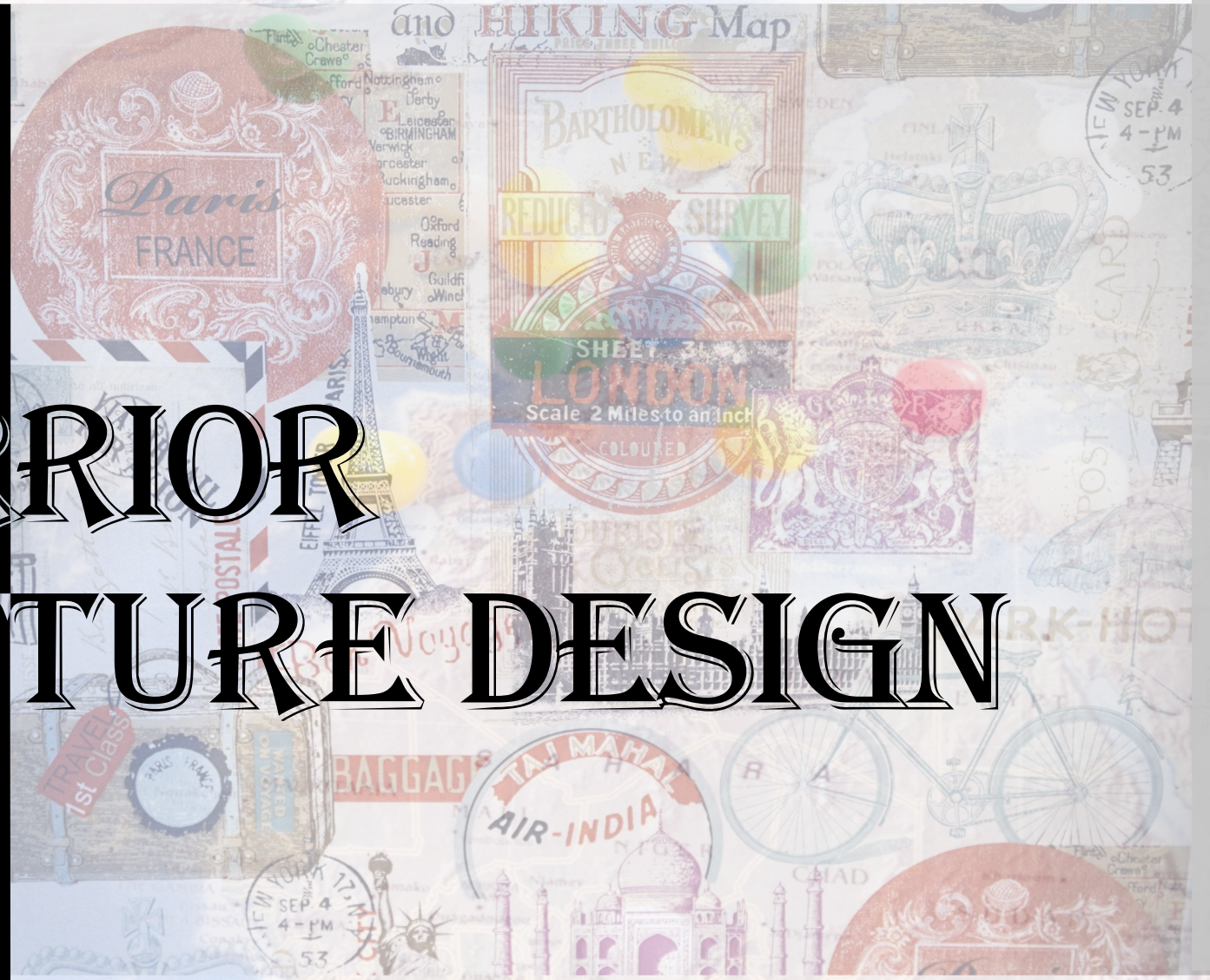
Next Gen Online Trip Management system

ROAD WARRIOR ARCHITECTURE DESIGN



PRESENTED BY
TEAM GENSIGHTS

Road Warrior

Overview

PRODUCT

ARCHITECTURAL CHARACTERISTICS

COMPONENT AND CONNECTION – WORKFLOW & ACTOR/ACTION ANALYSIS

COMPONENT AND CONNECTION OVERALL VIEW

ARCHITECTURAL DECISIONS

Road Warrior

Essential Functional Features

Booking & Reservation

- ✓ Smart Search
- ✓ Global Distribution System (GDS) connectivity like SABRE, APOLLO
- ✓ Integration with third-party Agencies (Hotel, Airlines, Car)
- ✓ E-mail Polling
- ✓ Payment system integration
- ✓ Booking confirmations
- ✓ Booking Management (Add, Update, Cancel or Delete)

Trip & Package

- ✓ Trip templates
- ✓ Trip publishing tools
- ✓ Insights and reporting
- ✓ Multi-language Support
- ✓ Geo location & Maps integration

CRM System

- ✓ Distribution of leads among agents
- ✓ Trip-specific customer categories
- ✓ Configurable dashboards
- ✓ Support for Different channels (booking websites, Helpline, etc.)

Financials

- ✓ Inventory management and tracking
- ✓ Connection to banks
- ✓ Invoicing
- ✓ Reconciliations
- ✓ Supplier data handling

Reporting

- ✓ Inventory management and tracking
- ✓ *end-of-year summary reports*
- ✓ Connection to banks
- ✓ Invoicing
- ✓ Reconciliations
- ✓ Supplier data handling

Road Warrior

Non-Functional Requirements

Availability & Security

- ✓ User Management
- ✓ Social Media Integration
- ✓ Data encryption
- ✓ International availability
- ✓ Push Notifications

Users

- ✓ 2M user/week
- ✓ 15M users total

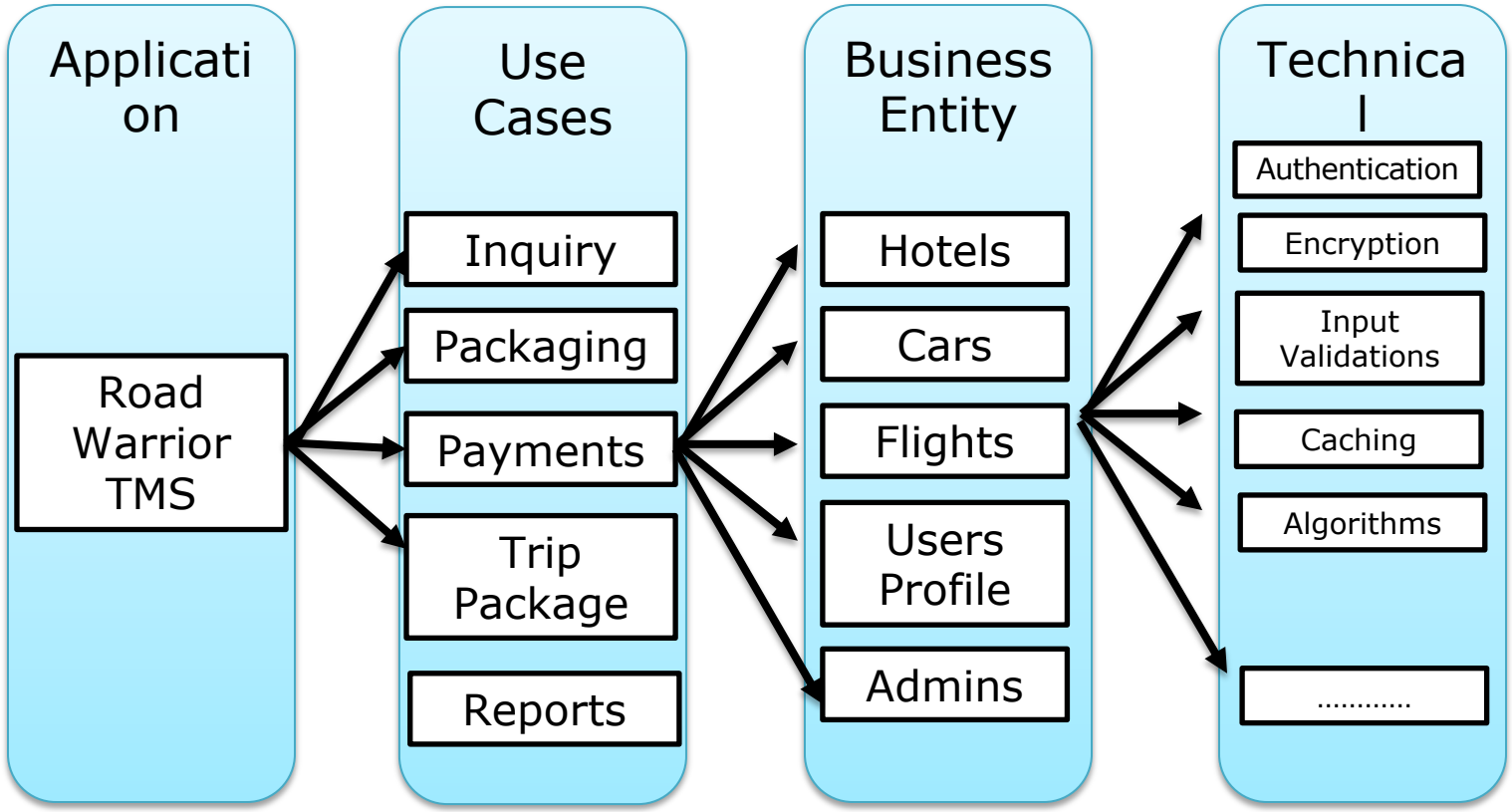
SLAs

- ✓ Max 5 Mins/month downtime
- ✓ *Updates must be presented in the app within 5 minutes*
- ✓ Web Response time – 800 ms
- ✓ Mobile Response time - 1.4 Sec
- ✓ Near real time flight tracking

High-level Solution

System's capabilities and choices

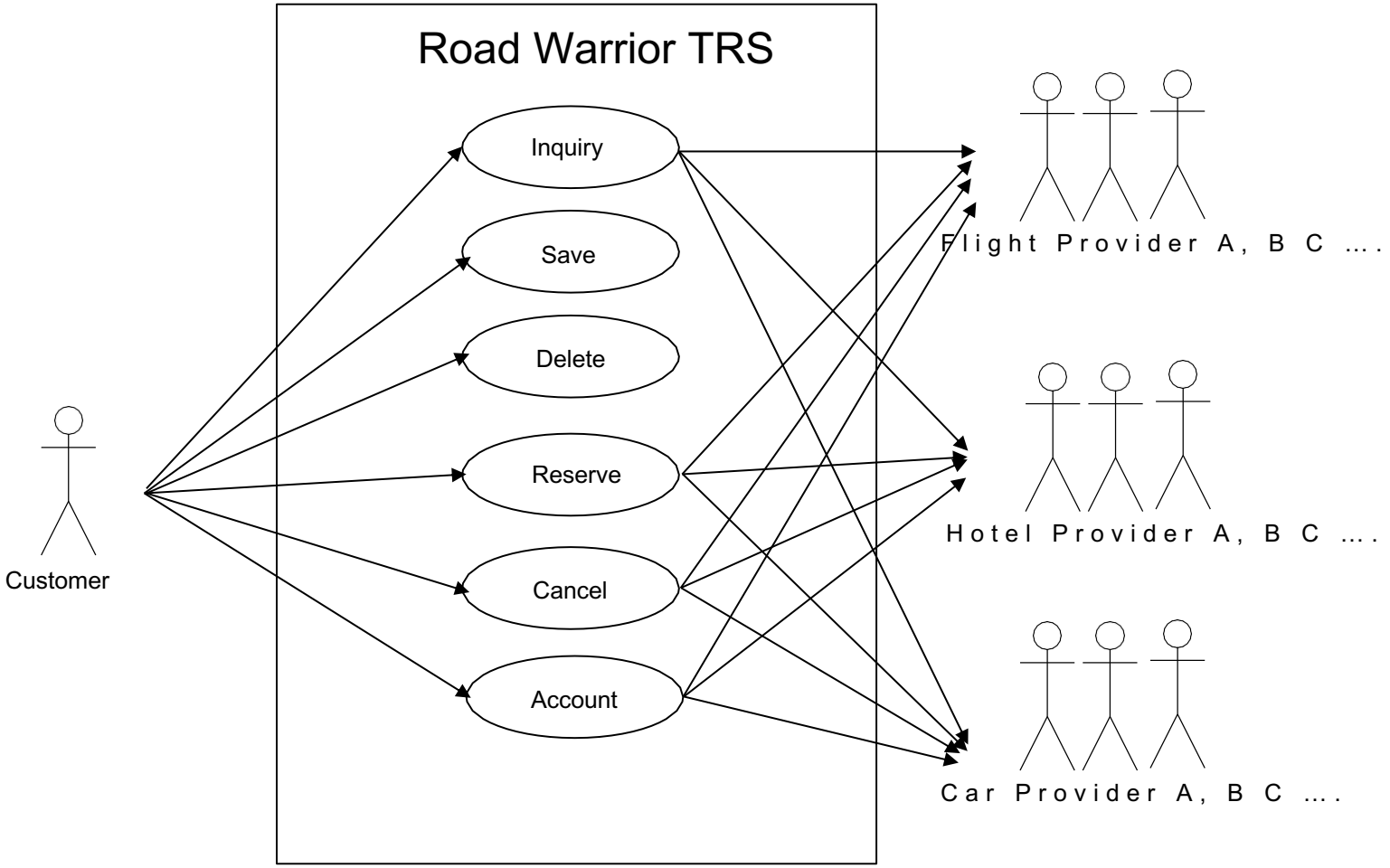
We are developing a Web/Mobile Portal to make reservation online and get help / recommendations



B.P.	Goal
Flights	❖ Ticket Quote Response Time. ❖ Airlines Agent Reliability ❖ Flight Ticket Quote Quality
Hotels	❖ Hotel Quote Response Time ❖ Hotel Agent Reliability ❖ Hotel Reservation Quote Quality
Cars	❖ Car Quote Response Time ❖ Car Agent Reliability ❖ Car Reservation Quote Quality

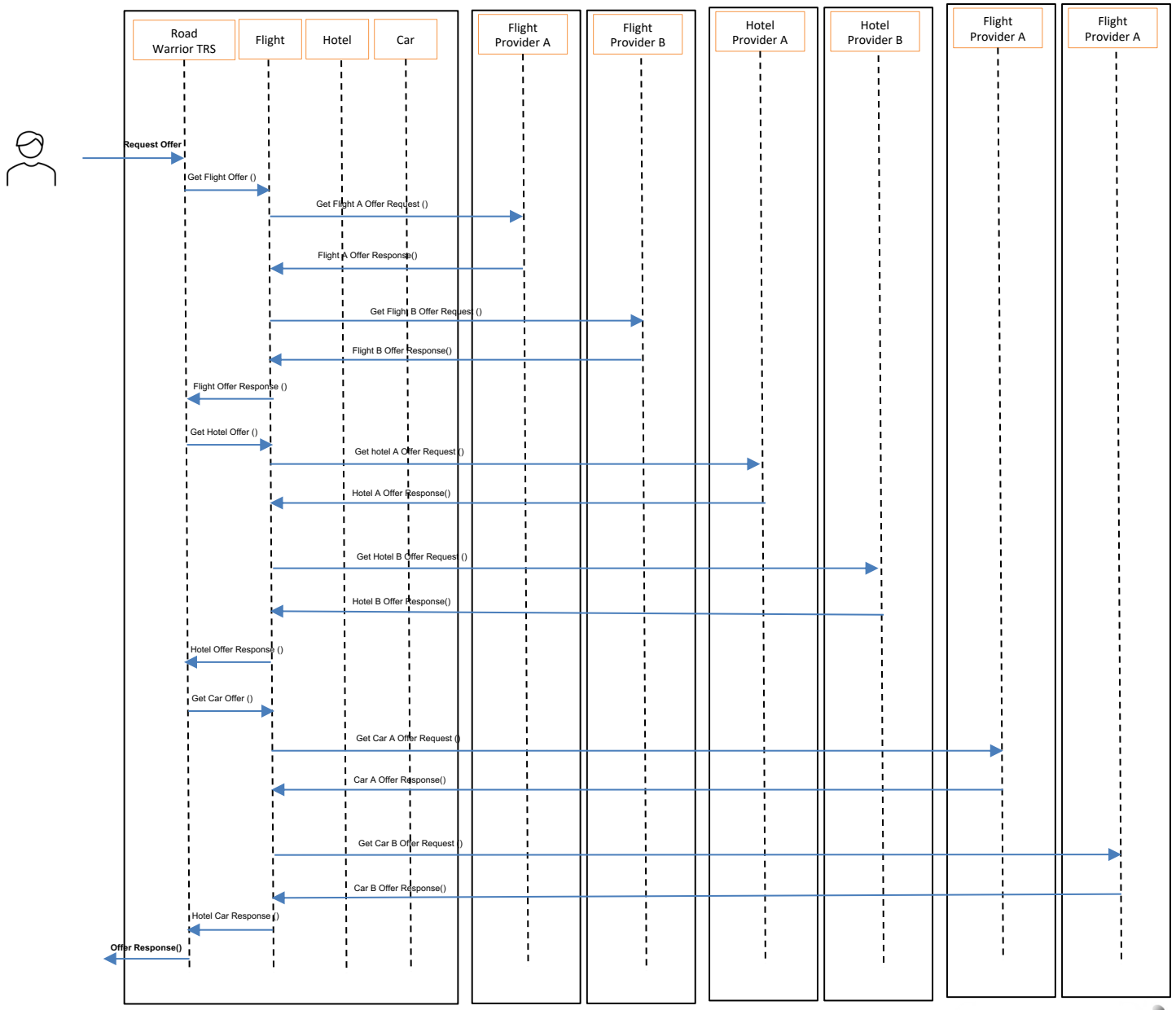
Road Warrior

Use Case Diagram Level 1

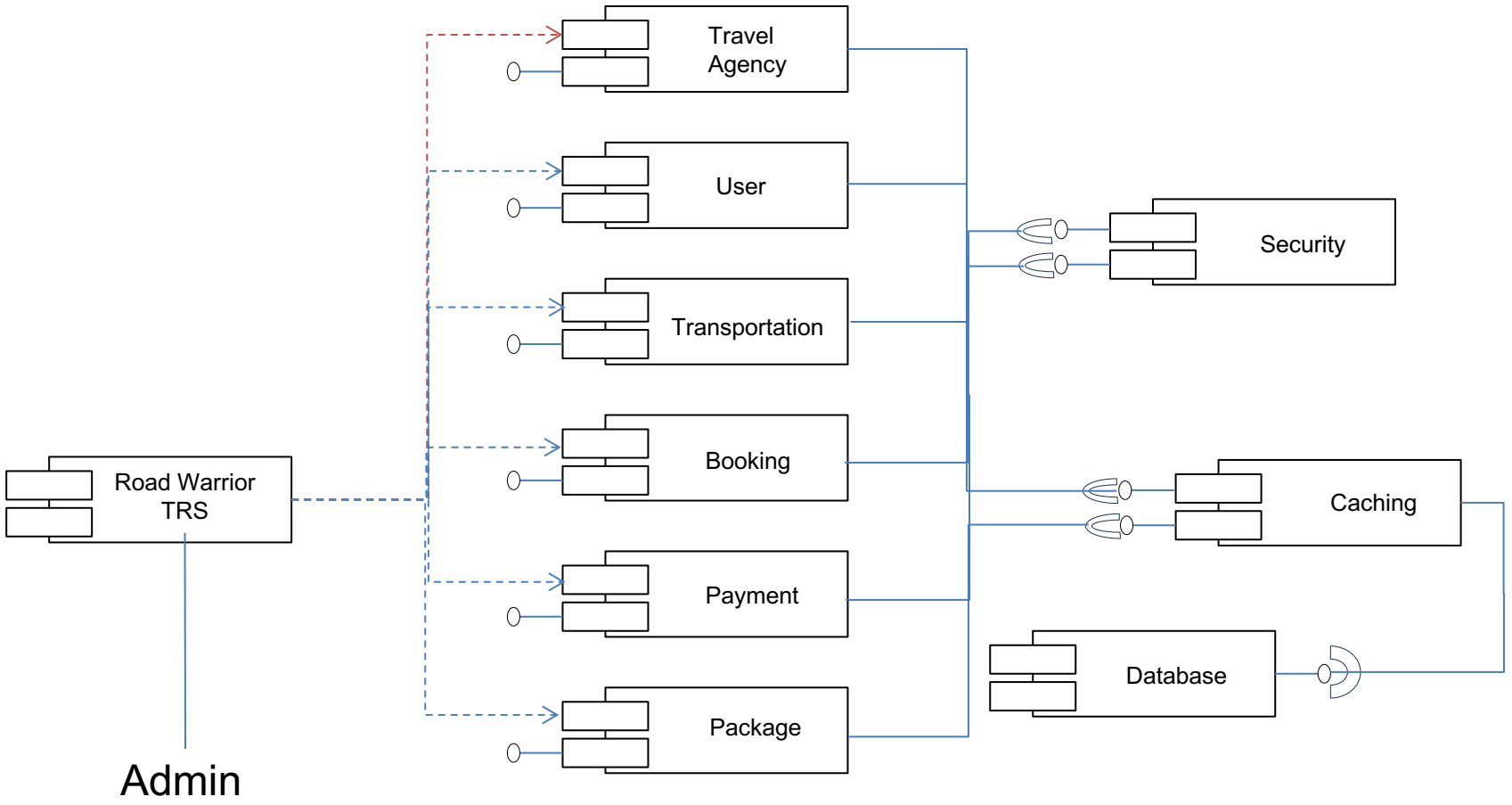


Sequence Diagram

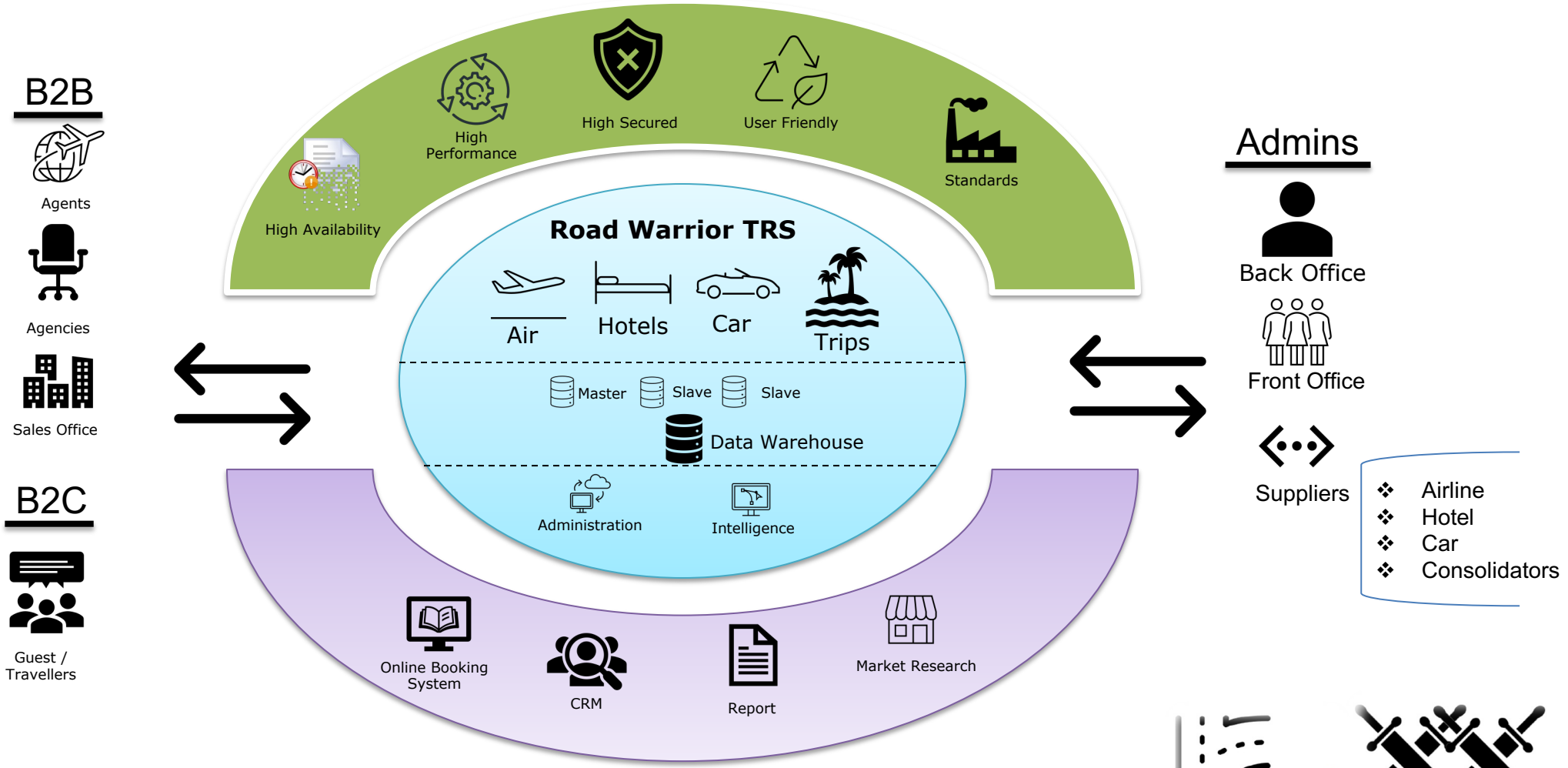
– Get Trip Offer



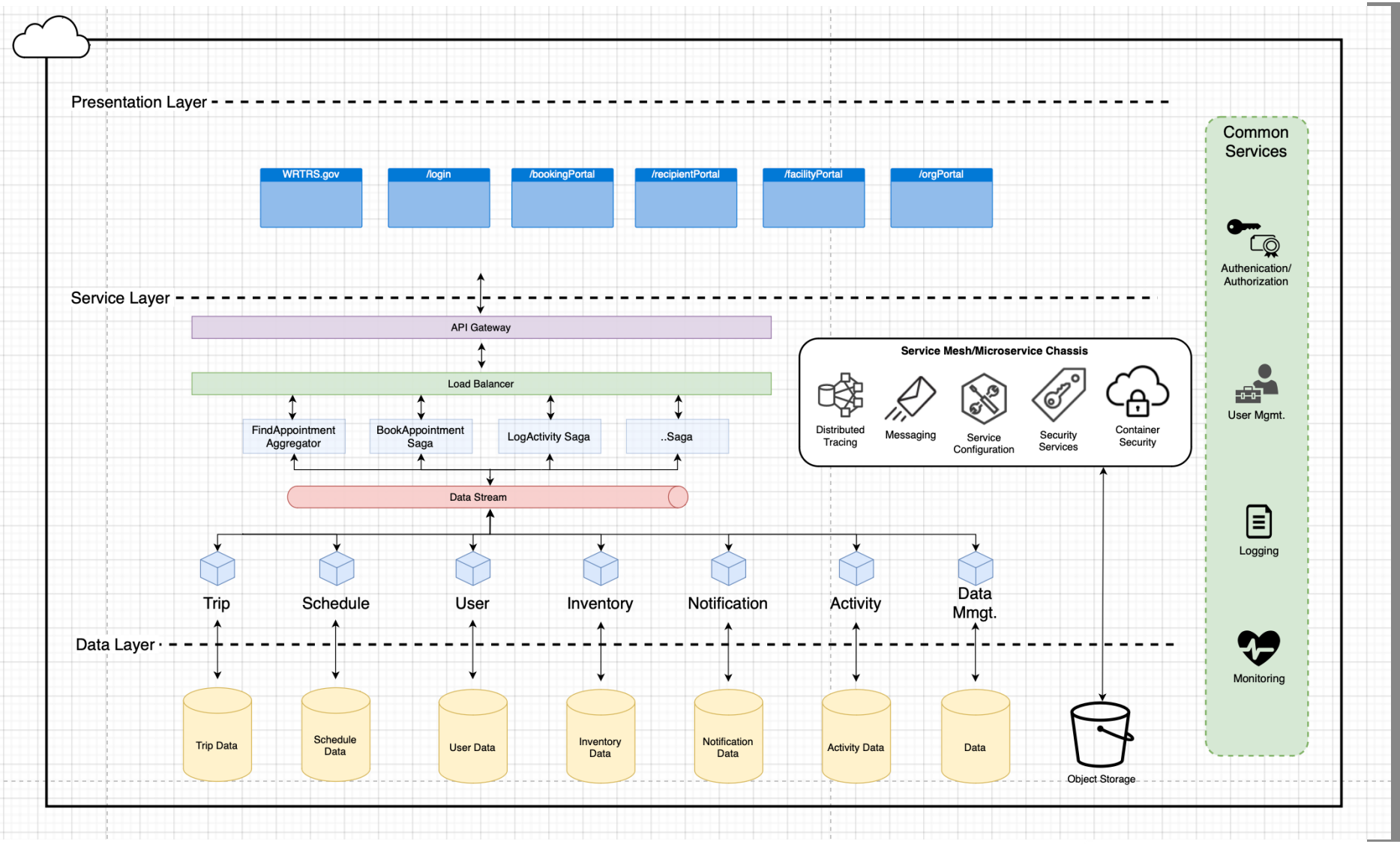
Component Diagram



Context Diagram

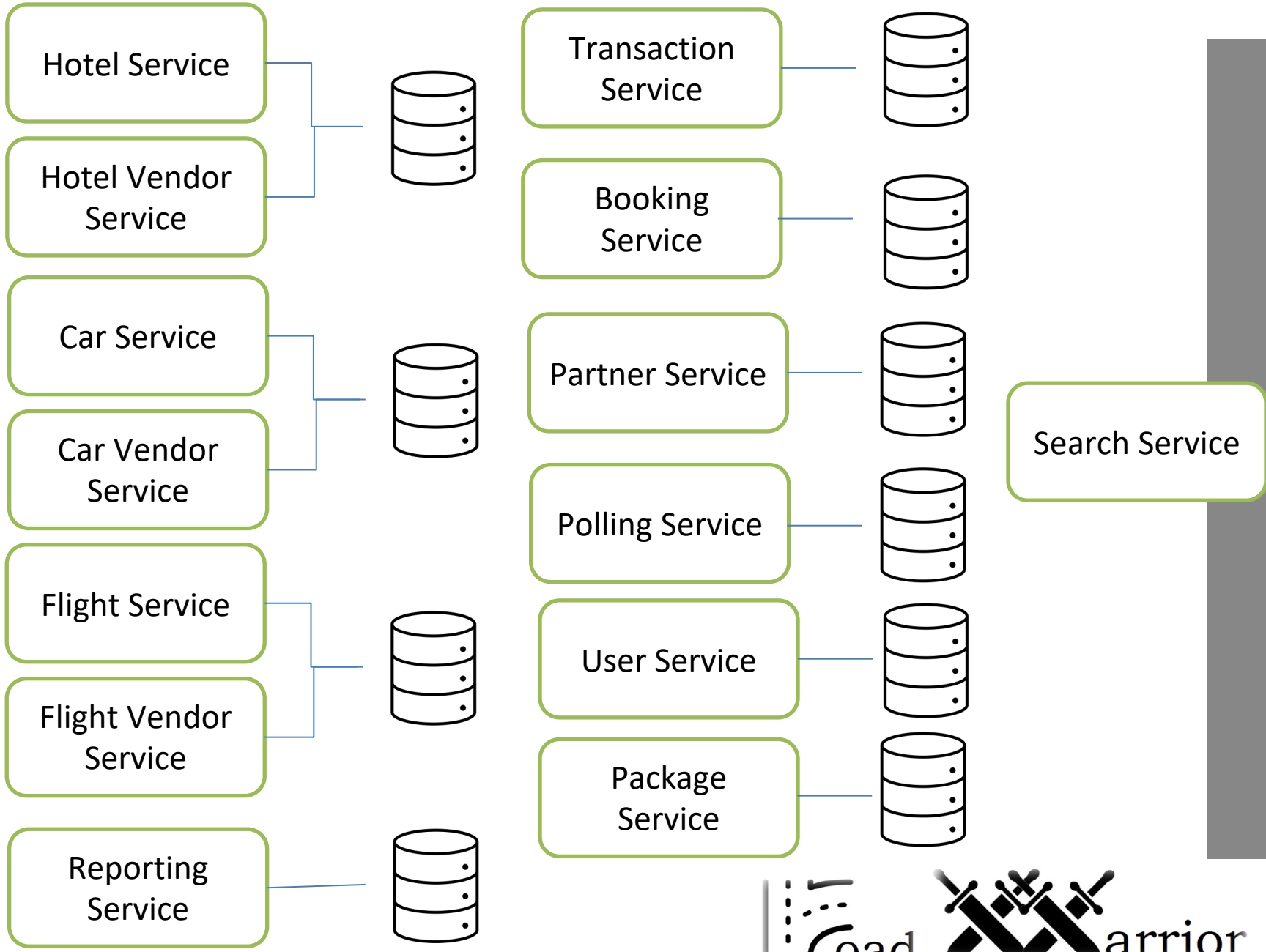


Logical Architecture diagram



Microservice Design

Microservice Candidates



Microservice Design

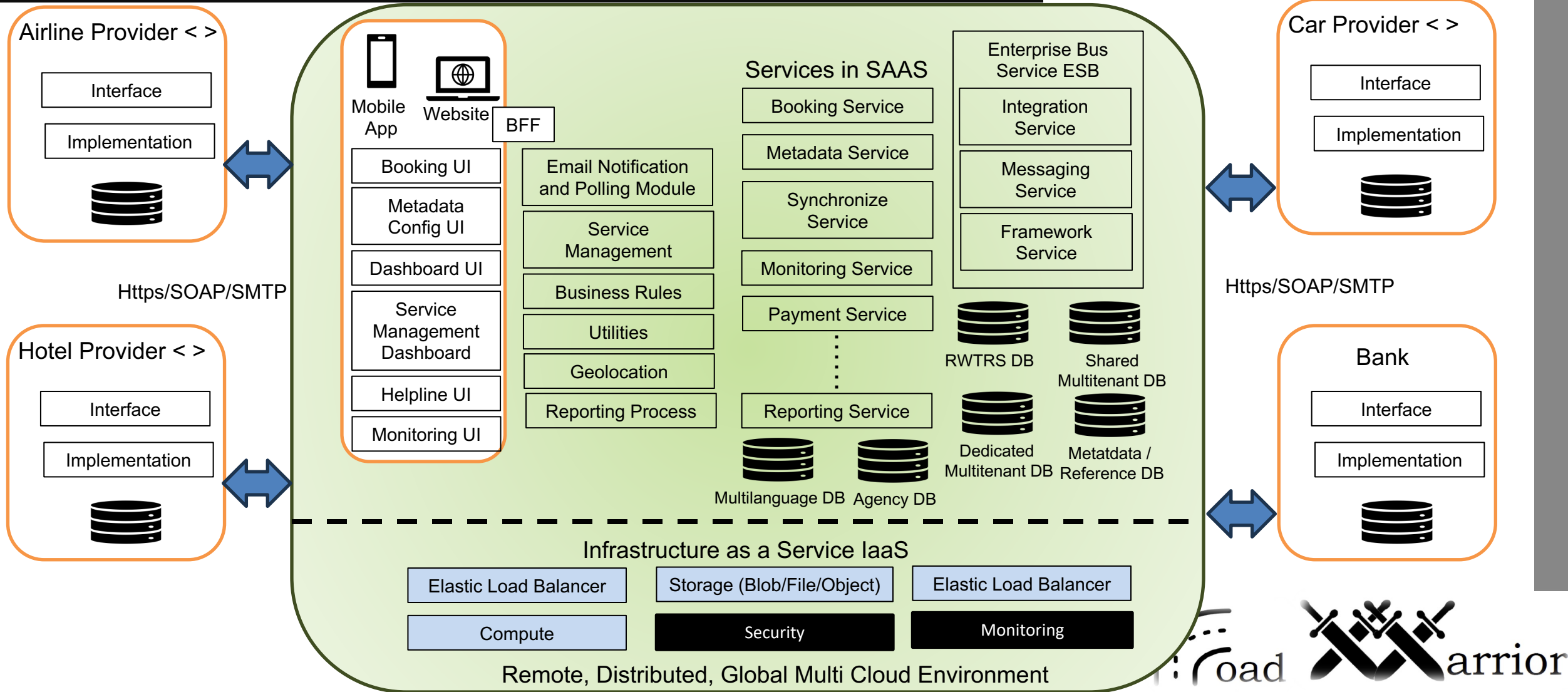
Service Communication Pattern
With Event driven Architecture

Service	Communication Pattern
Car Service	Synchronous (HTTPS, REST API)
Cars Vendor Service	Asynchronous (Message Bus, AMPQ)
Booking Service	Synchronous (HTTPS, REST API)
Reporting Service	Asynchronous (Message Bus, AMPQ)
Partner Service	Synchronous (HTTPS, REST API)
Flight Service	Synchronous (HTTPS, REST API)
Flight Vendor Service	Asynchronous (Message Bus, AMPQ)
Customer Service	Synchronous (HTTPS, REST API)
Transaction Service	Asynchronous (Message Bus, AMPQ)
Packages Service	Synchronous (HTTPS, REST API)
Payment Service	Asynchronous (Message Bus, AMPQ)
Hotel Service	Synchronous (HTTPS, REST API)
Hotel Vendor Service	Asynchronous (Message Bus, AMPQ)

AMPQ – Advance Messaging Queuing Protocol

Deployment Model

Provide Road Warrior-TRS As Service (RWAAS)



Architectural Characteristics

RELIABILITY

AVAILABILITY

COMPLIANCE

DATA SECURITY

SCALABILITY

ELASTICITY

PERFORMANCE

CUSTOMIZABILITY

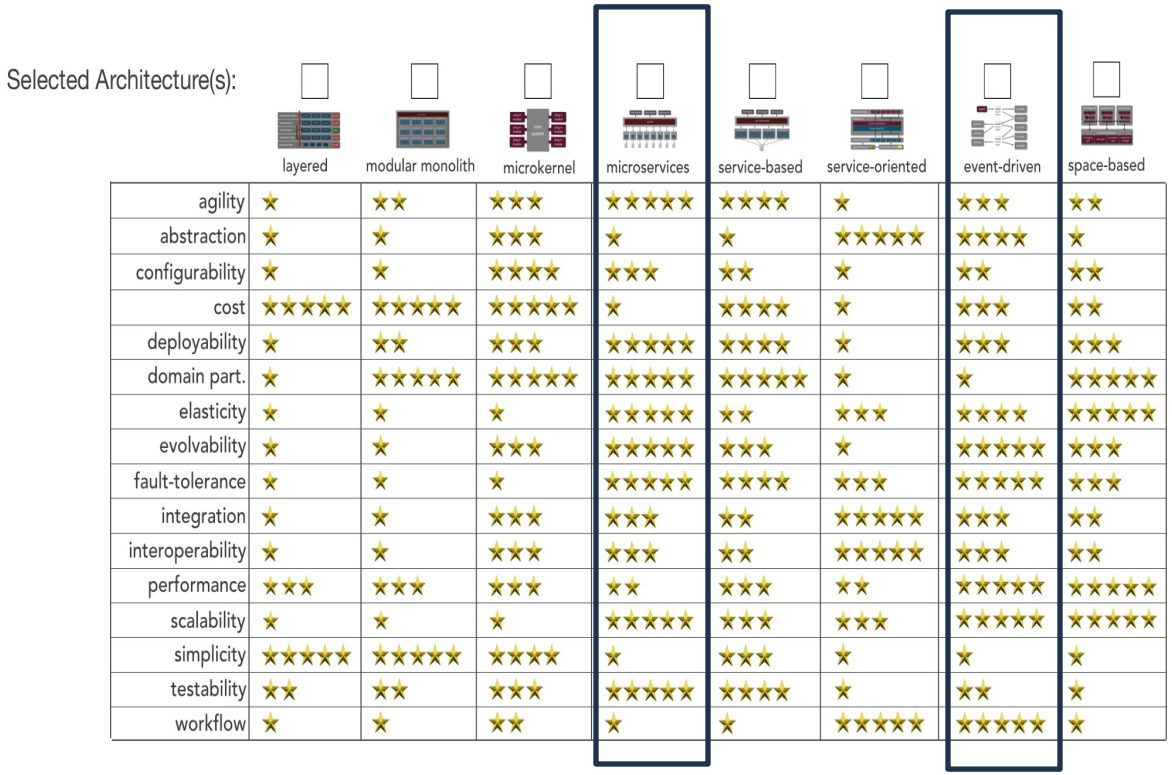
TRANSPARENCY

Road Warrior

Architectural Decision Record – ADR 001

Selected Architecture

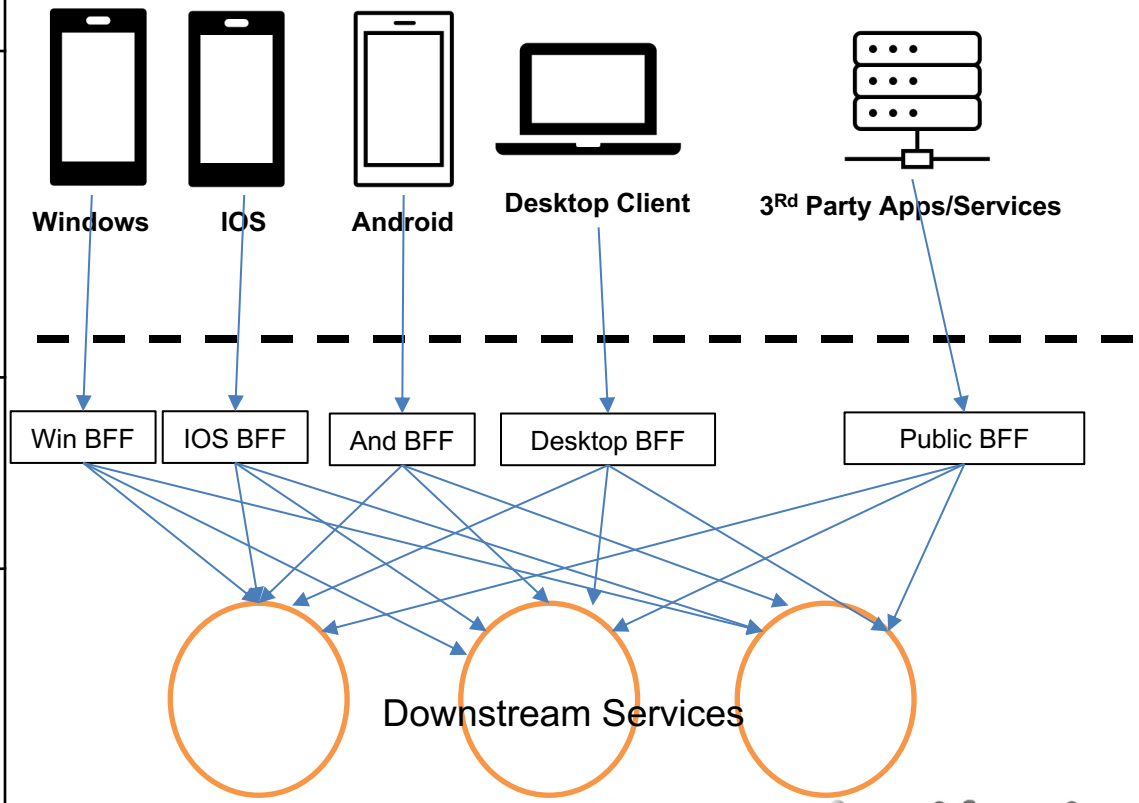
ADR001: Use of Microservice event driven style with Containers	
Status	Accepted
Context	The overall architectural should be simple, maintainable and easy to use. Team adopting a well defined established architecture that supports evolution.
Decision	We will use the event driven microservice style framework for our backend services. These service will be deployed using containers (K8)
Consequences	- Configurability - Maintainability - Agility - Scalability - Fault Tolerance - Elasticity



Architectural Decision Record – ADR 002

BFF Pattern

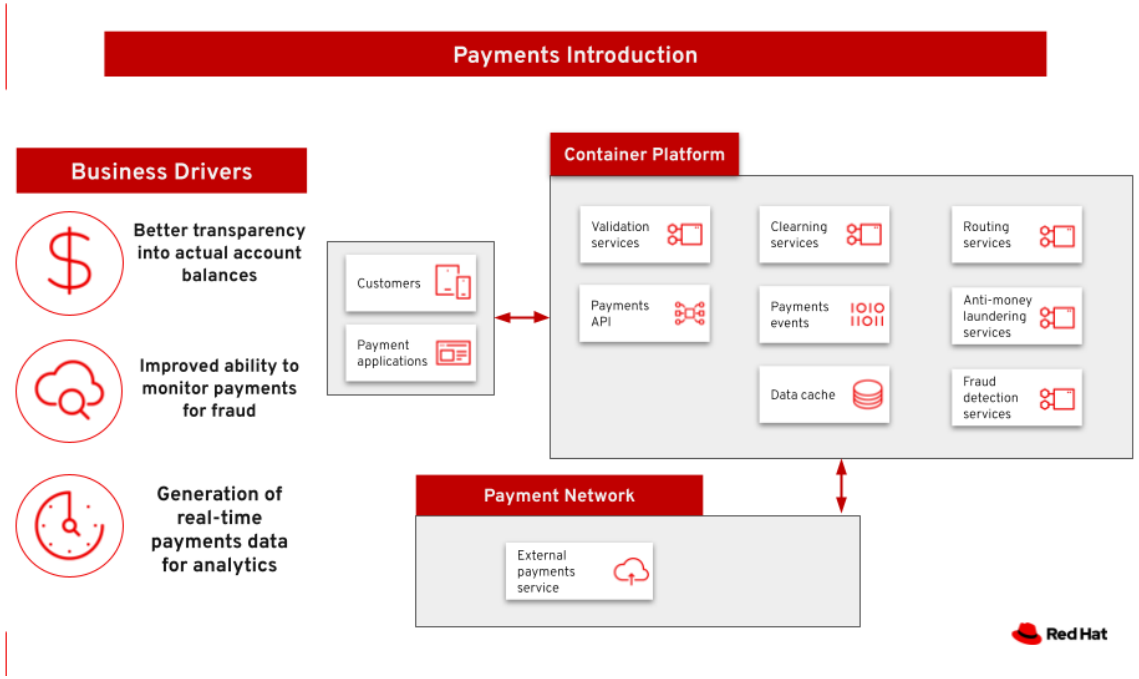
ADR002: Use of BFF Pattern	
Status	Accepted
Context	We have a microservice architecture with several REST services and different types of frontends: Web application, iOS application, Android application, public API clients. Different frontends may require slightly different message formats, message structures, headers, etc.
Decision	We will use Backend For Frontend (BFF) pattern
Consequences	- Interoperability - Adaptability - Abstraction



Architectural Decision Record – ADR 003

Payment Gateway

ADR003: Use Payment Gateway in self-hosted mode	
Status	Proposed
Context	The Road Warriors TRS app will be available in different platform and variety of devices. The app can take bookings and give customers flexible yet secure payment options. The payment solution should be simple and easy to use, provide scalability, and work on native mobile apps.
Decision	We will use a commercial payment gateway in integrated self-hosted mode with Modern Payments Architecture
Consequences	- Security - Integration - Flexibility - Acknowledgement in the background



Reference: https://www.redhat.com/architect/portfolio/detail/12-integrating-a-modern-payments-architecture

Road Warrior

Architectural Decision Record – ADR 004

Zero Trust Architecture

ADR004: Use Payment Gateway in self-hosted mode	
Status	Approved
Context	Our system will communicate with upstream and downstream systems in cloud environment. We can not rely on the assumption the requests are safe from the network.
Decision	Internal calls from modules should contain security info with auth and claims info from the very beginning.
Consequences	While presenting vast opportunity, the Adoption of Zero Trust is a journey that comes with potential hurdles along the way. A change in mindset is fundamental to success but must be followed with specific actions that put principles into practice.

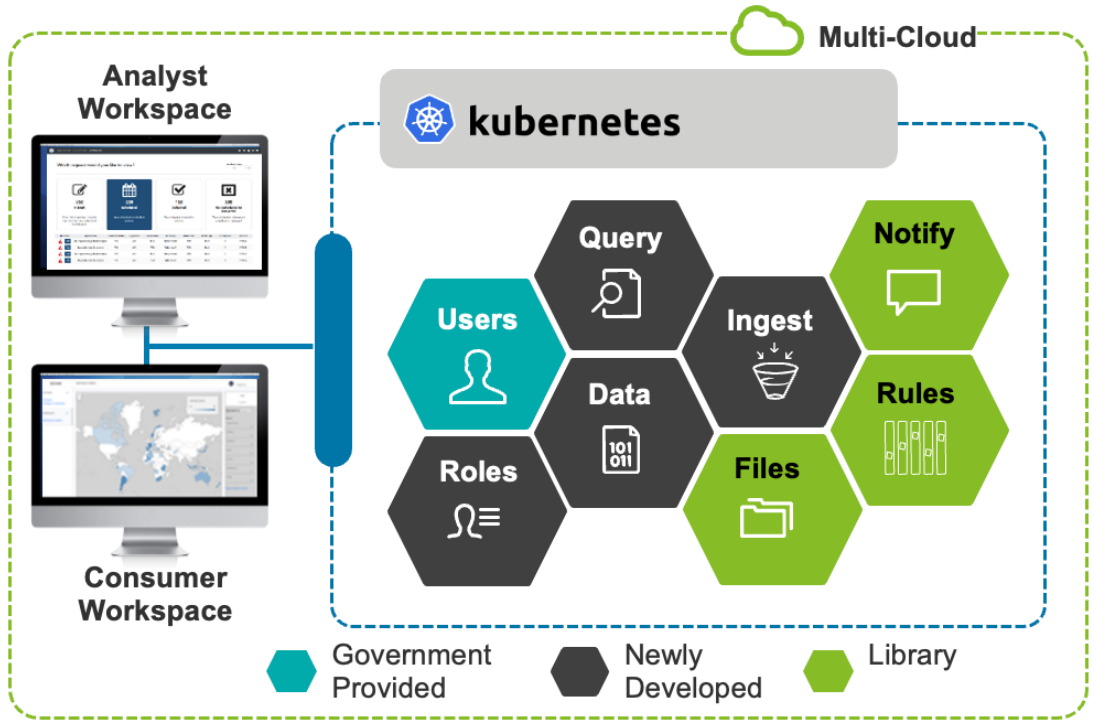
Principle	Actions
Focus on the fundamentals	• Get buy-in at the leadership level and head off any detraction • Conduct a current state assessment of foundational cyber hygiene • Establish a repeatable process to track and mature system, application, identity, and data inventories
Prioritize mission needs over technology	• Conduct a mission threat assessment • Set priorities for current operations as well as future mission needs • Determine the technology roadmap only after threat analysis is complete for your specific mission(s)
Centralize and federate identity services	• Define policies and standards • Centralize identity governance, authentication, and authorization services • Provide integration and application programming interface (API support across technology ecosystems
Minimize disruption by starting with low-risk use cases	• Find and execute “quick wins” • Map out an incremental implementation plan that sets near-term goals • Conduct pilots in low-risk environments and scale where success is found

Road Warrior

Architectural Decision Record – ADR 005

Use multi-Cloud hosting environment

ADR005: Use multi-Cloud hosting environment	
Status	Approved
Context	Our architecture needs to be hosted in multi-cloud environments. We need to use different services from different cloud providers based on cost, availability and ease of use.
Decision	We will be using Azure, AWS and GCP with different regions to have availability and performance worldwide.
Consequences	The implementation is expected to scale elastically on demand. We can run experiments and develop staging and test environments easily.



Architectural Decision Record – ADR 006

Use Infrastructure as a Code

ADR005: Use infrastructure as a Code	
Status	Approved
Context	Our architecture need to use infrastructure as a code pattern. We will automate the environment creation for faster and reliable deployment.
Decision	We will use Terraform, Ansible to use IaaC
Consequences	- Extensibility - Scalable – Reliability

Road Warrior

Architectural Decision Record – ADR 007

Use Redis for persistence caching and pub/sub

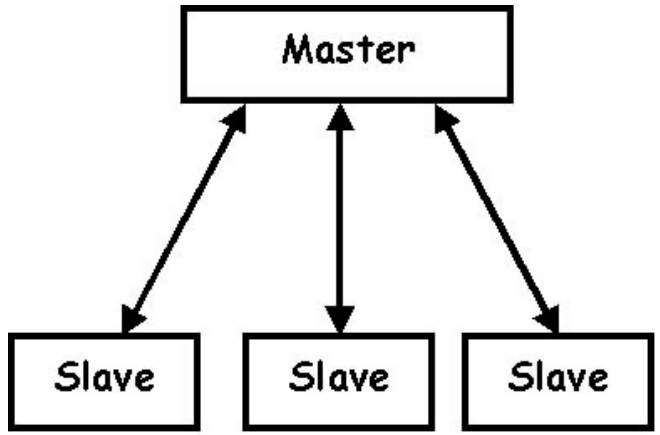
ADR005: use of Redis for queue and caching	
Status	Approved
Context	Our system require to manage booking, payments and event driven activities. Which require persistence caching and better message/event handling.
Decision	We will use Redis for handing the Caching and queue handling
Consequences	- Secure - Scalable – Reliability - Data integrity

Road Warrior

Architectural Decision Record – ADR 008

Use Master Slave Architecture Pattern

ADR005: Use of Master Slave architecture	
Status	Proposed
Context	Our system require to have least amount of delay time (Max 5 mins). We need to implement a solution which is with high performance
Decision	We are proposing to use master slave architecture for mostly search operations.
Consequences	- Fault tolerance - Parallel computing - Computational accuracy



TEAM GENSIGHTS

Alok Gupta

https://www.linkedin.com/in/enggalok/

Divya Gangula

https://www.linkedin.com/in/divya-gangula/

Kiran Vangala

https://www.linkedin.com/in/kiran-vangala-9155ab10/

Road Warrior


THANK YOU
Road Warrior