Ingegneria del Software e Progettazione Web

Progetto A.A. 2022/2023

CampusCarpool

0292416

Simone Niro

INDEX

Introduction

Aim of the documentation

Provide a full description of the software system developed following an approach based on practices of software engineering.

Little overview of the application

CampusCarpool is a system that connects passengers and drivers, making carpooling within the university community an even more eco-friendly experience and accessible to anyone, as there is no fee for a reservation.

This system primarily involves two types of users: **Passengers** and **Drivers**.

There are few, but essentials, key features:

- **Driver ride creation**: they can easily create and publish their available rides. They can specify the date, time, departure location, destination, and the number of available seats in their vehicle.
- **Passenger ride search and booking**: Passengers have the flexibility to search for available rides based on their criteria, such as date, time, and departure/arrival locations. They can then send booking requests for rides that meet their requirements.
- **Driver acceptance or rejection of requests**: Drivers receive booking requests from interested passengers for their rides. They can either accept or reject these requests based on their availability and preferences.

In the application, the user can interact with the software through two different interfaces:

- **Graphical User Interface (GUI)**: users can access the system's functionalities through a minimal graphical user interface, offering a clear overview of available rides and booking requests. Users can intuitively search, view ride details, and manage bookings.
- **Command-Line Interface (CLI)**: the system also offers a well-structured and user-friendly CLI. This interface allows users to perform the same essential operations using text-based commands.

HW and SW requirements

The software and hardware requirements:

- RAM: 2GB of free RAM
- CPU: any modern CPU

- Disk Space: 3.5GB
- Monitor resolution: 1024x768
- Operating System: Microsoft 8 or later, macOS 10.14 or later, any Linux distributions that supports Gnome, KDE or Unity DE

Related systems, Pros and Cons

There are many related systems:

- **Traditional Ride-Hailing Apps**: several ride-hailing apps (like Uber) offer transportation services. However, they focus primarily on on-demand rides and lack the carpooling and trip-sharing features provided by *CampusCarpool*.
- **Other Carpooling Services**: some carpooling services exist, but they might lack the dedicated focus on campus communities that CampusCarpool offers.

Some pros about *CampusCarpool* are:

- **Student-Centric Approach**: CampusCarpool is tailored specifically for students and offers a platform for easy carpooling within campus communities, creating a sense of trust and familiarity among users.
- **Direct Booking and Request Management**: Unlike many other transportation services, CampusCarpool empowers students and drivers to manage ride requests, bookings, and communication directly within the system, enhancing user control and convenience.

While some cons are:

- **Single-User Limitation**: the system currently supports single-user interactions and doesn't allow concurrent usage from different users, limiting its scalability for larger communities.
- **Limited Ride Customization**: even if *CampusCarpool* offers convenient (free) booking and ride-sharing options, it may not provide the same level of customization and flexibility as traditional ride-hailing apps.
- **Stringent Compatibility Search**: the compatibility search requires users to specify the driver's departure address. This requirement might be too stringent for some users as it assumes that users already know the exact address of drivers.

Absence of a Review Format: the system does not offer a review system with descriptions, which could help users make informed choices.

User Stories

1. As a passenger, I want to know the departure location of each ride, so that I know where I can find the driver.

2. As a driver, I want to view passenger's phone number in the ride request, so that I can contact them if necessary.

3. As a passenger, I want to receive immediate notifications if a driver cancels a ride I've booked, so that I have a chance to re-arrange.

NOTE: the third user story is NOT implemented.

Functional Requirements

1. The system shall provide the form to search a ride with the possibility to insert the ride details*.

2. The system shall show to the driver the requests of the passengers with ride details* and passenger details**.

3. The system shall allow drivers to specify the maximum number of passengers they can accommodate for each ride.

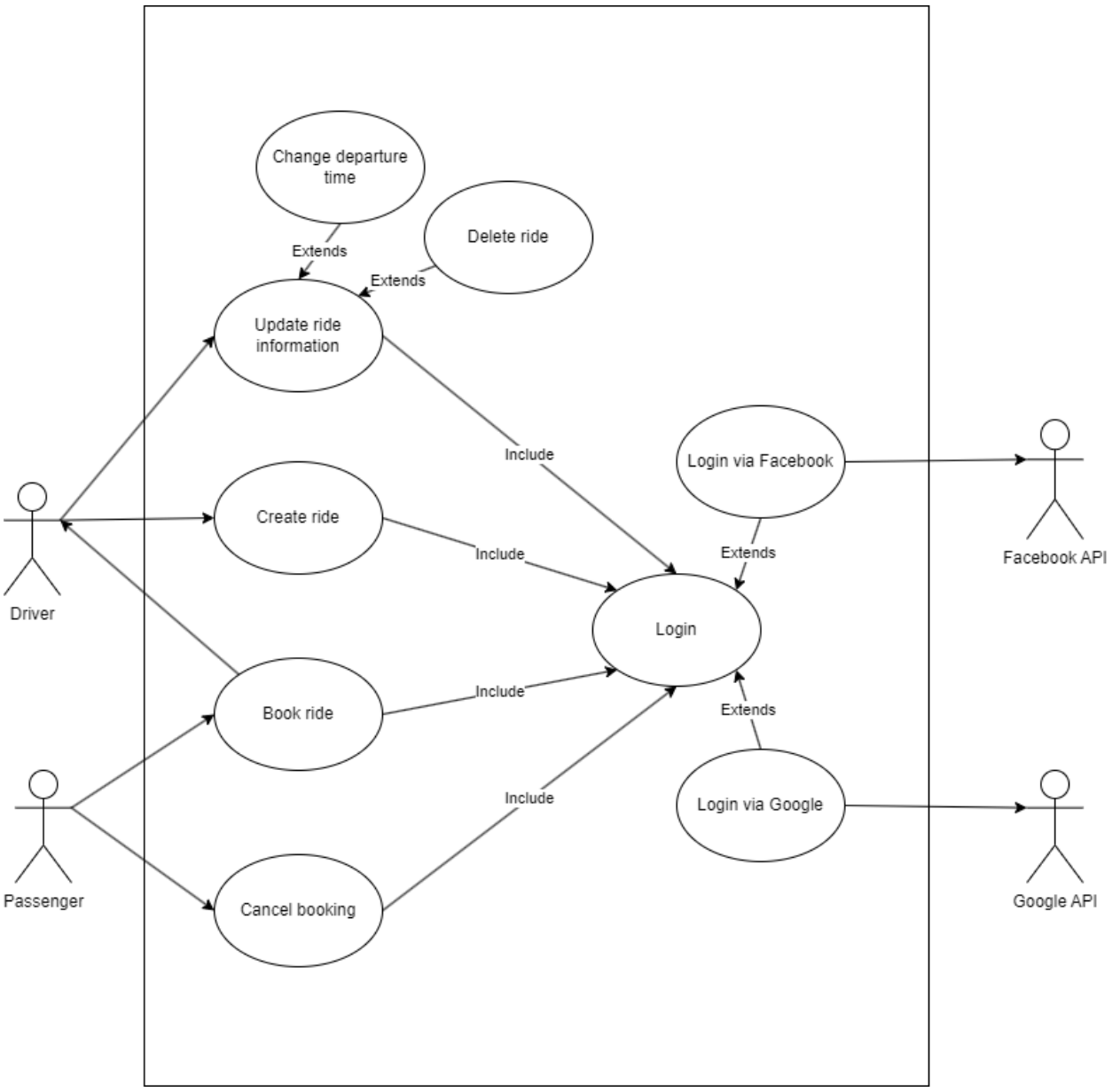
***ride details =** departure location, destination location, departure date, departure time.

****passenger details =** first name, last name, date of birth, phone number.

Use Case

Diagram

Link: https://github.com/simoneniro17/Deliverables_CampusCarpool/tree/main/Use%20Case%20Diagram



NOTE: Use Cases *Cancel booking, Update ride information, Change departure time, Delete ride, Login via Facebook* and *Login via Google* are NOT implemented.

Internal steps

Name: *Book ride (Passenger)*

1. The system authenticates the passenger via the use case Login.
2. The passenger requests to book a ride.
3. The system requests the passenger to insert the departure date.
4. The passenger inserts the departure date.
5. The system requests the passenger to insert the departure time.
6. The passenger inserts the departure time.
7. The system requests the passenger to insert the departure location.
8. The passenger inserts the departure location.
9. The system requests the passenger to insert the destination location.
10. The passenger inserts the destination location.
11. The system gets available rides compatible with the request.
12. The system displays the available rides.
13. The passenger sends a request.
14. The system saves the request.

Extensions

4a. *Invalid date*: the system displays to the passenger an error message ("Date cannot be in the past. Please, choose a valid date.") and allows the passenger to insert a new date.

11a. *There are no available rides*: the system displays to the passenger the message ("No available rides.").

13a. *The passenger already sent a request for that ride*: the system displays to the passenger an error message ("You already sent a request for this ride!") and terminates the use case.

NOTE: these internal steps do not correspond at all to the final version of the application. Many of the steps are written individually just to meet the requirements of the deliverable, but in the application many of them are implemented on the same page.

Storyboards

Link: https://github.com/simoneniro17/Deliverables_CampusCarpool/tree/main/UI%20Prototypes

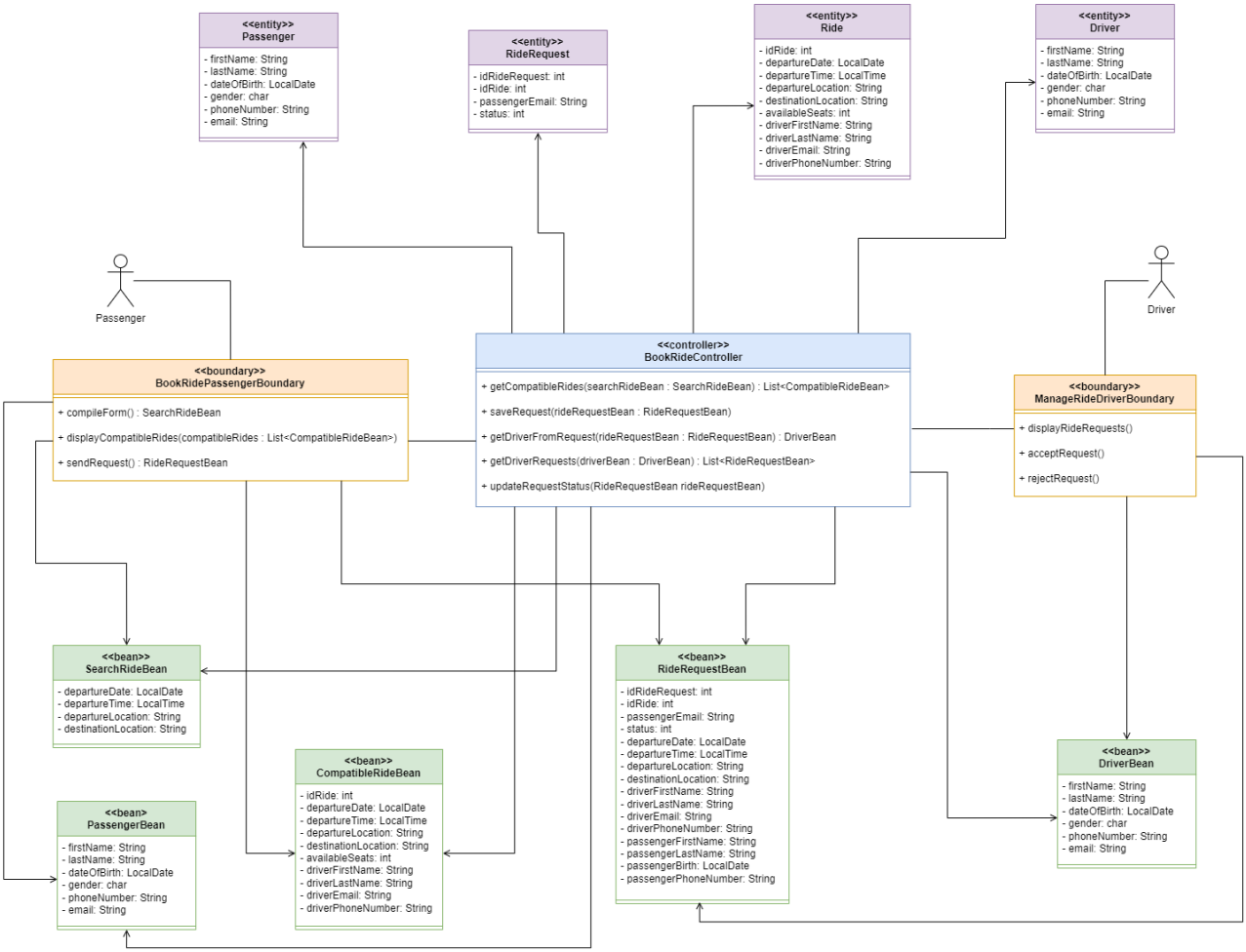
NOTE: As a Driver, start from DriverLogin.html

As a Passenger, start from PassengerLogin.html

Class Diagram

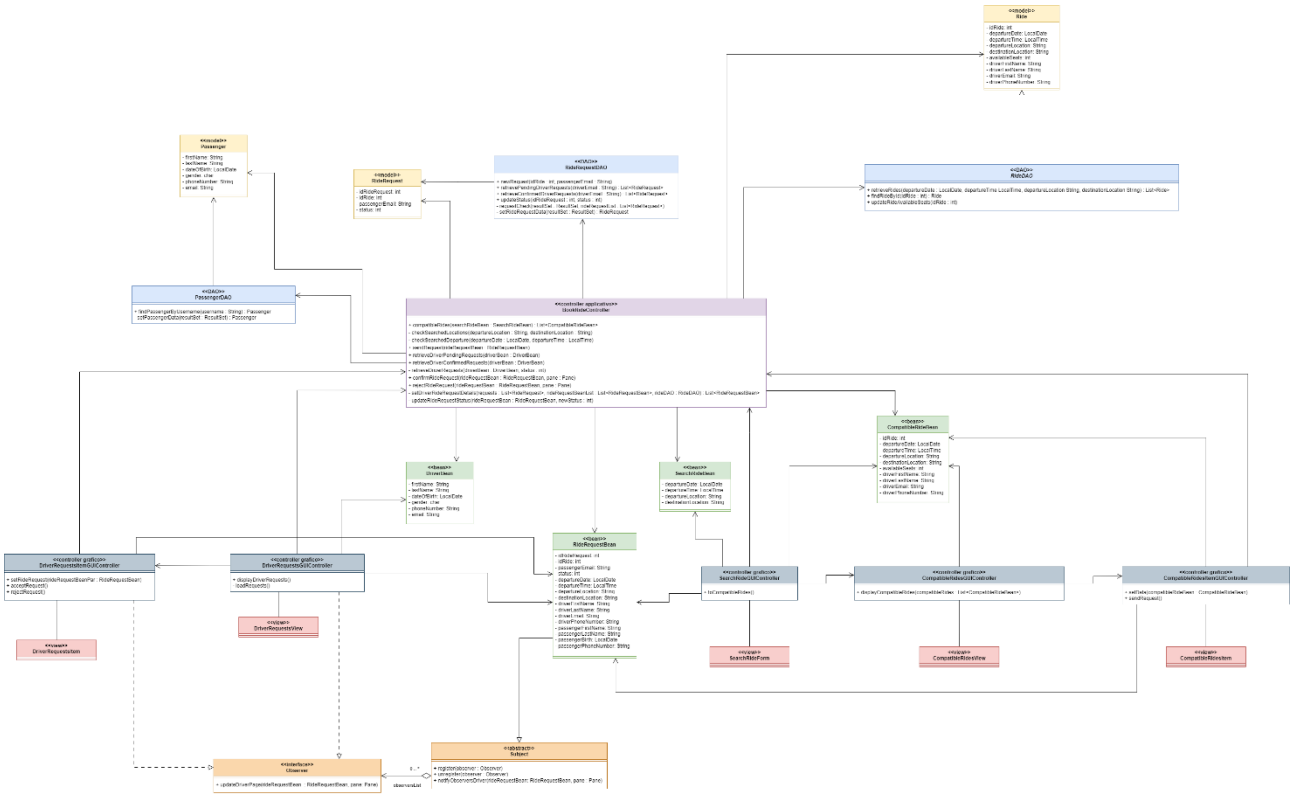
BCE

Link: https://github.com/simoneniro17/Deliverables_CampusCarpool/tree/main/BCE



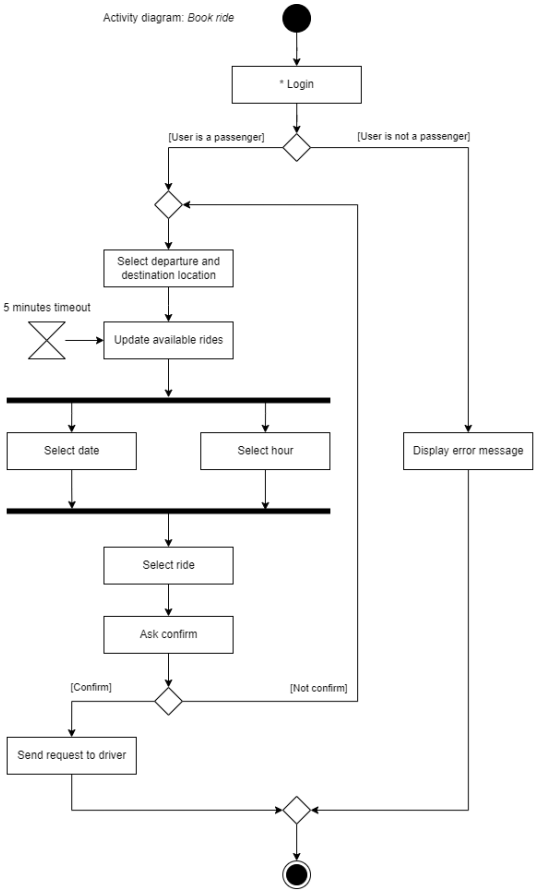
MVC

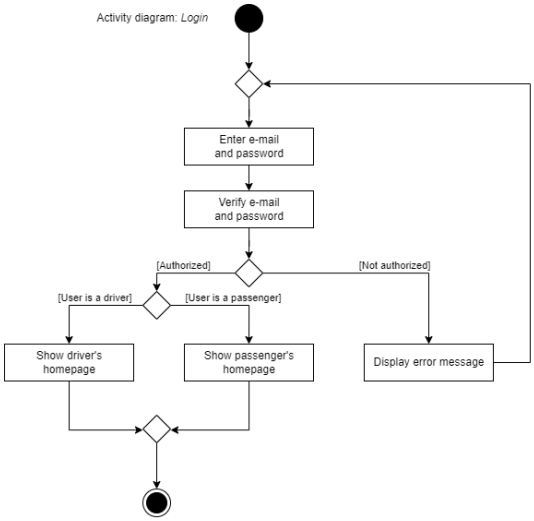
Link: https://github.com/simoneniro17/Deliverables_CampusCarpool/tree/main/MVC



Activity Diagram

Link: https://github.com/simoneniro17/Deliverables_CampusCarpool/tree/main/Activity%20Diagram



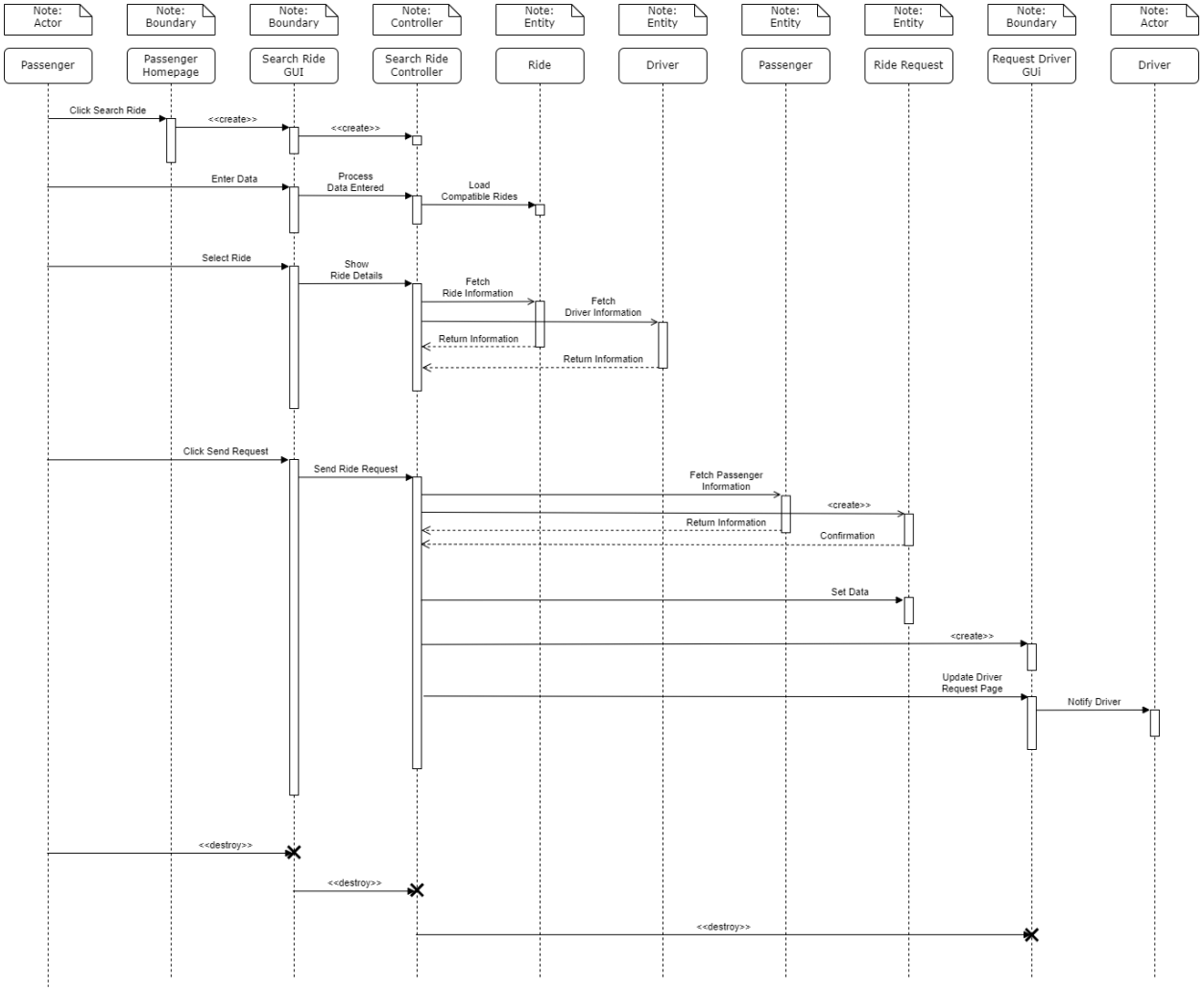


NOTE: this Activity Diagram does not correspond at all to the final version of the application. Many of the actions are written just to meet the requirements of the deliverable and are not implemented.

Sequence Diagram

Link: https://github.com/simoneniro17/Deliverables_CampusCarpool/tree/main/Sequence%20Diagram

Use Case: *Book ride (Passenger)*

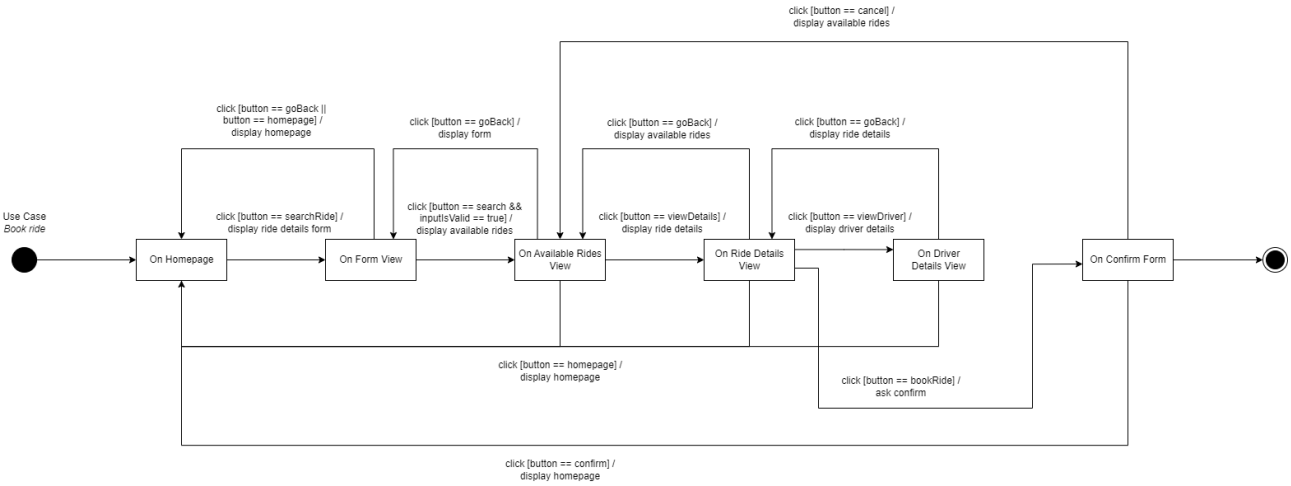


NOTE: in the application all the asynchronous calls are synchronous and the `<<destroys>>` are not implemented since the Garbage Collector oversees the deallocation in Java.

State Diagram

Link: https://github.com/simoneniro17/Deliverables_CampusCarpool/tree/main/State%20Diagram

Name: *Book ride (Passenger)*



NOTE: this State Diagram does not correspond at all to the final version of the application. Some of the states are written individually just to meet the requirements of the deliverable, but in the application, they are implemented on the same page or not implemented.

Testing

Selenium Test Via GUI

Link: https://github.com/simoneniro17/Deliverables_CampusCarpool/tree/main/Test%20Selenium%20GUI

Selenium Test Via API

Link: https://github.com/simoneniro17/Deliverables_CampusCarpool/tree/main/Test%20Selenium%20API

Code

Link: https://github.com/simoneniro17/CampusCarpool

Video

Link: https://github.com/simoneniro17/Deliverables_CampusCarpool/tree/main/Video

SonarCloud

Link: https://sonarcloud.io/summary/overall?id=simoneniro17_CampusCarpool