SPEEDYCHEERS

Architecture

Author

Prashanth Kumar

Contents

Business Requirement

Business proposes a new product called **SpeedyCheers** through which customers can request for a Cheer. A Cheerer goes to customer location and cheers them by laughing for jokes cracked by customer or encourages them in whatever the tasks they are doing. Cheerer gets paid for his service based on the pre-defined fee model. SpeedyCheers works similar to Uber model where customers can request for a ride based on his location. Only difference here is, instead of a ride Cheerer visits customer and cheers them.

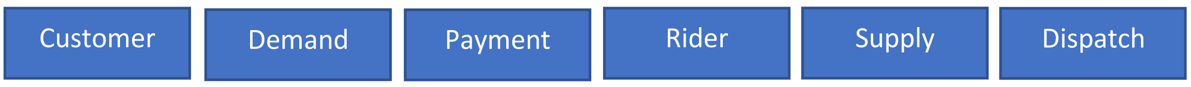
Proposed Solution

The Proposed solution is based on the micro-service architecture. Services are separated based on the bounded contexts of the domain and communicates with each other by using REST protocol or by using messaging backbone.

This way, organization can quickly benefit the following capabilities

- Scalability
- Availability
- Resiliency
- Independent
- Failure isolation
- Continuous testing & delivery

High level Domain/Sub Domain break-up with bounded contexts:



Sequence Diagram

The below sequence diagram shows the high-level interaction between customer and system.

Customer Service

Responsibility of this service is to - customer registration, profile management etc.

Rider Service

Responsibility if this service is to - Cheer Rider registration with various cheer services he/she can offer

Demand Service

Service takes the customer request and registers this request in dispatch system. It captures various details from customer like

- Location details
- Cheer requirement

Supply Service

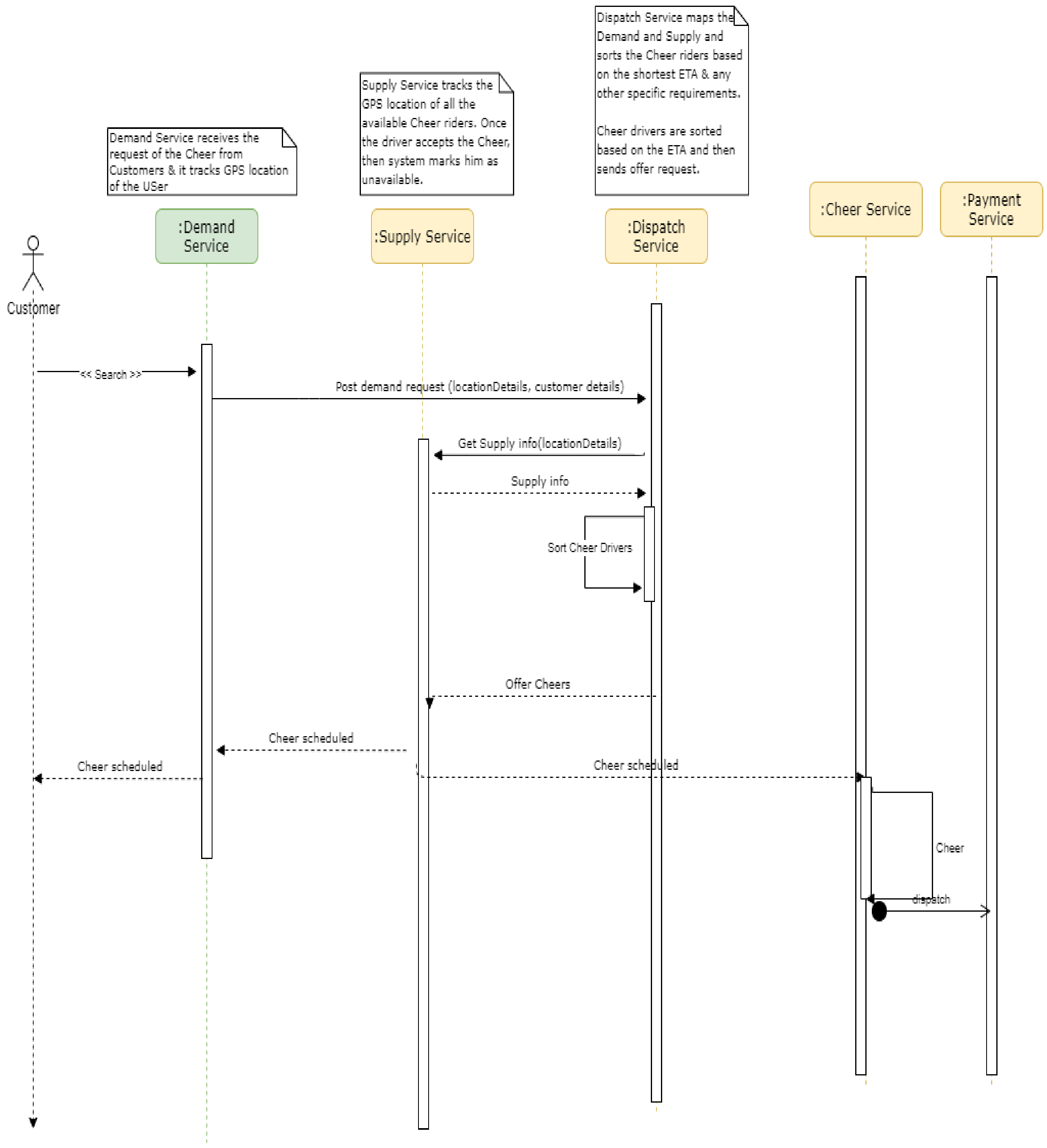
Responsibility of this service is to – capture the available Cheer rider details (in a pre-defined time frame like every 5 minutes) along with location details

Dispatch Service

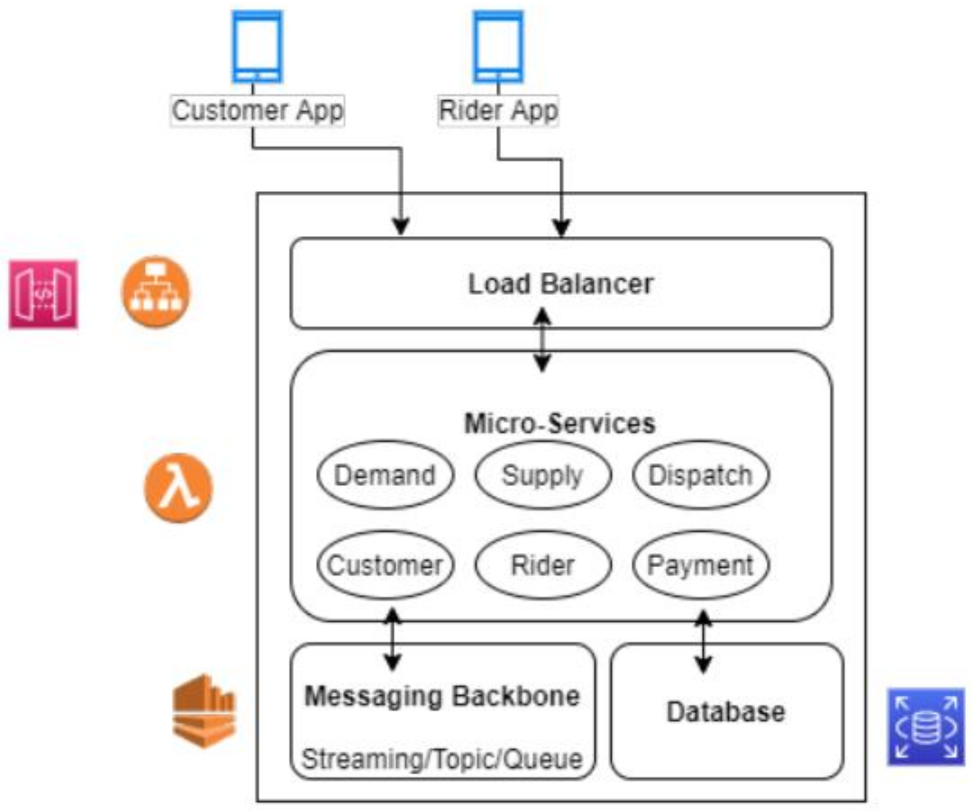
This is the core service which actually offers the cheer service to customer. When there is a new demand raised then the service will liaise with supply and sorts the available cheer drivers based on the location and any other special requirements. Once the Cheer driver details are sorted then it liaises with Supply service to offer a cheer to customer.

Payment Service

Responsible for calculating the fee for a cheer. Fee calculation is based on the pre-defined cost model like incremental, offer based etc.


Demand Service receives the request of the Cheer from Customers & it tracks GPS location of the USer
Supply Service tracks the GPS location of all the available Cheer riders. Once the driver accepts the Cheer, then system marks him as unavailable.
Dispatch Service maps the Demand and Supply and sorts the Cheer riders based on the shortest ETA & any other specific requirements.
Cheer drivers are sorted based on the ETA and then sends offer request.
:Demand Service
:Supply Service
:Dispatch Service
:Cheer Service
:Payment Service
Customer
<< Search >>
Post demand request (locationDetails, customer details)
Get Supply info(locationDetails)
Supply info
Sort Cheer Drivers
Offer Cheers
Cheer scheduled
Cheer scheduled
Cheer scheduled
Cheer
dispatch

Architecture Diagram



The above architecture diagram depicts the various components and interaction between those.

Customer/Rider Apps

Mobile apps installed in customer/Rider smartphones. Through which customer can raise a demand and track the order. At the same time, rider can offer a cheer ride.

Load Balancer

It makes routing decisions at the transport layer (TCP/SSL). It can handle millions of requests per second. After the load balancer receives a connection, it selects a target from the target group for the default rule using a flow hash routing algorithm.

Microservices

Microservices executing the business rules and algorithms. These services expose the REST endpoint and these endpoints can be configured in API gateway for better security. Traffic from load balancer can then route to API Gateway based on the pre-defined policy.

Messaging Backbone

High throughput messaging/streaming service offers various capabilities like streaming/topics/queue. This helps in building the distributed system. Communication between microservices can be either via REST endpoints or it can be routed via messaging backbone.

Database

Data persistency layer. At this moment RDBMS kind of database is considered in the solution.

Technology Map

Load Balancer & API Gateway
AWS Network Load Balancer
AWS API Gateway

Microservices
AWS Lambda

Messaging Backbone
AWS Kinesis – streaming platform

Database
Amazon RDS

Programming Language
NodeJs

Assumptions

- Location/Map services that provide the location details of customer and rider are deemed to be via third party services OR via separate services not depicted in the architecture
- Notification services and platform services (like logging, security) are out side of this MVP delivery
- Performance, Resilience, Disaster Recovery are considered as out of scope of this MVP delivery
- Mobile application, Reports, Admin modules are out of scope of this MVP delivery
- Continuous deployment, continuous delivery and continuous testing are out of scope of this MVP delivery