Software Engineering 2: myTaxiService

CASATI FABRIZIO, CASTELLI VALERIO

Context

- Aim: build a taxi management system for a medium-large city

- The city area is divided in taxi zones, there must be a taxi queue for each zone

- Taxis are matched with requests depending on the zone they come from

- Interaction with the system happens through smartphones, tablets and web browser

- Support both immediate requests and reservations (for registered users only)

- The existing taxi management service of the city is fully migrated and decommissioned

- 24/7 service (ideally)

Domain Assumptions (I)

- Taxis are uniquely associated to drivers and viceversa

- Passengers don't need to choose a particular taxi driver among those available for their ride

- Taxi drivers are provided with a mobile phone with an active data plan by the city council

- Passengers are not allowed to place reservations more than 15 days in advance or cancel reservations after a taxi has already been scheduled for them

- Rides can be requested by all passengers, while reservations can only be placed by registered passengers

- The only taxis eligible for fulfilling a reservation are the ones present in the queue of the zone associated with the reservation source address 10 minutes before the scheduled meeting time

Domain Assumptions (II)

- In order to receive ride requests, a taxi driver must explicitly mark himself as available

- A taxi driver will not be able to receive ride requests while he's unavailable

- Taxis that are considered to be out-of-city will not be able to receive calls

- A taxi driver who is currently on a ride will not be able to receive calls

- A taxi driver must always notify the system when he terminates a ride

- After a taxi driver has been associated to a call, he must confirm or refuse the request within two minutes. After that period of time, the call is considered refused

- If no taxis are available to fulfill a reservation 10 minutes before the scheduled meeting time, attempts of rescheduling are to be made at intervals of 2 minutes for at most 20 times

Functional Requirements

- The city administration must have the possibility to enter and update taxi driver data and the taxi zone division

- Taxi drivers must be able to
  - communicate their availability status
  - receive, accept, refuse and drop ride requests
  - communicate they have terminated a ride

- Passengers must be able to request rides and, if logged in, also place and manage their reservations

- The system must support third party expansion through plugins and remote services

- It must be possible to verify the identity of a passenger before taking him onboard

Stakeholders and actors

STAKEHOLDERS

- Passengers

- Taxi drivers

- City council

- Taxi drivers' union

- Mobile phone producers

- Wireless carriers

- Third party developers

ACTORS

- Guest passenger

- Guest taxi driver

- Logged in passenger

- Logged in taxi driver

- Administrative personnel

- Mapping service

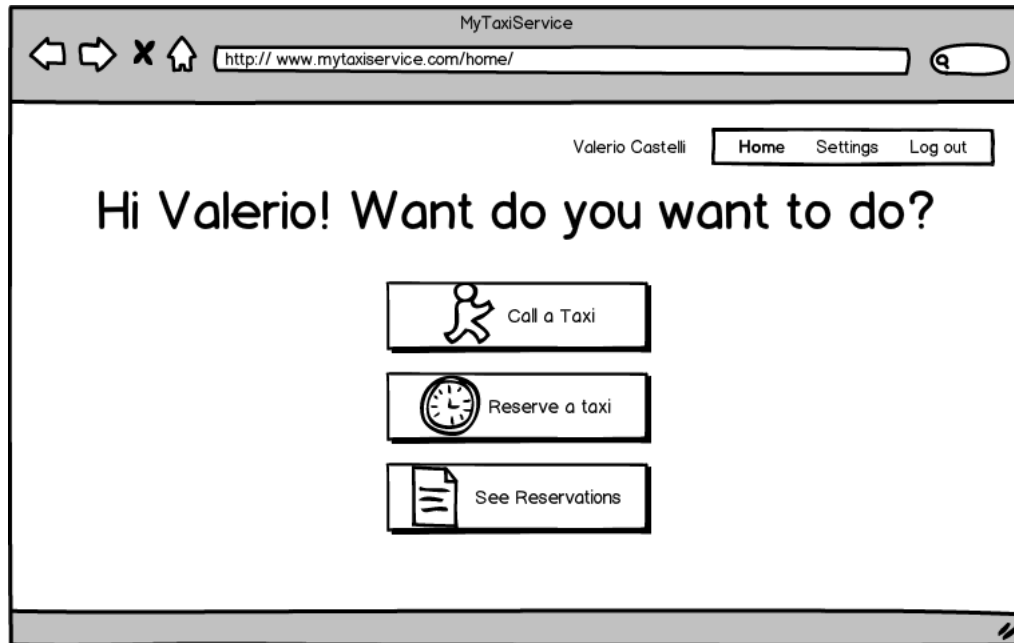- Remote services

UI Mockup (I)

HOMEPAGE

USER LOGIN

UI Mockup (II)

PASSENGER HOMEPAGE
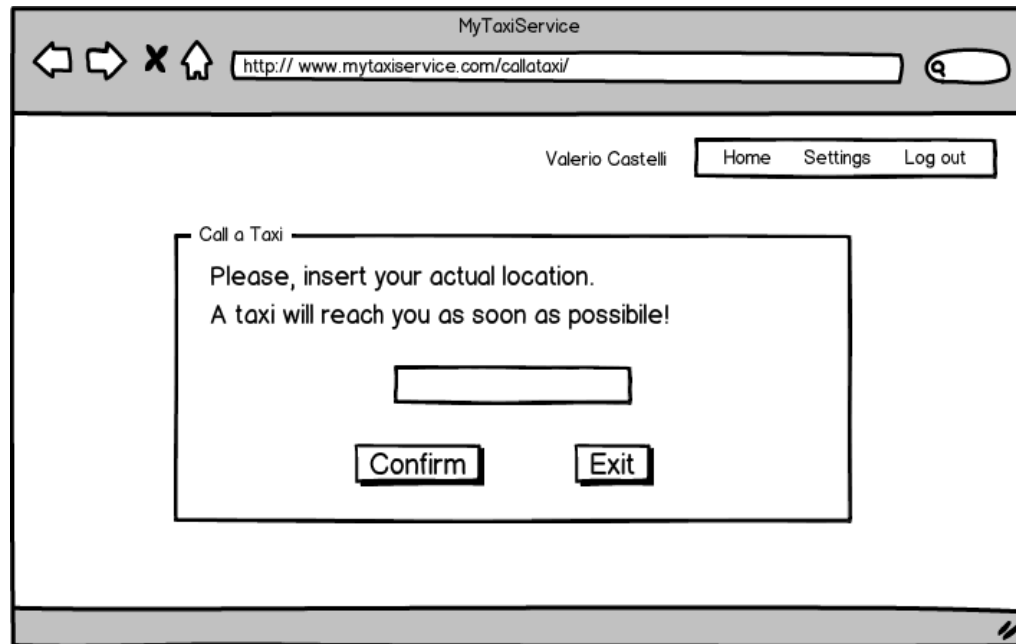
PASSENGER RESERVATIONS
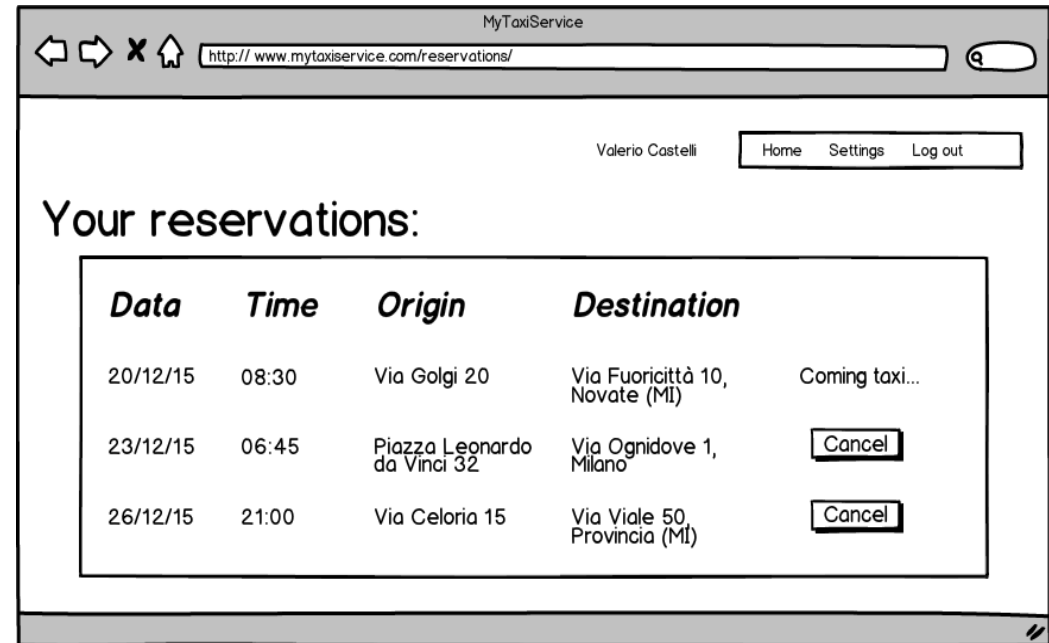
UI Mockup (III)

PASSENGER REQUEST
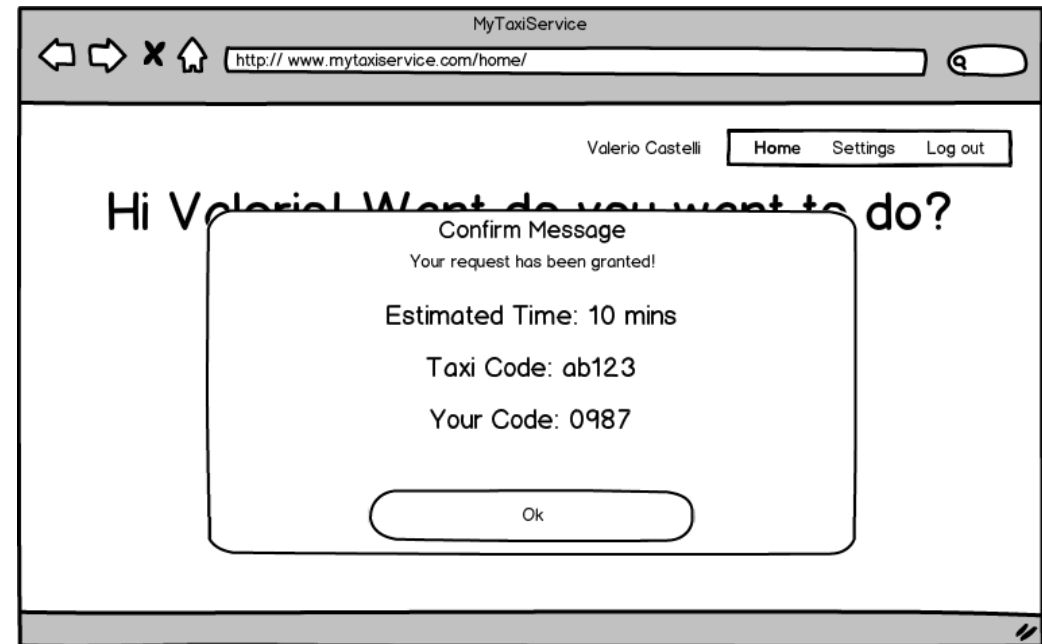
REQUEST CONFIRMATION

UI Mockup (IV)

MOBILE LOGIN
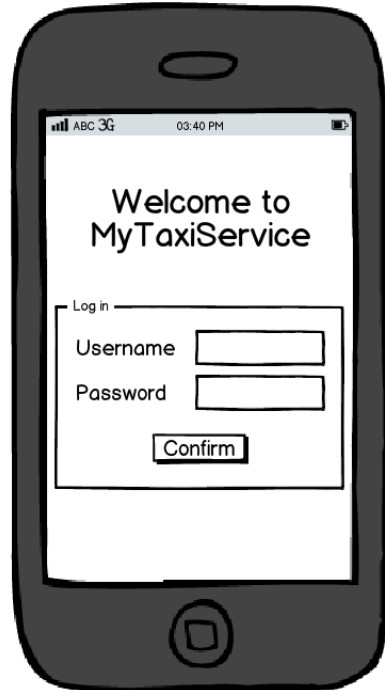
TAXI AVAILABLE
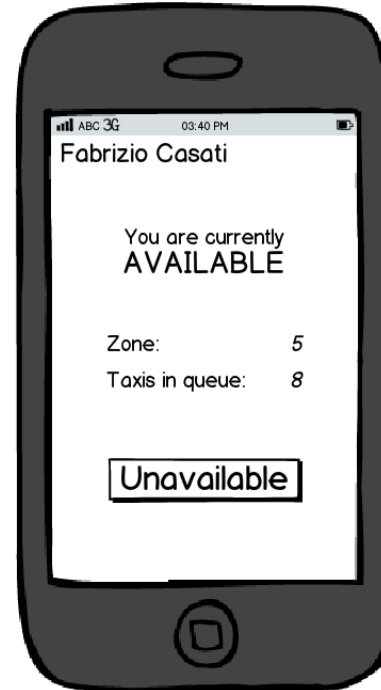
TAXI UNAVAILABLE

UI Mockup (V)
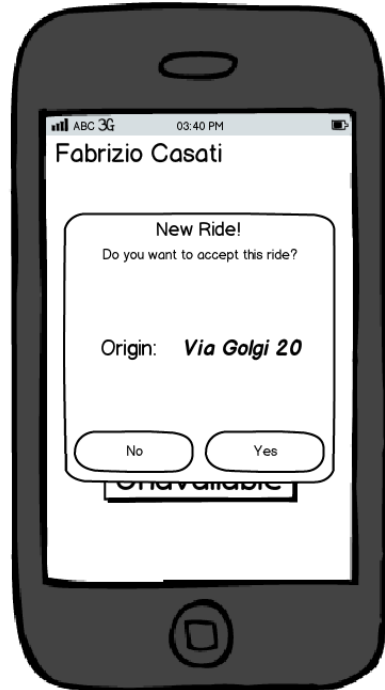
RIDE REQUEST   ACCEPTED RIDE   CURRENTLY RIDING
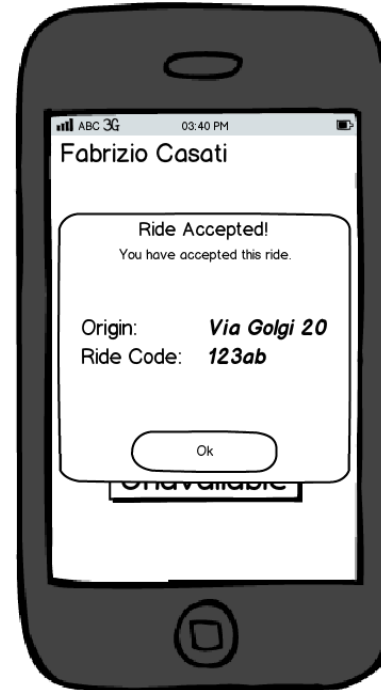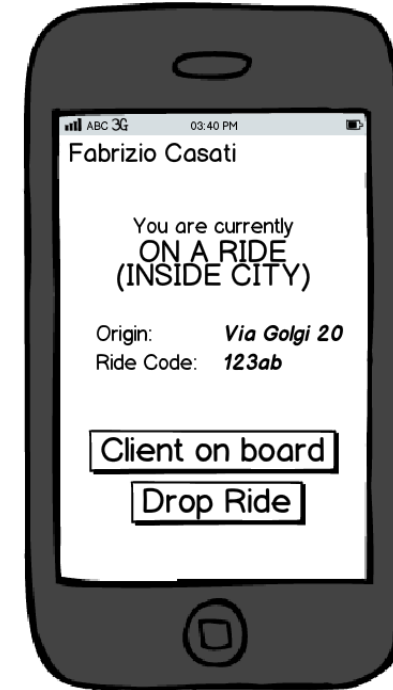
Architectural Design

- Multi-layer and multi-tier architecture

- Access to DBMS is mediated by an intermediate abstraction (Data Layer) for flexibility

- Business Layer implements core functionalities, exposed through an Application Façade

- A subset of functionalities is made available for remote calls via a Service Layer (SOAP and JAX-WS)

- Communication between remote clients and central system happens through a message broker and notification system

- Implementation based on Java EE
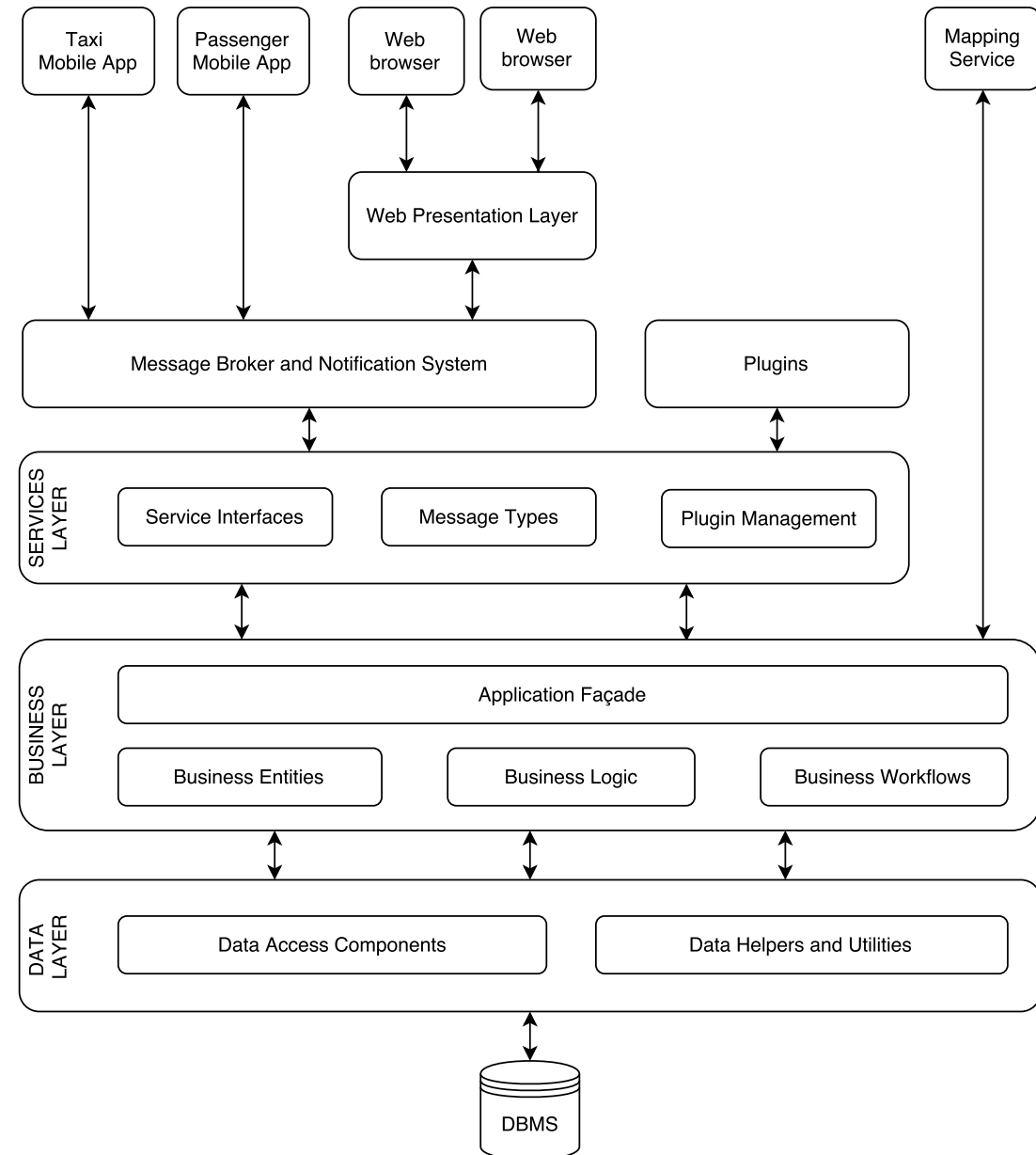
Architecture: high level components

- Account Management
  - Passenger registration and login procedure
  - Settings management and password retrieval

- Taxi Management System
  - Maintains the availability status of each taxi updated
  - Manages the taxi queue associated with each zone in the city
  - Accepts and handles taxi reservations
  - Fulfills taxi requests by selecting the first available taxi in the corresponding taxi zone

- Mapping Service
  - Reverse geocoding, maps and ETA

- System Administration
  - Insert, update, delete taxis and zones
  - API permission management
  - Statistical queries and service monitoring

- Remote Services Interface
  - Exposes APIs to third party services

- Notification System
  - Sends notification to users

- Data Access Utilities
  - Mediates access to DBMS

- Database Management System (DBMS)

High level components

- Taxi Management System:
  - Reservation Management
  - Request Management
  - Location Management
  - Taxi Management

- Account Management:
  - Passenger Registration
  - Login
  - Password Retrieval
  - Settings Management

- System Administration:
  - API Permission Management
  - Zone Division Management
  - Taxi Driver Management
  - Service Statistics
  - Plugin Management
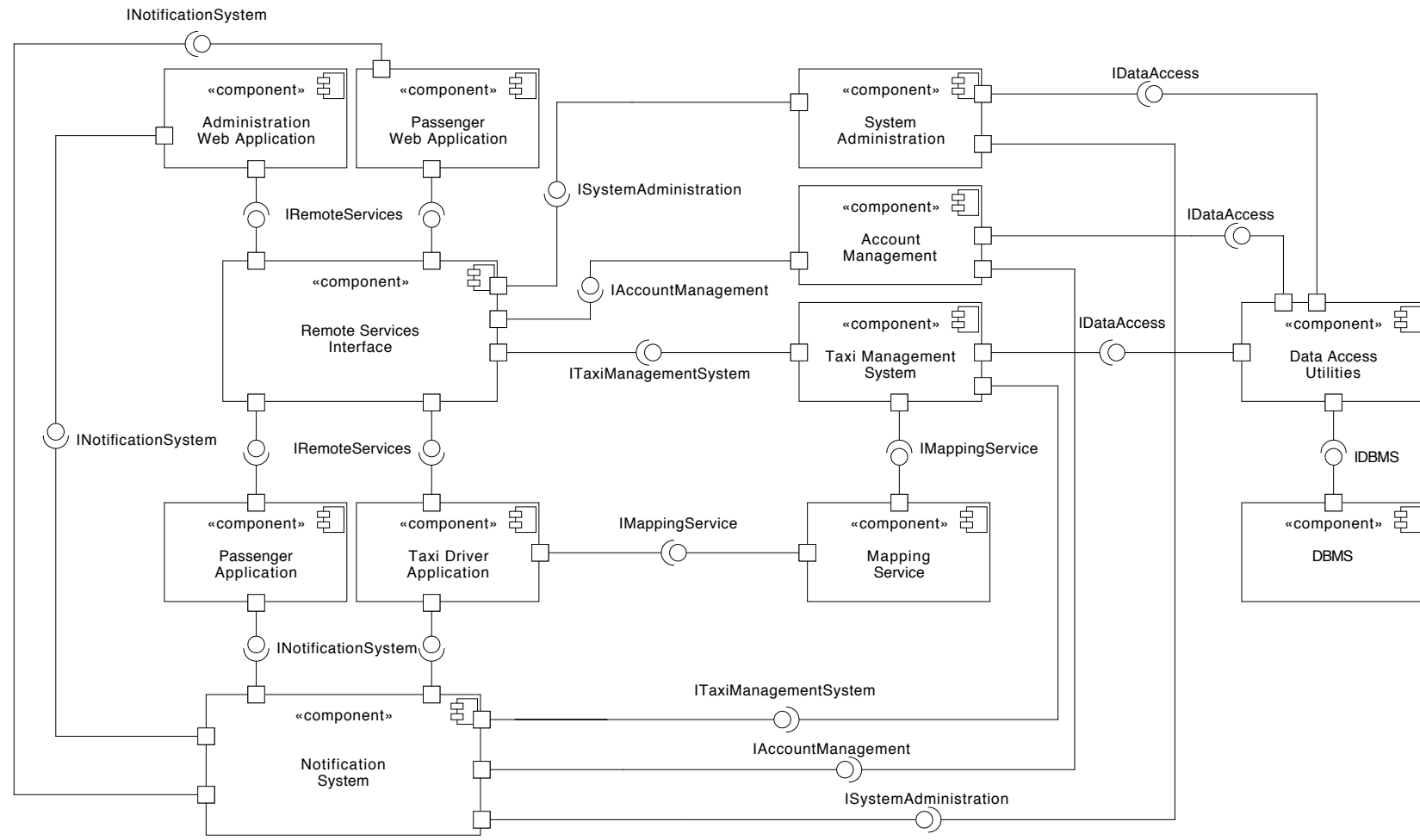
Deployment view
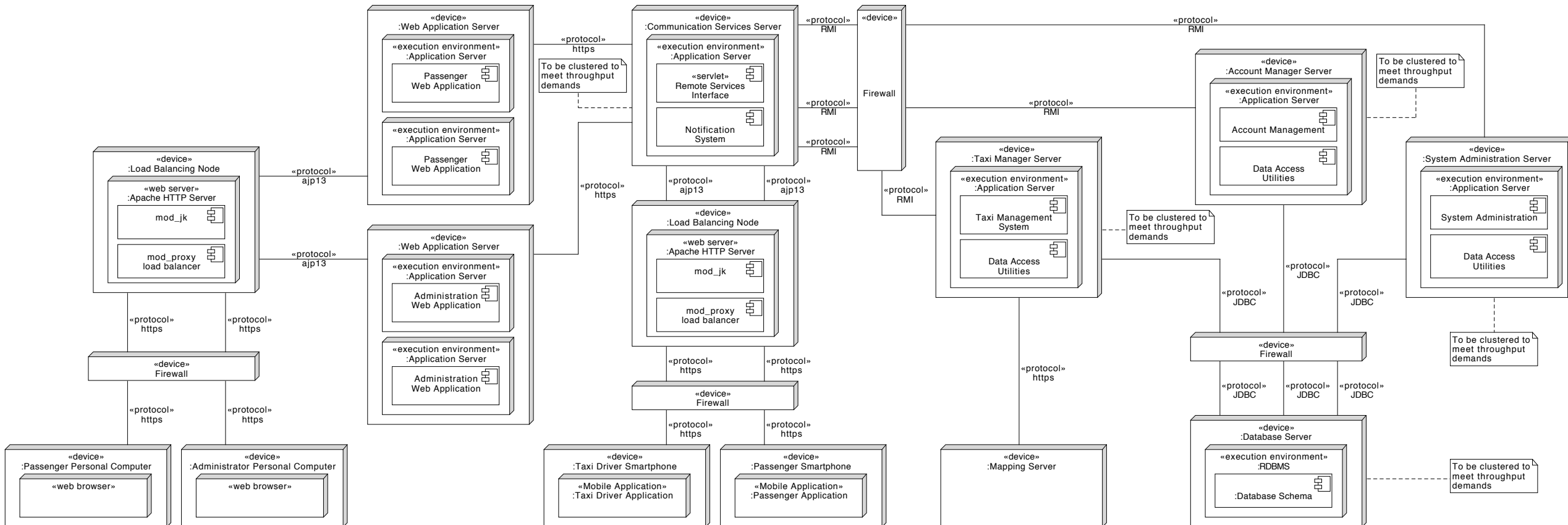
Remote API

- Methods exposed by the core system via a web service interface

- Service Oriented Architecture (SOA)

- XML-based requests (SOAP)

- Managed by the JAX-WS API of Java EE

- Each session bean exposes a subset of the available methods

- Plugins are allowed to extend the offered API

- Remote third party services can invoke the exposed methods

- Different levels of privileges are supported

- Details discussed in the DD (Component Interfaces section)

Architectural Styles

- Service Oriented Architecture (SOA)
  - Used to expose the APIs of the core system
  - Allows easily expansion of the offered functionalities

- Layered Architecture
  - Separation of concerns
  - Great clarity and flexibility
  - Different services run on different machines

- Client/Server Architecture
  - Business Logic implemented in servers
  - Clients used only for presentation purposes

- 4-tier Architecture
  - Better security and resilience
  - Critical services are isolated and protected from external attacks

- Cloud Architecture
  - Great scalability of hardware resources
  - Cost-effective
  - No necessity of maintaining a server farm

- Publisher/Subscriver Architecture
  - Used to implement the notification system
  - Flexible with respect to further expansions

Other design decisions

CORE SYSTEM IMPLEMENTATION: JAVA EE

- Supports multi-tiered applications

- Highly reliable

- Interoperable (JAX-WS)

- Automatically manages reliable DB transactions, resource allocation and secure network communicatons

- Supports load balancing

- Mature, well known, supported technology

APPS: NATIVE FRAMEWORKS AND HTML5

- Native look & feal on each platform

- Supports platform-specific advancements

- Very small codebase

- Only implements presentation

- WebApps implemented in HTML5

- Available both on desktop and mobile

- Tailored to the specific form-factor

Integration Testing

ENTRY CRITERIA

- RASD and DD fully written and reviewed

- Data Access Utilities: 100% complete

- Taxi Management System: 90% complete

- Account Management: 70% complete

- System Administration: 70% complete

- Client Apps: 50% complete

INTEGRATION STRATEGY

- Components are integrated when they reach 90% completion

- Bottom up approach
  - Reflects the development process
  - Discover errors earlier

- In case of ties: critical-module-first approach
  - Subsystems are fairly independent from one another
  - Gives precedence to most critical functionalities

Integration of components

TAXI MANAGEMENT SYSTEM
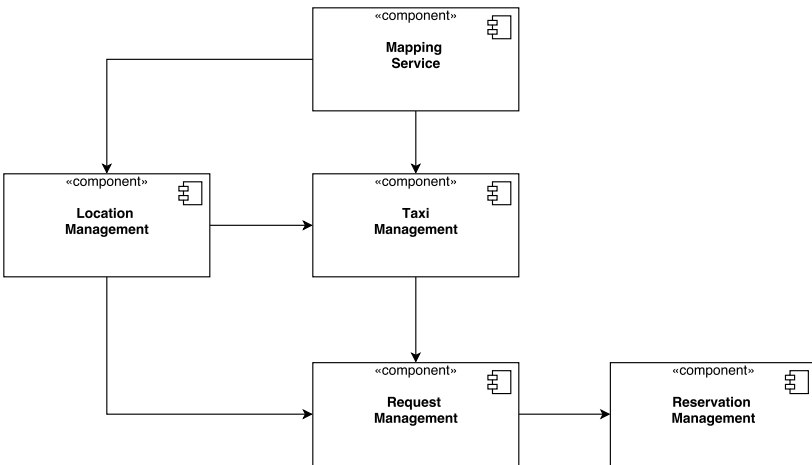
SYSTEM ADMINISTRATION
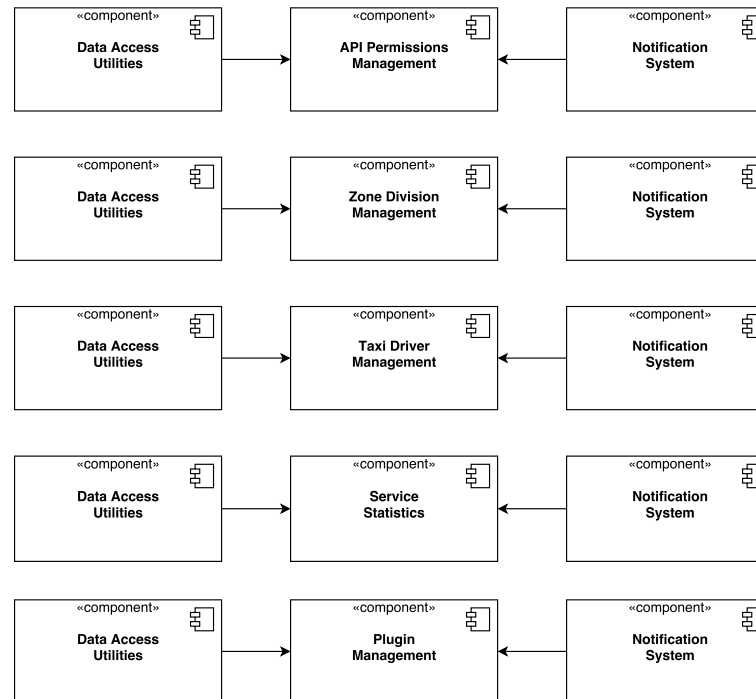
ACCOUNT MANAGEMENT
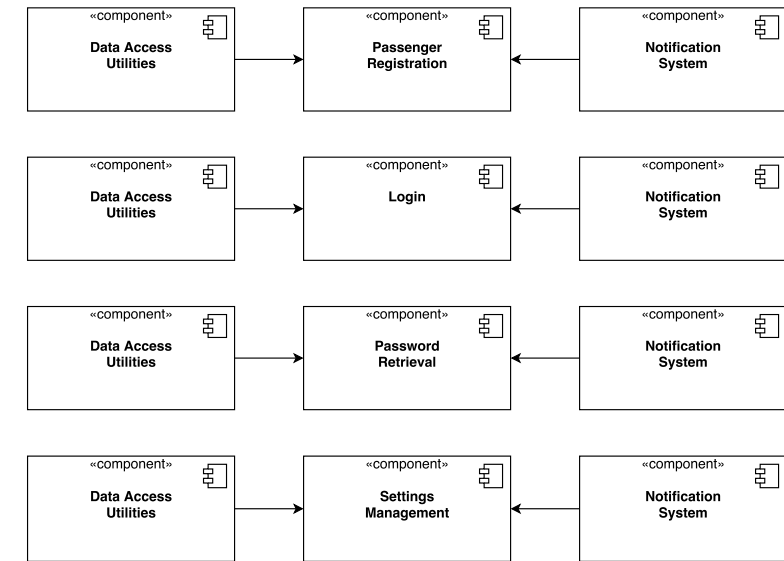
Integration of subsystems

- Bottom-up: start with components that haven't got unfinished depencencies

- To break ties: critical-module-first
  - Start with Taxi Management System
  - Then System Administration
  - Finally Account Management

- When business logic is integrated, add the Remote Services and the Notification System

- Last step: integrate the client apps
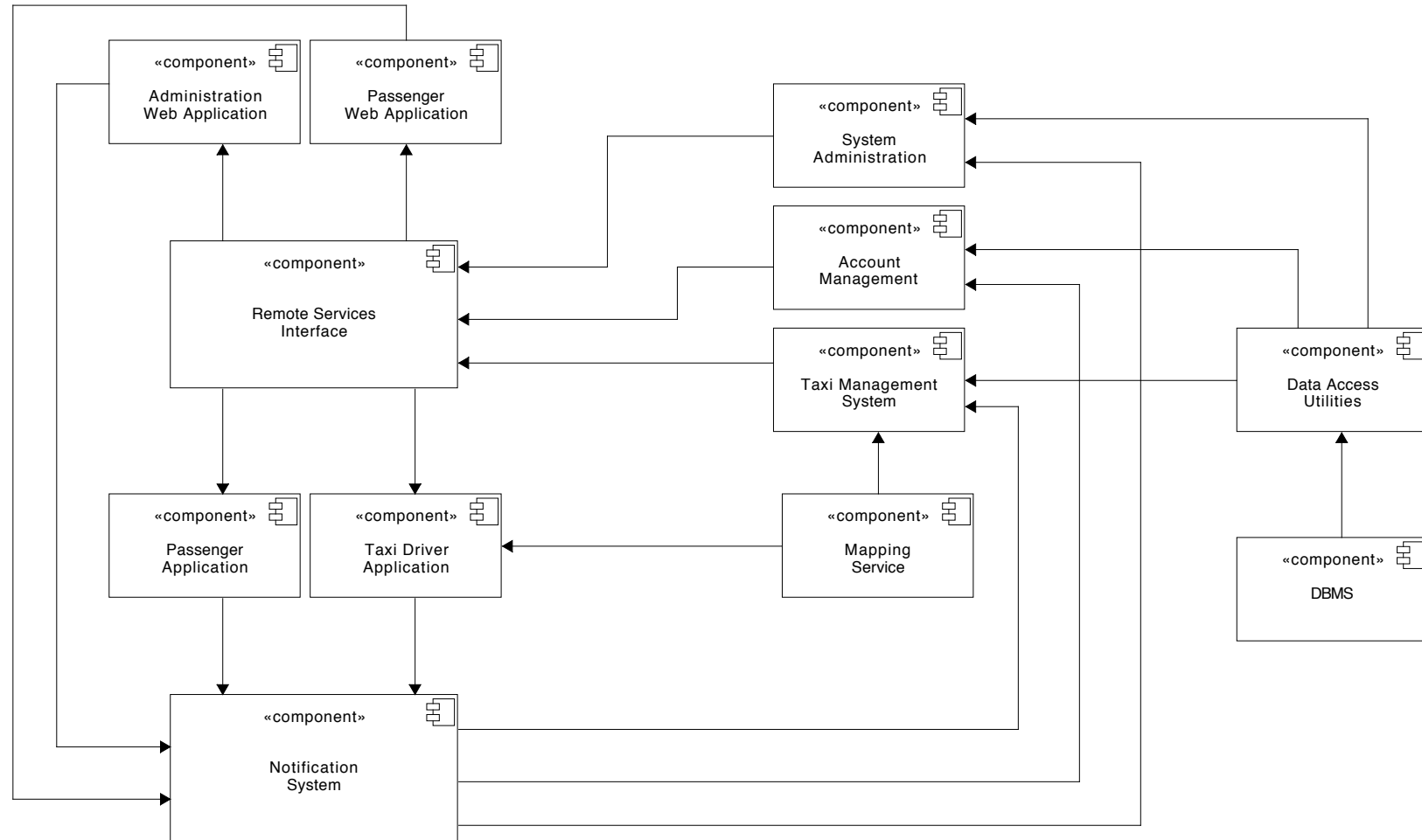
Test Equipment and Tools

TEST EQUIPMENT

- Mobile devices
  - All supported platforms (Android, iOS, Windows)
  - All supported screens (3"-6" phone, 7"-12" tablet)
  - Subject to revision depending on actual market share of the devices

- Desktop and notebook: no particular req.

- Cloud infrastructure
  - Possibly already the final one

TOOLS

- JUnit
  - Both unit testing and integration testing
  - In conjunction with Mockito

- Arquillian
  - Test how the containers interact with the beans

- Preliminary performance testing
  - Analysis tools for each mobile platform
  - Apache JMeter to test loads on both web layer and business layer

Stubs, drivers and test data

- Bottom-up approach: a driver for each component and subsystem

- Stubs to replace clients when developing the core system
  - Allow us to test the notification system without having the clients
  - Simply write the calls on a log

- Test data: stress the possible sources of anomalies
  - Taxi drivers: invalid taxi license, invalid driving license, already existing id…
  - Passengers: invalid email, address, phone number, already existing id…
  - Zones: overlapping zones, location vertices not producing a convex area, invalid or null locations..
  - Requests: location outside city
  - Reservations: source location outside city, meeting time outside the validity range
  - In general: null objects and/or fields

Function Points and COCOMO II

COCOMO II Scale Drivers:

- **Precedentedness: low**
  - We have some experience in designing software, but not at this scale and not in J2EE

- **Development flexibility: low**
  - Strict functional requirements

- **Risk resolution: very high**
  - Quite extensive analysis of the possible risks

- **Team cohesion: very high**
  - We've been working together for 3+ years

- **Process maturity: level 3 CMM**
  - First project of this kind, but we know well the tools and the environment

Function Points	Complexity	Type	FPs
Login data	Low	ILF	7
Passenger data	Low	ILF	7
Taxi drivers	Low	ILF	7
Zones	Low	ILF	7
Queues	Average	ILF	10
Reservations and requests	Low	ILF	7
API permissions	Average	ILF	10
ETA computation	Low	ELF	10
Reverse geocoding	Low	ELF	10
Map data retrieval	Low	ELF	10
Login/Logout	Low	EI	2x3
Password retrieval	Average	EI	4
Change settings	Average	EI	4
Request or reserve a taxi	High	EI	2x4
Delete a reservation	Low	EI	3
Register a new passenger account	Average	EI	4
View reservation history	Low	EI	3
Insert, delete and update zones	High	EI	3x6
Insert, delete and update taxi drivers	High	EI	3x6
Request service statistics	High	EI	6
Grant and revoke app privileges	Average	EI	2x4
Grant and revoke plugin privileges	Average	EI	2x4
Accept, refuse and end ride	High	EI	3x6
Set taxi availability	Average	EI	4
Retrieve taxi position in queue	Low	EQ	3
Retrieve passenger reservation history	Low	EQ	3
Retrieve list of taxi drivers	Low	EQ	3
Retrieve list of zones	Low	EQ	3
Retrieve list of passengers	Low	EQ	3
Retrieve list of approved applications	Low	EQ	3
Retrieve list of approved plugins	Low	EQ	3
Taxi request assignment notification	Low	EO	4
Request accepted notification	Low	EO	4
Request dropped notification	Low	EO	4
Zone changed notification	Low	EO	4
Position in the queue changed notification	Low	EO	4
Total			238

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)	Nominal	1.00
Platform volatility (PVOL)	Nominal	1.00
Analyst capability (ACAP)	High	0.85
Programmer capability (PCAP)	High	0.88
Application Experience (APEX)	Low	1.10
Platform Experience (PLEX)	Nominal	1.00
Language and Tool Experience (LTEX)	Nominal	1.00
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		1.54613

Scale Driver	Factor	Value
Precedentedness (PREC)	Low	4.96
Development flexibility (FLEX)	Low	4.05
Risk resolution (RESL)	Very high	1.41
Team cohesion (TEAM)	Very high	1.10
Process maturity (PMAT)	Level 3	3.12
Total		14.64

```
A = 2.94 (for COCOMO II)
EAF = product of all cost drivers (1.54613)
E = exponent derived from the scale drivers. It is computed a
    s:
    B + 0.01 * ∑_i SF[i] = B + 0.01 * 14.64 = 0.91 + 0.1464 =
    1.0564
      in which B is equal to: 0.91 for COCOMO II.
```

Effort and duration estimation

OPTIMISTIC SCENARIO

- Assuming a conversion rate of 46 LOC x FP
  - SLOC = 238 * 46 = 10948
  - "Average" suggested value

- Effort = A * EAF * KSLOC^E = 57 person-months
  - A = 2.94 (COCOMO II)
  - EAF = product of cost drivers = 1.54613
  - E = B+ 0.01 * scale_drivers = 1.0564

- Duration = 3.67 * Effort^F = 12.81 months
  - F = 0.28 + 0.2 ∗ (E − B) = 0.30928

PESSIMISTIC SCENARIO

- Assuming a conversion rate of 67 LOC x FP
  - SLOC = 238 * 67 = 15946
  - "High" suggested value

- Effort = A * EAF * KSLOC^E = 85 person-months
  - A = 2.94 (COCOMO II)
  - EAF = product of cost drivers = 1.54613
  - E = B+ 0.01 * scale_drivers = 1.0564

- Duration = 3.67 * Effort^F = 14.49 months
  - F = 0.28 + 0.2 ∗ (E − B) = 0.30928

Risk Analysis

POLITICAL ISSUES

- Issues with the city administration
  - Change of the city government
  - Budget crisis (spending review?)
  - Shift in local government priorities

- Issues with taxi drivers' union

- Changes in national legislation

- Countermeasures: let stakeholders have an active role in the project (meetings, reviews, demos…)

TECHNICAL ISSUES

- Changes in the map service API
  - Isolate the calls to the mapping service and make them portable

- Overestimate technical abilities
  - Try to hire good people

- Loss of source code
  - Use backups

- Changes in cloud pricing plans
  - Keep the system portable

HUMAN ISSUES

- Key members of the team leave the company or get ill
  - Make sure no single person is the sole responsible of a task

- Users refuse the app
  - Plan acceptance tests and marketing strategies, incentives

- Schedule slippage
  - Allocate extra time at the end of each phase

Extra: Code Inspection of GlassFish

- Class: com.sun.enterprise.connectors.jms.system.ActiveJmsResourceAdapter

- Methods: createManagedConnectionFactory(…), createManagedConnectionFactories(…)

- Issues:
  - Constant violation of information hiding
  - Exceptions are not properly handled in many cases (empty catch blocks, pure logging operations, never checks objects to be not-null)
  - There are several points in which the comments explicitly say the code is a patch, a hack or a certain function isn't fully implemented yet (TODOs)
  - Poor documentation (incomplete Javadoc, comments mostly missing, unexplained acronyms)
  - Not thread safe
  - Inconsistent formatting and poor choice of names (same name for constant and variable, one-letter variables and even plainly misleading names)