Topic : Construction Management System

Group no : MLB_02.02_04

Campus : Malabe

Submission Date : 19/05/2022

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**
IT21220760	Dias D.D.K.S.	0775986635
IT21218880	Dayan W.T.	0713745599
IT21219566	Thathsara S.M.K.	0776548285
IT21222436	Perera M.A.E.M.	0758683750
IT21222818	Dissanayaka R.K.D.	0717783332

Table of Contents

System Requirements

- Guests can register for new accounts by entering their details. They can only be registered as customers, suppliers, and investors. After creating an account, they can login to the system whenever they want.
- Suppliers and investors can view company strategies and procedures while all three parties can send feedbacks to the company.
- Customers can view project plans and designs while investors can view financial reports.
- Employees can only be added by the project administrator. Also, admin has the authority to edit or remove any other account including the employee accounts.
- QA engineers can add disciplinary actions, update project audit checklist, and generate quality audit reports.
- Commercial managers can publish business opportunities, update company strategies and procedures, and generate financial reports.
- Site supervisors can report on-site issues, manage inventory records, and generate inventory reports.
- Safety engineers can add warnings about potential hazards, generate safety evaluation reports, and recommend safety improvements.
- Civil engineers can add project plans & designs, publish implementation guidelines, and review Suppliers.
- Admin can add new projects and modify all the attributes related to a project. Also, admin can add equipment and vehicle details to the inventory while calculating the rental fees for them. If something is wrong about the system, admin can run diagnostics and generate a security report until the technicians get to repair the system.
- All the employees have access to third-party integrations and project administrator can view all the project reports.

Identified Classes

Registered Users
Customers
Suppliers
Investors
Company strategies
Feedbacks
Employees
Reports
QA Engineer
Commercial manager
Safety engineer
Site supervisor
Civil engineer
disciplinary actions
project audit checklist
quality audit reports
business opportunities
financial reports
on-site issues
inventory
warnings
safety evaluation reports
safety improvements
administrator
project plans & designs
guidelines
projects
equipment
vehicle
rental fees
security report
third-party apps

CRC Cards

Registered Users	
Responsibilities	Collaborations
Store account details	
Send feedbacks	Feedbacks
Edit feedbacks	Feedbacks
Edit account details	

Customers	
Responsibilities	Collaborations
Login to the system	Registered user
View project plans and designs	Project plans

Suppliers	
Responsibilities	Collaborations
Login to the system	Registered user
View company strategies and procedures	Company strategies

Investors	
Responsibilities	Collaborations
Login to the system	Registered user
View financial reports	Financial reports

Company strategies	
Responsibilities	Collaborations
Store strategic data	
Display strategic data	
Display report data	Reports

Feedbacks	
Responsibilities	Collaborations
Store feedbacks Display feedbacks	

Employees	
Responsibilities	Collaborations
Store account details Edit account details Connect with third-party apps	 Third-party integrations

Reports	
Responsibilities	Collaborations
Store report meta data Display meta data	

QA Engineer	
Responsibilities	Collaborations
Login to the system Add disciplinary actions Update project audit checklist Generate quality audit reports.	Employees Disciplinary actions Audit checklist QA reports

Commercial manager	
Responsibilities	Collaborations
Login to the system Publish business opportunities Update company strategies and procedures Generate financial reports	Employees Opportunities Company strategies Financial reports

Safety engineer	
Responsibilities	Collaborations
Login to the system	Employees
Add warnings about potential hazards	Warnings
Generate safety evaluation reports	Safety reports
Recommend safety improvements	Safety improvements

Site supervisors	
Responsibilities	Collaborations
Login to the system	Employees
Report on-site issues,	Issues
Manage inventory records	Inventory
Generate inventory reports	Inventory

Civil engineer	
Responsibilities	Collaborations
Login to the system	Employees
Add project plans & designs	Project plans
Publish implementation guidelines	Guidelines
Review suppliers	Suppliers

Disciplinary actions	
Responsibilities	Collaborations
Store disciplinary actions	
Display disciplinary actions	
Display report data	Reports

Audit checklist	
Responsibilities	Collaborations
Store Audit checklist items	
Display Audit checklist	

QA reports	
Responsibilities	Collaborations
Store quality audit details Display quality audit details Display report data	 Reports

Opportunities	
Responsibilities	Collaborations
Store business opportunities Display details Display availability Display report data	 Reports

Financial reports	
Responsibilities	Collaborations
Store financial data Display financial data Display report data	 Reports

Issues	
Responsibilities	Collaborations
Store issue data Display issue status	

Inventory	
Responsibilities	Collaborations
Store inventory data Generate inventory reports	

Warnings	
Responsibilities	Collaborations
Store warning data Display warning details	

Safety reports	
Responsibilities	Collaborations
Store safety report data Display safety report details Display report data	 Reports

Safety improvements	
Responsibilities	Collaborations
Store safety improvement data Display safety improvement details	

Administrator	
Responsibilities	Collaborations
Login to the system	
Edit account	Registered users, Employees
Remove account	Registered users, Employees
Add project	Projects
Edit projects	Projects
Add equipment	Equipment, Inventory
Add vehicle	Vehicle, Inventory
Calculate rental fees	Rental fees, Inventory
Generate security reports	Security reports
View reports	Reports

Project plans	
Responsibilities	Collaborations
Store project plan data Display project plan details	

Guidelines	
Responsibilities	Collaborations
Store guidelines data Show equipment guidelines data Show vehicle guidelines data	 Equipment Vehicle

Projects	
Responsibilities	Collaborations
Store project details Show project status Show assigned employees Show customer details	 Employees Customers

Equipment	
Responsibilities	Collaborations
Store equipment data Show rental fees Show equipment data	 Rental fees

Vehicle	
Responsibilities	Collaborations
Store vehicle data Show rental fees Show vehicle data	 Rental fees

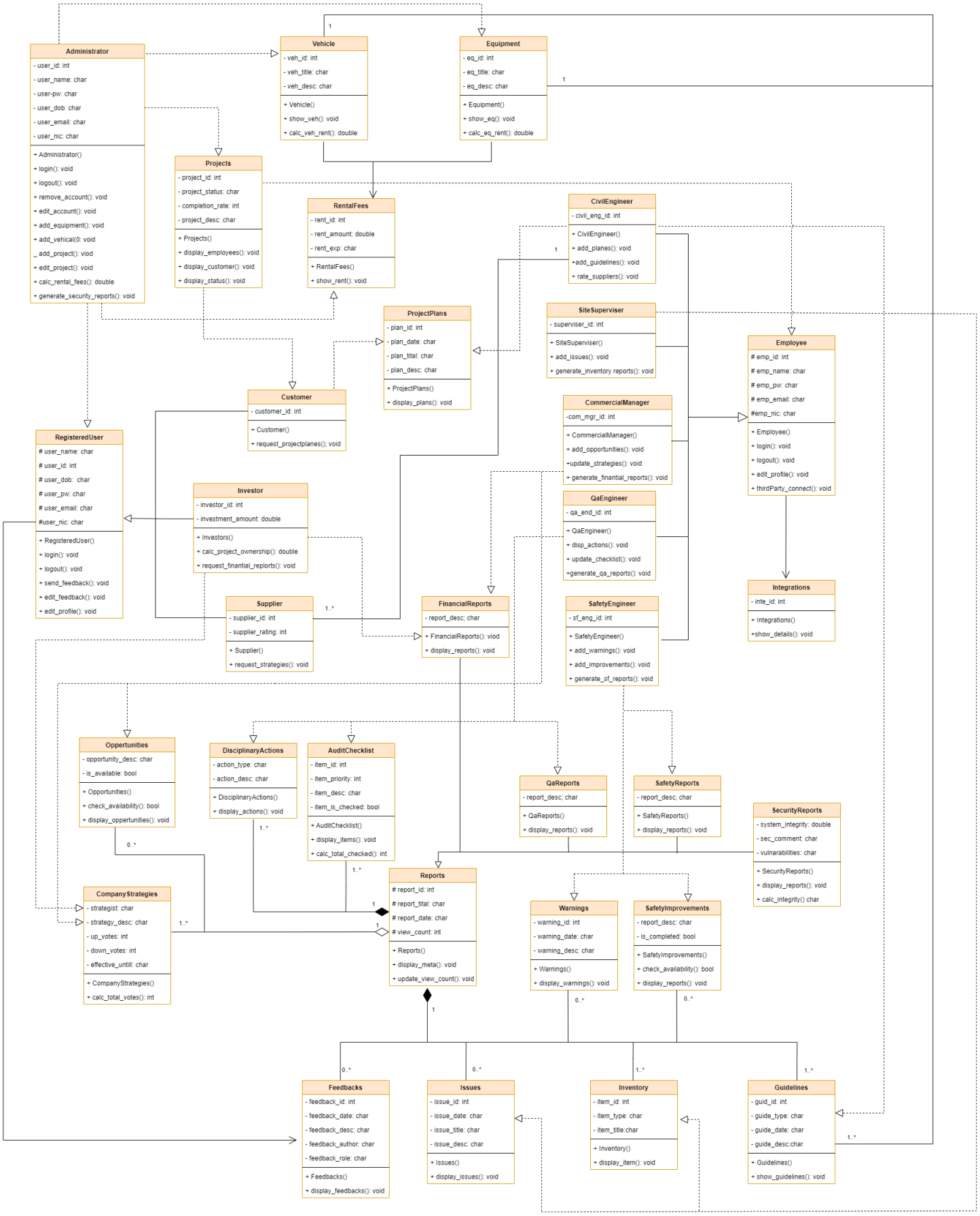
Rental fees	
Responsibilities	Collaborations
Store rental fees	

Security reports	
Responsibilities	Collaborations
Store security data Show system integrity status Show report data	 Reports

Third-part integrations	
Responsibilities	Collaborations
Store integration data Connect with employee accounts	 Employees

Class Diagram

[Original Image](#) (weblink)



Class Header Files

Administrator.h

```
#include "RegisteredUser.h"
#include "Equipment.h"
#include "Vehicle.h"
#include "Projects.h"
#include "RentalFees.h"
#include "SecurityReports.h"


class Administrator {
  private:
    int user_id;
    char name[20];
    char user_pw[8];
    char user_dob[10];
    char user_email[20];
    char user_nic[11];

  public:
    Administrator();
    void login();
    void logout();
    void remove_account(RegisteredUser* acc);
    void edit_account(RegisteredUser* acc);
    void add_equipment(Equipment* eq);
    void add_vehicle(Vehicle* veh);
    void add_project(Projects* prj);
    void edit_project(Projects* prj);
    double calc_rental_fees(RentalFees* fees);
    void generate_security_reports(SecurityReports* report);
    ~Administrator();
};
```

AuditChecklist.h

```
class AuditChecklist {
  private:
    int item_id;
    int item_priority;
    char item_desc[200];
    bool item_is_checked;

  public:
    AuditChecklist();
    AuditChecklist(int id, int priority, const char desc[], bool check);
    void display_items();
    int calc_total_checked();
    ~AuditChecklist();
};
```

CivilEngineer.h

```
#include "Supplier.h"
#include "ProjectPlans.h"
#include "Guidelines.h"
#include "Employee.h"


#define SIZE 5

class CivilEngineer : public Employee {
  private:
    int civil_eng_id;
    Supplier* sup[SIZE];

  public:
    CivilEngineer();
    CivilEngineer(int id);
    void add_plans(ProjectPlans* plan, int id);
    void add_guidelines(Guidelines* guide, int id);
    ~CivilEngineer();
};
```

CommercialManager.h

```
#include "Employee.h"
#include "Opportunities.h"
#include "FinancialReports.h"
#include "CompanyStrategies.h"


class CommercialManager : public Employee {
  private:
    int com_mgr_id;

  public:
    CommercialManager();
    CommercialManager(int id);
    void add_opportunities(Opportunities* opp, int id);
    void update_strategies(CompanyStrategies* strt);
    void generate_financial_reports(FinancialReports* report, int id);
    ~CommercialManager();
};
```

CompanyStrategies.h

```
class CompanyStrategies {
  private:
    char strategist[20];
    char strategy_desc[200];
    int up_votes;
    int down_votes;
    char effective_until[10];

  public:
    CompanyStrategies();
    CompanyStrategies(const char strategist[], const char desc[], int up, int down, const char until[]);
    int calc_total_votes();
    ~CompanyStrategies();
};
```

Customer.h

```
#include "RegisteredUser.h"
#include "ProjectPlans.h"

class Customer : public RegisteredUser {
  private:
    int customer_id;

  public:
    Customer();
    Customer(int id);
    void request_projectplanes(ProjectPlans* prpl, int id);
    ~Customer();
};
```

DisciplinaryActions.h

```
class DisciplinaryActions {
 private:
   char action_type[10];
   char action_desc[200];

 public:
   DisciplinaryActions();
   DisciplinaryActions(const char type[], const char desc[]);
   void display_actions();
   ~DisciplinaryActions();
}
```

Employee.h

```
#include "Integrations.h"

class Employee {
 protected:
   int emp_id;
   char emp_name[20];
   char emp_pw[8];
   char emp_email[20];
   char emp_nic[11];
   Integrations* intgr;

 public:
   Employee();
   Employee(int id, const char name[], const char pw[], const char email[], const char nic[]);
   void login();
   void logout();
   void edit_profile(const char name[], const char pw[], const char email[], const char nic[]);
   void thirdParty_connect();
   ~Employee();
};
```

Equipment.h

```
#include "Guidelines.h"
#include "RentalFees.h"

#define SIZE 5

class Equipment {
  private:
    int eq_id;
    char eq_title[20];
    char eq_desc[200];
    Guidelines* guide[SIZE];
    RentalFees* fee;

  public:
    Equipment();
    Equipment(int id, const char title[], const char desc[]);
    void show_eq();
    double calc_eq_rent();
    ~Equipment();
}
```

Feedbacks.h

```
class Feedbacks {
  private:
    int feedback_id;
    char feedback_date[10];
    char feedback_desc[200];
    char feedback_author[20];
    char feedback_role[10];

  public:
    Feedbacks();
    Feedbacks(int id, const char date[], const char desc[], const char author[], const char role[]);
    void display_feedbacks();
    ~Feedbacks();
};
```

FinancialReports.h

```
#include "Reports.h"

class FinancialReports : public Reports {
        private:
                char report_desc[200];
        public:
                FinancialReports();
                FinancialReports(const char desc[]);
                void display_reports();
                ~FinancialReports();
};
```

Guidelines.h

```
#include "Equipment.h"
#include "Vehicle.h"


class Guidelines {
  private:
    int guide_id;
    char guide_type[20];
    char guide_date[10];
    char guide_desc[200];
    Equipment* eq;
    Vehicle* veh;

  public:
    Guidelines();
    Guidelines(int id, const char type[], const char date[], const char desc[]);
    void show_guidelines();
    ~Guidelines();
};
```

Integrations.h

```
class Integrations{
        private:
                int inte_id;

        public:
                Integrations();
                Integrations(int id);
                void show_details();
                ~Integrations();
};
```

Inventory.h

```
class Inventory{
        private:
                int item_id;
                char item_type[20];
                char item_title[20];

        public:
                Inventory();
                Inventory(int id, const char type[],const char title[]);
                void display_item();
   ~Inventory();

};
```

Investor.h

```
#include "RegisteredUser.h"
#include "CompanyStrategies.h"
#include "FinancialReports.h"


class Investor : public RegisteredUser {
  private:
    int investor_id;
    double investment_amount;

  public:
    Investor();
    Investor(int id);
    double calc_project_ownership(CompanyStrategies* strt);
    void request_financial_reports(FinancialReports* reprt);
    ~Investor();
};
```

Issues.h

```
class Issues{
        private:
                int issue_id;
                char issue_date[20];
                char issue_title[20];
                char issue_desc[200];

        public:
                Issues();
   Issues(int id, const char date[], const char title[], const char desc[]);
                void display_issues();
   ~Issues();
};
```

Opportunities.h

```
class Opportunities {
  private:
    char opportunities_desc[200];
    bool is_available;

  public:
    Opportunities();
    bool check_availability();
    void display_opportunities();
    ~Opportunities();
};
```

ProjectPlans.h

```
class ProjectPlans {
 private:
   int plan_id;
   char plan_date[10];
   char plan_title[20];
   char plan_desc[200];

 public:
   ProjectPlans();
   ProjectPlans(int id, const char date[], const char title[], const char desc[]);
   void display_plans();
   ~ProjectPlans();
};
```

Projects.h

```
#include "Employee.h"
#include "Customer.h"


class Projects {
  private:
    int project_id;
    char project_status[10];
    int completion_rate;
    char project_desc[200];

  public:
    Projects();
    Projects(int id, const char status[], int rate, const char desc[]);
    void display_employees(Employee* emp);
    void display_customer(Customer* cus);
    void display_status();
    ~Projects();
};
```

QaEngineer.h

```
#include "Employee.h"
#include "DisciplinaryActions.h"
#include "AuditChecklist.h"
#include "QaReports.h"


class QaEngineer : public Employee {
  private:
    int qa_end_id;

  public:
    QaEngineer();
    QaEngineer(int id);
    void disp_actions(DisciplinaryActions* action, int id);
    void update_checklist(AuditChecklist* item, int id);
    void generate_qa_reports(QaReports* report, int id);
    ~QaEngineer();
};
```

QaReports.h

```
#include "Reports.h"

using namespace std;

class QaReports : public Reports {
        private:
                char report_desc[200];
        public:
                QaReports();
                QaReports(const char desc[]);
                void display_reports();
                ~QaReports();
};
```

RegisteredUser.h

```
#include "Feedbacks.h"

class RegisteredUser {
  protected:
    char user_name[20];
    int user_id;
    char user_dob[10];
    char user_pw[8];
    char user_email[20];
    char user_nic[11];
    Feedbacks* feedbk;

  public:
    RegisteredUser();
    void login();
    void logout();
    void send_feedback(int id, const char date[], const char desc[], const char author, const char
role[]);
    void edit_feedback(int id, const char date[], const char desc[], const char role[]);
    void edit_profile(const char name[], const char pw[], const char email[], const char nic[]);
    ~RegisteredUser();
};
```

RentalFees.h

```
class RentalFees{
  private:
    int rent_id;
    double rent_amount;
    char rent_exp[10];

  public:
    RentalFees();
    RentalFees(int id, double amount, const char exp[]);
    void show_rent();
    ~RentalFees();
};
```

Reports.h

```
#include "Warnings.h"
#include "SafetyImprovements.h"
#include "Feedbacks.h"
#include "Issues.h"
#include "Inventory.h"
#include "Guidelines.h"
#include "AuditChecklist.h"
#include "DisciplinaryActions.h"
#include "Opportunities.h"
#include "CompanyStrategies.h"

# define SIZE 5

class Reports {
        protected:
                int report_id;
                char report_title[20];
                char report_date[10];
                int view_count;

    Warnings* warn[SIZE];
    SafetyImprovements* si[SIZE];
    Feedbacks* feed[SIZE];
    Issues* issue[SIZE];
    Inventory* item[SIZE];
    Guidelines* guide[SIZE];
    AuditChecklist* audit[SIZE];
    DisciplinaryActions* actions[SIZE];
    Opportunities* opp[SIZE];
    CompanyStrategies*  cs[SIZE];

        public:
                Reports();
                Reports(int id, const char title[], const char date[], int count);
```

```
        void display_meta();
        void update_view_count();
        ~Reports();
};
```

SafetyEngineer.h

```
#include "Employee.h"
#include "Warnings.h"
#include "SafetyImprovements.h"
#include "SafetyReports.h"

class SafetyEngineer : public Employee {
  private:
    int sf_eng_id;
  public:
    SafetyEngineer();
    SafetyEngineer(int id);
    void add_warnings(Warnings* warn, int id);
    void add_improvements(SafetyImprovements* impr, int id);
    void generate_sf_reports(SafetyReports* report, int id);
    ~SafetyEngineer();
};
```

SafetyImprovements.h

```
class SafetyImprovements{
        private:
                char report_desc[200];
                bool is_completed;

        public:
                SafetyImprovements();
    SafetyImprovements(const char desc[], bool completed);
                bool check_availability();
                void display_reports();
    ~SafetyImprovements();

};
```

SafetyReports.h

```
class SafetyReports : public Reports {
        private:
                char report_desc[200];
        public:
                SafetyReports();
                SafetyReports(const char desc[]);
                void display_reports();
                ~SafetyReports();
}
```

SecurityReports.h

```
#include "Reports.h"

class SecurityReports : public Reports {
        private:
                double system_integrity;
                char sec_comment[200];
                char vulnarabilities[200];
        public:
                SecurityReports();
                SecurityReports(double integrity, const char comment[], const char vuln[]);
                void display_reports();
                double calc_integrity(double integrity);
                ~SecurityReports();
};
```

SiteSupervisor.h

```
#include "Employee.h"
#include "Issues.h"
#include "Inventory.h"


class SiteSupervisor : public Employee {
  private:
    int supervisor_id;

  public:
    SiteSupervisor();
    SiteSupervisor(int id);
    void add_issues(Issues* issue, int id);
    void generate_inventory_reports(Inventory* report, int id);
    ~SiteSupervisor();
};
```

Supplier.h

```
#include "RegisteredUser.h"
#include "CompanyStrategies.h"

class Supplier : public RegisteredUser {
   private:
     int supplier_id;
     int supplier_rating;

   public:
     Supplier();
     Supplier(int id);
     void request_strategies(CompanyStrategies* strt);
     ~Supplier();
};
```

Vehicle.h

```
#include "Guidelines.h"
#include "RentalFees.h"

#define SIZE 5

class Vehicle {
  private:
    int veh_id;
    char veh_title[20];
    char veh_desc[200];
    Guidelines* guide[SIZE];
    RentalFees* fee;

  public:
    Vehicle();
    Vehicle(int id, const char title[], const char desc[]);
    void show_veh();
    double calc_veh_rent();
    ~Vehicle();
};
```

Warnings.h

```
class Warnings{
        private:
                int warning_id;
                char warning_date[20];
                char warning_desc[200];

        public:
                Warnings();
                Warnings(int id, const char date[],const char desc[]);
                void display_warnings();
   ~Warnings();

};
```

Class CPP Files

Admininstrator.cpp

```
#include <cstring>
#include "Administrator.h"

Administrator::Administrator() {
  user_id = 0;
  strcpy(name, "");
  strcpy(user_pw, "");
  strcpy(user_email, "");
  strcpy(user_nic, "");
  strcpy(user_dob, "");
}
void Administrator::login(){}
void Administrator::logout(){}
void Administrator::remove_account(RegisteredUser* acc){}
void Administrator::edit_account(RegisteredUser* acc){}
void Administrator::add_equipment(Equipment* eq){}
void Administrator::add_vehicle(Vehicle* veh){}
void Administrator::add_project(Projects* prj){}
void Administrator::edit_project(Projects* prj){}
double calc_rental_fees(RentalFees* fees){}
void Administrator::generate_security_reports(SecurityReports* report){}
Administrator::~Administrator(){}
```

AuditChecklist.cpp

```
#include "AuditChecklist.h"
#include <cstring>


AuditChecklist::AuditChecklist() {
  item_id = 0;
  item_priority = 0;
  item_is_checked = false;
  strcpy(item_desc, "");
}
AuditChecklist::AuditChecklist(int id, int priority, const char desc[], bool check) {
  item_id = id;
  item_priority = priority;
  item_is_checked = check;
  strcpy(item_desc, desc);
}
void AuditChecklist::display_items(){}
int AuditChecklist::calc_total_checked(){}
AuditChecklist::~AuditChecklist(){}
```

CivilEngineer.cpp

```
#include "CivilEngineer.h"


CivilEngineer::CivilEngineer() {
  civil_eng_id = 0;
}
CivilEngineer::CivilEngineer(int id) {
  civil_eng_id = id;
}
void CivilEngineer::add_plans(ProjectPlans* plan, int id){}
void CivilEngineer::add_guidelines(Guidelines* guide, int id){}
CivilEngineer::~CivilEngineer(){}
```

CommercialManager.cpp

```
#include "CommercialManager.h"

CommercialManager::CommercialManager() {
  com_mgr_id = 0;
}
CommercialManager::CommercialManager(int id) {
  com_mgr_id = id;
}
void CommercialManager::add_opportunities(Opportunities* opp, int id){}
void CommercialManager::update_strategies(CompanyStrategies* strt){}
void CommercialManager::generate_financial_reports(FinancialReports* report, int id){}
CommercialManager::~CommercialManager(){}
```

CompanyStrategies.cpp

```
#include "CompanyStrategies.h"
#include <cstring>


CompanyStrategies::CompanyStrategies() {
  strcpy(strategist, "");
  strcpy(strategy_desc, "");
  strcpy(effective_until, "");
  up_votes = 0;
  down_votes = 0;
}

CompanyStrategies::CompanyStrategies(const char strategist[], const char desc[], int up, int down,
const char until[]) {
  strcpy(strategy_desc, desc);
  strcpy(effective_until, until);
  up_votes = up;
```

```
  down_votes = down;
}
int CompanyStrategies::calc_total_votes(){}
CompanyStrategies::~CompanyStrategies(){}
```

Customer.cpp

```
#include "Customer.h"

Customer::Customer() {
  customer_id = 0;
}
Customer::Customer(int id) {
  customer_id = id;
}
void Customer::request_projectplanes(ProjectPlans* prpl, int id){}
Customer::~Customer() {}
```

DisciplinaryActions.cpp

```
#include "DisciplinaryActions.h"
#include <cstring>

DisciplinaryActions::DisciplinaryActions(){
  strcpy(action_type, "");
  strcpy(action_desc, "");
}
DisciplinaryActions::DisciplinaryActions(const char type[], const char desc[]){
  strcpy(action_type, type);
  strcpy(action_desc, desc);
}
void DisciplinaryActions::display_actions(){}
DisciplinaryActions::~DisciplinaryActions(){}
```

Employee.cpp

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

Employee:: Employee() {
  emp_id = 0;
  strcpy(emp_name, "");
  strcpy(emp_pw, "");
  strcpy(emp_email, "");
  strcpy(emp_nic, "");
}
Employee:: Employee(int id, const char name[], const char pw[], const char email[], const char nic[])
{
```

```
  emp_id = id;
  strcpy(emp_name, name);
  strcpy(emp_pw, pw);
  strcpy(emp_email, email);
  strcpy(emp_nic, nic);
}
void Employee::login(){}
void Employee::logout(){}
void Employee::edit_profile(const char name[], const char pw[], const char email[], const char
nic[]){}
void Employee::thirdParty_connect(){}
Employee:: ~Employee(){}
```

Equipment.cpp

```
#include "Equipment.h"
#include <cstring>


Equipment::Equipment() {
  eq_id = 0;
  strcpy(eq_title, "");
  strcpy(eq_desc, "");
}
Equipment::Equipment(int id, const char title[], const char desc[]) {
  eq_id = id;
  strcpy(eq_title, title);
  strcpy(eq_desc, desc);
}
void Equipment::show_eq(){}
double Equipment::calc_eq_rent(){}
Equipment::~Equipment(){}
```

Feedbacks.cpp

```
#include "Feedbacks.h"
#include <cstring>


Feedbacks::Feedbacks() {
  feedback_id = 0;
  strcpy(feedback_date, "");
  strcpy(feedback_desc, "");
  strcpy(feedback_author, "");
  strcpy(feedback_role, "");
}
Feedbacks::Feedbacks(int id, const char date[], const char desc[], const char author[], const char
role[]) {
  feedback_id = id;
```

```
 strcpy(feedback_date, date);
 strcpy(feedback_desc, desc);
 strcpy(feedback_author, author);
 strcpy(feedback_role, role);
}
void Feedbacks::display_feedbacks(){}
Feedbacks::~Feedbacks(){}
```

FinancialReports.cpp

```
#include "FinancialReports.h"
#include <cstring>


FinancialReports :: FinancialReports() {
        strcpy(report_desc, "");
}

FinancialReports :: FinancialReports(const char desc[]) {
        strcpy(report_desc, desc);
}

void FinancialReports :: display_reports() {}

FinancialReports :: ~FinancialReports() {}
```

Guidelines.cpp

```
#include "Guidelines.h"
#include <cstring>


Guidelines::Guidelines(){
 guide_id = 0;
 strcpy(guide_date, "");
 strcpy(guide_desc, "");
 strcpy(guide_type, "");
}

Guidelines::Guidelines(int id, const char type[], const char date[], const char desc[]){
 guide_id = id;
 strcpy(guide_date, date);
 strcpy(guide_desc, desc);
 strcpy(guide_type, type);
}
void Guidelines::show_guidelines(){}
Guidelines::~Guidelines(){}
```

```
#include "Integrations.h"

Integrations::Integrations() {
  inte_id = 0;
}
Integrations::Integrations(int id) {
  inte_id = id;
}
void Integrations::show_details(){}
Integrations::~Integrations() {}
```

Inventory.cpp

```
#include "Inventory.h"
#include <cstring>


Inventory::Inventory() {
  item_id = 0;
  strcpy(item_type,"");
  strcpy(item_title,"");
}
Inventory::Inventory(int id, const char type[],const char title[]) {
  item_id = id;
  strcpy(item_type,type);
  strcpy(item_title, title);
}
void Inventory::display_item(){}
Inventory::~Inventory(){}
```

Investor.cpp

```
#include "Investor.h"

Investor::Investor() {
  investor_id = 0;
  investment_amount = 0.00;
}
Investor::Investor(int id) {
  investor_id = id;
}
double Investor::calc_project_ownership(CompanyStrategies* strt){}
void Investor::request_financial_reports(FinancialReports* reprt){}
Investor::~Investor() {}
```

Issues.cpp

```
#include "Issues.h"
#include <cstring>

Issues::Issues() {
  issue_id = 0;
  strcpy(issue_date, "");
  strcpy(issue_title, "");
  strcpy(issue_desc, "");
}
Issues::Issues(int id, const char date[], const char title[], const char desc[]) {
  issue_id = id;
  strcpy(issue_date, date);
  strcpy(issue_title, title);
  strcpy(issue_desc, desc);
}
void Issues::display_issues(){}
Issues::~Issues(){}
```

Opportunities.cpp

```
#include "Opportunities.h"
#include <cstring>

Opportunities::Opportunities() {
  strcpy(opportunities_desc, "");
  is_available = false;
}
bool Opportunities::check_availability(){}
void Opportunities::display_opportunities(){}
Opportunities::~Opportunities() {}
```

ProjectPlans.cpp

```
#include "ProjectPlans.h"
#include <cstring>

ProjectPlans::ProjectPlans() {
  plan_id = 0;
  strcpy(plan_date, "");
  strcpy(plan_title, "");
  strcpy(plan_desc, "");
}
ProjectPlans::ProjectPlans(int id, const char date[], const char title[], const char desc[]) {
  plan_id = id;
  strcpy(plan_date, date);
  strcpy(plan_title, title);
  strcpy(plan_desc, desc);
```

```
}
void ProjectPlans::display_plans(){}
ProjectPlans::~ProjectPlans(){}
```

Projects.cpp

```
#include "Projects.h"
#include <cstring>

Projects::Projects() {
  project_id = 0;
  strcpy(project_status, "Pending");
  completion_rate = 0;
  strcpy(project_desc, "lorem ipsum");
}
Projects::Projects(int id, const char status[], int rate, const char desc[]) {
  project_id = id;
  strcpy(project_status, status);
  completion_rate = rate;
  strcpy(project_desc, desc);
}
void Projects::display_employees(Employee* emp){}
void Projects::display_customer(Customer* cus){}
void Projects::display_status(){}
Projects::~Projects(){}
```

QaEngineer.cpp

```
#include "QaEngineer.h"

QaEngineer::QaEngineer(){
  qa_end_id = 0;
}
QaEngineer::QaEngineer(int id){
  qa_end_id = id;
}
void QaEngineer::disp_actions(DisciplinaryActions* action, int id){}
void QaEngineer::update_checklist(AuditChecklist* item, int id){}
void QaEngineer::generate_qa_reports(QaReports* report, int id){}
QaEngineer::~QaEngineer(){}
```

QaReports.cpp

```
#include "QaReports.h"
#include <cstring>

QaReports :: QaReports() {
	strcpy(report_desc, "");
}

QaReports :: QaReports(const char desc[]) {
	strcpy(report_desc, desc);
}

void QaReports :: display_reports() {}

QaReports :: ~QaReports() {}
```

RegisteredUser.cpp

```
#include "RegisteredUser.h"
#include <cstring>

RegisteredUser::RegisteredUser() {
  user_id = 0;
  strcpy(user_name, "");
  strcpy(user_dob, "");
  strcpy(user_pw, "");
  strcpy(user_email, "");
  strcpy(user_nic, "");
}
void RegisteredUser::login(){}
void RegisteredUser::logout(){}
void RegisteredUser::send_feedback(int id, const char date[], const char desc[], const char author,
const char role[]){}
void RegisteredUser::edit_feedback(int id, const char date[], const char desc[], const char role[]){}
void RegisteredUser::edit_profile(const char name[], const char pw[], const char email[], const char
nic[]){}
RegisteredUser::~RegisteredUser(){}
```

RentalFees.cpp

```
#include "RentalFees.h"
#include <cstring>


RentalFees::RentalFees() {
  rent_id = 0;
  rent_amount = 0.00;
  strcpy(rent_exp, "");
}

RentalFees::RentalFees(int id, double amount, const char exp[]) {
  rent_id = id;
  rent_amount = amount;
  strcpy(rent_exp, exp);
}
void RentalFees::show_rent(){}
RentalFees::~RentalFees(){}
```

Reports.cpp

```
#include "Reports.h"
#include <cstring>

Reports :: Reports() {
        report_id = 0;
        strcpy(report_title, "");
        strcpy(report_date, "");
        view_count = 0;
}

Reports :: Reports(int id, const char title[], const char date[], int count) {
        report_id = id;
        strcpy(report_title, title);
        strcpy(report_date, date);
        view_count = count;
}

void Reports :: display_meta() {}

void Reports :: update_view_count() {}

Reports :: ~Reports() {}
```

SafetyEngineer.cpp

```
#include "SafetyEngineer.h"

SafetyEngineer::SafetyEngineer() {
  sf_eng_id = 0;
}
SafetyEngineer::SafetyEngineer(int id) {
  sf_eng_id = id;
}
void SafetyEngineer::add_warnings(Warnings* warn, int id){}
void SafetyEngineer::add_improvements(SafetyImprovements* impr, int id){}
void SafetyEngineer::generate_sf_reports(SafetyReports* report, int id){}
SafetyEngineer::~SafetyEngineer(){}
```

SafetyImprovements.cpp

```
#include "SafetyImprovements.h"
#include <cstring>

SafetyImprovements::SafetyImprovements() {
  strcpy(report_desc, "");
  is_completed = false;
}

SafetyImprovements::SafetyImprovements(const char desc[], bool completed) {
  strcpy(report_desc, desc);
  is_completed = completed;
}
bool SafetyImprovements::check_availability(){}
void SafetyImprovements::display_reports(){}
SafetyImprovements::~SafetyImprovements(){

}
```

SecurityReports.cpp

```
#include "SecurityReports.h"
#include <cstring>

SecurityReports :: SecurityReports() {
        system_integrity = 0.0;
        strcpy(sec_comment, "");
        strcpy(vulnarabilities, "");
}

SecurityReports :: SecurityReports(double integrity, const char comment[], const char vuln[]) {
        system_integrity = integrity;
        strcpy(sec_comment, comment);
        strcpy(vulnarabilities, vuln);
```

```
}

void SecurityReports :: display_reports() {}

double SecurityReports :: calc_integrity(double integrity) {}

SecurityReports :: ~SecurityReports() {}
```

SiteSupervisor.cpp

```
#include "SiteSupervisor.h"

SiteSupervisor::SiteSupervisor() {
  supervisor_id = 0;
}
SiteSupervisor::SiteSupervisor(int id) {
  supervisor_id = id;
}
void SiteSupervisor::add_issues(Issues* issue, int id){}
void SiteSupervisor::generate_inventory_reports(Inventory* report, int id){}
SiteSupervisor::~SiteSupervisor(){}
```

Supplier.cpp

```
#include "Supplier.h"

Supplier::Supplier() {
  supplier_id = 0;
  supplier_rating = 0;
}
Supplier::Supplier(int id) {
  supplier_id = id;
}
void Supplier::request_strategies(CompanyStrategies* strt){}
Supplier::~Supplier() {}
```

Vehicle.cpp

```
#include "Vehicle.h"
#include <cstring>


Vehicle::Vehicle() {
  veh_id = 0;
  strcpy(veh_title, "");
  strcpy(veh_desc, "");
}
Vehicle::Vehicle(int id, const char title[], const char desc[]) {
  veh_id = id;
  strcpy(veh_title, title);
  strcpy(veh_desc, desc);
}
void Vehicle::show_veh(){}
double Vehicle::calc_veh_rent(){}
Vehicle::~Vehicle(){}
```

Warnings.cpp

```
#include "Warnings.h"
#include <cstring>


Warnings::Warnings() {
  warning_id = 0;
  strcpy(warning_date, "");
  strcpy(warning_desc, "");
}
Warnings::Warnings(int id, const char date[],const char desc[]) {
  warning_id = id;
  strcpy(warning_date, date);
  strcpy(warning_desc, desc);
}
void Warnings::display_warnings(){}
Warnings::~Warnings(){}
```

Main Program

main.cpp

```
#include "Administrator.h"
#include "AuditChecklist.h"
#include "CivilEngineer.h"
#include "CommercialManager.h"
#include "CompanyStrategies.h"
#include "Customer.h"
#include "DisciplinaryActions.h"
#include "Employee.h"
#include "Equipment.h"
#include "Feedbacks.h"
#include "FinancialReports.h"
#include "Guidelines.h"
#include "Integrations.h"
#include "Inventory.h"
#include "Investor.h"
#include "Issues.h"
#include "Projects.h"
#include "Opportunities.h"
#include "ProjectPlans.h"
#include "QaEngineer.h"
#include "QAReports.h"
#include "RegisteredUser.h"
#include "RentalFees.h"
#include "Reports.h"
#include "SafetyEngineer.h"
#include "SafetyImprovements.h"
#include "SafetyReports.h"
#include "SecurityReports.h"
#include "SiteSupervisor.h"
#include "Supplier.h"
#include "Vehicle.h"
#include "Warnings.h"

#include <iostream>
using namespace std;

int main() {

  //------Object Creation----------------

  Administrator* admin = new Administrator();
  AuditChecklist* list = new AuditChecklist();
  CivilEngineer* ce = new CivilEngineer();
  CommercialManager* cm = new CommercialManager();
  CompanyStrategies* cs = new CompanyStrategies();
  Customer* customer= new Customer();
```

```
DisciplinaryActions* da = new DisciplinaryActions();
Employee* emp = new Employee();
Equipment* eq = new Equipment();
Feedbacks* feed = new Feedbacks();
FinancialReports* fr = new FinancialReports();
Guidelines* guide = new Guidelines();
Integrations* inte= new Integrations();
Inventory* item = new Inventory();
Investor* inv = new Investor();
Issues* issue = new Issues();
Projects* prj = new Projects();
Opportunities* opp = new Opportunities();
ProjectPlans* pp = new ProjectPlans();
QaEngineer* qae= new QaEngineer();
QaReports* qar= new QaReports();
RegisteredUser* rUser= new RegisteredUser();
RentalFees* fees = new RentalFees();
Reports* rep = new Reports();
SafetyEngineer* se = new SafetyEngineer();
SafetyImprovements* si = new SafetyImprovements();
SafetyReports* sr = new SafetyReports();
SecurityReports* secR= new SecurityReports();
SiteSupervisor* ss= new SiteSupervisor();
Supplier* sup = new Supplier();
Vehicle* veh = new Vehicle();
Warnings* warn = new Warnings();

//----Deleting dynamic objects--------
delete admin;
delete list;
delete ce;
delete cm;
delete cs;
delete customer;
delete da;
delete emp;
delete eq;
delete feed;
delete fr;
delete guide;
delete inte;
delete item;
delete inv;
delete issue;
delete prj;
delete opp;
delete pp;
delete qae;
delete qar;
delete rUser;
delete fees;
```

```
  delete rep;
  delete se;
  delete si;
  delete sr;
  delete secR;
  delete ss;
  delete sup;
  delete veh;
  delete warn;
}
```