Construction Management System

Group Assignment Summarized Document (For External Viewers)

Overview

This document is made for the any external user who wants to view what this group assignment project is about.

This Document contains the following contents

Contents

Version 0: System Requirements

1. customer should be able to login to the system
2. customer can give feedback
3. customer should be able to make projects
4. customer should be able to contact the company regarding to any issue
5. customer should be able to view any blogs posted on the website
6. customer should be able to edit their profile
7. customer can give requirements that they need in their project
8. customer must provide payment to start a contract
9. employee should be able to read their reviews
10. employee should be able to get their dependent
11. employee should be able to supervise the other employees
12. employee should be able to see the payment
13. employee should be able to see their rating
14. employee should be able to view the project details
15. employee should be able to see the department details
16. employee's details must be stored into the system
17. department should store the details of the respective employees in the company
18. department should create, delete, and update the details of when the employee started working at the specific department
19. the department details must be stored in the system such as (name of department, when it was established, the number of employees, etc...)
20. contract should have received payment to conduct the project
21. contract details can be updated anytime
22. payment method should be specified when doing the contract
23. rating will be described according to how the customer views the service
24. rating will contain many different types according to the customers preference
25. rating must be given by the customer

26. rating details must be stored in the system

27. rating must be shown to customers

28. supplier should be able to update the details of their products and the quantity that it is in

29. supplier details must be stored in the system

30. Admin answer FAQ

31. Admin updates project details

32. Admin updates contract details

33. Admin should receive the payment details of the customers contract to keep record of their contract

Classes that were identified during the noun-verb analysis

* Employee
* Customer
* Department
* Contract
* Rating
* Supplier
* Construction Materials
* Payment
* Project
* Dependent
* Supervisor
* Admin

Version 0 : CRC Cards

CRC cards for the Construction Management System

Employee	
Responsibilities:	**Collaborations:**
Login to the system	
Read their own reviews	Rating
Get their dependent details	Dependent
View current project details	Project
View department details	Department
Send dependent details	Dependent
View their own details	

Customer	
Responsibilities:	**Collaborations:**
Login to the system	
Provide feedback	Rating
Make projects	Project
Contact issues with company	
Make contract	Contract, Payment
View the blogs on the site	
Edit their user profile	

Department	
Responsibilities:	**Collaborations:**
Store employee details	Employee
Add new employee details	Employee
Update employee details	Employee
Delete employee details	Employee
Store department details	

Contract	
Responsibilities:	**Collaborations:**
Check if payment received	
Update details of contract any time	Admin
Specify the payment method	Payment
Send Contract details	Project

Rating	
Responsibilities:	**Collaborations:**
Store the rating details	
Show rating to customer	Customer
Delete Rating	Customer
Update Rating	Customer

Supplier	
Responsibilities:	**Collaborations:**
Login to the system	
Update product details	Construction materials
Add product details	Construction materials
Delete product details	Construction materials
Receive order details	

Construction Materials	
Responsibilities:	**Collaborations:**
Set details of construction materials	Supplier
Update details of construction materials	Supplier
Add new construction material details	Supplier

Payment	
Responsibilities:	**Collaborations:**
Valid the payment	
Store Payment details	
Create receipt	
Send Receipt	Customer, Admin

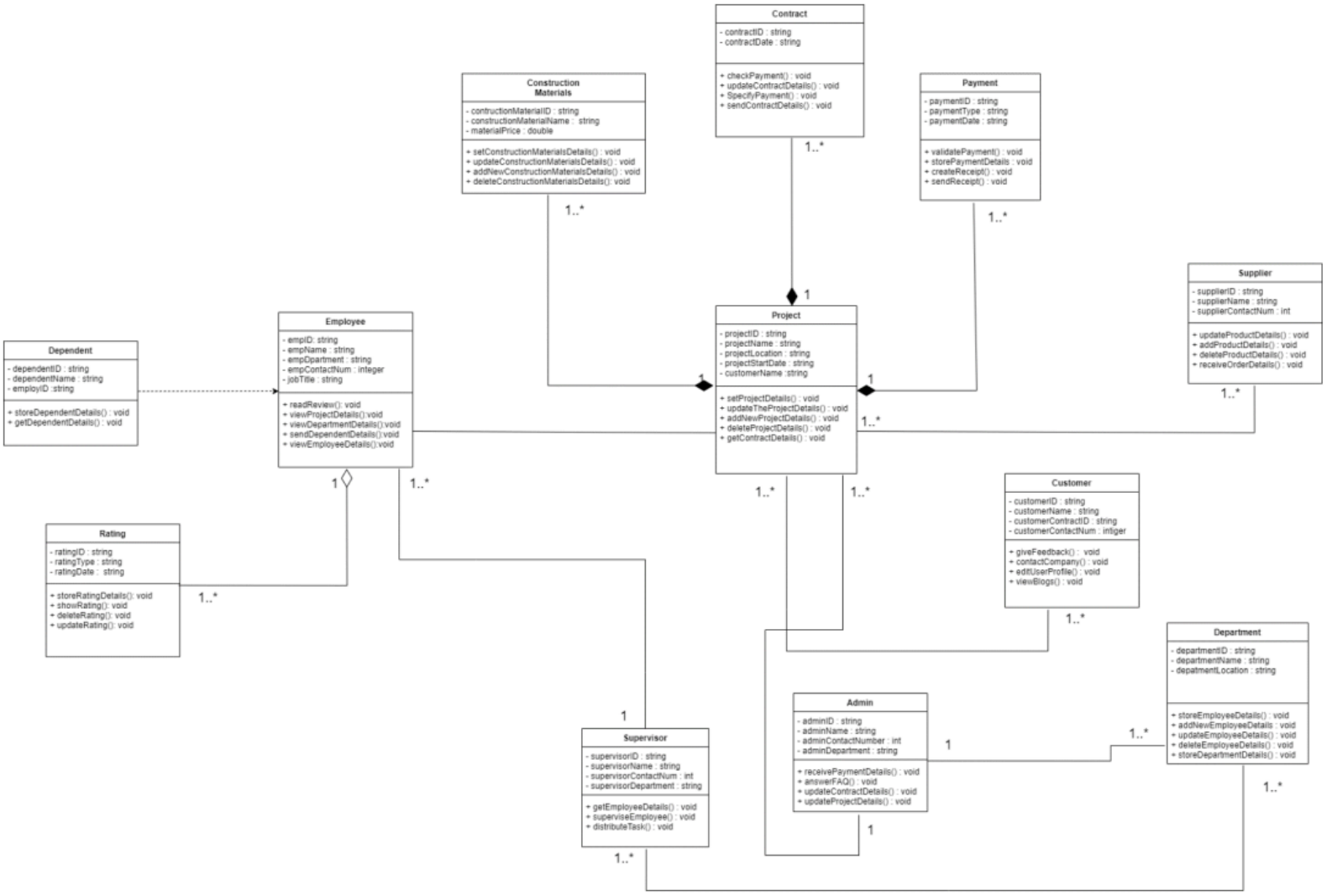
Project	
Responsibilities:	**Collaborations:**
Set the project details	
Update the project details	Admin
Add new project details	Admin
Delete project details	Admin
Get contract details	Contract

Dependent	
Responsibilities:	**Collaborations:**
Store dependent details	
Get dependent details	Employee

Supervisor	
Responsibilities:	**Collaborations:**
Get employee details	Employee
Supervise the employee	
Distribute task to employee	

Admin	
Responsibilities:	**Collaborations:**
Receive payment details	Payment
Answer FAQ	
Update Project details	Project
Update Contract details	Contract

Version 0: UML DIAGRAM



Version 0: Source Code

```
#pragma once
class Admin
{
private:
	//attributes of Admin Class
	char adminID[30];
	char adminName[30];
	int adminContactNumber;
	char adminDepartment[30];

public:
	//constructors
	Admin();
	Admin(const char pAdminID[], const char pAdminName[],int pAdminContactNumber,
const char pAdminDepartment[]);

	//destructor
	~Admin();

	//methods in the class
	void receivePaymentDetails();
	void answerFAQ();
	void updateContractDetails();
	void updateProjectDetails();
};

#pragma once
class ConstructionMaterials
{
private:
	//attributes of the class
	char constructionMaterialID[30];
	char constructionMaterialName[30];
	double materialPrice;

public:
	//constructors
	ConstructionMaterials();
	ConstructionMaterials(const char pConstructionMaterialID[], const char
pConstructionName[], double pMaterialPrice);

	//destructor
	~ConstructionMaterials();

	//methods in the class
	void setConstructionMaterialsDetails();
	void updateConstructionMaterialsDetails();
```

```
        void addNewConstructionMaterialsDetails();
        void deleteConstructionMaterialsDetails();

};

#pragma once
class Contract
{
private:
        //attributes of Contract Class
        char contractID[30];
        char contractDate[30];

public:
        //constructors
        Contract();
        Contract(const char pContractID[], const char pContractDate[]);

        //destructor
        ~Contract();

        //methods in the class
        void checkPayment();
        void updateContractDetails();
        void specifyPaymentMethod();
        void sendContractDetails();

};

#pragma once
class Customer
{
private:
        //attributes of the Customer Class
        char customerID[30];
        char customerName[30];
        char customerContractID[30];
        int customerContactNum;

public:
        //constructors
        Customer();
        Customer(const char pCustomerID[], const char pCustomerName[], const char
pContractContracctID[], int pCustomerContactNum);

        //destructor
        ~Customer();

        //methods in the class
        void giveFeedback();
        void contactCompany();
```

```
	void editUserProfile();
	void viewBlogs();

};

#pragma once
class Department
{
private:
	//attributes of the Department Class
	char departmentID[30];
	char departmentName[30];
	char departmentLocation[30];

public:
	//constructors
	Department();
	Department(const char pDepartmentID[],const char pDepartmentName[], const
char pDepartmentLocation[]);

	//destructor
	~Department();

	//methods in the class
	void storeEmployeeDetails();
	void addNewEmployeeDetails();
	void deleteEmployeeDetails();
	void updateEmployeeDetails();
	void storeDepartmentDetails();
};

#pragma once
class Dependant
{
private:
	//there is a dependancy relationship between Employee and Dependant
	//dependant class requre the employee class inorder to exist
	//if the class is deleted the data relating to dependant is deleted also

	//attributes of Dependant Class
	char dependantID[30];
	char dependantName[30];
	char employeeID[30];

public:
	//constructors
	Dependant();
	Dependant(const char pDependantID[], const char pDependantName[], const char
pEmployeeID[]);

	//destructor
```

```
        ~Dependant();

        //methods
        void storeDependantDetails();
        void getDependantDetails();
};
#pragma once
#include "Dependant.h"
class Employee
{
private:
        //attributes of Employee Class
        char empID[30];
        char empName[30];
        char empDpartment[30];
        int empContactNum;
        char jobTitle[30];

        //Dependant object is needed for employee to add their dependant details
        Dependant *dpend1;

public:
        //Constructors
        Employee();
        Employee(const char pEmpID[], const char pEmpName[], const char
pEmpDpartment[], int pEmpContactNum, const char pJobTitle[]);

        //destructor
        ~Employee();

        //methods in the class
        void readReview();
        void viewProjectDetails();
        void viewDepartmentDetails();
        void sendDependantDetails();
        void viewEmployeeDetails();
};

#pragma once
class Payment
{
private:
        char paymentID[30];
        char paymentType[30];
        char paymentDate[30];

public:
        //constructors
        Payment();
        Payment(const char pPaymentID[], const char pPaymentType[], const char
pPaymentDate[]);
```

```
	//destructor
	~Payment();

	//methods in the class
	void validatePayment();
	void storePaymentDetails();
	void createReceipt();
	void sendReceipt();

};
#pragma once
#include "ConstructionMaterials.h"
#include "Contract.h"
#include "Payment.h"
class Project
{
	//Composition relationship between the Project, Construction Materials,
Contract and Payment classes
	//Project class will not exist unless the construction materials, contract
and payment classes are existing
private:
	//attributes of the Project Class
	char projectID[30];
	char projectName[30];
	char projectLocation[30];
	char projectStartDate[30];
	char customerName[30];

	//objects of Construction Materials, Contract and Payment classes must exist
inorder for the Project class to exist
	ConstructionMaterials* conMat1;
	Contract *contrct1;
	Payment *pymnt1;


public:
	//constructors
	Project();
	Project(const char pProjectID[], const char pProjectName[], const char
pProjectLocation[], const char pProjectStartDate[], const char pCustomerName[]);

	//destructor
	~Project();

	//methods
	void setProjectDetails();
	void updateTheProjectDetails();
	void addNewProjectDetails();
	void deleteProjectDetails();
	void getContractDetails();
```

```
};
#pragma once
class Rating
{
private:
	//Aggregration Relationship between the Rating and the Employee Class
	//When Rating class gets deleted Employee class will remain

	//attributes of Rating Class
	char ratingID[30];
	char ratingType[30];
	char ratingDate[30];

public:
	//constructors
	Rating();
	Rating(const char pRatingID[], const char pRatingType[], const char
pRatingDate[]);

	//destructor
	~Rating();

	//methods in the class
	void storeRatingDetails();
	void showRating();
	void deleteRating();
	void updateRating();
};

#pragma once
class Supervisor
{
private:
	//attributes of the Supervisor Class
	char supervisorID[30];
	char supervisorName[30];
	int supervisorContactNum;
	char supervisorDepartment[30];

public:
	//constructor
	Supervisor();
	Supervisor(const char pSupervisorID[], const char pSupervisorName[],int
pSupervisorContactNum, const char pSupervisorDepartment[]);

	//destructor
	~Supervisor();

	//methods in the class
	void getEmployeeDetails();
```

```
        void superviseEmployee();
        void distributeTask();
};

#pragma once
class Supplier
{
private:
        //attributes of the class
        char supplierID[30];
        char supplierName[30];
        int supplierContactNum;

public:
        //constructors
        Supplier();
        Supplier(const char pSupplierID[], const char pSupplierName[], int
pSupplierContactNum);

        //destructor
        ~Supplier();

        //methods in the class
        void updateProductDetails();
        void addProductDetails();
        void deleteProductDetails();
        void receiveOrderDetails();
};

#include "Admin.h"
#include <iostream>
#include <cstring>

using namespace std;

//constructors
Admin::Admin() {

}
Admin::Admin(const char pAdminID[], const char pAdminName[], int
pAdminContactNumber, const char pAdminDepartment[]) {
        strcpy_s(adminID,pAdminID);
        strcpy_s(adminName,pAdminName);
        adminContactNumber = pAdminContactNumber;
        strcpy_s(adminDepartment,pAdminDepartment);
}

//destructor
Admin::~Admin() {

}
```

```
//methods in the class
void Admin::receivePaymentDetails() {

}
void Admin::answerFAQ() {

}
void Admin::updateContractDetails() {

}
void Admin::updateProjectDetails() {

	}

#include "ConstructionMaterials.h"
#include <iostream>
#include <cstring>

using namespace std;

//constructors
ConstructionMaterials::ConstructionMaterials() {

}
ConstructionMaterials::ConstructionMaterials(const char pConstructionMaterialID[],
const char pConstructionMaterialName[], double pMaterialPrice) {
	strcpy_s(constructionMaterialID,pConstructionMaterialID);
	strcpy_s(constructionMaterialName, pConstructionMaterialName);
	materialPrice = pMaterialPrice;
}

//destructor
ConstructionMaterials::~ConstructionMaterials() {

}

//methods in the class
void ConstructionMaterials::setConstructionMaterialsDetails() {

}
void ConstructionMaterials::updateConstructionMaterialsDetails() {

}
void ConstructionMaterials::addNewConstructionMaterialsDetails() {

}
void ConstructionMaterials::deleteConstructionMaterialsDetails() {

	}
```

```cpp
#include "Contract.h"
#include <iostream>
#include <cstring>

using namespace std;
//constructors
Contract::Contract() {

}
Contract::Contract(const char pContractID[], const char pContractDate[]) {
	strcpy_s(contractID, pContractID);
	strcpy_s(contractDate, pContractDate);
}

//destructor
Contract::~Contract() {

}

//methods in the class
void Contract::checkPayment() {

}
void Contract::updateContractDetails() {

}
void Contract::specifyPaymentMethod() {

}
void Contract::sendContractDetails() {

	}
```

```cpp
#include "Customer.h"
#include <iostream>
#include <cstring>

using namespace std;

//constructors
Customer::Customer() {

}
Customer::Customer(const char pCustomerID[], const char pCustomerName[], const char
pCustomerContracctID[], int pCustomerContactNum) {
	strcpy_s(customerID,pCustomerID);
	strcpy_s(customerName,pCustomerName);
	strcpy_s(customerContractID,pCustomerContracctID);
	customerContactNum = pCustomerContactNum;
}
```

```cpp
//destructor
Customer::~Customer() {

}

//methods in the class
void Customer::giveFeedback() {

}
void Customer::contactCompany() {

}
void Customer::editUserProfile() {

}
void Customer::viewBlogs() {

	}

#include "Department.h"
#include <iostream>
#include <cstring>

using namespace std;

//constructors
Department::Department() {

}
Department::Department(const char pDepartmentID[], const char pDepartmentName[],
const char pDepartmentLocation[]) {
	strcpy_s(departmentID,pDepartmentID);
	strcpy_s(departmentName,pDepartmentName);
	strcpy_s(departmentLocation,pDepartmentLocation);
}

//destructor
Department::~Department() {

}

//methods in the class
void Department::storeEmployeeDetails() {

}
void Department::addNewEmployeeDetails() {

}
void Department::deleteEmployeeDetails() {

}
```

```
void Department::updateEmployeeDetails() {

}
void Department::storeDepartmentDetails() {

	}

#include "Dependant.h"
#include <iostream>
#include <cstring>

using namespace std;

//Normal Constructor
Dependant::Dependant() {

}

//Overloaded Constructor
Dependant::Dependant(const char pDependantID[], const char pDependantName[], const
char pEmployeeID[]) {
	strcpy_s(dependantID, pDependantID);
	strcpy_s(dependantName, pDependantName);
	strcpy_s(employeeID, pEmployeeID);
}

//destructor
Dependant::~Dependant() {

}

//methods

void Dependant::getDependantDetails() {

}
void Dependant::storeDependantDetails() {

}

#include "Employee.h"
#include <iostream>
#include <cstring>

using namespace std;

//Normal Constructor
Employee::Employee() {

}
```

```cpp
//Overloaded Constructor
Employee::Employee(const char pEmpID[], const char pEmpName[], const char
pEmpDpartment[], int pEmpContactNum, const char pJobTitle[]) {
	strcpy_s(empID,pEmpID);
	strcpy_s(empName, pEmpName);
	strcpy_s(empDpartment, pEmpDpartment);
	empContactNum = pEmpContactNum;
	strcpy_s(jobTitle,pJobTitle);
}

//destructor
Employee::~Employee() {

}

//methods
void Employee::readReview() {

}
void Employee::viewProjectDetails() {

}
void Employee::viewDepartmentDetails() {

}
void Employee::sendDependantDetails() {

}
void Employee::viewEmployeeDetails() {

}


#include "Payment.h"
#include <iostream>
#include <cstring>

using namespace std;

//constructors
Payment::Payment() {

}
Payment::Payment(const char pPaymentID[], const char pPaymentType[], const char
pPaymentDate[]) {
	strcpy_s(paymentID,pPaymentID);
	strcpy_s(paymentType,pPaymentType);
	strcpy_s(paymentDate,pPaymentDate);
}

//destructor
```

```
Payment::~Payment() {

}

//methods in the class
void Payment::validatePayment() {

}
void Payment::storePaymentDetails() {

}
void Payment::createReceipt() {

}
void Payment::sendReceipt() {

	}

#include "Project.h"
#include <iostream>
#include <cstring>

using namespace std;
//constructors
Project::Project() {

}
Project::Project(const char pProjectID[], const char pProjectName[], const char
pProjectLocation[], const char pProjectStartDate[], const char pCustomerName[]) {
	strcpy_s(projectID, pProjectID);
	strcpy_s(projectName, pProjectName);
	strcpy_s(projectLocation, pProjectLocation);
	strcpy_s(projectStartDate, pProjectStartDate);
	strcpy_s(customerName, pCustomerName);
}
//destructor
Project::~Project() {

}

//methods
void Project::setProjectDetails() {

}
void Project::updateTheProjectDetails() {

}
void Project::addNewProjectDetails() {

}
void Project::deleteProjectDetails() {
```

```
}
void Project::getContractDetails() {

	}

#include "Rating.h"
#include <iostream>
#include <cstring>

using namespace std;

//constructors
Rating::Rating() {

}
Rating::Rating(const char pRatingID[], const char pRatingType[], const char
pRatingDate[]) {
	strcpy_s(ratingID, pRatingID);
	strcpy_s(ratingType, pRatingType);
	strcpy_s(ratingDate, pRatingDate);
}

//destructor
Rating::~Rating() {

}

//methods in the class
void Rating::storeRatingDetails() {

}
void Rating::showRating() {

}
void Rating::deleteRating() {

}
void Rating::updateRating() {

	}

#include "Supervisor.h"
#include <iostream>
#include <cstring>

using namespace std;

//constructor
Supervisor::Supervisor(){
}
```

```
Supervisor::Supervisor(const char pSupervisorID[], const char pSupervisorName[], int
pSupervisorContactNum, const char pSupervisorDepartment[]) {
	strcpy_s(supervisorID,pSupervisorID);
	strcpy_s(supervisorName,pSupervisorName);
	supervisorContactNum = pSupervisorContactNum;
	strcpy_s(supervisorDepartment,pSupervisorDepartment);
}

//destructor
Supervisor::~Supervisor() {

}

//methods in the class
void Supervisor::getEmployeeDetails() {

}
void Supervisor::superviseEmployee() {

}
void Supervisor::distributeTask() {

	}
```

```
#include "Supplier.h"
#include <iostream>
#include <cstring>

using namespace std;

//constructors
Supplier::Supplier() {

}
Supplier::Supplier(const char pSupplierID[], const char pSupplierName[], int
pSupplierContactNum) {
	strcpy_s(supplierID,pSupplierID);
	strcpy_s(supplierName,pSupplierName);
	supplierContactNum = pSupplierContactNum;
}

//destructor
Supplier::~Supplier() {

}

//methods in the class
void Supplier::updateProductDetails() {

}
void Supplier::addProductDetails() {
```

```
}
void Supplier::deleteProductDetails() {

}
void Supplier::receiveOrderDetails() {

	}

#include <iostream>
#include <cstring>

//calling class header files
#include "Admin.h"
#include "ConstructionMaterials.h"
#include "Contract.h"
#include "Customer.h"
#include "Department.h"
#include "Dependant.h"
#include "Employee.h"
#include "Payment.h"
#include "Project.h"
#include "Rating.h"
#include "Supervisor.h"
#include "Supplier.h"

using namespace std;

//main program
int main() {
	//Object for Admin Class
	Admin admn1("ADM001", "joe joena", 0712345671,"Database");
	admn1.answerFAQ();
	admn1.receivePaymentDetails();
	admn1.updateContractDetails();
	admn1.updateProjectDetails();

	//Object for Construction Material Class
	ConstructionMaterials conMat1("PLK001","Lanwa Steel",120000.99);
	conMat1.addNewConstructionMaterialsDetails();
	conMat1.deleteConstructionMaterialsDetails();
	conMat1.setConstructionMaterialsDetails();
	conMat1.updateConstructionMaterialsDetails();

	//Object for Contract Class
	Contract contrct1("CON001","2/2/2013");
	contrct1.checkPayment();
	contrct1.sendContractDetails();
	contrct1.specifyPaymentMethod();
	contrct1.updateContractDetails();
```

```
        //Object for Customer Class
        Customer cust1("Cid001","john richard", "CON001", 0123456712);
        cust1.contactCompany();
        cust1.editUserProfile();
        cust1.giveFeedback();
        cust1.viewBlogs();

        //Objects for Department Class
        Department dept1("DID001", "Engineering", "New Kandy Road");
        dept1.addNewEmployeeDetails();
        dept1.deleteEmployeeDetails();
        dept1.storeDepartmentDetails();
        dept1.storeEmployeeDetails();
        dept1.updateEmployeeDetails();

        //Objects for Dependant Class
        Dependant dpnd1("DPNN001","Maththew silva","Cid001");
        dpnd1.getDependantDetails();
        dpnd1.storeDependantDetails();

        //Object for Employee Class
        Employee emp1("CSK001","John silva","Engineering",0123456712,"Supervisor");
        emp1.readReview();
        emp1.sendDependantDetails();
        emp1.viewDepartmentDetails();
        emp1.viewEmployeeDetails();
        emp1.viewProjectDetails();

        //Object for Payment Class
        Payment pymnt1("PMY001","Visa","1/12/2012");
        pymnt1.createReceipt();
        pymnt1.sendReceipt();
        pymnt1.storePaymentDetails();
        pymnt1.validatePayment();

        //Object for Project Class
        Project proj1("PROJ001", "Siedal Towers", "New Kandy Street", "2/2/2013",
"john richard");
        proj1.addNewProjectDetails();
        proj1.deleteProjectDetails();
        proj1.getContractDetails();
        proj1.setProjectDetails();
        proj1.updateTheProjectDetails();

        //Object for Rating Class
        Rating rate1("RT001","star","2/2/2013");
        rate1.deleteRating();
        rate1.showRating();
        rate1.storeRatingDetails();
        rate1.updateRating();
```

```
//Object for Supervisor Class
Supervisor supvsr1("SUP001","John michaels",0122267721,"Engineering");
supvsr1.distributeTask();
supvsr1.getEmployeeDetails();
supvsr1.superviseEmployee();

//Object for Supplier Class
Supplier supp1("SPP001","John Super Steel",0623122223);
supp1.addProductDetails();
supp1.deleteProductDetails();
supp1.receiveOrderDetails();
supp1.updateProductDetails();


}
```