Software Engineering 2 Project

Power Enjoy

Niccolo' Raspa
Matteo Marinelli

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

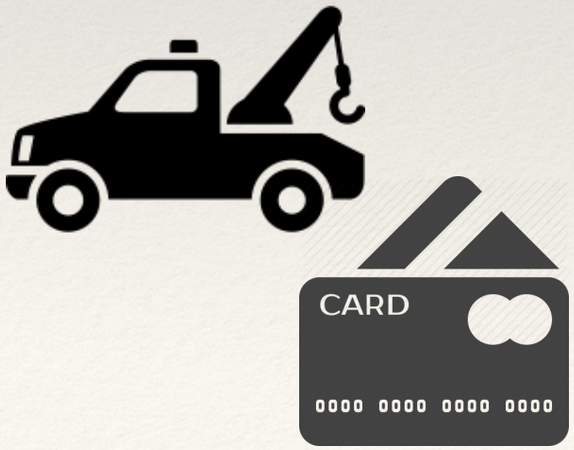
Goals

- User Functionalities
  - Allows USER to login to the system
- Reservation Functionalities
  - Allows POWER USER to reserve one AVAILABLE CAR in a SAFE AREA.
- Reservation Management
  - CAR RESERVATION expires after one hour
- User Interactions
  - POWER USER with pending payments can’t reserve cars

Domain Assumption

CORRECTNESS AND AVAILABILITY
OF INFORMATIONS

CAR FUNCTIONALITIES



EXTERNAL SERVICES

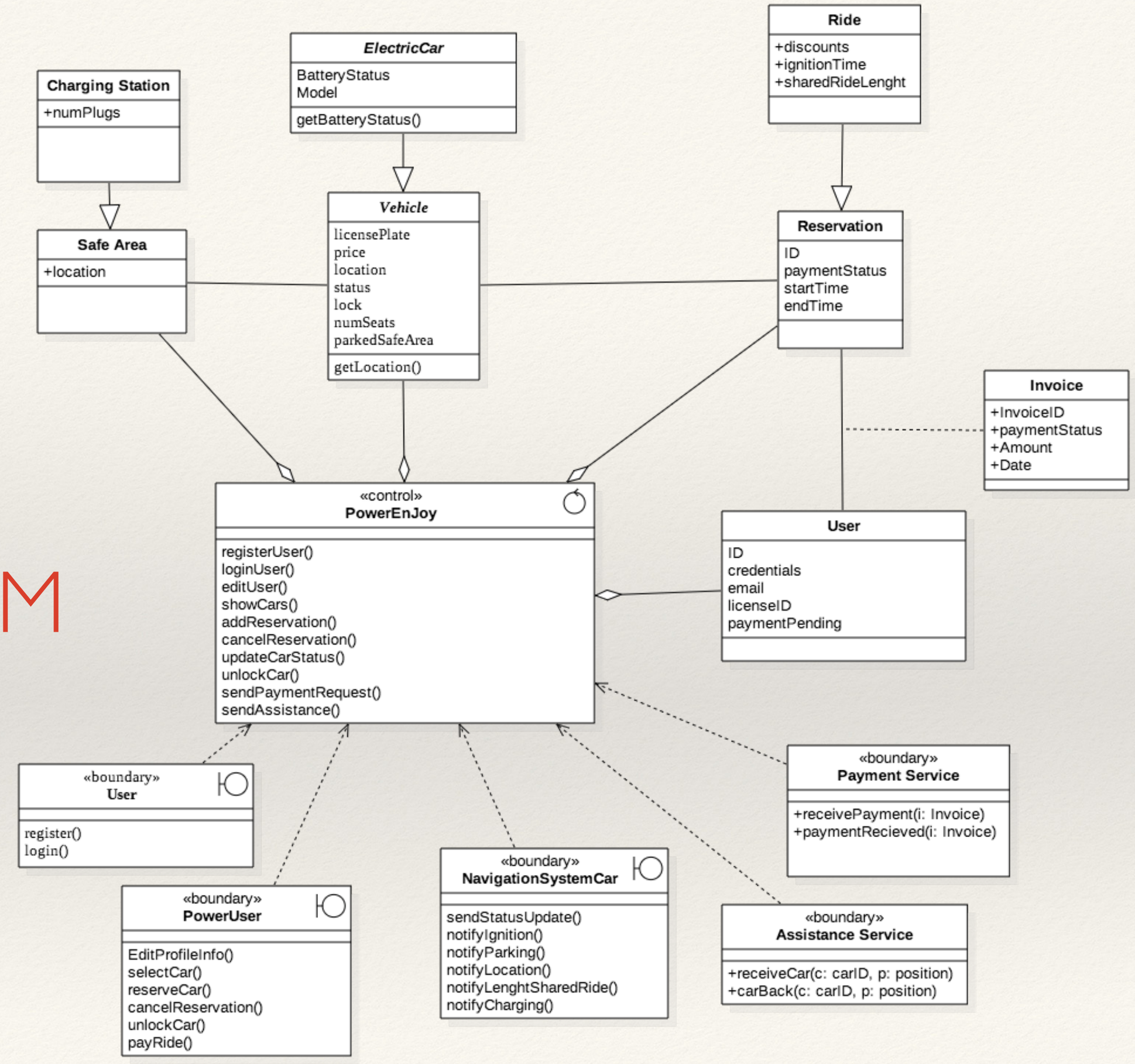
Domain Assumption - Car

- Power Enjoy service employs a particular model of electric car with specific functionalities:
  - **Weight sensors**
  - Ignition sensors
  - Battery Level sensors
  - Global Positioning System (GPS)
  - Automatic keyless entry
  - **Remote control**
  - **Lcd touchscreen**
  - **Internet connectivity**
- Models should also consider this non functional requirements :
  - Battery Length
  - Charging Time
  - Safety
  - Comfort
  - Performance
  - Navigation System

Requirements Derivation

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

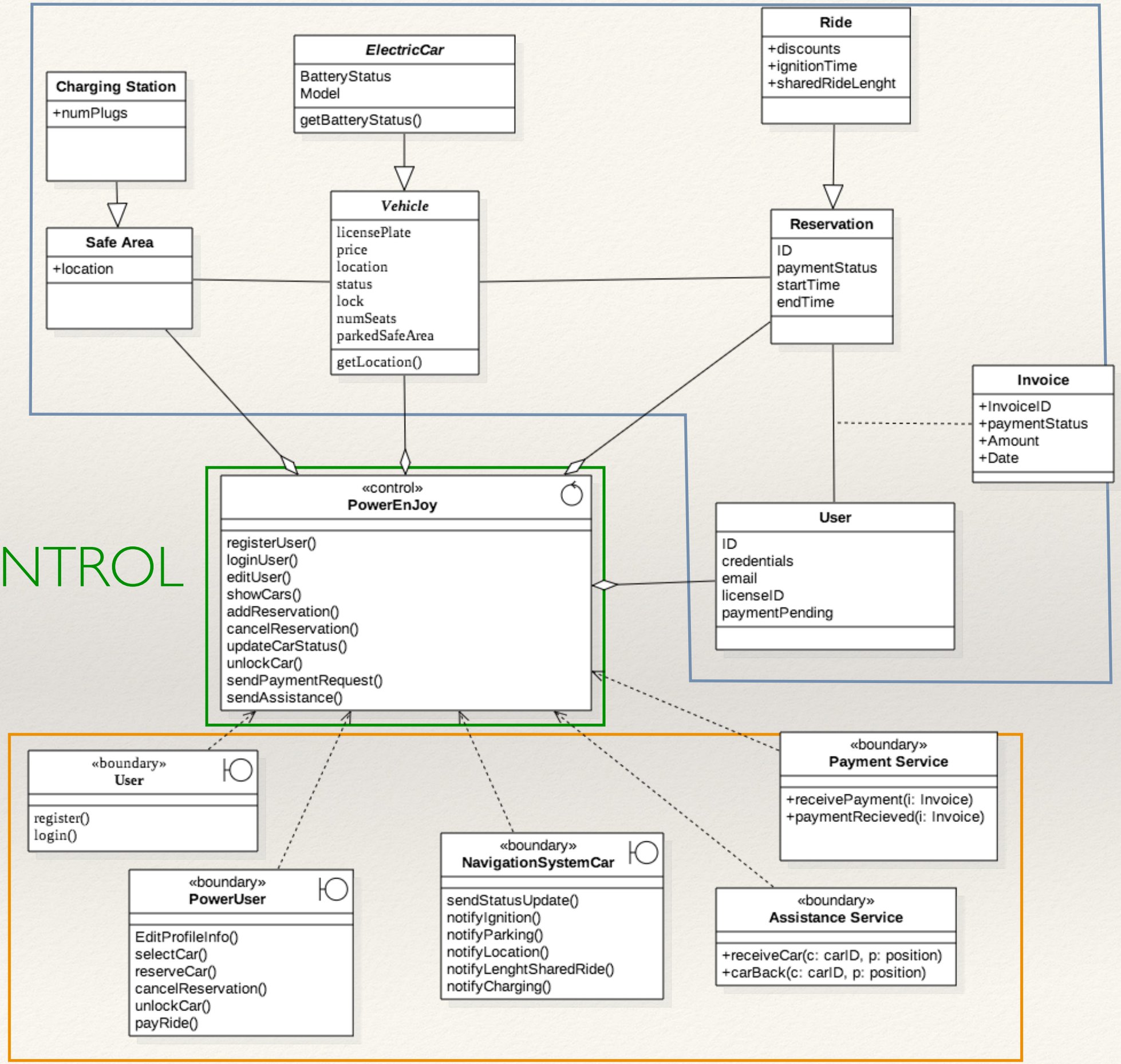
CLASS DIAGRAM



ENTITY

CONTROL

BOUNDARY

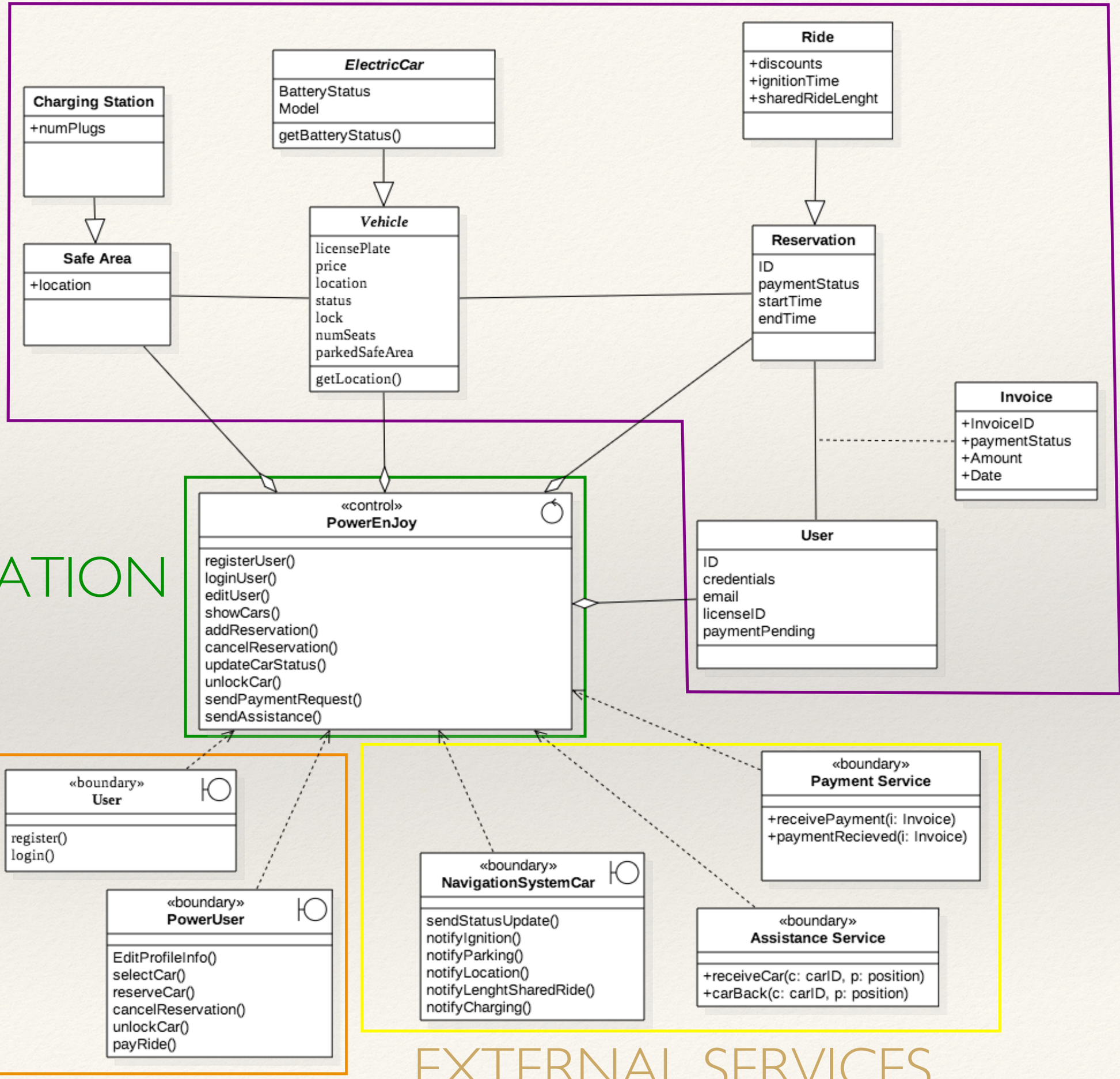


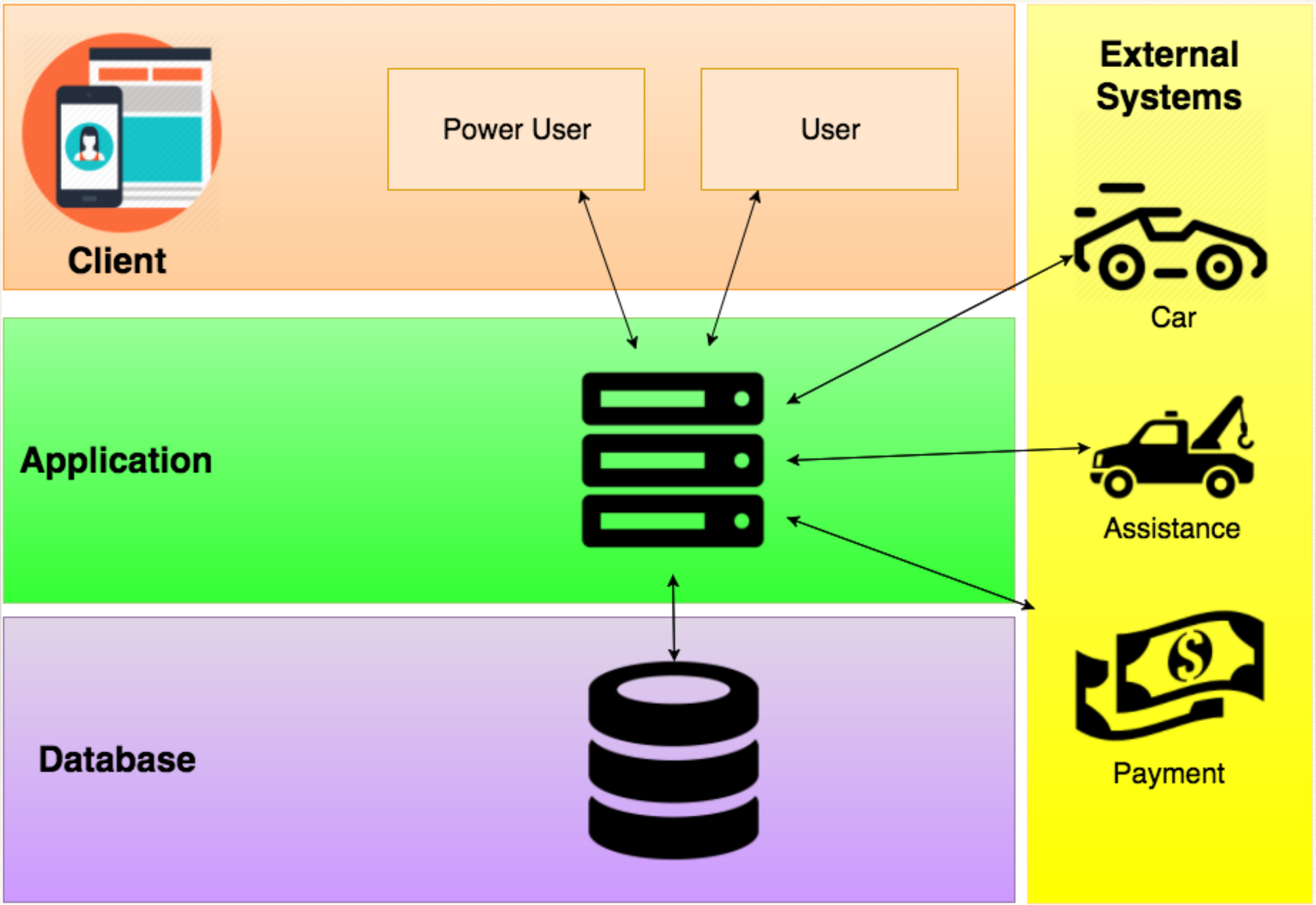
DATABASE

APPLICATION

CLIENT

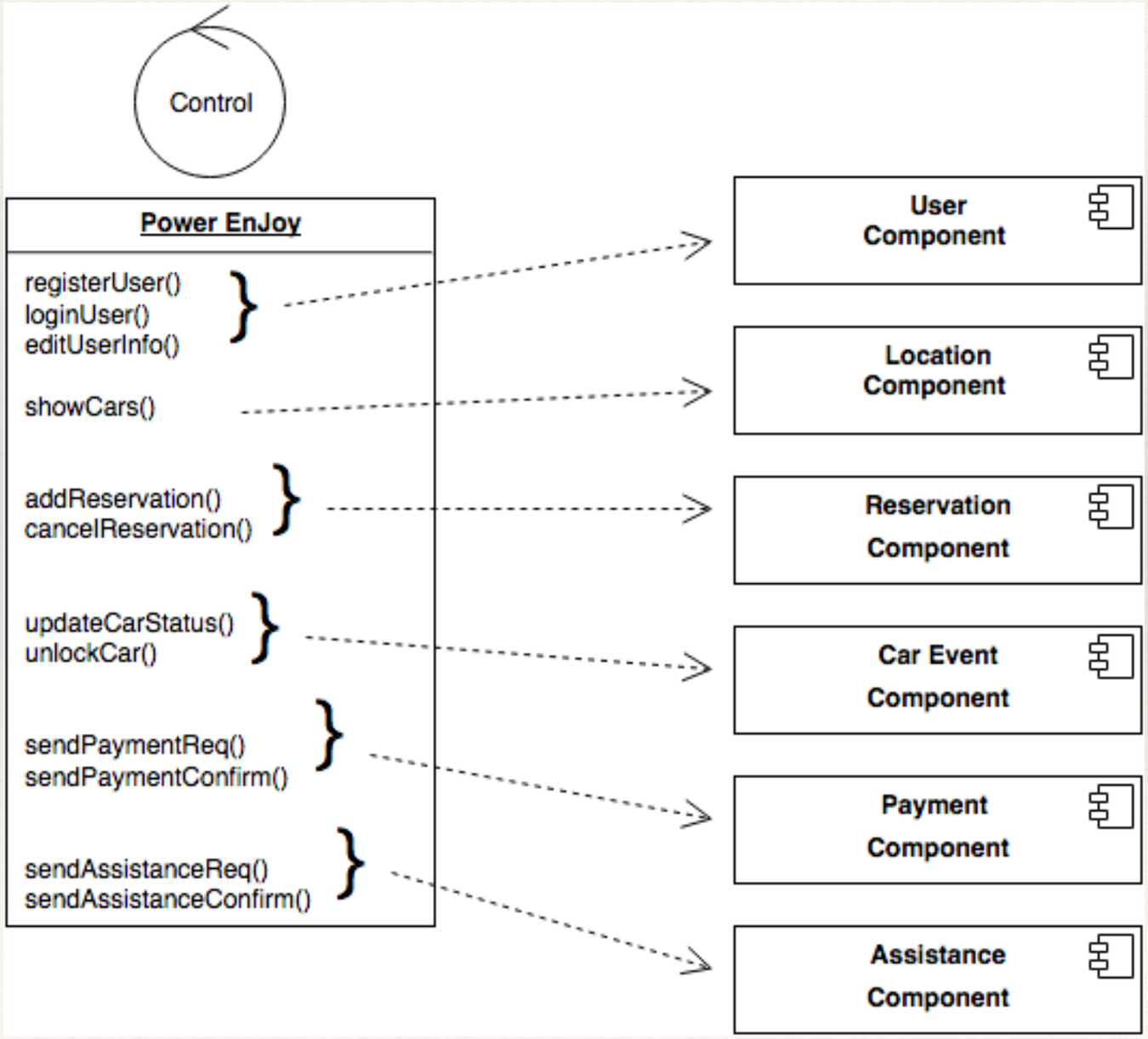
EXTERNAL SERVICES




Client
Power User
User
External Systems
Car
Assistance
Payment
Application
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)

Client Layer

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

Client Layer

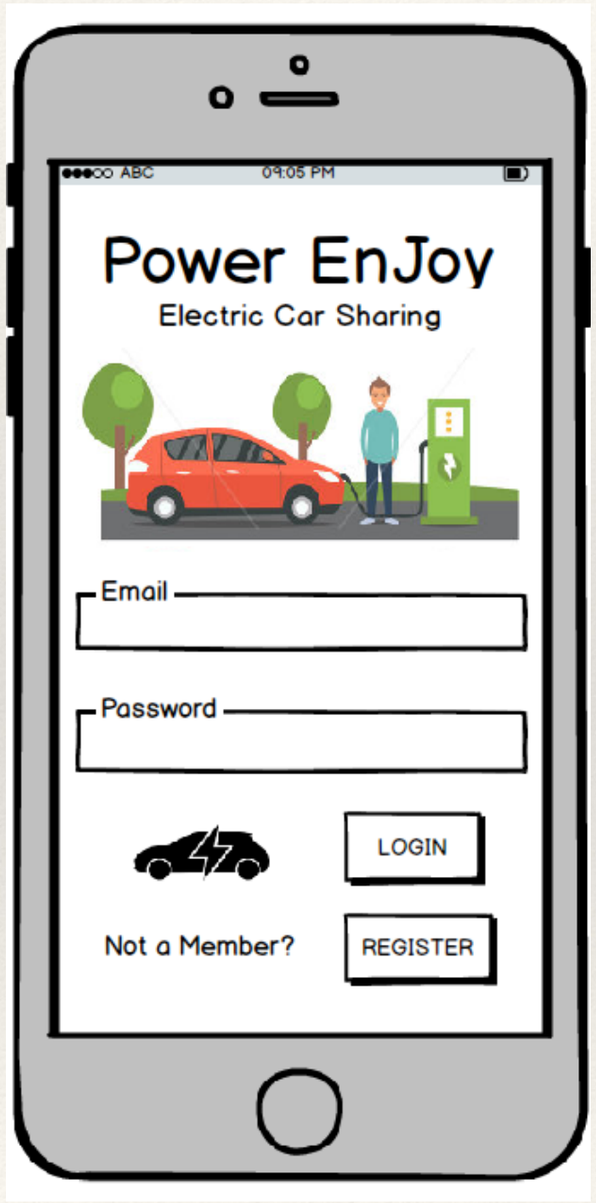
- **Considerations:**
  - Mobility In Mind
  - Mobile First
- **Expected Functionalities:**
  - Registration
  - Login
  - Edit Profile
  - See Recent Rides

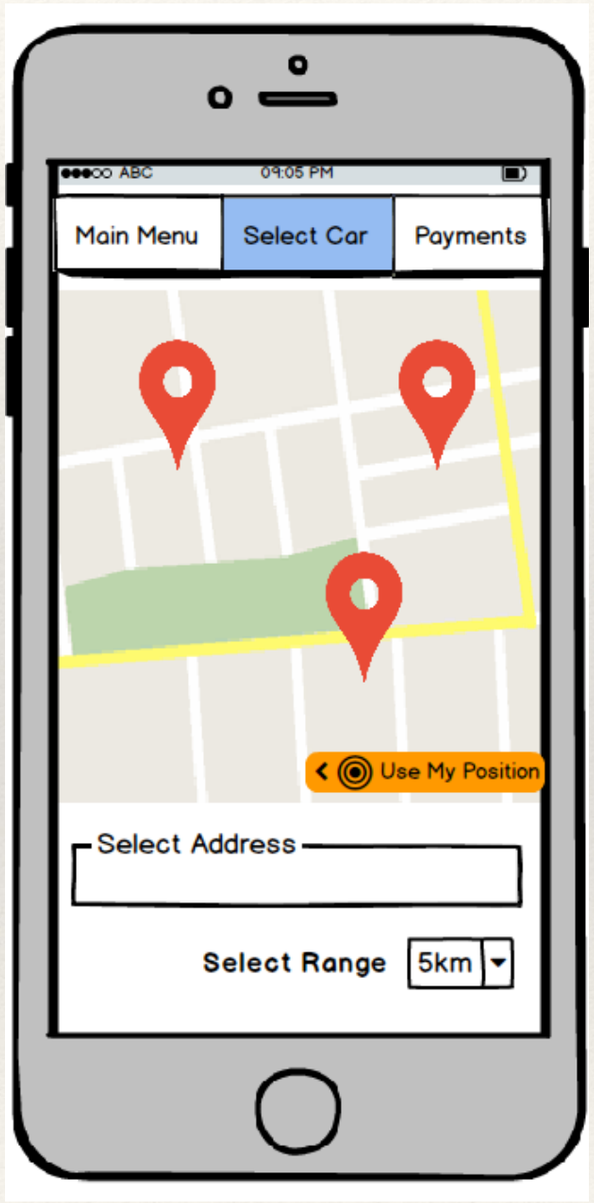
  } Profile Management

  - Reservation/Ride
  - Make Payment

  } Car Sharing

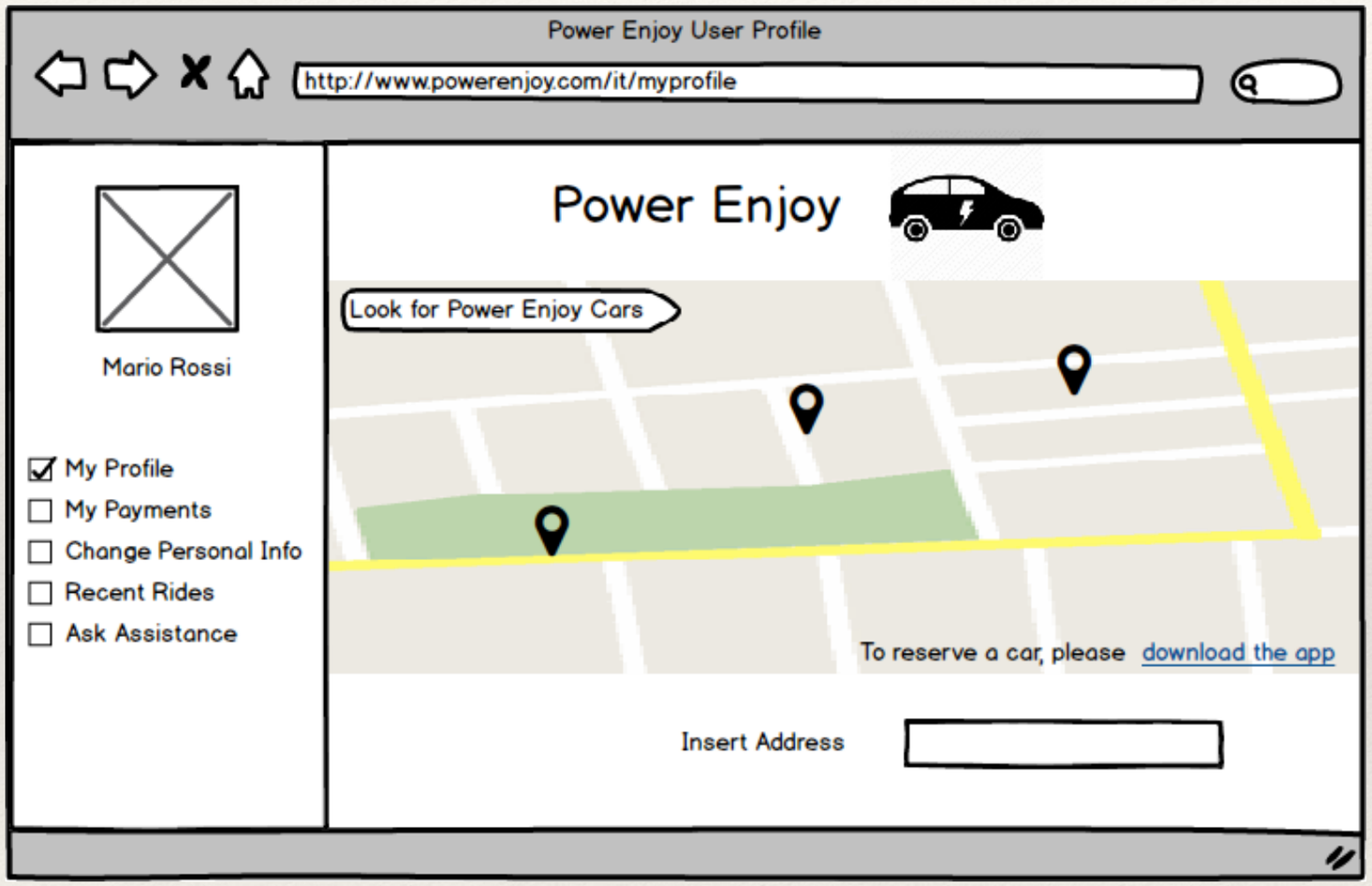
Client Layer - Mobile App



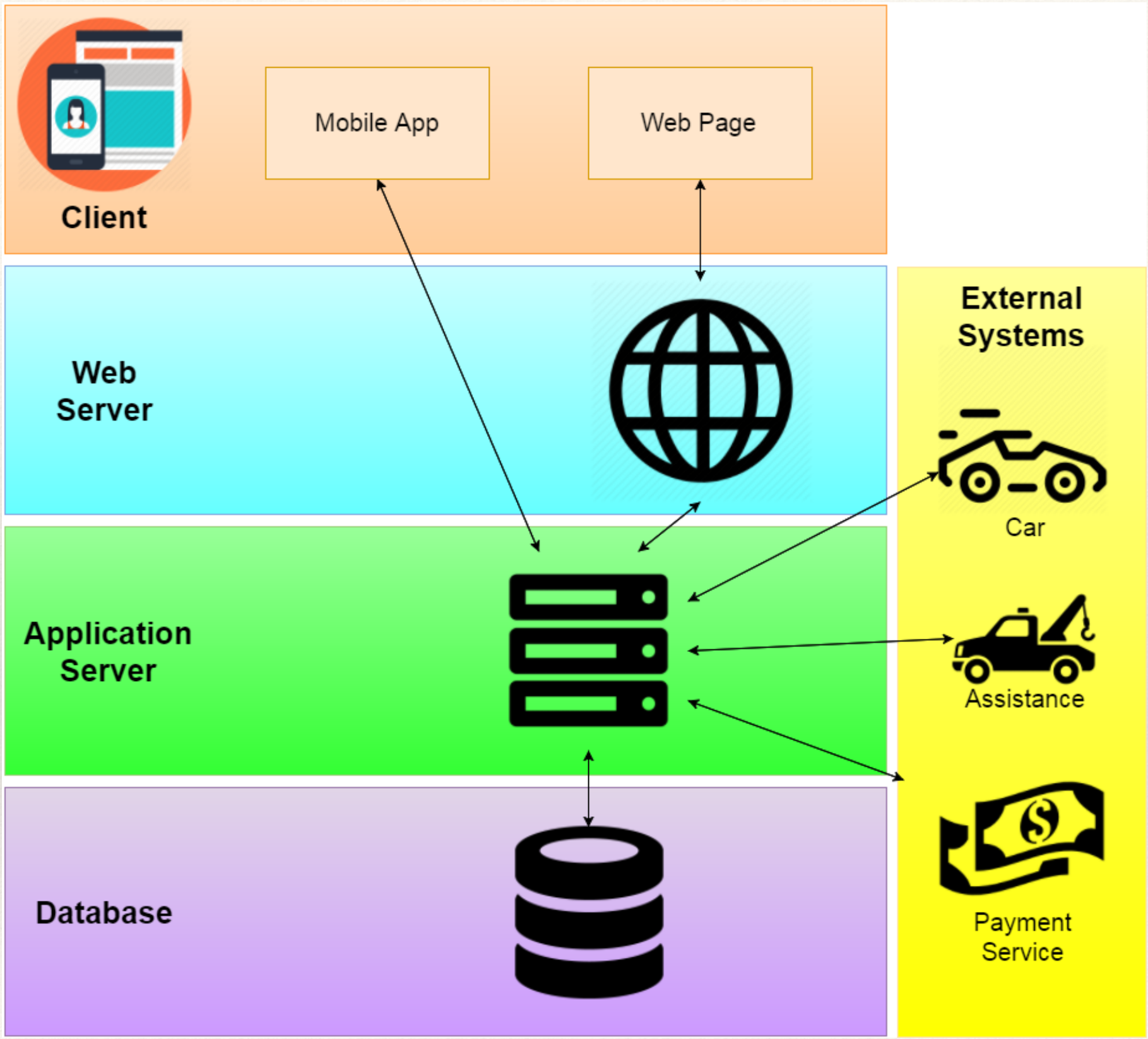


- Mobile App implements all expected functionalities
- Mobile First but not Mobile Only:
  - visibility
  - accessibility
  - scalability

Client Layer - Web Server



- Support to Mobile App
- New Tier?

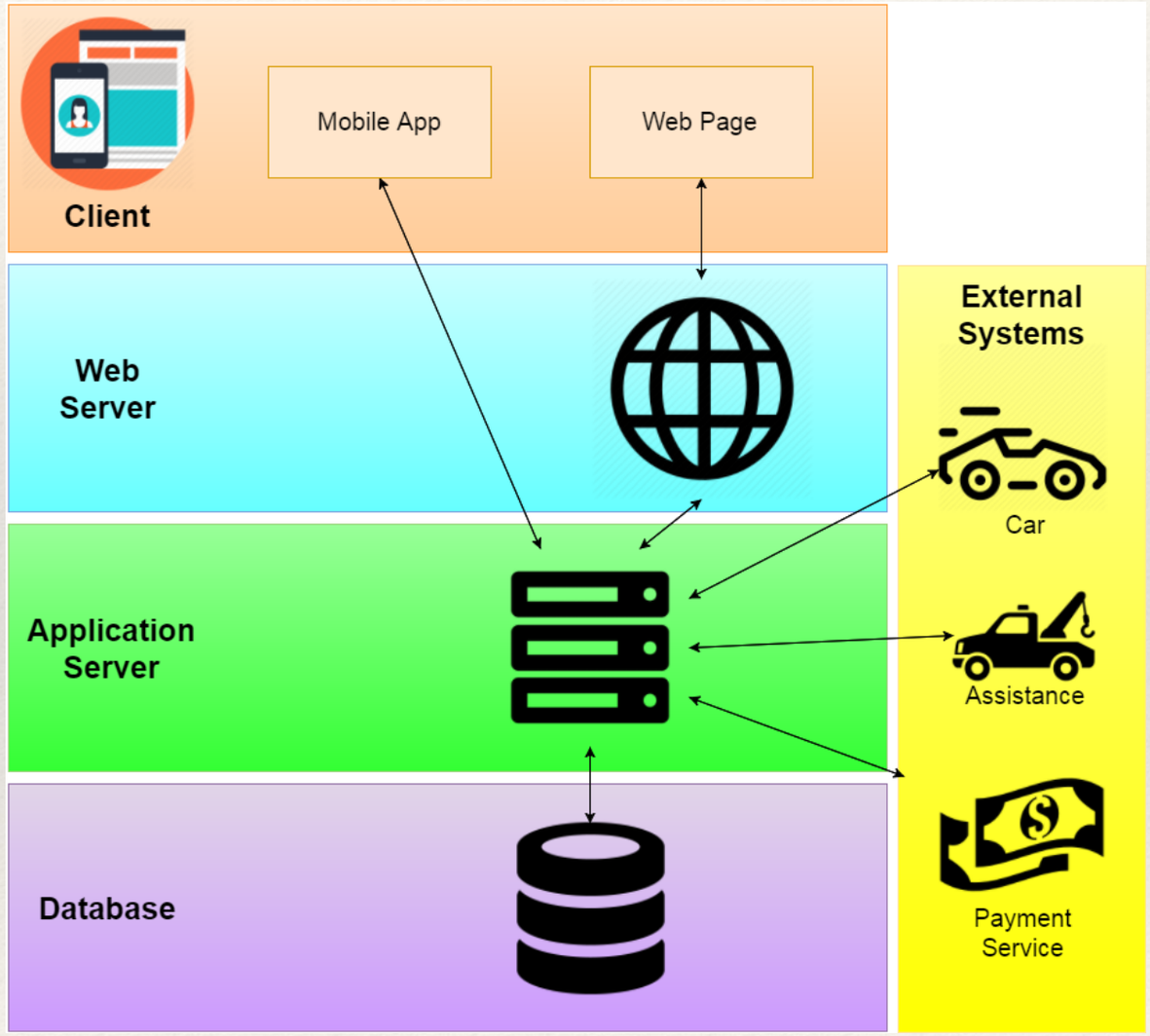

Client
Mobile App
Web Page
Web
Server
External
Systems
Car
Application
Server
Assistance
Database
Payment
Service

Client Layer - Implementation

- **Web Server:**
  - GlassFist with Java Server Pages
  - Communication to Application Server via RESTful APIs
- **Mobile App:**
  - Cordova
  - Cost Effective: free
  - Easy to modify: open source
  - Resource Effective: target multiple devices with one codebase

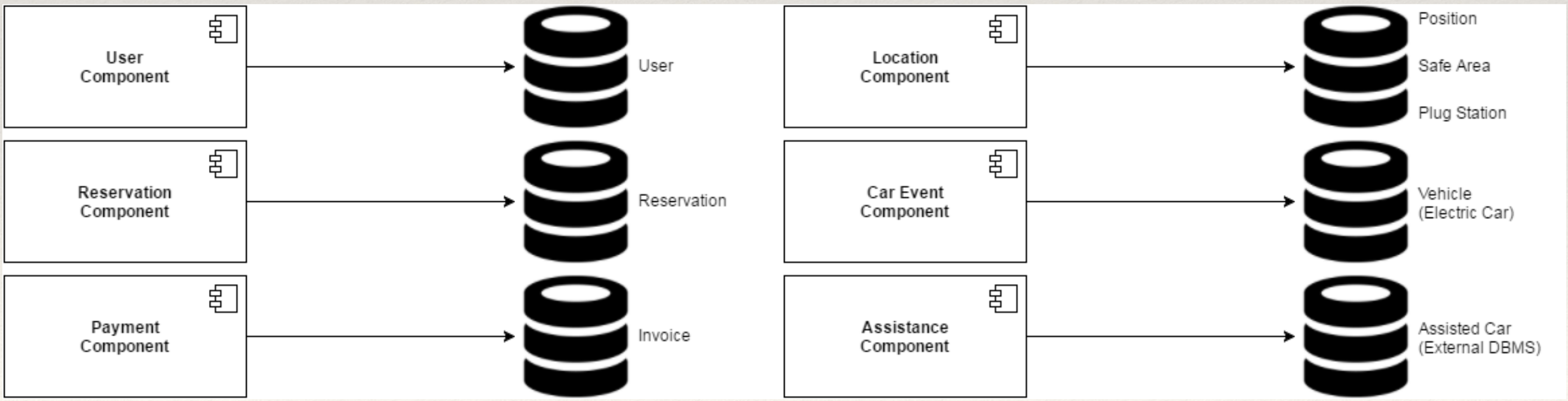
Some Problems…

- **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

Reservation Service

User Service

Location Service

Database

Assistance Service

Payment Service

Car Service

Assistance System

Payment System

Car On Board System

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

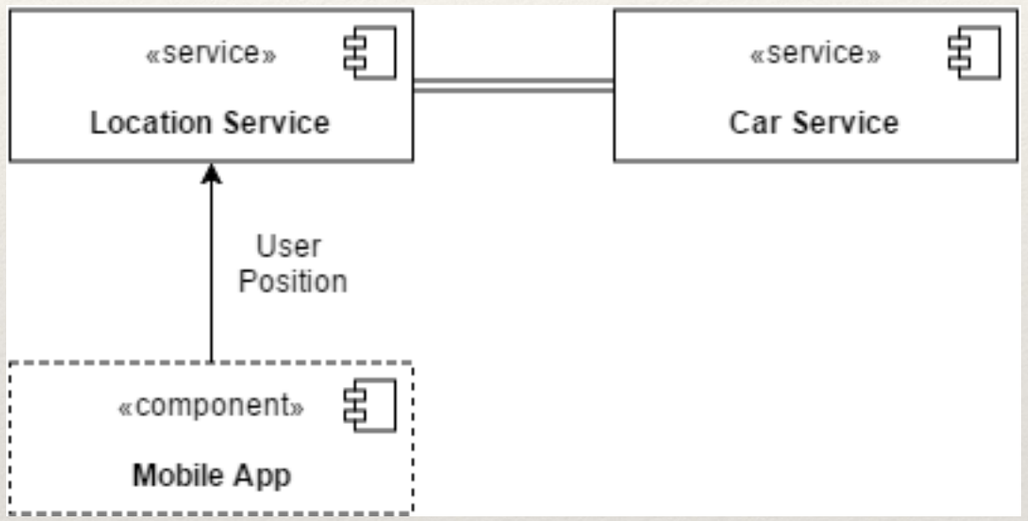
Integration Strategy

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

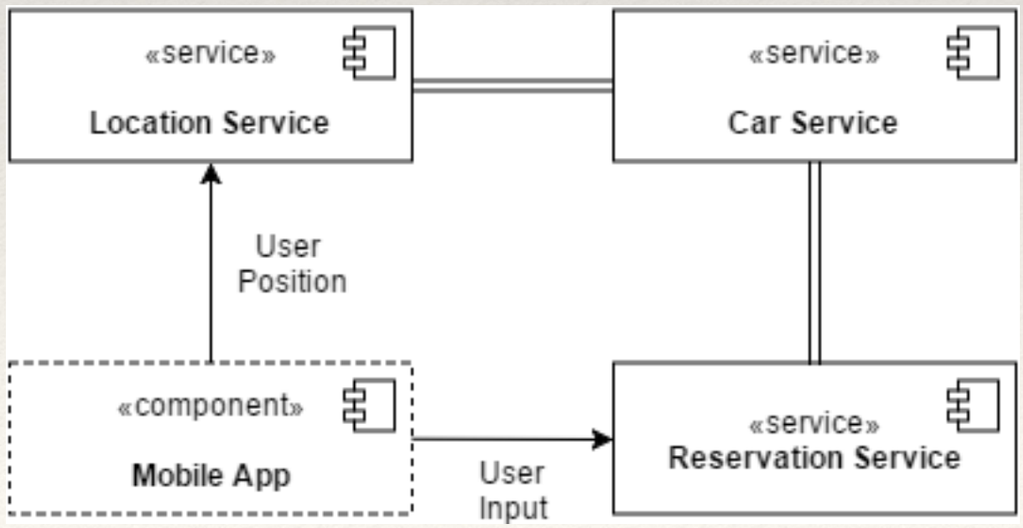
Integration of Services

LOCATION OF VEHICLES

LOCATION OF USER

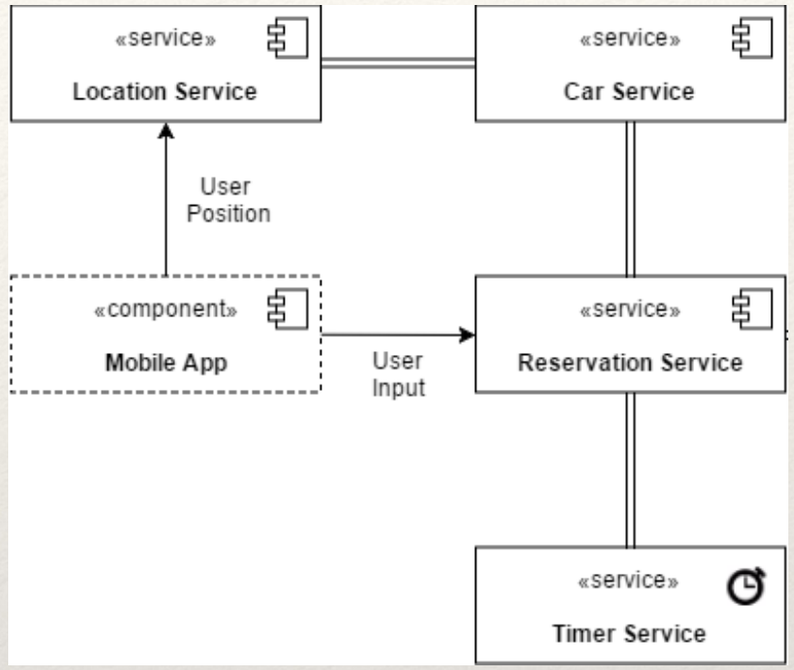


RESERVATION

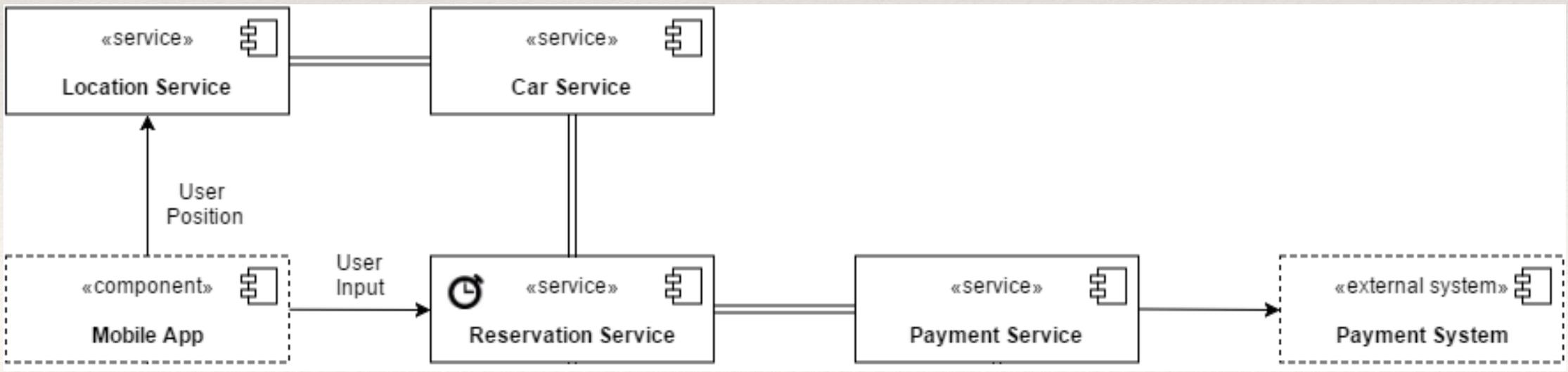


Integration of Services

TIMING

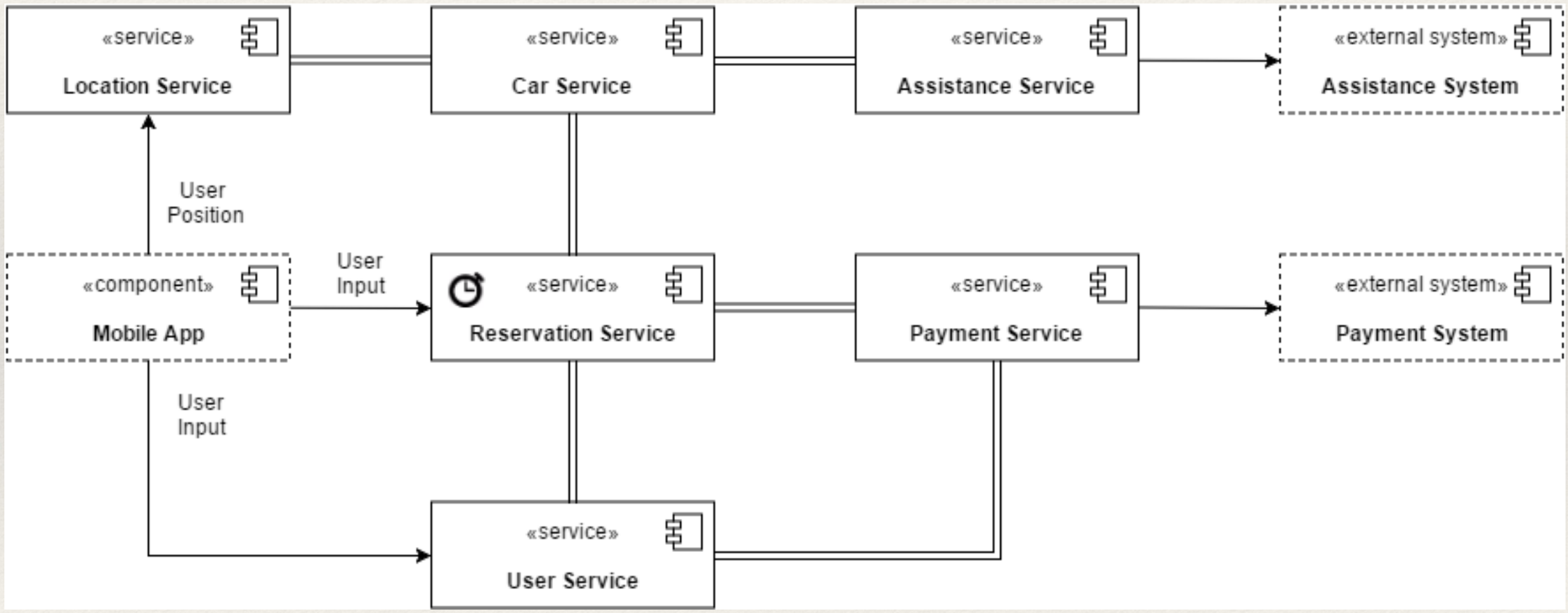


PAYMENT



Integration of Services

WHOLE SYSTEM



Tools Used

- Mockito
- Arquilian
- JUnit
- Manual Testing

Project Plan

- Beginning of the project → Early Design Approach
- Function Points and Drivers (COCOMO) values based on our real skill and experience
- $SLOC_{Avg}$=7728 lines of code
- Effort=31,47 PM
- Duration ≈ 16 Months with 2 developers
  - High value but reasonalbe since Early Design approach

Scheduling

	Research & PP		MILESTONE	RASD		MILESTONE	DD	
	SEP 2016			OCT 2016			NOV 2016	
Matteo	Customer Analysis	FPs Estimation		Domain, Goals	Use Cases		Component View	Runtime View
	Cost/Benefits Analysis	COCOMO II Estimation		Functional Requirements	Class Diagram		System Architecture	Requirements Traceability
Niccolo'	Market Analysis	Scheduling		Functional Requirements	Non Functional Requirements		System Architecture	UI Mockups
	Resource Allocation	Feasibility Study		World & Machine	UML Diagrams		Deployment View	High Level Components

	ITPD		MILESTONE	Implementation		MILESTONE	Integration	
	DEC 2016 - JAN 2017			JAN 2017 - SEP 2017			SEPT 2017 - DEC 2017	
Matteo	Driver & Stubs Required	Test Cases		Single Unit Coding	Code Inspection		Subsystems Integrations	Alpha Test
	Individual Test Description			Single Unit Testing			Deployment	
Niccolo'	Integration Strategy	Test Data & Equiment		Single Unit Coding	Code Inspection		Components Integration	Alpha Test
	Individual Test Description			Single Unit Testing			Deployment	

Poject Risks

- Changing Requirements
  - Why? Most recurrent and also imprevedible
  - Strategy: Traceability Information and information hiding in design
- Personnel Shortfall
  - Why? Just 2 developers 1 fall = +50% delay
  - Strategy: Positive Work Environment

Technical Risks

- Defecting Components of the Car System
  - Why? Delays in the Testing Stage
  - Strategy: Present partial developement

- Software too difficult to use for avarage costumers
  - Why? Important point for the Success or Failure of the product
  - Strategy: Revisit UI and UX

- Also others…

Questions