POLITECNICO DI MILANO

Department of Electronic, Information and Biomedical Engineering

Master Degree course in Computer Science Engineering

Project of Software Engineering 2

PowerEnJoy

Integration Test Plan Document

(ITPD)

Version 1.0 (22 January 2017)

Reference professor: Prof. Elisabetta Di Nitto

Authors:

Davide Anghileri matr. 879194

Antonio Paladini matr. 879118

Revision History

Name	Date	Reason for Changes	Version
First complete version			1.0

1 Contents

1 Introduction

This section contains a brief introduction to the **P**roject **P**lan Document.

1.1 Purpose and scope

The purpose of this document is to explain the Project Plan devised for the system to be.
In this document, we are going to evaluate the dimension of the code of the entire project and the **time and effort needed** by our team to realize it. For the estimation of the code size we have used the **Functional Point** approach and for the evaluation of the effort and time needed we have used the **COCOMO II** model.
All the people involved in the project could be considered as possible readers of the document, but the document itself is more of a guide for the Project Manager and the Management in general.
The Project Plan consists in tables, Gantt diagrams, charts and natural language descriptions of the planning, scheduling and management of PowerEnJoy development.

1.2 Definitions Acronyms and abbreviations

1.2.1 Definitions

- Others definitions are in the RASD, DD and ITPD

1.2.2 Acronyms

- DD: Design Document
- RASD: Requirements Analysis and Specifications Document
- ITPD: Integration Test Plan Document
- PP: Project Plan
- PPD: Project Plan Document
- FP: Functional Points
- ILF: Internal Logic File
- ELF: External Logic File
- EI: External Input
- EO: External Output
- EIQ: External InQuiry
- DBMS: DataBase Management System
- API: Application Programming Interface
- GPS: Global Positioning System

1.2.3 Abbreviations

- COCOMO = COnstructive COst Model

1.3 Reference Documents

- RASD produced before 2.1
- DD produced before 2.0
- Assignments AA 2016-2017
- Example of PPD from past year
- Software Engineering 2 course slides
- COCOMO II: Model Definition Manual
- The Function Points complexity evaluation tables.

2 Function points and COCOMO

In this section, we want to provide an estimation of the expected cost, size and required effort of PowerEnJoy application.
For the estimation, we consider the **Function points approach** in order to predict the number of lines of code that the developers will write in Java.
Instead, for the cost and effort estimation we consider the **COCOMO approach**, using as initial value the amount of lines of code computed with the Function Points.

2.1 Functions points

Through the Function Point technique, we are going to assess the effort needed to design and develop PowerEnJoy application.
This technique is based on the assumption that the dimension of a software can be characterized based on the **functionalities** that it has to offer.
The analysis will be based on a combination of the following program characteristics:

- Data structures;
- Inputs and outputs;
- Inquiries;

- External interfaces

The number of function points is determined by splitting them in different **function types** and then multiplying the subtotal by a **complexity weight** previously defined for each of them.
The function points which are going to be considered are:

- Internal Logical File
- External Interface File
- External Input
- External Output
- External Inquiry

Through these five categories is possible to define an external representation of PEJ application.
The estimation is based on the usage of figures obtained through statistical analysis of real projects, which have been properly normalized and condensed in the following tables:

Reference table for Internal Logic Files and External Logic Files

Record Elements	**Data Elements**		
	1-19	20-50	51+
1	Low	Low	Avg
2-5	Low	Avg	High
6+	Avg	High	High

Reference table for External Output and External Inquiry

File Types	**Data Elements**		
	1-5	6-19	20+
0-1	Low	Low	Avg
2-3	Low	Avg	High
4+	Avg	High	High

Reference table for External Input

File Types	**Data Elements**		
	1-4	5-15	16+
0-1	Low	Low	Avg
2-3	Low	Avg	High
4+	Avg	High	High

Reference table for calculation of UFP

Function Types	**Complexity weights**

	Low	Average	High
External Inputs	3	4	6
External Outputs	4	5	7
External Inquiries	3	4	6
Internal Logical Files	7	10	15
External Interface Files	5	7	10

2.1.1 Internal logic file

Internal Logical Files (ILF) are user identifiable group of logically related data that resides entirely within the application boundary. Now we are going to identify each ILF concerned with PEJ application and to define the complexity weight for each of them.

Starting from the users, the system stores basic information about each of them, like username, password, email and so on. Moreover, each user is related to other entities like: *Identity Card*, *Driving License* and *Method of Payment* which are more complex data required in order to perform a registration.

The *User entity* and the other entities related have a *high rate of insertion and reading* because they are used any time a user performs a login procedure or a registration procedure. A user can also modify his personal data by the dedicated area accessible from the main menu of the application, but *modifications* of data are quite *infrequent*.

Any user can reserve a car, when this happens a Reservation is created. The *Reservation entity* is related with other entities like the *Drive Session* which saves information like the date of a drive session and its route, and the *Payment* related to it.

These entities are characterized by a *high rate of insertion* and by *infrequent modifications and readings*.

Then we move to the *Car entity*, which saves basic information like the model, the number of seats and the status (available or not) of a car.

For this entity *insertion are quite infrequent* while *modification and reading are repeated very often;* this is basically because when a user search for a car to reserve it, it is necessary to check in these entities for available cars and their position.

The *Parking Area entity* behaves in a similar way, in fact it has to be read and modified every time someone park a car into it, or when someone takes out a car from it for a drive session. *Instead, insertion of new Parking Areas is very infrequent.*

Finally, we have to analyse the two figures of Maintainer and Administrator.

For the *Maintainer entity* we can repeat a reasoning similar to the one we made for the user. Read-only access to data is required any time a maintainer try to Login. *Modifications are quite infrequent* as the maintainers do not have a personal area through which they can modify their data. Also *insertions are very rare* compared to the rate of insertion of new users.

Every time a user requires the assistance there is an insertion in the *Request Assistance* entity, which later is modified by the maintainers who interview to fulfil the request. So it has a *high rate of insertion and modification.* Also reading are very frequent as any maintainer access to this entity any time he login into the application.

The *Administrator entity* saves very basic information about the administrators like their Name and Username. *This entity has a low rate of insertion, modification and reading.*

ILF	Complexity	FPs
User	High	15
Reservation	Avg	10
Drive Session	Avg	10
Car	Avg	10
Parking Area	Avg	10
Maintainer	Low	7
Administrator	Low	7

Total	69

2.1.2 External interface file

EIF's are user identifiable group of logically related data that is used for reference purposes only. The data resides entirely outside the application and is maintained by another application. The external interface file is an internal logical file for another application.

As we described in the RASD PEJ relies on Google Maps API's in order to create and customize the map and calculate directions between locations.

These functionalities are used during all the drive sessions in order to visualize map on the devices on board of each car. Moreover, they are used each time there is the need to visualize maps and calculate distances, like when a user tries to search for a nearby car through his mobile device.

There are two main kind of interactions:

- Given the coordinates of two locations, get an estimate of the time that is necessary to drive from one to the other
- Given an address, get the correspondent pair of coordinates (reverse geocoding)

Given these considerations we define the weights as follows:

EIF	**Complexity**	**FPs**
Time of Arrival Estimation	Low	5
Reverse geocoding	Low	5
Map data retrieval	Low	5
Total		15

2.1.3 External Input

An External input is an elementary process in which data crosses the boundary from outside to inside. This data may come from a data input screen or another application.

In PEJ we can categorize external inputs in three different group:

Generated by the User

- Registration: a user provides his personal information which are elaborated by *Registration Manager* in order to perform the reservation.
- Login: a user provides his personal information and PEJ elaborate them through *Log-in Manager*
- Password Retrieval: when a user tries to retrieve his password the *Log-in Manager* performs some action which check if he is able to perform the action, and then executes the retrieval.
- Modifications of personal Data: the user can modify some of his personal data providing new information which are elaborated by the *User Manager* which check also if the modifications are actionable.
- Make a reservation: a user is able to perform a reservation by choosing an available car on the map. Once the car is chosen the *Reservation Manager* checks if the reservation is actionable by checking if the user is banned or not and if his method of payment is working.
- Delete a reservation: a user can delete a reservation he has done.
- Pay for a reservation: when a reservation expires the *Payment Manager* work in order to perform the payment by charging the user on the method of payment that he has chosen.
- Make a request of Assistance: A user is able to make a request of assistance by contacting the support of PEJ,

then the information provided are used to create an instance of request of assistance.

Generated by the Maintainer

- Choose a request: once a maintainer has accessed to the list of pending requests of assistance he can decide which of those he want to take care about. The *Request Assistance Handler* simply flag the request as already chosen by a maintainer.
- Fulfill a request: when a maintainer has fulfilled a request communicates it to PEJ which flag it as resolved, through the *RequestAssistanceHandler*.

Generated by the Administrator

- Insertion/Deletion/Update of Parking Areas: an administrator is able to insert a new Parking Area in the system or to update or delete an existing one. All these changes, as also the ones which follow are performed by the *ChangesManager*.
- Insertion/Deletion/Update of Cars: an administrator is able to insert a new Car in the system or to update or delete an existing one.
- Banning/Removals of Users: an Administrator is able to ban a user or to remove him from the system.
- Insertion/Deletion/Update of Maintainers: an Administrator is able to insert a new Maintainer into the system and to delete an existing one.
- Update of Prices Condition: an Administrator is able to modify the pricing system of PowerEnJoy.

EI	**Complexity**	**FPs**
Registration	High	6
Login	Low	3
Password Retrieval	Avg	4

Modification of Personal Data	Avg	4
Make a Reservation	High	6
Delete a Reservation	Low	3
Pay for a Reservation	Low	3
Make a Request of Assistance	Avg	4
Choose a Request	Low	3
FullFill a Request	Avg	4
I/D/U of Parking Areas	High	6
I/D/U of Cars	High	6
Banning/Removals of Users	High	6
I/D/U of Maintainers	High	6
Update of Prices Conditions	Avg	4
Total		68

2.1.4 External output

External output are elementary processes that create data directed to the external environment. These reports and files are created from one or more internal logical files and external interface file.

We are going to list below the operations of this type that we have identified in PEJ application:

- Notify a user about the outcome of the registration procedure.[1]
- Notify a user he is near to the car he has reserved.
- Notify a user the total amount to be payed at the end of a reservation.
- Notify a user he has been banned or removed by the application.

[1] We retain this an external output because the completion of a user's registration is not immediate. Personal data provided by the user have to be analysed before and then PEJ communicates to the user the outcome of the procedure

- Notify a maintainer a new request of assistance was issued.

Now we define the weights of these operations as follows:

EO	Complexity	FPs
Registration outcome	Low	4
Near to a reserved car	Low	4
Notify bans and removals	Low	4
New request of assistance	Low	4
Total		16

2.1.5 External inquiry

An external inquiry is an elementary process with both input and output components that result in data retrieval from one or more internal logical files and external interface files.

In PEJ application there are different type of external inquiry, we will briefly describe them in the following list:

- Reservation of a Car: when a user tries to make a reservation he accesses to a list of all the available cars. Then he can visualize more detailed information about the single car he is interested to. If the user doesn't perform the reservation no internal file is modified.
- Access to Past Reservations: the user can access to a list of all his past reservations.
- Access to Personal Area: the user can access to a personal area in which he can visualize all his personal details
- Requests of Assistance: when a maintainer login into the application the first page he sees contains a list of all the requests of assistance active at the moment.
- Access to Statistics Monitor: an administrator is able to access to a statistics monitor which allows him to retrieve

information about the history of reservations performed by the users.

- Access to a Real-time Monitor: an administrator is able to access to a Real-Time monitor which shows him what's happening at the moment: This means that he can see how many cars are reserved at the moment and by who, how many cars are to be recovered by a maintainer and so on.

Now we have identified all the possible types of external inquiry we define their weights as follow:

EQ	**Complexity**	**FPs**
Reservation of a Car	Avg	4
Access to Past Reservations	Low	3
Access to Personal Area	Avg	4
Requests of Assistance	Low	3
Access to Statistics Monitor	Avg	4
Access to RealTime Monitor	Avg	4
Total		22

2.1.6 Summary

Now we have estimated the number of **Function Points** for each category we sum them up in order to conclude the FP count.

Considering Java Enterprise Edition as a development platform and disregarding the aspects concerning the implementation of the mobile applications (which can be thought as pure presentation with no business logic) we estimate the **SLOC (Source Lines of Code)** by multiplying the total count of FP by an adjusting coefficient.

Function Types	Complexity weights			
	Low	Average	High	**Total**
External Inputs	**4***3=12	**5***4=20	**6***6=36	**68**
External Outputs	**4***4=16	**0***5=0	**0***7=0	**16**
External Inquiries	**2***3=6	**4***4=16	**0***6=0	**22**
Internal Logical Files	**2***7=14	**4***10=40	**1***15=15	**69**
External Interface Files	**3***5=15	**0***7=0	**0***10=0	**15**
Total FP				**190**
Adjusting Coefficient				**55**
SLOC				**10450**

2.2 COCOMO II

In this section, we are going to estimate costs and efforts referred to the developing of PEJ.

We use COCOMO II to perform this analysis. COCOMO II is an evolution of COCOMO 81 (Constructive Cost Model v.1981) which was elaborated with a statistical approach.

COCOMO II takes into account some of the new characteristics of software development activities.

Now will follow estimations of different factors which will used at the end of this section in order to estimate the effort in developing PEJ application in terms of *Person-Months*.

2.2.1 Scale Factors

Scales factors are some of the most important factors which contribute to the duration and cost of the developing of a software.

All of them are used to define the exponent used in the **Effort Equation**.

Follows a reference table with values defined in the COCOMO II for each couple of factor and relative level.

Scale Factors	Very Low	Low	Nominal	High	Very High	Extra High
PREC	thoroughly unprecedented	largely unprecedented	somewhat unprecedented	generally familiar	largely familiar	thoroughly familiar
SF_i:	6.20	4.96	3.72	2.48	1.24	0.00
FLEX	rigorous	occasional relaxation	some relaxation	general conformity	some conformity	general goals
SF_i:	5.07	4.05	3.04	2.03	1.01	0.00
RESL	little (20%)	some (40%)	often (60%)	generally (75%)	mostly (90%)	full (100%)
SF_i:	7.07	5.65	4.24	2.83	1.41	0.00
TEAM	very difficult interactions	some difficult interactions	basically cooperative interactions	largely cooperative	highly cooperative	seamless interactions
SF_i:	5.48	4.38	3.29	2.19	1.10	0.00
PMAT	The estimated Equivalent Process Maturity Level (EPML) or					
	SW-CMM Level 1 Lower	SW-CMM Level 1 Upper	SW-CMM Level 2	SW-CMM Level 3	SW-CMM Level 4	SW-CMM Level 5
SF_i:	7.80	6.24	4.68	3.12	1.56	0.00

Precedentedness

This factor considers our rate of experience in developing of similar projects. As we have previous experience using Java SE for small-size projects, but we had never used Java EE for developing a large-size application and we never experienced developing of mobile applications, we decide this parameter to be **Low.**

Development Flexibility

This factor estimates how much we have to conform to pre-established requirements and external interface specs.

Since we have to respect general requirements and functionalities without any external interface specifications we set this parameter to **High**.

Risk Resolution

This factor is used to define how much our application will be solid once completed, this mean how much precise is our risk management plan and how much we are prepared to face possible unexpected problems. Since we have followed a strict and organised phase of designing PEJ application, including the risk analysis which in section 5 of this document, we set this parameter to **Very High**.

Team Cohesion

The Team Cohesion scale factor accounts for the sources of project turbulence and entropy due to difficulties in synchronizing the members of the group. Since the members of our group as already worked together without problems, and we agree on mostly every developing choice we have encountered until now this value is **Very High.**

Process Maturity

Since we didn't have big problems during the development of the project, we set this value to **Level 4**.

Finally, we sum up all the values identified in order to calculate the

Scale Factor	Level	Value
Precedentedness	Low	4.96
Development Flexibility	High	2.03
Risk Resolution	Very High	1.41
Team Cohesion	Very High	1.10
Process Maturity	Level 4	1.56
Total		11.06

exponent **E** considered in the Effort Equation.

$$E = B + 0.01 \times \sum_{1<=j<=5} SF_j, \text{ where } B = 0.91$$

E=0.91+0.01*(11.06) **= 1.02**

2.2.2 Cost Drivers

Follows the cost driver analysis for the case of post-architecture.

Product Factors

- **Required Software Reliability (RELY):** This is the measure of the extent to which the software must perform its intended function over a period of time. No type of failure should risk human life but some could bring to high financial loss, then we set this value to **high.**
- **Data Base Size (DATA):** This measure attempts to capture the affect large data requirements have on product development. The rating is determined by calculating Database size (bytes)/Program size (SLOC). We estimated the dimension of our DB around 2 GB, then we set this driver to a **very high** value.
- **Product Complexity (CPLX):** The product complexity is calculated by analysing five different areas: control operations, computational operations, device-dependent operations, data management operations, and user interface management operations. Making an average of the complexity in these areas we chosen **high.**
- **Developed for Reusability (RUSE):** This cost driver accounts for the additional effort needed to construct components intended for reuse on the current or future projects. Since we decided to don't develop component reusable in other application this value is set to **low.**
- **Documentation Match to Life-Cycle Needs (DOCU):** This cost driver is evaluated in terms of the suitability of the project's documentation to its life-cycle needs. Since

the standard level of documentation is required for this driver we set a **nominal** value.

Platform Factors

- **Execution Time Constraint (TIME):** This is a measure of the execution time constraint imposed upon a software system. For PEJ this value should be **high.**
- **Main Storage Constraint (STOR):** This parameter describes the expected amount of storage usage with respect to the availability of the hardware. As current disk drives can easily contain several terabytes of storage, this value is set to **nominal**.
- **Platform Volatility (PVOL):** For what concerns the core system, we don't expect our fundamental platforms to change very often. However, the client applications may require at least a major release once every six months to be aligned with the development cycle of the main mobile operating systems. For this reason, this parameter is set to **nominal**.

Personnel Factors

- **Analyst Capability (ACAP):** This cost driver aims to measure the analysts' analysis and design ability, efficiency and thoroughness, and their ability to communicate and cooperate. This value is set to **nominal.**
- **Programmer Capability (PCAP):** This cost driver is focused on evaluating the capability of the programmers as a team. Since our group is very. As our group has already worked together for programming tasks we set this parameter to **high.**
- **Personnel Continuity (PCON):** The rating scale for PCON is in terms of the project's annual personnel turnover. We think people working on the project will be

more or less the same until the end of the project, so this driver is set to **very high.**

- **Applications Experience (APEX):** This rating is dependent on the level of applications experience of the project team developing the software system or subsystem. We have some experience in the development of Java applications, but we never tackled a Java EE system of this kind. For this reason, we're going to set this parameter to **low**.
- **Platform Experience (PLEX):** We don't have any experience with the Java EE platform, but we have some previous experience with databases, user interfaces and server side development. For this reason, we're going to set this parameter to **nominal**.
- **Language and Tool Experience (LTEX):** This is a measure of the level of programming language and software tool experience of the project team developing the software system or subsystem. Since we started programming in java 2 years ago and we have general knowledges over the used tool, these driver is set to **nominal.**

Project Factors

- **Use of Software Tools (TOOL):** for developing of PEJ we will make use of mature life-cycle tools and moderately integrated, then this driver is set to **high.**
- **Multisite Development (SITE):** The team is in average fully located, so the chosen level is **nominal.**
- **Required Development (SCED):** This rating measures the schedule constraint imposed on the project team developing the software. We have not a particular constraint oppression, then this driver is set to **nominal.**

Follows a summary of the cost drivers identified with the relative values:

Scale Factor	Level	Value
RELY	High	1.10
DATA	Very high	1.28
CPLX	High	1.17
RUSE	Low	0.95
DOCU	Nominal	1.00
TIME	High	1.11
STOR	Nominal	1.00
PVOL	Nominal	1.00
ACAP	Nominal	1.00
PCAP	High	0.88
PCON	Very high	0.81
APEX	Low	1.10
PLEX	Nominal	1.00
LTEX	Nominal	1.00
TOOL	High	0.90
SITE	Nominal	1.00
SCED	Nominal	1.00
Product of all cost drivers		1.23

2.2.3 Effort Equation

Now we calculate the total effort required for the developing of PEJ application. It is measured in Person-Months and is calculated as follows:

$$PM = A \times Size^{E} \times \prod_{1<=i<=n} EM_i$$

Where:

- **A=2.94** → This value approximates a productivity constant in PM/KSLOC (Person-Months/Kilo-Source Lines of Code)
- **Size=10.45** → is the estimated size of the project in KSLOC which we calculated above through the *Function Points*.
- **EM=1.23** → is the Effort Multiplier calculated above with the method of the *Cost Drivers*.
- **E=1.02** → is an aggregation of the five *Scale Factors* and we calculated it in section 2.2.1.

In conclusion:

$PM=2.94*10.45^{1.02}*1.23=39.61$

2.2.4 Schedule estimation

We are going to estimate the final schedule with the following formula:

$Duration= 3.67 * Effort^{F}$

Where:

- 3.67 is a coefficient defined in COCOMO II
- Effort is the PM value calculated above
- F= 0.28+0.2*(E-1.01) → with E we refer to the E value considered in the Effort Equation in the previous paragraph.

Then:

F=0.28+0.2*(1.0206-1.01) = 0.28212

In conclusion:

$Duration=3.67*39.61^{0.28212}$=10.36 months

3 Project tasks and scheduling

3.1 Project tasks

Several **tasks** have been identified in our project. They are summarized all together in the following table, which associates a label, a description and a completion state to each task:

Task	Description	Completed
T1a	Reading Assignment	100%
T1b	RASD - Introduction	100%
T1c	RASD - Assumptions	100%
T1d	RASD - Requirements	100%
T1e	RASD - UML	100%
T1f	RASD - Alloy	100%
T1g	RASD - Refinement	100%
T1h	RASD - Presentation	100%
T2a	DD - Introduction	100%
T2b	DD - Architectural Design	100%
T2c	DD - Algorithm	100%
T2d	DD - Refinement	100%
T2e	DD - Presentation	100%
T3a	ITPD - Introduction	100%
T3b	ITPD - Integration Strategy	100%
T3c	ITPD - Individual Steps	100%
T3d	ITPD - Refinement	100%
T4a	PP - Introduction	100%
T4b	PP - Function Points	70%
T4c	PP - Gantt	50%
T4d	PP - Refinement	0%
T5	Final Presentation	0%
T6	Development	0%
T7	Unit Test	0%
T8	Integration Testing	0%
T9	System Testing	0%
T10	User Acceptance	0%
T11	Release to market	0%

3.2 Tasks scheduling

In this section is solved the **scheduling** problem for the identified tasks. We provide two tables and a diagram in order to have the most effective visualization of the scheduling.

3.2.1 Tasks, durations and dependencies

The following table summarizes for each task identified:

- The **effort** needed to perform such task (in terms of [person • hour])
- The dependencies of the specific tasks to other tasks.

Task	Description	Effort	Dependencies
T1a	Reading Assignment	32	
T1b	RASD - Introduction	18	T1a
T1c	RASD - Assumptions	10	T1a
T1d	RASD - Requirements	64	T1c
T1e	RASD - UML	26	T1d
T1f	RASD - Alloy	18	T1e
T1g	RASD - Refinement	6	T1d
T1h	RASD - Presentation	12	T1g
T2a	DD - Introduction	14	T1h
T2b	DD – Architectural Design	80	T2a
T2c	DD - Algorithm	12	T2a
T2d	DD - Refinement	10	T2c
T2e	DD - Presentation	12	T2d
T3a	ITPD - Introduction	8	T2e
T3b	ITPD - Integration Strategy	24	T3a
T3c	ITPD - Individual Steps	32	T3b
T3d	ITPD - Refinement	8	T3c
T4a	PP - Introduction	8	T3d
T4b	PP - Function Points	24	T4a
T4c	PP - Gantt	12	T4b
T4d	PP - Refinement	8	T4c
T5	Final Presentation	32	T4d
T6	Development	340	T5
T7	Unit Test	120	T6
T8	Integration Testing	80	T7
T9	System Testing	64	T8

T10	User Acceptance	48	T9
T11	Release to market	60	T10

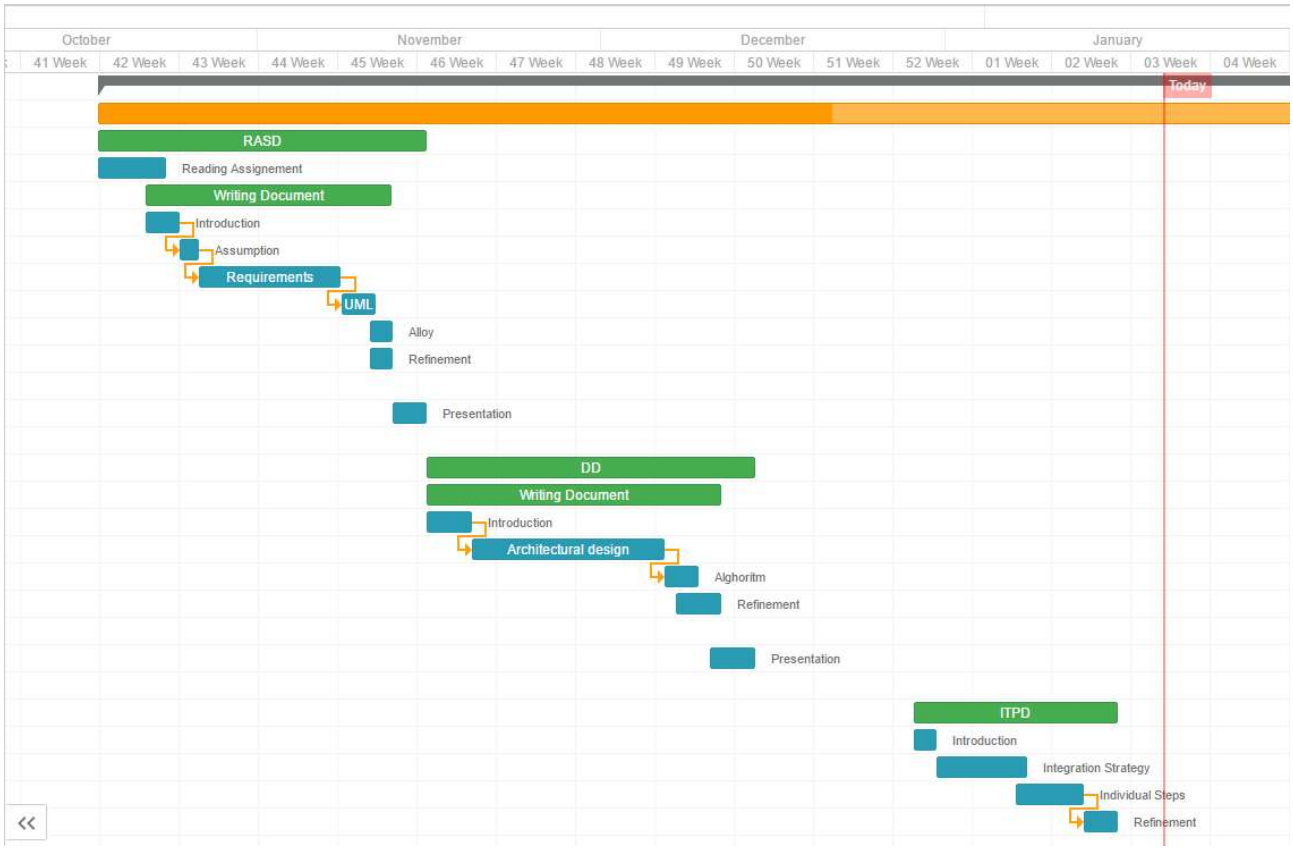
Then, for each task, are identified:

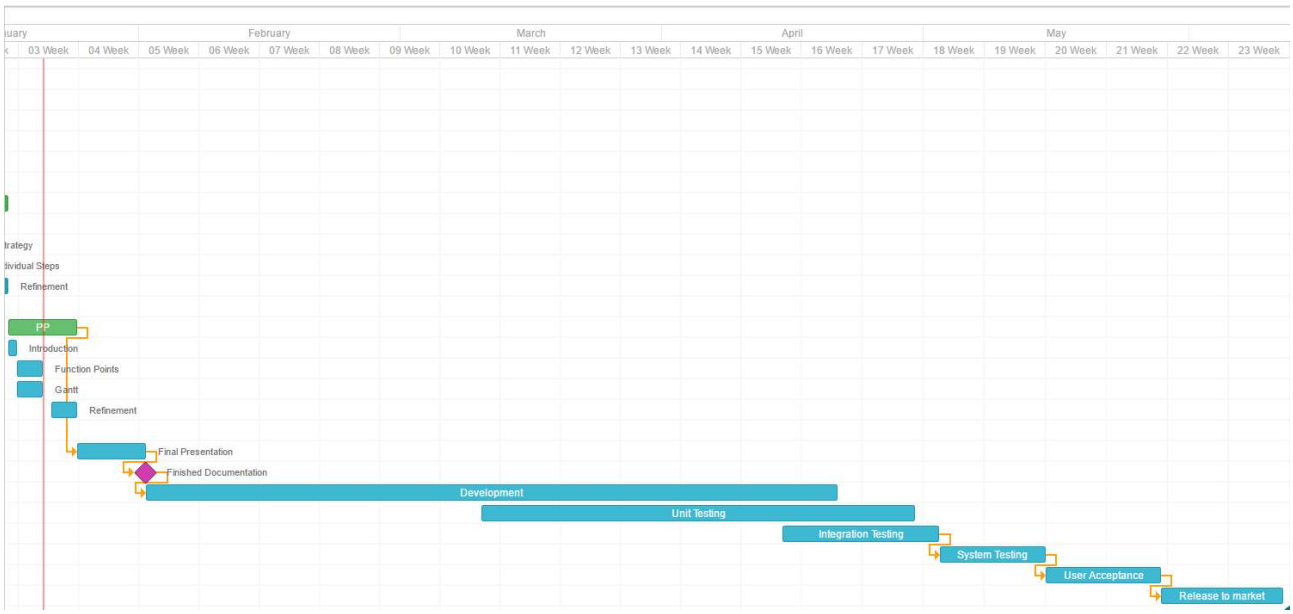
- The date in which the given task starts,
- The date in which the given task ends,
- The interval in [day] that separates the starting date from the ending date.

Task	Start	End	Interval
T1a	16/10/2016	22/10/2016	6
T1b	21/10/2016	24/10/2016	3
T1c	24/10/2016	26/10/2016	2
T1d	25/10/2016	6/11/2016	12
T1e	07/11/2016	10/11/2016	3
T1f	09/11/2016	11/11/2016	2
T1g	09/11/2016	11/11/2016	2
T1h	11/11/2016	14/11/2016	3
T2a	14/11/2016	18/11/2016	4
T2b	18/11/2016	5/12/2016	17
T2c	05/12/2016	08/12/2016	3
T2d	06/12/2016	10/12/2016	4
T2e	09/12/2016	13/12/2016	4
T3a	27/12/2016	29/12/2016	2
T3b	29/12/2016	9/01/2016	6
T3c	05/01/2017	08/01/2017	3
T3d	11/01/2017	19/01/2017	8
T4a	14/01/2017	15/01/2017	1
T4b	15/01/2017	18/01/2017	3
T4c	15/01/2017	18/01/2017	3
T4d	19/01/2017	22/01/2017	3
T5	22/01/2017	30/01/2017	8
T6	30/01/2017	21/04/2017	80
T7	10/03/2017	30/04/2017	50
T8	14/04/2017	01/05/2017	18
T9	02/05/2017	24/05/2017	12
T10	15/05/2017	28/05/2017	13
T11	28/05/2017	11/06/2017	14

3.2.2 Gantt diagram

Here we have the **Gantt Diagram** showing the schedule that we have defined in the previous paragraphs (Since the diagram was too large, we've divided it into 2 images).





4 Resources allocation

This section covers the problem of allocating the human resources to each task to respect the identified scheduling.
Since the fact that we are a group of only two people, and we have only few concurrency in the tasks that we identified, **we have worked together** on the same task at the same time during the whole duration of the project.
The only exceptions concern the following tasks:

During the writing of the RASD:

Task	Resource
Alloy	Antonio
Refinement	Davide

During the writing of the DD:

Task	Resource
Algorithm	Davide
Refinement	Antonio

5 Project risks

In order to have a global Proactive Risk Management Strategy we decided to identify all the risks that in our opinion could affect the PowerEnJoy application.
Also, for each of them we estimate the probability that it will occur and the impact that it will have on the project if it does occur.

5.1 Risk Types

All the risk types identified are here reported:

- Project Risks

1. Members of the team are ill during the project.

   - Probability: High
   - Effect: Moderate
   - Management: Organize the team so that also with a maximum of 1 person ill, the team can keep on with the project.

2. The work force available is not enough to satisfy the tasks

   - Probability: Moderate
   - Effect: Moderate
   - Management: consider and provide extra time during the scheduling phase to prevent the risk to be late.

3. The developer team is not able to perform some high-complexity tasks

   - Probability: Low
   - Effect: Moderate
   - Management: consider the possibility to change something during the development phase.

- Law Risks

1. The Legislation regarding the use of electric car, the use of device during the driving or the sharing cars may change

   - Probability: Moderate
   - Effect: Serious
   - Management: Forecast if something is changing in the legislation or in the culture of the people and be prepared to react to this changes if necessary

- Technical Risks

1. The External software components used in the application cannot process as many operations per second as expected.

   - Probability: Moderate
   - Effect: Serious
   - Management: Overestimate the operations throughput in or-der to choose the appropriate support infrastructure.

2. The loss of all or a big part of the code of the project

   - Probability: Low
   - Effect: Catastrophic
   - Management: Use distribute technique to develop the project and use a backup system.

- Business Risks

1. The project, once developed, is significantly different from what was required

   - Probability: Low
   - Effect: Catastrophic
   - Management: Design and maintain an accurate version of the RASD document accepted by our customers.

2. The project, once developed, comes out that is not easily maintainable

   - Probability: Moderate
   - Effect: Catastrophic
   - Management: Design and develop the project in a structured way with a high level of detail

3. The application doesn't reach a desired level of users

- Probability: Moderate
- Effect: Serious
- Management: Use advertisement to make the application more popular and so to increase the number of users

4. The user interfaces developed are not user-friendly

- Probability: Low
- Effect: Moderate
- Management: Follow the guidelines provided by the designers for developing modern, thin and usable UIs.

5. The company decides to reduce the budget for the development of PowerEnJoy

- Probability: Low
- Effect: Serious
- Management: Prepare a briefing document for management showing how cutting the budget for the project would not be cost-effective.

6. A developer of the application decides to stop the development for personal reasons.

- Probability: Low
- Effect: Serious
- Management: Investigate the possibility of hiring new team members.

6 References

6.1 Used Tools

To draw up this document we've used the following tools:

- Microsoft Office World 2016: to redact and format this document
- Paint: to create and adapt the images of this document
- GitHub: to save and control the version of the document
- GanttPRO: to design and to redact the Gantt Diagram (https://app.ganttpro.com)

6.2 Hours of work

Date	Antonio's hours	Davide's hours
2017/01/16	4h	4h
2017/01/17	--	3h
2017/01/19	3h	--
2017/01/20	6h	6h
2017/01/22	2h	2h
Total	**15h**	**15h**

Hours for review:

Date	Antonio's hours	Davide's hours
Total		