

BỘ GIÁO DỤC VÀ ĐÀO TẠO  
ĐẠI HỌC KINH TẾ TP HỒ CHÍ MINH (UEH)  
TRƯỜNG CÔNG NGHỆ VÀ THIẾT KẾ



## ĐỒ ÁN MÔN HỌC

### ĐỀ TÀI

# XÂY DỰNG CHƯƠNG TRÌNH KẾT NỐI TÀI XẾ

Học Phần: Lập trình hướng đối tượng

Danh Sách Nhóm:

1. NGUYỄN THỊ KỲ DUYÊN
2. PHẠM ĐỨC PHÚ
3. DƯƠNG QUANG PHÚC
4. NGUYỄN LÊ ĐỨC TRÍ

Chuyên Ngành: CÔNG NGHỆ THÔNG TIN

Khóa: K49

Giảng Viên: TS. Đặng Ngọc Hoàng Thành

Tp. Hồ Chí Minh, Ngày xx tháng 11 năm 2024

# MỤC LỤC

<b>CHƯƠNG 1. PHẦN MỞ ĐẦU.....</b>	<b>3</b>
1.1. Tính cấp thiết của đề tài.....	3
1.2. Giới thiệu về Kỹ thuật lập trình hướng đối tượng (LTHĐT).....	3
1.3. Tầm quan trọng của LTHĐT trong Kỹ thuật phần mềm.....	5
<b>CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ LỚP.....</b>	<b>6</b>
2.1. Phân tích bài toán.....	6
2.2. Thiết kế lớp và Sơ đồ lớp (Class diagrams).....	8
2.3. Cài đặt các lớp chức năng .....	10
<b>CHƯƠNG 3. XÂY DỰNG ỨNG DỤNG .....</b>	<b>16</b>
3.1. Thiết kế giao diện chương trình .....	16
3.2. Phát triển các chức năng của ứng dụng .....	18
3.3. Các kịch bản thực thi ứng dụng .....	18
<b>CHƯƠNG 4. THẢO LUẬN &amp; ĐÁNH GIÁ.....</b>	<b>21</b>
4.1. Các kết quả nhận được.....	21
4.2. Một số tồn tại.....	22
4.3. Hướng phát triển.....	23
<b>PHỤ LỤC .....</b>	<b>24</b>
<b>TÀI LIỆU THAM KHẢO .....</b>	<b>26</b>

# CHƯƠNG 1. PHẦN MỞ ĐẦU

## 1.1. Tính cấp thiết của đề tài

- Đề tài "Xây dựng chương trình kết nối tài xế" thể hiện tính cấp thiết trong bối cảnh hiện nay rất cao, vì các dịch vụ đặt xe đang trở nên phổ biến và đóng vai trò quan trọng trong xã hội hiện đại. Việc phát triển một hệ thống kết nối tài xế hiệu quả không chỉ góp phần tối ưu hóa việc vận chuyển, giảm bớt tình trạng kẹt xe mà còn hỗ trợ khách hàng trong việc di chuyển một cách an toàn, tiện lợi và nhanh chóng. Hơn nữa, ứng dụng này sẽ giúp cải thiện khả năng quản lý và giám sát các chuyến xe, nâng cao trải nghiệm khách hàng, đồng thời tạo ra cơ hội công việc ổn định cho nhiều người lao động, nhất là trong ngành vận tải công nghệ cao.

- Đề tài này cũng mang ý nghĩa khoa học và thực tiễn lớn khi khai thác các nguyên lý lập trình hướng đối tượng (OOP) để xây dựng một hệ thống có tính tương tác cao, dễ bảo trì và mở rộng. Việc áp dụng OOP giúp hệ thống dễ dàng nâng cấp, tích hợp các chức năng mới như quản lý hồ sơ tài xế, theo dõi vị trí và đánh giá an toàn, đáp ứng nhu cầu phát triển lâu dài của thị trường

## 1.2. Giới thiệu về Kỹ thuật lập trình hướng đối tượng (LTHĐT)

- Lập trình hướng đối tượng (OOP) là một phương pháp phát triển phần mềm dựa trên việc mô hình hóa các đối tượng trong thế giới thực. Thay vì xử lý dữ liệu qua các dòng lệnh, OOP giúp lập trình viên xây dựng các **lớp** (classes) và **đối tượng** (objects) có thể tương tác với nhau trong chương trình. Mỗi đối tượng là một đơn vị bao gồm cả dữ liệu và các phương thức (methods) để xử lý dữ liệu đó, giúp chương trình dễ quản lý và dễ hiểu hơn.

- Một phần mềm OOP được xây dựng xoay quanh các đối tượng, mỗi đối tượng được định nghĩa bởi các thuộc tính và hành vi riêng. Phương pháp này ưu tiên việc tổ chức các đối tượng thay vì tập trung vào logic xử lý, phù hợp cho các ứng dụng lớn và phức tạp.

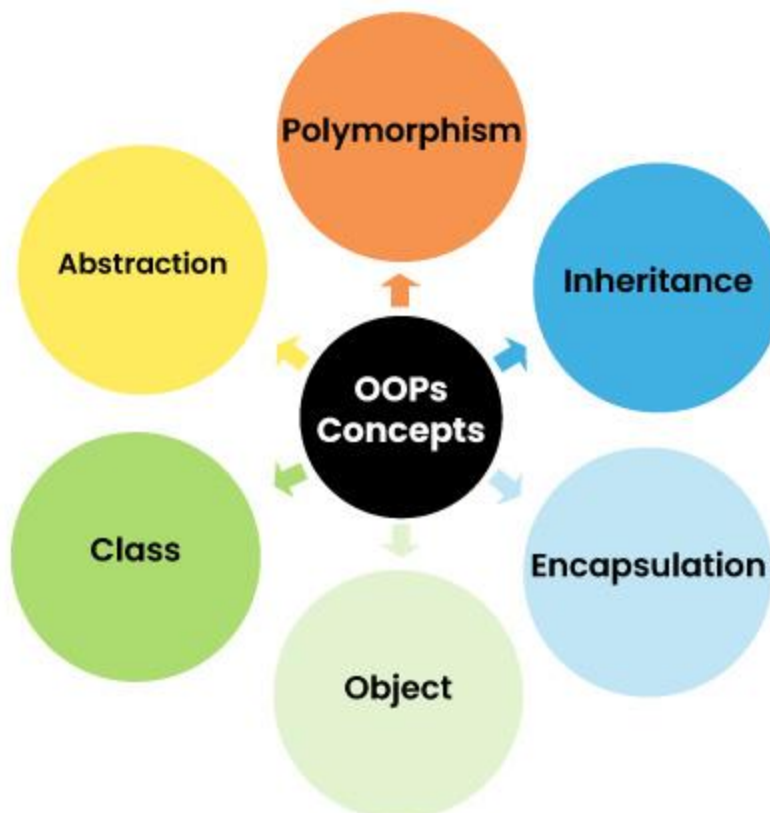
- Phương pháp này dựa trên bốn nguyên lý chính: **tính đóng gói, tính kế thừa, tính đa hình, và tính trừu tượng**:

- **Đóng gói:** Tính đóng gói giúp bảo vệ dữ liệu bên trong đối tượng, ngăn chặn sự truy cập từ bên ngoài trừ khi được phép. Điều này giống như cách các viên thuốc trong vỉ thuốc được bọc kín để bảo vệ khỏi tác động từ bên ngoài. Thông qua đóng gói, dữ liệu và phương thức của một đối tượng được bảo vệ, chỉ cho phép truy cập theo những cách được định nghĩa trước.
- **Trừu tượng hóa:** Trừu tượng hóa giúp đơn giản hóa sự phức tạp bằng cách chỉ hiển thị những chi tiết cần thiết và che giấu các chi tiết thực hiện. Điều này giống như

cách bạn sử dụng điện thoại mà không cần biết chi tiết bên trong nó hoạt động như thế nào. Trừu tượng hóa giúp dễ dàng cập nhật và bảo trì mã nguồn mà không làm ảnh hưởng đến cách sử dụng các đối tượng.

- **Kế thừa:** Kế thừa cho phép tạo ra một lớp mới dựa trên một lớp đã tồn tại, thừa hưởng các thuộc tính và phương thức của lớp cũ. Điều này giúp tái sử dụng mã nguồn một cách hiệu quả, cho phép mở rộng và tùy chỉnh các lớp con mà không cần viết lại logic từ đầu.
- **Đa hình:** Đa hình cho phép một phương thức có thể hoạt động trên các lớp khác nhau, dù chúng có cùng tên nhưng có thể thực hiện khác nhau tùy theo lớp con. Điều này giúp linh hoạt trong việc sử dụng các lớp con như lớp cha, tránh nhầm lẫn và bối rối khi xử lý các đối tượng khác nhau.

- Nhờ vào những tính năng này, OOP giúp phát triển phần mềm một cách có cấu trúc, dễ bảo trì và mở rộng hơn. OOP còn cho phép xây dựng phần mềm phức tạp một cách hiệu quả và giảm thiểu lỗi, nhờ khả năng kiểm soát và tổ chức tốt hơn trong quá trình phát triển.



**Hình 1.1.** Các tính chất của Lập trình hướng đối tượng

### **1.3. Tầm quan trọng của LTHĐT trong Kỹ thuật phần mềm**

- Lập trình hướng đối tượng đóng vai trò quan trọng trong Kỹ thuật phần mềm, đặc biệt khi phát triển các dự án lớn và phức tạp. Cách tiếp cận này giúp chúng ta quản lý mã nguồn dễ dàng và hiệu quả hơn, nhờ vào khả năng tổ chức chương trình thành các đối tượng và lớp riêng biệt. Điều này không chỉ giúp mã nguồn trở nên rõ ràng, dễ hiểu mà còn giúp lập trình viên có thể bảo trì và nâng cấp dễ dàng khi dự án cần sửa đổi.

- Lập trình hướng đối tượng cũng hỗ trợ khả năng tái sử dụng mã một cách linh hoạt. Bằng cách xây dựng các lớp và đối tượng có thể sử dụng lại trong nhiều phần của chương trình, chúng ta tiết kiệm được thời gian và công sức trong phát triển phần mềm. Khả năng mở rộng cũng là một lợi thế lớn của Lập trình hướng đối tượng. Khi dự án cần thêm tính năng mới, lập trình viên chỉ cần xây dựng các lớp mới hoặc mở rộng các lớp hiện có mà không ảnh hưởng đến toàn bộ hệ thống.

- Tính đóng gói và tính kế thừa – hai đặc điểm quan trọng của Lập trình hướng đối tượng – cũng giúp bảo vệ tính toàn vẹn của dữ liệu và giảm thiểu lỗi khi làm việc với mã. Thay vì phải sửa đổi trực tiếp các phần quan trọng, lập trình viên có thể mở rộng các lớp sẵn có để tạo ra chức năng mới. Kết hợp tất cả những yếu tố trên, Lập trình hướng đối tượng thực sự mang lại sự linh hoạt, hiệu quả, và dễ dàng trong việc quản lý dự án phần mềm, làm cho nó trở thành một phương pháp chủ đạo trong Kỹ thuật phần mềm hiện đại.

## CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ LỚP

### 2.1. Phân tích bài toán

#### a. Mô tả bài toán

- Trong thời đại công nghệ số, nhu cầu đặt xe và di chuyển trở nên phổ biến và tiện lợi hơn. Với các dịch vụ đặt xe trực tuyến, người dùng có thể dễ dàng kết nối với các tài xế, lựa chọn loại xe phù hợp, và theo dõi hành trình của mình một cách an toàn.

- Hiểu được nhu cầu đó, nhóm chúng em đã phát triển một chương trình kết nối tài xế, hướng tới việc cung cấp cho người dùng một trải nghiệm đặt xe dễ dàng và hiệu quả, tương tự các ứng dụng phổ biến hiện nay như Grab và Be.

#### b. Đối tượng sử dụng

- **Tài xế (Driver):** những người mong muốn nhận được các đơn đặt xe từ Khách hàng, theo dõi hành trình và hoàn thành chuyến đi để nhận tiền.

- **Khách hàng (User):** là những người có nhu cầu đặt xe di chuyển từ điểm A đến điểm B một cách tiện lợi và nhanh chóng nhờ sự hỗ trợ của những Tài xế.



Hình 2.1. Tài xế và Khách hàng

#### c. Các chức năng chính của chương trình

- **Tìm kiếm và đặt xe:** Khách hàng có thể nhập điểm đón và điểm đến, sau đó hệ thống sẽ tìm kiếm các tài xế khả dụng.

- **Xác nhận và hủy chuyến đi:** Khách hàng có thể xác nhận hoặc hủy đơn đặt xe trước khi chuyến đi bắt đầu.

- **Quản lý thông tin tài xế và khách hàng:** Lưu trữ và quản lý thông tin cá nhân, vị trí và tình trạng tài xế để đảm bảo tính chính xác và an toàn cho các chuyến đi.

- **Lưu trữ lịch sử chuyến đi:** Ghi nhận thông tin về các chuyến đi đã hoàn thành để khách hàng và tài xế có thể dễ dàng tra cứu lịch sử.

- **Quản lý thanh toán:** Cung cấp phương thức thanh toán tiện lợi, ghi nhận và tính toán chi phí mỗi chuyến đi.

#### ***d. Một số vấn đề cần giải quyết***

- Chương trình cần giải quyết một số vấn đề chính để đáp ứng tốt nhu cầu của người dùng. Đầu tiên là vấn đề kết nối và tìm kiếm tài xế phù hợp. Hệ thống phải có khả năng xác định tài xế gần nhất dựa trên vị trí hiện tại và thời gian sẵn sàng nhận chuyến, giúp khách hàng có thể nhanh chóng tìm được phương tiện di chuyển phù hợp.

- Tiếp theo, việc quản lý trạng thái của tài xế đóng vai trò quan trọng. Tài xế có thể cập nhật trạng thái "Sẵn sàng" hoặc "Bận" để hệ thống phân bổ đơn đặt xe một cách chính xác hơn, đồng thời giảm thiểu thời gian chờ đợi cho khách hàng.

- Vấn đề bảo mật dữ liệu cũng là một yếu tố không thể thiếu. Hệ thống cần áp dụng các phương pháp mã hóa để bảo vệ thông tin cá nhân của cả tài xế và khách hàng, đảm bảo an toàn trong quá trình lưu trữ và truyền tải dữ liệu. Điều này giúp người dùng yên tâm khi sử dụng ứng dụng.

- Cuối cùng, hệ thống cần có khả năng tự động tính toán chi phí chuyến đi dựa trên khoảng cách hoặc thời gian di chuyển. Chi phí sẽ được tính toán linh hoạt theo các mức giá quy định để tạo sự minh bạch và tiện lợi cho người dùng. Những yếu tố này là nền tảng quan trọng để chương trình hoạt động hiệu quả và an toàn cho tất cả người dùng.

#### ***e. Mô hình hóa các thực thể trong chương trình***

Dựa vào những phân tích phía trên, chúng ta có thể xây dựng một số lớp và thực thể chính trong chương trình, bao gồm:

- **Lớp User (Người dùng):** Thể hiện thông tin chung của Tài xế và Khách hàng (ID, tên, số điện thoại)

- **Lớp Driver (Tài xế):** Kế thừa từ lớp Người dùng và chứa thêm thông tin trạng thái, vị trí hiện tại, và số chuyến đã hoàn thành.

- **Lớp Rider (Khách hàng):** Kế thừa từ lớp Người dùng, chứa các thông tin của Người dùng (tài khoản, số điện thoại, email, mật khẩu), lưu trữ các yêu cầu đặt xe và lịch sử chuyến đi.

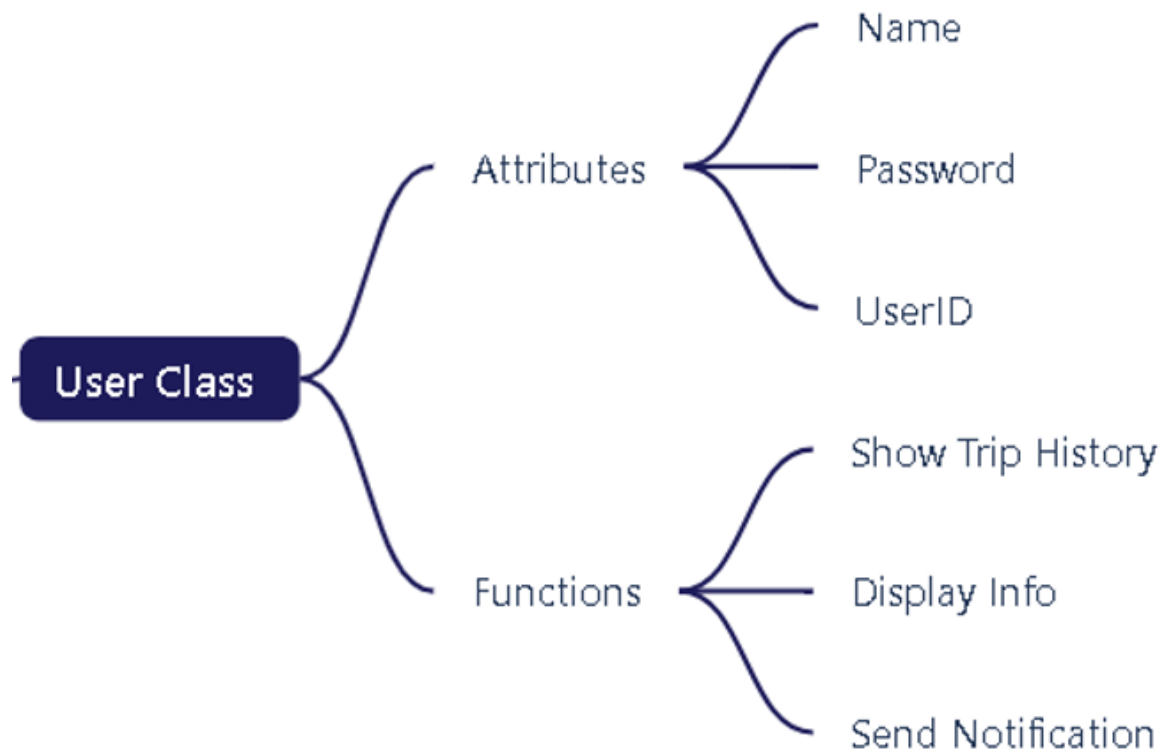
- **Lớp Trip (Chuyến đi):** Lưu trữ thông tin về điểm đón, điểm đến, tài xế, khách hàng, và chi phí chuyến đi.

- **Lớp System (Hệ thống):** lớp điều phối trung tâm, xử lý toàn bộ quy trình của một chuyến đi, từ khởi tạo, quản lý tiến trình, thông báo, đến lưu trữ thông tin vào cơ sở dữ liệu, tạo ra một hệ thống hoàn chỉnh cho việc quản lý đặt xe và các chuyến đi.

Ngoài ra, còn có những lớp hỗ trợ khác như lớp **Vehicle (Phương tiện)** hiển thị thông tin về phương tiện của Tài xế, lớp **TripHistory (Lịch sử chuyến đi)** lưu trữ thông tin chuyến đi của Khách hàng, lớp **Init (Khởi tạo)** chứa những thông tin cần thiết để chạy chương trình và thêm nhiều lớp hỗ trợ khác.

## 2.2. Thiết kế lớp và Sơ đồ lớp (Class diagrams)

- Trong dự án này, sẽ có một lớp User đóng vai trò bản thiết kế cho các thuộc tính và chức năng cơ bản mà một người dùng cần có. Người dùng ở đây bao gồm cả Driver và Rider. Những thuộc tính chung của User có thể bao gồm tên (Name), mật khẩu (Password), ID (UserID), và một số phương thức như showTripHistory (hiển thị lịch sử chuyến đi), displayInfo (hiển thị thông tin cá nhân), và sendNotification (nhận thông báo từ hệ thống). Việc xây dựng lớp User cho phép Driver và Customer kế thừa và sử dụng lại các thuộc tính và phương thức chung, giảm thiểu sự trùng lặp và tiết kiệm công sức phát triển.



**Hình 2.2.** Triển khai lớp User

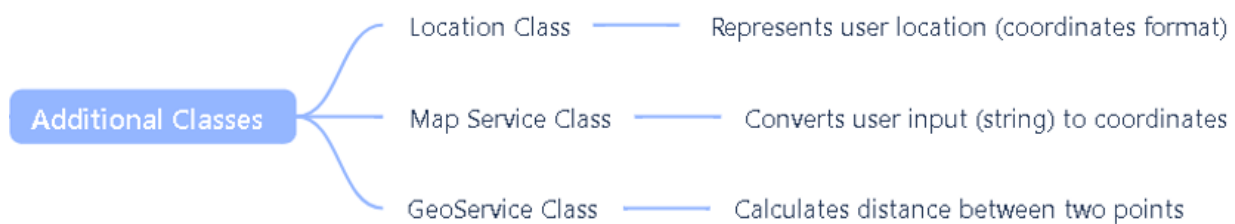
- Về phần hệ thống, sẽ có lớp System để quản lý giao diện và kết nối giữa các User, đồng thời đảm nhận các logic cơ bản của ứng dụng. Để hỗ trợ quy trình làm việc của hệ thống, có thể thêm ba lớp hỗ trợ. Đầu tiên là lớp Location, đại diện cho vị trí của người dùng dưới dạng tọa độ. Tiếp theo là lớp MapService, lớp này chuyển đổi địa điểm đầu vào của người dùng (dưới dạng chuỗi) thành tọa độ trên bản đồ. Cuối cùng là lớp GeoService, cung cấp các phương pháp tính toán giữa hai điểm, chẳng hạn như tính khoảng cách. Thiết kế này cho phép che giấu những chi tiết xử lý phức tạp bên trong GeoService và MapService, trong khi chỉ cung cấp những phương thức cần thiết cho các tính năng liên quan đến bản đồ.



- Xét về các phương thức, MapService và GeoService có thể nằm chung một class. Nhưng để phân biệt các phương thức hỗ trợ việc tạo map object lên Front-end, các phương thức liên quan sẽ được di chuyển vào MapService. Các phương thức và dữ liệu liên quan đến bản đồ được sắp xếp và quản lý chặt chẽ trong lớp MapService, đảm bảo việc xử lý dữ liệu một cách an toàn và có tổ chức.

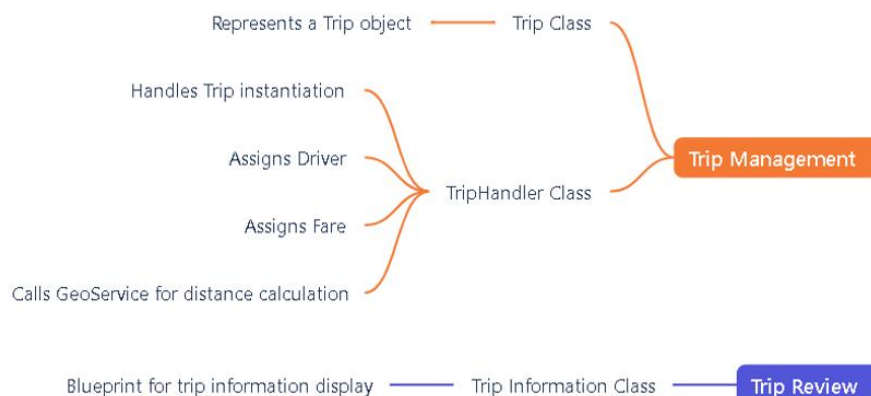


**Hình 2.3.** Triển khai lớp System



**Hình 2.4.** Các lớp hỗ trợ cho lớp System (Location, MapService, GeoService)

- Về tính năng chính là yêu cầu chuyến đi, sẽ tạo hai lớp Trip và TripHandler. Lớp Trip đại diện cho một chuyến đi, còn TripHandler sẽ xử lý việc khởi tạo chuyến đi, như tạo một đối tượng Trip, gán tài xế, tính tiền chuyến đi, và gọi phương thức tính khoảng cách từ lớp GeoService. Ngoài ra, phương thức showTripHistory trong lớp User có thể được các lớp con như Driver và Customer mở rộng, cho phép hiển thị thông tin lịch sử chuyến đi theo cách phù hợp cho từng loại người dùng khác nhau, mang đến sự linh hoạt trong việc xử lý dữ liệu cho từng đối tượng.



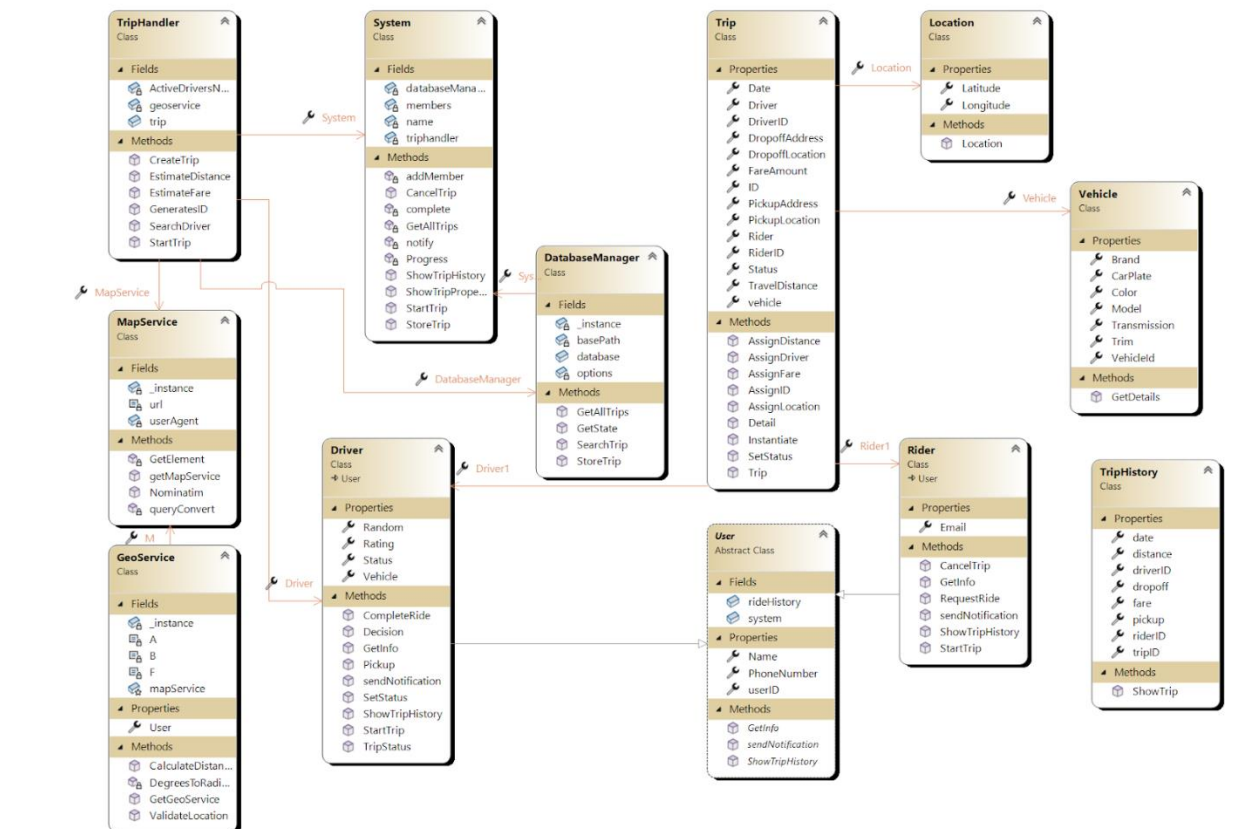
**Hình 2.5.** Triển khai lớp Trip

- Dữ liệu sẽ được lưu trữ dưới dạng JSON, và có thể tạo một lớp DatabaseManager để truy xuất dữ liệu này. Đối với việc hiển thị thông tin chuyến đi khi người dùng muốn xem lại, có thể tạo một lớp làm bản thiết kế cho thông tin chuyến đi cần hiển thị.



**Hình 2.6.** Lớp truy xuất dữ liệu dưới dạng JSON

- Tổng kết lại, ta có được Sơ đồ lớp cho chương trình kết nối tài xế:



**Hình 2.7.** Sơ đồ lớp Xây dựng chương trình kết nối tài xế

## 2.3. Cài đặt các lớp chức năng

Mỗi lớp khai báo được thiết kế để thực hiện một tập hợp các nhiệm vụ cụ thể nhằm quản lý các khía cạnh khác nhau của ứng dụng, bao gồm định vị, điều phối chuyến đi, quản lý thông tin người dùng và phương tiện. Các lớp này là xương sống của hệ thống, đóng vai trò hỗ trợ các hoạt động chính từ việc xử lý yêu cầu của người dùng cho đến tính toán lộ trình và chi phí chuyến đi.

Dưới đây là khai báo hàm của các lớp trong chương trình:

### ***a. Lớp User***

```
public abstract class User
{
    // Basic methods to manage a User object
    // Show user's info
    public abstract Dictionary<string, string> GetInfo();

    // Send a notification to the user
    public abstract void sendNotification(object message, object channel);

    // Show history
    public abstract void ShowTripHistory();
}
```

### ***b. Lớp Rider***

```
public class Rider : User
{
    // Basic methods
    // Receive message from system
    public override void sendNotification(object message, object channel)

    //Show info
    public override Dictionary<string, string> GetInfo()

    //Show trip
    public override void ShowTripHistory()

    // Create a trip
    public async Task<Trip> RequestRide(Dictionary<string, object> pickup,
Dictionary<string, object> dropoff)

    // This method represent the user's decision whether to proceed with the ride
    public async Task<bool> StartTrip(object trip)
    public async Task CancelTrip()
}
```

### ***c. Lớp Driver***

```
public class Driver : User
{
    // Basic methods
```

```

public override Dictionary<string, string> GetInfo()
public override void sendNotification(object message, object channel)
public override void ShowTripHistory()

// UI/UX
public bool Decision()
public bool Pickup()
public async Task<bool> CompleteRide()
public void SetStatus(bool available)

// Marked the trip as completed
public bool TripStatus()

// Passenger picked up, heading to the destination
public bool StartTrip()
}

```

#### ***d. Lớp Trip***

```

public class Trip
{
    public Dictionary<string, string> Detail()
    public Trip Instantiate(Rider rider)
    public Trip AssignID(string id)

    // Updating trip status
    public Trip AssignDriver(Driver driver)
    public Trip AssignLocation(Dictionary<string, object> pickupLocation,
Dictionary<string, object> dropoffLocation)
    public Trip SetStatus(string status)
    public Trip AssignDistance(double distance)
    public Trip AssignFare(double fare)
}

```

#### ***e. Lớp TripHandler***

```

public class TripHandler
{
    // Create instance of the trip.
    public async Task<Trip> CreateTrip(Rider customer, Dictionary<string, object>
pickup, Dictionary<string, object> dropoff)

```

```

// Begins a trip
public Trip StartTrip()

// Searching for nearest driver
public Driver SearchDriver()

// ID generataaah
public void GeneratesID()

// Distance estimator
public double EstimateDistance()

// Fare estimator
public double EstimateFare()
}

f. Lớp TrisHistory
public class TripHistory
{
    public Dictionary<string, string> ShowTrip()
}

g. Lớp System
public class System
{
    // Cast location
    public static Dictionary<string, object> CastLoc(string loc)

    // UI
    public async Task<Trip> ShowTripProperties(Rider customer, Dictionary<string,
object> pickup, Dictionary<string, object> dropoff)

    public async Task<bool> StartTrip(Trip trip)
    private void addMember(object member)
    private void notify(string message)

    // Trip progress
    public async Task CancelTrip()
    private async Task<bool> Progress(Trip trip)
    private void complete()

```

```

//Database Section
public List<TripHistory> ShowTripHistory(string UserID)

//Store trip into database
public void StoreTrip(Trip trip)
private List<TripHistory> GetAllTrips()
}

h. Lớp GeoService
public class GeoService
{
    // Singleton implementation
    public static GeoService GetGeoService()

    // Method to validate the location
    public async Task<Dictionary<string, object>>
ValidateLocation(Dictionary<string, object> location)

    // Helper method to convert degrees to radians
    private double DegreesToRadians(double degrees)

    // Calculate the distance between two locations
    public double CalculateDistance(Location loc1, Location loc2)
}

i. Lớp MapService
public class MapService
{
    public static MapService getMapService()

    // Method to interact with the Nominatim API
    private string queryConvert(Dictionary<string, object> parameters)

    // Use Jsonelement instead of deserialize method
    private async Task<Dictionary<string, object>> GetElement(string jsonResponse)

    // Geocoding API
    public async Task<Dictionary<string, object>> Nominatim(Dictionary<string,
object> location)

```

```
}
```

***j. Lớp Location***

```
public class Location
```

```
{
```

```
    public Location(double Latitude, double Longitude)
```

```
}
```

***k. Lớp DatabaseManager***

```
public class DatabaseManager
```

```
{
```

```
    public static DatabaseManager GetState()
```

```
    // Store trip
```

```
    public void StoreTrip(Trip trip)
```

```
    //Search for user's trips
```

```
    public List<TripHistory> SearchTrip(string UserID)
```

```
    //Get all trips
```

```
    public List<TripHistory> GetAllTrips()
```

```
}
```

***l. Lớp Vehicle***

```
public class Vehicle
```

```
{
```

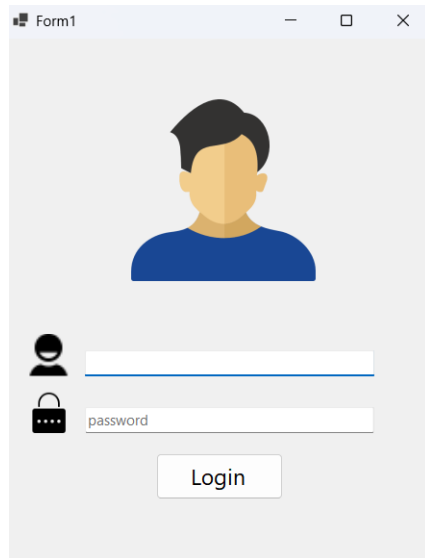
```
    public Dictionary<string, string> GetDetails()
```

```
}
```

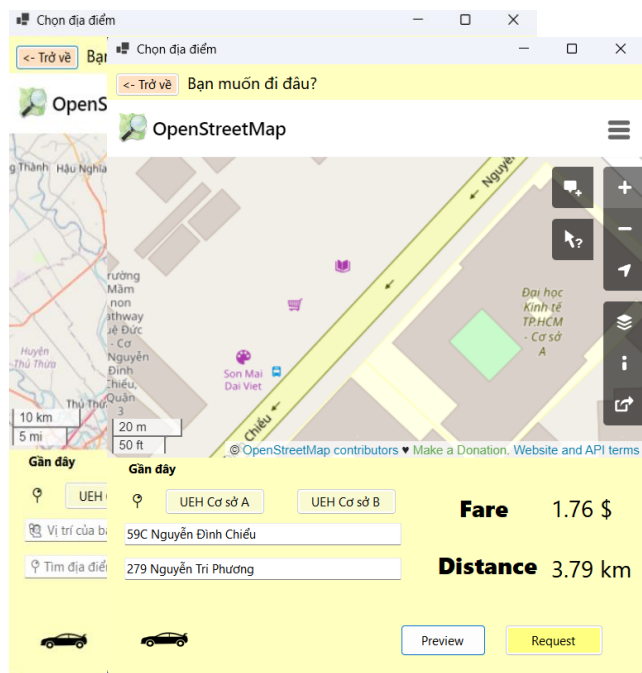
## CHƯƠNG 3. XÂY DỰNG ỨNG DỤNG

### 3.1. Thiết kế giao diện chương trình

#### A. Giao diện đăng nhập

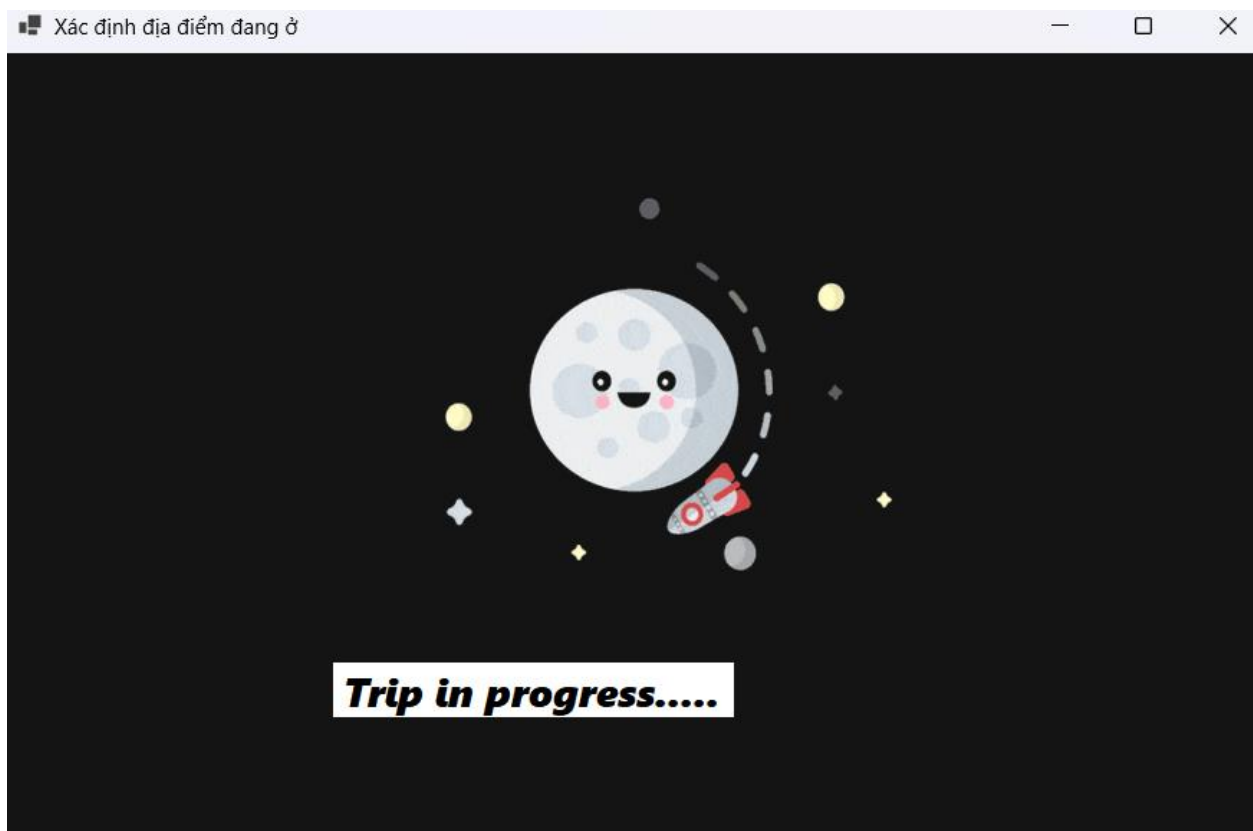
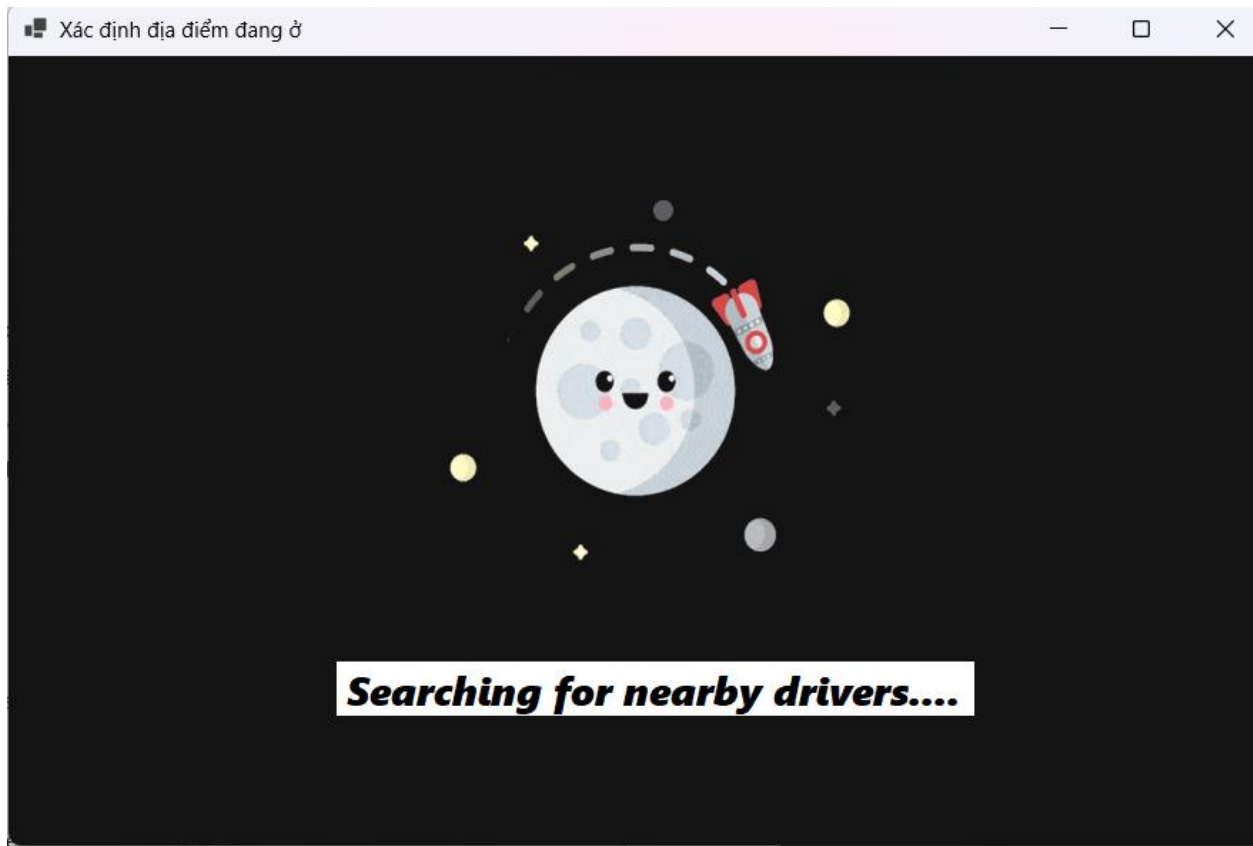


#### B. Giao diện đặt xe

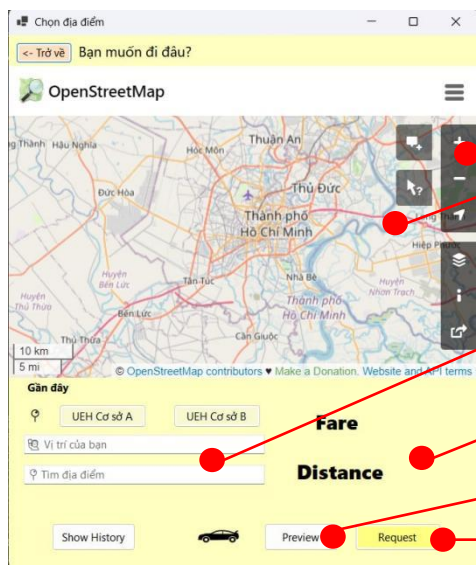


#### C. Giao diện chạy chương trình





### 3.2. Phát triển các chức năng của ứng dụng

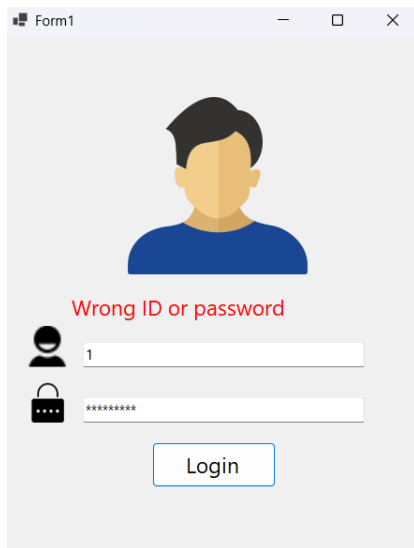


The screenshot shows a web application interface for a ride-sharing service. It features an OpenStreetMap of Ho Chi Minh City. Below the map, there are input fields for 'Gần đây' (Nearby), 'UEH Cơ sở A', 'UEH Cơ sở B', 'Fare', 'Distance', 'Vi trí của bạn' (Your location), and 'Tìm địa điểm' (Find location). At the bottom, there are buttons for 'Show History', 'Preview', and 'Request'. Red lines connect specific UI elements to descriptive text boxes on the right.

- Tiện ích khác
- Bản đồ, hiển thị thông tin vị trí hiện tại của người dùng (Có thể dùng để tìm kiếm vị trí)
- Điểm đón, trả
- Thông tin cơ bản về chuyến đi
- Xác nhận thông tin chuyến đi
- Đặt chuyến

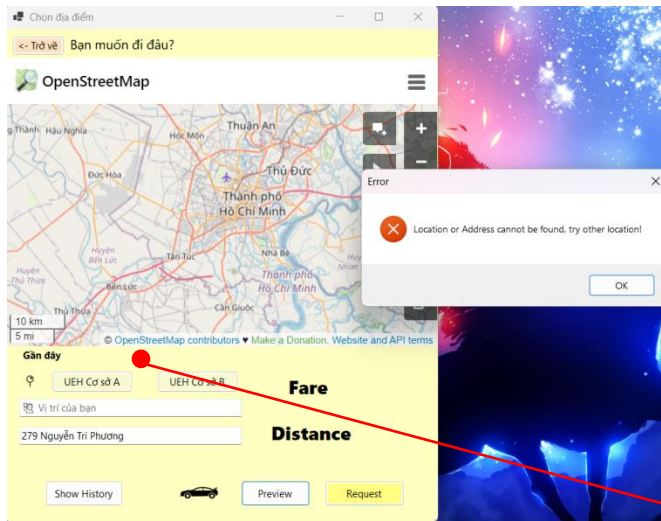
### 3.3. Các kịch bản thực thi ứng dụng

#### 1. Đăng nhập

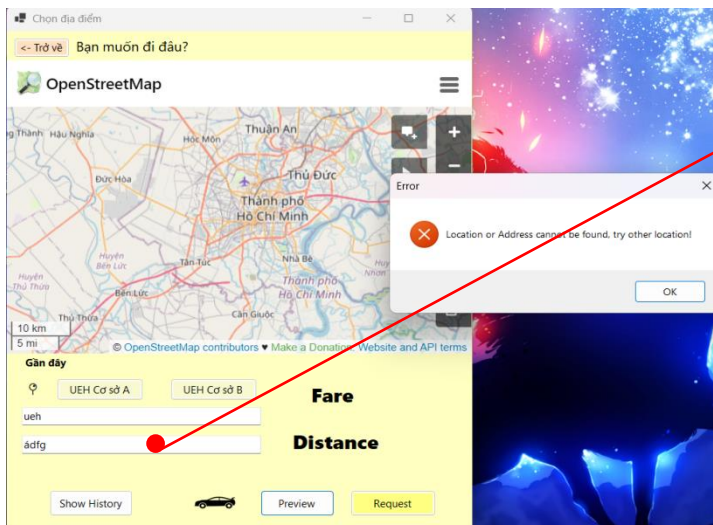


The screenshot shows a web application window titled "Form1". It features a login interface with a user icon at the top. Below the icon, a red error message "Wrong ID or password" is displayed. Underneath the message are two input fields: the first is preceded by a person icon and contains the text "1"; the second is preceded by a padlock icon and contains seven asterisks "\*\*\*\*\*". A "Login" button is positioned below the password field.

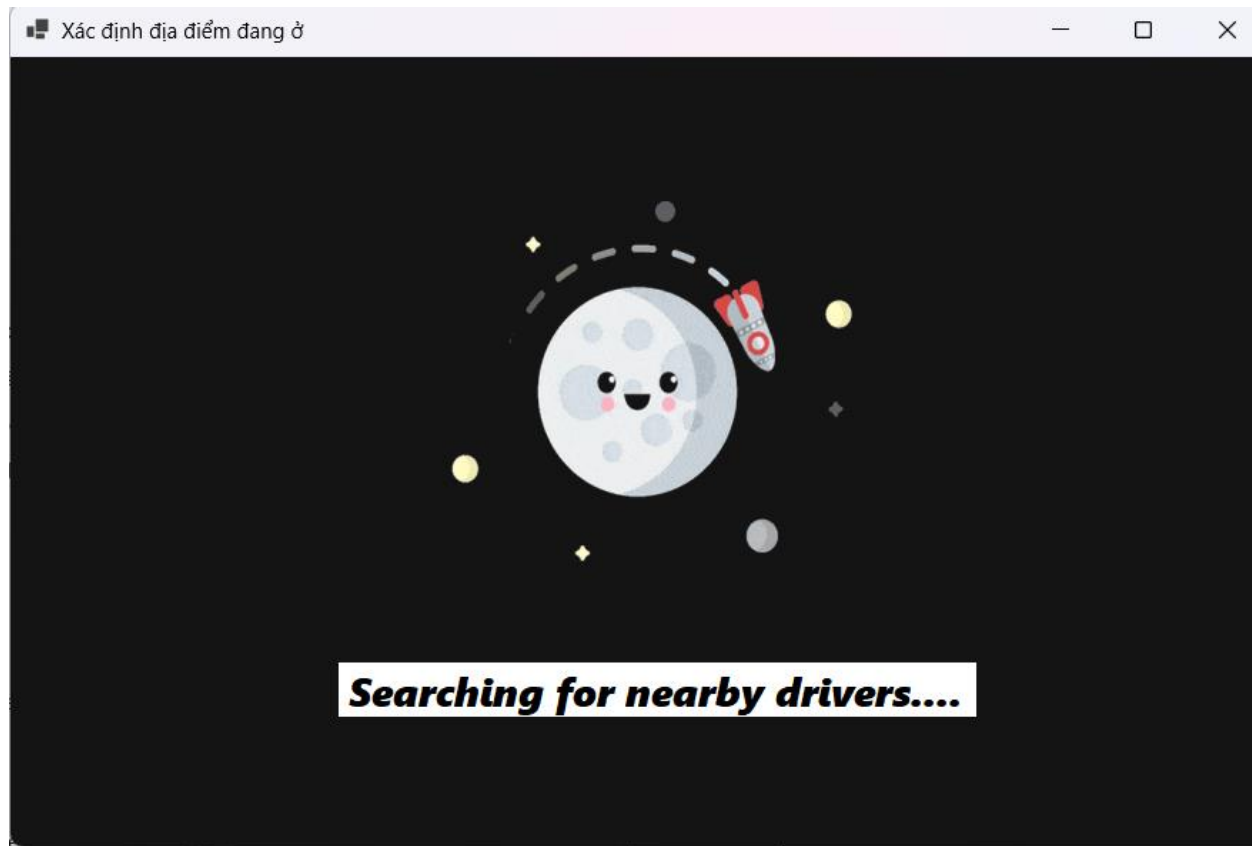
Khi tài khoản, mật khẩu không tồn tại hoặc không hợp lệ sẽ báo lỗi



Khi ô địa chỉ hoặc điểm  
đến bị bỏ trống hoặc  
không tồn tại sẽ báo lỗi



Khi không còn những lỗi trên, chương trình sẽ chạy một ô window, mô phỏng một chuyến đi



## CHƯƠNG 4. THẢO LUẬN & ĐÁNH GIÁ

### 4.1. Các kết quả nhận được

Các tính năng chính đã triển khai:

- **Quản lý chuyến đi:** Các lớp Trip và TripHandler quản lý chi tiết chuyến đi như vị trí đón, trả, khoảng cách và giá cước. TripHandler cung cấp logic chính cho việc tạo và bắt đầu chuyến đi, gán tài xế, và ước tính giá cước.
- **Dịch vụ vị trí:** Lớp GeoService thực hiện việc tính toán khoảng cách giữa các địa điểm và xác nhận tính hợp lệ của vị trí đón và trả bằng cách sử dụng các dịch vụ bản đồ, đảm bảo dữ liệu vị trí chính xác.

- **Tính toán giá cước:** Phương thức tính giá cước sử dụng tỉ lệ dựa trên khoảng cách và các yếu tố ngoại cảnh, như nhu cầu và thời tiết, giúp cung cấp một mô hình giá cước thực tế cho chuyến đi.
- **Quản lý người dùng:** Giao diện Login xác nhận thông tin đăng nhập của người dùng, cho phép truy cập vào giao diện chính của ứng dụng khi đăng nhập thành công.

## 2. Quy trình chuyến đi và phân công tài xế:

- Khi người dùng yêu cầu một chuyến đi, TripHandler sẽ tìm kiếm tài xế gần nhất sẵn sàng, ước tính khoảng cách và tính giá cước. Thông tin này sẽ được cung cấp cho người dùng trước khi xác nhận chuyến đi.
- Quá trình chuyến đi được theo dõi theo thời gian thực, bao gồm cập nhật tình trạng đón khách, trạng thái chuyến đi, và xác nhận hoàn thành khi đến nơi.

## 3. Cơ sở dữ liệu và lưu trữ dữ liệu:

- **Quản lý lịch sử chuyến đi:** Lớp TripHistory lưu trữ và truy xuất các chuyến đi đã hoàn thành, cho phép người dùng xem lại lịch sử chuyến đi của mình.
- **Tích hợp cơ sở dữ liệu:** Dữ liệu được quản lý bởi DatabaseManager, chịu trách nhiệm lưu trữ và truy xuất thông tin chuyến đi cũng như dữ liệu liên quan đến người dùng, đảm bảo khả năng lưu trữ dữ liệu lâu dài.

## 4. Giao diện người dùng và tính tương tác:

- **Giao diện đăng nhập:** Cho phép người dùng nhập thông tin đăng nhập để truy cập ứng dụng.
- **Giao diện đặt chuyến đi:** Giúp người dùng chọn vị trí đón và trả, xem chi tiết chuyến đi và xác nhận hoặc hủy chuyến.

### 4.2. Một số tồn tại

Đề tài "Xây dựng chương trình kết nối tài xế" đã đạt được nhiều tính năng cơ bản, nhưng vẫn còn một số điểm cần tối ưu để tăng hiệu quả và khả năng bảo trì:

- Giao diện Front End xây dựng trên nền tảng Window Forms còn nhiều hạn chế từ đó khiến cho giao diện người dùng (UI) thực tế chưa quá thân thiện và bắt mắt.

- Trong lớp Trip, mặc dù đã có constructor nhưng chỉ khởi tạo một số ít thuộc tính, trong khi nhiều thuộc tính khác cũng quan trọng cho mỗi chuyến đi. Một số thuộc tính như PickupLocation, DropoffLocation, Status, và TravelDistance chỉ được sửa qua các phương thức cụ thể, giúp tăng tính bảo mật, nhưng cần định nghĩa rõ ràng là chỉ đọc hoặc chỉ được gán trong quá trình khởi tạo để tránh nhầm lẫn. Lớp này cũng cho phép method chaining, tạo tiện lợi khi sử dụng, nhưng có thể làm mã phức tạp khó đọc.

- Trong lớp GeoService, phương thức tính toán khoảng cách phức tạp, do đó có thể chia thành các phương thức nhỏ hơn để dễ bảo trì. Bên cạnh đó, phương thức ValidateLocation hiện ném ra ngoại lệ chung, dễ gây nhầm lẫn; do đó, việc tạo một lớp ngoại lệ tùy chỉnh sẽ giúp xác định lỗi tốt hơn. Lớp này cũng sử dụng mẫu thiết kế singleton để đảm bảo chỉ có một thể hiện được tạo, nhưng cần kiểm soát chặt chẽ hơn để tránh lỗi trong môi trường đa luồng.

- Lớp Vehicle chủ yếu được thiết kế như một bộ chứa thông tin xe cộ, nhưng để đảm bảo dữ liệu nhập vào luôn hợp lệ, cần bổ sung kiểm tra định dạng cho các thuộc tính quan trọng như CarPlate hoặc Color. Nếu các lớp khác cũng cần lấy chi tiết dữ liệu dưới dạng từ điển, chúng ta có thể tạo một lớp cơ sở để tránh lặp lại mã, giúp mã nguồn trở nên gọn gàng và dễ duy trì hơn.

### **4.3. Hướng phát triển**

- Tích hợp thêm hệ thống định vị toàn cầu GPS (Global Positioning System). Việc tích hợp tính năng định vị GPS để khách hàng có thể theo dõi vị trí tài xế và ước lượng thời gian đến theo thời gian thực. Điều này giúp khách hàng cảm thấy yên tâm hơn và có thể chuẩn bị tốt hơn khi tài xế sắp tới.

- Bổ sung hệ thống đánh giá, tích điểm thưởng giúp giữ chân khách hàng cũ bằng cách cho phép khách hàng đổi các mã giảm giá,...

- Tạo ra hệ thống phân cấp tài khoản như tài khoản cấp Đồng → Bạc → Vàng → Kim Cương. Việc tạo ra hệ thống này sẽ giúp các tài khoản có cấp cao hơn khi thực hiện xong chuyến đi sẽ sở hữu nhiều điểm thưởng hơn và nhiều ưu đãi hơn so với tài khoản có cấp thấp hơn và việc thăng cấp thông qua việc tích điểm nói trên.

- Tạo ra thêm nhiều phương thức thanh toán như thanh toán thông qua ngân hàng (Internet Banking) hay liên kết thêm với ví điện tử (Momo, Viettel Pay,...) để giúp người dùng có thể dễ dàng thanh toán mà không cần phải đem theo tiền mặt. Mặt khác, việc sử dụng ví điện tử cũng có nhiều lợi ích cho người dùng vì thường ví điện tử sẽ có nhiều ưu đãi hơn cho người dùng so với ngân hàng.

- Đồng thời, xây dựng một ứng dụng di động đơn giản, thân thiện với giao diện dễ sử dụng cũng sẽ giúp trải nghiệm của người dùng trở nên liền mạch và thuận tiện, đặc biệt là đối với các bạn trẻ thường xuyên sử dụng điện thoại.

## PHỤ LỤC

- Đưa toàn bộ mã nguồn lên github và đưa link vào đây
- Viết hướng dẫn cách cài đặt để chạy
- Phân công công việc rõ ràng

<b>Nhiệm vụ</b> <b>Thành viên</b>	<b>Phần code</b>	<b>Phần giao diện</b>	<b>Phần đồ án</b>
Nguyễn Thị Kỳ Duyên	Quản lý class Database Manager, và các tính năng lưu trữ chuyển đi,	Giao diện xem lịch sử chuyển đi	Phần 4: Thảo luận & Đánh giá
Phạm Đức Phú	Vehicle, các class trong folder User. Quản lý dữ liệu của các lớp này.	Giao diện đăng nhập	Phần 3: Xây dựng ứng dụng
Dương Quang Phúc	Quản lý các class trong folder System và GeoFeature, khởi	Giao diện đặt xe và Giao diện Thực hiện chuyển đi	Phần 2: Phân tích và thiết kế lớp
Nguyễn Lê Đức Trí	tạo các tính năng và logic của chúng.	Giao diện đặt xe và Giao diện Thực hiện chuyển đi	Phần 1: Mở đầu

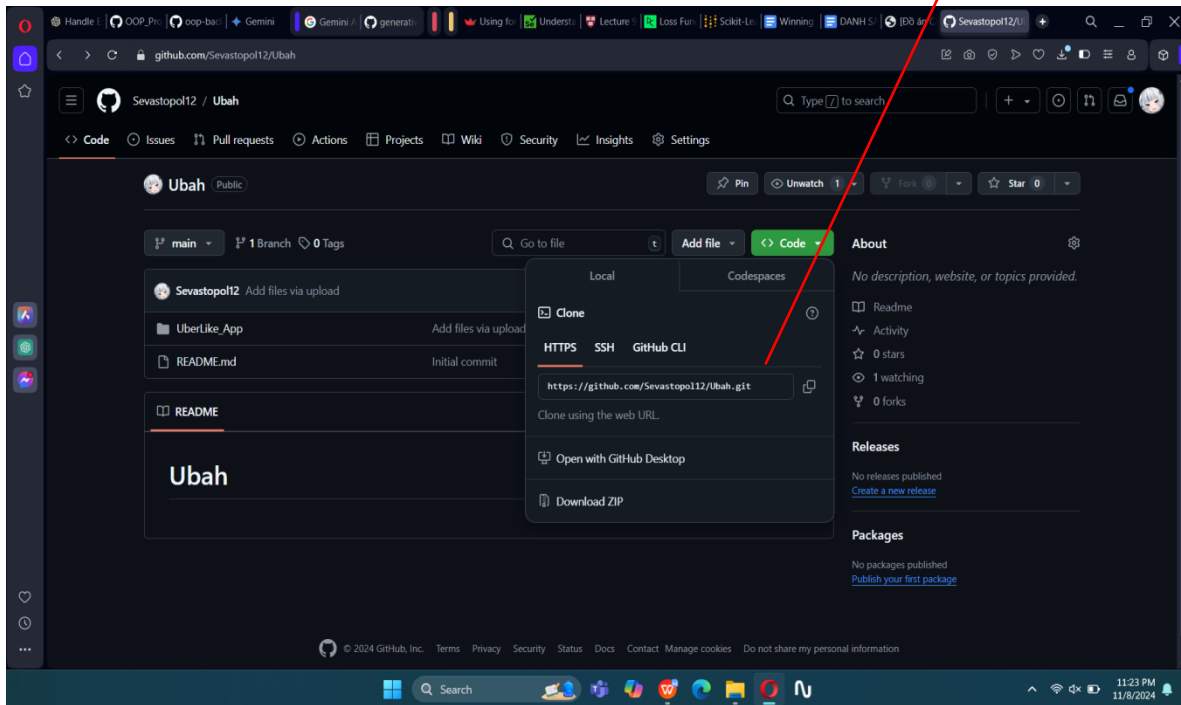
Mã nguồn GitHub :

[https://github.com/Sevastopol12/Ubah/tree/main/UberLike\\_App](https://github.com/Sevastopol12/Ubah/tree/main/UberLike_App)

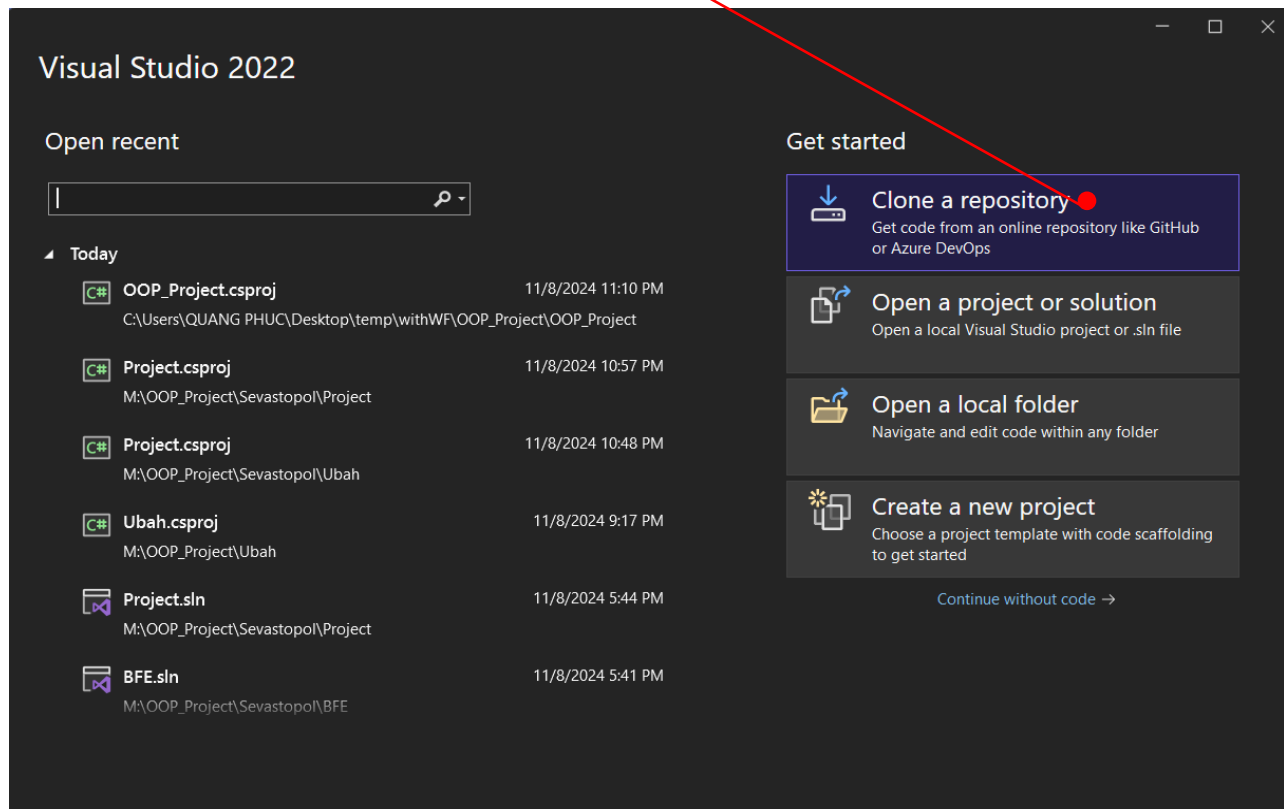


# Hướng dẫn chạy

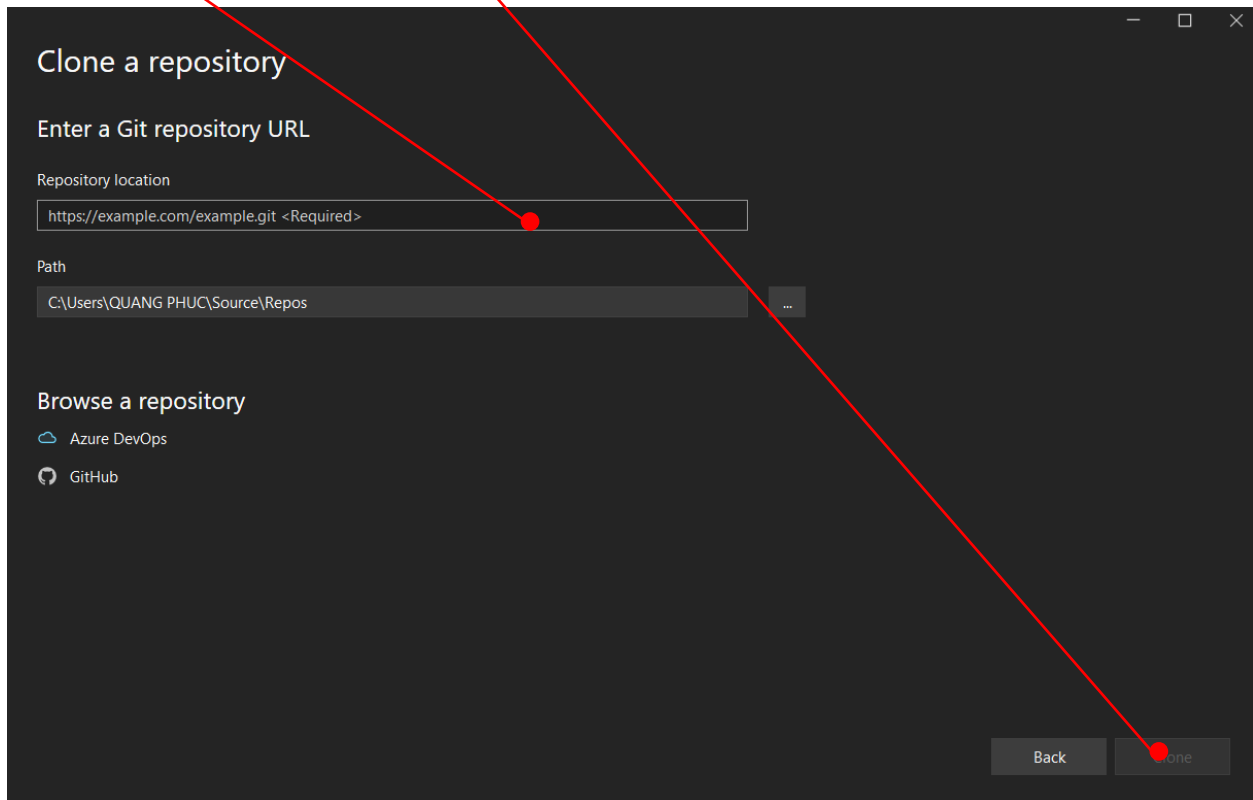
1. Vào link github đã gửi, chọn nút **Code** có màu xanh lục và copy link



## 2. Mở VisualStudioCode, chọn Clone Repository



### 3. Chép link vào và chọn Clone



## TÀI LIỆU THAM KHẢO

1. Tên tác giả. Tên tài liệu, năm xuất bản. Link nếu có.

2.

200lab Blog \_ LTHĐT là gì? \_ 2023

ITviec blog \_ 4 tính chất cơ bản của lập trình hướng đối tượng \_ 2020

TesterProVN \_ Vai trò của LTHĐT trong phát triển phần mềm \_ 2022