - Application could use a token for unique client identification to avoid request tampering
- Performance
  - Apache ignite cache helps for better search results
- Scalability
  - Micro-service/docker deployments supports horizontal scalability
  - Query design is implemented in such a way that cater most of the future needs
- Maintenance
  - Minimal npm modules for frontend development for better maintenance
  - Spring Boot avoids boilerplate coding, enhances developer productivity

Assumptions

- Admin user already creates the master entries for product pricing
- Pricing feeds will be huge and concurrent from different countries
- Only the recent product price is considered for any monetary calculations