MS Project-like System

(Basic PM)

Final Project Report

CMPE 412

Team members: Elmehdi, Charidi, 149164

Muyiwa, Akinpelu, 147704

Hasan, Soygazi, 123379

Serhat, Kocagoz, 131113

Naim, Al Tarabichi, 137823

Aliza, Wasim, 137843

Abdulrahman, Zurghani, 147592

Waseem, Nasir, 147878

Saidu, Sokoto, 147912

Muhammad, Saleh, 15701299

Computer Engineering Department

Eastern Mediterranean University

May 2017

ABSTRACT

The main aim of this project is the design and implementation of a project management application that is similar to Microsoft Project. An application that is similar in functionality but easier to use. In order to achieve this a detailed analysis of the requirements set by the client was done and way of implemented them where devised. It was concluded, as a group, that the best software engineering model to use is the water fall model, because the requirements were well understood. At the end after months of working, a working prototype was made, which has a seamless user interface and at the same time satisfying all the requirements of the client in addition to other features deemed suitable to be added. These requirements include, but are not limited to, having task names, predecessors, start dates, duration, Gantt charts, network diagrams, resources pool and organizations charts.

Keywords: Tasks, time, cost, calendar, resource.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The main objective of this project is to develop a product where students can easily plan and manage their projects. The product provides more control and hence saving both time and money. When there are tasks to be achieved and the need to keep up with deadlines, our product will definitely be helpful. One can track time, people or money all through this product using the Gantt chart. It is mainly for CMPE/CMSE students to use free of charge. This application software will be different from other project such as MS project in that it will have a more simplified user interface incorporating and laying out the most important and used features used by the students in the CMPE/CMSE department of EMU. In addition to this many of functions and features that are commonly used will be automated by default, unlike other software packages in which changes have to be manually made. Due to its simplicity, it will be easier to learn and use compared other project management applications. By making almost all features automatic, it will also save time and consequently cost.

2. REQUIREMENTS ANALYSIS

2.1 Functional Requirements

2.1.1 Portability and storage

Description:

When developing the system, we used Dataset and DataTables rather than Database, thing which will make our .exe file not heavy. If we used SQL database, we should include it in the deployment which will make the size of our project very large. In addition to that, using SQL database will force all the project and data to be stored in one database, which will make life harder if we want to copy a project from one laptop to another one (we should copy the whole database and link it to the other application in the other laptop). In our case, once the user open the file, all the data taken from the file will be stored in the Dataset which exist in the application.

Priority: High

Stimulus/Response Sequences:

When the use click on "create project" the application will ask him/her to fill a form which has some important information about the project he/she wants to create (starting time, directory of the project, name of the project, etc.). After filling the form successfully, the application creates a file with extension (.ale) in with the given name in the given directory.

Functional Requirements:

REQ-1: The user must be able to create a project file in any location.

REQ-2: The user must be able to open a project from any laptop which has the application.

REQ-3: The user must be able to save any changes he/she makes while working in a project.

REQ-4: The application must provide a form which must be filled by the user which has the necessary information about the project.

2.1.2 Automatic calculations and smartness:

Description:

when dealing with the Tasks, the user does not have to mention the finish time, he/she should just specify the starting and the duration and type of duration (Hour-Day-Week-Month) and automatically the finishing time is computed and placed in its place, the application consider that the duration typed is in Working days, our application is smart enough to detect Saturday and Sunday, and it does not allow the user to start a task at those days.

The application is able to calculate the cost of each task by referring to the price for the resources and see how long does this task require.

Priority: Medium

Stimulus/Response Sequences:

The user starts filling the grid for the tasks, when he/she comes to the duration, he/she must write a valid duration (integer value), then he/she puts a starting time. After this, he/she can only click on the cell for finishing time, then automatically the value for this cell will be calculated. Same thing implies for calculation the cost. The user must only to choose the corresponding resource for the task, and automatically the cost for the selected task will be calculated and placed in its place.

Functional Requirements:

REQ-1: The user must be able to add tasks to the project.

REQ-2: The application must be able to hold as many as tasks the user wants to have.

REQ-3: The user must be able to add resources to his/her project and assign tasks to them.

REQ-4: The application must be able to calculate the cost for each task, the finishing time for each task and distinguish between a working day and weekend.

2.1.3 Gantt chart:

Description and priority:

Representing the tasks in a Gantt chart with their starting and finishing time, which will make things more clear.

Priority: High

Stimulus/Response Sequences:

Once a task is fully filled in a row in the grid corresponding to the tasks, when leaving that row, the Gantt chart is automatically updated.

Functional Requirements:

REQ-1: The Gantt chart must be updated according to the data entered by the user.

REQ-2: The user must be able to create/delete a task and see those things in the chart.

REQ-3: The user must be able how much is done in each task in the chart.

2.1.4 Activity Network Diagram:

Description and priority:

A tool which shows tasks as a diagram, different look of the Gantt chart. It also gives a better understanding of how tasks are related to each other.

In addition, it shows the critical paths in a different colors.

Priority: High

Stimulus/Response Sequences:

The user must first enter some tasks information in the grid corresponding to the tasks, then he/she can easily click on the button (activity network) to be able to see the network diagram.

Functional Requirements:

REQ-1: The diagram must show the critical paths.

REQ-2: The user must be able to see how tasks are related.

REQ-3: The user must be able to export the diagram as an image.

REQ-4: The user must be able to zoom in and zoom out in the chart.

2.1.5 Organization chart:

Description and priority:

A chart which shows how the structure of the team.

Priority: High

Stimulus/Response Sequences:

The user must first fill the resources grid, then give some ranking data, specifies the teams and so on.

Functional Requirements:

REQ-1: The user must be able to make as much as teams he/she wants.
REQ-2: The user must be able to see an organization chart in a proper way.
REQ-3: The user must be able to export the diagram as an image.
REQ-4: The user must be able to zoom in and zoom out in the chart.

2.2 Non-Functional Requirements

2.2.1 Security Requirements

- The PC on which the software will reside will have its own security depending on the preferences of the owner. Only the owner, or those given permission will have physical access to the computer and the software's on it (including Basic PM)
- For the encryption of user data, a static key will be securely generated that will only be known to all Basic PM application
- This key will be used to encrypt and decrypt any file that has the. ale extension to prevent other application from having access to a user's data.
- When an existing project is trying to be opened, the key will be used to decrypt the data to retrieve the data and them immediately encrypt it again to prevent unauthorized access
- Whenever a project is saved either manually or automatically it will immediately be decrypted to allow writing and then immediately decrypted afterwards.
- This will ensure the protection of user data by preventing unauthorized access to user data.

2.2.2 Performance Requirements

- The application must be interactive and responsive

- Any delay must be minimal.
- Immediately a new task is added, the corresponding Gantt chart and network diagrams should be immediately visible.
- Immediately the date and duration of a new task is added, the corresponding end date should be computed excluding working days.
- Error messages should be displayed immediately an error occurs.
- Data access should be instantaneous to ensure smooth operations

2.2.3 Safety Requirements

- The user should maintain a backup of the local file with (. ale extension) on an external storage device or on the cloud (e.g. Google Drive, OneDrive, Drop Box etc.)
- To ensure the safety of user data, the system should maintain a good backup: the database must be secured so that it can be easily restored in case of a failure
- On the part of the system, there is a mechanism in place to automatically save the user data every one (1) minute while its active. As a result of this the user does not have to manually save the work.
- In case of a power failure, a temporary file will be created in which the user will be able to retrieve and continue his work from where he stopped at least one minute before the power failure.

2.2.4 Usability

- The platform should be user friendly.
- The graphical user interface should be well informed.
- There should be a way for a user to find help immediately in case of a problem.
- The user must be satisfied.

Security: How difficult should it be for people to hack the system? Reliability: How often will the system be allowed to fail or be unavailable? Usability: How easy should it be for users to use the system? Accessibility: Should people with disabilities be able to use the system? What are the requirements for evolution of the system, such as testability, maintainability, extensibility and scalability?

2.3 Realistic constraints

- Economic

The product is economically feasible, and any individual can use the system without a lot of investment. Using this project for proper planning will help one to prepare for any unfortunate events, and hence saving both time and money. With a methodical project planning that this product will provide, a successful business is achievable, and thus, helping the economy of North Cyprus and other countries. In addition to this, because our software will be free; any organization, state or country that decides to adopt it will save a lot of money.

- Environmental

The use of this product does not consume a lot of power, it only requires an adequately charged system. Using this product does not lead to pollution of the environment due to the fact that the product is used in the confinements of people`s houses or offices.

- Social

The project is socially feasible, and all age-groups can use this product for various project-related purpose provided that they are computer literate. The product has simplified the Microsoft Project features, so due to this factor, the product gives inspiration to students who might want to do software projects because of its ease of use.

- Political

There are no political constraints related to this product, therefore it is politically feasible.

- Ethical

There are no ethical constraints that must be observed in the use of this product.

- Health and safety

Using this product is very safe and it doesn't endanger the health and safety of the society because it is mainly going to be used for official purpose like project management and scheduling.

- Manufacturability

The source code is available and the application can be installed on any number of systems.

- Sustainability

The product can be used over a long term because it is user-friendly and easy to use for specific individuals that finds it difficult to understand Microsoft Project features and its functionalities.

2.4 Ethical issues

There is no ethical issue related to this product. Furthermore, the product cannot be used for any unprofessional or criminal activity due to its nature, because it is primarily used for project management and planning.

3. DESIGN

3.1 High level design (architectural)

The software we coded was designed to work with three parts, these parts are the basic elements of most running programs.

First is the User Interface (UI), the UI was designed to be simple and self-explanatory, it follows the same pattern of most project management software's so users can adapt to use fast and easy, when the program is opened it is programmed to give the use an interface where he can start a new project, open an already created project and an exit option, when starting a new project a simple form will appear asking the user to name the project and give start/end dates, and can enter an additional information and project description, the next screen will show the user the standard view, where user will see the task list where he can enter the details, and on the left side he will have the Gant-chart which also shows the critical path.

In the code, the code starts with creating a file for the project where it stores the information, now in this software no database was used, since there is not that much of complex data relations that require using data base, the code can calculate working days and days of slack and lag, also operates on resources.

The file has a specified structure where all project information is stored in systemic way that it can be reliably stored retrieved and edited.

3.2 Low level design (components used)

The UI essentially consists of the DataGridView and the Gantt chart, DataGridView is were user can add a task by entering its information name, duration, duration type, start and end time, predecessor, and other specific information, after the user have add the task to DataGridView, it will automatically be added to the Gannt-Chart, the Gannt-Chart will represent the duration of the task, and will link it to tasks before and after it, the critical path has red color to differ it from the rest of the path.

When designing the UI the "**Metro Modern UI - Metro Framework 1.4.0**" since it provides a different design and graphic elements that are not available on the standard UI provided in .NET framework. The UI elements were connected to the C# code and programmed to invoke events. The network diagram was drawn using tool named **GLEE,** provided by Microsoft and its free for use.

The language the software was coded in was C#, the first thing was representing tasks in the proper data structure to preform operation on them. with the help of "**guncharts dll**" which had the support of tasks, and tasks related operation with added benefit of supporting the Gann-chart and its representation, instead of using a database a data table was used as a data structure.

The code will save the project in an encrypted file that has a specific format, the format has the following structure, file is divided into sections:

- First line contains the date and time of file creation.
- Second line contains the country where the project is taking part.
- Third line specifies the county's currency.
- Fourth line stats the number of working hours.
- Fifth line starts the first section which is table1 that holds task information.
- Second section is the table of resources.
- Third section is the table hold the relation between tasks and resources.
- Fourth section holds the dependencies of tasks.
- Fifth section hold the Gannt-Chart information.
- Sixth section is simply the data in the DataGridView.
- Last the document ends with the word "Finish" that indicates the end of the file.

4. IMPLEMENTATION

4.1 Tools, technologies and platforms used

MYSQL

MS Project

Adobe Photoshop CS6

Visual Studio 2015

MS Word

C#

.NET framework

4.2 Use of Software Engineering Process Steps

4.2.1 Project planning and management

a. Project Organization Activities & Roles and Responsibilities Activities:

We draw the Organization chart to indicate the roles and responsibilities of each person, the relationships of the project team and the coordination of project activities. The diagram is shown as follows:

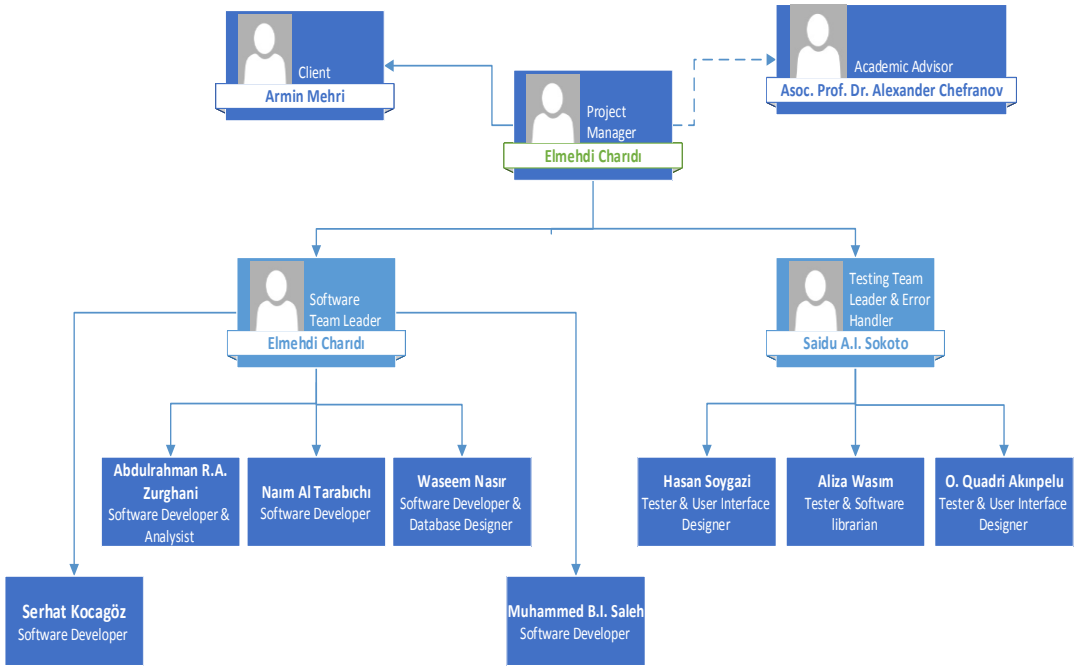

Figure 1. Organization Chart

b. Managerial Process Plan Activities:

The reason for starting this project as stated earlier is to simplify project management. This project like other projects has some risks which by using risk management plan we try to identify different risks and prepare an alternative plan for the risky parts of the project to minimize the effects of these risks on the project. List of risks are shown in table below:

Table 1. Risk Management List

Risk	Probability	Effects	Your Strategy
Exchange rate variability costs are incurred in foreign currencies	Moderate	Tolerable	Make use of two currencies (i.e. US dollars and Turkish lira) and use

exchange rates can have a dramatic impact.			the most stable one for calculations
Resource performance issues Resources who perform below expectations.	Moderate	Tolerable	Use more efficient resources and have backup resources.
Project team misunderstand requirements	High	Serious	Improve communication, explain clearly, and speak with the client.
Design lacks flexibility A poor design makes change requests difficult and costly	Moderate	Serious	Organize design carefully and plan a useful and flexible design
Decision delays impact project	Moderate	Serious	Establish guidelines for decision turnaround time.
Technical change impacts project. A technology innovation changes might impact the project.	Moderate	Insignificant	Ensure all team members are aware and knowledgeable of the technology being used.
The expected yield is not obtained from the database used.	Moderate	Serious	Investigate the use of a higher performance database.

c. Estimation Plan Activities, Work Plan Activities and Schedule Allocation:

We use Gantt chart to define the work distribution and resources allocation. Also in this part we draw activity networks diagram and find critical path networks to apply PERT analysis and Crashing to try to reduce the completion time. Mentioned diagrams are shown as follows:

- **Gantt chart**

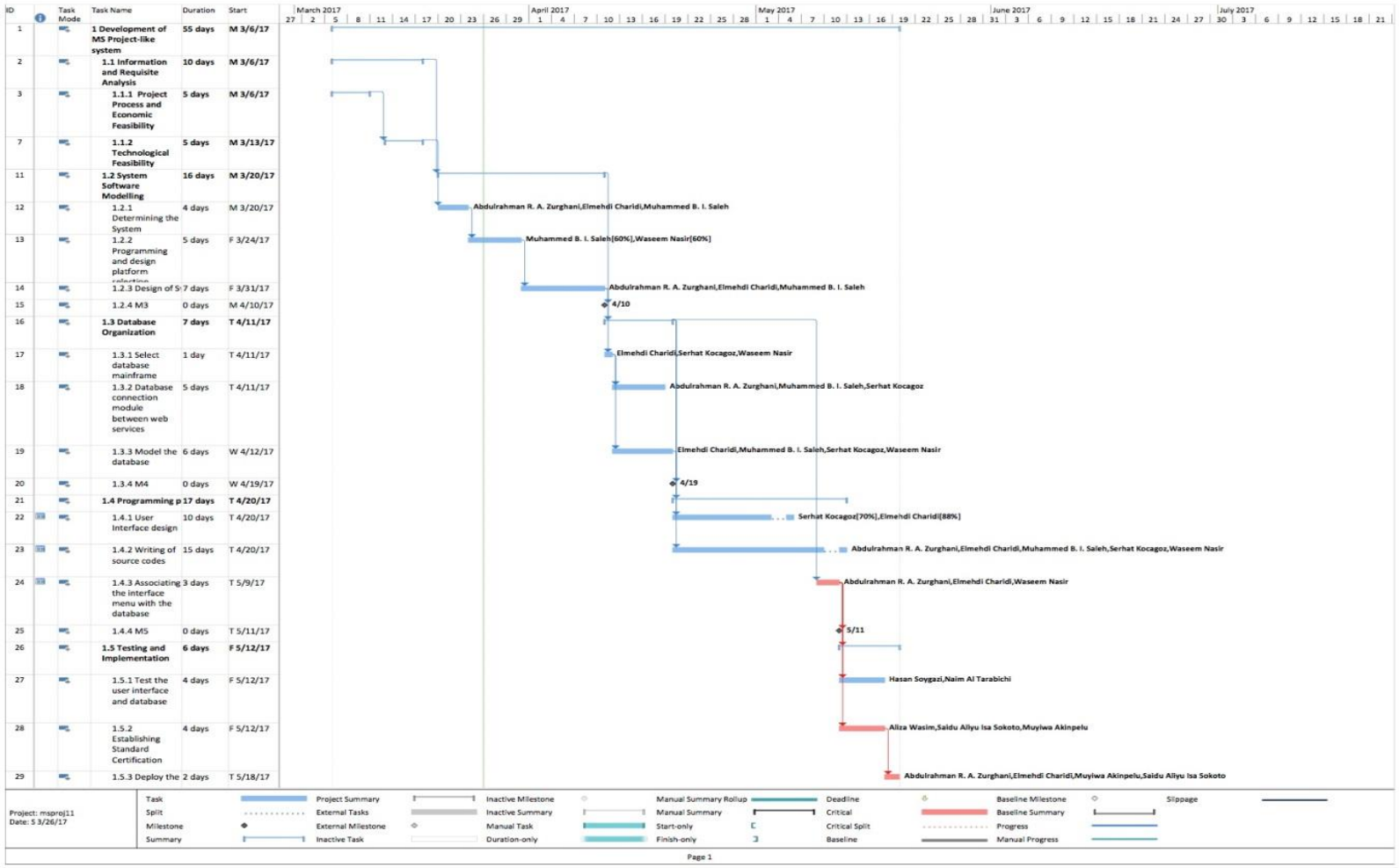

Figure 2. Gantt chart

- **Resource Allocation**

RESOURCE SHEET

	Resource Name	Type	Material	Initials	Group	Max.	Std. Rate	Ovt. Rate	Cost/Use	Accrue	Base	Code	Add New Column
1	Muyiwa Akinpelu	Work		MA	CMPE	100%	10.00 ₺/hr	15.00 ₺/hr	10.00 ₺	Prorated	Standard		
2	Elmehdi Charidi	Work		ES	CMPE	100%	10.00 ₺/hr	15.00 ₺/hr	10.00 ₺	Prorated	Standard		
3	Saidu Aliyu Isa Sokoto	Work		SA	CMPE	100%	10.00 ₺/hr	15.00 ₺/hr	10.00 ₺	Prorated	Standard		
4	Hasan Soygazi	Work		HS	CMPE	100%	10.00 ₺/hr	15.00 ₺/hr	10.00 ₺	Prorated	Standard		
5	Waseem Nasir	Work		WN	CMPE	100%	10.00 ₺/hr	15.00 ₺/hr	10.00 ₺	Prorated	Standard		
6	Naim Al Tarabichi	Work		NA	CMPE	100%	10.00 ₺/hr	15.00 ₺/hr	10.00 ₺	Prorated	Standard		
7	Aliza Wasim	Work		Aw	CMPE	100%	10.00 ₺/hr	15.00 ₺/hr	10.00 ₺	Prorated	Standard		
8	Abdulrahman R. A. Zurghani	Work		AZ	CMPE	100%	10.00 ₺/hr	15.00 ₺/hr	10.00 ₺	Prorated	Standard		
9	Serhat Kocagoz	Work		SK	CMPE	100%	10.00 ₺/hr	15.00 ₺/hr	10.00 ₺	Prorated	Standard		
10	Muhammed B. I. Saleh	Work		MS	CMPE	100%	10.00 ₺/hr	15.00 ₺/hr	10.00 ₺	Prorated	Standard		
11													
12													
13	C#	Material		C			0.00 ₺		4,000.00 ₺	Prorated			
14	MySQL	Material		M			0.00 ₺		1,500.00 ₺	Prorated			
15	MS Project	Material		M			0.00 ₺		2,200.00 ₺	Prorated			
16	Oracle NetBeans 7	Material		O			0.00 ₺		900.00 ₺	Prorated			
17	Adobe Photoshop CS6	Material		AP			0.00 ₺		2,500.00 ₺	Prorated			
18	Visual Studio 2015	Material		V			0.00 ₺		0.00 ₺	Prorated			
19	Microsoft Visio	Material		MV			0.00 ₺		0.00 ₺	Prorated			
20													
21													
22	Microsoft	Cost		MS						Prorated			
23	Travel	Cost		T						Prorated			
24	Local Service Acquis	Cost		L						Prorated			

Figure 3. Resource Allocation

Step1. Project work break down

Table 2. Project work break down

Activity ID	Task Name	Duration (day)	predecessors
a	Development of MS project-like system	1	-
b	Information and Requisite Analysis	10	-
c	System Software Modelling	16	b
d	Database Organization	7	c
e	Programming phase	17	d
f	Testing and Implementation	6	e

Step2. Network Diagram

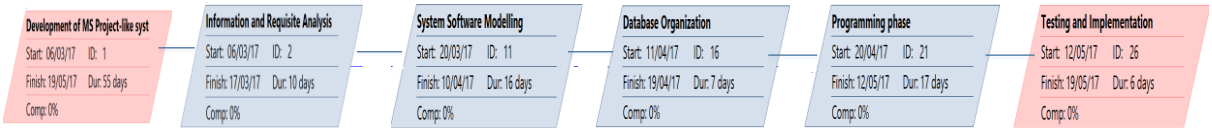

Figure 4. Network diagram

Step3. Calculate the project completion time

Table 3. Project completion time

Paths	Path duration
abcdef	**56(critical path)**

Step4. Calculating expected task times

$$\text{Expected time} = \frac{\text{optimistic} + 4(\text{realistic}) + \text{pessimisti c}}{6} \qquad (1)$$

Table 4. Expected task time

Activity ID	Optimistic time	Realistic time	Pessimistic time	Expected time
a	0	1	2	1
b	7	10	13	9.6
c	14	16	19	15.8
d	6	7	10	7.16
e	15	17	19	16.8
f	5	6	9	6

Step5. Network diagram with expected activity times

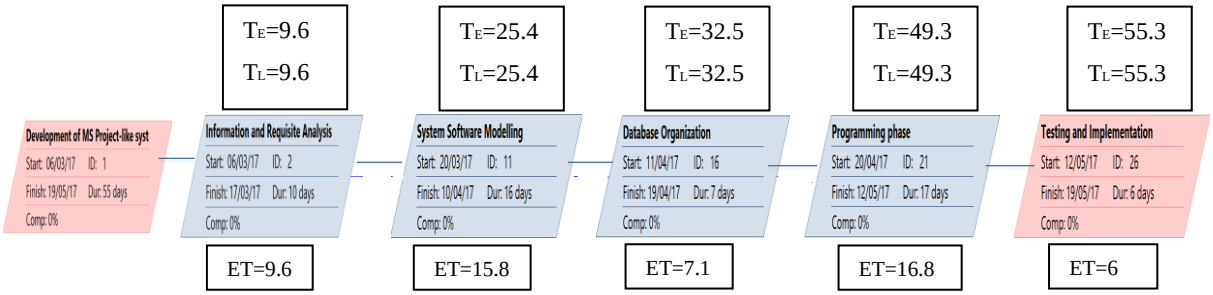

Figure 5. Netwrok diagram with expected time

Step6. Estimated Path Durations through the Network

Table 5. Path duration

Paths	Expected duration
abcdefg	**55.3 (critical path)**

Step7. Estimating the probability of completion dates for each path

$$\sigma^2 = \left(\frac{p-o}{6}\right)^2 \qquad (2)$$

Table 6. Probability of completion dates

Activity ID	Optimistic time	Realistic time	Pessimistic time	Expected time	Variance
a	0	1	2	1	0.11
b	7	10	13	9.6	0.44
c	14	16	19	15.8	0.25
d	6	7	10	7.16	0.25
e	15	17	19	16.8	0.25
f	5	6	9	6	0.11

Step8. Variance of each path through the network

Table 7. Variance of each path

Path number	Activities on Paths	Path variance (days)
1	abcdef	1.416

Step9. Calculating the Probability of Completing the Project in 56 Days

$$z = \frac{\text{specified time} - \text{path expected time}}{\text{path standard time}} = \left(\frac{D_T - EF_P}{\sqrt{\sigma_P^2}} \right) \quad (3)$$

Table 8. Probability of completion in 56 days

Path number	Activities on Paths	Path variance (days)	Z-value	Probability of completion
1	abcdef	1.416	0.494	1

Step10. Crashing

The project manager wants to reduce the new product project for 2 days.

Table 9. Crashing table

Activity ID	Normal time	Normal cost ($)	Crash time	Crash cost	Max. days of reduction	Reduce cost per day
a	1	120	1	240	1	120
b	10	1200	3	1560	3	360
c	16	1920	3	2280	2	240
d	7	840	3	1200	1	120
e	17	2040	2	2280	2	240
f	6	720	1	840	1	120
Totals	56	6840	13	8400	8 days	1200

Cost Management Processes:

We use COCOMO II as one of cost models. To calculate this cost model we need to follow the steps below and use COCOMOII Formula (4):

$$E = 2.45 \times (KLOC)^{b} \times EAF \qquad (4)$$

$$b = 1.01 + 0.01 \sum_{j=1}^{5} SF_i \qquad 0.91 \leq b \leq 1.23$$

- First we need to define program productivity. It depends on the developer's experience and capability in conjunction with the capabilities of the CASE tools. We calculated the productivity rate as follows:
    - The developers' experience is Nominal = 13
    - The CASE tool which is used is low = 7
    - PROD = (13+7) / 2 = 10
- After that we use objects Point Analysis to rate the system. This application has 6 screens which they defined as follows:
    - Create project Screen: Create new project.
    - Display interface Screen: Shows the page after creating project.
    - Enter task & resources Screen: Show the section that user can enter information
    - Display for user Screen: Display page after entering tasks and resources.
    - Request activity chart Screen: User can request activity chart.
    - Result Screen: Display the final result for user.

Table 10. Rating result

Name	**Object**	**Complexity**	**Weight**
Creating project	Screen	Simple	1

Display interface	Screen	Medium	2
Enter task & resources	Screen	Medium	2
Display user after entering	Screen	Simple	1
Request activity chart	Screen	Medium	2
Display result	Screen	Simple	1
		Total	9

- Then calculate the effort in person-months by using formula (5):

$$PM = (NAP * (1 - \%reuse / 100)) / PROD \quad (5)$$

After applying formula (5) we have:

  - PM = 9 * (1 – 0 / 100) / 10 = 0.9 person-months

- After that we use objects Point Analysis to rate the system. This application has 6 screens which they defined as follows:

  - Create project Screen: Create new project.
  - Display interface Screen: Shows the page after creating project.
  - Enter task & resources Screen: Show the section that user can enter information
  - Display for user Screen: Display page after entering tasks and resources.
  - Request activity chart Screen: User can request activity chart.
  - Result Screen: Display the final result for user.

Table 11. Rating result

Name	**Object**	**Complexity**	**Weight**
Creating project	Screen	Simple	1

Display interface	Screen	Medium	2
Enter task & resources	Screen	Medium	2
Display user after entering	Screen	Simple	1
Request activity chart	Screen	Medium	2
Display result	Screen	Simple	1
		Total	9

- Then calculate the effert in person-months by using formula (5):

$$PM = (NAP * (1 - \%reuse / 100)) / PROD \quad (5)$$

After applying formula (5) we have:

- PM = 9 * (1 – 0 / 100) / 10 = 0.9 person-months

4.2.2 Requirements analysis and development:

We use the IEEE standard and we used Adobe Photoshop Cs6 and Microsoft PowerPoint to draw the UML diagrams. The diagrams are shown below:

A. ER diagram

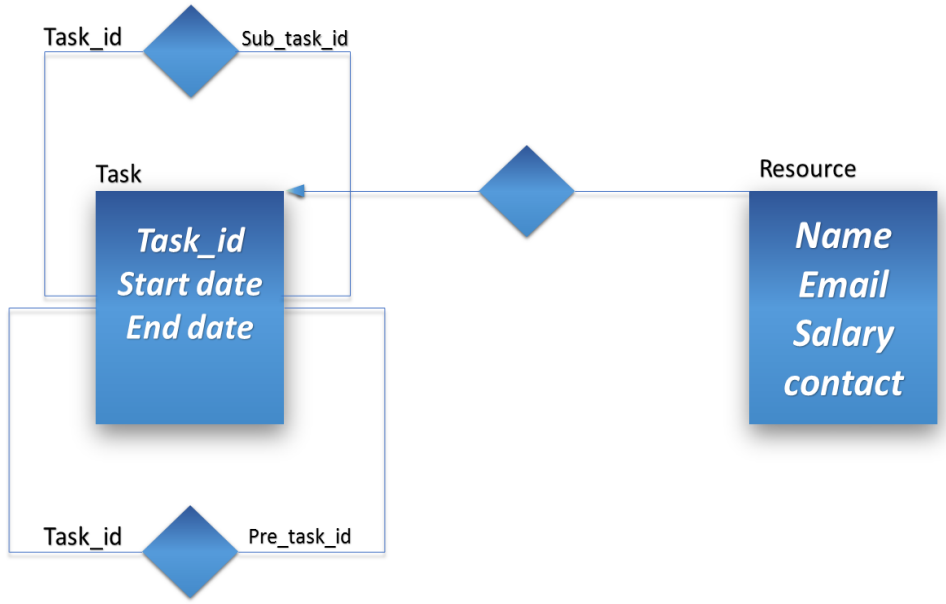

Figure 6 ER Diagram

B. Sequence Diagram

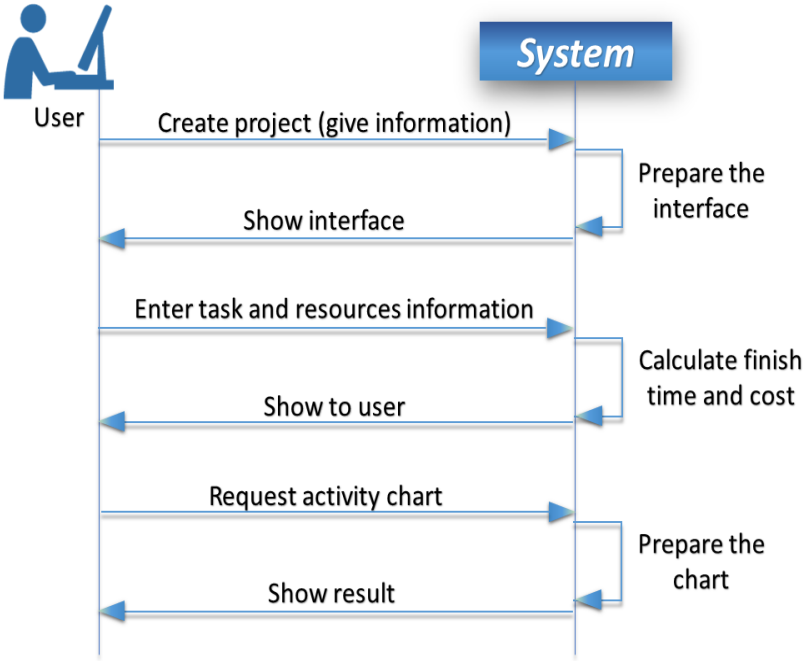

Figure 7 Sequence Diagram

C. Use Case diagrams

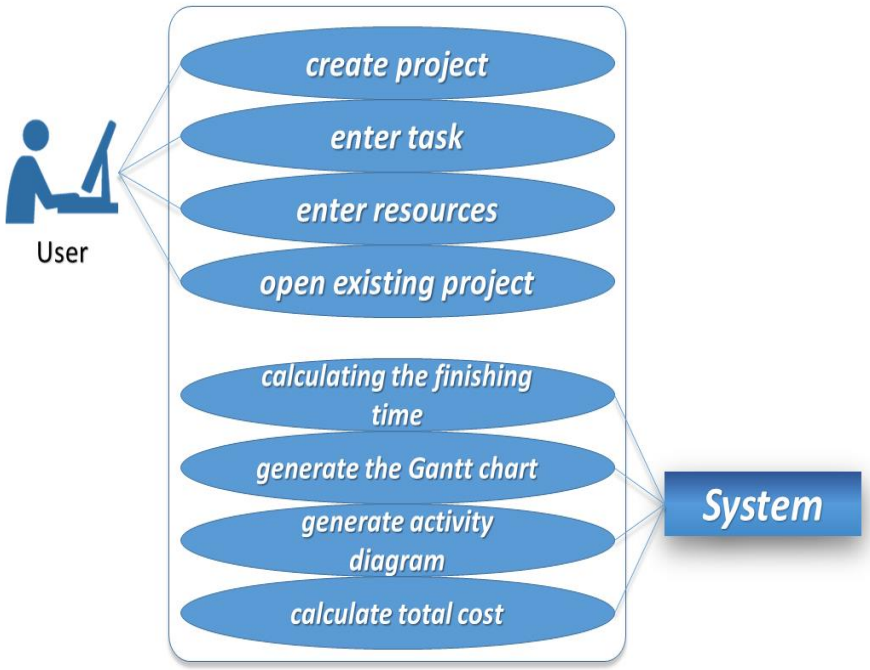

Figure 8 Use Case Diagram

4.3 Algorithms

Our application had many algorithms, however, here we will focus on only the main algorithms that we used.

1- Calculating the duration in different types:

When the user is filling the data for each task, he/she is asked to choose a duration (integer) and a type of duration (Hour / Day / Week / Months). Since our Gantt chart knows only days, we must convert everything to days, so when the user types a duration and type, the application will functions as follow:

Switch(type_of_duration)

Case “Hour” ➔ Divide the duration by the number of working hours.

Case “Day”➔ Leave it as it is.

Case “Week”➔ Multiply the duration by 7

Case “Month”➔Multiply the duration by 30.

End

Finally, we store the new value of duration in our dataset.

2- Excluding Weekend:

The programming language that we are using is able know the exact day of every given date, for example, 25/05/2017, our program can easy know that it is Thursday.

When the user fills the starting time, and the duration and the type of duration, the algorithms of “calculating the duration in different types” is executed and then we have a new duration, then the following algorithm will take place:

DateTime dt = the starting time of the task.

Int new_duration=the duration that is calculated from the above algorithm.

Int i=0,

Int count_working_days=0;

While (i<new_duration)

{

If(dt+i)==Saturday or Sunday➔ count_working_days+= 2// if the algorithm found Saturday, it must scape Saturday and Sunday that’s why we added 2.

Else ➔ count_working_days+=1 in the normal days, we add one.

}

By this algorithm, we will have another new duration which is count_working_days, the difference between this and the other one, is that the newer one excluded the weekends.

3- Calculating the finishing time:

Once a user successfully filled the starting time and the duration and type of duration, the finishing time will be calculated automatically and placed in its place, by adding the new duration to the starting time.

4- Calculating the cost of a task:

When the user chooses a resource for a task, the application goes to the resources information and take its corresponding salary, and multiply it by the new duration and number of working hour, since the salary of each user is a salary per hour.

5- Calculating the total cost of the project:

By simply add the total cost of all tasks.

6-Finding Critical path:

The essential technique for using CPM(Critical Path Method) is to construct a model of the project that includes the following:

1. A list of all activities required to complete the project (typically categorized within a work breakdown structure),
2. The time (duration) that each activity will take to complete,
3. The dependencies between the activities and,
4. Logical end points such as milestones or deliverable items.

Using these values, CPM calculates the longest path of planned activities to logical end points or to the end of the project, and the earliest and latest that each activity can start and finish

without making the project longer. This process determines which activities are "critical" (i.e., on the longest path) and which have "total float" (i.e., can be delayed without making the project longer). In project management, a critical path is the sequence of project network activities which add up to the longest overall duration, regardless if that longest duration has float or not. This determines the shortest time possible to complete the project. There can be 'total float' (unused time) within the critical path. For example, if a project is testing a solar panel and task 'B' requires 'sunrise', there could be a scheduling constraint on the testing activity so that it would not start until the scheduled time for sunrise. This might insert dead time (total float) into the schedule on the activities on that path prior to the sunrise due to needing to wait for this event. This path, with the constraint-generated total float would actually make the path longer, with total float being part of the shortest possible duration for the overall project. In other words, individual tasks on the critical path prior to the constraint might be able to be delayed without elongating the critical path; this is the 'total float' of that task. However, the time added to the project duration by the constraint is actually critical path drag, the amount by which the project's duration is extended by each critical path activity and constraint.

A project can have several, parallel, near critical paths; and some or all of the tasks could have 'free float' and/or 'total float'. An additional parallel path through the network with the total durations shorter than the critical path is called a sub-critical or non-critical path. Activities on sub-critical paths have no drag, as they are not extending the project's duration.

CPM analysis tools allow a user to select a logical end point in a project and quickly identify its longest series of dependent activities (its longest path). These tools can display the critical path (and near critical path activities if desired) as a cascading waterfall that flows from the project's start (or current status date) to the selected logical end point.

In the coding stage, the code about critical path will be shown.

4.4 Standards

- For the design of components, the MetroFramework was used.
- Adding comments add the beginning of functions to serve as descriptions of the functions.

5. TESTING

Test-Case ID:	TC-01
Test-Case Name:	Create new project
Pass/Fail Criteria :	The test passes if the user fill all requirement fields correctly.
Input Data:	Numeric and alphabet key code
Test Procedure:	**Expected Outcomes :**
Step-1: An empty requirement field (Name, Type, Starting Date, Directory, Description, Country, Currency, Number of Working hour per day)	System indicate failure; An appropriate error message should be displayed and the user shouldn't be allowed to create new project by clicking next button.
Step-2: Fill in all requirement fields in correct format	System indicate success; The user should be directed to the next page by clicking the next button which became active by filling all required fields correctly.

Table 12 "Create New Project" Test

Test Case Result

Test Case ID: TC-01

Test Designed By: Serhat Kocagöz

Module Name: Required fields

Test Designed Date: 23/5/2017

Test Title: Create new project

Test Executed By: Serhat Kocagöz

Description: Required fields tested

Test Execution Date: 23/5/2017

Pre-Conditions: User has fill all requirement fields correctly.

Step No.	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	An empty requirement field	An empty requirement field	An appropriate error message should be shown an the user shouldn't be allowed to move next page	An appropriate error message shown (blinking error icons at right of textboxes) and user not allowed to move next page	**Pass**
2	Fill in all requred fields in correct format	Name: testProject Type: testProjectType Starting Date: Default (as today) Directory: A valid directory	If all the required spaces are correctly filled, user can navigate to next page.	After all fields correctly filled, next button became enabled so that the user can navigate to next page by using this button	**Pass**

		Description: This is a test project. Country: Turkey Currency: TRY (Automated selection) Number of working hour per day: 8			

Table 13 "Create New Project" Test Results

Test-Case ID:	TC-02
Test-Case Name:	Adding Tesks
Pass/Fail Criteria :	The test passes if the user can not add a valid resource or enters an invalid resource
Input Data:	Resource name, salary per hour and E-mail data
Test Procedure:	**Expected Outcomes :**
Step-1: An empty field (Name, Salary per hour or E-mail) in resources form	System indicate failure; If the resource is accepted even with a missing data, system fails.

Step-2: Fill in all requirement fields in correct format	System indicate success; Resource is automatically added to the resources pool and user can assign the added resource to task

Table 14 "Addidng Resources" Test

Test Case Result

Test Case ID: TC-02

Test Designed By: Serhat Kocagöz

Module Name: Resource adder

Test Designed Date: 23/5/2017

Test Title: Adding new resource

Test Executed By: Serhat Kocagöz

Description: Resource adder tested

Test Execution Date: 23/5/2017

Pre-Conditions: User has fill all requirement fields correctly.

Step No.	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	An empty requirement field	An empty requirement field	Resource shouldn't added to the resources pool	After adding invalid resource, it didn't shown in the resource name list in main page	**Pass**
2	Fill in all requred fields in correct format	Name: testResource1 Salary:100 E-mail: testResource1@testDomain.com	If all the required spaces are correctly filled, user can see new resource on the list.	After all fields correctly filled, new resource appeared on the available resources list	**Pass**

Table 15 "Adding New Resource" Test Results

Test-Case ID:	TC-03
Test-Case Name:	Adding Tesks
Pass/Fail Criteria :	The test passes if the user can not add a valid task or enters an invalid task
Input Data:	Task name, duration, type of duration, start time, finish time, cost, resource name
Test Procedure:	**Expected Outcomes :**
Step-1: An empty field or inappropriate data is entered (Name, Salary per hour or E-mail) in resources form	System indicate failure; If the resource is accepted even with a missing data, system fails.
Step-2: Fill in all requirement fields in correct format	System indicate success; Resource is automatically added to the resources pool and user can assign the added resource to task

Table 16 "Addidng New Task" Test

Test Case Result

Test Case ID: TC-03

Test Designed By: Serhat Kocagöz

Module Name: Task adder

Test Designed Date: 23/5/2017

Test Title: Adding new task

Test Executed By: Serhat Kocagöz

Description: Task adder tested

Test Execution Date: 23/5/2017

Pre-Conditions: User has fill all requirement fields correctly.

Step No.	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	An empty requirement field	An empty requirement field	Resource shouldn't added to the resources pool	After adding invalid resource, it didn't shown in the resource name list in main page	**Pass**
2	Fill in all requred fields in correct format	Task Name: testTask1 Duration: 30 Type of Duration: Hour Start Time: 23/5/2017 Finish Time: 26/5/2017 Cost: 3200 (Auto-Generated) Resource Name: testResource1	If all the required spaces are correctly filled, user can see new task on the list.	After all fields correctly filled, new task appeared on the tasks list.	**Pass**

Table 17 "Addidng New Task" Test

Test-Case ID:	TC-04
Test-Case Name:	Network diagram creating
Pass/Fail Criteria :	The test passes if the program can generate a network diagram according to the entered valid datas
Input Data:	Task list
Test Procedure:	**Expected Outcomes :**
Step-1: Creating a network diagram	System indicate failure; If the network diagram is not created or there is a miscalculation on the paths. System indicate success; If the network diagram is created and drawn.

Table 18 "Network Diagram Created" Test

Test Case Result

Test Case ID: TC-04

Test Designed By: Serhat Kocagöz

Module Name: Diagram Drawer

Test Designed Date: 24/5/2017

Test Title: Network diagram creating

Test Executed By: Serhat Kocagöz

Description: Network diagram tested

Test Execution Date: 24/5/2017

Pre-Conditions: User has fill all requirement fields correctly.

Step No.	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	Creating a network diagram	Task list	Network diagram either created successfully (**Pass**) or not created (**Fail)**	After adding valid tasks and resources, network diagram is created successfully.	**Pass**

Table 19 "Network Diagram Created" Test Results

Test-Case ID:	TC-05
Test-Case Name:	Total calculation testing
Pass/Fail Criteria :	The test passes if the program can calculate the total cost of the project according to the entered datas
Input Data:	Project datas
Test Procedure:	**Expected Outcomes :**
<u>Step-1:</u> Calculating the total	System indicate failure; If the calculation is wrong or not generated System indicate success; If the calculation is true according to the entered datas

Table 20 "Total calculation" Test

Test Case Result

Test Case ID: TC-05

Test Designed By: Serhat Kocagöz

Module Name: Calculate Total

Test Designed Date: 24/5/2017

Test Title: Total calculation testing

Test Executed By: Serhat Kocagöz

Description: Total calculator tested

Test Execution Date: 24/5/2017

Pre-Conditions: User has fill all requirement fields correctly.

Step No.	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	Calculating the total	Porject datas	Total amount is either calculated successfully (**Pass**) or not calculated (**Fail)**	After adding valid project datas, total amount is calculated successfully.	**Pass**

Table 21 "Total Calculation" Test Results

Test-Case ID:	TC-06
Test-Case Name:	Saving project
Pass/Fail Criteria :	The test passes if the user can not save a project with missing fields
Input Data:	Project data
Test Procedure:	**Expected Outcomes :**
Step-1: An empty field or inappropriate data is left in the task list.	System indicate failure; An appropriate error message should be displayed and the user shouldn't be allowed to save project.
Step-2: Fill in all requirement fields in correct format	System indicate success; The user should be able to save the project with correct data.

Table 22 "Saving Project" Test

Test Case Result

Test Case ID: TC-06

Test Designed By: Serhat Kocagöz

Module Name: Project Saver

Test Designed Date: 24/5/2017

Test Title: Saving Project

Test Executed By: Serhat Kocagöz

Description: Project saver tested

Test Execution Date: 24/5/2017

Pre-Conditions: User has fill all requirement fields correctly.

Step No.	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	An empty requirement field	An empty requirement field	Project can not be saved, error message should be shown	With a missing or incorrect field, project is not saved and error message is shown.	Pass
2	Fill in all requred fields in correct format	Correctly filled project data fields.	If all the required spaces are correctly filled, user can save the project.	After all fields correctly filled, project is successfully saved.	Pass

Table 23 "Saving Project" Test Results

Test-Case ID:	TC-07
Test-Case Name:	Loading Project
Pass/Fail Criteria :	The test passes if the user can load a project with correct file.
Input Data:	Saved project file
Test Procedure:	**Expected Outcomes :**
Step-1: A corrupted or wrong file type is loaded to the program	System indicate failure; An appropriate error message should be displayed and the program should close
Step-2: A correct file is loaded to the program	System indicate success; The user should be able to load the file and see the project correctly

Table 24 "Loading Project" Test

Test Case Result

Test Case ID: TC-07

Test Designed By: Serhat Kocagöz

Module Name: Project Loader

Test Designed Date: 25/5/2017

Test Title: Loading Project

Test Executed By: Serhat Kocagöz

Description: Project loader tested

Test Execution Date: 25/5/2017

Pre-Conditions: User has fill all requirement fields correctly.

Step No.	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	A corrupted or wrong file type is loaded to the program	An empty textFile	File can not be loaded, error message should be shown	A missing or corrupted file is not loaded and error message is shown.	**Pass**
2	A correct file is loaded to the program	An correct project file	If the file integrity is intact, file should be loaded and the project details should be showned	A correct file is loaded and the project details are shown.	**Pass**

Table 25 "Saving Project" Test Results

6. USER GUIDE OF THE SYSTEM

Homepage

1- After clicking the application icon, the application homepage shown in figure 9 appears.
2- If a new project is to be created, click New project button.
3- If opening a existing project, click “Open existing Project”,
4- If information about the team who made the application is needed, click “About Team”
5- if you want to close the application, you can either click “Exit” or the cloes button on the top right corner of the homepage.

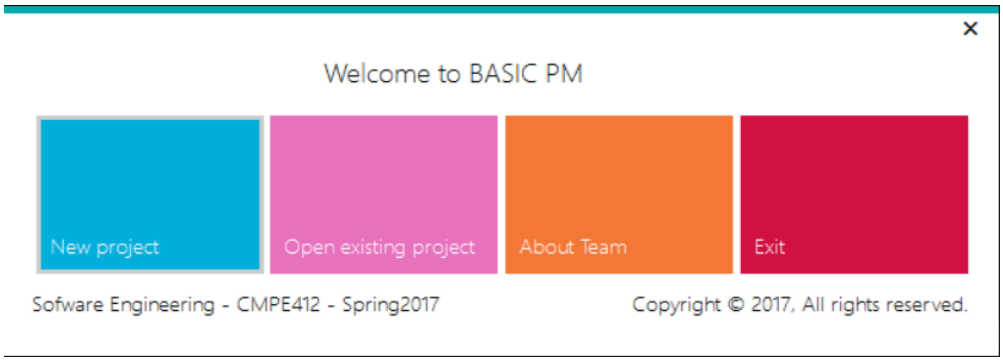

Figure 9 Application Homepage

Project Creation Form

After the user either clicks “New project” or “Open existing project”, the project creation form, figure 10, appears. This is where the user gives information about his/her project. It has the following fields:

1- Name field – this field is mandatory
2- Type – this field is optional
3- Starting date – this field is mandatory
4- End date – this field is mandatory
5- Directory – this is where you want the filed to be save and is also mandatory
6- Description – this is an option field for a brief description of the project you are about to create.
7- Country – this is the country in which the project will be executed and is mandatory.
8- Currency – this is automatically field by the application.

9- Number of working hours per day – this is a mandatory field where the user specifies the number of working hours of the personnel.

After all the mandatory fields have be filled the application allows the user to click the next button, if not he/she will not be able to continue.

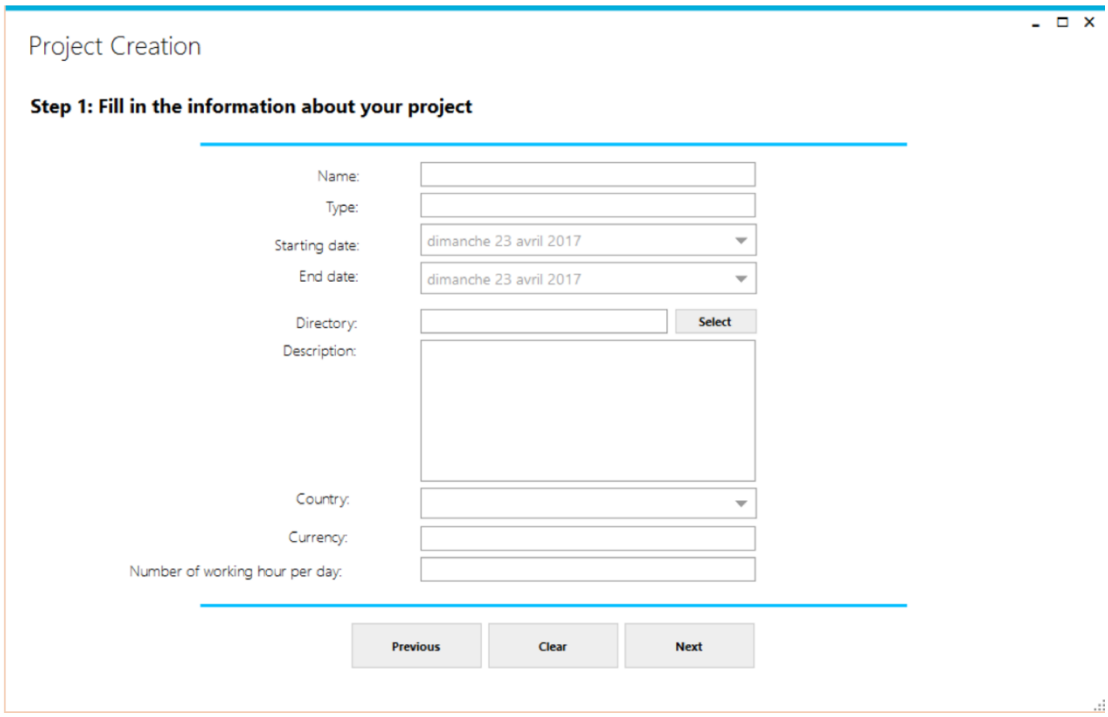

Figure 10 Project Creation Form

Main Interface

After filling all the necessary information about the Project, the main interface, figure 11, appears

1- A user can add a new task by clicking a cell under the task name and adding the name of a task
2- After adding a task, a user has to give the duration of the task in the duration cell to the right of the Task Name.
3- After adding the duration of the task, the user next selects the "Type of Duration". This can either be a "day", "week", "month" or "year".
4- Next, the user has to give the start time of the specific Task.
5- The finish time will be automatically calculated by the application
6- In addition to all the above which are necessary, a user can add the cost and resource name, but these two are optional.

7- Finally, a user can add the predecessor of a task, i.e. the task that has to be completed before the current task can start. Of course this is only added if a task has a predecessor.
8- After these fields have been properly field, the Gantt chart related to this task immediately appear on the right half of the main Interface.
9- A user can add as many task as might be needed to complete a project.
10- Resources can be added to any task in the project. However these resources have to be added to the resource pool, which is the top left icon title “resource”.
11- The user can decide to save after adding the tasks by clicking the save button.

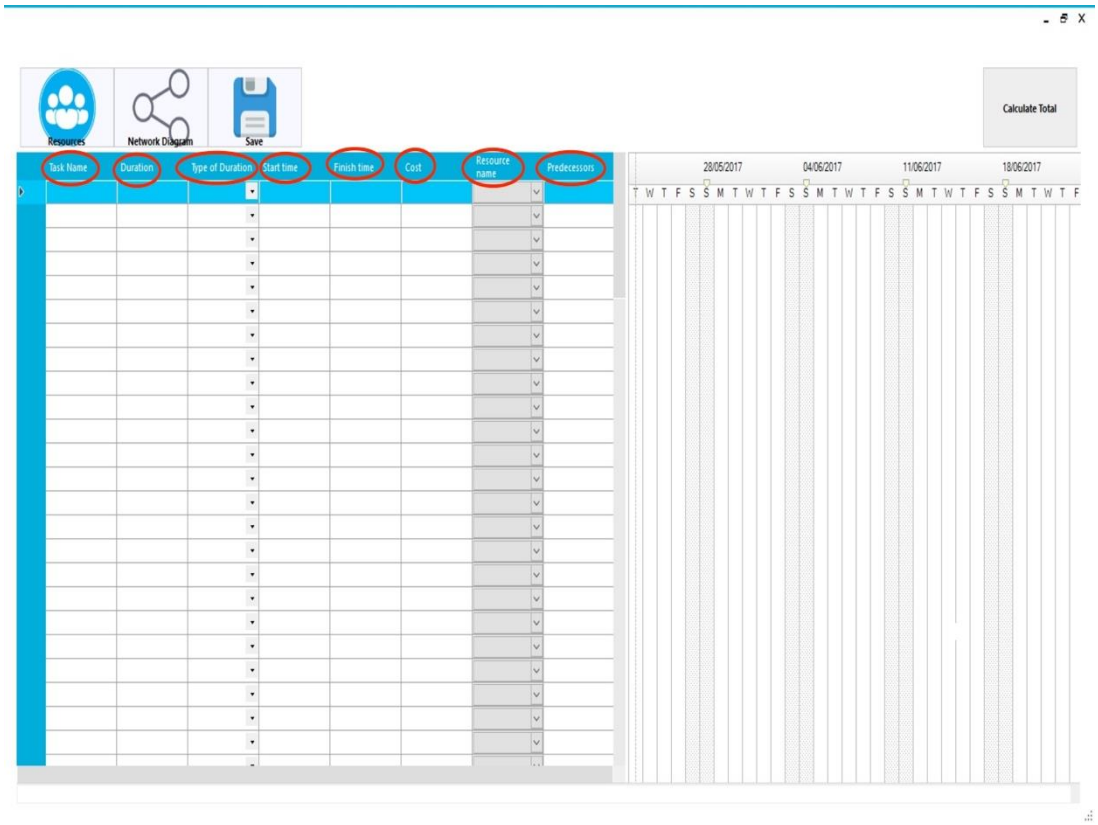

Figure 11 Main Interface

Network Diagram

Each project has a network Diagram that shows project start date, tasks, dependencies and end dates. After adding all the necessary tasks, a user can view the network diagram of a project. figure 12, by clicking on the network diagram button (figure 13). In addition, the user can do the following in the network diagram interface.

1- Export the diagram as an image by clicking the export button
2- Print the diagram by clicking the print button

3- Zoom in and zoom out from the network diagram.

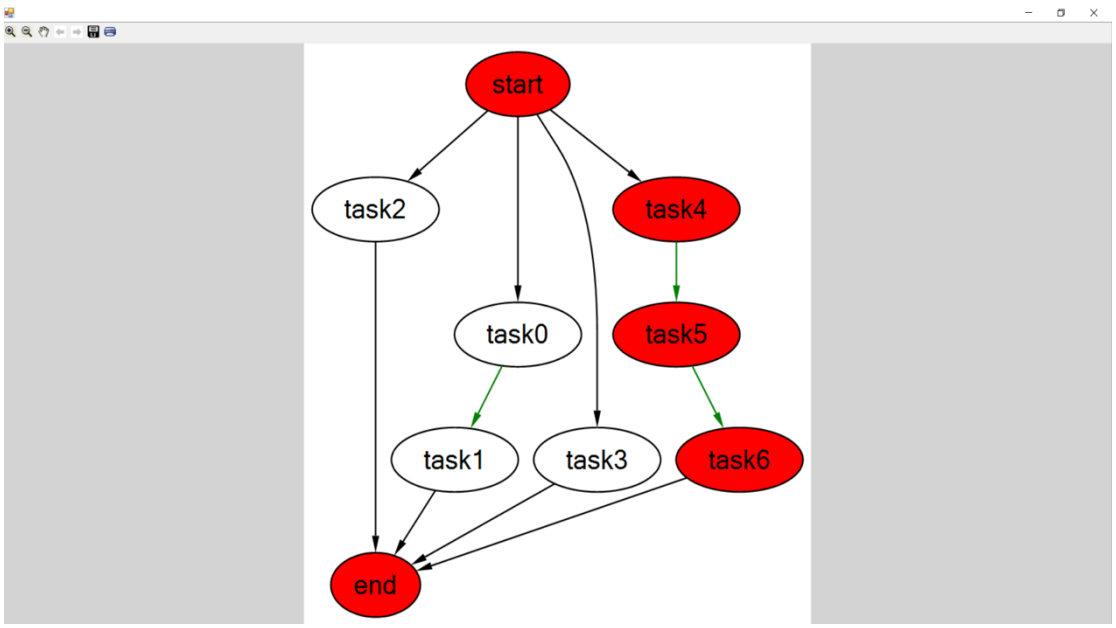

Figure 12 Network Diagram

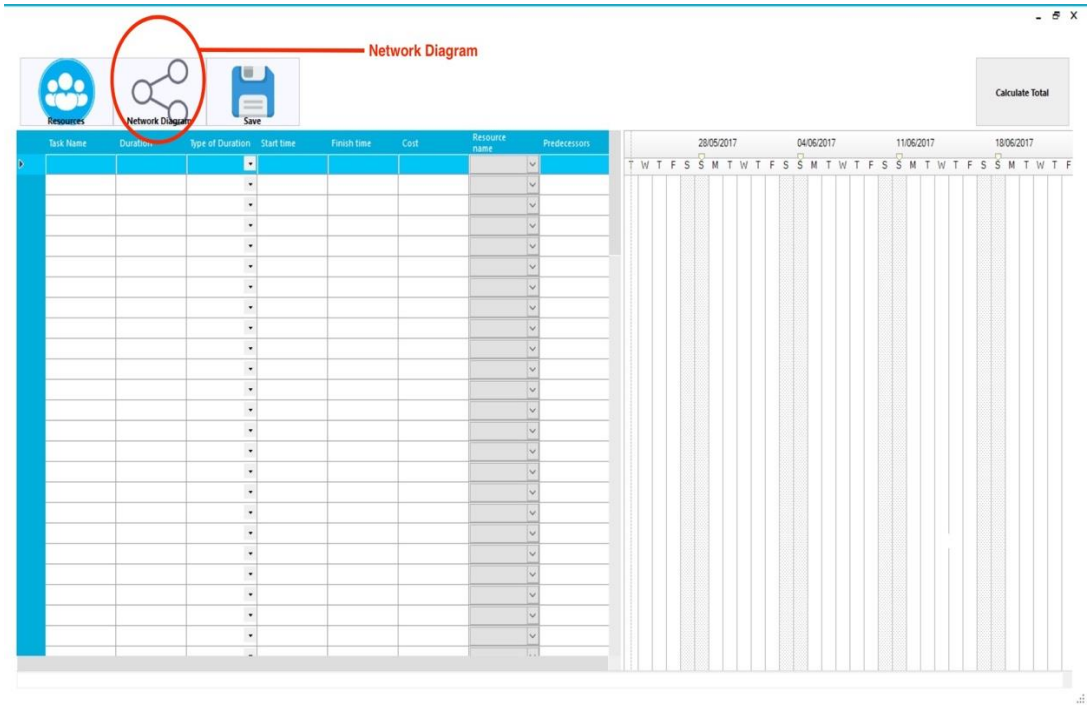

Figure 13 Network Diagram Button

Closing the application

1- Click the save button

2- Click the X on the top left corner of the main interface

7. DISCUSSION

7.1Economic Impact

Our application will save a company, school or organization a lot of money for the following reason. Its free cost in comparison with same products in the market with the same functionalities will help the company using our product to save money. This amount of saved money could be used by the company to progress further in their field which will certainly create jobs for people therefore helping in improving the local and global economy.

7.2 Social Impact

This application will help in organizing and in making it easier to keep up with any project progress therefore making it a simpler job. This will help in saving a lot of time and it will also help in handling tasks better which will result in creating a better working atmosphere for employees. Good working atmosphere is an essential thing in creating a healthy productive society.

7.3 Environmental Impact

Saving trees will be a huge environmental advantage of our application since our application focus is the digitalization of project organization which is usually done on paper.

7.4 Global Impact

The simplicity, free cost and solid functionality of the application will have a huge impact on the social, economic and environmental context of our users' local community and since the local community is a building block of the global community, we would be contributing to the development of the whole globe.

8. CONCLUSION

This project is a project management application similar to the famous and professionally used Microsoft Project application.

Basically it allows project management by project managers, company staff and everyone who has affiliation with this duty, doing everything related to it, from creating projects, tasks, resources to creating Gantt charts, activity network diagrams, exporting data, importing data and everything necessary to do project management in an application.

This project is useful because, this application can reduce time spent in the process of project management. It can also do many tasks more efficiently than plain planning on paper, thereby enabling big companies, managers and individuals to plan their tasks using this project in a simpler way. Because this application separates project planning stages into one at a time, it is more reliable way to finish work since human errors can occur without this project, and this exactly what the system developed does, not mentioning it's ability to detect errors and reliability features.

In this project we managed to create an application that is more efficient and easier to use than Microsoft Project. With a more streamlined way of doing things by cutting out the complexity found in large application like MS Project, you can create projects and manage them on the fly with our application. In addition to this, we managed to have clean interface with the aid of the MetroFramework. In order to increase speed and efficiency, we avoided the use of a database by using datatables and datasets instead of a database.

This application can also do majority of core tasks done by MS project such as:

- Creating work break down schedule.
- Creating Gantt chart.
- Doing complex calculations of finish times and durations, swiftly and effectively.
- Inserting milestones to tasks.
- Offering Timeline View
- Having different views.
- Data protection
- Data Integrity

We achieved a program that is portable since it is made in .exe format, light and user friendly, so it can be taken anywhere and its saved files can be used on any system with a similar application installed.

This project helped our team members on many levels, starting from team work and organizing jobs, we learned to stay on touch with other teammates to synchronize the work. It developed our communication and integrity in doing our assigned task. Besides, working on developing this software gave us an experience of software engineering development, and a better insight and understanding of the ins and outs of project management systems, how to use minimal design to simplify a large project into more reasonable system, we learned how to create the program with development resources available.

Of course each team member developed personally in their own field such as, ER design, C# programming, .Net framework, UI design, algorithms and all the necessary documentation work necessary for the project.

We learned how to use .NET components such as datagrid, textbox. Etc, how to use latest Metroframework, how to do export and import data on files from a gridview and how to properly use visual studio to make a functional application.

All in all, it was a great opportunity to apply software engineering and learn about software development.

9. REFERENCES

[1] “Gantt Chart Control: Braincase.GanttChart.Task Class Reference,” *Gantt Chart Control: Braincase.GanttChart.Task Class Reference*. [Online]. Available: http://jakesee.com/docs/ganttchart/class_braincase_1_1_gantt_chart_1_1_task.html. [Accessed: 25-May-2017].

[2] “Metro Modern UI - Metro Framework,” *NuGet Gallery | Metro Modern UI - Metro Framework 1.4.0*. [Online]. Available: https://www.nuget.org/packages/MetroModernUI/. [Accessed: 25-May-2017].

[3] “Automatic Graph Layout,” *Microsoft*. [Online]. Available: https://www.microsoft.com/en-us/download/details.aspx?id=52034&from=http%3A%2F%2Fresearch.microsoft.com%2Fen-us%2Fdownloads%2Ff1303e46-965f-401a-87c3-34e1331d32c5%2F. [Accessed: 25-May-2017].

[4] U. K., “UweKeim/dot-net-transitions,” *GitHub*, 16-Jul-2015. [Online]. Available: https://github.com/UweKeim/dot-net-transitions. [Accessed: 25-May-2017].

APPENDICES

A. Instructions for installing the system

1- Get the executable file

Figure 14 Executable File

1- Double click on the executable file to get the set up wizard

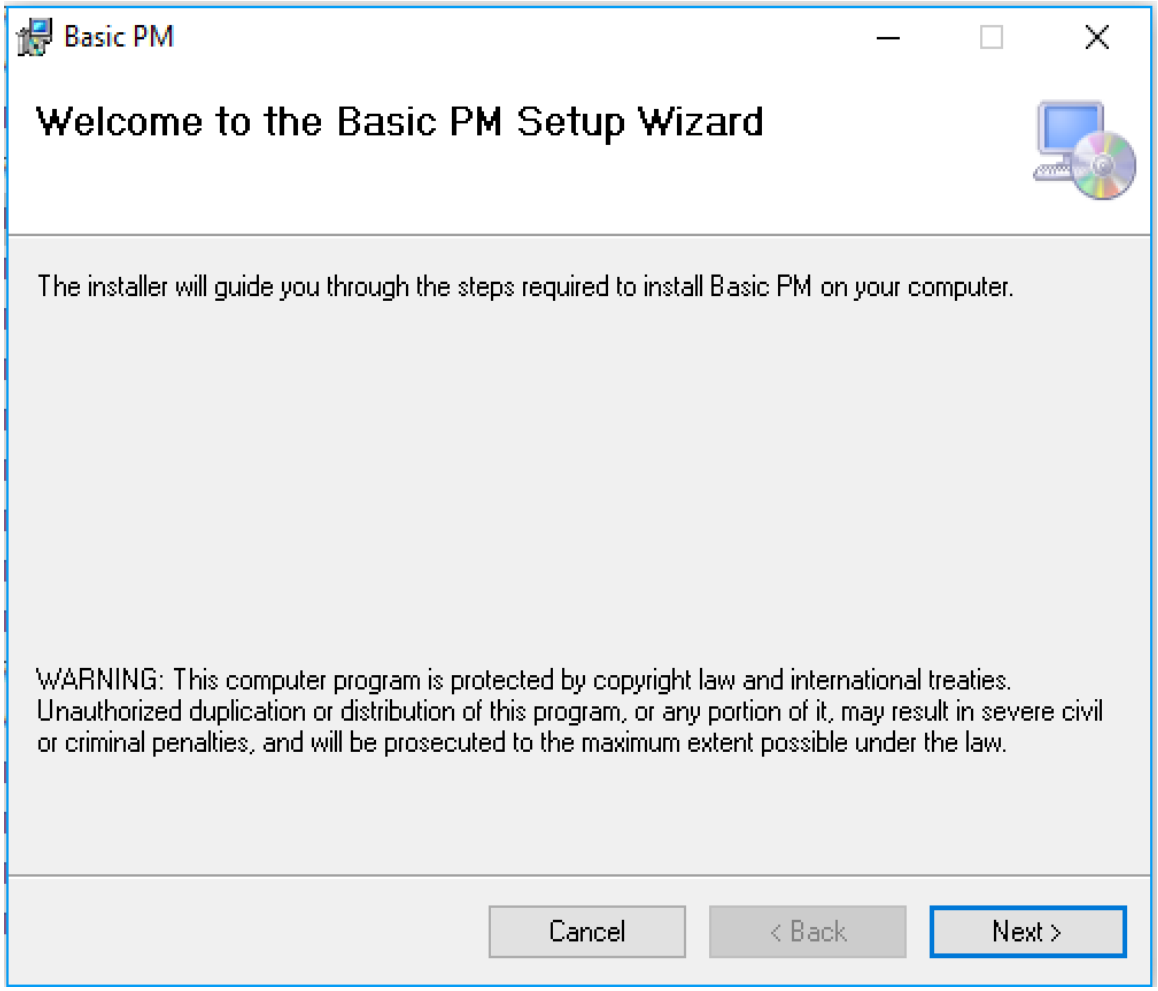

Figure 15 Setup Wizard

2- click on “Next”

3- After this, select the installation folder as shown in figure 16, and choose how many users you want to use the application i.e.

   i. Select “Everyone” to give access to all users.

   ii. Select “Just Me” to give access to only the current user.

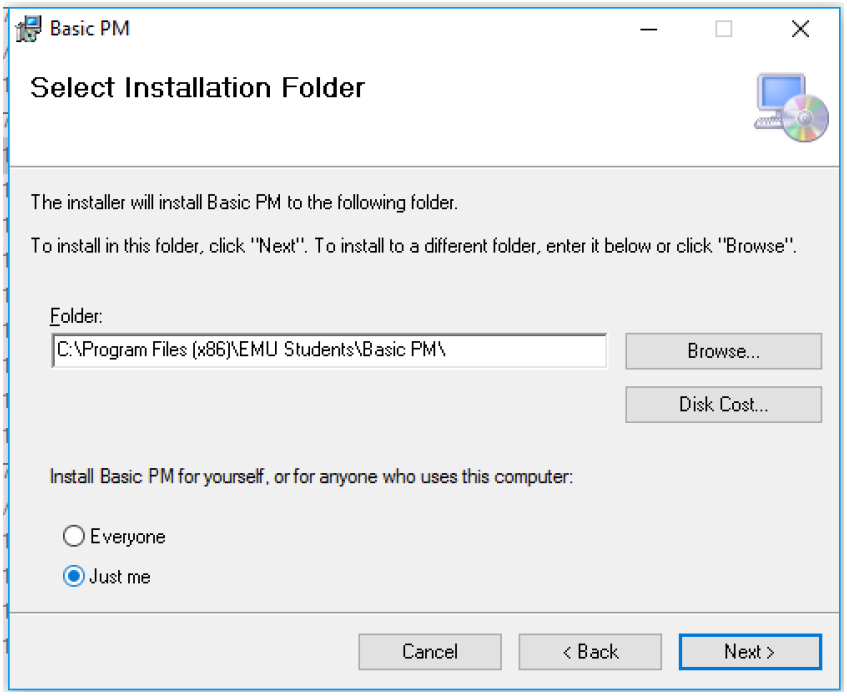

Figure 16 Select Installation Path

4- click Next, as show in figure 17, to confirm your installation

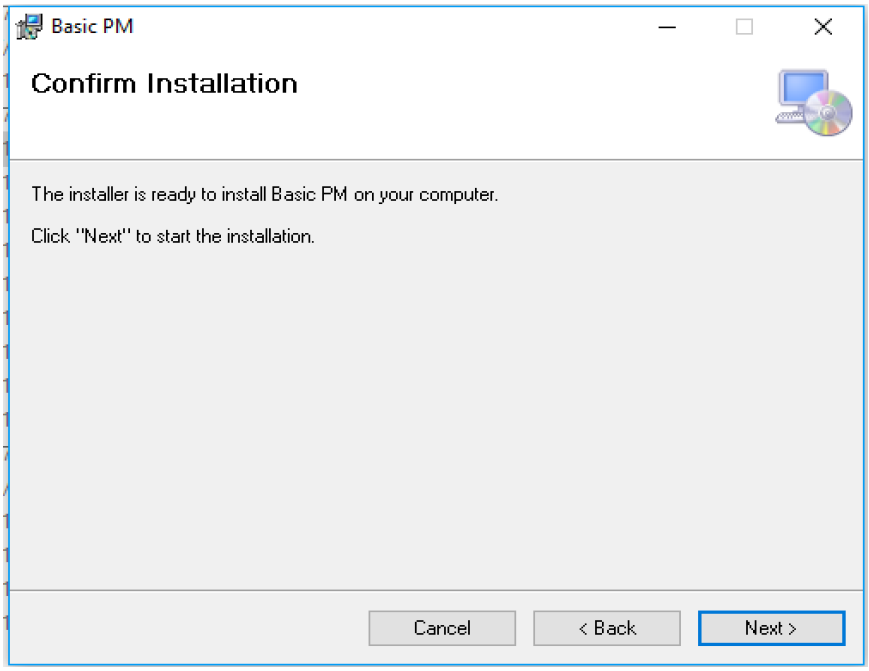

Figure 17 Comfirm installation

B. Code for the system

We have used multiple forms in our applications, here we show the necessary code for application in order to function properly.

The application starts with a Menu, in the Menu we use some classes for animation written by students from MIT details of which can be found in reference [4]. The Content of those classes are shown as follow:

```
using System;

using System.Collections.Generic;

using System.Text;

using System.Reflection;

using System.Timers;

using System.Diagnostics;

using System.Windows.Forms;

using System.ComponentModel;

namespace Transitions

{

    /// <summary>

    /// Lets you perform animated transitions of properties on arbitrary objects. These

    /// will often be transitions of UI properties, for example an animated fade-in of

    /// a UI object, or an animated move of a UI object from one position to another.

    ///

    /// Each transition can simulataneously change multiple properties, including properties

    /// across multiple objects.

    ///

    /// Example transition

    /// ------------------

    /// a.      Transition t = new Transition(new TransitionMethod_Linear(500));

    /// b.      t.add(form1, "Width", 500);

    /// c.      t.add(form1, "BackColor", Color.Red);

    /// d.      t.run();

    ///
```

```
/// Line a:         Creates a new transition. You specify the transition method.
///
/// Lines b. and c: Set the destination values of the properties you are animating.
///
/// Line d:         Starts the transition.
///
/// Transition methods
/// ------------------
/// TransitionMethod objects specify how the transition is made. Examples include
/// linear transition, ease-in-ease-out and so on. Different transition methods may
/// need different parameters.
///
/// </summary>
public class Transition
      {
            #region Registration


            /// <summary>
   /// You should register all managed-types here.
   /// </summary>
   static Transition()
   {
      registerType(new ManagedType_Int());
      registerType(new ManagedType_Float());
                  registerType(new ManagedType_Double());
      registerType(new ManagedType_Color());
      registerType(new ManagedType_String());
   }
```

```
        #endregion

        #region Events

        /// <summary>
        /// Args passed with the TransitionCompletedEvent.
        /// </summary>
        public class Args : EventArgs
        {
        }

        /// <summary>
        /// Event raised when the transition hass completed.
        /// </summary>
        public event EventHandler<Args> TransitionCompletedEvent;

        #endregion

        #region Public static methods

        /// <summary>
        /// Creates and immediately runs a transition on the property passed in.
        /// </summary>
        public static void run(object target, string strPropertyName, object destinationValue, ITransitionType transitionMethod)
        {
            Transition t = new Transition(transitionMethod);
            t.add(target, strPropertyName, destinationValue);
            t.run();
        }
```

```
/// <summary>
/// Sets the property passed in to the initial value passed in, then creates and
/// immediately runs a transition on it.
/// </summary>
public static void run(object target, string strPropertyName, object initialValue, object destinationValue, ITransitionType transitionMethod)
{
   Utility.setValue(target, strPropertyName, initialValue);
   run(target, strPropertyName, destinationValue, transitionMethod);
}

/// <summary>
/// Creates a TransitionChain and runs it.
/// </summary>
public static void runChain(params Transition[] transitions)
{
   TransitionChain chain = new TransitionChain(transitions);
}

#endregion

#region Public methods

/// <summary>
/// Constructor. You pass in the object that holds the properties
/// that you are performing transitions on.
/// </summary>
publicTransition(ITransitionType transitionMethod)
{
```

```
                m_TransitionMethod = transitionMethod;
        }


                /// <summary>
                /// Adds a property that should be animated as part of this transition.
                /// </summary>
                public void add(object target, string strPropertyName, object destinationValue)
                {
                // We get the property info...
                Type targetType = target.GetType();
                PropertyInfo propertyInfo = targetType.GetProperty(strPropertyName);
        if (propertyInfo == null)
                        {
throw new Exception("Object: " + target.ToString() + " does not have the property: " + strPropertyName);
                        }


                        // We check that we support the property type...
                        Type propertyType = propertyInfo.PropertyType;
                        if (m_mapManagedTypes.ContainsKey(propertyType) == false)
                        {
                                throw new Exception("Transition does not handle properties of type: " + propertyType.ToString());
                        }


        // We can only transition properties that are both getable and setable...
        if (propertyInfo.CanRead == false || propertyInfo.CanWrite == false)
        {
          throw new Exception("Property is not both getable and setable: " + strPropertyName);
        }
```

```
    IManagedType managedType = m_mapManagedTypes[propertyType];

    // We can manage this type, so we store the information for the
                    // transition of this property...
            TransitionedPropertyInfo info = new TransitionedPropertyInfo();
                    info.endValue = destinationValue;
                    info.target = target;
                    info.propertyInfo = propertyInfo;
                    info.managedType = managedType;

    lock (m_Lock)
    {
        m_listTransitionedProperties.Add(info);
    }
            }

/// <summary>
/// Starts the transition.
/// </summary>
public void run()
{
    // We find the current start values for the properties we
    // are animating...
    foreach (TransitionedPropertyInfo info in m_listTransitionedProperties)
    {
        object value = info.propertyInfo.GetValue(info.target, null);
        info.startValue = info.managedType.copy(value);
    }
```

```
            // We start the stopwatch. We use this when the timer ticks to measure
                // how long the transition has been runnning for...
                m_Stopwatch.Reset();
                m_Stopwatch.Start();

    // We register this transition with the transition manager...
    TransitionManager.getInstance().register(this);
            }

#endregion

#region Internal methods

/// <summary>
/// Property that returns a list of information about each property managed
/// by this transition.
/// </summary>
internal IList<TransitionedPropertyInfo> TransitionedProperties
{
    get { return m_listTransitionedProperties; }
}

/// <summary>
/// We remove the property with the info passed in from the transition.
/// </summary>
internal void removeProperty(TransitionedPropertyInfo info)
{
    lock (m_Lock)
```

```
        {
            m_listTransitionedProperties.Remove(info);
        }
    }

    /// <summary>
    /// Called when the transition timer ticks.
    /// </summary>
    internal void onTimer()
    {
        // When the timer ticks we:
        // a. Find the elapsed time since the transition started.
        // b. Work out the percentage movement for the properties we're managing.
        // c. Find the actual values of each property, and set them.

        // a.
        int iElapsedTime = (int)m_Stopwatch.ElapsedMilliseconds;

        // b.
        double dPercentage;
        bool bCompleted;
        m_TransitionMethod.onTimer(iElapsedTime, out dPercentage, out bCompleted);

        // We take a copy of the list of properties we are transitioning, as
        // they can be changed by another thread while this method is running...
        IList<TransitionedPropertyInfo> listTransitionedProperties = new
List<TransitionedPropertyInfo>();
        lock (m_Lock)
        {
            foreach (TransitionedPropertyInfo info in m_listTransitionedProperties)
```

```
            {
                listTransitionedProperties.Add(info.copy());
            }
        }

        // c.
        foreach (TransitionedPropertyInfo info in listTransitionedProperties)
        {
            // We get the current value for this property...
            object value = info.managedType.getIntermediateValue(info.startValue,
info.endValue, dPercentage);

            // We set it...
            PropertyUpdateArgs args = new PropertyUpdateArgs(info.target, info.propertyInfo,
value);
            setProperty(this, args);
        }

        // Has the transition completed?
        if (bCompleted == true)
        {
            // We stop the stopwatch and the timer...
            m_Stopwatch.Stop();

            // We raise an event to notify any observers that the transition has completed...
            Utility.raiseEvent(TransitionCompletedEvent, this, new Args());
        }
    }

    #endregion
```

```
#region Private functions

            /// <summary>
            /// Sets a property on the object passed in to the value passed in. This method
            /// invokes itself on the GUI thread if the property is being invoked on a GUI
            /// object.
            /// </summary>
            private void setProperty(object sender, PropertyUpdateArgs args)
            {
   try
   {
      // If the target is a control that has been disposed then we don't
      // try to update its proeprties. This can happen if the control is
      // on a form that has been closed while the transition is running...
      if (isDisposed(args.target) == true)
      {
         return;
      }

      ISynchronizeInvoke invokeTarget = args.target as ISynchronizeInvoke;
      if (invokeTarget != null && invokeTarget.InvokeRequired)
      {
         // There is some history behind the next two lines, which is worth
         // going through to understand why they are the way they are.

         // Initially we used BeginInvoke without the subsequent WaitOne for
         // the result. A transition could involve a large number of updates
         // to a property, and as this call was asynchronous it would send a
```

```
            // large number of updates to the UI thread. These would queue up at
            // the GUI thread and mean that the UI could be some way behind where
            // the transition was.

            // The line was then changed to the blocking Invoke call instead. This
            // meant that the transition only proceded at the pace that the GUI
            // could process it, and the UI was not overloaded with "old" updates.

            // However, in some circumstances Invoke could block and lock up the
            // Transitions background thread. In particular, this can happen if the
            // control that we are trying to update is in the process of being
            // disposed - for example, it is on a form that is being closed. See
            // here for details:
            //     http://social.msdn.microsoft.com/Forums/en-US/winforms/thread/7d2c941a-0016-431a-abba-67c5d5dac6a5

            // To solve this, we use a combination of the two earlier approaches.
            // We use BeginInvoke as this does not block and lock up, even if the
            // underlying object is being disposed. But we do want to wait to give
            // the UI a chance to process the update. So what we do is to do the
            // asynchronous BeginInvoke, but then wait (with a short timeout) for
            // it to complete.
            IAsyncResult asyncResult = invokeTarget.BeginInvoke(new EventHandler<PropertyUpdateArgs>(setProperty), new object[] { sender, args });
            asyncResult.AsyncWaitHandle.WaitOne(50);
        }
        else
        {
            // We are on the correct thread, so we update the property...
            args.propertyInfo.SetValue(args.target, args.value, null);
```

```
        }
    }
    catch (Exception)
    {
        // We silently catch any exceptions. These could be things like
        // bounds exceptions when setting properties.
    }
            }

/// <summary>
/// Returns true if the object passed in is a Control and is disposed
/// or in the process of disposing. (If this is the case, we don't want
/// to make any changes to its properties.)
/// </summary>
private bool isDisposed(object target)
{
    // Is the object passed in a Control?
    Control controlTarget = target as Control;
    if (controlTarget == null)
    {
        return false;
    }

    // Is it disposed or disposing?
    if (controlTarget.IsDisposed == true || controlTarget.Disposing)
    {
        return true;
    }
    else
```

```
    {
        return false;
    }
}

        #endregion

        #region Private static functions

        /// <summary>
        /// Registers a transition-type. We hold them in a map.
        /// </summary>
        private static void registerType(IManagedType transitionType)
        {
            Type type = transitionType.getManagedType();
            m_mapManagedTypes[type] = transitionType;
        }

        #endregion

        #region Private static data

        // A map of Type info to IManagedType objects. These are all the types that we
    // know how to perform transactions on...
    private static IDictionary<Type, IManagedType> m_mapManagedTypes = new Dictionary<Type, IManagedType>();

    #endregion

        #region Private data
```

```
        // The transition method used by this transition...
        private ITransitionType m_TransitionMethod = null;

        // Holds information about one property on one taregt object that we are performing
        // a transition on...
        internal class TransitionedPropertyInfo
        {
            public object startValue;
            public object endValue;
            public object target;
            public PropertyInfo propertyInfo;
            public IManagedType managedType;

    public TransitionedPropertyInfo copy()
    {
       TransitionedPropertyInfo info = new TransitionedPropertyInfo();
       info.startValue = startValue;
       info.endValue = endValue;
       info.target = target;
       info.propertyInfo = propertyInfo;
       info.managedType = managedType;
       return info;
    }
        }

        // The collection of properties that the current transition is animating...
        private IList<TransitionedPropertyInfo> m_listTransitionedProperties = new List<TransitionedPropertyInfo>();
```

```
            // Helps us find the time interval from the time the transition starts to each timer
tick...
            private Stopwatch m_Stopwatch = new Stopwatch();


    // Event args used for the event we raise when updating a property...
            private class PropertyUpdateArgs : EventArgs
            {
                public PropertyUpdateArgs(object t, PropertyInfo pi, object v)
                {
                    target = t;
                    propertyInfo = pi;
                    value = v;
                }
                public object target;
                public PropertyInfo propertyInfo;
                public object value;
            }


    // An object used to lock the list of transitioned properties, as it can be
    // accessed by multiple threads...
    private object m_Lock = new object();


            #endregion
        }
}
```

We used some methods from the class above, in order to make some animation in our application, especially in the Menu.

Menu Class:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Transitions;

namespace CMPE412_Project
{
   public partial class Menu : MetroFramework.Forms.MetroForm
   {
      private const string STRING_ENGLISH = "Welcome to BASIC PM";
      private const string STRING_FRENCH = "Bienvenue au BASIC PM";
      private const string STRING_ARABIC = "BASIC PM مرحبا بكم في";
      private const string STRING_TURKISH = "BASIC PM Projesine Hoş Geldiniz";

         public Menu()
      {
         InitializeComponent();
      }

      private void metroUserControl1_Load(object sender, EventArgs e)
      {

      }

      private void Form1_Load(object sender, EventArgs e)
      {
       Label_location=lblTextTransition1.Location.X;
       timer1.Start();
      }

      private void metroTile1_Click(object sender, EventArgs e) //click on Make new project
      {
         MetroFramework.Forms.MetroForm f = new Project();
         f.Show();
         this.Hide();

      }

      private void Title_2_Click(object sender, EventArgs e) //click on open existing project
      {
         OpenFileDialog op = new OpenFileDialog();
         //op.Filter = "Ale Files (.ale) | *.ale | Text Files (.txt)|*.txt";
```

```
        if (op.ShowDialog() == System.Windows.Forms.DialogResult.OK)
        {
            Program.path = op.FileName;
        }
        if(Program.path!="")
        {
            MetroFramework.Forms.MetroForm f = new Basic_PM();
            f.Show();
            this.Hide();
        }

    }


    public int Label_location;
    private void timer1_Tick(object sender, EventArgs e) //Animation
    {

        string strText1;
        if (lblTextTransition1.Text == STRING_ENGLISH)
            strText1 = STRING_FRENCH;
        else
            if (lblTextTransition1.Text == STRING_FRENCH)
        {
            strText1 = STRING_ARABIC;
            Label_location += 9;
            lblTextTransition1.Location = new Point(Label_location,
lblTextTransition1.Location.Y);
            Label_location -= 9;
        }
             else
              if (lblTextTransition1.Text == STRING_ARABIC)
               {
                  strText1 = STRING_TURKISH;
                  Label_location -= 15;
                  lblTextTransition1.Location = new Point(Label_location,
lblTextTransition1.Location.Y);
                  Label_location += 15;
               }
              else
                //if (lblTextTransition1.Text == STRING_TURKISH)
                //{
                //    strText1 = STRING_GREEK;
                //    Label_location -= 24;
                //                lblTextTransition1.Location = new Point(Label_location,
lblTextTransition1.Location.Y);
                //    Label_location += 24;
                //}
                //else
                {
```

```
                strText1 = STRING_ENGLISH;
                lblTextTransition1.Location = new Point(Label_location,
lblTextTransition1.Location.Y);
            }



        // We create a transition to animate all four properties at the same time...
        Transition t = new Transition(new TransitionType_Linear(2000));
        t.add(lblTextTransition1, "Text", strText1);
        // t.add(lblTextTransition2, "Text", strText2);
        t.run();

    }

    private void metroTile3_Click(object sender, EventArgs e) //click on Exit
    {
        if(MessageBox.Show("Are you sure that you want to exit
?","Confirmation",MessageBoxButtons.YesNo,MessageBoxIcon.Question)==DialogResult.
Yes)
        {
            timer1.Stop();
            Application.Exit();
        }
    }

    private void metroTile2_Click(object sender, EventArgs e) //click on about team.
    {
        MessageBox.Show("EMU STUDENTS", "About team", MessageBoxButtons.OK,
MessageBoxIcon.Information);
    }
  }
}
```

Program Class (Global and shared class)

in this class we put all the information that must be shared among all forms.

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace CMPE412_Project
{

  static class Program
```

```
{
    public static int number_of_working_hours;
    public static  DateTime Starting_time;
    public static string path;
    public static string name;
    public static string type;
    public static string description;
    public static DateTime start;
    public static DateTime finish;
    public static string country;
    public static string currency;

        public static DataSet ods = new DataSet();
    /// <summary>
    /// The main entry point for the application.
    /// </summary>
    [STAThread]

    static void Main()
    {

        DataTable task = new DataTable("task");
        DataColumn task_name = new DataColumn("task_name");
        task_name.DataType =Type.GetType("System.String") ;
        DataColumn task_duration = new DataColumn("task_duration");
        task_duration.DataType = Type.GetType("System.Int32");
        DataColumn task_duration_type = new DataColumn("task_duration_type");
        task_duration_type.DataType = Type.GetType("System.String");
        DataColumn task_starting_time = new DataColumn("task_starting_time");
        task_starting_time.DataType = Type.GetType("System.DateTime");
        DataColumn task_finishing_time = new DataColumn("task_finishing_time");
        task_finishing_time.DataType = Type.GetType("System.DateTime");


        task.Columns.Add(task_name);
        task.Columns.Add(task_duration);
        task.Columns.Add(task_duration_type);
        task.Columns.Add(task_starting_time);
        task.Columns.Add(task_finishing_time);



        DataTable resources = new DataTable("resources");
        DataColumn res_name = new DataColumn("res_name");
        res_name.DataType = Type.GetType("System.String");
        DataColumn res_salary = new DataColumn("res_salary");
        res_salary.DataType = Type.GetType("System.Double");
        DataColumn res_email = new DataColumn("res_email");
        res_email.DataType = Type.GetType("System.String");
        resources.Columns.Add(res_name);
```

```
            resources.Columns.Add(res_salary);
            resources.Columns.Add(res_email);


            DataTable Tas_res = new DataTable("Tas_res");
            DataColumn tres_tname = new DataColumn("tres_tname");
            tres_tname.DataType = Type.GetType("System.String");
            DataColumn tres_rname = new DataColumn("tres_rname");
            tres_rname.DataType = Type.GetType("System.String");
            Tas_res.Columns.Add(tres_tname);
            Tas_res.Columns.Add(tres_rname);

            DataTable Tas_dep = new DataTable("Tas_dep");
            DataColumn Tas_dep_t = new DataColumn("Tas_dep_t");
            Tas_dep_t.DataType = Type.GetType("System.String");
            DataColumn Tas_dep_tp = new DataColumn("Tas_dep_tp");
            Tas_dep_t.DataType = Type.GetType("System.String");
            Tas_dep.Columns.Add(Tas_dep_t);
            Tas_dep.Columns.Add(Tas_dep_tp);
            ods.Tables.Add(task);
            ods.Tables.Add(resources);
            ods.Tables.Add(Tas_res);
            ods.Tables.Add(Tas_dep);
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Menu ());
        }
    }
}
```

Project Class (this form is opened when the user want to make new project)

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using DevComponents.DotNetBar;
using System.IO;
using System.Reflection;

namespace CMPE412_Project
{

    public partial class Project : MetroFramework.Forms.MetroForm
    {
        string Succes_icon, failure_icon,countries;
        ErrorProvider err = new ErrorProvider();
```

```
ErrorProvider err1 = new ErrorProvider();
ErrorProvider err2 = new ErrorProvider();
int count_filled = 0;

public Project()
{
   InitializeComponent();
}

private void Project_Load(object sender, EventArgs e)
{
   currencyText.Enabled = false;
   directoryText.Enabled = false;
   string line;
   string executableLocation = Path.GetDirectoryName(
      Assembly.GetExecutingAssembly().Location);
       Succes_icon = Path.Combine(executableLocation, "suc.ico");
      failure_icon = Path.Combine(executableLocation, "fail.ico");
      countries = Path.Combine(executableLocation, "Countries.txt");
          StreamReader file =
          new StreamReader(countries);
   while (true)
   {
       line = file.ReadLine();
      if (line == null)
          break;
      String[] t = line.Split(';');
      countryCmbox.Items.Add(t[0]);

   }
   countryCmbox.DropDownHeight = countryCmbox.Font.Height * 6;
   file.Close();

}

private void timer1_Tick(object sender, EventArgs e)
{
   if(nameText.Text=="Name" && nameText.Focused==true)
   {
      nameText.Text = "";
   }
}



private void panel1_Paint(object sender, PaintEventArgs e)
{
   metroButton5.Enabled = false;
   timer1.Start();
}
```

```
private void metroButton5_Click(object sender, EventArgs e)
{
   try
   {
      Program.path = directoryText.Text + @"\" + nameText.Text + ".ale";
      FileStream fs = File.Create(Program.path);
      fs.Close();
      Program.number_of_working_hours = int.Parse(nmb.Text);
      Program.start = startDate.Value;
      Program.country = countryCmbox.Text;
      Program.currency = currencyText.Text;
      if (typeText.Text != null || typeText.Text != "")
      {
         Program.type = typeText.Text;
      }
      if (descriptionText.Text != null || descriptionText.Text != "")
      {
         Program.type = descriptionText.Text;
      }
      MetroFramework.Forms.MetroForm f = new Interface();
      f.Show();
      this.Hide();
   }catch(Exception e1)
   {
      MessageBox.Show(e1.Message);
   }


}



private void metroButton1_Click(object sender, EventArgs e)
{
   FolderBrowserDialog op = new FolderBrowserDialog();


   if (op.ShowDialog() == DialogResult.OK)
   {
      string path = op.SelectedPath;
      directoryText.Text = path;
   }

}

private void timer1_Tick_1(object sender, EventArgs e)
{
   if(nameText.Text=="" || startDate.Text == "" || countryCmbox.Text == "" ||
      currencyText.Text == "" || nmb.Text == "" || directoryText.Text=="")
```

```
    {
        metroButton5.Enabled = false;
    }
    else
    {
        metroButton5.Enabled = true;
    }
}

private void countryCmbox_SelectedIndexChanged(object sender, EventArgs e)
{
    string executableLocation = Path.GetDirectoryName(
      Assembly.GetExecutingAssembly().Location);
    countries = Path.Combine(executableLocation, "Countries.txt");
    StreamReader file =
    new StreamReader(countries);
    String country = countryCmbox.Text ;

    while (true)
    {
        String line = file.ReadLine();
        String[] t = line.Split(';');
        if (t[0] == country)
        {
            currencyText.Text = t[1];
            break;
        }

    }
}

private void metroButton2_Click(object sender, EventArgs e)
{
    MetroFramework.Forms.MetroForm f = new Menu();
    f.Show();
    this.Hide();

}

private void metroButton3_Click(object sender, EventArgs e)
{
    nameText.Clear();
    typeText.Clear();
    startDate.Value = DateTime.Now;
    descriptionText.Clear();
    countryCmbox.Text = "";
    currencyText.Clear();
    nmb.Clear();
    directoryText.Clear();
}
```

```
        private void countryCmbox_TextChanged(object sender, EventArgs e)
        {
            if (countryCmbox.Text == "" || countryCmbox.Text == null)
            {

                err2.SetError(countryCmbox, "Missing");
                err2.Icon = new Icon(failure_icon, new Size(8, 8));
                err2.BlinkStyle = ErrorBlinkStyle.AlwaysBlink;
                count_filled--;

            }
            else
            {
                err2.Clear();
                err2.SetError(countryCmbox, "This field had been filled successfully");
                err2.Icon = new Icon(Succes_icon);
                err2.BlinkStyle = ErrorBlinkStyle.NeverBlink;
                count_filled++;
            }
        }

        private void nameText_TextChanged(object sender, EventArgs e)
        {

            if (nameText.Text == "" || nameText.Text == null){

                err.SetError(nameText, "working");
                err.Icon = new Icon(failure_icon, new Size(8, 8));
                err.BlinkStyle = ErrorBlinkStyle.AlwaysBlink;
                count_filled--;

            }
            else
            {
                err.Clear();
                err.SetError(nameText, "This field had been filled successfully");
                err.Icon = new Icon(Succes_icon);
                err.BlinkStyle = ErrorBlinkStyle.NeverBlink;
                count_filled++;
            }
        }
    }
}
```

Critical path Function:

```
private int critical_path()
```

```
{
    DataTable task = new DataTable("task");
    DataColumn task_name = new DataColumn("task_name");
    task_name.DataType = Type.GetType("System.String");
    DataColumn task_duration = new DataColumn("task_duration");
    task_duration.DataType = Type.GetType("System.Int32");
    DataColumn task_duration_type = new DataColumn("task_duration_type");
    task_duration_type.DataType = Type.GetType("System.String");
    DataColumn task_starting_time = new DataColumn("task_starting_time");
    task_starting_time.DataType = Type.GetType("System.DateTime");
    DataColumn task_finishing_time = new DataColumn("task_finishing_time");
    task_finishing_time.DataType = Type.GetType("System.DateTime");
    task.Columns.Add(task_name);
    task.Columns.Add(task_duration);
    task.Columns.Add(task_duration_type);
    task.Columns.Add(task_starting_time);
    task.Columns.Add(task_finishing_time);
    DataTable Tas_dep = new DataTable("Tas_dep");
    DataColumn Tas_dep_t = new DataColumn("Tas_dep_t");
    Tas_dep_t.DataType = Type.GetType("System.String");
    DataColumn Tas_dep_tp = new DataColumn("Tas_dep_tp");
    Tas_dep_tp.DataType = Type.GetType("System.String");
    Tas_dep.Columns.Add(Tas_dep_t);
    Tas_dep.Columns.Add(Tas_dep_tp);

    DataTable flip = new DataTable("flip");
    DataColumn fatherr = new DataColumn("fatherr");
    fatherr.DataType = Type.GetType("System.String");
    DataColumn sonn = new DataColumn("sonn");
```

```
sonn.DataType = Type.GetType("System.String");
flip.Columns.Add(fatherr);
flip.Columns.Add(sonn);
DataTable dep = new DataTable("dep");

DataColumn f = new DataColumn("f");
Tas_dep_t.DataType = Type.GetType("System.String");

DataColumn s = new DataColumn("s");
Tas_dep_tp.DataType = Type.GetType("System.String");

dep.Columns.Add(f);
dep.Columns.Add(s);

ods.Tables.Add(task);
ods.Tables.Add(Tas_dep);
ods.Tables.Add(flip);
ods.Tables.Add(dep);

graph.CleanNodes();
for (int k=0;k<ods.Tables[0].Rows.Count;k++)
{
   ods.Tables[0].Rows.RemoveAt(k);
}
for (int k = 0; k < ods.Tables[1].Rows.Count; k++)
{
   ods.Tables[1].Rows.RemoveAt(k);
}
for (int k=0;k<Program.ods.Tables[0].Rows.Count;k++)
```

```
{
    string name = Program.ods.Tables[0].Rows[k][0].ToString();
    int dur = int.Parse(Program.ods.Tables[0].Rows[k][1].ToString());
    string type= Program.ods.Tables[0].Rows[k][2].ToString();
    switch (type)
    {

        case "Month": { dur = dur * 30; } break;
        case "Hour":
            {
                double duration_ = ((double)dur / Program.number_of_working_hours);
                dur = (int)Math.Ceiling(duration_);
            }
            break;
        case "Week": { dur = dur * 7; } break;

    }
    string d1 = Program.ods.Tables[0].Rows[k][3].ToString();
    string d2 = Program.ods.Tables[0].Rows[k][4].ToString();
    string[] d11 = d1.Split(' ');

    DateTime st = DateTime.Parse(d11[0]);

    //DateTime fn= null;
    ods.Tables["task"].Rows.Add(name,dur,type,st);
}
for (int k = 0; k < Program.ods.Tables[3].Rows.Count; k++)
{
    string col1 = Program.ods.Tables[3].Rows[k][0].ToString();
```

```
    string col2= Program.ods.Tables[3].Rows[k][1].ToString();
    ods.Tables[1].Rows.Add(col1, col2);
}

for (int q = 0; q < Tas_dep.Rows.Count; q++)
{
    flip.Rows.Add(null, null);
    flip.Rows[q][0] = Tas_dep.Rows[q][1];
    flip.Rows[q][1] = Tas_dep.Rows[q][0];
}

for (int q = 0; q < flip.Rows.Count; q++)
{
    dep.Rows.Add(null, null);
    dep.Rows[q][0] = flip.Rows[q][0];
    dep.Rows[q][1] = flip.Rows[q][1];

}

for (int q = 0; q < task.Rows.Count; q++)//for father
{
    int y = 0;
    for (int w = 0; w < flip.Rows.Count; w++)//if there is father
    {
        if (task.Rows[q][0] == flip.Rows[w][1]) y = 1;
    }
    if (y != 1)
    {
        dep.Rows.Add(null, task.Rows[q][0]);
```

```
    }
    y = 0;

}
for (int q = 0; q < task.Rows.Count; q++)//for son
{
    int y = 0;
    for (int w = 0; w < flip.Rows.Count; w++)//if there is son
    {
        if (task.Rows[q][0] == flip.Rows[w][0]) y = 1;
    }
    if (y != 1)
    {
        dep.Rows.Add(task.Rows[q][0], null);
    }
    y = 0;

}
Console.WriteLine("this is the dep that u are going to use\n");
for (int q = 0; q < dep.Rows.Count; q++)
{

    Console.WriteLine("fa:{0}  son:{1}\n", dep.Rows[q][0], dep.Rows[q][1]);
}



int n;
n = ods.Tables[0].Rows.Count;
int i;
```

```
init(n);
net_work[0].flag = 1;
net_work[n + 1].flag = 1;
net_work[0].early_start = 0;
net_work[0].early_end = 0;
for (i = 1; i < n + 1; i++)
{
   int x = int.Parse(ods.Tables[0].Rows[i - 1][1].ToString());
   net_work[i].value = x;
}
int m;
m = dep.Rows.Count;
int fa = new int(), son = new int();
for (i = 0; i < m; i++)
{

   string sonName, sonNameToCheck, fatherName, fatherNameToCheck;
   //for the father
   for (int j = 0; j < n; j++)
   {
      fatherName = dep.Rows[i][0].ToString();
      fatherNameToCheck = ods.Tables[0].Rows[j][0].ToString();
      if (fatherName == "")
      {
         fa = 0;
         break;
      }

      if (fatherName == fatherNameToCheck)
```

```
        {
            fa = j + 1;
            break;
        }// comparison for parent to find its number from a string
    }

    // for the son
    for (int j = 0; j < n; j++)
    {

        sonName = dep.Rows[i][1].ToString();
        sonNameToCheck = ods.Tables[0].Rows[j][0].ToString();
        if (sonName == "")
        {
            son = n + 1;
            break;
        }
        if (sonName == sonNameToCheck)
        {
            son = j + 1;
            break;
        }
    }

    //fa = int.Parse(ods.Tables[1].Rows[i][0].ToString());
    //son = int.Parse(ods.Tables[1].Rows[i][1].ToString());
    add_node(fa, son);
}
```

```
for (int z = 0; z < n + 1; z++)
{
   Console.WriteLine("value:{0}\n", net_work[z].value);
}
topological_order();
show_item(n);
net_work[n + 1].last_start = net_work[n + 1].early_start;
net_work[n + 1].last_end = net_work[n + 1].early_end;
inverse_topological_order(n + 1);

show_item(n);

//create a form
System.Windows.Forms.Form form = new System.Windows.Forms.Form();
//create a viewer object
        Microsoft.Glee.GraphViewerGdi.GViewer viewer = new
Microsoft.Glee.GraphViewerGdi.GViewer();
//create a graph object
//create the graph content
//graph.AddEdge("Start", "T1");

for (i = 0; i < m; i++)
{
   if (dep.Rows[i][0].ToString() == "")
   {
      graph.AddEdge("start", dep.Rows[i][1].ToString());
      continue;
   }
   if (dep.Rows[i][1].ToString() == "")
   {
```

```
            graph.AddEdge(dep.Rows[i][0].ToString(), "end");
            continue;
        }

        graph.AddEdge(dep.Rows[i][0].ToString(),  dep.Rows[i][1].ToString()).Attr.Color
= Microsoft.Glee.Drawing.Color.Green;
    }

    //graph.AddEdge("Start", "End").Attr.Color = Microsoft.Glee.Drawing.Color.Green;
    //graph.FindNode("start").Attr.Fillcolor = Microsoft.Glee.Drawing.Color.Magenta;

    find_critical_path(0);
    //graph.FindNode("T1").Attr.Fillcolor = Microsoft.Glee.Drawing.Color.MistyRose;
    //Microsoft.Glee.Drawing.Node c = graph.FindNode("End");
    //c.Attr.Fillcolor = Microsoft.Glee.Drawing.Color.PaleGreen;
    //c.Attr.Shape = Microsoft.Glee.Drawing.Shape.Diamond;
    //bind the graph to the viewer
    viewer.Graph = graph;

    //associate the viewer with the form
    form.SuspendLayout();
    viewer.Dock = System.Windows.Forms.DockStyle.Fill;
    form.Controls.Add(viewer);
    form.ResumeLayout();
    //show the form
    form.ShowDialog();

    return 0;
}
```

We believe that this is enough for the coding, our CD which will be delivered with this document will have all the source code.

The other remaining classes has more than 3000 lines of code, it will be too much to include it in this stage, and a little bit hard to be understood.