Software Engineering Course Final Project

Table of Contents

Chapter I – introducing the Problem Domain

Introduction to the Problem Domain

The startup company "Rupintremp" has decided to develop a new system in favor of efficient management of cooperative travel, to reduce the congestion from public transportation and the roads and reduce CO2.

These days, there is no public / cooperative transportation system in Israel that guarantees availability and efficiency in real time for passengers who do not want to rely on inefficiency and rigid public transportation times.

Therefore, the company decided to set up a system that will connect drivers and passengers who are nearby, and provide an additional envelope for solving public transportation - the "**Get-Ride**" app.

The system will automatically connect passengers and drivers so they will be able to travel together, resulting in reduced traffic jams, and as a result of the more efficient traffic situation on the roads, the annual general GDP as well as the annual GDP, will both increase significantly as people spend less time standing in traffic jams and more Time at work and in personal activities, which can lead to an increase in the national happiness index.

In addition, the association of passengers with drivers encourages a market and a free economy, since now every driver who meets the criteria can profit legally and openly, which will encourage consumerism and competition in the public transportation market, which can indirectly streamline the national public transportation system.

- Users of the system are the general population aged 16 and over.
  - A user can be both a driver and a daily commuter.
  - * A driver must be 18+ years old.

• When a passenger wants to travel to a destination (booking a new trip), he will log in to the mobile app and coordinate a trip.

• For each order the customer will be asked to enter the destination address and source address as well as the following details:

- ▪ Desired Connection Time
- ▪ Number of passengers
  - ○ The Source address can be entered automatically according to location permissions given to the app on the smartphone.

- The system will check if there is at least one driver available within the 30 minutes' drive zone from the pickup point.
  - ○ Using an external API to calculate travel times.
  - If available, the system will show the passenger his\her data (profile picture, number of stars and reviews, as well as the cost of travel by travel time).
  - If it doesn’t exist, the system will show the passenger an error message stating that no driver was found nearby.
- If the passenger chooses a driver, the driver will be notified in his\her applicable interface, where he can choose whether to accept on reject the shuttle offer.
  - If the passenger chooses to receive the drive, the driver will confirm it in his applicable interface, the passenger will receive confirmation message in his\her application, and the ride will start.
  - If the driver chooses to reject the shuttle offer, a suitable alert will be sent to the passenger's app, and it will be returned to the screen of possible drivers to be updated according to the current status of available drivers.
    - ○ Rejected drivers will not be shown to the passenger again.
- The system will store all the details of users and drivers in dedicated Database for each type of user (passenger or driver)

- The system will be able to display personal statistics for each driver- ie the amount of trips he made, the number of passengers he took in total, how much money he earned, fuel expenses, etc.
- The system will be able to display personal statistics for each passenger – how many trips he\her made this month, total cost, how many KM traveled, how many reviews given. In addition, it will be possible to view – but not to edit – them.
- At the end of each trip the system will ask the passenger to enter a driver review.
    - Review based on text, and star rating (between 1 [lowest] and 5 [highest])
        - A **negative** review will lower the driver's rating and he\she will appear lower in the app's drivers menu.
        - A positive review will increase the driver's rating so that more passengers are recommended nearby.
- A passenger will be able to add a driver he likes to a list of favorites, which he\her will be able to order more easily (depending on the distance as mentioned above)
    - In addition, the system will present to the passenger, the preferred drivers (if available) – in higher priority for the desired trip.

Chapter II – Requirements Definition

Functional requirements

Req #	Priority	Requirement
Req #1	5	The system will allow registration as follows: Registration of a passenger to the system: The following details: first name, last name, sex, city of residence, telephone number, email address, date of birth, credit card / PayPal information (for debit or credit), is an accessible vehicle needed, Preferred languages. Registration of a driver for the system: The following details: All the details the system asks of a passenger, but in addition the following parameters: ID card, what languages are spoken, mother of an accessible vehicle, type of vehicle, number of seats in the vehicle, license plate number, photocopy of valid driver's license, Israel Police conviction sheet, profile picture.
Req #2	5	To make a reservation the system will ask the passenger to enter the following details: destination, whether you are interested in a private trip (without picking up additional passengers) and a free text field (note field)
Req #3	5	After entering a collection point and destination, the system will show the passenger all the relevant drivers who are in his area (within a range of up to 30 minutes' drive) and ready to receive trips. * An available driver will be defined according to the following criteria: The driver must be connected to the system, the driver must be within the collection range.
Req #4	3	The system will show the passenger the reviews and ratings of the drivers in descending order (from a driver with five service stars to a driver with one service star).
Req #5	2	When choosing a driver, the system will display the estimated time of arrival at the collection point, and will also display the following details: a picture of the type of vehicle the driver should arrive with + license plate tax, number of seats in the vehicle and the cost of travel with the same driver.
Req #6	5	The system will display an option to select the driver and place the trip booking.
Req #7	4	The system will display to the passenger and driver the "travel" screen that shows the travel route on a map by using an external API as well as the arrival times.
Req #8	5	After selecting a driver for the trip and submitting a new trip: The system will send the selected driver a push notification about the trip booking, which will allow him to decide whether he accepts the booking or rejects it. If the driver rejects the request, the system will cancel the ride submitted and allow the passenger to select another driver from the list of available drivers. If the driver received the ride, the passenger and driver will be shown a "ride" screen (Req # 9). * In any case, the system will send the passenger a push alert that updates the status of the reservation (accepted or rejected by the driver). * If a driver did not respond within 10 minutes of sending the request - the system will cancel the trip and send an update message to the passenger
Req #9	5	At the end of the trip, the system will automatically debit the passenger's credit card or PayPal account and will send the user an invoice by email. The credit to the driver's account will be made by the credit card company
Req #10	3	The system will allow drivers to be rated. Driver rating will be performed according to the following criteria and according to the star rating method (1-5 stars): driving safety, ride comfort, vehicle cleanliness, punctuality. In addition, after entering these parameters, the passenger will be allowed to enter free text as a free review of the driver. The system will keep the rating given by the user to the star table and ratings of the same driver.
Req #11	3	The system will allow cancellation of a travel booking by the passenger after it is performed (up to 5 minutes) for a payment of 25% of the planned trip cost. If the passenger cancels the travel booking, the system will send a push alert to the driver stating that the trip is canceled.
Req #12	3	The system will allow you to update and change all user details (driver or passenger) at any time through a suitable menu within the app except for the ID card and date of birth and will also allow password reset
Req #13	2	The system will allow each user's travel history to be displayed as follows: For each passenger his travel history according to the following parameters: driver name, vehicle type, travel date, review given to that driver, departure point and destination, travel time, arrival time and so on. Also, the cost of travel in addition to the option to view the invoice. For each driver his transportation history according to the following parameters: names of the passengers on the same trip, date of travel, point of departure and destination.
Req #14	1	The system will make it possible to issue a monthly report for each user that contains travel / transportation statistics
Req #15	2	The system will detect if the passenger has entered a low star rating (e.g., 1 or 2), and if so will present him with a question as to whether it is necessary to contact customer service. If the passenger chooses yes, the referral will be forwarded to the customer service department, where he will be analyzed by a person, and if necessary, the driver will be reprimanded, and even more serious measures will be taken against him at his discretion.
Req #16	2	The system will allow a customer service representative to make a refund to a passenger who was not satisfied in exceptional cases that require it. It should be noted that the company is the one that returns the money to the passenger, and in such a situation the system will alert the driver about the incident and that he is in financial debt to the company and must arrange it with customer service. If no refund is made by the driver, he will be removed from the system.
Req #17	1	The system will allow users to search for and rate other drivers, coordinate a new ride.
Req #18	1	The system will allow users to share the profile of other drivers on social media and instant messaging apps (WhatsApp / SMS, etc.). * A driver can share himself
Req #19	3	The system will allow users to add favorite driver profiles to their favorites list, for quick contact for future trips.

Req #20	1	The system will check the user activity in the app every 10 minutes. If there is no use of the application (and there is no active travel) or alternatively the application is closed on the mobile device, the system will disconnect the user from the system
Req #21	5	The system will show the user the home screen that contains all the options that can be made (start a new trip, view history, search for a driver, file a complaint, edit details, view favorite drivers)

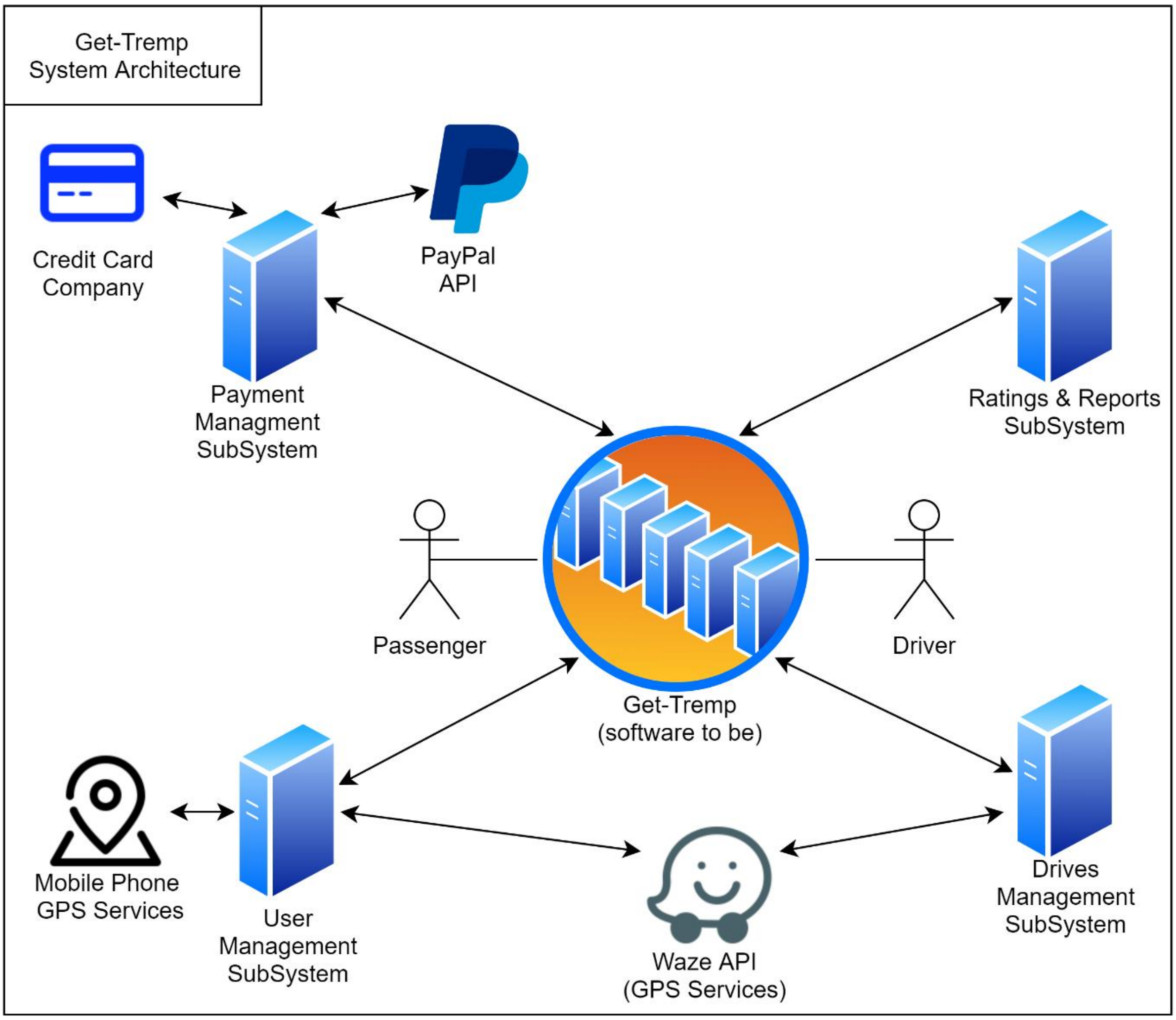
Quality Requirements

Quality Req #	Category	Requirement	Comments
Req #1	Data Security	Prevention of counterfeit travel booking	For example by creating verification and authentication mechanisms (e.g. using 2FA) or connecting biometrically
Req #2		Reducing system hacking risks	For example, by charging users the use of passwords of a certain character length / combination.
Req #3		Using the SSL protocol and the HTTPS protocol in all communication to the server	
Req #4		Store credit card and PayPal account information encrypted on a DB server and according to PCI standard	
Req #5		Payment confirmation in the application will only be made by user verification biometrically or with the help of a unique PIN code for the user	If the user selects a payment using a PIN code, the user will select the code, and he will be solely responsible for it - this method is considered less secure than the biometric and recommended method (recommended by the system using a Recommended graphic indication)
Req #6		DB user data stored will be encrypted using the AES algorithm	
Req #7		Each user will only be able to access his personal information	
Req #8		Information stored in the application will be stored on the end devices with end to end encryption so that no other application / system running on the device or alternatively attacks the server, you will not be able to "intercept" information from the application and / or the server	

Req #9		The system will back up the database every 24 hours to the Offsite database	
Req #10	UI\UX	AutoComplete support	
Req #11		Language support - Hebrew, English (default language), Arabic and Russian	
Req #12		Minimal and intuitive interface	
Req #13		An application adapted for Android and iOS	There is no need for compatibility with smart TVs
Req #14	performance	Switching from menu to menu and between steps should not exceed 0.5 seconds	
Req #15		All images will be saved in DB as JPG files, and all files will be saved as PDF files - (such as receipts)	
Req #16		Updating a record in DB does not exceed 1 second	
Req #17		The minimum time required to book a new trip should not exceed 10 seconds in total	
Req #18		Access to DB for about 10,000 users simultaneously (drivers and passengers)	
Req #19		Storage and management of up to about 100,000 trips per year	
Req #20		Maintenance of DB and storage servers with a volume of about 100TB	Adding storage volumes according to traffic and storage needs
Req #21		The system will allow a deviation of up to 60 meters in the GPS position	
Req #22		The app will verify network connectivity on the end device, before displaying the main menu	
Req #23	maintenance	Code-level system maintenance will include adding features and	

		new functionality, as well as fixing bugs in existing features	
Req #24		The system will be written and set up in a modular way so that adding a new feature will not result in significant changes in the way the code is implemented	
Req #25	technology	Development using #C server side, JAVA for Android devices and Swift 5 for iPhone devices	
Req #26		The system will calculate driving distances with an accuracy of tens of meters (displayed in kilometers)	For example, 1.04 km
Req #27		The system will calculate the travel costs with the accuracy of a penny (two digits after the point)	For example, 42.86 NIS
Req #28	budget	The cost of the development will not exceed NIS 500,000	
Req #29		The cost of monthly maintenance will not exceed NIS 20,000, including server maintenance	
Req #30		For each credit clearing (payment for travel) a commission of 5% of the transaction amount will be charged	For example, if a trip cost NIS 100, the system will bring the company NIS 5
Req #31	development	The duration of development shall not exceed six months	
Req #32		The development will be carried out according to a meticulous timetable	
Req #33	employees	The company will employ ten employees who will function as customer service	For dissatisfied customers and exceptional public inquiries

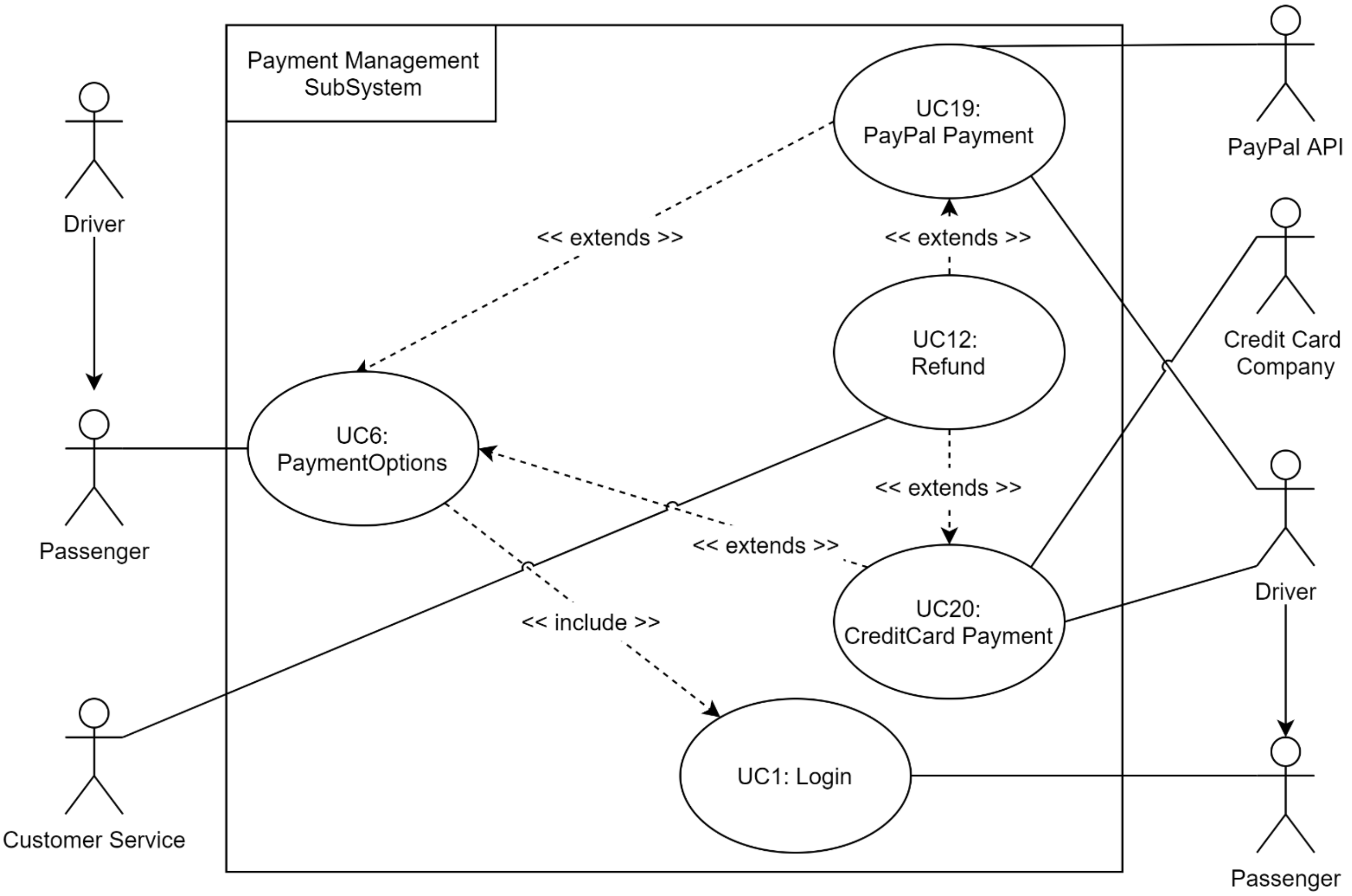
System Architecture



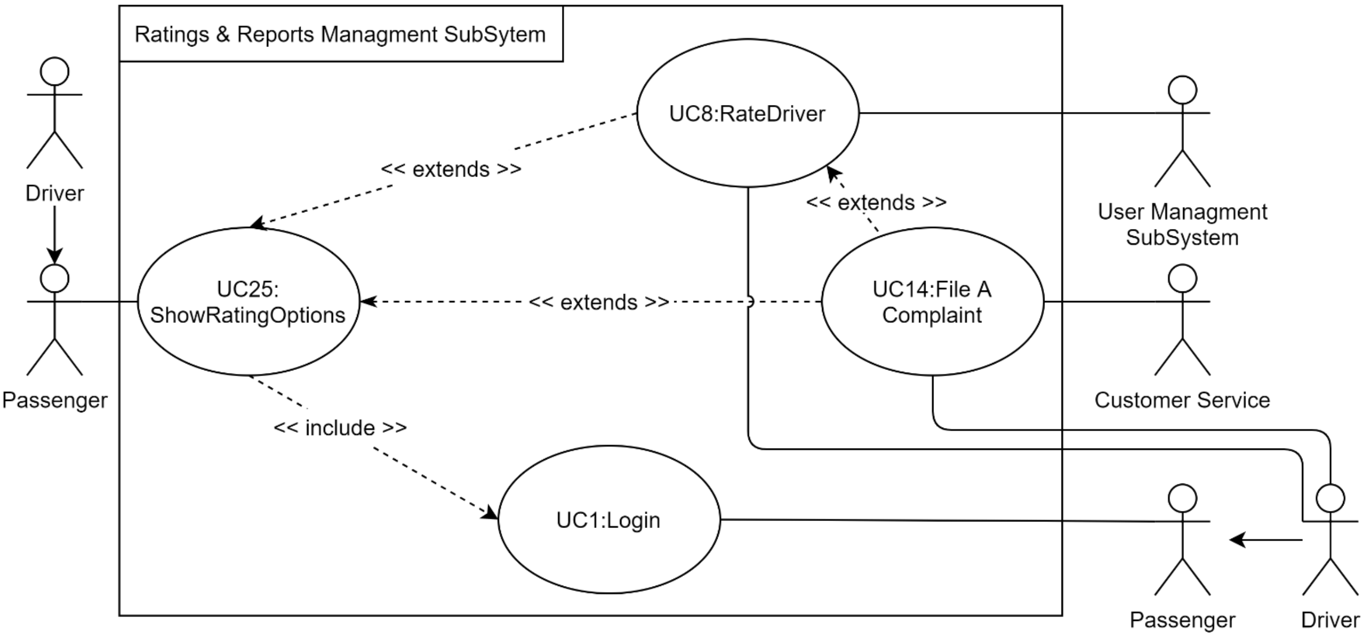
Chapter III – Requirements Specification

Use Case Diagrams

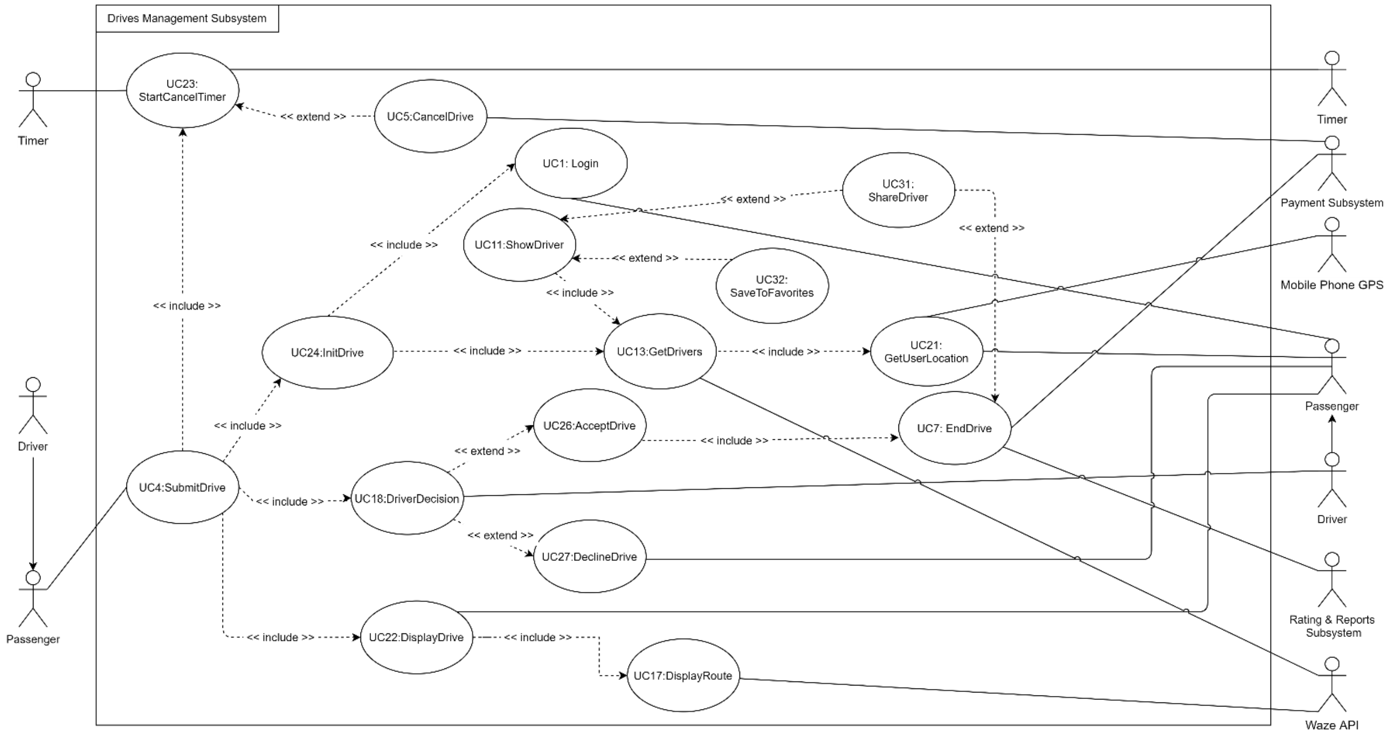
Payment Management Subsystem



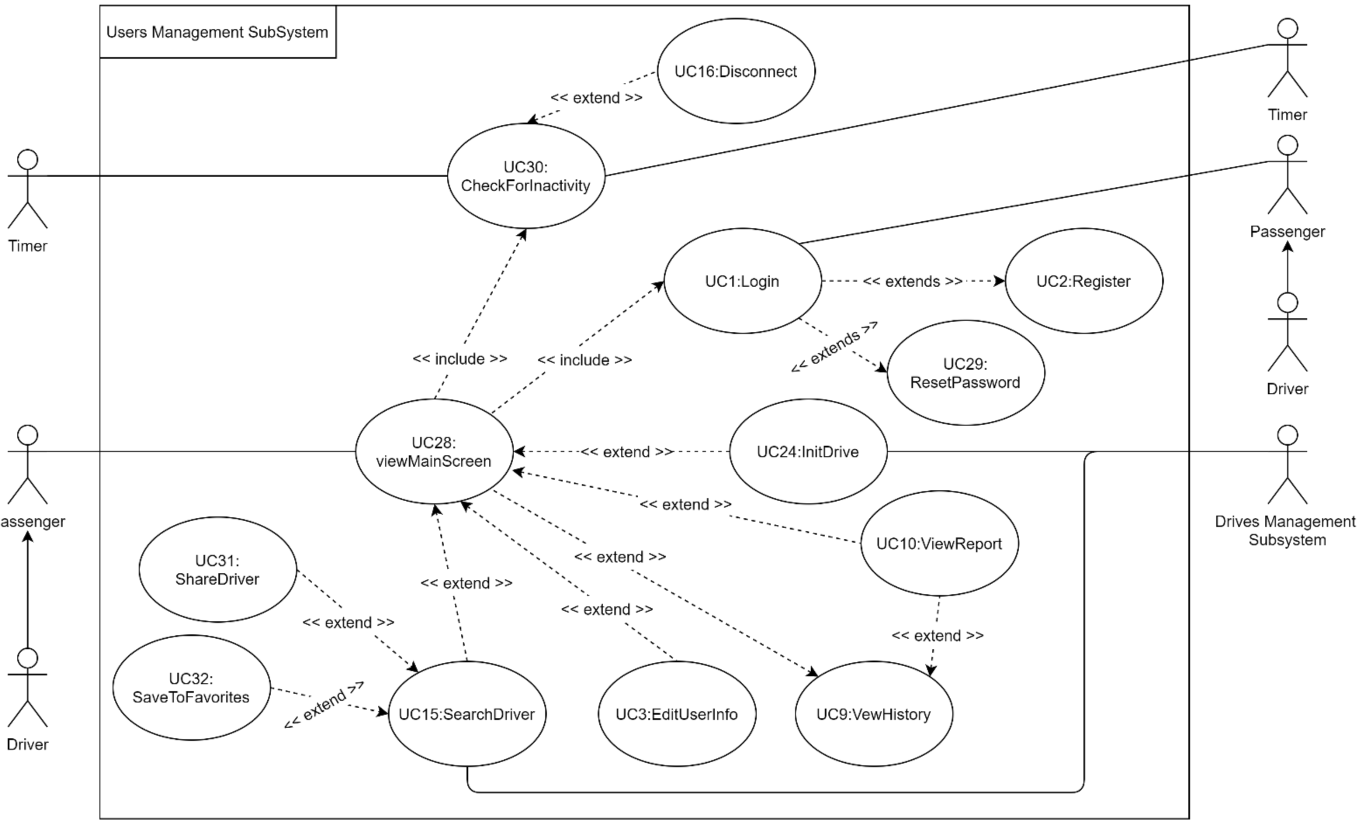
Ratings & Reports Management Subsystem



Drives Management Sub System



Users Management Sub System



Glossary

UC#	UC	Goal	Actors	Scenarios
1	login	Let the Drivers nor Passengers log into the system or asking to register if user does not exist in DB	-Driver -Passenger	-->User will enter to the application from mobile device home screen.
				<--System prompt user for login info (email & password).
				-->User will provide info.
				<--System confirms user login.
				<--System lets the user into the main menu
				Alternative scenario:
				<--User login failed.
				<--System will not let the user into the system and will prompt the user with a choice to register as new user or resetting password.
2	register	Let new user register as driver or just as a passenger	-Driver -Passenger	<--System will ask the user whether to register as a driver or as a passenger.
				-->User will select desired role.
				<--System will display the proper registration form (REF Func req #1)
				-->User enters all required and optional fields and press the “register” button.
				<--System validates entered info.
				<--System saves the entered info in the DB, and automatically logs the user in
				Alternative scenario:
				<--Validation failed.
				<--System prompts the user for the mismatched field and wait for him to re-enter info into the field.
3	EditUserInfo	The system will enable users to change their data except certain-crusial information fields (FuncREQ#26)	-Driver -Passenger	-->User will press “edit info” button
				<--System will show him registration form filled with the current user info.
				-->User will select field\s and make changes as he wishes.
				-->User will press “Save” button
4	SubmitDrive	The system will enable the passenger to order new drives	-Passenger	-->Driver approved the passenger's drive order (REF UC 18)
				<-- System will notify the passenger and transfer him to the drive screen
				<-- System will start cancelation timer (REF UC 5)
				<-- System will display the drive screen (REF UC 22)
				Alternative scenario:
				-->Driver declined the passenger's drive order (REF UC 18)
				<-- System will discard the submitted drive
				<-- System will notify the passenger and take him back to initDrive stage (REF UC 24)
5	CancelDrive	Passenger will be able to cancel ordered drive before the driver comes to pick him up (5 minutes from submitting a drive (ref UC:23)	-Driver -Passenger -Timer -Payment Subsystem	-->After ordering a drive, passenger can cancel it within 5 minutes after submitting
				<-- Payment Subsystem will handle a 25% payment from the passenger for the driver as a cancelation fine
				<--System will stop the current drive and send push notification to the driver stating the drive was canceled.
				Alternative scenario: --> Driver denied Passenger's submitted drive
				<-- System will cancel the submitted drive
				Alternative scenario:
				--> Passenger submitted a drive
				--> Driver did not reply within 10 minutes of submitting

				<-- System will cancel the submitted drive <-- System will notify the passenger and take him back to initDrive stage (REF UC 24)
6	PaymentOptions	system will display the passenger with a payment options screen	-Passenger	<--System will notify the passenger to approve and pay drive costs via a credit card or PayPal. -->Passenger will choose from the two. (Ref UC 19 & 20)
7	EndDrive	System will end the drive and save it to history	-Passenger	<--System will end the drive and save drive info to the driver's and passenger's history DB
8	RateDriver	Passenger will be able to rate the driver after drive ends	-Passenger	<--At drive end, system will ask the user to enter review for the driver. -->Passenger chose to enter a review <--System will display the passenger with a review screen asking for: stars review (mandatory, REF FuncReq10), free text review(optional). **Alternative scenario:** -->Passenger chose to decline and continue <--System will return the passenger to the main screen (Ref UC28)
9	ViewHistory	System will display history of all drives (ordered top down from the latest to earliest)	-Passenger -Driver	<-- System will present "view history" button in the main menu --> User will choose "view History" option. <--System will display the user with a preview of he's drives history (5 per at a time)
10	ViewReport	System will display full drives history Report including costs and other statistical info	-Passenger -Driver	-->Both passengers and drivers will have "view Report" button inside the view history screen <--System will display the user with he's full drives history Report include costs, receipts and statistical info
11	ShowDriver	Let passengers see driver's fare rates (drive costs) and reviews, having the option to start a drive.	-Passenger	-->Passenger will select a specific driver from available drivers list (FuncREQ#3) <--System will request the DB for the driver's statistics and present it to the passenger. -->Passenger can now press the "select" button to select the driver and start a new drive (UC#4).
12	Refund	System will refund Passenger account if decided by customer service representative	-Credit Card Company -PayPal API -customer service	<--System sends to the credit card company a request to refund passenger's account <-- Credit card company transfers the refund directly into Passenger's account.
13	getDrivers	System will get drivers and passenger's current location and calculate best route for each driver, and display the drivers list to the passenger	-Mobile Phone GPS - Waze API -Passenger -Driver	<--System will query available drivers mobile devices to get their current position. (Available driver = logged in to the system) <--System will display to the passenger a list of available drivers in he's area with all of their rating and info
14	File a complaint	An unsatisfied Passenger can file a complaint against an offensive or dangerous driver	-Passenger -Customer service	-->Passenger files a complaint after an unsatisfied drive. -->Customer service deals with the issue and find the best solution for this case. -->Customer service decides that the passenger deserves a refund for the drive. (REF UC12) -->Customer service closes the dispute alerting the driver about the incident.

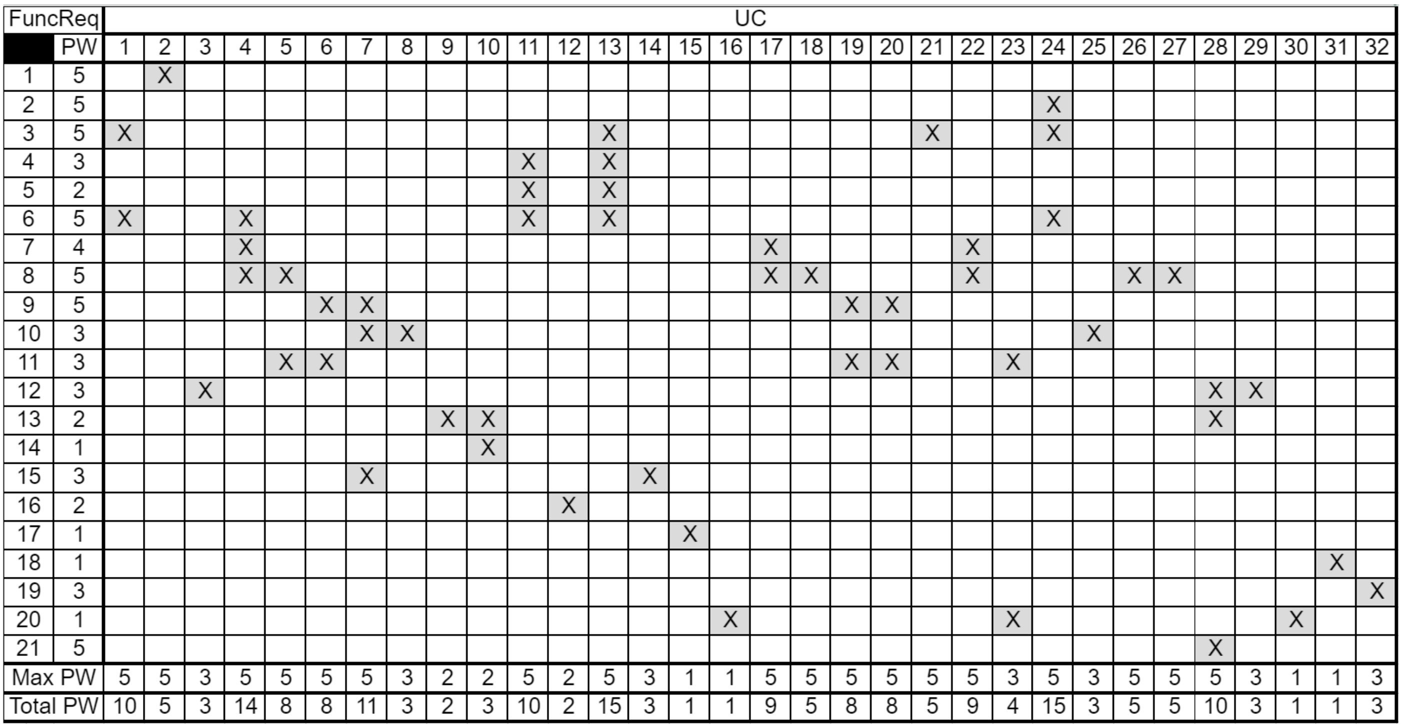
				Alternative scenario:
				-->Customer service decides that the driver should no longer provide services as part of “Get-Tremp” system and suspends he's account.
15	SearchDriver	To find a specific driver	-Passenger	-->Passenger taps “search driver” button
				<--System displays textbox to enter driver’s name\id
				<--System displays to the passenger the driver info
				-->User can select the driver for a new drive, and "order" him, which will initiate a new drive automatically (REF UC 24)
				Alternative scenario:
				<--System did not find searched driver in the DB.
				<--System will display the passenger a message that no divers were found and takes him back to the search screen.
16	disconnect	system will disconnect the user from the system	-Passenger -Driver	<-- System will disconnect user automatically (log him out).
17	displayRoute	The system will display drive route on a live map	-Waze API -Phone GPS	<--System will display drive route on a live map on the “drive” screen, using Waze API.
18	DriverDecision	The system will give the driver the ability to accept or decline incoming drives	-Driver	<-- System will display the recent passengers submitted drives, that want to drive with him
				Alternative scenario: the driver accepted the drive - REF UC 26
				Alternative scenario: the driver chose not to take the passenger - REF UC 27
19	PaypalPayment	system will allow payments (and refunds) with PayPal account	-Passenger -Driver -PayPal API	-->User chose to pay with PayPal
				<--system will send transaction info to PayPal’s API.
				<--PayPal API will perform transaction and send receipt via email to passenger.
20	CreditCardPayment	system will allow payments (and refunds) with credit card	-Passenger -Driver -Credit Card Company	-->User chose to pay with Credit Card
				<--System will send transaction info to CCC
				<--CCC will perform transaction and send receipt via email to passenger.
21	getUserLocation	system will use phone GPS to get his current location	-Passenger -Driver -Phone GPS	<--System will query user's GPS for current location
				-->User’s mobile device will return its current location
22	DisplayDrive	System will display the Drive Screen	-Driver -Waze API	--> Driver accepted the drive (REF UC# 4)
				<-- System will display the "Drive screen" which will show ETA to destination, estimated remaining drive time and a live map of the route
23	StartCancelTimer	System will start countdown timer (5 minutes)	-Timer	<--System will start cancelation timer (5 min) to enable drive cancelation within the given quota
24	initDrive	System will provide the user with required fields to Submit a drive	-Passenger	-->Passenger will press “new drive” button
				-->User will select driver and fill the required fields (REF FuncREQ 2)
				<--System will send push notification to driver’s mobile app asking for drive approval (REF UC18)
25	ShowRating Options	System will display rating and reviews menu	-Passenger -Driver	<--System will display rating and reviews menu
				-->User will choose weather to input a review or just watch driver's reviews
26	AcceptDrive	driver will accept the drive	-Driver	-->Driver will accept the drive
				<--System will submit the drive (REF UC4)
27	DeclineDrive	driver will decline the job offer	-Passenger -Driver	-->Driver will decline the drive
				<--System will notify the passenger and reset the drive, and take him back to the “drivers” screen to reselect driver (REF UC13)
28	viewMainScreen	system will display to the user the main menu	Passenger, Driver, Timer	<--System will display the logged in user with the main screen showing him the main screen options (Ref UC: 3,9,15,24)

29	ResetPassword	system will reset user's password	-Passenger -Driver	-->User forgot password and pressed "ResetPassword" button
				<--System will ask the user for a new password (will ask to enter twice for validation) and update DB accordingly.
30	CheckFor Inactivity	the system will automatically disconnect a user from the system after 10 minutes of inactivity	-Timer	<--System will check for user inactivity every 10 minutes.
				-->User active – do nothing
				Alternative Scenario:
				-->User inactive –app minimized \closed on mobile device, and no ongoing drives*
31	ShareDriver	system will enable users to share drivers profiles	-Passenger -Driver	-->user will be able to press "share driver profile" button
				<--System will share driver via social medias
32	SaveToFavorites	system will enable passengers to save drivers to favorite	-Passenger -Driver	-->Passenger will be able to "press save driver's profile" button
				<-- System will save driver profile into a Passengers favorite list

Actors Table

Actors	Actor's Goal	Goal UC#
Passenger	Edit profile info	UC3:EditUserInfo
	Submit a new drive	UC4:SubmitDrive
	Rate the driver	UC8:RateDriver
	See recent drives	UC9:ViewHistory
	View full drives report	UC10:ViewReport
	File a Complaint against misbehaving driver	UC14:File A Complaint
	Search a driver	UC15:SearchDriver
	Pay with paypal	UC19:Paypal Payment
	Pay with credit card	UC20:CreditCardPayment
	See an ongoing drive screen	UC22:DisplayDrive
	Share driver's profile via social media (facebook, whatsapp, twitter etc)	UC31:ShareDriver
	Save a driver to favorites	UC32:SaveToFavorites
Driver	Edit profile info	UC3:EditUserInfo
	Watch recent drives	UC9:ViewHistory
	Watch full drives report	UC10:ViewReport
	Accept the submitted drive	UC26:AcceptDrive
	Decline the submitted drive	UC27:DeclineDrive
Customer Service	Refund a user	UC12:Refund
	Handle the complaint filed by the passenger	UC14:File A Complaint
CreditCardCompany	Handle credit card payments	UC20:CreditCardPayment
Paypal API	Handle Paypal payments	UC19:Paypal Payment
Payment Subsystem	Initiate payment procedure and Handle the drive payment	UC7:EndDrive
	To fine the passenger that canceled the drive (25% of drives cost)	UC5:CancelDrive
UserManagment SubSystem	Save driver's review and rating	UC8:RateDriver
Ratings & Reports Subsystem	Offer rating after the drive ended and handle it	UC7:EndDrive
Timer	Cancel a drive within given time quota	UC5:CancelDrive
	Start Countdown before disabling the option to cancel drive	UC23:StartCancelTimer
	Check for user inactivity every x time units	UC30:CheckForInactivity
Mobile Phone GPS	Get the Passenger's current location in order to get nearby drivers	UC21:GetUserLocation
Waze API	Get drivers within 30min range of passengers location	UC13:GetDrivers
	Display the user with a live map of the drive	UC17:DisplayRoute

Traceability Matrix



FuncReq		UC																															
	PW	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
1	5		X																														
2	5																								X								
3	5	X												X								X			X								
4	3											X		X																			
5	2											X		X																			
6	5	X			X							X		X											X								
7	4				X													X					X										
8	5				X	X												X	X				X				X	X					
9	5						X	X												X	X												
10	3							X	X																	X							
11	3					X	X													X	X			X									
12	3			X																									X	X			
13	2									X	X																		X				
14	1										X																						
15	3							X							X																		
16	2												X																				
17	1															X																	
18	1																															X	
19	3																																X
20	1																X							X							X		
21	5																												X				
Max PW		5	5	3	5	5	5	5	3	2	2	5	2	5	3	1	1	5	5	5	5	5	5	3	5	3	5	5	5	3	1	1	3
Total PW		10	5	3	14	8	8	11	3	2	3	10	2	15	3	1	1	9	5	8	8	5	9	4	15	3	5	5	10	3	1	1	3

Conclusion

We can use the traceability matrix to figure out which use cases are important, which are less, and which are nice to haves.

High Priority Requirements

REQ: 1, 2, 3, 6, 7, 8, 9, 21

these requirements are classified with priority 4 and 5.

We gave those requirements the highest priority because they are crucial to the basic behavior of our system, which is to connect between a driver and a passenger in order to perform a drive together.

these requirements are the “beating heart” of our system, and without them the system will loose from its purpose.

UCs: 13, 24, 4, 7 ,1 ,11 ,28, 17, 22

These UCs are critical to our system.

We can see that they have a priority of 9 and above.

Without those, our system will be incomplete and key functionality of the system will be missed, causing the system to not work properly (if not work at all)

Low Priority Requirements

Req: 4, 5, 10, 11, 12, 13, 15, 16, 19

We gave those requirements a low priority of 2 and 3.

These requirements are features that are less critical to the system basic functionality, which is to connect between a driver and a passenger to perform a drive in a unifying system.

UCs: 2, 3, 5, 6, 8, 9, 10,12, 14, 18, 19, 20, 21, 23, 25, 26, 27, 29, 32

These UCs received priority from 3 to 8.

Those UCs help and support the high priority UCs and Functional requirements, and without those the system will function, but will not function at 100%.

Therefore, they are needed but they are not critical to the main purpose of the system.

Nice To Have

Req: 14,17,18,20

We gave those requirements the lowest priority of 1.

Those requirements will be "nice to haves" because they do not reflect a system functionality.

They are in the form of "features" that will be nice if were added to the system in a later version.

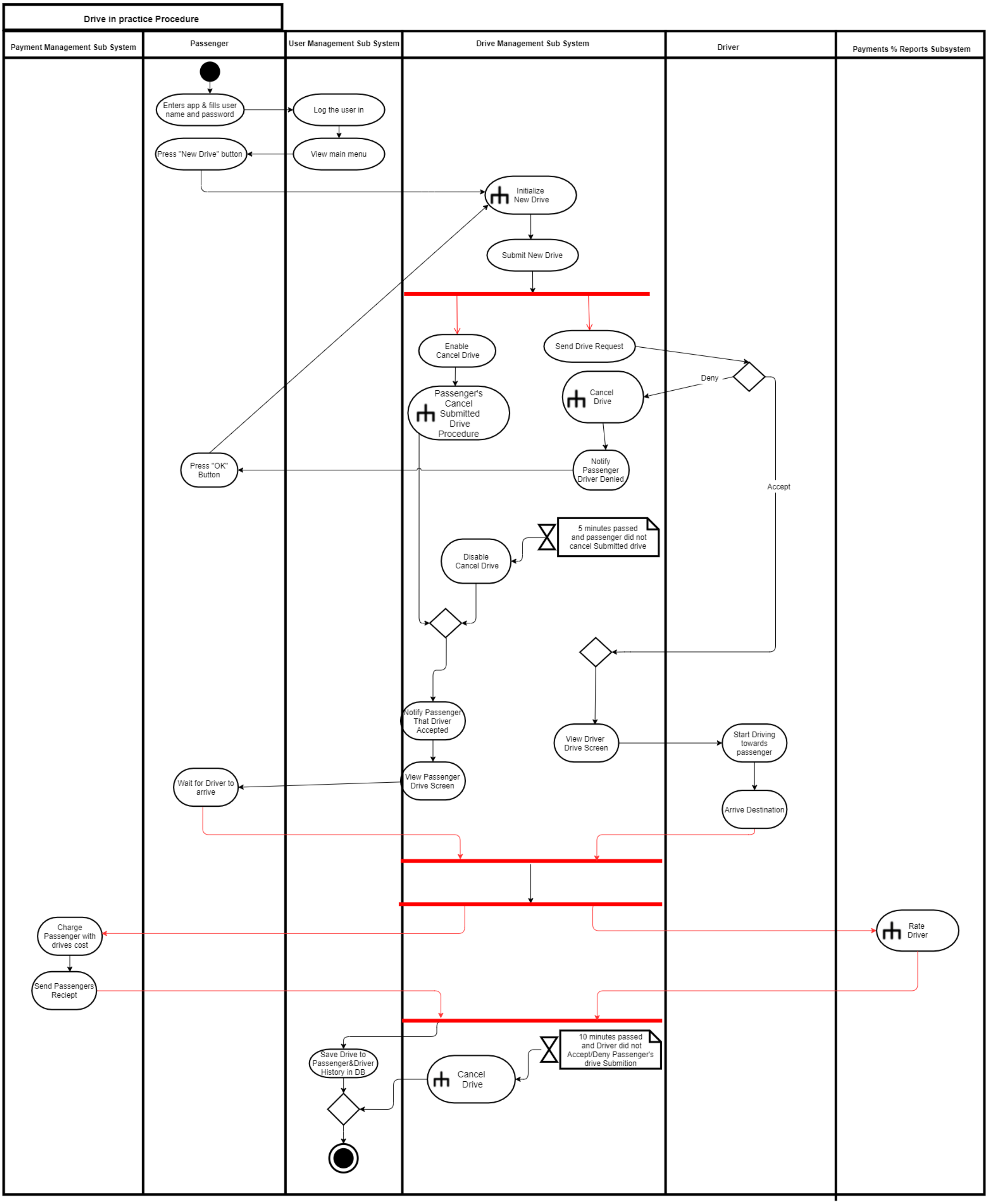
The system can work fluently without them.

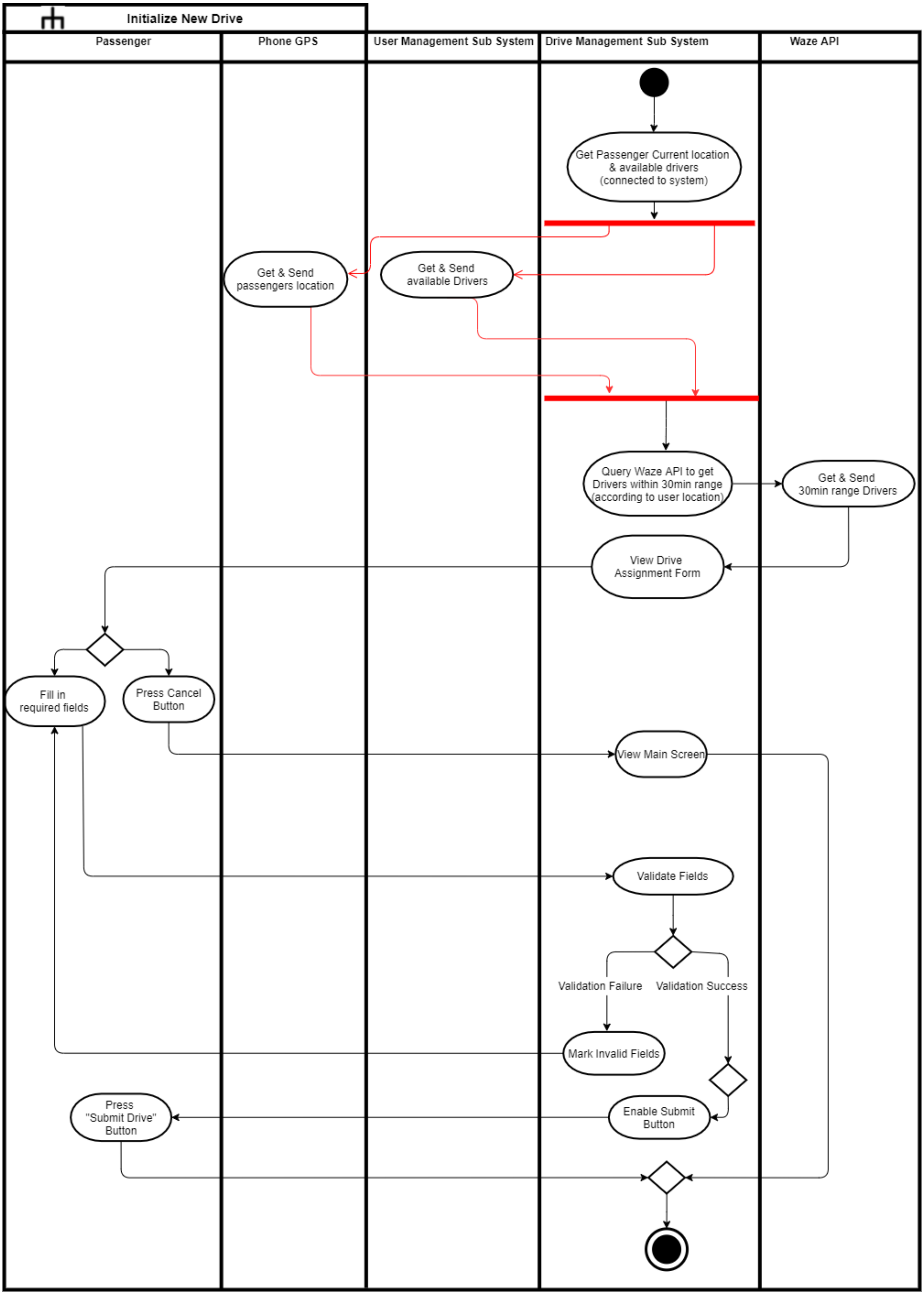
UCs:15, 16, 30, 31

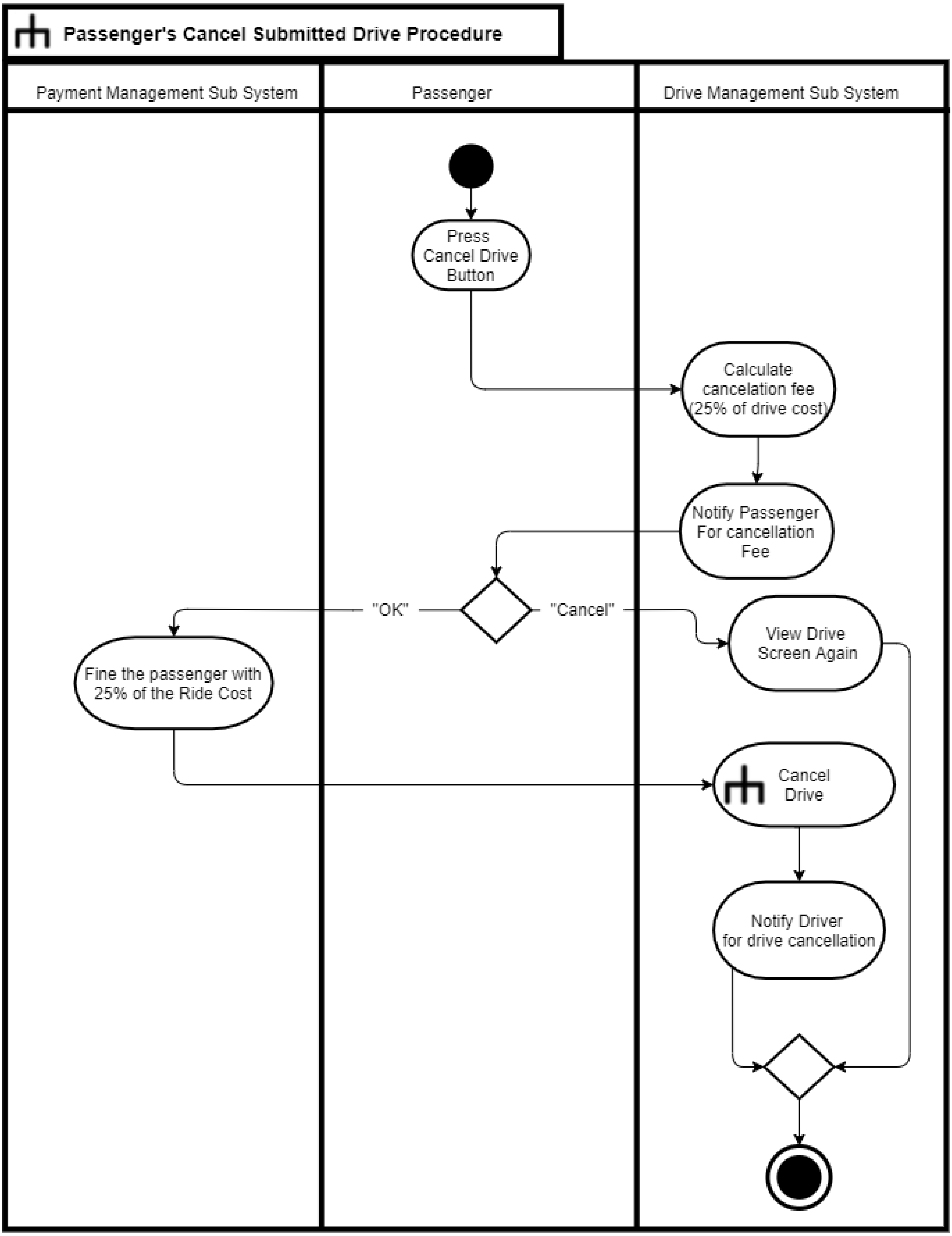
These UCs received the lowest priority and are within the bounds of "nice to have", which mean that their development will be done in a later version (if developed at all) and that the system can do well without them at all.

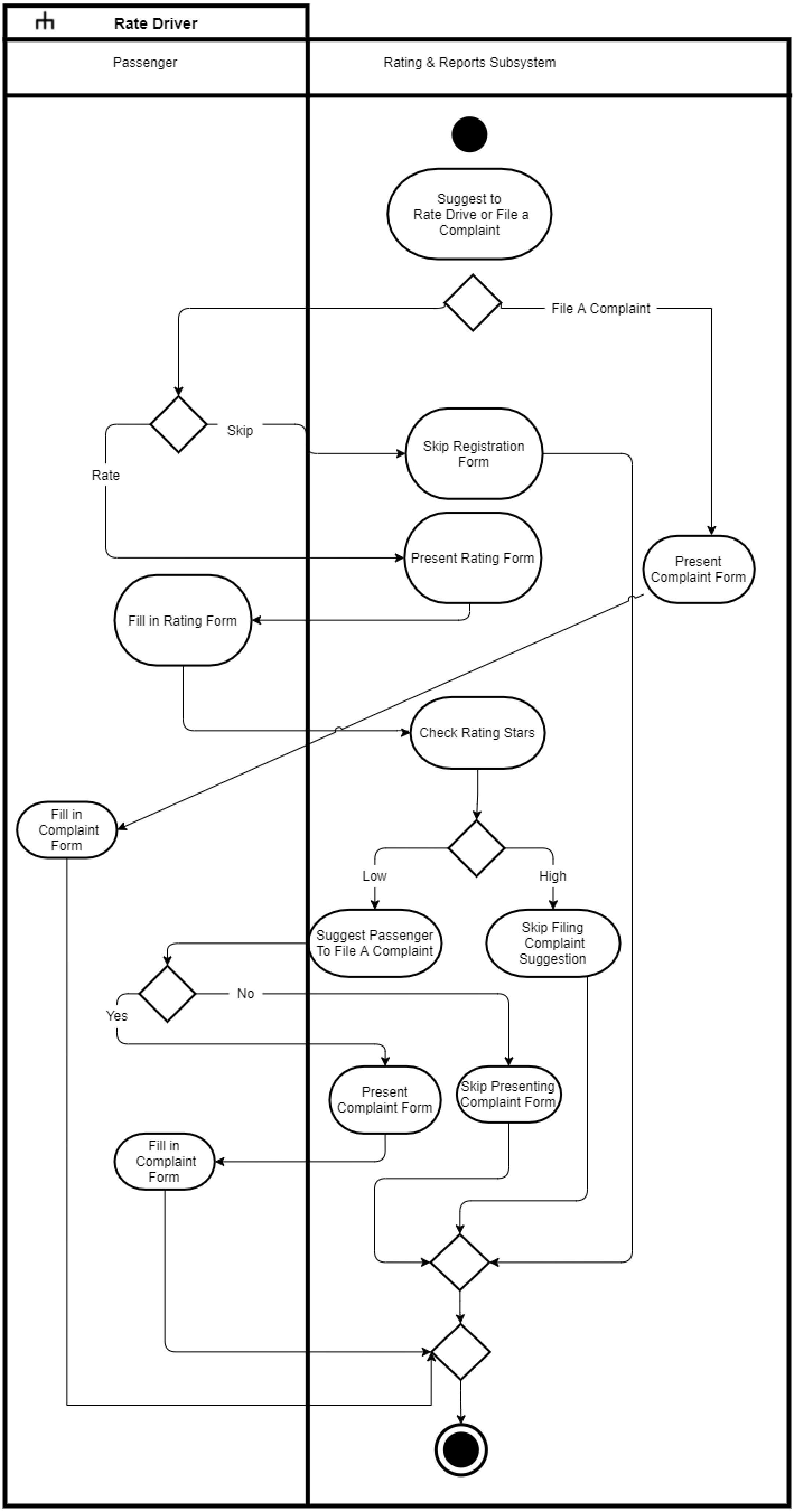
They are extensions of existing and functioning use cases and requirements.

Activity Diagrams


Drive in practice Procedure
Payment Management Sub System
Passenger
User Management Sub System
Drive Management Sub System
Driver
Payments % Reports Subsystem
Enters app & fills user name and password
Log the user in
View main menu
Press "New Drive" button
Initialize New Drive
Submit New Drive
Enable Cancel Drive
Send Drive Request
Deny
Cancel Drive
Passenger's Cancel Submitted Drive Procedure
Notify Passenger Driver Denied
Press "OK" Button
Accept
5 minutes passed and passenger did not cancel Submitted drive
Disable Cancel Drive
Notify Passenger That Driver Accepted
View Driver Drive Screen
Start Driving towards passenger
View Passenger Drive Screen
Wait for Driver to arrive
Arrive Destination
Charge Passenger with drives cost
Rate Driver
Send Passengers Reciept
10 minutes passed and Driver did not Accept/Deny Passenger's drive Submition
Save Drive to Passenger&Driver History in DB
Cancel Drive

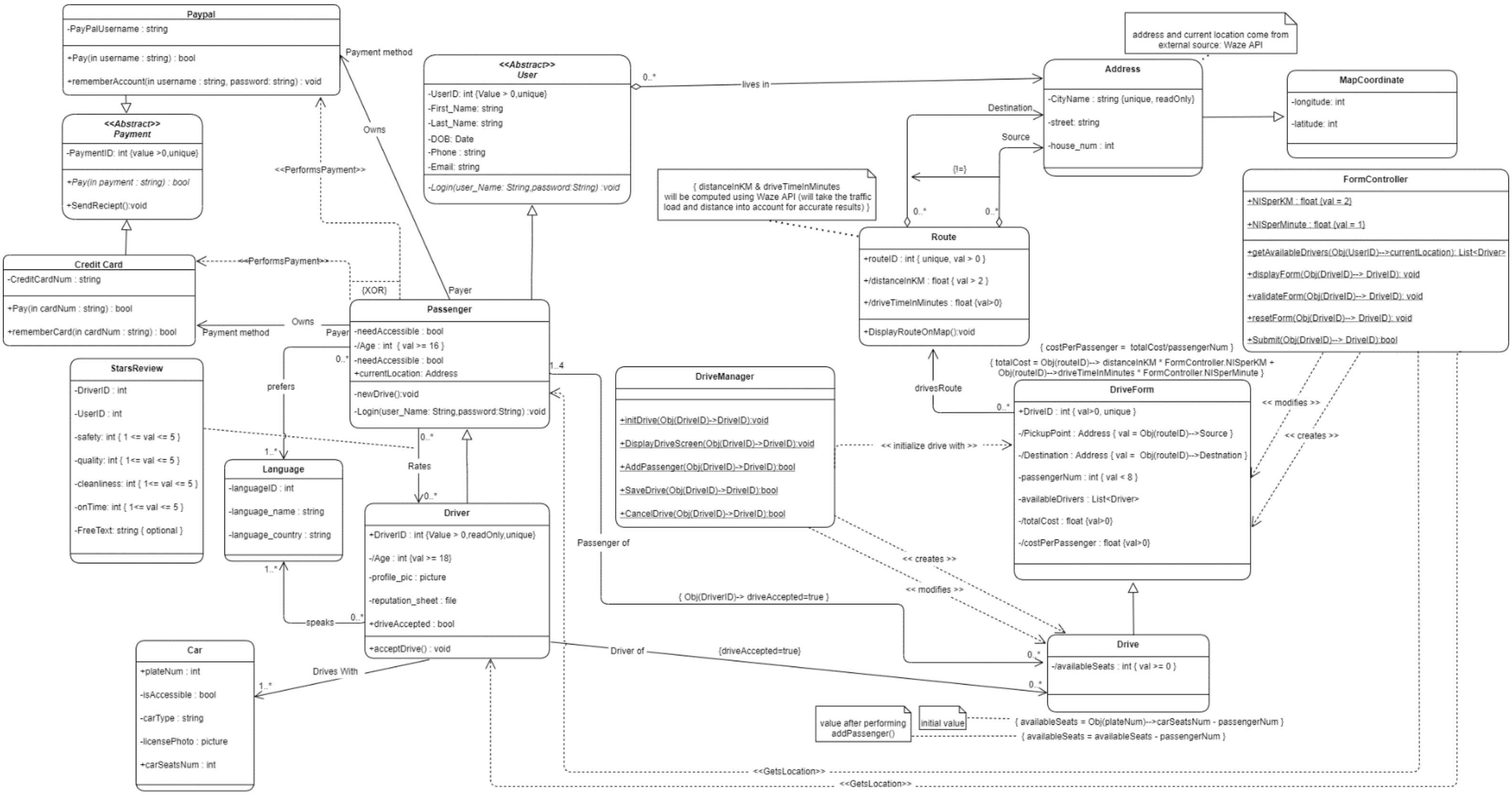

Initialize New Drive
Passenger
Phone GPS
User Management Sub System
Drive Management Sub System
Waze API
Get Passenger Current location & available drivers (connected to system)
Get & Send passengers location
Get & Send available Drivers
Query Waze API to get Drivers within 30min range (according to user location)
Get & Send 30min range Drivers
View Drive Assignment Form
Fill in required fields
Press Cancel Button
View Main Screen
Validate Fields
Validation Failure
Validation Success
Mark Invalid Fields
Press "Submit Drive" Button
Enable Submit Button


Passenger's Cancel Submitted Drive Procedure
Payment Management Sub System
Passenger
Drive Management Sub System
Press Cancel Drive Button
Calculate cancelation fee (25% of drive cost)
Notify Passenger For cancellation Fee
"OK"
"Cancel"
View Drive Screen Again
Fine the passenger with 25% of the Ride Cost
Cancel Drive
Notify Driver for drive cancellation


Rate Driver
Passenger
Rating & Reports Subsystem
Suggest to Rate Drive or File a Complaint
File A Complaint
Skip
Rate
Skip Registration Form
Present Rating Form
Present Complaint Form
Fill in Rating Form
Check Rating Stars
Fill in Complaint Form
Low
High
Suggest Passenger To File A Complaint
Skip Filing Complaint Suggestion
No
Yes
Present Complaint Form
Skip Presenting Complaint Form
Fill in Complaint Form

Chapter IV – Design

Class Diagram


Paypal
-PayPalUsername : string
+Pay(in username : string) : bool
+rememberAccount(in username : string, password: string) : void
Payment method
Owns
<<PerformsPayment>>
<<Abstract>>
Payment
-PaymentID: int {value >0,unique}
+Pay(in payment : string) : bool
+SendReciept():void
Credit Card
-CreditCardNum : string
+Pay(in cardNum : string) : bool
+rememberCard(in cardNum : string) : bool
<<PerformsPayment>>
{XOR}
Payer
Owns
Payment method
Payer
<<Abstract>>
User
-UserID: int {Value > 0,unique}
-First_Name: string
-Last_Name: string
-DOB: Date
-Phone : string
-Email: string
-Login(user_Name: String,password:String) :void
0..*
lives in
Passenger
-needAccessible : bool
-/Age : int { val >= 16 }
-needAccessible : bool
+currentLocation: Address
-newDrive():void
-Login(user_Name: String,password:String) :void
0..*
prefers
1..*
1..4
StarsReview
-DriverID : int
-UserID : int
-safety: int { 1 <= val <= 5 }
-quality: int { 1<= val <= 5 }
-cleanliness: int { 1<= val <= 5 }
-onTime: int { 1<= val <= 5 }
-FreeText: string { optional }
Language
-languageID : int
-language_name : string
-language_country : string
0..*
Rates
0..*
Driver
+DriverID : int {Value > 0,readOnly,unique}
-/Age : int {val >= 18}
-profile_pic : picture
-reputation_sheet : file
+driveAccepted : bool
+acceptDrive() : void
1..*
speaks
0..*
Car
+plateNum : int
-isAccessible : bool
-carType : string
-licensePhoto : picture
+carSeatsNum : int
Drives With
1..*
Passenger of
Driver of
{driveAccepted=true}
{ Obj(DriverID)-> driveAccepted=true }
address and current location come from external source: Waze API
Address
-CityName : string {unique, readOnly}
-street: string
-house_num : int
MapCoordinate
-longitude: int
-latitude: int
Destination
Source
{!=}
0..*
0..*
{ distanceInKM & driveTimeInMinutes will be computed using Waze API (will take the traffic load and distance into account for accurate results) }
Route
+routeID : int { unique, val > 0 }
+/distanceInKM : float { val > 2 }
+/driveTimeInMinutes : float {val>0}
+DisplayRouteOnMap():void
drivesRoute
0..*
FormController
+NISperKM : float {val = 2}
+NISperMinute : float {val = 1}
+getAvailableDrivers(Obj(UserID)-->currentLocation): List<Driver>
+displayForm(Obj(DriveID)--> DriveID): void
+validateForm(Obj(DriveID)--> DriveID): void
+resetForm(Obj(DriveID)--> DriveID): void
+Submit(Obj(DriveID)--> DriveID):bool
{ costPerPassenger = totalCost/passengerNum }
{ totalCost = Obj(routeID)--> distanceInKM * FormController.NISperKM + Obj(routeID)-->driveTimeInMinutes * FormController.NISperMinute }
DriveManager
+initDrive(Obj(DriveID)->DriveID):void
+DisplayDriveScreen(Obj(DriveID)->DriveID):void
+AddPassenger(Obj(DriveID)->DriveID):bool
+SaveDrive(Obj(DriveID)->DriveID):bool
+CancelDrive(Obj(DriveID)->DriveID):bool
<< initialize drive with >>
DriveForm
+DriveID : int { val>0, unique }
-/PickupPoint : Address { val = Obj(routeID)-->Source }
-/Destination : Address { val = Obj(routeID)-->Destnation }
-passengerNum : int { val < 8 }
-availableDrivers : List<Driver>
-/totalCost : float {val>0}
-/costPerPassenger : float {val>0}
<< modifies >>
<< creates >>
<< creates >>
<< modifies >>
Drive
-/availableSeats : int { val >= 0 }
0..*
0..*
value after performing addPassenger()
initial value
{ availableSeats = Obj(plateNum)-->carSeatsNum - passengerNum }
{ availableSeats = availableSeats - passengerNum }
<<GetsLocation>>
<<GetsLocation>>