**Meeting Manager**

**Date: 12/03/2018**

**Group: 6**

**Student Names: Ross Maider, Sarah Hartley, Arran Roy and Oliver Esdaile**

Team Leader: Ross

Secretary: Sarah

Morale Manager: Arran

**Introduction**

The problem we have been given is to create a program that is a 'Meeting Manager'. This program must be able to store the diaries of several employees and store information about the meetings they have scheduled. It must also be able to add new meetings and delete/edit existing meetings. An undo feature must also be included, to reverse the users last action. And finally, the program must be able to compare all employees diaries to find times that are available to everyone, so that a meeting can be set up. This meeting would then be added to the diary of each employee involved.

**Requirements**

Functional requirements:

R1: The program shall be able to add employees

R2: The program shall be able to create meetings

R3: The program shall have an undo feature

R4: The program shall be able to delete meetings

R5: The program shall be able to edit meetings

R6: The program shall be able to display diary entries for a given employee

R7: The program shall be able to save diary entries for an employee

R8: The program shall be able to load diary entries for an employee

R9: The program shall be able to search for free time for a number of employees, and display all of the times available for meetings.

Non-functional requirements:

R1: The user must have a java virtual machine installed to be able to run the program.

R2: The program shall comply with the data protection acts

R3: The program shall run smoothly

Assumptions:

1: The user will be able to use a keyboard and mouse

**Task Allocations**

Design:

Arran:

* Report
* Introduction

Sarah:

* Requirements
* Pseudocode/flowcharts

Ross:

* Class diagram
* Pseudocode/flowcharts

Oliver:

* User interface/sketches

Coding:

Arran:

* Stack class
* Menu class

Ross:

* Meeting class
* Diary class

Sarah:

* Employee class
* EmployeeList class

**Designs**

Pseudocode for addEmployee(){

Get name and ID from user

Create new employee

Add employee to list

}

Pseudocode for deleteEmployee(){

Find employee to delete

Get previous employee in list

Previous.setNext(toDelete.getNext())

}

Pseudocode for findEmployee(int ID){

While (idToFind != current.getID() && current.getNext() != null){

Current.getNext()

If (current == idToFind){

Return true;

}

}

Return false;

}

Pseudocode for searchMeeting(){

Get names, start date, end date and … from user

For each day{

For each hour of day{

Set holdMeeting to true

For each name {

If node in tree at the time already exists then set holdMeeting to false;

}

If (holdMeeting == true){

Display possible time

}

}

}

Get chosen time from user

Add meeting to each employees diary

}

**Management document:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| DATE | TIME | LOCATION | ATTENDANCE | PROGRESS MADE | TASKS ALLOCATED |
| 08/03/2018 | 15:00-16:50 | QMB | Ross & Sarah | Requirements, class diagram and pseudocode for add, delete and find employee done. Worked on by Ross & Sarah. | Arran to do project description and any additional research needed. Oliver to design a user interface. |
| 12/03/2018 | 15:00-17:50 | QMB | Ross, Sarah & Arran | Everyone helped finalise class design. Decided on using a linked list to store the employees, and a binary tree to store the meetings. | Sarah to design user interface, Arran to structure the report and Ross to finalise and submit it. |

**User interface sketch:**



