**Speedy Cheers Application**

High-level application architecture

Table of Contents

[Functional Requirements: 2](#_Toc161611258)

[Non-functional Requirements: 3](#_Toc161611259)

[Workflow of cheerleader booking steps: 3](#_Toc161611260)

[Services broad classification 3](#_Toc161611261)

[State changes services: 4](#_Toc161611262)

[Immutable services: 4](#_Toc161611263)

[Metrics logging for Business activity monitoring: 4](#_Toc161611264)

[Capacity estimations: 4](#_Toc161611265)

[High-level design 4](#_Toc161611266)

[Component diagram: 5](#_Toc161611267)

[State change diagram: 5](#_Toc161611268)

[Architecture Decisions: 6](#_Toc161611269)

[System design 7](#_Toc161611270)

[Service components: 7](#_Toc161611271)

[Geo-location service 7](#_Toc161611272)

[ETA service 7](#_Toc161611273)

[Matching Service 8](#_Toc161611274)

[Pricing Service 8](#_Toc161611275)

[Cheer Requestor and Cheer Leader Management Service 8](#_Toc161611276)

[Trip Management Service 9](#_Toc161611277)

[Payment Service 9](#_Toc161611278)

## Functional Requirements:

1. Cheers Requestor book rides from A to B now.
2. Cheers Rider reaches the requestor pickup place and performs the cheers tasks.

Assumptions:

1. Cheer Leader and Cheer Rider are the same.

Constraints:

1. Book only one cheer rider at a time.
2. Later time requesting the cheer leader is not in scope.
3. Cancel or modifying the cheers duration and location after start is not in scope.
4. Credit/debit cards information is not allowed to be stored.
5. Minimum price for the request for default distance and minimum time.

Cheers Requestor actions:

1. Book the Cheer Rider and provide the task duration.
2. Use GPS to locate the geo-location.
3. Provide Pickup location.
4. Cancel the ride before.

Cheer Leader actions:

1. Cheers leader can say he is available or not available.
   1. Not available: when the driver is in a lunch break or starting from home or going back home.
2. Receives the proposal for the cheers task.
   1. Pickup location
   2. Time to start now
   3. How long the whole cheers task takes placed
   4. Estimated price

## Non-functional Requirements:

1. Scalability: The app must handle millions of concurrent users, especially during peak demand, across various geographic locations.
2. High Availability: The rider app needs to be always accessible for booking rides. Downtime would be disastrous for the business.
3. Real-time Experience: Riders expect to see immediate updates on driver availability, location tracking, and estimated time of arrival (ETA).
4. Location-Awareness: Geospatial calculations and data form the core of Uber's functionality.

## Workflow of cheerleader booking steps:

1. Request for the cheerleader.
2. The application finds the cheers leader.
3. Check if the cheerleader is accepted in 10 seconds
4. else look for another cheerleader near by
5. if the cheerleader accept, then the pickup location is shared in real-time
   1. ETA will be provided according to the cheerleader's progress.
   2. The Cheer requestor will be notified about the ETA of the cheerleader.

## Services broad classification

The services are broadly divided into state change services and read-only services.

### State changes services:

In these services, there are chances of collision happening due to distribution in nature. So, the actions should be synchronized. If there are multiple requestors and the system is distributed logically it is a single machine.

The data should be strictly consistent always.

Example: Requestor to cheerleader matching state machine synchronization. Only one cheers leader to be selected for one requestor.

## Immutable services:

This service will request only read data, there is no live object state is maintained. So these services can be serverless and scalable.   
Ex: How many cheerleaders are available in a given location or region? A 5-second display of delay data is still ok. If it shows only a few cheerleaders and later more cheerleaders. These services provide eventually consistent data and are scalable due to no state management.

Synchronization of the data specific to a particular ride.

1. Share the data with the requestor and cheers the driver.
2. Every 10 seconds update the position of the cheers driver.
   1. What happens when the trip manager instance dies, which consists of active trip details, cheer rider position, and requestions pickup position?

## Metrics logging for Business activity monitoring:

1. Traffic numbers in every component.
2. The users should not see performance degradation at any time.
3. User-facing metrics.
   1. Number of cheerleaders required.
   2. I would look into the length of rides
   3. Average pickup time
   4. Short or longer rides
   5. A/B testing.

## Capacity estimations:

1. Traffic
2. Storage
3. Bandwidth

## High-level design

### Component diagram:

A diagram of a company

Description automatically generated

### State change diagram:

#### Rider State management:

A diagram of a flowchart

Description automatically generated

#### Cheer Leader state management:

A diagram of a cheer leader

Description automatically generated

## Architecture Decisions:

1. Use micro service architecture for customer-facing applications secure REST API.
2. Use WebSockets for user communication to keep the continuous communication between the system and the customers**.**
3. Use Cassandra to management the trip in real time
4. Use 3rd party payment service, do not write your own service.
5. Use 3rd party map service.
6. Inter micro services communication: gRPC + protobuf

## System design

A diagram of a diagram

Description automatically generated

## Service components:

### Geo-location service

In the SpeedyCheers app rider application, the geolocation service performs below actions

* When you open the app and request a ride, the geolocation service using GPS determines your exact pickup location. This allows SpeedyCheers app to accurately pinpoint you on a map for the driver.
* **Provides the location of Cheers requestor and Cheerleader that help matching service to identify the near by cheerleader.**
* **With the help of geolocation and the map details provide by location service will be used to calculate the ETA and Estimated price for the trip**
* **This service is used the cheerleader to navigate to the pickup location.**

### ETA service

The ETA (Estimated Time of Arrival) service in the SpeedyCheers app rider application is all about providing you with an informed timeframe for when your driver will arrive. Here's how it works: It calculates the ETA using the requestor location, traffic data, cheer leader location and shortest path.

During the cheerleader reaching the pickup destination, depending on the driver position the ETA will be re-calculated and shared with Cheers Requestor and Cheerleader.

The ETA service in the SpeedyCheers app rider application aims to help you plan your time accordingly and offers an estimation time when your SpeedyCheers app will arrive.

Redis Cache:

* Use Redis to update the cached ETA information whenever new traffic data or driver location updates are received. This ensures riders see the most recent estimated arrival time.
* Store the frequently accessed ETA of current ride in progress and this information is used to re-caculate ETA as per position changes and reduce number of calls to database.

### Matching Service

The matching service in the SpeedyCheers app rider application is the heart of connecting you with a driver quickly and efficiently. Here's how it works behind the scenes:

1. When you request a ride, your location and other details are sent to the matching service.
2. The service identifies a pool of nearby drivers who meet the matching criteria mentioned above.
3. The matching service analyze all the available cheerleaders near distance, reach quickly consider the distance and traffic.
4. The chosen driver receives a notification on their app with your pickup location and ride details..

### Pricing Service

The pricing service in the SpeedyCheers app rider app determines the fare you pay for your trip. It calculate the fare based on the 2 main factors.

1. Distance and time taken to reach the location.
2. Time to spend on cheers task
3. Pricing service will calculate the price for the Requestor and cheerleader
4. A detailed price breakdown will be provided and the task will be completed (including tax).
5. Price calculation to cheers task:
   1. Minimum distance and minimum task duration = minimum stand price
   2. More than minimum distance and/or more task duration = calculate the cost where price paid for 5-min increment charging $10 per 5 mins.

### Cheer Requestor and Cheer Leader Management Service

Assuming the information of cheer requestor and cheerleader is higher in volume to large users in the future, the profiles are maintained by two separate microservices.

* **Cheers Requestor Management Service:**
  + Signup and login
  + Storing user profile information (name, contact details, payment methods)
  + Trip history (for analysis)
* **Cheerleader Management Service:**
  + Registration and onboarding process
  + Number of cheerleaders available online to take new requests at a location.

By dividing as two services, allows us to have separate security risk levels if required, easily scalable(meet specific demand) and maintainable(upgrade functionalities) and follows SRP principle.

### Trip Management Service

The Trip Management Service in the SpeedyCheers app rider application plays a central role in coordinating the entire trip after accepting the request by cheerleader

1. Receives the request for the trip between cheerleaders and Requestor that includes pickup location.
2. Real time tracking based on trip id:
   1. Keep on tracking the progress of cheerleaders location and to calculate the time he spent in a location either on the road or in the pickup location to do cheers task.
   2. **Keep tracking the new ETA depending on cheer leader gps position and notify all.**
3. **Manages the state of trip request, state management of requestor and state management of cheerleader based on the trip.**
4. **Managed trip details are provided to billing service.**
5. **After cheers task is completed, send notification to do final price calculations to price service and send payment request notification to the Cheers requestor.**

**Overall Orchestrator:**

The Trip Management Service essentially acts as the conductor of the entire Speedy Cheers trip. It has to provide the real-time updates to the requestor and cheerleader.

The service is a write-state service, so this service should be always available, resilient and fault tolerant. The recovery time should be very fast. So the status of the trip must be maintained in database that is shard, backup and replicated. No data loss is allowed.

### Payment Service

The payment service will talk to third party payment processing system and make sure the payment is taken from the requestor and payment is done for the cheer leader in the planned time. The payment process can be asynchronous. It should be able to accept various payment methods.