ONLINE VACCINATION PORTAL

Topic : Online Vaccination Portal
Group no : MLB_08.02_08
Campus : Malabe
Submission Date : 05/20/2022

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
IT21186974	K.A.N.N. Perera	0766980579
IT21211850	A.C.M. Arangalla	0775351306
IT21271250	C. Ranaweera	0705618525
IT21231650	D.H. Peramunaarachchi	0786150482
IT21206528	K.H.D. Kawishan	0779933629

Table of Contents

PART 1

1. Requirements for the system identified – Online Vaccination Portal

I. Guests/unregistered users visit the online vaccination portal (web system) and access set of selected services provided by the system.

II. Guests sign up to system and create user profile.

III. Guests view vaccinating details pages.

IV. Guests access media and guidelines

V. Guests download guidelines

VI. Once a registered user enters credentials to log in, system validate the user to give access permissions.

VII. That person views all the pages related to vaccination and authorized pages by the administrators.

VIII. This user reserve vaccination place according to the location.

IX. Registered user should register to the reservation process to save details to the database.

X. After registering user can view and edit selected parts of information that the system stores.

XI. After registering to the reservation process user can manually enter location to track the nearest vaccinating place and select most convenient place from the list given for receive the vaccine.

XII. If the user needs system to track the user's location and detect nearest place, that facility is also given.

XIII. By each method, after selecting the most convenient place, user directs to get a list of vaccines available at the selected vaccinating place.

XIV. User can select preferable vaccine type.

XV. After the selection user must stick to the scheduled time slots as vaccinating venues are managed by government bodies.

XVI. One registered user can only reserve one vaccine at a time.

XVII. After a successful reservation user gets all the details about the due vaccinating process via an E-mail.

XVIII. For a user who has reserved a place to get vaccinated, automatically creates a User ID same as their NIC for the use of managers in the vaccinating place.

XIX. Using the ID or received e-mail, managers at the venue can verify and update user details to the system.

XX. After receiving vaccine dose if user have no digital vaccine card, data entry manager can create the card and enter details to the system according to user ID. If user already have card, it can be updated by the data entry manager

XXI. Any user can leave feedback for the system.

XXII. User can contact for a resourced person for further details using this system.

XXIII. E-channeling is one of those processes. Any user can channel a doctor to get instructions regarding the vaccinating process and health condition after or before receiving a vaccine.

XXIV. Management staff should verify payment before E-channeling.

XXV. System admins create statistic and detail reports on monthly vaccination progress and send to health and statistic departments of the state.

XXVI. Super administrators manage system administrators, managers and collaborated service systems such as location tracking system

- Red colored words are identified as nouns for identify classes according to the Noun-Verb Analysis method.
- Blue colored words are the set of verbs which were identified as the methods.
- Nouns have further refined according to the Rejecting Rules of Nouns and finalized as class names.

2. CLASSES

a) User

b) Patient

c) Manager

d) Admin

e) Vaccine

f) Digital_Vaccine_Card

g) Reservation

h) E-channel

i) Doctor

j) Payment

k) Feedback

l) Report

*These classes have been extracted according to the Noun-Verb Analyzing method. All the red colored words counted as nouns and by applying the five rules of dropping unnecessary class names the list has finalized.

3. CRC CARDS

1. User

User	
Responsibilities	**Collaborations**
Verify user login	Doctor, Customer, Manager, Administrator

2. Customer

Customer	
Responsibilities	**Collaborations**
Register for services	Reservation
Channel a doctor	Doctor
Pay for E-channeling	E-channel
Reserve Vaccination Place	Reservation
Give system feedbacks	Feedback

3. Manager

Manager	
Responsibilities	**Collaborations**
Update reservation details	User, Reservation
Create/Update vaccine card details	DigitalVaccineCard
Update user profiles	

4. Administrator

Administrator	
Responsibilities	**Collaborations**
Manage system staff and collaborated systems	Manager
Create reports	Report

5. Vaccine

Vaccine	
Responsibilities	**Collaborations**
Generate vaccine ID	
Re-store or update vaccination place details	
Update vaccine details	

6. Digital_Vaccine_Card

Digital_Vaccine_Card	
Responsibilities	**Collaborations**
Update vaccine card	Manager, Doctor
Generate vaccine card	Manager

7. Reservation

Reservation	
Responsibilities	**Collaborations**
Update reservation details	Manager

8. E-channel

E-channel	
Responsibilities	**Collaborations**
Verifying payment	Payment
Generate channeling details	Doctor

9. Doctor

Doctor	
Responsibilities	**Collaborations**
Update digital vaccine card	DigitalVaccineCard
Check patients (online mode)	Customer

10. Payment

Payment	
Responsibilities	**Collaborations**
Make new payments and verify	E-channel
Generate ID for payments	

11. Feedback

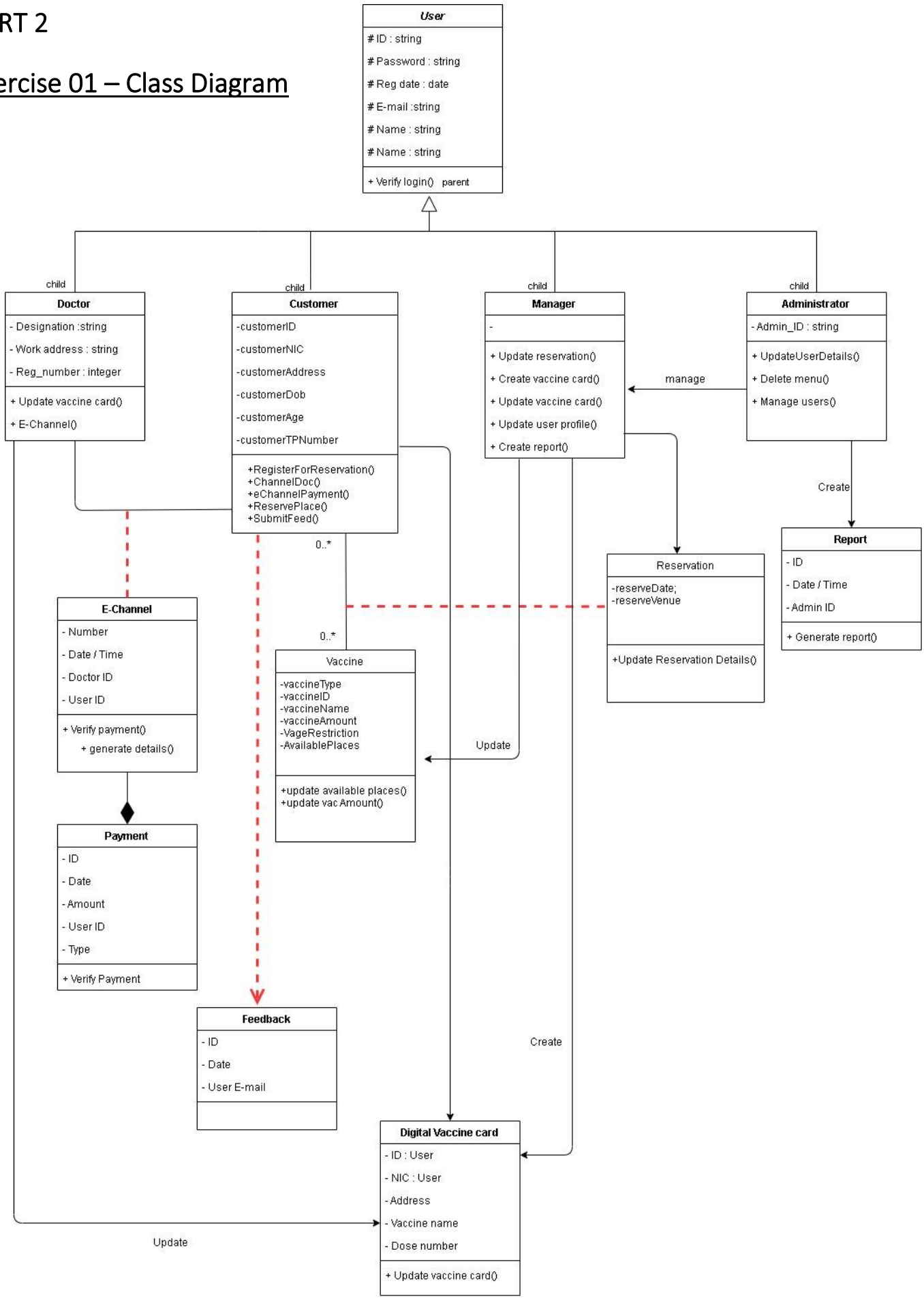
Feedback	
Responsibilities	**Collaborations**
Store feedbacks	Customer

12. Report

Report	
Responsibilities	**Collaborations**
Generate weekly/monthly reservation reports	Admin
Update vaccine statistic reports	Admin
Generate payment reports	Admin

PART 2

Exercise 01 – Class Diagram



Exercise 02 - Coding of classes

```
#include <iostream>
#include <cstring>
using namespace std;
#define SIZE 20

////////////////////////////////////////////////////////////
class User
{
	protected :
		char userID[8];
		char userName[20];
		char userPassword[15];
		int userDate;
		char userEmail[20];

	public :
		User();
		void verify_login();

};
//-------------------------------------------------------------

//Implementing user public methods

User::User()
{
	cout<<"User default constructor"<<endl;
}

void User::verify_login()
{
  cout<<"User verify login method successful " <<endl;
}


//-------------------------------------------------------------

class Customer :public User
 {
	private :
		char customerID[8];
		char customerNIC[12];
		char customerAddress[30];
		char customerDob[10];
```

```
                int customerAge;
                int customerTPNumber;

        public :
                Customer();
                Customer(char cID[],char cNIC[],char cAddress[],char Cdob[],int cage,int cTPN);
                void RegisterForReservation();
                void ChannelDoc();
                void eChannelPayment();
                void ReservePlace();
                void SubmitFeed();

};

//Implementing Customer class public methods

Customer::Customer()
{
  cout<<"Customer default"<<endl;
}

Customer::Customer(char cID[],char cNIC[],char cAddress[],char Cdob[],int cage,int cTPN)
{
        strcpy(customerID,cID);
        strcpy(customerNIC,cNIC);
        strcpy(customerAddress,cAddress);
        strcpy(customerDob,Cdob);
        customerAge=cage;
        customerTPNumber=cTPN;

        cout<<"Customer Overloaded constructor succsessful " <<endl;
}

void Customer::RegisterForReservation()
{
  cout<<"Customer RegisterForReservation method succsessful " <<endl;
}

void Customer::ChannelDoc()
{
 cout << "Customer ChannelDoc method successful"<< endl;
}

void Customer::eChannelPayment()
{
 cout << "Customer eChannelPayment method successful"<< endl;
}
```

```
void Customer::ReservePlace()
{
  cout << "Customer ReservePlace method successful"<< endl;
}

void Customer::SubmitFeed()
{
  cout << "Customer SubmitFeed method successful"<< endl;
}


//--------------------------------------------------

class Vaccine {
        private :
                char vaccineType[20];
                char vaccineID[10];
                char vaccineName[20];
                int VacgeRestriction;
                int vaccineAmount;
                char AvailablePlaces[50];

        public :
                Vaccine();
                Vaccine(char vtype[],char vID[],char Vname[],int vageres);
                void update_availableplaces(char Aplaces[]);
                void update_vacAmount(int vacAmount);
};




//Implementing Vaccine class public methods

Vaccine::Vaccine()
{
        cout<<"Vaccine default constructor called"<<endl;
}

Vaccine::Vaccine(char vtype[],char vID[],char Vname[],int vageres)
{
         strcpy(vaccineType,vtype);
         strcpy(vaccineID,vID);
        strcpy(vaccineName,Vname);
         VacgeRestriction = vageres;

         cout << "Vaccine Constructer successful"<< endl;
}
```

```
void Vaccine::update_availableplaces(char Aplaces[])
{
	strcpy(AvailablePlaces,Aplaces);
	cout << "Vaccine update_availableplaces method successful"<< endl;
}

void Vaccine::update_vacAmount(int vacAmount)
{
	vaccineAmount=vacAmount;
	cout << "Vaccine update_vacAmount method successful"<< endl;

}


//-------------------------------------------------------------------------------------------------------

class Reservation {
	private :
		Customer *customer;
		Vaccine *vaccine;
		int reserveDate;
		char reserveVenue[];
	public :
		void Add_ReservationDetails(Customer *rcus,Vaccine *rvac,int rDate,char rVenue[]);
		~Reservation();
};


void  Reservation::Add_ReservationDetails(Customer *rcus,Vaccine *rvac,int rDate,char rVenue[])
{
	cout << "Reservation Add_ReservationDetails method successful"<< endl;
}

Reservation::~Reservation()
{
	cout << "Reservation deleted" << endl;
}

//-------------------------------------------------------------------------------------------------------

class Administrator {
	private :
		 User*us;
		Manager*mgr;
		Report*rep;
		char adminid[10];
```

```
        public :
                Administrator();
                void update_menu();
                void Delete_menu();
                void Manage_users();

};

Administrator::Administrator(char aid[])
{
 strcpy(adminid,aid);
        cout<<"Administor default constructor"<<endl;
}

void Administor::update_menu()
{
 cout<<"Administor update menu method successful" <<endl;
}

void Administor::Delete_menu()
{
 cout<<"Administor Delete menu method successful" <<endl;
}

void Administrator::Manage_users()
{
 cout<<"Administor Manage users method successful" <<endl;
}


//-------------------------------------------------------------------------------------------------------
class Manager : public User

{
        private:
                Reservation *reserve;
        public:
                void updateVaccineCard();
                void updateReservation(Reservation *res);
};

void Manager::updateReservation(Reservation *res)
{
 cout<<"UpdateReservation method has called"<<endl;
}
```

```
//-------------------------------------------------------------------------------------------------
```

class Report

```
{
        private :
                Administrator *Administrator;
                char reportID[8];
                char reporttime[30];
                char reportdate[10];

        public :
           Report();
           Report(char rID[],char rtime[],char rDate[],Administrator *aid;);
           void Generatereport();

};

Report::Report()
{
  cout<<"Report default"<<endl;
}

Report::Report(char rID[],char rtime[],char rDate[],Administrator *aid;)
{
        strcpy(reportID,rID);
        strcpy(reporttime,rtime);
          strcpy(reportdate,rDate);
        cout<<"Report Overloaded constructor succsessful " <<endl;
}

void Report::Generatereport()
{
  cout<<"Generate report method succsessful " <<endl;
}
```

```
//-------------------------------------------------------------------------------------------------
```

class Feedback

```
{
private:
     int feedbackId;
     int date;
     char email[20];
public:
     Feedback(int fbId,int dte,char mail[]);
    void displayFeedback();
```

```
    ~Feedback();
};

Feedback::Feedback(int fbId,int date,char mail[])
{
    feedbackId=fbId;
    strcpy(email,mail);
    date = dte;
}

void Feedback::displayFeedback()
{
  cout<<"Here you see the displayFeedback function" <<endl;
}

Feedback::~Feedback()
{}

//---------------------------------------------------------------------------------------------------
```

class Doctor : public User

```
{
        private:
                char workAddress[30];
                char Designation[20];
        public:
                Doctor();
                void updateVaccineCard();
                void Echannel();
};


//---------------------------------------------------------------------------------------------------
```

class Echannel

```
{
        private:
                int number;
                char dateAndTime[40];
                int doctorId;
                int userId;
                Payment* p1
        public:
                Echannel();
                Echannel(int tnumber, char tchardateAndTime[], int tdoctorId, int tuserId);
                ~Echannel();
};
```

```
//-----------------------------------------------------------------------------------------------
```

```
class Payment
{
        private:
                int id;
                char date[40];
                float amount;
                int userId;
                char type[20];
        public:
                Payment();
                void verifyPayment();};
```

```
//-----------------------------------------------------------------------------------------------------------------
```

//Main program

```
int main()
{
  //dynamic object

   User *c1[3];
   c1[0]= new User();
   c1[1]->verify_login();

  Customer *cus[4];
   cus[0] = new Customer();
   cus[2]= new Customer("cm001","1e2qwda","gadaca,as","3234",20,077);
   cus[1]->ChannelDoc();
   cus[1]->eChannelPayment();

   Vaccine *vac;
   vac=new Vaccine();

   Reservation *r;
   r->Add_ReservationDetails(cus[2],vac,22,"cm");


  return 0;
}
```