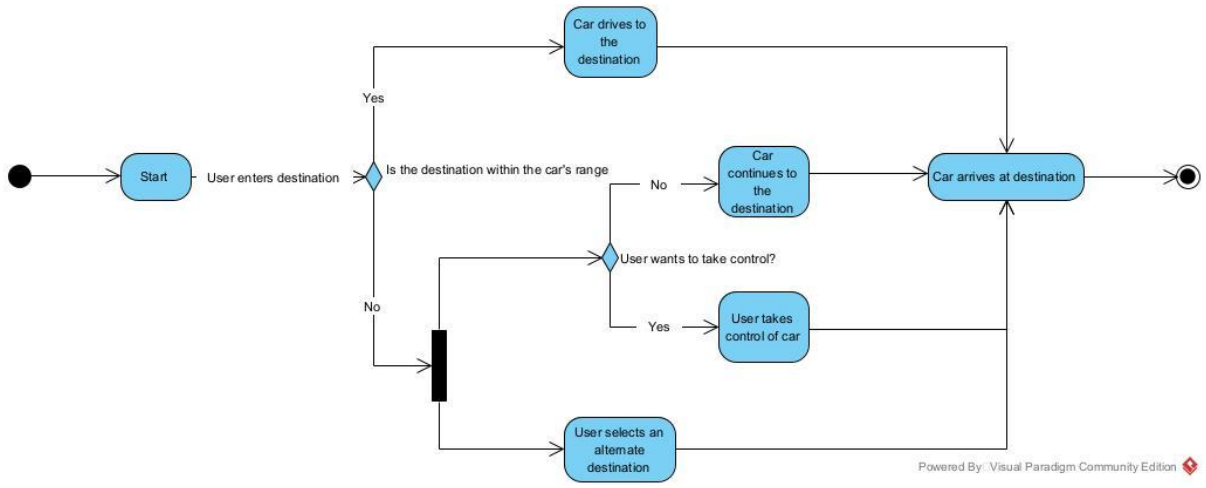
1. **Design an activity diagram which shows the relationships and interactivity between the user's behaviour within the driverless car.**

An activity graph of nodes and flows represents the flow of control (and possibly data) across the many computation steps. Steps can be carried out concurrently or sequentially. Similar to but more potent than a standard flow chart, which only offers sequential and branching structures, an activity involves synchronisation and branching constructs (Booch et al., 2004).

According to Komáromi (2021) An activity diagram is a dynamic UML behaviour graph diagram that highlights the control flow's sequence and circumstances by accentuating the control flow's nodes (activities, actions, and controls) and edges (control flows).

A vehicle that employs a combination of sensors, cameras, radar, and artificial intelligence (AI) to move between locations without a human driver is called a self-driving car (also known as an autonomous or driverless car). A vehicle must be capable of navigating to a predefined location across roads that have not been modified for its usage to be considered entirely autonomous (Lutkevich, N.D.).

The activity diagram below demonstrates the connections and interaction between user conduct inside the driverless car:



A destination is entered into the navigation system of the vehicle. The vehicle then determines whether the location is within its range. If so, the vehicle continues on its way. If not, the user is given the option to choose an alternative location. When the vehicle reaches its destination, it stops, and the user gets off.

There are many methods the user can communicate with the vehicle. They can input a destination, choose a different one, and ask the vehicle to stop and wait. The vehicle can also communicate with the user by giving updates on its location, including the distance remaining before arrival and the anticipated arrival time.

Also, the user can take over driving at any time. When the user takes over, the automobile switches into a mode where it can no longer drive itself, and the user is in charge of steering.

Here are some more specifics on the various activities shown in the diagram:

Start: user inserts a destination into the navigation system of the vehicle.

Decision: The vehicle determines whether the destination is within reach.

Yes: The vehicle travels to the desired location.

No: The user is prompted to choose a different location.

Decision: Will the user assume control?

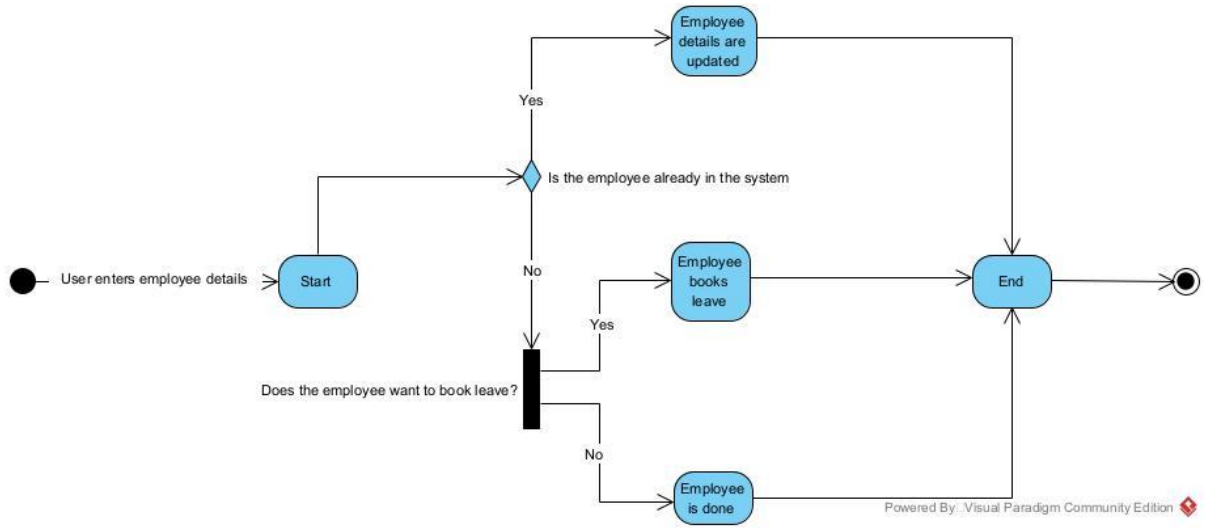
Yes: The driver takes over the wheel.

No: The vehicle keeps moving towards its destiny.

End: The user exits the vehicle after it reaches its destination.

2. **Expand upon the activity diagram by developing a class diagram using UML to support a system with basic employee-related functionality. This should include the retention of employee details and allowing an employee to book a day of annual leave.**

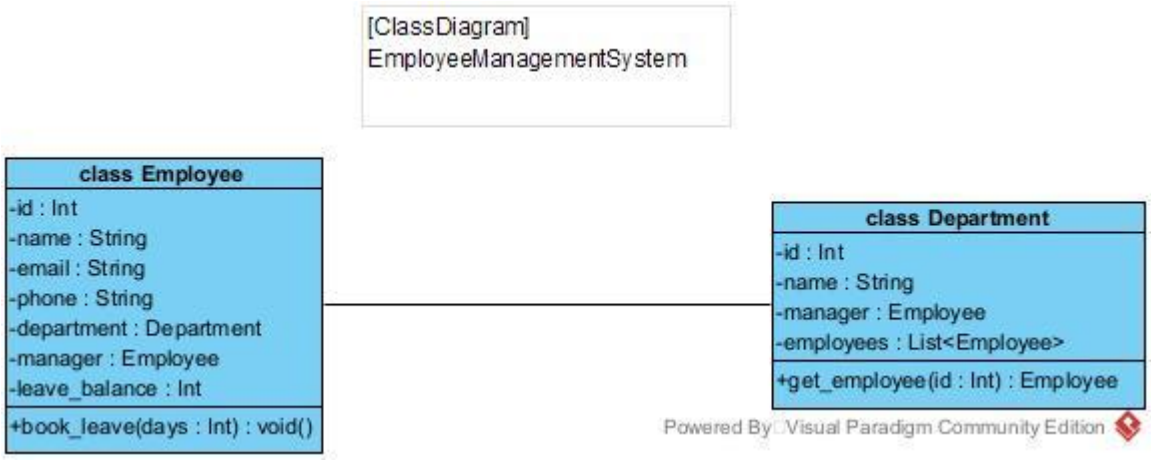
This is the activity diagram:



This activity diagram starts with the user entering their own information, such as their name, manager, department, and phone number. After that, the system determines whether he is already in the system. If the employee is already in the system, the employee information is updated. If the employee is not already in the system, the employee's information is added.

The user has the option to schedule time off. The system will prompt the user for the number of days the employee wants to book if the user chooses to book leave. The employee's leave balance will then be updated by the system.

An enhanced activity diagram for creating a class diagram in UML to support a system with essential employee-related functions is shown below:



The Employee class represents an employee. It has a name, an id, an email, a phone number, a manager, a department, and a balance of leave. The Employee class may reserve days of leave.

The Department class represents a department. It has a list of employees, a manager, an ID, and a name. A department class can identify an employee using their ID.

3. Develop the Python program to implement the class model.

```
class Employee:
    def __init__(self, employee_id, first_name, last_name, email, remaining_days):
        self.employee_id = employee_id
        self.first_name = first_name
        self.last_name = last_name
        self.email = email
        self.remaining_days = remaining_days

    def book_days(self):
        if self.remaining_days > 0:
            self.remaining_days -= 1
            print("Your holidays were booked successfully.")
        else:
            print("You don't have any days left to book your holiday.")
```

References:

Booch, G., Jacobson, I. & Rumbaugh, J. (2004). *Advanced Praise for The Unified Modeling Language Reference Manual, Second Edition. (n.d.).* Available at: https://personal.utdallas.edu/~chung/Fujitsu/UML_2.0/Rumbaugh--UML_2.0_Reference_CD.pdf.

Komáromi, A.L. (2021). A proposal for the usage of OntoUML and UML diagrams for conceptual modeling in philosophy. *hcommons.org*. [online] Available at: https://hcommons.org/deposits/item/hc:34111/ [Accessed 10 Jun. 2023].

Lutkevich, B. (N.D.). *What are Self-Driving Cars and How Do They Work?* [online] Available at: https://www.techtarget.com/searchenterpriseai/definition/driverless-car#:~:text=To%20qualify%20as%20fully%20autonomous.