Online Movie Ticket Booking Platform

Online movie ticket booking platform that caters to both B2B (theatre partners) and B2C (end customers) clients

Brief steps towards design

Being Proactive

Step that involves exhaustive discussion for requirements and thinking wide across all corners of all possible use cases

Requirements

Enable theatre partners to onboard their theatres over this platform and get access to a bigger customer base while going digital.

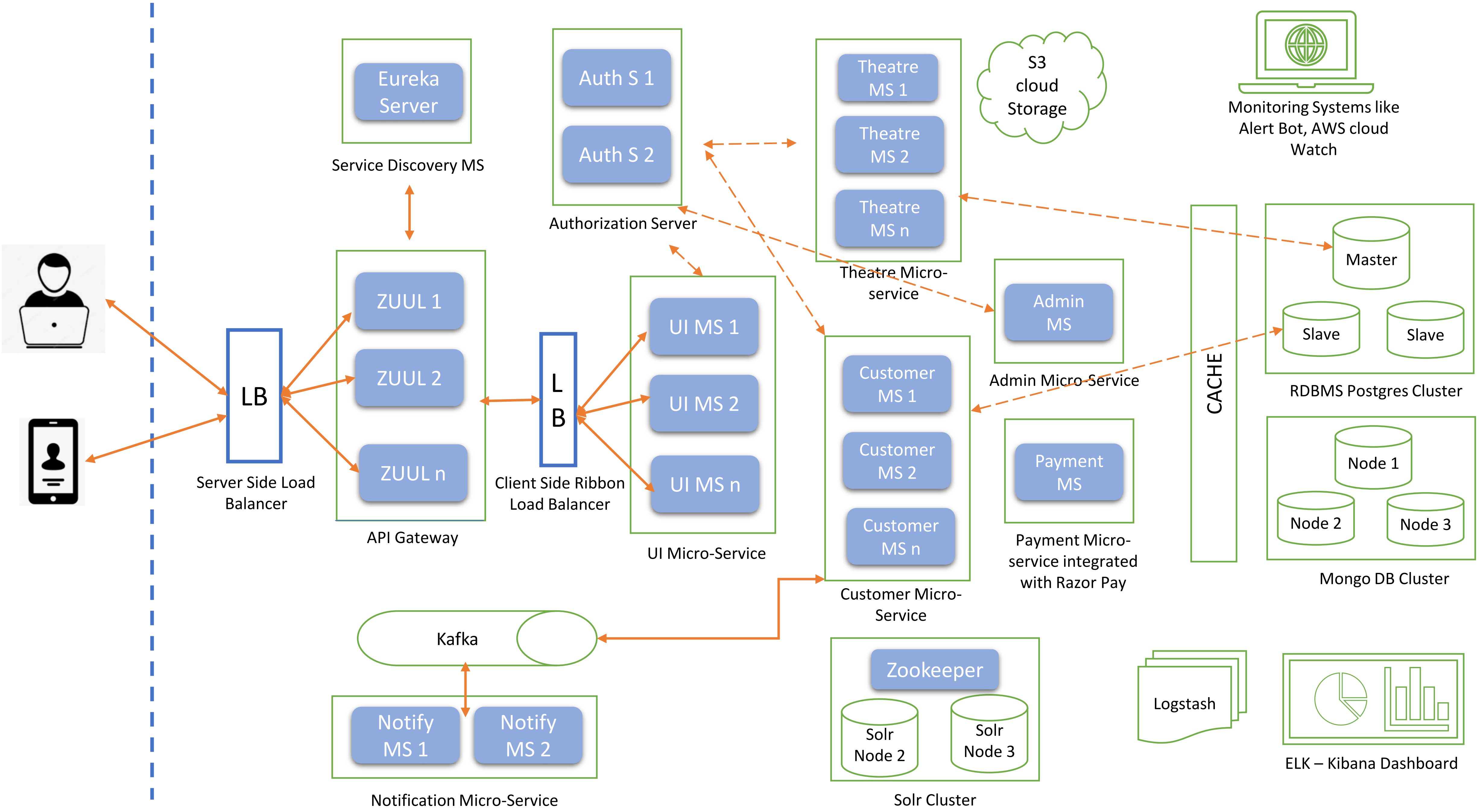
- Theatre partners shall be able to login to the system and should have privileges to create, update, read and delete their data like theatre, screens, shows, upcoming shows, seating categories, offers, etc.
- They shall be able to publish the future engagements as well as track history for their transactions on platform.
- They shall be able to update seat availability and their prices at runtime.
- System should have ability to integrate APIs exposed from existing IT systems of the existing theatres.

Enable end customers to browse the platform to get access to movies across different cities, languages, and genres, as well as book tickets in advance with a seamless experience.

- Any user shall be able to visit platform from any device like mobile, laptop, desktop over internet.
- They should be able to browse all the shows, theatres by location, show timings, etc. search criteria without needing to login to the system.
- They should be able to select the movie show for which they want to book the tickets, should be able to register themselves into system and book the ticket by paying online.
- They should be able to like, comment, share on the show also they should be able to see their previous history on the platform.

Begin with End In Mind

Step that defines our final milestone



Eureka Server

Service Discovery MS

Auth S 1

Auth S 2

Authorization Server

Theatre MS 1

Theatre MS 2

Theatre MS n

Theatre Micro-service

S3 cloud Storage

Monitoring Systems like Alert Bot, AWS cloud Watch

LB

Server Side Load Balancer

ZUUL 1

ZUUL 2

ZUUL n

API Gateway

L B

Client Side Ribbon Load Balancer

UI MS 1

UI MS 2

UI MS n

UI Micro-Service

Admin MS

Admin Micro-Service

Customer MS 1

Customer MS 2

Customer MS n

Customer Micro-Service

Payment MS

Payment Micro-service integrated with Razor Pay

CACHE

Master

Slave

Slave

RDBMS Postgres Cluster

Node 1

Node 2

Node 3

Mongo DB Cluster

Kafka

Notify MS 1

Notify MS 2

Notification Micro-Service

Zookeeper

Solr Node 2

Solr Node 3

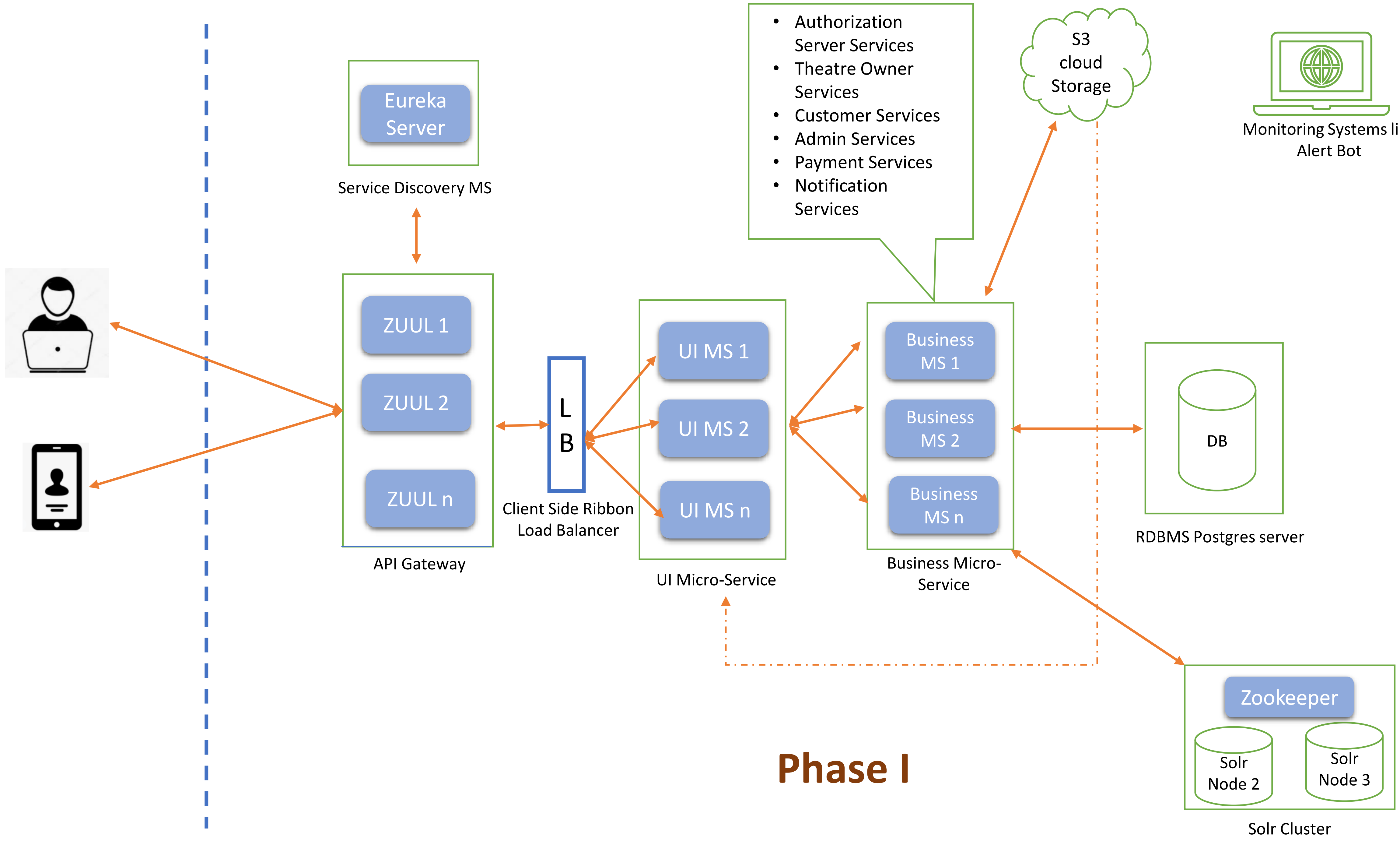
Solr Cluster

Logstash

ELK – Kibana Dashboard

Put First Things First

Step that defines achievable milestone within given timelines



Eureka Server
Service Discovery MS
- Authorization Server Services
- Theatre Owner Services
- Customer Services
- Admin Services
- Payment Services
- Notification Services
S3 cloud Storage
Monitoring Systems like Alert Bot
ZUUL 1
ZUUL 2
ZUUL n
API Gateway
L B
Client Side Ribbon Load Balancer
UI MS 1
UI MS 2
UI MS n
UI Micro-Service
Business MS 1
Business MS 2
Business MS n
Business Micro-Service
DB
RDBMS Postgres server
Zookeeper
Solr Node 2
Solr Node 3
Solr Cluster

Phase I

Technology and Frameworks

Spring Boot

Spring Cloud Netflix OSS

- Eureka Server
- ZUUL API Gateway
- Config Server
- Ribbon Load Balancer
- Hystrix Circuit Breaker
- Feign Client

Spring Framework

- Spring MVC
- Spring Security
- Spring AOP
- Spring thymeleaf template Engine

User Interface

- HTML
- CSS
- JavaScript
- jQuery
- Bootstrap 4

Database

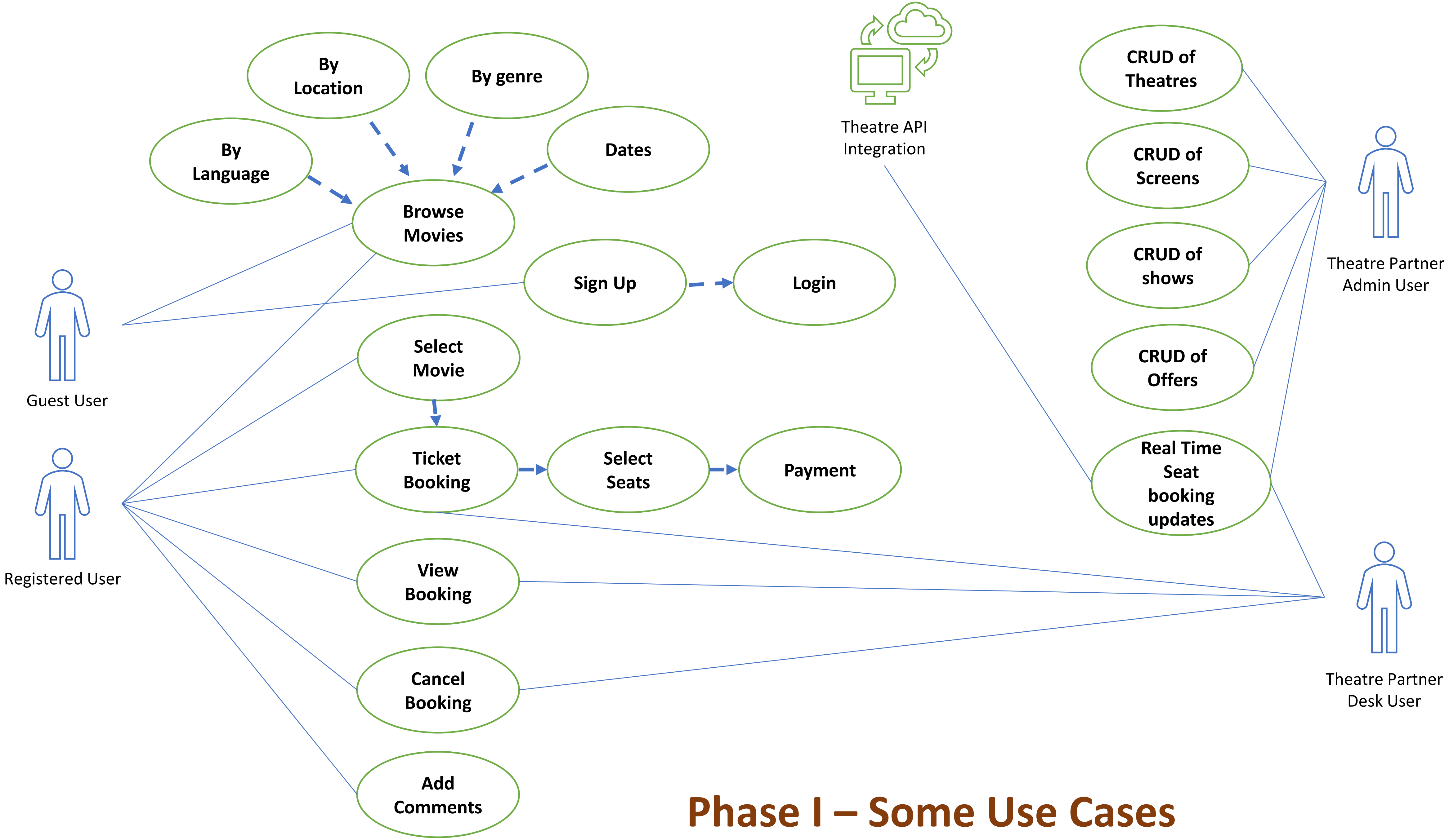
- Postgres SQL

Cloud Resources

- Simple Storage Service (S3)
- EC2 Instances
- RDS Postgres Instance

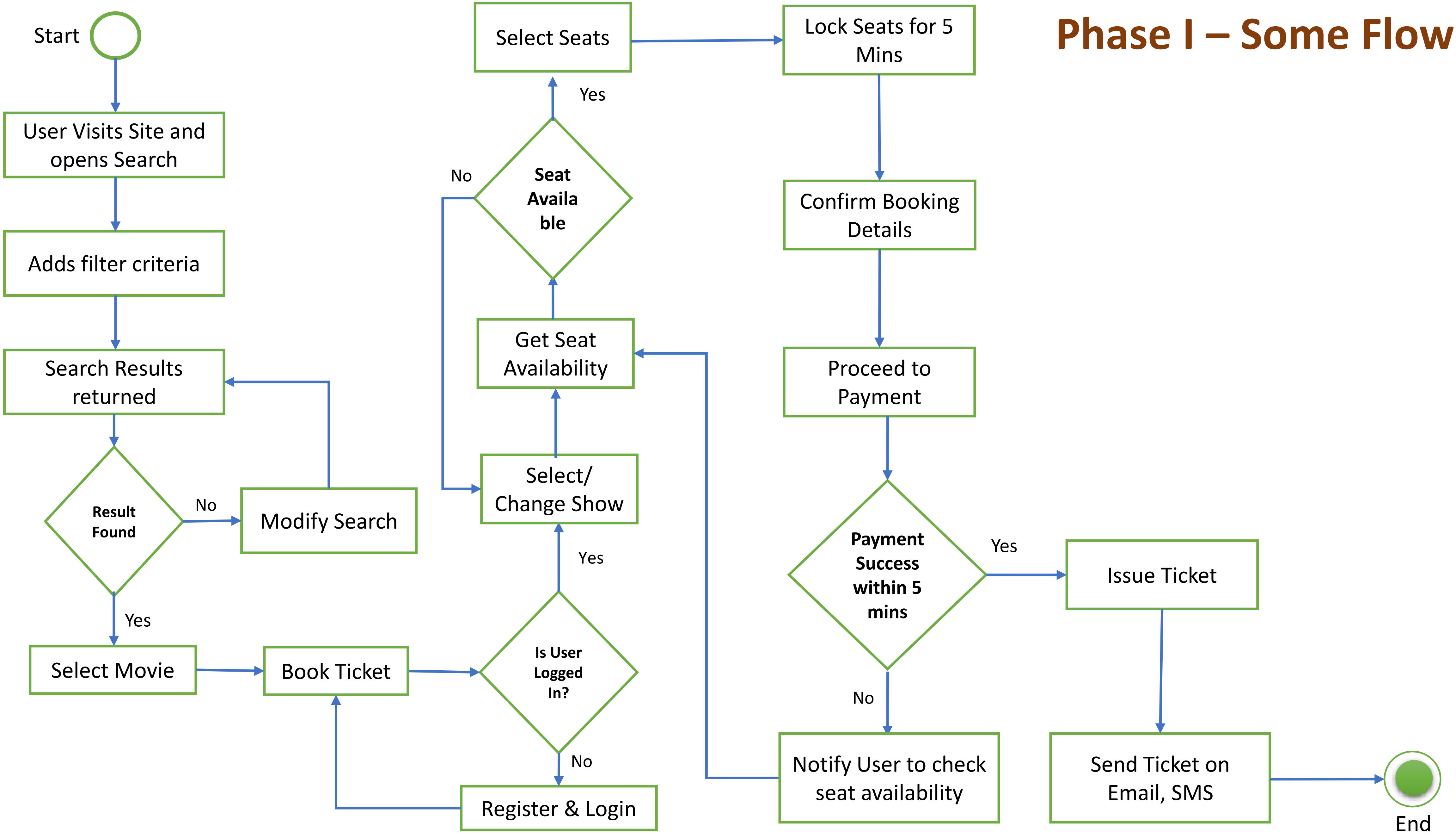
Build Tool

- Gradle
- Batch Files
- SH files



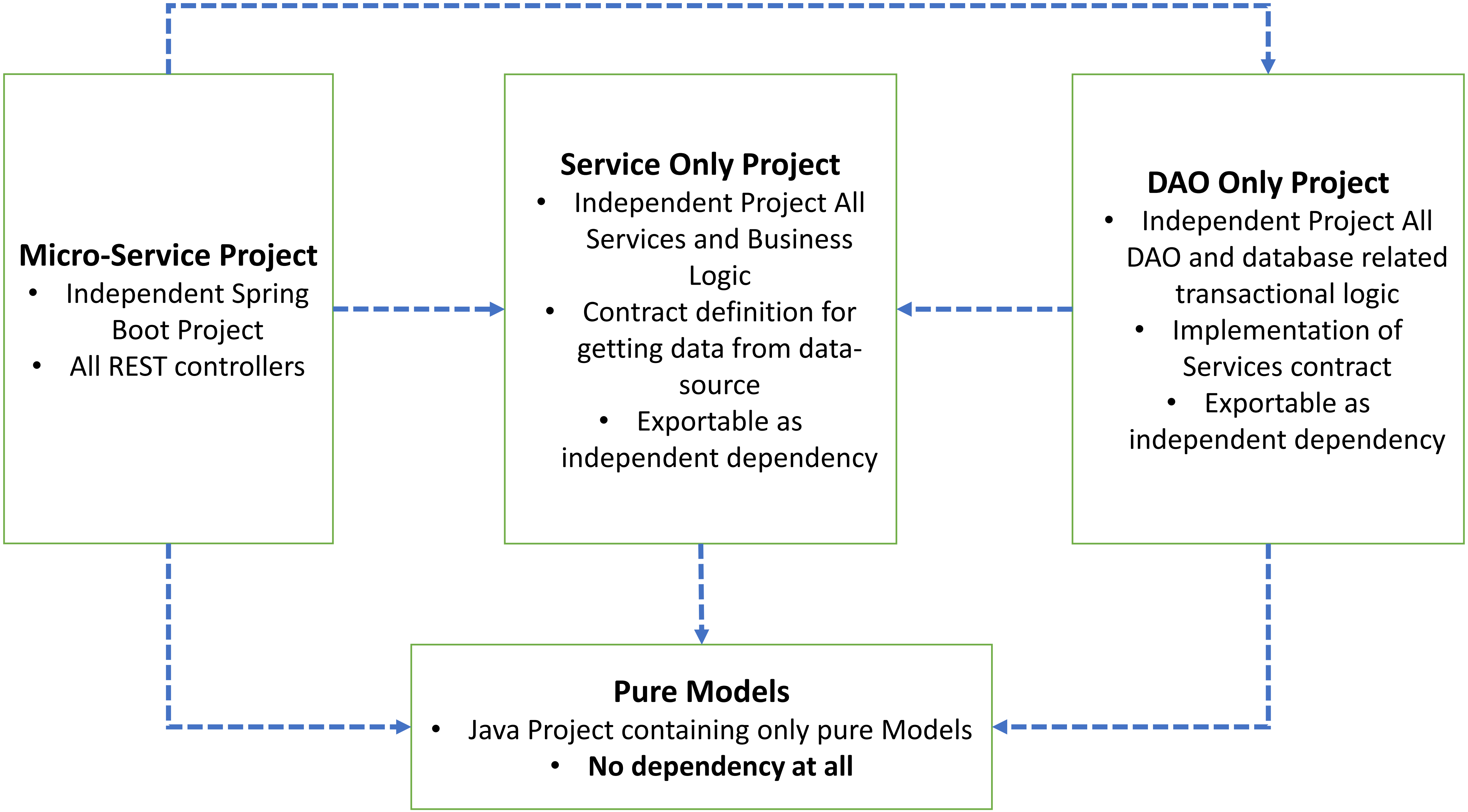
Phase I – Some Use Cases

Phase I – Some Flows



Micro-Modules with Purity of Models

Step that introduces unique concept and set of principles for designing a single unit of Micro-service



Micro-Service Project

- Independent Spring Boot Project
- All REST controllers

Service Only Project

- Independent Project All Services and Business Logic
- Contract definition for getting data from data-source
- Exportable as independent dependency

DAO Only Project

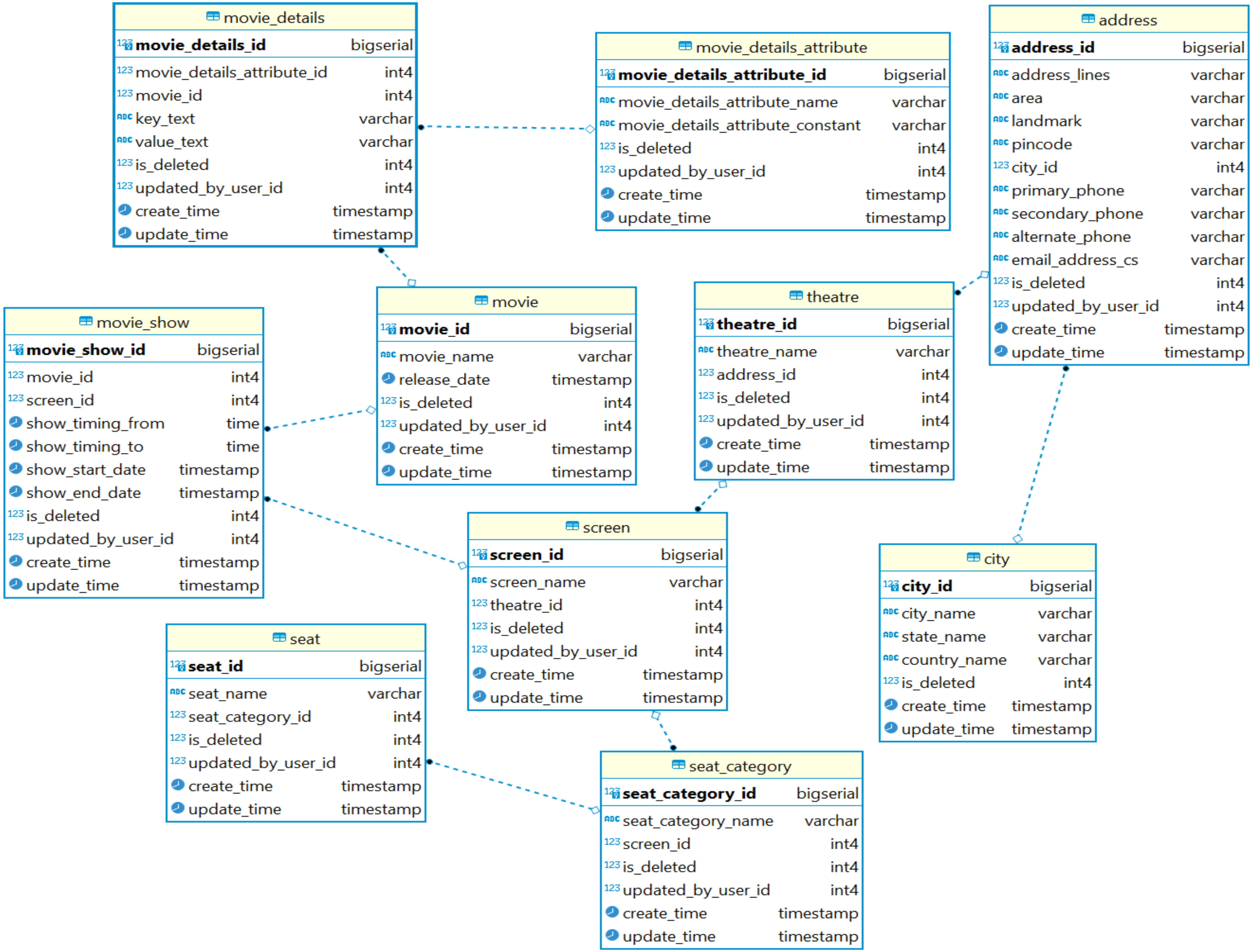
- Independent Project All DAO and database related transactional logic
- Implementation of Services contract
- Exportable as independent dependency

Pure Models

- Java Project containing only pure Models
- **No dependency at all**

Core DB design

Core business related tables



movie_details	
movie_details_id	bigserial
movie_details_attribute_id	int4
movie_id	int4
key_text	varchar
value_text	varchar
is_deleted	int4
updated_by_user_id	int4
create_time	timestamp
update_time	timestamp

movie_details_attribute	
movie_details_attribute_id	bigserial
movie_details_attribute_name	varchar
movie_details_attribute_constant	varchar
is_deleted	int4
updated_by_user_id	int4
create_time	timestamp
update_time	timestamp

address	
address_id	bigserial
address_lines	varchar
area	varchar
landmark	varchar
pincode	varchar
city_id	int4
primary_phone	varchar
secondary_phone	varchar
alternate_phone	varchar
email_address_cs	varchar
is_deleted	int4
updated_by_user_id	int4
create_time	timestamp
update_time	timestamp

movie_show	
movie_show_id	bigserial
movie_id	int4
screen_id	int4
show_timing_from	time
show_timing_to	time
show_start_date	timestamp
show_end_date	timestamp
is_deleted	int4
updated_by_user_id	int4
create_time	timestamp
update_time	timestamp

movie	
movie_id	bigserial
movie_name	varchar
release_date	timestamp
is_deleted	int4
updated_by_user_id	int4
create_time	timestamp
update_time	timestamp

theatre	
theatre_id	bigserial
theatre_name	varchar
address_id	int4
is_deleted	int4
updated_by_user_id	int4
create_time	timestamp
update_time	timestamp

screen	
screen_id	bigserial
screen_name	varchar
theatre_id	int4
is_deleted	int4
updated_by_user_id	int4
create_time	timestamp
update_time	timestamp

city	
city_id	bigserial
city_name	varchar
state_name	varchar
country_name	varchar
is_deleted	int4
create_time	timestamp
update_time	timestamp

seat	
seat_id	bigserial
seat_name	varchar
seat_category_id	int4
is_deleted	int4
updated_by_user_id	int4
create_time	timestamp
update_time	timestamp

seat_category	
seat_category_id	bigserial
seat_category_name	varchar
screen_id	int4
is_deleted	int4
updated_by_user_id	int4
create_time	timestamp
update_time	timestamp