Rent-A-Car System

Architecture Specification Document

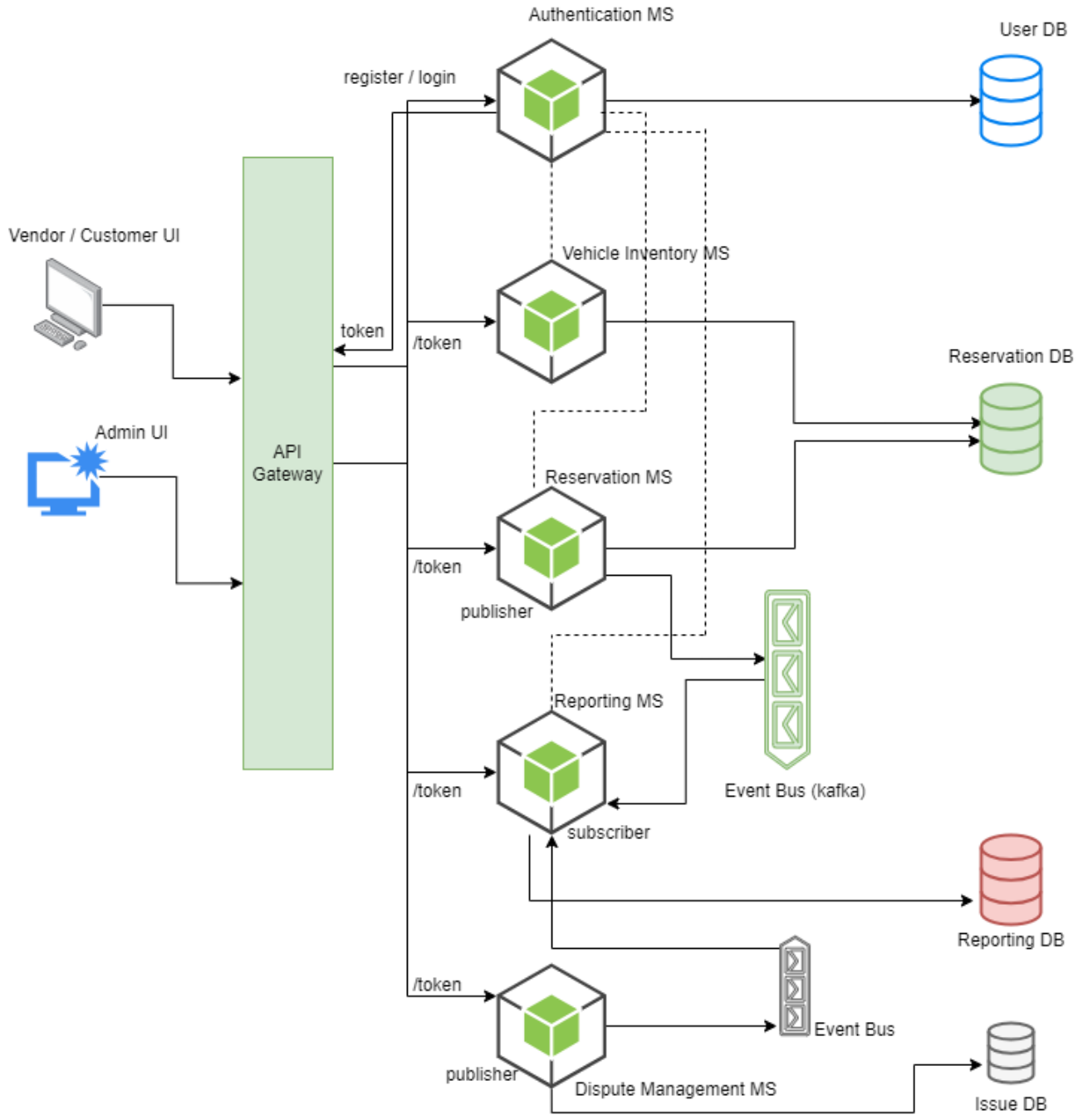
High Level Design

Team members

Tanvir Zobair Mahboob (615163)
Md Siam Biswas (615195)
Md Atikur Rahman (615231)
Md Shahab Uddin (615180)

Architecture Specification

High Level Design



Physical Tier

- The system is designed in microservices architecture. In the physical tier,
    - there will be client (customer/vendor and administrators with different roles)
    - Microservices will be part of the Application Tier
    - Several databases dedicated to different microservices will be part of Data Tier.

Logical Tier

- Client (users) will be connected to API gateway through HTTPS, which is an abstraction between user and microservices. Based on routing path, API gateway will redirect requests to respective microservices via REST and respond with JSON data.
- Initially the registration of users/vendors will be requested from UI. All the user management, registration and login and user validation will be handled by Authentication Microservice. Upon valid user registration and successful login, each user will be provided with a valid JWT token from Authentication MS.
- All other subsequent requests for other activities or actions will be passed with the token to the API gateway. API gateway will not route any request if there is no token.
- All other microservices will receive token and validate before sending response.
- List of vendor vehicles, their rates, status will be managed by Vehicle Inventory microservice which has a database to store persistent inventory data (Reservation DB).
- Reservation, billing, payment services are handled by Reservation microservice which will also be connected to Reservation DB.
- Issue, risk and compliance management will be handled by Dispute management microservice that has another dedicated database connected with it.
- Reporting for Business Intelligence will be handled by separate microservice – Reporting MS that has a dedicated denormalized database behind.
- Communication between reservation microservice and reporting microservice will be event-driven. There will be a message broker (kafka) in between where reservation MS will be publisher and reporting MS will be subscriber. Any reservation/billing/cancellation event will be published to event bus that will trigger the reporting MS to log that into Reporting DB.
- The same architecture will be used between Dispute Management MS and Reporting MS.

Technology:

- Web UX – Angular, Bootstrap
- Microservices – Spring Boot
- Message Queue – Kafka
- Database – MySql/MS Sql/Postgres OR mongo DB/Dynamo DB
- Container (for cloud native microservices) – Docker
- Deployment – on premises (kubernetes) or AWS (EKS) – *to be decided later*