My Taxi Service

Massimo Beccari - Davide Citterio – Lorenzo Cunial

Requirement Analysis and Specification Document

Goals

- The user has to be able to sign up and log in with the mobile application and the web page
- The user has to be able to require a taxi specifying the starting address
- The user must be able to use the service of taxi sharing
- The taxi driver has to be able to answer to the calls generated by the system after the users request
- For each user asking for a request or a reservation of a taxi is guaranteed the punctuality
- The system has to allow the administrator to manage driver
- The administrator has to be able to change the price of the ride, deactivate some services and know the position of the taxi

Domain Properties

- Taxi drivers have to take up no more than 4 people simultaneously
- Taxi work within the jurisdiction of Municipality of LargeCity
- Taxi locations are known by GPS signal
- The connection between the system and taxi drivers devices is made by UMTS technology: all the LargeCity jurisdiction is covered by UMTS signal so it guarantee a very fast connection
- Each sharing-reservation (composed of many users) involves at most 4 people (in order to ease our system and the driver)

Assumptions

- Each zone is reachable in less than 10 minutes by at least one taxi
- In the previous system users had only one kind of service: the request. This was made by telephone call
  (so we haven’t got integration with a pre-existent informative system)
- Mobile devices of users don't guarantee a precise geolocation service (accordingly user insert manually his position)
- Sometimes user lost connection
  (accordingly we send a confirmation by SMS Message)

Actors Identifying

- Guest
- User
- Driver
- Administrator

Functional Requirements (1)

- User
  - Log in
  - Recover password
  - Make a taxi request
  - Make a taxi reservation
  - Receive a confirmation sms
  - Modify a taxi reservation
  - Delete a taxi reservation
  - See all reservations
  - Filter reservations by date and price

Functional Requirements (2)

- Driver
  - Log in
  - Set his status
  - Accept or deny a request;
  - See the route toward a destination
  - See the fee for a ride
  - See the route toward a zone
  - Being notified when a user wants to join a shared ride

Functional Requirements (3)

- Administrator
  - See the current position of every taxi
  - See the current status of every on-line taxi
  - See the information about the rides
  - Add and remove taxi drivers from the system
  - Set the hourly fee
  - Set the fixed daily fee
  - Set the fixed nightly fee
  - Delete a driver
  - Enable/disable additional services

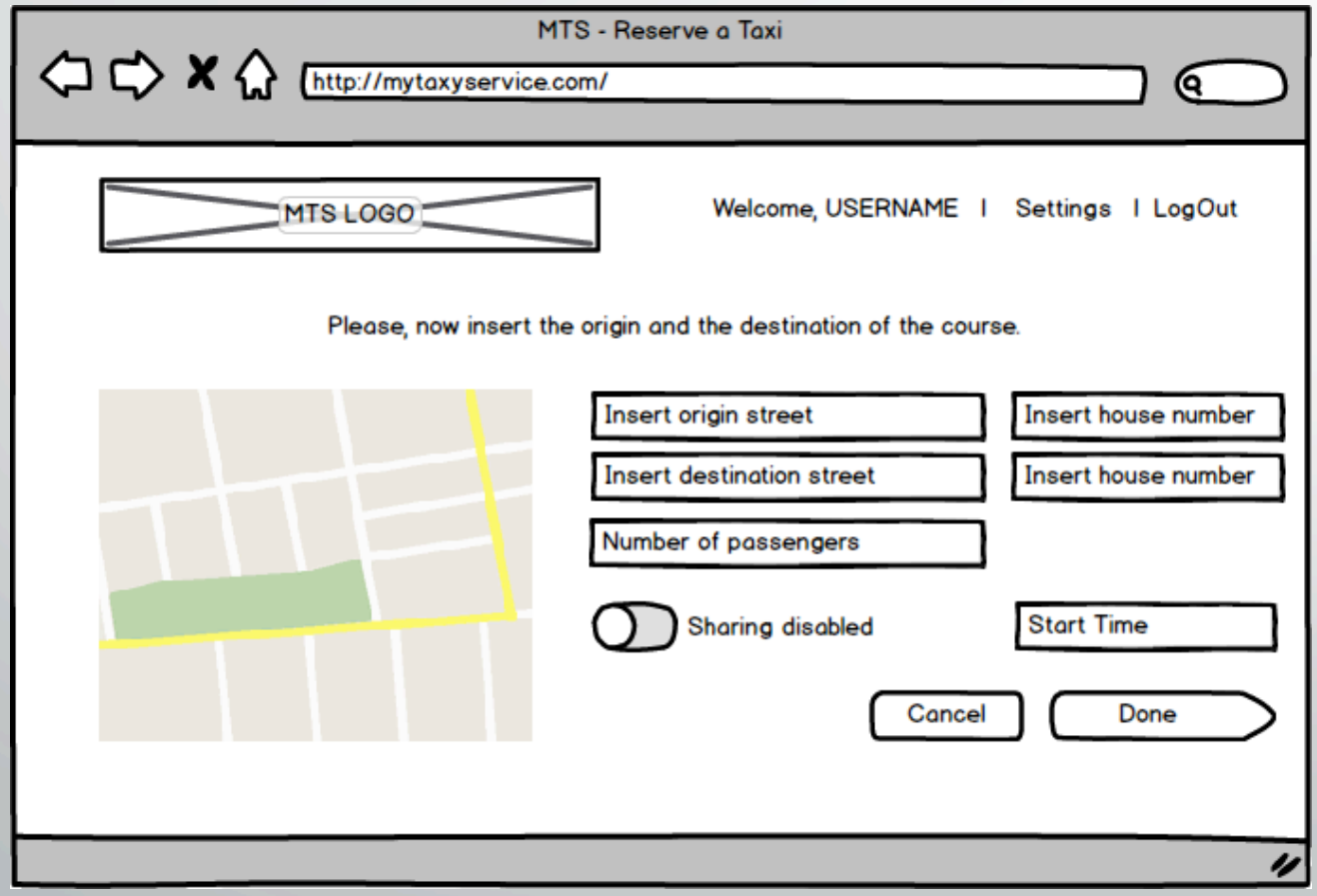
Software system attributes

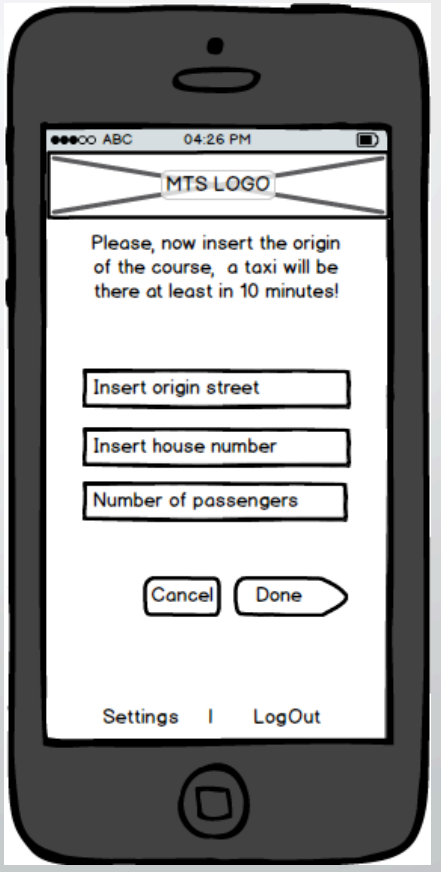
- Reachable anytime (possible 24/24h - 7/7 days)
- High level of security and privacy on user's personal data
- High scalability
- The server of the system has to be allocate outside the municipality buildings due to the not enough space and resources to managed the infrastructure by the stakeholder

Software system attributes – Security Focus

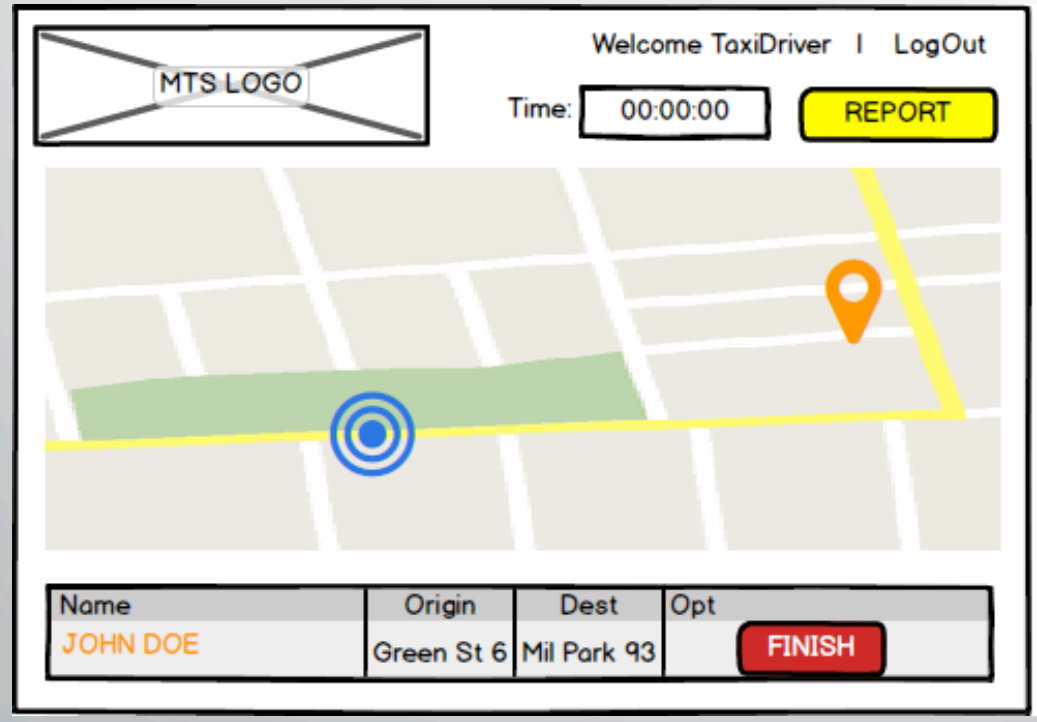
- User-side
  - Login/Logout with email address and password
  - Password has to be encrypted
- Driver-side
  - Login/Logout with id and password given by the administrator random generated by the system
- Administrator side
  - Login/logout with password given by developers -> cannot modify it
- Application-side
  - Prevent any kind of SQL Injection
  - Possibly use HTTPS Protocol to improve security (at least on user side)

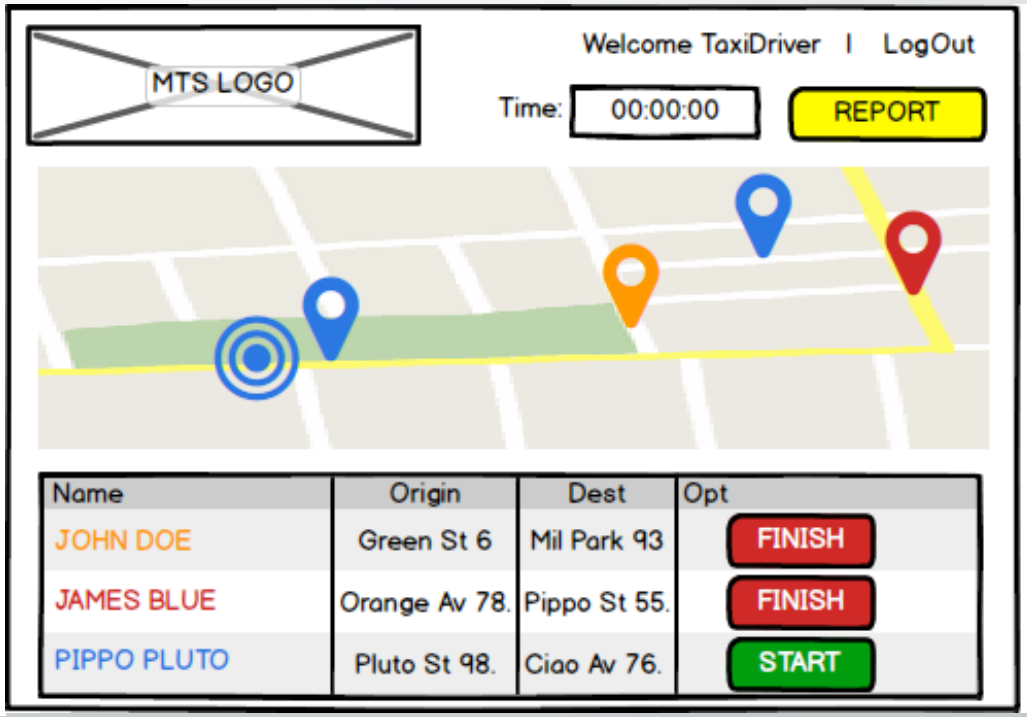
Mockup – User interfaces



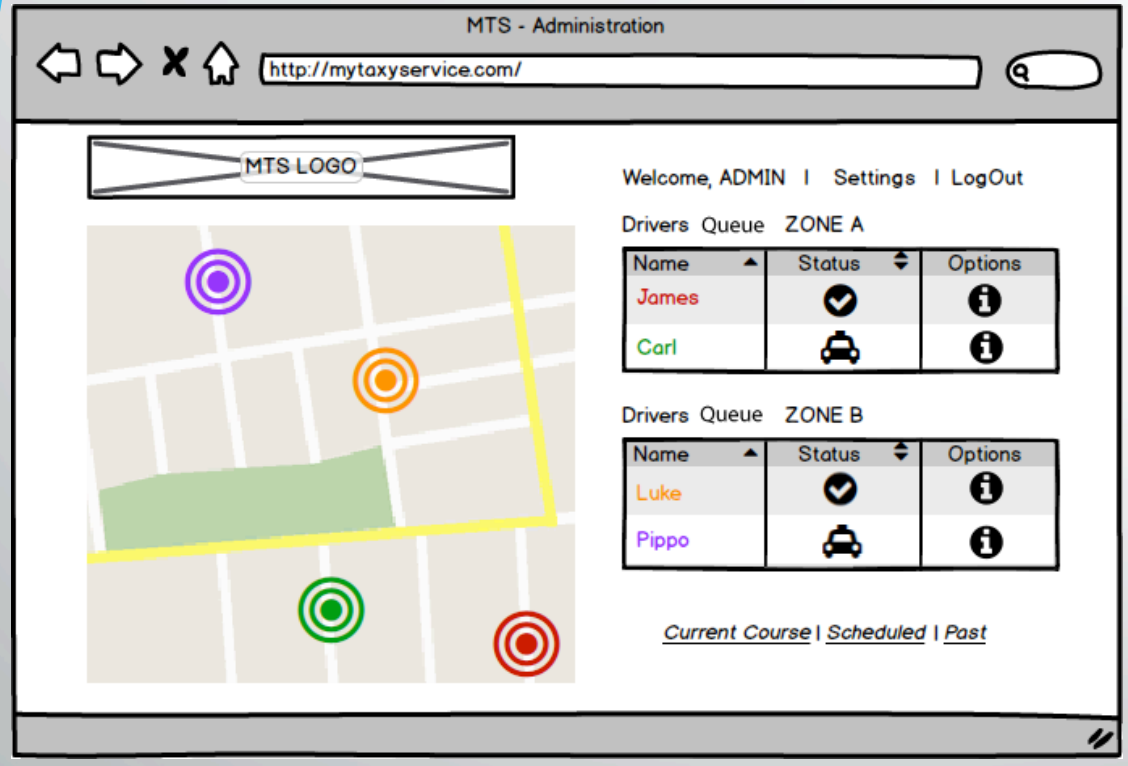


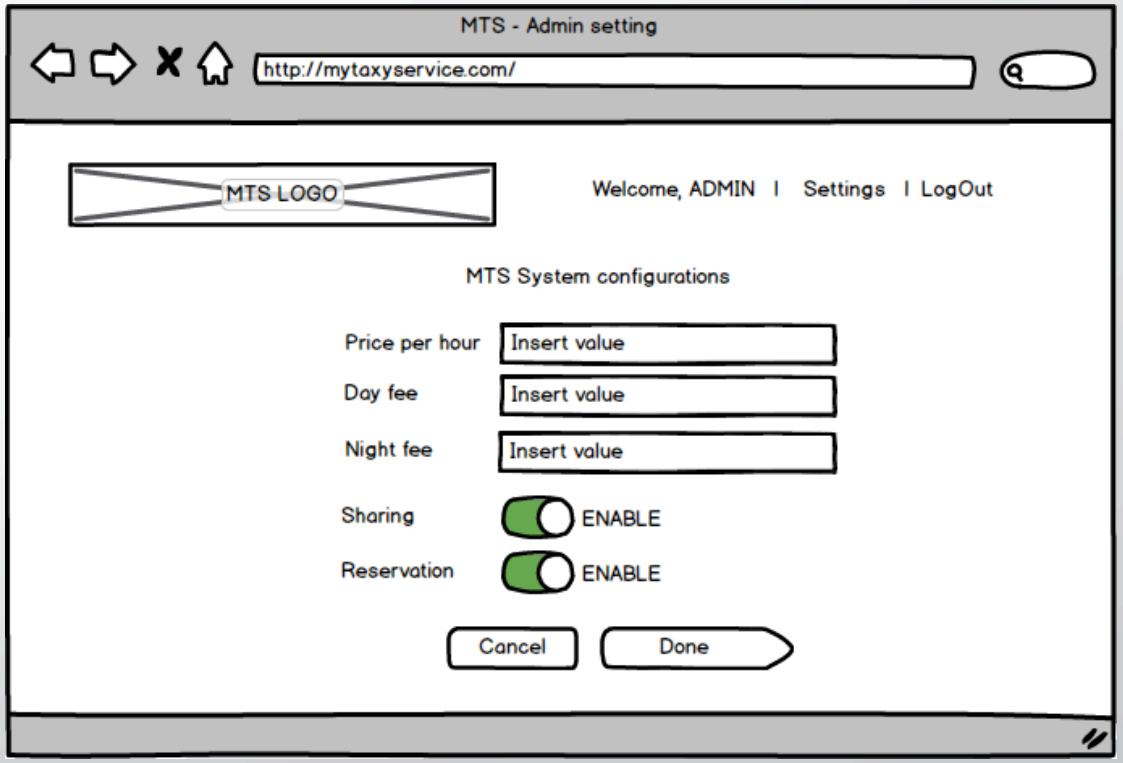
Mockup – Driver interfaces





Mockup – Administrator interfaces

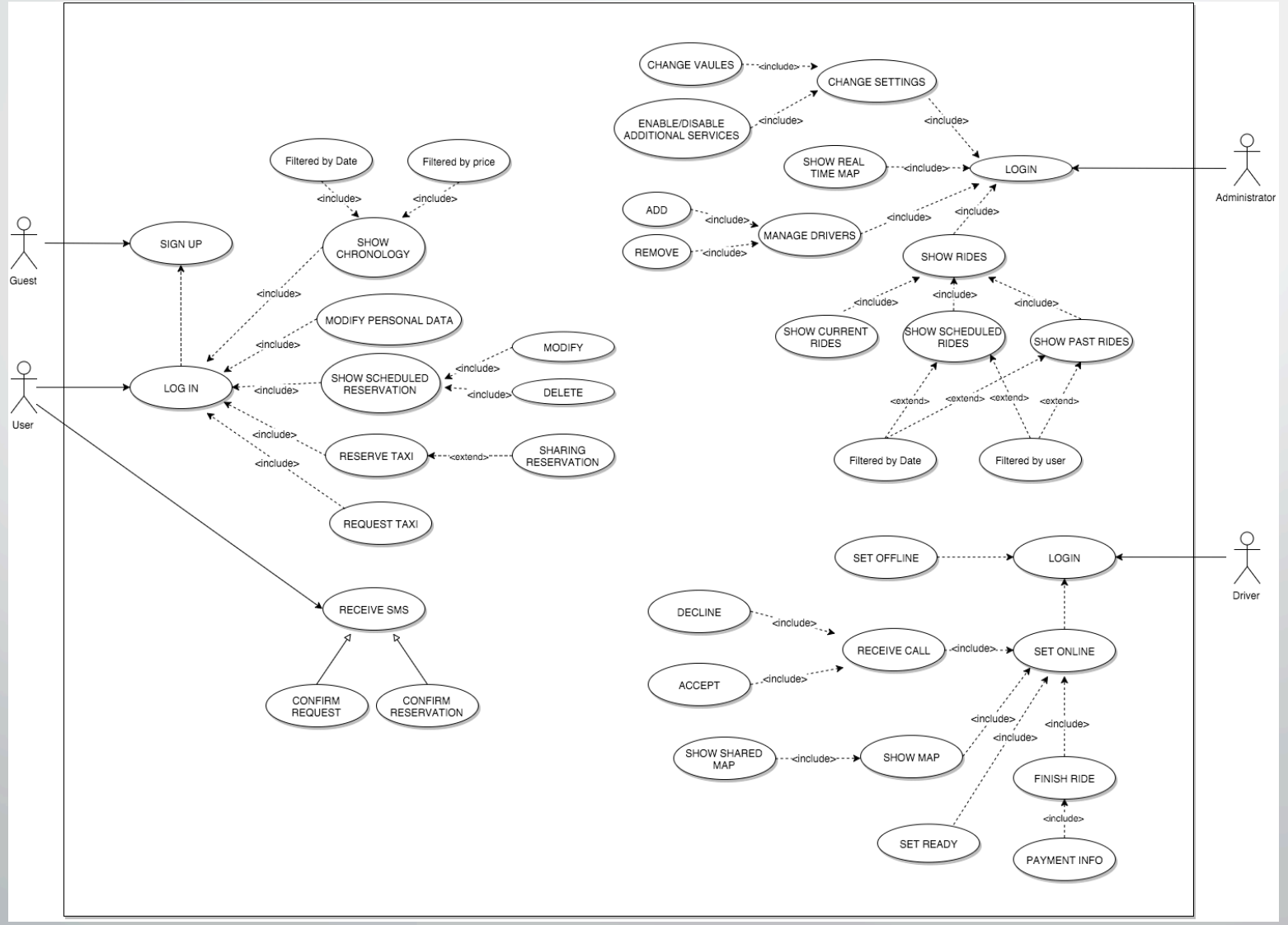




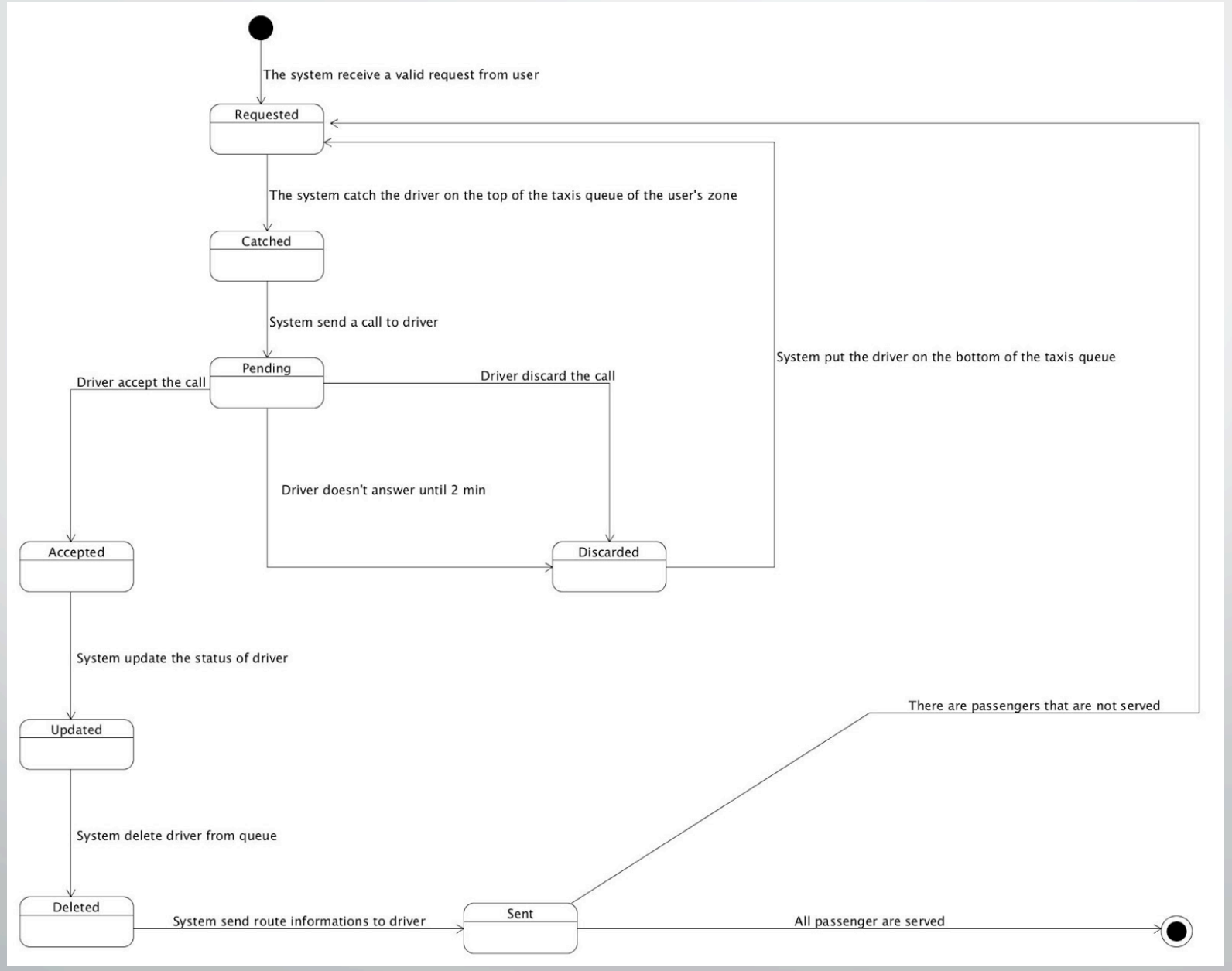
Scenarios Identifying

1. Download mobile app and procedure of sign up
2. Procedure of request taxi from mobile app
3. Procedure of request taxi from mobile app for more than 1 person
4. Procedure of Reservation with sharing (both mobile app + browser)
5. Driver procedure of login and wait for a call
6. Driver procedure of manage a call
7. Driver procedure to refuse a call
8. Driver procedure of report missing User
9. Administrator procedure of manage MTS

Use Case Diagram

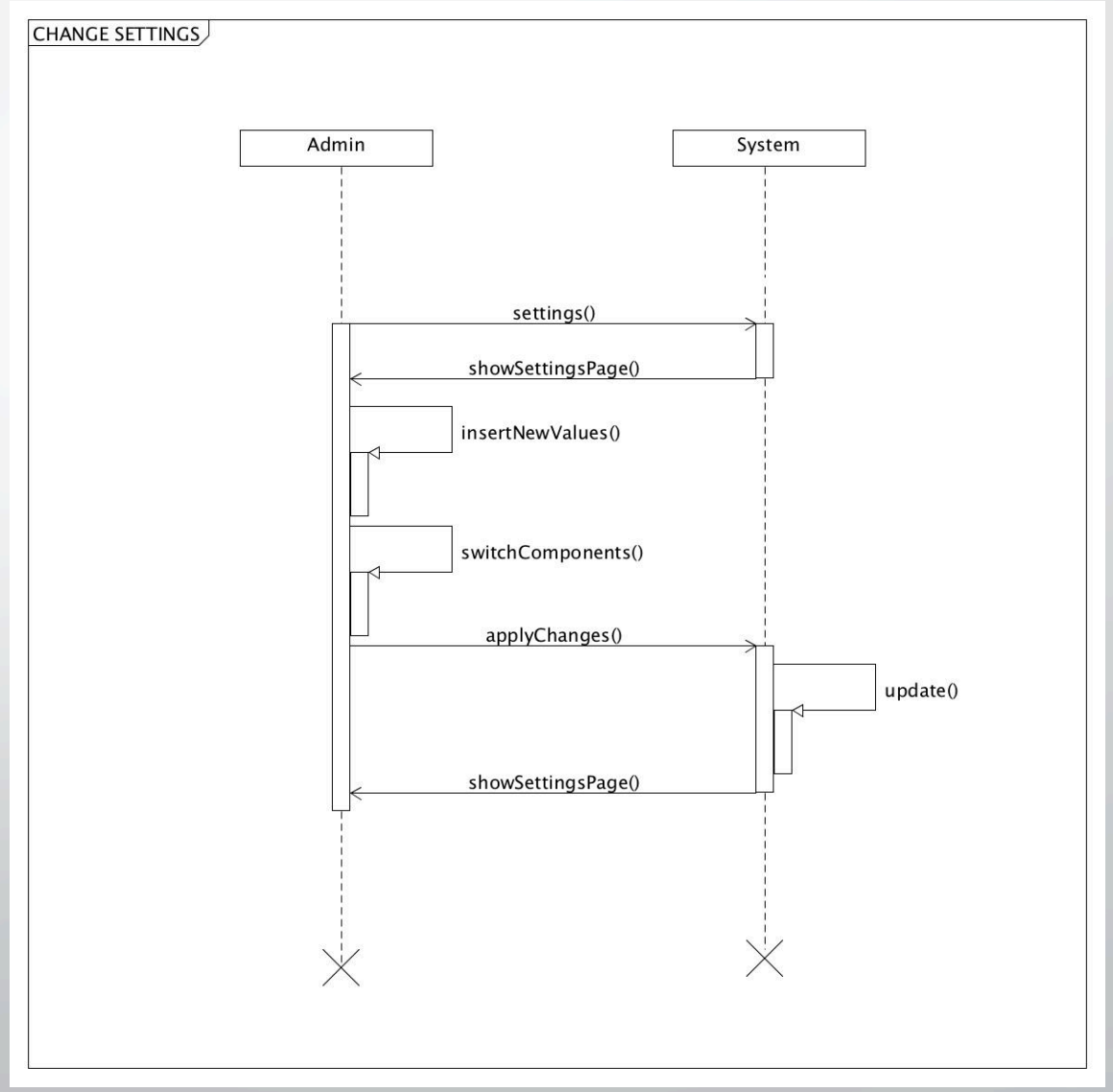


Statechart Diagram: Allocate Taxi



Sequence Diagram: Change System Settings

Name	CHANGE SETTINGS
Actors	Administrator
Entry Conditions	Administrator has already made login
Flow Events	• Administrator clicks settings button • He changes one or more values among the ones available • He switch on/off the additional services buttons • He clicks on done button
Exit Conditions	System shows him the same page with new values.
Exceptions	none



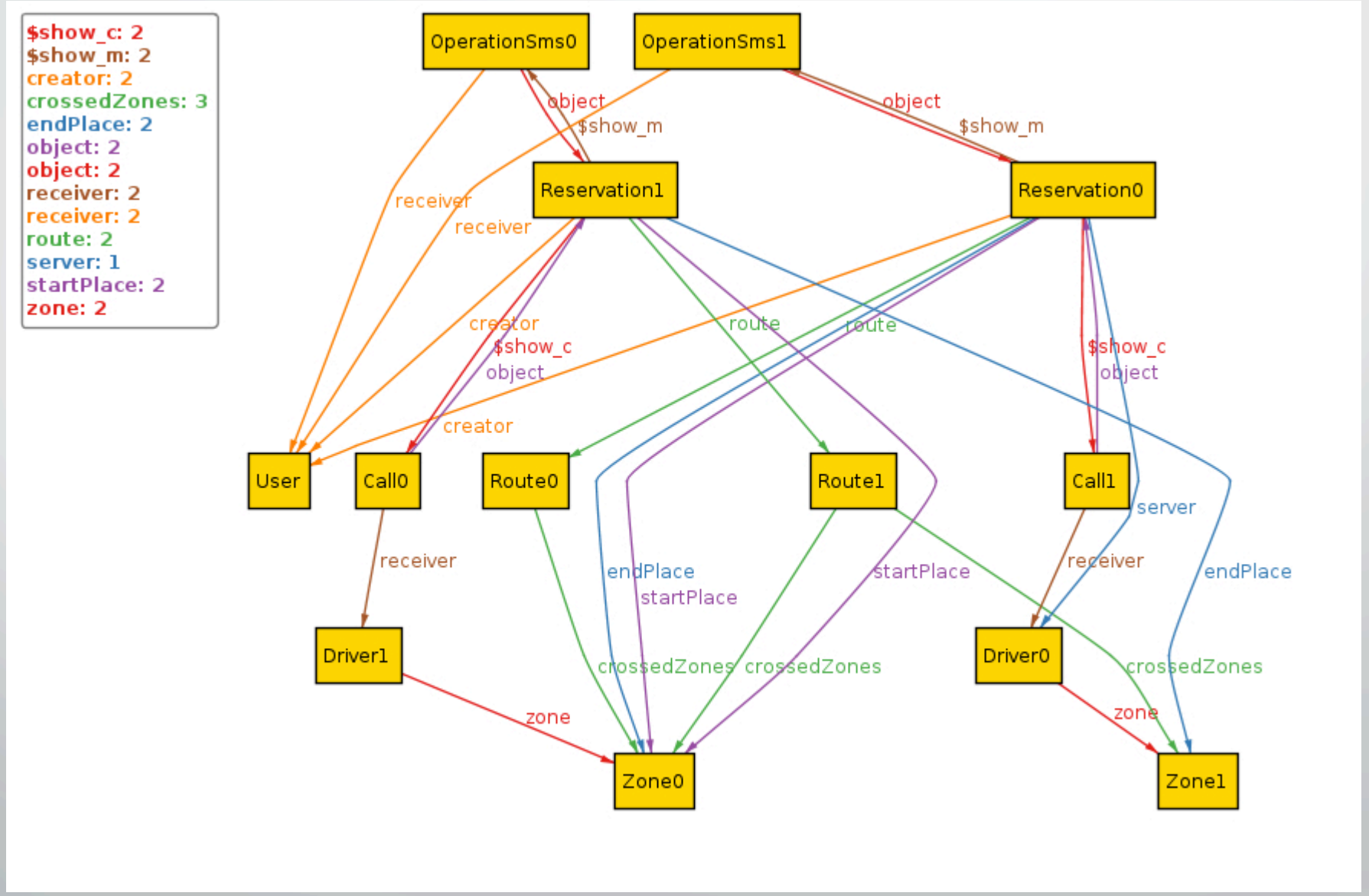
Alloy: signatures (1)

- sig User
- sig Driver
- abstract sig UserOperation
- sig Request extends UserOperation
- sig Reservation extends UserOperation
- sig Route

Alloy: signatures (2)

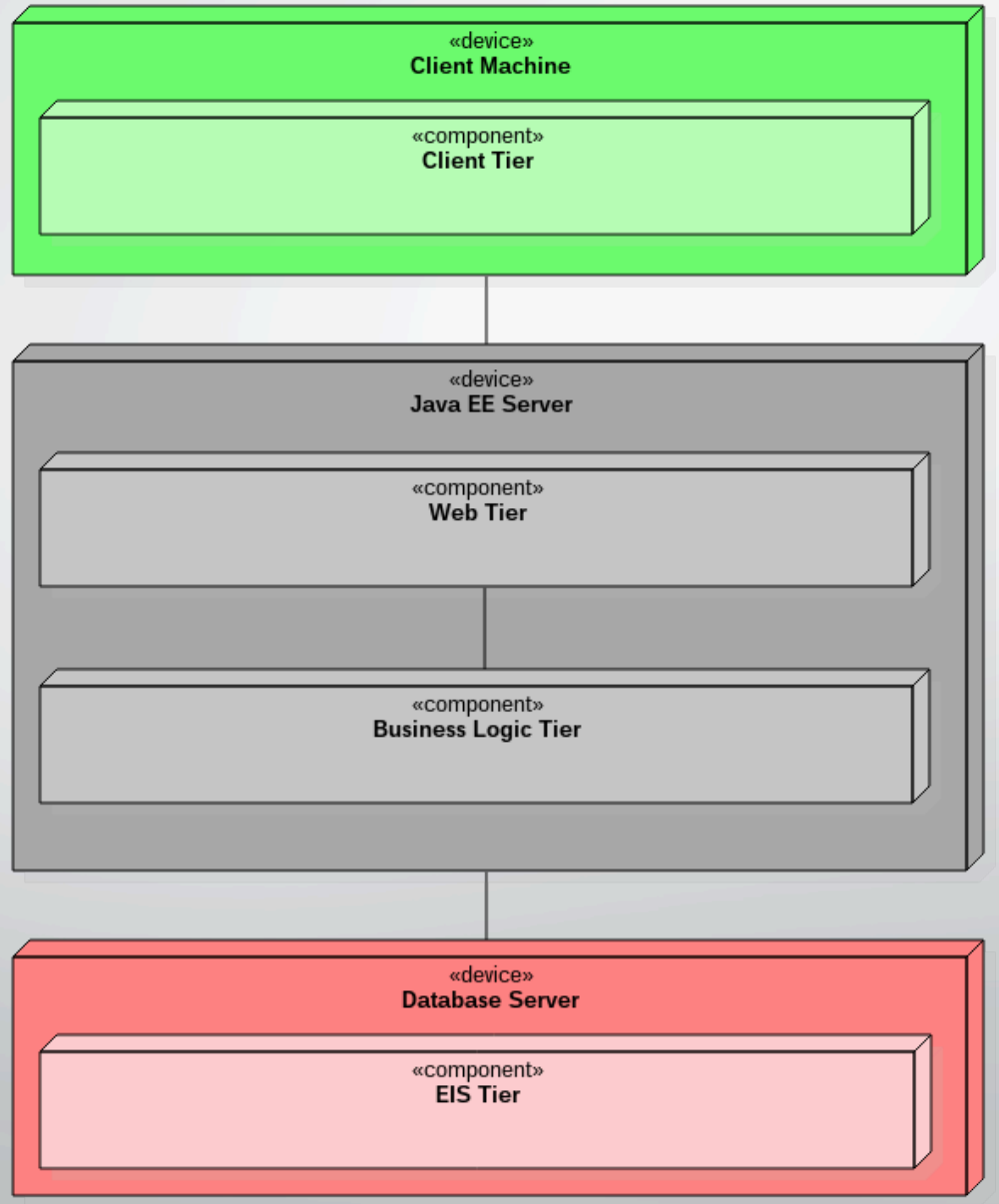
- sig Zone
- sig Call
- abstract sig SmsNotification
- sig RegSms extends SmsNotification
- sig OperationSms extends SmsNotification
- sig UpdateSms extends SmsNotification

Alloy: a world

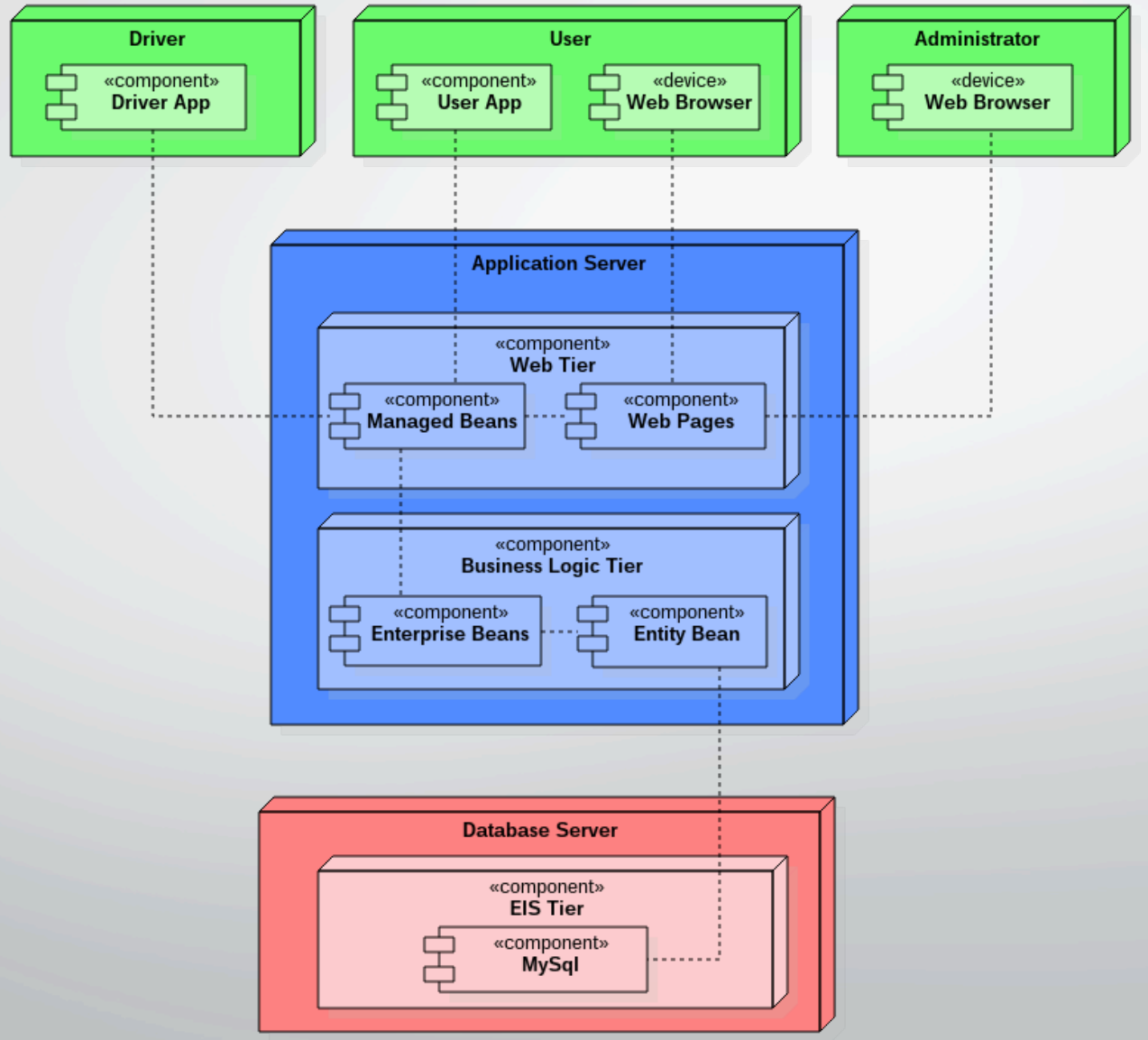


Design Document

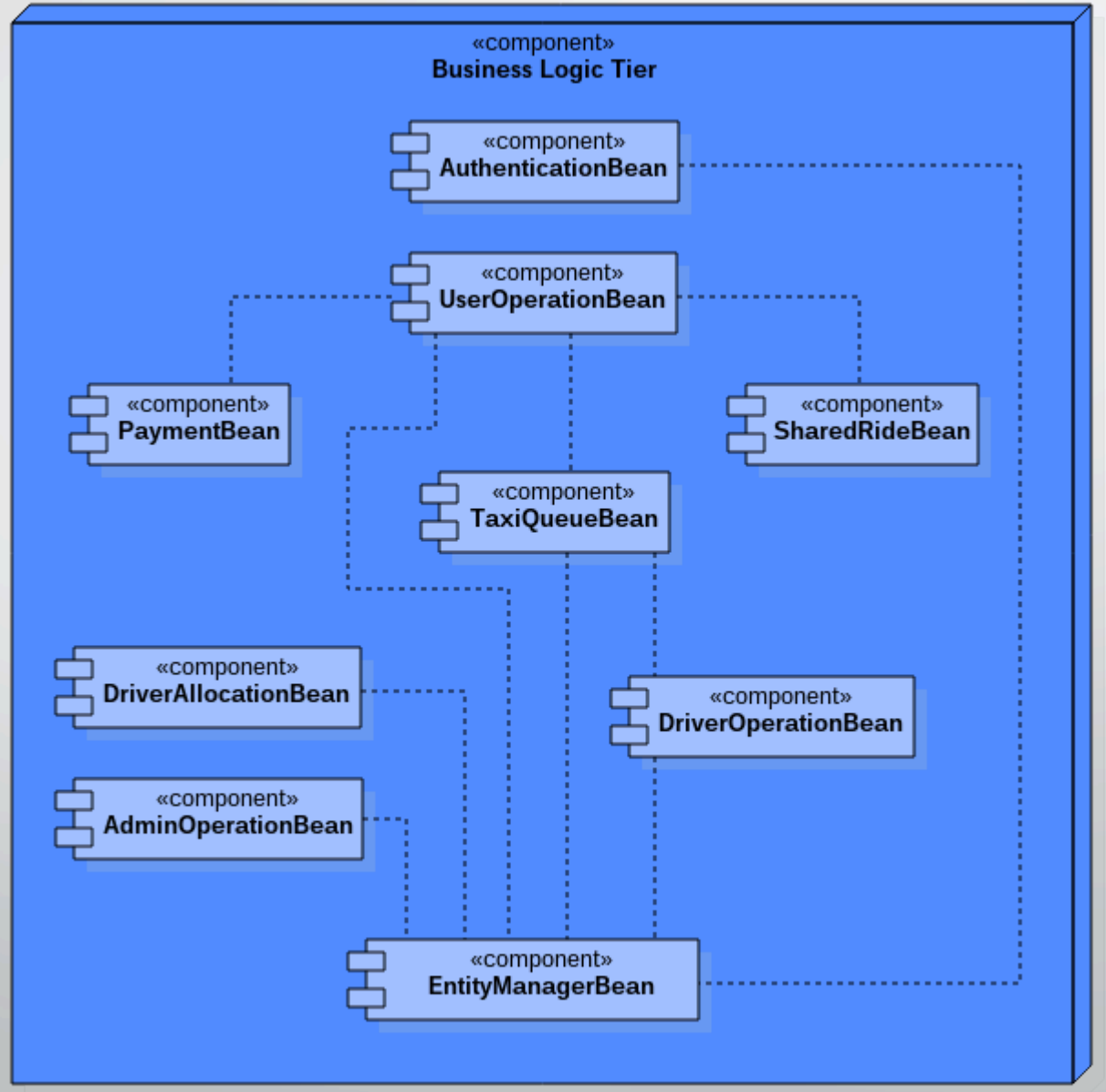
Architectural Overview



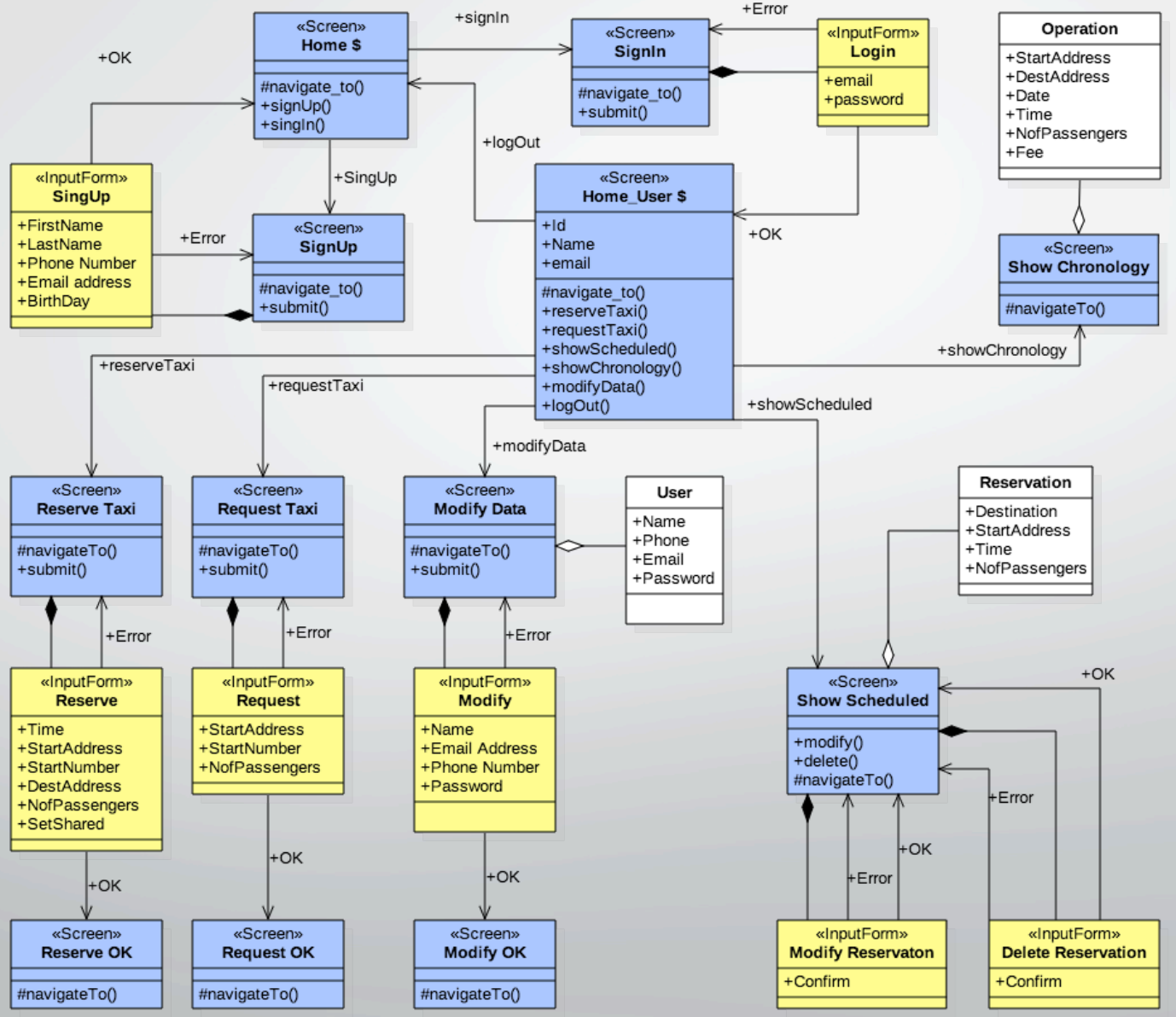
High Level Architecture



Business Logic Tier



User Experience Diagram



Architectural Style

- Client-Server architecture
- Style tier baser
  - 3 tiers
  - Client Tier
  - Web and Business tier (App Server)
  - Data Tier
- High cost of building infrastructure -> possibility to adopt cloud solution

Architectural Style

- System based on Java Enterprise Edition
- Uses Enterprise Java Beans
- Web presentation implemented by JSF and servlets
- JavaMail for sending Mail
- Java SMS, library that allows communication with external SMS gateway
- GlassFish server
- MySQL Server for the database

Patterns & other design decisions

- Observer Pattern: monitoring status of each taxi (offlline, free, busy, returning)
- Proxy Pattern: monitoring position of each taxi
- Singleton Pattern: used to create administrator
- About Mobile Applications
  - One release of user app for each apps market (Android, iOs, WindowsPhone)
  - One release of driver app only in android version
  - Communication via HTTP parsing request with XML language

- We also provide ER and logic rapresentation of data level

Algorithm Design: management of shared payment

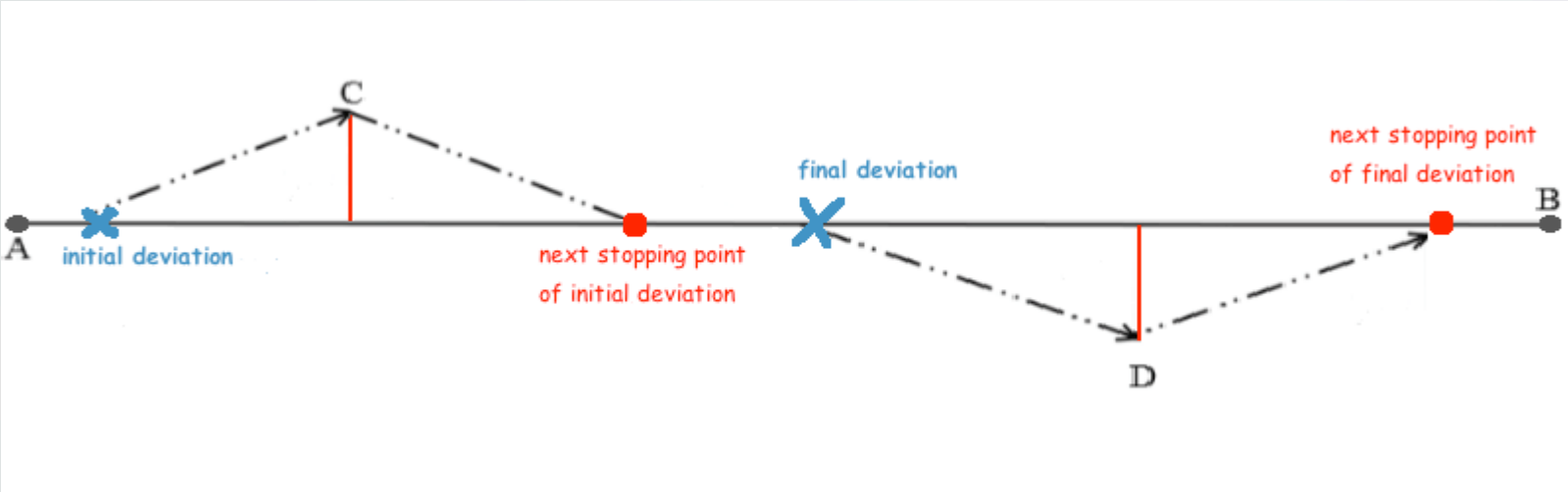
In order to reward all the people for using the sharing, and in order to ease our system, we pre-calculate the price of sharing before the allocation of the taxi. Even so the taxi take 30 minutes from A to B and the optimal time is 15 minutes, the users will pay the price for 15 minutes. The computation happens after the checking of a compatibility (10 minutes before the start-time of a reservation):

```
int cost(Address origin ,Address destination ,...) {
        cost:=fixed-fee ;
        prec:= origin;
        foreach (address as {all crossed addresses from next(origin) to destination following the best route in terms of time } )
        {
          cost+=(variable-fee* requested_time(prec , address ))/(number of user from prec to address );
          prec:=next(address);
        }
        return cost ;
}
```

Algorithm Design: composition of shared route

```
boolean checkCompatibilty (Reservation reservation) {
        foreach (container as Container)
            if (residual capacity of the container >= reservation capacity)
                if (there is affinity between container and reservation) {
                    insert reservation into container in a position that mantains
                    the ascending order of estimated time arrival ;
                    return true ;
            }
        return false ;
}
```

Algorithm Design: affinity



- The next stopping point is the nearest destination/starting address (of another user or of the same) following the route. The function affinity(...) handles 3 case:

1) when the final deviation is situated from C and the next stopping point;
2) when there is at least a next stopping point from C to final deviation;
3) when the route is entirely contained in A-B;

Therefore in the function there is a term **value** (choosable) that represent the maximum time from initial-deviation to C.

Algorithm Design: management of the driver allocation

- If a driver set is status to ready either he will be driven by the system in a place where he has to wait for a new call or he will wait where is located. This choice is made by the system according to the following policy: the driver remain in his zone if and only if the average of drivers in each zone is greater than the number of drivers located in the same zone of him. In the event that the condition isn’t satisfied the driver will be driven in the nearest zone where it’s valid

Code Inspection Document

InjectionManager-Impl.java

It does/provide:

- Resource Injection
- Injection of classes and instances
- Lifecycle callback methods:
  - PostConstruct
  - PreDestroy

Most Common Issues found

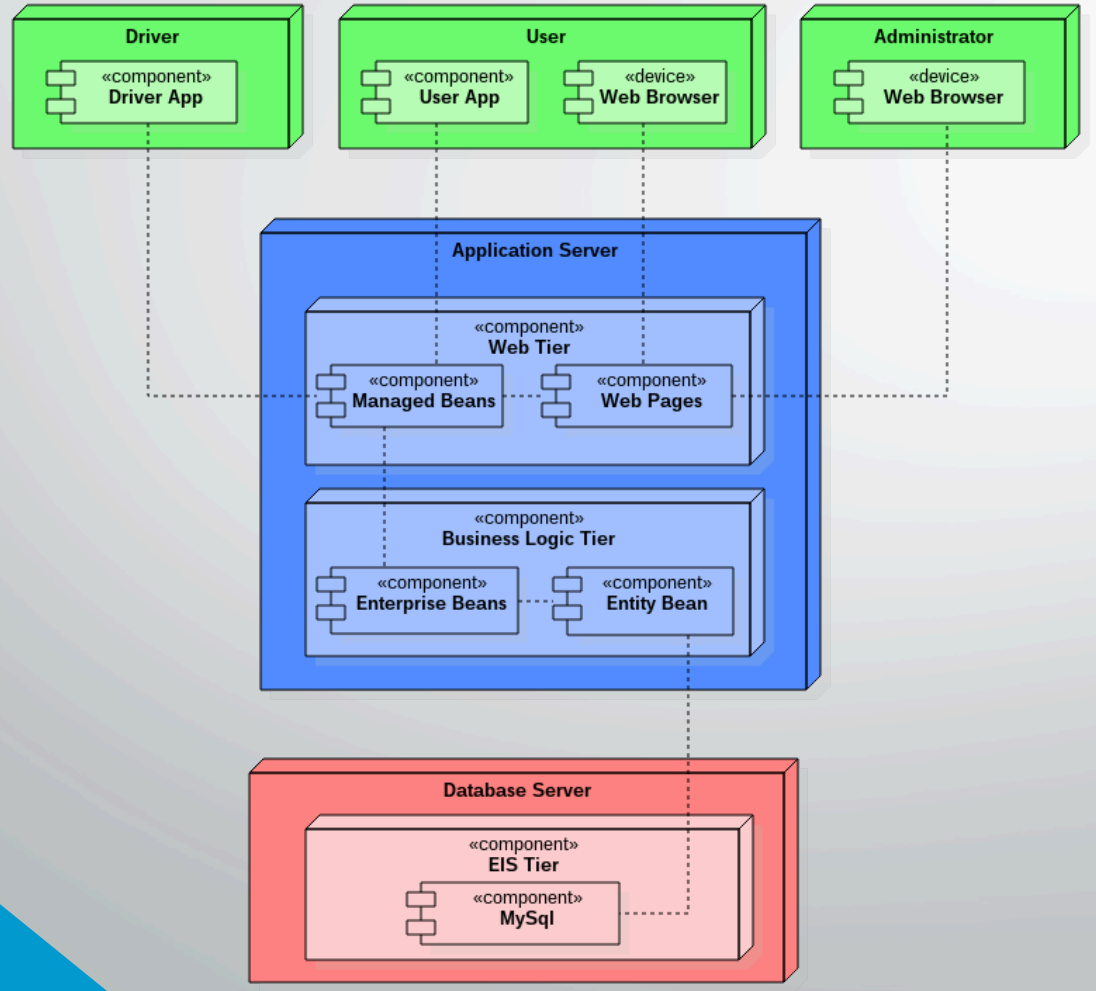
- Parameters without meaningful names
- Absent or too short indentation
- Line break splitting method’s names
- Too long lines
- Wrong alignment of lines in the same level

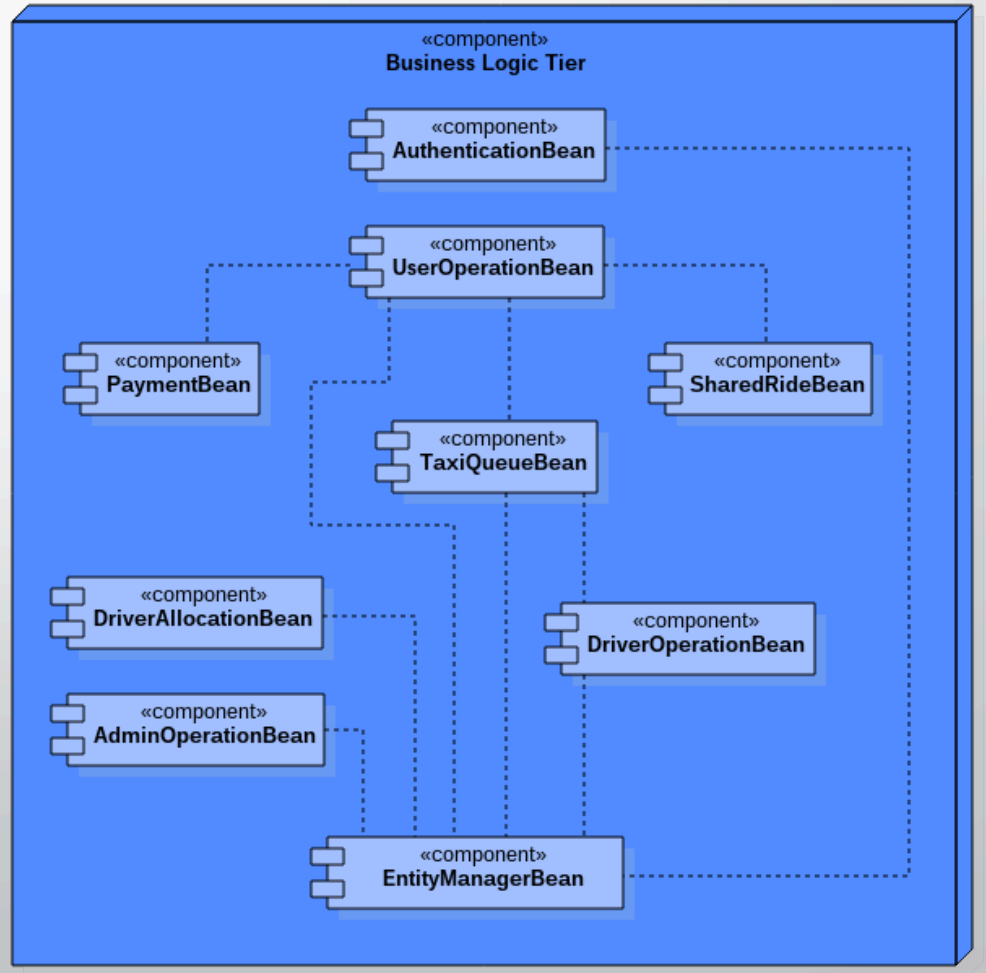
Integration Test Plan Document

Integration Test Plan – Entry Criteria

- Assumption : for each stand-alone component developers has to provide at least 95% of unit test coverage
- Assumption: all APIs used works correctly and are in a stable version
- Integration Test can be done if each component involved are completely developed and ready to use
- Otherwise integration can be made with specific stubs or drivers

Elements to be integrated

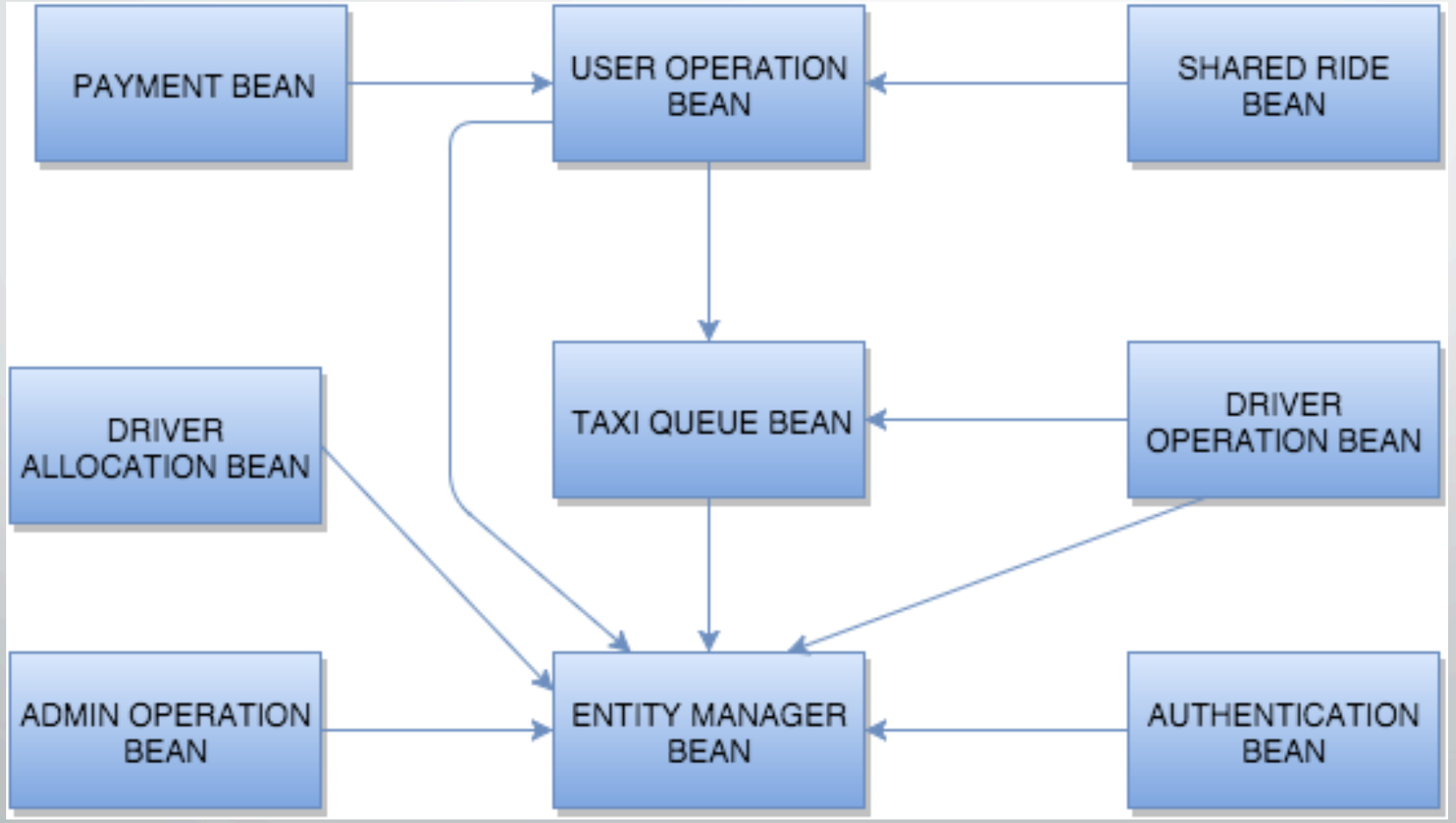




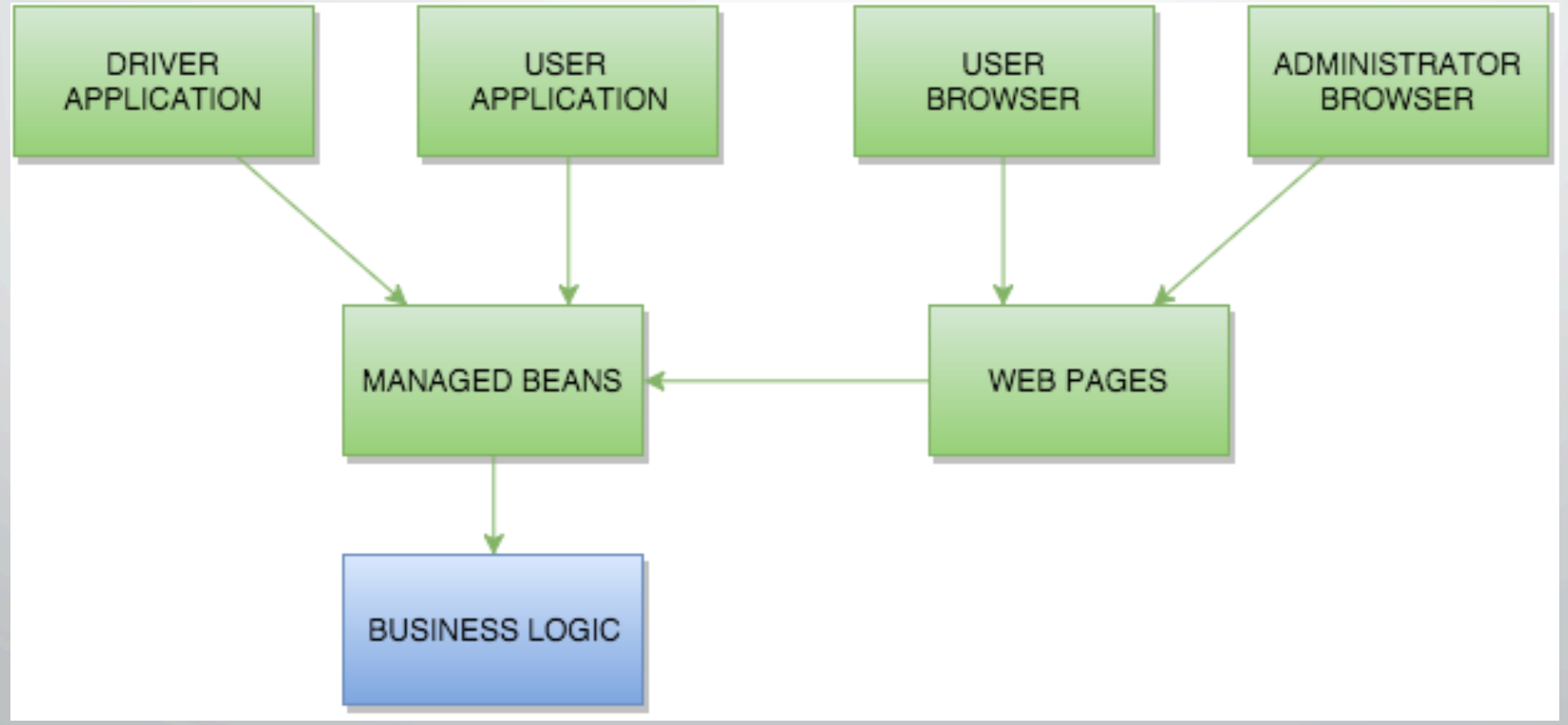
Integration Test Plan – Strategy

- Bottom Up approach -> high modularity
- First step: integrate server side subsystem
- Used specific drivers and stub to simulate behavior of clients
- Second integration of server side with clients side (User, Driver, Administration)

Integration sequence: business logic tier



Integration Sequence: front-end components



Integration test procedures

3.2.1 Integration Test Procedure TP4

Test Procedure Identifier	TP4
Purpose	This test procedure verifies that the driver's frontend part works correctly (for the list of admin's operations see the RASD, page 12-13)
Procedure Steps	I9 → I12 → I13 → I14 → I11 → I10

3.2.2 Integration Test Procedure TP5

Test Procedure Identifier	TP5
Purpose	This test procedure verifies that the admin's frontend part works correctly (for the list of driver's operations see the RASD, page 12)
Procedure Steps	I15 → I20 → I17 → I16 → I18 → I19

Integration test cases

3.2.7 Integration Test Case I9

Test Case Identifier	I9T1
Test Item(s)	Driver Application -> Managed Beans
Input Specification	Fulfill login form
Output Specification	Successful login
Environmental Needs	Device

Test Case Identifier	I9T2
Test Item(s)	Driver Application-> Managed Beans
Input Specification	Insert wrong login-data
Output Specification	Login Not permitted
Environmental Needs	Device

3.2.8 Integration Test Case I10

Test Case Identifier	I10T1
Test Item(s)	Driver Application -> Managed Beans
Input Specification	Make logout
Output Specification	Successful logout
Environmental Needs	Device,I9 succeeded

Project Plan Document

Functional Point & COCOMO (Approximately)

- TOTAL FP NUMBER = **132**
- SLOC = FPtotal * 46 = 132 * 46 = **6072**
- EFFORT = $2.94 * EAF * (KSLOC)^{E}$ = 2.94 * 1.00 * $(6.072)^{1.0977}$ = **21.4 Person / Month**
- DURATION = $3.67 * (21.4)^{0.3179}$ = **9.72 Months**

- NofPEOPLE = EFFORT / DURATION =
  = 21.4 / 9.71 = 2.25 people → 3 PEOPLE

COCOMO – Test Online Result

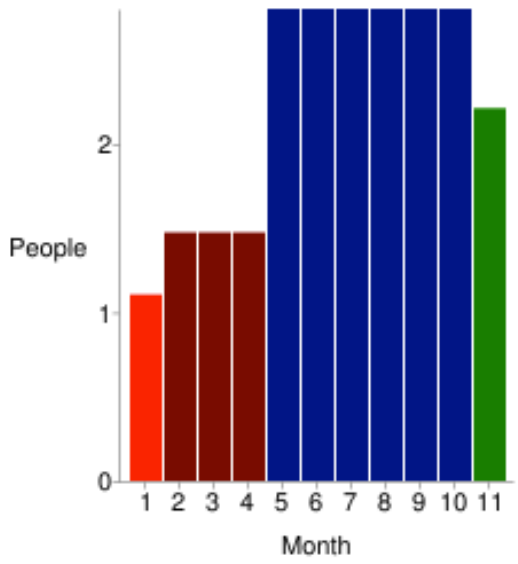
Results

Software Development (Elaboration and Construction)

Effort = 24.2 Person-months
Schedule = 10.5 Months
Cost = $38684

Total Equivalent Size = 6072 SLOC

Staffing Profile



Acquisition Phase Distribution

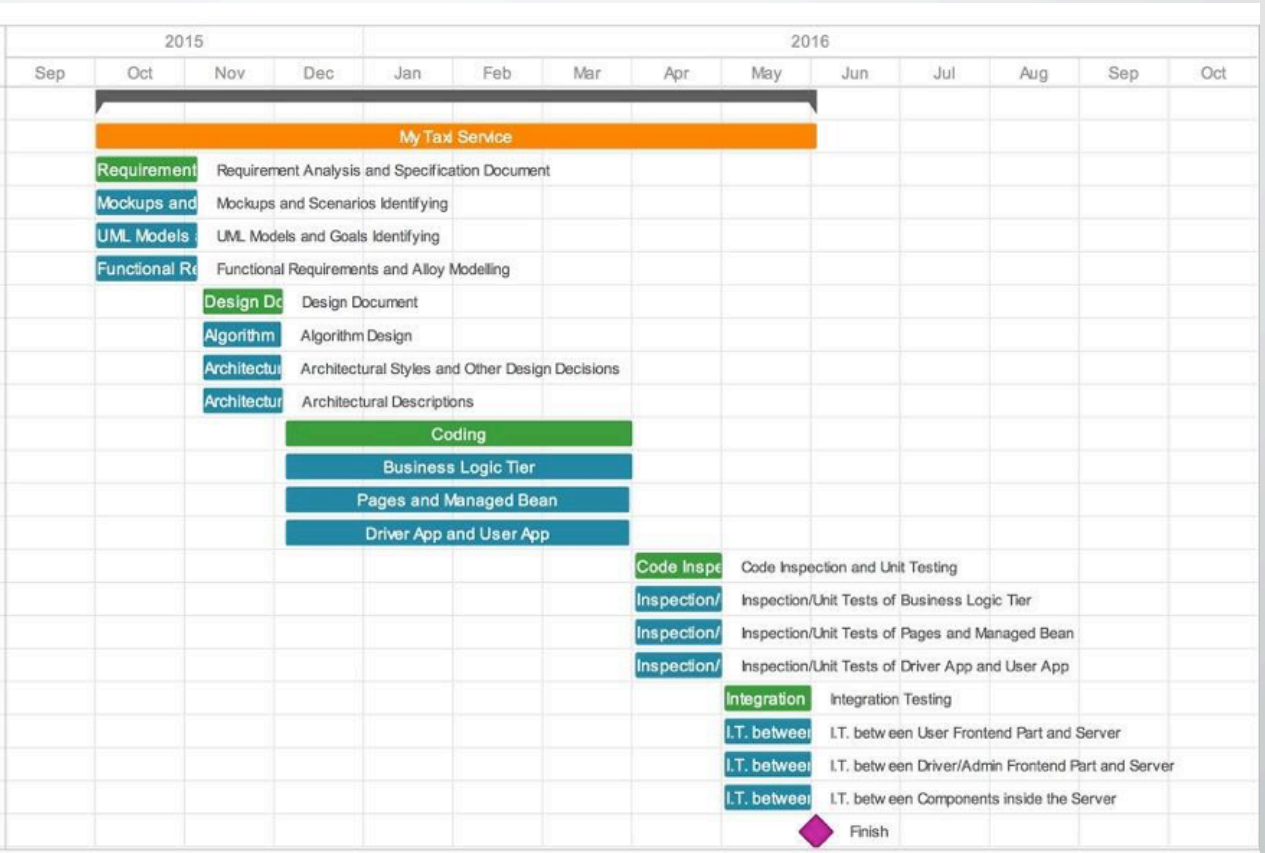
Phase	Effort (Person-months)	Schedule (Months)	Average Staff	Cost (Dollars)
Inception	1.5	1.3	1.1	$2321
Elaboration	5.8	3.9	1.5	$9284
Construction	18.4	6.6	2.8	$29400
Transition	2.9	1.3	2.2	$4642

Software Effort Distribution for RUP/MBASE (Person-Months)

Phase/Activity	Inception	Elaboration	Construction	Transition
Management	0.2	0.7	1.8	0.4
Environment/CM	0.1	0.5	0.9	0.1
Requirements	0.6	1.0	1.5	0.1
Design	0.3	2.1	2.9	0.1
Implementation	0.1	0.8	6.2	0.6
Assessment	0.1	0.6	4.4	0.7
Deployment	0.0	0.2	0.6	0.9

Tasks

Task name	Start time	End time	Assigned
Total Estimate	01/10/15 00:00	02/06/16 18:31	
My Taxi Service	01/10/15 00:00	02/06/16 18:31	
Requirement Analysis and Specification Document	01/10/15 00:00	05/11/15 05:00	
Mockups and Scenarios Identifying	01/10/15 00:00	05/11/15 05:00	Davide
UML Models and Goals Identifying	01/10/15 00:00	05/11/15 05:00	Lorenzo
Functional Requirements and Alloy Modelling	01/10/15 00:00	05/11/15 05:00	Massimo
Design Document	07/11/15 00:00	04/12/15 00:00	
Algorithm Design	07/11/15 00:00	03/12/15 05:00	Lorenzo
Architectural Styles and Other Design Decisions	07/11/15 00:00	03/12/15 05:00	Davide
Architectural Descriptions	07/11/15 00:00	04/12/15 00:00	Massimo
Coding	05/12/15 00:00	01/04/16 05:00	
Business Logic Tier	05/12/15 00:00	01/04/16 05:00	Massimo
Pages and Managed Bean	05/12/15 00:00	31/03/16 05:00	Lorenzo
Driver App and User App	05/12/15 00:00	31/03/16 05:00	Davide
Code Inspection and Unit Testing	02/04/16 00:00	01/05/16 00:00	
Inspection/Unit Tests of Business Logic Tier	02/04/16 00:00	01/05/16 00:00	Massimo
Inspection/Unit Tests of Pages and Managed Bean	02/04/16 00:00	01/05/16 00:00	Lorenzo
Inspection/Unit Tests of Driver App and User App	02/04/16 00:00	01/05/16 00:00	Davide
Integration Testing	02/05/16 00:00	01/06/16 00:00	
I.T. between User Frontend Part and Server	02/05/16 00:00	01/06/16 00:00	Davide
I.T. between Driver/Admin Frontend Part and Server	02/05/16 00:00	01/06/16 00:00	Lorenzo
I.T. between Components inside the Server	02/05/16 00:00	01/06/16 00:00	Massimo
Finish	02/06/16 18:31	02/06/16 18:31	



Resource Allocation

- We suppose that a student like us works 25-30 hours a week.
  Accordingly, each component of the group works 100-120 hours/month.
- The total amount of hours of work is obtained by
  (100-120)hours * 3 student * 8 months : is equal to **2400 - 2880 hours**. Instead, the duration of Cocomo II is
  21.4 person/month *152 hours : is equal to **3252.8 hours**.

  We can obviously notice that the estimation of COCOMO II is oversized respect to the real time we suppose to spend for the project. This could be linked to the statistic nature of COCOMO II.
- Inspite of all, the time we suppose to spend for the project differs from the COCOMO II estimation more or less by 20%: this make our project plan acceptable.

Risks (1)

- Type: project risk
- Probability: moderate
- Impact: serious
- Summary: unavailability of a member delays other member work
- Contingency Plan: reallocation of tasks

Risks (2)

- Type: technical risk
- Probability: moderate
- Impact: serious
- Summary: underestimation of the database cloud service
- Contingency Plan: simulation with many parallel operation, evaluation of performances and possible purchase of a more powerful service