Informatics Institute of Technology

In Collaboration with

University of Westminster, UK

Intelligent Sorting Mechanism using Multiobjective Optimization For Effective Task Sorting in Project Management

A dissertation by

Vinusha Perera

Supervised by

Mr. Achala Aponsu

Submitted in partial fulfillment of the requirements for the

BSc (Hons) Software Engineering degree

Department of Computing

May 2018

Declaration

I hereby certify that this project report and all the artifacts associated with it is my own work and it has not been submitted before nor is currently being submitted for any degree programme.

Full Name of the Student:

Registration Number:

Signature: …………………………………….. Date: ……………………………………..

Abstract

Effectively managing tasks is one of the key aspects of an ongoing project. By managing the tasks effectively, chance of failing the project comes to a minimum risk. Furthermore this could be beneficial to both the project owner and the project undertaking company. In order to effectively manage the tasks, a person with a good understanding of the project needs manage the overall project. Through an in-depth literature survey, a new system is proposed to increase the efficiency of this manually done process.

For this research a new approach to conventional task sorting mechanism was introduced based on multiobjective optimization algorithms. The proposed system is capable of sorting given number of tasks considering multiple objectives producing more efficient results compared to the conventional sorting mechanisms.

Keywords:

Multiobjective Optimization, Non-dominated Sorting

Acknowledgment

I would like to give my special thanks to my supervisor, Mr. Achala Aponsu, who guided me through the project from the beginning. Also I would like to thank the lecture panel of Informatics Institute of Technology for the guidance provided during this project. Finally I would like to give my special thanks to my loving parents for the support they gave me to make this project a success. It would not have been possible without the support of these incredible people.

Table of Contents

Table of Tables

Table of Figures

Chapter 01

Introduction

1.1 Chapter Overview

This chapter contains the introduction to the project and its background in order to give a clear understanding about the problem domain this project addresses. This detailed introduction will provide a better understanding of the project background, problem domain, its scope, aims, objectives, resource requirements and the features of the prototype. It also highlights the research and development methodologies which are to use and the timeline of the project.

1.2 Project Background

Project management is a field which incorporates with lots of other sectors, from manufacturing to services. When it comes to managing a project there are lots of key aspects to consider in each phase. Not only the project planning phase, but also the project execution stage is equally important since this is the phase most of the projects fail. (Discenza, Forman, 2007) There could be many reasons behind this and some of the reasons may even lie within the project planning stage. However when considering only the project execution stage, there are several important things to be considered and one of which is the task management. (Masood, Hoda, Blincoe, 2017) This may not lead projects directly to failure, but this could cause many other issues in terms of the project timeline, cost and overall experience with the client.

Nowadays there are lots of talented project managers out there who are experts in managing projects. Since most of the companies which handle projects have project managers working for them, most of the projects are properly planned and executed. Even though this is a hectic task, project managers oversees the project from the project planning phase through the execution phase to the concluding phase. When it comes to the execution phase of the project, it gets complicated even though the whole project is properly planned. (Gosenheimer, 2017) This happens due to the needs of the project owners, who might need to add new things to the project or make amendments to the existing features. According to a study done by Project Management Institute also known as PMI, this is a one of many reasons to a project to fail in the execution phase. (Ika, 2012)

Since many issues could arise in the project execution phase it is not that easy to manage all the task of a project. When it comes to agile development methodologies like scrum, this gets a little easier since only a portion of the project backlog will be taken for a sprint. In this taken portion of tasks, still there are a lot of things to consider when arranging the tasks in a specific order to complete. This sorting is usually done by a project manager or someone who has the overall picture of the project, considering many aspects from timeframe to the resources available. However this task is not an easy one since there could be a lot of tasks in an active project with various deadlines, priorities and task dependencies. In this stage of a project, it is crucial to properly arrange the remaining task, which can save a lot of time leaving more time to properly test the functionalities or to evaluate the overall project properly.

With the growth of the IT industry, more and more tools have been developed over time to easily manage projects. These tools have lots of helpful features to plan and schedule a project, track the time, report issues, budget and map team collaboration. When managing active tasks of a project these tools can be really helpful because it allows the users to easily collaborate with others and do things through the tool itself. (Dayani, 2017) However one of the major drawbacks of these tools is the lack of sorting options they provide when it comes to task management. These tools only provide basic sorting mechanism where the user can sort using one field or the other. Even though these tools can help the project managers in various ways, still they have to arrange the tasks to be done considering multiple objectives, as a basic sorting mechanism will not take those in to account.

When it comes to agile planning of a project, there are various ways to decide the complexity of a task. (Coelho and Basu, 2012) Story points system is one of the common ways to do it but it does not give a clear picture on the project timeline. Time allocations for the tasks can be different from task to task even though it has the same story point value. However the story points can be a big help when planning a project and distributing the tasks since it gives an idea about the complexity of that specific task.

In order to get over this problem, an application can be developed to effectively sort the tasks provided in an efficient way using problem solving algorithms. This sorting mechanism should consider more objectives in order to increase the accuracy of the sorted list so it will increase the efficiency of the overall project. This is very important considering the fact that even the project managers might not be able to arrange the tasks in a most efficient manner when the number of tasks and the objectives increase. However with an application, with a sorting mechanism which consider multiple objectives to produce an optimal sorted list, not only the project managers, even the stakeholders might get benefits out of it.

1.3 Project Aim

Aim of this project is to research, design, implement and evaluate a prototype application to sort tasks in a project considering the given factors maximizing the effectiveness of the project. The solution proposed will help improving the efficiency of a project by effectively laying out the tasks at hand so the waiting time and unnecessary conflicts get minimized.

1.4 Scope of the Project

1.4.1 Inclusions

1. Solution will allow the users to change the sorting parameters according to their needs.
2. Solution will allow users to export the sorted tasks in Excel format.
3. Solution will produce an optimal set of sorted items.
4. Solution will allow the user to select from multiple solutions if needs be.
5. Solution will allow the user to switch between producing the optimal sorted list and the optimal sorted lists set.

1.4.2 Exclusions

1. Solution will not be integrated to any project management tools.
2. Solution will not consider the priority level of tasks on its own.
3. Solution will not consider anything other than the sorting parameters when sorting.

1.5 Project Objectives

1. Propose an efficient sorting mechanism using multiobjective optimization and genetic algorithms to produce the most efficient work backlog in projects.
2. Conduct an in-depth research to understand the existing solutions, existing procedures and algorithms used, drawbacks of using the specified algorithms and perform surveys to get the feedback from general users.
3. Conduct a thorough literature survey to evaluate the past research material on multiobjective optimization sorting and genetic algorithms for a better understanding of the domain.
4. Gather functional and non-functional requirements and special requirements of the stakeholders by discussing with the domain experts and considering the information received from the surveys conducted.
5. Finalize the features of the proposed system considering all the requirements and user experience aspects.
6. Design an efficient system for sorting the work backlog of a project considering all the functional, non-functional requirements identified.
7. Implement a prototype showcasing the identified functional, non-functional requirements and certain user experience aspects.
8. Evaluate the prototype thoroughly by making a complete set of test cases testing each and every aspect of the prototype.
9. Complete the documentation of the prototype.

1.6 Features of Prototype

High level features of the proposed system is listed below.

1. **Feeding the tasks –** Users can input the tasks to the system manually or using Microsoft Excel format so they can directly export the list from their project management tool and import it to the proposed system.
2. **Sorting the task using proposed mechanism –** Users can sort the imported list of tasks using the new mechanism introduced in this project.
3. **Exporting the tasks –** Users can export the tasks list in the same format that they have given to the software so they can easily import it back to their project management tool.
4. **Tune the sorting mechanism –** Users will be allowed to fine tune the sorting mechanism to adjust into new scenarios and situations.
5. **Add additional parameters** – Users will be allowed add more parameters to the existing mechanism.
6. **Manage the sorted tasks –** By creating an account users can save the sorted tasks and use them later so they do not have to worry about data lose. This will also allow to manage the tasks using the proposed system to some extent.

Below figure shows an overview for the proposed system.

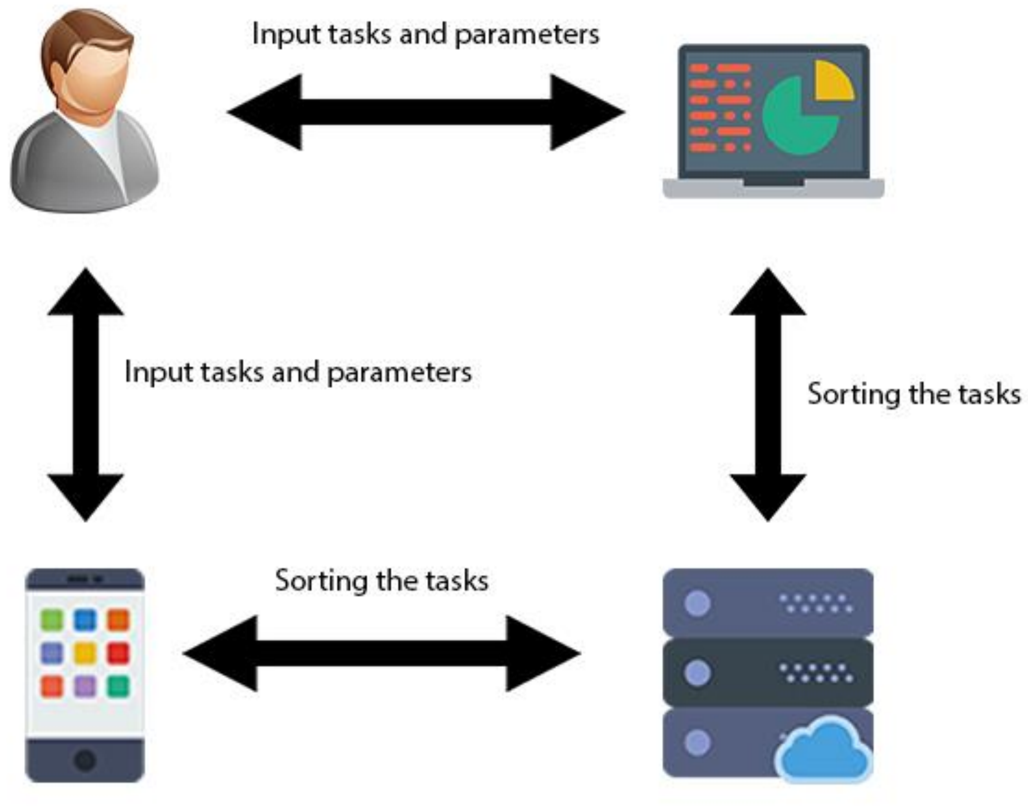


Figure 1.1 Overview of the System

1.7 Resource Requirements

Minimum system requirements for this application development is stated below. These requirements may change in the future as the project moves forward.

1.7.1 Hardware Requirements

- Intel core i3 processor
- 8 GB RAM
- 1 GB VGA
- 100 GB of HDD

1.7.2 Software Requirements

- Visual Studio 2015
- Astah UML Design tool
- GitHub
- Microsoft Word 2013
- Microsoft Project 2013
- Microsoft PowerPoint 2013

1.8 Research and Development Methodology

1.8.1 Research Methodology

The main goal of this project is to find a best sorting mechanism using multiobjective optimization and genetic algorithms for the given number of tasks. Since this focus on finding a new mechanism, inductive research methodology will be chosen.

1.8.2 Software Development Methodology

After comparing some available methodologies, this project will take Iterative and incremental development methodology. Due to the limited time period, single person development approach and the research nature of this project, this development method will be the most suitable since it allows the developer to add and refine the features in each iteration as the project moves on. Below is a brief comparison of the selected development process models.

Table 1.1 Software Development Methodology Comparison

Model	Pros	Cons
Waterfall	• Strictly structured. • Simple and easy to follow. • Easy to manage the phases.	• All the requirements need to be gathered before starting the development. • Cannot go back and make changes.
Iterative and Incremental	• Add and refine features at every iteration. • Working prototype can be produced after each iteration. • Iterations can be tested separately for the specific feature testing.	• Design conflicts might arise since the requirements are not upfront. • May take more time.
Spiral	• Additional functionalities can be added later. • Risks can be avoided mostly.	• Success of the project is highly dependent on the risk analysis phase.

	• Prototype is built early and therefore the final version will be very stable.	• Not suitable for small projects. • Highly specific expertise is needed in risk analysis.
Prototype	• Errors can be identified in early stages. • Missing functionalities can be identified easily. • Final version will be very well refined due to many iterative prototyping.	• May increase the complexity when giving patches to the identified bugs. • Scope might get increased.

1.8.3 Project Management Methodology

Considering the fact that this project is a research based one, the best approach would be the PRINCE2 methodology. Since this project focuses a lot on the designing phase due to the research aspects, it would be better to move on with a methodology like this. When it comes to Agile like methodologies, it is necessary to have a prototype from the early stages onwards and this will be a bit hard considering the limited time frame this project has.

Table 1.2 Project Management Methodology Comparison

Methodology	Pros	Cons
Waterfall	• Easy to follow. • Each phase is strictly structured.	• All the project requirements need to be gathered to start the project. • Cannot go back and make changes after that phase is passed.
Agile	• Working prototype is always there. • Changes can be done rather easily. • Can keep track with the research aspect of the project.	• Scope might get increased. • Cannot get a time estimation due to lack of planning.
Scrum	• Product is available from the early stages of development.	• Difficult to plan the project which lacks a clear definition.

	• Testing can be conducted thoroughly before moving to the next sprint. • Changes can be made rather easily.	• Scope might get increased. • Depends on a cross functional team.
PRINCE2	• Each stage can be closely monitored and therefore the outcome is predictable. • Utilizes the best practices.	• Complexity of the methodology. • Changes cannot be made easily.

1.8.4 Data Gathering Methodologies

Following data gathering methods have been selected for this project.

Table 1.3 Data Gathering Methodology Comparison

Method	Reason
Domain Expert Interviews	Best way to get the expert knowledge in a specific area and their feedback on the technologies used and the approaching techniques to the proposed system.
Surveys and Questionnaires	Can gather data easily from people all around the world so I can get a clear understanding about the requirements and improving user experience as well.
Research Material Analysis	Easy to find over the internet and more time can be spend learning the new approaches to the problem.
Focus Groups	Can get a clear understanding about the users out there, their needs, special requirements and

	what needs to be done to improve the user experience aspects.

1.9 Chapter Summary

This chapter included a detailed introduction of the project and its aims and objectives. At the end of this project an application will be developed to efficiently sort the given number of tasks considering selected number of important parameters. The important parameters will be limited to a few in order to scope the project. The next chapter will contain a detailed literature survey regarding the problem domain this project addresses.

Chapter 02

Literature Review

2.1 Chapter Overview

This chapter contains more details on the problem domain, existing applications available and also the techniques which can be used to successfully implement this application. In addition, this chapter includes detailed information on available approaches, its use, accuracy and performance.

2.2 Problem Domain

Usually when it comes to a project, the person who is in charge is responsible for planning and distribution of the tasks. This is what happens in most of the projects with the help of the existing tools. However this process is highly dependent on the person who is in charge since the tools mostly used for the recording purpose. When sorting relatively large number of tasks, lots of parameters - needs to be considered. (Filho, Pinheiro, Albuquerque, 2015) Due time, priority level, needed resources, availability of resources and human resource allocations are some of the main parameters that needs to be taken in to consideration. Processing these manually by single person can be quite challenging and therefore it will consume a lot of time. Even though the tasks are sorted according to the person in charge's knowledge, there is no guarantee that it is efficient. (Almeida, Pinheiro, Albuquerque, 2011)

Most common issue here is not taking the task dependencies in to consideration when done manually. Tasks might be arranged considering the priority level of each task and the availability of human and other resources but it might not be efficient when considering the whole project and the time line. (Dayani, 2017) When it comes to the task management tools, the problem is that they only consider a very few parameters in terms of sorting the tasks like due time, priority level and dependencies to a certain extent. These solutions do not consider the dependencies of the tasks and the achievable optimizations to a satisfactory level. A lot can be achieved if more parameters were to consider when developing these tools. At the same time manual effort can be reduced to a greater extent and a lot of time can be saved in the overall project.

2.3 Problem Solving Algorithms

There have been several problem solving algorithm developments and some of them are used to solve famous problems like Travelling Salesman Problem.

Intelligent Water Drops algorithm (Shah-Hosseini, 2009) is one of the algorithms that have been used to solve this problem. It was built observing the way a natural water drop moves in a water source like a river. When water moves from a higher terrain to a lower terrain there is always the shortest path but the water drop will have to adjust its path to get to the destination due to the barriers lie ahead. In order to prove this algorithm, an artificial water drop is being created giving some of the unique aspects of a natural water drop. This water drop has two important properties. First one is the amount of soil it carries at the moment, Soil (IWD) and its current moving velocity, Velocity (IWD). These two properties will have different values as it moves from different environment and this environment depends on the problem at hand. When it comes to applying this to a problem there could be one or more than one solutions, in other terms different paths to reach the destination. If the destination is known, the target of this to find the best path, most probably the shortest one possible. (Shah-Hosseini, 2008) If the destination is unknown the goal is to find the optimum destination based on the other factors given like the cost. When it comes to solving the Travelling Salesman Problem using this algorithm covers fast to optimum solutions and gives promising results.

2.4 Multiobjective Evolutionary Algorithms

In some cases there might not be a single solution satisfying all the objects so an algorithm which gives multiple alternative solutions have a real value (Kalyanmoy D, Samir A, Amrit P, Meyarivan T, 2009). When it comes to a multi objective optimization problem there are several solutions that are superior to the rest of the solutions when the complete set of objectives are taken into consideration but inferior to other solutions in the problem space when one or more objects are considered. These solutions are called as Pareto-optimal or non-dominated solutions (Chankong and Haimes, 1983; Hans, 1988). Since these non-dominated solutions are not absolutely better than any of the solutions chosen, selecting one solution over the others requires a consideration of other factors relating to the problem itself.

One of the common difficulties of the multi objective optimization is the objective conflict since none of the feasible solutions will provide an optimal solution to all the objectives. When it comes to this, most mathematically favored non-dominated solution would be the one which gives least objective conflict comparing to the others. (Deb K., 2015) Even though this seems like a promising way, this might not satisfy the decision maker since that person might have some associated priorities of the objectives. When it comes to this point, most of the classical methods will scalarize the objective vector in to one objective.

However there are some drawbacks when it comes to the classical methods of multi-objective optimization. Most of these classical methods will combine multi objectives to one objective in order to get a non-dominated (Pareto-optimal) solution but it will result in a single point solution. (Lara, Sanchez and Coello, 2009) When it comes to the real world problem solving, more alternatives might be needed for decision making. One of the biggest drawbacks of these classical ways is these methods are very sensitive towards the weights or demands of the objective. When combining these objectives in to a single dimension, the decision maker needs to have a good knowledge in each of the objectives and its priority levels. In this way the same problem needs to be solved more than one time and according to the situation different weight vectors should be used in each case.

In a 1999 study Multiobjective Revolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach by Eckart Zitzler and Lothar Thiele, they have taken 5 algorithms in order to compare them and find the effectiveness of these algorithms. In order to do that they have chosen the multiobjective 0/1 knapsack problem. This was chosen since it is understandable and the experiments can be repeatable on this problem. Also this ideally represent a certain class of a real world problems. Though the problem description is simple enough to understand, solving the problem is quite challenging. In a 0/1 knapsack problem, formulated as a multiobjective optimization problem, consists of a set of items which are associated with a weight and a profit and an upper bound for the capacity of a knapsack. Main task of this is to find a subset which maximizes the total of the profits in the subset and still the selected items should fit to the knapsack. Another main thing to note here is that the total weight should not exceed the given weight.

In the above study they have tested 4 existing algorithms and one new algorithm using nine different problem settings. Achieved non-dominated sets quality was measured quantitatively by the size of the covered space. In addition, these approaches were compared directly taking the outcomes regarding Pareto dominance. In this study all the multiobjective evolutionary algorithms were proven effective against a pure random search strategy. This search strategy randomly generates new points in the search space without exploiting the similarities between the solutions. Besides all the algorithms used in this study, non-dominated sorting genetic algorithm got the best results on all test problems.

2.5 Strength Pareto Evolutionary Algorithm

Apart from all the algorithms presented in this study, the new evolutionary algorithm introduced by this paper (SPEA) has outperformed all in this 0/1 knapsack problem by a huge margin. Strength Pareto Evolutionary Algorithm uses established and new techniques to find multiple Pareto-optimal solutions in parallel. According to the study it has similarities to the existing multiobjective evolutionary algorithms like, it stores the Pareto-optimal solutions found so far externally and it uses Pareto-dominance concept to assign scalar fitness values to individuals. (Sheng et al., 2012) Also it uses clustering to reduce the number of Pareto-optimal solutions found and stored without exploiting the characteristics of the tradeoff front.

Besides all these, Strength Pareto Evolutionary Algorithm uses some unique techniques according to the study. (Zitzler, Thiele, 1999) It uses above techniques by combining those into one and it only determines the fitness of an individual from the solutions stored in the external non-dominated set. Members of the population dominating over each other is irrelevant. In addition to that, all solutions in the external non-dominated set participate in the selection. (Sheng et al., 2012) A new niching method was provided to keep the diversity of the populations as it is. This method which was introduced was a Pareto-based method and it did not require any distance parameter like the niche radius for sharing

Following are the steps of the Strength Pareto Evolutionary Algorithm according to the study.

1. Generate an initial population P and create the empty external non-dominated set $\mathbf{P_1}$.
2. Copy non-dominated members of **P** to $\mathbf{P_1}$.
3. Remove solutions within $\mathbf{P_1}$ which are covered by any other member of $\mathbf{P_1}$.
4. If the number of externally stored non-dominated solutions exceeds a given maximum $\mathbf{N^1}$, prune $\mathbf{P_1}$ by means of clustering.
5. Calculate the fitness of each individual in **P** as well as in $\mathbf{P_1}$.
6. Select individuals from $\mathbf{P+P_1}$ (multi-set union), until the mating pool is filled. In this study, binary tournament selection with replacement is used.
7. Apply problem-specific crossover and mutation operators as usual.
8. If the maximum number of generations is reached, then stop, else go to Step 2.

2.6 Nondominated Sorting Methods

Nondominated sorting methods have been used in literatures for the past few years since Goldberg suggested their use on Multiobjective Evolutionary Algorithms. The first algorithm to adopt this nondominated sorting strategy was Nondominated Sorting Genetic Algorithm, also known as NSGA. In this algorithm each and every solution will be compared with each other and the solutions which are not dominated by others will be assigned to a front. (McClymont and Keedwell, 2012) The solutions which are in this front will be removed from the current solutions temporarily. Then the solutions in the remaining set will be compared with each other and all the nondominated solutions will be assigned to a different front. This process continue till all the solutions in the current population is assigned to a front. Since this method has many redundant comparisons, this will be highly time consuming and computationally inefficient when it comes to large populations.(Ahmed and Deb, 2012) However there is an improved version of this algorithm, where the comparison between two solutions happen only once. Due to this the time complexity comes down to $O(MN^2)$, where the normal NSGA costs $O(MN^3)$.

Recursive nondominated sorting approach was also being introduced using divide and conquer mechanism which reduces the time complexity by a great amount. (McClymont and Keedwell, 2012) When it comes to bi-objective multiobjective optimization problems, time complexity is reduced to $O(MN \log N)$ and for the problems which have more than two objectives it is reduced to $O(N \log^{M-1} N)$. However the problem with this approach is that the time complexity grows exponentially when increasing the number of objectives in a problem. According to this study, Jensen's sort will most likely to consume more runtime in simulation, if there are a high number of objectives, due to its recursive nature. (Ahmed and Deb, 2012) Apart from these, this algorithm will not be applicable in scenarios like strong-dominance or E-dominance is used in the comparison or the population contains duplicate entries.

There are some algorithms which are based on arena's principle in order to assign solutions to a front. In this method, it randomly selects a solution as the arena host and compare all the remaining solutions with the host. One that dominates the arena will be the next arena host and it goes on like that. This method has a better computational efficiency than fast nondominated sort and Jensen's sort. Time and space complexity of this method will be $O(MN^2)$ and $O(N)$. Even though there are other algorithms similar to these, in most cases efficiency drops when the objectives are increased.

2.7 Analyzing the existing nondominated sorting algorithms

Most of the nondominated sorting approaches focus on assigning the available solutions to a front in different manners. However the assignment can only be done for a front at a time and the already assigned solutions must be removed from the population before assigning the others to the next front. When it comes to the nondominated sorting mechanisms (Zhang, Tian, Cheng, Jin, 2015) the main operation is the dominance comparison between the solutions. Efficiency of the sorting approach rely on the number of comparisons been done on the solution sets. In order to maximize the efficiency most of these approaches remove unnecessary comparisons between solutions.

When looked closely, four categories can be identified in these approaches. If two solutions were taken to compare with each other - *pm and pn,* the first case would be the one where *pm* is dominated by *pn* or the other way around. The second case would be the one where *pm* and *pn* both are nondominated and they belong to the same front and that front being the current front. The third case would be very much similar to these where both the solutions are nondominated and belongs to the same front but that front is not being the current front at the moment. The last category would be where both the solutions are nondominated but belongs to different fronts. (Zhang, Tian, Cheng, Jin, 2015) In case one both the solutions will fall into different fronts since one dominates the other, the additional comparisons will not be necessary in this case. In case two, since both the solutions are nondominated and belongs to the same and the current front, comparison between these two should happen to ensure that none would dominate over each other. In addition to that all the solutions in the current front should be compared with each other to make sure that all of them are nondominated to exist in the current front. However when it comes to the third case there will be at least one dominated solution over *pm* and *pn* since both the solutions are not belonging to the current front. Since it is the case, the comparison between these two solutions can be skipped. In case four there will at least be one dominating solution over these two solutions and the comparison between *pm* and *pn* can be skipped.
In a 2015 study, they have introduced an efficient nondominated sorting framework which is different from most of the nondominated approaches in the previous studies. In this approach front of the each solution will be determined one by one where most of the other nondominated approaches determine the front of all solutions on the same front as a whole. When it is performed in this manner, duplicate comparisons can be avoided. Reason behind this is that a solution to be assigned only needs to be compared with the solutions that have already been assigned to a respective front.

2.8 Efficient Nondominated Sorting Algorithm

2.8.1 Unnecessary Comparisons

When it comes to this algorithm there are key plus points comparing to most of the other nondominated sorting algorithms. One of the main advantage of this algorithm is that it does not compare any unnecessary solutions.

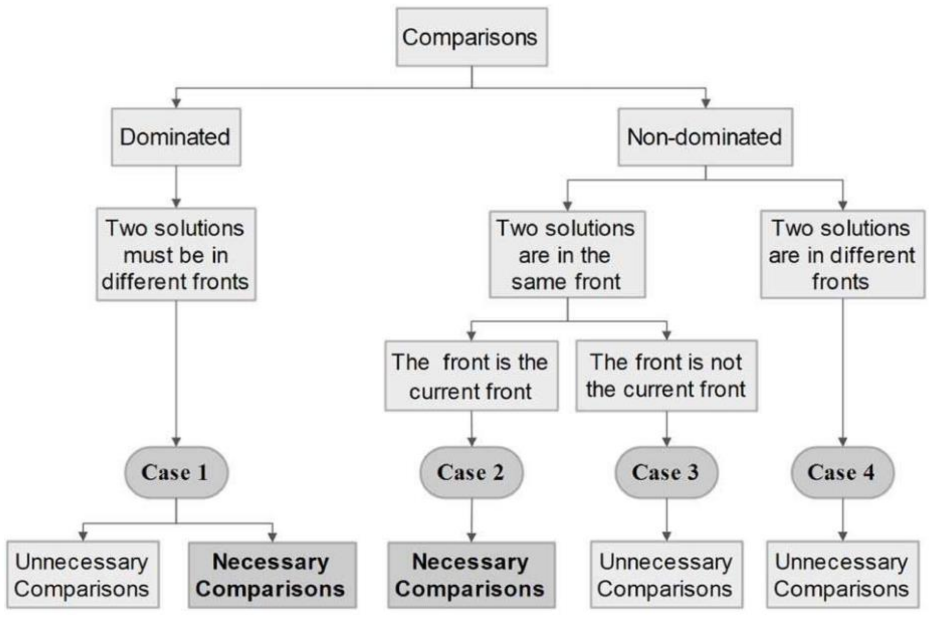

Comparisons
Dominated
Non-dominated
Two solutions must be in different fronts
Two solutions are in the same front
Two solutions are in different fronts
The front is the current front
The front is not the current front
Case 1
Case 2
Case 3
Case 4
Unnecessary Comparisons
Necessary Comparisons
Necessary Comparisons
Unnecessary Comparisons
Unnecessary Comparisons

Figure 2.1 ENS Comparisons

In order to avoid these unnecessary comparisons, the population has to be sorted in ascending order giving priority to one parameter. According to this study, the problem should be a minimization problem in order to this to work. However when the population is sorted into the ascending order, comparison can be started from the bottom of the population so whenever an item in the population is assigned to a front, other less dominating items do not have to be compared with that front.

2.8.2 Computationally Expensive Front Assigning Strategies

Below diagram shows a common strategy for finding fronts for the solutions.

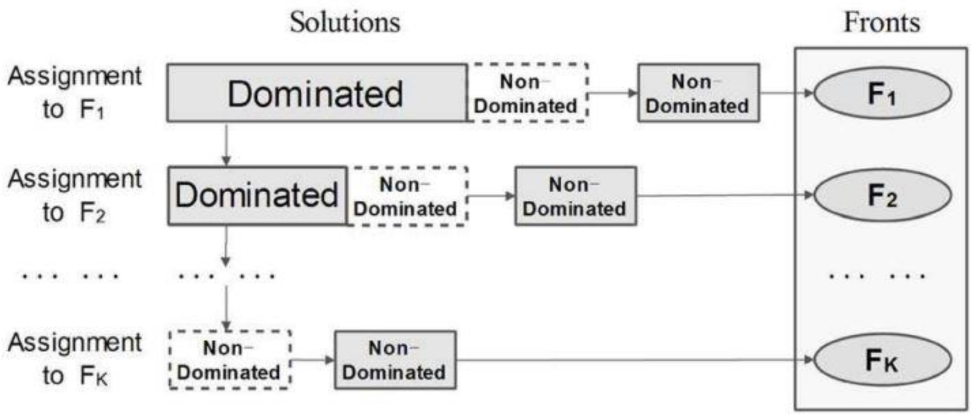


Figure 2.2 Common strategy for finding fronts

In this common strategy, the same procedure has to be performed over and over again until all the solutions in a given population are assigned to a front. Even though this method can assign multiple solutions to a specific front, still it has to loop through the whole population multiple times.
The initial step of this strategy is to create a new front and assign all the solutions in the population to that specific front. After that it checks for the nondominated solutions among the population and all the other solutions are moved out from that specific front and assigns to a new front. This procedure goes on till all the solutions have a front in which there exist no dominating solutions.

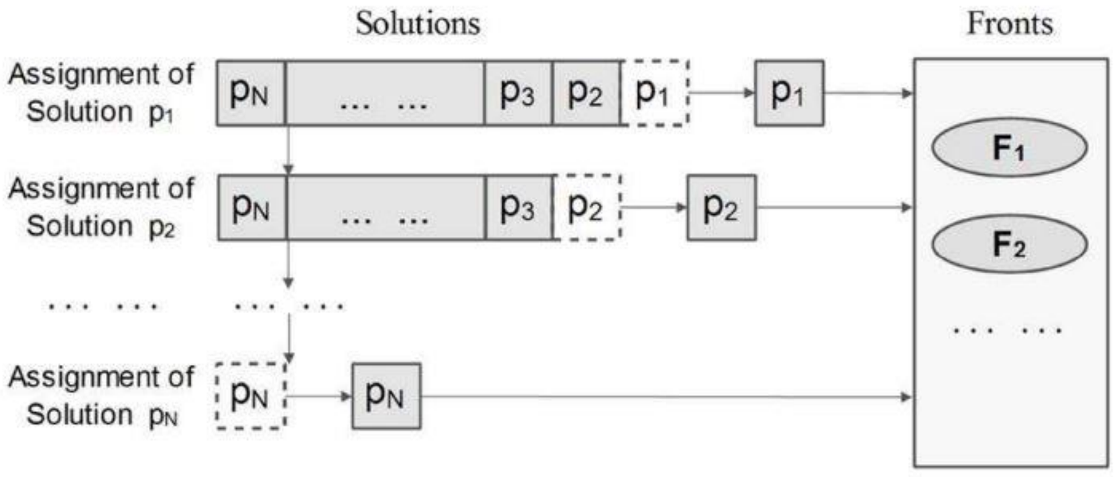

Solutions
Fronts
Assignment of Solution p1
pN p3 p2 p1
p1
Assignment of Solution p2
pN p3 p2
p2
F1
F2
... ...
Assignment of Solution pN
pN
pN

Figure 2.3 Proposed Strategy in ENS algorithm

By assigning fronts individually to each solution, duplicate comparisons can be avoided. This is due to the simple reason that once a solution is compared with all the other solutions in a population and assign to a front, other dominating solutions do not have to be compared with the fronts which are below the respective front.

Due to these efficient strategies, a better performance can be achieved by using this nondominated sorting algorithm.

2.9 Chapter Summary

This chapter concluded the previous work which has been done on the considered approaches for the problem that this system trying to solve. Also several strategies and algorithms for nondominated sorting was discussed in this chapter. In the next chapter, system requirements will be identified using several requirement elicitation techniques and the finding of this chapter will be used to finalize the approach of the system.

Chapter 03

Software Requirement Specification

3.1 Chapter Overview

Since the previous chapter focuses on the past work done in this area, this chapter aims at identifying the stakeholders, requirement elicitation techniques and use case diagram with use case descriptions. This will help identify the functional and non-functional requirements of this project.

3.2 Stakeholder Analysis

3.2.1 Stakeholders of the System

According to the Project Management Institute's definition, "a project stakeholder refers to an individual, group or organization, who may affect, be affected by, or perceive itself to be affected by a decision, activity or outcome of a project". (Project Management Institute, 2013) It is important to identify these stakeholders, in order to make a project a success. These stakeholders could have positive or negative interests towards the system, so it is very important to identify and categorize these stakeholders accordingly.

These stakeholders can be categorized according to their roles towards the system as below.

- Beneficiary
    - Functional beneficiary
    - Financial Beneficiary
- Negative
- Regulatory
- Operational

3.2.2 Onion Model Diagram

Below diagram shows an Onion Model describing the stakeholders who are associated with this system.

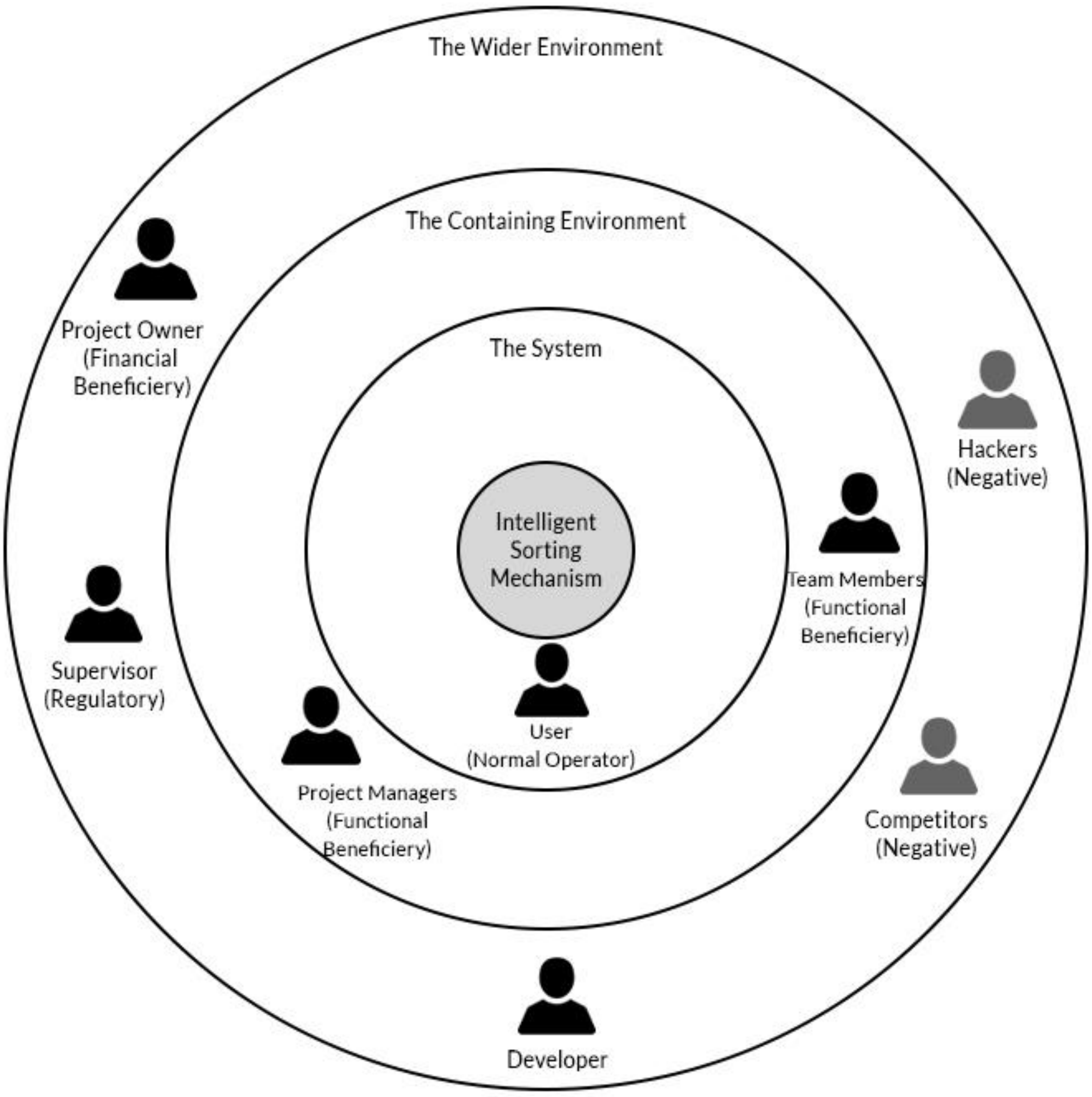

The Wider Environment
The Containing Environment
The System
Intelligent Sorting Mechanism
Project Owner (Financial Beneficiery)
Hackers (Negative)
Team Members (Functional Beneficiery)
Supervisor (Regulatory)
User (Normal Operator)
Project Managers (Functional Beneficiery)
Competitors (Negative)
Developer

Figure 3.1 Onion Model Diagram

3.2.3 Stakeholder Descriptions

Below table shows the stakeholders of the system, their roles and the viewpoints of those stakeholders of the proposed system.

Table 3.1 Stakeholder Descriptions

Stakeholder	Stakeholder Role	Viewpoint
User	Normal Operator	Uses the application to effectively sort the tasks given to himself/herself.
Team Members	Functional Beneficiary	Need to minimize the waiting time of the overall team and increase the effectiveness.
Project Manager	Functional Beneficiary	Wants to minimize the overall time consumption of the project by effectively handling the tasks of a project.
Project Owner	Financial Beneficiary	Needs to minimize the cost of the project by minimizing the overall time consumption of the project.
Supervisor	Regulatory	Set standards to the project to keep it in track.
Competitors	Negative Stakeholders	May build similar systems.
Hackers	Negative Stakeholders	Attack the system in order to access data or fail the system.

3.3 Requirement Elicitation Techniques

Following requirement gathering methods have been considered for this project.

Table 3.2 Requirement Elicitation Techniques

Method	Reason
Domain Expert Interviews	Best way to get the expert knowledge in a specific area and their feedback on the approaching techniques to the problem and the technologies used.
Surveys and Questionnaires	Can easily gather data from a large number of people. Also this will allow to get insights on the user experience aspect of the product as well.
Research Material Analysis / Literature Survey	Best way to gather information on the field in a short amount of time. Also it is easy to gain a good understanding on the previous work which has been done on the topic.
Focus Groups	Can get some direct opinions from the possible future users of the product. Very helpful in terms of getting the requirements tuned in for a better user experience.

3.4 Results of Requirement Elicitation

From the above identified methods, following requirement elicitation techniques were conducted.

3.4.1 Literature Review

Through the literature review several approaches including problem solving algorithms, multiobjective optimization and nondominated sorting techniques were covered. During this process lots of indirect requirements, approaches and limitations were discovered. Most of the data gathered through this method was highly related to the approach of the system, because there were not any similar products developed in this domain. However there were some promising research materials on multiobjective optimization and nondominated sorting approaches. This was extremely helpful in order to finalize the overall approach and the techniques to be used in the prototype.

Even though the literature survey provided lots of information on the approach, it was hard to finalize the requirements by only considering this method. In order to get specific requirements for the prototype, few domain experts had to be consulted.

3.4.2 Domain Expert Interviews

When it came to the domain experts, they were divided in to two groups. One group consisted with people who are related to project management and the other group consisted with people with a strong technical background. Responses received from the people who are related to project management was really helpful in order to draft the first set of requirements to the prototype. Those gathered requirements were enhanced and finalized with the help of the second group of domain experts interviewed. This way of conducting the interviews was really helpful since the domain experts with a strong technical background were able to provide a great deal of help in terms of the overall approach and techniques.

Mainly with these requirement elicitation techniques, both the functional and non-functional requirements were finalized.

3.5 Use Case Diagram

Below figure shows a Use Case Diagram of the proposed system.

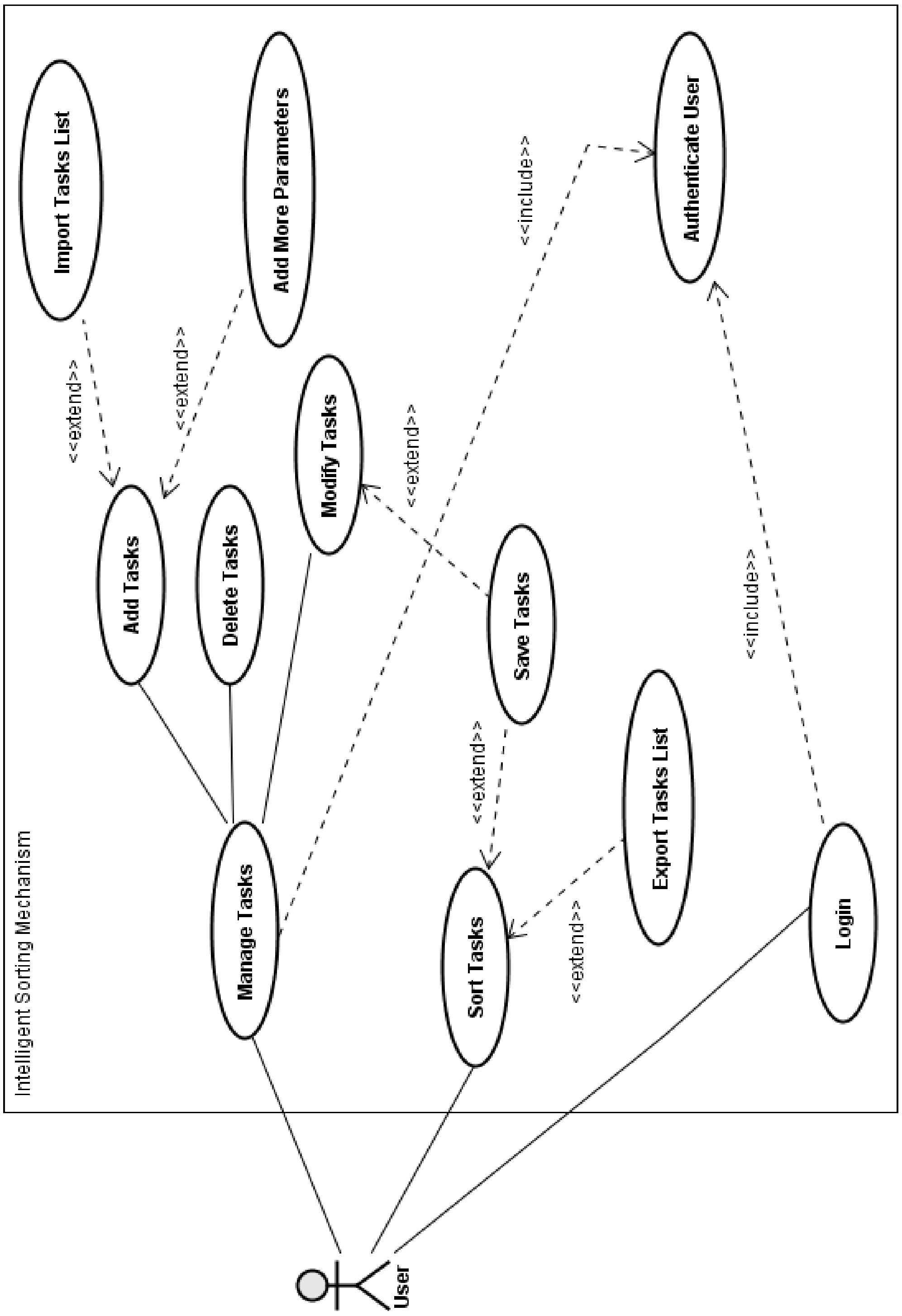


Figure 3.2 Use Case Diagram

3.6 Use Case Descriptions

Following are the Use Case Descriptions for the above Use Case Diagram. More descriptions can be found in appendix.

Table 3.3 Use Case Description 01

Name:	**Add Tasks**
Description:	This inputs list of tasks in order to proceed with the sorting.
Actor:	User
Pre-Condition:	The system must be functional. A logged in user should be there.
Including:	Authenticate User
Extending:	Import Tasks List, Add More Parameters
Main Flow of Events:	1. Captures the tasks and the parameters. 2. Validate the parameters. 3. Adds to a list and save.
Alternative Flows:	1. In step 3, if there are no lists created for the current session, a new list will be created. 2. Adds tasks to that new list.
Exceptions:	E1. In step 2, if the parameters have illegal values it produces an error message and asks to re-enter the task.
Post Condition:	A list of tasks is saved.

Table 3.4 Use Case Description 02

Name:	**Sort Tasks**
Description:	This sorts the input list of tasks.
Actor:	User
Pre-Condition:	The system must be functional. A logged in user should be there. A valid set of tasks should be there.
Including:	None
Extending:	Export Tasks List
Main Flow of Events:	1. Validate the task list.

	2. Sort the tasks using the first parameter and then the second. 3. Assign it to fronts using multiobjective optimization. 4. Display sorted list.
Alternative Flows:	None
Exceptions:	1. In step 1 if the list does not have enough tasks to sort, display an error message. Terminates the use case.
Post Condition:	A list of sorted tasks is saved.

Table 3.5 Use Case Description 03

Name:	**Import Tasks List**
Description:	This imports list of tasks in order to proceed with the sorting.
Actor:	User
Pre-Condition:	The system must be functional. A logged in user should be there.
Including:	None
Extending:	None
Main Flow of Events:	1. Validate the imported file. 2. Extract data from the file. 3. Identify the parameters of tasks.
Alternative Flows:	1.
Exceptions:	E1. In step 1, if the file extension is not supported, display an error message. Terminates the use case. E2. In step 2, if the data in the file is not readable, display an error message. Terminates the use case. E3. In step 3, if the parameters are not marked correctly, display an error. Terminates the use case.
Post Condition:	A list of imported tasks is saved.

3.7 Requirements

3.7.1 Functional Requirements

Table 3.6 Functional Requirements

No.	Requirement	Use Case	Priority
01	Allow the user to add tasks to the system.	Add Tasks	Essential
02	Allow the user to modify the added tasks.	Modify Tasks	Essential
03	Sort the tasks according to the parameters given.	Sort Tasks	Essential
04	Display the sorted list to the user.	Display Task List	Essential
05	Allow the user to import tasks directly from their project management tool using Microsoft Excel format.	Import Tasks	Essential
06	User should be able to export the sorted tasks using Microsoft Excel format so it can be directly imported back to their tools.	Export Tasks	Essential
07	User should be able to write custom equations to adjust the objective weight.	Provide Custom Equations	Luxury

3.7.2 Non-functional Requirements

- NFR1 – **Accuracy**: When it comes to sorting, system should always provide an accurate sorted list.
- NFR2 – **Performance**: Sorting process should not take a long time to complete even when the number of tasks get increased by a reasonable amount.
- NFR3 – **Usability**: User Interfaces should be clear and self-explanatory.
- NFR4 – **Extensibility**: More parameters should be able to add to the system later so the sorting process gets even more accurate.
- NFR5 – **Reliability**: Application should perform without getting stuck when the number of tasks get increased.

3.8 Scope Refinements

The scope was refined to only 2 parameters which are priority and deadline. The final prototype will be able to sort the given tasks effectively using these 2 parameters as the objectives.

3.9 Chapter Summary

This chapter focuses on requirement elicitation and getting the both functional and non-functional requirements accurately in order to make a good design for the proposed system, which will be discussed in the next chapter. Since this is a research based project, some requirements might change over the project timeline but the main functions of the project will remain as it is.

Chapter 04

Design Specification

4.1 Chapter Overview

This chapter discusses the design of the prototype of this project. This is based on the important aspects gathered in the previous chapter like functional and non-functional requirements. In this chapter design goals of the prototype, high level architecture, low level architecture and its detailed UML representation will be discussed.

4.2 Architectural Goals and Constraints

Accuracy – Sorting accuracy should always be high to achieve a time advantage in a project this application promises.

Solution: Proper design techniques should be used to avoid the design flows which compromises the accuracy.

Scalability – Application should be able to scale when it comes to a huge projects with a lot of number of tasks to process and store.

Solution: Use of the dynamic data structures to store and process the data fed to the system without compromising the performance of the system.

Extensibility – This application should be able provide a solid base for adding more parameters to the sorting mechanism in future to make this process more accurate.

Solution: A well-structured design should be there to accommodate future extensions to the project.

Performance – Since there could be a lot of tasks to process in a project, the response time should always be kept to a minimum.

4.3 Design Methodology

Object oriented methodology will be used in this project since it is supported by the major software development tools and languages. Apart from that it is known to be time saving when it comes to reusability and the maintenance of the code. Considering the limited time schedule and the single person development approach, object oriented design and development methodology will be selected for this project.

Table 4.1 Software Design Methodology

Methodology	Pros	Cons
Object Oriented	• Can be easily maintained. • High reusability. • Improved flexibility and reliability. • Modeling is based on objects rather than on data and processes. • UML design approaches can be used	• Can be quite complex sometimes. • Hard to determine all required classes and objects.
Process Oriented	• Process is easy to understand. • Comparatively low upfront planning needed. • Easy to keep track of the program flow.	• Does not support reusability. • Hard to get adapted to the requirement changes.

4.4 System Architecture

The proposed system has a 3-tier architecture as shown in the below diagram.

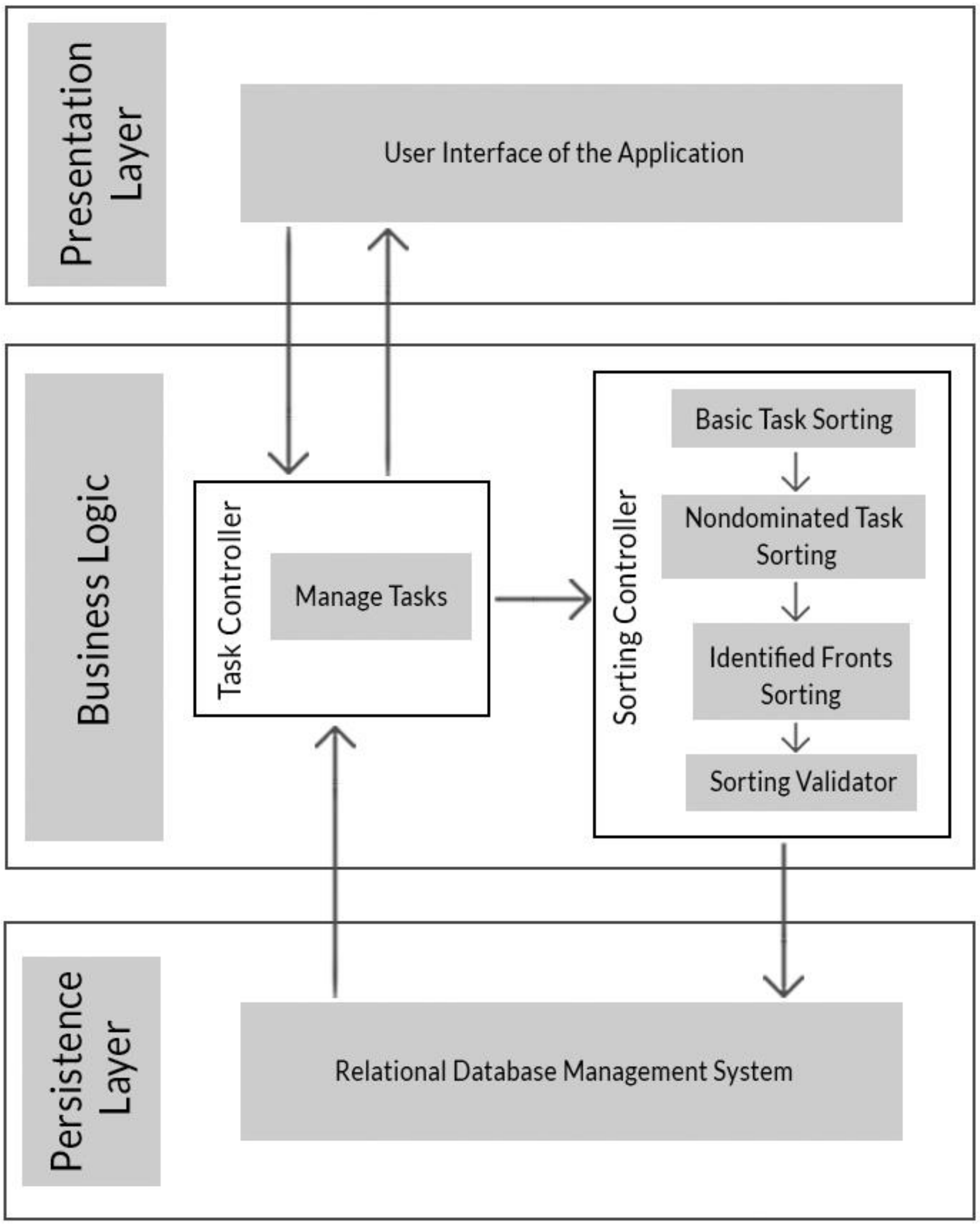


Figure 4.1 System Architecture Diagram

4.5 System Architecture Description

Below section contains a detailed explanation of the layers shown in the 3-tier architecture diagram which was in the previous section.

4.5.1 Persistence Layer

All the task lists which have been added to the system can be stored in the database included in this layer. This data storage will contain all the added tasks to the system in list templates and the users who register with the system.

4.5.2 Business Logic Layer

This layer basically contains all the major logics associated with the current system. The most important component here is the Sorting Controller, which handles the task sorting process using several mechanisms.

Initially these tasks go through a basic sorting phase to get them ordered in an ascending order according to the parameters considered. Then the sorted tasks will go through the nondominated sorting mechanism, which at the end assigns the tasks to different fronts using the given parameters and objective weights. After all the tasks in the current list are assigned to respective fronts, sorting validating process happens in each front finalizing the sorting order of the list.

At the end, the finalized sorted list will be handed down to the data storage, which was discussed in the Persistence Layer.

4.5.3 Presentation Layer

This layer basically contains the user interfaces for the application.

4.6 Low Level Design Diagrams

4.6.1 Class Diagram

Below figure shows a class diagram of the proposed system.

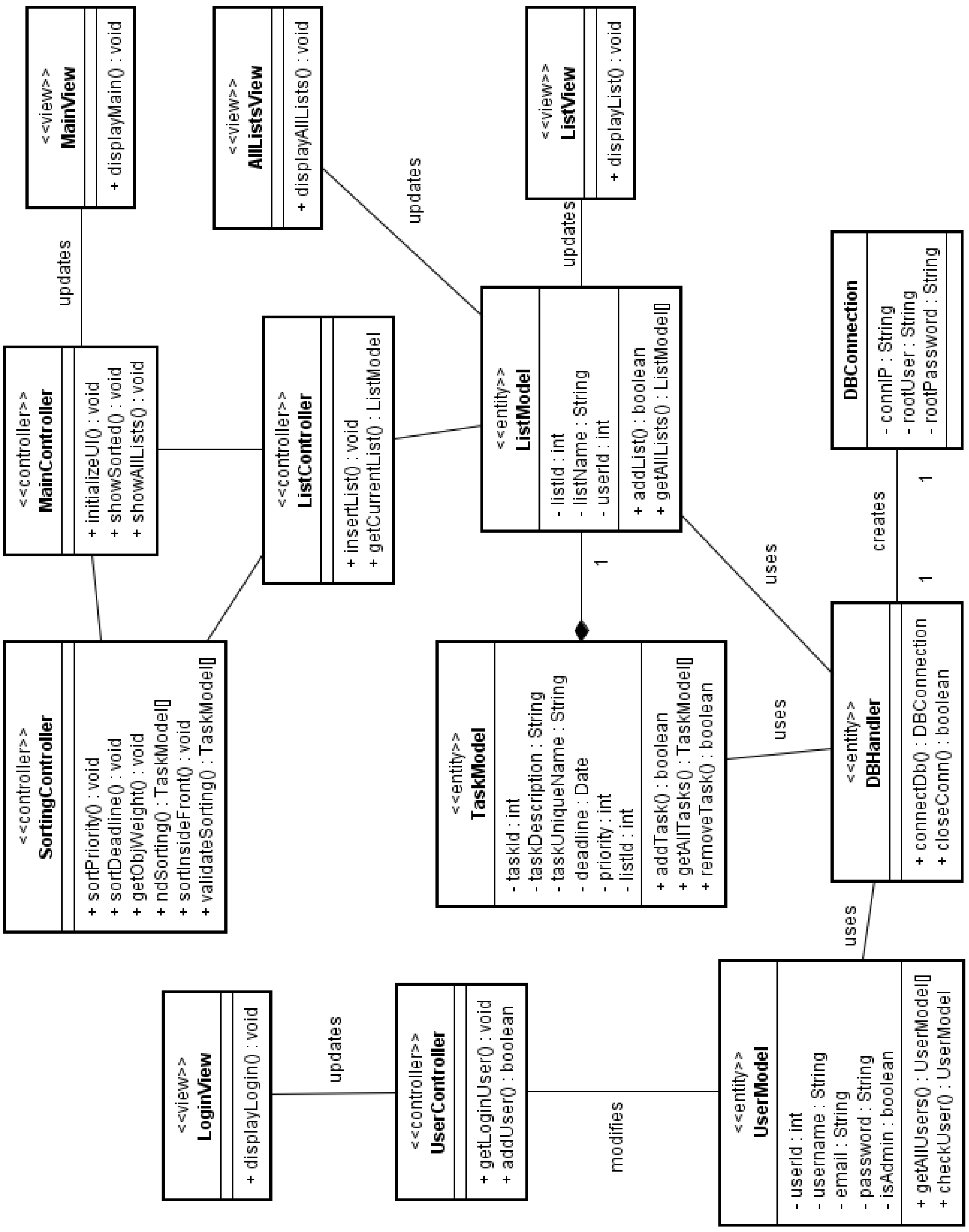


Figure 4.2 Class Diagram

4.6.2 Sequence Diagrams

Sequence diagram for the main flow, which is sorting a list of tasks is shown in the below figure.

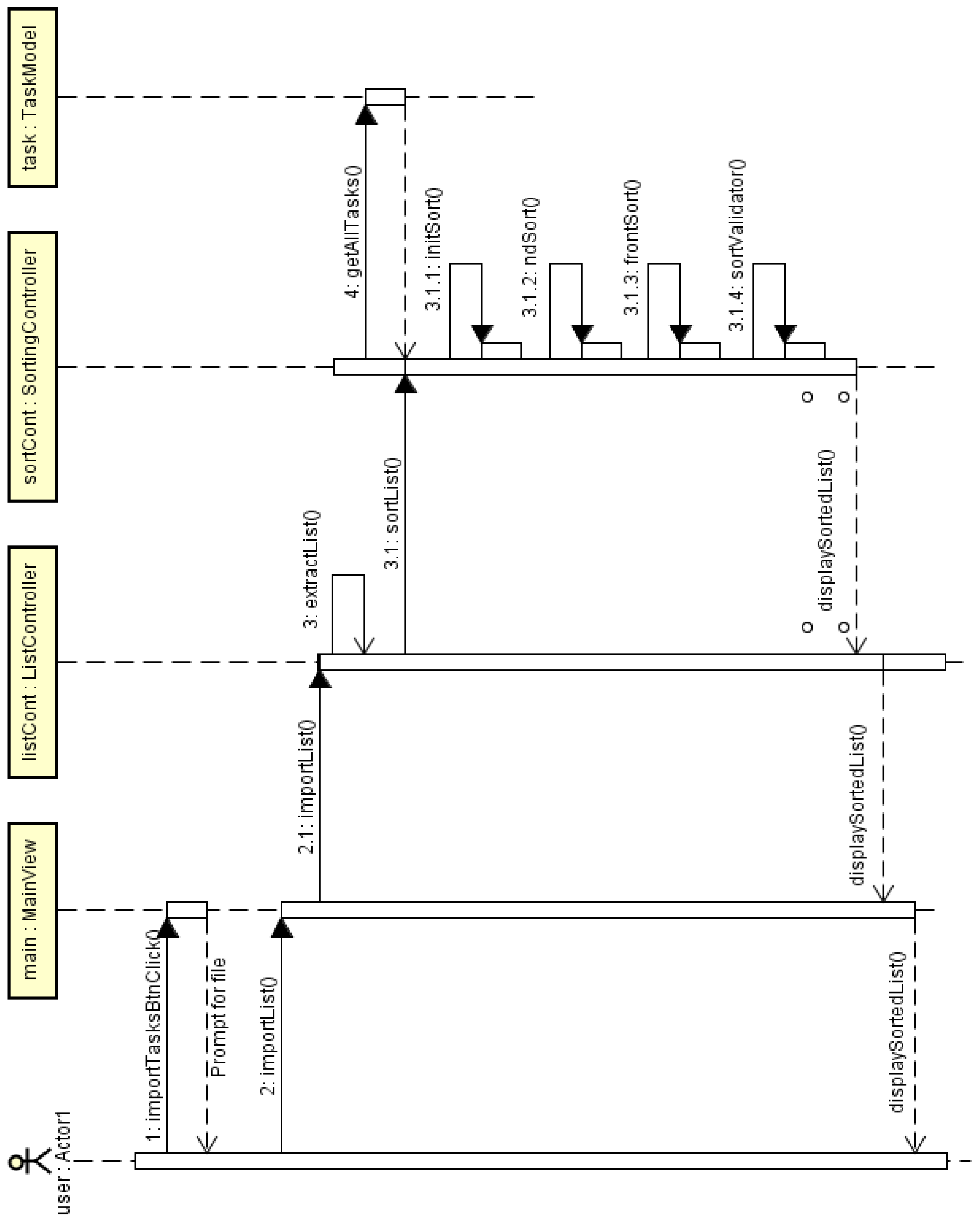


Figure 4.3 Sequence Diagram

4.7 Chapter Summary

This chapter covered the design aspects, goals, system architecture and detailed UML diagrams of the system. Since the designing of the system is completed, the next chapter will focus on the implementation part of the system according to the above design specifications.

Chapter 05

Implementation

5.1 Chapter Overview

This chapter explains the implementation process of the system according to the design specifications provided in the previous chapter. This will include the technology selection process, problems occurred during the implementation and the algorithms used in the implementation.

5.2 Application Overview

The high level design of the system is presented in the below diagram. There are a few components to the system and the main component is the nondominated sorting component. How this component works is discussed in the section below.

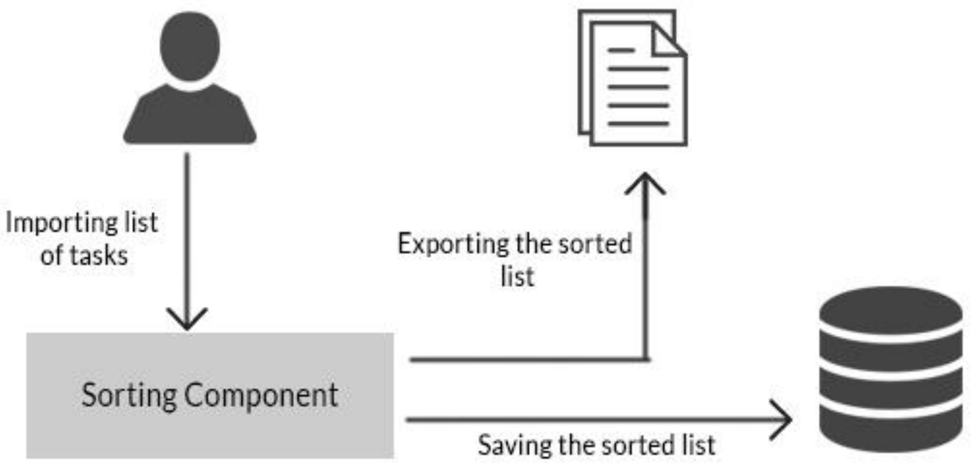


Figure 5.1 High Level View

5.2.1 Nondominated Sorting Component

In this component, the initially sorted list will be considered as the population and the items in the population will be added to different fronts according to the parameters given.

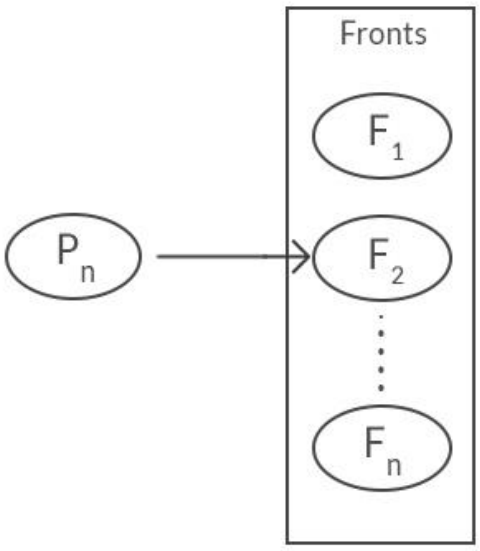
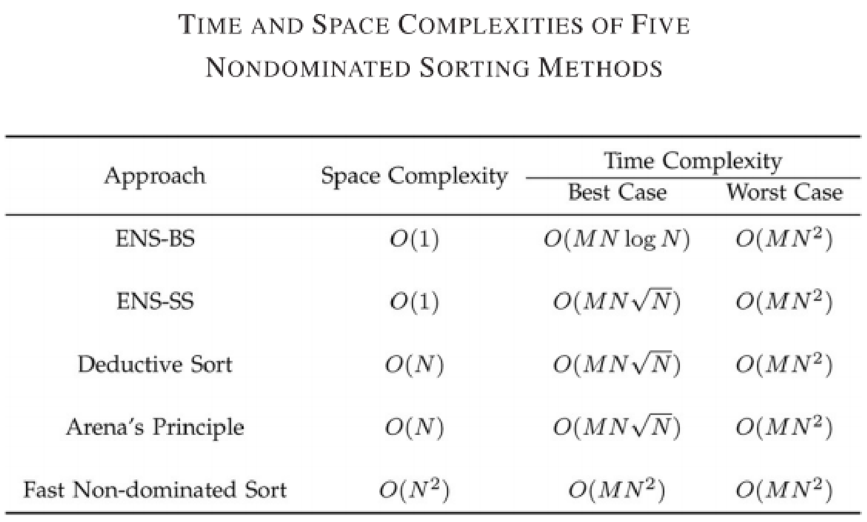


Figure 5.2 Front Allocation

According to the above diagram, if an item P_n in the population assigned to the F_2 front, there exist at least one solution in each front above F_2 which dominates the task P_n. When it comes to the fronts below F_2, there does not exist any solution which dominates the task P_n.

5.2.2 Nondominated Sorting Approach Selected

There are two main ways of assigning items to the respective fronts. In this approach the binary search strategy was selected over the sequential search strategy. Both of these approaches have a same space complexity of $O(1)$ and a worst case time complexity of $O(MN^2)$. (In this case N being the number of solutions and M being the number of objectives.) However when it comes to the best case time complexity, nondominated sorting with binary search strategy comes to an $O(MN \log N)$ where the sequential search strategy only achieves an $O(MN\sqrt{N})$.


TIME AND SPACE COMPLEXITIES OF FIVE NONDOMINATED SORTING METHODS

Approach	Space Complexity	Time Complexity	
		Best Case	Worst Case
ENS-BS	$O(1)$	$O(MN\log N)$	$O(MN^2)$
ENS-SS	$O(1)$	$O(MN\sqrt{N})$	$O(MN^2)$
Deductive Sort	$O(N)$	$O(MN\sqrt{N})$	$O(MN^2)$
Arena's Principle	$O(N)$	$O(MN\sqrt{N})$	$O(MN^2)$
Fast Non-dominated Sort	$O(N^2)$	$O(MN^2)$	$O(MN^2)$

Figure 5.3 Time and Space Complexity Comparison

5.3 Technology Selection Rationale

This section describes the reasons for selecting below technologies for developing the prototype by comparing those with one another.

5.4 Selection of Technologies

5.4.1 Programming Language

Both Java and C# languages were considered in this process. Since both the languages are not that different from each other some other aspects had to be taken in to consideration. When it comes to java it can be used in many other platforms but C# is deeply integrated with Windows platform in a way. However when it comes to C#, the support given from the .NET platform cannot be ignored in this case. With the extensive support for its core features it is easier to use C# in this project rather than Java. Not only the language aspects, the development environment aspects were also considered in this process. Since most of the Java IDEs are open source it is easier to find a reasonable IDE to work with the language. However when it comes to C# language, Microsoft Visual Studio cannot be discarded since it has so many features to make the developing life easier. Apart from that C# supports WPF (Windows Presentation Foundation), which can be used to build really great dynamic user interfaces with data binding. Due to these reasons, C# was selected as the development language of the prototype.

5.4.2 Data Storage

MySQL was selected as the relational database management system since it is a reliable and highly scalable database management system. Also with the interfaces provided it is extremely easy to work with. Besides it has an excellent data structure and it can be integrated with many platforms, making it easier to add more platforms to the current system.

5.5 Libraries Used

5.5.1 Microsoft Office Interop Services

Microsoft Office Interop Services were chosen to read and write excel files in the prototype. Even though there are faster approaches to read and write excel files, this one is easiest comparing to other methodologies.

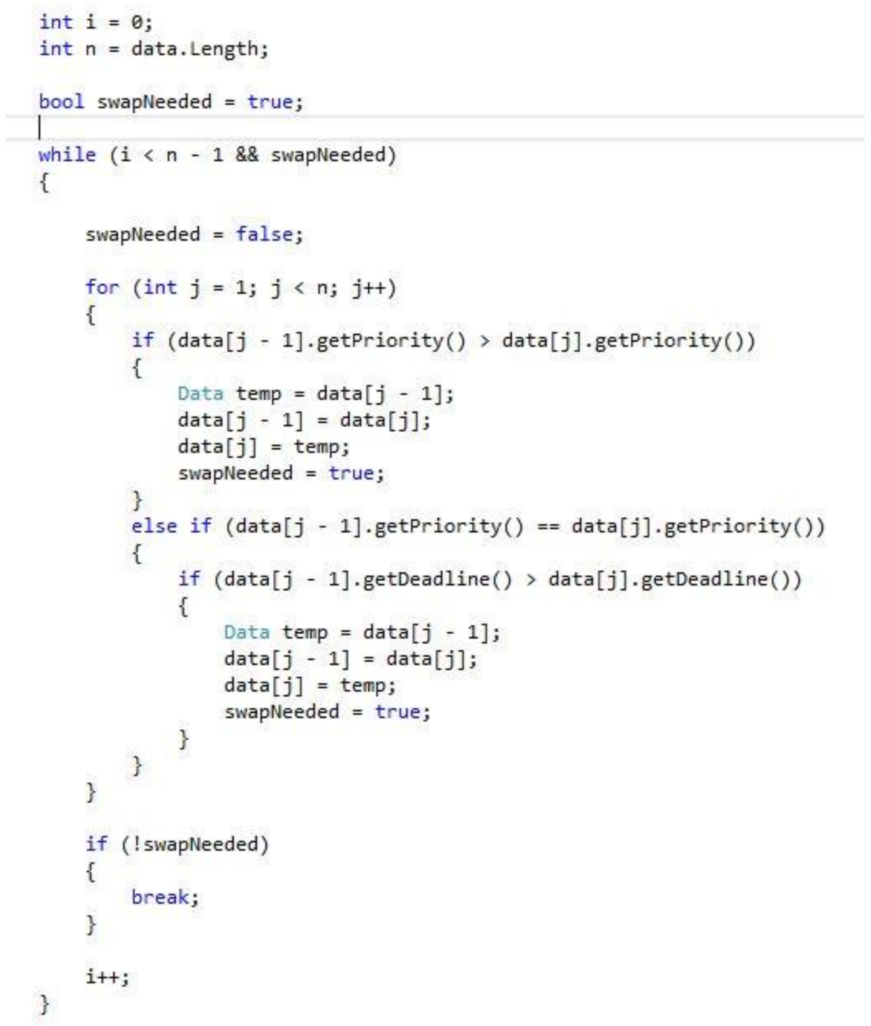
One of the main ideas considered during the selection was reading an Excel file using the zip method. Since all the Excel Workbooks can be viewed as a zip file, it can be opened using a zip file opener and the values can be read using the XML files inside the workbook. This method is very fast comparing to the Microsoft Interop Services but this is only effective when working with huge number of data.

Since this does not contain that much data to process, performance of the Microsoft Interop Service was acceptable.

5.6 Implementation of the Prototype

In the sorting component, data is being processed in four different methods. In this section the key implementation aspects of the sorting component will be discussed.

Before the actual nondominated sorting process begins, the tasks imported will be sorted using a basic sorting algorithm, according to the objectives.

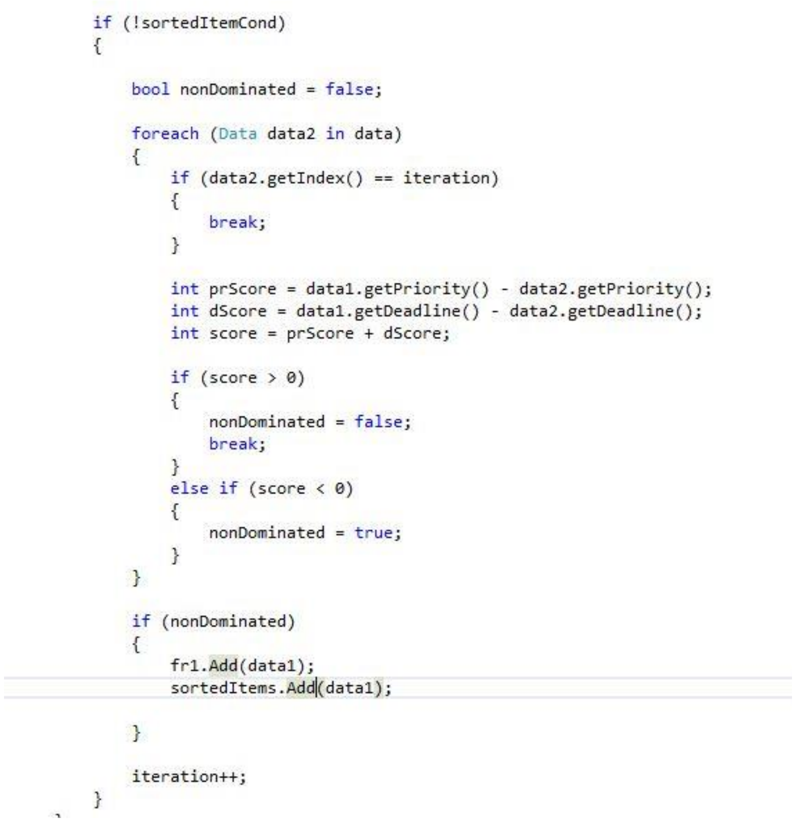


```
int i = 0;
int n = data.Length;

bool swapNeeded = true;

while (i < n - 1 && swapNeeded)
{

    swapNeeded = false;

    for (int j = 1; j < n; j++)
    {
        if (data[j - 1].getPriority() > data[j].getPriority())
        {
            Data temp = data[j - 1];
            data[j - 1] = data[j];
            data[j] = temp;
            swapNeeded = true;
        }
        else if (data[j - 1].getPriority() == data[j].getPriority())
        {
            if (data[j - 1].getDeadline() > data[j].getDeadline())
            {
                Data temp = data[j - 1];
                data[j - 1] = data[j];
                data[j] = temp;
                swapNeeded = true;
            }
        }
    }

    if (!swapNeeded)
    {
        break;
    }

    i++;
}
```

Figure 5.4 Implementation of the Basic Sorting function

This will produce a list of data which has been sorted in ascending order according to the parameters considered. Even though this is a very simple task, this will help avoid the duplicate comparisons in the nondominated sorting process and make the algorithm more efficient.

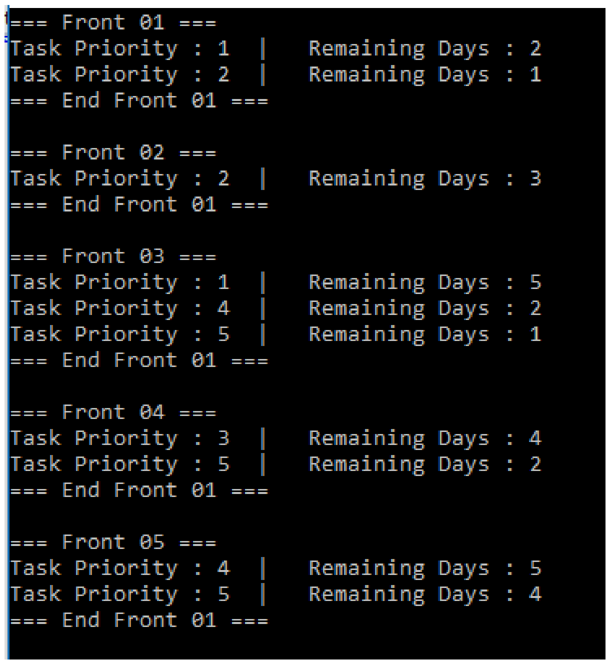
When it comes to the nondominated sorting process, it takes the initially sorted list and do the necessary comparisons with the other tasks in the same population space. When comparing two tasks it takes the objective weight into consideration and decide which task is the most suitable nondominated solution and assigns that to a front. Below code snippet shows the most basic way of calculating the objective weight by simply deducting the values of both the minimizing values.



```
if (!sortedItemCond)
{
    bool nonDominated = false;

    foreach (Data data2 in data)
    {
        if (data2.getIndex() == iteration)
        {
            break;
        }

        int prScore = data1.getPriority() - data2.getPriority();
        int dScore = data1.getDeadline() - data2.getDeadline();
        int score = prScore + dScore;

        if (score > 0)
        {
            nonDominated = false;
            break;
        }
        else if (score < 0)
        {
            nonDominated = true;
        }
    }

    if (nonDominated)
    {
        fr1.Add(data1);
        sortedItems.Add(data1);

    }

    iteration++;
}
```

Figure 5.5 Basic Objective Function Implementation

At the end of the nondominated sorting process, tasks will be assigned to a respective front as shown in the figure below.



```
=== Front 01 ===
Task Priority : 1  |   Remaining Days : 2
Task Priority : 2  |   Remaining Days : 1
=== End Front 01 ===

=== Front 02 ===
Task Priority : 2  |   Remaining Days : 3
=== End Front 01 ===

=== Front 03 ===
Task Priority : 1  |   Remaining Days : 5
Task Priority : 4  |   Remaining Days : 2
Task Priority : 5  |   Remaining Days : 1
=== End Front 01 ===

=== Front 04 ===
Task Priority : 3  |   Remaining Days : 4
Task Priority : 5  |   Remaining Days : 2
=== End Front 01 ===

=== Front 05 ===
Task Priority : 4  |   Remaining Days : 5
Task Priority : 5  |   Remaining Days : 4
=== End Front 01 ===
```

Figure 5.6 Front Allocation Results

5.7 Complications Occurred During Implementation

Some of the complications occurred during the implementation phase and the given solutions are described in the below table.

Table 5.1 Complications Occurred During Implementation

Complication	Solution
When importing Excel sheets some of the column names conflict with the objective parameter columns.	An option was given to the user to mark the correct objective parameter columns.
Some of the priority standards were not identified by the system.	A set of standard priority levels were declared and priority mapping procedure was introduced to the user.
When importing tasks, if there are multiple error values in the data, user had to re-enter the whole data set.	A new list was created to hold the invalid data rows and prompted separately for the user to correct them.
Hard to check the accuracy of the data produced by the sorting functions.	A few data sets were chosen and nondominated sorting was performed manually to check the accuracy.
Sometimes the output data was not accurate enough.	More validations were added in order to make the sorting process more accurate.

5.8 Chapter Summary

This chapter concluded the implementation of the proposed system justifying the technologies, tools and algorithms used in the development process. At this point the prototype is completed successfully and the testing of the prototype will be discussed in the next chapter.

Chapter 06

Testing and Evaluation

6.1 Chapter Overview

This chapter covers the testing of the implemented system using various techniques and critically evaluating the results according to the selected evaluation methodologies. Also the strengths and weaknesses of the prototype and future enhancements will be covered in this chapter too.

6.2 Objectives of Testing

There are few objectives to be achieved in this phase of the project development process. The testing phase is necessary to ensure that the functionalities of the system are working correctly without any bugs or performance issues. The following goals have been identified to ensure that the prototype is thoroughly tested.

- Completion of the identified functional requirements.
- Completion of the identified non-functional requirements.
- Ensure that the system performs well without any bugs.

6.3 Testing Methodology

The prototype was tested using several methodologies. During the implementation process unit testing of the prototype was conducted. Then after combining the sorting components integration testing was conducted. At the end of the development phase, functional, accuracy and performance testing were conducted.

Table 6.1 Testing Methodology

Test Type	**Description**
Unit Testing	To test the individual components as the development process moves forward. This will help identify bugs in individual components making it easier to fix.
Integration Testing	This testing can be carried out after integrating the individual components. This testing will help

	identify the issues occur when integrating the components.
Functional Testing	The prototype will be tested function wise, in order to identify any issues related to a specific functionality.
Accuracy Testing	This testing will be conducted to see the accuracy of the data produced by the system.
Performance Testing	System will be tested for its performance using various number of tasks.
System Testing	Prototype will be tested for the functional requirements provided in Chapter 03, to ensure that all the requirements are fulfilled in the system.

6.4 Testing Results

6.4.1 Unit Testing and Integration Testing

Unit testing was conducted as the prototype was being developed. Each of the components which was completed had been tested using both black box and white box testing. It was easy to identify bugs using this procedure and all the bugs came up at the end of the testing procedure were fixed.

In order to find the incompatibilities among the system components, integration testing was conducted aggregating all the related components. When conducting the integration testing, bottom up approach was selected, because it was easier to test the low level components first as the development process moved forward. All the bugs occurred during these two phases of testing were fixed and moved on to the next set of testing procedures.

6.4.2 Functional Testing

Functional testing was conducted as the development of the prototype was completed. This was done in a black box testing form. During this process each of the functionalities were tested using different test cases and the summarized test results are available in the below table.

Table 6.2 Functional Testing

ID	Test Case	Expected Results	Actual Results	Status
01.	Input blank or invalid values for the parameters.	Prompt an error message.	Prompted an error message.	Passed.
02.	Input a different file type other than Microsoft Excel.	Import button should be grayed out and unable to select.	Import button was disabled.	Passed.
03.	Import an Excel file with tasks but without a parameter column.	Error message should be displayed and should be prompted for the missing values.	An error message was given and asked to enter the missing values.	Passed.
04.	Sort a list with only one items.	Error message should be displayed and should be prompted to input more tasks.	An error message was displayed and a popup to enter more tasks were appeared.	Passed.
05.	View sorted lists without logging in to the system.	The sorted list area should be disabled restricting non users to enter.	The sorted list area was disabled.	Passed.
06.	Export a list without sorting it.	The list should be exported.	The list was successfully exported.	Passed.

6.4.3 System Testing

The prototype was tested for the functional requirements provided in Chapter 03 – System Requirement Specification. Below table contains the requirements and the testing results of this process.

Table 6.3 System Testing

ID	Requirement	Status
01.	Allow the user to add tasks to the system. This may include single or multiple tasks.	Passed
02.	Allow the user to modify added tasks.	Passed
03.	Sort the added tasks according to the parameters given by the user.	Passed
04.	User should be able to see the final sorted list of tasks.	Passed
05.	Allow the user to import tasks directly using Microsoft Excel format.	Passed
06.	Allow the user to export the sorted task list using Microsoft Excel format.	Passed
07.	Allow the user to write custom equations to define objective weights.	Failed

6.4.4 Accuracy Testing

Accuracy of the data produced by the prototype was tested against sample sorted lists. Since the accuracy cannot be checked completely using those data, expert opinions were taken. Those results will be discussed in the evaluation section of this chapter under the Quantitative Evaluation sub heading.

6.4.5 Performance Testing

In order to check the performance of the system, the nondominated sorting algorithms used were compared with other similar algorithms. Apart from that, a complete performance test was conducted by feeding increasing number of tasks and calculated the processing time. This will also be discussed under the Quantitative Evaluation section.

6.5 Project Evaluation

In order to assess the project conducted, it is very important to evaluate the project according to the below criteria. In here not only the project implementation also the project process is evaluated based on the objectives achieved, completion of the project aim and finally the completion of the requirements.

6.6 Evaluation Methodology

This can be broken down to two main components.

- Qualitative Evaluation

  This will be based on the expert feedbacks and the user ratings for the product. This will be discussed in detail in this section.

- Quantitative Evaluation

  This will be based on the test results and the performance of the prototype tested. This will further evaluated by comparing with the existing algorithms performance and accuracy.

In addition to above evaluation methods, a critical evaluation by the author will also be discussed in this chapter.

6.7 Qualitative Evaluation

In order to get the project evaluated by the external evaluators, a questionnaire was prepared and an interview was conducted based on the questionnaire.

6.7.1 Selection Criteria for Evaluators

Table 6.4 Selection Criteria for Evaluators

Evaluator	**Reason**
Project Managers	Since they have the most experience on handling large projects considering all the factors from project owner's end to developer's end.
Senior Software Engineers	Since they are experts on the technical domain, it is necessary to have them evaluate the system based on the technical aspects, like system architecture and implementation.

6.7.2 Summary of the Evaluation Questionnaire

Table 6.5 Summary of the Evaluation Questionnaire

Evaluation Criteria	No.	Question
Concept and the Approach	1.	Does this prototype provide a better solution to the described problem?
	2.	Do you think that this sorting mechanism can provide efficient data comparing to the sorting mechanisms in existing project management tools?
	3.	How was the accuracy of the sorting mechanism comparing to the way that you would manually sort?
Overall User Experience	6.	How was the overall performance of the current prototype?
	7.	What was the overall user experience gained by working with prototype?

	8.	How satisfied are you with the UI experience provided in the prototype?
Limitations and Future Enhancements	9.	What are the limitations of the prototype demonstrated? (If any)
	10.	What are the features that can be added to improve this solution in future?

6.7.3 Summary of Feedbacks

6.7.3.1 The Concept and the Approach

According to the responses given by the above evaluators, the concept and the chosen approach was really impressive since all the conventional sorting approaches are based on simple mechanisms which cannot consider multiple objectives at a time. Project Managers who evaluated the project mentioned that this approach can actually produce more accurate data compared to the inbuilt sorting mechanisms in the project management tools that they are currently using. Even though there are only two objectives considered in the prototype, still the sorting order was more efficient than the normal sorting approaches according to the Software Developers who tried the prototype.

However when it came to the large data sets, these evaluators found it a bit slow due to the computational complexity of the algorithms used, but since the tasks were sorted in a more effective manner they still accepted the current approach. Some of the software experts participated in this evaluation process stated that this can be more optimized when adding more objectives in future so this approach remains as it is.

6.7.3.2 Prototype

All the evaluators have agreed that the prototype has done its justice to the project by providing all the necessary features to efficiently sort a given number of tasks. In addition, they have mentioned that the prototype performance is really impressive, since most of the multiobjective optimization algorithms are computationally expensive. When it comes to the usability of the prototype, the evaluators liked the simple, easy to use design and the extra features like Import Excel, which increased the overall experience of the user.

6.7.3.3 Limitations of the Prototype

Even though the evaluators were impressed with the overall project, there were a few limitations that they have pointed out.

- Sorting mechanism only consider two objectives at the moment.

  Even though the sorting mechanism is more efficient than the conventional sorting mechanisms, all the evaluators have agreed that there should be more parameters given to the system as objectives in order to increase the accuracy. Some of the evaluators also said that by adding more objectives, the amount of conditions that should be checked at the end of the sorting process will go down or might change, making it more flexible to any sort of scenario.

- Does not let the user to handle the objective weight.

  At the moment the objective weight is not adjustable to the user and some evaluators mentioned that by letting the user handle the objective weight, it makes the sorting mechanism more flexible.

6.8 Quantitative Evaluation

This step is crucial in order to identify the suitability of the proposed system. Quantitative evaluation is based on the statistical data gathered in the testing phase of the project. Some of these results might change according to the environment that it is being tested.

6.8.1 Accuracy of the System

This is a very important factor of the proposed system, but hard to analyze only using the algorithmic data. In order to test the accuracy of the system, sample sorted lists were compared against the sorted lists produced by the system. In addition to that, expert feedbacks were taken in order to decide whether the task arrangement was acceptable. According to the evaluators who oversaw this process agreed that the accuracy of the sorted lists were quite high considering the fact that these data were based on two parameters.

When compared to the sample sorted lists the accuracy of the system produced lists were really impressive. Even though some task orders were not as expected, these results were also acceptable. In addition, these accuracy ratings did not go down when the number of tasks were increased.

6.8.2 Performance of the System

Even though the nondominated sorting algorithms used in the system are complex and computationally expensive than the basic sorting algorithms, the performance testing results were promising. The performance testing was conducted in a computer with an Intel Core i3 3.5 GHz processor and an 8 GB RAM. Since the overall performance of the sorting process depends on the number of tasks given to the system, with a large set of tasks the overall performance of the system went down.

The overall performance drop is noticeable only when the system handles large number of tasks. Compared to other algorithms, performance of this prototype is really impressive since almost all the nondominated algorithms have performance issues due to duplicate comparisons. However in the ENS algorithm those duplicate comparisons are avoided since it effectively assigns the nondominated members of the population to different fronts in each iteration.

6.9 Evaluation by the Author

Sorting plays a major role in algorithms. There are a few basic sorting algorithms which optimizes the use of many other algorithms. However when it comes to sorting a list of elements, in most cases, only one parameter is considered and the set of elements get sorted in a specific order. This is acceptable in most scenarios but when it comes to elements which needs to be sorted considering multiple objectives this is not an effective way. Especially when it comes to project task management, most of the times it is being done by a project manager or someone who has the knowledge about the project and the available resources. Besides, the project management tools currently exist only provide basic sorting mechanisms to arrange the tasks in a sprint or the project backlog. This process of manually sorting the tasks is time consuming and it might not be efficient always since there could be large number of tasks with various objectives to consider. The motivation behind this project was to increase the efficiency of the sorting process and also to simplify this process at the same time.

With this in mind, the research aspects of the project was started and the first task was to conduct a proper literature survey to lay a proper foundation to the project. In this stage, lots of different methods have been considered, but since the solution should accommodate a simple yet an efficient sorting mechanism, multiobjective optimization was selected as the approach to this problem. Multiobjective Optimization is now being used in several engineering fields to design highly complex systems which consider multiple criteria when selecting a solution to optimize the overall solution. In addition to the literature survey conducted, the author consulted several domain experts to gather more knowledge on the approach and also to get some valuable insights on mapping the solution with the found approach.

In order to come up with the requirements needed for the prototype, various methods have been considered. Apart from the continuous literature survey conducted, the author had to consult experts on both the software development side as well as the project management side to get an overall idea about the requirements needed for this application. Through the responses collected from those experts and the results gained through the literature survey, the requirements were finalized.

The requirements were formalized in the design phase of the project by making the blueprints to the actual system using UML diagrams. In this phase, requirements and other necessary features were added to the formalized design making the system more user friendly and reliable.

In the implementation phase, all the core requirements, non-functional requirements and other necessary features were implemented. However the author was faced with various challenges when working with multiobjective optimization algorithms due to lack of research materials and tutorials. In addition to that, most of the algorithms were computationally expensive and therefore it could have affected the performance of the overall application. These challenges were successfully overcame by the author with the help and guidance of the project supervisor and the domain experts.

As the final phase of the project development, the prototype was thoroughly tested and the remaining bugs were fixed. At the end of the testing phase the prototype was given to the selected external evaluators to get their feedback. According to the testing of the prototype and the expert feedbacks given, all the functional and non-functional requirements have been completed.

Even though the prototype contains all the requirements planned in the earlier phases of the project, there are some limitations in the prototype since the project had to be done under a strict time frame. However with the expert feedbacks received these limitations can be lifted and these aspects were noted down in order to enhance the current product in future. With these enhancements in mind, the author believes that this sorting mechanism can be improved by adding more objective functions and it will give a much better performance in future.

6.10 Completion of Requirements

Below table contains the functional requirements found and presented in Chapter 03 – Software Requirements Specification, and the current status of those requirements.

Table 6.6 Completion of Functional Requirements

No.	Functional Requirement	Priority	Status
01	Allow the user to add tasks to the system.	Essential	Completed
02	Allow the user to modify the added tasks.	Essential	Completed
03	Sort the tasks according to the parameters given.	Essential	Completed
04	Display the sorted list to the user.	Essential	Completed
05	Allow the user to import tasks directly from their project management tool using Microsoft Excel format.	Essential	Completed
06	User should be able to export the sorted tasks using Microsoft Excel format so it can be directly imported back to their tools.	Essential	Completed
07	User should be able to write custom equations to adjust the objective weight of the sorting mechanism.	Luxury	Future Enhancement

Table 6.7 Completion of non-functional Requirements

No.	Non-functional Requirement	Status
01	When it comes to sorting, system should always provide an accurate sorted list.	Completed
02	Sorting process should not take a long time to complete even when the number of tasks get increased by a reasonable amount.	Completed
03	User Interfaces should be clear and self-explanatory.	Completed

04	More parameters should be able to add to the system later so the sorting process gets even more accurate.	Future Enhancement
05	Application should perform without getting stuck when the number of tasks get increased.	Completed

6.11 Chapter Summary

In this chapter a critical evaluation for the project was discussed. This was done in two main areas and at the end of the chapter, a critical evaluation done by the project author was included. In this chapter, the whole project and its process was evaluated. In the next chapter, the whole project will be concluded.

Chapter 07

Conclusion

7.1 Chapter Overview

This final chapter will summarize all the accomplishments throughout the project, challenges faced and solutions found to overcome those challenges during this project. It also includes the future enhancements of this project, learning outcomes and the overall help provided by the other modules in this degree program.

7.2 Achievement of Project Objectives

Table 7.1 Achievement of Objectives

Objective	Description	Status	Evidence
Obj1	Establish the scope of the project.	Completed	Chapter 01 - Introduction
Obj2	Conduct a background study in order to find existing solutions and mechanisms.	Completed	Chapter 02 – Literature Review
Obj3	Conduct a thorough literature survey to evaluate research material on multiobjective optimization.	Completed	Chapter 02 – Literature Review
Obj4	Gather the requirements and stakeholder aspects using the literature survey conducted and the other requirement elicitation techniques used.	Completed	Chapter 03 – Software Requirement Specification
Obj5	Finalize the features of the prototype considering all the factors.	Completed	Chapter 03 – Software Requirement Specification

Obj6	Design the system using identified requirements and stakeholder needs.	Completed	Chapter 04 – Design Specification
Obj7	Implement a prototype according to the design specifications.	Completed	Chapter 05 - Implementation
Obj8	Evaluate the prototype thoroughly with the test data.	Completed	Chapter 06 – Testing and Evaluation
Obj9	Complete the research documentation of the project.	Completed	Chapter 07 - Conclusion

7.3 Obstacles Faced During the Project

During the course of this project lots of challenges came up and some of the major challenges are listed below.

- Finding a research area and establishing a proper scope.

  It was difficult at first to come up with a research area for the project. After consulting several lecturers the research area was finalized. However when it comes to the project scope it was difficult to achieve as an undergraduate student due to the complexity of the research area. In order to remedy this situation, a literature survey had to be carried out with the expert opinions and the scope was fixed according to the time frame and complexity of the project.

- Less research materials to refer.

  Even though the multiobjective optimization is currently being used in several engineering fields, it is a fairly new research area. Therefore it was very hard to find lots of research materials on the internet. However by referring to web libraries like IEEE and some engineering threads, some of the best research materials came up to light.

- Limited time frame.

  Since the project had a time constraint, all the tasks had to be managed carefully and in some cases some tasks were done parallel to the others.

- Understanding the complex algorithms.

  Some of the multiobjective optimization algorithms were complex to understand at first. However with the help of experts in the industry, those difficult areas were broken down to small areas and implemented in the prototype.

7.4 Learning Outcomes of the Project

This project cleared way for lots of new paths and provided lots of learning opportunities along the way. The following are the most important learning outcomes of this project.

- How to manage a project in an effective manner.
- Conducting a proper scientific research on a subject.
- Effectively managing time and the tasks.
- Problem solving skills.
- Requirement gathering skills.
- Working with multiobjective optimization algorithms.

7.5 Modules Contributing to the Project

Table 7.2 Modules Contributing to the Project

Module Name	Contribution
Software Development Principles	Provided the basic foundation in to programming and provided more lessons on different approaches in programming.
Object Oriented Principles	One of the key methodologies used in the project was object oriented development

	methodology. This module provided great information on object oriented principles and object oriented design aspects.
Information Systems	This module gave a good understanding on designing the database systems and modeling them. This was a great help when it came to designing the information architecture of this project.
Mathematics for Computing	Provided the basic mathematics knowledge which is required when it comes to software engineering. This module was really helpful when understanding the objective functions in some multiobjective optimization algorithms.
Algorithms and Complexity	One of the most helpful modules, which lay the ground work for this project. It gave a solid background on working with algorithms, creating new algorithms and amending the current algorithms. This was extremely helpful when working with NSGA-II and ENS algorithms.
Project Management	This module provided some valuable insights to manage an actual project minimizing all the issues and tackling an issue when comes to light. The overall module helped us tackle this one year long project in the most effective manner.

7.6 Future Enhancements

Since this project was completed as a part of the degree program, only the essential features were implemented. During this process some great features were identified and kept aside as the future enhancements of this project. Following are the future enhancements identified in this project.

- Ability to add more parameters to the sorting function.

  Current prototype only allows 2 parameters which are deadline and priority. By allowing the user to add more parameters to the sorting process increases the accuracy of the sorting mechanism and it will add more flexibility to the product.

- Allow user to write custom equations to tune the sorting function according to their needs.

  At the moment the weighting of the parameters are fixed. By allowing the users to write custom equations to change the weights of the parameters, users can add more parameters to the process and weight them according to their preference.

- Allow multiple platforms so this can be used from anywhere.

  By adding more platforms, especially the mobile platform, people can use this product in a very convenient manner and they can simply arrange the tasks and give it to their team. In this case they can do this on-the-go so they can save more time.

- Make a plugin so it can be integrated with project management tools directly.

  At the moment users have to export the tasks from their project management tool and sort using this product before importing it back to their project management tool. This can be a somewhat inconvenient to the user. By making a plugin, users can easily sort their tasks without having to export and import back to their tool. Also this increases the security since user does not have to keep separate excel sheets of their project tasks.

- Increase the security of the overall application.

  When it comes to the industry, security of their data is a huge priority because it can give some valuable insights to their competitors about their new products. In order to prevent this sort of breach, it is mandatory to use encryption when handling the data.

7.7 Concluding Remarks

Primary objective of this project was to design and implement a system which can be used to sort given number of tasks in an effective manner. At this point this whole project can be concluded as a success considering all the aims and objectives achieved, all the knowledge gain in various fields from conducting a research to developing a fully functional prototype. This thesis has given a comprehensive account on the research aspects of this project as well as the designing, implementation, testing and evaluation aspects too. Furthermore the next steps to this project have also been discussed in this last chapter, in order to make this sorting mechanism a more accurate and a usable one.

To sum up, this journey was not a smooth one but was an extremely knowledgeable one. It was a journey with many challenges, obstacles and uncertainty but at the end, this journey was a successful one.

References

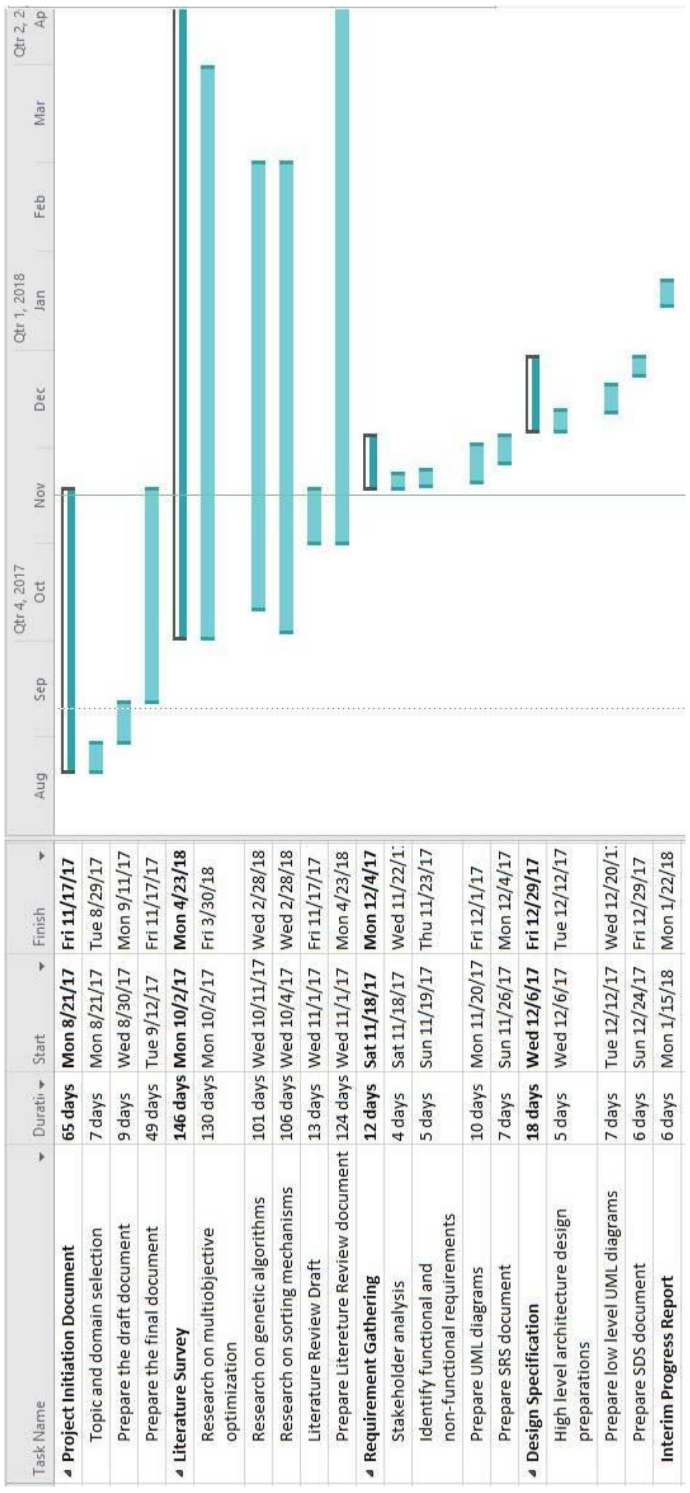
- Abramov I.V., Illarionov I.V., Matveev M.G. (2016) The Process-Oriented Approach for Designing a Project Management System. In: Becker J., Kozyrev O., Babkin E., Taratukhin V., Aseeva N. (eds) Emerging Trends in Information Systems. Progress in IS. Springer, Cham
- Ahmed, F. and Deb, K. (2012). Multi-objective optimal path planning using elitist non-dominated sorting genetic algorithms.
- Ali, L. and A, V. (2017). *What is Agile model – advantages, disadvantages and when to use it?*. [online] Istqbexamcertification.com. Available at: http://istqbexamcertification.com/what-is-agile-model-advantages-disadvantages-and-when-to-use-it/ [Accessed 8 Nov. 2017].
- Almeida, L.H., Pinheiro, P.R., Albuquerque, A.B.: Applying Multi-Criteria Decision Analysis to Global Software Development with Scrum Project Planning. In: Yao, J., Ramanna, S., Wang, G., Suraj, Z. (eds.) RSKT 2011. LNCS, vol. 6954, pp. 311–320. Springer, Heidelberg (2011)
- Clymont, K. and Keedwell, E. (2012). *Deductive sort and climbing sort: New methods for nondominated sorting*.
- Coelho, E. and Basu, A. (2012). Effort Estimation in Agile Software Development using Story Points. [online] Available at: https://pdfs.semanticscholar.org/0ae7/5cc0678f6b01f1066093b19cad9dd4379385.pdf [Accessed 24 Jan. 2018].
- Das, M., Jagdale, T. and sheheer, T. (2017). *What is Spiral model- advantages, disadvantages and when to use it?*. [online] Istqbexamcertification.com. Available at: http://istqbexamcertification.com/what-is-spiral-model-advantages-disadvantages-and-when-to-use-it/ [Accessed 15 Nov. 2017].
- Dayani, M. (2017). Software Project Planning Through Simulation of Entire Project's Problem-Space.
- Deb K. (2015) Multi-Objective Evolutionary Algorithms. In: Kacprzyk J., Pedrycz W. (eds) Springer Handbook of Computational Intelligence. Springer, Berlin, Heidelberg
- Discenza, R. & Forman, J. B. (2007). Seven causes of project failure: how to recognize them and how to initiate project recovery. Paper presented at PMI® Global Congress 2007—North America, Atlanta, GA. Newtown Square, PA: Project Management Institute.
- Filho M.S., Pinheiro P.R., Albuquerque A.B. (2015) Task Allocation Approaches in Distributed Agile Software Development: A Quasi-systematic Review. In: Silhavy R., Senkerik R., Oplatkova Z., Prokopova Z., Silhavy P. (eds) Software Engineering in Intelligent Systems. Advances in Intelligent Systems and Computing, vol 349. Springer, Cha

- Gosenheimer, C. (2017). *PROJECT PRIORITIZATION. A STRUCTURED APPROACH TO WORKING ON WHAT MATTERS MOST*. [ebook] Office of Quality Improvement. Available at: https://oqi.wisc.edu/resourcelibrary/uploads/resources/Project_Prioritization_Guide_v_1.pdf [Accessed 5 Nov. 2017].
- Hussain, M., gemeda, T. and Moe, N. (2017). *What is Waterfall model- advantages, disadvantages and when to use it?*. [online] Istqbexamcertification.com. Available at: http://istqbexamcertification.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/ [Accessed 15 Nov. 2017].
- Ika, L. A. (2012). Project management for development in Africa: why projects are failing and what can be done about it. Project Management Journal, 43(4), 27–41. doi: http://dx.doi.org/10.1002/pmj.21281
- Ishibuchi, H., Imada, R., Setoguchi, Y. and Nojima, Y. (2016). *Performance Comparison of NSGA-II and NSGA-III on Various Many-Objective Test Problems*.
- Kacprzyk, J. and Pedrycz, W. (2015). Springer Handbook of Computational Intelligence. Berlin, Heidelberg: Springer, pp.995-1015.Lara, A., Sanchez, G. and Coello, C. (2009). HCS: A New Local Search Strategy for Memetic Multiobjective Evolutionary Algorithms.
- Lara, A., Sanchez, G. and Coello, C. (2009). HCS: A New Local Search Strategy for Memetic Multiobjective Evolutionary Algorithms.
- Masood Z., Hoda R., Blincoe K. (2017) Exploring Workflow Mechanisms and Task Allocation Strategies in Agile Software Teams. In: Baumeister H., Lichter H., Riebisch M. (eds) Agile Processes in Software Engineering and Extreme Programming. XP 2017. Lecture Notes in Business Information Processing, vol 283. Springer, Cham
- Mazikana, T. and dogzky, S. (2017). *What is Prototype model- advantages, disadvantages and when to use it?*. [online] Istqbexamcertification.com. Available at: http://istqbexamcertification.com/what-is-prototype-model-advantages-disadvantages-and-when-to-use-it/ [Accessed 11 Nov. 2017].
- McClymont, K. and Keedwell, E. (2012). Deductive Sort and Climbing Sort: New Methods for Non-Dominated Sorting.
- Shah-Hosseini, H. (2007). *Problem Solving by intelligent water drops*.
- Shah-Hosseini, H. (2008). *Intelligent Water drops algorithm: a new optimization method for solving the multiple knapsack problem*.
- Shah-Hosseini, H. (2009). *The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm*.
- Sheng, W., Liu, Y., Meng, X. and Zhang, T. (2012). An Improved Strength Pareto Evolutionary Algorithm 2 with application to the optimization of distributed generations.

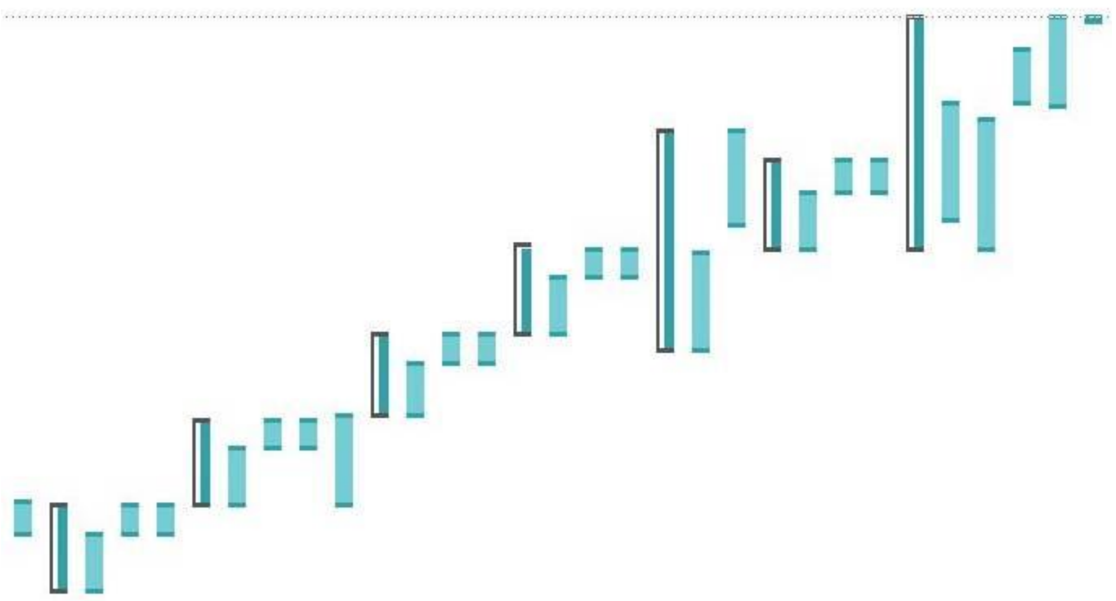
- Verma, V. (2017). *What is Iterative model- advantages, disadvantages and when to use it?*. [online] Istqbexamcertification.com. Available at: http://istqbexamcertification.com/what-is-iterative-model-advantages-disadvantages-and-when-to-use-it/ [Accessed 15 Nov. 2017].
- Zhang, X., Tian, Y., Cheng, R. and Jin, Y. (2015). *An Efficient Approach to Nondominated Evolutionary Multiobjective Optimization*.
- Zitzler, E. and Thiele, L. (1999). *Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach*.

Appendix A – Project Management

9.1 Activity Schedule

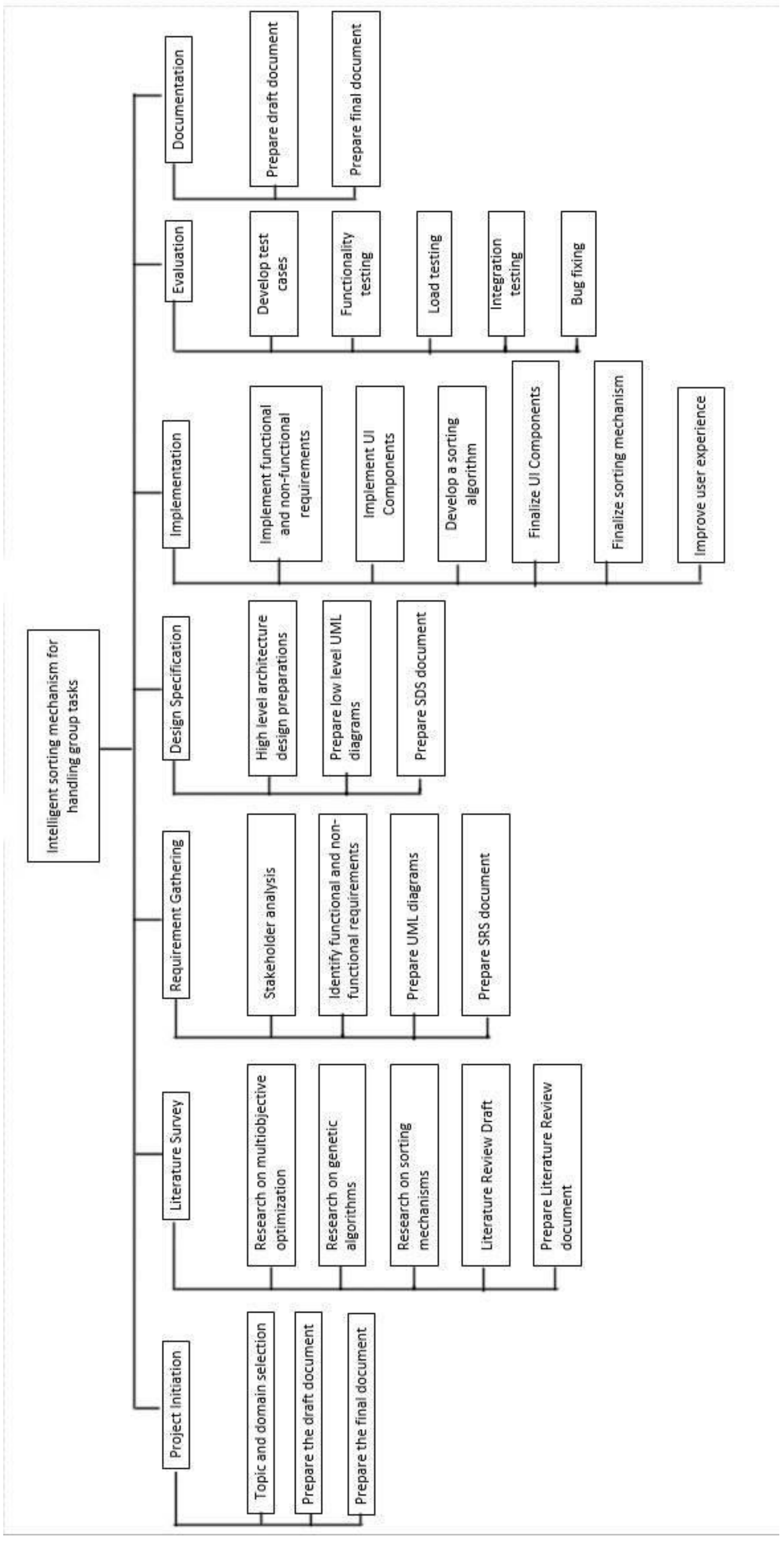


Task Name	Duration	Start	Finish
Project Initiation Document	**65 days**	**Mon 8/21/17**	**Fri 11/17/17**
Topic and domain selection	7 days	Mon 8/21/17	Tue 8/29/17
Prepare the draft document	9 days	Wed 8/30/17	Mon 9/11/17
Prepare the final document	49 days	Tue 9/12/17	Fri 11/17/17
Literature Survey	**146 days**	**Mon 10/2/17**	**Mon 4/23/18**
Research on multiobjective optimization	130 days	Mon 10/2/17	Fri 3/30/18
Research on genetic algorithms	101 days	Wed 10/11/17	Wed 2/28/18
Research on sorting mechanisms	106 days	Wed 10/4/17	Wed 2/28/18
Literature Review Draft	13 days	Wed 11/1/17	Fri 11/17/17
Prepare Litereture Review document	124 days	Wed 11/1/17	Mon 4/23/18
Requirement Gathering	**12 days**	**Sat 11/18/17**	**Mon 12/4/17**
Stakeholder analysis	4 days	Sat 11/18/17	Wed 11/22/1
Identify functional and non-functional requirements	5 days	Sun 11/19/17	Thu 11/23/17
Prepare UML diagrams	10 days	Mon 11/20/17	Fri 12/1/17
Prepare SRS document	7 days	Sun 11/26/17	Mon 12/4/17
Design Specification	**18 days**	**Wed 12/6/17**	**Fri 12/29/17**
High level architecture design preparations	5 days	Wed 12/6/17	Tue 12/12/17
Prepare low level UML diagrams	7 days	Tue 12/12/17	Wed 12/20/1
Prepare SDS document	6 days	Sun 12/24/17	Fri 12/29/17
Interim Progress Report	**6 days**	**Mon 1/15/18**	**Mon 1/22/18**



Interim Progress Report	6 days	Mon 1/15/18	Mon 1/22/18
◢ Implementation Phases 01	**16 days**	**Mon 1/1/18**	**Sun 1/21/18**
Implementing the key functions	11 days	Mon 1/1/18	Sun 1/14/18
Testing the prototype	6 days	Mon 1/15/18	Sun 1/21/18
Bug Fixing	6 days	Mon 1/15/18	Sun 1/21/18
◢ Implementation Phases 02	**16 days**	**Mon 1/22/18**	**Sun 2/11/18**
Implementing the key functions	11 days	Mon 1/22/18	Sun 2/4/18
Testing the prototype	6 days	Mon 2/5/18	Sun 2/11/18
Bug Fixing	6 days	Mon 2/5/18	Sun 2/11/18
Prototype Demonstration	16 days	Mon 1/22/18	Mon 2/12/18
◢ Implementation Phases 03	**15 days**	**Tue 2/13/18**	**Sun 3/4/18**
Implementing the key functions	10 days	Tue 2/13/18	Sun 2/25/18
Testing the prototype	6 days	Mon 2/26/18	Sun 3/4/18
Bug Fixing	6 days	Mon 2/26/18	Sun 3/4/18
◢ Implementation Phases 04	**16 days**	**Mon 3/5/18**	**Mon 3/26/18**
Implementing the key functions	11 days	Mon 3/5/18	Sun 3/18/18
Testing the prototype	6 days	Mon 3/19/18	Sun 3/25/18
Bug Fixing	6 days	Mon 3/19/18	Sun 3/25/18
◢ Documentation	**38 days**	**Thu 3/1/18**	**Mon 4/23/18**
Prepare a draft document	18 days	Thu 3/1/18	Sat 3/24/18
Prepare the final document	17 days	Sun 4/1/18	Mon 4/23/18
◢ Implementation Phases 05	**16 days**	**Mon 3/26/18**	**Mon 4/16/18**
Implementing the key functions	11 days	Mon 3/26/18	Sun 4/8/18
Testing the prototype	6 days	Mon 4/9/18	Mon 4/16/18
Bug Fixing	6 days	Mon 4/9/18	Mon 4/16/18
◢ Final Implementation Phase	**41 days**	**Mon 3/26/18**	**Mon 5/21/18**
Finalizing the UI components	21 days	Mon 4/2/18	Mon 4/30/18
Finalizing the sorting mechanism	24 days	Mon 3/26/18	Thu 4/26/18
Improving the user experience	10 days	Tue 5/1/18	Sun 5/13/18
Testing the prototype	16 days	Mon 4/30/18	Mon 5/21/18
Viva	1 day	Mon 5/21/18	Mon 5/21/18

9.2 Work Breakdown Structure


Intelligent sorting mechanism for handling group tasks
Project Initiation
Topic and domain selection
Prepare the draft document
Prepare the final document
Literature Survey
Research on multiobjective optimization
Research on genetic algorithms
Research on sorting mechanisms
Literature Review Draft
Prepare Literature Review document
Requirement Gathering
Stakeholder analysis
Identify functional and non-functional requirements
Prepare UML diagrams
Prepare SRS document
Design Specification
High level architecture design preparations
Prepare low level UML diagrams
Prepare SDS document
Implementation
Implement functional and non-functional requirements
Implement UI Components
Develop a sorting algorithm
Finalize UI Components
Finalize sorting mechanism
Improve user experience
Evaluation
Develop test cases
Functionality testing
Load testing
Integration testing
Bug fixing
Documentation
Prepare draft document
Prepare final document

Appendix B – Risks and Mitigation Plan

01. Lack of domain knowledge | High Risk

- Do in-depth background studies.
- Conduct a proper literature-survey.
- Get constant feedbacks from domain experts.

02. Failing to achieve deadlines | High Risk

- Allocate more working hours to the project.
- Use already developed algorithms if possible.
- Use third party tools and libraries for UI optimization.

03. Often requirement changes | High Risk

- Avoid any major changes to the core code.
- Reject unnecessary requirements.
- Re-evaluate the important features so it will not change again.

04. Bugs in the final version of the prototype | High Risk

- Use a versioning tool for the development.
- Follow best practices in coding.
- Always follow the designed architecture.

05. Loss of data | High Risk

- Backup day-to-day data to cloud.
- Synchronize the code and documentation with a real-time cloud based tool.

06. Hardware Failures | High Risk

- Configure an extra machine in case of an emergency.
- Keep all the necessary software to configure a new computer immediately.