SRS Document

Project: Uber

TA: Radwa Moustafa

Team Id: 53

Team Members:

Student Name	ID	Department
نانيس عادل شحاتة حسن عبدالعال	20201700909	IS
نادين عمرو خالد الزغبي	20201700901	IS
نوران سمير أحمد حسب عوف	20201700934	IS
يوسف عبدالرؤوف أمين محمود عوض	20201701019	IS
أحمد رمضان اسماعيل عبدالعليم	20201701049	IS
كريم أسامة مرسي محمد الاباصيري	20201701186	IS
فيرينا وجيه فهمي	20201700590	IS

1. Introduction:

The Uber app is a transportation network platform that connects passengers with drivers who use their own vehicles or vehicles provided by the company. The app allows users to request a ride, track the driver's location, and pay for the ride all within the app. It also provides information about the driver, the type of car they are driving, and the estimated time of arrival. The app uses GPS technology to match riders with the nearest available driver and provides a cashless payment system, making it a convenient and popular option for transportation. Additionally, the app offers other features such as the ability to schedule a ride in advance and rate drivers/customer based on their experience.

2. User Requirements:

- The System should enable the user to book a ride to anywhere & at any time.
- A user (driver) shall be able to confirm / reject request while being on online mode.
- He (driver / rider) can cancel the trip by specifying the reason.
- User (driver / rider) can report about an issue or lost item during the trip to the help center and support will guide them in the process.
- A user (rider) shall be able to rent a car with his driving liscence.
- Hr can review the drivers applications to accept or decline them.
- He (driver / rider) shall be able to login to the application to start booking a ride.
- User (driver) can choose the driver mode that he desire.
- A user (rider / driver) shall be to register into the application.
- A user can apply for the application to be a driver.
- I (rider) can edit/update my trip like change the destination or add multiple stops .
- The system should provide different payment method in the application like cash , wallets , credit cards.
- User (rider) also can contact with driver or the support.

- A user (rider / driver) shall be to view the history of rides.
- The user feels free to give a rating about driver or the trip or vise versa .
- guarantee user safety like adding real-time tracking .
- the user should be able to see his history of rides .

3. Functional Requirements:

- **Manage request:**

  - Description / Action:

    -Driver shall be able to confirm or reject a ride after receiving a request

  - Requirements / Inputs:

    -Trip information which includes pick-up, drop-off location, cost, estimated time and type of trip which can be a reserved trip or instant trip.

  - Source: System.

  - Pre-Condition:

    -Driver must have chosen online mode which enable him to receive requests.

  - Post-Condition:

    - if driver confirmed request, system should increment driver's number of rides.

    - if driver rejected request, match will be executed again.

  - Output:

    - If driver confirmed the request, the ride should start, pickup location, rider info and estimated time should appear and driver should head to it.

    -Otherwise, system displays other requests

- **Cancel ride:**

- Description / Action:

-The driver/rider shall be able to cancel the ride

- Requirements / Inputs:

- trip information
- the reason why driver/rider wants to cancel the ride

- Source: system and driver/rider

- Pre-Condition:

- Rider must have Booked a ride, be matched with a driver and the ride started.
- Checking the number of cancels the driver/rider made through the day.
- If it exceeds the maximum allowed number of cancels per day, the driver/rider should be informed that if he/she cancels the drive, he/she will pay a fine.

- Post-Condition:

- System should increment driver/rider 's number of cancels

- Output:

- message that confirms the cancellation

- **Help:**

- Description / Action:

- Driver/Rider shall be able to report for a situation or for lost/found personal belongings.
-support can review reports and take the appropriate decision.

- Requirements / Inputs:

    - rider/driver information

    - type of report which can be issue with account, help with trip and guide.

- Source: driver/rider and system

- Pre-Condition:

- Post-Condition:

    - system should notify customer support that there is a new report

- Output:

    - a message states that report state is pending till support contacts the reporter.

- **Match driver:**

- Description / Action:

    - rider is matched with a driver that is within the area.

- Requirements / Inputs:

    - pick-up location, driver location, driver information

- Source: system

- Pre-Condition:

    - rider must have booked a ride.
    - driver must have chosen online mode.

- Post-Condition:

- Output:

    - rider and driver will be matched and each one will view other's information

- **Generate path:**

- Description / Action:

    - this functionality generates path according to pick-up location and drop-off location of the ride.

- Requirements / Inputs:

    - pick-up location, drop-off location

- Source: system

- Pre-Condition:

    - rider must have booked a ride.

- Post-Condition:

- Output:

    - path is generated and is displayed for rider and driver

- **Rent car:**

- Description / Action:

    - Rider shall be able to rent a car for himself to use it and return it back at the agreed time.

- Requirements / Inputs:

    - renting duration, rider information and driving license

- Source:

  - rider

- Pre-Condition:

  - Rider should have enough money in his wallet/visa in order to make this functionality start.

  - Rider can use this functionality once his rating stars are greater than 3.

- Post-Condition:

  - system should track the car & make sure the rider returned the car at the agreed time.

  - if the rider is late, rider will be charged a fine that increases every hour that passes.

- Output:

  - confirmation message

- **Track live location:**

- Description / Action:

  -monitor the current whereabouts of the car during the trip .

-Requirements / Inputs:

  - the location of car.

- Source: system

- Pre-Condition:

- the trip must start first .

- Post-Condition:

- the positions of the car will be shown/drawn on the map .

- Output:

-the car with the route to destination will be shown on the map and could be shared to other platforms .

- **Evaluate job application:**

- Description / Action:

- Hr shall be able to review applicant's application, evaluate it and take the appropriate decision.

- Requirements / Inputs:

- job application

- Source: system

- Pre-Condition:

- there must be at least one application

- Post-Condition:

- if driver is accepted, he should be able to use the application as a driver & his data should be added to database.

- if there is missing data in the application, email is sent to the applicant

- status of application is changed from pending into finished

- Output:

- an email is sent to the desired person states if he is accepted or not, depend on Hr 's evaluation.

- **Log in:**

- Description / Action:

  - rider/driver/support shall be able to log in with his valid email and passcode

- Requirements / Inputs:

  - driver/rider/support 's phone number or email, password

- Source: driver/rider

- Pre-Condition:

  - driver/rider must have an account first

- Post-Condition:

  - rider/driver/support 's data should be retrieved from database.

- Output:

  - if data is valid, driver/rider will be logged in successfully. Otherwise, message states that there is something that is incorrect.

- **Change mode:**

- Description / Action:

  - driver shall be able to choose between online and offline modes. In online mode, driver can receive requests. Otherwise, driver cannot receive requests

- Requirements / Inputs:

  - state of driver, driver info

- Source: driver, system

- Pre-Condition:

  - driver can't change his mode to offline while he is in ongoing ride.

- Post-Condition:

  - In online mode, receiving requests will be enabled.
  - In offline mode, receiving requests will be disabled.

- Output:

  - message states that mode is switched

- **Register:**

- action:

  - create an account for the user

- inputs:

  - email, password, phone number and username.

- source: rider

- pre-condition:

- the user must be a rider.

- post-condition:

- the user data will successfully be added to our database .

- output:

- rider features will be enabled to the user.

- **Fill job application:**

- action:

- enable the user to use the app as a driver .

- inputs:

- email, username, phone number, age, password and more info about the cat and its owner like driver's license status, driving experience ,driving history ,criminal record, provide proof of residency, along with proof of auto insurance , car registration and car insurance coverage.

- source: driver.

- pre-condition:

-the driver data should be valid like the car license

- post-condition:

-the driver application will be added to the system with status "pending" until evaluation ends .

- output:

- the driver will receive a message to endicate a sucessfull application, awaiting screen will appear until the status changes to "accepted".

- **Book a ride:**

- action:

-enable the customer to book a trip defining a source and destination points considering adding multiple stops.

- calculate cost and estimated time.

- inputs:

- pick-up and drop-off points from the map.

- source: rider

- pre-condition:

-the user must have an account and logged in successfully.

- post-condition:

- the app will look for the nearest cars around the customer area.

- output:

-the trip will start once there is a match and path is shown on the map.

- **update trip:**

- action:

  - enabling the user to access and change in his trip like changing the destination , add multiple stops and path and change payment method in the current trip.

- inputs:

  - new drop-off location, other payment method , choose multiple locations for stops.

- source: rider

- pre-condition:

  - the trip must start first.

-post-condition:

  - the trip will get altered (the trip attributes/ specification will get altered to the user likes).

- output:

  - the trip will get altered to the user likes.

- **Manage wallet:**

- action:

  - enabling multiple methods for payment like cash , wallets , credit cards , payment services for the user to choose for an example the driver/rider should add/ remove card , the support should access the payment method during the trip in case of emergency situations .

- inputs:

    - choosing from these methods and extra info for credit card information.

- source: driver/rider/customer support

- pre-condition:

    - the user must have an account and logged in successfully .

- post-condition:

    - The new payment method is successfully added to the user's account and is available for use when making a ride booking.

- output:

    - The user can now select the newly added payment method as their preferred option when booking a ride. The payment will be processed using the account information provided during the setup process.

- **Contact:**

- action :

    - enable the user to communicate with the driver after the match through chat or calls .

- inputs :

    - phone number of both from the database , messages from both .

- source: driver/rider, database

- pre-condition:

-the match must happen first, good internet connection.

- post-condition:

-A chat starts between both sides and have the access to each other phone number .

- output:

-a chat is opened if they choose to communicate through chat .

- **Access history:**

- action :

- show the history of rides in detail for the user .

- inputs :

- price , pick-up , drop-off points , driver name .

- source: database

- pre-condition :

- the trip must be completed successfully .

- post-condition:

- output :

- the user will be able to see all previous trips in detail .

- **Rate:**

- action :

- enable the user to give his feedback and rate the driver / customer .

- inputs :

-numerical rating or score, choosing from a scale of 1 to 5 . -

- source: driver/ rider

- pre-condition :

-the trip must end first

- post-condition:

- the new rate of the user will be calculated .

- output :

-update the rating of the user .

- **Show profile information:**

- action :

- enable the user to show the profile info of others .

- inputs :

- id or name of the user .

- source: database .

- pre-condition :

- both the user viewing and the user being viewed must have an account on the platform .

- post-condition:

- more data will get fetched from the user table in the database .

- output :

- the user profile will be showed to the viewer .

- **Select multiple destinations:**

- action :

- enable the user to add multiple stops in his trip .

- inputs :

- multiple locations from the map.

- source: rider

- pre-condition:

- the user must have an account and logged in successfully and in the process of booking a trip .

- post-condition:

- the trip specifications will get updated and the driver will get notified with the changes.

- output :

- multiple stops will be shown in the trip map .

4. **Non-functional Requirements:**
   1. Safety: The system should prevent harm to people or damage to the environment.

Example: The rider can share his location and his itinerary with his friends to know the time of his arrival at the place he wants to reach and his current location.

Driver must have valid car license

2. Security: The software's safety should be protected against espionage or sabotage. Unauthorized access to the data is not permissible. The data must be backed up daily and stored in a secured location, at a distance from different facilities of the system.

Example: *No one can see user's profile information unless there is a ride booked with restricted knowledge or it's the user itself.
*The payment processing gateway must ask questions that only the user knows the answer to. This can help verify a user's identity .
*Encrypting all sensitive data like user details, payment information, and trip history to ensure it is protected from potential hackers.
*After a certain number of login attempts, a security system may lock an account to protect a user's information from potential hackers. To unlock their account, a user can typically call the company to verify their identity and set a new password accept only secure passwords that have sufficient length and non-alphabetic characters

3. Maintainability: The system should be restored or repaired to a specified condition within a specified period or time when maintenance is performed.

The system must be easy-to-maintain right from its launch.

Example : The mean time to restore the system following a system failure must not be greater than 10 minutes.

If there is a bug, user can report and will be maintained as soon as possible.

In case of a hardware failure or database corruption, a replacement page will be shown. Also in case of a hardware failure or database corruption, backups of the database should be retrieved from the server and saved by the administrator

4. Availability: The system is highly accessible to a user at any given point in time.

example: The system may be available 98 percent of the time during a month.

5. Environmental(I/O DEVICE): The system can be ran on any other operating systems and on all kinds of devices. The interfaces must automatically adjust to devices with different screen sizes,
and allow to change interface size and color scheme to improve readability

Example : The system will be running on IOS , Android and Windows in any version.

6.Localization : Application has features that match the geographical location of its users, including aspects such as:
Languages, Currencies, Measurements, such as pounds vs. kilograms and Time zones.

Example : The system will provide multiple currencies and different languages such as Arabic and English.

7. Performance: Using a high-speed and reliable server to ensure minimal downtime and fast response times.

Example : multi-client system that must provide same response time targets for each of the clients and users. The system shall be able to run a target number of transactions without failure.
The application’s homepage should load in less than 4 seconds
When a user goes to Navigation screen and enters the destination, the route should be calculated within 4 seconds.

8. Usability: The system must provide different graphical interfaces for users.
Providing a clear and intuitive interface that allows passengers to easily book a ride and drivers to easily accept them.

All system interfaces must be user-friendly and simple to learn, including helping hints and messages and intuitive workflow to enable users to perform tasks efficiently while enjoying the experience.

example :- the client must be able to fast learn and use the interface without prior knowledge of system terminologies or rules.

- The error rate of users submitting their payment details at the checkout page mustn't exceed 10 percent.

9. Accessibility : Make the software available to people with the broadest range of characteristics and capabilities, including users with deafness, blindness, colorblindness, and more.

Example : The system will provide many features to help users such as voice assistants ,screen reader and apply high-contrast and color schemes that are easier to read for users with color blindness

10. scalability :System should be able to handle a growing amount of work, data and transactions and define how the application can grow and increase its features and functionality without impacting the performance

example : The system must be scalable enough to support changed transactions like change drop off or cancel ride without causing any failure. The limit of maximum attendance of the website must be scalable enough to support 300,000 visits at the same time while maintaining optimal performance without a negative impact on the website load speed.

11. Consistency: After an operation executes, the data should be consistent across all the nodes, and thus all clients see the same data at the same time, no matter which node they connect to .
Ensuring consistent user experience across different platforms and devices like smartphones, tablets, and laptops same UI design and function implementation over time, across different contexts or situations. System should behave the same way every time it is used, regardless of the user, the environment, or other factors.

example: Consistent Design: Ensure that the design of the system is consistent across all pages and sections, including color schemes, typography, and layout.
Consistent Performance: Ensure that the booking system consistently performs at a high level, with minimal downtime and fast page load times.

12. Programming language: C# and SQL

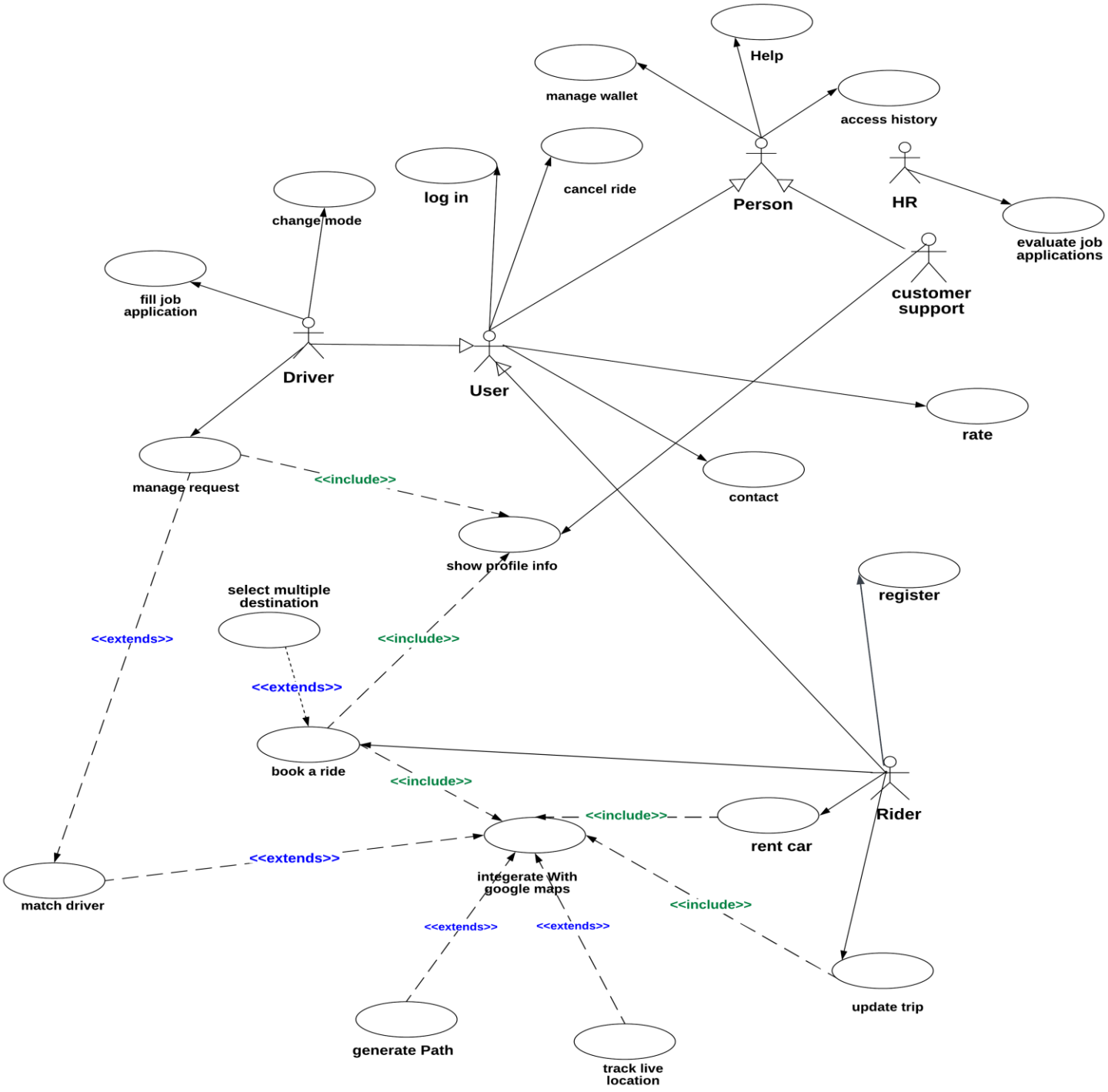
13. Reliability: The system should perform its intended function or task consistently and accurately, without any unexpected failures or errors

Example: accurately display available cars based on matching area , accurately generate path after select source and destination , accurately handle bookings such as payment transaction and live location cancellations

14. Authorization: Giving someone the ability to access a resource.
- Mechanism to determine access levels or user privileges related to system resources.

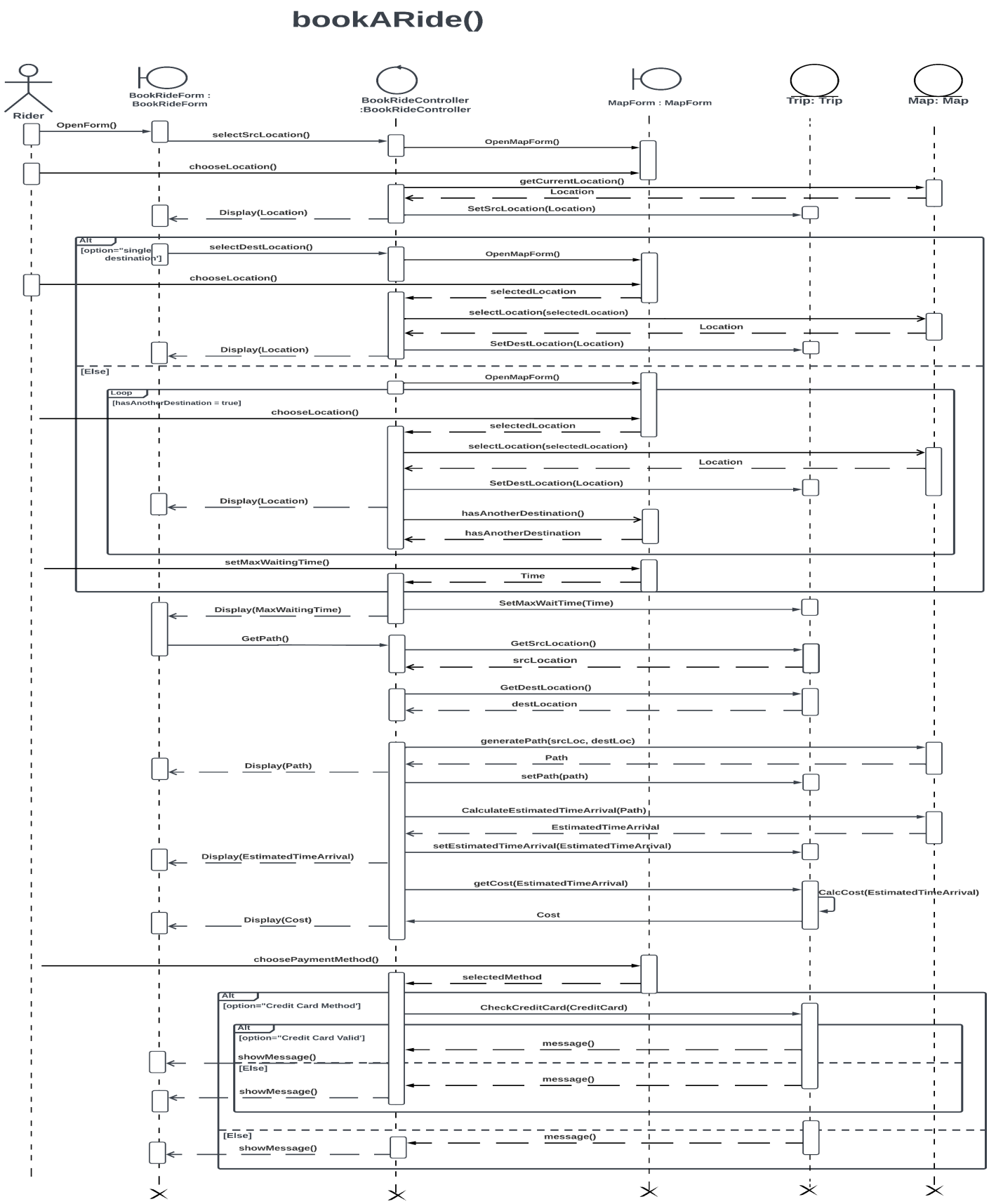
Example: Users can securely sign in with their Uber username and password to access their information and their allowed function whether he was rider/driver/support/HR.

5. Use case:



6. Sequence diagram:

- **Use case: Book a ride**


bookARide()
Rider
BookRideForm : BookRideForm
BookRideController :BookRideController
MapForm : MapForm
Trip: Trip
Map: Map
OpenForm()
selectSrcLocation()
OpenMapForm()
chooseLocation()
getCurrentLocation()
Location
Display(Location)
SetSrcLocation(Location)
Alt
[option="single destination']
selectDestLocation()
OpenMapForm()
chooseLocation()
selectedLocation
selectLocation(selectedLocation)
Location
Display(Location)
SetDestLocation(Location)
[Else]
OpenMapForm()
Loop
[hasAnotherDestination = true]
chooseLocation()
selectedLocation
selectLocation(selectedLocation)
Location
SetDestLocation(Location)
Display(Location)
hasAnotherDestination()
hasAnotherDestination
setMaxWaitingTime()
Time
Display(MaxWaitingTime)
SetMaxWaitTime(Time)
GetPath()
GetSrcLocation()
srcLocation
GetDestLocation()
destLocation
generatePath(srcLoc, destLoc)
Path
Display(Path)
setPath(path)
CalculateEstimatedTimeArrival(Path)
EstimatedTimeArrival
setEstimatedTimeArrival(EstimatedTimeArrival)
Display(EstimatedTimeArrival)
getCost(EstimatedTimeArrival)
CalcCost(EstimatedTimeArrival)
Cost
Display(Cost)
choosePaymentMethod()
selectedMethod
Alt
[option="Credit Card Method']
CheckCreditCard(CreditCard)
Alt
[option="Credit Card Valid']
message()
showMessage()
[Else]
message()
showMessage()
[Else]
message()
showMessage()