Project Report: Ride Sharing Service

Objective:

- To create a ride-sharing system that connects passengers with nearby drivers.

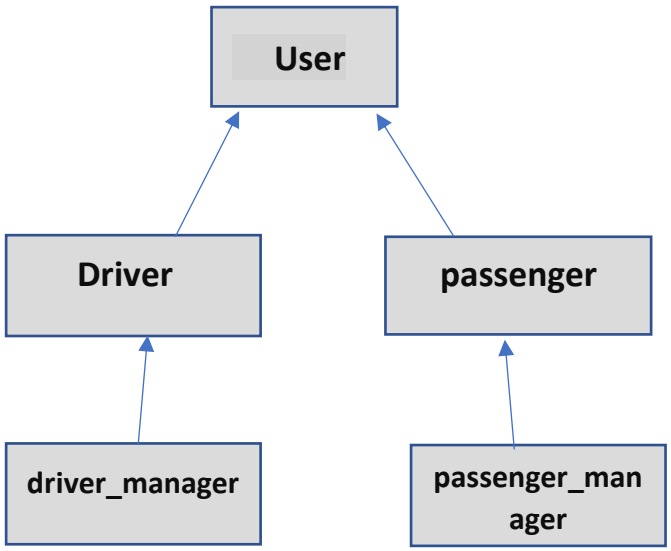
Introduction:

The project simulates a basic ride-sharing platform. It consists of two main user types: passengers and drivers. Passengers can sign up, book rides, login and make payments, while drivers can sign up and offer rides. The program uses object-oriented programming (OOP) principles to manage user data and methods efficiently.

Program Overview:

- The base class “User” contains user information and virtual function
- “Driver” class and “passenger” class inherits from “User” class and constructors have been used to initialize data. These two classes also contain some methods for example to show personal details.
- Besides there are two classes called “passenger_manager” and “driver_manager” which also contains some methods to access data and containers of STL to store data. These two inherits from class “passenger” and “Driver” respectively.
- Some functions are used in the program to determine time, destination using latitudes and longitudes, and to generate random numbers. And for this purpose some extra header files were added.
- File handling has been used to memory management.

UML Diagram:



Code:

```
#include <iostream>
#include <fstream>
#include <algorithm>
#include<cmath>
#include<vector>
#include <unordered_map>
#include<string>
#include <cstdlib>
#include<sstream>
#include <ctime>
using namespace std;
class passenger;     //forward declaration
class Driver;
class passenger_manager;
class driver_manager;

const double EarthRadius= 6371.0;   //function for calculating distance
double degreesToRadians(double degrees)
{
   return degrees*M_PI/180.0;
}
// Function to calculate the distance between two points
double calculateHaversineDistance(double lati1, double long1, double lati2, double long2)
{
   lati1 = degreesToRadians(lati1);
   long1 = degreesToRadians(long1);
   lati2 = degreesToRadians(lati2);
   long2 = degreesToRadians(long2);
   double dLat = lati2-lati1;
   double dLon = long2-long1;
   double a = sin(dLat / 2.0)*sin(dLat/2.0)+cos(lati1)*cos(lati2)*sin(dLon/2.0)*sin(dLon/2.0);
```

```
    double c = 2.0*atan2(sqrt(a),sqrt(1.0-a));
    double distance = EarthRadius*c;
    return distance;
}
int RandomCar(int lower, int upper)     // function for generating random number in a given range
{
    return lower + rand() % (upper-lower + 1);
}

class User   // base class that inherits passenger and driver class
{
public:
    string Name;
    string Phone;
    string Email;
    string Location;

    User() {}
    User(string n,string p, string em,string l):Name(n),Phone(p),Email(em),Location(l) {}
    virtual void ShowProfile()=0;  //pure virtual function for using polymorphism
};
class Driver: public User    //inheritence
{
protected:
    int licence_no;
public:
    string car_model;
    Driver() {}
    Driver(string n,string p, string em,string l,string cm,int licence):User(n,p,em,l)
    {
        car_model=cm;
        licence_no=licence;
    }
    friend class passenger_manager;  //use of friend class for accessing data of other class
    void ShowProfile()override   //for showing personal details of the user
```

```
    {
        cout<<"\n\nDriver's Profile"<<endl;
        cout<<"Name: "<<Name<<endl;
        cout<<"Mobile: "<<Phone<<endl;
        cout<<"Email: "<<Email<<endl;
        cout<<"Location: "<<Location<<endl;
        cout<<"Vehicle: "<<car_model<<endl;
        cout<<"Licence: "<<licence_no<<endl;
    }
    friend void show_driver(driver_manager &chalok, int licence);  //use of friend function for accesing private data of another class

};

class passenger: public User
{
protected:
    string source,destination;
    int shal,mash,tarikh,car;
    string PIN;
public:
    string Payment_method;
    passenger() {}
    passenger(string n,string p, string em,string l,string pm,string pin):User(n,p,em,l)
    {
        Payment_method=pm;
        PIN=pin;
    }
    passenger(string pin,string s, string d,int rc,int yr,int mnth,int dt) //parameterized constructor
    {
        PIN=pin;
        source=s;
        destination=d;
        car=rc;
        shal=yr;
```

```
        mash=mnth;
        tarikh=dt;
    }
    void ShowProfile()override //for showing personal details of the user
    {
        cout<<"Passenger's Profile"<<endl;
        cout<<"Name: "<<Name<<endl;
        cout<<"Mobile: "<<Phone<<endl;
        cout<<"Email: "<<Email<<endl;
        cout<<"Location: "<<Location<<endl;
        cout<<"Payment method: "<<Payment_method<<endl;
    }
    void display_booking()   //for showing details after booking a ride
    {
        cout<<"Booked a trip from "<<source<<" to "<<destination<<" on date "<<shal<<"-"<<mash<<"-"<<tarikh<<endl;
        cout<<"Driver's licence no: "<<car<<endl;
    }
    friend void passenger_status(passenger_manager &jatri, string mobile, string PIN);
};
class passenger_manager:public passenger   // inherited from passenger class
{
public:
    unordered_map<string,string> password; //using stl container for storing password with unique key
    vector<passenger>psngr;
    vector<passenger>book_ride;  //vector container for storing info
    vector<Driver> drv;
    void sign_in_as_passenger();
    void Trip();
    double make_payment();
    friend class driver_manager;
    void addPassword(const string &mobile, const string &pin)  //function for adding password
    {
        password[mobile] = pin;
    }
    string getPassword(const string &mobile)   //function for getting password
```

```
{
    string key = mobile;
    if (password.find(key) != password.end())
    {
        return password[key];
    }
    else
    {
        cout << "No user found" << endl;
        return "";
    }
}
void savePassengerDataToFile()
{
    ofstream outFile;
    outFile.open("passengers.txt");
    if (!outFile)
    {
        cerr << "Unable to open passengers.txt" << endl;
        return;
    }
    else
    {
        for (passenger &p : psngr)
        {
            outFile << p.Name << "," << p.Phone << "," << p.Email << "," << p.Location << "," << p.Payment_method << endl;
        }
        outFile.close();
    }
}
void loadPassengerDataFromFile()
{
    ifstream inFile("passengers.txt");
    if (!inFile)
    {
```

```
            cerr << "No passenger data loaded." << endl;
            return;
        }
        psngr.clear();
        string line;
        while (getline(inFile, line))
        {
            stringstream ss(line);
            string name, phone, email, location, payment_method, pin;
            getline(ss, name, ',');
            getline(ss, phone, ',');
            getline(ss, email, ',');
            getline(ss, location, ',');
            getline(ss, payment_method, ',');
            getline(ss, pin, ',');
            psngr.push_back(passenger(name, phone, email, location, payment_method, pin));
        }
        inFile.close();
    }
};
class driver_manager: public Driver   // inherited from Driver class
{
public:
    vector<Driver>drv;     // vector for storing info
    void sign_in_as_driver();
    void Show_all_driver();
    void initial_drivers();
    void saveDriverDataToFile()
    {
        ofstream outFile("drivers.txt");
        if (!outFile)
        {
            cerr << "Unable to open drivers.txt." << endl;
            return;
        }
```

```
    for (const Driver &d : drv)
    {
        outFile << d.Name << "," << d.Phone << "," << d.Email << "," << d.Location << "," << d.car_model<< endl;
    }
    outFile.close();
  }
  void loadDriverDataFromFile()
  {
    ifstream inFile;
    inFile.open("drivers.txt");
    if (!inFile)
    {
        cerr << "No driver data loaded." << endl;
        return;
    }
    string line;
    while (getline(inFile, line))
    {
        stringstream ss(line);
        string name, phone, email, location, cm;
        int lic;
        getline(ss, name, ',');
        getline(ss, phone, ',');
        getline(ss, email, ',');
        getline(ss, location, ',');
        getline(ss, cm, ',');
        ss >> lic;

        drv.push_back(Driver(name, phone, email, location, cm, lic));
    }
    inFile.close();
  }
};

void passenger_manager::sign_in_as_passenger()      //function for adding passenger info
```

```
{
    string name, mobile, email,loc,pay,pin;
    cout << "Enter Name:";
    cin.ignore();
    getline(cin, name);
    cout << "Enter Mobile No:";
    getline(cin, mobile);
    cout << "Enter email address:";
    getline(cin, email);
    cout << "Enter Location:";
    getline(cin, loc);
    cout << "Enter Payment method:";
    getline(cin, pay);
    cout<<"Set Password: ";
    getline(cin, pin);
    addPassword(mobile,pin);
    psngr.push_back(passenger(name,mobile,email,loc,pay,pin));
    cout << "\n\nRegistration completed!" << endl;
    cout<<"Login Info "<<endl;
    cout<<"Phone Number: "<<mobile<<endl;
    cout<<"Your password is: "<<pin<<endl;
}

void driver_manager::sign_in_as_driver()    //function for adding driver info
{
    string name, mobile, email,loc,car;
    int licence;
    cout << "Enter Name:";
    cin.ignore();
    getline(cin, name);
    cout << "Enter Mobile No:";
    getline(cin, mobile);
    cout << "Enter Email Address:";
    getline(cin, email);
    cout << "Enter Location:";
```

```
    getline(cin, loc);
    cout << "Enter Your Car:";
    getline(cin, car);
    cout << "Enter Your Licence:";
    cin>>licence;
    drv.push_back(Driver(name,mobile,email,loc,car,licence));
    cout << "\nRegistration completed!" << endl;
}
void driver_manager::initial_drivers()   //initialized some driver info manually
{
    drv.push_back(Driver("Faruk Ahmed","01358974522","faruk@gmail.com","Mirpur 10,Dhaka","AM General MV-
1",10001));
    drv.push_back(Driver("Sheikh Miraj","01457220350","miraj@gmail.com","Mirpur 2,Dhaka","Acura MDX",10002));
    drv.push_back(Driver("Md. abu Taleb","01358474527","abu1101@gmail.com","Keraniganj","Tata Indigo ECS",10003));
    drv.push_back(Driver("Chinmoy Saha","01785974572","saha78@gmail.com","SHEIKH MUJIB RD,Chittagong","Acura
TLX",10004));
    drv.push_back(Driver("Md Sumon","01358457865","sumon551@gmail.com","CDA AVENUE","Suzuki Alto 800",10005));
    saveDriverDataToFile();
}
double passenger_manager::make_payment()  //for calculating payment based on distance
{
    double lat1, lat2, lon1, lon2;
    cout << "\nEnter latitude and longitude of your current location: ";
    cin >> lat1 >> lon1;
    cout << "\nEnter latitude and longitude of your destination: ";
    cin >> lat2 >> lon2;

    double range=calculateHaversineDistance(lat1, lon1, lat2, lon2);
    return range*10;
}
void passenger_manager::Trip()   // function for booking a trip
{
    string locate,destin,pass;

    cout<<"Enter password ";
    cin.ignore();
```

```
    getline(cin,pass);
    cout<<"Enter your current location: ";
    getline(cin,locate);
    cout<<"Enter destination: ";
    getline(cin,destin);
    int largestLicence = 0;
    for (const Driver& driver : drv)  //for loop for getting the upper bound of random function(highest licence no.)
    {
        if (driver.licence_no > largestLicence)
        {
            largestLicence = driver.licence_no;
        }
    }
    int lower=10001,upper=largestLicence;
    int random_car=RandomCar(lower,upper);
    cout<<"Making a trip from "<<locate<<" to "<<destin<<endl;
    cout<<"Driver with licence "<<random_car<<" is in proximity to you"<<endl;

    time_t currentTime = time(nullptr);  //getting time on the system(booking time)
    struct tm* currentDate =localtime(&currentTime);
    int year = currentDate->tm_year + 1900;
    int month = currentDate->tm_mon + 1;
    int day = currentDate->tm_mday;

    book_ride.push_back(passenger(pass,locate,destin,random_car,year,month,day));
}

void passenger_status(passenger_manager &jatri, string mobile, string PIN)   //function for logging into profile using
password
{
    auto it = jatri.password.find(mobile);
    if (it != jatri.password.end())
    {
        if (it->second == PIN)
        {
```

```
            for(int i=0; i<jatri.psngr.size(); i++)
            {
               if(jatri.psngr[i].PIN==PIN)
               {
                  jatri.psngr[i].ShowProfile();
               }
            }
         }
         if (it->second == PIN)
         {
            for(int i=0; i<jatri.book_ride.size(); i++)
            {
               if(jatri.book_ride[i].PIN==PIN)
               {
                  jatri.book_ride[i].display_booking();
                  return;
               }
            }
         }
         else
            cout << "Invalid login information!" << endl;
      }
}
void show_driver(driver_manager &chalok, int licence) // function for showing driver's details
{
   int i;
   for (i = 0; i < chalok.drv.size(); i++)
   {
      if (chalok.drv[i].licence_no == licence) // matching passengers with available drivers
      {
         chalok.drv[i].ShowProfile();
         return;
      }
   }
   if(i==chalok.drv.size())
```

```
        cout<<"No Driver Found!"<<endl;
}
void driver_manager::Show_all_driver()
{
    for (int i = 0; i <drv.size(); i++)
    {
        drv[i].ShowProfile();
    }
    cout << endl;
}
int main()
{
    srand(static_cast<unsigned int>(time(nullptr)));
    driver_manager chalok;
    passenger_manager jatri;
    chalok.initial_drivers();
    jatri.drv = chalok.drv;
    jatri.loadPassengerDataFromFile();
    chalok.loadDriverDataFromFile();
    int choice;
    do
    {
        cout << "\n\n----------------------------------------------------------------------------------------------------------------------------------------" <<
endl;
        cout << "                                        RIDE SHARE                " << endl;
        cout << "----------------------------------------------------------------------------------------------------------------------------------------" <<
endl;
        cout << "\n   1. Sign up as passenger              2. Register as driver        3. Show driver list             4. Request a trip" <<
endl;
        cout << "   5. Login to passenger's profile       6. Show my driver           7. Make payment                 8. Leave" <<
endl<<endl;;
        cin >> choice;
        switch(choice)
        {
        case 1:
        {
            jatri.sign_in_as_passenger();
```

```
    break;
}
case 2:
{
    chalok.sign_in_as_driver();
    break;
}
case 3:
{
    chalok.Show_all_driver();
    break;
}
case 4:
{
    jatri.Trip();
    break;
}
case 5:
{
    string pswrd,phn;
    cout<<"Login"<<endl;
    cout<<"Enter mobile no: ";
    cin.ignore();
    getline(cin,phn);
    cout<<"Enter password: ";
    getline(cin,pswrd);
    passenger_status(jatri,phn,pswrd);
    break;
}
case 6:
{
    int lcn;
    cout<<"Enter driver's licence number: ";
    cin>>lcn;
    show_driver(chalok,lcn);
```

```
            break;
        }
        case 7:
        {
            double payment=jatri.make_payment();
            cout<<"\nYou have to pay "<<payment<<" taka."<<endl;
            break;
        }
        case 8:
        {
            cout<<"Leaving app..."<<endl;
            break;
        }
        }
    }
    while(choice!=8);
    jatri.savePassengerDataToFile();
    chalok.saveDriverDataToFile();
    return 0;
}
```

Output:

```
------------------------------------------------------------------------------------------------------------------------

                                        RIDE SHARE

------------------------------------------------------------------------------------------------------------------------


  1. Sign up as passenger       2. Register as driver     3. Show driver list       4. Request a trip
  5. Login to passenger's profile    6. Show my driver       7. Make payment          8. Leave
1
Enter Name: Shormi Ghosh
Enter Mobile No:01587946525
Enter email address:shormi111@gmail.com
Enter Location:khulna
Enter Payment method:bkash
Set Password: 1234ag
```

Registration completed!

Login Info

Phone Number: 01587946525

Your password is: 1234ag

RIDE SHARE

---

1. Sign up as passenger 2. Register as driver 3. Show driver list 4. Request a trip

5. Login to passenger's profile 6. Show my driver 7. Make payment 8. Leave

1

Enter Name:Tareq Rahman

Enter Mobile No:01919320102

Enter email address:rahman@gmail.com

Enter Location:mirpur

Enter Payment method:bkash

Set Password: 8520a

Registration completed!

Login Info

Phone Number: 01919320102

Your password is: 8520a

RIDE SHARE

---

1. Sign up as passenger 2. Register as driver 3. Show driver list 4. Request a trip

5. Login to passenger's profile 6. Show my driver 7. Make payment 8. Leave

2

Enter Name:Nazim

Enter Mobile No:0124578989856

Enter Email Address:nazim@gmail.com

Enter Location:keraniganj

Enter Your Car:Alto 800

Enter Your Licence:10006

Registration completed!

RIDE SHARE

---

1. Sign up as passenger 2. Register as driver 3. Show driver list 4. Request a trip

5. Login to passenger's profile 6. Show my driver 7. Make payment 8. Leave

3

Driver's Profile

Name: Faruk Ahmed

Mobile: 01358974522

Email: faruk@gmail.com

Location: Mirpur 10,Dhaka

Vehicle: AM General MV-1

Licence: 10001

Driver's Profile

Name: Sheikh Miraj

Mobile: 01457220350

Email: miraj@gmail.com

Location: Mirpur 2,Dhaka

Vehicle: Acura MDX

Licence: 10002

Driver's Profile

Name: Md. abu Taleb

Mobile: 01358474527

Email: abu1101@gmail.com

Location: Keraniganj

Vehicle: Tata Indigo ECS

Licence: 10003

Driver's Profile

Name: Chinmoy Saha

Mobile: 01785974572

Email: saha78@gmail.com

Location: SHEIKH MUJIB RD,Chittagong

Vehicle: Acura TLX

Licence: 10004

Driver's Profile

Name: Md Sumon

Mobile: 01358457865

Email: sumon551@gmail.com

Location: CDA AVENUE

Vehicle: Suzuki Alto 800

Licence: 10005

Driver's Profile

Name: Nazim

Mobile: 0124578989856

Email: nazim@gmail.com

Location: keraniganj

Vehicle: Alto 800

Licence: 10006

RIDE SHARE

---

1. Sign up as passenger 2. Register as driver 3. Show driver list 4. Request a trip
5. Login to passenger's profile 6. Show my driver 7. Make payment 8. Leave

4

Enter password 8520a

Enter your current location: Dhaka

Enter destination: Chittagong

Making a trip from Dhaka to Chittagong

Driver with licence 10004 is in proximity to you

RIDE SHARE

---

1. Sign up as passenger 2. Register as driver 3. Show driver list 4. Request a trip

5. Login to passenger's profile 6. Show my driver 7. Make payment 8. Leave

6

Enter driver's licence number: 10004

Driver's Profile

Name: Chinmoy Saha

Mobile: 01785974572

Email: saha78@gmail.com

Location: SHEIKH MUJIB RD,Chittagong

Vehicle: Acura TLX

Licence: 10004

RIDE SHARE

---

1. Sign up as passenger 2. Register as driver 3. Show driver list 4. Request a trip

5. Login to passenger's profile 6. Show my driver 7. Make payment 8. Leave

7

Enter latitude and longitude of your current location: 23.8041 90.4152

Enter latitude and longitude of your destination: 22.3752 91.8349

You have to pay 2152.47 taka.

RIDE SHARE

```
  1. Sign up as passenger      2. Register as driver      3. Show driver list      4. Request a trip
  5. Login to passenger's profile      6. Show my driver      7. Make payment      8. Leave
5
Login
Enter mobile no: 01919320102
Enter password: 8520a
Passenger's Profile
Name: Tareq Rahman
Mobile: 01919320102
Email: rahman@gmail.com
Location: mirpur
Payment method: bkash
Booked a trip from Dhaka to Chittagong on date 2023-9-24
Driver's licence no: 10004

------------------------------------------------------------------------------------------------------------------------

                                                RIDE SHARE

------------------------------------------------------------------------------------------------------------------------

  1. Sign up as passenger      2. Register as driver      3. Show driver list      4. Request a trip
  5. Login to passenger's profile      6. Show my driver      7. Make payment      8. Leave
8
Leaving app...
```

Explanation of Code:

1. **Classes and Objects:** The program defines several classes, such as “User”, “Driver”, “passenger”, “passenger_manager”, and “driver_manager”, each representing a different entity within the ride-sharing system. Objects of these classes are created to store and manage user data.

2. **Inheritance:** The “Driver” and “passenger” classes inherit from the “User” class, inheriting its properties and methods. “passenger_manager” and “driver_manager” which inherits from class “passenger” and “Driver” (indirectly from ”User” class) respectively. These inheritances create a hierarchy of classes.

3. **Polymorphism:** Polymorphism is achieved through the use of pure virtual functions in the “User” class. Both “Driver” and “passenger” classes implement the “ShowProfile()” function, allowing them to provide their own implementations for displaying user profiles.

4. **Friend Classes and Functions:** Friend classes and functions were used to access private data of other classes. For example, “friend class passenger_manager” , “friend void show_driver(driver_manager &chalok, int licence)” , “friend void passenger_status(passenger_manager &jatri, string mobile, string PIN)” allow certain functions to access private members of “passenger_manager” and “driver_manager” respectively.

5. **Encapsulation:** Data encapsulation is employed by declaring member variables as private in the classes and providing public member functions to access and modify the data.

6. **Abstraction:** The ShowProfile() function declared in the User class is a pure virtual function. It means that it has no implementation in the base class and must be overridden in derived classes (passenger and Driver). This enforces the concept of abstraction by ensuring that every derived class must provide its own implementation of ShowProfile().

7. **Constructor Overloading:** The program demonstrates constructor overloading in both the “Driver” and “passenger” classes, allowing for the creation of objects with different sets of parameters.

8. **Standard Template Library:** STL like Vector, Unordered_Map is used to store data in the program. These containers make code more easier and maintainable.

User Experience:

- Passengers can sign up, providing their personal information and setting a password for their accounts.

- Drivers can sign up, providing their details such as name, mobile number, email, location, car model, and license number.

- Passengers can book rides by specifying their current location and destination. The system matches them with available drivers based on proximity.

- Passengers can view driver profiles, which include their personal information, car details using driver’s license number.

- Users can log in using their mobile number and password and view their profiles and booking details.

Conclusion:

The Ride Sharing System mentioned above provides a user-friendly interface for passengers and drivers to interact with the ride-sharing platform. This project showcases how OOP principles like inheritance, polymorphism, abstraction, object, class and encapsulation can be applied to create a real-world application. Further improvements could include integrating databases for data persistence, exception handling and expanding the system's features for a wider ride-sharing experience.