Politecnico di Milano

Scuola di Ingegneria Industriale e dell'Informazione

M.Sc. in Computer Science and Engineering

Authors:

Angelo Gallarello

Edoardo Longo

Giacomo Locci

Project Overview

- Requirements Analysis and Specification Document
- Design Document
- Integration Test Plan Document
- Project Plan Document
- Code Inspection Document

Requirements Analysis and Specification Document

- Goals
- Applications to be developed
- Scenarios and Use Cases
- Functional Requirements

The list of functional requirements is available in section 3.5

Goals

- **Intuitive Service:**
  - Requesting a service is intuitive for **every kind** of potential user
- **Fast:**
  - **Request** time ≤ 30s
  - **Booking** time ≤ 1.5m
- **Availability:**
  - Wherever an **internet** connection is available
- **Fair Queues:**
  - Minimise the **idle time** of every taxi driver

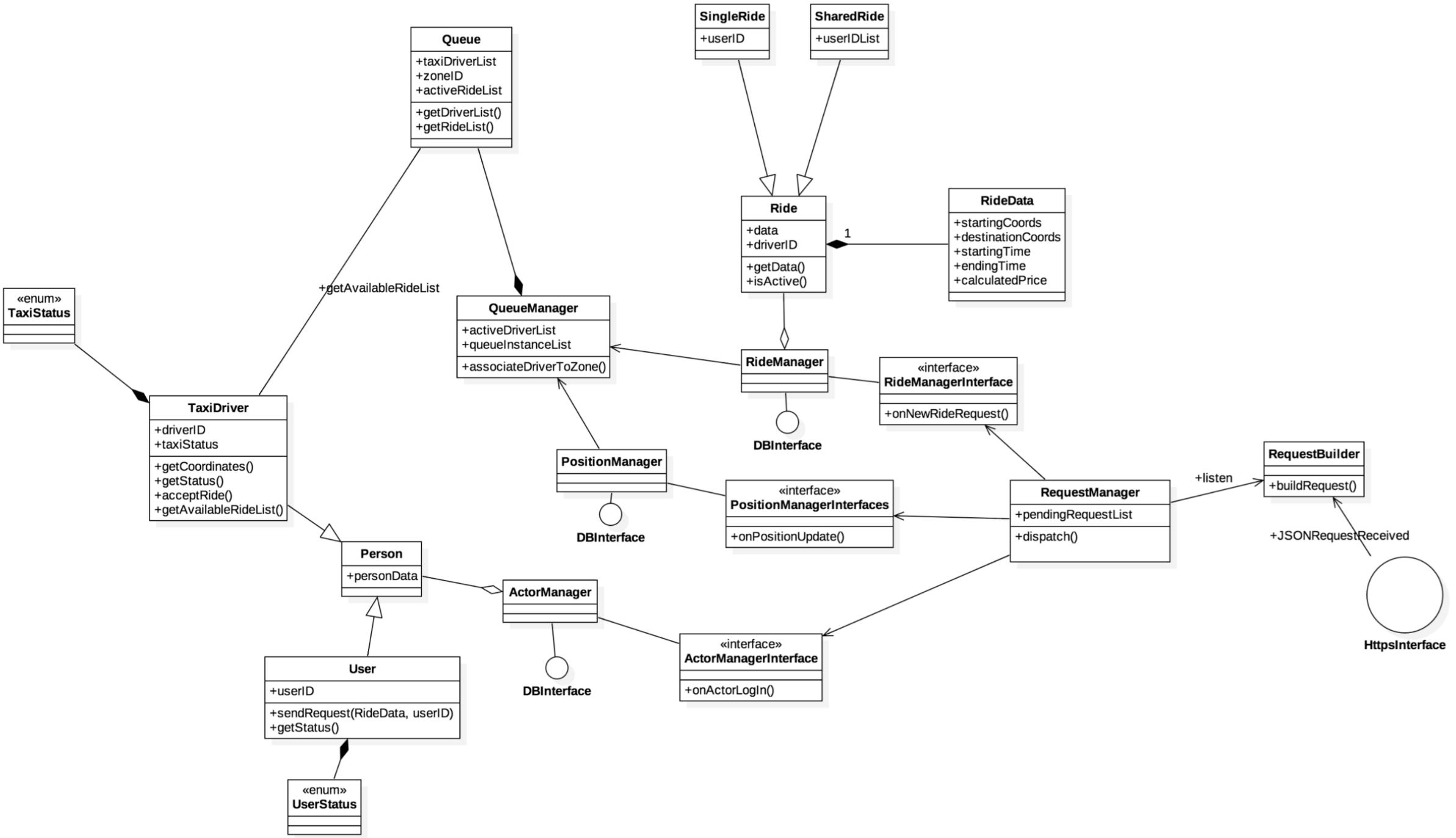
Applications

Client-Server Paradigm

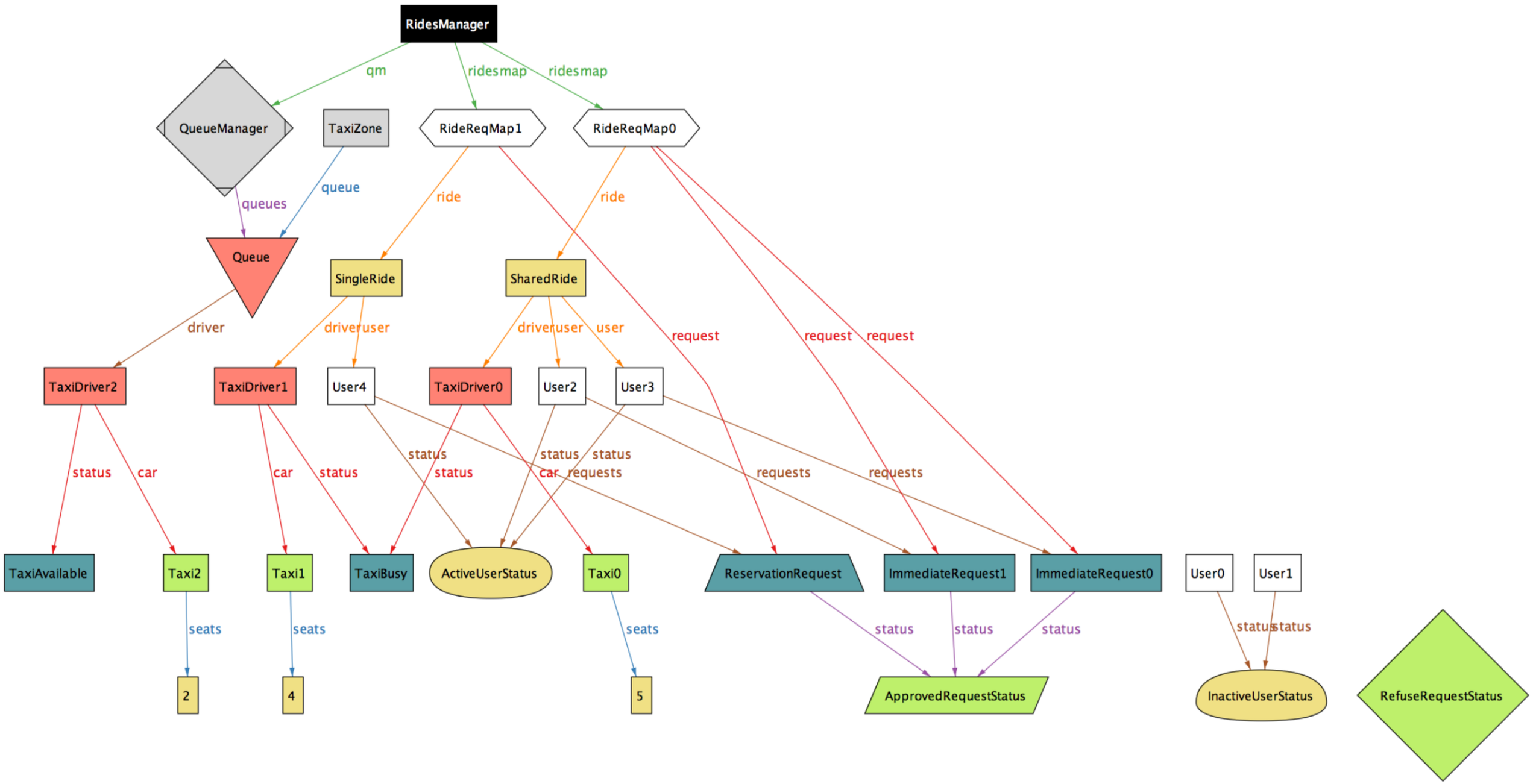
- Back End (Server)
- User Client
- Taxi Driver Client
- Web Client

Applications – Back end

- Manages:
  - Request system
  - Queue system
  - Ride system



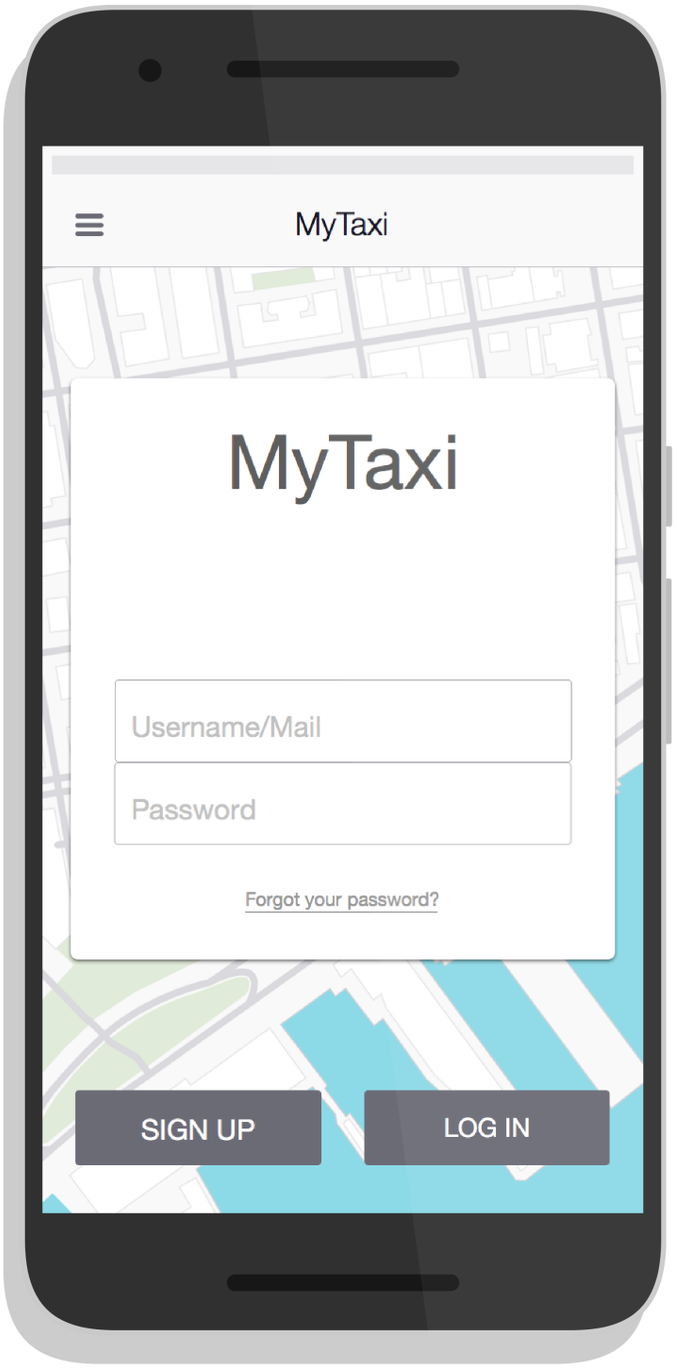
Class Diagram – Back end



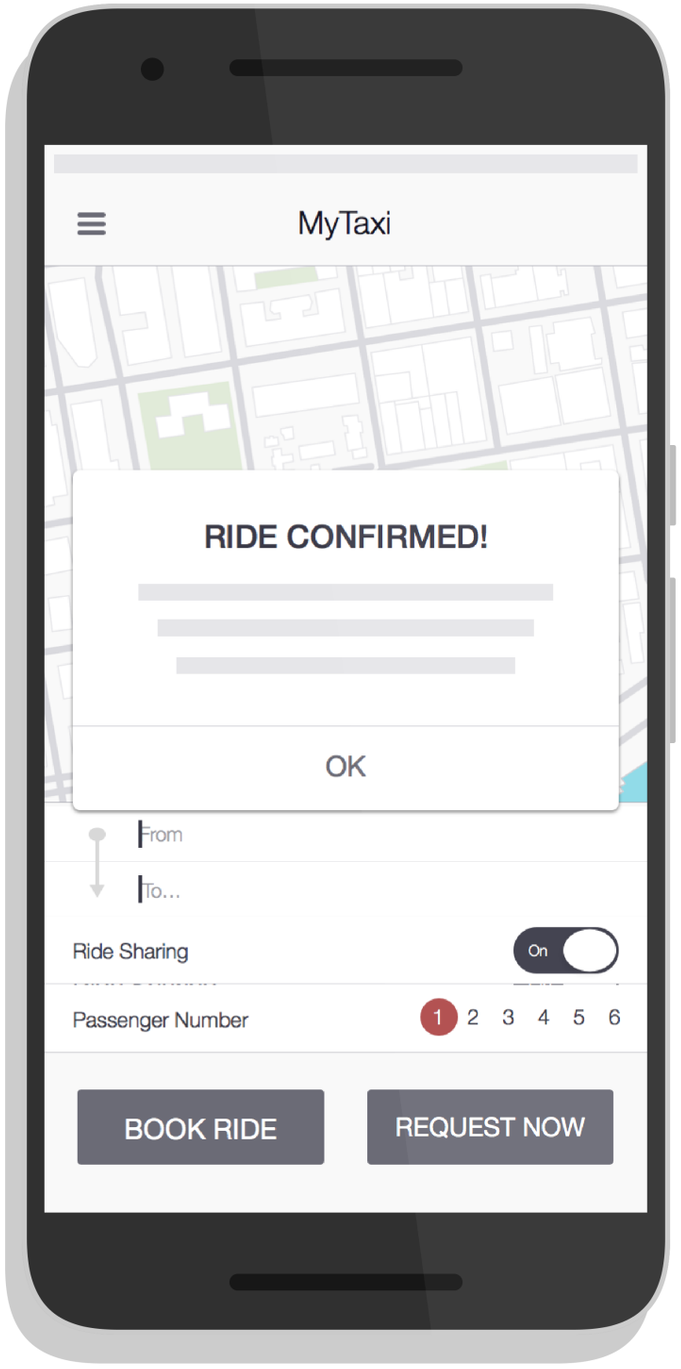
Alloy generated world

Application – User Client

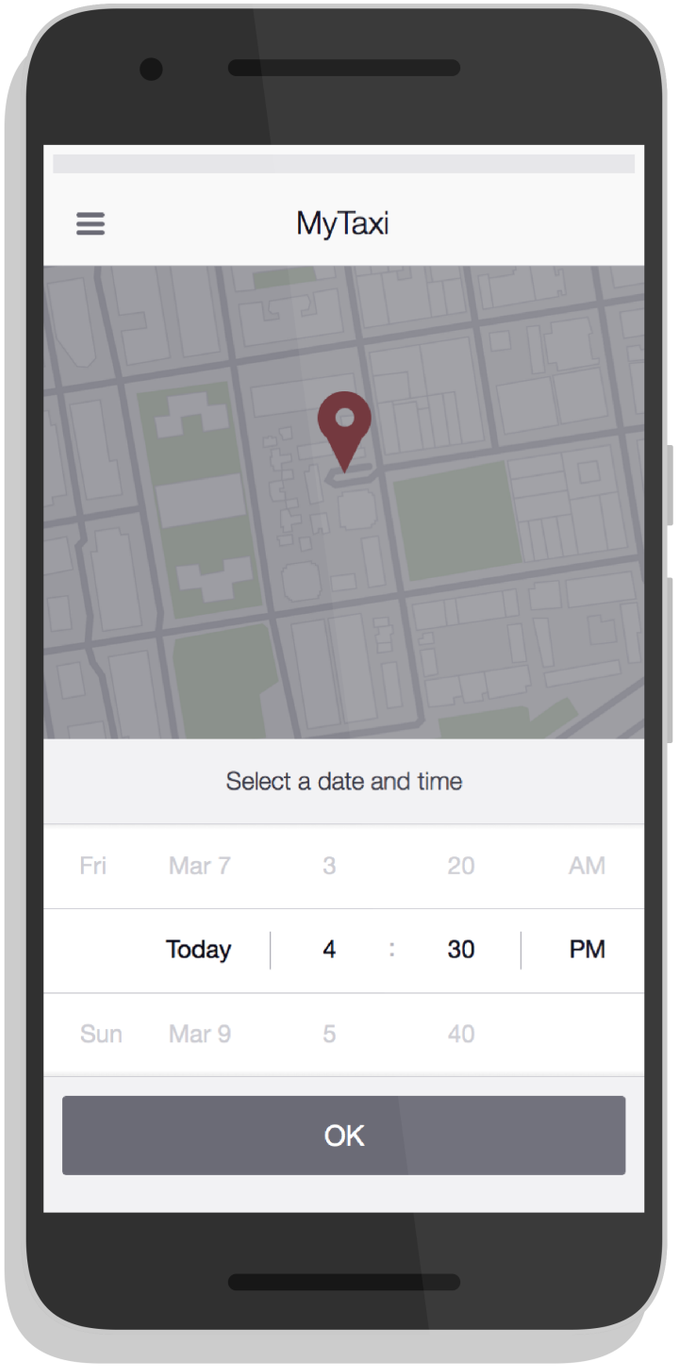
- Available for all major mobile OSs:
  - Android, iOS, Windows Mobile, Blackberry
- Functionalities:
  - Manage profile
  - Request a Taxi
  - Make / Delete reservation



Login



Request Ride

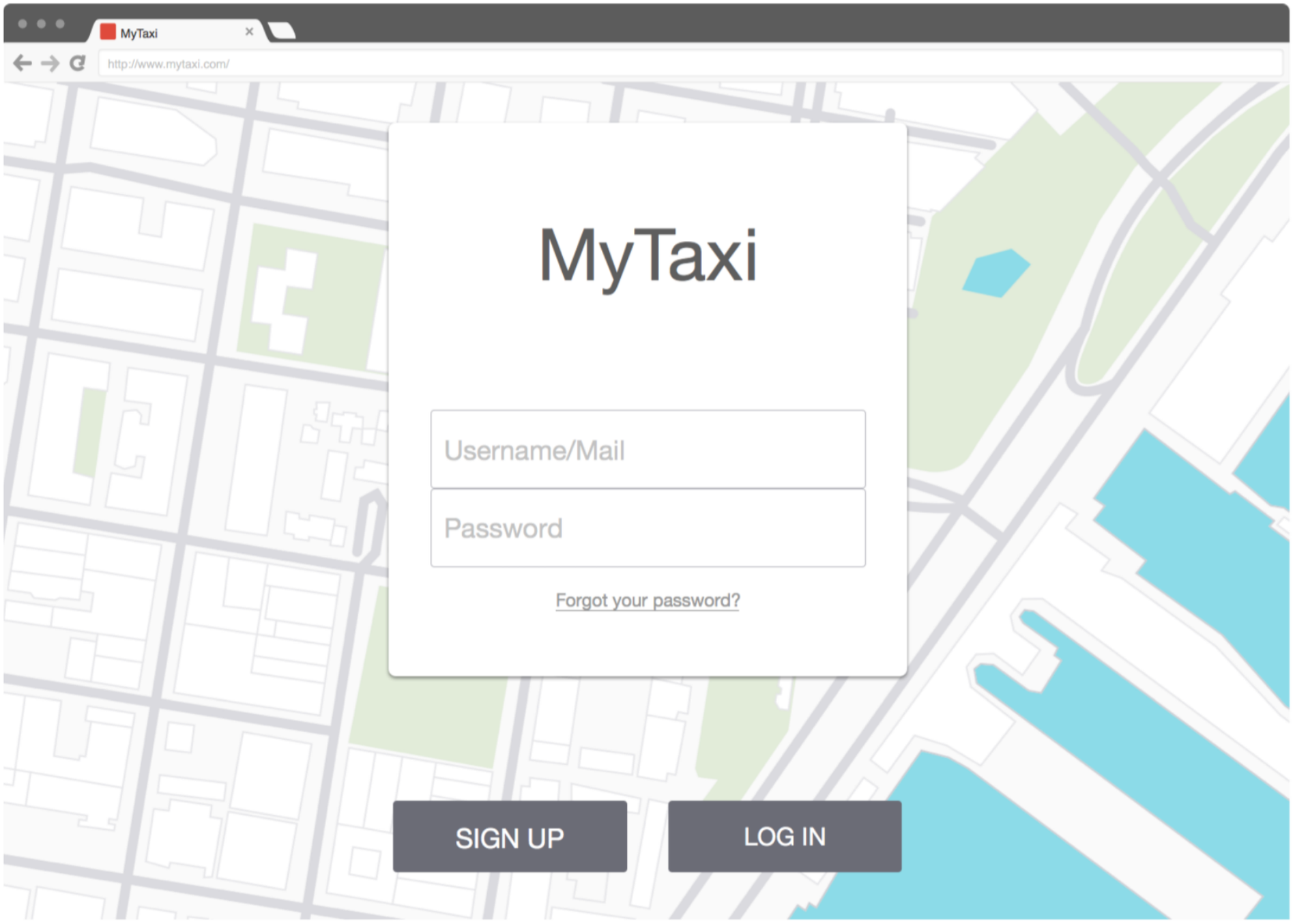


Make a reservation

(Other mockups in the RASD sec. 3.1)

Application – Web Client

- Addressed towards the User
- Accessible from any internet enabled device
- Functionalities:
  - Manage profile
  - Request a Taxi
  - Make / Delete reservation

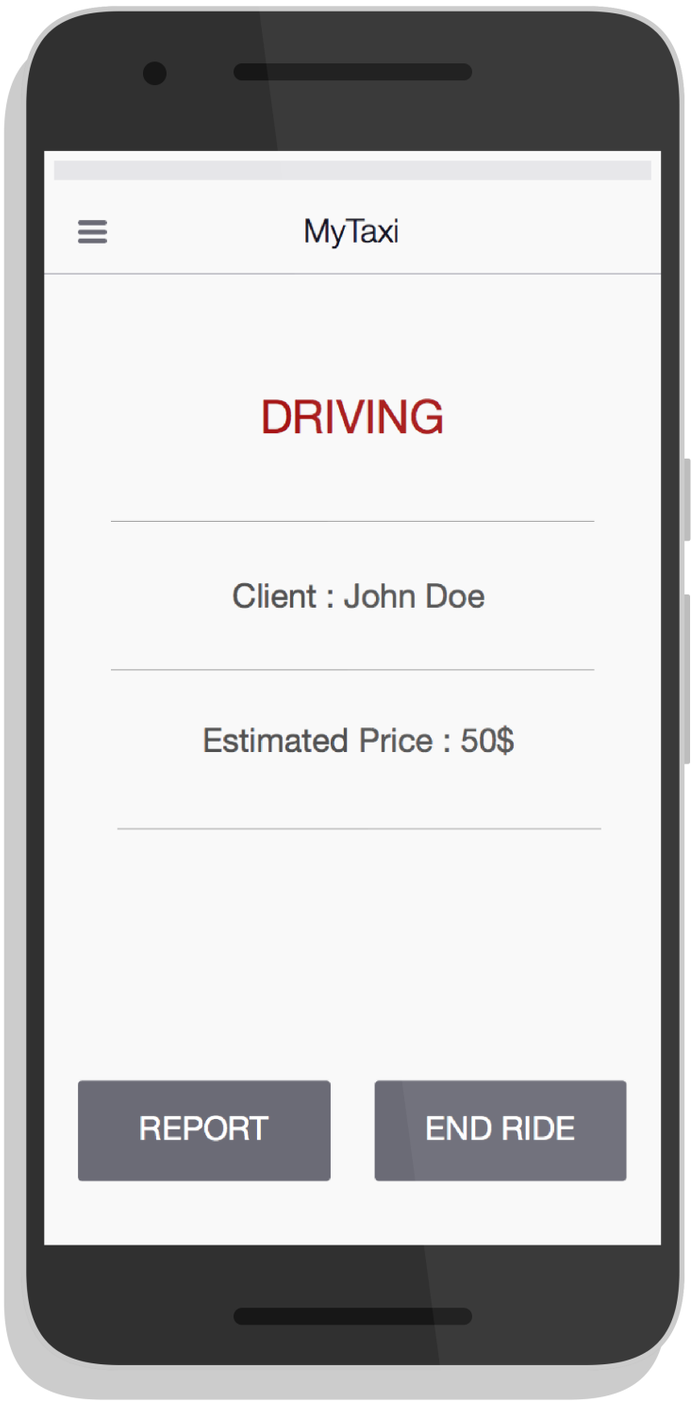


Login

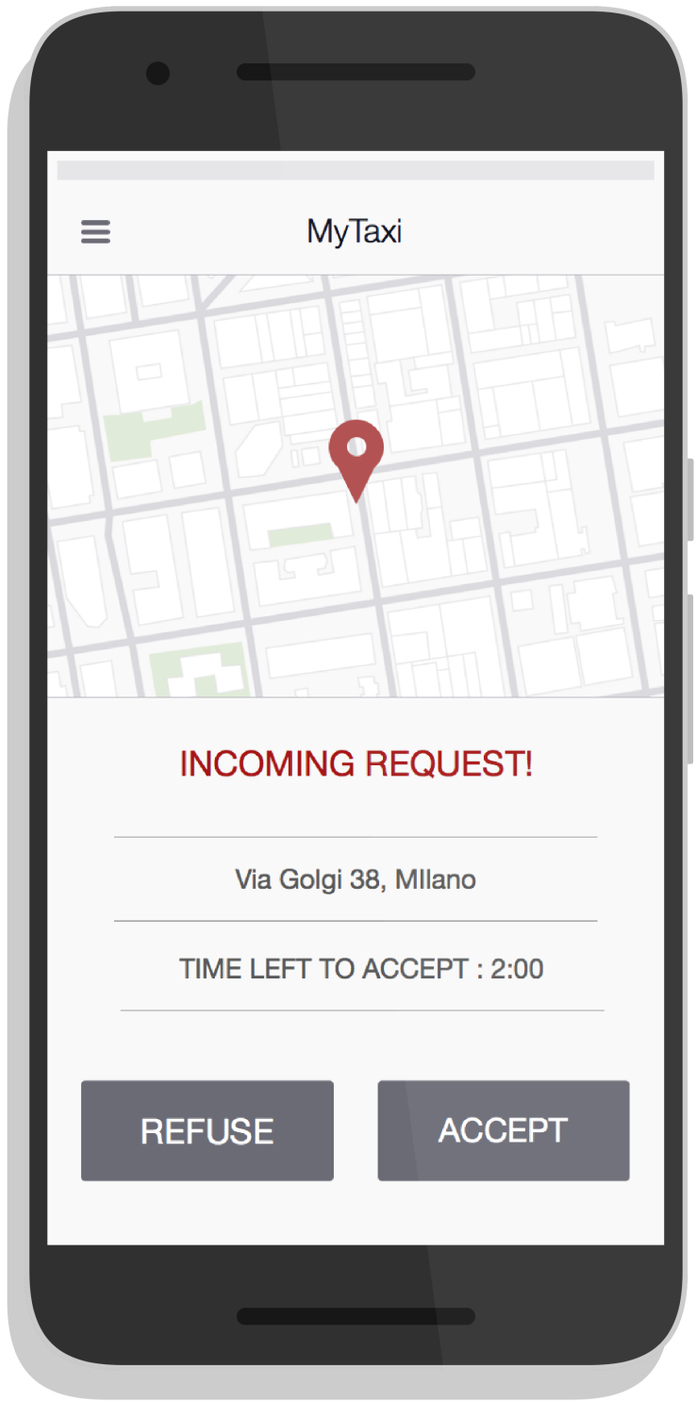
(Other mockups in the RASD sec. 3.1)

Application – Taxi Driver Client

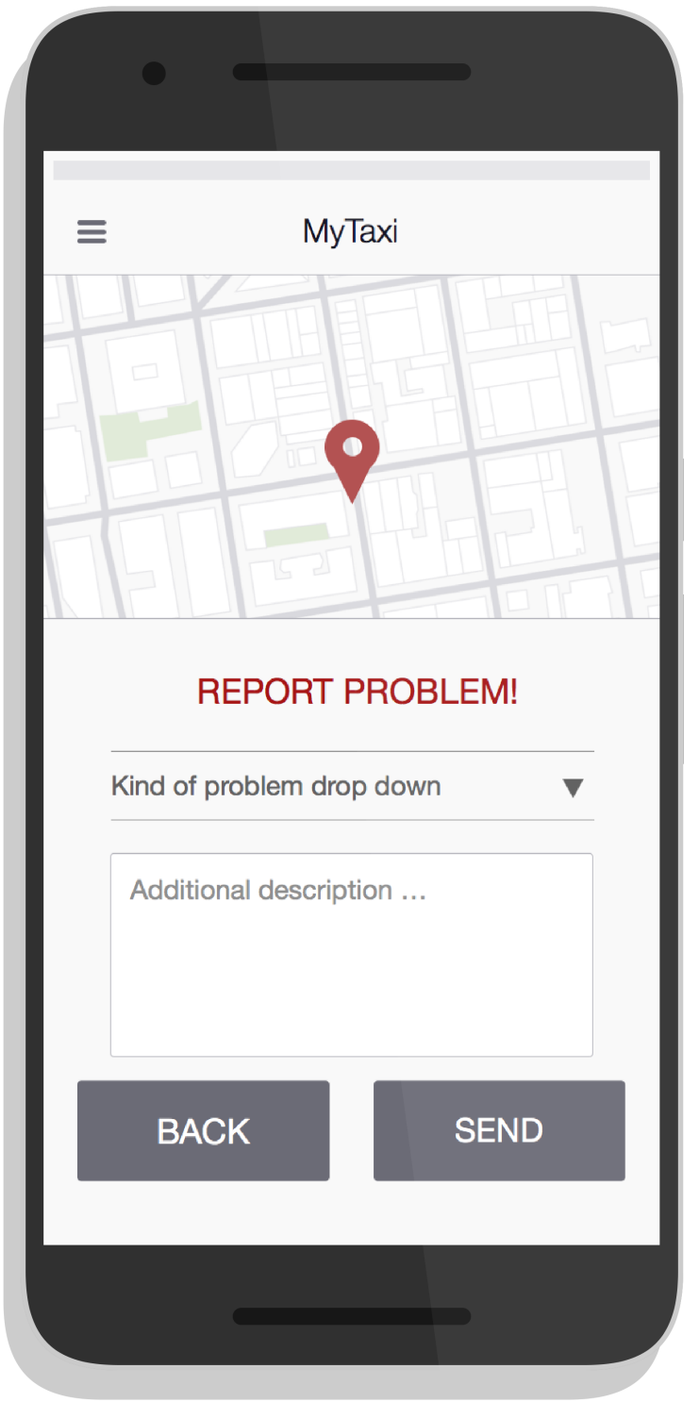
- Available for all major mobile OSs:
  - Android, iOS, Windows Mobile, Blackberry
- Functionalities:
  - Accept a request
  - Keep track of the work shifts
  - Report any problem to the operator



Work Shifts



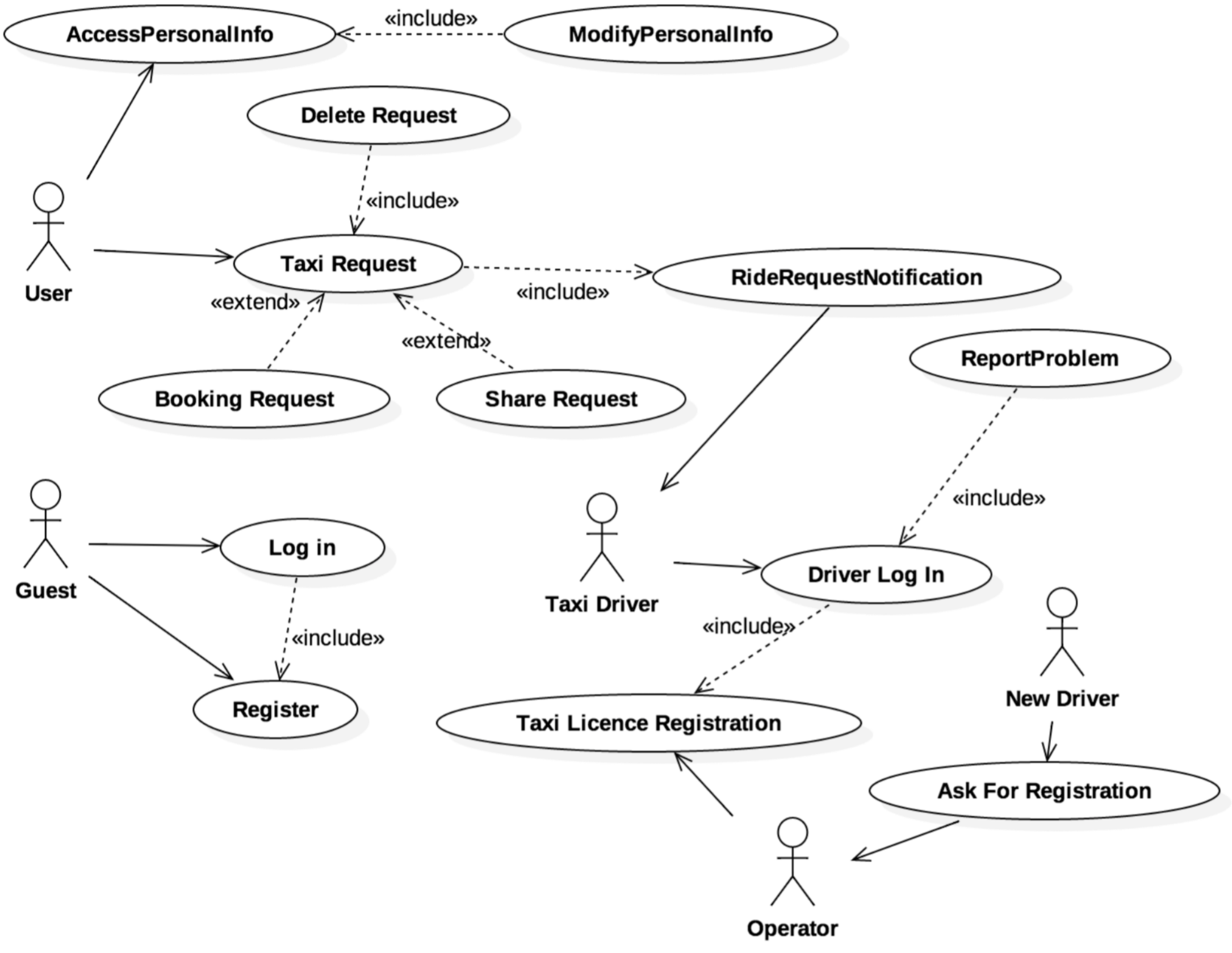
Accept Request



Report

(Other mockups in the RASD sec. 3.1)

Use cases and scenarios



Scenarios – An example
Taxi Request

Actor	User
Goal	Request a taxi
Input condition	User is correctly logged in.
Event flow	1. User press the request taxi button 2. User fill the form 3. Used decides if he want to perform an Immediate Request or an Reservation Request 4. User press confirm button 5. Backend confirm the reservation
Output condition	NULL
Exception	1. Connection unavailable 2. Data in the form are not consistent with requirements

More in sec. 3.2

Design Document

- Already discussed…

- We'll talk about the changes

Changes

- Added Overview Diagram as suggested
- Clarification on Data Caching Policy
- Specified which component implements each algorithm
- Added requirements traceability section
- Fixed paragraphing issues
- Added Changelog

Integration Test Plan Document

- Entry criteria
- Integration strategy
- Elements to be integrated
- Program stubs and test data

Entry criteria

- Design and check of Unit Tests
- Design and implementations of drivers/stub
- Design and creation of test data

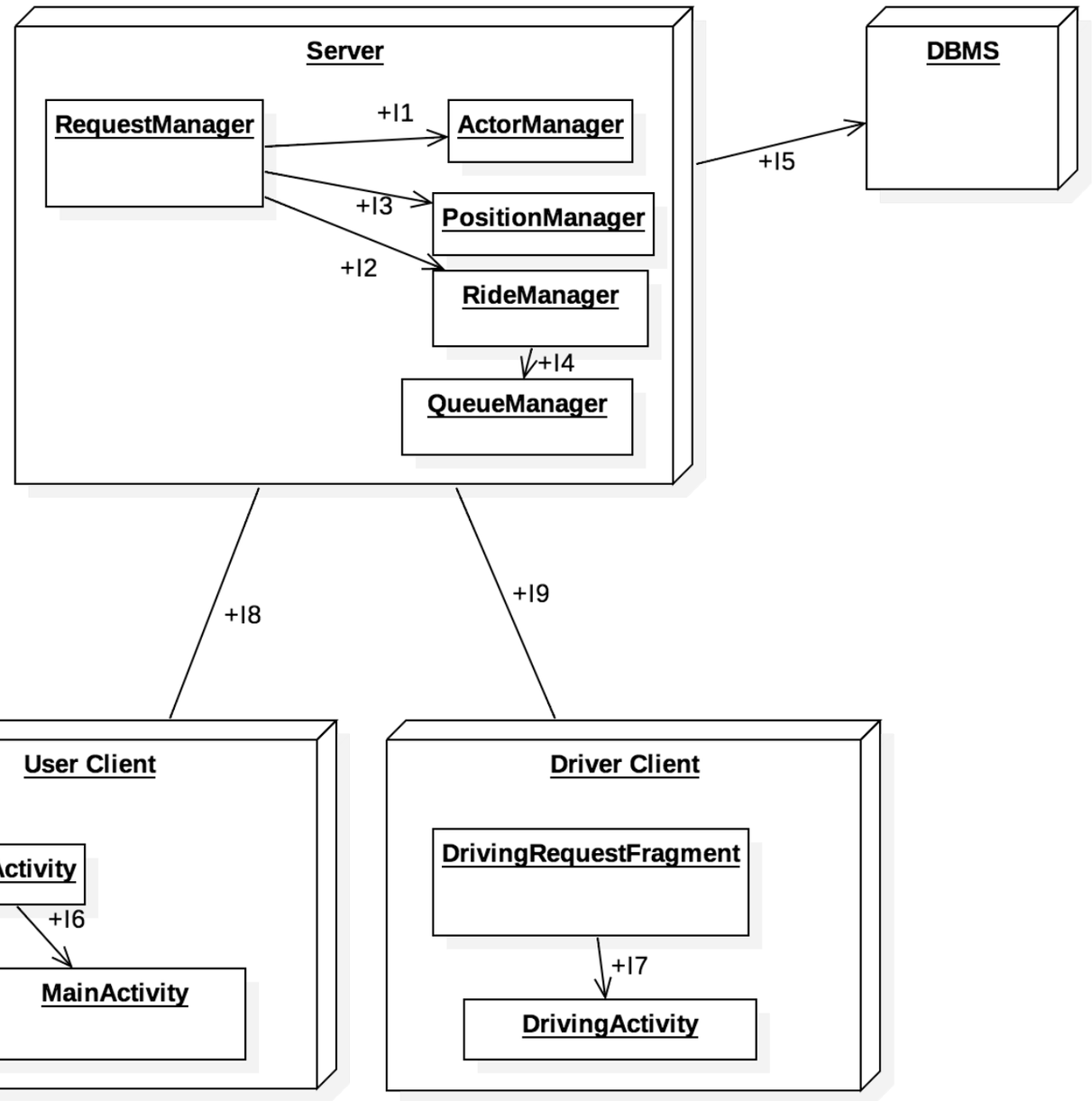
Integration strategy

- Bottom-up approach:
- Use of existing code
- Lower number of stubs/driver

Elements to be integrated

- **Server components:**
  - Queue Manager
  - Ride Manager
  - Actor Manager
  - Position Manager
  - Request Manager
  - Database Interface
- **User Client components:**
  - Main Activity
  - Signup Activity
- **Driver Client components:**
  - Driving Request Fragment
  - Driving Activity

Elements to be integrated



Tools and test equipment

- Manual testing
  - Client / Server requests
- Arquillian + JUnit
  - Integration test environment for JEE components

Project Plan Document

- Project size, effort and costs
- Tasks
- Resources allocation
- Risks

Project size, effort and costs

- Function points approach
- Cocomo II approach

Function points approach

Function	Weight	Points
External Input		
User sign-up	LOW	3
User login	LOW	3
User data update	LOW	3
User ride request	AVG.	4
User reservation request	AVG.	4
Taxi driver response	LOW	3
Taxi driver login	LOW	3
Taxi driver problem report	LOW	3
External Outputs		
Ride request response	AVG	5
Ride sharing response	AVG	5
Ride cost estimation	LOW	4
External Inquiry		
Ride sharing request	AVG	4
Internal Logical Files		
User	AVG	10
Taxi Driver	AVG	10
Shared Ride	AVG	10
Taxi Queue	AVG	10
External Logical Files		
Google Maps	AVG	7

Function Type	Points
External Input	26
External Outputs	14
External Inquiry	4
Internal Logical Files	40
External Logical Files	7
TOTAL	**91 UFP**

Estimation
for the **Android** platform

COCOMO II approach

Effort expressed in Person-Months

$$\text{Effort} = A \times SIZE^{E} \times \prod_{i} EM_i = 12.15PM$$

SIZE : KLOC of our project

E: Scale Factors

EM: Effort multipliers

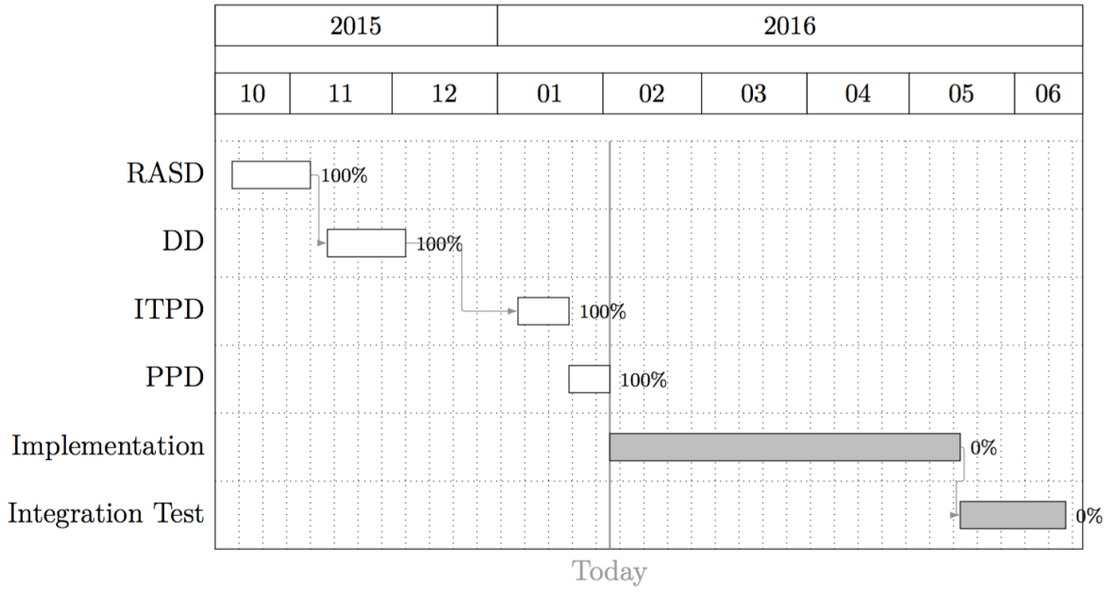
Name	**Factor**	**Value**
Precedentedness	Nominal	3.72
Development flexibility	Nominal	3.04
Risk resolution	High	2.83
Team cohesion	Very High	1.10
Process maturity	High	3.12
Total	$E = 0.91 + 0.01 \times \sum_i SF_i$	1.0481

Tasks

- The list of task to be completed in the following order.

Order	Task
1	RASD
2	DD
3	ITPD
4	PPD
5	Implementation & Unit Testing
6	Integration Testing

- Gantt Chart of the schedule



Resources allocation

DD

Subtask	**Resource**
Overview	Edoardo
High Level Components	Angelo
Component View	Angelo Giacomo
Deployment View	Angelo Edoardo
Component Interfaces	Angelo
Architectural Styles and Patterns	Angelo Giacomo Edoardo
Data Management View	Edoardo
Algorithm Design	Giacomo
User Interface View	Angelo
Requirements Traceability	Angelo Giacomo Edoardo
Appendices	Angelo Giacomo Edoardo

Risks

- **Project risks**
  - Delays
  - Communication issues
  - Lack of experience
- **Technical risks**
  - System fail
  - Data loss / Data leaks
- **Economical risks**
  - Competitors

Code Inspection Document

- Functional role of assigned classes
- Issues of assigned methods
- Other class issues
- Tools used

Functional role of assigned classes

- EntityContainer class :
  - Implements the container interface for BMP and CMP
  - Manage their instance and lifecycle
    - Pooled, Ready, Invoking, Incomplete_Tx, Destroyed

Issues of assigned methods

- Single char name used for a non throwaway variable

- Missing Javadoc

- Equals not used for object comparison

- Wrong indentation or formatting

- Not auto-explicative method naming

Other class issues

- Wrong indentation
- Excessive line length
- Brackets position / Missing brackets
- Tab character
- Naming patterns

Tool used

- Checklist provided
- Manual checking for methods
- *Checkstyle* tool for class issues

Any questions ?