Topic : Resource Booking System

Group no : MLB_WD_CSNE_13_03

Campus : Malabe

Submission Date : 20/05/2022

We declare that this is our own work and this Assignment does not incorporate without acknowledgment any material previously submitted by anyone else in SLIIT or any other university/Institute. And we declare that each one of us equally contributed to the completion of this Assignment.

Registration No	**Name**	**Contact Number**
IT21302558	Mathuranage P. N. M	0788263616
IT21259784	Serasinghe D. A. Y. A	0702385345
IT21300196	Manathunga M. D. T. L	070 2763025
IT21293108	Bogahawatta C. A	076 655 3831
IT21307812	Vaffa M. A. M. A	071 2906494

System Requirements

•In a construction company there are many resources.

•Main resources are vehicles, materials and tools.

•Vehicle details, tool details, material details and cost per unit of materials are stored.

•These resources are used to do various projects.

•According to the nature of the project each part of the project will be assigned to its respective department.

•Employees and projects are managed by the department they belong to.

•A Resource Booking System is used to manage the resources booked to these projects and to store project details.

•This system contains the details and cost of resources.

•In this system there are two types of members, they are employees and executive.

•Employee can view the project there assigned to and details of the project.

•Executives can view, add, edit or delete resources and projects.

•The resource booking system automatically generates a cost report and assigned resources report for given project.

•Relevant resources for a project are booked by executives.

Noun Verb analysis

Classes and Methods

Resources	- details and cost of resources
Projects	- project details
Department	- Employees and projects are managed
Employee	- view the project there assigned to
Executives	- view, add, edit or delete resources and projects
Vehicles	- vehicles details
Materials	- material details, cost per unit
Tools	- tools details
Report	- Cost report, Assigned resources report

CRC Cards

Resources	
Responsibility	Collaborators
Details and cost of resources	Vehicles Materials Tools

Projects	
Responsibility	Collaborators
Store details about projects	

Departments	
Responsibility	Collaborators
Store details about departments	Employee Project

Vehicles	
Responsibility	Collaborators
Store details about vehicles	

Tools	
Responsibility	Collaborators
Store details about tools	

Materials	
Responsibility	Collaborators
Store details about materials	

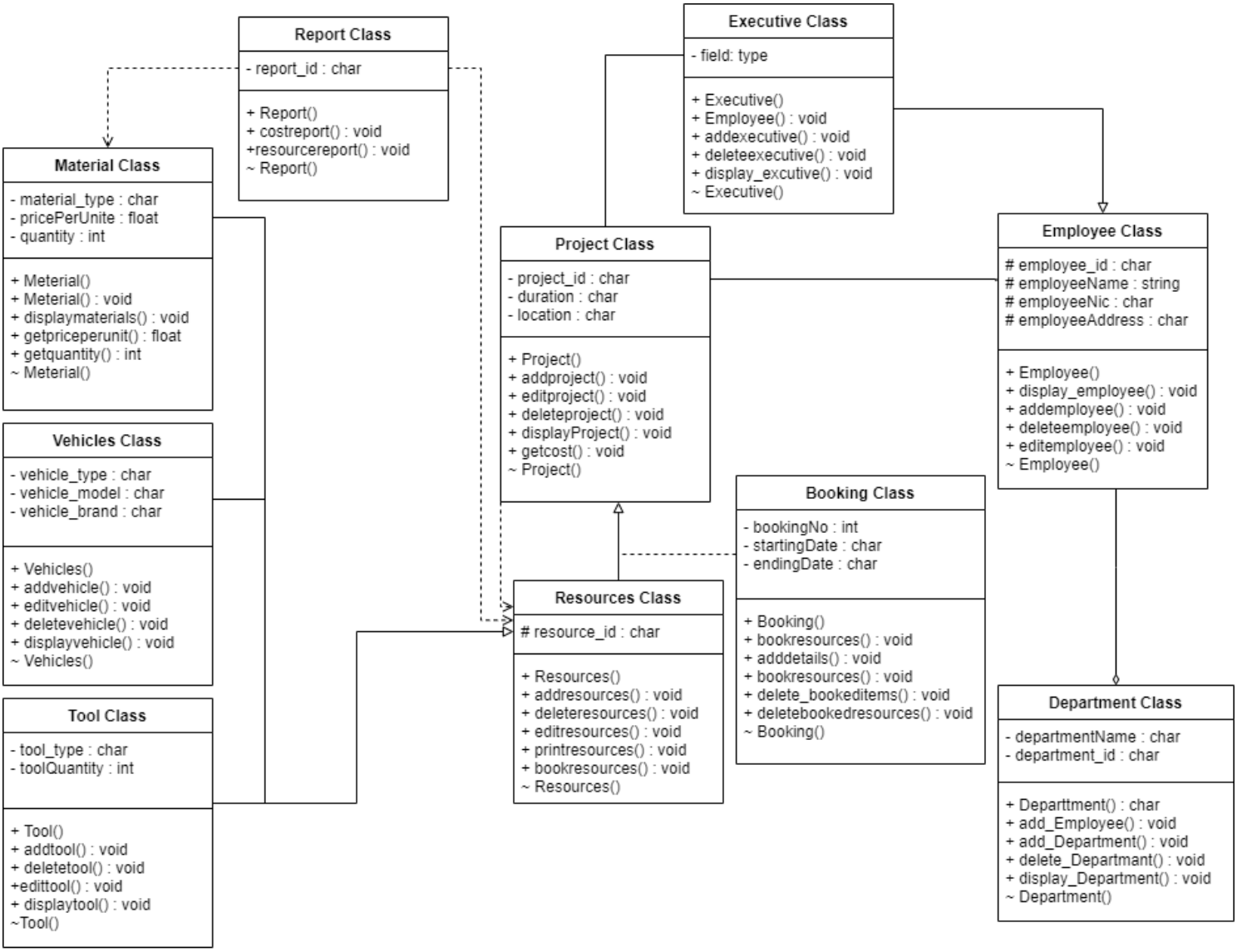
Employee	
Responsibility	Collaborators
Store details about Employees	

Executives	
Responsibility	Collaborators
Add, edit or remove details of resources	Resources
Add, edit or remove project details	Project

Report	
Responsibility	Collaborators
Cost report	Materials
Assigned resources report for a project	Resources Project

Booking	
Responsibility	Collaborators
Book resources to projects	Resources Projects
Assign employees to projects	Employees Projects

UML Notation



Main.cpp

```
#include <iostream>
#include "Employee.h"
#include "Executive.h"
#include "Department.h"
#include "Resources.h"
#include "Tool.h"
#include "Vehicle.h"
#include "Material.h"
#include "Project.h"
#include "Report.h"
#include "Booking.h"
using namespace std;
int main() {
	Employee *em1;
	Employee  *em2;
	em1 = new Employee("E01", "Saman Kumara", "198526354V", "No 6,
	Colombo 7" );
	em2 = new Employee("E02", "Malindu Kumara", "199856234V", "No 5,
	Kaduwela, Malabe");
	em2->display_employee();
	Executive *manager;
	manager = new Executive("Manager", em1);
	manager->display_executive();
	Department  *IT;
	IT = new Department("IT");
	IT-> add_Employee(em2);

	IT->display_Department();
	Resources *resource1, *resource2, *resource3;
	resource1 = new Resources("T001");
	resource2 = new Resources("V001");
	resource3 = new Resources("M001");
	resource1->printrsources();
	Tool *tool1;
	tool1 = new Tool("Drill", 5, resource1);
	tool1->displayTools();
	Vehicle *vehicle1;
	vehicle1 = new Vehicle("Lorry", "58E", "TATA", resource2);
	vehicle1->displayVehicles();
	Material *mat1;
	mat1 = new Material("Sand", 25000, 45879, resource3);
	mat1->displaymaterials();
	Project *pr1;
	pr1 = new Project("P001", "Year", "Colombo", manager, resource1,
	resource2, resource3);
	Report *report[2];
```

```
        report[0] = new Report();
        report[1] = new Report();
        report[0]->costreport(001, mat1);
        report[1]->resourcereport(002, pr1);
        Booking *book1;
        book1 = new Booking();

        delete em1;
        delete em2;
        delete manager;
        delete IT;
        delete resource1;
        delete resource2;
        delete resource3;
        delete tool1;
        delete vehicle1;
        delete mat1;
        delete pr1;
        delete report[0];
        delete report[1];
        delete book1;
        return 0;
}
```

IT21302558 - Mathuranage P.N.M

Report.h

```
#pragma once
#include "Material.h"
#include "Project.h"
class Report{
	private:
		 int report_no;
	public:
		Report();
		void costreport(int no, Material *m1);
		void resourcereport(int no, Project *p1);
		~Report();
};
```

Report.cpp

```
#include "Report.h"

#include "Material.h"

#include "Project.h"

#include <iostream>

using namespace std;

Report::Report(){

}

void Report::costreport(int no, Material *m1){

        cout << "\n" << endl;

        m1->displaymaterials();

        cout << "\nTotal  Cost - "<<( m1->getquantity() * m1->getpriceperunit()) <<
        "\n" << endl;

}

void Report::resourcereport(int no, Project *p1){

        p1->displayProject();

}

Report::~Report(){

        cout << "Deleting Report" << endl;

}
```

Project.h

```
#pragma once
#include "Executive.h"
#include "Resources.h"


class Project{
 private:
        Executive *exucutive[1];
        Resources *resources[3];
        char projectid[5];
        char duration[10];
        char location[10];
 public:
        Project();
        Project(const char id[5], const char pduration[10], const char plocation[10], Executive *ex1, Resources *re1, Resources *re2, Resources *re3);
        void addproject();
        void editproject();
        void deleteproject();
        void displayProject();
        void getcost();
        ~Project();
};
```

Project.cpp

```
#include "Project.h"

#include "Executive.h"

#include <cstring>

#include <iostream>

using namespace std;

Project::Project(){

}

        Project::Project(const char id[5], const char pduration[10], const char
plocation[10], Executive *ex1, Resources *re1, Resources *re2, Resources *re3){

        strcpy(projectid, id);

         strcpy(duration, pduration);

         strcpy(location, plocation);

        exucutive[0] = ex1;

        resources[0] = re1;

         resources[1] = re2;

         resources[2] = re3;

}

void Project::displayProject(){

         cout << "Project ID      - " << projectid << endl;

         cout << "       Duration - " << duration << endl;

         cout << "       Location - " << location << endl;

         cout << "Booked by       - " ;

         exucutive[0]->display_executive();

         cout << "Booked Resources   " << endl;


        for(int count = 0; count < 3 ; count ++){

                resources[count]->printrsources();

        }
```

```
}
        void Project::editproject(){
}
        void Project::deleteproject(){
}
 void Project::getcost(){
         resources[0] ;
         resources[1] ;
         resources[2] ;
}
Project::~Project(){
        cout << "Delete Project" << endl;
}
```

IT21259784 - Serasinghe D. A. Y. A

Tools.h

```
#pragma once

#include "Resources.h"


class Tool : public Resources {

 private:

        Resources *resource[1];

        char tool_Name[20];

        int quantity;


 public:

        Tool();

       Tool(const char name[20], int tquantity, Resources *res1);

        void addtools();

        void edittools();

        void deletetools();

        void displayTools();

        ~Tool();

};
```

Tools.cpp

```
#include "Tool.h"
#include "Resources.h"
#include <cstring>
#include <iostream>
using namespace std;
Tool::Tool(){
        strcpy(tool_Name, "");
        quantity = 0;
}
Tool::Tool(const char name[20], int tquantity, Resources *res1){
        strcpy(tool_Name, name);
        quantity = tquantity;
         resource[0] = res1;
}
void Tool::addtools(){

}
void Tool::edittools(){
}
void Tool::deletetools(){
}

void Tool::displayTools(){
        cout << "\n" << endl;
        cout << "Tool ID      - " ;
```

```
 for(int count = 0; count < 1; count++){
            resource[count]->printrsources();
 }
          cout << "     Name     - " << tool_Name << endl;
          cout << "     Quantity - " << quantity << endl;
 }
Tool::~Tool(){
          cout << "Deleting Tool" << endl;
}
```

Resources.h

```
#pragma once
class Resources{
 protected:
        char resource_id[5];
 public:
         Resources();
         Resources(const char rid[5]);
         void addresources();
        void deletereouces();
        void editresources();
        void printrsources();
        void bookresources();
         ~Resources();
};
```

Resources.cpp

```
#include "Resources.h"
#include <cstring>
#include <iostream>
using namespace std;
Resources::Resources(){
        strcpy(resource_id, "");
}
Resources::Resources(const char rid[5]){
        strcpy(resource_id, rid);
}
        void addresources(){
}
        void deletereouces(){
}
        void editresources()
}
void Resources::printrsources(){
        cout << resource_id << endl;
}
        void Resources::bookresources(){
}
Resources::~Resources(){
        cout << "Deleting Resource " << endl;
}
```

IT21293108 - Bogahawatta C.A.

Executive.h

```
#pragma once
#include "Employee.h"
class Executive: public Employee
{
        private:
                Employee *emp[1];
   char position[10];
        public:
                Executive();
                Executive(const char post[10], Employee *employee1);
   void addexecutive();
   void deleteexecutive();
   void display_executive();
                ~Executive();
};
```

Executive.cpp

```
#include "Executive.h"
#include <iostream>
#include <cstring>
using namespace std;
Executive::Executive() {
        strcpy(position , "");
}
Executive::Executive(const char post[10], Employee *employee1) {
        strcpy(position, post);
  emp[0] = employee1;
}
void Executive::addexecutive(){
}
void Executive::deleteexecutive(){
}
void Executive::display_executive(){
        cout <<  "\n" << position;
  for(int count = 0; count < 1; count++){
         emp[count]->display_employee();
  }
}
Executive::~Executive(){
        cout << "\nDeleting Executive - " << employee_ID << endl;
}
```

Employee.h

```
#pragma once

class Employee

{

        protected:

          char employee_ID[4];

          char name[20];

          char nic[11];

          char address[25];

        public:

          Employee();

          Employee(const char e_id[4], const char e_name[20], const char e_nic[11],
          const char e_address[25]);

          void display_employee();

          void addemployee();

           void deleteemployee();

           void editemployee();

          ~Employee();


};
```

Employee.cpp

```
#include "Employee.h"

#include <iostream>

#include <cstring>

using namespace std;

Employee::Employee() {

        strcpy(employee_ID , "");

        strcpy(name , "");

        strcpy(nic , "");

        strcpy(address , "");

}

Employee::Employee(const char e_id[4], const char e_name[20], const char
e_nic[10], const char e_address[25]){

        strcpy(employee_ID , e_id);

        strcpy(name , e_name);

        strcpy(nic , e_nic);

        strcpy(address , e_address);

}

void Employee::display_employee(){

        cout << "\n" << endl;

        cout << "ID      -"  << employee_ID << endl;

        cout << "Name    -" << name << endl;

        cout << "NIC     -" << nic << endl;

        cout << "Address -" << address << endl;

}


void Employee::addemployee(){

}

void Employee::deleteemployee(){
```

```
}
void Employee::editemployee(){
}
Employee::~Employee() {
        cout << "\nDeleting employee - " << employee_ID << endl;
}
```

IT21307812 - Vaffa M. A. M. A

Materials.h

```
#pragma once

#include "Resources.h"

class Material : public Resources{

	private:

		Resources *materials[1];

		char type[10];

		float priceperunit;

		int quantity;

	public:

		Material();

		Material(const char mtype[10], float price, int quantity, Resources
		*mat1);

		void displaymaterials();

		float getpriceperunit();

		int getquantity();

		~Material();

};
```

Materials.cpp

```
#include "Material.h"

#include <cstring>

#include <iostream>

using namespace std;

Material::Material(){

}

Material::Material(const char mtype[10], float price, int mquantity, Resources
*mat1){

        strcpy(type, mtype);

        priceperunit = price;

        quantity = mquantity;

        materials[0] = mat1;

}

void Material::displaymaterials(){

        cout << "\nMaterial  ID          -";

        for(int count = 0; count < 1; count++){

                materials[count]->printrsources();

        }

        cout << "        Type         -" << type << endl;

        cout << "        Price Per Unit -" << priceperunit << endl;

        cout << "        Quantity     -" << quantity << endl;

}

float Material::getpriceperunit(){

        return priceperunit;

}

int Material::getquantity(){

        return quantity;

}
```

```
Material::~Material(){
        cout << "Deleting Material " << endl;
 }
```

Vehicles.h

```
#pragma once

#include "Resources.h"

class Vehicle : public Resources{

        private:

                Resources *vehicle[1];

                char type[10];

                char model[15];

                char brand[15];

        public:

                Vehicle();

                Vehicle(const char vtype[10],const char vmodel[15], const char
                vbrand[15], Resources *VR1);

                void addvehicles();

                void editvhicles();

                void deletevehicles();

                void displayVehicles();

                ~Vehicle();

};
```

Vehicle.cpp

```
#include "Vehicle.h"

#include <iostream>

#include <cstring>

using namespace std;

Vehicle::Vehicle(){

}

Vehicle::Vehicle(const char vtype[10],const char vmodel[15], const char vbrand[15],
Resources *VR1){

        strcpy(type, vtype);

        strcpy(model, vmodel);

        strcpy(brand, vbrand);

        vehicle[0] = VR1;

}

void Vehicle::displayVehicles(){

        cout << "\nVehicle ID   -" ;

        for(int count = 0; count < 1; count++){

                vehicle[count]->printrsources();

        }

        cout << "       Type  -" << type << endl;

        cout << "       model -" << model << endl;

        cout << "       Brand -" << brand << endl;

}

void Vehicle::addvehicles(){


}

void Vehicle::editvhicles(){


}
```

```
void Vehicle::deletevehicles(){


}
Vehicle::~Vehicle(){
        cout << "Deleting Vehicle" << endl;
}
```

IT21300196 - Manathunga M. D. T. L

Booking.h

```
#include "Project.h"
#include "Resources.h"
class Booking{
        private:
                Resources *resource[1];
                Project *project[1];
                char bookingID[5];
                char startdate[10];
                char enddate[10];
         public:
                Booking();
                void bookresources();
                void adddetails();
                void delete_bookeditems();
                void deletebookedresources();
                ~Booking();
};
```

Booking.cpp

```
#include "Booking.h"
#include <iostream>
using namespace std;
Booking::Booking(){
}
void Booking::adddetails(){
}
void Booking::bookresources(){
}
void Booking::delete_bookeditems(){
}
void Booking::deletebookedresources(){
}
Booking::~Booking(){
        cout << "Deleting Booked Items " << endl;
}
```

Department.h

```
#pragma once
#include "Employee.h"
class Department{
	private:
		Employee *em[1];
		char department_name[10];
	public:
		Department(const char dname[10]);
		void add_Employee(Employee *emp1);
		void add_Department();
		void delete_Department();
		void display_Department();
		~Department();

};
```

Department.cpp

```
#include <iostream>
#include <cstring>
#include "Department.h"
using namespace std;
Department::Department(const char dname[10]){
	strcpy(department_name, dname);
}
void Department::add_Employee(Employee *emp1){
	em[0] = emp1;
}
void Department::add_Department(){
}
void Department::delete_Department(){
}
void Department::display_Department(){
	cout << "\n" <<department_name << " Department" << endl;
	for(int count = 0; count < 1; count++){
		em[count]->display_employee();
	}
}
Department::~Department(){
	cout << "\nDeleting Department - " << department_name << endl;
}
```