Bike Rental System

Team 3
Kang-Hua Wu
Hongyuan Li

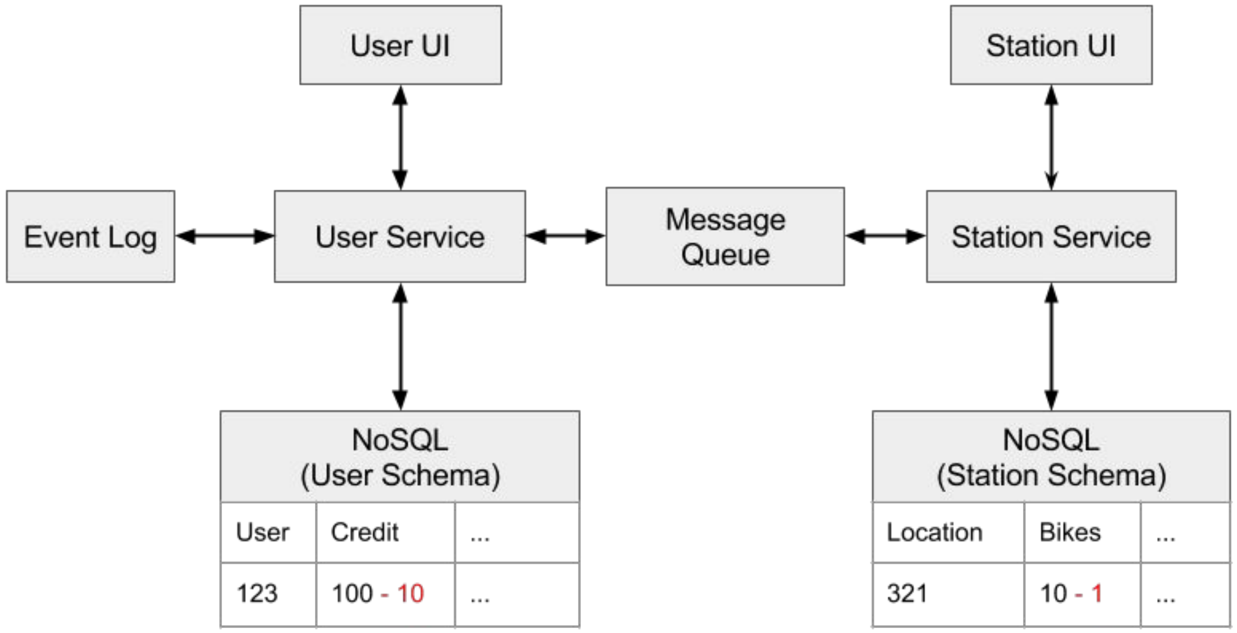
Design Goals

- Business
  - Users can book bikes online
  - Users can check out / in bikes at stations

- Non-blocking Transaction System
  - An scenario (All bikes checked out during, while transaction waiting for station's response over the network)

- Event Sourcing (ES) & Command Query Responsibility Segregation (CQRS)

High Level Architecture



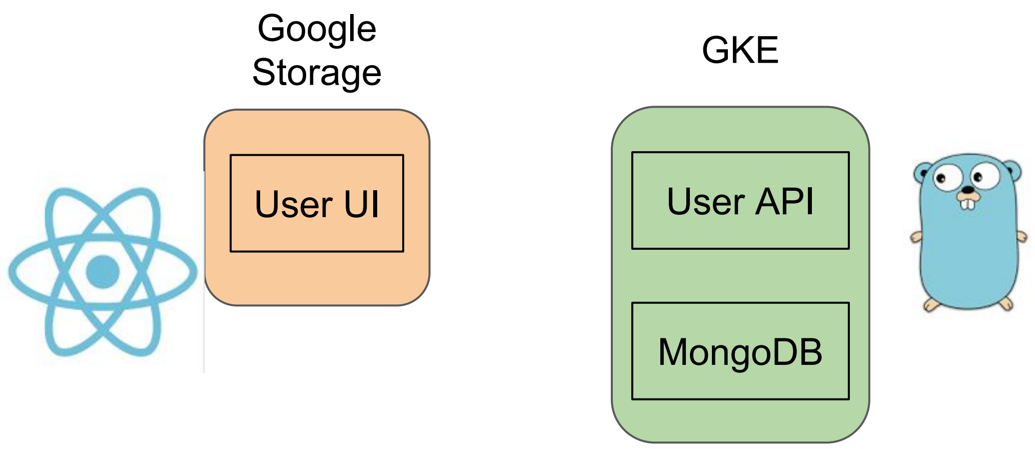
User	Credit	...
123	100 - 10	...

Location	Bikes	...
321	10 - 1	...

- User Request to book a bike at User UI
  - User Service Return OK if user account has enough balance
  - User Service sends out Reservation Message to MQ
- User Check in bike at Station UI
  - User Station sends out txn completion message to MQ

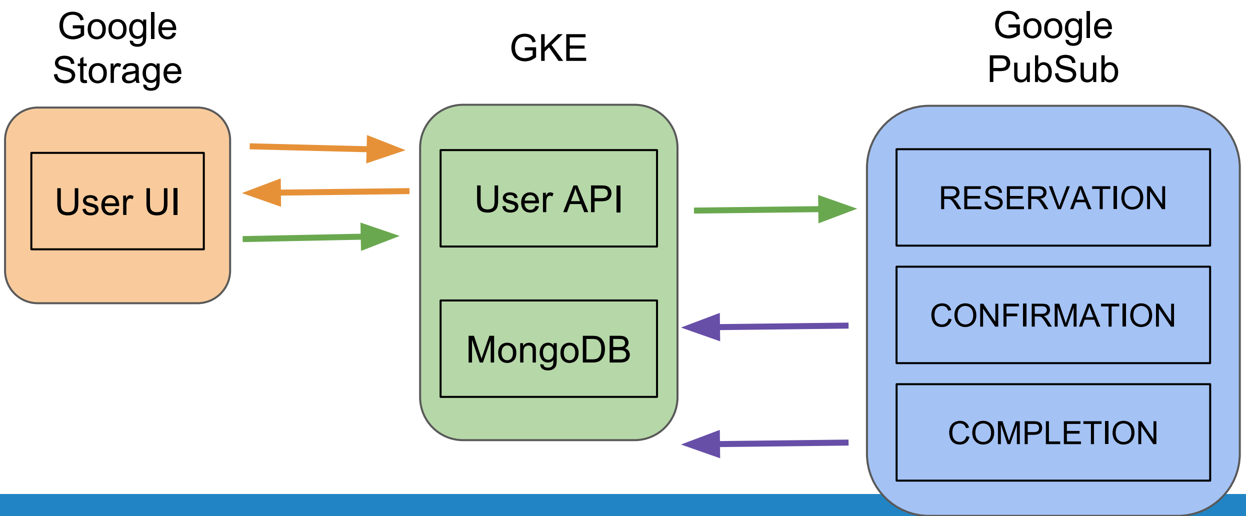
User Service

- Components:
  - User UI - Google Storage
    - React, Redux
  - User Backend - GKE
    - 2 containers
    - Restful service (stateless) - Golang
    - Data Persistence (stateful) - MongoDB



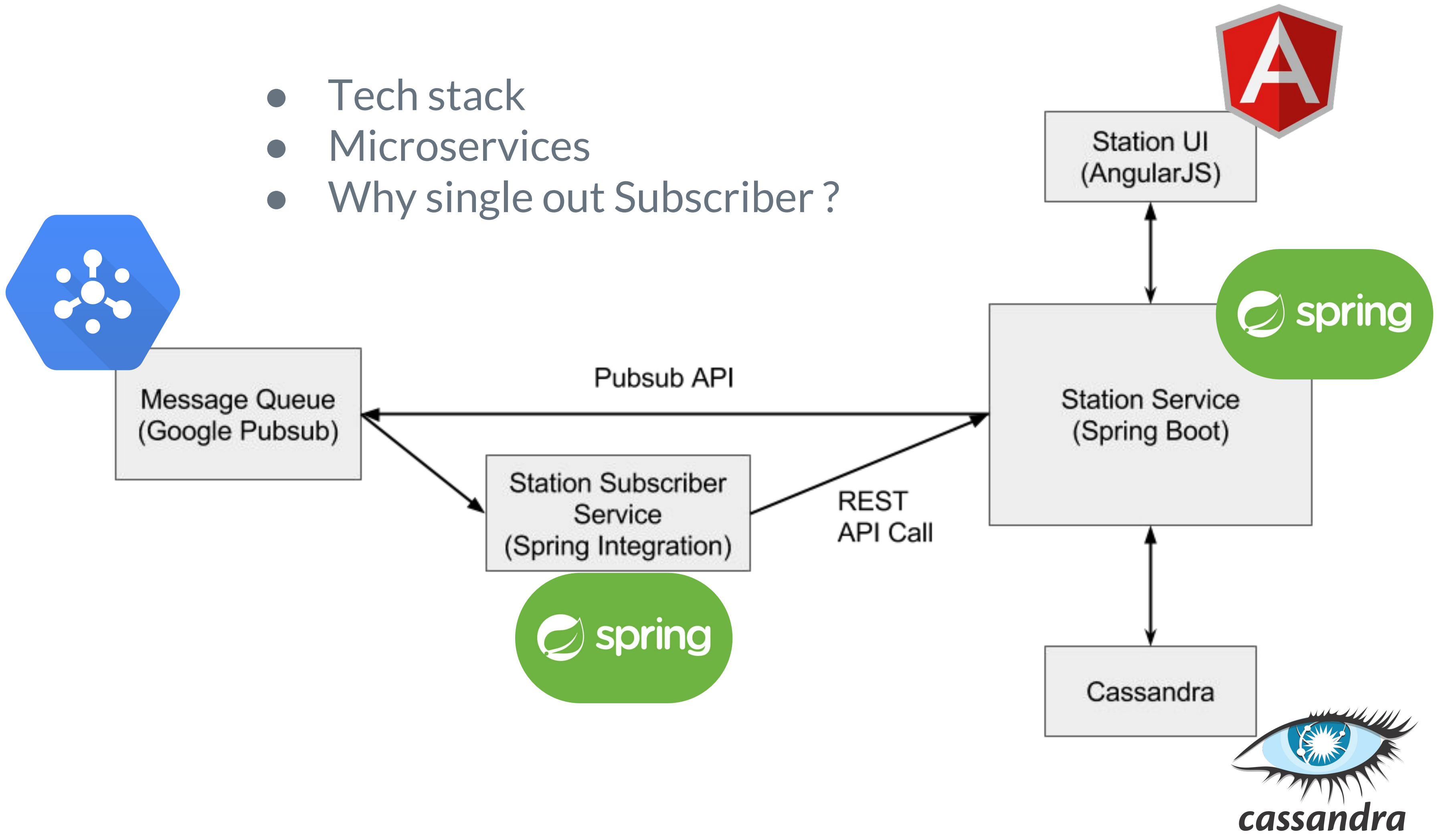
Data flow

- Pubsub
  - Act as intermediate between 2 services (User/Station)
- Functionalities
  - Retrieve User Info and orders
  - Create order
  - Update order status

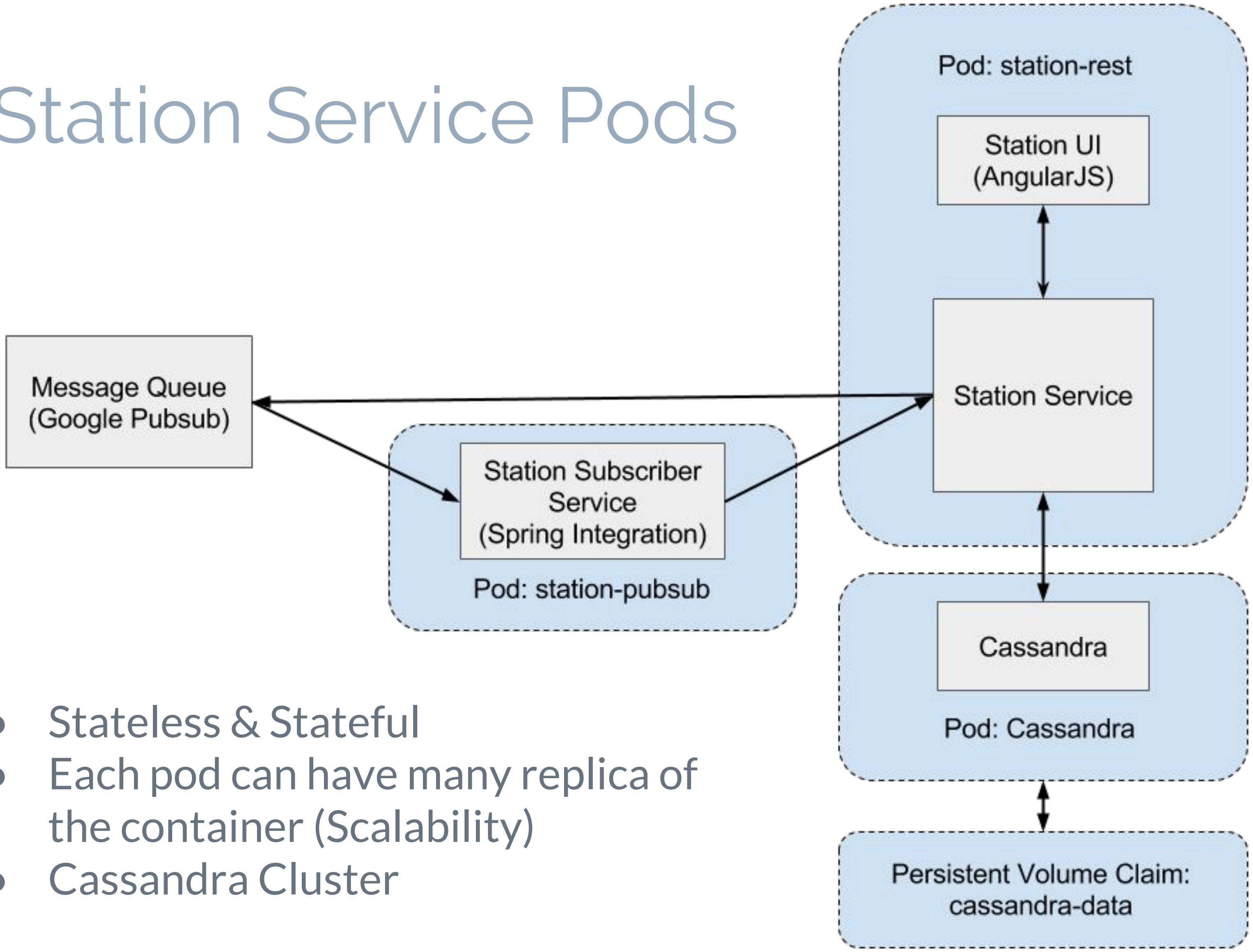


Station Service Architecture

- Tech stack
- Microservices
- Why single out Subscriber ?



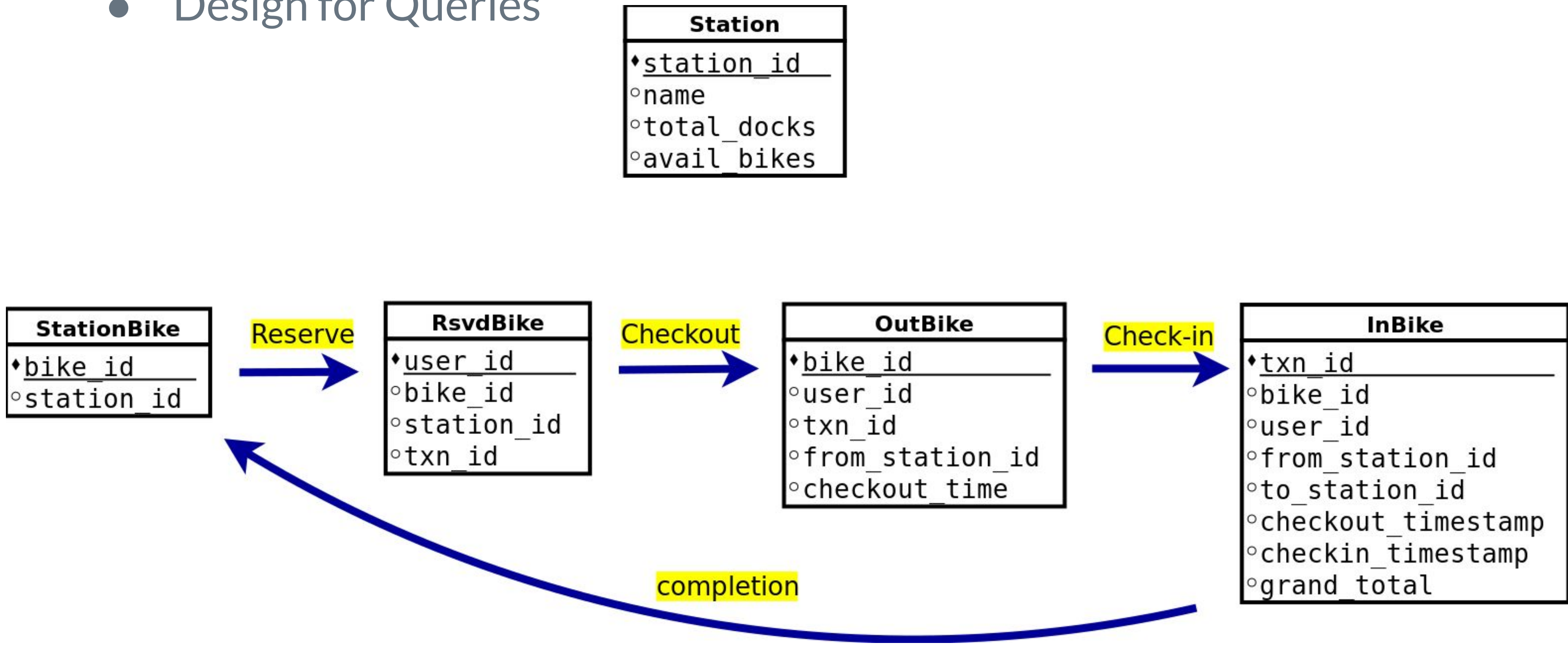
Station Service Pods


Pod: station-rest
Station UI (AngularJS)
Station Service
Message Queue (Google Pubsub)
Station Subscriber Service (Spring Integration)
Pod: station-pubsub
Cassandra
Pod: Cassandra
Persistent Volume Claim: cassandra-data

- Stateless & Stateful
- Each pod can have many replica of the container (Scalability)
- Cassandra Cluster

Station Service Data Model

- Duplicated data is OK
- Select with non-key filter is inefficient
- Join is Hard
- Design for Queries



Demo