POLITECNICO DI MILANO
Computer Science and Engineering
Project of Software Engineering 2

Project Planning

Authors: Falci Angelo 875123
Lanzuise Valentina 807364
Lazzaretti Simone 875326

Reference Professor: Di Nitto Elisabetta

SUMMARY

1 Introduction

1.1 Revision History
Version 1.0 22/01/2017

1.2 Purpose and Scope
This document represents the Project Plan of PowerEnjoy. The aim of the analysis that we are going to do is providing an idea of the complexity of our project and giving an estimation of the costs, risks and efforts that its development comports. The information obtained from this analysis are useful to define the required budget, the resources allocation and the schedule of the activities.
In the first section, we are going to use a function point approach, followed by a COCOMO approach, both used in order to provide an estimation of PowerEnjoy in terms of cost/effort and code lines required to proceed in the development.
In the second section, we are going to provide a possible schedule and organization of the developing process, which will include all the activities of the requirements identification, design of the architecture, implementation and testing of the entire project. In the third section, we will elaborate our conclusion of this analysis and suggest the risks that could be faced in the different phases of development of PowerEnjoy.

1.3 Acronyms
FP: Function Points
ILF: Internal logic file
ELF: External logic file
EI: External Input
EO: External Output
EQ: External Inquiries
DBMS: Database Management System
UI: User Interface
REGISTERED USER: identify the person registered who can use the service.
USER: identify the generic person who is using the service.
SAFE AREA: identify the area where the user can leave a car to have a discount.
POWER GRID: identify the power station that a safe area can have where the user can recharge the car to obtain a discount.
SYSTEM: identify server and database that manage the web-application service, and the software that manages the car.
CAR: identify every single car provided by PowerEnJoy.
POSITION: indicate the specific position about car, safe area and power grid using latitude and longitude.
PERSONAL USER DEVICE: identify a generic device used by user with an Internet connection.

MSO: identify the “Money Saving Option” function.
VoIP: is a acronym to identify Voice over Internet Protocol is a methodology and group of technologies for the delivery of voice communications over Internet Protocol (IP) networks, such as the Internet.

1.4 Reference documents

- Assignment document
- RASD
- DD
- ITPD
- Old projects of the past years
- Lecture slides

2 Function Point Approach

The Function Points approach provides an estimation of the size of a project taking as inputs the amount of functionalities to be developed and their complexity.
For the calculation of the Function Points, we will use the following tables:

For Internal Logic Files and External Logic Files

	Data Elements		
Record Elements	1 to 19	29 to 50	51 and over
1 2-5 6>=	Low Low Average	Low Average High	Average High High

For External Output and External Inquiry

	Data Elements		
Record Elements	1 to 5	6 to 19	20 and over
0-1 2-3 4>=	Low Low Average	Low Average High	Average High High

For External Input

	Data Elements		
Record Elements	1 to 4	5 to 15	16 and over
0-1	Low	Low	Average
2-3	Low	Average	High
4>=	Average	High	High

UFP Complexity Weights

	Complexity Weight		
Function Type	Low	Average	High
Internal Logic Files	7	10	15
External Logic Files	5	7	10
External Inputs	3	4	6
External Outputs	4	5	7
External Inquiry	3	4	6

2.1 Internal Logic Files

PowerEnJoy must be able to manage personal data (credentials and payment information), cars information, safe area information, power grid information, booking and renting of users, assistant information and assistance service information.

Let's start from personal data, they consist in: nickname, password, name, surname, date of birth, payment information, e-mail, address, identity card number and driving license ID. All these data are stored in a single record, each for a client. Looking at the table, we will associate a “Low” complexity, 7 FP.

For cars information, the reason is the same, the data consist in: position, car plate, car code, state of the car, battery level, a description of car and of how many seats it is equipped. All of these are contained in a single record so, again, we will associate a “Low” complexity cost, 7 FP.

For areas, the information stored are the position of the area, the presence of a power grid, the amount of available free spaces and the amount of energy stored in the power grid (if this is situated in a safe area). Looking at the table, we will associate a “Low” complexity cost, 7 FP.

Booking information consists in: when start, when finish, userID and car information. We will use three records for all these data, one for the userID, one for car information and the last one named “booking” that contains the previous records and the two “time” information. Looking at the table, we will associate a “Low”

complexity cost, 7 FP.
For the last set of information, the renting one, the information stored are: the userID, the car used, the final charge without discounts and/or raises, the final charge with discounts and/or raises applied (if any), the time when the rent begins and finishes, the number of passengers revealed by sensors, the battery level at the end, the presence of discounts and/or raises, if the renter activated the money saving option, the road driven and possible fines (if user gets them during the renting). All of these information are stored in three records, one for the userID, one for car information and the last one named “renting” that contains the previous records and the other information we said in the previous list. So we will associate a “Low” complexity level, 7 FP.
As a final sum of this part, the amount is 49 FPs.

ILF	Complexity	FPs
Personal data	Low	7
Cars information	Low	7
Safe Area information	Low	7
Booking	Low	7
Renting	Low	7
Assistant	Low	7
Assistance service information	Low	7
Total		49

2.2 External Logic Files

The only data managed in this section are data for maps. They taken from the web to be displayed on web browser in user device and they are included in the car device in order to show the graphical representation of the city map.
We decide to adopt a “Low” weight. As a result, we get 2 x 5 = 10 FP.

ELF	Complexity	FPs
Data for maps in user device	Low	5
Data for maps in car display	Low	5
Total		10

2.3 External Inputs

PowerEnjoy manages different interactions with users, which differs for the typology of users but also for the devices used (car device or personal device).
We are now going to summarize the impact of the offered features, in the following order:
User Not Registered:

- Sign Up: it leads the compile of many fields, and the verification of them. So we can consider it an operation of average complexity: 4FPs
- Personal code release: this operation has a low complexity, it just sends for email the personal code that user must be used to turn on the car. For this reason, it contributes 4 FPs

Registered user from their device:

- Login/Logout: they are simple operations so we can adopt the simple weight for them. 2 x 3 = 6 FPs
- Modify Information: it is just the modification of old data of the user, so it implies only one entity but its complexity depends on the number of data to be changed: 4FPs
- Show map: this operations involves many entities, like users, cars, safe area, power grid and their positions. Its complexity is high: 6 FPs
- Select cars: this operations is quite simple and involves only two entities: 3 FPs
- Book a car: it is a complex operation because it includes a lot of steps and components to be completed: 6 FPs
- Rent a car/unlock it: it involves at least four entities, implies the change of the different status of cars/renting/booking and includes many components, so has high complexity for each: 6 FPs
- View booking: it is a simple operation that permits to user to view its booking: 3 FPs
- Search an address: it is like “show map” but it doesn't consider the user position. For this reason the complexity decreases a little bit: 4 FPs
- Safe area/Power grids: it implies the two entities which are shown in the nearby position of users, so it is still a average complex operation for each entities: 4x2=8 FPs
- Delete booking: it involves three entities (car, user and booking) but it is a straightforward operation, so its complexity is average: 4 FPs
- View notification history: it is a simple operation, that allows user to visualize old notification received: 3 FPs
- Contact Assistance: it involves the starting of a call with VPS to the system that manages the assistance. It is a average complexity operation: 4 FPs

Registered user from the car device:

- Search destination: from the position of the car, it search the destination and computes the best rout, so it is a high complexity: 6 FPs

- Activate MSO: it involves many entities and components, and must reveal the best way in order to get discount: for this reason its complexity is high: 6 FPs
- View SafeArea/PowerGrid it is the same for the personal device: 4x2 FPs
- Activate Park Function: this operations just permit to continue the reservation, but release the key of the car in order to permit the user to exit from it. It is a simple operation: 3 FPs
- View Charge: it allows user to visualize the current charge. It is not high complex operation: 3 FPs
- Turn on car: it permits to user to insert the code, which is verified, while the car is turned on manually. It is a simple operation: 3 FPs

Assistant:

- View cars: in the working station assistants can see the position of cars in order to take them, bring them in the safe areas and refuel them. The displaying of cars in the map works exactly as for users, but without any functionality (like selecting, booking). For this reason the complexity is average: 4 FPs
- View reports: assistants visualize the all votes of the cars sent by user during the renting. It is a simple operation: 3 FPs

Administrators:

- Insert, cancel, update Safe Area/Power grids: administrators insert/delete in the system the zones in which there are safe areas and power grids. These are very complex operations that involves a great number of components. For this reason they account for 6 FPs each. 6X3= 18 FPs
- Add/delete new cars: cars need to be add in the system, for then being displaced in the map and used for the service. It has an high complexity:12 FPs
- Insert/cancel Assistants: as for the zones, the complexity of these operations is high, so they account for another 6 FPs each: 12 FPs.
- Insert/delete bonus/fee: administrators add new discounts for special offers, and decide how and for what manage them, idem for the fee. They have a high complexity both: 2x2x6=24 FPs

EI	Complexity	FPs
Sign Up	Average	4
Personal code release	Average	4
Login/Logout	Low	2x3=6
Modify Information	Average	4
Show map	High	6
Select cars	Low	3
Book a car/Rent a car	High	6x2=12
View booking, delete booking, view notification history	Low	3x3=9

Search an address, Search Safe area/Power grids (user)	Average	4x3=12
Contact assistance	Average	4
Search destination/activate MSO	High	6x2=12
View Safe Areas /Power Grids	Average	4x2=8
Activate Park Function, turn on car, view charge	Low	3x3=9
View cars (assistant)	Average	4
View reports (assistant)	Low	3
Administrators operations	High	6x11=66
Total		166

2.4 External Output

PowerEnjoy has to communicate with user through notification, error message and map update.

So, all the notification can be considered as simple action, so we can associate them a “Low” complexity level. For error message the reason it's the same, because it's like a notification, but “negative”.

For map update the system sends coordinates, they are simple integer date, so we can associate them a “Low” complexity level.

So the cost is 4 FP for each operation, for a final sum of about 30 FP (must analyze better)

EO	Complexity	FPs
Login confirm/error	Low	4x2
Modify information confirm/error	Low	4x2
Booking confirm/error	Low	4x2
Start renting confirm/error	Low	4x2
Stop renting confirm	Low	4
Payment notification	Low	4
Best safe area for MSO	Low	4
Address request	Low	4
Total		30

2.5 External Inquiry

For “Inquiry”, we means the list of data that can be requested by user from the service. The lists that can be retrieved by registered users are: booking list, safe areas list, car information list and notification list.
For assistants, the unique list they can access is the pending request one.
All these operations are simple, so we will associate a “Low” complexity cost, 3 FP for each operation, for a result of 18 FP.

EI	Complexity	FPs
List of booking	Low	3
List of notifications	Low	3
List of information about a car	Low	3
List of pending requests	Low	3
List of safe areas with power grid	Low	3
List of safe areas without power grid	Low	3
Total		18

2.6 Overall Estimation

Now that we have all the FP, we can establish the total sum

Function Type	Value
Internal Logic Files	49
External Logic Files	10
External Inputs	166
External Inquiries	30
External Outputs	18
Total	272

Considering Java Enterprise Edition as a development platform and disregarding the aspects concerning the implementation of the browser pages for user device we can estimate the total number of lines of code.

Language	QSM SLOC/FP Data			
	Avg	Median	Low	High
J2EE	46	49	15	67

Depending on the conversion rate, we have a average bound of:
SLOC = 272 x46 = 12512
and an upper bound of
SLOC = 272 x67 = 18224

3 COCOMO Approach

In this section we're going to use the COCOMO II approach to estimate the cost and effort needed to develop. With this model we take in account not only the characteristics of the product but also the people will be involved and the process that will be used.

3.1 Scale drivers

The following table will be used to evaluate the scale drivers, whose cost is situated in the lower part of each cell

Scale Factors	Very Low	Low	Nominal	High	Very High	Extra High
PREC	Thoroughly unprecedented 6.20	Largely unprecedented 4.96	Somewhat unprecedented 3.72	Generally familiar 2.48	Largely familiar 1.24	Thoroughly familiar 0.00
FLEX	Rigorous 5.07	Occasional relaxation 4.05	Some relaxation 3.04	General conformity 2.03	Some conformity 1.01	General goals 0.00
RESL	Little 7.07	Some 5.65	Often 4.24	Generally 2.83	Mostly 1.41	Full 0.00
TEAM	Very difficult interactions 5.48	Some difficult interactions 4.38	Basically cooperative interactions 3.29	Largely cooperative 2.19	Highly cooperative 1.10	Seamless interactions 0.00
PMAT	Level 1 Lower 7.80	Level 1 Upper 6.24	Level 2 4.68	Level 3 3.12	Level 4 1.56	Level 5 0.00

Precedentedness: it reflects the previous experience of our team with the development of large scale projects. Since we are not expert in the field, this value will be low.
Development flexibility: it reflects the degree of flexibility in the development process with respect to the external specification and requirements. Since there are very strict requirements on the functionalities but nothing specific is stated as for the technology to be used, this value will be low.
Risk resolution: reflects the level of awareness and re-activeness with respect to risks. The risk analysis we performed is quite extensive, so the value will be set to very high.(?)
Team cohesion: it is an indicator of how well the team members know each other and

work together in a cooperative way. For our team, the value is normalize.
Process maturity: it is the first project of this kind, and we tried to satisfy the requests the best we could, in the way we thought to be the appropriate and complete, even our inexperience. For this reason this value is set to high.

The following table summarize the total cost:

Scale Driver	Factor	Value
Precedentedness (PREC)	Low	4.96
Development exibility (FLEX)	Low	4.05
Risk resolution (RESL)	Very High	1.41
Team cohesion (TEAM)	Normalize	3.29
Process maturity (PMAT)	High	3.12
Total		16,83

3.2 Cost Drivers

- Required Software Reliability: in the city people can have a lot choice between vehicles to use in order to travel, and they will choose the most convenient. In order to be competitive and to ensure an efficient system, while the malfunctioning could lead to important financial losses. For this reason the RELY Cost Drivers is set to high.

Table 17. RELY Cost Driver

RELY Descriptors:	slight inconven-ience	low, easily recoverable losses	moderate, easily recoverable losses	high financial loss	risk to human life	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	0.82	0.92	1.00	1.10	1.26	n/a

- Database size:
  Our test database size is equal to 200 KB and then the average estimate of the program size is equal to 12512 SLOC, the division D/P=15,98. This parameter has a nominal value.

Table 18. DATA Cost Driver

DATA* Descriptors		Testing DB bytes/Pgm SLOC < 10	10 ≤ D/P < 100	100 ≤ D/P < 1000	D/P ≥ 1000	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	0.90	1.00	1.14	1.28	n/a

* DATA is rated as Low if D/P is less than 10 and it is very high if it is greater than 1000. P is measured in equivalent source lines of code (SLOC), which may involve function point or reuse conversions.

- Product Complexity:
  Set to Nominal, according to the new COCOMO II CPLEX rating scale.

Table 20. CPLX Cost Driver

Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	0.73	0.87	1.00	1.17	1.34	1.74

- Required Reusability:
  In the server side we divided our functionalities in more modules. So every module can be reuse for other new services that will take shape in the future. For this reason this parameter is set to very high.

Table 21. RUSE Cost Driver

RUSE Descriptors:		none	across project	across program	across product line	across multiple product lines
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	0.95	1.00	1.07	1.15	1.24

- Documentation match to life-cycle needs:
  This parameter measures how much we respect the requirements of the project. Based on our RASD and DD Documents we think to cover the requirements but we don't just based our project to them, so we can assign a nominal value.

Table 22. DOCU Cost Driver

DOCU Descriptors:	Many life-cycle needs uncovered	Some life-cycle needs uncovered.	Right-sized to life-cycle needs	Excessive for life-cycle needs	Very excessive for life-cycle needs	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	0.81	0.91	1.00	1.11	1.23	n/a

- Execution Time Constraint:
  For some operations, about booking function and renting function, the time needed to satisfy the request must be enough short. For this reason we can set this parameter to very high.

Table 23. TIME Cost Driver

TIME Descriptors:			≤ 50% use of available execution time	70% use of available execution time	85% use of available execution time	95% use of available execution time
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	n/a	1.00	1.11	1.29	1.63

- Main Storage Constraint:
  We manage simple type of data like “double” or “string”, this mean that the server will spend more year to fill 1 T Byte, so we can set this parameter as nominal.

Table 24. STOR Cost Driver

STOR Descriptors:			≤ 50% use of available storage	70% use of available storage	85% use of available storage	95% use of available storage
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	n/a	1.00	1.05	1.17	1.46

- Platform Volatility:
  We chose to distribute the service as application web, so the service doesn't need update if the user device change because in every device can run a browser. So we can set this parameter to low.

Table 25. PVOL Cost Driver

PVOL Descriptors:		Major change every 12 mo.; Minor change every 1 mo.	Major: 6 mo.; Minor: 2 wk.	Major: 2 mo.;Minor: 1 wk.	Major: 2 wk.;Minor: 2 days	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	0.87	1.00	1.15	1.30	n/a

- Analyst Capability:
  In our previous documents we spend a lot of time to analyze the problem in order to satisfy all the requirements but also to provide solutions for eventual problems occur in the real world. We discuss about this mainly in our RASD document, when we also resolve the ambiguities in the initial description.
  So we can set this parameter to high.

Table 26. ACAP Cost Driver

ACAP Descriptors:	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.42	1.19	1.00	0.85	0.71	n/a

- Programmer Capability:
  We got along during the project but we don't have a real experience as a team, but also in this kind of project, so we can set this parameter as nominal.

Table 27. PCAP Cost Driver

PCAP Descriptors	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.34	1.15	1.00	0.88	0.76	n/a

- Application Experience:
  We have experience with Java but we have never used Java EE before so we can set this parameter as very low.

Table 29. APEX Cost Driver

APEX Descriptors:	≤ 2 months	6 months	1 year	3 years	6 years	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.22	1.10	1.00	0.88	0.81	n/a

- Platform Experience:
  Though at the beginning of the project we didn't know Java EE, we had some experience with database, user interfaces and client-server application so we we can assume nominal this parameter.

Table 30. PLEX Cost Driver

PLEX Descriptors:	≤ 2 months	6 months	1 year	3 years	6 year	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.19	1.09	1.00	0.91	0.85	n/a

- Language and Tool Experience:
  For this parameter is valid the things we said in the previous parameter so we can set this like nominal.

Table 31. LTEX Cost Driver

LTEX Descriptors:	≤ 2 months	6 months	1 year	3 years	6 year	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.20	1.09	1.00	0.91	0.84	

- Personnel Continuity:
  We had definitive and clear delivery so we work continuously and in an organized manner to respect our deadlines. So we can set this parameter to high.

Table 28. PCON Cost Driver

PCON Descriptors:	48% / year	24% / year	12% / year	6% / year	3% / year	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.29	1.12	1.00	0.90	0.81	

- Usage of Software Tools:
  Our application environment is complete and well integrated, so we'll set this parameter as high.

Table 32. TOOL Cost Driver

TOOL Descriptors	edit, code, debug	simple, frontend, backend CASE, little integration	basic life-cycle tools, moderately integrated	strong, mature life-cycle tools, moderately integrated	strong, mature, proactive life-cycle tools, well integrated with processes, methods, reuse	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.17	1.09	1.00	0.90	0.78	n/a

- Multi-site Development:
  we live in different cities, hence we worked mainly using Internet wide band, with messaging applications and file hosting service, but also meeting us in college to complete and discuss the main things of the project. For this reason the parameter is set to nominal.

Table 33. SITE Cost Driver

SITE: Collocation Descriptors: SITE: Communications Descriptors:	Inter-national Some phone, mail	Multi-city and Multi-company Individual phone, FAX	Multi-city or Multi-company Narrow band email	Same city or metro. area Wideband electronic communicat ion.	Same building or complex Wideband elect. comm., occasional video conf.	Fully collocated Interactive multimedia
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.22	1.09	1.00	0.93	0.86	0.80

- Required Development Schedule:
  The time spent for the project was limited by the assignment deadlines, but we think to have spent the right hours in order to complete and satisfy the requirements adequately. For this reason this parameter is set to nominal.

Table 34. SCED Cost Driver

SCED Descriptors	75% of nominal	85% of nominal	100% of nominal	130% of nominal	160% of nominal	
Rating Level	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multiplier	1.43	1.14	1.00	1.00	1.00	n/a

Our results are expressed in the following table:

Scale Driver	Factor	Value
Required Software Reliability	High	1,1
Database Size	Nominal	1
Product Complexity	Nominal	1
Required Re-usability	Very High	1,15

Documentation match to life-cycle needs	Nominal	1
Execution Time Constraint	Very high	1,29
Main Storage Constraint	Nominal	1
Platform Volatility	Low	0,87
Analyst Capability	High	0,85
Programmer Capability	Nominal	1
Application Experience	Very Low	1,22
Platform Experience	Nominal	1
Language and Tool Experience	Nominal	1
Personnel Continuity	High	0,9
Usage of Software Tool	High	0,9
Multi-site Development	Nominal	1
Required Development Schedule	Nominal	1
Total:		1,19

The total number with all the decimal digits is 1,192513388715

3.3 Effort Equation

This final equation gives us the effort estimation measured in Person-Months(PM):
Effort = A EAF*(KSLOC)^E*

where *A=2.94* (for COCOMO II)

EAF=is the total of cost drivers calculated previously= *1,192513388715*

KSLOC=SLOC:1000= 12512:1000=*12,512*

E= Exponent derived from Scale Drivers=It is computed as :
B + 0.01 *ΣiSF[i] = B + 0.01*16,83=*1,0783* in which B is equal to 0.91 for COCOMO II .

With this parameters we can compute the effort value:
Effort=2,94* 1,192513388715* (12,512)^1,0783=*53,46357712* Person-Months , so we can approximate to 53PM.

3.4 Schedule Estimation

Regarding the final schedule, we are going to use the following formula:

Duration= 3,67(Effort)^F*
Effort is the same we calculated before: *53,46357712 PM*

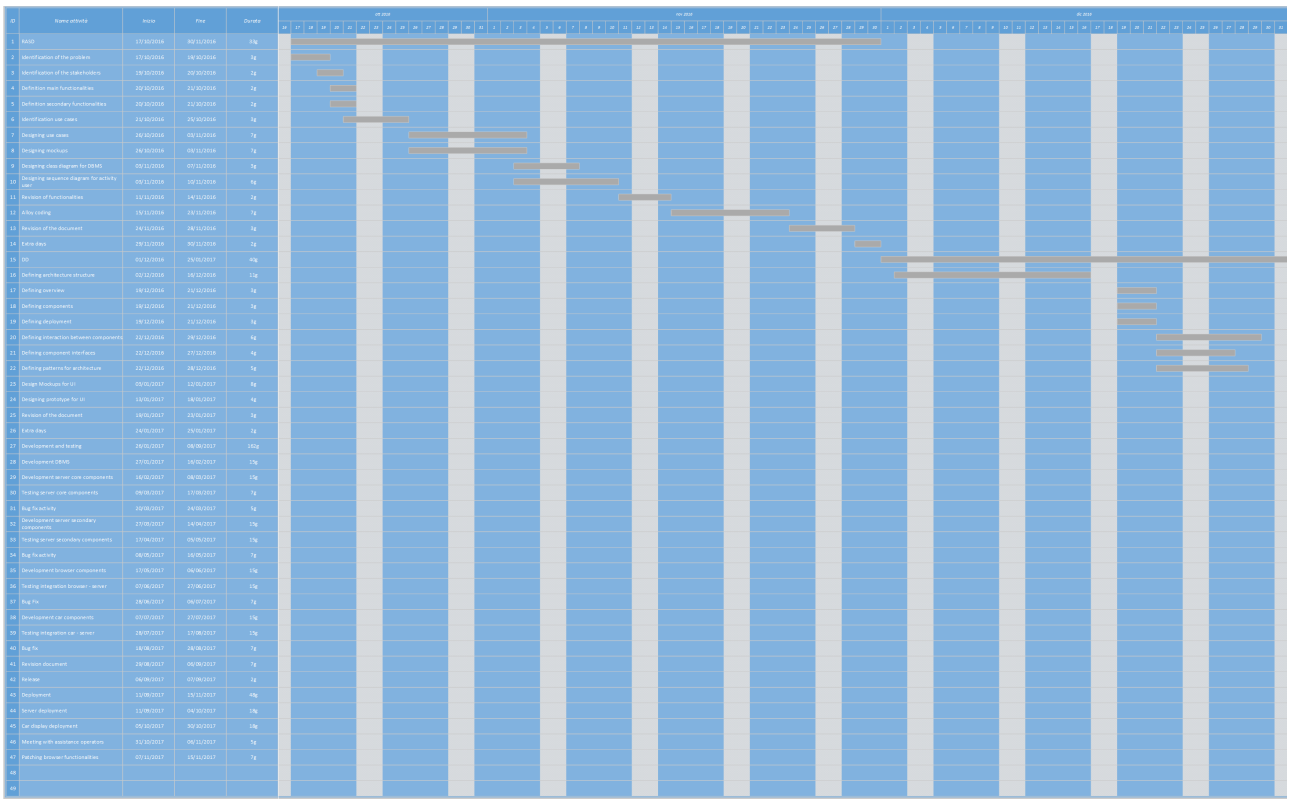
F is the schedule equation exponent derived from the five Scale Drivers:
0.28 + 0 . 2 *(E – B), where E and B are the same defined before.
F=0,28+0,2*(*1,0783- 0.91)=0,31366*

The final result with this parameters is:
Duration=3,67*(*53,46357712)^(0,31366)=12,78464699 months* which can be approximated to 13 months.

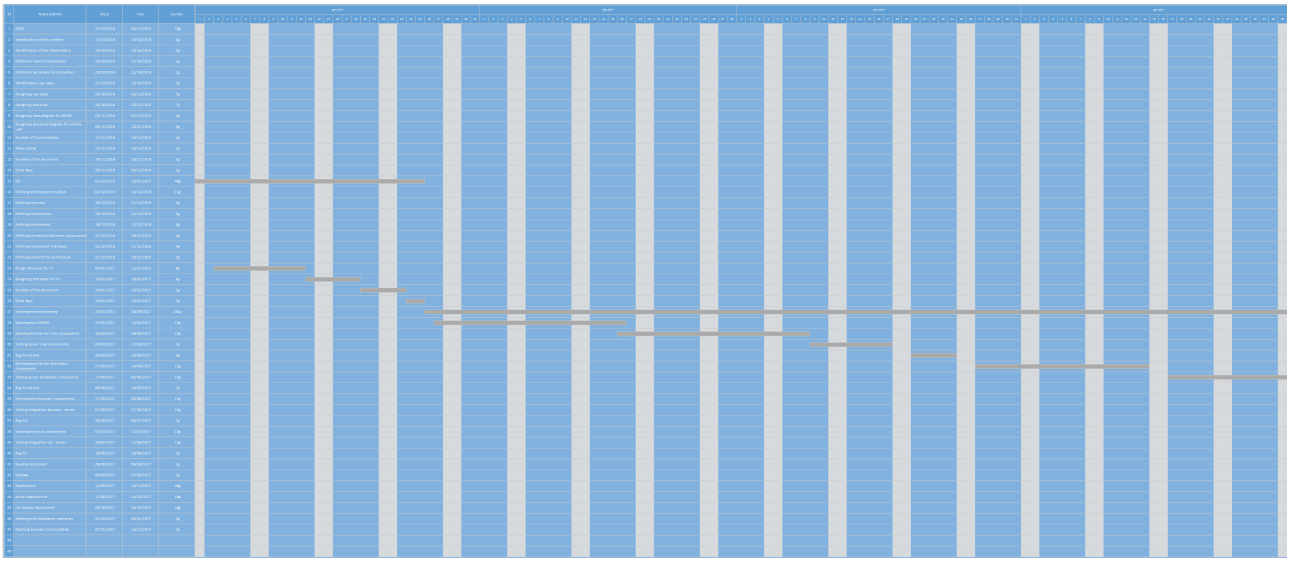
At last, we can estimate the required people for the work with the formula:
Npeople= Effort/Duration=*53,46357712 PM / 12,78464699 M=4,18 Person* which can be approximated to 4.

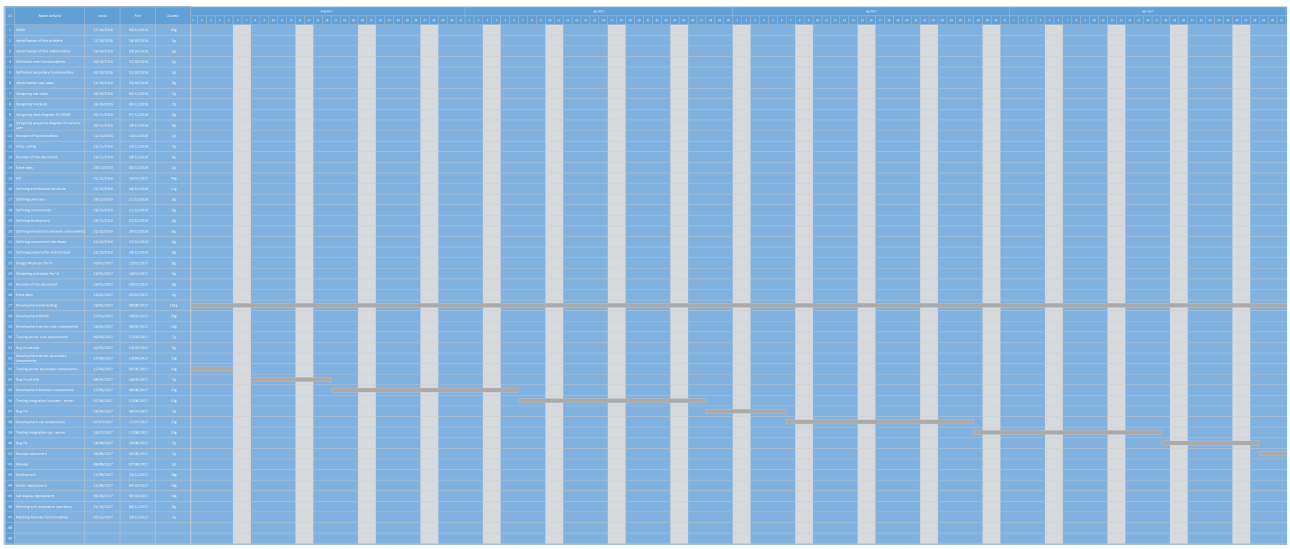
4 Schedule

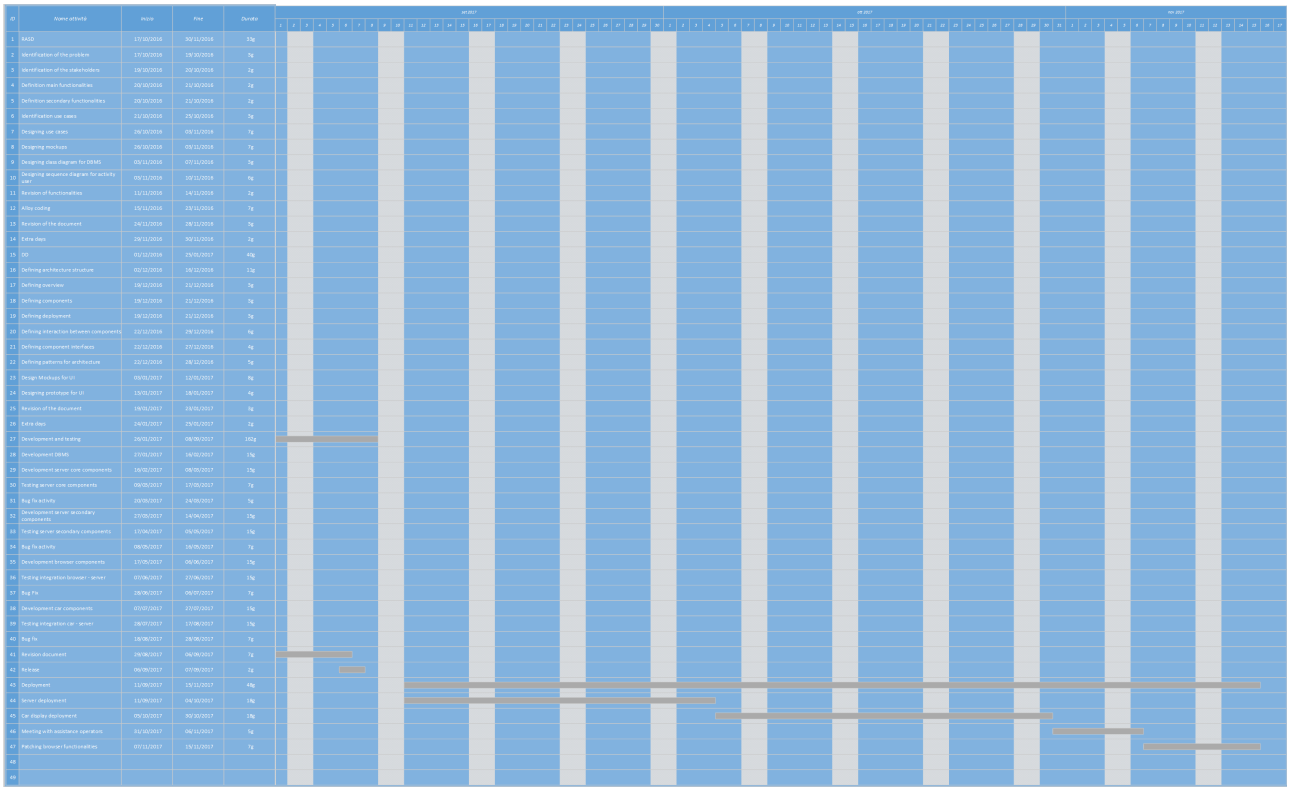
In this chapter it will be shown, according to COCOMO results, the whole schedule of the project, taking in considerations the main phases of the first life-cycle of the software (RASD, Design, Development, Testing, Deployment). It must be said that all the activities reported are the most relevant and the one that our groups faced during all the project period: these means that, in a real situation, others activities could belong to this list. Also, the time schedule takes in consideration the fact all the schedule is reduced from four people (the number of people underlined by COCOMO) to three (the actual number of people in our team)
Here the Gantt graph, illustrating the whole schedule:



ID	Name attività	Inizio	Fine	Durata
1	RAID	17/10/2016	30/11/2016	31g
2	Identification of the problem	17/10/2016	19/10/2016	2g
3	Identification of the stakeholders	19/10/2016	20/10/2016	2g
4	Definition main functionalities	20/10/2016	21/10/2016	2g
5	Definition secondary functionalities	20/10/2016	21/10/2016	2g
6	Identification use cases	21/10/2016	24/10/2016	3g
7	Designing use cases	24/10/2016	03/11/2016	7g
8	Designing mockups	24/10/2016	03/11/2016	7g
9	Designing class diagram for DBMS	03/11/2016	07/11/2016	5g
10	Designing sequence diagram for activity UART	03/11/2016	10/11/2016	6g
11	Revision of functionalities	11/11/2016	14/11/2016	2g
12	Alloy coding	15/11/2016	23/11/2016	7g
13	Revision of the document	24/11/2016	28/11/2016	3g
14	Extra days	29/11/2016	30/11/2016	2g
15	DD	01/12/2016	25/01/2017	40g
16	Defining architecture structure	02/12/2016	16/12/2016	11g
17	Defining overview	19/12/2016	21/12/2016	3g
18	Defining components	19/12/2016	21/12/2016	3g
19	Defining deployment	19/12/2016	21/12/2016	3g
20	Defining interaction between components	22/12/2016	29/12/2016	6g
21	Defining component interfaces	22/12/2016	27/12/2016	4g
22	Defining patterns for architecture	22/12/2016	28/12/2016	5g
23	Design Modcuss for UI	03/01/2017	12/01/2017	8g
24	Designing prototype for UI	13/01/2017	18/01/2017	4g
25	Revision of the document	19/01/2017	22/01/2017	3g
26	Extra days	24/01/2017	25/01/2017	2g
27	Development and testing	26/01/2017	08/09/2017	162g
28	Development DBMS	27/01/2017	16/02/2017	15g
29	Development server core components	16/02/2017	08/03/2017	15g
30	Testing server core components	09/08/2017	17/08/2017	7g
31	Bug fix activity	20/03/2017	24/03/2017	5g
32	Development server secondary components	27/03/2017	14/04/2017	15g
33	Testing server secondary components	17/04/2017	05/05/2017	15g
34	Bug fix activity	08/05/2017	16/05/2017	7g
35	Development browser components	17/05/2017	04/06/2017	15g
36	Testing integration browser - server	07/06/2017	27/06/2017	15g
37	Bug Fix	28/06/2017	06/07/2017	7g
38	Development car components	07/07/2017	27/07/2017	15g
39	Testing integration car - server	28/07/2017	17/08/2017	15g
40	Bug fix	18/08/2017	24/08/2017	7g
41	Revision document	29/08/2017	06/09/2017	7g
42	Release	06/09/2017	07/09/2017	2g
43	Deployment	11/09/2017	15/11/2017	48g
44	Server-deployment	11/09/2017	04/10/2017	18g
45	Car display deployment	05/10/2017	30/10/2017	18g
46	Meeting with assistance operators	31/10/2017	09/11/2017	9g
47	Patching browser functionalities	07/11/2017	15/11/2017	7g
48				
49				





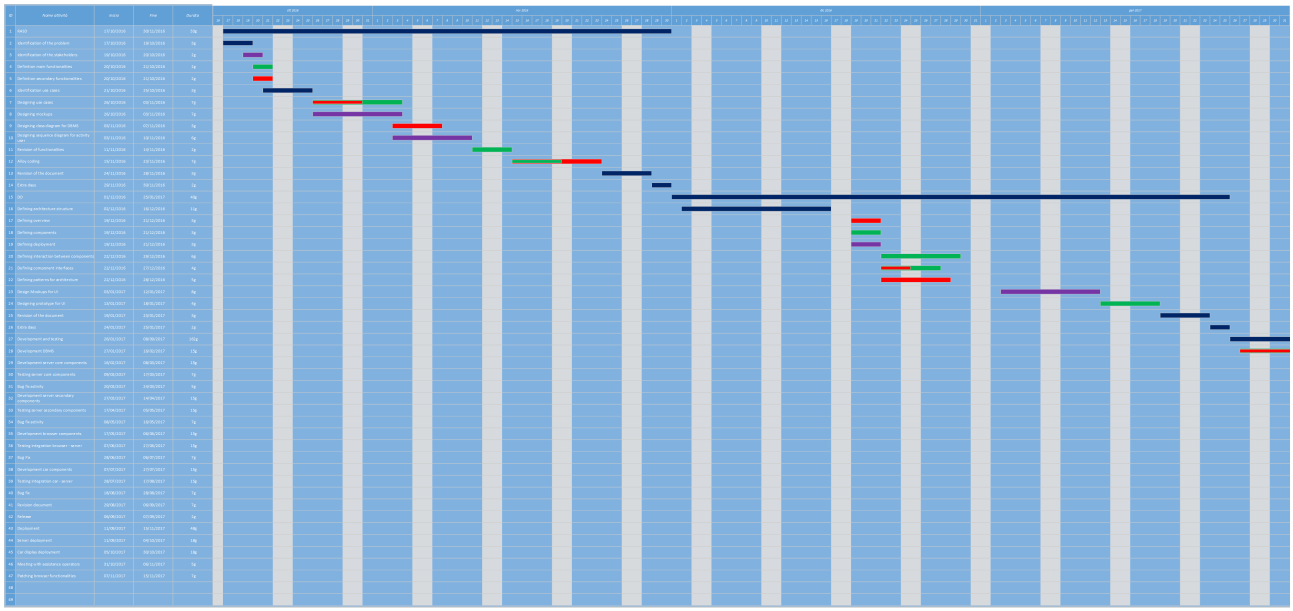


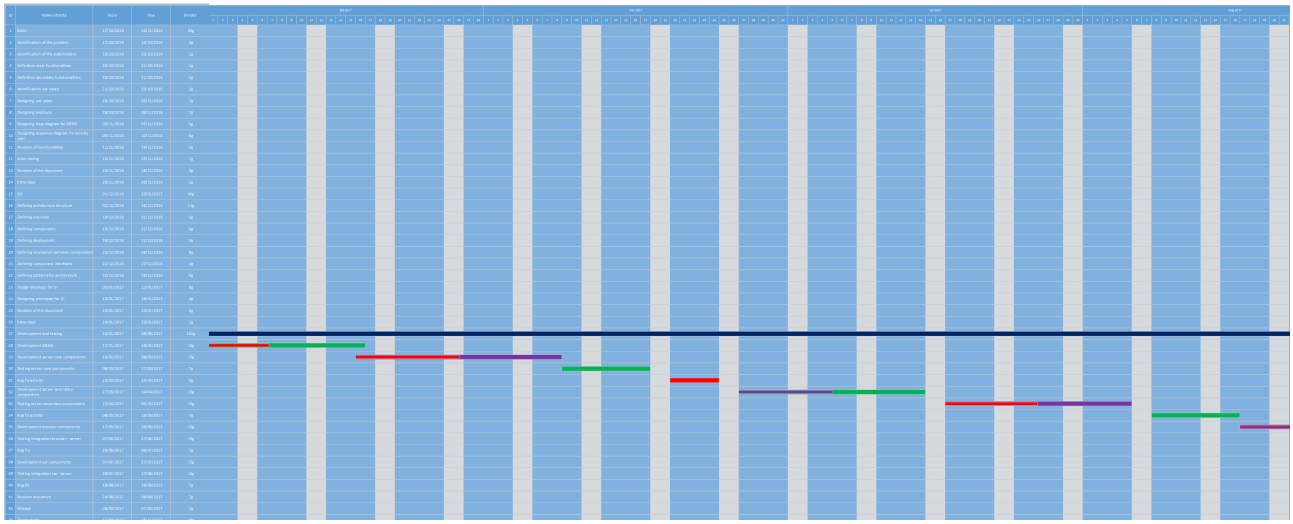
ID	Nome attività	Inizio	Fine	Durata
1	BASD	17/10/2016	30/11/2016	33g
2	Identification of the problem	17/10/2016	19/10/2016	1g
3	Identification of the stakeholders	19/10/2016	20/10/2016	2g
4	Definition main functionalities	20/10/2016	21/10/2016	2g
5	Definition secondary functionalities	20/10/2016	21/10/2016	2g
6	Identification use cases	21/10/2016	25/10/2016	5g
7	Designing use cases	26/10/2016	03/11/2016	7g
8	Designing mockups	26/10/2016	03/11/2016	7g
9	Designing class diagram for DBMS	03/11/2016	07/11/2016	5g
10	Designing sequence diagram for activity user	03/11/2016	10/11/2016	6g
11	Revision of functionalities	11/11/2016	14/11/2016	2g
12	Alloy coding	15/11/2016	23/11/2016	7g
13	Revision of the document	24/11/2016	28/11/2016	5g
14	Extra days	29/11/2016	30/11/2016	2g
15	DD	01/12/2016	23/01/2017	40g
16	Defining architecture structure	02/12/2016	16/12/2016	11g
17	Defining overview	19/12/2016	21/12/2016	3g
18	Defining components	19/12/2016	21/12/2016	3g
19	Defining deployment	19/12/2016	21/12/2016	3g
20	Defining interaction between components	22/12/2016	29/12/2016	6g
21	Defining component interfaces	22/12/2016	27/12/2016	4g
22	Defining patterns for architecture	22/12/2016	28/12/2016	5g
23	Design Mockups for UI	03/01/2017	12/01/2017	8g
24	Designing prototype for UI	13/01/2017	18/01/2017	4g
25	Revision of the document	18/01/2017	23/01/2017	3g
26	Extra days	24/01/2017	25/01/2017	2g
27	Development and testing	26/01/2017	08/09/2017	162g
28	Development (DBMS)	27/01/2017	16/02/2017	15g
29	Development server core components	16/02/2017	08/09/2017	15g
30	Testing server core components	09/05/2017	17/05/2017	7g
31	Bug fix activity	20/05/2017	24/05/2017	5g
32	Development server secondary components	27/05/2017	14/06/2017	15g
33	Testing server secondary components	17/06/2017	05/09/2017	15g
34	Bug fix activity	08/05/2017	16/05/2017	7g
35	Development browser components	17/05/2017	05/06/2017	15g
36	Testing integration browser - server	07/06/2017	27/06/2017	15g
37	Bug Fix	28/06/2017	06/07/2017	7g
38	Development car components	07/07/2017	27/07/2017	15g
39	Testing integration car - server	28/07/2017	17/08/2017	15g
40	Bug fix	18/08/2017	28/08/2017	7g
41	Revision document	29/08/2017	06/09/2017	7g
42	Release	06/09/2017	07/09/2017	2g
43	Deployment	11/09/2017	15/11/2017	48g
44	Server deployment	11/09/2017	03/10/2017	18g
45	Car display deployment	03/10/2017	30/10/2017	18g
46	Meeting with assistance operators	31/10/2017	06/11/2017	5g
47	Patching browser functionalities	07/11/2017	15/11/2017	7g
48				
49				

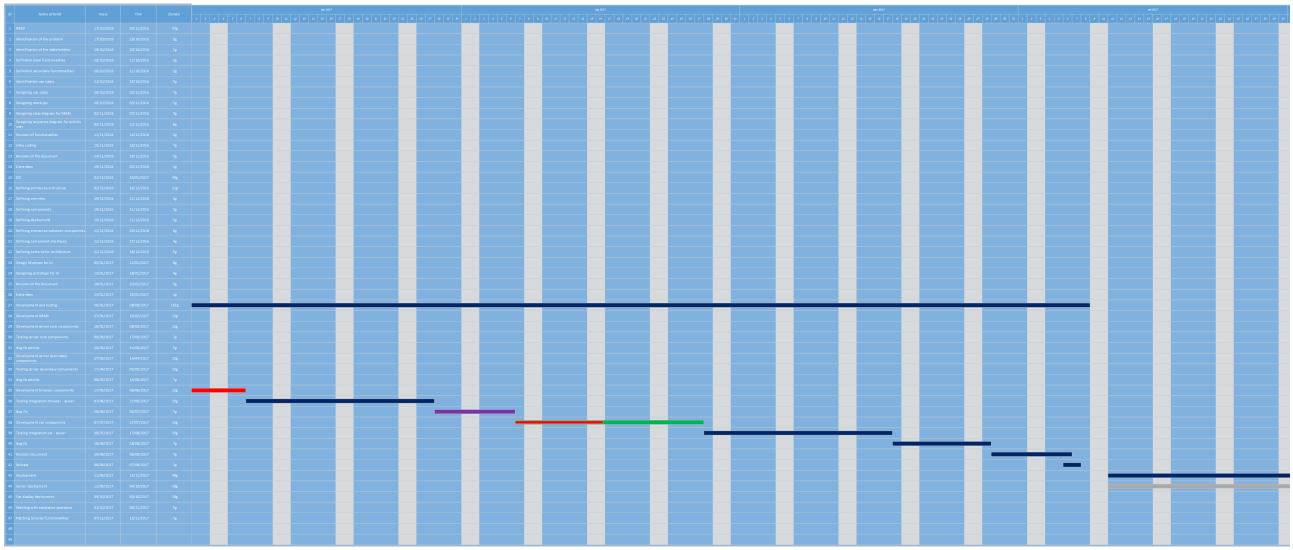
5 Resource allocation

In this chapter we're going to provide a general overview of how the tasks dened by the schedule in the previous section will be divided between the three members of our team. As we already mentioned in the previous section, we have also included activities in the requirement analysis and design phases that won't actually take place, like the stakeholders meetings, as well as the full implementation phase. This has been purposefully done to have a more realistic description of how the development process could go.
The following diagrams shows with the different colors how the work would be devided: in blue are shown the activities that take place with all the members, while

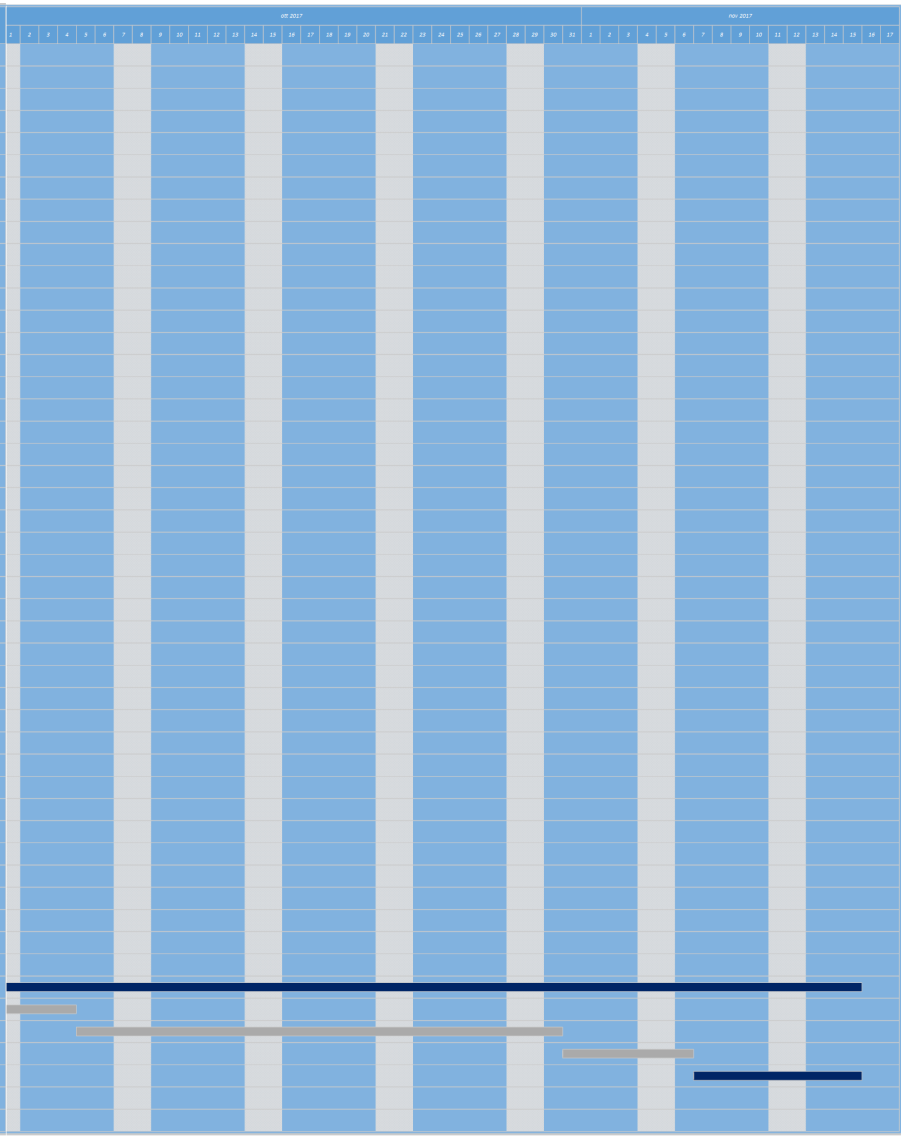
the colors red, green and purple highlight the activities of each team member.







ID	Nome attività	Inizio	Fine	Durata
1	RASD	17/10/2016	30/11/2016	33g
2	Identification of the problem	17/10/2016	19/10/2016	3g
3	Identification of the stakeholders	19/10/2016	20/10/2016	2g
4	Definition main functionalities	20/10/2016	21/10/2016	2g
5	Definition secondary functionalities	20/10/2016	21/10/2016	2g
6	Identification use cases	21/10/2016	25/10/2016	3g
7	Designing use cases	26/10/2016	03/11/2016	7g
8	Designing mockups	26/10/2016	03/11/2016	7g
9	Designing class diagram for DBMS	03/11/2016	07/11/2016	3g
10	Designing sequence diagram for activity user	03/11/2016	10/11/2016	6g
11	Revision of functionalities	11/11/2016	14/11/2016	2g
12	Alloy coding	15/11/2016	23/11/2016	7g
13	Revision of the document	24/11/2016	28/11/2016	3g
14	Extra days	29/11/2016	30/11/2016	2g
15	DD	01/12/2016	25/01/2017	40g
16	Defining architecture structure	02/12/2016	16/12/2016	11g
17	Defining overview	19/12/2016	21/12/2016	3g
18	Defining components	19/12/2016	21/12/2016	3g
19	Defining deployment	19/12/2016	21/12/2016	3g
20	Defining interaction between components	22/12/2016	29/12/2016	6g
21	Defining component interfaces	22/12/2016	27/12/2016	4g
22	Defining patterns for architecture	22/12/2016	28/12/2016	5g
23	Design Mockups for UI	03/01/2017	12/01/2017	8g
24	Designing prototype for UI	13/01/2017	18/01/2017	4g
25	Revision of the document	19/01/2017	23/01/2017	3g
26	Extra days	24/01/2017	25/01/2017	2g
27	Development and testing	26/01/2017	08/09/2017	162g
28	Development DBMS	27/01/2017	16/02/2017	15g
29	Development server core components	16/02/2017	08/03/2017	15g
30	Testing server core components	09/03/2017	17/03/2017	7g
31	Bug fix activity	20/03/2017	24/03/2017	5g
32	Development server secondary components	27/03/2017	14/04/2017	15g
33	Testing server secondary components	17/04/2017	05/05/2017	15g
34	Bug fix activity	08/05/2017	16/05/2017	7g
35	Development browser components	17/05/2017	06/06/2017	15g
36	Testing integration browser - server	07/06/2017	27/06/2017	15g
37	Bug Fix	28/06/2017	06/07/2017	7g
38	Development car components	07/07/2017	27/07/2017	15g
39	Testing integration car - server	28/07/2017	17/08/2017	15g
40	Bug fix	18/08/2017	28/08/2017	7g
41	Revision document	29/08/2017	06/09/2017	7g
42	Release	06/09/2017	07/09/2017	2g
43	Deployment	11/09/2017	15/11/2017	48g
44	Server deployment	11/09/2017	04/10/2017	18g
45	Car display deployment	05/10/2017	30/10/2017	18g
46	Meeting with assistance operators	31/10/2017	06/11/2017	5g
47	Patching browser functionalities	07/11/2017	15/11/2017	7g
48				
49				



6 Risk management

In this chapter we try to find, analyze and resolve the main problems we can meet during and after the realization of this project.
These problems can be directly connected to the project in course or they can be indirect problems.
Here a short list about the most critical risks we must be ready to face:

- The company PowerEnjoy can fail while we are developing the project they commissioned us. It can be a real big problem, mostly if it happens when we have almost finished because we wasted a lot of time in a complex project, but we will not take the entire revenue.
  We can solve this problem, if it happens, offering our project to another automotive company that can be interested in electrical cars.
- Some member of the team can leave the team before the end of the project, because its own decision or for external forces. This is a problem because

maybe he/she leaves its work unfinished and the new member will understand the beginning work in order to complete it.
We can't resolve this problem because depend of the people, but we can divide the responsibilities of project between more people, so that does not exist a single person with huge responsibilities, in order to prevent we waste a lot of time in case that a single person with huge responsibilities would leave us.

- The breaking of the server is the most dangerous risk for our project because we virtually “loose” all cars. The assistant couldn't know where the cars are, the service will be not usable and the people that rent a car before the breaking of the server couldn't stop it.
  We must be sure to guarantee that our server will be always available and usable. A possible solution is to create a backup server. It is a server equal to the main but not working normally. When the main server collapse or has a problem, this second server starts to satisfy the requests.
  Probably it is better to have more than one backup server, at least two backup server in addition to the main server.
- An other similar problem is about the database.
  Our data are very important to ensure that entire system works. They can be lost after a software/hardware error or they can be modified from other bad people.
  We can reduce this risk with two different approaches. We can use at least two back up memories, similar to the previous case, to be sure that an hardware problem does not compromise the validity of the data. We also can put a firewall between the DBMS and the other function of the server.
  For a final safety measure we give the permission to interact with the data to the less people possible.

7 Effort spent

Angelo Falci: 7h
Valentina Lanzuise: 10h
Simone Lazzaretti: 6h