Topic : Bus Scheduling & Booking System

Group no : MLB_WD_CSNE_13_05

Campus : Malabe

Submission Date:

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**
IT21253508	A.R.B.L. Athapattu	0763313304
IT21229916	S.C.W. Dissanayake	0773288284
IT21014468	T.M.L.D. Wickremasinghe	0711874249
IT21031670	A.M.W.Y. Abayakoon	0774674802
IT21212154	M.M.R.S. Costa	0767091471

Table of Contents

1. Description of the requirements

Our database and system are made for online bus scheduling and booking. For this system we need user details, bus details, staff details and payment details. In this system we store this information and display them when they wanted.

User will give their information like name, email, contact numbers, address, and password for their accounts. After login user has the chance to select their bus, bus number, bus type, route, destination, time, and payment methods. if user wants to cancel the booking it also can be happen through the process. User can leave feedback if they like. Finally, user will have a ticket with all relevant details.

Bus staff will be assigned by the organization and details will be stored in the database.

2. Identified Classes

- Users.
- Feedback.
- Booking.
- Bus Info.
- Payment.
- Cards.
- Bus Category.
- Route Category.
- Cancellation.
- Bus staff.
- Ticket.

3. System Requirements

• Application will store data entries of users and make a profile.

• System will allow new registrations.

• System will store and display the bus types, destination, road type and route.

• System will display bus and available seat.

• Then system will display ticket and calculate the fee.

• System will update after the process.

4. Method

- Users – new user will enter new credentials. (Name, birthday, con number)

  Registered users will enter their credentials and log in to system.

- Feedback – users will give feedback about service.
- Booking – users will book seats according to their preferences.
- Bus Info – store information about buses, drivers, conductors, and routes.
- Payment – users will give payment details.
- Cards – store information about card.
- Bus Category – view bus id, number, and type of bus.
- Route Category – view routes and route numbers.
- Cancellation – gives a cancellation code, user will input their payment id.
- Bus staff – store staff members information.
- Ticket – display information that are on the ticket.

5. CRC Cards

5.1. IT21229916 - S.C.W. Dissanayake

Class name - Users	
Responsibilities:	Collaborations:
New user register to system by giving credentials.	-
Registered user inputs credentials.	-

Class name – Feedback and Review	
Responsibilities:	Collaborations:
User enters credentials.	Users
User review the service as he preferred.	-

5.2. IT21031670 - A.M.W.Y. Abayakoon

Class name - Booking	
Responsibilities:	Collaborations:
Users must choose buses, route, destination and others. Booking section of organize will provide a Booking number	Route_Category

Class name - Cancellation	
Responsibilities:	Collaborations:
User input name and ID	User
User input payment ID for refund	Payment
System will provide a code	-

5.3. IT21253508 - A.R.B.L. Athapattu

Class name – Bus information	
Responsibilities:	Collaborations:
Gives information about bus and route	Booking
Store staff members ID	Bus Staff

Class name - Payment	
Responsibilities:	Collaborations:
User input their name	Users
User will choose payment type and they will receive an ID	-

Class name – Route Category	
Responsibilities:	Collaborations:
User choose details of route	-

5.4. IT21014468 -T.M.L.D. Wickremasinghe

Class name - Card	
Responsibilities:	Collaborations:
User gives relevant information about card	Users
User will get ticket price	Ticket

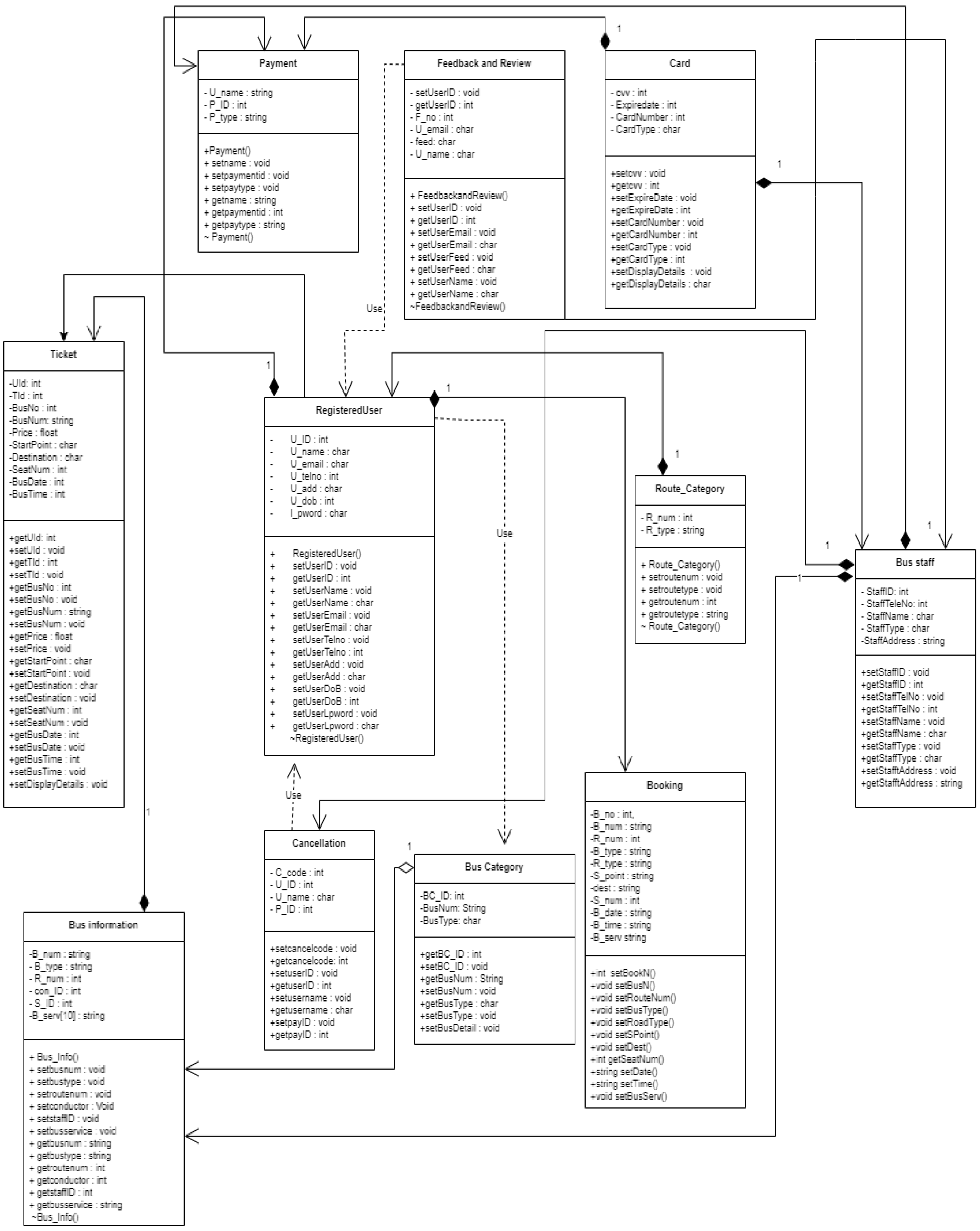
Class name – Bus Staff	
Responsibilities:	Collaborations:
Include Staff ID	Bus_Information
Include information about staff member	-

5.5. IT21212154 - M.M.R.S. Costa

Class name - Ticket	
Responsibilities:	Collaborations:
Give ticket price and ticket number	-
Ticket display user ID	User
Ticket display Seat details	Booking
Ticket display payment details	Payment

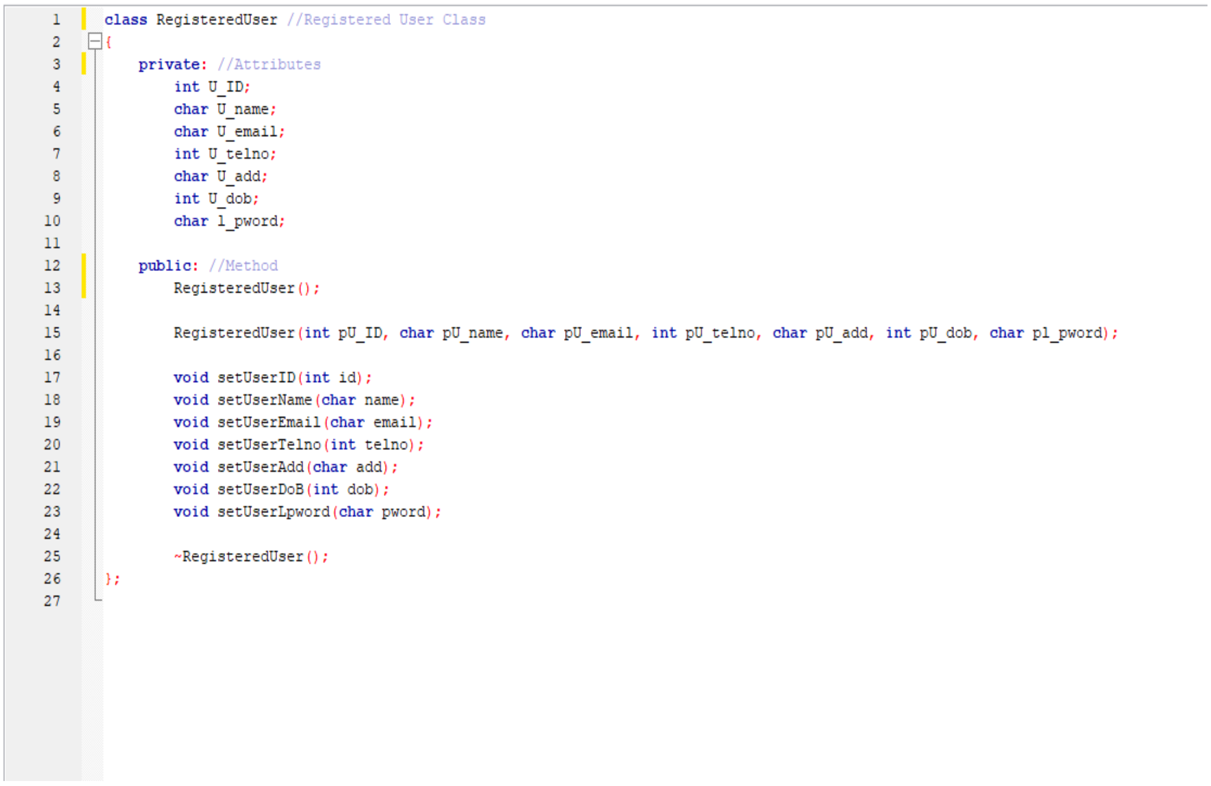
Class name – Bus Category	
Responsibilities:	Collaborations:
Gives bus type and bus number	Booking
Provide a bus category ID	-

6. Class Diagram (UML Notation)



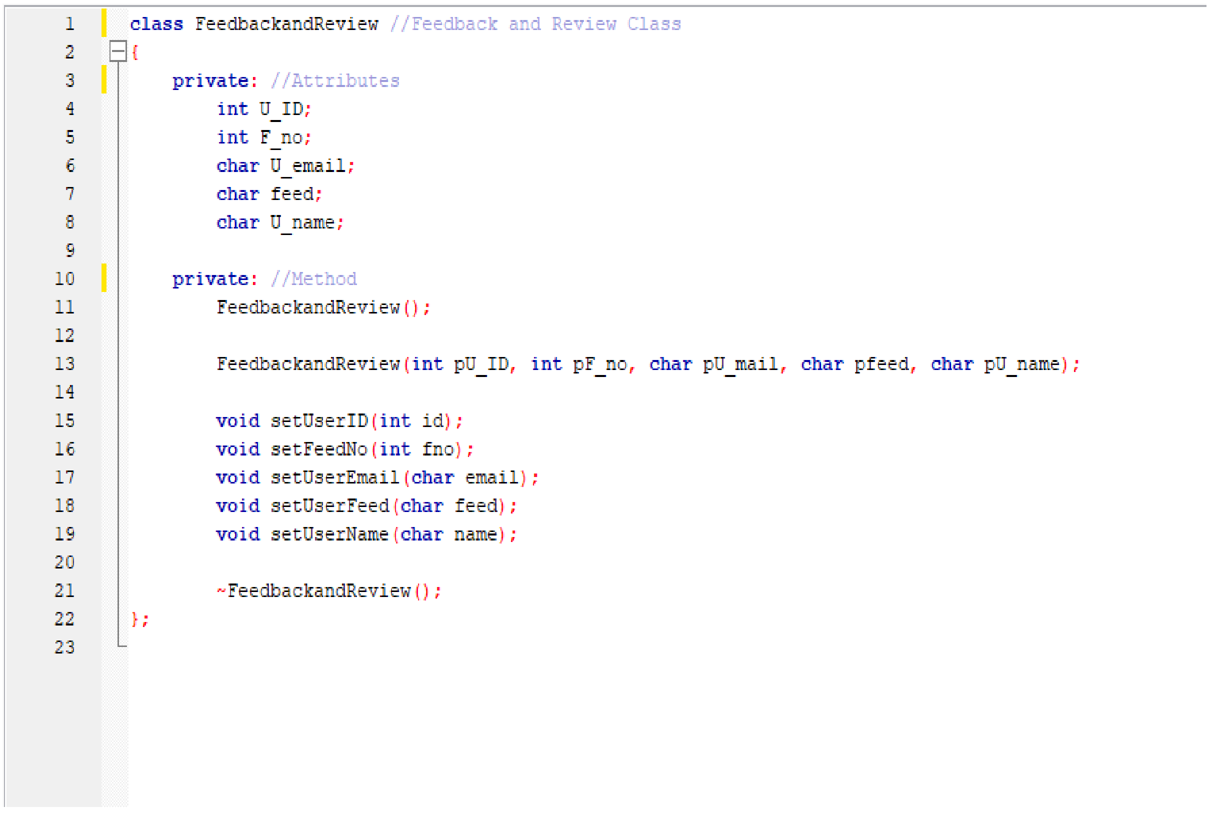
7. Class Headers Files

7.1. Users.



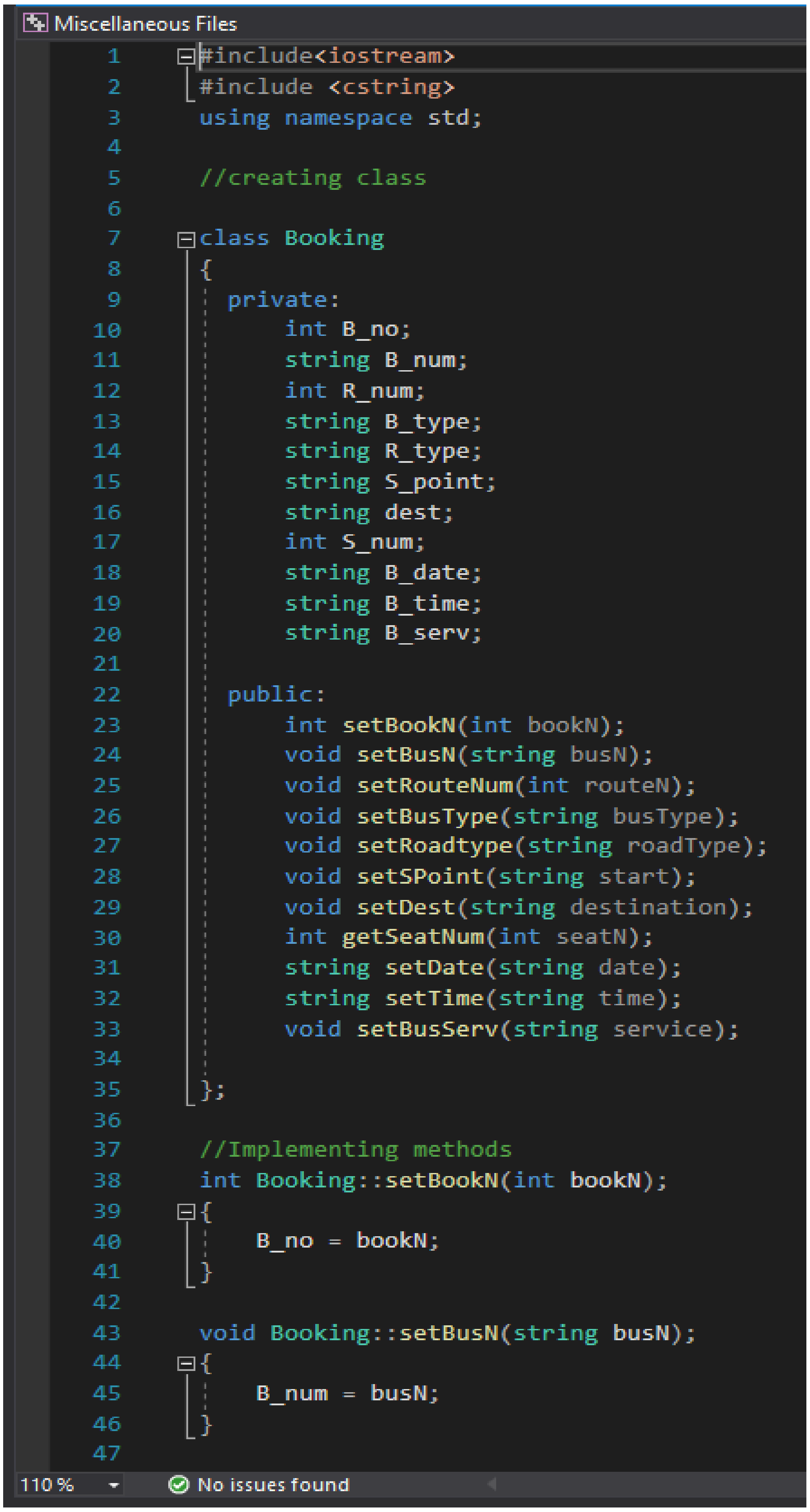
```
class RegisteredUser //Registered User Class
{
    private: //Attributes
        int U_ID;
        char U_name;
        char U_email;
        int U_telno;
        char U_add;
        int U_dob;
        char l_pword;

    public: //Method
        RegisteredUser();

        RegisteredUser(int pU_ID, char pU_name, char pU_email, int pU_telno, char pU_add, int pU_dob, char pl_pword);

        void setUserID(int id);
        void setUserName(char name);
        void setUserEmail(char email);
        void setUserTelno(int telno);
        void setUserAdd(char add);
        void setUserDoB(int dob);
        void setUserLpword(char pword);

        ~RegisteredUser();
};
```

7.2. Feedback.



```
class FeedbackandReview //Feedback and Review Class
{
    private: //Attributes
        int U_ID;
        int F_no;
        char U_email;
        char feed;
        char U_name;

    private: //Method
        FeedbackandReview();

        FeedbackandReview(int pU_ID, int pF_no, char pU_mail, char pfeed, char pU_name);

        void setUserID(int id);
        void setFeedNo(int fno);
        void setUserEmail(char email);
        void setUserFeed(char feed);
        void setUserName(char name);

        ~FeedbackandReview();
};
```

7.3. Booking

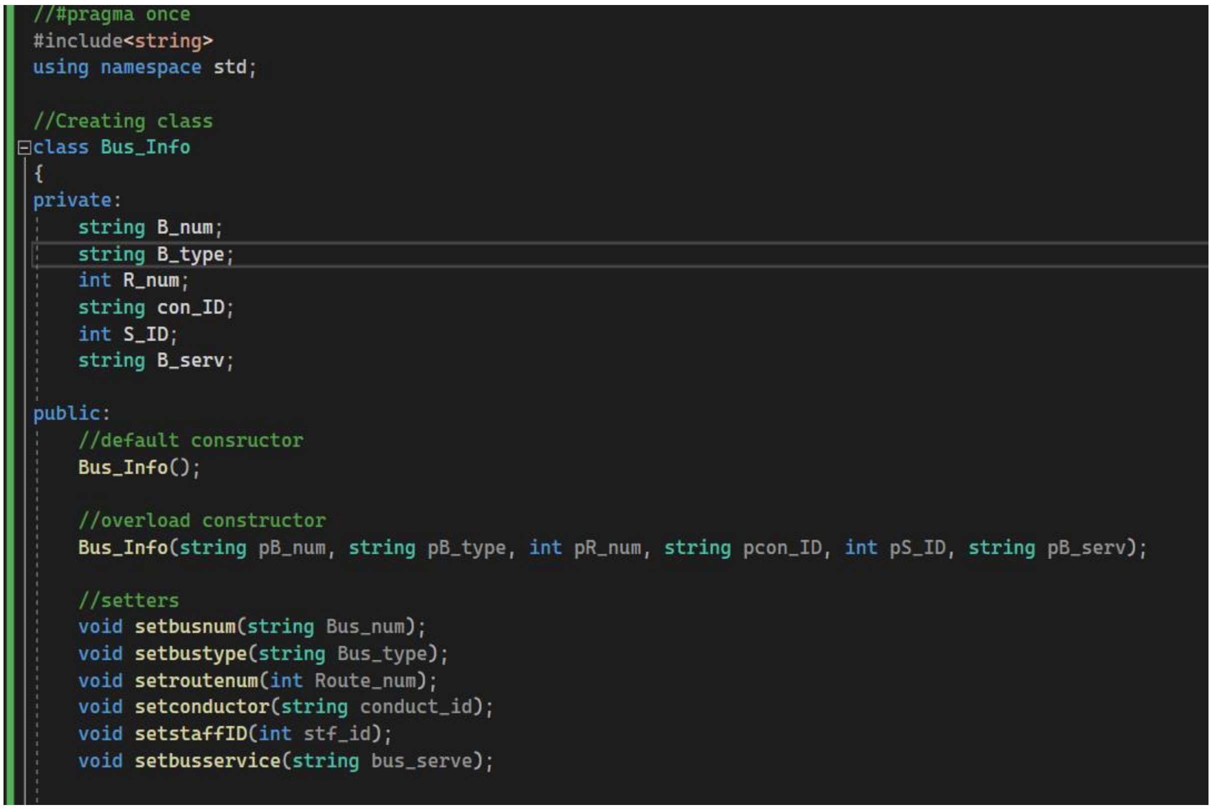


Miscellaneous Files

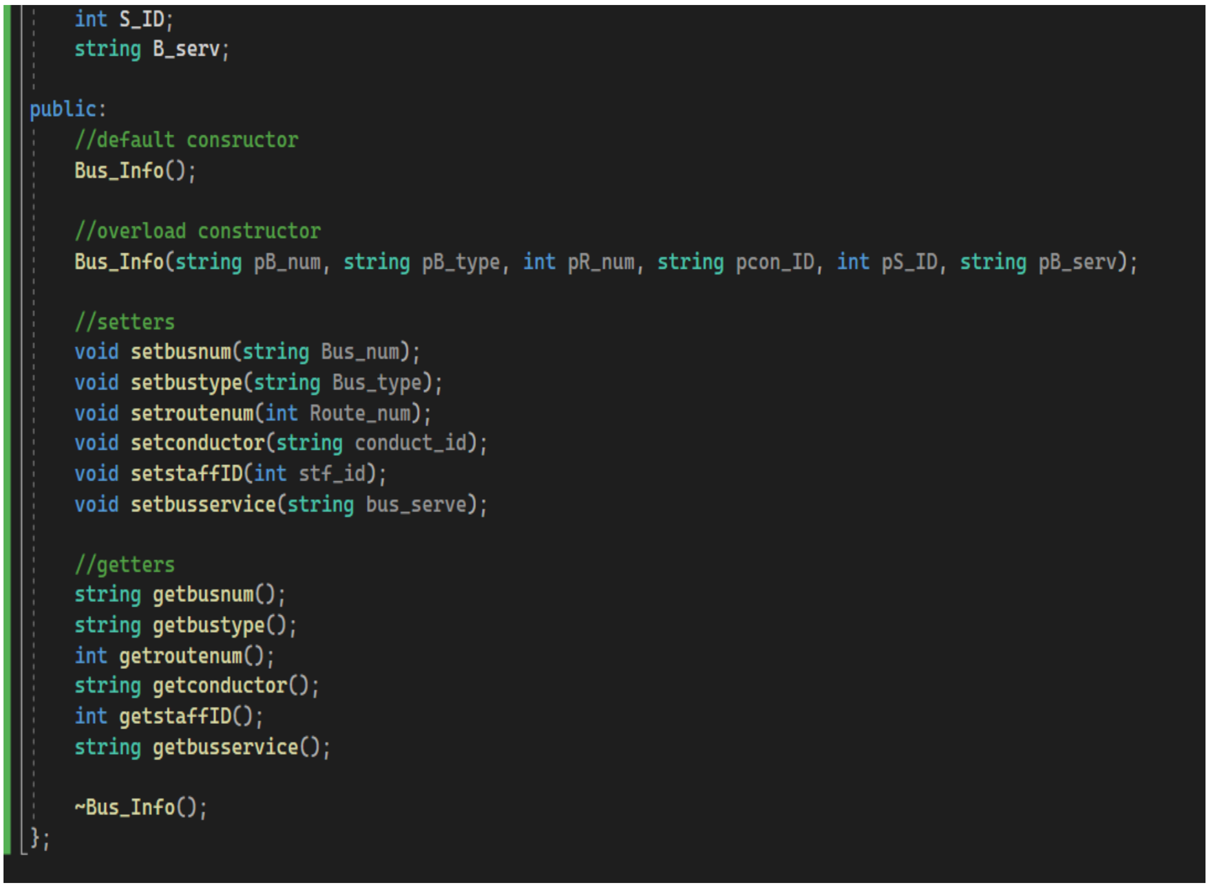
```
#include<iostream>
#include <cstring>
using namespace std;

//creating class

class Booking
{
    private:
        int B_no;
        string B_num;
        int R_num;
        string B_type;
        string R_type;
        string S_point;
        string dest;
        int S_num;
        string B_date;
        string B_time;
        string B_serv;

    public:
        int setBookN(int bookN);
        void setBusN(string busN);
        void setRouteNum(int routeN);
        void setBusType(string busType);
        void setRoadtype(string roadType);
        void setSPoint(string start);
        void setDest(string destination);
        int getSeatNum(int seatN);
        string setDate(string date);
        string setTime(string time);
        void setBusServ(string service);

};

//Implementing methods
int Booking::setBookN(int bookN);
{
    B_no = bookN;
}

void Booking::setBusN(string busN);
{
    B_num = busN;
}

```

110 %   No issues found

7.4. Bus Info.

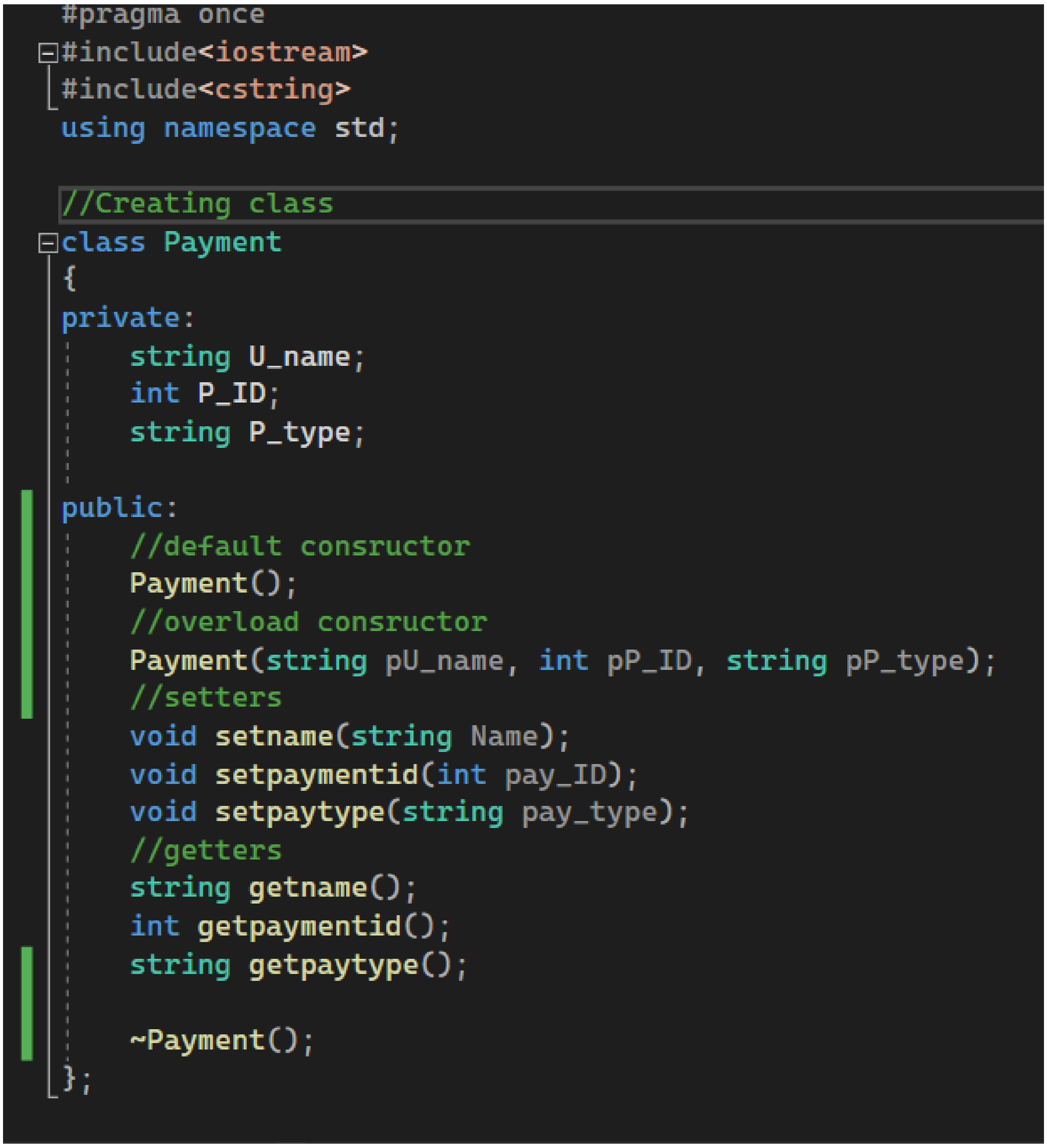


```
//#pragma once
#include<string>
using namespace std;

//Creating class
class Bus_Info
{
private:
    string B_num;
    string B_type;
    int R_num;
    string con_ID;
    int S_ID;
    string B_serv;

public:
    //default consructor
    Bus_Info();

    //overload constructor
    Bus_Info(string pB_num, string pB_type, int pR_num, string pcon_ID, int pS_ID, string pB_serv);

    //setters
    void setbusnum(string Bus_num);
    void setbustype(string Bus_type);
    void setroutenum(int Route_num);
    void setconductor(string conduct_id);
    void setstaffID(int stf_id);
    void setbusservice(string bus_serve);
```



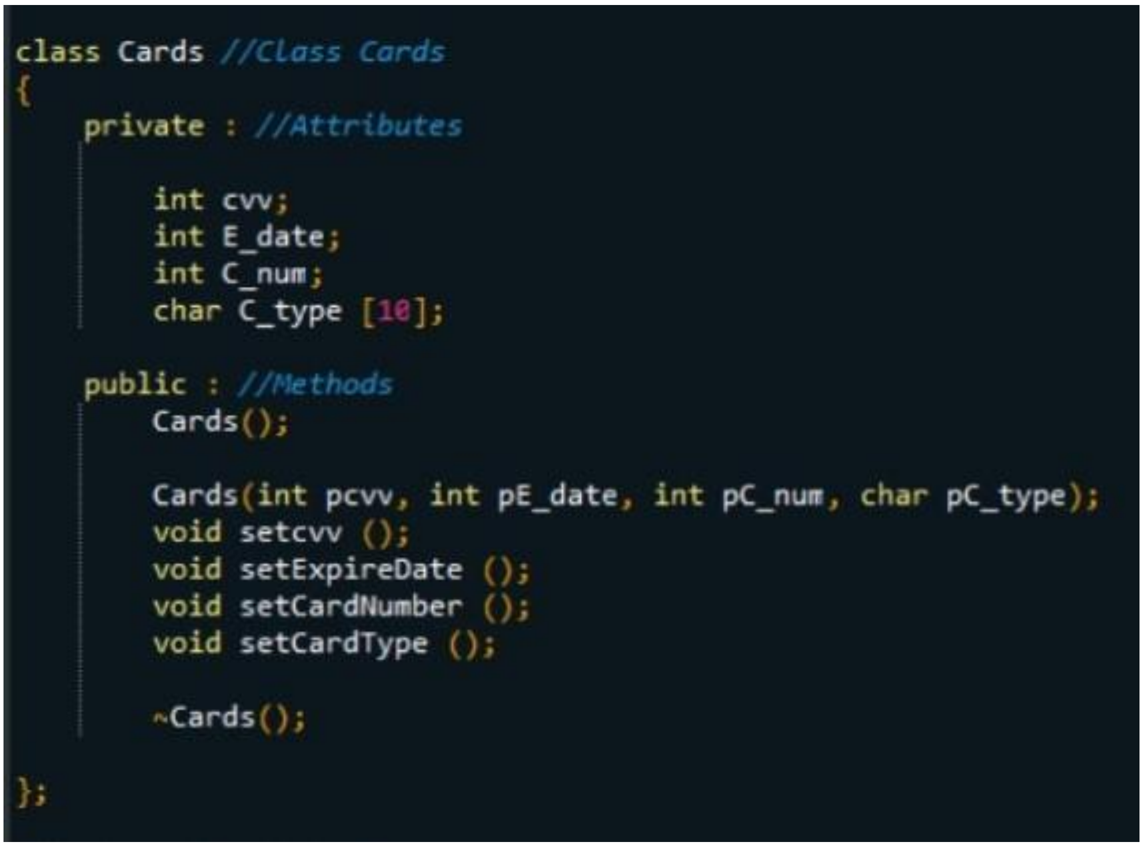
```
    int S_ID;
    string B_serv;

public:
    //default consructor
    Bus_Info();

    //overload constructor
    Bus_Info(string pB_num, string pB_type, int pR_num, string pcon_ID, int pS_ID, string pB_serv);

    //setters
    void setbusnum(string Bus_num);
    void setbustype(string Bus_type);
    void setroutenum(int Route_num);
    void setconductor(string conduct_id);
    void setstaffID(int stf_id);
    void setbusservice(string bus_serve);

    //getters
    string getbusnum();
    string getbustype();
    int getroutenum();
    string getconductor();
    int getstaffID();
    string getbusservice();

    ~Bus_Info();
};
```

7.5. Payment.



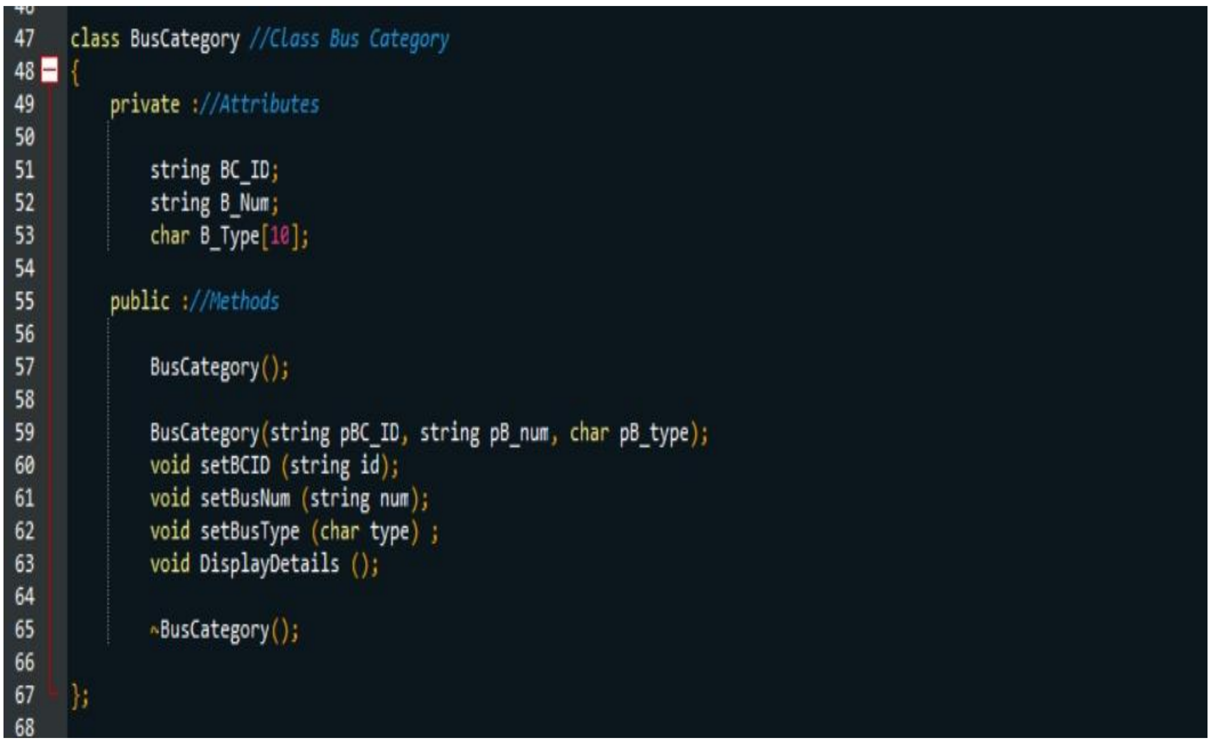
```
#pragma once
#include<iostream>
#include<cstring>
using namespace std;

//Creating class
class Payment
{
private:
    string U_name;
    int P_ID;
    string P_type;

public:
    //default consructor
    Payment();
    //overload consructor
    Payment(string pU_name, int pP_ID, string pP_type);
    //setters
    void setname(string Name);
    void setpaymentid(int pay_ID);
    void setpaytype(string pay_type);
    //getters
    string getname();
    int getpaymentid();
    string getpaytype();

    ~Payment();
};
```

7.6. Cards.



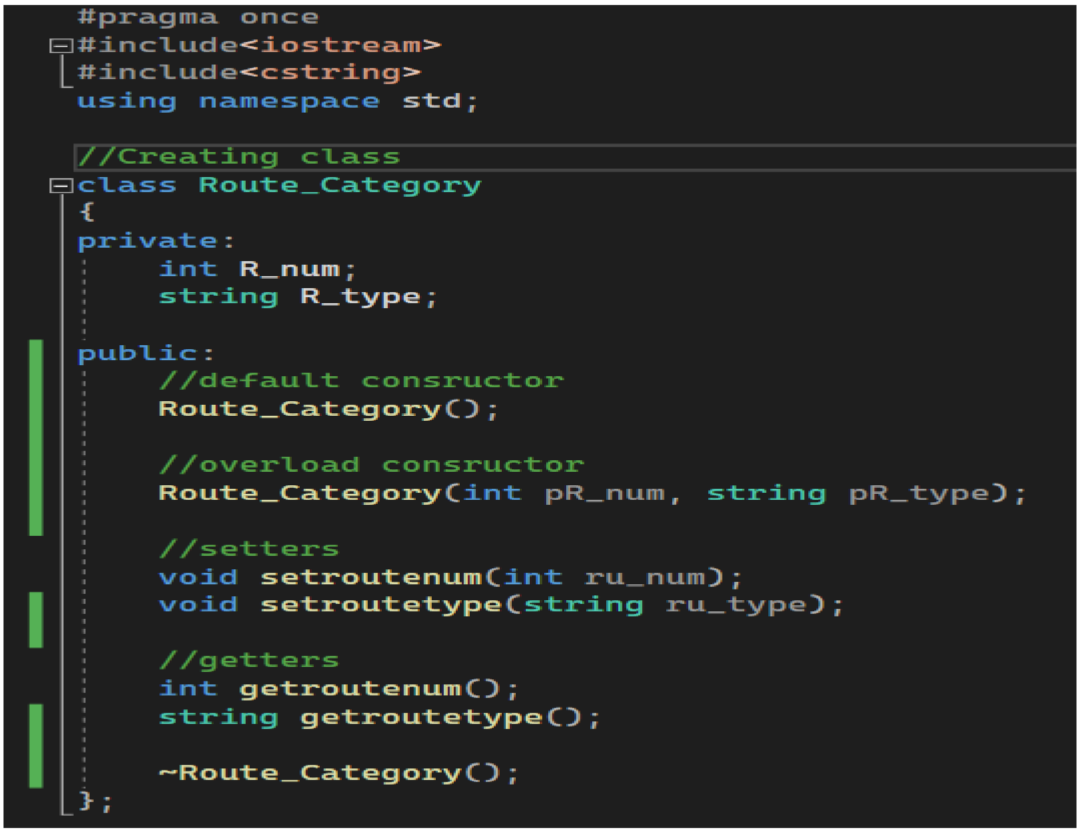
```
class Cards //Class Cards
{
    private : //Attributes

        int cvv;
        int E_date;
        int C_num;
        char C_type [10];

    public : //Methods
        Cards();

        Cards(int pcvv, int pE_date, int pC_num, char pC_type);
        void setcvv ();
        void setExpireDate ();
        void setCardNumber ();
        void setCardType ();

        ~Cards();

};
```

7.7. Bus Category.



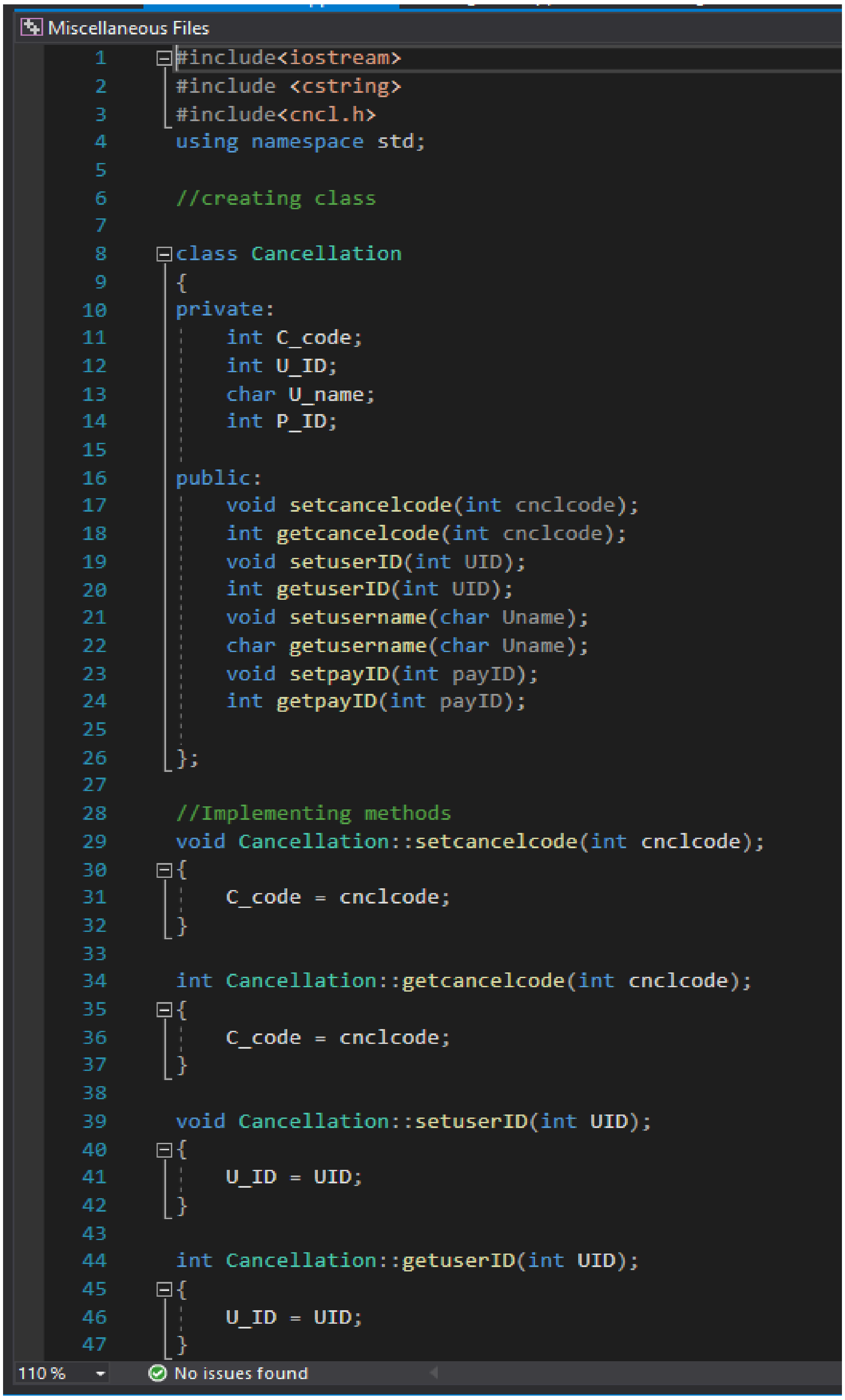
```
class BusCategory //Class Bus Category
{
    private ://Attributes

        string BC_ID;
        string B_Num;
        char B_Type[10];

    public ://Methods

        BusCategory();

        BusCategory(string pBC_ID, string pB_num, char pB_type);
        void setBCID (string id);
        void setBusNum (string num);
        void setBusType (char type) ;
        void DisplayDetails ();

        ~BusCategory();

};
```

7.8. Route Category.



```
#pragma once
#include<iostream>
#include<cstring>
using namespace std;

//Creating class
class Route_Category
{
private:
	int R_num;
	string R_type;

public:
	//default consructor
	Route_Category();

	//overload consructor
	Route_Category(int pR_num, string pR_type);

	//setters
	void setroutenum(int ru_num);
	void setroutetype(string ru_type);

	//getters
	int getroutenum();
	string getroutetype();

	~Route_Category();
};
```

7.9. Cancellation.

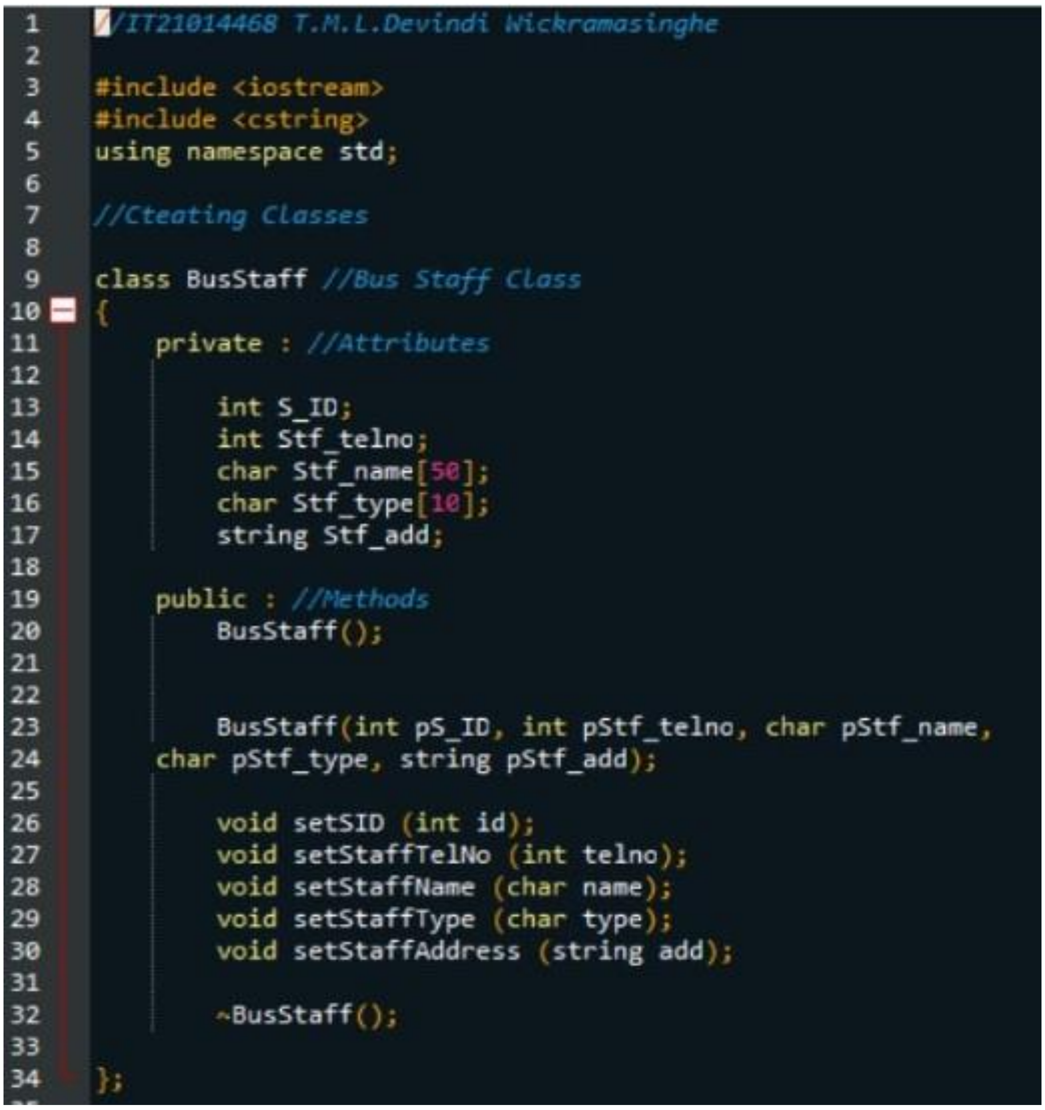


Miscellaneous Files

```
#include<iostream>
#include <cstring>
#include<cncl.h>
using namespace std;

//creating class

class Cancellation
{
private:
    int C_code;
    int U_ID;
    char U_name;
    int P_ID;

public:
    void setcancelcode(int cnclcode);
    int getcancelcode(int cnclcode);
    void setuserID(int UID);
    int getuserID(int UID);
    void setusername(char Uname);
    char getusername(char Uname);
    void setpayID(int payID);
    int getpayID(int payID);

};

//Implementing methods
void Cancellation::setcancelcode(int cnclcode);
{
    C_code = cnclcode;
}

int Cancellation::getcancelcode(int cnclcode);
{
    C_code = cnclcode;
}

void Cancellation::setuserID(int UID);
{
    U_ID = UID;
}

int Cancellation::getuserID(int UID);
{
    U_ID = UID;
}
```

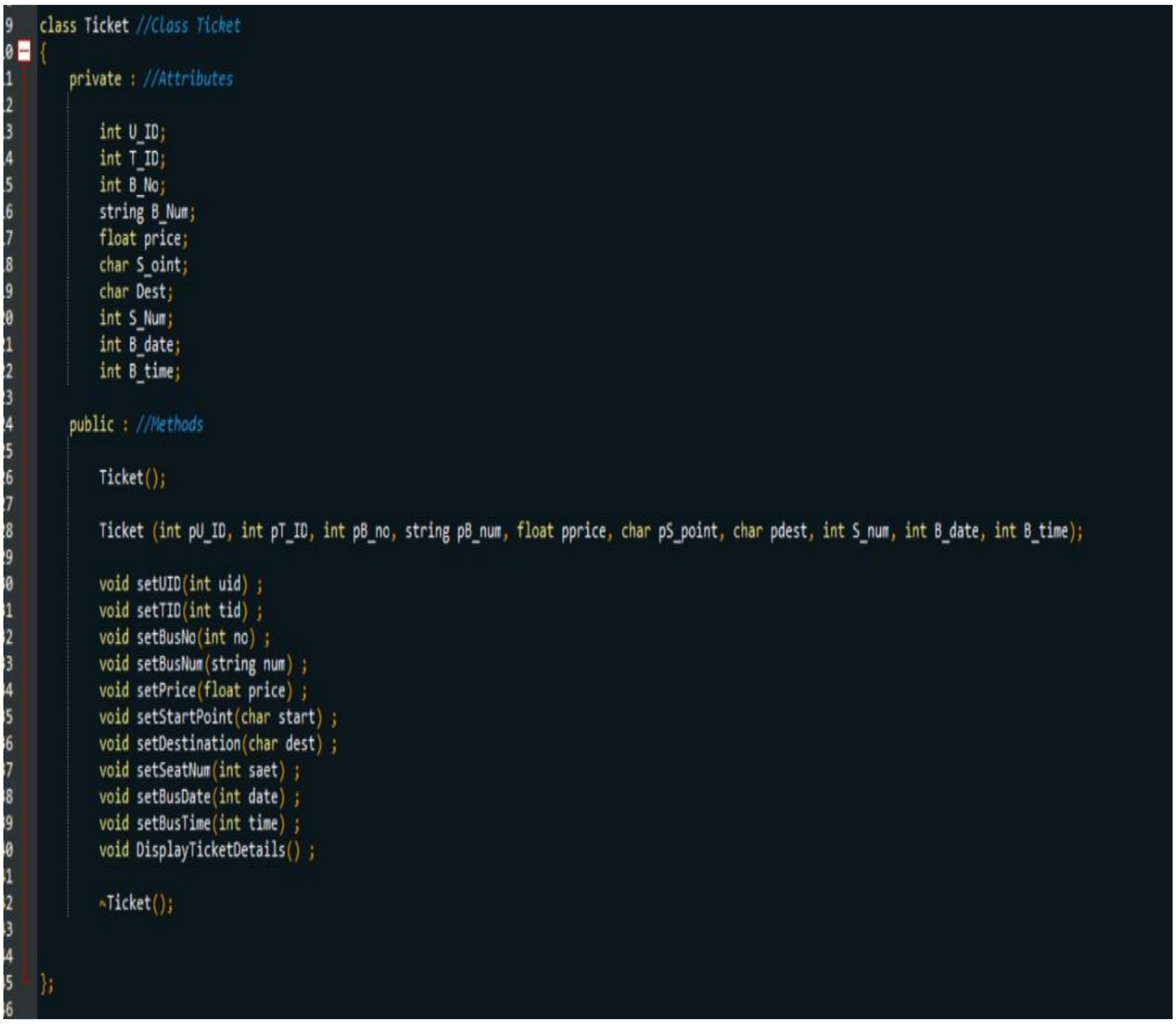
110 % No issues found

7.10. Bus staff.



```
//IT21014468 T.M.L.Devindi Wickramasinghe

#include <iostream>
#include <cstring>
using namespace std;

//Cteating Classes

class BusStaff //Bus Staff Class
{
    private : //Attributes

        int S_ID;
        int Stf_telno;
        char Stf_name[50];
        char Stf_type[10];
        string Stf_add;

    public : //Methods
        BusStaff();


        BusStaff(int pS_ID, int pStf_telno, char pStf_name,
    char pStf_type, string pStf_add);

        void setSID (int id);
        void setStaffTelNo (int telno);
        void setStaffName (char name);
        void setStaffType (char type);
        void setStaffAddress (string add);

        ~BusStaff();

};
```

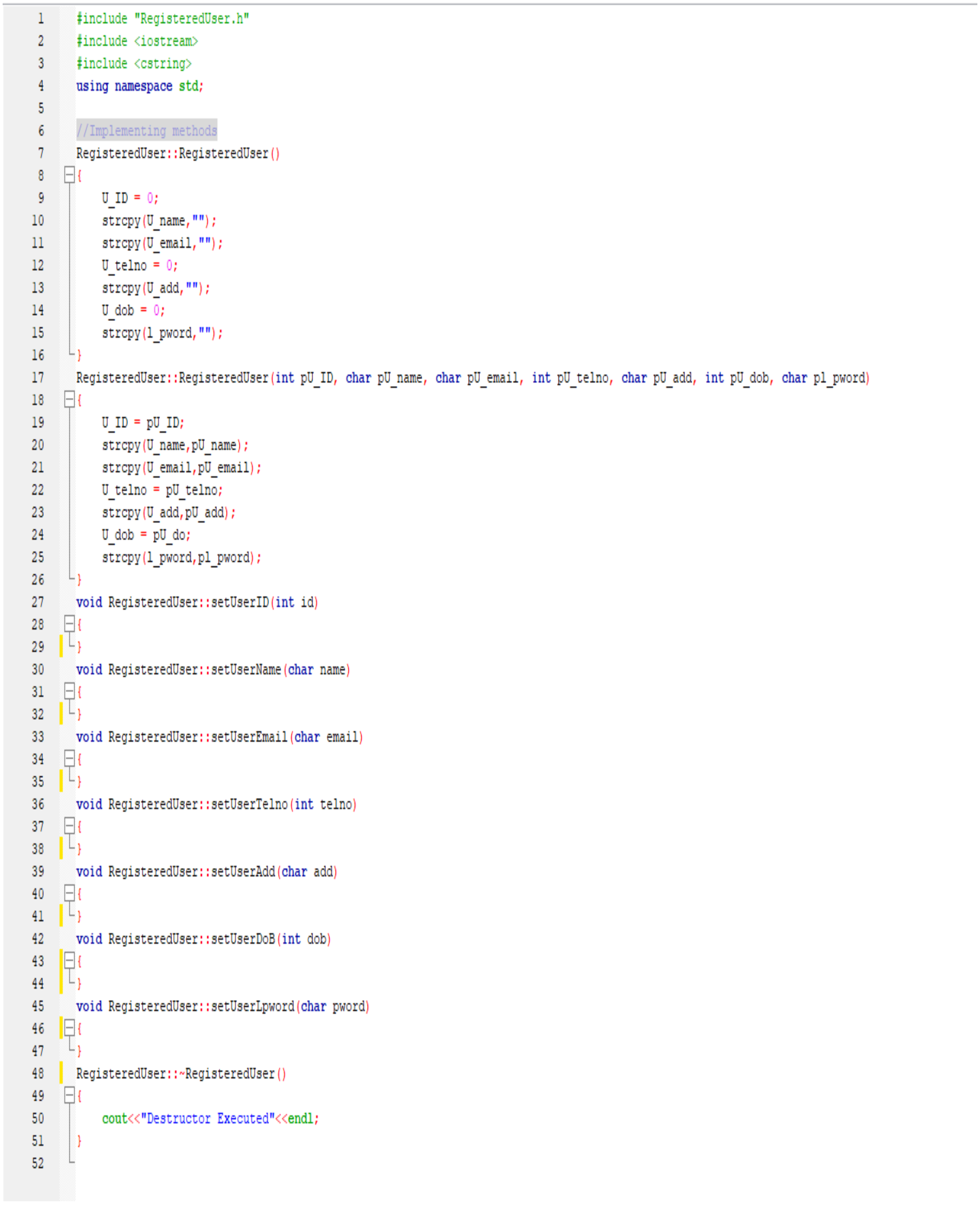
7.11. Ticket.



```
class Ticket //Class Ticket
{
    private : //Attributes

        int U_ID;
        int T_ID;
        int B_No;
        string B_Num;
        float price;
        char S_oint;
        char Dest;
        int S_Num;
        int B_date;
        int B_time;

    public : //Methods

        Ticket();

        Ticket (int pU_ID, int pT_ID, int pB_no, string pB_num, float pprice, char pS_point, char pdest, int S_num, int B_date, int B_time);

        void setUID(int uid) ;
        void setTID(int tid) ;
        void setBusNo(int no) ;
        void setBusNum(string num) ;
        void setPrice(float price) ;
        void setStartPoint(char start) ;
        void setDestination(char dest) ;
        void setSeatNum(int saet) ;
        void setBusDate(int date) ;
        void setBusTime(int time) ;
        void DisplayTicketDetails() ;

        ~Ticket();


};
```

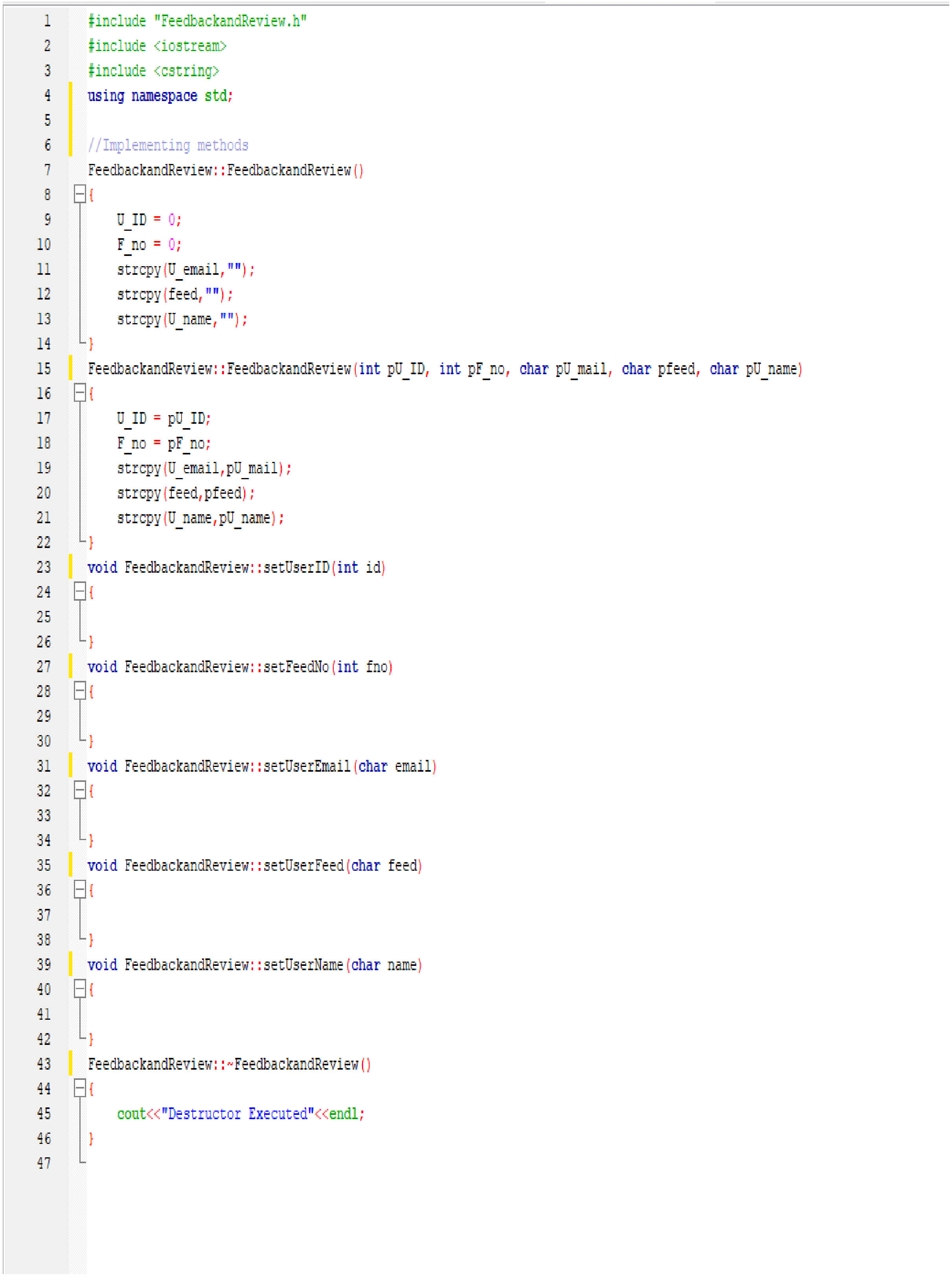
8. Class Cpp files

8.1. Users.



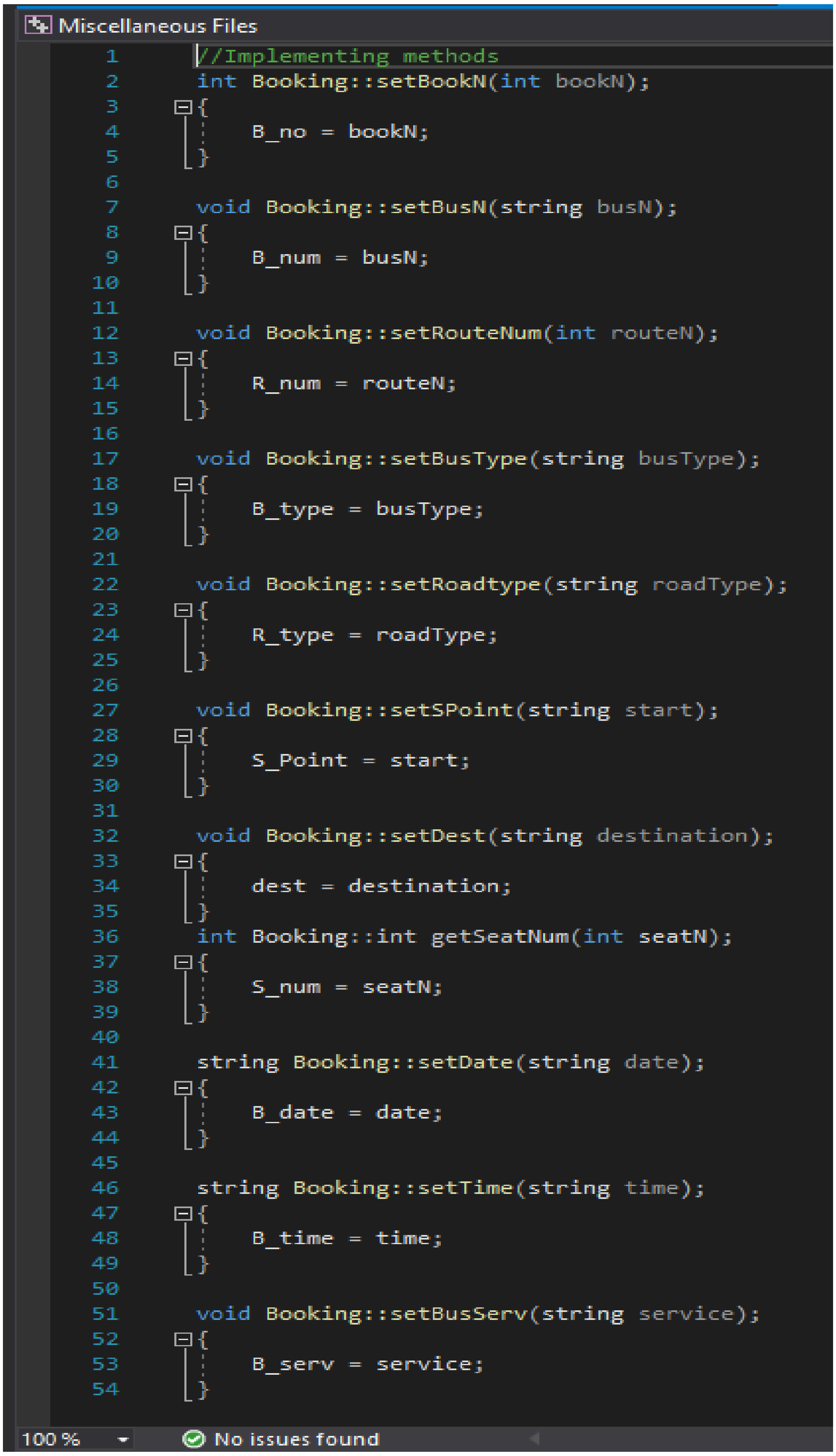
```cpp
#include "RegisteredUser.h"
#include <iostream>
#include <cstring>
using namespace std;

//Implementing methods
RegisteredUser::RegisteredUser()
{
    U_ID = 0;
    strcpy(U_name,"");
    strcpy(U_email,"");
    U_telno = 0;
    strcpy(U_add,"");
    U_dob = 0;
    strcpy(l_pword,"");
}
RegisteredUser::RegisteredUser(int pU_ID, char pU_name, char pU_email, int pU_telno, char pU_add, int pU_dob, char pl_pword)
{
    U_ID = pU_ID;
    strcpy(U_name,pU_name);
    strcpy(U_email,pU_email);
    U_telno = pU_telno;
    strcpy(U_add,pU_add);
    U_dob = pU_do;
    strcpy(l_pword,pl_pword);
}
void RegisteredUser::setUserID(int id)
{
}
void RegisteredUser::setUserName(char name)
{
}
void RegisteredUser::setUserEmail(char email)
{
}
void RegisteredUser::setUserTelno(int telno)
{
}
void RegisteredUser::setUserAdd(char add)
{
}
void RegisteredUser::setUserDoB(int dob)
{
}
void RegisteredUser::setUserLpword(char pword)
{
}
RegisteredUser::~RegisteredUser()
{
    cout<<"Destructor Executed"<<endl;
}

```

8.2. Feedback.



```cpp
#include "FeedbackandReview.h"
#include <iostream>
#include <cstring>
using namespace std;

//Implementing methods
FeedbackandReview::FeedbackandReview()
{
    U_ID = 0;
    F_no = 0;
    strcpy(U_email,"");
    strcpy(feed,"");
    strcpy(U_name,"");
}
FeedbackandReview::FeedbackandReview(int pU_ID, int pF_no, char pU_mail, char pfeed, char pU_name)
{
    U_ID = pU_ID;
    F_no = pF_no;
    strcpy(U_email,pU_mail);
    strcpy(feed,pfeed);
    strcpy(U_name,pU_name);
}
void FeedbackandReview::setUserID(int id)
{

}
void FeedbackandReview::setFeedNo(int fno)
{

}
void FeedbackandReview::setUserEmail(char email)
{

}
void FeedbackandReview::setUserFeed(char feed)
{

}
void FeedbackandReview::setUserName(char name)
{

}
FeedbackandReview::~FeedbackandReview()
{
    cout<<"Destructor Executed"<<endl;
}

```

8.3. Booking.

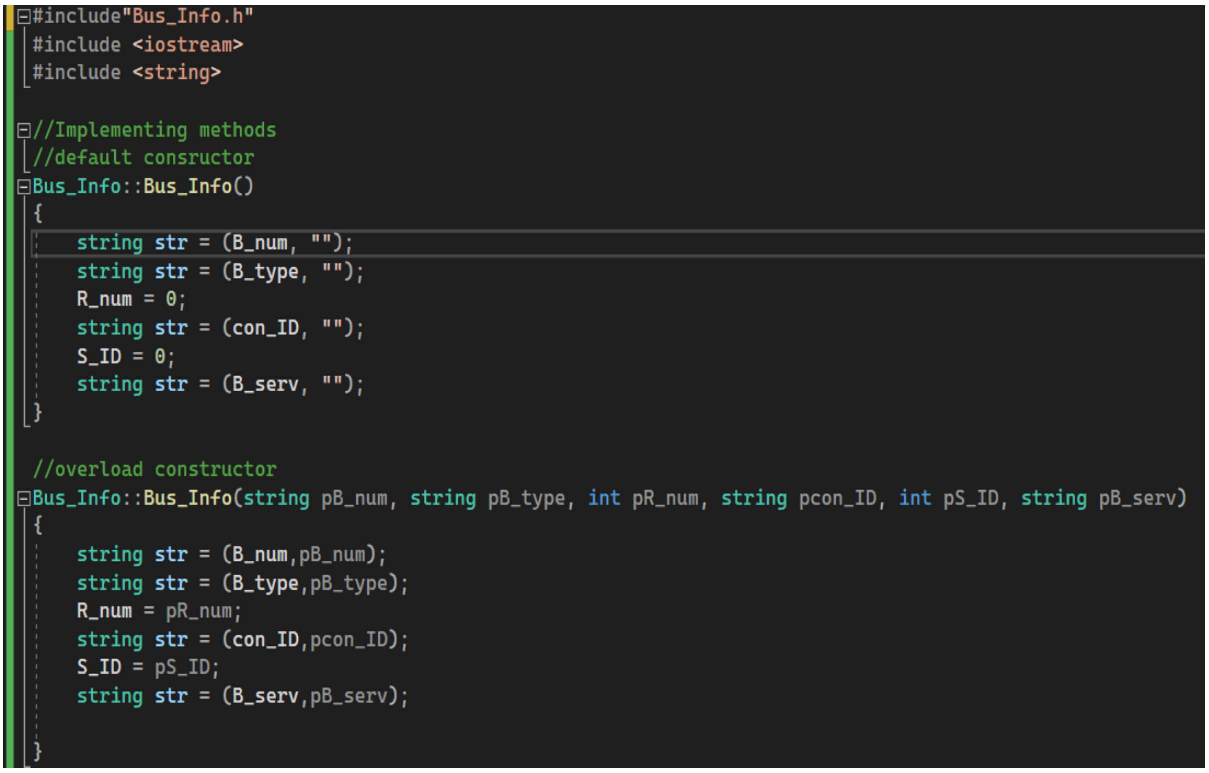


Miscellaneous Files

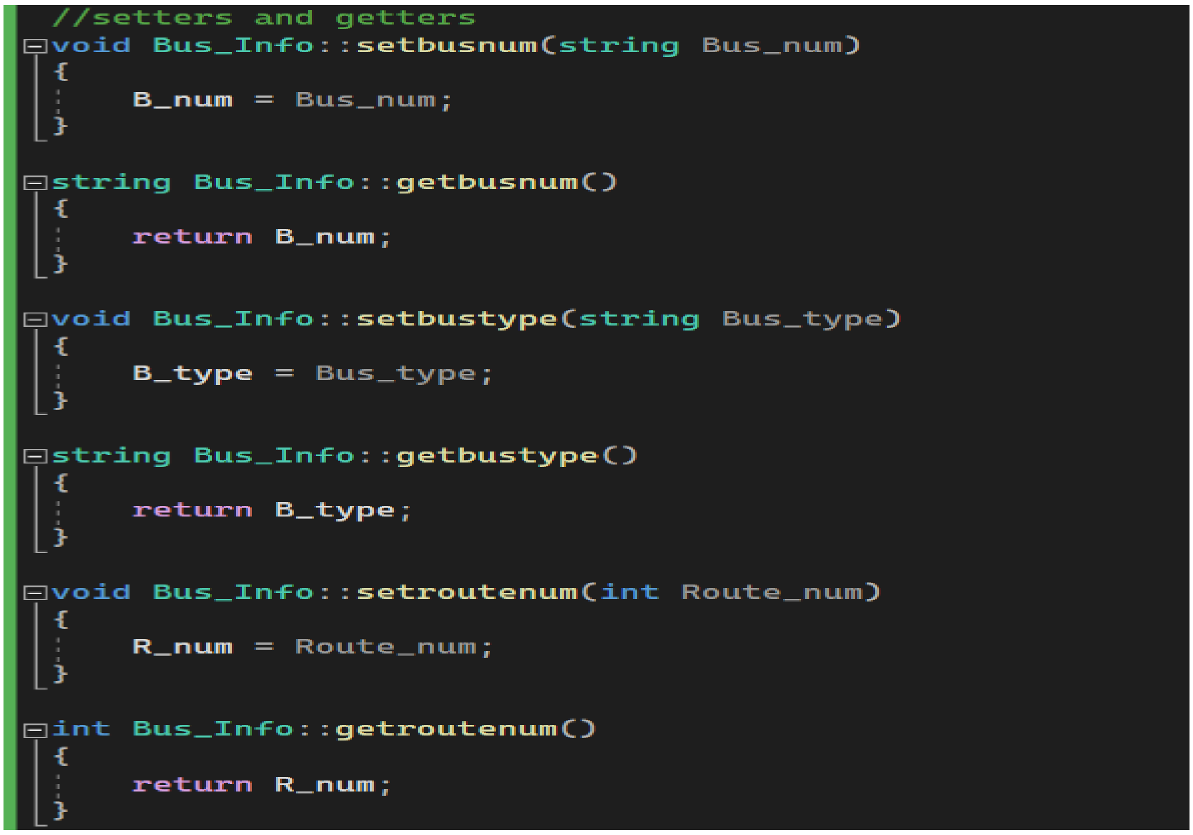
```
//Implementing methods
int Booking::setBookN(int bookN);
{
    B_no = bookN;
}

void Booking::setBusN(string busN);
{
    B_num = busN;
}

void Booking::setRouteNum(int routeN);
{
    R_num = routeN;
}

void Booking::setBusType(string busType);
{
    B_type = busType;
}

void Booking::setRoadtype(string roadType);
{
    R_type = roadType;
}

void Booking::setSPoint(string start);
{
    S_Point = start;
}

void Booking::setDest(string destination);
{
    dest = destination;
}
int Booking::int getSeatNum(int seatN);
{
    S_num = seatN;
}

string Booking::setDate(string date);
{
    B_date = date;
}

string Booking::setTime(string time);
{
    B_time = time;
}

void Booking::setBusServ(string service);
{
    B_serv = service;
}
```

100 %   No issues found

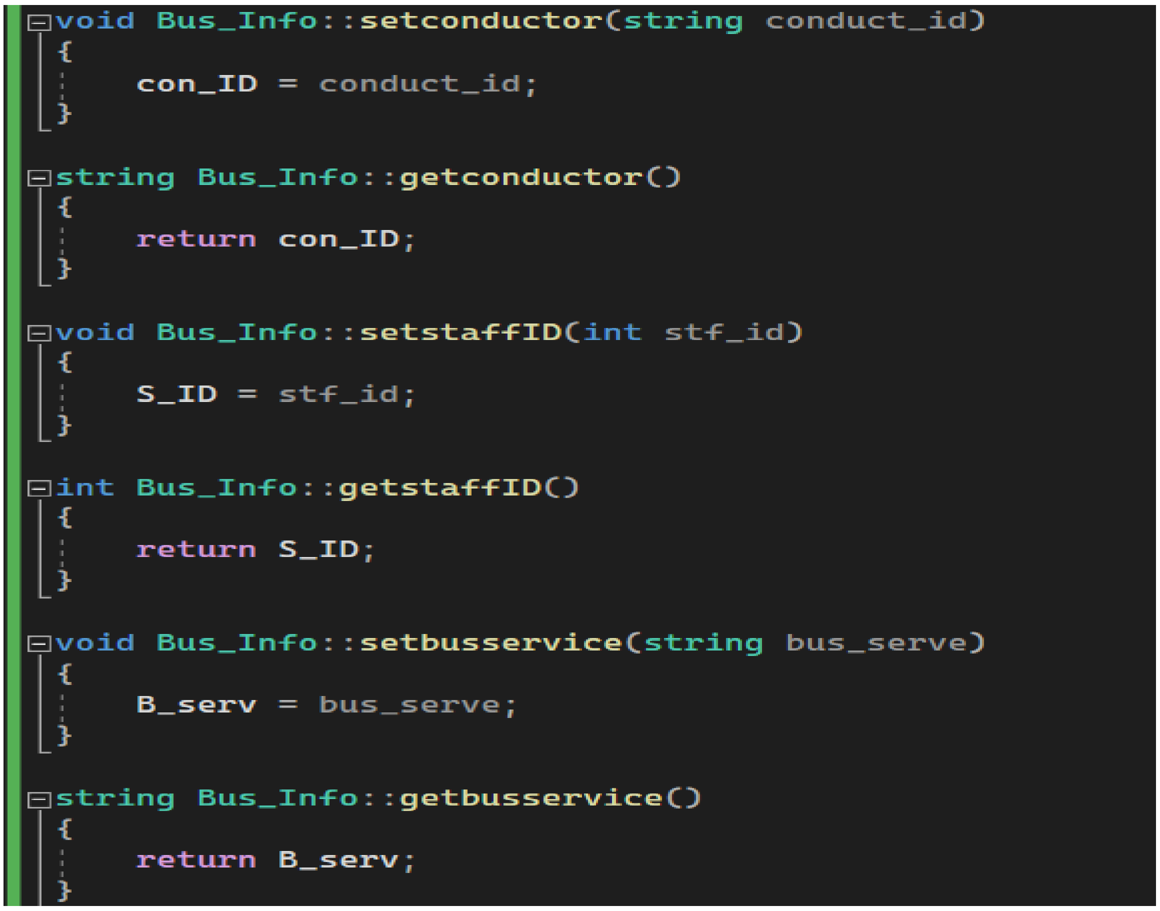
8.4. Bus Info.



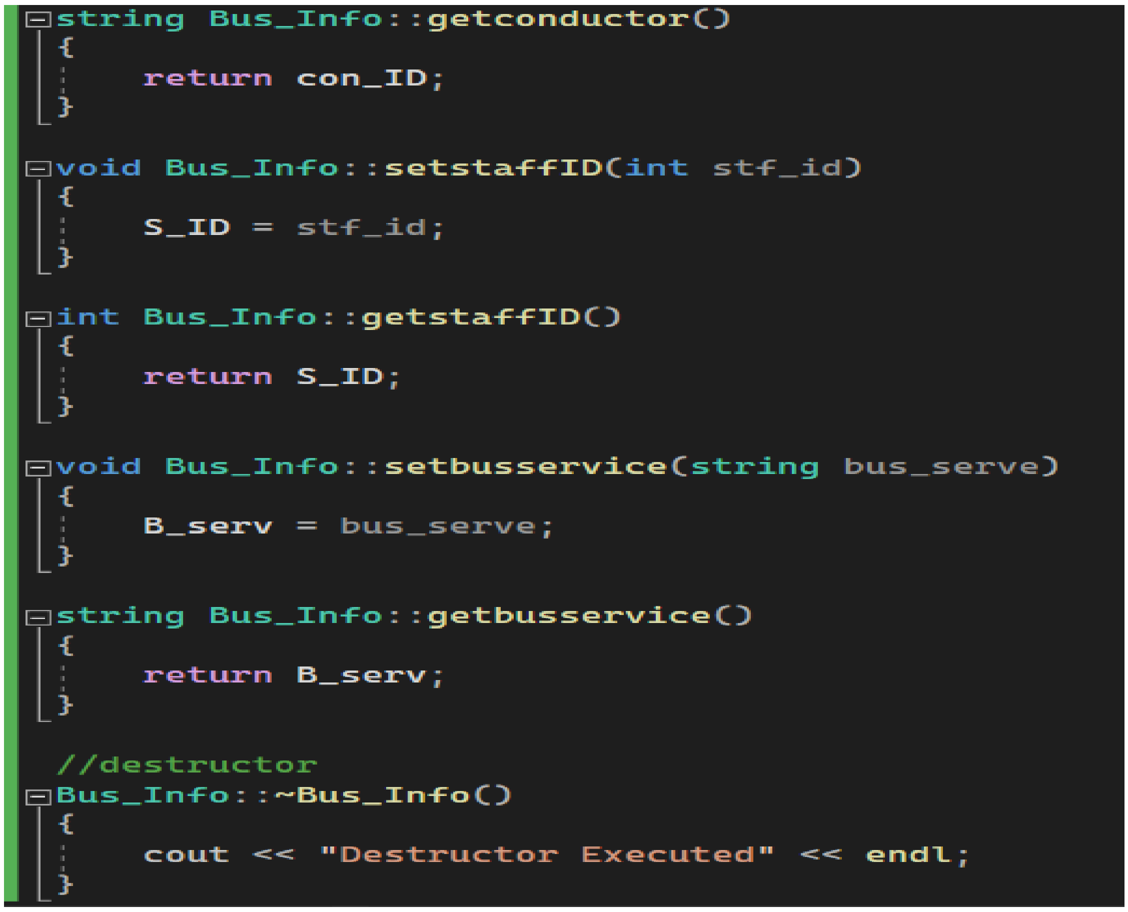
```
#include"Bus_Info.h"
#include <iostream>
#include <string>

//Implementing methods
//default consructor
Bus_Info::Bus_Info()
{
    string str = (B_num, "");
    string str = (B_type, "");
    R_num = 0;
    string str = (con_ID, "");
    S_ID = 0;
    string str = (B_serv, "");
}

//overload constructor
Bus_Info::Bus_Info(string pB_num, string pB_type, int pR_num, string pcon_ID, int pS_ID, string pB_serv)
{
    string str = (B_num,pB_num);
    string str = (B_type,pB_type);
    R_num = pR_num;
    string str = (con_ID,pcon_ID);
    S_ID = pS_ID;
    string str = (B_serv,pB_serv);

}
```



```
//setters and getters
void Bus_Info::setbusnum(string Bus_num)
{
    B_num = Bus_num;
}

string Bus_Info::getbusnum()
{
    return B_num;
}

void Bus_Info::setbustype(string Bus_type)
{
    B_type = Bus_type;
}

string Bus_Info::getbustype()
{
    return B_type;
}

void Bus_Info::setroutenum(int Route_num)
{
    R_num = Route_num;
}

int Bus_Info::getroutenum()
{
    return R_num;
}
```

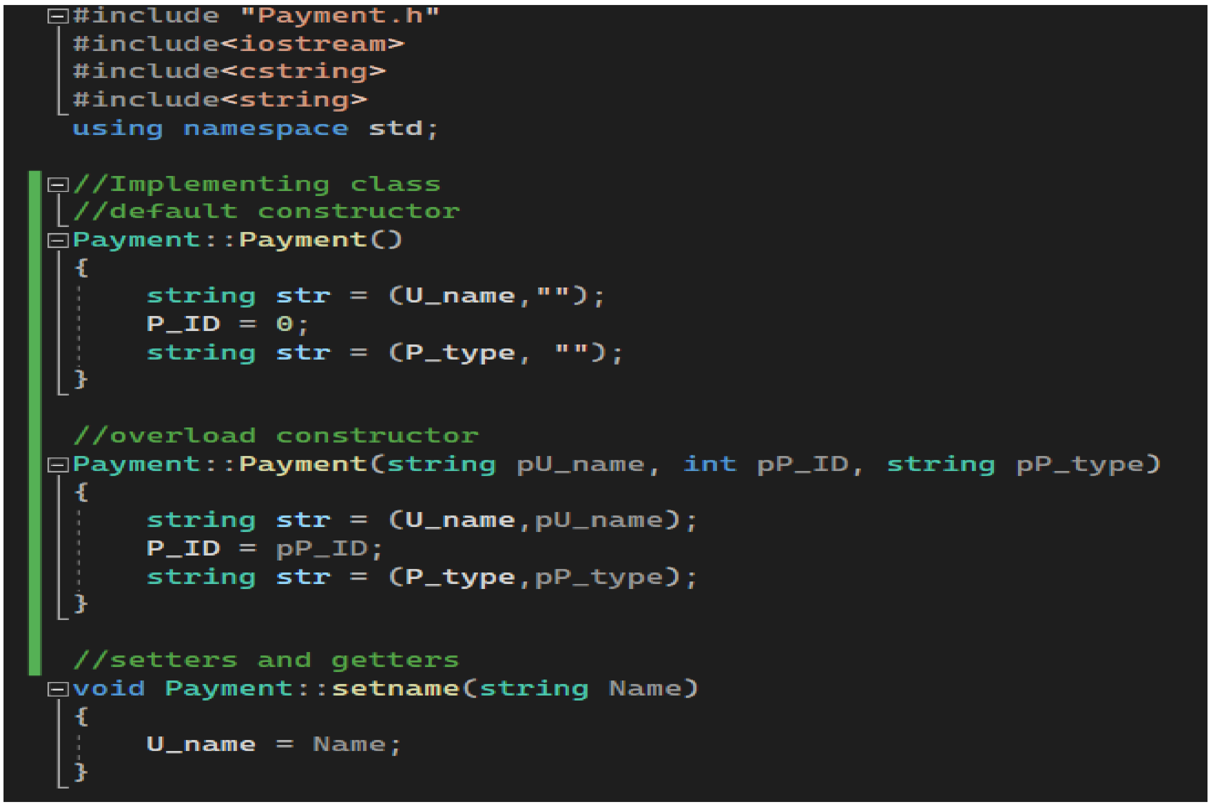


```cpp
void Bus_Info::setconductor(string conduct_id)
{
    con_ID = conduct_id;
}

string Bus_Info::getconductor()
{
    return con_ID;
}

void Bus_Info::setstaffID(int stf_id)
{
    S_ID = stf_id;
}

int Bus_Info::getstaffID()
{
    return S_ID;
}

void Bus_Info::setbusservice(string bus_serve)
{
    B_serv = bus_serve;
}

string Bus_Info::getbusservice()
{
    return B_serv;
}
```

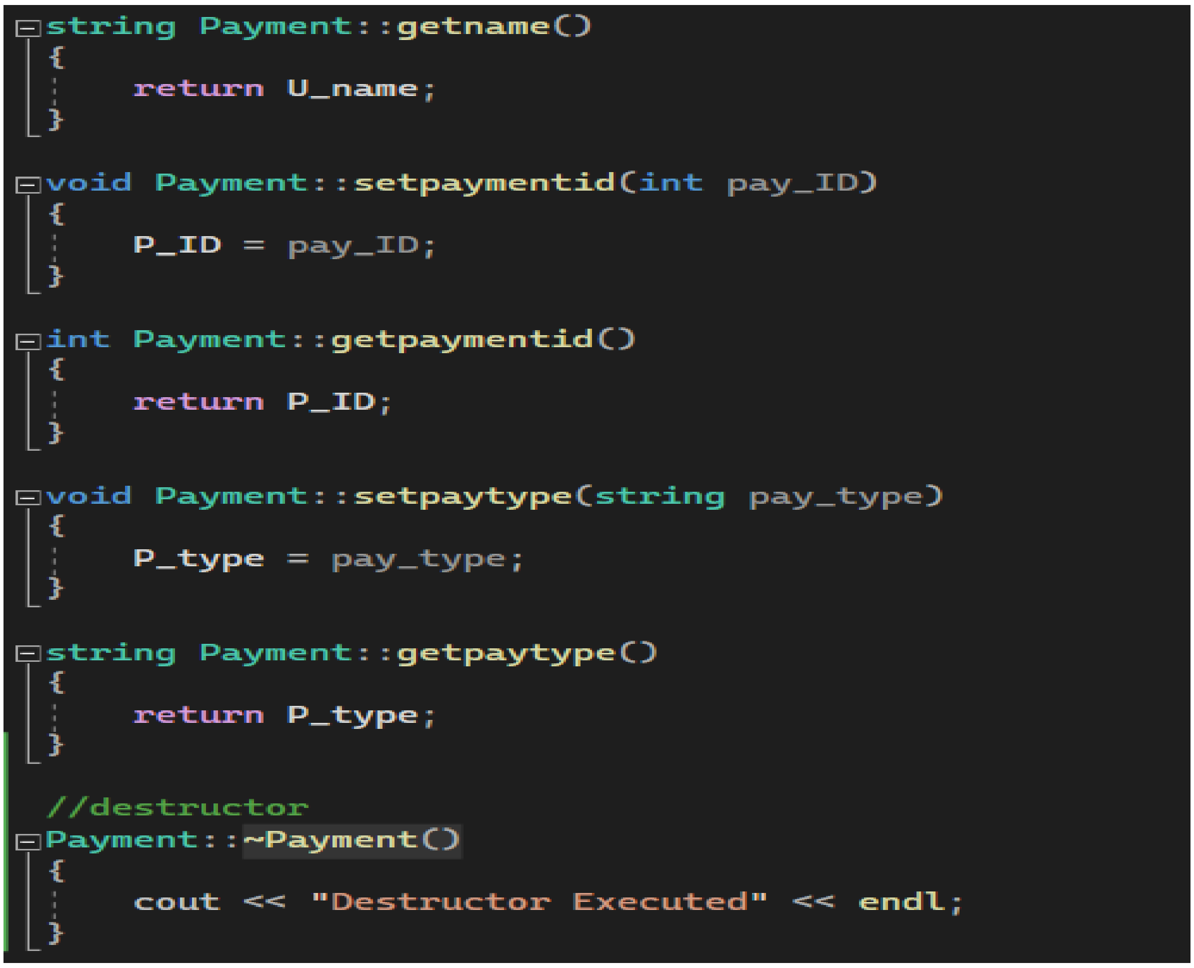


```cpp
string Bus_Info::getconductor()
{
    return con_ID;
}

void Bus_Info::setstaffID(int stf_id)
{
    S_ID = stf_id;
}

int Bus_Info::getstaffID()
{
    return S_ID;
}

void Bus_Info::setbusservice(string bus_serve)
{
    B_serv = bus_serve;
}

string Bus_Info::getbusservice()
{
    return B_serv;
}

//destructor
Bus_Info::~Bus_Info()
{
    cout << "Destructor Executed" << endl;
}
```

8.5. Payment.

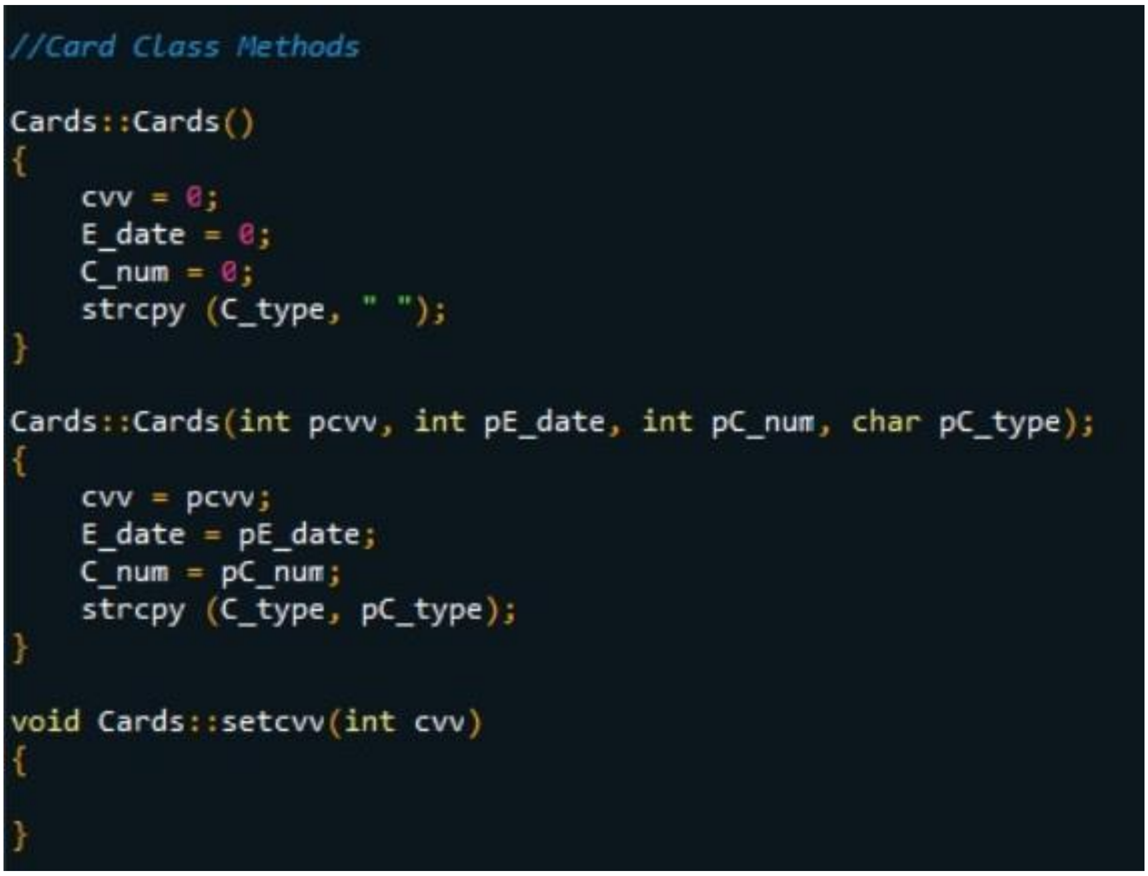


```
#include "Payment.h"
#include<iostream>
#include<cstring>
#include<string>
using namespace std;

//Implementing class
//default constructor
Payment::Payment()
{
    string str = (U_name,"");
    P_ID = 0;
    string str = (P_type, "");
}

//overload constructor
Payment::Payment(string pU_name, int pP_ID, string pP_type)
{
    string str = (U_name,pU_name);
    P_ID = pP_ID;
    string str = (P_type,pP_type);
}

//setters and getters
void Payment::setname(string Name)
{
    U_name = Name;
}
```

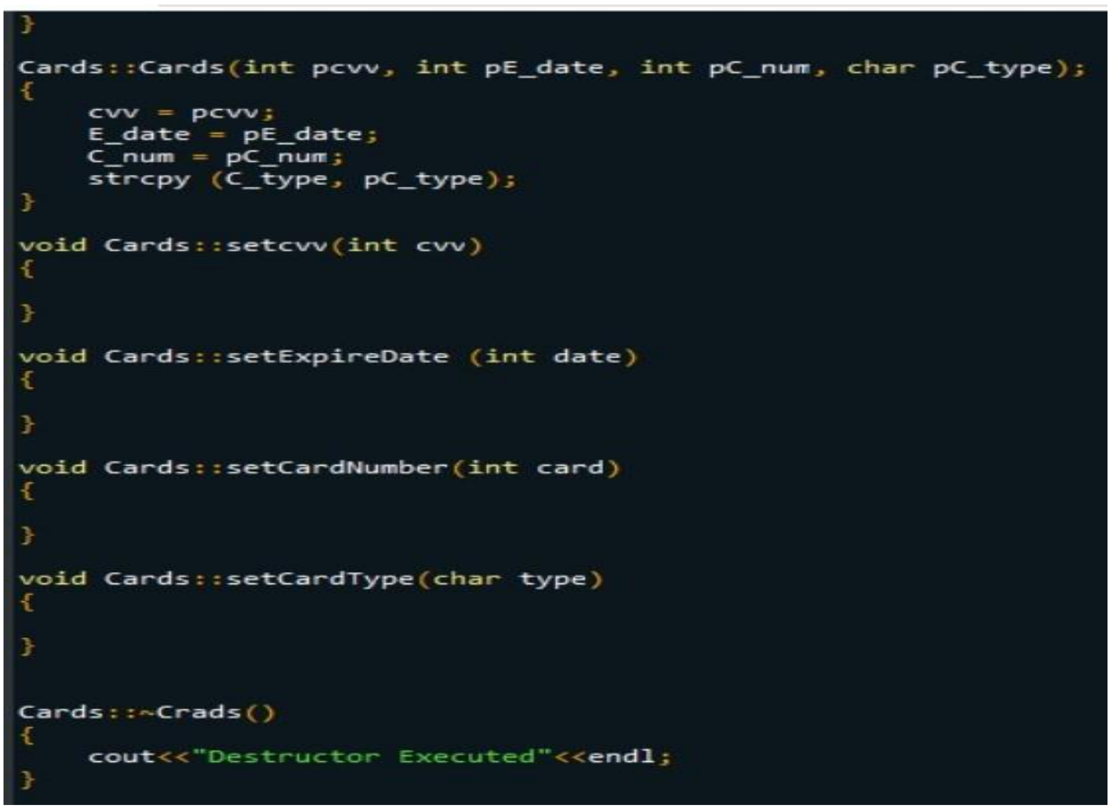


```
string Payment::getname()
{
    return U_name;
}

void Payment::setpaymentid(int pay_ID)
{
    P_ID = pay_ID;
}

int Payment::getpaymentid()
{
    return P_ID;
}

void Payment::setpaytype(string pay_type)
{
    P_type = pay_type;
}

string Payment::getpaytype()
{
    return P_type;
}

//destructor
Payment::~Payment()
{
    cout << "Destructor Executed" << endl;
}
```

8.6. Cards.

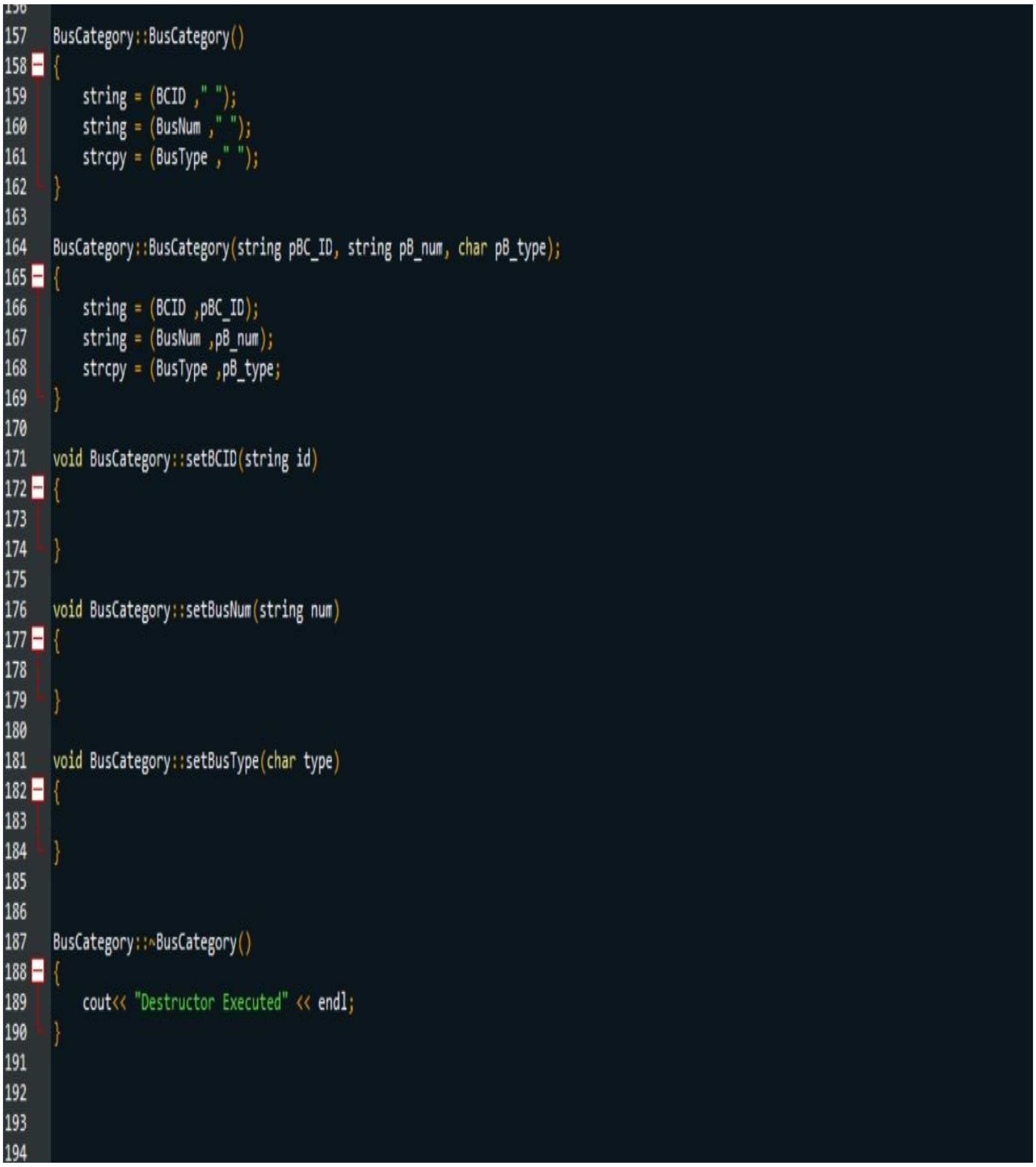


```
//Card Class Methods

Cards::Cards()
{
    cvv = 0;
    E_date = 0;
    C_num = 0;
    strcpy (C_type, " ");
}

Cards::Cards(int pcvv, int pE_date, int pC_num, char pC_type);
{
    cvv = pcvv;
    E_date = pE_date;
    C_num = pC_num;
    strcpy (C_type, pC_type);
}

void Cards::setcvv(int cvv)
{

}
```



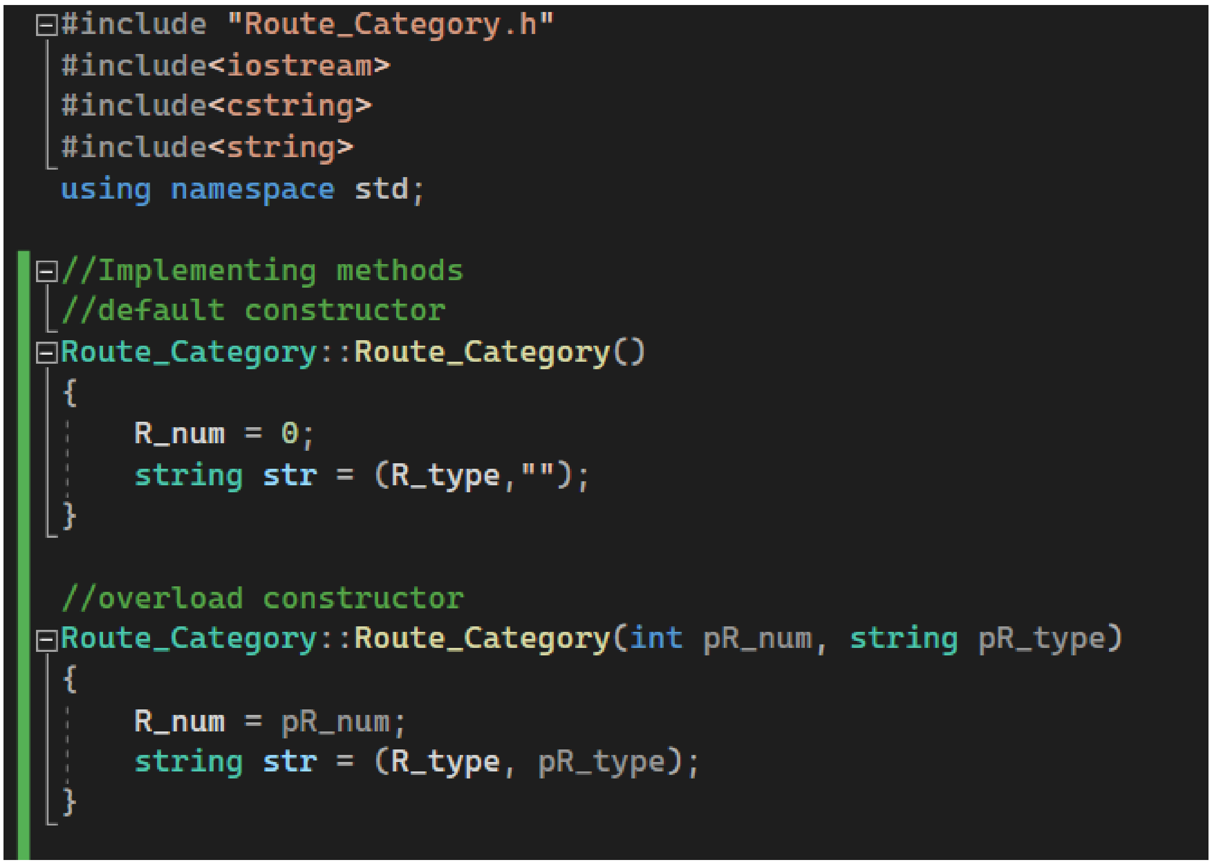
```
}
Cards::Cards(int pcvv, int pE_date, int pC_num, char pC_type);
{
    cvv = pcvv;
    E_date = pE_date;
    C_num = pC_num;
    strcpy (C_type, pC_type);
}
void Cards::setcvv(int cvv)
{

}
void Cards::setExpireDate (int date)
{

}
void Cards::setCardNumber(int card)
{

}
void Cards::setCardType(char type)
{

}

Cards::~Crads()
{
    cout<<"Destructor Executed"<<endl;
}
```

8.7. Bus Category.

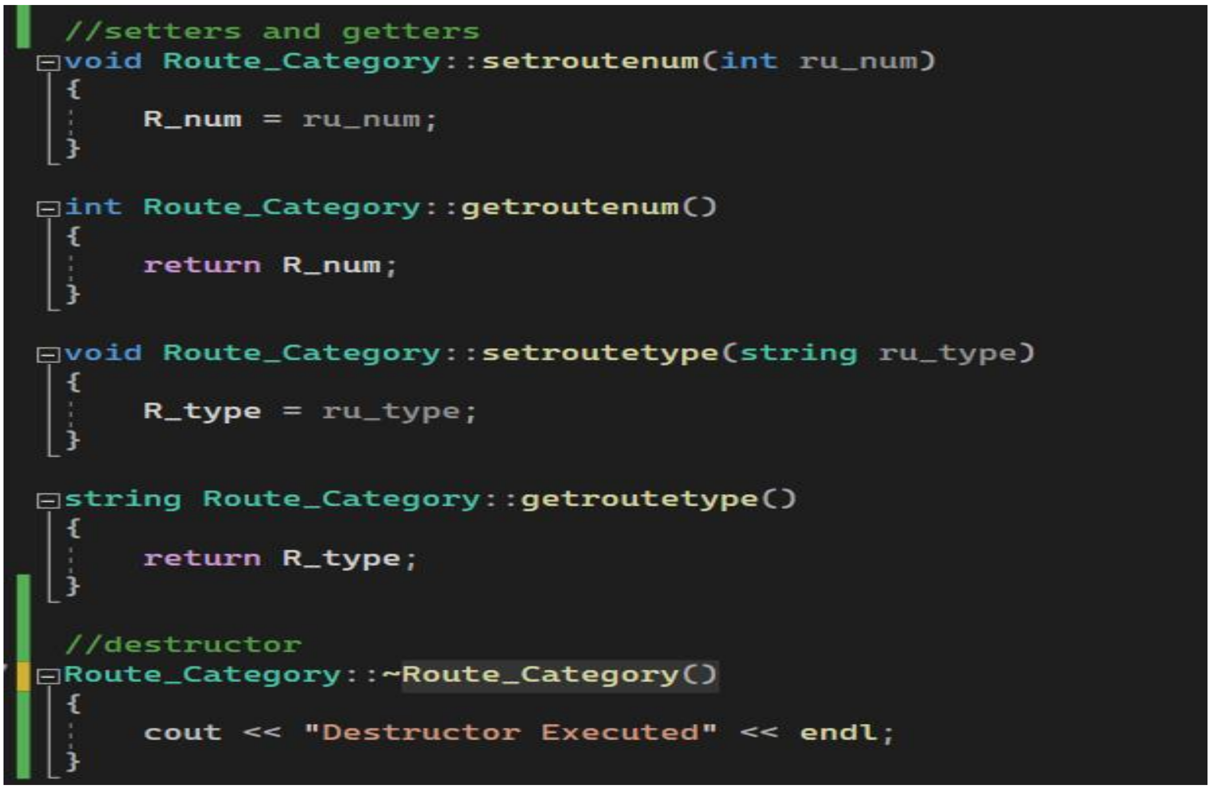


```
BusCategory::BusCategory()
{
    string = (BCID ," ");
    string = (BusNum ," ");
    strcpy = (BusType ," ");
}

BusCategory::BusCategory(string pBC_ID, string pB_num, char pB_type);
{
    string = (BCID ,pBC_ID);
    string = (BusNum ,pB_num);
    strcpy = (BusType ,pB_type;
}

void BusCategory::setBCID(string id)
{

}

void BusCategory::setBusNum(string num)
{

}

void BusCategory::setBusType(char type)
{

}


BusCategory::~BusCategory()
{
    cout<< "Destructor Executed" << endl;
}
```

8.8. Route Category.

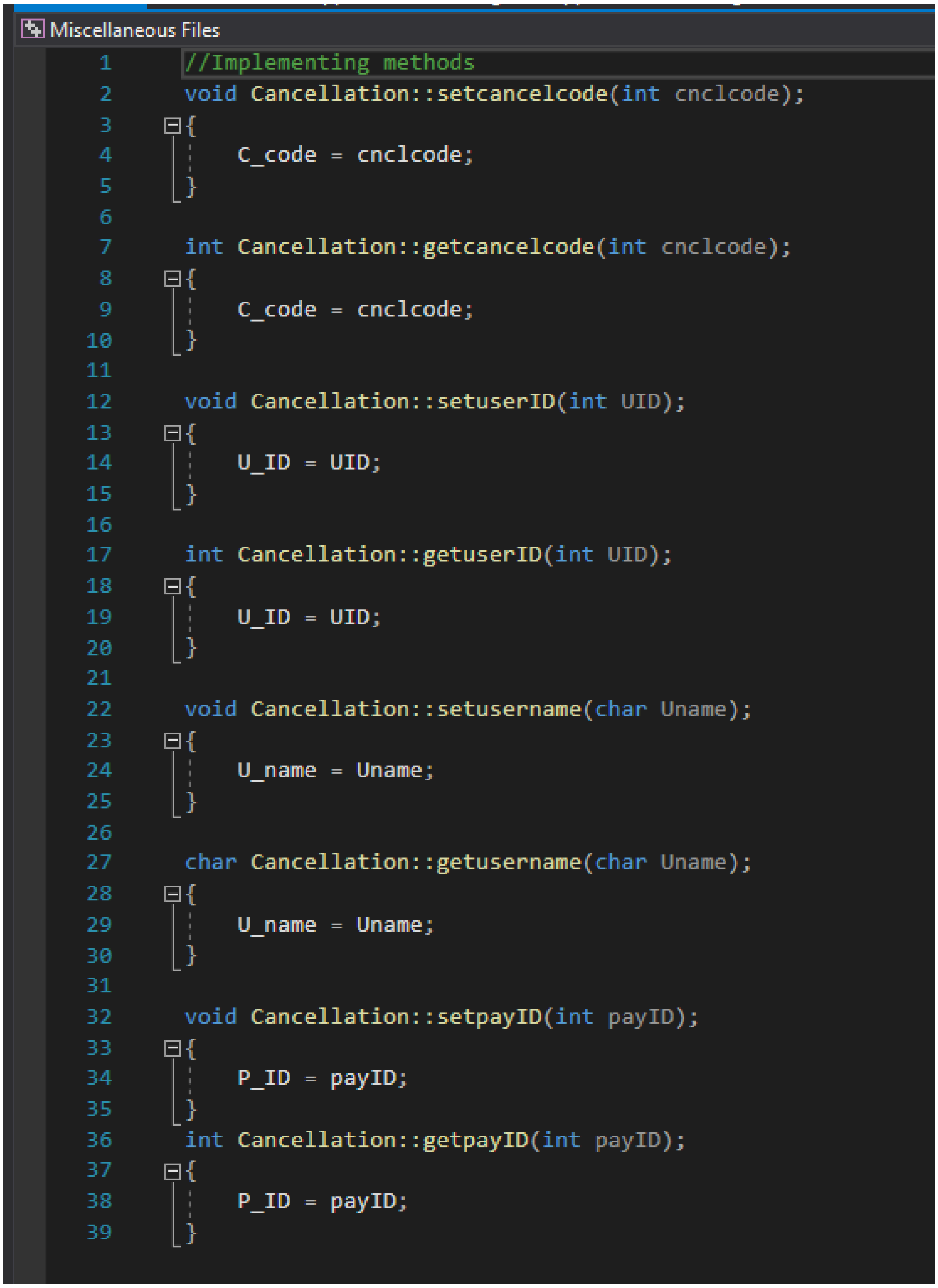


```
#include "Route_Category.h"
#include<iostream>
#include<cstring>
#include<string>
using namespace std;

//Implementing methods
//default constructor
Route_Category::Route_Category()
{
    R_num = 0;
    string str = (R_type,"");
}

//overload constructor
Route_Category::Route_Category(int pR_num, string pR_type)
{
    R_num = pR_num;
    string str = (R_type, pR_type);
}
```



```
//setters and getters
void Route_Category::setroutenum(int ru_num)
{
    R_num = ru_num;
}

int Route_Category::getroutenum()
{
    return R_num;
}

void Route_Category::setroutetype(string ru_type)
{
    R_type = ru_type;
}

string Route_Category::getroutetype()
{
    return R_type;
}

//destructor
Route_Category::~Route_Category()
{
    cout << "Destructor Executed" << endl;
}
```

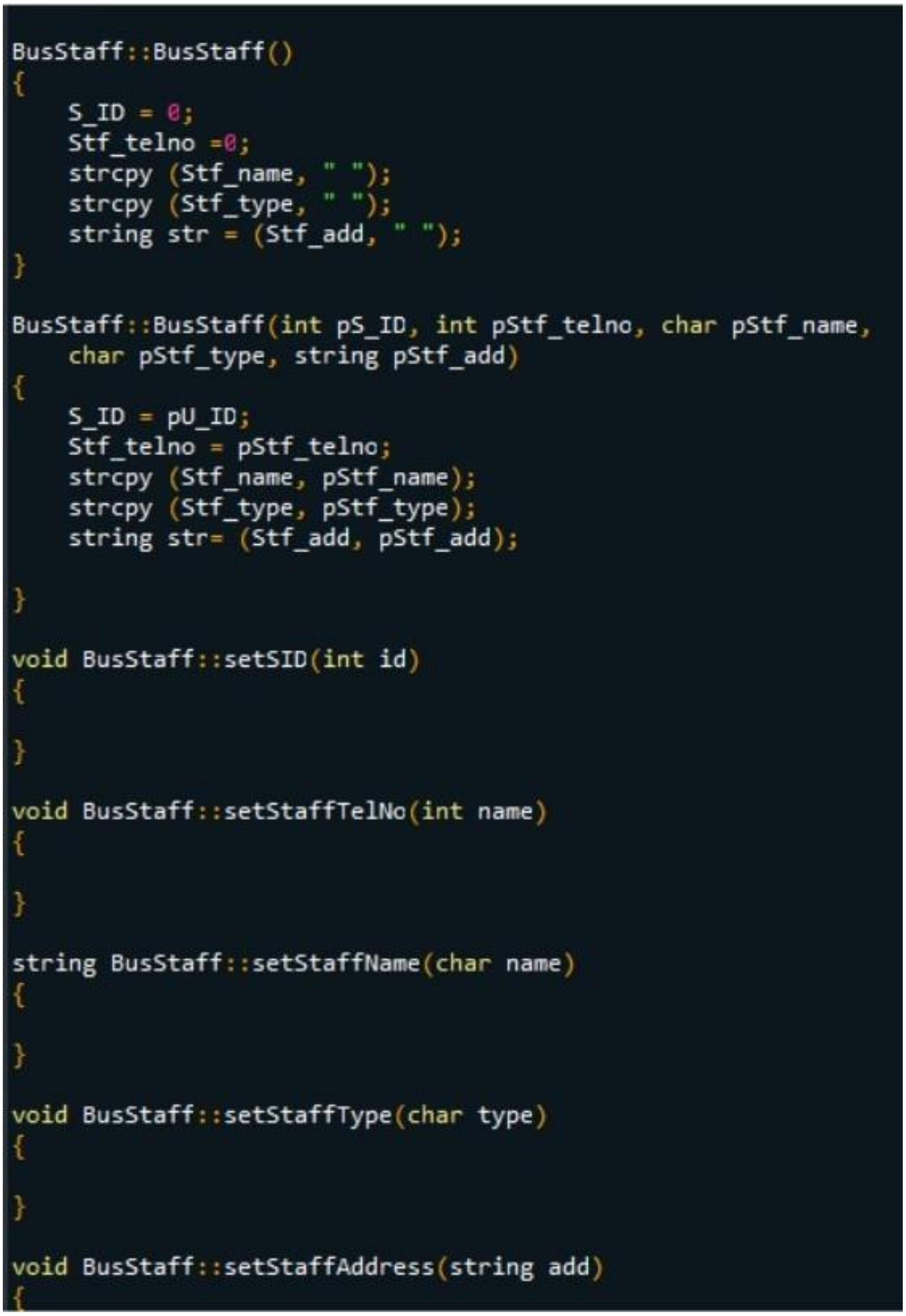
8.9. Cancellation.



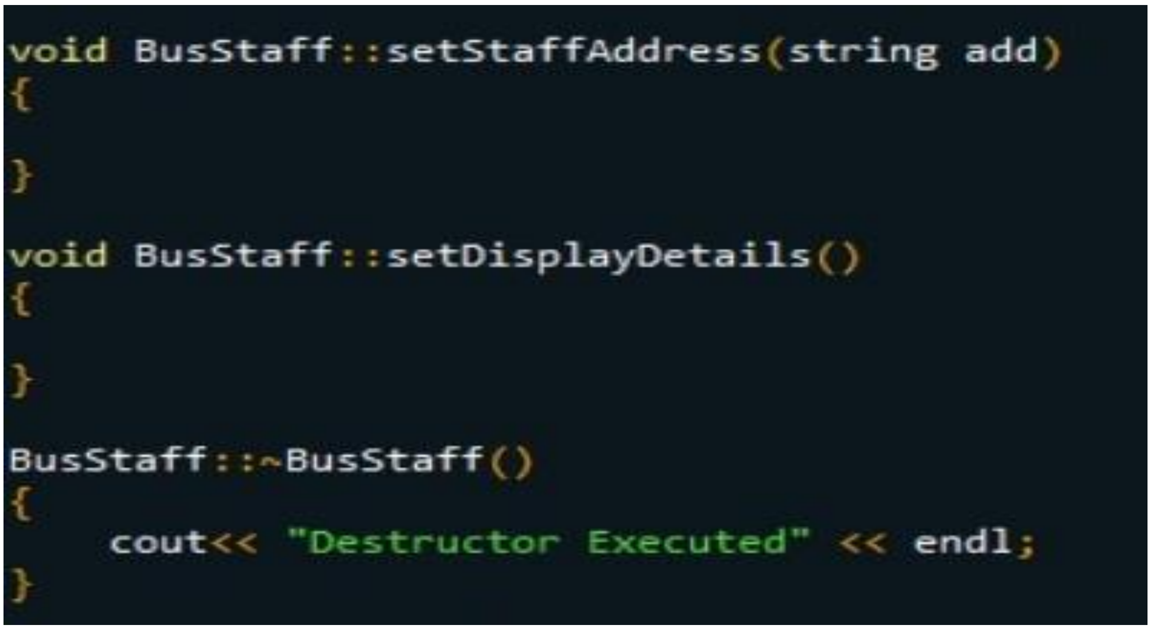
Miscellaneous Files

```
//Implementing methods
void Cancellation::setcancelcode(int cnclcode);
{
    C_code = cnclcode;
}

int Cancellation::getcancelcode(int cnclcode);
{
    C_code = cnclcode;
}

void Cancellation::setuserID(int UID);
{
    U_ID = UID;
}

int Cancellation::getuserID(int UID);
{
    U_ID = UID;
}

void Cancellation::setusername(char Uname);
{
    U_name = Uname;
}

char Cancellation::getusername(char Uname);
{
    U_name = Uname;
}

void Cancellation::setpayID(int payID);
{
    P_ID = payID;
}
int Cancellation::getpayID(int payID);
{
    P_ID = payID;
}
```

8.10. Bus Staff.

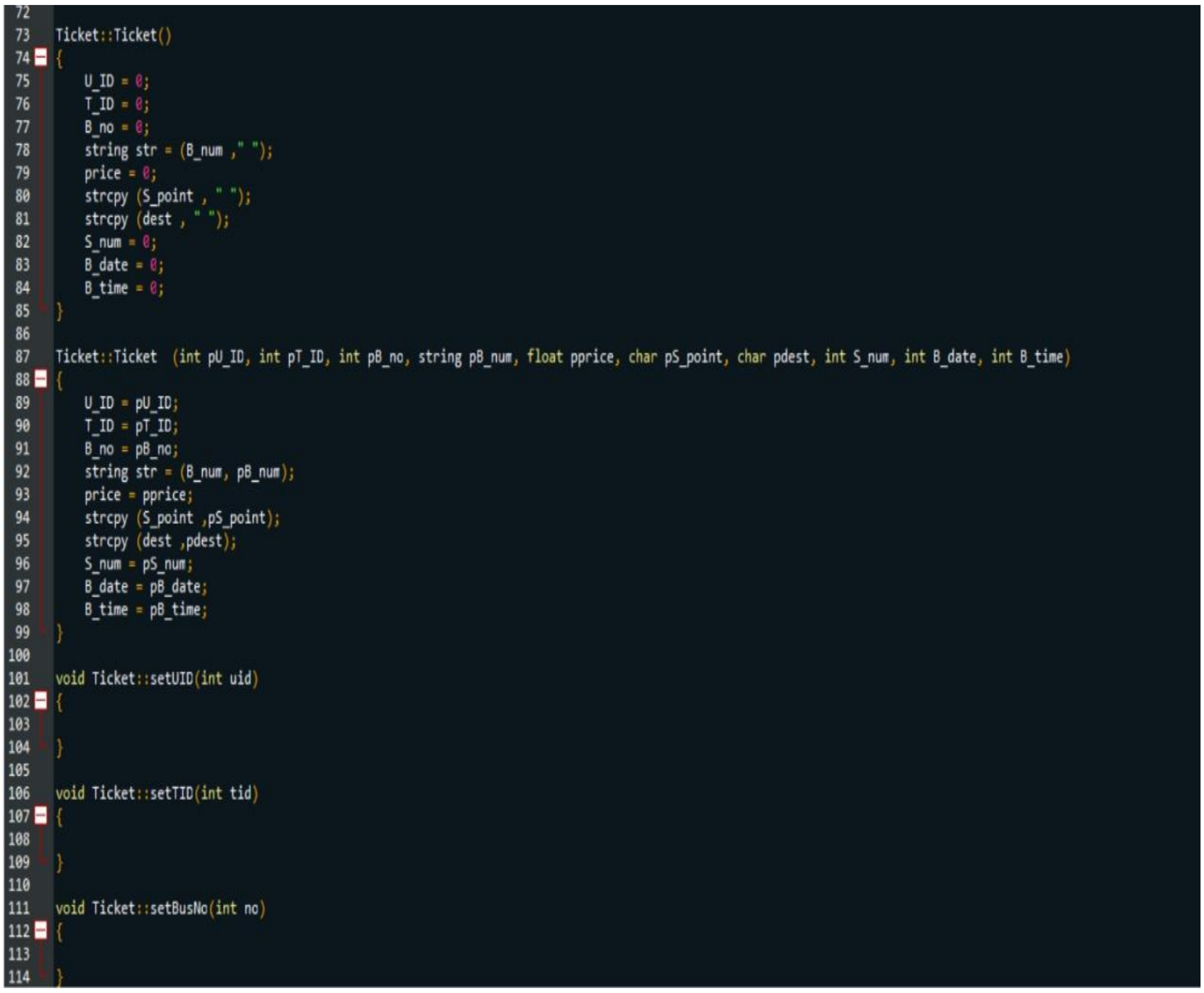


```
BusStaff::BusStaff()
{
    S_ID = 0;
    Stf_telno =0;
    strcpy (Stf_name, " ");
    strcpy (Stf_type, " ");
    string str = (Stf_add, " ");
}

BusStaff::BusStaff(int pS_ID, int pStf_telno, char pStf_name,
    char pStf_type, string pStf_add)
{
    S_ID = pU_ID;
    Stf_telno = pStf_telno;
    strcpy (Stf_name, pStf_name);
    strcpy (Stf_type, pStf_type);
    string str= (Stf_add, pStf_add);

}

void BusStaff::setSID(int id)
{

}

void BusStaff::setStaffTelNo(int name)
{

}

string BusStaff::setStaffName(char name)
{

}

void BusStaff::setStaffType(char type)
{

}

void BusStaff::setStaffAddress(string add)
{
```



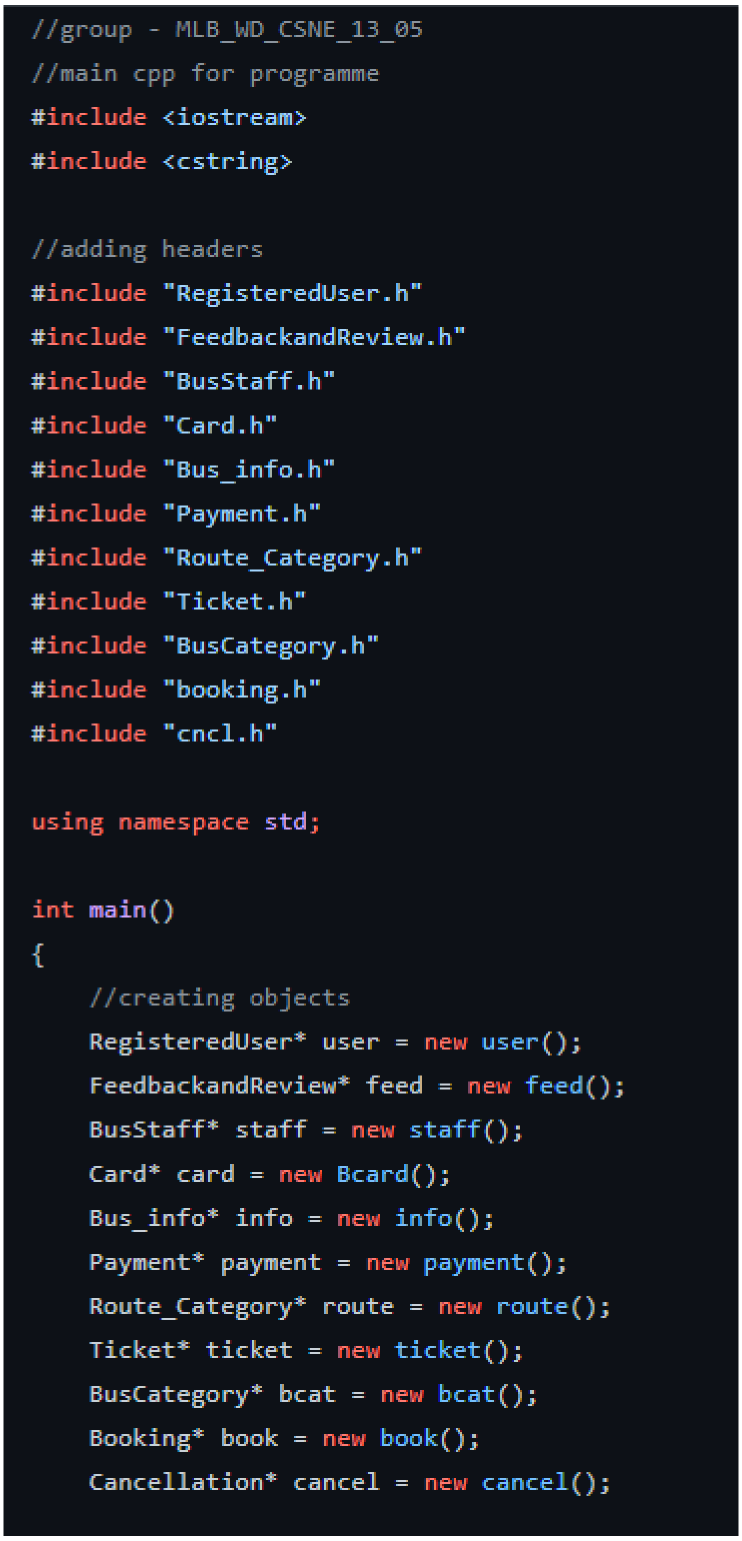
```
void BusStaff::setStaffAddress(string add)
{

}

void BusStaff::setDisplayDetails()
{

}

BusStaff::~BusStaff()
{
     cout<< "Destructor Executed" << endl;
}
```

8.11. Ticket.

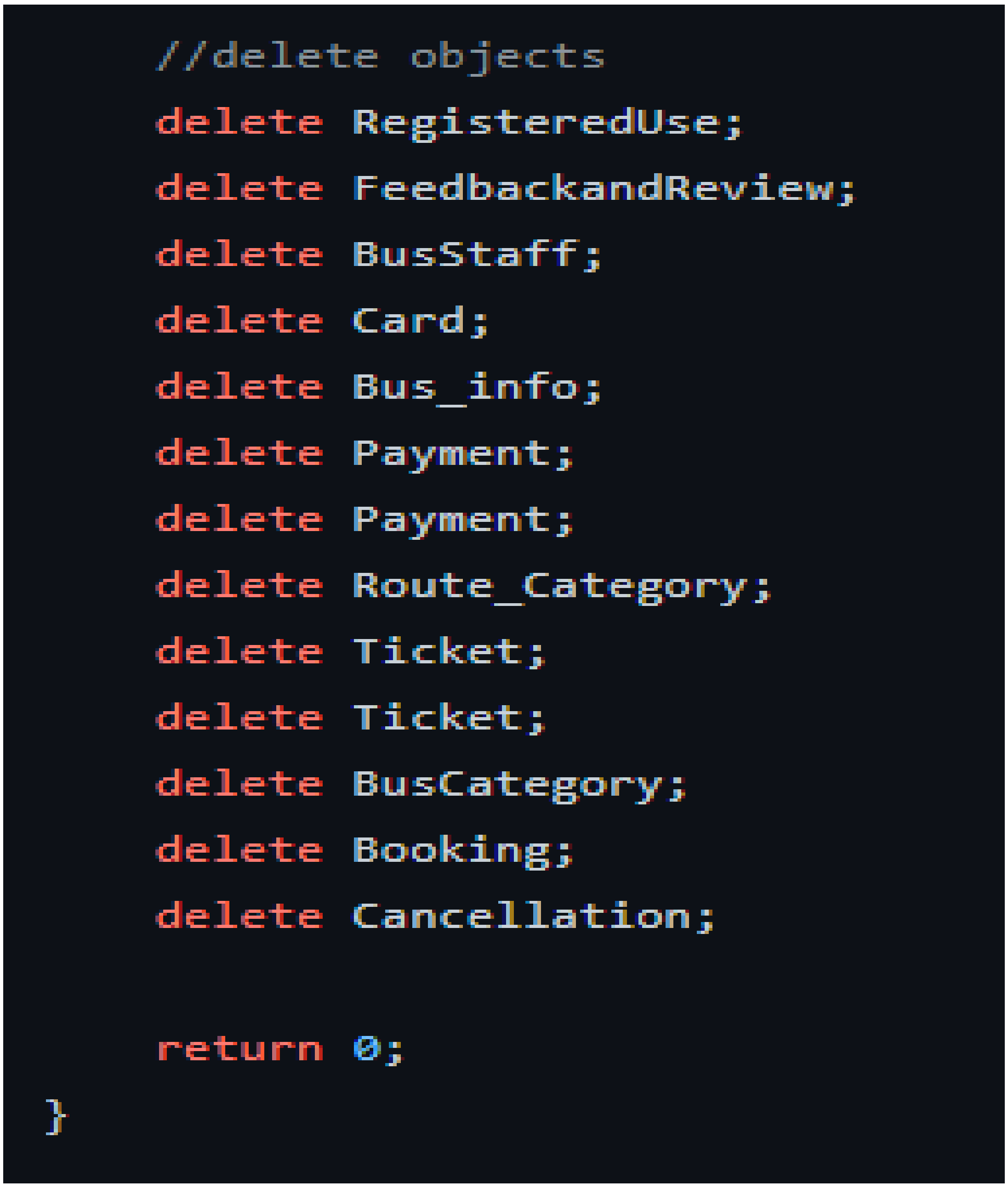


```

Ticket::Ticket()
{
    U_ID = 0;
    T_ID = 0;
    B_no = 0;
    string str = (B_num ," ");
    price = 0;
    strcpy (S_point , " ");
    strcpy (dest , " ");
    S_num = 0;
    B_date = 0;
    B_time = 0;
}

Ticket::Ticket  (int pU_ID, int pT_ID, int pB_no, string pB_num, float pprice, char pS_point, char pdest, int S_num, int B_date, int B_time)
{
    U_ID = pU_ID;
    T_ID = pT_ID;
    B_no = pB_no;
    string str = (B_num, pB_num);
    price = pprice;
    strcpy (S_point ,pS_point);
    strcpy (dest ,pdest);
    S_num = pS_num;
    B_date = pB_date;
    B_time = pB_time;
}

void Ticket::setUID(int uid)
{

}

void Ticket::setTID(int tid)
{

}

void Ticket::setBusNo(int no)
{

}
```

9. Main cpp



```
//group - MLB_WD_CSNE_13_05
//main cpp for programme
#include <iostream>
#include <cstring>

//adding headers
#include "RegisteredUser.h"
#include "FeedbackandReview.h"
#include "BusStaff.h"
#include "Card.h"
#include "Bus_info.h"
#include "Payment.h"
#include "Route_Category.h"
#include "Ticket.h"
#include "BusCategory.h"
#include "booking.h"
#include "cncl.h"

using namespace std;

int main()
{
    //creating objects
    RegisteredUser* user = new user();
    FeedbackandReview* feed = new feed();
    BusStaff* staff = new staff();
    Card* card = new Bcard();
    Bus_info* info = new info();
    Payment* payment = new payment();
    Route_Category* route = new route();
    Ticket* ticket = new ticket();
    BusCategory* bcat = new bcat();
    Booking* book = new book();
    Cancellation* cancel = new cancel();
```



```
    //delete objects
    delete RegisteredUse;
    delete FeedbackandReview;
    delete BusStaff;
    delete Card;
    delete Bus_info;
    delete Payment;
    delete Payment;
    delete Route_Category;
    delete Ticket;
    delete Ticket;
    delete BusCategory;
    delete Booking;
    delete Cancellation;

    return 0;
}
```