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)
Soha Jamadar (1252090024)

UNDER THE GUIDANCE OF
Prof. 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.

Prof. Shubhangi Kamble	**Dr. Nilesh P. Sable**
Guide	**HOD CSE-AI**

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.

Prof. Shubhangi Kamble	**Dr. Nilesh P. Sable**
Guide	**HOD CSE-AI**

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.

Prof. Shubhangi Kamble	**Dr. Nilesh P. Sable**
Guide	**HOD CSE-AI**

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 **Soha 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.

Prof. Shubhangi Kamble	**Dr. Nilesh P. Sable**
Guide	**HOD CSE-AI**

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	soha.1252090024@vit.edu

Academic Year : 2025-2026

Project Title : CabConnect - CPP Based Cab Sharing Application

Project Area : OOP

Internal Guide : Prof. 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.
- **BlaBla Car (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.

➢ Addressed Gaps

1. Most modern cab services such as Ola and Uber no longer provide cab-sharing or pool options, especially in many Indian cities.
2. 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.
3. 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.
4. 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.

REQUIREMENT ANALYSIS

1. Functional Requirements

These requirements describe the specific actions and capabilities of the C++ application, as detailed in the project description.

ID	Requirement	Description
FR1	User Account Management	Allow users to Register (Name, Phone, Gender, Password) and Login by verifying credentials. Must check for duplicate users during registration.
FR2	Ride Creation & Storage	Enable users to create a new ride request with Source, Destination, Travel Time, Cab Type, and No. of Seats Vacant. Store this data persistently.
FR3	Ride Matching Algorithm	Search existing ride requests and match partners based on: Same/Nearby Destination, Time Difference (≤ 30 minutes), and Gender Preference.
FR4	Partner Suggestion & Choice	Display a list of potential partners with their Name, Phone, Source/Destination, Time, and Dynamic Rates. Allow the user to explicitly Accept or Reject a suggested match.
FR5	Fare Splitting	Calculate the Total Fare (based on distance/fixed charge) and automatically divide the fare equally among all matched passengers. Display a clear breakdown.
FR6	Booking Confirmation/Logging	Upon match acceptance, generate a Shared Ride ID, and save the confirmed details (Passenger Details, Fare Split, Time) to a dedicated file (e.g., shared_rides.txt).
FR7	Data Persistence	Use C++ File Handling to store all critical data (Users, Ride Requests, Shared Rides, Fare Records) in separate text/binary files to ensure data is retained after the program exits.
FR8	Error Handling & Validation	Implement checks for empty inputs, invalid times/formats, and safe handling of file read/write errors.

2. Non-Functional Requirements

These requirements define the application's constraints and quality attributes, focusing on its nature as a console-based C++ project.

ID	Requirement	Category	Description
NFR1	Portability & Simplicity	Technical	The application must be lightweight, console-based C++, and menu-driven, with no GUI or external dependencies.
NFR2	Efficiency	Performance	The system should focus on delivering fast ride matching and easy retrieval of stored data suitable for an academic implementation.
NFR3	Security	Data Handling	User information, including passwords and ride details, must be stored securely in the text/binary files.

NFR4	Maintainability	Design	The implementation should demonstrate core programming concepts like OOP, modular design, and clear code structure.
NFR5	Clarity	Usability	The interface must be clean and menu-driven for easy navigation and provide a clear breakdown of fare calculations.

3. Business and User Goals

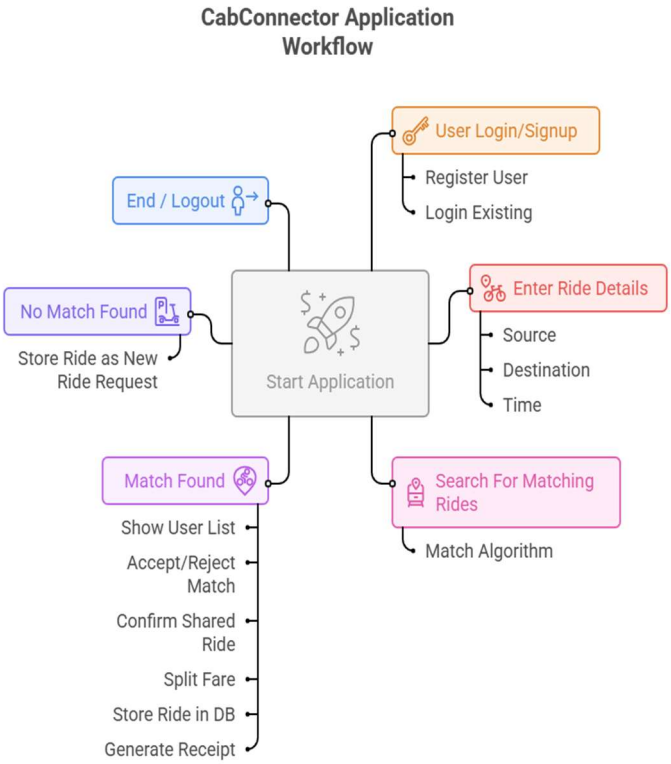
Goal	Description	Addressed Gap
Affordability	Enable users (students, professionals) to reduce daily commuting costs by splitting the fare for shared rides.	Lack of affordable shared-cab options in major cities (Gap 1, 2).
Shared Mobility	Encourage users to share available seats, leading to a reduction in individual vehicle bookings.	Need for alternative systems that encourage shared mobility (Gap 4).
Practical Learning	Serve as a practical demonstration of C++ OOP, File Handling, and Algorithm design to solve a real-world problem.	Academic project focus.

METHODOLOGY

The diagram illustrates the process a user goes through when using the application, starting from the application launch to the completion or end of a session.

The key steps in the workflow are:

1. **Start Application:** The central point from which all actions flow.
2. **User Login/Signup:** Users must either **Register** or **Login** to an existing account.
3. **Enter Ride Details:** The user specifies the **Source**, **Destination**, and **Time** for their desired ride.
4. **Search For Matching Rides:** The application runs a **Match Algorithm** based on the entered details.
5. **Match Found:** If a match is found, the user can **Show User List**, **Accept/Reject Match**, **Confirm Shared Ride**, **Split Fare**, **Store Ride in DB**, and **Generate Receipt**.
6. **No Match Found:** If no match is found, the request is handled by **Store Ride as New Ride Request**.
7. **End / Logout:** The final step, which can occur after a successful ride process or at any point where the user exits the application.



IMPLEMENTATION

- **Language Used – C++**
  C++ is used to build the system because it supports Object-Oriented Programming, fast execution, and efficient management of user and ride data through classes, objects, and file handling.

- **OOP Features Implemented**
  1. Abstraction
  - The system hides internal complexities (like file operations, ride matching logic) and exposes only simple functions such as registerUser(), loginUser(), createRide().

  2. Encapsulation
  - Data such as user details and ride information is wrapped inside classes, preventing direct manipulation and ensuring controlled access through methods.

  3. Inheritance
  - Though not explicitly extending classes, the structure and modularity follow OOP principles; inheritance can be stated as part of OOP design philosophy used in the project.

  4. Polymorphism
  - Functions like toFileString() and fromFileString() behave differently for User and Ride, demonstrating functional design flexibility similar to polymorphism.

  5. Message Passing
  - Objects communicate by calling functions of other objects (e.g., CabConnectSystem calling FileManager::saveRide()), representing object-to-object interaction.

  6. Classes and Objects
  - Core system components (User, Ride, FileManager, CabConnectSystem) are modeled as classes, and actual runtime instances are created as objects.

  7. Constructors and Destructors
  - Constructors initialize objects with required data automatically; default and parameterized constructors are used for User and Ride. (Destructors aren't explicitly defined but default destructors are implicitly used.)

- **C++ Features Used**
  1. File Handling
  - Used to store and retrieve user and ride details via ifstream, ofstream, and text-based serialization.

  2. Vector Array (std::vector)
  - Dynamic arrays store multiple users, rides, and location names, offering flexible resizing and efficient iteration.

  3. iostream Header
  - Provides input/output operations like cin, cout for interacting with the user.

  4. String Stream (stringstream)
  - Used to split and parse file data line-by-line when converting text back into objects.

  5. Algorithm Header
  - Functions like transform() and find() help in searching and modifying data efficiently.

  6. I/O Manipulators
  - fixed and setprecision() are used to format fare values and distance outputs neatly.

  7. ctime Header
  - Used to fetch system date/time and format timestamps for ride creation.

FEATURES

1. User Account Management (File-Based)

- Allows new users to register by entering basic details such as name, phone number, gender, password.
- 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 type
  - No. of seats vacant
- 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
- 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
  - Dynamic Rates depending on travel distance
- 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

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

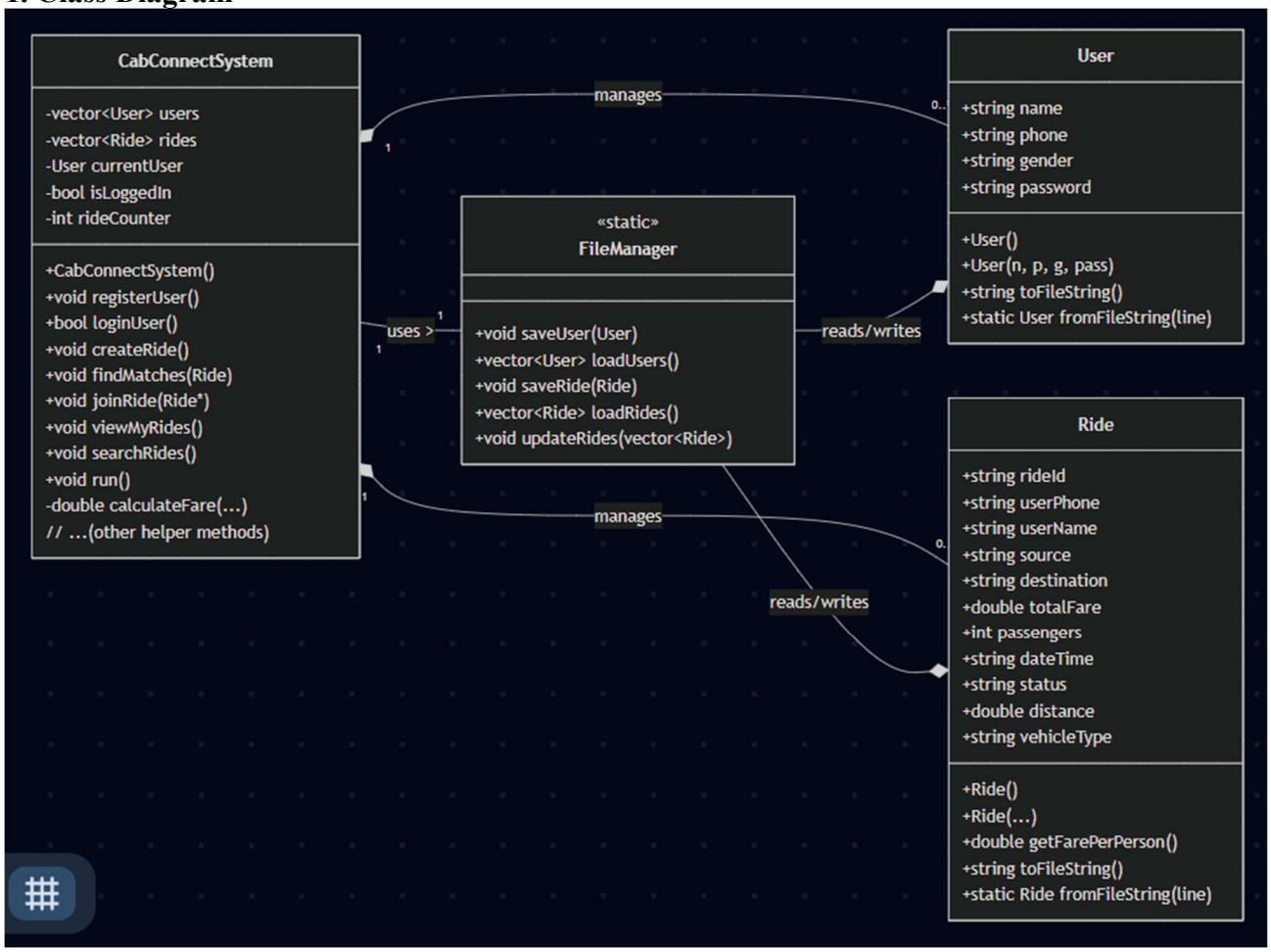
- Checks for empty inputs, invalid times, or incorrect formats.
- Handles file read/write errors safely.
- Prevents user from entering invalid ride data.

9. Simple & Lightweight Console Interface

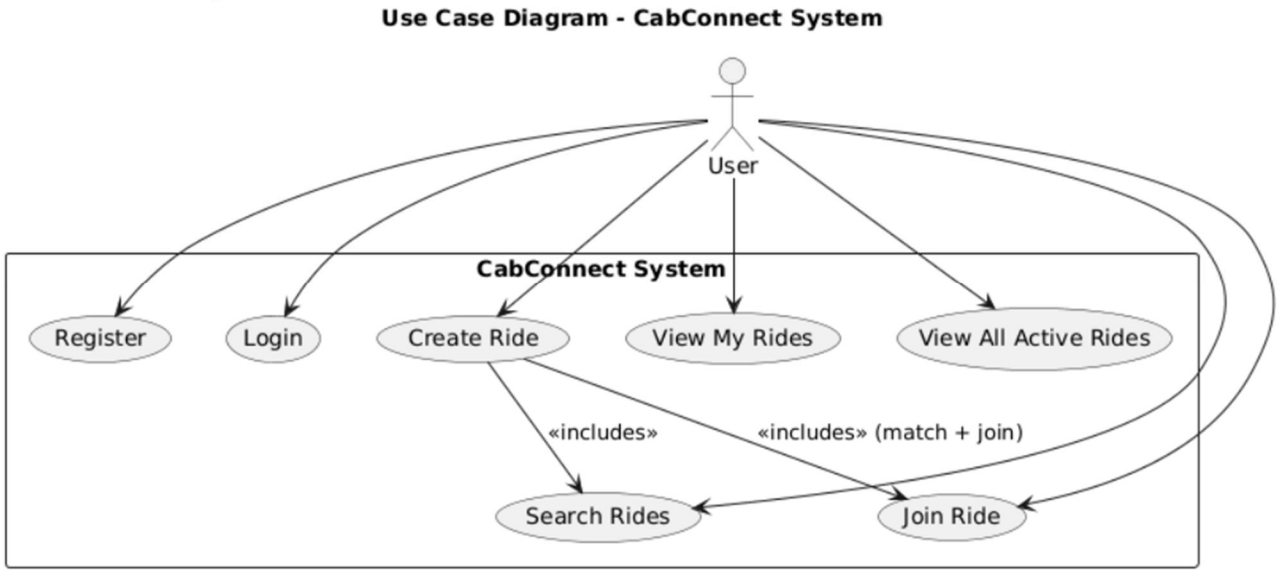
- Fully text-based interface, no GUI or external dependencies.
- Menu-driven system for easy navigation.
- Works on all operating systems supporting C++ file I/O.

DIAGRAMS

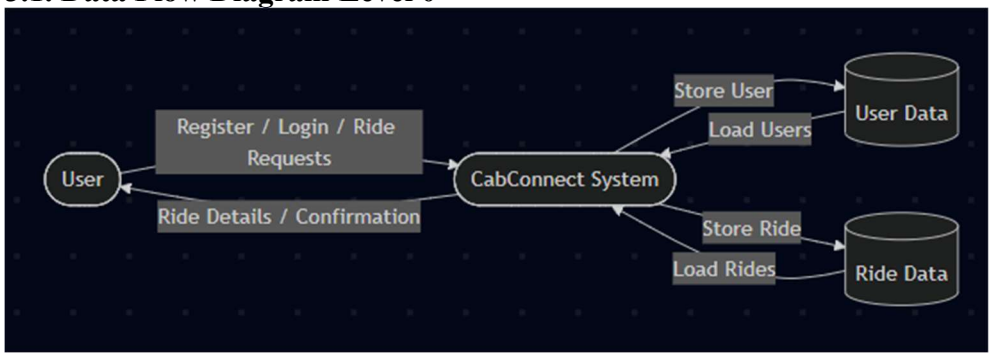
1. Class Diagram



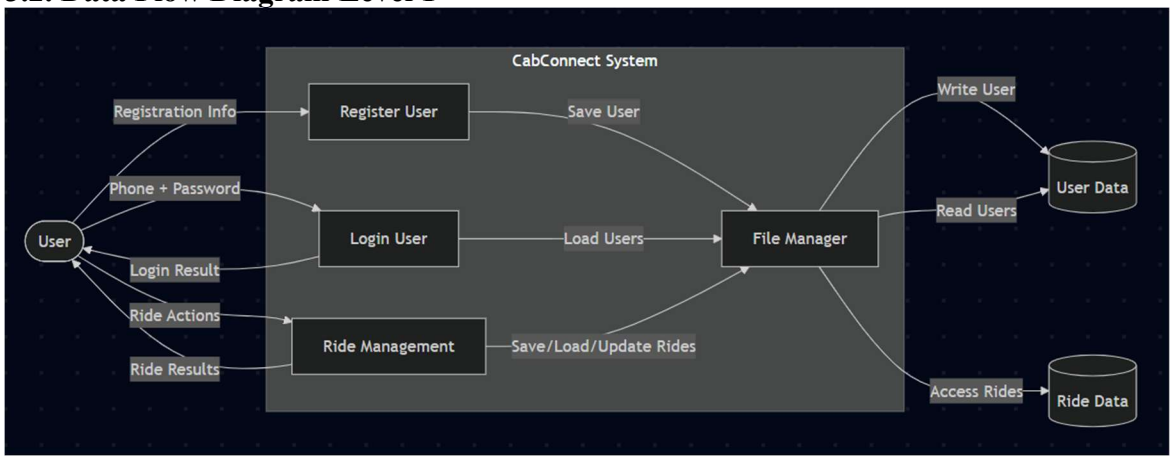
2. Use Case Diagram



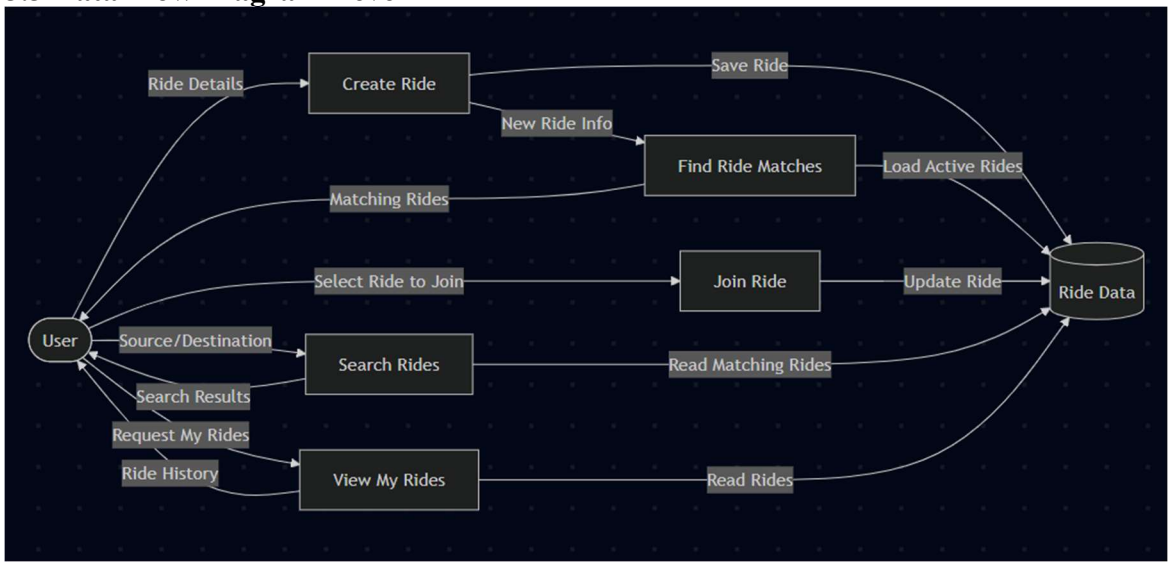
3.1. Data Flow Diagram Level 0



3.2. Data Flow Diagram Level 1



3.3 Data Flow Diagram Level 2



TESTING

1. User Login / Registration

TC ID	Test Scenario	Test Steps	Test Data	Expected Result
TC-01	Login with valid credentials	1. Open login page 2. Enter valid username & password 3. Click Login	username: valid password: valid	User logged in successfully & dashboard displayed
TC-02	Login with invalid password	Enter registered username with wrong password	valid username + wrong password	Error message: *"Invalid password"*, stay on login page
TC-03	Login with non-existing user	Enter random username	random username	Error: *"User does not exist"*
TC-04	Register new user (valid details)	Fill registration form & submit	Valid name, email, password	Registration successful, redirected to login
TC-05	Register with already registered email	Enter duplicate email	email used before	Error: *"Email already registered"*
TC-06	Empty fields during login	Click Login without filling fields	—	Show validation message: *"Fields cannot be empty"*

2. Display List of Events

TC ID	Test Scenario	Test Steps	Expected Result
TC-07	Display events after successful login	Login → Navigate to Events page	Events list should load with event names & dates
TC-08	No events available	Backend returns empty list	Show message: *"No events available"*
TC-09	Event list fails to load (server error)	Trigger API failure	Show error: *"Unable to load events"*

3. Event Selection

TC ID	Test Scenario	Test Steps	Expected Result
TC-10	Select a valid event	Click event from list	Event details should display
TC-11	Select event not open for registration	Choose event with closed status	Display message: *"Registration Closed"*
TC-12	Select event with invalid/removed ID	Enter/Click invalid event ID	Error: *"Event not found"*

4. Registration Open Check

TC ID	Scenario	Steps	Expected Result
TC-13	Registration open	Event status = OPEN	Allow user to proceed
TC-14	Registration closed	Event status = CLOSED	Stop flow, show message and exit

5. Queue Registration

TC ID	Scenario	Steps	Expected Result
TC-15	Successful queuing	Click "Register" when open	User is added to queue data structure
TC-16	Queue full	Max participants reached	Show: *"Registration Full"*
TC-17	Duplicate registration attempt	Try to register again	Show: *"Already Registered"*

6. Confirmation & Notification

TC ID	Scenario	Steps	Expected Result
TC-18	Successful registration confirmation	After queue insert	Display: *"Registration Successful"*
TC-19	Notification sent	Complete registration	Notification pushed to stack/queue
TC-20	Notification failure	Trigger fail scenario	Show: *"Notification could not be delivered"*

7. Negative / Boundary Test Cases

TC ID	Scenario	Steps	Expected Result
TC-21	Very long username	Enter 100+ chars	Validation error
TC-22	Special characters in name fields	Enter invalid characters	Validation error
TC-23	Slow internet/no connection	Disconnect internet & perform action	App shows connection error
TC-24	Multiple rapid registration clicks	Double-click register	Only one entry in queue

8. System / Performance Test Cases

TC ID	Scenario	Expected Result
TC-25	Load test for 1000 login requests	System handles without crash
TC-26	Queue scalability	Queue handles large number without error
TC-27	Notification stack under load	Stack does not overflow/crash

9. Security Test Cases

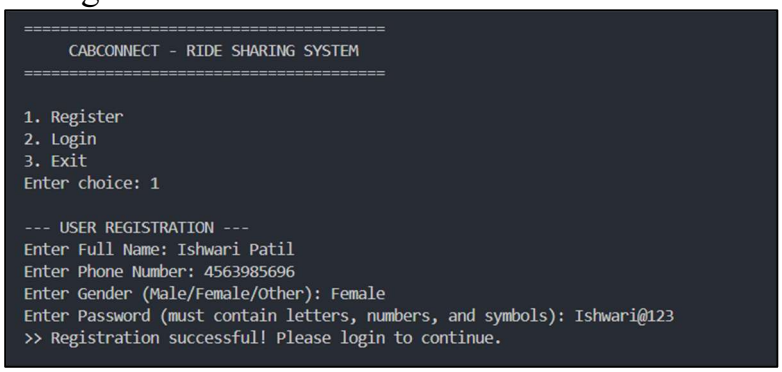
TC ID	Scenario	Expected Result
TC-28	SQL Injection in login	Inputs blocked & sanitized

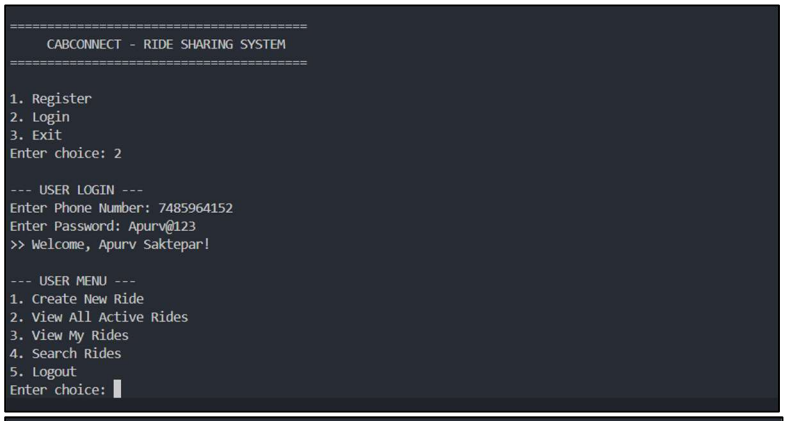
TC ID	Scenario	Expected Result
TC-29	Unauthorized event access	System denies access
TC-30	Session timeout	User is logged out after inactivity

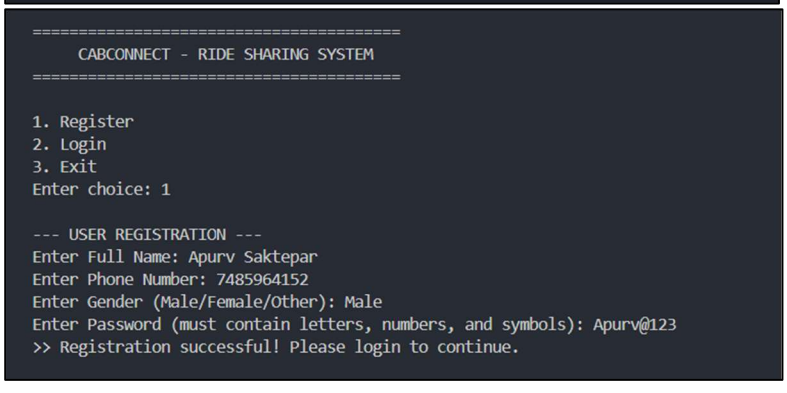
SCREENSHOTS OF RESULTS

---

1. Registered 2 Accounts







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

```
--- 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!
```

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

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.

CONCLUSION

The CabConnect system shows how core programming concepts of OOP, file handling, and logical decision-making can be used to solve real transportation challenges. With features like ride matching and fare splitting, it offers an affordable and organized travel solution. Its simple console-based design keeps the project lightweight and easy to learn, while still demonstrating effective system thinking and modular design. Overall, CabConnect proves that even a basic text-driven application can deliver practical, scalable, and sustainable mobility benefits.

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.