Topic : Vehicle Insurance Management System

Group no : MLB_PR_17

Campus : Malabe

Submission Date: 15/10/2021

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**
IT18027648	L.A.A.E.Senarathna	0718993204
IT18003192	T.M.I.A.Munasinghe	0712927713
IT20225810	K.Thisanth	0778259037

Exercise 1:

System Requirements for Vehicle Insurance Management System

- This system allows all users to look for vehicle insurance.
- Any visitor can sign up to become a member.
- The vehicle insurance management system, all members can access the website by entering their email address and password.
- Once user registered, he/she can view his profile, delete profile, or edit profile details.
- A member can request for a selected vehicle insurance.
- A member can buy a vehicle categorizing in brand, service, payment, secure.
- If a member needs vehicle insurance first, they must fill the request form.
- Once a member filled the form, they can add their own information to the website for insurance company.
- A member can make payments online through our website via using credit cards, online money transfer or EFT.
- Admin also needs to login to the system before do the activities.
- Admin can generate reports such as vehicle insurance reports, cash flow of the company.
- Admin can view requests for insurance policy, other benefits and decide whether it accepts.
- Admin can also generate reports.
- Admin can apply different level of filters on report.
- Admin can track the detailed information of insurance Management.

Nouns/Verbs analysis

- Nouns in Red colour
- Verbs in Blue colour

2)

This system allows all users to look for vehicle insurance.

Any visitor can sign up to become a member.

The vehicle insurance management system, all members can access the website by entering their email address and password.

Once user registered, he/she can view his profile, delete profile, or edit profile details.

A member can request for a selected vehicle insurance.

A member can buy a vehicle categorizing in brand, service, payment, secure.

If a member needs vehicle insurance first, they must fill the request form.

Once a member filled the form, they can add their own information to the website for insurance company.

A member can make payments online through our website via using credit cards, online money transfer or EFT.

Admin also needs to login to the system before do the activities.

Admin can generate reports such as vehicle insurance reports, cash flow of the company.

Admin can view requests for insurance policy, other benefits and decide whether it accepts.

Admin can also generate reports.

Admin can apply different level of filters on report.

Admin can track the detailed information of insurance Management.

Identified Classes using Noun Verb Analysis

Identified Classes:

- User
- Role
- Customer
- Payment
- Permission
- Bill
- Insurance

Nouns

- User
- Vehicle insurance
- Visitor
- Member
- Management system
- Website
- Adders
- Password

- Brand
- Service
- Secure
- Payment
- Form
- Information
- Credit card
- Online transfer
- EFT
- Company
- Admin
- Insurance policy

Reasons for Rejecting Other Nouns

Redundant:

> In an insurance sales system user, visitor, member refers to the same person as "Member".

Outside scope of system

> Website, company, system, user profile, request form, information is outside scope of the insurance management system.

An attribute

- Email
- Password
- Credit card
- EFT
- Money transfer
- Date slots

Exercise 2

CRC Cards for Vehicle Insurance Management System

Role	
Responsibilities	Collaborations
Add roles	
Restock	
Update	

Customer	
Responsibilities	Collaborations
Login to the system (Provide customer information)	
Search insurance	Insurance
View profile	
Delete profile	
Edit profile details	
Request for an insurance	Admin(user)
Apply for a loan	Admin(user)
Make payment	Payment
Request to add insurance	Admin(user)
Buy a vehicle insurance	Insurance

User (Admin)	
Responsibilities	Collaborations
Login to the system	
Accept for the customer request	Customer
Add a customer	Customer
Update insurance policy	Insurance Permission
Generate report	Insurance

Permission	
Responsibilities	Collaborations
Check permission	Customer Insurance Payment

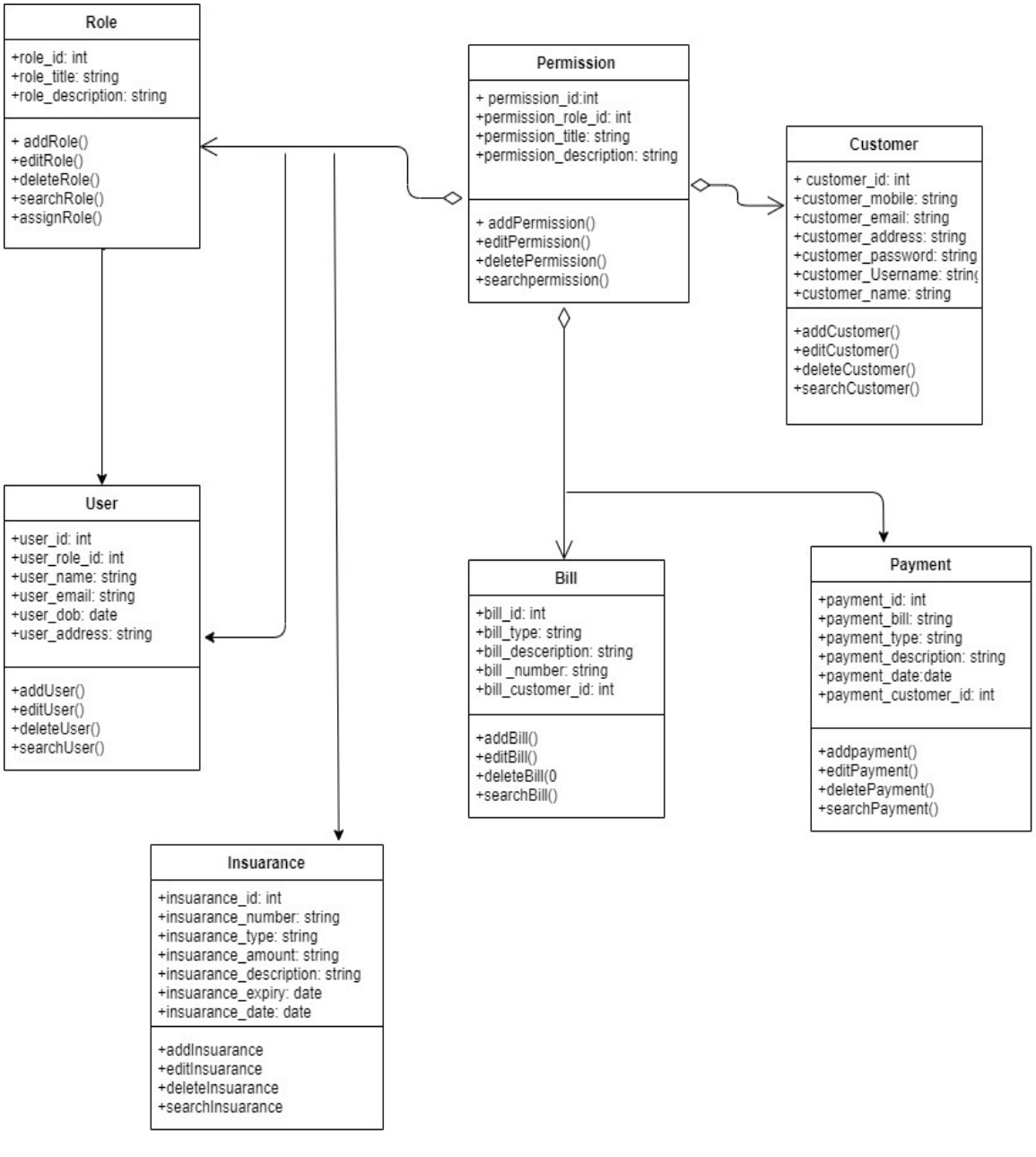
Bill	
Responsibilities	Collaborations
List of restock vehicles	Item (vehicles)
List of previse vehicles	Customer

Payment	
Responsibilities	Collaborations
View payment details	Customer
Validate the payment	
Keep a record of all customer payments	
Calculate and store amount of deposit returned	
Issue receipt	Customer, Bill

Insurance	
Responsibilities	Collaborations
Issue the receipt	Payment
Issue the insurance policy	Permission

Exercise 3

Class Diagram for the Vehicle Insurance Management System



Exercise 4

Coding for the Classes in Class Diagram

```
// Customer Class

class Customer{
    private:
        int Customer_id;
        std ::string Customer_mobile;
        std ::string Customer_email;
        std ::string Customer_address;
        std ::string Customer_password;
        std ::string Customer_Username;
        std ::string Customer_name;
    public:
        void addCustomer();
        void editCustomer();
        void deleteCustomer();
        void searchCustomer();

}
Customer::Customer(){
}
Customer::Customer(int cID,std ::string cMonum,std ::string cEmil,std ::string
 cAdres,std ::string cPwd,std ::string cUname,std::String cName )
{
    Customer_id = cID;
  Customer_mobile = cMonum;
  Customer_email=  cEmil;
  Customer_address = cAdres;
    Customer_password = cPwd;
  Customer_Username = cUname;
  Customer_name=cName;
}

void Customer::addCustomer(){

}

void Customer::editCustomer(){
```

```
}
void Customer::deleteCustomer(){

}
void Customer::searchCustomer(){

}
Customer::~Customer(){}

};
```

```
// Bill Class

  class Bill{
    private:
        int bill_id;
        std ::string bill_type;
        std ::string bill_description;
        std ::string bill_number;
        int bill_customer_id;

    public:
        void addBill();
        void editBill();
        void deleteBill();
        void searchBill();
}
Bill::Bill(){

}
Bill::Bill(int bID,std ::string bType,std ::string bDes,std ::string bNum,std
::string bCid )
{
    bill_id = bID;
  bill_type=   bType;
    bill_description = bDes;
    bill_number= bNum;
  bill_customer_id = bCid;

}

void Bill::addBill(){

}
```

```
void Bill::editBill(){

}
void Bill::deleteBill(){

}
void Bill::searchBill(){

}
Bill::~Bill(){}

};
```

```
// Permission Class

  class Permission{
    private:
        int permission_id;
        int permission_role;
        std ::string permission_title;
        std ::string permission_description;
    public:
        void addpermission();
        void editpermission();
        void deletepermission();
        void searchRequestedpermission();
    }


Permission::permission() {

}
Permission::Permission(int pID,int pRole,std ::string pTit,std ::string pDes )
{
    permission_id = pID;
    permission_role = pRole;
    permission_title = pTit;
    permission_description = pDes;
}
void Permission::addPermission(){

}

void Permission::editpermission(){

}
```

```
void Permission::deletepermission(){

}
void Permission::searchRequestedpermission(){

}
void displayPermission(){

}
permission::~Permission(){

};
```

```
// Role Class

class Role{
    private:
        int role_id;
        std ::string role_title;
        std ::string role_description;
    public:
        void addRole();
        void editRole();
        void deleteRole();
        void searchRole();
        void assignRlole();
}

Role::Role(){

}
Role::Role(int rID,std ::string rTit,std ::string rDes )
{
    role_id = rID;
    role_title = rTit;
    role_description = rDes;
}
void Role::addRole(){

};
```

```
// User Class

  class User{
    private:
        int user_id;
        std ::string user_role;
```

```
        std ::string user_name;
        std ::string user_email;
        std ::string user_dob;
        std ::string user_address;

    public:
        void addUser();
        void editUser();
        void deleteUser();
        void searchUser();

}

User::User(){

}
User::User(int uID,std ::string uRol,std ::string uname,std ::string uemail,st
d ::string udob,std ::string uadres )
{
    user_id = uID;
  user_role=  uRol;
    user_name = uname;
    user_email = uemail;
  user_dob = udob;
  user_address = uadres;
}

void User::addUser(){

}

void User::editUser()(){

}
void User::deleteUser(){

}
void User::searchUser(){

}
User::~User(){}

};


// Insuarance Class

  class Insuarance{
    private:
```

```
        int insuarance_id;
        std ::string insuarance_number;
        std ::string insuarance_type;
        std ::string insuarance_amount;
        std ::string insuarance_description;
        std ::string insuarance_expiry;
        std ::string insuarance_date;
    public:
        void addInsuarance();
        void editInsuarance();
        void deleteInsuarance();
        void searchInsuarance();
}

Insuarance::Insuarance(){

}
Insuarance::Insuarance(int iID,std ::string inum,std ::string itype,std ::stri
ng iAmot,std ::string iDes,std ::string iExp,std::String iDat )
{
    insuarance_id = iID;
  insuarance_number = inum;
  insuarance_type=  itype;
  insuarance_amount = iAmot;
    insuarance_description = iDes;
  insuarance_expiry = iExp;
  insuarance_date=iDat;
}

void Insuarance::addInsuarance(){

}

void Insuarance::editInsuarance(){

}
void Insuarance::deleteInsuarance(){

}
void Insuarance::searchInsuarance(){

}
Insuarance::~Insuarance(){}

};
```

```
// Payment Class
```

```
  class Payment{
    private:
        int payment_id;
        std ::string payment_bill;
        std ::string payment_type;
        std ::string payment_description;
        std ::string payment_date;
        std ::string payment_customer_id;
    public:
        void addPayment();
        void editPayment();
        void deletePayment();
        void searchPayment();
};

Payment::Payment(int yID,std ::string ybil,std ::string ytype,std ::string yde
s,std ::string ydat,std ::string ycuid )
{
    payment_id = yID;
  payment_bill = ybil;
  payment_type=  ytype;
    payment_description = ydes;
     payment_date = ydat;
  payment_customer_id =  ycuid;
}

void Payment::addPayment(){

}

void Payment::editPayment(){

}
void Payment::deletePayment(){

}
void Payment::searchPayment(){

}

    }
```

https://github.com/ishanayantha/IT18003192.git