Object Oriented Programming

Course Project Report on

"CabConnect - CPP Based Cab Sharing Application"

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
BACHELOR OF

TECHNOLOGY IN

"Computer Science and Engineering (Artificial Intelligence)"

OF

VISHWAKARMA INSTITUTE OF TECHNOLOGY

Savitribai Phule Pune University

Shantanu Kaute (1252090007)
Apurv Saktepar (1252090011)
Nisha Pragane (1252090013)
Sohaa Jamadar (1252090024)

UNDER THE GUIDANCE OF
Mrs. Shubhangi D. Kamble

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (ARTIFICIAL INTELLIGENCE)

BANSILAL RAMNATH AGARWAL CHARITABLE TRUST'S

VISHWAKARMA INSTITUTE OF TECHNOLOGY

(An Autonomous Institute affiliated to Savitribai Phule Pune University)

2025 - 2026

BANSILAL RAMNATH AGARWAL CHARITABLE TRUST'S

VISHWAKARMA INSTITUTE OF TECHNOLOGY

(An Autonomous Institute affiliated to Savitribai Phule Pune University)

PUNE – 411037

CERTIFICATE

This is to certify that the Course Project titled **"CabConnect - CPP Based Cab Sharing Application"** submitted by **Shantanu Kaute (1252090007)** is in partial fulfillment for the award of Degree of Bachelor of Technology in **Computer Science And Engineering (Artificial Intelligence)** of Vishwakarma Institute of Technology, Savitribai Phule Pune University. This project report is a record of Bonafide work carried out by him/her under my guidance during the academic year 2025-26.

Guide	HOD (CSE-AI)
Mrs. Shubhangi Kamble	**Dr. Nilesh P. Sable**

Place: VIT, Pune

Date:

BANSILAL RAMNATH AGARWAL CHARITABLE TRUST'S

VISHWAKARMA INSTITUTE OF TECHNOLOGY

(An Autonomous Institute affiliated to Savitribai Phule Pune University)

PUNE – 411037

CERTIFICATE

This is to certify that the Course Project titled **"CabConnect - CPP Based Cab Sharing Application"** submitted by **Apurv Saktepar (1252090011)** is in partial fulfillment for the award of Degree of Bachelor of Technology in **Computer Science And Engineering (Artificial Intelligence)** of Vishwakarma Institute of Technology, Savitribai Phule Pune University. This project report is a record of Bonafide work carried out by him/her under my guidance during the academic year 2025-26.

Guide	HOD (CSE-AI)
Mrs. Shubhangi Kamble	**Dr. Nilesh P. Sable**

Place: VIT, Pune

Date:

BANSILAL RAMNATH AGARWAL CHARITABLE TRUST'S

VISHWAKARMA INSTITUTE OF TECHNOLOGY

(An Autonomous Institute affiliated to Savitribai Phule Pune University)

PUNE – 411037

CERTIFICATE

This is to certify that the Course Project titled **"CabConnect - CPP Based Cab Sharing Application"** submitted by **Nisha Pragane (1252090013)** is in partial fulfillment for the award of Degree of Bachelor of Technology in **Computer Science And Engineering (Artificial Intelligence)** of Vishwakarma Institute of Technology, Savitribai Phule Pune University. This project report is a record of bonafide work carried out by him/her under my guidance during the academic year 2025-26.

Guide	HOD (CSE-AI)
Mrs. Shubhangi Kamble	**Dr. Nilesh P. Sable**

Place: VIT, Pune

Date:

BANSILAL RAMNATH AGARWAL CHARITABLE TRUST'S

VISHWAKARMA INSTITUTE OF TECHNOLOGY

(An Autonomous Institute affiliated to Savitribai Phule Pune University)

PUNE – 411037

CERTIFICATE

This is to certify that the Course Project titled **"CabConnect - CPP Based Cab Sharing Application"** submitted by **Sohaa Jamadar (1252090024)** is in partial fulfillment for the award of Degree of Bachelor of Technology in **Computer Science And Engineering (Artificial Intelligence)** of Vishwakarma Institute of Technology, Savitribai Phule Pune University. This project report is a record of bonafide work carried out by him/her under my guidance during the academic year 2025-26.

Guide	HOD (CSE-AI)
Mrs. Shubhangi Kamble	**Dr. Nilesh P. Sable**

Place: VIT, Pune

Date:

Bansilal Ramnath Agarwal Charitable Trust's

Vishwakarma Institute of Technology, Pune-37

Department of Computer Science and Engineering (Artificial Intelligence)

PROJECT DETAILS

Group No : 1

Members:

Roll No.	PRN	Name of Student	Contact No.	Email ID
07	1252090007	Shantanu Kaute	7498829225	shantanu.1252090007@vit.edu
11	1252090011	Apurv Saktepar	7058229202	apurv.1252090011@vit.edu
13	1252090013	Nisha Pragane	9373639198	nisha.1252090013@vit.edu
24	1252090024	Sohaa Jamadar	9975130249	sohaa.1252090024@vit.edu

Academic Year : 2025-2026

Project Title : CabConnect - CPP Based Cab Sharing Application

Project Area : OOP

Internal Guide : Mrs. Shubhangi D. Kamble

Signature of Internal Guide

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to the Department of **Computer Science and Engineering (Artificial Intelligence)** for introducing the subject "**Object-Oriented Programming (OOP)**" in our curriculum. This subject has significantly enhanced my understanding of core programming concepts such as classes, objects, inheritance, polymorphism, and abstraction, thereby strengthening my analytical and problem-solving skills in software development.

I extend my heartfelt thanks to our course teacher **Prof. Shubhangi Kamble Mam** for her insightful teaching, continuous support, and dedicated efforts in simplifying complex programming concepts through practical examples and interactive sessions. I am also deeply thankful to our respected Head of Department **Dr. Nilesh Sable Sir** for his constant encouragement and for fostering a learning environment that promotes creativity, logic, and structured thinking.

Additionally, I appreciate the contribution of our course in-charge for efficiently coordinating the subject, ensuring smooth conduct of lectures and practical sessions, and providing timely guidance throughout the semester. Their commitment to academic excellence has played a vital role in enhancing my understanding of Object-Oriented Programming.

INTRODUCTION

The rapid rise in cab fares across major Indian cities has made daily commuting increasingly expensive for students, professionals, and frequent travelers. Many individuals travel to common destinations such as colleges, offices, airports, and metro stations without realizing that others nearby are heading the same way. To make transportation more affordable and sustainable, the concept of cab sharing has gained significant importance.

In this project, CabConnect is implemented as a lightweight, console-based C++ application that uses file handling to store and manage user and ride information. This approach makes the system simple, portable, and efficient for academic use while still demonstrating real-world functionality. By storing data such as user profiles, ride requests, shared rides, and fare summaries in structured text or binary files, the application ensures persistence without the need for external dependencies. The system focuses on delivering fast ride matching, easy retrieval of stored data, and clear fare calculations—all within a clean, menu-driven interface. This practical implementation showcases how core programming concepts like OOP, file handling, algorithms, and modular design can be combined to solve everyday transportation challenges through an optimized and user-friendly solution.

LITERATURE SURVEY

Existing Cab & Ride-Sharing Applications

- **Ola Share / Uber Pool**
  - Offered shared cab options to reduce fare for users.
  - Used GPS-based route matching.
  - However, services were discontinued in many cities due to operational difficulty and lack of demand post-pandemic.

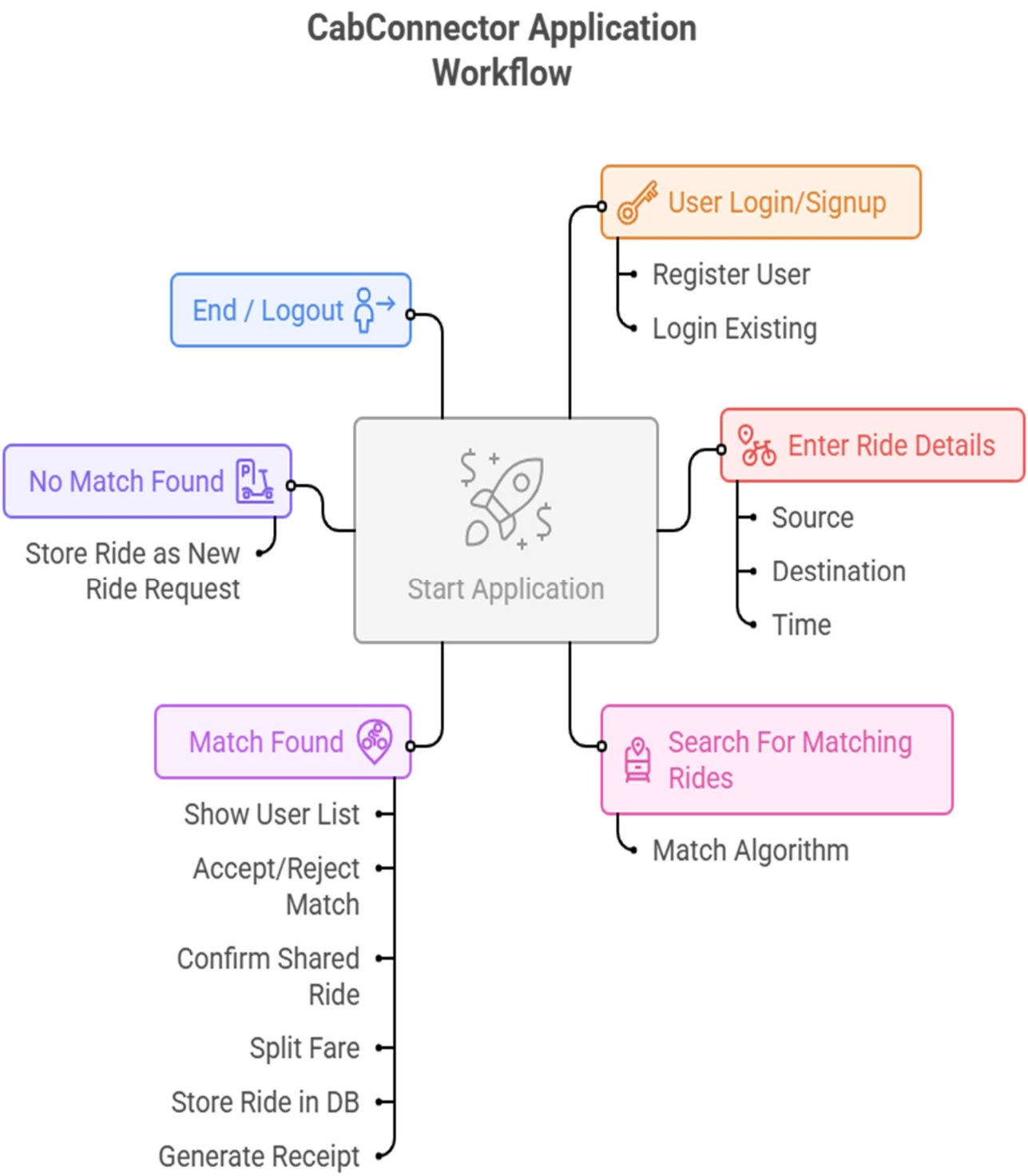
- **BlaBlaCar (Inter-city ride sharing)**
  - Connects long-distance travelers with empty car seats.
  - Focuses on intercity routes rather than daily commuting.

- **QuickRide (Bike & Car Pooling App)**
  - Popular among IT hubs in Bangalore.
  - Matches users based on office routes and timings.
  - Requires real-time location access and app installation.

- **SRide Carpool App**
  - Provides matching based on source–destination and flexible timings.
  - Fare is shared automatically, but requires GPS tracking.

Most modern cab services such as Ola and Uber no longer provide cab-sharing or pool options, especially in many Indian cities. Earlier features like *Ola Share* and *Uber Pool* were gradually discontinued due to operational constraints, reduced demand after the pandemic, and challenges in coordinating shared rides. As a result, users today are forced to book individual rides even when multiple passengers are traveling to the same destination, leading to higher travel costs and increased traffic congestion. This absence of built-in cab-sharing features highlights a significant gap and creates a strong need for alternative systems like **CabConnect**, which encourage shared mobility and cost reduction.

METHODOLOGY



FEATURES

1. User Account Management (File-Based)

- Allows new users to register by entering basic details such as name, phone number, gender, password, source, and destination.
- Stores all user information securely in text/binary files using file handling.
- Provides login functionality by verifying credentials stored in the user file.
- Prevents duplicate users by checking existing entries during registration.

---

2. Ride Creation & Storage

- Users can create a new ride request by entering:
  - Source
  - Destination
  - Travel time
  - Cab preference (optional)
- Ride details are stored in a rides file (e.g., `rides.txt`) for future matching.
- Rides are stored in a structured format, making it easy to retrieve and process.

---

3. Ride Matching System

- Reads all existing ride entries from the rides file.
- Matches users based on:
  - Same or nearby destination
  - Time difference ≤ 30 minutes
  - Gender preference (if enabled)
- Displays a list of potential partners who have similar travel routes.
- Offers a simple match-making algorithm suitable for console applications.

---

4. Cab Sharing Partner Suggestions

- Suggests the best possible ride-sharing partners from stored ride data.
- Provides partner details such as:
  - Name
  - Phone Number
  - Source & Destination
  - Travel Time
- Provides user-driven choice to accept or reject suggested matches.

---

5. Fare Splitting (Offline Calculation)

- Calculates estimated fare based on distance or a fixed charge.
- Automatically divides the fare equally among matched passengers.
- Shows clear breakdown of:
    - Total Fare
    - Number of passengers
    - Individual share
- Stores fare summary in a separate file for record-keeping.

---

6. Booking Confirmation (File Logging)

- Once users accept a match, the system generates:
    - Shared Ride ID
    - Passenger Details
    - Fare Split
    - Time of Ride
- Saves the confirmed ride details in a file (e.g., `shared_rides.txt`).
- Enables easy retrieval of previous ride confirmations for future reference.

---

7. File-Based Data Management

- All data such as:
    - Users
    - Ride Requests
    - Shared Rides
    - Fare Records
- is stored in separate files using file handling.
- Ensures persistent data storage even after program exit.

---

8. Data Validation & Error Handling

- Checks for empty inputs, invalid times, or incorrect formats.
- Handles file read/write errors safely.
- Prevents user from entering invalid ride data.

---

9. Simple & Lightweight Console Interface

- Fully text-based interface – no GUI or external dependencies.

- Menu-driven system for easy navigation.
- Works on all operating systems supporting C++ file I/O.

---

10. Basic Admin Panel (Optional Feature)

- Admin can log in using a special password.
- View all registered users stored in files.
- View all ride requests and shared rides.
- Delete or reset files if needed for maintenance.

CODE

```
#include <iostream>
#include <vector>
#include <string>
#include <fstream>
#include <sstream>
#include <algorithm>
#include <iomanip>
#include <ctime>
#include <cctype>
#include <regex>

using namespace std;

class User {
public:
    string name, phone, gender, password;

    User() {}
    User(string n, string p, string g, string pass)
        : name(n), phone(p), gender(g), password(pass) {}

    string toFileString() {
        return name + "|" + phone + "|" + gender + "|" + password;
    }

    static User fromFileString(string line) {
        stringstream ss(line);
        string n, p, g, pass;
        getline(ss, n, '|');
        getline(ss, p, '|');
        getline(ss, g, '|');
        getline(ss, pass, '|');
        return User(n, p, g, pass);
    }
};

// Ride Class
class Ride {
public:
    string rideId, userPhone, userName, source, destination;
    double totalFare;
    int passengers;
    string dateTime, status;
    double distance;
    string vehicleType;

    Ride() {}
    Ride(string id, string uPhone, string uName, string src, string dest,
         double fare, int pass, string dt, double dist, string vehicle)
        : rideId(id), userPhone(uPhone), userName(uName), source(src),
          destination(dest), totalFare(fare), passengers(pass),
          dateTime(dt), status("ACTIVE"), distance(dist), vehicleType(vehicle) {}

    double getFarePerPerson() {
        return totalFare / passengers;
    }

    string toFileString() {
        return rideId + "|" + userPhone + "|" + userName + "|" + source + "|" +
               destination + "|" + to_string(totalFare) + "|" +
               to_string(passengers) + "|" + dateTime + "|" + status + "|" +
```

```
                to_string(distance) + "|" + vehicleType;
    }

    static Ride fromFileString(string line) {
        stringstream ss(line);
        Ride r;
        string fare, pass, dist;
        getline(ss, r.rideId, '|');
        getline(ss, r.userPhone, '|');
        getline(ss, r.userName, '|');
        getline(ss, r.source, '|');
        getline(ss, r.destination, '|');
        getline(ss, fare, '|');
        getline(ss, pass, '|');
        getline(ss, r.dateTime, '|');
        getline(ss, r.status, '|');
        getline(ss, dist, '|');
        getline(ss, r.vehicleType, '|');
        r.totalFare = stod(fare);
        r.passengers = stoi(pass);
        r.distance = stod(dist);
        return r;
    }
};

// File Manager Class
class FileManager {
public:
    static void saveUser(User u) {
        ofstream file("users.txt", ios::app);
        file << u.toFileString() << endl;
        file.close();
    }

    static vector<User> loadUsers() {
        vector<User> users;
        ifstream file("users.txt");
        string line;
        while (getline(file, line)) {
            if (!line.empty())
                users.push_back(User::fromFileString(line));
        }
        file.close();
        return users;
    }

    static void saveRide(Ride r) {
        ofstream file("rides.txt", ios::app);
        file << r.toFileString() << endl;
        file.close();
    }

    static vector<Ride> loadRides() {
        vector<Ride> rides;
        ifstream file("rides.txt");
        string line;
        while (getline(file, line)) {
            if (!line.empty())
                rides.push_back(Ride::fromFileString(line));
        }
        file.close();
        return rides;
    }
```

```
    static void updateRides(vector<Ride> rides) {
        ofstream file("rides.txt");
        for (auto& r : rides) {
            file << r.toFileString() << endl;
        }
        file.close();
    }
};

// System Class
class CabConnectSystem {
private:
    vector<User> users;
    vector<Ride> rides;
    User currentUser;
    bool isLoggedIn = false;
    int rideCounter = 1000;

    vector<string> puneLocations = {
        "Shivajinagar", "Kothrud", "Hinjewadi", "Viman Nagar", "Wakad",
        "Baner", "Aundh", "Pimpri", "Chinchwad", "Katraj"
    };

    vector<string> vehicleTypes = {
        "Bike", "Cab", "Rickshaw"
    };

    string toLowerCase(string str) {
        transform(str.begin(), str.end(), str.begin(), ::tolower);
        return str;
    }


    double calculateFairShare(double totalFare, int passengers) {
        return totalFare / passengers;
    }


    bool isValidPhone(string phone) {
        if (phone.length() != 10) return false;
        for (char c : phone) {
            if (!isdigit(c)) return false;
        }
        return true;
    }


    bool isValidPassword(string password) {
        bool hasChar = false, hasDigit = false, hasSymbol = false;

        for (char c : password) {
            if (isalpha(c)) hasChar = true;
            else if (isdigit(c)) hasDigit = true;
            else if (!isspace(c)) hasSymbol = true;
        }

        return hasChar && hasDigit && hasSymbol;
    }


    bool isValidGender(string gender) {
        string lowerGender = toLowerCase(gender);
```

```
        return (lowerGender == "male" || lowerGender == "female" || lowerGender
== "other");
    }


    int displayLocationsMenu(string type) {
        cout << "\n--- SELECT " << type << " LOCATION ---\n";
        for (int i = 0; i < puneLocations.size(); i++) {
            cout << (i+1) << ". " << puneLocations[i] << "\n";
        }
        cout << "11. Other Location\n";
        cout << "Enter your choice (1-11): ";

        int choice;
        cin >> choice;
        return choice;
    }


    int displayVehicleMenu() {
        cout << "\n--- SELECT VEHICLE TYPE ---\n";
        for (int i = 0; i < vehicleTypes.size(); i++) {
            cout << (i+1) << ". " << vehicleTypes[i] << "\n";
        }
        cout << "Enter your choice (1-3): ";

        int choice;
        cin >> choice;
        return choice;
    }


    double getDistance(string source, string destination) {

        vector<string> locations = {
            "Shivajinagar", "Kothrud", "Hinjewadi", "Viman Nagar", "Wakad",
            "Baner", "Aundh", "Pimpri", "Chinchwad", "Katraj"
        };

        auto it1 = find(locations.begin(), locations.end(), source);
        auto it2 = find(locations.begin(), locations.end(), destination);

        if (it1 != locations.end() && it2 != locations.end()) {
            int idx1 = it1 - locations.begin();
            int idx2 = it2 - locations.begin();
            return (abs(idx1 - idx2) + 1) * 5.0;
        }


        return 10.0;
    }


    double calculateFare(double distance, string vehicleType) {
        double ratePerKm;

        if (vehicleType == "Bike") {
            ratePerKm = 8.0;
        } else if (vehicleType == "Cab") {
            ratePerKm = 12.0;
        } else if (vehicleType == "Rickshaw") {
            ratePerKm = 10.0;
        } else {
```

```
            ratePerKm = 10.0;
        }

        return distance * ratePerKm;
    }


    string getLocationName(int choice) {
        if (choice >= 1 && choice <= 10) {
            return puneLocations[choice - 1];
        }
        return "";
    }


    string getVehicleType(int choice) {
        if (choice >= 1 && choice <= 3) {
            return vehicleTypes[choice - 1];
        }
        return "Cab";
    }

public:
    CabConnectSystem() {
        users = FileManager::loadUsers();
        rides = FileManager::loadRides();
        rideCounter = 1000 + rides.size();
    }

    void displayHeader() {
        cout << "\n";
        cout << "========================================\n";
        cout << "     CABCONNECT - RIDE SHARING SYSTEM   \n";
        cout << "========================================\n\n";
    }

    void registerUser() {
        cout << "\n--- USER REGISTRATION ---\n";
        User newUser;

        cout << "Enter Full Name: ";
        cin.ignore();
        getline(cin, newUser.name);


        while (true) {
            cout << "Enter Phone Number: ";
            cin >> newUser.phone;

            if (!isValidPhone(newUser.phone)) {
                cout << ">> ERROR: Phone number must be exactly 10 digits! Please
try again.\n";
            } else {

                bool phoneExists = false;
                for (auto& u : users) {
                    if (u.phone == newUser.phone) {
                        phoneExists = true;
                        break;
                    }
                }
                if (phoneExists) {
```

```
                    cout << ">> ERROR: Phone number already registered! Please
try again.\n";
                } else {
                    break;
                }
            }
        }


        while (true) {
            cout << "Enter Gender (Male/Female/Other): ";
            cin >> newUser.gender;

            if (!isValidGender(newUser.gender)) {
                cout << ">> ERROR: Gender must be 'Male', 'Female', or 'Other'!
Please try again.\n";
            } else {
                break;
            }
        }


        while (true) {
            cout << "Enter Password (must contain letters, numbers, and symbols):
";
            cin >> newUser.password;

            if (!isValidPassword(newUser.password)) {
                cout << ">> ERROR: Password must contain at least one letter, one
number, and one symbol! Please try again.\n";
            } else {
                break;
            }
        }

        users.push_back(newUser);
        FileManager::saveUser(newUser);

        cout << ">> Registration successful! Please login to continue.\n";
    }

    bool loginUser() {
        cout << "\n--- USER LOGIN ---\n";
        string phone, pass;

        cout << "Enter Phone Number: ";
        cin >> phone;
        cout << "Enter Password: ";
        cin >> pass;

        for (auto& u : users) {
            if (u.phone == phone && u.password == pass) {
                currentUser = u;
                isLoggedIn = true;
                cout << ">> Welcome, " << currentUser.name << "!\n";
                return true;
            }
        }

        cout << ">> Invalid credentials!\n";
        return false;
    }
```

```
    void createRide() {
        if (!isLoggedIn) {
            cout << ">> Please login first!\n";
            return;
        }

        cout << "\n--- CREATE NEW RIDE ---\n";
        string src, dest, vehicleType;
        double fare, distance;
        int pass;


        int sourceChoice = displayLocationsMenu("SOURCE");
        if (sourceChoice == 11) {
            cout << "Enter your source location: ";
            cin.ignore();
            getline(cin, src);
        } else if (sourceChoice >= 1 && sourceChoice <= 10) {
            src = getLocationName(sourceChoice);
        } else {
            cout << ">> Invalid choice!\n";
            return;
        }


        int destChoice = displayLocationsMenu("DESTINATION");
        if (destChoice == 11) {
            cout << "Enter your destination location: ";
            cin.ignore();
            getline(cin, dest);
        } else if (destChoice >= 1 && destChoice <= 10) {
            dest = getLocationName(destChoice);
        } else {
            cout << ">> Invalid choice!\n";
            return;
        }


        int vehicleChoice = displayVehicleMenu();
        if (vehicleChoice >= 1 && vehicleChoice <= 3) {
            vehicleType = getVehicleType(vehicleChoice);
        } else {
            cout << ">> Invalid choice! Defaulting to Cab.\n";
            vehicleType = "Cab";
        }


        cout << "Enter Number of Passengers needed: ";
        cin >> pass;


        cout << "\n>> ROUTE INFORMATION:\n";
        cout << "   From: " << src << "\n";
        cout << "   To: " << dest << "\n";
        cout << "   Vehicle Type: " << vehicleType << "\n";
        cout << "   Passengers: " << pass << "\n";


        if (sourceChoice != 11 && destChoice != 11) {
            distance = getDistance(src, dest);
            cout << "   Distance: " << fixed << setprecision(1) << distance << "
km\n";
        } else {
```

```
            cout << "Enter distance in km: ";
            cin >> distance;
            cout << "   Distance: " << fixed << setprecision(1) << distance << "
km\n";
        }


        fare = calculateFare(distance, vehicleType);
        cout << "   Total Fare: ₹" << fixed << setprecision(2) << fare << "\n";


        cout << "   Fare Rates:\n";
        cout << "   - Bike: ₹8 per km\n";
        cout << "   - Cab: ₹12 per km\n";
        cout << "   - Rickshaw: ₹10 per km\n";


        time_t now = time(0);
        tm* ltm = localtime(&now);
        char dt[100];
        strftime(dt, sizeof(dt), "%Y-%m-%d %H:%M", ltm);
        string dateTime(dt);

        string id = "R" + to_string(++rideCounter);
        Ride newRide(id, currentUser.phone, currentUser.name, src, dest, fare,
pass, dateTime, distance, vehicleType);

        rides.push_back(newRide);
        FileManager::saveRide(newRide);

        cout << "\n>> Ride created successfully!\n";
        cout << ">> Ride ID: " << id << "\n";
        cout << ">> Date & Time: " << dateTime << "\n";
        cout << ">> Route: " << src << " → " << dest << "\n";
        cout << ">> Vehicle: " << vehicleType << "\n";
        cout << ">> Passengers: " << pass << "\n";
        cout << ">> Distance: " << fixed << setprecision(2) << distance << "
km\n";
        cout << ">> Total Fare: ₹" << fare << " | Per Person: ₹"
             << fixed << setprecision(2) << calculateFairShare(fare, pass) <<
"\n";


        findMatches(newRide);
    }

    void findMatches(Ride myRide) {
        vector<Ride*> matches;

        for (auto& r : rides) {
            if (r.rideId != myRide.rideId &&
                r.status == "ACTIVE" &&
                toLowerCase(r.source) == toLowerCase(myRide.source) &&
                toLowerCase(r.destination) == toLowerCase(myRide.destination) &&
                r.vehicleType == myRide.vehicleType) {
                matches.push_back(&r);
            }
        }

        if (matches.empty()) {
            cout << "\n>> No matches found yet. You'll be notified when someone
posts a similar ride.\n";
```

```
            return;
        }

        cout << "\n>> FOUND " << matches.size() << " MATCHING RIDE(S)!\n";
        cout << "========================================\n";

        for (size_t i = 0; i < matches.size(); i++) {
            Ride* r = matches[i];
            cout << "\n[" << (i+1) << "] Ride ID: " << r->rideId << "\n";
            cout << "    Traveler: " << r->userName << "\n";
            cout << "    Route: " << r->source << " → " << r->destination <<
"\n";
            cout << "    Vehicle: " << r->vehicleType << "\n";
            cout << "    Distance: " << r->distance << " km\n";
            cout << "    Time: " << r->dateTime << "\n";
            cout << "    Available Seats: " << r->passengers << "\n";
            cout << "    Total Fare: ₹" << r->totalFare << "\n";
            cout << "    Fare per person: ₹" << fixed << setprecision(2)
                 << r->getFarePerPerson() << "\n";
        }

        cout << "\n========================================\n";
        cout << "Would you like to join any ride? (1-" << matches.size() << " or
0 to skip): ";

        int choice;
        cin >> choice;

        if (choice > 0 && choice <= matches.size()) {
            joinRide(matches[choice - 1]);
        }
    }

    void joinRide(Ride* targetRide) {

        Ride* myRidePtr = nullptr;
        for (auto& r : rides) {
            if (r.userPhone == currentUser.phone && r.status == "ACTIVE") {
                myRidePtr = &r;
                break;
            }
        }

        if (!myRidePtr) {
            cout << ">> Error: Could not find your active ride!\n";
            return;
        }


        double myOriginalFare = myRidePtr->totalFare;


        targetRide->passengers += 1;


        myRidePtr->status = "MERGED";


        double newFarePerPerson = calculateFairShare(targetRide->totalFare,
targetRide->passengers);

        FileManager::updateRides(rides);
```

```
        cout << "\n>> RIDE SHARED SUCCESSFULLY!\n";
        cout << "========================================\n";
        cout << ">> SHARING DETAILS:\n";
        cout << "   Total Passengers: " << targetRide->passengers << "\n";
        cout << "   Combined Total Fare: ₹" << fixed << setprecision(2) <<
targetRide->totalFare << "\n";
        cout << "   >> New Fare Per Person: ₹" << newFarePerPerson << "\n";
        cout << "   >> You Save: ₹" << fixed << setprecision(2)
             << (myOriginalFare - newFarePerPerson) << "\n";
        cout << "========================================\n";


        targetRide->userName += " & " + currentUser.name;
    }

    void viewMyRides() {
        if (!isLoggedIn) {
            cout << " Please login first!\n";
            return;
        }

        cout << "\n--- MY RIDES ---\n";
        bool found = false;

        for (auto& r : rides) {
            if (r.userPhone == currentUser.phone) {
                found = true;
                cout << "\n------------------------------------------------
\n";
                cout << "Ride ID: " << r.rideId << "\n";
                cout << "Route: " << r.source << " → " << r.destination << "\n";
                cout << "Vehicle: " << r.vehicleType << "\n";
                cout << "Distance: " << r.distance << " km\n";
                cout << "Time: " << r.dateTime << "\n";
                cout << "Status: " << r.status << "\n";
                cout << "Passengers: " << r.passengers << "\n";
                cout << "Total Fare: ₹" << r.totalFare << "\n";
                if (r.status == "ACTIVE") {
                    cout << "Your Share: ₹" << fixed << setprecision(2) <<
r.getFarePerPerson() << "\n";
                }
                cout << "------------------------------------------------\n";
            }
        }

        if (!found) {
            cout << "No rides found.\n";
        }
    }

    void viewAllActiveRides() {
        cout << "\n--- ALL ACTIVE RIDES ---\n";
        bool found = false;

        for (auto& r : rides) {
            if (r.status == "ACTIVE") {
                found = true;
                cout << "\n------------------------------------------------
\n";
                cout << "Ride ID: " << r.rideId << "\n";
                cout << "Posted by: " << r.userName << "\n";
                cout << "Route: " << r.source << " → " << r.destination << "\n";
                cout << "Vehicle: " << r.vehicleType << "\n";
```

```
                cout << "Distance: " << r.distance << " km\n";
                cout << "Time: " << r.dateTime << "\n";
                cout << "Available Seats: " << r.passengers << "\n";
                cout << "Total Fare: ₹" << r.totalFare << "\n";
                cout << "Fare per person: ₹" << fixed << setprecision(2) <<
r.getFarePerPerson() << "\n";
                cout << "-------------------------------------------------\n";
            }
        }

        if (!found) {
            cout << "-- No active rides available.\n";
        }
    }

    void searchRides() {
        cout << "\n--- SEARCH RIDES ---\n";


        int sourceChoice = displayLocationsMenu("SOURCE");
        string src;
        if (sourceChoice == 11) {
            cout << "Enter source location: ";
            cin.ignore();
            getline(cin, src);
        } else if (sourceChoice >= 1 && sourceChoice <= 10) {
            src = getLocationName(sourceChoice);
        } else {
            cout << ">> Invalid choice!\n";
            return;
        }


        int destChoice = displayLocationsMenu("DESTINATION");
        string dest;
        if (destChoice == 11) {
            cout << "Enter destination location: ";
            cin.ignore();
            getline(cin, dest);
        } else if (destChoice >= 1 && destChoice <= 10) {
            dest = getLocationName(destChoice);
        } else {
            cout << ">> Invalid choice!\n";
            return;
        }


        int vehicleChoice = displayVehicleMenu();
        string vehicleType;
        if (vehicleChoice >= 1 && vehicleChoice <= 3) {
            vehicleType = getVehicleType(vehicleChoice);
        } else {
            vehicleType = "Any";
        }

        bool found = false;
        cout << "\nSearch Results for: " << src << " → " << dest;
        if (vehicleType != "Any") {
            cout << " [" << vehicleType << "]";
        }
        cout << "\n";

        for (auto& r : rides) {
```

```
            if (r.status == "ACTIVE" &&
                toLowerCase(r.source).find(toLowerCase(src)) != string::npos &&
                toLowerCase(r.destination).find(toLowerCase(dest)) !=
string::npos &&
                (vehicleType == "Any" || r.vehicleType == vehicleType)) {

                found = true;
                cout << "\n----------------------------------------------------
\n";
                cout << "Ride ID: " << r.rideId << "\n";
                cout << "Posted by: " << r.userName << "\n";
                cout << "Route: " << r.source << " → " << r.destination << "\n";
                cout << "Vehicle: " << r.vehicleType << "\n";
                cout << "Distance: " << r.distance << " km\n";
                cout << "Time: " << r.dateTime << "\n";
                cout << "Available Seats: " << r.passengers << "\n";
                cout << "Total Fare: ₹" << r.totalFare << "\n";
                cout << "Fare per person: ₹" << fixed << setprecision(2) <<
r.getFarePerPerson() << "\n";
            }
        }

        if (!found) {
            cout << " No rides found for your search criteria.\n";
        }
    }

    void userMenu() {
        while (true) {
            cout << "\n--- USER MENU ---\n";
            cout << "1. Create New Ride\n";
            cout << "2. View All Active Rides\n";
            cout << "3. View My Rides\n";
            cout << "4. Search Rides\n";
            cout << "5. Logout\n";
            cout << "Enter choice: ";

            int choice;
            cin >> choice;

            switch (choice) {
                case 1: createRide(); break;
                case 2: viewAllActiveRides(); break;
                case 3: viewMyRides(); break;
                case 4: searchRides(); break;
                case 5:
                    isLoggedIn = false;
                    cout << "Logged out successfully!\n";
                    return;
                default: cout << " Invalid choice!\n";
            }
        }
    }

    void run() {
        while (true) {
            displayHeader();
            cout << "1. Register\n";
            cout << "2. Login\n";
            cout << "3. Exit\n";
            cout << "Enter choice: ";

            int choice;
```

```
            cin >> choice;

            switch (choice) {
                case 1: registerUser(); break;
                case 2:
                    if (loginUser()) {
                        userMenu();
                    }
                    break;
                case 3:
                    cout << "\n Thank you for using CabConnect!\n";
                    return;
                default:
                    cout << "Invalid choice!\n";
            }
        }
    }
};

int main() {
    CabConnectSystem system;
    system.run();
    return 0
```

SCREENSHOTS OF RESULTS

1. Registered 2 Accounts

```
 ======================================
      CABCONNECT - RIDE SHARING SYSTEM
 ======================================

 1. Register
 2. Login
 3. Exit
 Enter choice: 1

 --- USER REGISTRATION ---
 Enter Full Name: Ishwari Patil
 Enter Phone Number: 4563985696
 Enter Gender (Male/Female/Other): Female
 Enter Password (must contain letters, numbers, and symbols): Ishwari@123
 >> Registration successful! Please login to continue.
```

```
 ======================================
      CABCONNECT - RIDE SHARING SYSTEM
 ======================================

 1. Register
 2. Login
 3. Exit
 Enter choice: 1

 --- USER REGISTRATION ---
 Enter Full Name: Apurv Saktepar
 Enter Phone Number: 7485964152
 Enter Gender (Male/Female/Other): Male
 Enter Password (must contain letters, numbers, and symbols): Apurv@123
 >> Registration successful! Please login to continue.
```

```
======================================
     CABCONNECT - RIDE SHARING SYSTEM
======================================

1. Register
2. Login
3. Exit
Enter choice: 2

--- USER LOGIN ---
Enter Phone Number: 7485964152
Enter Password: Apurv@123
>> Welcome, Apurv Saktepar!

--- USER MENU ---
1. Create New Ride
2. View All Active Rides
3. View My Rides
4. Search Rides
5. Logout
Enter choice:
```

```
  --- SELECT VEHICLE TYPE ---
  1. Bike
  2. Cab
  3. Rickshaw
  Enter your choice (1-3): 2
  Enter Number of Passengers needed: 4

  >> ROUTE INFORMATION:
     From: Shivajinagar
     To: Katraj
     Vehicle Type: Cab
     Passengers: 4
     Distance: 50.0 km
     Total Fare: ₹600.00
     Fare Rates:
     - Bike: ₹8 per km
     - Cab: ₹12 per km
     - Rickshaw: ₹10 per km

  >> Ride created successfully!
  >> Ride ID: R1004
  >> Date & Time: 2025-11-23 22:58
  >> Route: Shivajinagar → Katraj
  >> Vehicle: Cab
  >> Passengers: 4
  >> Distance: 50.00 km
  >> Total Fare: ₹600.00 | Per Person: ₹150.00
```

```
--- USER MENU ---
1. Create New Ride
2. View All Active Rides
3. View My Rides
4. Search Rides
5. Logout
Enter choice: 2

--- ALL ACTIVE RIDES ---

--------------------------------------------------
Ride ID: R1001
Posted by: Shantanu Kaute
Route: Shivajinagar → Katraj
Vehicle:
Distance: 50.00 km
Time: 2025-11-22 19:44
Available Seats: 1
Total Fare: ₹500.00
Fare per person: ₹500.00
--------------------------------------------------
```

```
--------------------------------------------------
Ride ID: R1003
Posted by: Shantanu Kaute
Route: Kothrud → Bibvewadi
Vehicle: Rickshaw
Distance: 8.00 km
Time: 2025-11-23 22:13
Available Seats: 2
Total Fare: ₹80.00
Fare per person: ₹40.00
--------------------------------------------------

--------------------------------------------------
Ride ID: R1004
Posted by: Apurv Saktepar
Route: Shivajinagar → Katraj
Vehicle: Cab
Distance: 50.00 km
Time: 2025-11-23 22:58
Available Seats: 4
Total Fare: ₹600.00
Fare per person: ₹150.00
--------------------------------------------------
```

```
--- USER MENU ---
1. Create New Ride
2. View All Active Rides
3. View My Rides
4. Search Rides
5. Logout
Enter choice: 5
Logged out successfully!
```

CONCLUSION

The CabConnect system successfully demonstrates how everyday transportation challenges can be solved using fundamental programming concepts such as Object-Oriented Programming, file handling, and algorithmic decision-making. By offering features like ride matching, partner suggestions, and fare splitting, the project shows how technology can make commuting more affordable, organized, and collaborative. The console-based design keeps the application simple, lightweight, and easily portable, making it a perfect academic model for understanding real-world system implementation without the complexity of external databases or servers.

Beyond solving a practical problem, the project highlights the importance of structured thinking and modular software design. CabConnect proves that even a text-based interface, when built with clean logic and efficient storage techniques, can deliver meaningful results. This project not only enhances technical learning but also encourages innovation in sustainable travel solutions. It sets a strong foundation for transforming a simple idea into a scalable mobility system that promotes shared rides, reduces traffic, and contributes to smarter, budget-friendly transportation for everyday users.

FUTURE SCOPE

- **Integration with GUI (Qt/GTK):** Upgrade the console application to a graphical interface for better usability.

- **Addition of Database Support:** Replace file handling with MySQL or SQLite to enable faster search, filtering, and data scalability.

- **Real-Time GPS Integration:** Add location-based matching for more accurate partner suggestions.

- **Mobile Application Version:** Develop Android/iOS apps using Flutter or Kotlin for broader usability.

- **Fare Prediction Model:** Use machine learning to estimate fares based on distance, time, and traffic patterns.

- **Safety Enhancements:** Add OTP verification, emergency contact alerts, and user verification features.

- **Rating and Review System:** Allow passengers to rate ride partners and improve the reliability of matches.

- **Notification System:** Send SMS or email alerts when a matching ride is found.

- **Advanced Matchmaking Algorithm:** Use clustering or graph algorithms to match multiple users more efficiently.

REFERENCES

1. https://github.com/rajattekriwal/Cab-Booking-System
2. https://github.com/vivekmittal01/Transport-Management-System
3. https://github.com/NeelPatel17/Taxi-Management-System
4. https://github.com/nikhilroxtomar/Cab-Booking-System
5. https://core.ac.uk/reader/322683583
6. https://github.com/mohitkhedkar/Ride-Sharing-System.