Project Plan
"MyTaxiService" Application
A.Y. 2015 - 2016

Ennio Visconti (mat. 790382)
Simone Zocchi (mat. 852910)
Khanh Huy Paolo Tran (mat. 852496)

February 2, 2016

POLITECNICO
MILANO 1863

Contents

1 Introduction

1.1 Revision History

Version	Date	Authors	Summary
1.0	02/02/2016	Ennio Visconti, Simone Zocchi, Khanh Huy Paolo Tran	Document creation.

1.2 Purpose and Scope

This document contains a general analysis of the project made using Function Points and COCOMO approaches to estimate its size and costs.
According to the results found, tasks for the development of the application are identified and associated to the available resources. Finally the main risks for the project are identified and recovery actions are analyzed.

1.3 Definitions and Abbreviations

- FP: Function Points
- UFP: Unadjusted Function Points
- COCOMO: COnstructive COst MOdel
- ILF: Internal Logical Files
- EIF: External Interface Files
- SF: Scaling Factor
- EM: Effort Multiplier
- EPML: Estimated Process Maturity Level

1.4 Reference Document

- Software Engineering 2 Project, AA 2015-2016 Assignments 5 – Project plan
- Requirements Analysis & Specification Document - "myTaxiService" Application, AY 2015-2016; Ennio Visconti, Simone Zocchi, Khanh Huy Paolo Tran
- Design Document - "myTaxiService" Application, AY 2015-2016; Ennio Visconti, Simone Zocchi, Khanh Huy Paolo Tran
- *COCOMO II Model Definition Manual*

2 Project size estimation - Function Points

In this section the project size will be evaluated in terms of unadjusted function points. Given the following table summarizing the complexity weights of various function types, a detailed analysis will follow in the upcoming paragraphs.

Table 3. UFP Complexity Weights

Function Type	Complexity-Weight		
	Low	Average	High
Internal Logical Files	7	10	15
External Interfaces Files	5	7	10
External Inputs	3	4	6
External Outputs	4	5	7
External Inquiries	3	4	6

Figure 1: Weights values

2.1 INPUT

Passenger

- **Login/Logout**: These operations con be considered with a *Low* weight as they involve only one entity. 2 x 3 = **6 FPs**
- **Registration**: This operation involves a few more input data but still involves one entity, so it can be considered as *Low* weight. 1 x 3 = **3 FPs**
- **Ride request / reservation**: These are quite complex, in fact they require some data from the user and the interaction with other entities. So consider these as *High* weight. 2 x 6 = **12 FPs**

Taxi Driver

- **Accept / Decline request**: This operation can be considered of *Average* weight, as it requires only an interaction from the user, but in involves two other entities, request and passenger. 2 x 4 = **8 FPs**
- **Start / Stop ride**: Again this operation requires a single interaction from user, but involves other two entities, ride and Passenger. So we adopt an *Average* weight. 2 x 4 = **8 FPs**

2.2 OUTPUT

- **Total fare of a ride**: This operation requires information from three different entities, ride, driver and passenger, it can be considered of *High* weight. 1 x 7 = **7 FPs**
- **Notification for accepted ride**: This operation requires information from two different entities, driver and passenger, it can be considered of *Average* weight. 1 x 5 = **5 FPs**
- **Notification for ride request**: This operation requires information from three different entities, request, driver and passenger, it can be considered of *High* weight. 1 x 7 = **7 FPs**

2.3 INQUIRY

Passenger

- **Rides Log**: This operation requires data from Passenger and Rides. It is a quite easy operation so can be considered of *Low* weight. 1 x 3 = **3 FPs**

Taxi Driver

- **Notifications**: This operation requires data Requests and Driver entities but still remains an easy operation, so can be considered of *Low* weight. 1 x 3 = **3 FPs**
- **Passengers on board**: This operation requires information Passengers, Rides and Driver entities so can be considered of *High* weight. 1 x 6 = **6 FPs**

2.4 ILF

The following entities will be used to store information needed for the application: **Passengers**, **Taxis**, **Requests**, **Rides** and **Drivers**. They all have a simple structure, except for the Request entity that is a little bit more complicated. So we assigned a *Low* weight to all the entities, except for Request that has an *Average* weight. 4 x 7 + 1 x 10 = **38 FPs**

2.5 EIF

- **GPS location**: This file contains the GPS coordinates of the vehicle. Can be considered with a simple structure so we assigned a *Low* weight. 1 x 5 = **5 FPs**
- **Payment data**: This file contains data about the payment. It is quite simple so we assigned a *Low* weight. 1 x 5 = **5 FPs**

- **Path data**: This is a complex structure acquired from the Google Maps API, we assigned an *High* weight. 1 x 10 = **10 FPs**

2.6 Summary

Function Type	**Total FP**
Input	6 + 3 + 12 + 8 + 8 = 37 FPs
Output	7 + 5 + 7 = 19 FPs
Inquiry	3 + 3 + 6 = 12 FPs
ILF	38 FPs
EIF	5 + 5 + 10 = 20 FPs
Total	37 + 19 + 12 + 38 + 20 = **126 FPs**

3 Effort and Cost estimation - COCOMO

In this section effort and costs of the project will be evaluated, in terms of month-person units. A detailed analysis of the scale and cost drivers will follow in the upcoming paragraphs, as defined in the COCOMO II estimation standard.

3.1 Scale Drivers

- **Precedentedness** [**PREC**]: This is the first Enterprise project developed by our team. However the team members are quite familiar with the Java language and basic language constructs and patterns. The level is therefore **low**.
- **Development Flexibility** [**FLEX**]: The customer only set general goals, without restricting the flow to specific processes. However, at an high level, the taxi reservation process is quite standardized, which therefore means the level is **high**.
- **Architecture/risk resolution** [**RESL**]: The architecture/risk resolution requires a more in depth analysis of some specific points, in particular:
  - The Risk Management Plan generally identifies critical risk items and establishes milestones for resolving them. Rating Level: Nominal.
  - Schedule and budget have been set, but at some level of abstraction. Rating Level: Nominal.
  - The percent of development schedule spent on the architecture is at a Nominal rating.
  - Percent of required top software architects available: High.
  - Tool support available to resolve risk items: Nominal.
  - Level of uncertainty in key architecture drivers: High
  - Number and criticality of risk items: 2-4 =¿ Nominal.
    Overall rating: **High**
- **Team cohesion** [**TEAM**]: The development team works really well and in harmony. Developers knew each other years before starting the project and there are practically no communication problems between them. So the level is **extra high**.
- **Process maturity** [**PMAT**]: Given the calculation KPAs, by doing an arithmetic approximation, the result of the EPML is 3: **High**.

3.2 Cost Drivers

1. **Required Software Reliability [RELY]**: Since the software simply replaces a manual procedure, no particular losses would be generated by a system failure. Rating Level: **Low**.

2. **Data Base Size [DATA]**: Test data is considered to don't be particularly huge. Rating Level: **Nominal**.

3. **Product Complexity [CPLX]**: The product complexity can be divided in some more specific topics. An overview of the specific ratings is presented below:

   - Control Operations: High.
   - Computational Operations: High.
   - Device-dependent Operations: Nominal.
   - Data Management Operations: High.
   - User Interface Management Operations: Nominal.

   So the average rating level is **High**.

4. **Developed for Reusability [RUSE]**: The project development will be done with the idea of reusable components to the level of the project (there are no plans of other programs to develop which will interact with it). Rating level: **Nominal**.

5. **Documentation Match to Life-Cycle Needs [DOCU]**: The most life-cycle needs are covered by previous documents. However some parts still need some more documentation. Rating Level: **Low**.

6. **Execution Time Constraint [TIME]**: There are no particular constraints about the execution times, but still the application must respond quickly to user inputs. Rating Level: **High**.

7. **Main Storage Constraint [STOR]**: Since only the database must be stored, a minimum part of the available storage will be used. Rating Level: **Nominal**.

8. **Platform Volatility [PVOL]**: Most of the technologies are really stable and there is no perception of significant changes in less than one year. Rating Level: **Low**.

9. **Analyst Capability [ACAP]**: The analysts are quite efficient in extracting the needed design and in cooperating to find the best architectural solutions. Rating Level: **High**.

10. **Programmer Capability** [**PCAP**]: The programmers as a team ave a really effective communication and can cooperate really well. Rating Level: **Very High**.

11. **Personnel Continuity** [**PCON**]: There are no information about the statistical personnel turnover. Rating Level: **Nominal**.

12. **Application Experience** [**APEX**]: The team is composed by only beginners software engineers. Therefore there is practically no experience in the development of this kind of products at an enterprise level. Rating Level **Very Low**.

13. **Platform Experience** [**PLEX**]: All team members have quite a basic comprehension and few experience of the chosen platform. Rating Level: **Low**.

14. **Language and Tool Experience** [**LTEX**]: All team members are quite experienced with the Java language and the common tools usually used for developing programs in that context. Rating Level: **High**.

15. **Use of Software Tools** [**TOOL**]: Given the semi-mature definition of the application life cycle, and the choice of a continuous integration tool, an effective tool set has been defined. Rating Level: **High**.

16. **Multisite Development** [**SITE**]: The development is fully collocated. Rating Level: **Extra High**

17. **Required Development Schedule** [**SCED**]: The development schedule is properly set without any particular stretch-outs or compressions. Rating Level: **Nominal**.

3.3 Summary

Given the effort (ϵ) equation, which is:

$$\epsilon = A \times Size^E \times EAF \tag{1}$$

Where:

- A = 2.94 as in COCOMO II.2000
- Size: actual size of the product to be developed in terms of KSLOC (thousands of Source Lines Of Code).

$$Size = \frac{UFP \times LF}{1000} \tag{2}$$

Where LF = 46 SLOCs as for the avarage value of J2EE applications.

- E : Exponent derived from the five Scale Drivers, with

$$E = B + 0.01 \times \sum_{j=1}^{5} SF_j \tag{3}$$

With B = 0.91 as in COCOMO II.2000

- EAF : Effort Adjustment Factor derived from the Cost Drivers, with

$$EAF = \prod_{i=1}^{N} EM_i \tag{4}$$

Scale Factors	**Rating Level**	**SF Value**
PREC	Low	4.96
FLEX	High	2.03
RESL	High	2.83
TEAM	Extra High	0.00
PMAT	Nominal	4.68
TOTAL (Sum)		**14.5**

Table 3: Scale Drivers Recap

Cost Drivers	**Rating Level**	**Effort Multiplier**
RELY	Low	0.92
DATA	Nominal	1.00
CPLX	High	1.17
RUSE	Nominal	1.00
DOCU	Low	0.91
TIME	High	1.11

STOR	Nominal	1.00
PVOL	Low	0.87
ACAP	High	0.85
PCAP	Very High	0.76
PCON	Nominal	1.00
APEX	Very Low	1.22
PLEX	Low	1.09
LTEX	High	0.91
TOOL	High	0.90
SITE	Extra High	0.80
SCED	Nominal	1.00
TOTAL (Product)		**0.53**

Table 4: Cost Drivers Recap

Therefore the estimated effort is:

$$\epsilon = 2.94 \times (\frac{126 \times 46}{1000})^{0.91+0.01\times14.5} \times 0.53 = \mathbf{9.95\ person\text{-}month} \quad (5)$$

with an estimated duration of:

$$\delta = C \times \epsilon^{D+0.2\times(E-B)} = 3.67 \times 9.95^{0.28+0.2\times0.01\times14.5} = \mathbf{7.5\ months} \quad (6)$$

Note: C = 3.67 and D = 0.28 as for COCOMO II.2000

And a (minimum) required number of people of:

$$\pi = \lceil \epsilon/\delta \rceil = \mathbf{2\ persons} \quad (7)$$

4 Tasks identification

In this section the main tasks of the project will be described with related duration and dependencies. Tasks are divided in terms of the activity (or super-task) they are related to and some milestones, described with deliveries versioning) are set.
Note: versions are also tracked as "releases" on the GitHub repository.

4.1 Requirement Analysis & Specification

	Task	**Duration (days)**	**Dependencies**
1.	Scenarios identification	5	
2.	Goals collection	10	1
3.	Constraints analysis	10	1
4.	Interface requirements collection	7	2,3
5.	Functional requirements collection	7	2,3
6.	Use cases writing	4	1
7.	Alloy verification	4	5
v.0.1	**Milestone: RASD**		

Table 5: Requirements Analysis & Specification Tasks

4.2 Design & Architecture Development

	Task	**Duration (days)**	**Dependencies**
8.	Main blocks design and definition	2	
9.	Front end components definition	7	8
10.	Back end components definition	7	8
11.	Architectural styles and design patterns selection	5	9,10
12.	Components interfaces and Public API design	7	10
13.	Requests handling analysis	5	11
14.	Request arrival and ride notification analysis	5	11
15.	Fare algorithm design	4	

16.	Shared request algorithm design	8	13
v.0.2	**Milestone: DD**		

Table 6: Design & Architecture Development Tasks

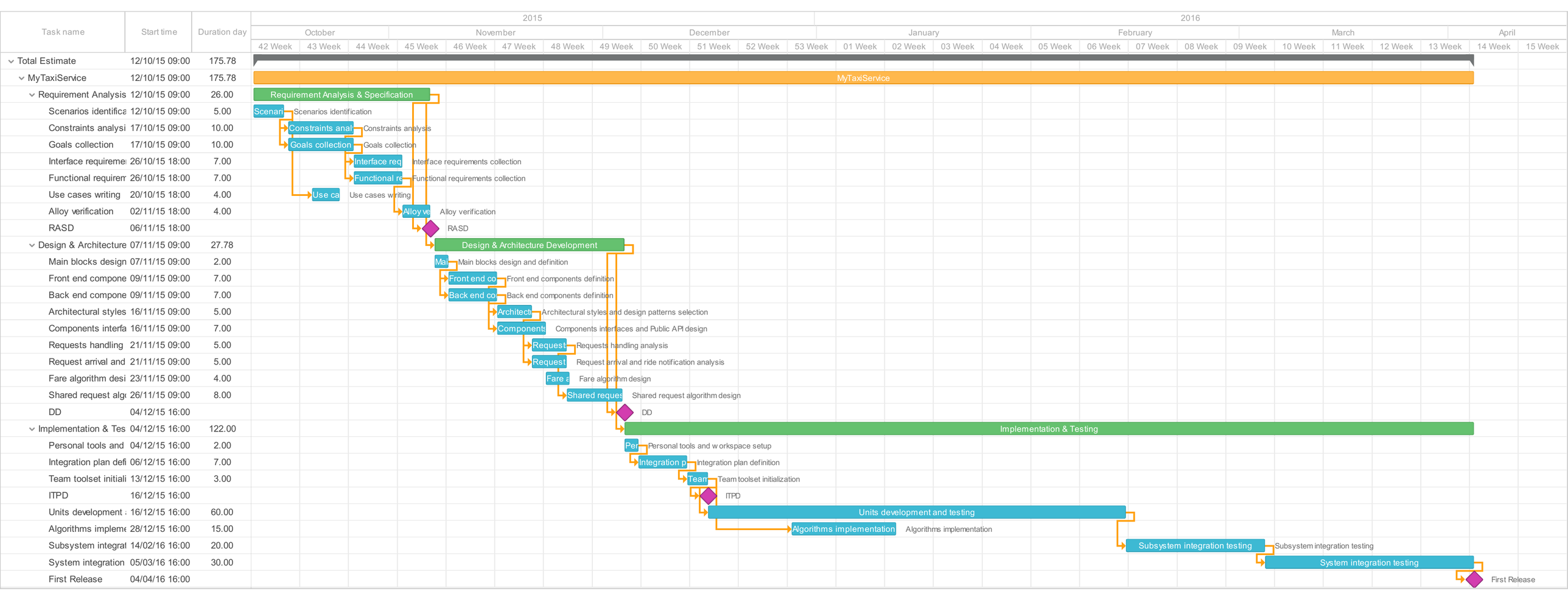
4.3 Implementation & Testing

	Task	**Duration (days)**	**Dependencies**
17.	Personal tools and workspace setup	2	
18.	Integration plan definition	7	17
19.	Team toolset initialization	3	17,18
v.0.3	**Milestone: ITPD**		
20.	Units development and testing	60	19
21.	Algorithms implementation	15	19
22.	Subsystem integration testing	20	20
23.	System integration testing	30	22
v.1.0	**Milestone: First Release**		

Table 7: Implementation & Testing Tasks

4.4 Project Scheduling

The Gantt chart with the scheduling of the project is available at the next page, or at the following link: *Gantt chart*


Task name
Start time
Duration day
2015
2016
October
November
December
January
February
March
April
42 Week
43 Week
44 Week
45 Week
46 Week
47 Week
48 Week
49 Week
50 Week
51 Week
52 Week
53 Week
01 Week
02 Week
03 Week
04 Week
05 Week
06 Week
07 Week
08 Week
09 Week
10 Week
11 Week
12 Week
13 Week
14 Week
15 Week
Total Estimate 12/10/15 09:00 175.78
MyTaxiService 12/10/15 09:00 175.78
Requirement Analysis 12/10/15 09:00 26.00
Scenarios identifica 12/10/15 09:00 5.00
Constraints analysi 17/10/15 09:00 10.00
Goals collection 17/10/15 09:00 10.00
Interface requireme 26/10/15 18:00 7.00
Functional requirem 26/10/15 18:00 7.00
Use cases writing 20/10/15 18:00 4.00
Alloy verification 02/11/15 18:00 4.00
RASD 06/11/15 18:00
Design & Architecture 07/11/15 09:00 27.78
Main blocks design 07/11/15 09:00 2.00
Front end compone 09/11/15 09:00 7.00
Back end compone 09/11/15 09:00 7.00
Architectural styles 16/11/15 09:00 5.00
Components interfa 16/11/15 09:00 7.00
Requests handling 21/11/15 09:00 5.00
Request arrival and 21/11/15 09:00 5.00
Fare algorithm desi 23/11/15 09:00 4.00
Shared request alg 26/11/15 09:00 8.00
DD 04/12/15 16:00
Implementation & Tes 04/12/15 16:00 122.00
Personal tools and 04/12/15 16:00 2.00
Integration plan defi 06/12/15 16:00 7.00
Team toolset initiali 13/12/15 16:00 3.00
ITPD 16/12/15 16:00
Units development 16/12/15 16:00 60.00
Algorithms impleme 28/12/15 16:00 15.00
Subsystem integrat 14/02/16 16:00 20.00
System integration 05/03/16 16:00 30.00
First Release 04/04/16 16:00
MyTaxiService
Requirement Analysis & Specification
Scenarios identification
Constraints analysis
Goals collection
Interface requirements collection
Functional requirements collection
Use cases writing
Alloy verification
RASD
Design & Architecture Development
Main blocks design and definition
Front end components definition
Back end components definition
Architectural styles and design patterns selection
Components interfaces and Public API design
Requests handling analysis
Request arrival and ride notification analysis
Fare algorithm design
Shared request algorithm design
DD
Implementation & Testing
Personal tools and w orkspace setup
Integration plan definition
Team toolset initialization
ITPD
Units development and testing
Algorithms implementation
Algorithms implementation
Subsystem integration testing
Subsystem integration testing
System integration testing
First Release

5 Resource allocation

In this section all the tasks are assigned to the members of the group. Assignments has been done taking into account both personal skills and time availability. Task identifiers are the ones defined in the previous section.

Group Member	**Tasks assigned**
Ennio Visconti	1, 3, 4, 8, 10, 12, 16, 17, 18, 19, 20, 21, 22, 23
Simone Zocchi	1, 2, 5, 8, 10, 11, 14, 15, 17, 18, 19, 20, 22, 23
Khanh Huy Paolo Tran	1, 6, 7, 8, 9, 11, 13, 15, 17, 18, 19, 20, 22, 23

Table 8: Task assignment

6 Risks identification

	Risk	Probability	Effect
R1	Member of the group are ill and this will cause a shift in the plan.	Moderate	Marginal
R2	Change in requirements and design are proposed.	Moderate	Critical
R3	Architectural components cannot support the load of requests.	Low	Critical
R4	Faults in external components used (i.e. Google Maps, PayPal)	Low	Catastrophic
R5	Faults in architectural components	Low	Catastrophic
R6	Underestimate development time	Moderate	Serious

Risk	Recovery Action
R1	Organize tasks in resource allocation so that more overlapping are present.
R2	Make a clear and traceable RASD and DD and project a flexible architecture, so that changing are less effective.
R3	Project the physical architecture of the system so that it can be distributed and easily expandable.
R4	Take into account different alternatives for external components and implement different interfaces for the different ones. It will be possible to change the components used easily.
R5	Consider the possibility of redundancy of the most critical architectural components (for example Open Street Maps instead of Google Maps and Skrill instead of PayPal).
R6	Investigate the possibility to buy already done component to be integrated in the application and not implemented from scratch.