2DV608

Assignment 3
Design Document

Author1: Zejian Wang <zw222bb>
Author2: Katarina Simakina <es225hi>
Semester: Spring 2023
Supervisor: Mauro Caporuscio

Contents

1 Assumptions and dependencies

1.1 Critical areas of the application:

- Mainly users have smartphones.
- The application supports Android and iOS systems.
- The application route matching algorithms are accurate and efficient.
- The application withstands high-concurrency as end users are more likely to use the application in certain periods of time during a day.
- The payment system is secure and efficient. End users should do the payment without any issues.
- The rating system is accurate as it is critical for users to agree on the carpooling.

1.2 Availability of important resources

- Database server MySQL is used to store and retrieve data.
- Web Server Apache Tomcat is used for handling the requests from clients.
- Application Server is used for the carpooling software to host.
- The GPS application installed on the phone interacts with Google Map to show a real-time map and location of users.
- Swish and Bankid are key for users to do the payment.
- The development team is small, so development milestones should be scheduled appropriately to satisfy the requirements.

1.3 Design approach discussion

There are three design approach options for us to choose – Top-bottom, Bottom-top, a mix of both top-down and bottom-top approach. Top-bottom design is first to design the high-level structure of the carpooling system, and then gradually work down into smaller and detailed decisions. When taking this approach, overall functionalities such as registration, ride searching, route matching and payment processing in the carpooling system are defined and structured first. And then the detailed decisions such as the searching interface and the route matching algorithm. Bottom-up design starts at small and reusable low-level modules such as creating route matching algorithms, searching interface, and then deciding how these modules will be put together to create high-level structure. [1]

The approach of a mix of both top-bottom and bottom-top design will be used in the carpooling system as it can benefit from both design approaches. Top-down design can give a good overall structure to align with the requirements, and the bottom-up design approach can be used to create

reusable specific functionalities of the software. This approach makes the design structure more reasonable, and it saves time as it reduces the probabilities of modifying the design during the whole design process.

2 Architecturally significant requirements

ID	Name	Requirement Description
FR1	User Registration	The user should be able to be registered and verified through email address, personal number, phone number and Bankid.
FR2	User Login	The user should be able to log in via username and password or Bankid.
FR3	Search Rides	The passenger should be able to search for rides by entering start and destination addresses in the system.
FR4	Book rides	The user should be able to book rides if there's a match after inputting departure, destination and time.
FR5	Role Switch	The user should be able to change his/her role to the other.
FR6	Ride Information Display	The system should be able to display the time, route and cost of the ride after the carpooling is done.
FR7	Necessary Information Sharing	The necessary personal information should be shared between the passengers and the driver after both accept the ride.
FR8	Real-time Map Display	The system should be able to show the route and current location of the vehicle during the whole trip.
FR9	Payment Method	The system should be able to provide a way to let the passenger do the payment via Swish after both the passenger and the driver confirm the trip is finished.
FR10	Rating	The passenger and the driver should be able to rate each other after the trip is finished. The rating is from 1 to 5 stars of each option associated with a comment.

2.1. Usability

Requirement: FR1 & FR2 & FR5

Rationale: Users use this application by registering through the authentication processes and log in with a specific role(Driver or Passenger).

2.2 Implementing Trip management

Requirement: FR3 & FR4

Rationale: The application architecture should have a reliable and efficient module to find available trips that match the user's input, the cost, time of which should be presented and the whole trip's route should be in real-time map.

2.3 Compatibility with Third parties

Requirement: FR6 & FR8 & FR9

Rationale: As the application should show the device's current location and the real-time map, and processes the payment, third parties service should be imported and compatible with the application.

2.4 Communicability between users

Requirement: (FR7).

Rationale: Necessary information of users should be presented to both sides of the carpooling trip to ensure the communication between each other.

2.5 Improvability

Requirement: FR10

Rationale: A Rating System allows users to rate services which they have used to do the submission of the ratings to improve the services of the drivers as well as the software itself.

3 Sustainability Concerns

3.1 Strategies for reducing the carbon impact.

- Energy-saving features can be designed to be implemented in software to reduce energy consumption. For example, software can be designed to switch to a simplified mode that does not display unnecessary information or can be designed for users to switch to a low-power mode when it is not in use to reduce the number of background threads running.
- Developers writing efficient code and optimizing energy-intensive parts of the software can significantly reduce the energy consumption of the software. Developers can use open source because it is often developed by volunteers who focus on efficiency and optimization
- The development team can use renewable energy resources to power their own hardware. It can partner with renewable energy providers.
- Avoid developing software functionalities that track unnecessary users data as it reduces the overall workload for page loads. [2]
- Developers can work from home and can have virtual collaboration to reduce the need for commuting and in-person meetings to reduce the carbon emissions.

3.2 Measurements of carbon impact

1) Define the boundaries of the carpooling software and the components of the carpooling software:

- Application Server
- Web Server
- Database server
- GPS application
- Front end mobile application
- Network traffic between client and application server and back
- Network traffic between application servers and database servers

2) Choose the ***functional unit***, here is the function of one searching trip.

3) Measuring the ***energy consumption*** of each carpooling software component by using hardwares that directly measures it. And the method to calculate the energy value is *Calculate*, the utilization of servers used will be measured and a model will be used to estimate the energy consumption.[1]

4) The region of the application runs is **Sweden** and the statistic of ***Carbon Intensity*** is from the official source.

5) Calculate the ***Embodied Carbon*** of each component according to the equation `M = TE * (TR/EL) * (RR/TR)`:

- TE = Total Embodied Emissions, the sum of LCA emissions for all hardware components associated with the application server.
- TR = Time Reserved, the length of time the hardware is reserved for use by the software.
- EL = Expected Lifespan, the anticipated time that the equipment will be installed.
- RR = Resources Reserved, the number of resources reserved for use by the software.
- TR = Total Resources, the total number of resources available.

6) From the results of calculation above, we can calculate the ***SCI***(Software Carbon Intensity) of each component of carpooling software according to the equation: `SCI = (E * I) + M Per R`:

- R: Functional Unit
- E: Energy Consumption
- I: Carbon Intensity
- M: Embodied Carbon

7) Finally, The total SCI score of the carpooling software application is the combined score of all the different components.

4 Decisions, constraints, priorities and justifications

Client/server architecture

Rationale: Since it is a progressive mobile application, the architecture of the application will be based on a client/server architecture, as clients make requests to the server, and the server is responsible for providing information requested. Because the carpooling system has much personal information that requires security, and Client-server architecture allows for better security of an application. By keeping sensitive information and logic on the server, it is easier to control access to that information and prevent unauthorized access or manipulation. client-server can also make it easier to manage and update the application as it has better maintenance.

"Thin" Client

Rationale: The server has the responsibility to apply logic and process huge data to present to the client side while there are few responsibilities for the client side to take. Furthermore, A thin client architecture can be easily scaled up or down to meet changing demand and processing and storage resources can be allocated on the server to accommodate changing requirements.

Third-party module

Rationale: As the key features real-time map and payment processing are based on third-party modules such as Google-map and Swish and Bankid, so they are designed to call third party modules instead of creating their own modules. There are also other third party modules that can be used as it can save a significant amount of time and can be easily integrated into. Third parties are also reliable and stable as they are rigorously tested over time.

Java Programming language

Rationale: Java has built-in security features which can help prevent unauthorized access to system resources; Java has community and ecosystem, which means that there are many resources available. There are also many third-party libraries and tools available, which can help accelerate development and reduce the amount of code that needs to be written from scratch. These features make it easier for programmers to develop, deploy, and maintain carpooling software.

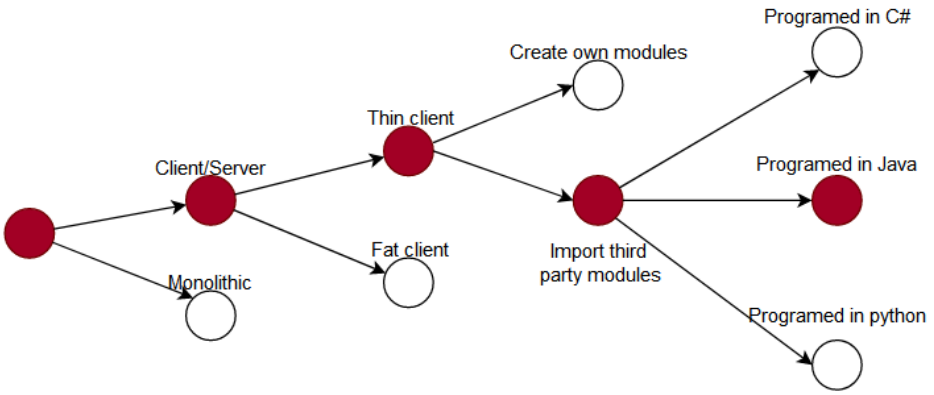

Figure 4-1. Decisions processing

5 Architectural Patterns

The architectural pattern that will be used is Microservice Architecture. In this pattern, the carpool software is divided into small services such as user management, trip management, payment processing, etc. Each component has its own responsibility and is not dependent on other components, as a result, each functionality will not affect other functionalities and each service that can be built, maintained, and deployed independently[4]. The green software pattern that will be used is Avoid tracking unnecessary data, which will reduce time spent on transferring and processing pages.

The above patterns will provide consistency and uniformity by ensuring that each microservice is responsible for a specific functionality. Also, each service can communicate with each other through APIs, which ensures that the application is loosely coupled and can be easily maintained.

As mentioned above, the carpooling software includes three service components: user management service, trip management service and payment management service. The user management service would be responsible for user registration, login and profile management. The trip management service would manage the trip creation, cancellation. The payment processing service would handle payment transactions between passengers and the system.

The design principles of Divide and Conquer, Cohesion, Reduce Coupling, Reusability, Flexibility and Testability are addressed as the carpooling system is break down into multiple independent services, each of it is responsible for its own functionality, not affect other function, each of it can be scaled can be developed, tested and maintained individually, each with each other through its API, reducing coupling as mentioned earlier. Each service is an individual module, which means it is reusable.

6 Architectural views

6.1 Component Diagram

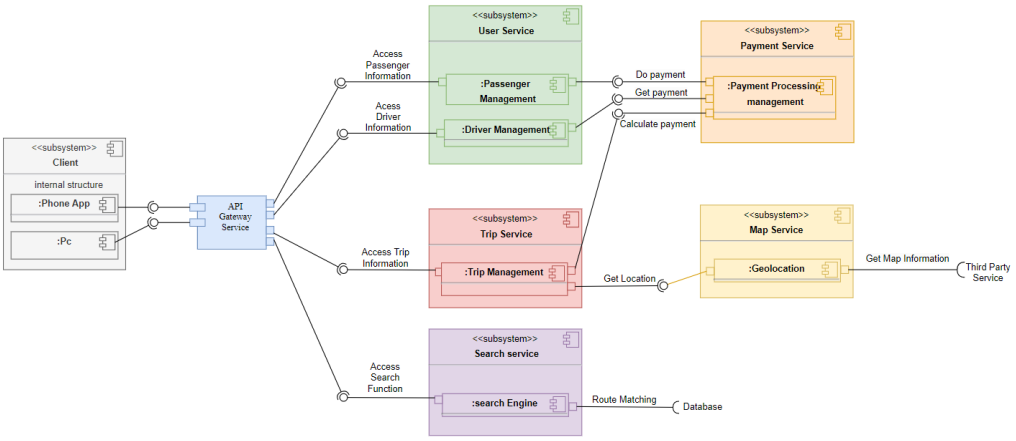

Figure 6-1. Carpooling Component Diagram

As the above figure shows, there are several service components based on the microservices architecture of the carpooling software system. It shows what the subsystems are and how they interact with each other. We can see from the figure that all requests will be handled via API Gateway Service, The User Service is responsible for user management, including passenger and driver profile management, authentication and authorization, it calls the Payment Service to do the payment and get the payment. Trip Service is responsible for trip management, containing trip creation, cancellation and trip booking. It calls the Payment Service to calculate the trip payment. It also calls Map Service to get the current location and the real-time map which should call the third party map service. The search service is responsible for the route matching, which should call the database.

6.2 Deployment Diagram

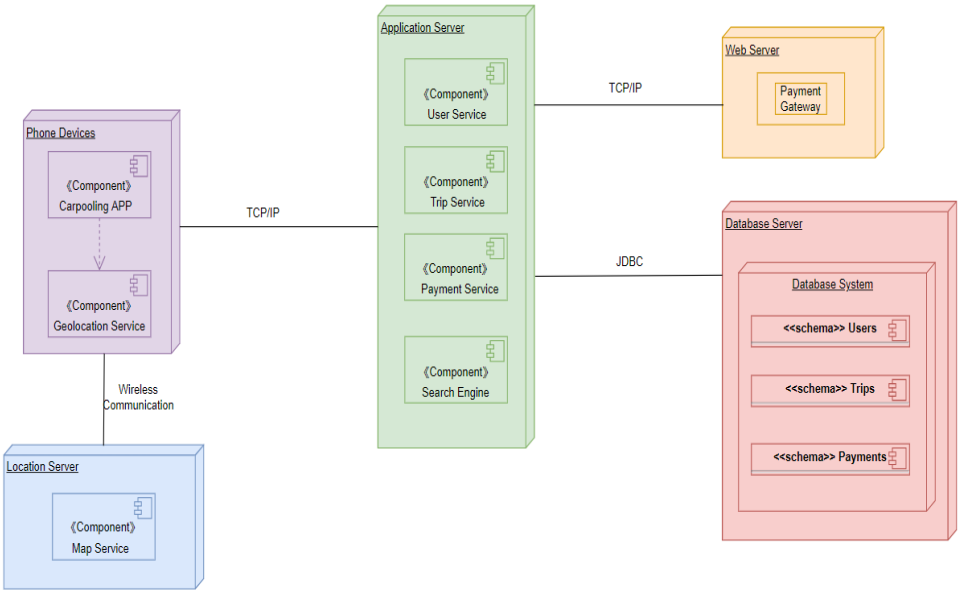

Figure 6-2. Deployment Diagram

Deployment diagram describes how the carpooling system application will be deployed. This application will be used by users via their smartphone. Smart phones have an embedded geolocation service. Geolocation service accesses the Location Server via wireless communication. This server contains a component Map Service which provides real-time maps. Phone device will connect with the Application Server via TCP/IP. Application Server has the following components: user service, trip service, payment service and search engine. All information processed will be stored in Database Server. Connection between Application Server and Database Server is JDBC (java database connectivity) which provides a standard interface for accessing and manipulating databases. This connection will help to improve performance and security in the application. Database server has a Database system which consists of a database of the users, trips, payments. Application Server has connection with Web server via TCP/IP. Web Server will get requests. This Server has a payment gateway which will help the process of the requests to create and process payments.

6.3 Sequence Diagrams

The sequence diagrams below show activities of some functions of carpooling software respectively.

6.3.1 User Login

The sequence diagram of user login showed below, after user entering the username and its password via GUI, which interacts with User Service, User Service made a request to database to check if Username and its password matches, the message of User account will be replied to User Service if Username and its password match, then the message of Successfully login will be returned to Gui; The message of User account not found will be returned to User Service if the username and its password not match, then the message of Failure login will be return to Gui to remind the user.

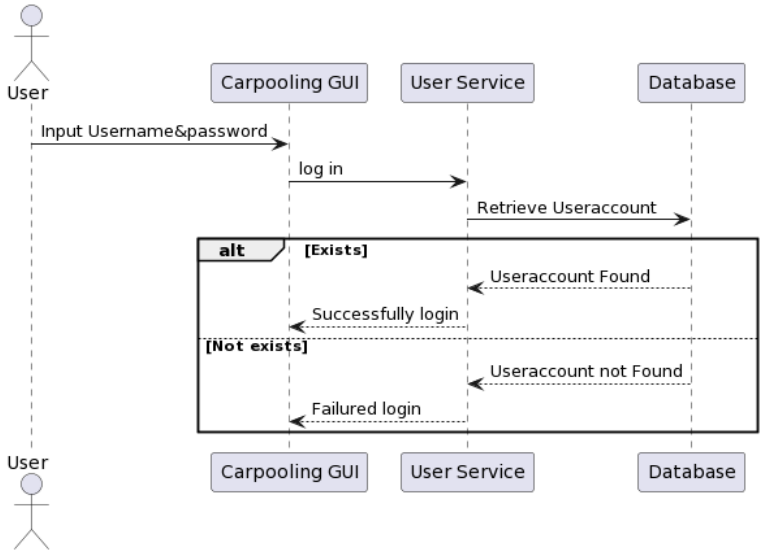

Figure 6-3 Sequence diagram of User login

6.3.2 Role Switch

Precondition: User has logged in and in the default role “passenger” .
The sequence diagram role switch has been shown below. The user enters the “Profile” button via Carpooling GUI. The Carpooling GUI interacts with the User Service by sending the request to show a profile. User Service replies to the Carpooling GUI. After it the user clicks “Switch role” via Carpooling GUI. In turn, the Carpooling GUI sends the request to the User Service to switch the role. The User Service interacts with the database by sending the request. Database replies to the User Service which replies to Carpooling GUI that the role has been switched to Driver.

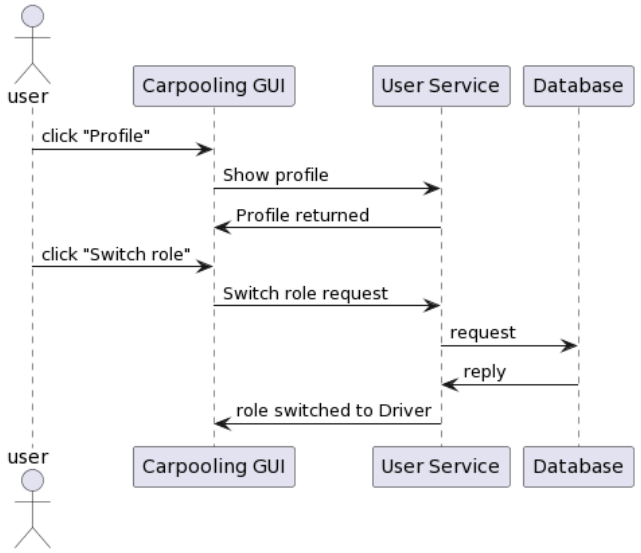

Figure 6-4 Role Switch

6.3.3 Searching Trip

Precondition: User has logged in.

The sequence diagram Searching trip has been shown below. The precondition is that the user has logged in and in a passenger role. Carpooling GUI makes a request to Trip Service. Trip Service interacts with the Map Service by requesting the current location. Map Service replies to Trip Service that the current location has been obtained. After it, Trip Service interacts with the Database by request of the route matching. There would be one the two alternatives, one is there are routes matching, and the other is there is no match route. If there are route matches, there will be three categories: matching percentage 75% above, matching percentage 50 % - 75 % and matching percentages 50% below. After the Trip Service gets replies from the Database, it continues interaction with Map Service by request of route loading. Map Service gives replies to the Carpooling GUI to show the route and estimated time. And then a request is sended by Trip Service to Payment service to request an estimated payment. Payment interacts with Carpooling GUI by giving a reply that show the estimated payment.

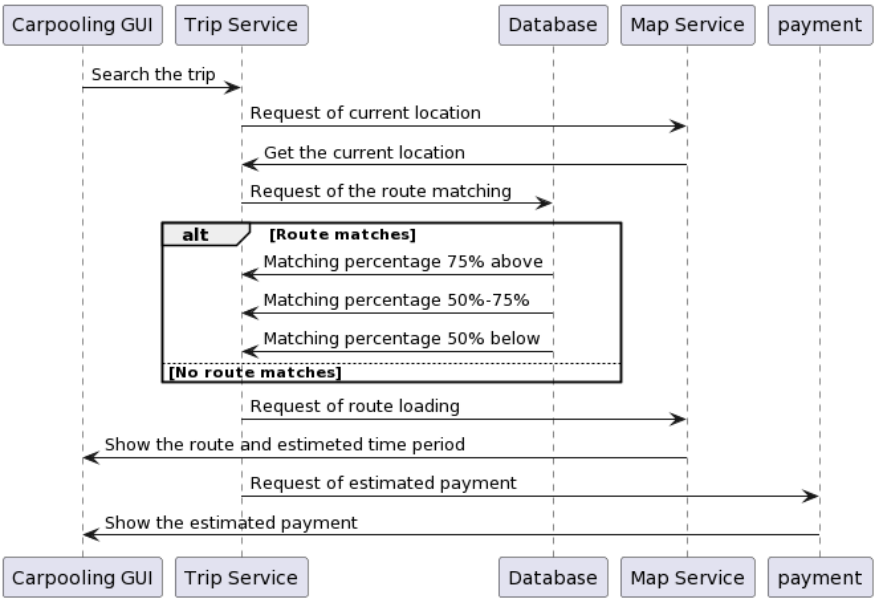

Figure 6-5 Sequence diagram of Searching Trip

6.3.4 Booking Trip

Precondition: Trip route has been successfully matched.
Below is the sequence diagram of Booking Trip, After user press book button on the GUI, a message of Trip booking will be sent to the component of Trip Service, a request will be made to database to record the carpooling trip booked, a reply will be return to Trip Service and then return Trip booked message to the GUI to inform the user that trip has been successfully booked. Trip Service also made a request and interacts with Map Service and then the real-time map will be presented on GUI.

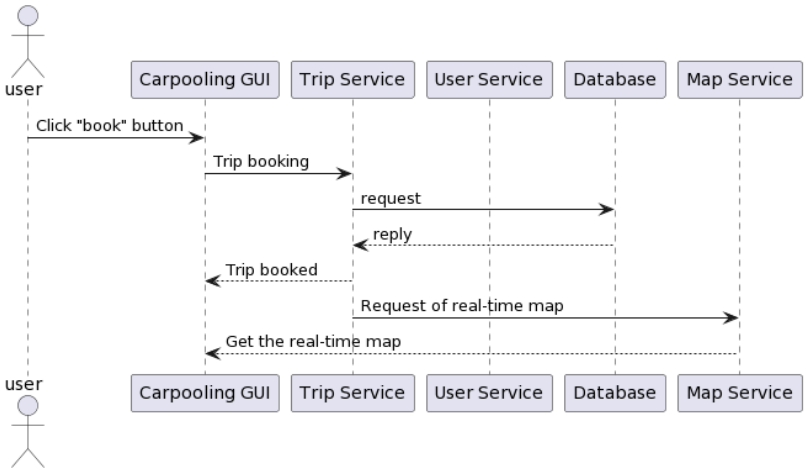

Figure 6-6. Sequence diagram of booking trip

6.3.5 Doing Payment

Precondition: Trip has finished and confirmed by passenger and driver
The sequence diagram of doing payment shows below, after the passenger entering the Pay button, a request was made from Gui to the Payment Service, Third party for example Swish and Bankid will be called by the Payment Service to require password and returned to GUI, then user enter password on the GUI, GUI made request to Third party, which made request to database to check if the password matches. If it matches, a corresponding message will be returned to the Third party, and a third sent message to Payment Service to do the payment processing. Payment service will make a request to the database to store the payment information and finally payment information will be presented on GUI. If the password does not match, the third party will directly return an error message to the GUI to remind the user.

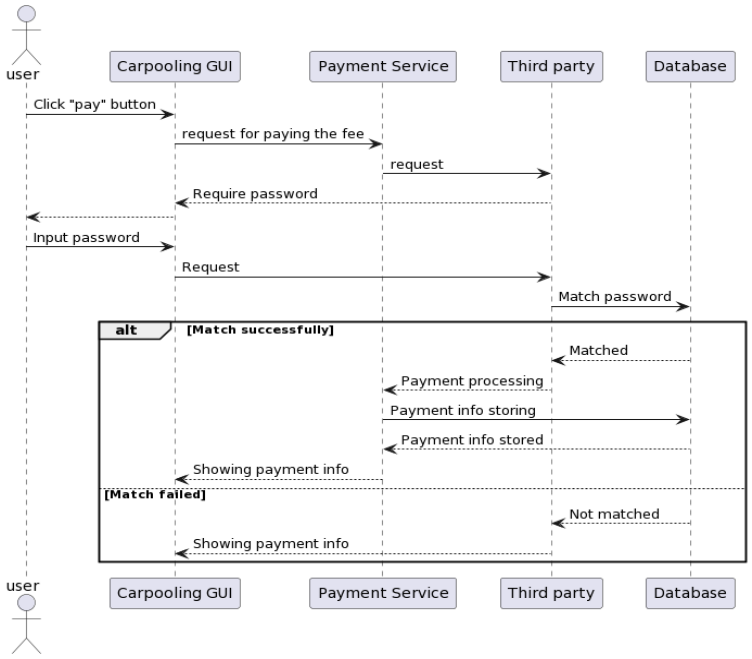

Figure 6-7. Sequence diagram of doing payment

6.3.6 Rating

Precondition: payment has been done.
The sequence diagram rating has been shown below. The user checks the rating options via Carpooling GUI. Carpooling GUI makes a request of submitting the rating to User Service. User Service interacts with the Database to store rating information. Database replies that rating information has been stored. The next interaction is going between User Service and Carpooling GUI: User Service informs Carpooling GUI that rating has been done.

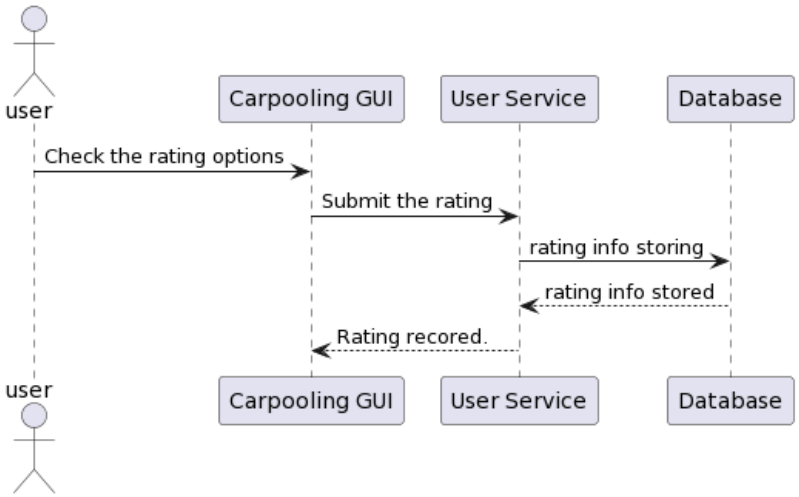

Figure 6-8. Sequence diagram of rating trip

7 Reference

[1]https://github.com/Green-Software-Foundation/sci-guide/blob/dev/use-case-submissions/msft-eShoppen.md

[2]https://medium.com/@softwarearchitectur/an-overview-of-software-architecture-patterns-55dfeaea552a