Software Engineering 2: PowerEnJoy

Francesco Fabiani
Jagadesh Manivannan
Niccolò Pozzolini

Politecnico di Milano

February 14, 2017

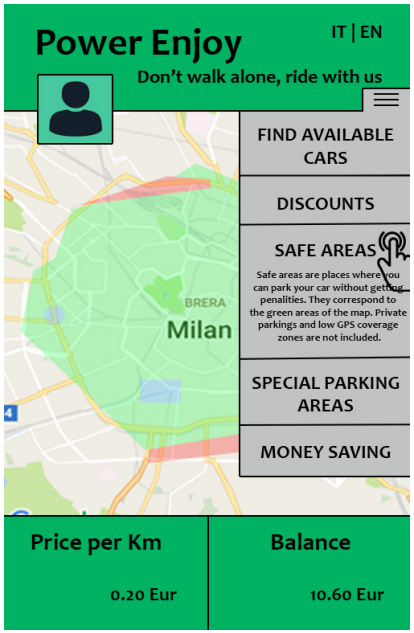
Contents

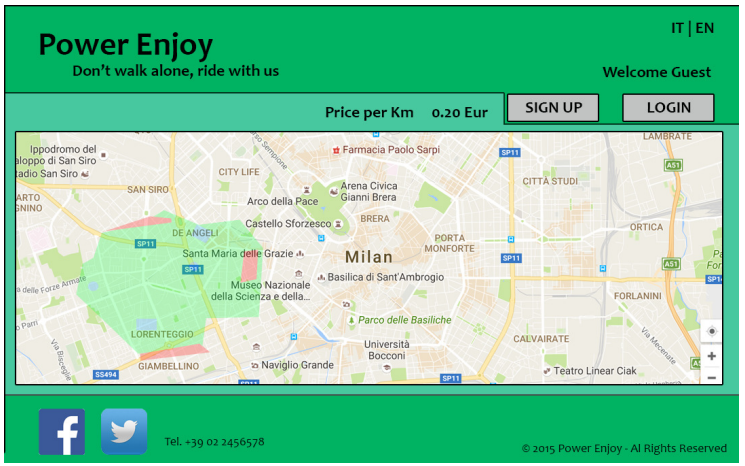
Requirement Analysis and Specification Document

Overview

High level functionality for the overall implementation. That includes:

- Defining and differentiating between requirements, domain properties and goals.
- World and the Machine diagram.
- Identifying various scenarios.
- UML-Use case diagram derived from scenarios identified.
- Dynamic modeling: through sequence diagram, class diagram, activity diagram.
- Alloy modeling for the world and the machine.





Goals and Domain Properties

Goals:

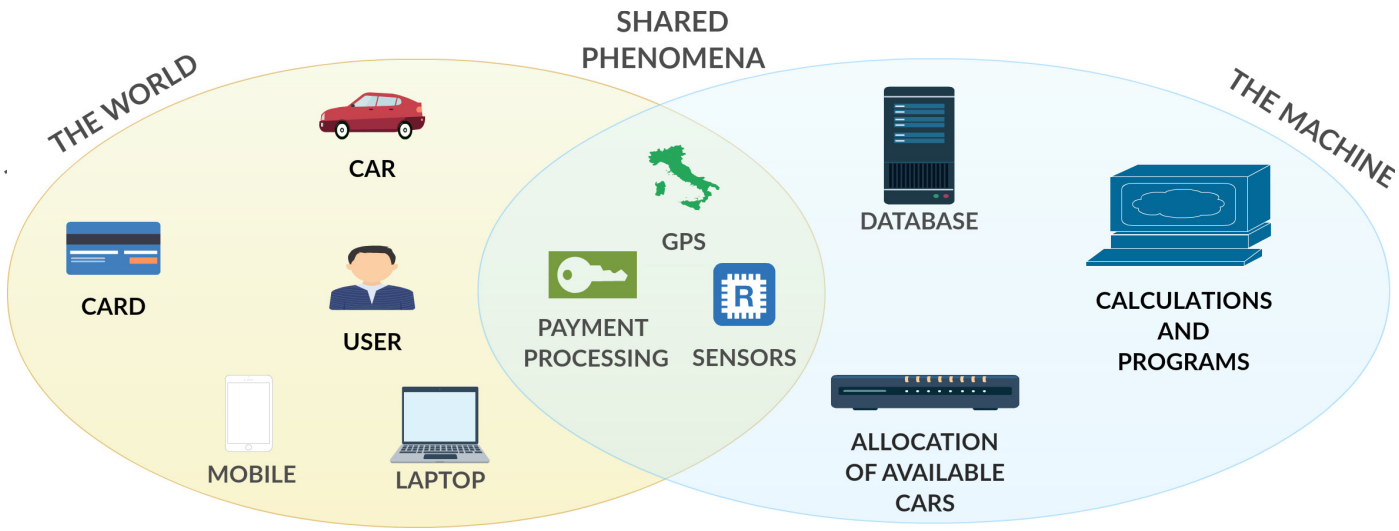
- [G1] Registration of a user to the system
- [G2] Finding the locations of the available cars
- [G3] Reservation of a car
- [G4] Expiration of reservation and penalization
- [G5] Entry of registered user into the car
- [G6] Start charging and notifying the registered user
- [G7] Stop charging the registered user and lock the car
- [G8] Safe areas for parking the reserved cars
- [G9] Detection of extra passengers and applying discount
- [G10] Detection of the battery status and applying discount
- [G11] Detection of special parking areas and applying discount
- [G12] Checking parking and battery constraints and penalization

Domain properties:

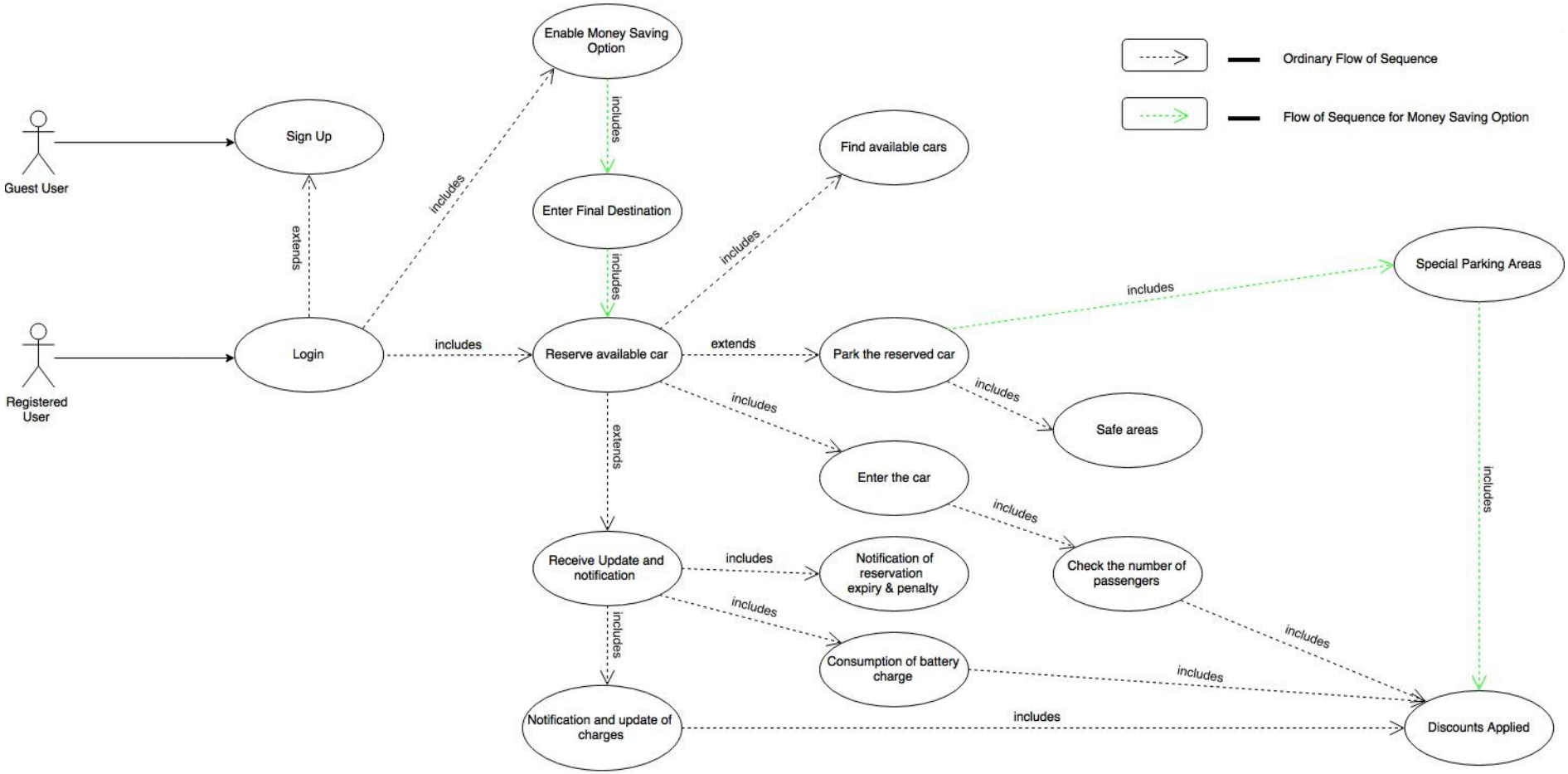
- User's data are always valid.
- Location reported by the GPS is always accurate.
- Every user can reserve just a car per time.
- The service is only available to validated users.
- The service has employees to move the cars if their distribution in the map is unbalanced.
- The license office provides the online service to let the system instantly check the user's documents during the registration.
- If the user's driver license expires, he gets unvalidated until he renew it.
- A call center is provided by the service to handle difficult situations.

Requirements

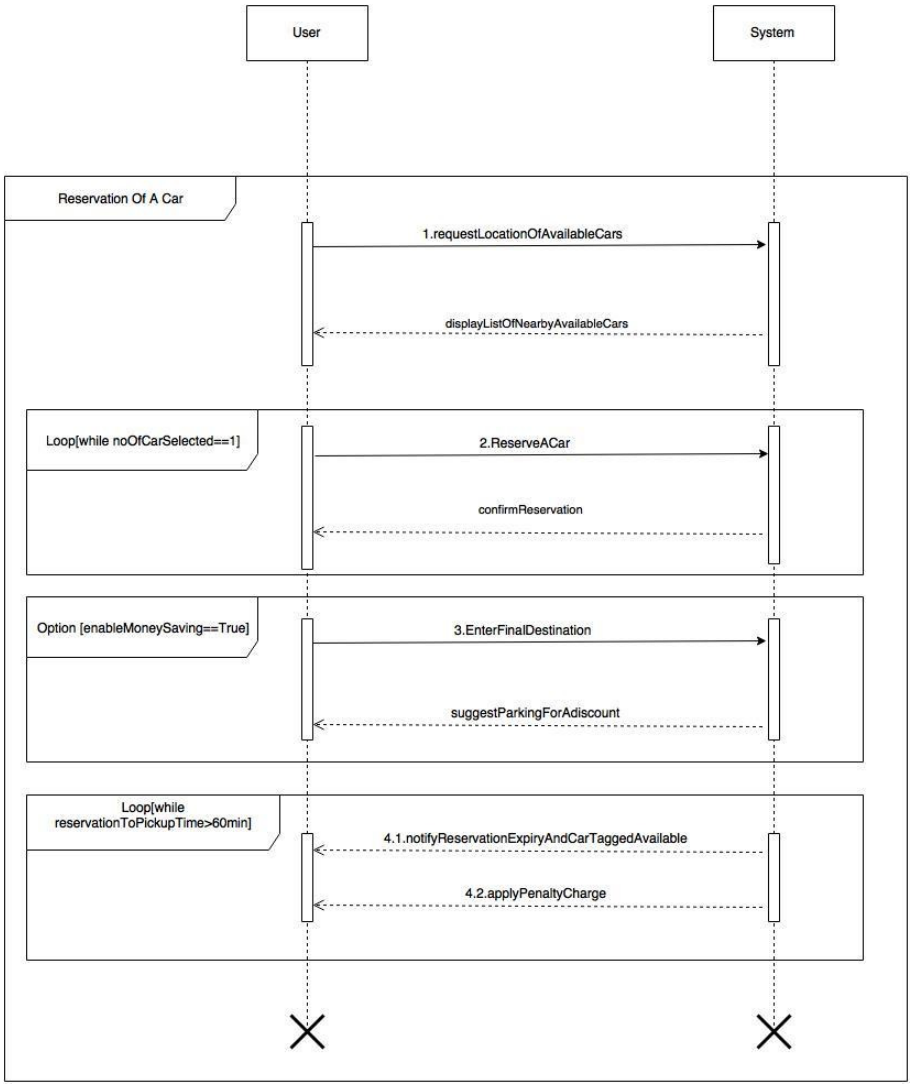
- The system needs to provide mandatory sign up and payment options or the guest users who wants to register to use the car sharing service.
- Once the payment is successful and the guest user is registered, the registered user receives a password that can be used to access (login into) the system.
- The system needs to provide the exact location of the cars that are available within a certain distance either from the current location of the registered users or from a specified address given(entered) by the registered users.
- The system provides provision such that the registered users must be able to reserve only a single car among the available cars in a certain geographical region for up to one hour before they pick it up.
- The system checks if a reserved car is picked-up within one hour. If not, the system tags the car as available again and the reservation expires.
- The system penalizes the registered user who made the reservation and did not pick the reserved car within an hour, by making him to pay a fee of 1EUR.

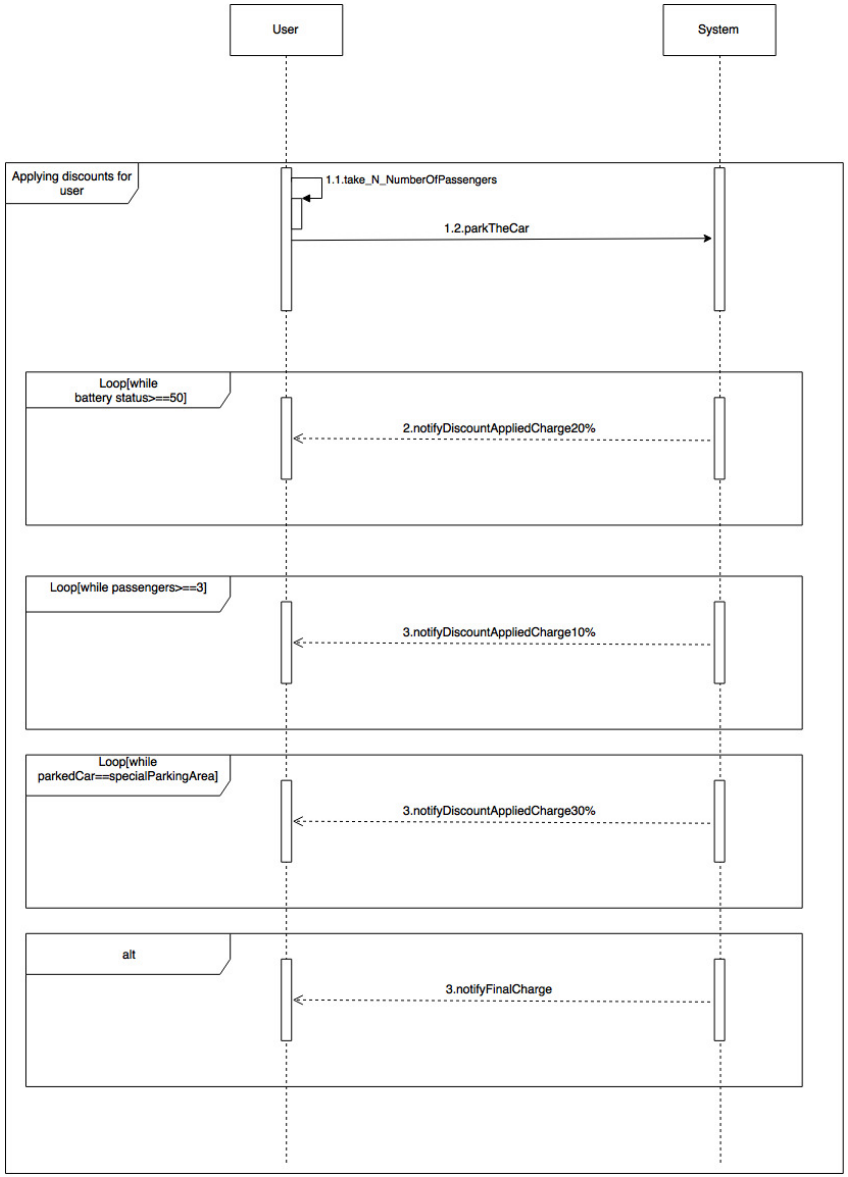


UML Model - Use Case Diagram

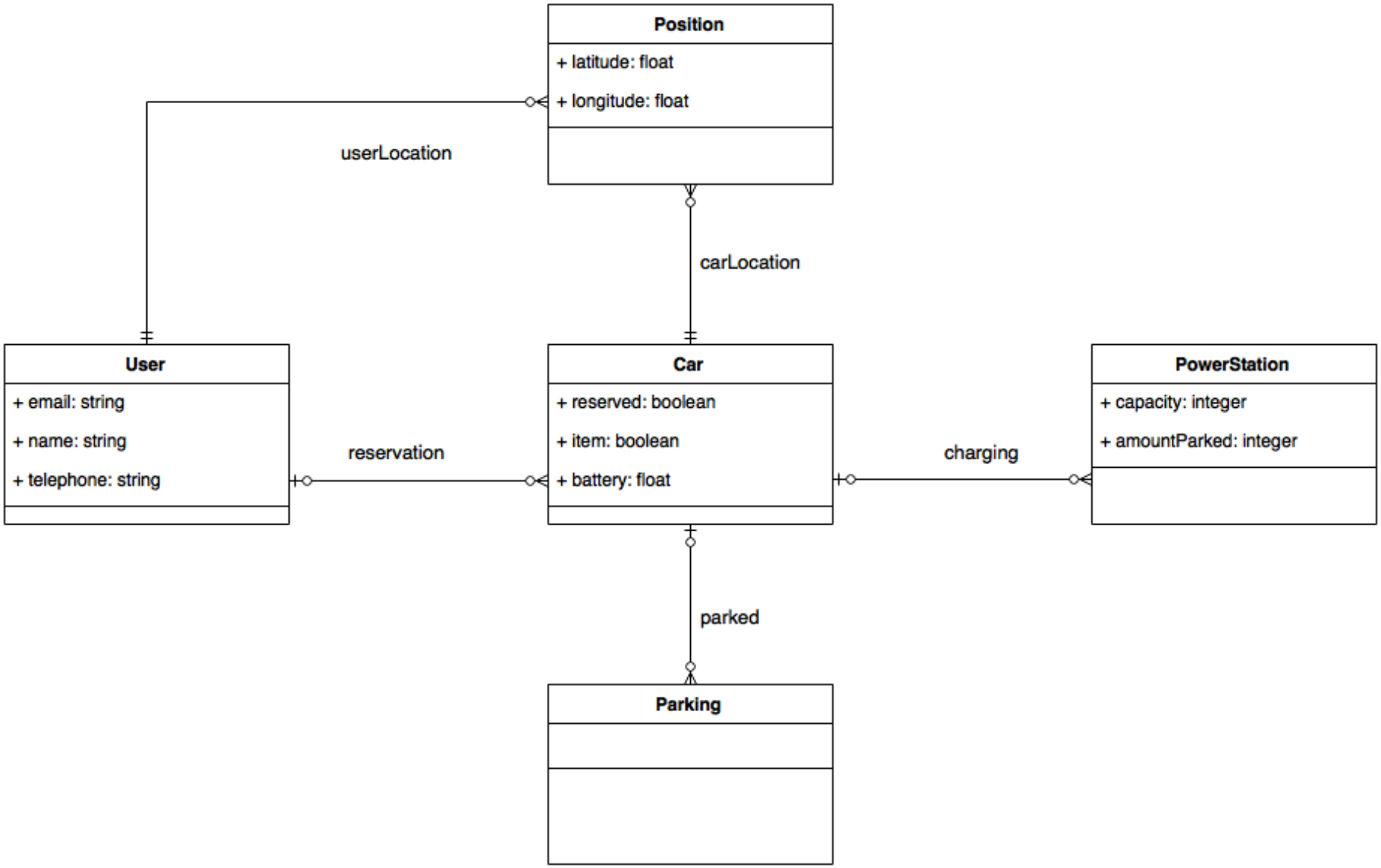


UML Model - Sequence Diagrams

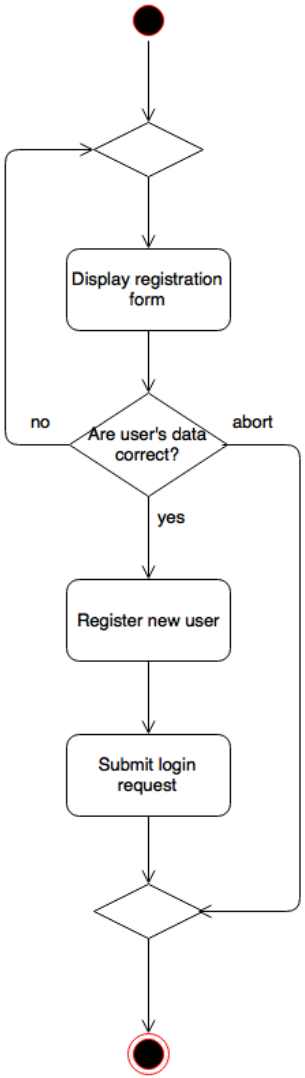


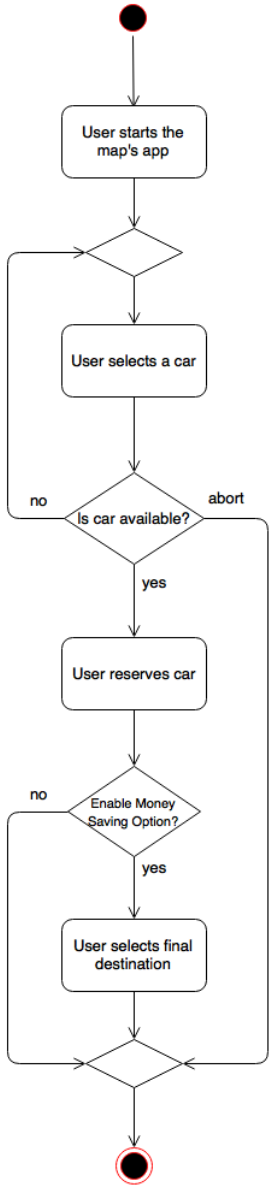


UML Model - Class Diagram

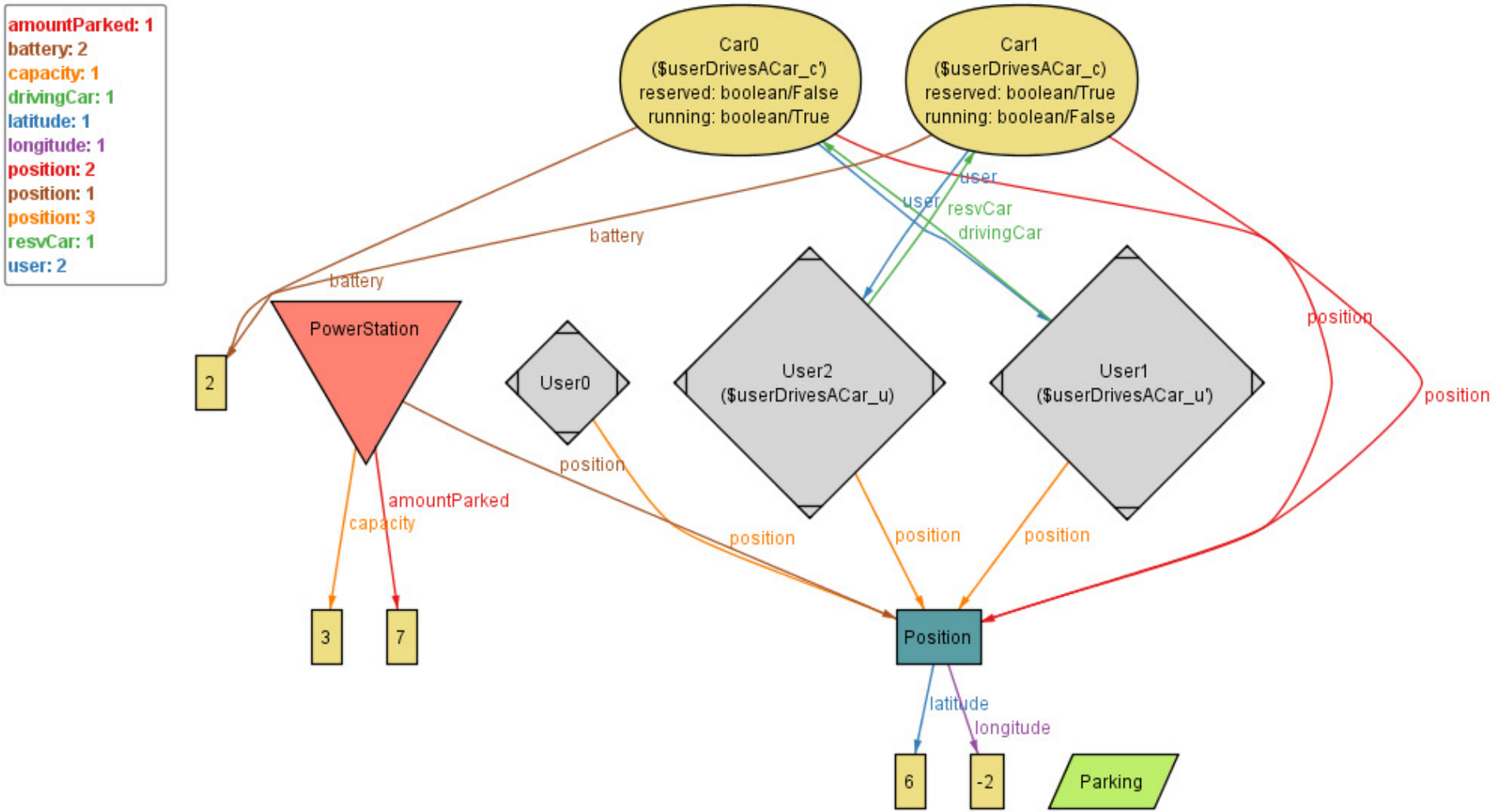


UML Model - Activity Diagrams



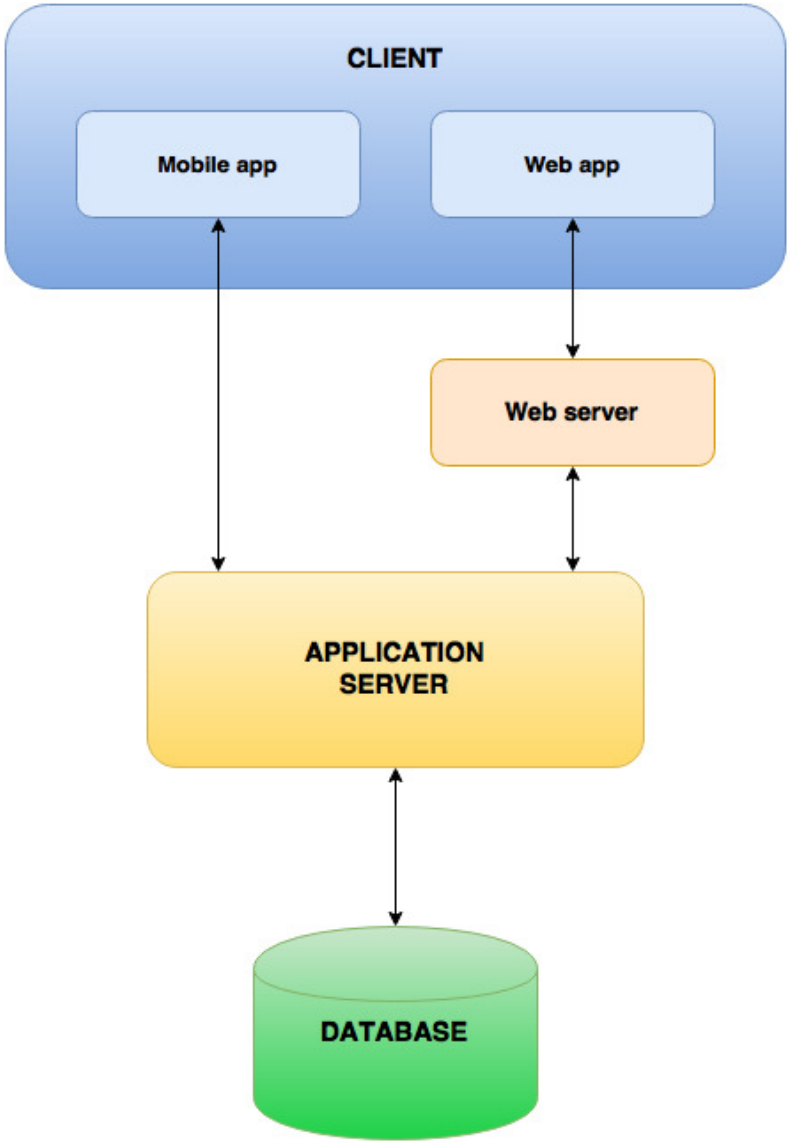


Alloy Model and World Generated

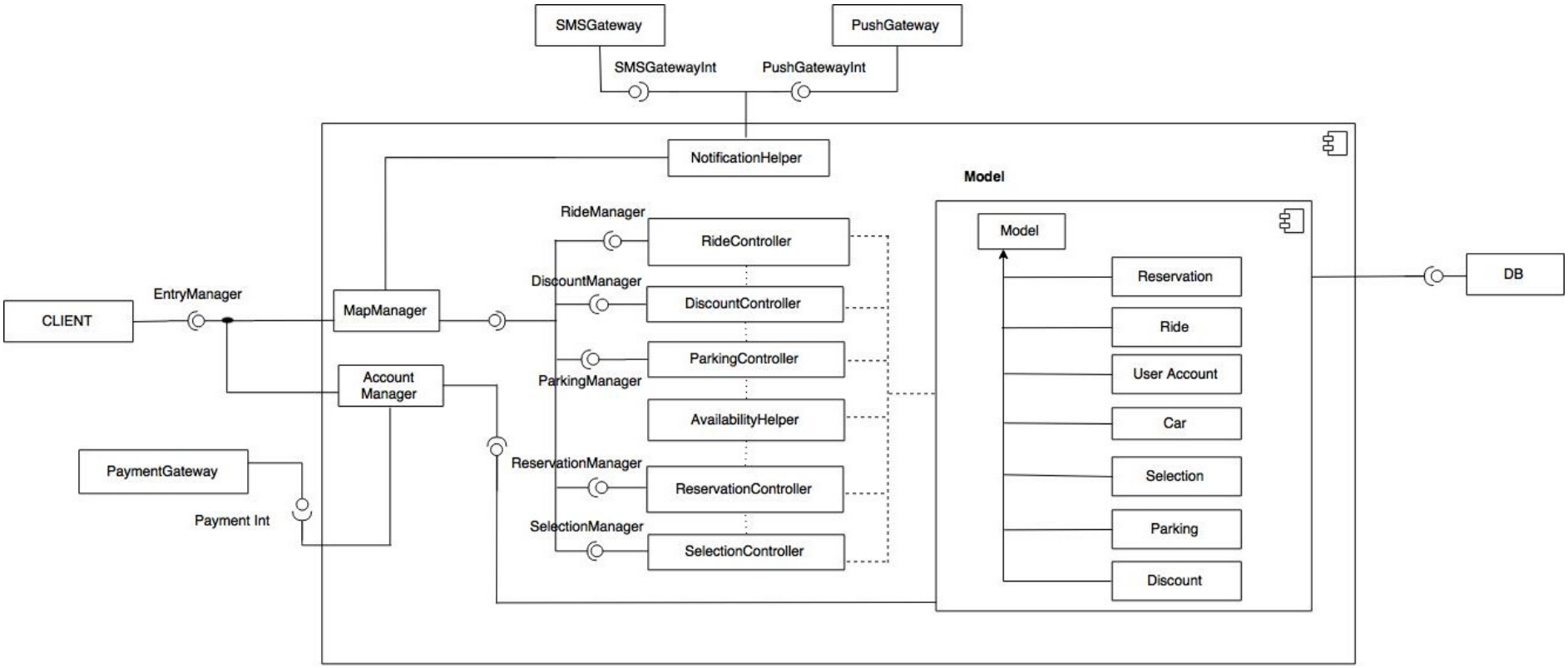


Design Document

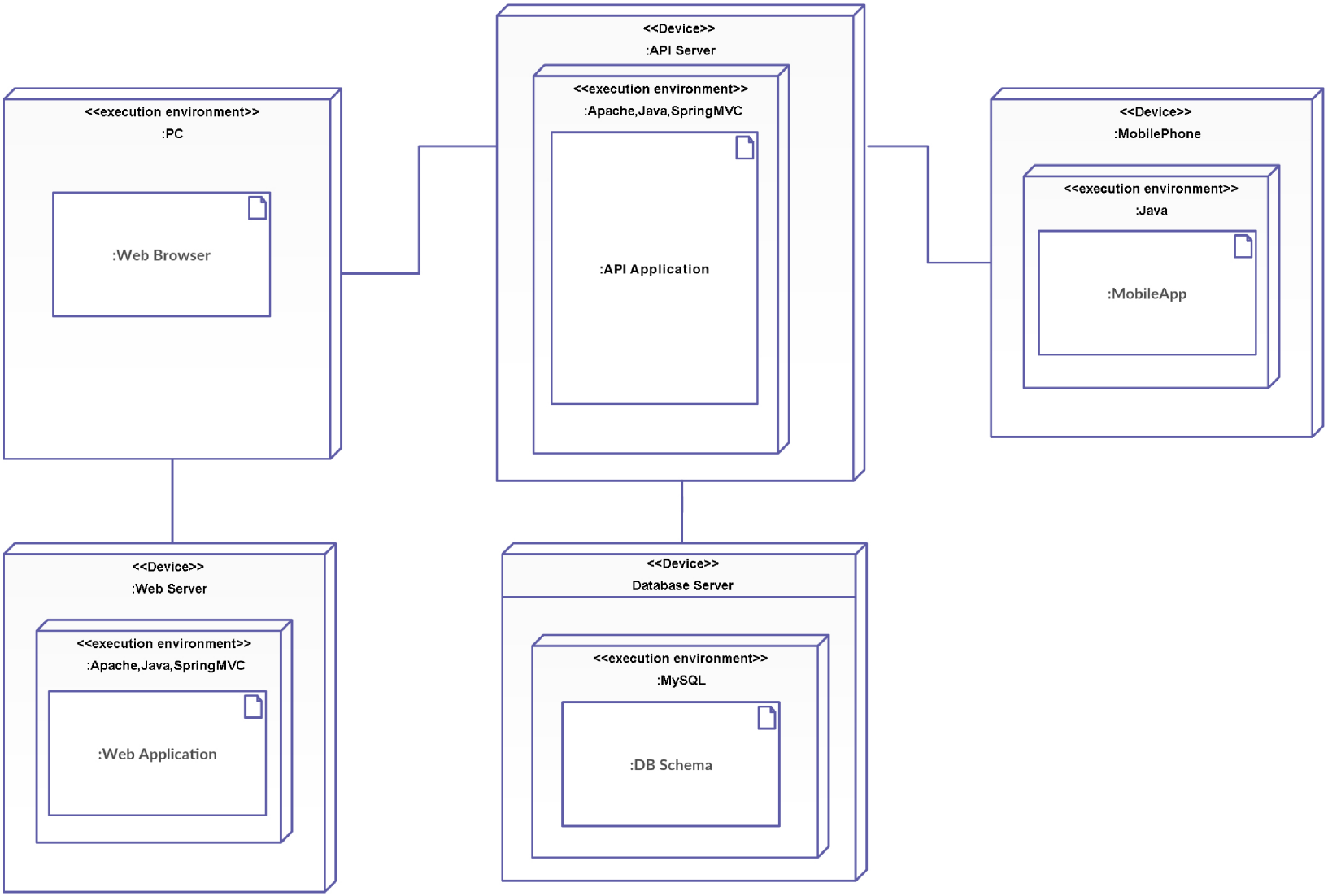
Three-tier Architecture



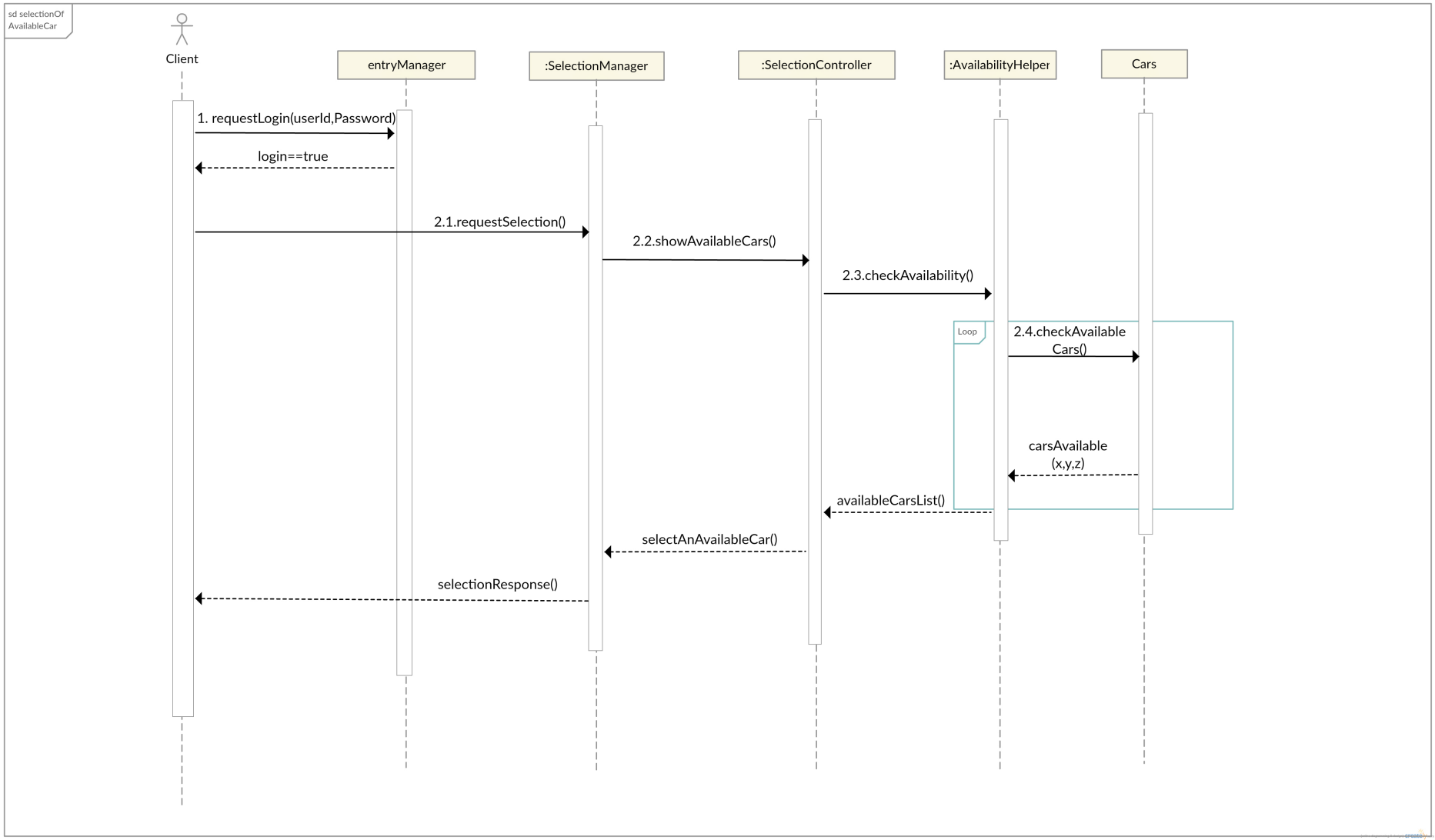
Component View



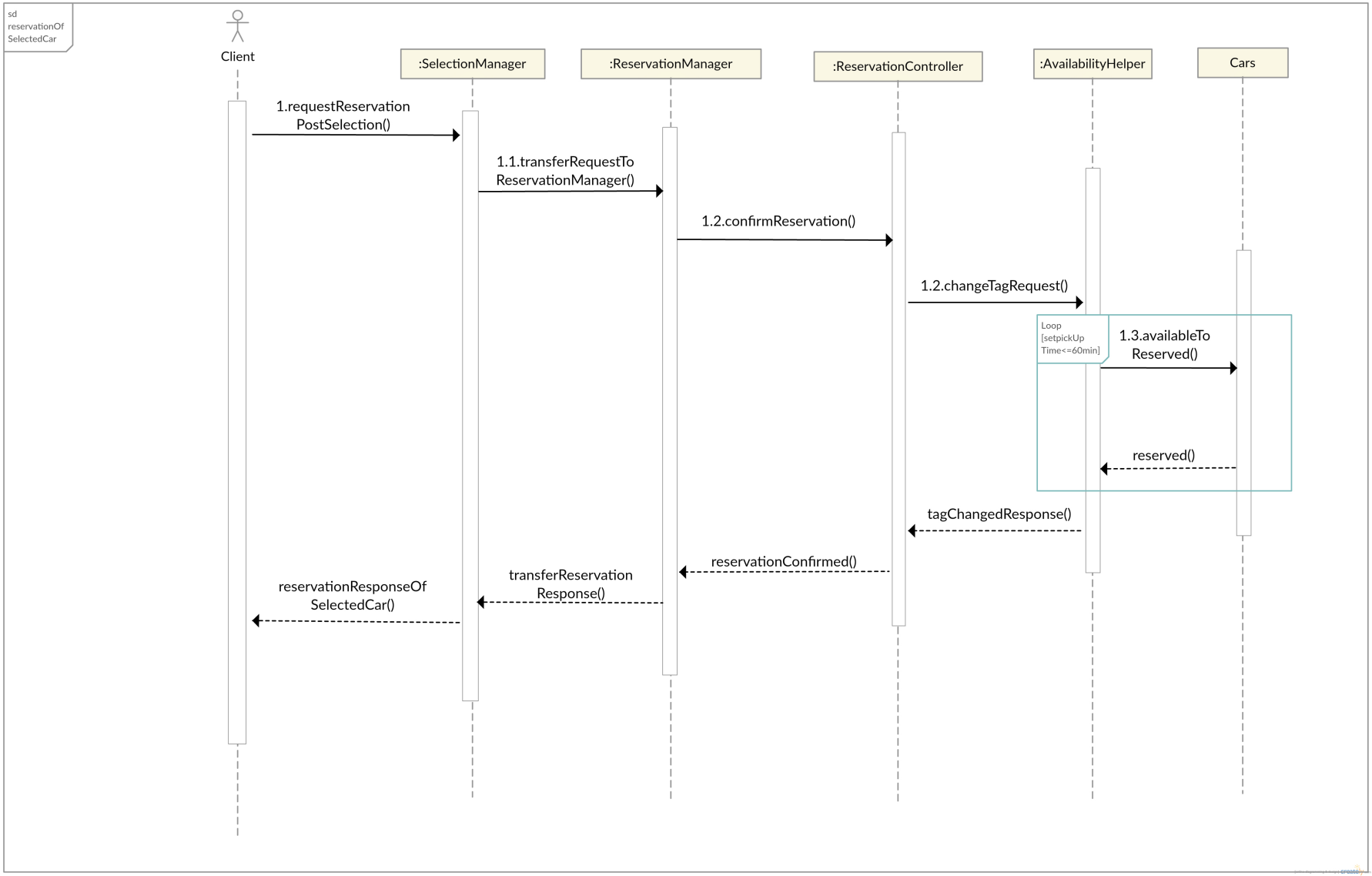
Deployment View



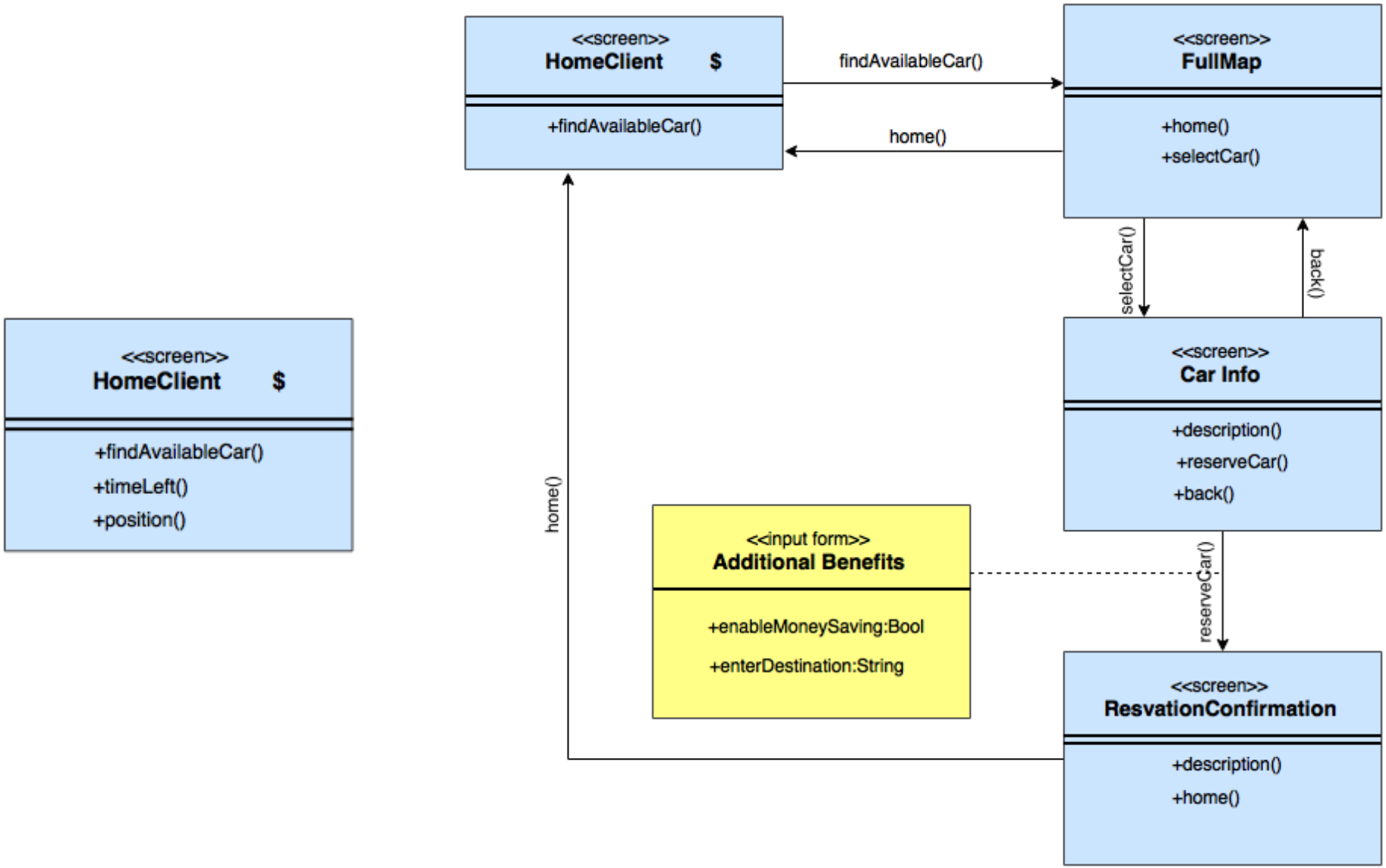
Selection of Available Car



Reservation of Selected Car



UX Diagrams


<<screen>>
HomeClient $
+findAvailableCar()
+timeLeft()
+position()
<<screen>>
HomeClient $
+findAvailableCar()
findAvailableCar()
home()
<<screen>>
FullMap
+home()
+selectCar()
selectCar()
back()
<<screen>>
Car Info
+description()
+reserveCar()
+back()
home()
<<input form>>
Additional Benefits
+enableMoneySaving:Bool
+enterDestination:String
reserveCar()
<<screen>>
ResvationConfirmation
+description()
+home()

Integration Test Plan Document

Entry Criteria

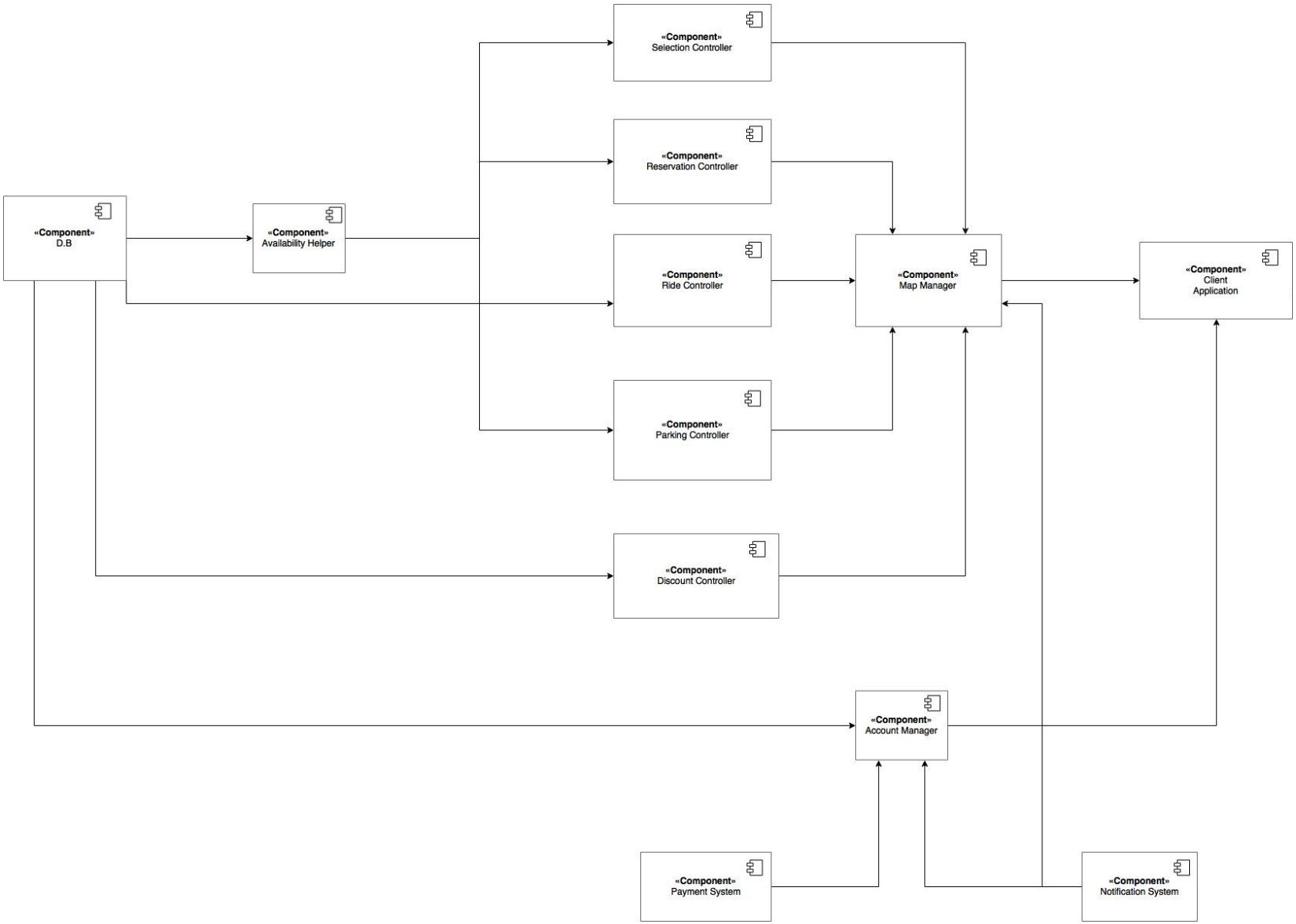
- RASD and DD documents are completely composed
- Development goes forward along with their unit testing
- Every component has at least 90% of its functionalities completed
- The integration process starts at the achievement of the following percentages of development:
  - 100% of the Database and Availability Helper components
  - at least 80% of the Controller components
  - at least 50% for the client applications

Integration testing strategy

Bottom-up approach

- Components integrated from the most independent to the less one
- Avoiding useless stubs
- Creation of an integrated subsystem
- Allowing developers to perform integration testing earlier in the development process
- Parallelism and efficiency are maximized

Components


«Component»
Selection Controller
«Component»
Reservation Controller
«Component»
D.B
«Component»
Availability Helper
«Component»
Ride Controller
«Component»
Map Manager
«Component»
Client
Application
«Component»
Parking Controller
«Component»
Discount Controller
«Component»
Account Manager
«Component»
Payment System
«Component»
Notification System

Integration Test Cases

Test case identifier	I5T2
Test item(s)	Reservation Controller → Map Manager
Input specification	Post a ReservationController request
Output specification	Check if the request is correctly routed
Environmental needs	I1, I2, I3 succeeded

placeReservation(car) returns response	
Input	*Effect*
A valid car	ReservationController will invoke AvailabilityHelper.changeTagRequest() and get the response R. R is then returned.
A null or invalid car	A NullCarException or InvalidCarException is generated.

Program Stubs

- Reduced number of drivers is required to perform necessary method invocations
  - *AvailabilityHelper driver*: invokes the methods exposed by the component AvailabilityHelper in order to test its interaction with the database management system
  - *Controllers driver*: invokes the methods exposed by the Controllers (acting like MapManager) in order to test its interaction with each other
- Only one stub acting like the client to fulfill the request/response protocol during client/system interaction

Project Plan

Function Points

Function	Type	Compl.	FPs
Reservation	ILF	Medium	10
Ride	ILF	Medium	10
User	ILF	Complex	15
Payment	ILF	Simple	7
Car	ILF	Simple	7
Parking	ILF	Simple	7
Discount	ILF	Simple	7
Get available cars (zone)	EQ	Simple	3
Get payment history	EQ	Simple	3
Get active reservation	EQ	Simple	3
Get available discounts	EQ	Simple	3
Notify	EO	Complex	7

Function	Type	Compl.	FPs
Register	EI	Middle	4
Login	EI	Simple	3
Logout	EI	Simple	3
Update licence	EI	Simple	3
Pay	EI	Middle	4
Reserve car	EI	Complex	6
Abort reservation	EI	Complex	6
Open car	EI	Complex	6
End ride	EI	Complex	6
Plug	EI	Middle	4
Ban user	EI	Simple	3
Add/remove car	EI	Simple	3

Source Lines of Code

Function type	Value
Internal Logic Files (ILF)	63
External Logic Files (ELF)	0
External Inputs (EI)	51
External Inquiries (EQ)	12
External Outputs (EO)	7
Total	133

- Lower bound: $SLOC = 133 \times 51 = 6783$
- Upper bound: $SLOC = 133 \times 73 = 9709$

COCOMO

Cost Driver	Factor	Value
Required software reliability (RELY)	High	1.10
Database size (DATA)	High	1.14
Product complexity (CPLX)	Very high	1.34
Required reusability (RUSE)	Nominal	1.00
Documentation match to life-cycle needs (DOCU)	Nominal	1.00
Execution time constraint (TIME)	Very high	1.29
Main storage constraint (STOR)	High	1.11
Platform volatility (PVOL)	Nominal	1.00
Analyst capability (ACAP)	High	0.85
Programmer capability (PCAP)	High	0.88
Application experience (APEX)	High	0.88
Platform Experience (PLEX)	High	0.91
Language and Tool Experience (LTEX)	High	0.91
Personnel continuity (PCON)	Very low	1.12
Usage of Software Tools (TOOL)	High	0.90
Multisite development (SITE)	Very high	0.86
Required development schedule (SCED)	High	1.00
Total		1.13694

Effort and Duration

$$\textbf{Effort} = 2.91 \times EAF \times KSLOC^{E}$$

$EAF = 1.13694$ (product of all cost drivers)
$E = B + 0.01 \times \sum_i SF[i] = B + 0.01 \times 11.84 = 0.91 + 0.1184 = 1.0284$

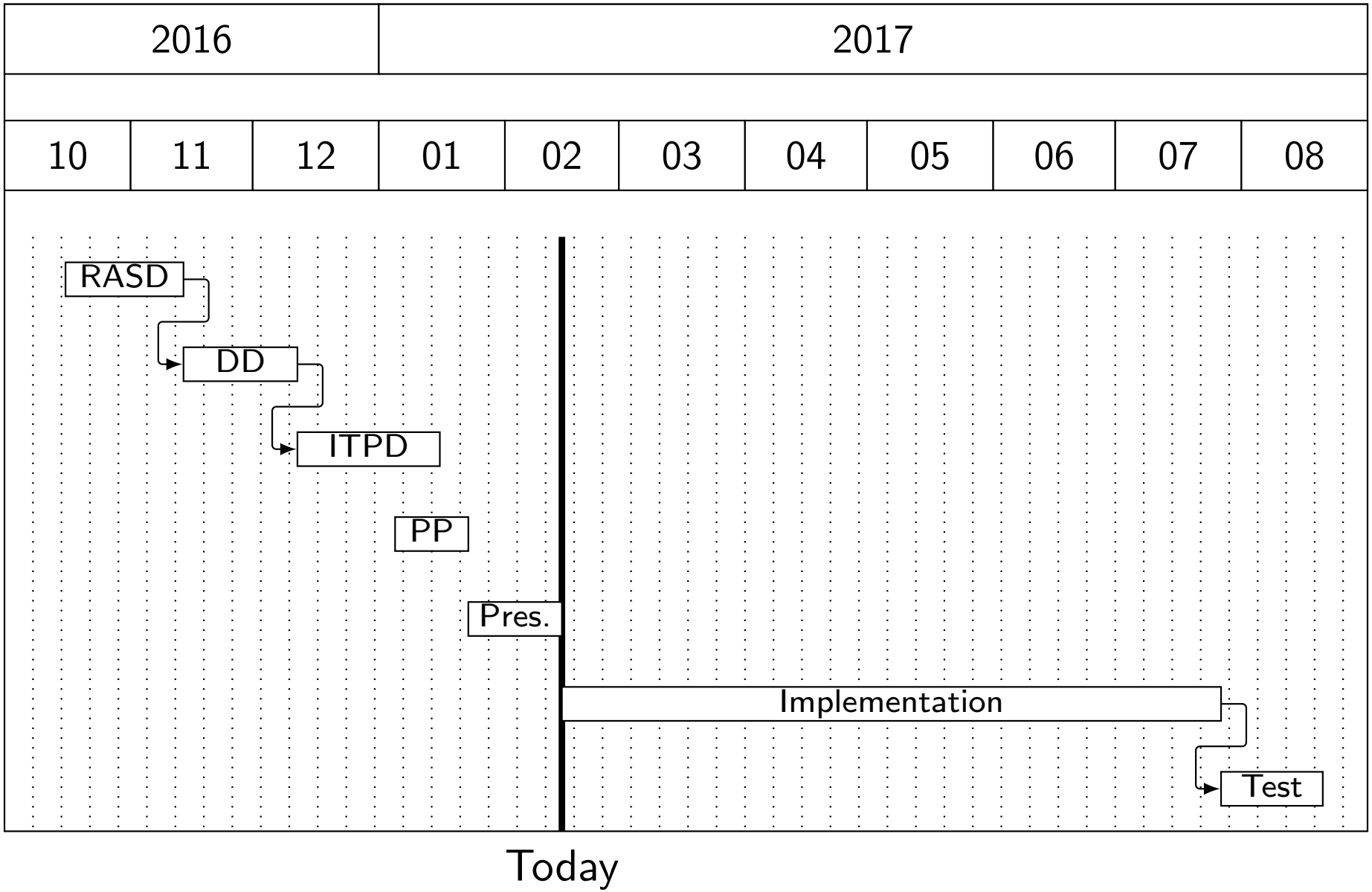
- LB: $Effort = 2.94 \times 1.13694 \times 6.783^{1.0284} = 23.94$ PM ≈ 24 PM
- UP: $Effort = 2.94 \times 1.13694 \times 9.709^{1.0284} = 34.61$ PM ≈ 35 PM

$$\textbf{Duration} = 3.67 \times Effort^{F}$$

$F = 0.28 + 0.2 \times (E - B) = 0.28 + 0.2 \times 0.1184 = 0.31368$

- LB: $Duration = 3.67 \times 23.94^{0.31368} = 9.94$ months
- UB: $Duration = 3.67 \times 34.61^{0.31368} = 11.15$ months

Gantt Chart


2016
2017
10
11
12
01
02
03
04
05
06
07
08
RASD
DD
ITPD
PP
Pres.
Implementation
Test
Today