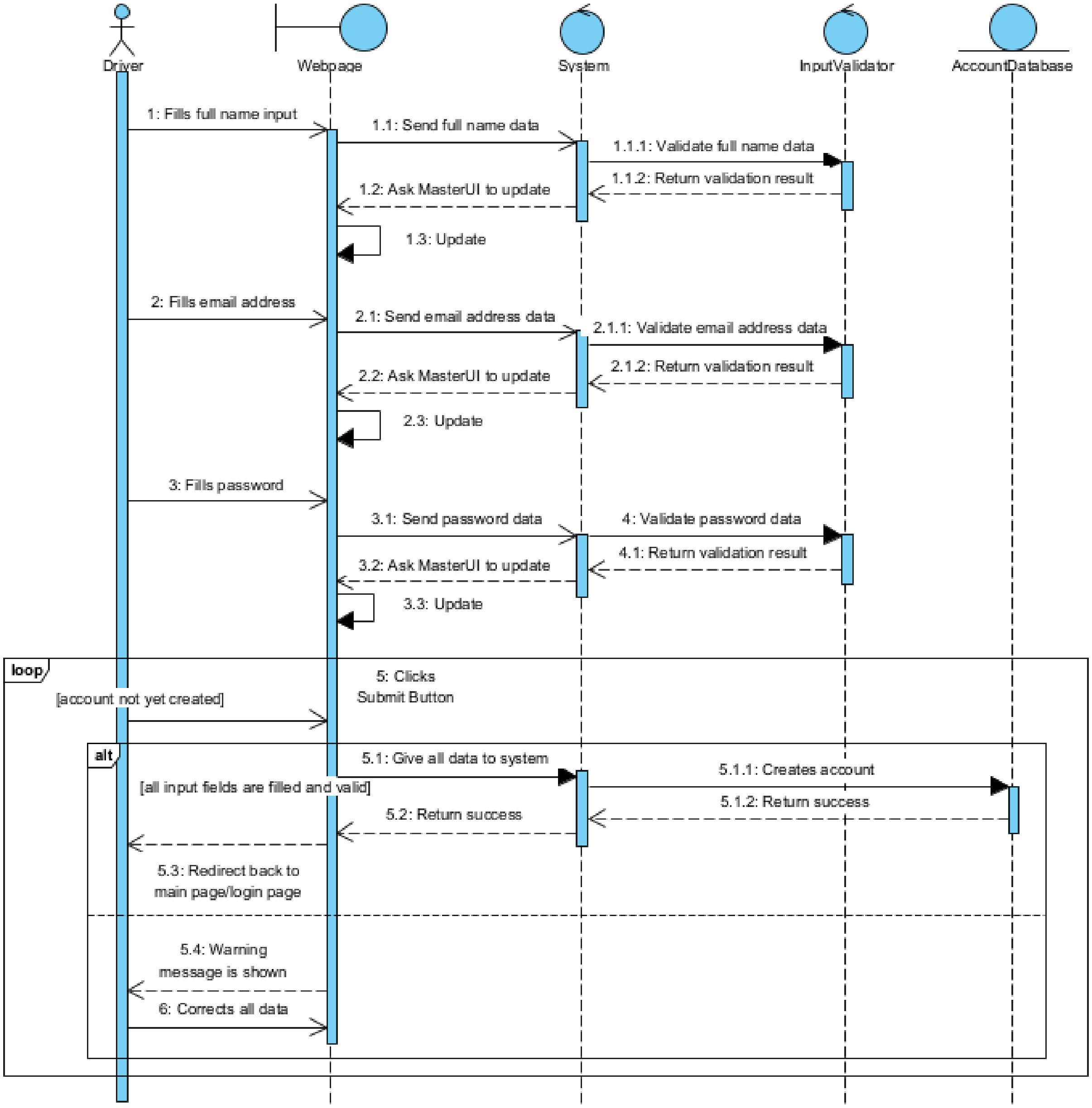
1. Account Creation

Use Case ID:	1		
Use Case Name:	Account creation		
Created By:	Min Khant	Last Updated By:	Ruxing
Date Created:	16/02/2023	Date Last Updated:	03/03/2023

Actor:	Driver
Description:	The driver can create an account using a form to access features which are available only to users with accounts.
Preconditions:	- The driver can access the account creation page. - The driver is already on the account creation page.
Postconditions:	- An additional account with correct information is created in the user base - The driver can login to the page, if he gives valid credentials
Priority:	Normal. Drivers can still use the website without any information. But, features requiring accounts will not be usable.
Frequency of Use:	0.8 times per user. Assuming that 80% of users will want to register an account.
Flow of Events:	1. User fills the inputs in the form in any order, as follows: a. Email b. Full name c. Password d. Re-enter password. 2. User clicks on the submit button. 3. All the inputs are filled in and are valid. The system creates a valid user account in the entity. 4. The User is directed back to the main page, with a flash message informing the user the account is created.

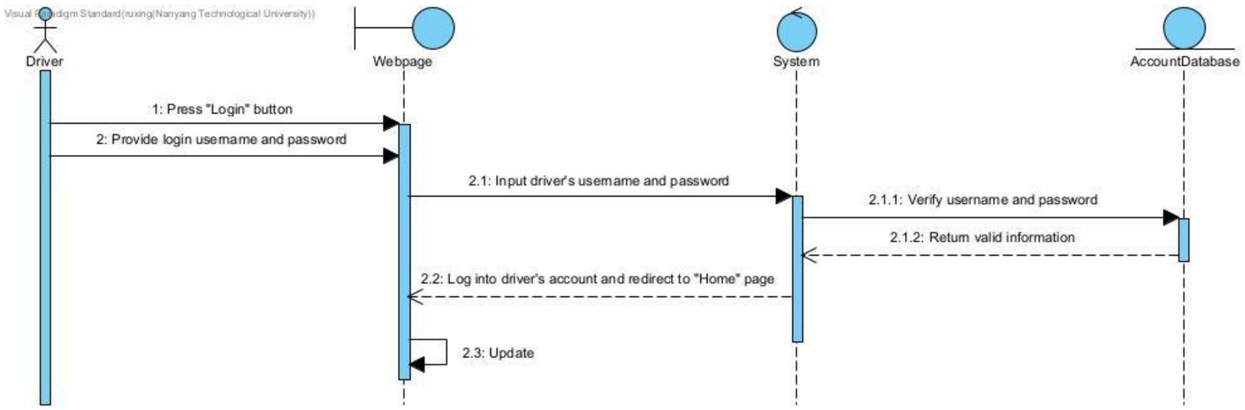
Alternative Flows:	If an input in invalid, there will be an error message, and the driver will have to re-enter details.
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	For the form, we want *continuous feedback* for the user and for the user to not refill everything input if they submit the form and got one input wrong. This is a minor detail but important for UX.


Driver
Webpage
System
InputValidator
AccountDatabase
1: Fills full name input
1.1: Send full name data
1.1.1: Validate full name data
1.1.2: Return validation result
1.2: Ask MasterUI to update
1.3: Update
2: Fills email address
2.1: Send email address data
2.1.1: Validate email address data
2.1.2: Return validation result
2.2: Ask MasterUI to update
2.3: Update
3: Fills password
3.1: Send password data
4: Validate password data
4.1: Return validation result
3.2: Ask MasterUI to update
3.3: Update
loop
[account not yet created]
5: Clicks
Submit Button
alt
5.1: Give all data to system
[all input fields are filled and valid]
5.1.1: Creates account
5.1.2: Return success
5.2: Return success
5.3: Redirect back to
main page/login page
5.4: Warning
message is shown
6: Corrects all data

2. Driver Login

Use Case ID:	2		
Use Case Name:	Driver Login		
Created By:	Tan Jun Xiong	Last Updated By:	Ruxing
Date Created:	11/02/2023	Date Last Updated:	03/03/2023

Actor:	Driver
Description:	Driver will login to his account, after which he can save destinations and indicate interest in going to car parks.
Preconditions:	1. Driver has created an account. 2. Driver is looking at the default page of the website, which is the login page.
Postconditions:	Driver is logged in and redirected to the main page
Priority:	-
Frequency of Use:	Once a day
Flow of Events:	1. Driver enters email and password into the provided text input boxes. 2. Driver presses on the “Login” button. 3. The website queries the account Database with the input information. 4. The website obtains the driver’s account and stores the information as a dynamic memory of the user’s current session. 5. The website redirects the driver to the home page.
Alternative Flows:	-
Exceptions:	If the website does not obtain any account from the database at Step 4, there will be an error flash message and driver is redirected back to Login Page (i.e. Step 1)
Includes:	-
Special Requirements:	6. Entire Use Case completed by Driver within 1 minute.
Assumptions:	-
Notes and Issues:	-



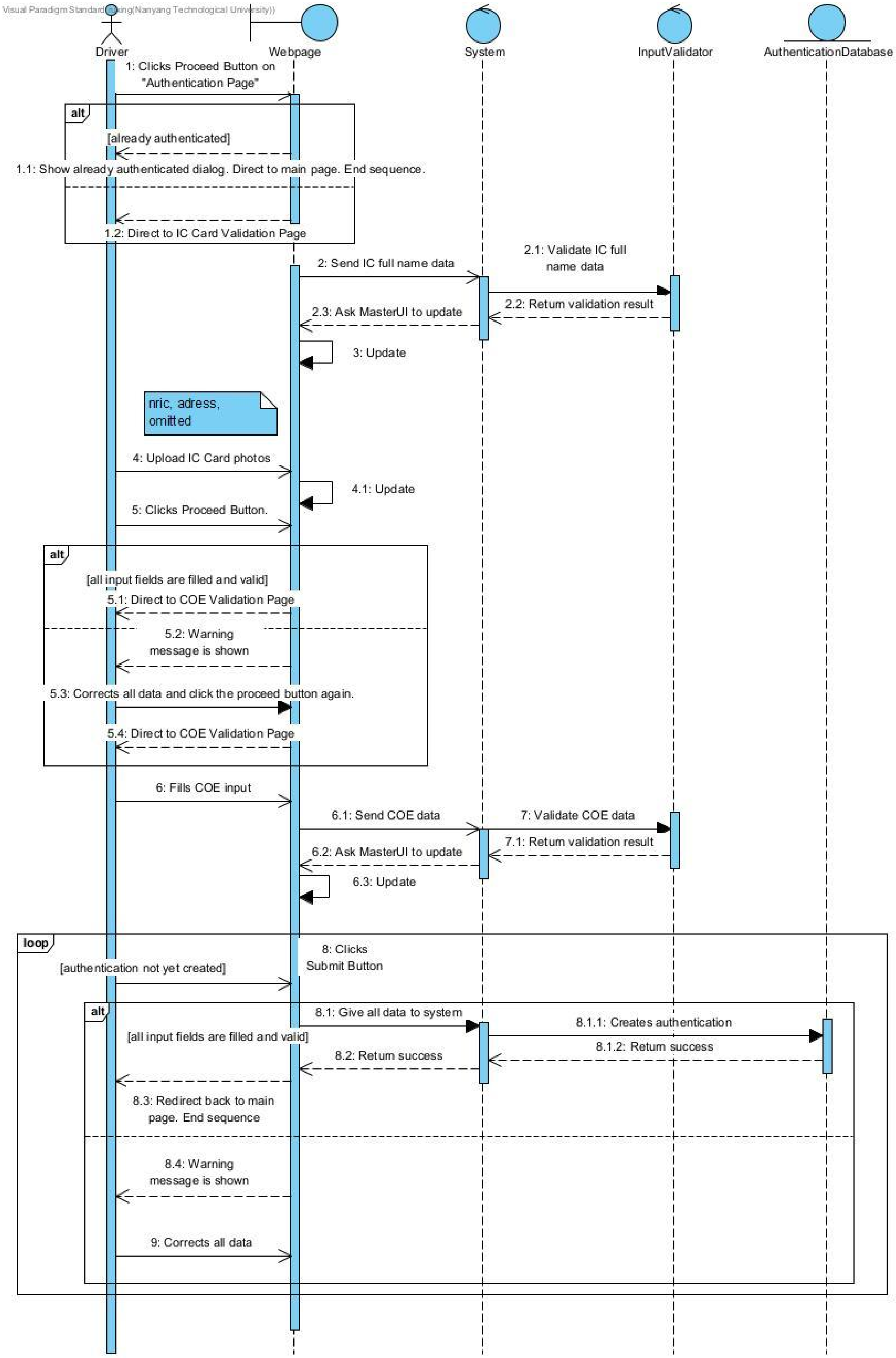
Driver
Webpage
System
AccountDatabase
1: Press "Login" button
2: Provide login username and password
2.1: Input driver's username and password
2.1.1: Verify username and password
2.1.2: Return valid information
2.2: Log into driver's account and redirect to "Home" page
2.3: Update

3. COE Registration

Use Case ID:	3		
Use Case Name:	COE Registration		
Created By:	Min Khant	Last Updated By:	Ruxing
Date Created:	16/02/2023	Date Last Updated:	03/03/2023

Actor:	Driver
Description:	Users with accounts are entitled to exclusive rewards and discounts. We want to ensure that users won’t abuse the system. To do this, we must check that they own a car. We can do this by checking their ID and COE.
Preconditions:	- Driver has an account - Driver is logged on to the account. - Driver is on the COE registration page
Postconditions:	- A relationship is made between driver and vehicle in the database - Driver is able to access rewards and discounts

Priority:	Normal. Should be done after account creation.
Frequency of Use:	0.64 times per user. Assuming 80% of registered users want rewards and discounts.
Flow of Events:	1. User is to upload a photo of images of their physical IC Card (front and back) 2. User inputs the fields and uploads the images and clicks the Submit button. 3. User inputs COE details. 4. User inputs the fields and uploads the images and clicks the Submit button. 5. All fields are filled and valid. The system validates the input and flashes a success message, redirects user to the home page. *Similar to account creation, we want continuous feedback when an input is filled. This will not be described here.*
Alternative Flows:	AF-1: 1. User already authenticated his account. The user is directed back to the main page and shown an “Account already authenticated.” dialog box. AF-4 and 6: 1. At least one of the inputs is either not filled in or is not valid. An alert box pops up in the browser, containing names of invalid inputs. 2. User goes to each invalid input and corrects the data. 3. User clicks the submit button again.
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	- Users are all acting in good faith - they won’t fake their ID Card and COE. - Users can take good quality images of the front and back of their IC Card.
Notes and Issues:	-

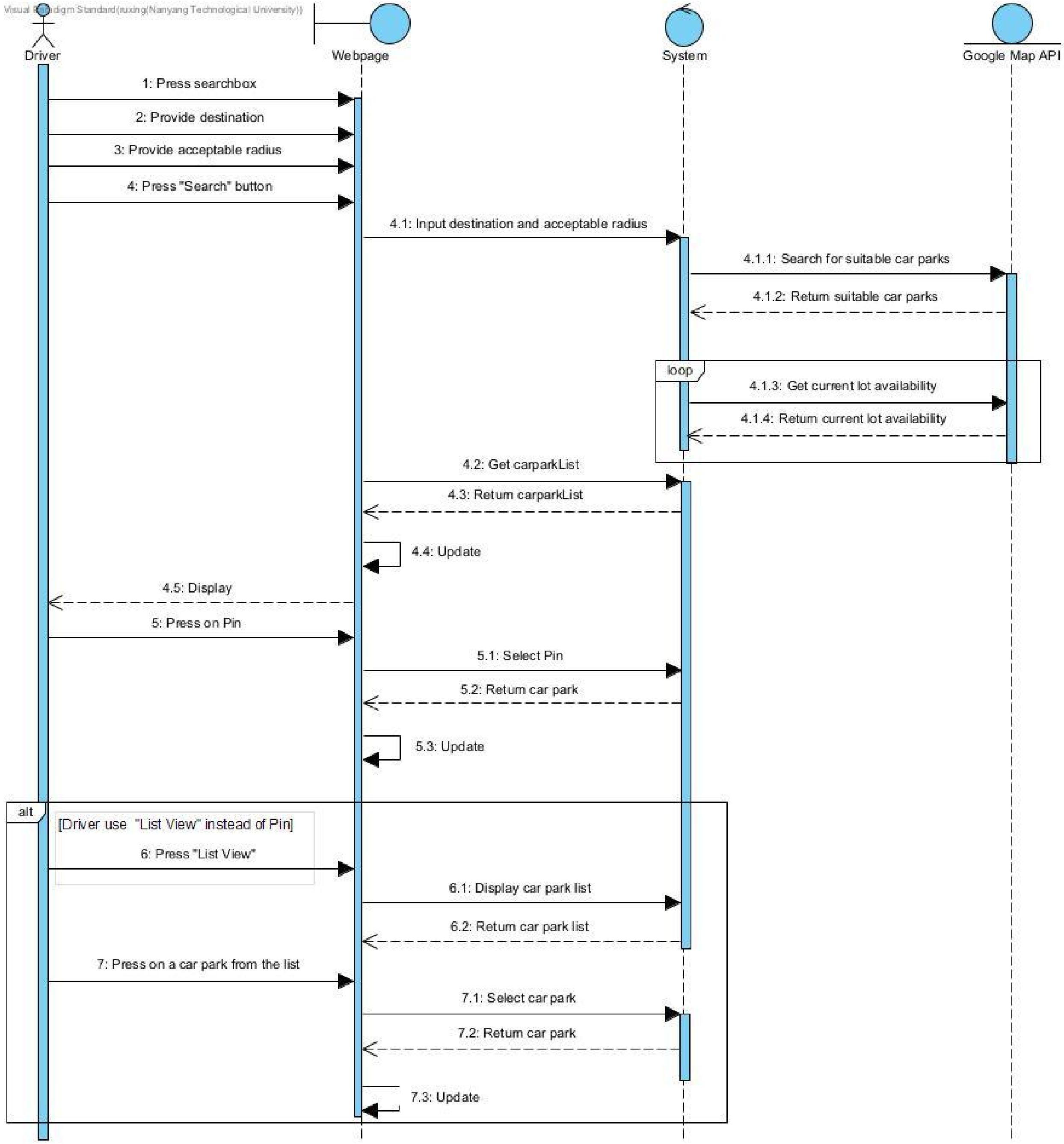

Visual Paradigm Standard(...king(Nanyang Technological University))
Driver
Webpage
System
InputValidator
AuthenticationDatabase
1: Clicks Proceed Button on "Authentication Page"
alt
[already authenticated]
1.1: Show already authenticated dialog. Direct to main page. End sequence.
1.2: Direct to IC Card Validation Page
2: Send IC full name data
2.1: Validate IC full name data
2.2: Return validation result
2.3: Ask MasterUI to update
3: Update
nric, adress, omitted
4: Upload IC Card photos
4.1: Update
5: Clicks Proceed Button.
alt
[all input fields are filled and valid]
5.1: Direct to COE Validation Page
5.2: Warning message is shown
5.3: Corrects all data and click the proceed button again.
5.4: Direct to COE Validation Page
6: Fills COE input
6.1: Send COE data
7: Validate COE data
7.1: Return validation result
6.2: Ask MasterUI to update
6.3: Update
loop
[authentication not yet created]
8: Clicks Submit Button
alt
8.1: Give all data to system
[all input fields are filled and valid]
8.1.1: Creates authentication
8.1.2: Return success
8.2: Return success
8.3: Redirect back to main page. End sequence
8.4: Warning message is shown
9: Corrects all data

4. Querying Car Parks

Use Case ID:	4		
Use Case Name:	Finding a car park		
Created By:	Tan Jun Xiong	Last Updated By:	Ruxing
Date Created:	11/02/2023	Date Last Updated:	03/03/2023

Actor:	Driver
Description:	Driver obtains information of car parks near his destination.
Preconditions:	1. Driver is logged in (Optional) 2. Driver is looking at the Home page.
Postconditions:	1. Relevant carpark information is provided to the driver.
Priority:	High. This is a main feature of the web application.
Frequency of Use:	2-3 times a day
Flow of Events:	1. Driver enters the destination postal code and acceptable radius into the provided text input boxes. 2. Driver presses on the “Search” button. 3. The website queries Dataset for a list of nearby car parks. 4. The website queries API on the current lot availability of the car parks in the list. 5. The website updates the Map Display with Pins representing car parks at the respective location of the map. 6. Driver selects a Pin to view details of the car park. 7. The website displays information of the car park. 8. (Optional) Driver selects another Pin, repeating Steps 6-7.
Alternative Flows:	4.1.AC.1. Driver chooses to view the car parks in a list format. After Step 5, the driver selects the “List View” button instead. Website displays a list of the car parks. Driver then continues with Step 6 by selecting Entries in the list instead of Pins on the Map.
Exceptions:	4.1.EX.1. Either destination could not be found, or no car parks could be found nearby (Step 3). Website displays “Destination could not be found or no car parks available in your choice of radius”. There will thus be no Pins on the Map or

	Entries in the car park list. Driver returns to Step 1.
Includes:	-
Special Requirements:	1. Entire Use Case completed by Driver within 2-4 minutes.
Assumptions:	-
Notes and Issues:	-

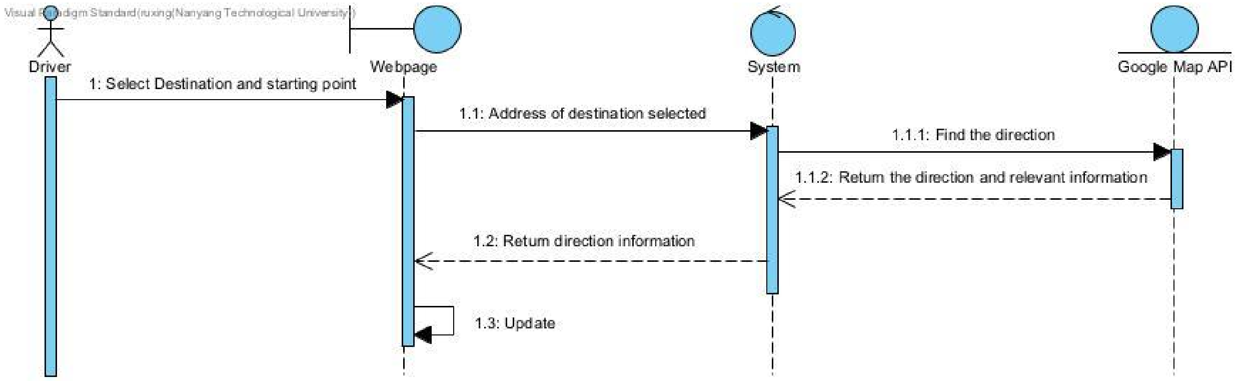


Visual Paradigm Standard(ruxing(Nanyang Technological University))
Driver
Webpage
System
Google Map API
1: Press searchbox
2: Provide destination
3: Provide acceptable radius
4: Press "Search" button
4.1: Input destination and acceptable radius
4.1.1: Search for suitable car parks
4.1.2: Return suitable car parks
loop
4.1.3: Get current lot availability
4.1.4: Return current lot availability
4.2: Get carparkList
4.3: Return carparkList
4.4: Update
4.5: Display
5: Press on Pin
5.1: Select Pin
5.2: Return car park
5.3: Update
alt
[Driver use "List View" instead of Pin]
6: Press "List View"
6.1: Display car park list
6.2: Return car park list
7: Press on a car park from the list
7.1: Select car park
7.2: Return car park
7.3: Update

5. Display Directions

Use Case ID:	5		
Use Case Name:	Display directions		
Created By:	Qiu Zhen	Last Updated By:	Ruxing
Date Created:	18/02/2023	Date Last Updated:	03/03/2023

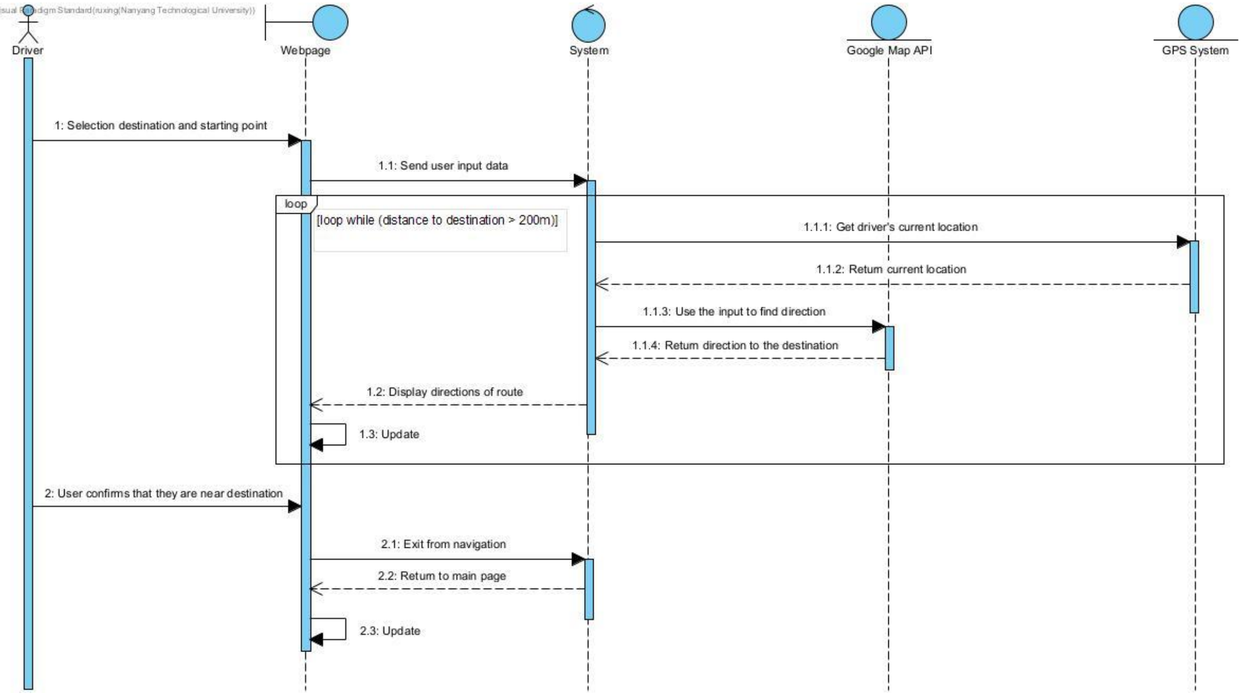
Actor:	Driver, Mapbox API
Description:	The website can display directions to the destination on the map for the drivers to view. The driver can view the estimated time, road condition and the directions to the destination so that the driver can make a more informed decision of which car park to go to.
Preconditions:	1. The website is displaying the car park pins on the map. 2. The user has selected one of the car parks as a destination.
Postconditions:	Direction is displayed on the website
Priority:	High. This is the main feature of the web application.
Frequency of Use:	5-6 times per day
Flow of Events:	1. The user selects the destination 2. The input address (or postal code) is used to calculate direction using Google API 3. The direction information is displayed on the website
Alternative Flows:	-
Exceptions:	5.EX.1: The website fails to load the direction The website displays an error message and then is prompted to return to the main page
Includes:	-
Special Requirements:	Mapbox API
Assumptions:	Login is not compulsory for users to view the directions on the website
Notes and Issues:	


Driver
Webpage
System
Google Map API
1: Select Destination and starting point
1.1: Address of destination selected
1.1.1: Find the direction
1.1.2: Return the direction and relevant information
1.2: Return direction information
1.3: Update

6. Navigation

Use Case ID:	6		
Use Case Name:	Navigation		
Created By:	Qiu Zhen	Last Updated By:	Ruxing
Date Created:	18/02/2023	Date Last Updated:	03/03/2023

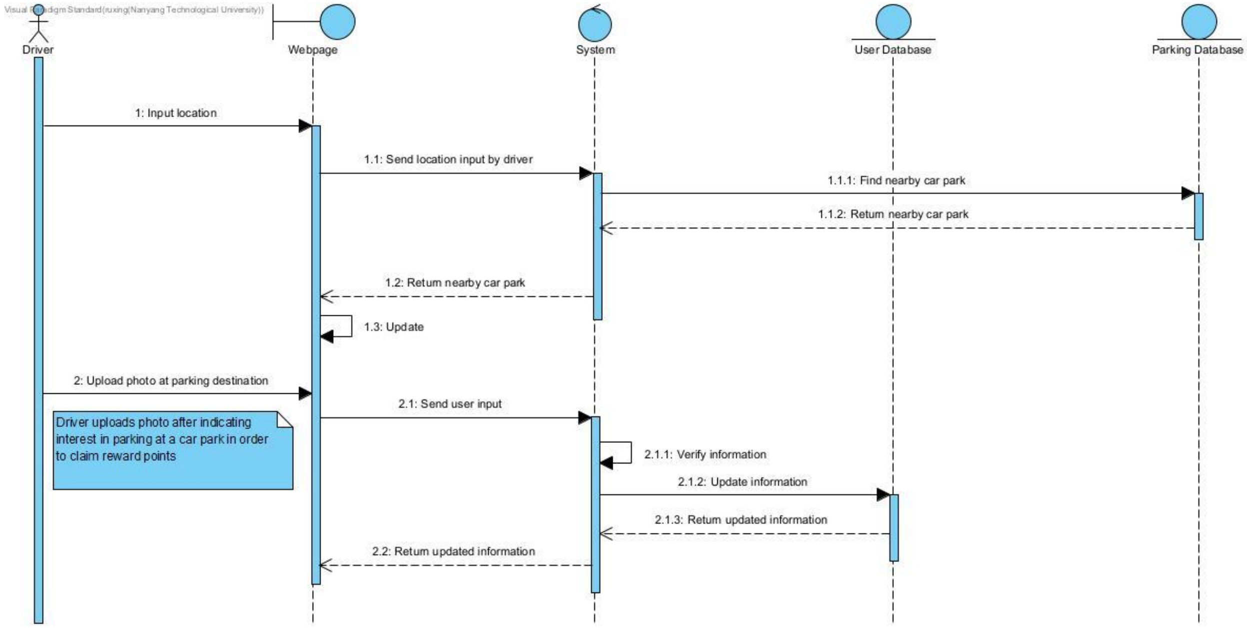
Actor:	Driver, Mapbox API
Description:	The website can help the drivers to navigate to the destination. The driver can use the website to navigate to the destination they choose.
Preconditions:	1. The driver has selected a destination. 2. Mapbox API is used 3. The current location of the driver can be accessed
Postconditions:	The driver is near the destination
Priority:	-
Frequency of Use:	1-2 times a day
Flow of Events:	1. The driver input the destination. 2. The destination and the starting location is used to calculate a direction through Google Map 3. The website uses Google Map to provide navigation 4. The navigation ends when the driver is near the destination
Alternative Flows:	-
Exceptions:	
Includes:	Show Directions
Special Requirements:	The driver must allow access of current location
Assumptions:	The user can use this function without login
Notes and Issues:	The navigation service is dependent on Google Map API or open Google Map Website to complete navigation


Driver
Webpage
System
Google Map API
GPS System
1: Selection destination and starting point
1.1: Send user input data
loop
[loop while (distance to destination > 200m)]
1.1.1: Get driver's current location
1.1.2: Return current location
1.1.3: Use the input to find direction
1.1.4: Return direction to the destination
1.2: Display directions of route
1.3: Update
2: User confirms that they are near destination
2.1: Exit from navigation
2.2: Return to main page
2.3: Update

Use Case ID:	7		
Use Case Name:	Saving destination		
Created By:	Ivan	Last Updated By:	Ruxing
Date Created:	17/02/2023	Date Last Updated:	03/03/2023

Actor:	Driver
Description:	The driver would be allowed to save the destination that he/she has visited.
Preconditions:	- Driver's identity has been authenticated. - Driver is logged in.
Postconditions:	- Driver's visited destination has been saved in the database.
Priority:	Normal. Not essential to key feature of finding car parks.
Frequency of Use:	Daily, assuming that everyday there exists someone who will drive
Flow of Events:	1. Driver input location via search function. a. The website sends user input to the system. b. System sends information to the database to find the nearest car park. 2. The website returns the nearest car park. 3. The website generates recommended routes. 4. Driver drives to the location and uploads a photo. 5. The website validates information. 6. The website stores the location under "Saved Places".
Alternative Flows:	AF- 4: Driver does not travel to the location. 1. Driver inputs location. 2. Driver does not travel to the location. 3. Void event.

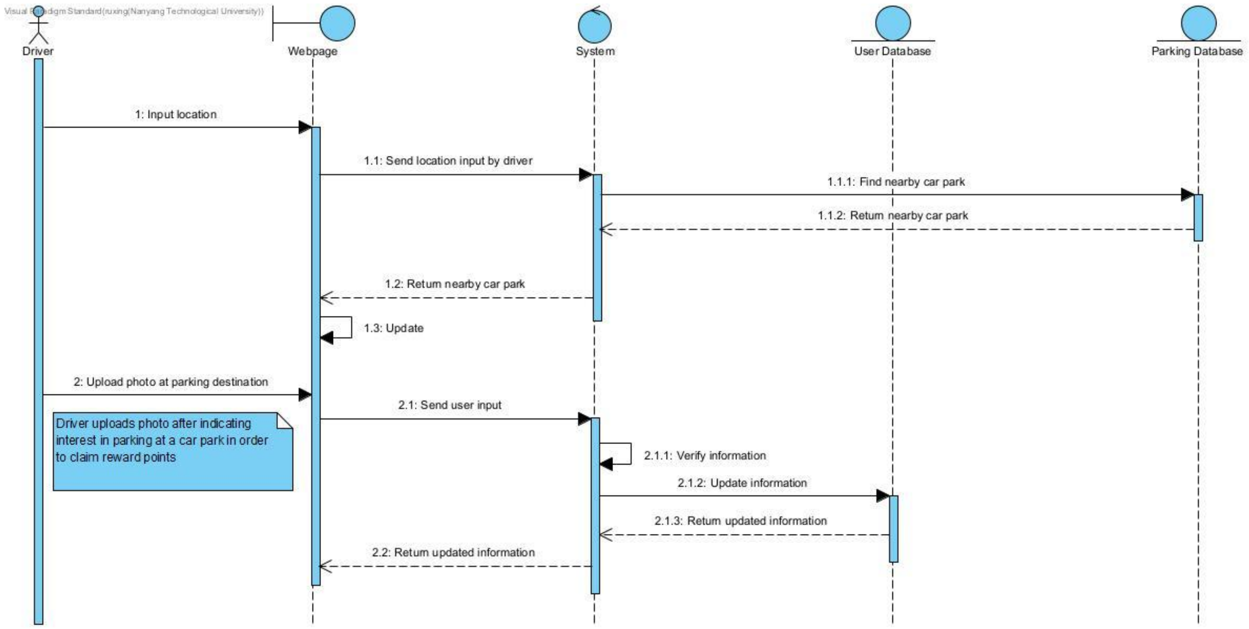
Exceptions:	-
Includes:	Filter search by map or list.
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-


Driver
Webpage
System
User Database
Parking Database
1: Input location
1.1: Send location input by driver
1.1.1: Find nearby car park
1.1.2: Return nearby car park
1.2: Return nearby car park
1.3: Update
2: Upload photo at parking destination
Driver uploads photo after indicating interest in parking at a car park in order to claim reward points
2.1: Send user input
2.1.1: Verify information
2.1.2: Update information
2.1.3: Return updated information
2.2: Return updated information

Use Case ID:	8		
Use Case Name:	Going interest		
Created By:	Ivan	Last Updated By:	Ruxing
Date Created:	17 Feb 2023	Date Last Updated:	27/02/2023

Actor:	Driver
Description:	The driver can travel to destinations saved under “Saved Places”.
Preconditions:	- Driver’s identity has been authenticated. - Driver is logged in.
Postconditions:	- Driver’s interested destination has been saved in the database
Priority:	-
Frequency of Use:	Once a week, assuming that every week, there would be someone who is interested in visiting certain places and saving them.
Flow of Events:	1. The driver inputs location via search function. a. The website sends the system the user input. b. The website finds nearby car parks in the parking database. c. The website returns a nearby carpark and updates the master UI. 2. Driver wants to add/remove “Saved Places” and sends a request to the website. 3. The website verifies the current status and updates the database accordingly.
Alternative Flows:	AF-3: Driver does not store the destination.

Exceptions:	EX 3: If current status is the same as intended status, raise an error message.
Includes:	Filter search by map or list.
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

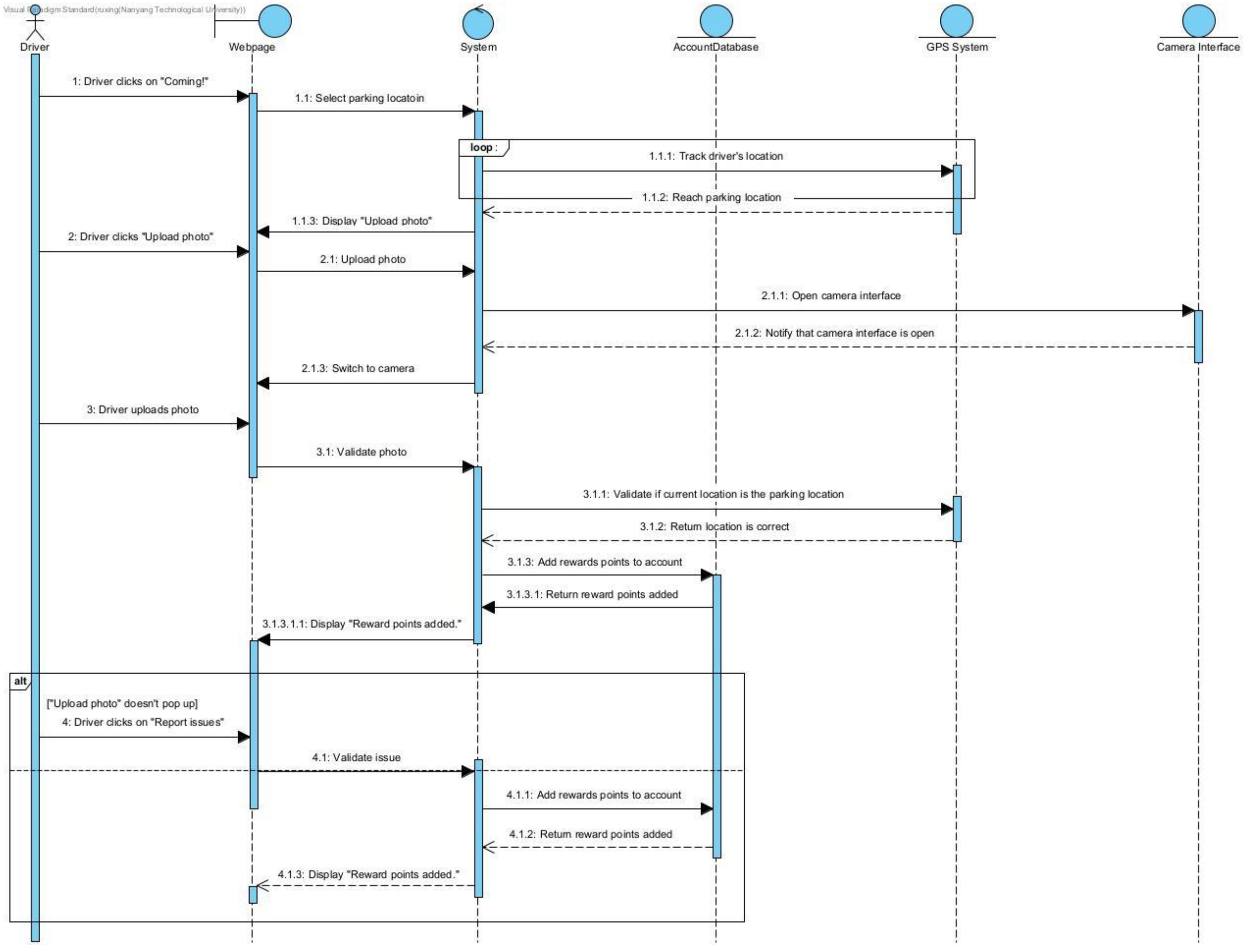

Driver
Webpage
System
User Database
Parking Database
1: Input location
1.1: Send location input by driver
1.1.1: Find nearby car park
1.1.2: Return nearby car park
1.2: Return nearby car park
1.3: Update
2: Upload photo at parking destination
Driver uploads photo after indicating interest in parking at a car park in order to claim reward points
2.1: Send user input
2.1.1: Verify information
2.1.2: Update information
2.1.3: Return updated information
2.2: Return updated information

Use Case ID:	9		
Use Case Name:	Receive Reward Points		
Created By:	Ruxing	Last Updated By:	Ruxing
Date Created:	19/02/2023	Date Last Updated:	03/03/2023

Actor:	Driver
Description:	To get reward points after the driver has indicated his interest in parking at a parking location and successfully parked at that location.
Preconditions:	- Driver has an account with the web-site. - Driver indicates his interest in parking at a location.
Postconditions:	- Driver receives reward points.
Priority:	Driver has to be logged into his account.
Frequency of Use:	Daily, assuming that everyday, someone drives and claims reward points after parking.
Flow of Events:	1. Driver chooses a parking location and indicates his interest in parking at that location. 2. Driver drives to the car park where he has indicated interest. 3. Driver takes a photo of his car parked at that location and uploads that photo. a. Driver selects the button "Coming!" to indicate his interest in using the car park. b. Driver proceeds to the car park. c. A prompt pops up on the web-site and displays "Upload photo." when the driver reaches the indicated location. d. Driver takes a photo of his car parked at the indicated car park.

	e. 3. 5) The web-site verifies the photo against his current location. 4. Driver receives reward points.
Alternative Flows:	AF- 3: The prompt "Upload photo." is not displayed and the driver is unable to upload the photo. 1. Driver clicks on "Report issue" under the account tab. 2. Driver provides a short description of the issue and uploads a photo of the car parked at the indicated location. 3. Admin reviews the issue and awards or denies reward points to the driver accordingly.
Exceptions:	EX1: Driver does not have an account yet. 1. Driver is not able to indicate his interest for a parking location. “Please sign in.” is displayed.
Includes:	-
Special Requirements:	GPS and camera interface is working.
Assumptions:	- Users are all acting in good faith – they won’t misuse the “Report issue” option. - Users’ GPS works well doing the duration of them using.
Notes and Issues:	-

Get Points

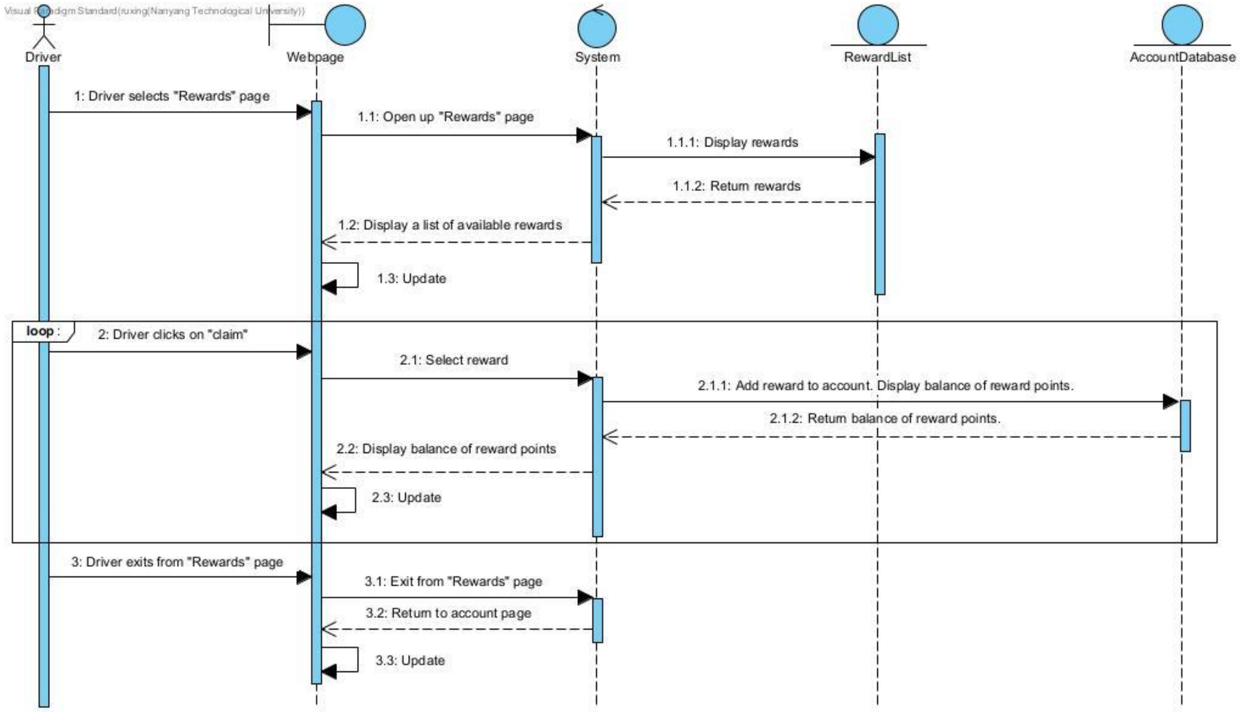

Driver
Webpage
System
AccountDatabase
GPS System
Camera Interface
1: Driver clicks on "Coming!"
1.1: Select parking locatoin
loop :
1.1.1: Track driver's location
1.1.2: Reach parking location
1.1.3: Display "Upload photo"
2: Driver clicks "Upload photo"
2.1: Upload photo
2.1.1: Open camera interface
2.1.2: Notify that camera interface is open
2.1.3: Switch to camera
3: Driver uploads photo
3.1: Validate photo
3.1.1: Validate if current location is the parking location
3.1.2: Return location is correct
3.1.3: Add rewards points to account
3.1.3.1: Return reward points added
3.1.3.1.1: Display "Reward points added."
alt
["Upload photo" doesn't pop up]
4: Driver clicks on "Report issues"
4.1: Validate issue
4.1.1: Add rewards points to account
4.1.2: Return reward points added
4.1.3: Display "Reward points added."

Use Case ID:	10		
Use Case Name:	Claim Reward		
Created By:	Ruxing	Last Updated By:	Ruxing
Date Created:	19/02/2023	Date Last Updated:	03/03/2023

Actor:	Driver
Description:	The driver claims rewards using his reward points.
Preconditions:	- Driver has an account with the web-site. - Driver clicks on “claim” to claim a reward.
Postconditions:	- Driver receives the reward he chose and his reward points balance. OR - Drivers receive the message “Insufficient reward points.
Priority:	The driver has to be logged into his account.
Frequency of Use:	Once every week, assuming that every week someone collects enough points to claim rewards or he simply wants to scroll through the reward list.
Flow of Events:	1. Driver logs into his account, using his username and password. 2. Driver selects "Rewards", which displays his reward point and a list of available rewards. 3. Driver clicks on the “claim” button to claim a reward from the list of available rewards and the balance of his reward points displays. 4. Driver exits from the "Rewards" page and returns to the “Home” page.
Alternative Flows:	AF- 3: Driver does not claim any rewards. 1. Driver scrolls through the list of available rewards.

	2. Driver exits from the “Rewards” page and returns to the “Home” page.
Exceptions:	EX1: Driver does not have an account yet. 1. No rewards are displayed on the “Rewards” page. “Please sign in.” is displayed on the “Rewards” page. EX3: Driver does not have sufficient reward points. 1. Driver cannot claim the reward. “Insufficient reward points.” is displayed.
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	The available rewards are provided by corporates working with us.

Claim Reward



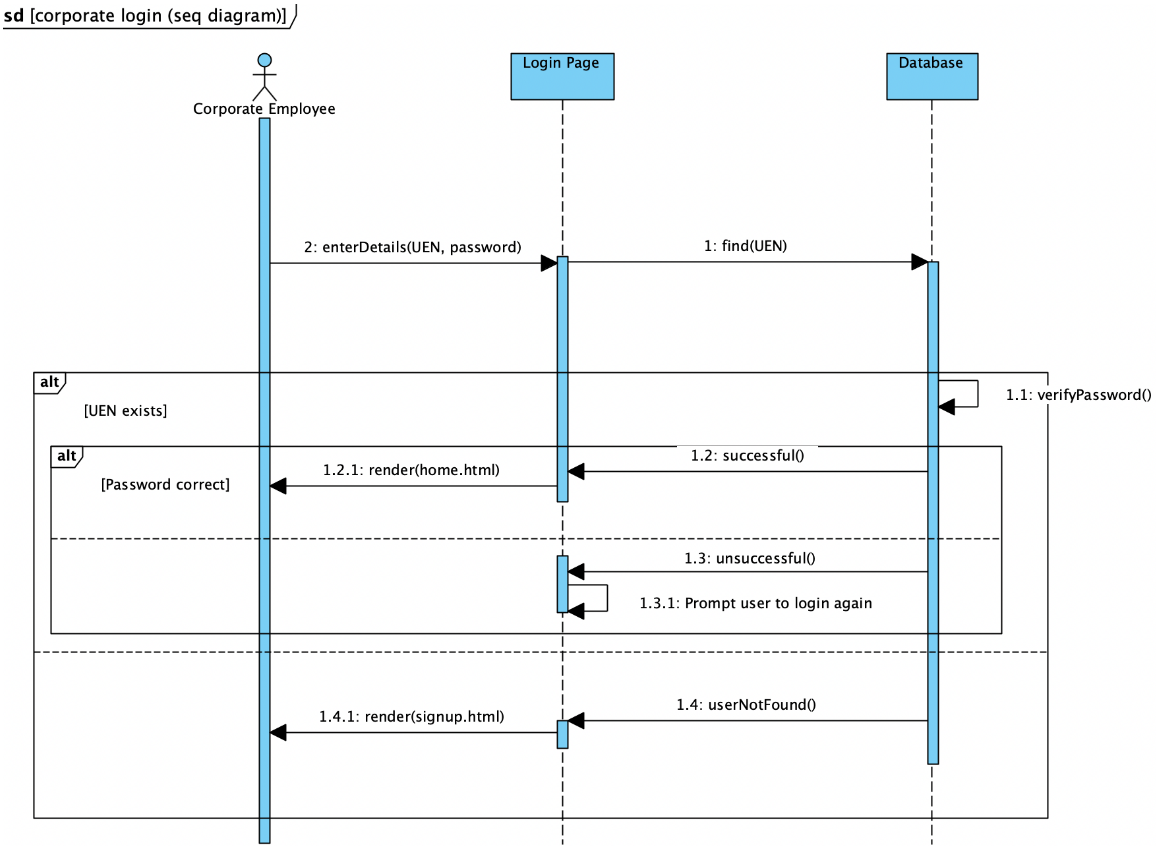
Driver
Webpage
System
RewardList
AccountDatabase
1: Driver selects "Rewards" page
1.1: Open up "Rewards" page
1.1.1: Display rewards
1.1.2: Return rewards
1.2: Display a list of available rewards
1.3: Update
loop
2: Driver clicks on "claim"
2.1: Select reward
2.1.1: Add reward to account. Display balance of reward points.
2.1.2: Return balance of reward points.
2.2: Display balance of reward points
2.3: Update
3: Driver exits from "Rewards" page
3.1: Exit from "Rewards" page
3.2: Return to account page
3.3: Update

11. Corporate Login

Use Case ID:	11		
Use Case Name:	Corporate Login		
Created By:	Joanna	Last Updated By:	Joanna
Date Created:	18/02/2023	Date Last Updated:	27/02/2023

Actor:	Corporate Employee, Database
Description:	Login flow for corporate employee
Preconditions:	1. Corporate account already exists
Postconditions:	1. Account is logged in and user is redirected to the rewards creation page
Priority:	-
Frequency of Use:	Once a day
Flow of Events:	1. Employee enters UEN and password. 2. Database checks if UEN exists. 3. If UEN exists, the database checks if the password is correct. 4. If the password is correct, login is successful, the home page is rendered.
Alternative Flows:	1. Employee enters UEN and password. 2. Database checks if UEN exists. 3. If UEN does not exist, an error message is flashed. 1. Employee enters UEN and password. 2. Database checks if UEN exists. 3. If UEN exists, the database checks if the password is correct. 4. If the password is incorrect, login is unsuccessful, and the login page prompts the employee to re-enter details.

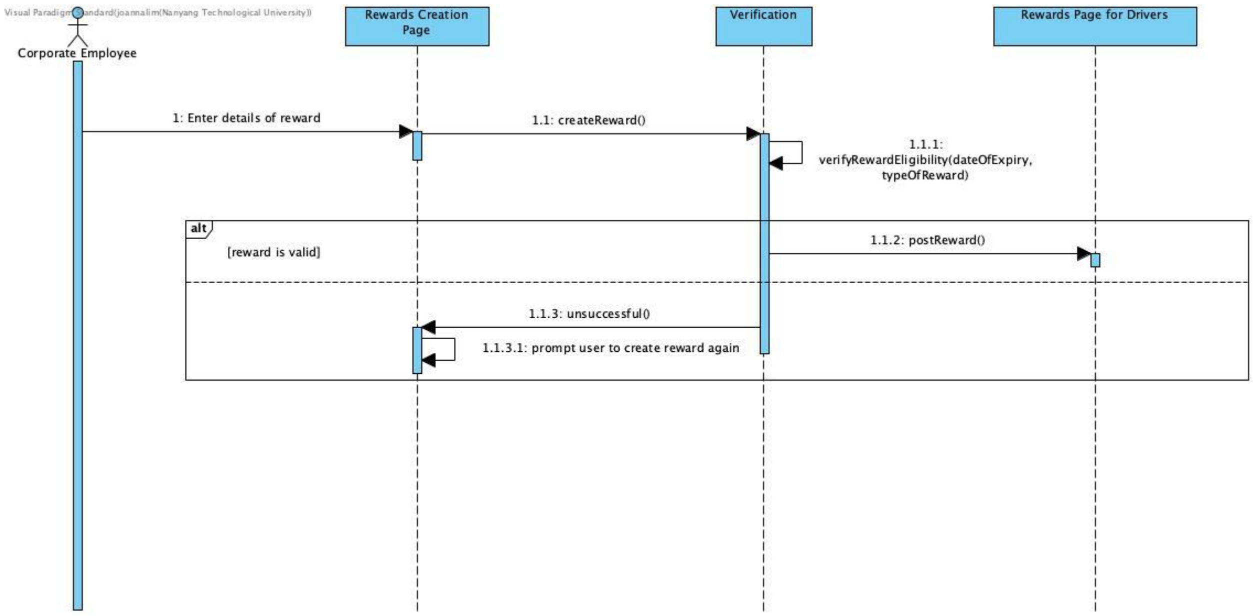
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	UEN will not be verified against ARCA due to a lack of a publicly provided API.



12. Rewards Creation

Use Case ID:	12		
Use Case Name:	Create Rewards		
Created By:	Joanna	Last Updated By:	Joanna
Date Created:	18/02/2023	Date Last Updated:	27/02/2023

Actor:	Corporate Employee
Description:	Reward Creation process
Preconditions:	Employee account exists and is logged in
Postconditions:	Reward will be posted to the rewards page
Priority:	-
Frequency of Use:	-
Flow of Events:	1. Employee creates a reward 2. Website verifies the eligibility of the reward 3. If eligible, reward is posted to the Rewards Page
Alternative Flows:	1. Employee creates a reward 2. Website verifies the eligibility of the reward 3. If ineligible, reward creation page prompts the employee to recreate a reward
Exceptions:	-
Includes:	-
Special Requirements:	Employee account
Assumptions:	-
Notes and Issues:	-


Corporate Employee
Rewards Creation Page
Verification
Rewards Page for Drivers
1: Enter details of reward
1.1: createReward()
1.1.1: verifyRewardEligibility(dateOfExpiry, typeOfReward)
alt
[reward is valid]
1.1.2: postReward()
1.1.3: unsuccessful()
1.1.3.1: prompt user to create reward again