Software Engineering 2 Project

Power Enjoy

Niccolo' Raspa
Matteo Marinelli

What is Power Enjoy?

- Digital management system for car-sharing service
- Eco friendly, exclusively employs electric car
- Based in Milan

Text Assumption

- **Park in an Unsafe Area**
  - Restrictive to Prevent this situation from happening
  - One Hour Clock
- **Payments**
  - External Service that takes care payment process
- **Multiple Discounts**
  - Discount only applied if car is a safe area
  - Only Shared Ride discount is cumulative
  - Fines over Discount

Domain Assumption

CORRECTNESS AND AVAILABILITY
OF INFORMATIONS

CAR FUNCTIONALITIES

EXTERNAL SERVICES

Domain Assumption - Car

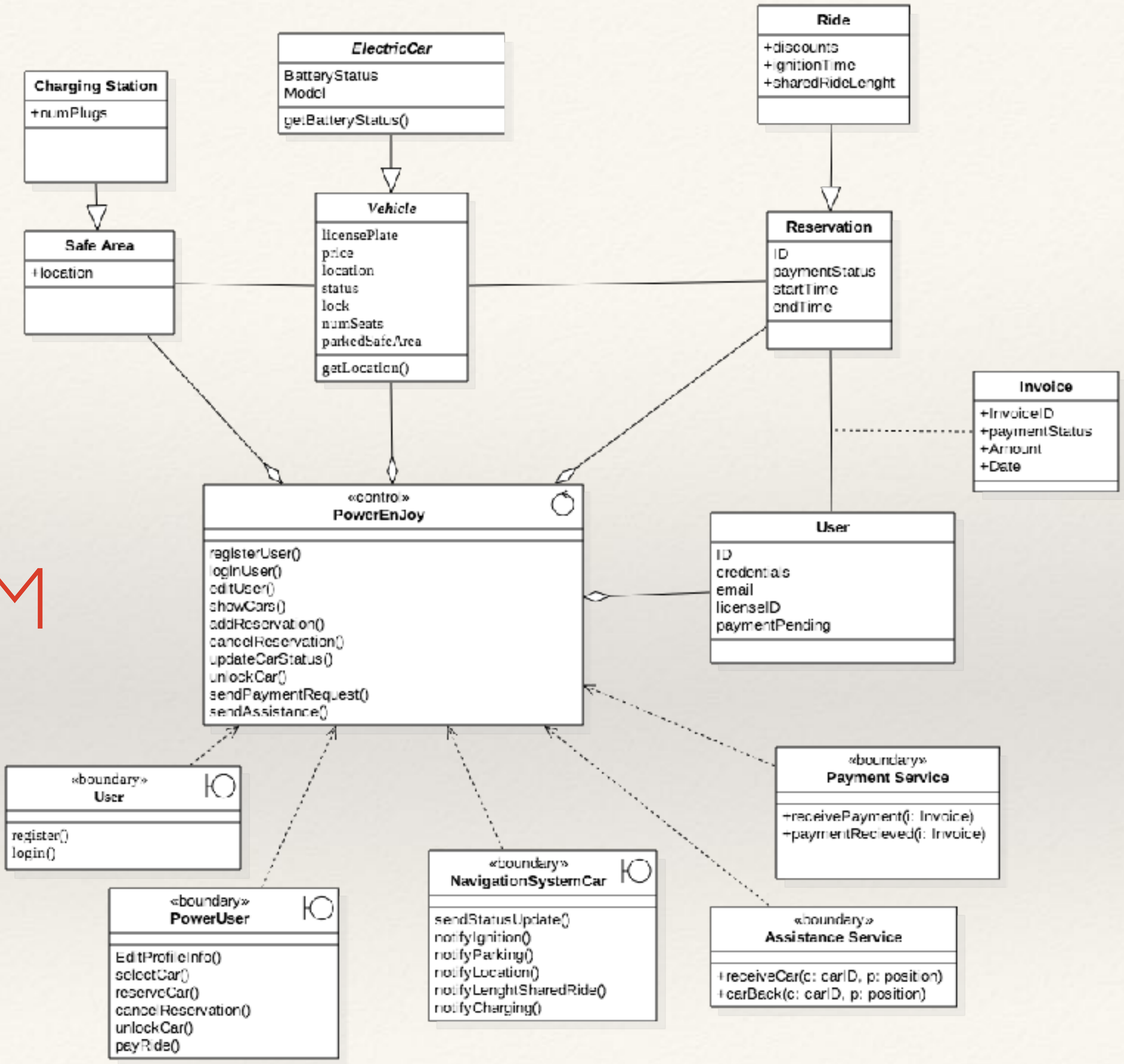
- Power Enjoy service employs a single model of electric car in the first release, but the system is designed to allow any electric vehicle as long as is provides the minimum functionality needed:
  - Weight sensors
  - Ignition sensors
  - Battery Level sensors
  - Global Positioning System (GPS)
  - Automatic keyless entry
  - Remote control
  - Lcd touchscreen
  - Internet connectivity
- Other models should also consider this non functional requirements that will highly affect the quality of the service we provide:
  - Battery Length
  - Charging Time
  - Safety
  - Comfort
  - Performance
  - Navigation System

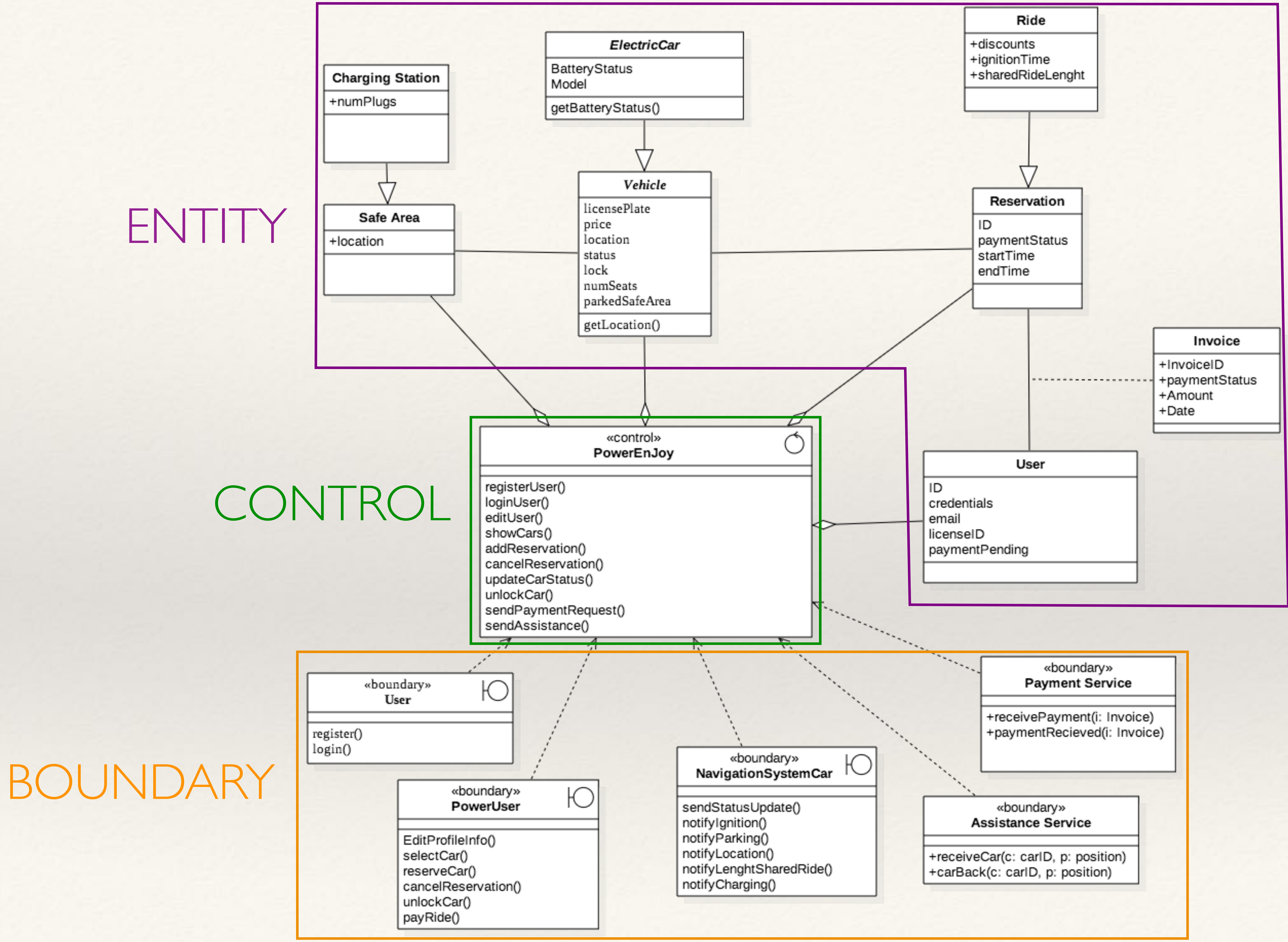
Goal ???

Requirements Derivation

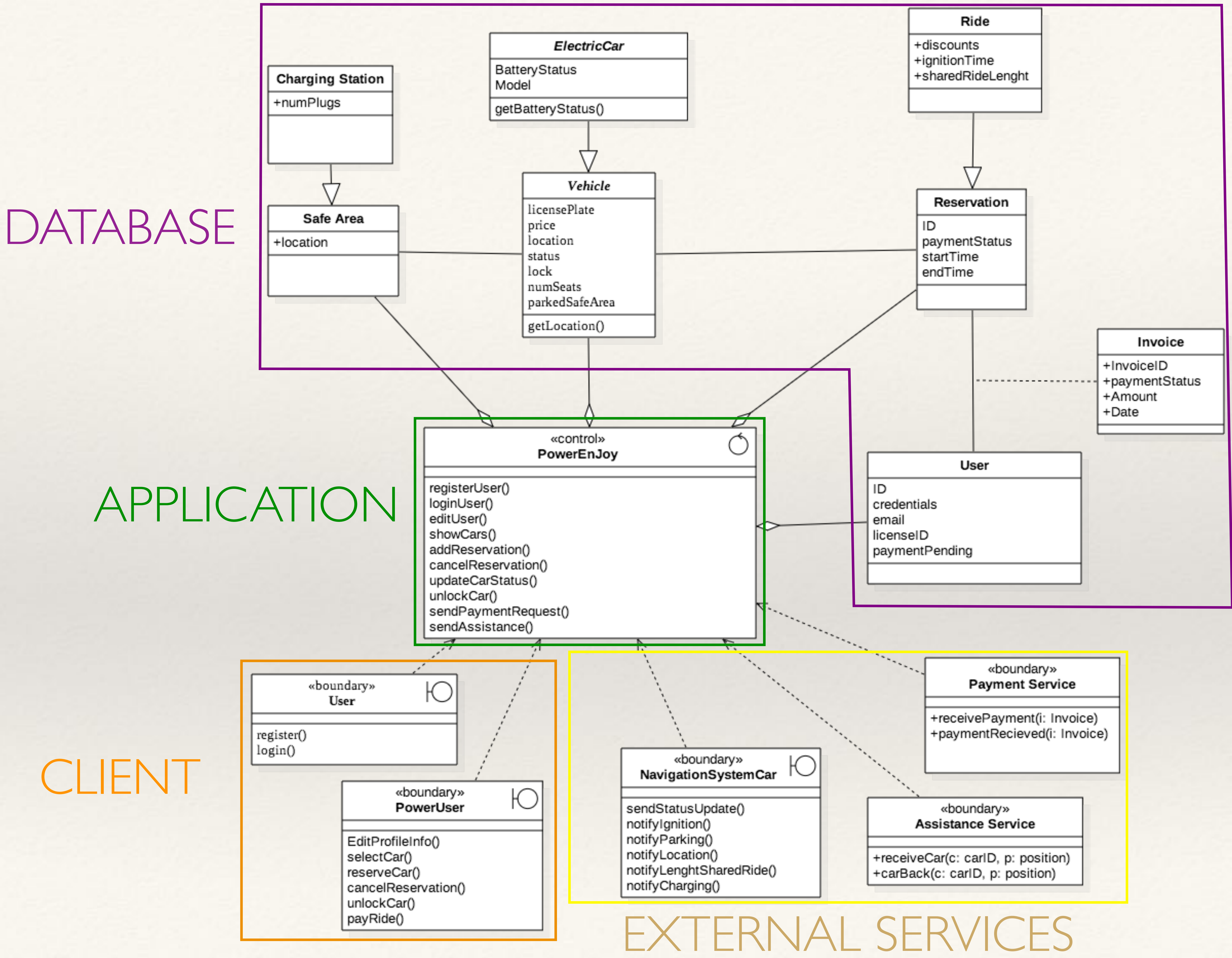
- Scenarios
- Use Cases
- Identification Requirements
- Traceability matrix ensure G = D + R

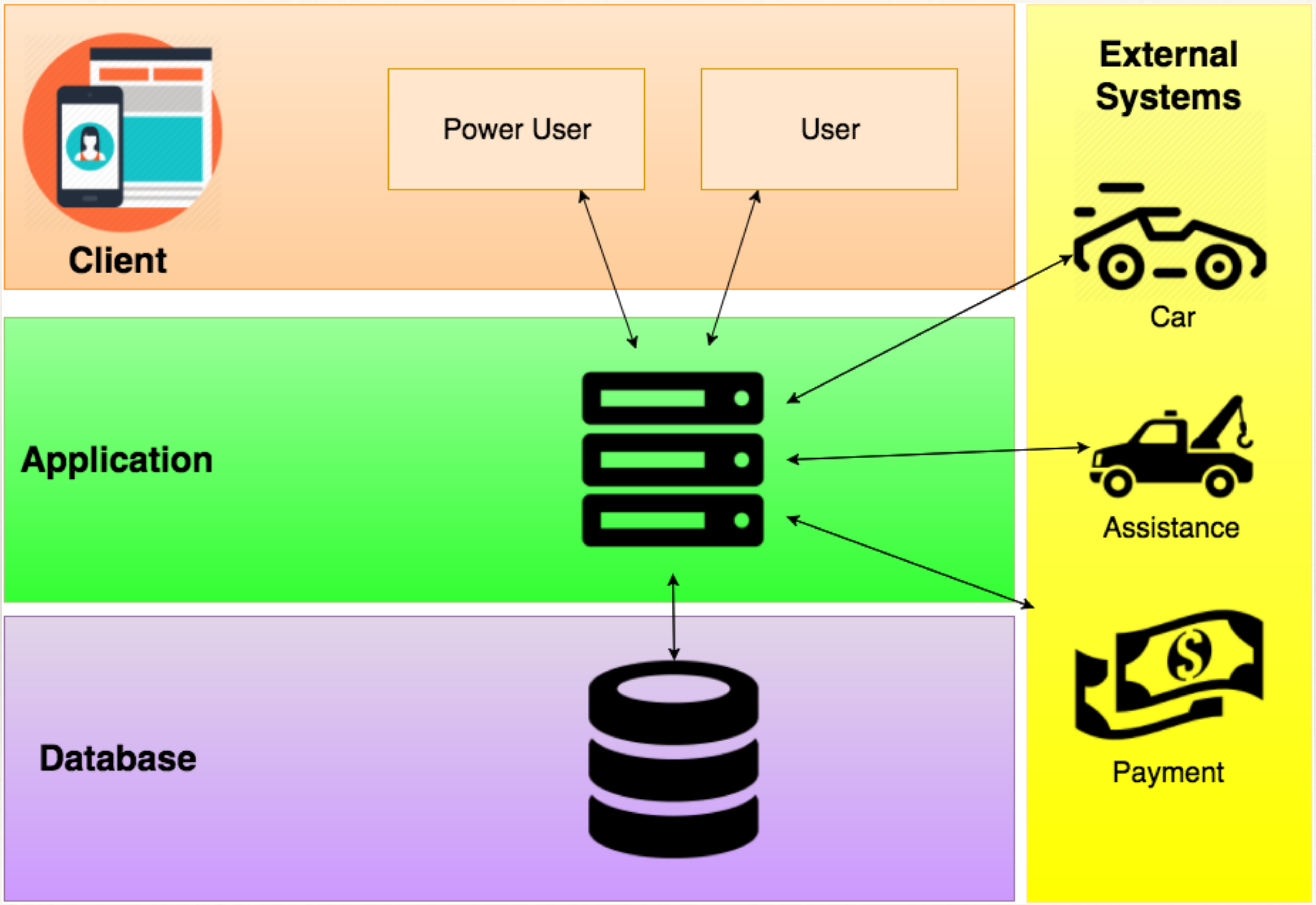
CLASS DIAGRAM





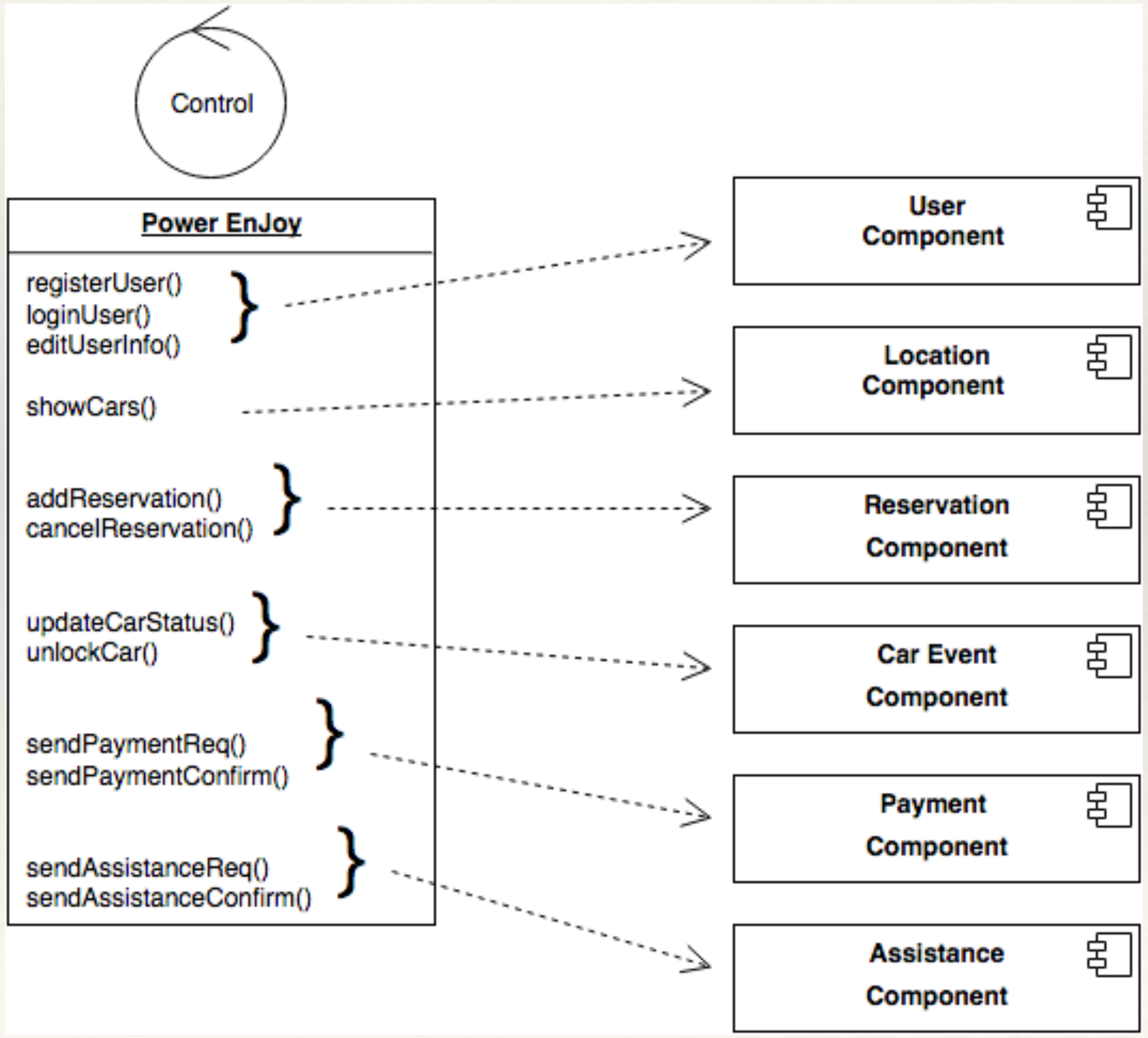
ENTITY
Charging Station
+numPlugs
ElectricCar
BatteryStatus
Model
getBatteryStatus()
Ride
+discounts
+ignitionTime
+sharedRideLenght
Safe Area
+location
Vehicle
licensePlate
price
location
status
lock
numSeats
parkedSafeArea
getLocation()
Reservation
ID
paymentStatus
startTime
endTime
Invoice
+InvoiceID
+paymentStatus
+Amount
+Date
CONTROL
«control»
PowerEnJoy
registerUser()
loginUser()
editUser()
showCars()
addReservation()
cancelReservation()
updateCarStatus()
unlockCar()
sendPaymentRequest()
sendAssistance()
User
ID
credentials
email
licenseID
paymentPending
BOUNDARY
«boundary»
User
register()
login()
«boundary»
PowerUser
EditProfileInfo()
selectCar()
reserveCar()
cancelReservation()
unlockCar()
payRide()
«boundary»
NavigationSystemCar
sendStatusUpdate()
notifyIgnition()
notifyParking()
notifyLocation()
notifyLenghtSharedRide()
notifyCharging()
«boundary»
Payment Service
+receivePayment(i: Invoice)
+paymentRecieved(i: Invoice)
«boundary»
Assistance Service
+receiveCar(c: carID, p: position)
+carBack(c: carID, p: position)


DATABASE
Charging Station
+numPlugs
ElectricCar
BatteryStatus
Model
getBatteryStatus()
Ride
+discounts
+ignitionTime
+sharedRideLenght
Safe Area
+location
Vehicle
licensePlate
price
location
status
lock
numSeats
parkedSafeArea
getLocation()
Reservation
ID
paymentStatus
startTime
endTime
Invoice
+InvoiceID
+paymentStatus
+Amount
+Date
APPLICATION
«control»
PowerEnJoy
registerUser()
loginUser()
editUser()
showCars()
addReservation()
cancelReservation()
updateCarStatus()
unlockCar()
sendPaymentRequest()
sendAssistance()
User
ID
credentials
email
licenseID
paymentPending
CLIENT
«boundary»
User
register()
login()
«boundary»
PowerUser
EditProfileInfo()
selectCar()
reserveCar()
cancelReservation()
unlockCar()
payRide()
«boundary»
NavigationSystemCar
sendStatusUpdate()
notifyIgnition()
notifyParking()
notifyLocation()
notifyLenghtSharedRide()
notifyCharging()
«boundary»
Payment Service
+receivePayment(i: Invoice)
+paymentRecieved(i: Invoice)
«boundary»
Assistance Service
+receiveCar(c: carID, p: position)
+carBack(c: carID, p: position)
EXTERNAL SERVICES


Client
Power User
User
External Systems
Car
Application
Assistance
Payment
Database

Application Layer

- This components implements the logic of the Power Enjoy Application, it's the core of our business



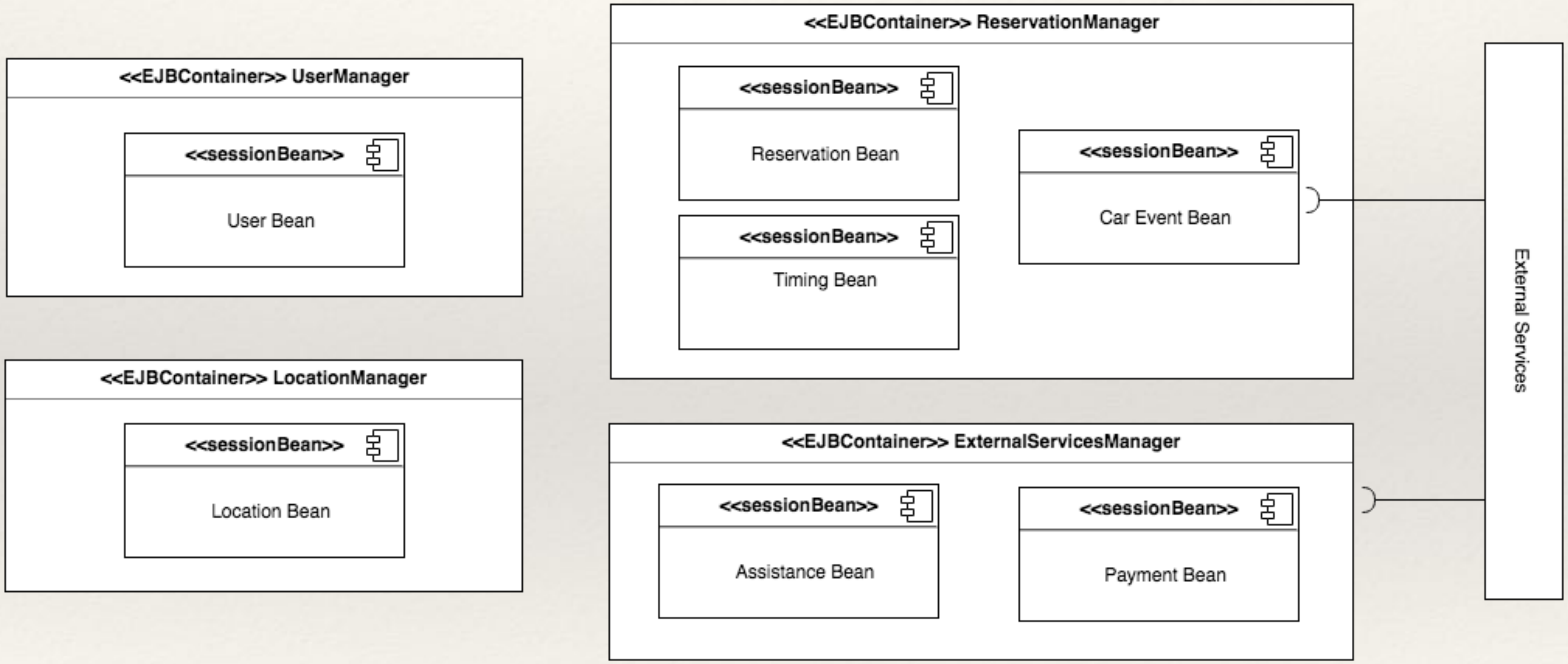
Application Layer - Implementation

- **Java Enterprise Edition 7 (JEE)**
  - Modular Components
  - Large Scale
  - Multi Tiered
  - Scalable

- **Enterprise Java Beans (EJB)**
  - Encapsulate Business Logic

- **GlassFish as Application Server**
  - Supports JEE7
  - Additional Features (Security, Load Balancing)

Application Layer - EJB



Client Layer

- **Considerations:**
  - Mobility In Mind
  - Mobile First
- **Expected Functionalities:**
  - Registration
  - Login
  - Edit Profile
  - See Recent Rides
  - Reservation/Ride
  - Make Payment

Client Layer

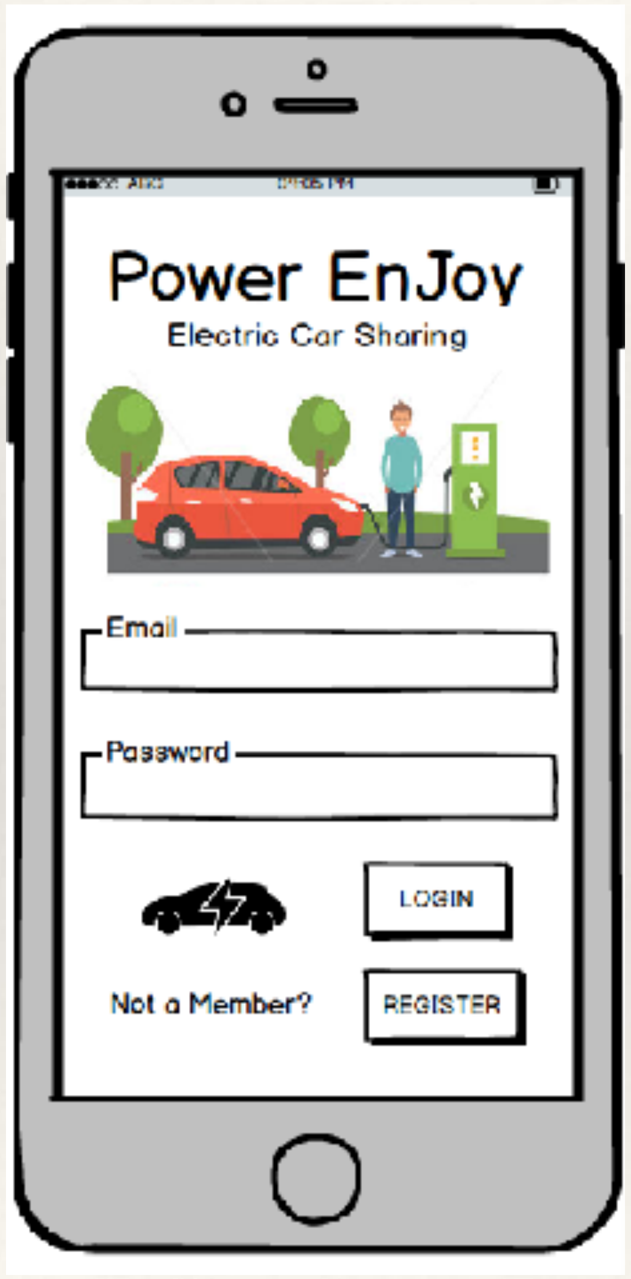
- **Considerations:**
  - Mobility In Mind
  - Mobile First
- **Expected Functionalities:**
  - Registration
  - Login
  - Edit Profile
  - See Recent Rides

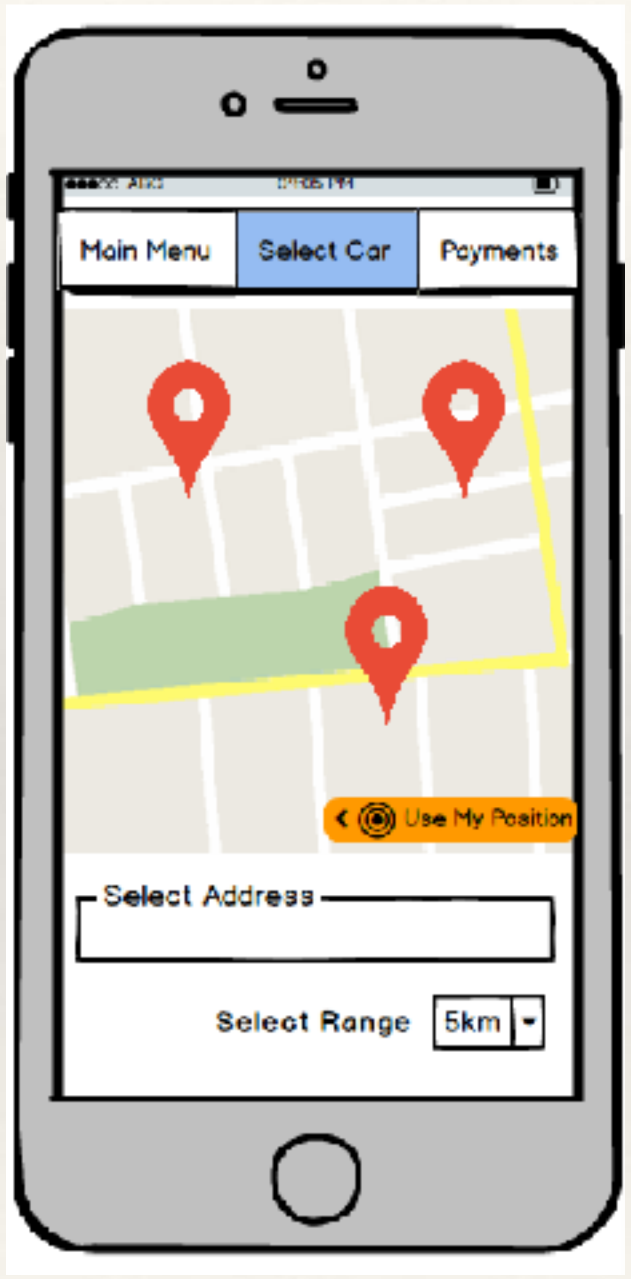
  } Profile Management

  - Reservation/Ride
  - Make Payment

  } Car Sharing

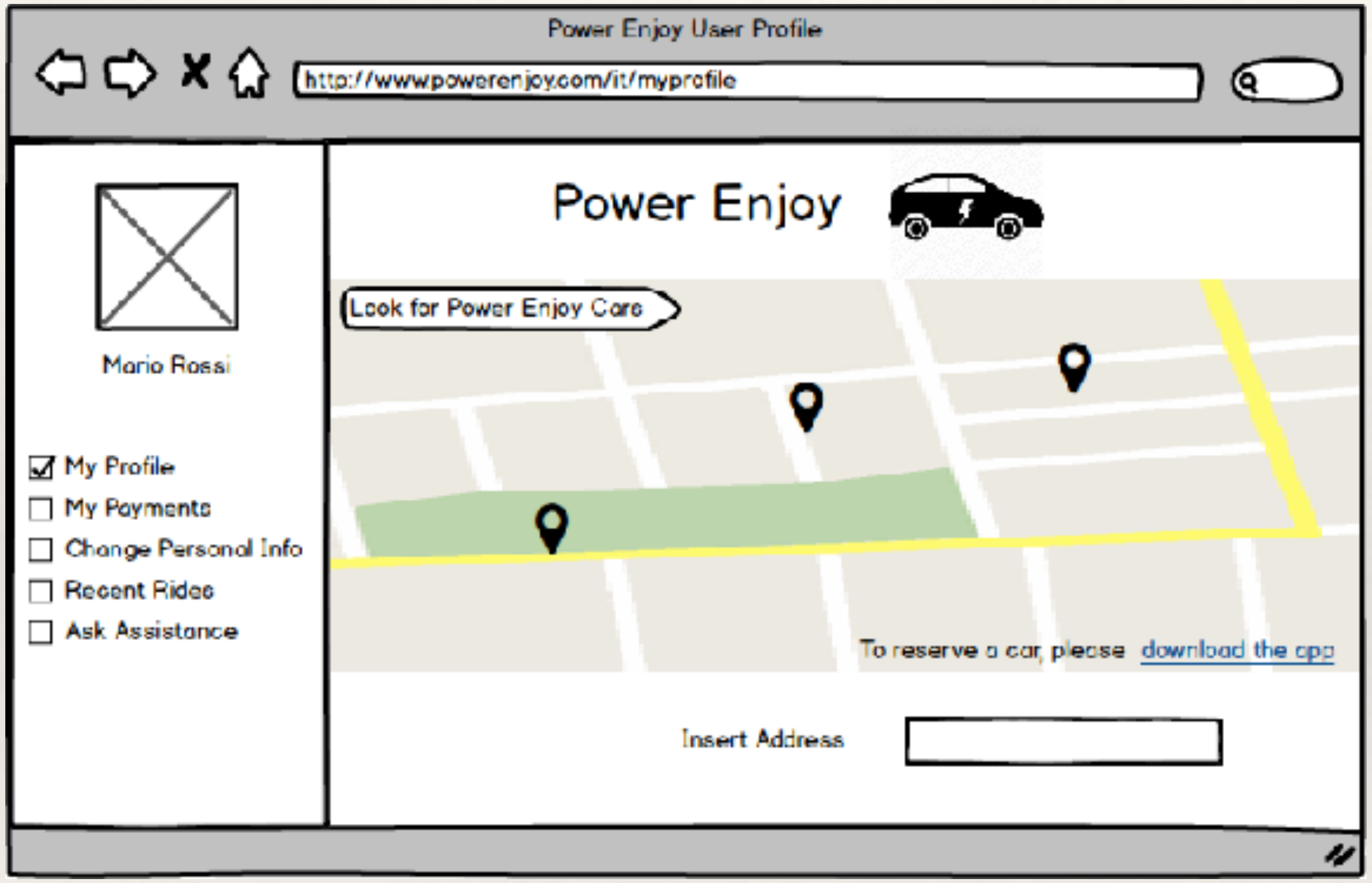
Client Layer - Mobile App



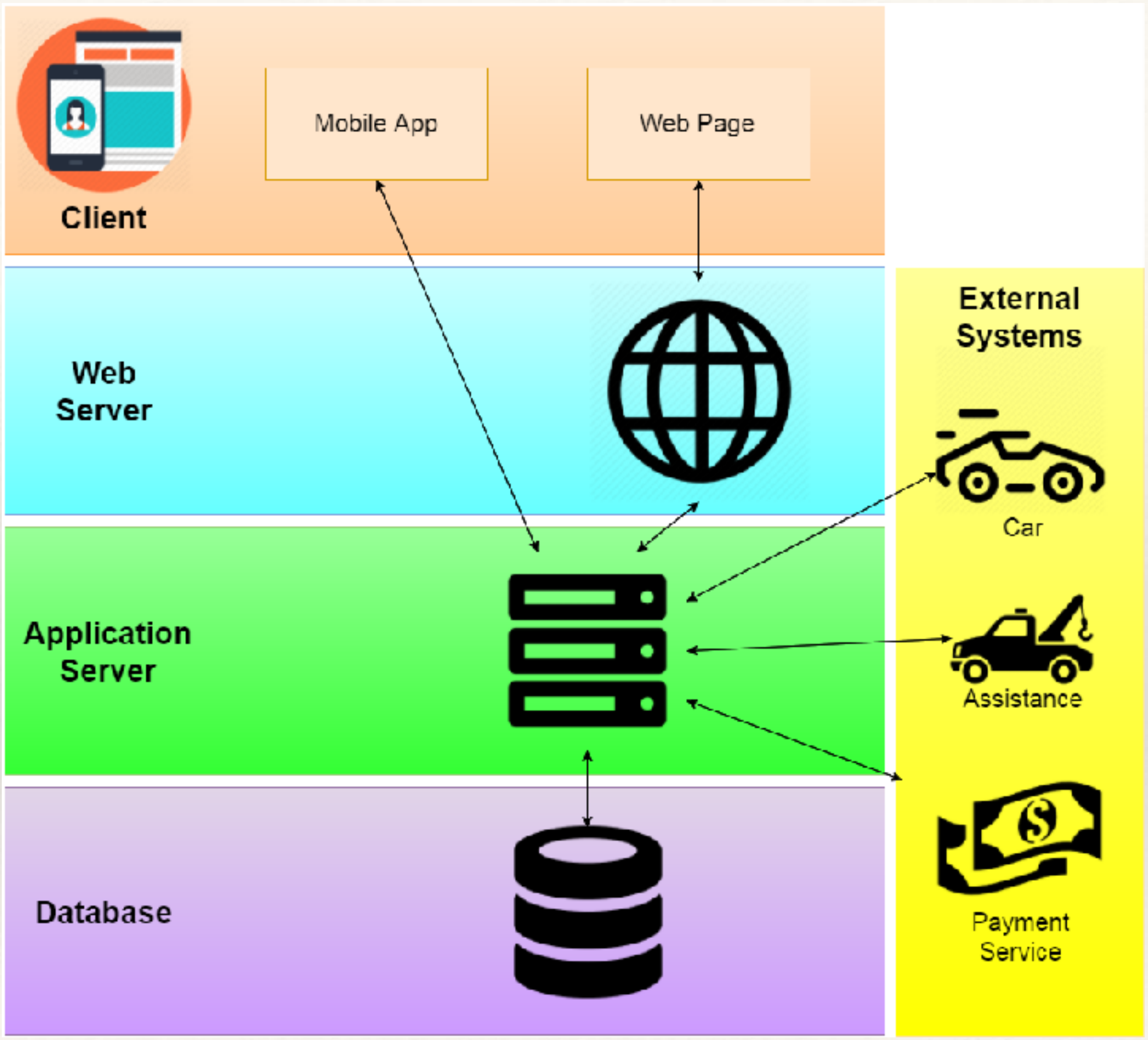


- Mobile App implements all expected functionalities
- Mobile First but not Mobile Only:
  - visibility
  - accessibility
  - scalability

Client Layer - Web Server

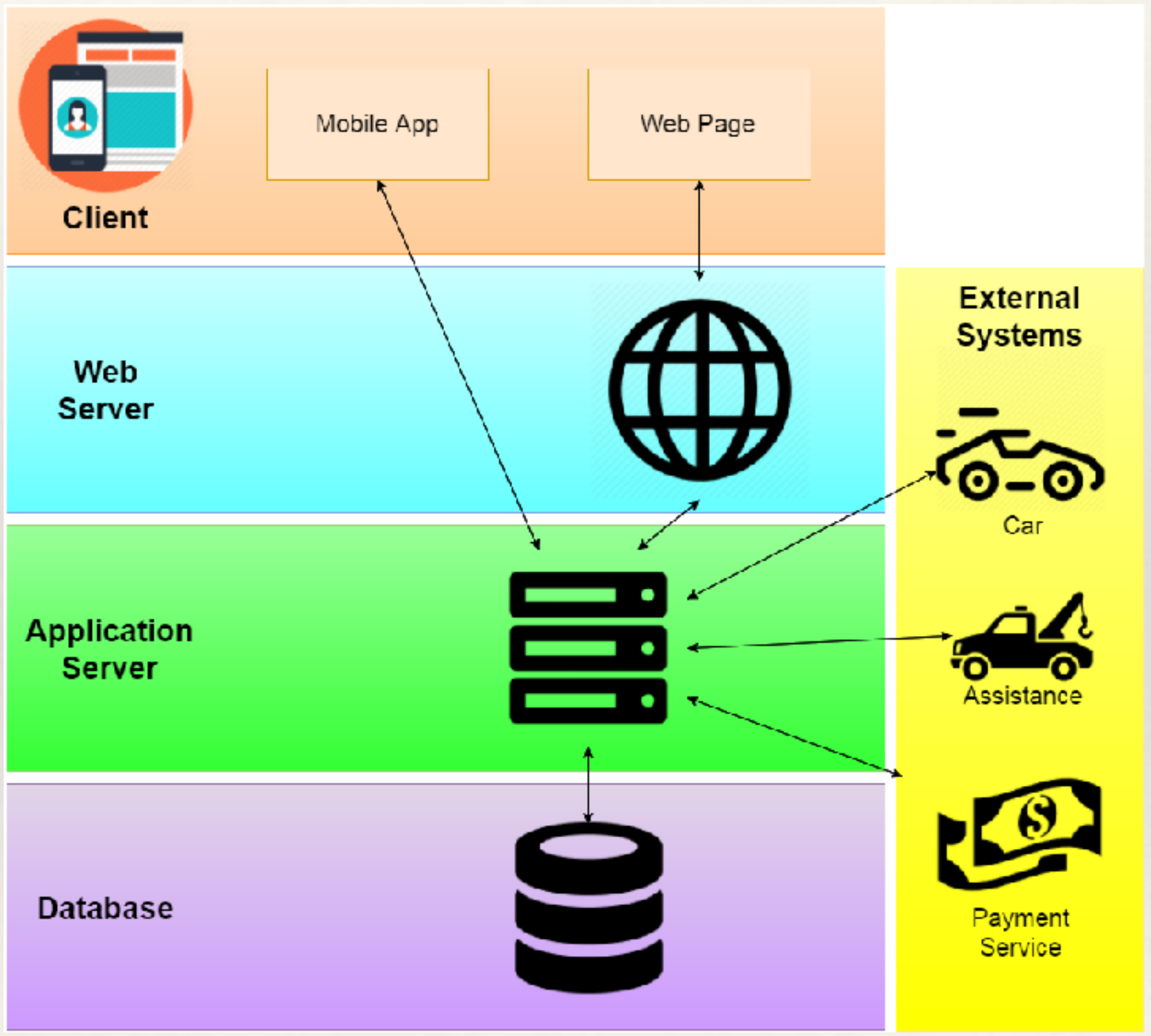


- Support to Mobile App
- New Tier?


Client
Mobile App
Web Page
Web Server
External Systems
Car
Application Server
Assistance
Database
Payment Service

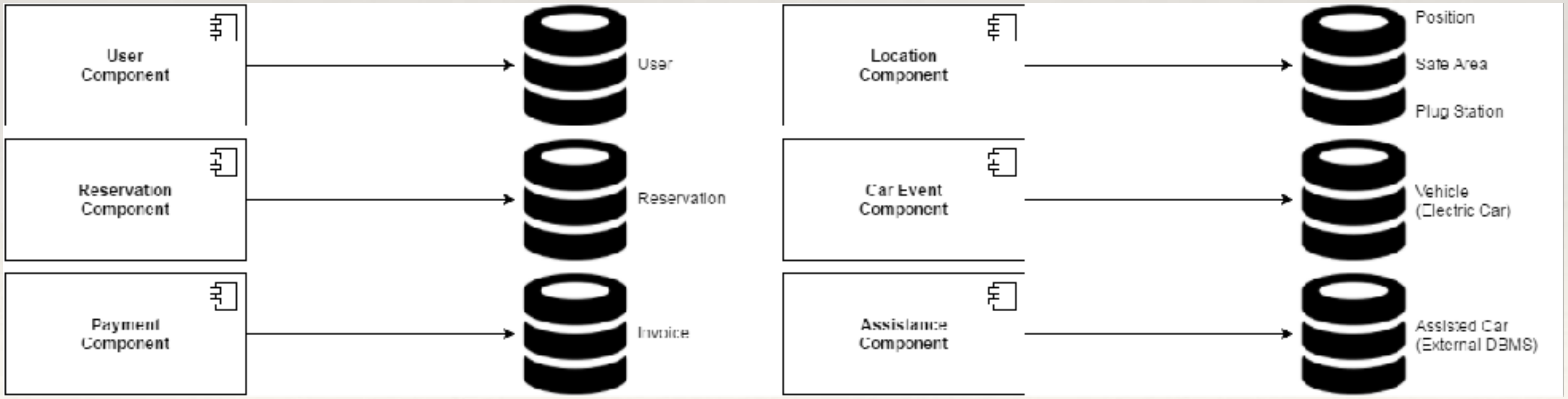
Some Problems…

- **Problems**
- Application Server is the bottleneck of our system
- The performance of this layer is strictly related to the overall performance of the system
- **Solutions**
- Multithreading?
- Sure, but we can do better…



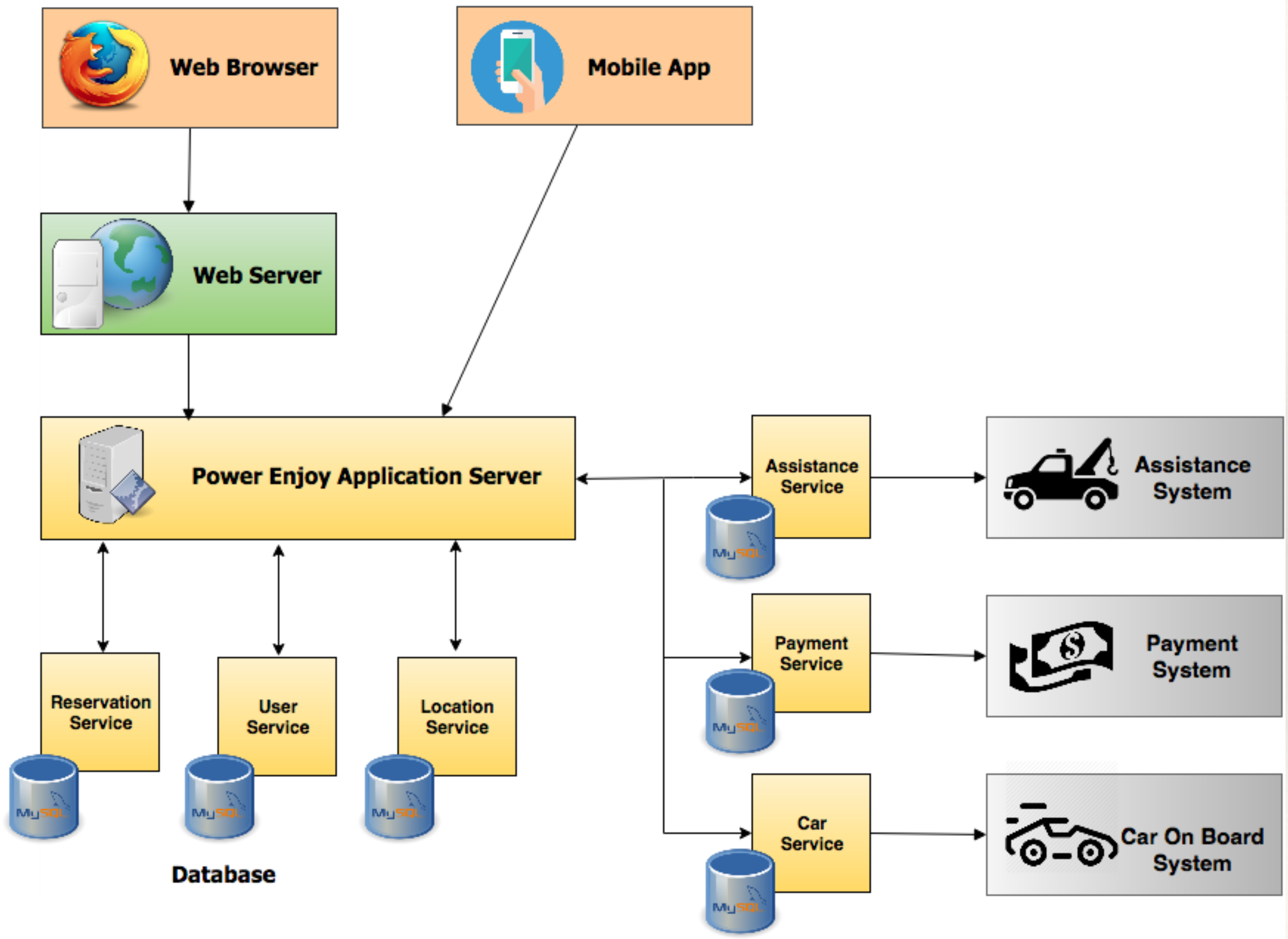
Moving to a SOA approach

- Split the workload among different services
  - Simple and clear Interface to other components
  - Each component is responsible for some entities in the database



Benefits

- It’s a more **clean architecture**. Every component implements a service and provides an interface to all the other services.
- Changing/optimising each module **will not affect the whole system** as long as we maintain the same interface for each component.
- It’s very flexible, it’s will be easy in the future to **add new functionalities**.
- We can **divide the databases among different regions** (e.g. for the city of Milan we don’t need to keep track of the cars in Turin)


Web Browser
Mobile App
Web Server
Power Enjoy Application Server
Assistance Service
Assistance System
Payment Service
Payment System
Car Service
Car On Board System
Reservation Service
User Service
Location Service
MySQL
Database

Entry Criteria

- **Stakeholder Approval**
- **Website and Mobile App**
  - communication between the Application Server and Clients, both via the Mobile App and via the Web Server, must have clearly structured
- **Coding and Testing Application Layer**
  - Unit test
  - Documentation and Code Inspection
  - Object Relational Mapping (each service to its respective database)

Integration and Testing

- **Elements to be integrated are:**
  - Integration of the different services inside the Application Layer
  - Integration of different tiers (Client - Web - Application)
  - Integration and configuration with third party systems (Payment System, Assistance System)

Integration Strategy

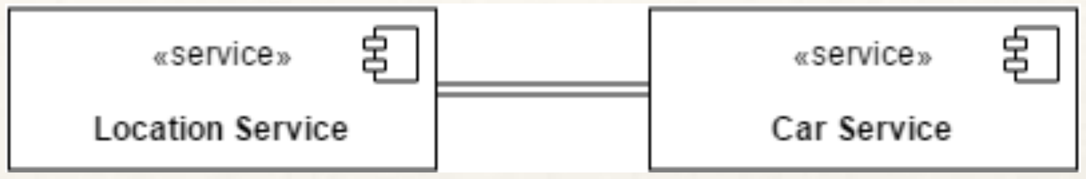
- **Mixture of the bottom-up and functional-grouping**
  - critical components first
  - start from small independent service
  - group them to implement complex functionalities
- **Relation with third party**
  - fixes might delay the process

Integration Strategy

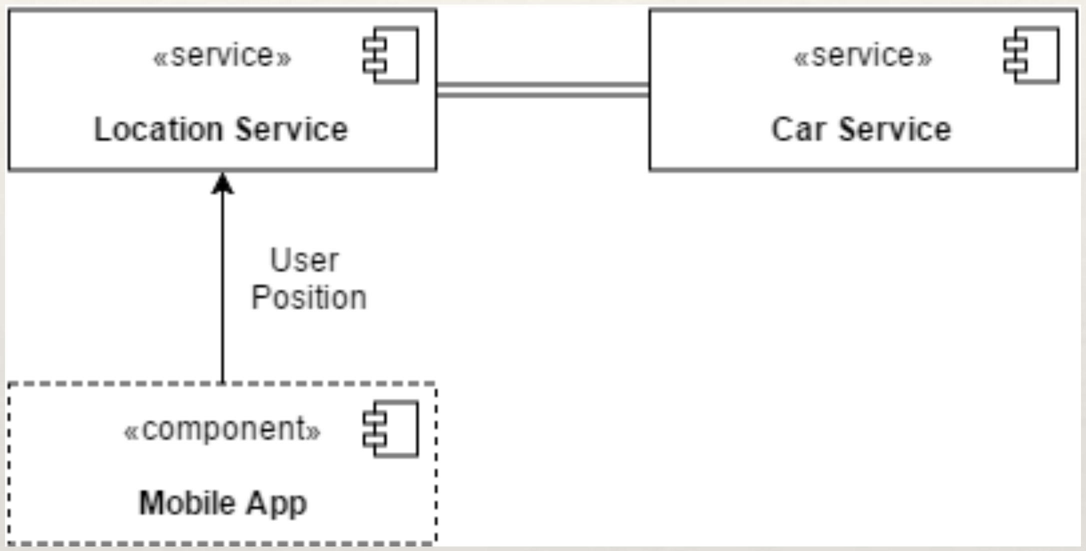
1. Ensure that services in relations with external systems works as expected
2. Ensure that we have control over the Car
3. Integration of Services
4. Integration with top layers
5. Alpha Test

Integration of Services

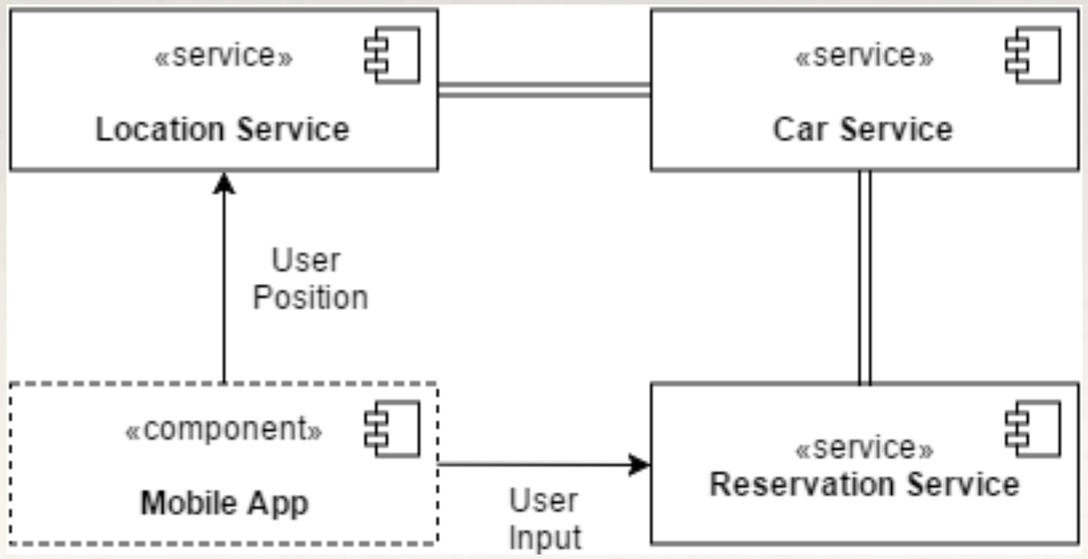
LOCATION OF VEHICLES



LOCATION OF USER

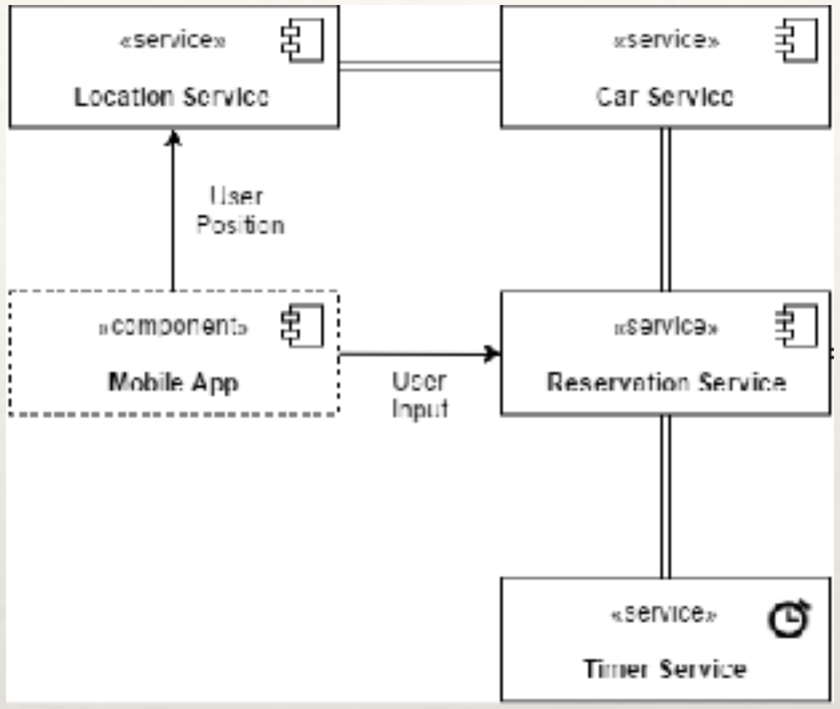


RESERVATION

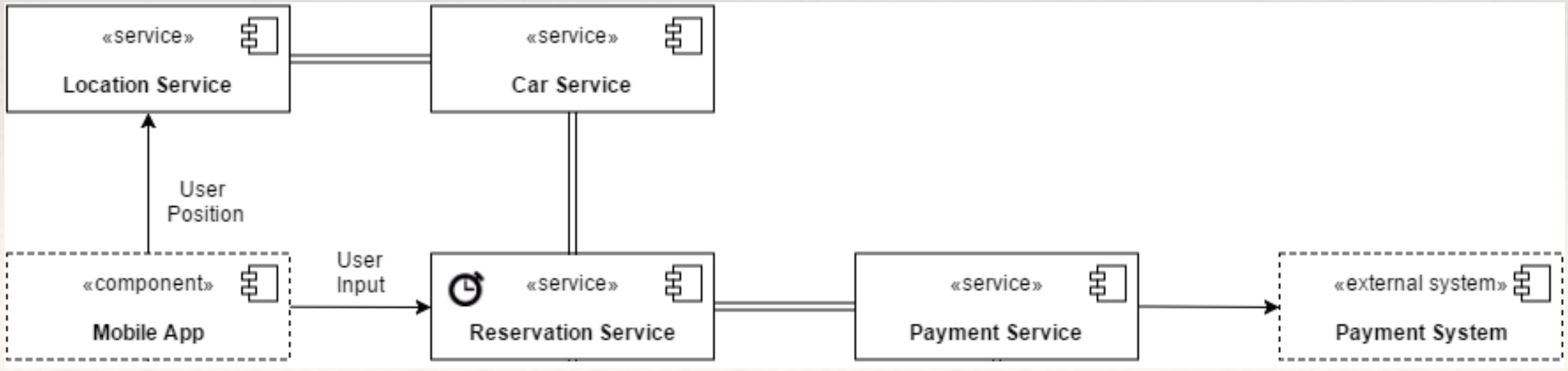


Integration of Services

TIMING

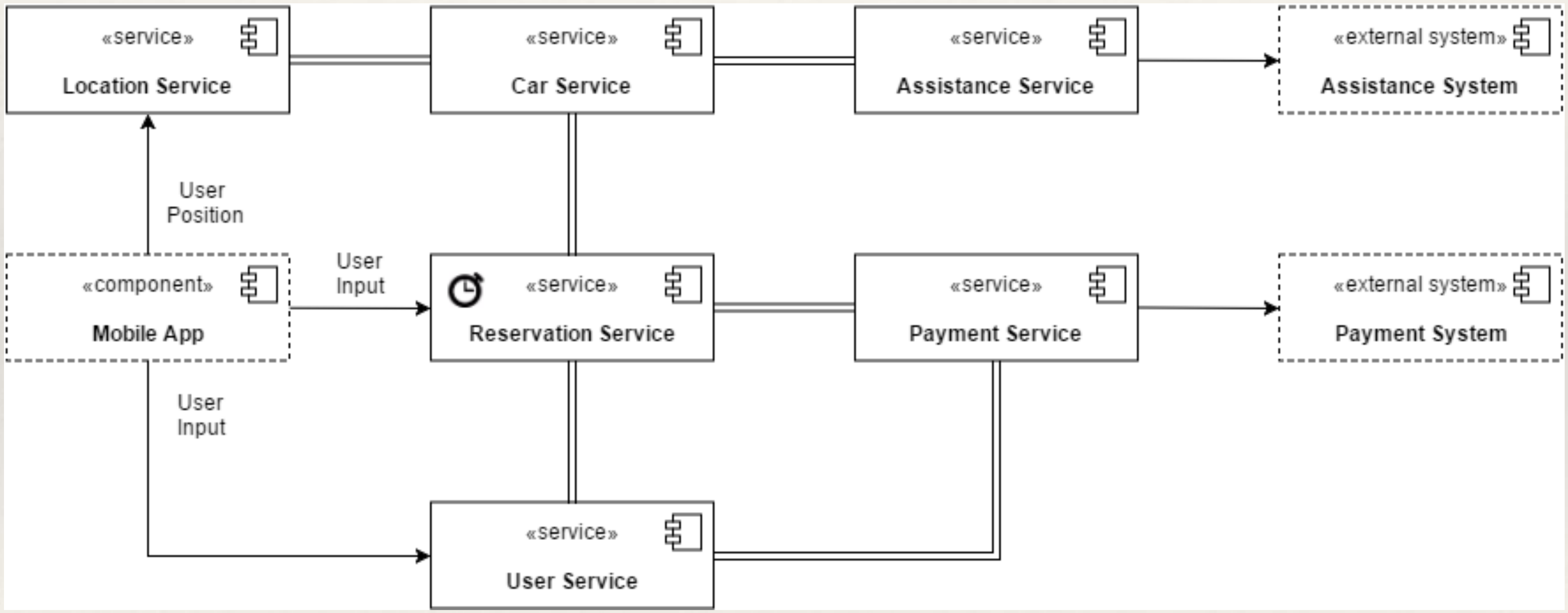


PAYMENT



Integration of Services

WHOLE SYSTEM



Tools Used

- Mockito
- Arquilian
- JUnit
- Manual Testing