The Eco-Drivana App

“Carpool Efficiently with Confidence”

Course: MIS-6308

Instructor: Dr. Srinivasan Raghunathan

Group #3:

- Kevin Sy
- Anamika Soni
- Amey Sudam Saste
- Monika Malik
- Sanjana Buchala

Table of Contents

EXECUTIVE SUMMARY

Currently, the Dallas-Fort Worth area has been growing rapidly as new companies and more students move into the area. In fact, Toyota, Boeing, Liberty Mutual, JP Morgan Chase, Samsung, and FedEx all have recently moved into Plano, Texas. As a result, all these companies' employees have added traffic to the already strained highways such as the Dallas Tollway North and I-635. In Dallas-Garland-Plano, the average worker commutes 55.8 minutes a day roundtrip. Additionally, the University of Dallas at Texas (UTD) has been experiencing huge amounts of growth from both domestic and international students. In fact, UTD grew to enrolled 24,532 students in 2015 (). More recently in 2019, UTD has announced plans to expand the campus to accommodate up to 35,000 students (). To keep up with pace of the huge growth of student enrollments, parking lots, roads, and housing must be expanded. Both the parking lots and roads will continue to become congested, creating both emotional and financial strain on the UTD students. If UTD and local governments do not address the issue of traffic, more residents and students will begin to turn towards other cities.

There have been several existing solutions to address the issue of increased traffic. UTD has public transportation available for students who live within 5 miles. Unfortunately, many students also live outside of the 5-mile radius, so their vehicles end up contributing to the already heavy traffic. Uber and Lyft both provide for-profit ride sharing services, but can both be expensive, especially during rush hour; students rarely have large amounts of cash to sustainably ride Uber and Lyft on a daily basis. Local governments have also expanded the number of lanes for highways and residential roads, but this is a very time consuming and costly process.

As a result of the all these existing issues and expensive solutions, the Eco-Drivana App has sought to create an affordable social media platform that aims to both reduce traffic and help UTD students save time and money on gasoline. The mission statement states, "With the Eco-Drivana App, we empower people to confidently and efficiently carpool to their destinations so we together can save money, the environment, and time on the roads." The Eco-Drivana App enables riders to search for drivers within their social network with similar schedules, routes, and destinations so they can carpool with others. The cost of each share ride is split between the driver and passengers based on the distance and average fuel cost of the day. To keep costs low, advertisement revenue and a small transaction fee will be collected to support the app's overhead costs such as the server maintenance in the cloud. The advertisements will be displayed to passengers to both provide ad revenue and provide passengers with coupons that can both help save students money and help local Dallas businesses.

PROBLEM STATEMENT

PROBLEMS:

1. The current traffic and public transportation scenario:

- On an average, a Dallas driver is delayed by more than 60 hours per year. Traffic can be fierce during rush hours, especially from 6:30 am to 9 am and 4 pm to 6:30 pm.
- Heavy traffic can create both financial and physical strain on students.
- It has been estimated that more than 42 percent of households in Dallas are underserved by public transit leaving students with minimal and expensive options.

2. Uber, Lyft, and Van Pool are uneconomical options for students.

- Cab Ride Apps are extremely expensive during rush hours, and not a viable option if the students live far away from work
- Students moving closer to work doesn't solve the problem since commuting to the university for morning or evening classes will result in the same issue at hand.
- Van Pool is only economical if the Van Pool has at least 10 people riding inside. Also, riding with complete strangers can be a huge risk for passengers.

3. The UT Dallas Parking lot is saturated with cars from new incoming students.

- Parking spots are designated according to color of pass which is available for purchase and if all the spots are occupied then the student will have to find alternative options
- When students spend ample amount of time finding parking spaces, it will result in them reaching late to class with a frustrated mind.
- The customer satisfaction and university rating will suffer.
- Tuition and parking permits prices will rise to record levels if new parking garages must be constructed to accommodate growing number of students.

OBJECTIVES:

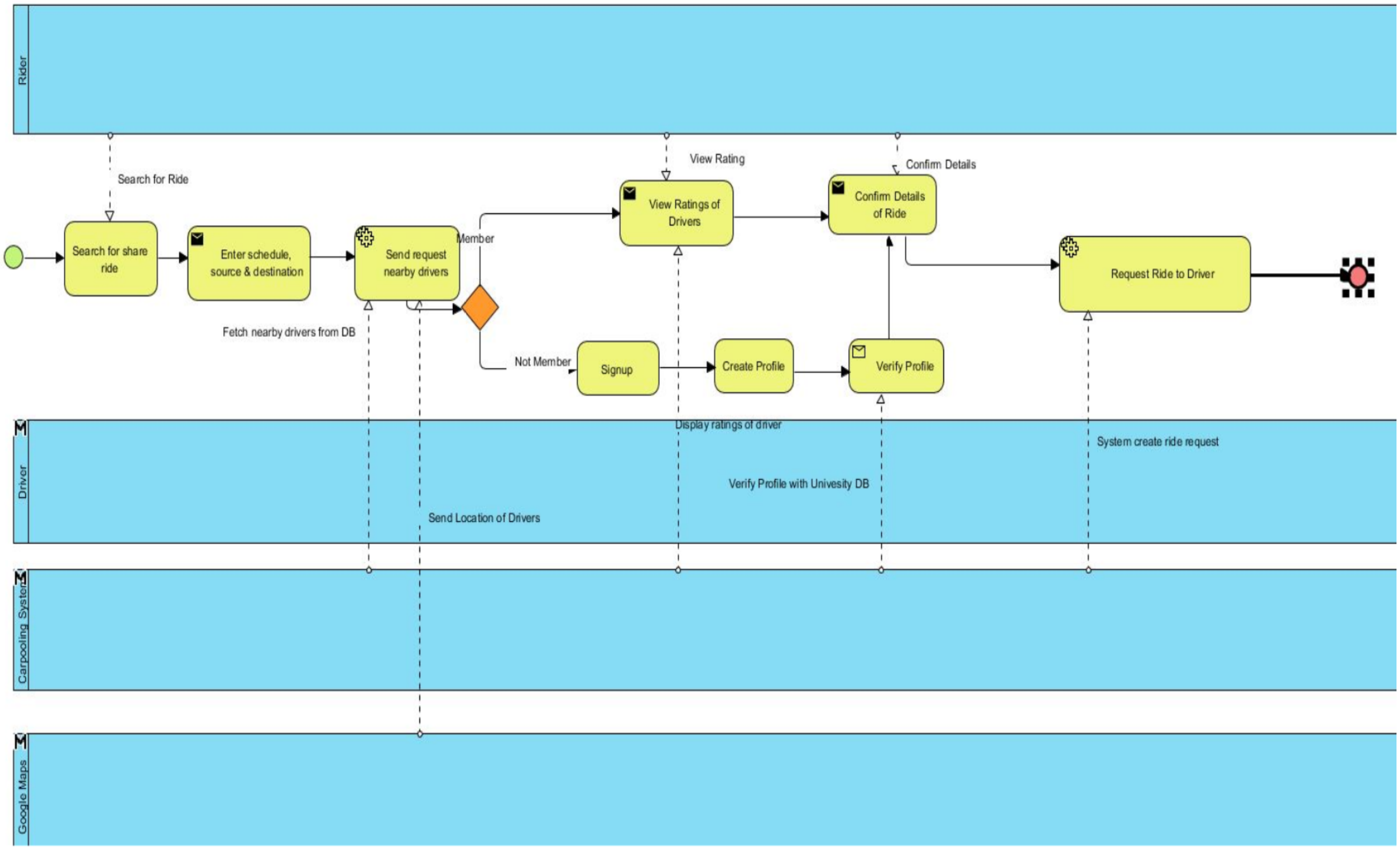
1. Create an economical app that provides affordable ride sharing service for drivers and passengers
2. New system provides services like group creation for users travelling similar routes frequently
3. Provide businesses with an additional platform to market their products or services to passengers
4. Reduce the number of vehicles on the road to reduce greenhouse gases and road traffic

SCOPE:

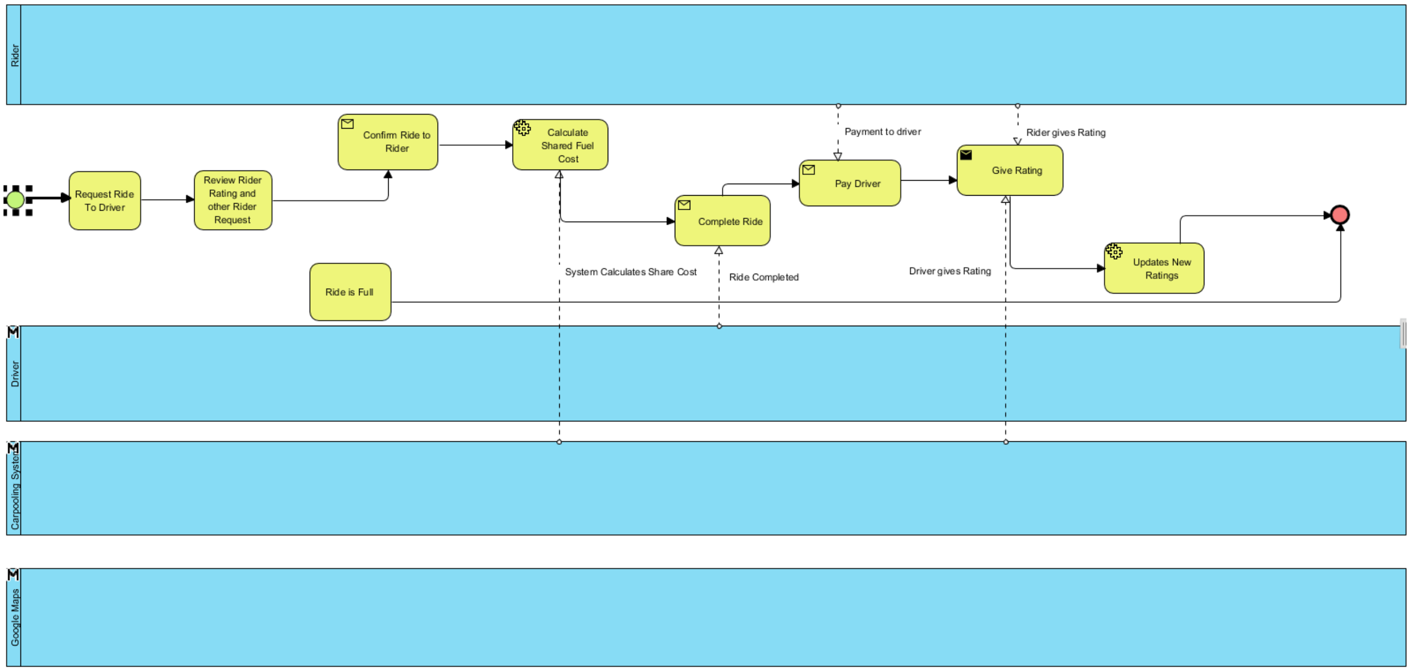
1. An estimated cost of $500,000 will be needed to support the entire operation.
2. The Eco-Drivana App will be limited to the scope of UT Dallas students and workers first as a proof of concept before expanding to other businesses and locations.
3. The business operations and development will require expertise related to negotiating offers, discounts, and advertisements with local vendors.
4. The technology expertise will require knowledge of Android Development (in Java), iOS Development (in Swift), Application Development, Google Maps API, Gas Buddy API, Payment Gateway API, Database Management, Cloud Deployments, and Java.
5. The system will be equipped with database servers for storing shared fuel cost calculations, user profiles, rides and rating history.

CHOREOGRAPHY DIAGRAM:

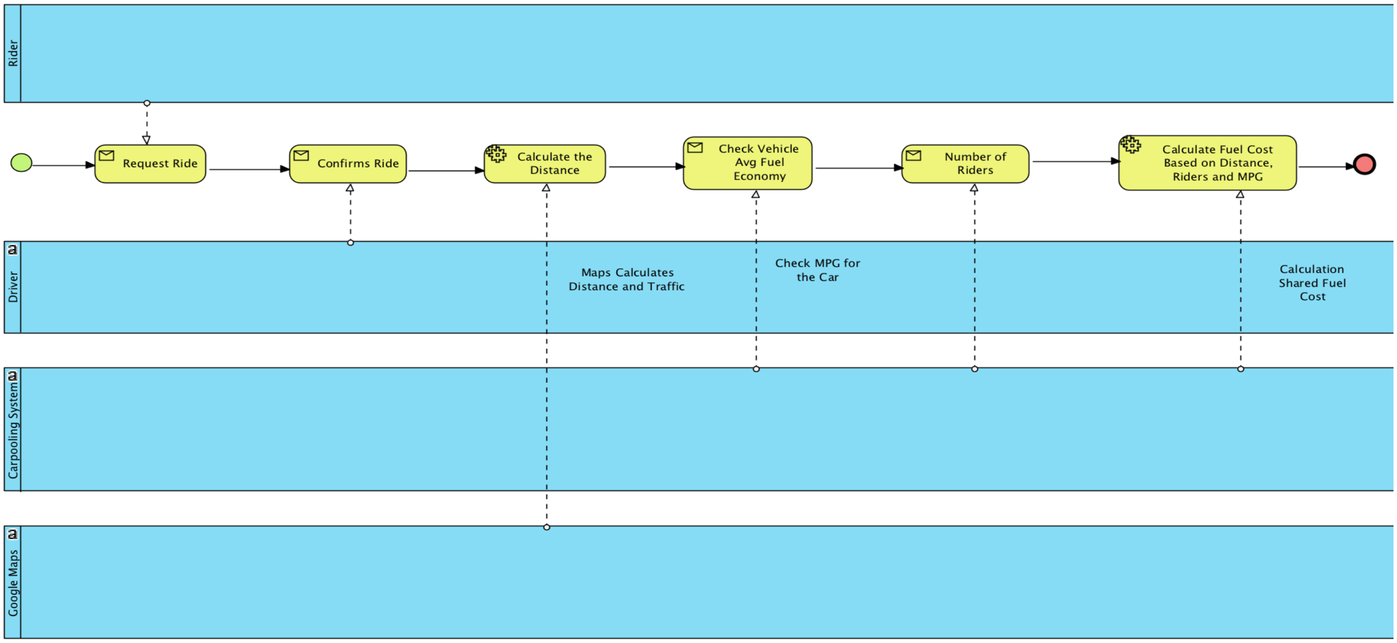
***Choreography Diagram**: Carpooling System*



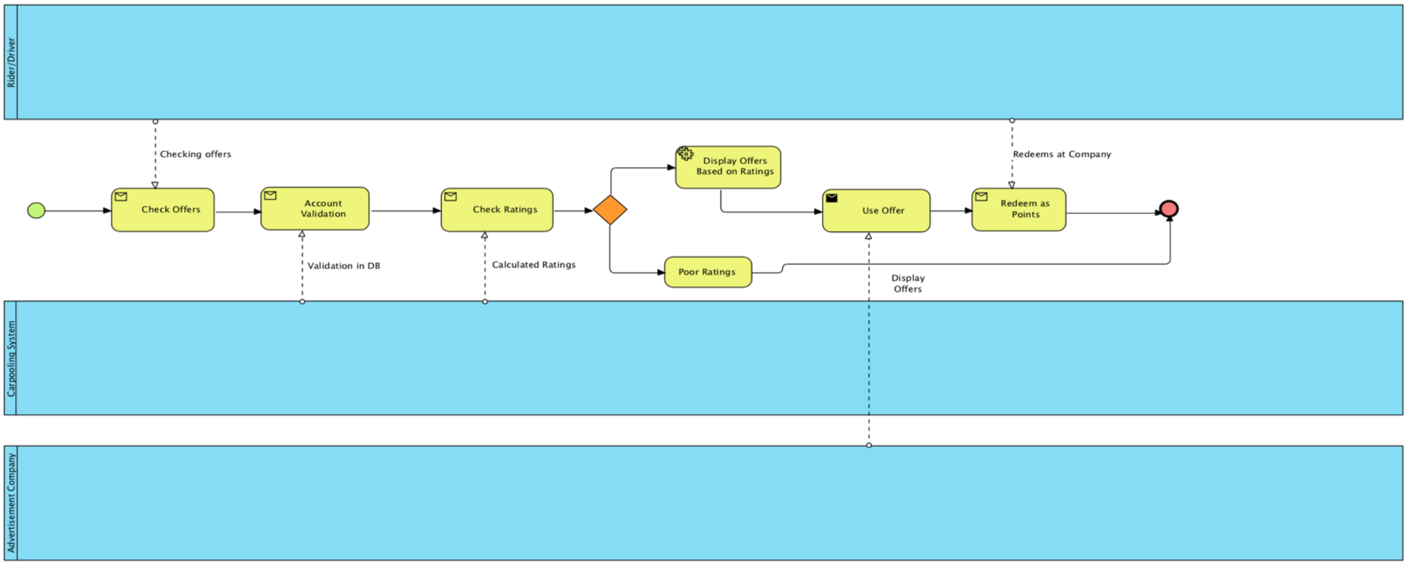
***Choreography Diagram**: Carpooling System (continued)*



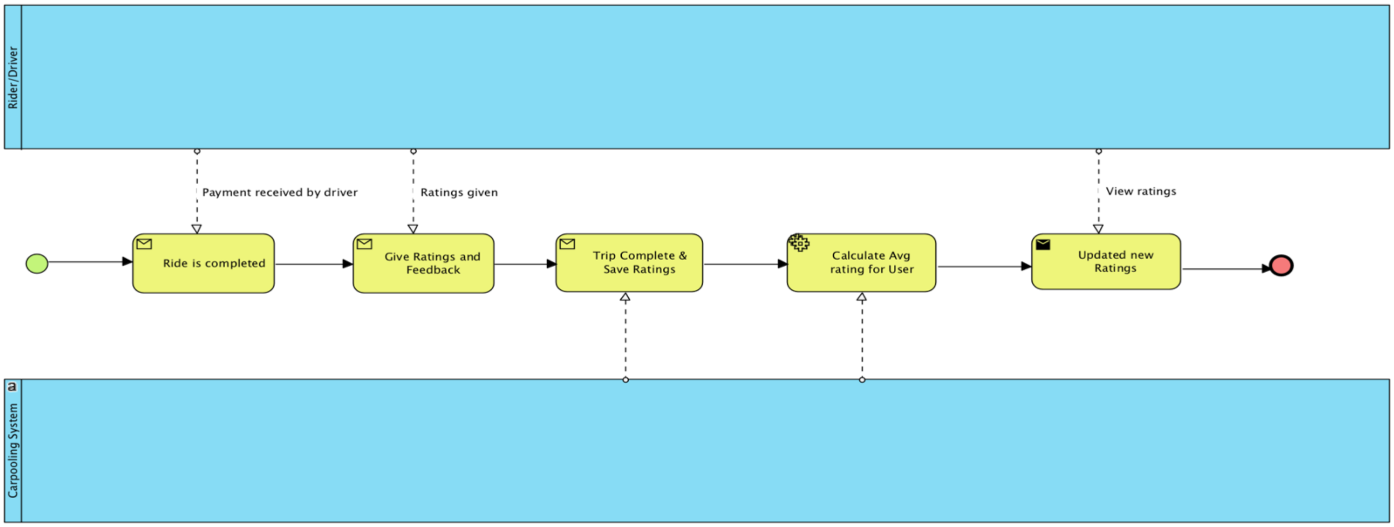
***Choreography Diagram**: Shared Fuel Costs*


Rider
Request Ride
Confirms Ride
Calculate the Distance
Check Vehicle Avg Fuel Economy
Number of Riders
Calculate Fuel Cost Based on Distance, Riders and MPG
Driver
Maps Calculates Distance and Traffic
Check MPG for the Car
Calculation Shared Fuel Cost
Carpooling System
Google Maps

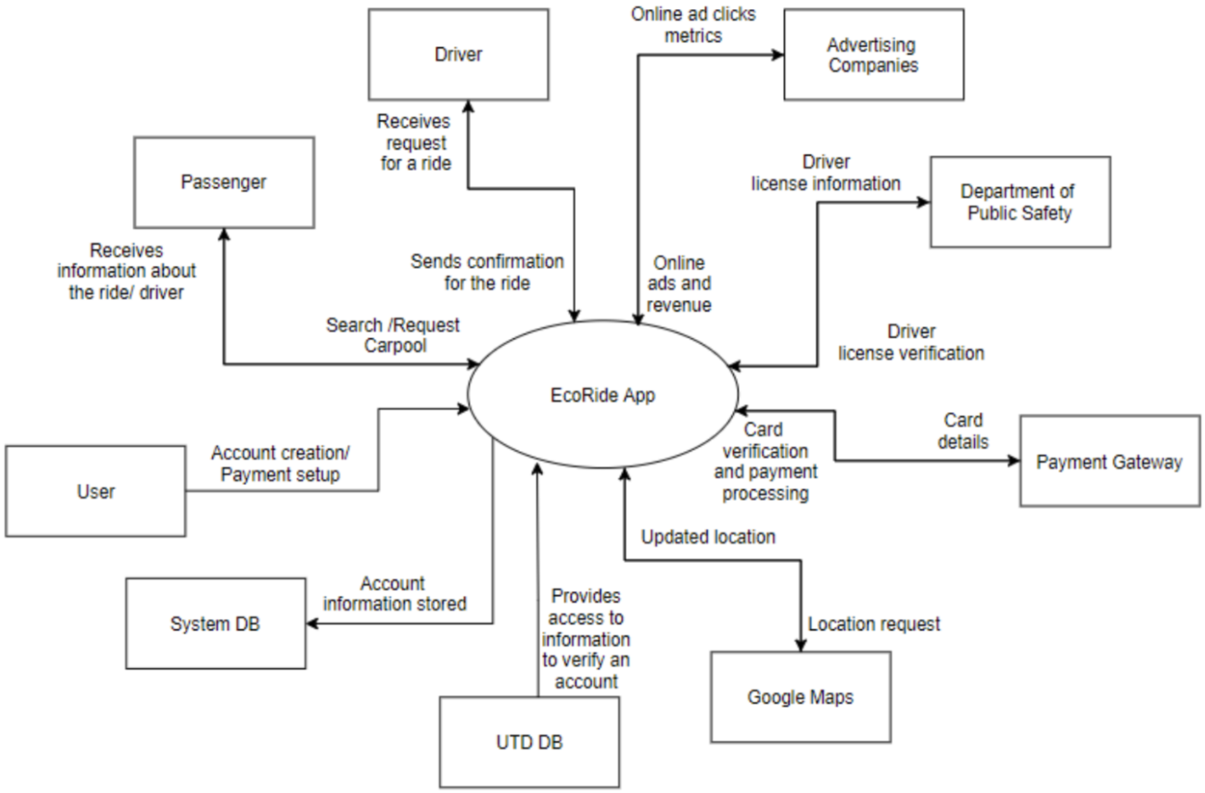
***Choreography Diagram**: Check Offers and Coupons*



***Choreography Diagram**: Calculate Ratings*

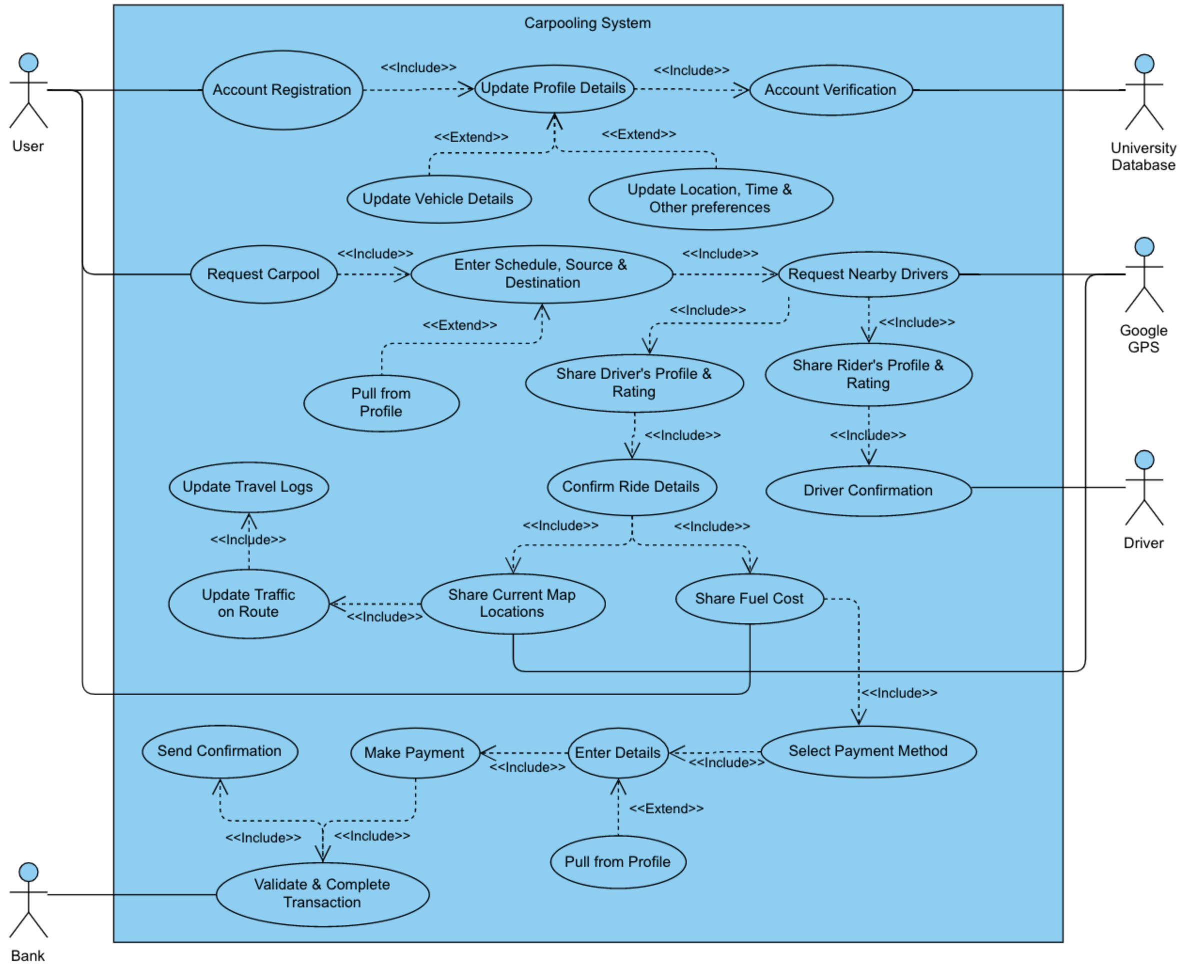

Rider/Driver
Payment received by driver
Ratings given
View ratings
Ride is completed
Give Ratings and Feedback
Trip Complete & Save Ratings
Calculate Avg rating for User
Updated new Ratings
Carpooling System

CONTEXT DIAGRAM:

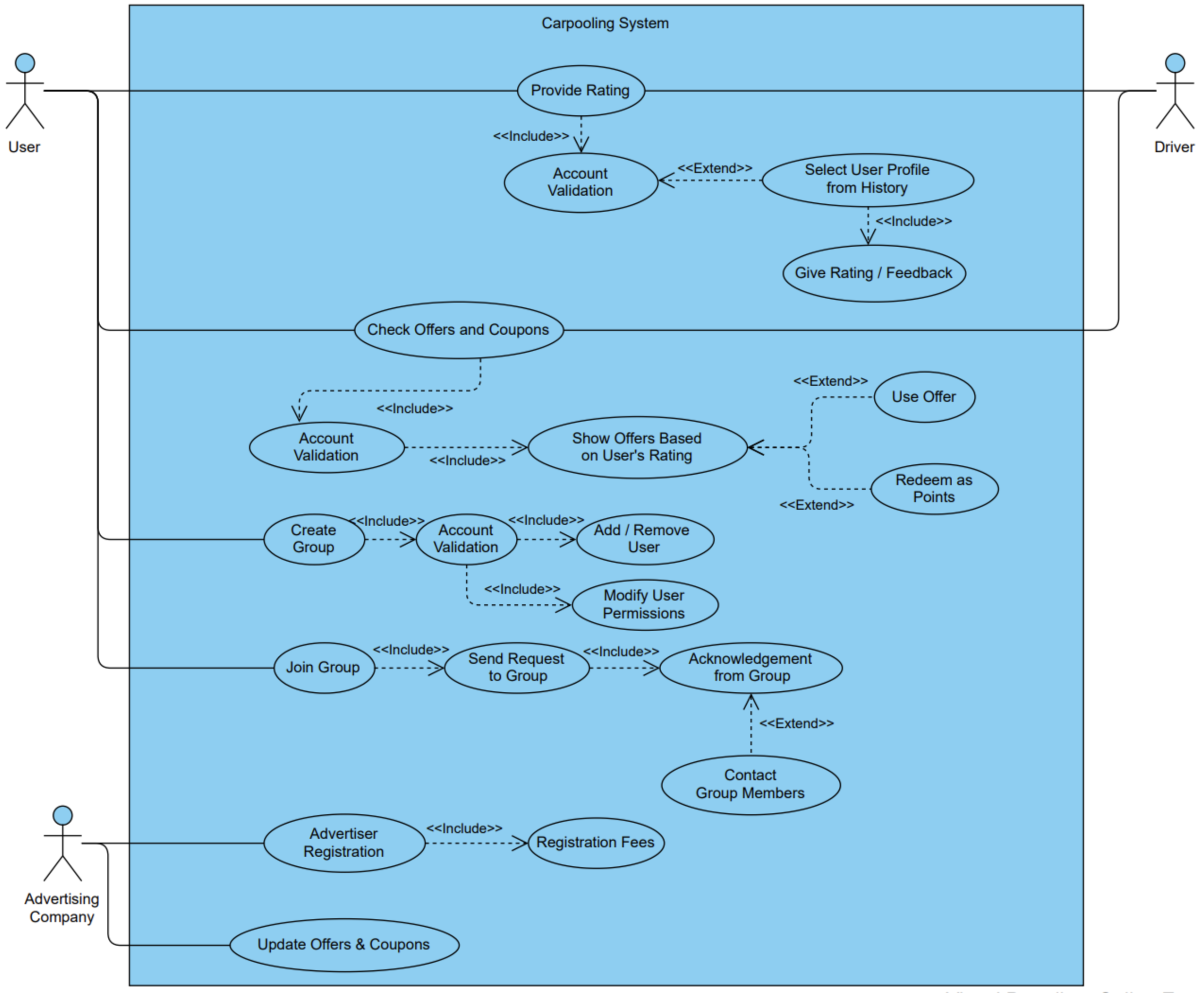

Driver
Online ad clicks metrics
Advertising Companies
Receives request for a ride
Passenger
Driver license information
Department of Public Safety
Receives information about the ride/ driver
Sends confirmation for the ride
Online ads and revenue
Search /Request Carpool
Driver license verification
EcoRide App
Card verification and payment processing
Card details
Payment Gateway
User
Account creation/ Payment setup
Updated location
Account information stored
System DB
Provides access to information to verify an account
Location request
Google Maps
UTD DB

USE CASE DIAGRAM:

Use Case Diagram:



Use Case Diagram (Continued):



Note: The University Database is referring to the UTD DB since the app will be tested at UTD

USE CASE DESCRIPTIONS:

Use Case Description 1:

Use Case Name: Account Registration

Primary Actor: Driver, Rider

Stakeholders: Carpooling System, University Database, System DB

Brief Description: When user wants to sign up as driver and/or rider

Trigger: When user clicks on Sign Up Button

Normal flow of events:

1. The User navigates through Carpooling Website or App
2. The User inputs personal information First Name, Last Name, Email (a UTDallas email), Date of Birth, Phone Number, Driver's License, Driver's License Expiration Date, and Fingerprint Code.
3. The Carpooling System then verifies with University Database.
4. The User clicks on "Sign Up" or "Create Your Free Account" button on main screen.
5. An Account Creation Confirmation Email is sent to the user requesting for the user to confirm his or her email.
6. The User then confirms with the email with the random generated code provided.
7. The User creates profile or search for rides or riders.

Exception:

1. If the User enters invalid details, then display "Sign-up failed".
2. If the User enters invalid university details, then display "Invalid Verification".

Use Case Description 2:

Use Case Name: Update Profile Details

Primary Actor: Driver, Rider, University DB, System DB

Stakeholders: Carpooling System

Brief Description: When user wants to create or update profile

Trigger: When user clicks on Create or Update Profile

Normal flow of events:

1. The User navigates through website or mobile app to Carpooling System Website or App
2. The User signs in with Username and Password or with a Fingerprint authentication
3. The User creates a User Profile as a rider or driver. For a rider profile, he or she updates the Favorite Schedule Routes. For a driver profile, he or she updates Vehicle Details, favorite route and time.
4. The User creates or updates Payment Card Details to receive and send money. Credit card or PayPal information.
5. If the User profile, Vehicle Details, and Payment Card Details information is already there, user can update the information.

Exception:

1. If user enters invalid User details, then display "Profile info not updated".
2. If user enters invalid Payment Card Details, then display "Invalid Payment Method".

Use Case Description 3:

Use Case Name: Request Carpool

Primary Actor: Driver, Rider, System DB, Google Maps

Stakeholders: Carpooling System

Brief Description: Rider is looking for share ride in carpooling app

Trigger: When user clicks on Request Ride button in carpooling app

Normal flow of events:

1. The Rider enters his/her Departure Time, Arrival Time, Start Location, and Destination Location to search for rides.
2. The carpooling system will fetch all drivers from the System DB based on Start Location and filters nearby drivers using Google Maps.
3. The carpooling system shares Drivers' Ratings and time to pick up the Rider.
4. The Rider confirms one of the drivers by clicking on "Confirm Driver" button.
5. The Driver receives rider's User information, Ratings, and ride details.
6. The Driver confirms ride by clicking on button "Confirm Ride".
7. The carpooling system calculates the Total Shared Fuel Cost for the shared fuel, and then updates the travel logs for auditing requirements.

Exception:

1. If there are no rides available in that timing, locations, then display "No rides available, please check later".
2. If the driver rejects the request for the ride, then display "The driver cancelled your ride".

Use Case Description 4:

Use Case Name: Share Fuel Cost

Primary Actor: Driver, Rider

Stakeholders: Carpooling System, Payment Gateway

Brief Description: When rider completes the ride and rider shares fuel cost with driver

Trigger: When driver and rider click on "Complete Ride"

Normal flow of events:

1. The Rider checks system calculated Total Shared Fuel Cost for shared fuel for completed ride
2. The Rider selects the available payment option to pay the driver using the selected Payment Card Details
3. The Rider selects the saved payment option or creates a new payment option.
4. The Rider makes the payment and driver click on "Confirm Payment". In the background, the Bank authorizes the payment transaction for the Total Shared Fuel Cost

Exception:

1. If rider enters invalid Payment Card Details, then display "Invalid Payment Method".

Use Case Description 5:

Use Case Name: Provide Rating
Primary Actor: Driver, Rider, System DB
Stakeholders: Carpooling System
Brief Description: Rider and driver give ratings and feedback after the payment process.
Trigger: After confirmation of payment, rider and driver gives the rating in "Rating/Feedback"
Normal flow of events: 1. The Rider or Driver checks the completed ride in ride history (also called the Payment Transaction History). 2. The Rider or Driver clicks on "Give Rating/Feedback" button. 3. The Rider or Driver gives a Rating out of 5 stars and writes a brief Feedback about the Rider. 4. The Rider and Driver can view the Feedback and Rating in view Ratings.
Exception: 1. If user writes feedback more than 500 characters, then display, "Please write brief feedback."

Use Case Description 6:

Use Case Name: Check Offers and Coupons
Primary Actor: Driver, Rider, System DB, Google Maps
Stakeholders: Carpooling System
Brief Description: When the Rider or Driver is checking for offers and coupons
Trigger: When user clicks on "Check Offers" in carpooling system
Normal flow of events: 1. When User clicks on "Check Offers", system validates the account of end user. 2. System shows Offers based on user Ratings, Location, and number of rides completed by the User. 3. The User can check the relevant Offers. 4. The User can redeem offers as points or rewards and it can be viewed in his/her account.
Exception: 1. If there are no offers available for user, then display "No offers available, please check later"

Use Case Description 7:

Use Case Name: Vendor Registration

Primary Actor: Advertising Company, Bank

Stakeholders: Carpooling System

Brief Description: A company is required register to display advertisements to passenger in the carpooling system

Trigger: When company clicks on “Register for Ads”

Normal flow of events:

1. Company navigates through Carpooling Website or App
2. The Vendor User inputs personal information Username, Password, First Name, Last Name, Email (any vendor email), Date of Birth, Phone Number, Driver’s License, Driver’s License Expiration Date, and Fingerprint Code.
3. The Vendor User inputs company information such as Vendor Name, Street Address, Building Number, City, State, Country, Email, and Phone Number.
4. The Vendor User updates the Payment Card Details under the payment method option so that when a user views an advertisement, Eco-Drivana get paid for Ad-Click revenue.
5. The company initiates the Registration Fee for Advertisement
6. The company rep clicks on “Sign Up” button on main screen, and a Bank completes the transaction. The Vendor is now registered and can now see advertisements in carpooling system and register new advertisements

Exception:

1. If the company enters invalid Payment Card Details or Vendor Lookup information, then display “Invalid Payment Method or Company Info.”

Use Case Description 8:

Use Case Name: Create Group

Primary Actor: Rider, Driver, System DB

Stakeholders: Carpooling System

Brief Description: When users want to create for certain routes with common riders and drivers for convenient communication

Trigger: When users click on “Create Group”

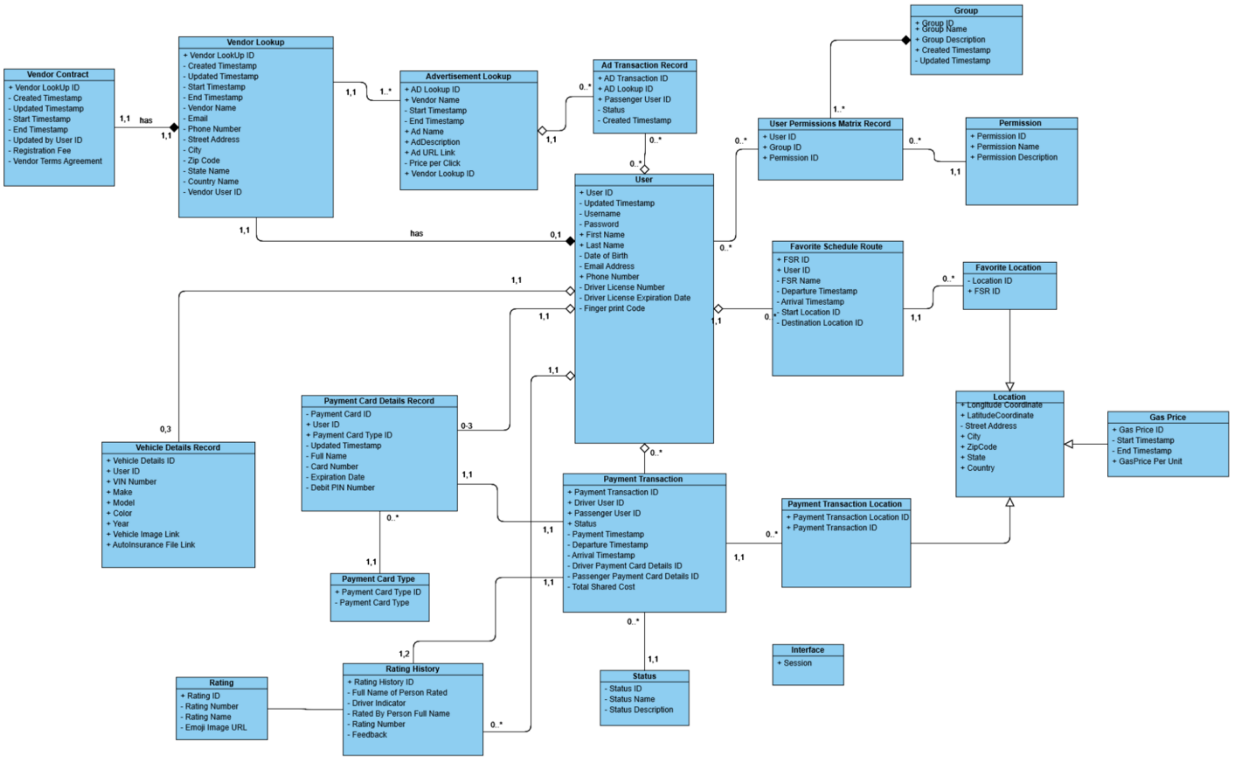
Normal flow of events:

1. A Rider or Driver clicks on Create Group for common Users, and the system validates the account.
2. A Group Admin who has created the Group can add or remove group members based on user request.
3. The Group Admin add or remove Permissions for Users.

Exception:

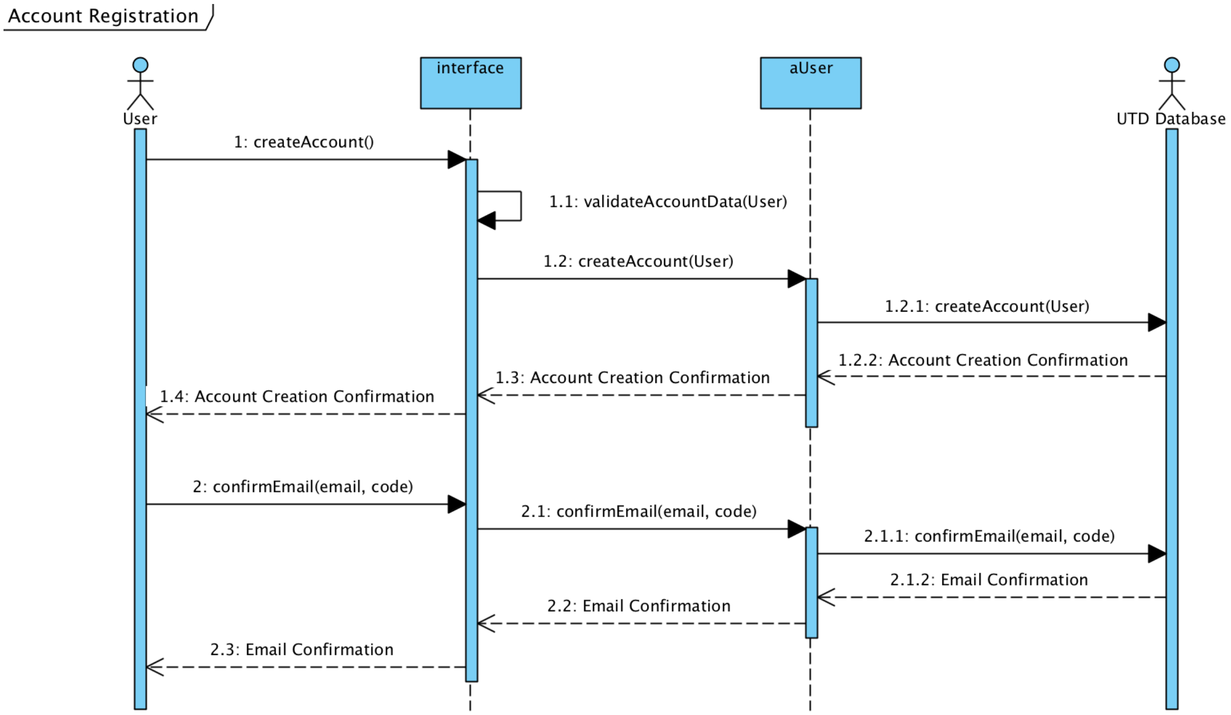
1. If the User attempts to create a Group that already exists under the same name, then display a warning message that states, “A Group with that name already exists. Please choose a different name for your Group.”
2. If the Group Admin attempts to add a member that already exists within the Group, then display a warning message that states, “The User <Name> is already a member of the Group <Group Name>.”

CLASS DIAGRAM WITHOUT METHODS (DATA MODEL)

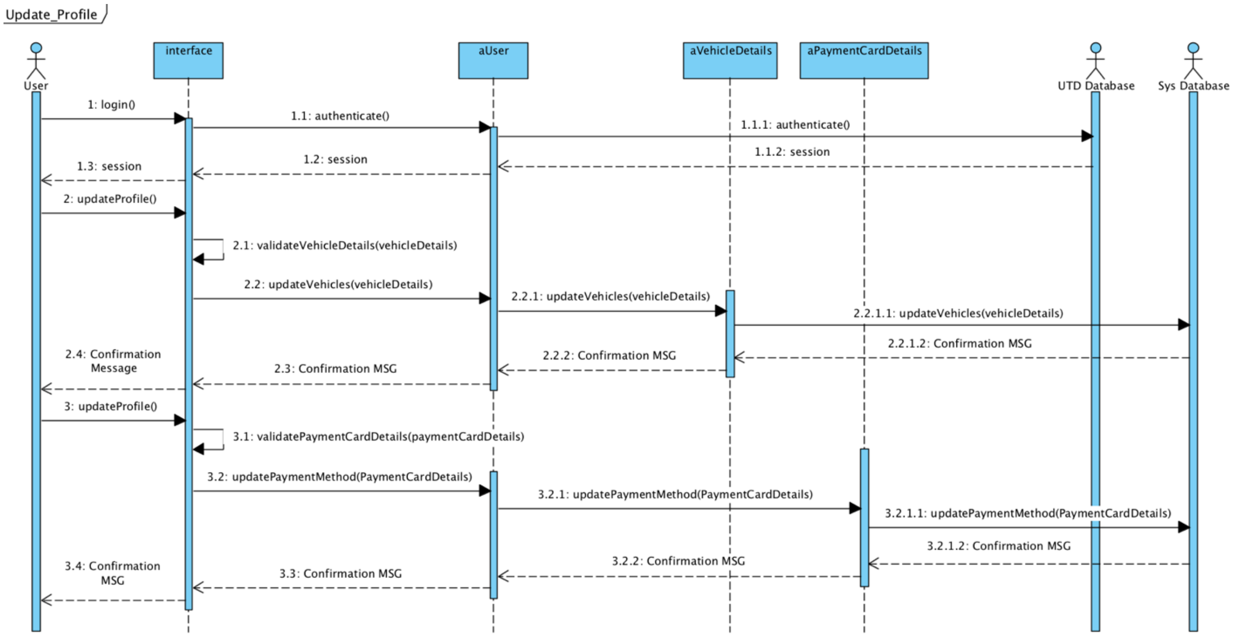


SEQUENCE DIAGRAM

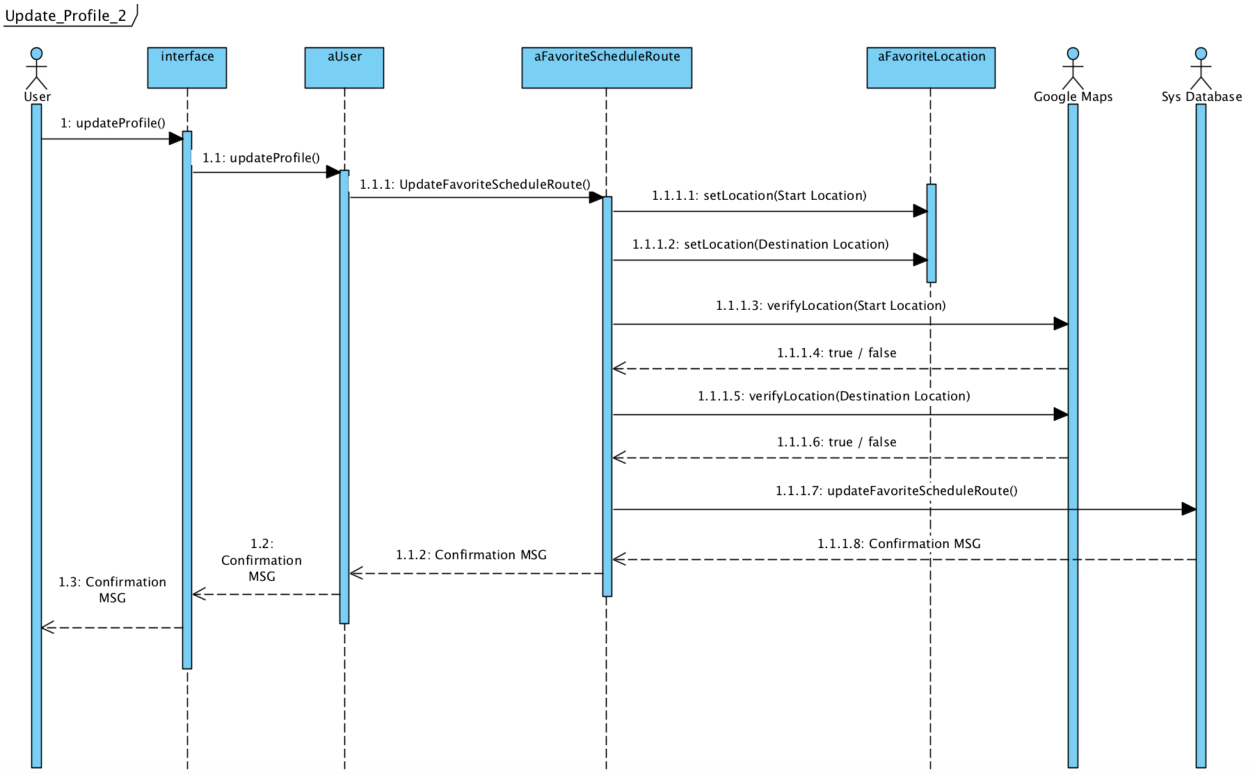
***Sequence Diagram:** Account Registration*


Account Registration
interface
aUser
User
UTD Database
1: createAccount()
1.1: validateAccountData(User)
1.2: createAccount(User)
1.2.1: createAccount(User)
1.2.2: Account Creation Confirmation
1.3: Account Creation Confirmation
1.4: Account Creation Confirmation
2: confirmEmail(email, code)
2.1: confirmEmail(email, code)
2.1.1: confirmEmail(email, code)
2.1.2: Email Confirmation
2.2: Email Confirmation
2.3: Email Confirmation

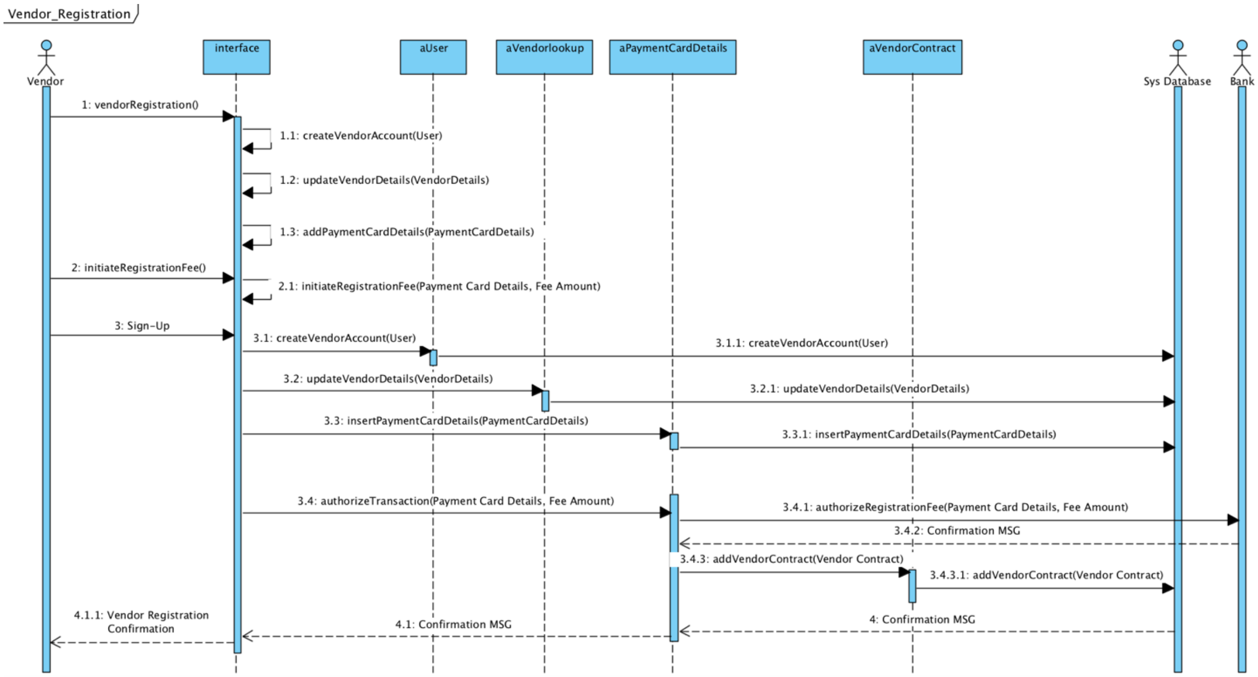
***Sequence Diagram:** Update Profile (for Vehicle Details and Payment Card Details)*


Update_Profile
User
interface
aUser
aVehicleDetails
aPaymentCardDetails
UTD Database
Sys Database
1: login()
1.1: authenticate()
1.1.1: authenticate()
1.1.2: session
1.2: session
1.3: session
2: updateProfile()
2.1: validateVehicleDetails(vehicleDetails)
2.2: updateVehicles(vehicleDetails)
2.2.1: updateVehicles(vehicleDetails)
2.2.1.1: updateVehicles(vehicleDetails)
2.2.1.2: Confirmation MSG
2.2.2: Confirmation MSG
2.3: Confirmation MSG
2.4: Confirmation Message
3: updateProfile()
3.1: validatePaymentCardDetails(paymentCardDetails)
3.2: updatePaymentMethod(PaymentCardDetails)
3.2.1: updatePaymentMethod(PaymentCardDetails)
3.2.1.1: updatePaymentMethod(PaymentCardDetails)
3.2.1.2: Confirmation MSG
3.2.2: Confirmation MSG
3.3: Confirmation MSG
3.4: Confirmation MSG

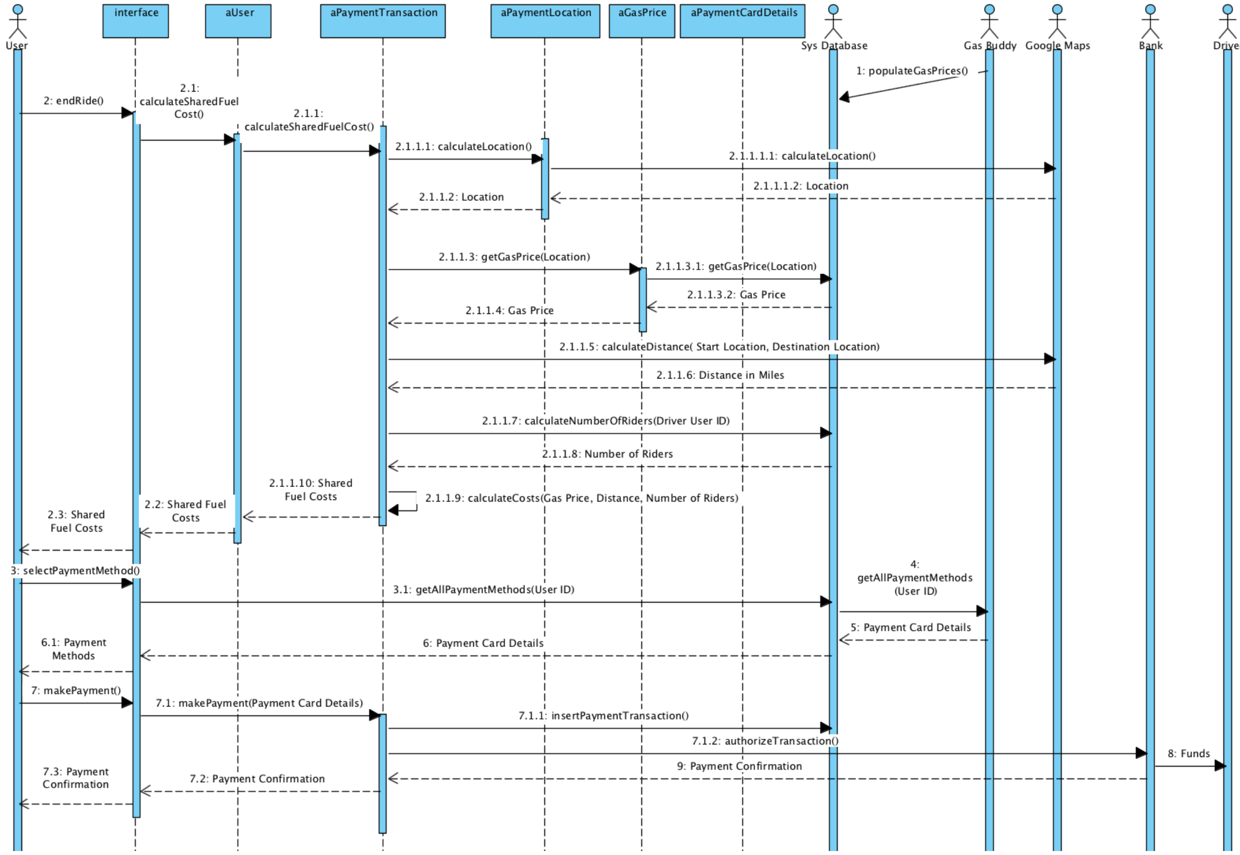
Sequence Diagram: *Update Profile (for Favorite Schedule Routes)*


Update_Profile_2
User
interface
aUser
aFavoriteScheduleRoute
aFavoriteLocation
Google Maps
Sys Database
1: updateProfile()
1.1: updateProfile()
1.1.1: UpdateFavoriteScheduleRoute()
1.1.1.1: setLocation(Start Location)
1.1.1.2: setLocation(Destination Location)
1.1.1.3: verifyLocation(Start Location)
1.1.1.4: true / false
1.1.1.5: verifyLocation(Destination Location)
1.1.1.6: true / false
1.1.1.7: updateFavoriteScheduleRoute()
1.1.1.8: Confirmation MSG
1.1.2: Confirmation MSG
1.2: Confirmation MSG
1.3: Confirmation MSG

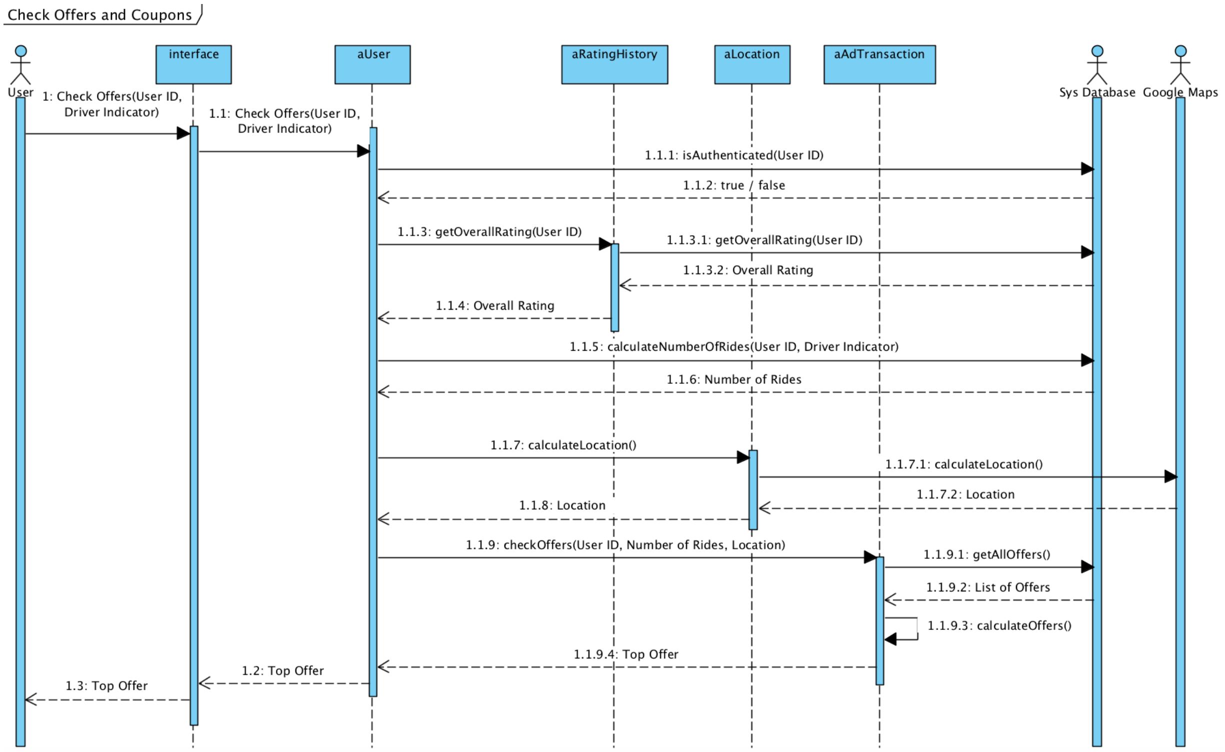
Sequence Diagram: *Vendor Registration*


Vendor_Registration
Vendor
interface
aUser
aVendorlookup
aPaymentCardDetails
aVendorContract
Sys Database
Bank
1: vendorRegistration()
1.1: createVendorAccount(User)
1.2: updateVendorDetails(VendorDetails)
1.3: addPaymentCardDetails(PaymentCardDetails)
2: initiateRegistrationFee()
2.1: initiateRegistrationFee(Payment Card Details, Fee Amount)
3: Sign-Up
3.1: createVendorAccount(User)
3.1.1: createVendorAccount(User)
3.2: updateVendorDetails(VendorDetails)
3.2.1: updateVendorDetails(VendorDetails)
3.3: insertPaymentCardDetails(PaymentCardDetails)
3.3.1: insertPaymentCardDetails(PaymentCardDetails)
3.4: authorizeTransaction(Payment Card Details, Fee Amount)
3.4.1: authorizeRegistrationFee(Payment Card Details, Fee Amount)
3.4.2: Confirmation MSG
3.4.3: addVendorContract(Vendor Contract)
3.4.3.1: addVendorContract(Vendor Contract)
4: Confirmation MSG
4.1: Confirmation MSG
4.1.1: Vendor Registration Confirmation

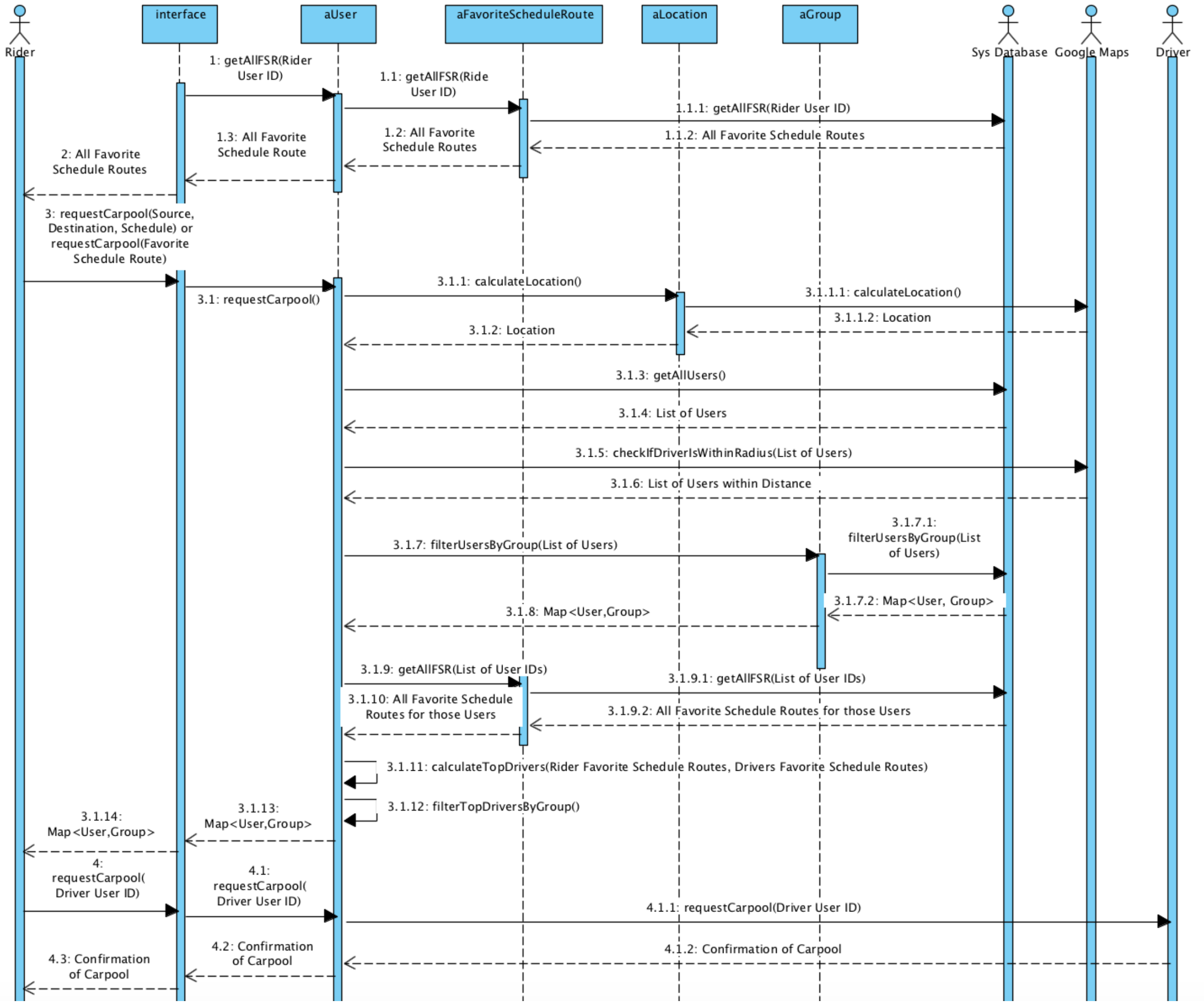
Sequence Diagram: *Share Fuel Costs*


User
interface
aUser
aPaymentTransaction
aPaymentLocation
aGasPrice
aPaymentCardDetails
Sys Database
Gas Buddy
Google Maps
Bank
Driver
1: populateGasPrices()
2: endRide()
2.1: calculateSharedFuelCost()
2.1.1: calculateSharedFuelCost()
2.1.1.1: calculateLocation()
2.1.1.1.1: calculateLocation()
2.1.1.1.2: Location
2.1.1.2: Location
2.1.1.3: getGasPrice(Location)
2.1.1.3.1: getGasPrice(Location)
2.1.1.3.2: Gas Price
2.1.1.4: Gas Price
2.1.1.5: calculateDistance(Start Location, Destination Location)
2.1.1.6: Distance in Miles
2.1.1.7: calculateNumberOfRiders(Driver User ID)
2.1.1.8: Number of Riders
2.1.1.10: Shared Fuel Costs
2.1.1.9: calculateCosts(Gas Price, Distance, Number of Riders)
2.2: Shared Fuel Costs
2.3: Shared Fuel Costs
3: selectPaymentMethod()
3.1: getAllPaymentMethods(User ID)
4: getAllPaymentMethods (User ID)
5: Payment Card Details
6: Payment Card Details
6.1: Payment Methods
7: makePayment()
7.1: makePayment(Payment Card Details)
7.1.1: insertPaymentTransaction()
7.1.2: authorizeTransaction()
8: Funds
9: Payment Confirmation
7.2: Payment Confirmation
7.3: Payment Confirmation

Sequence Diagram: *Check Offers and Coupons*


Check Offers and Coupons
User
interface
aUser
aRatingHistory
aLocation
aAdTransaction
Sys Database
Google Maps
1: Check Offers(User ID, Driver Indicator)
1.1: Check Offers(User ID, Driver Indicator)
1.1.1: isAuthenticated(User ID)
1.1.2: true / false
1.1.3: getOverallRating(User ID)
1.1.3.1: getOverallRating(User ID)
1.1.3.2: Overall Rating
1.1.4: Overall Rating
1.1.5: calculateNumberOfRides(User ID, Driver Indicator)
1.1.6: Number of Rides
1.1.7: calculateLocation()
1.1.7.1: calculateLocation()
1.1.7.2: Location
1.1.8: Location
1.1.9: checkOffers(User ID, Number of Rides, Location)
1.1.9.1: getAllOffers()
1.1.9.2: List of Offers
1.1.9.3: calculateOffers()
1.1.9.4: Top Offer
1.2: Top Offer
1.3: Top Offer

Sequence Diagram: *Request Carpool*


Rider
interface
aUser
aFavoriteScheduleRoute
aLocation
aGroup
Sys Database
Google Maps
Driver
1: getAllFSR(Rider User ID)
1.1: getAllFSR(Ride User ID)
1.1.1: getAllFSR(Rider User ID)
1.1.2: All Favorite Schedule Routes
1.2: All Favorite Schedule Routes
1.3: All Favorite Schedule Route
2: All Favorite Schedule Routes
3: requestCarpool(Source, Destination, Schedule) or requestCarpool(Favorite Schedule Route)
3.1: requestCarpool()
3.1.1: calculateLocation()
3.1.1.1: calculateLocation()
3.1.1.2: Location
3.1.2: Location
3.1.3: getAllUsers()
3.1.4: List of Users
3.1.5: checkIfDriverIsWithinRadius(List of Users)
3.1.6: List of Users within Distance
3.1.7: filterUsersByGroup(List of Users)
3.1.7.1: filterUsersByGroup(List of Users)
3.1.7.2: Map<User, Group>
3.1.8: Map<User,Group>
3.1.9: getAllFSR(List of User IDs)
3.1.9.1: getAllFSR(List of User IDs)
3.1.9.2: All Favorite Schedule Routes for those Users
3.1.10: All Favorite Schedule Routes for those Users
3.1.11: calculateTopDrivers(Rider Favorite Schedule Routes, Drivers Favorite Schedule Routes)
3.1.12: filterTopDriversByGroup()
3.1.13: Map<User,Group>
3.1.14: Map<User,Group>
4: requestCarpool(Driver User ID)
4.1: requestCarpool(Driver User ID)
4.1.1: requestCarpool(Driver User ID)
4.1.2: Confirmation of Carpool
4.2: Confirmation of Carpool
4.3: Confirmation of Carpool

DATA DICTIONARY:

Ad Transaction Record = AD Transaction ID + AD Lookup ID + Passenger User ID + Status + Created Timestamp

AD Transaction ID = Data Element

AD Lookup ID = Data Element

Passenger User ID = Data Element

Created Timestamp = Data Element

Advertisement Lookup = AD Lookup ID + Vendor Name + Start Timestamp + End Timestamp + Ad Name + Ad Description + Ad URL Link + Price per Click

Vendor Name = Data Element

Start Timestamp = Data Element

End Timestamp = Data Element

Ad Name = Data Element

Ad Description = Data Element

Ad URL Link = Data Element

Price per Click = Data Element

Country = Country Name

Country Name = Data Element

Favorite Location = Location ID + FSR ID + Longitude Coordinate + Latitude Coordinate + Street Address + City + Zip Code + State + Country

Location ID = Data Element

FSR Name = Data Element

Longitude Coordinate = Data Element

Latitude Coordinate = Data Element

Street Address = Data Element

City = Data Element

Zip Code = Data Element

State = Data Element

Country = Data Element

**Favorite Schedule Route = FSR ID + User ID + FSR Name + Departure Timestamp + Arrival Timestamp + Start Location ID + Destination Location ID **

FSR ID = Data Element

FSR Name = Data Element

Departure Timestamp = Data Element

Arrival Timestamp = Data Element

Full Name = First Name + (Middle Name) + Last Name

Gas Price = Gas Price ID + Country ID + State ID + Start Timestamp + End Timestamp + Gas Price Per Unit

Gas Price ID = Data Element

Gas Price Per Unit = Data Element

Group = Group ID + Start Timestamp + End Timestamp + Group Name + Group Description

Group ID = Data Element

Group Name = Data Element

Group Description = Data Element

Payment Card Details Record = Payment Card ID + User ID + Payment Card Type ID + Start Timestamp + Updated Timestamp + Full Name + Card Number + Expiration Date + (Debit PIN Number)

Payment Card ID = Data Element

User ID = Data Element

Payment Card Type ID = Data Element

Card Number = Data Element

Expiration Date = Data Element

Debit PIN Number = Data Element

Payment Card Type = Payment Card Type ID + [Payment Card Type]

Payment Card Type ID = Data Element

Payment Card Type ID = [Debit | Credit]

Payment Transaction = Payment Transaction ID + Driver User ID + Passenger User ID + Status + Payment Timestamp + Departure Timestamp + Arrival Timestamp + Driver Payment Card Details ID + Passenger Payment Card Details ID + Total Shared Fuel Cost

Payment Transaction ID = Data Element

Total Shared Fuel Cost = Data Element

Payment Transaction Location = Payment Transaction Location ID + Payment Transaction ID + Longitude Coordinate + Latitude Coordinate + Street Address + City + Zip Code + State Name + Country Name

Payment Transaction Location ID = Data Element

State Name= Data Element (joined from State table)

Country Name= Data Element (joined from Country table)

Permission = Permission ID + Permission Name + Permission Description

Permission ID = Data Element

Permission Name = Data Element

Permission Description = Data Element

Rating History Record = Rating History ID + User ID to be Rated + Payment Transaction ID + Driver Indicator + Rated By User ID + Rating Name + Feedback

Rating History ID = Data Element

User ID to be Rated = Data Element

Payment Transaction ID = Data Element

Driver Indicator = Data Element

Rated By User ID = Data Element

Rating Name = Data Element

Feedback = Data Element

State = State ID + Country Name + State Code + State Name

State ID = Data Element

Country ID = Data Element

Country Name = Data Element

State Code = Data Element

State Name = Data Element

Status = Status Name

Status Name = Data Element

User = User ID + Username + Password + First Name + Last Name + Date of Birth + Email Address + Phone Number + Driver's License + Driver's License Expiration Date + 1{User Permission Matrix Record} + 0{Vehicle Details Record}3 + 1{Payment Card Details Record}3 + 1{User Permissions Matrix Matrix Record}

User Ad Transaction History = UserID + UserID + 0{Ad Transaction Record}

User Permissions Matrix Record = User ID + Group ID + Permission ID

User ID = Data Element

Group ID = Data Element

Permission ID = Data Element

User Payment Transaction History = UserID + 0{Payment Transaction Record}

Vendor Contract = Vendor LookUp ID + Created Timestamp + Updated Timestamp + Start Timestamp + End Timestamp + User ID + Registration Fee + Vendor Terms Agreement

Vendor Lookup = Vendor LookUp ID + Created Timestamp + Updated Timestamp + Start Timestamp + End Timestamp + Vendor Name + Email + Phone Number + Street Address + City + Zip Code + State Name + Country Name

Vehicle Details Record= Vehicle Details ID + User ID + VIN Number+ Make + Model + Color + Year + Vehicle Image Link + Auto Insurance File Link

FUNCTIONAL SPECIFICATIONS DOCUMENT:

Functional Specifications		
Category	**Subcategory**	**Details**
User Specifications	Account Creation Specifications	• User must be able to sign up for a new account by entering the following information: o Username o Password o First Name o Last Name o Date of Birth o Phone Number o Email o Driver’s License o Driver’s License Expiration Date
		• Initial account creation includes email confirmation to confirm the user’s identity
	Advertisement Transaction Specifications	• Users should be able view an advertisement transaction history in a UI o Allow users to check if offer was redeemed o Could potentially also be a way for detecting fraud
	Authentication Specifications	• User must be able to authenticate using a username and password to authenticate with the UTDallas database
		• A secondary option is to authenticate using fingerprints on the mobile app. The fingerprint would be stored as a long-encrypted number to protect the privacy of users’ biometric information
		• User must be able reset and change password
	Group Specifications	• When managing carpooling groups and their associated users, Users must be able to do the following: o Create groups o Add new users o Modify user permissions within the group o Join other groups with admin approval
	Instant Chat Specifications	• User and drivers should be able to instant chat with each other privately
		• Users within a group should be able to make social media posts and instant chat with each other
	Payment Card Details	• For ease of making payments for sharing fuel costs, a

	Specifications	User must be able to save his or her credit or debit card information securely with VISA, Discover, and MasterCard: o Credit Card ▪ Full Name ▪ Credit Card Number ▪ Expiration Date o Debit Card ▪ Full Name ▪ Debit Card Number ▪ Expiration Date ▪ Debit PIN
	Payment Transaction History Specifications	• Drivers should be able to see all transaction history related to passenger paying for their ride. o Payment Transaction ID o Status o Driver’s Full Name o Passenger’s Full Name o Payment Time o Departure Time o Arrival Time o Start Location o Destination Location o Total Cost
		• Users must be able to view all his or her payment transaction history. For each record, the following fields must be displayed to user: o Payment Transaction ID o Status o Driver’s Full Name o Passenger’s Full Name o Payment Time o Departure Time o Arrival Time o Start Location o Destination Location o Total Cost
	Payment Transaction Processing Specifications	• Making payments with a credit card should require the following parameters and be authorized by a credit card company: o Full Name o Card Number o Expiration Date
		• Making payments with a debit card should require the

		following parameters and be authorized by a bank: o Full Name o Card Number o Expiration Date o Debit PIN
	Vehicle Details Specifications	• If the User desires to be a driver within a group, then he or she must submit following information and documents: o VIN (Vehicle Identification Number) o Make o Model o Color o Year o Proof of Auto Insurance o Image of Vehicle
Customer Service Specifications	Customer Service Specifications	• Customer Service employees should have the ability to cancel fraudulent transactions related to ads and payments.
Vendor Specifications	Vendor Specifications	• Vendors should be able to sign-in into a portal
		• Vendors must be to perform the functions: o Pay annual registration fee o View and renew vendor agreement o Update advertisements, offers, and coupon o Monitor the transaction history of all customer (or passengers) who have clicked or viewed their advertisements, offers, and coupons

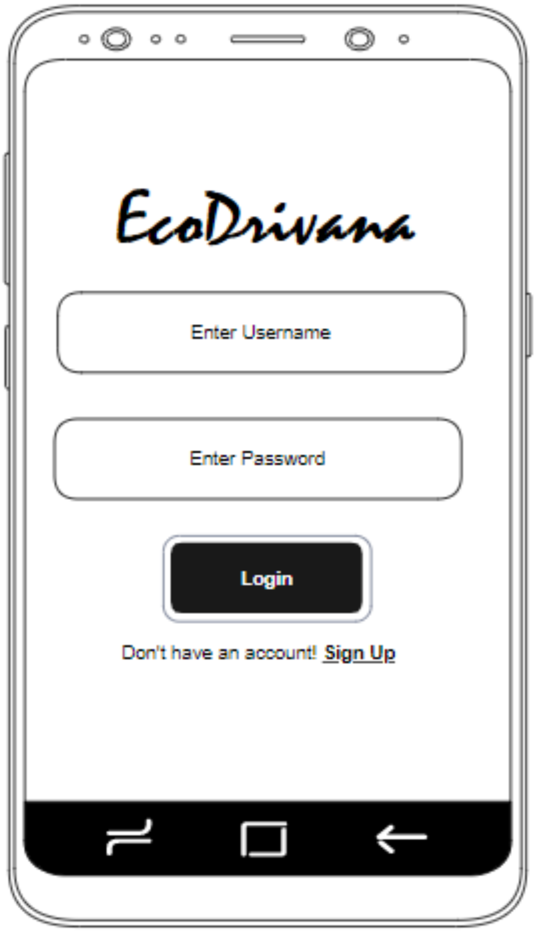
NON-FUNCTIONAL SPECIFICATION DOCUMENT:

Non-Functional Specifications		
Category	**Sub-Category**	**Details**
Platform	Platform	• Software must be compatible with Android and iOS.
		• Code must be developed, stored, and tracked in GitHub
		• API binaries must be deployed to the Amazon Web Services (AWS), a well-known cloud provider
		• Databases should be deployed to AWS RDS (Relational Database Service)
		• UTDallas Authentication API must be used to authenticate UTDallas students

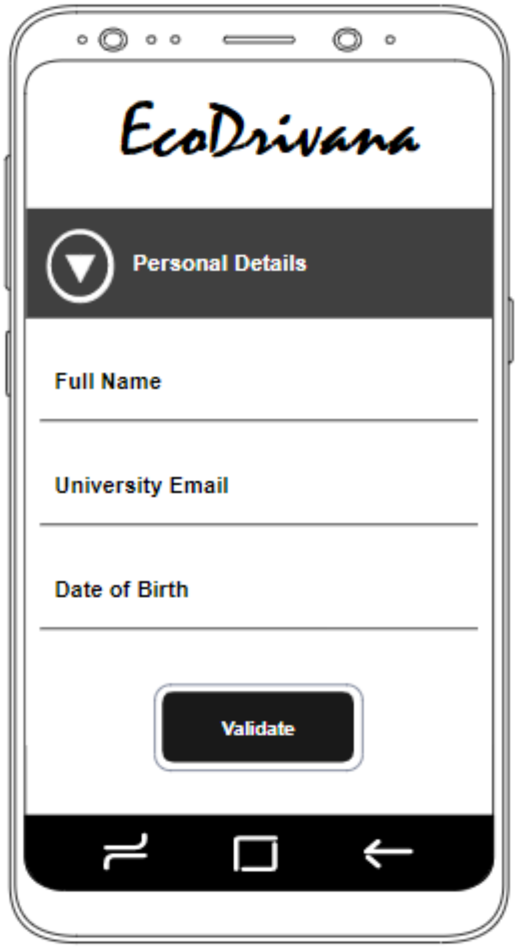
		• Department of Public Safety API must be able to verify driver's licenses.
		• Gas Buddy API should be used for Gas Pricing through a batch job that copies gas prices into the Sys Database
		• Google Maps API must be used to track the user's location and routes
		• Payment Gateways / Banks must be used to process transactions related to Paypal, credit card, and debit card
PCI Compliance	PCI Compliance	• Credit card and debit card related data must be encrypted across networks and at rest
		• Credit card numbers must be stored as hashes
		• Credit card security code must not be stored
		• Credit card and debit card related data must be encrypted in database with strict access control
Performance	Performance	• Server response time: < 2 seconds
		• User friendliness
Security and Access Control	Authentication	• Email Confirmation must be used to verify a person's identity
		• Username and password authentication
		• Fingerprint authentication
	Authorization	• By default, users are given default permission to modify their own profile details, favorite schedule routes, payment details, and vehicle details.
		• Customer representatives can modify user account only with approval from users when assistance is needed.
		• Group administrators can add users, remove users, and grant user specific permissions.

USER INTERFACES

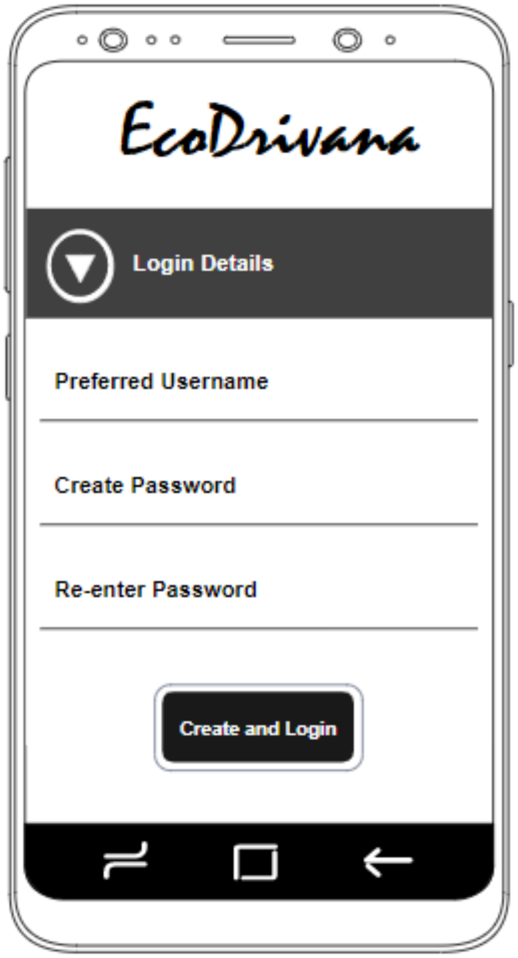
Login UI:



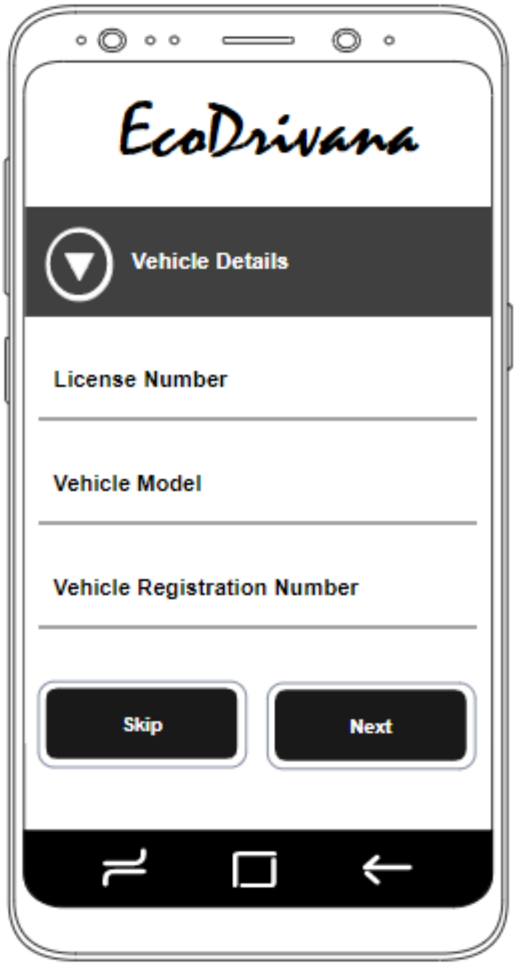
Create Profile UI:



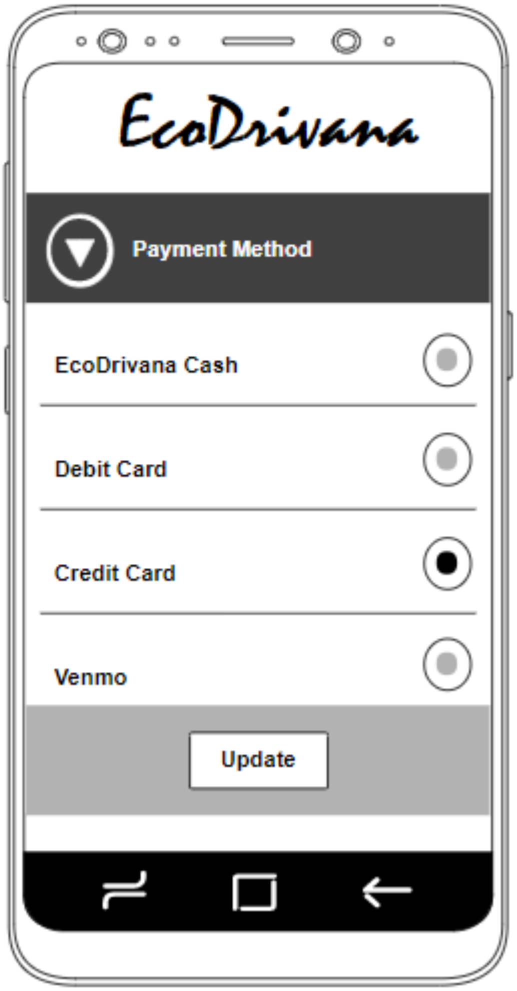
Create Profile UI:



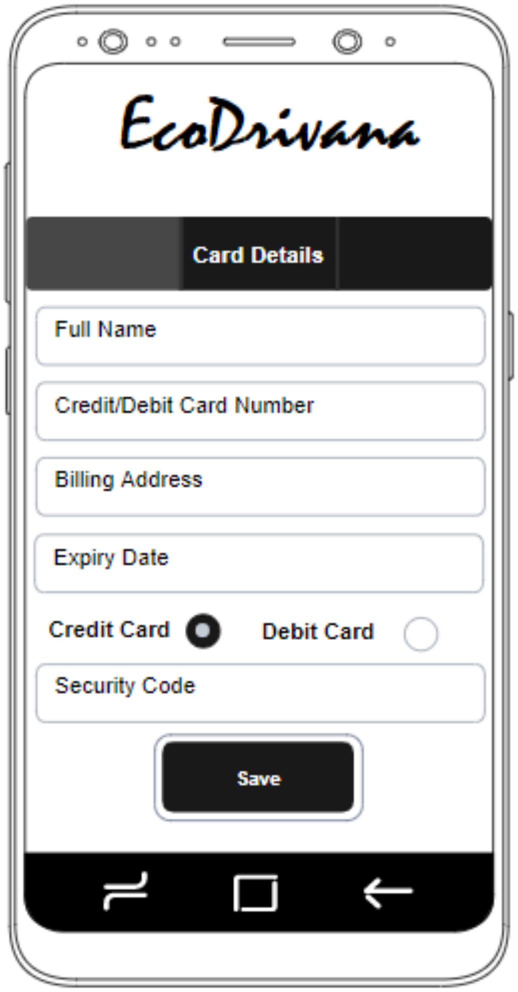
Add or Update Vehicle Details UI:



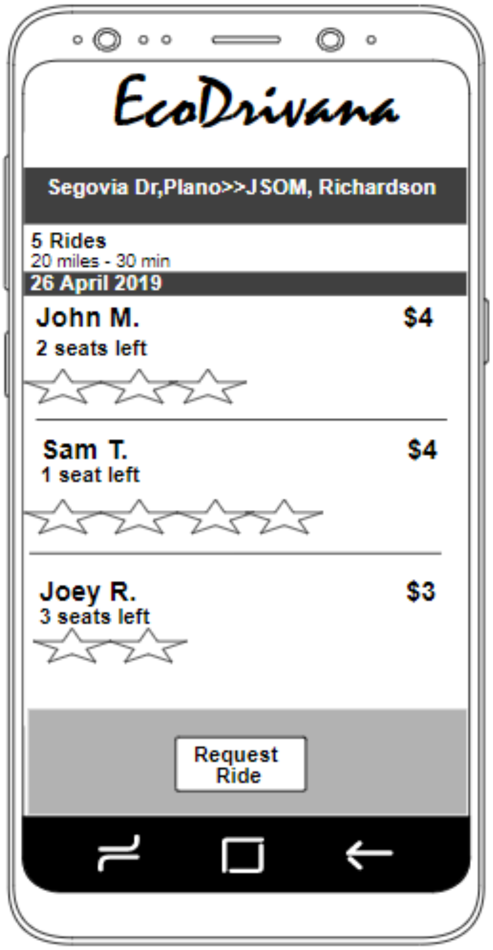
Add or Update Payment Method UI:



Add or Update Card Details UI:



Request Driver UI:



EcoDrivana

Segovia Dr,Plano>>JSOM, Richardson

5 Rides
20 miles - 30 min

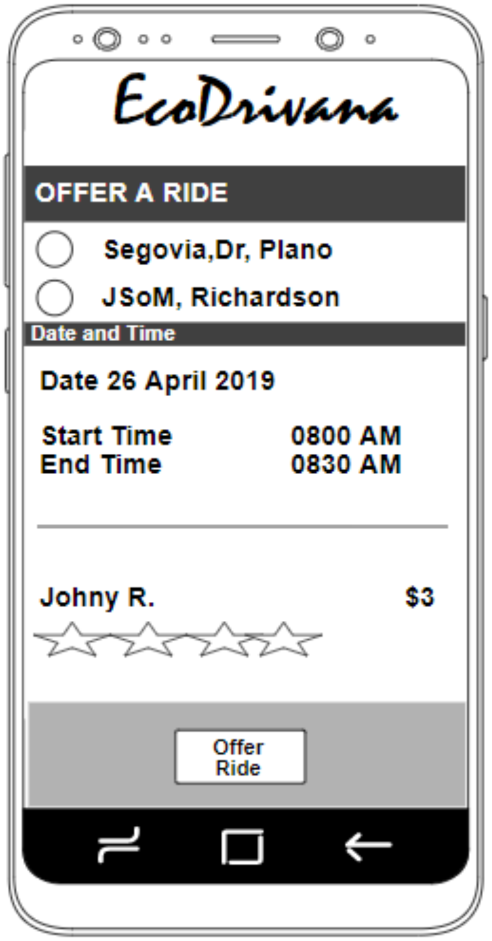
26 April 2019

John M. $4
2 seats left

Sam T. $4
1 seat left

Joey R. $3
3 seats left

Offer Ride UI:



EcoDrivana

OFFER A RIDE

○ Segovia,Dr, Plano
○ JSoM, Richardson

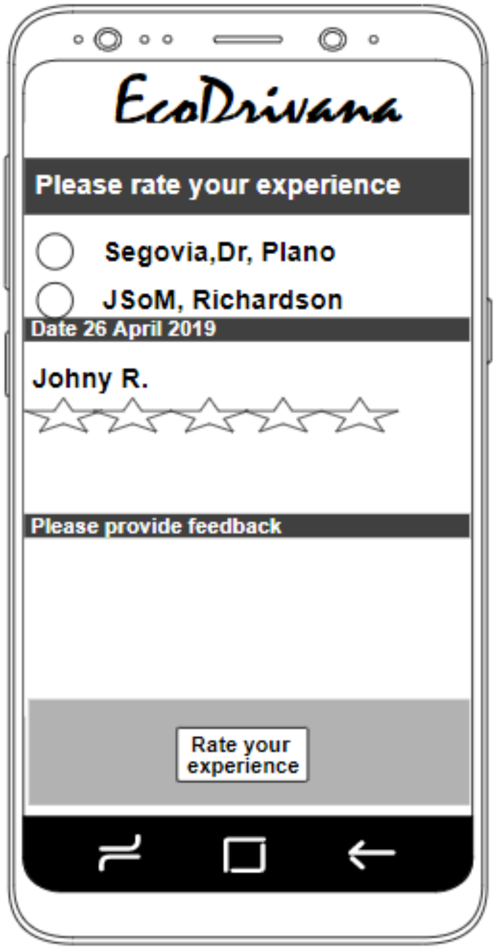
Date and Time

Date 26 April 2019

Start Time 0800 AM
End Time 0830 AM

Johny R. $3

Provide Rating (by Driver) UI:



EcoDrivana

Please rate your experience

○ Segovia,Dr, Plano

○ JSoM, Richardson

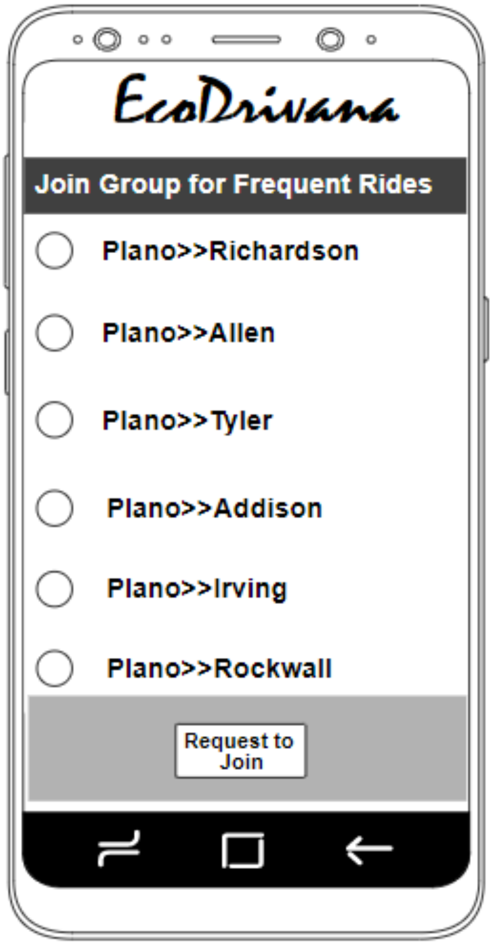
Date 26 April 2019

Johny R.

☆☆☆☆☆

Please provide feedback

Join Group UI:



EcoDrivana

Join Group for Frequent Rides

○ Plano>>Richardson

○ Plano>>Allen

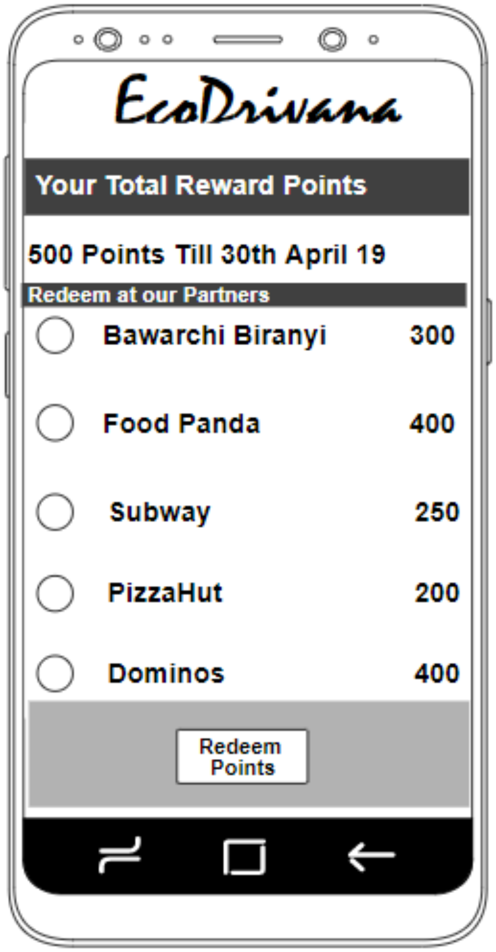
○ Plano>>Tyler

○ Plano>>Addison

○ Plano>>Irving

○ Plano>>Rockwall

View Reward Points UI:



EcoDrivana

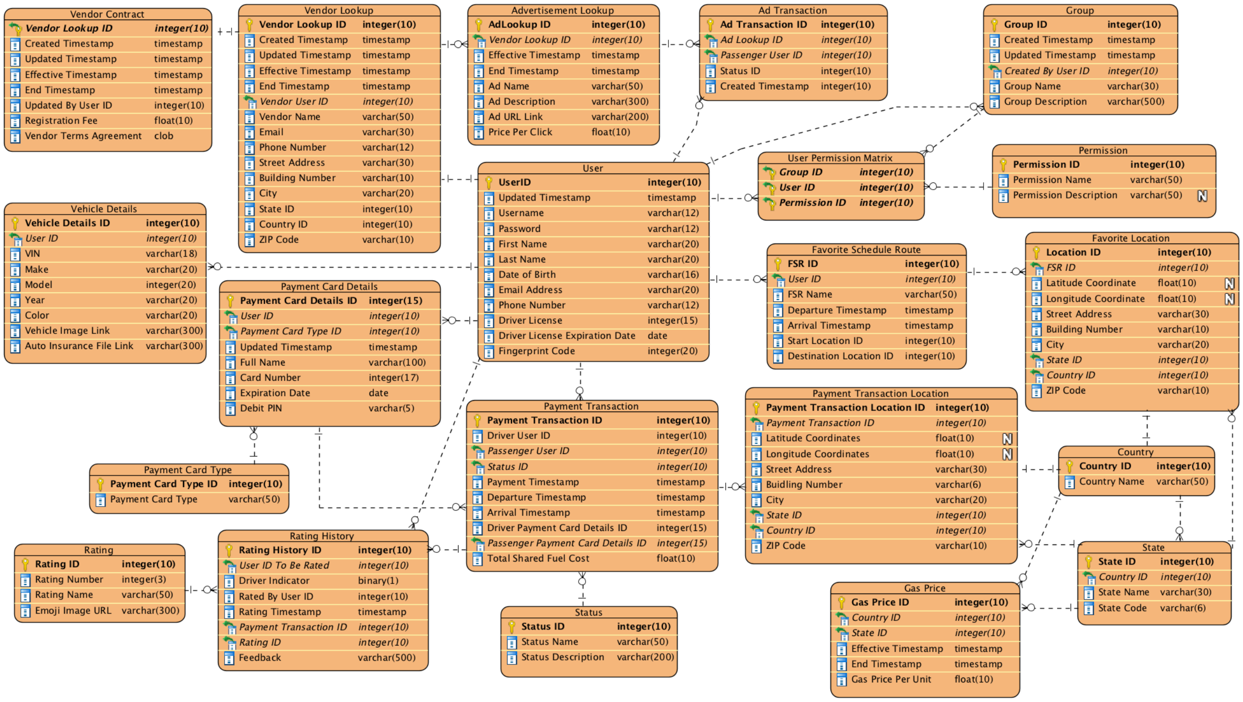
Your Total Reward Points

500 Points Till 30th April 19

Redeem at our Partners

○	Bawarchi Biranyi	300
○	Food Panda	400
○	Subway	250
○	PizzaHut	200
○	Dominos	400

DATABASE DESIGN:



Vendor Contract
Vendor Lookup ID integer(10)
Created Timestamp timestamp
Updated Timestamp timestamp
Effective Timestamp timestamp
End Timestamp timestamp
Updated By User ID integer(10)
Registration Fee float(10)
Vendor Terms Agreement clob
Vendor Lookup
Vendor Lookup ID integer(10)
Created Timestamp timestamp
Updated Timestamp timestamp
Effective Timestamp timestamp
End Timestamp timestamp
Vendor User ID integer(10)
Vendor Name varchar(50)
Email varchar(30)
Phone Number varchar(12)
Street Address varchar(30)
Building Number varchar(10)
City varchar(20)
State ID integer(10)
Country ID integer(10)
ZIP Code varchar(10)
Advertisement Lookup
AdLookup ID integer(10)
Vendor Lookup ID integer(10)
Effective Timestamp timestamp
End Timestamp timestamp
Ad Name varchar(50)
Ad Description varchar(300)
Ad URL Link varchar(200)
Price Per Click float(10)
Ad Transaction
Ad Transaction ID integer(10)
Ad Lookup ID integer(10)
Passenger User ID integer(10)
Status ID integer(10)
Created Timestamp integer(10)
Group
Group ID integer(10)
Created Timestamp timestamp
Updated Timestamp timestamp
Created By User ID integer(10)
Group Name varchar(30)
Group Description varchar(500)
User Permission Matrix
Group ID integer(10)
User ID integer(10)
Permission ID integer(10)
Permission
Permission ID integer(10)
Permission Name varchar(50)
Permission Description varchar(50) N
User
UserID integer(10)
Updated Timestamp timestamp
Username varchar(12)
Password varchar(12)
First Name varchar(20)
Last Name varchar(20)
Date of Birth varchar(16)
Email Address varchar(20)
Phone Number varchar(12)
Driver License integer(15)
Driver License Expiration Date date
Fingerprint Code integer(20)
Vehicle Details
Vehicle Details ID integer(10)
User ID integer(10)
VIN varchar(18)
Make varchar(20)
Model integer(20)
Year varchar(20)
Color varchar(20)
Vehicle Image Link varchar(300)
Auto Insurance File Link varchar(300)
Payment Card Details
Payment Card Details ID integer(15)
User ID integer(10)
Payment Card Type ID integer(10)
Updated Timestamp timestamp
Full Name varchar(100)
Card Number integer(17)
Expiration Date date
Debit PIN varchar(5)
Favorite Schedule Route
FSR ID integer(10)
User ID integer(10)
FSR Name varchar(50)
Departure Timestamp timestamp
Arrival Timestamp timestamp
Start Location ID integer(10)
Destination Location ID integer(10)
Favorite Location
Location ID integer(10)
FSR ID integer(10)
Latitude Coordinate float(10) N
Longitude Coordinate float(10) N
Street Address varchar(30)
Building Number varchar(10)
City varchar(20)
State ID integer(10)
Country ID integer(10)
ZIP Code varchar(10)
Payment Card Type
Payment Card Type ID integer(10)
Payment Card Type varchar(50)
Payment Transaction
Payment Transaction ID integer(10)
Driver User ID integer(10)
Passenger User ID integer(10)
Status ID integer(10)
Payment Timestamp timestamp
Departure Timestamp timestamp
Arrival Timestamp timestamp
Driver Payment Card Details ID integer(15)
Passenger Payment Card Details ID integer(15)
Total Shared Fuel Cost float(10)
Payment Transaction Location
Payment Transaction Location ID integer(10)
Payment Transaction ID integer(10)
Latitude Coordinates float(10) N
Longitude Coordinates float(10) N
Street Address varchar(30)
Buidling Number varchar(6)
City varchar(20)
State ID integer(10)
Country ID integer(10)
ZIP Code varchar(10)
Country
Country ID integer(10)
Country Name varchar(50)
Rating
Rating ID integer(10)
Rating Number integer(3)
Rating Name varchar(50)
Emoji Image URL varchar(300)
Rating History
Rating History ID integer(10)
User ID To Be Rated integer(10)
Driver Indicator binary(1)
Rated By User ID integer(10)
Rating Timestamp timestamp
Payment Transaction ID integer(10)
Rating ID integer(10)
Feedback varchar(500)
Status
Status ID integer(10)
Status Name varchar(50)
Status Description varchar(200)
Gas Price
Gas Price ID integer(10)
Country ID integer(10)
State ID integer(10)
Effective Timestamp timestamp
End Timestamp timestamp
Gas Price Per Unit float(10)
State
State ID integer(10)
Country ID integer(10)
State Name varchar(30)
State Code varchar(6)

DATABASE CONSTRAINTS:

Table: Ad Transaction
Constraints:

- AD Transaction ID must be non-null and unique to be a primary key
- AD Lookup ID must be non-null, unique, and reference the Advertisement Lookup Table to satisfy the foreign key constraint
- Passenger User ID must be non-null, unique, and reference the User Table to satisfy the foreign key constraint
- Created Timestamp must non-null

Table: Advertisement Lookup
Constraints:

- AD Lookup ID must be non-null and unique to be a primary key
- Vendor Lookup ID must be non-null, unique, and reference the Vendor Lookup Table to satisfy the foreign key constraint
- Effective Timestamp, End Timestamp, Ad Name, Ad Description, Ad URL Link, and Price per Click should be non-null

Table: Country
Constraints:

- Country ID must be non-null and unique to be a primary key
- Country Name must non-null and unique

Table: Favorite Location
Constraints:

- Location ID must be non-null and unique to be a primary key

- The FSR ID must be non-null, unique, and reference the Favorite Schedule Route Table to satisfy the foreign key constraint
- Longitude Coordinate, Latitude Coordinate, Street Address, City, and Zip Code must all be non-null.
- State ID must be non-null and reference the State table to satisfy the foreign key constraint
- Country ID must be non-null and reference the Country table to satisfy the foreign key constraint

Table: Favorite Schedule Route (FSR)
Constraints:

- The FSR ID must be non-null and unique to be a primary key
- The User ID must be non-null, unique, and reference the User Table to satisfy the foreign key constraint
- The Departure Timestamp and Arrival Timestamp must non-null and unique
- The Start Location ID and Destination Location ID must be non-null, unique, and reference the Country Table to satisfy the foreign key constraint

Table: Gas Price
Constraints:

- Gas Price ID must be non-null and unique to be a primary key
- The Country ID must be non-null, unique, and reference the Country Table to satisfy the foreign key constraint
- The State ID must be non-null, unique, and reference the State Table to satisfy the foreign key constraint
- The Effective Timestamp, End Timestamp, and Gas Price Per Unit must be non-null

Table: Group
Constraints:

- Group ID must be non-null and unique to be a primary key
- Created Timestamp, Updated Timestamp, Group Name, Group Description must be non-null
- The Created by User ID is a foreign key that references the User table. This field is non-null.

Table: Payment Card Details
Constraints:

- Payment Card Details ID must be non-null and unique to be a primary key
- The User ID must be non-null, unique, and reference the User Table to satisfy the foreign key constraint
- Payment Card Type ID must be non-null, unique, and reference the Payment Card Type Table to satisfy the foreign key constraint
- The Created Timestamp, Updated Timestamp, Full Name, Card Number, and Expiration Date must be non-null
- The Card Number must be 16 digits long
- The Expiration Date must be in the format ‘mm/yyyy’
- The Debit PIN Number is required for Debit Cards
- **Note**: All Payment Card Details data must be encrypted with AES 128-bit encryption to be compliant with PCI data security standards.

Table: Payment Card Type
Constraints:

- Payment Card Type ID must be non-null and unique to be a primary key
- The Payment Card Type must be non-null and be a member of the set (‘Credit’, ‘Debit’).

Table: Payment Transaction
Constraints:

- Payment Transaction ID must be non-null and unique to be a primary key
- The Driver User ID and Passenger User ID must be non-null, unique, and reference the User Table to satisfy the foreign key constraint
- The Payment Timestamp, Departure Timestamp, and Arrival Timestamp must be non-null.
- The Driver Payment Card Details ID and Passenger Payment Card Details ID must be non-null, unique, and reference the Payment Card Details Table to satisfy the foreign key constraint
- The Total Shared Cost must be floating point number greater than 0.0

Table: Payment Transaction Location
Constraints:

- Payment Transaction Location ID must be non-null and unique to be a primary key
- The Payment Transaction ID must be non-null, unique, and reference the Payment Transaction Table to satisfy the foreign key constraint
- Longitude Coordinate, Latitude Coordinate, Street Address, City, and Zip Code must all be non-null.
- State ID must be non-null and reference the State table to satisfy the foreign key constraint
- Country ID must be non-null and reference the Country table to satisfy the foreign key constraint

Table: Permission
Constraints:

- Permission ID must be non-null and unique to be a primary key
- Permission Name must be non-null and unique
- Permission Description must be non-null

Table: Rating
Constraints:

- Rating ID is a primary key
- Rating Number must be a positive integer greater than 0 and is non-null
- Rating Name must be non-empty string and is non-null
- Emoji Image is a non-null string.

Table: Rating History
Constraints:

- Rating History ID is a primary key.
- User ID to be Rated and Rated by User ID are foreign keys referencing the User table. Both fields are non-null
- Driver Indicator is non-null field that can only be true or false.
- Rating Timestamp is non-null
- Payment Transaction ID is a foreign key referencing the Payment Transaction table. This field is non-null
- Rating ID is foreign key referencing the Rating table. This field is non-null.
- Feedback is restricted to 500 characters and is nullable.

Table: State
Constraints:

- State ID must be non-null and unique to be a primary key
- The Country ID must be non-null, unique, and reference the Country Table to satisfy the foreign key constraint
- State Code must non-null, unique, and no longer than 3 characters

- State Name must non-null and unique

Table: User
Constraints:

- The UserID must be non-null and unique to be a primary key
- The Username must non-null and unique as part of a unique constraint.
- The Password must non-null and hashed for security reasons
- The Email Address, Date of Birth, Phone Number, Driver's License, Driver's License Expiration Date must all be non-null
- The Fingerprint Code is nullable, and is only used for secondary authentication
- **Note**: All User fields must be encrypted to protect the privacy and security of user data.

Table: User Permissions Matrix
Constraints:

- The User ID must be non-null, unique, and reference the User Table to satisfy the foreign key constraint
- The Group ID must be non-null, unique, and reference the Group Table to satisfy the foreign key constraint
- The Permission ID must be non-null, unique, and reference the User Table to satisfy the foreign key constraint

Table: Vendor Contract
Constraints:

- Vendor LookUp ID must be non-null and unique since it is a primary key. Also, the primary key references the Vendor Lookup table
- Created Timestamp, Updated Timestamp, Effective Timestamp, and End Timestamp must be non-null
- Updated By UserID must be in the User table
- Vendor Terms Agreement must non-null

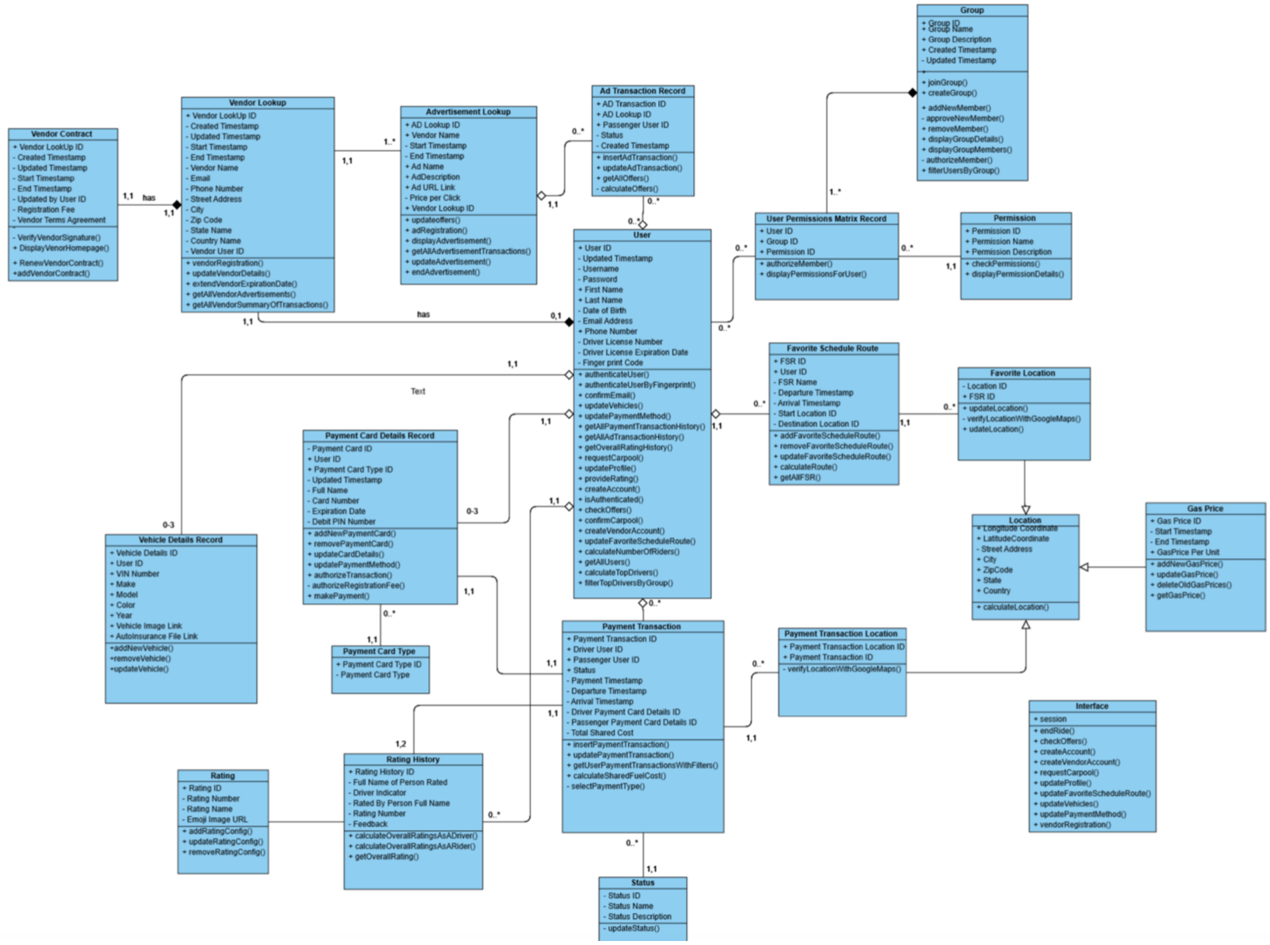
Table: Vendor Lookup
Constraints:

- Vendor LookUp ID must be non-null and unique since it is a primary key
- Created Timestamp, Updated Timestamp, Effective Timestamp, and End Timestamp must be non-null
- The Vendor User ID must be non-null, unique, and reference the User Table to satisfy the foreign key constraint
- Vendor Name should be non-null and not an empty string
- Email, Phone Number, Street Address, City, and Zip Code must all be non-null.
- State ID must be non-null and reference the State table to satisfy the foreign key constraint
- Country ID must be non-null and reference the Country table to satisfy the foreign key constraint

Table: Vehicle Details
Constraints:

- Vehicle Details ID must be non-null and unique to be a primary key
- The User ID must be non-null, unique, and reference the User Table to satisfy the foreign key constraint
- VIN Number, Make, Model, Color, Year, Vehicle Image Link, and Auto Insurance File Link must non null.

CLASS DIAGRAMS WITH METHODS:



SOFTWARE DESIGN CONTRACTS:

Signature:

Method Name: calculateSharedFuelCost()

Class Name: Payment Transaction **ID:** Payment Transaction ID

Clients (Consumers): User, Driver

Associated Use Cases: Shared Fuel Cost

Description of Responsibilities: Calculates the shared fuel cost using the distance, number of riders, and gas price from the Gas Buddy API for the user and driver

Arguments Received: Driver User ID

Type of Value Returned: Calculated Shared Fuel amount in dollars

Pre-Conditions: User and Driver must confirm the ride

Post-Conditions: Displaying the Shared Fuel Cost to the User and Driver

LOGIC:

1) Fetch the calculated distance in miles from the Google Maps based on the Start and Destination Location
2) Fetch the number of riders from Payment Transaction table according to the number of users have confirmed the ride
3) Fetch the gas price from the Gas Price Table by using the Start Location and Destination Location's State and Country ID (populated by a batch job that calls the Gas Buddy API)
4) Calculate the Shared Fuel Cost by performing the formula:

Shared Fuel Cost = (Calculated Distance in miles * Price of gas usage per mile) / (Number of Riders + 1)

Signature:

Method Name: calculateOffers(String userID)

Class Name: Ad Transaction **ID:** Ad Transaction ID

Clients (Consumers): User

Associated Use Cases: Check Offers and Coupons, Account Validation, Show Offers based on User's Rating, Use Offer, Redeem as Points

Description of Responsibilities: Calculates the offers available for each User based on their individual Average Rating and Number of Rides taken.

Arguments Received: User ID

Type of Value Returned: List of Offers

Pre-Conditions: User must have an average rating and taken at least one ride.

Post-Conditions: Displaying the available list of offers

LOGIC:

1. Calculate the Average Rating from Rating History table using averaging and aggregations for the given User ID
2. Use Google Maps to calculate the real time location using latitude and longitude coordinates
3. Calculate the number of rides from the Payment Transaction Table
4. Get Advertisements and Vendors within a specific radius from the Advertisement Lookup and Vendor Lookup tables
5. In the UI, display the top offer that the user is eligible for.

Signature:

Method Name: requestCarpool()

Class Name: User **ID:** UserID

Clients (Consumers): User, Driver

Associated Use Cases: Request Carpool, Enter Schedule, Source & Destination, Request Nearby Driver, Share Driver's Profile & Rating, Share Rider's Profile & Rating, Confirm Ride Details, Driver Confirmation

Description of Responsibilities: Requesting for a Carpool by entering the Start location and Destination location, leading to the search for nearby drivers and sharing their respective Profile and Rating if their routes match.

Arguments Received: Schedule, Source & Destination Location

Type of Value Returned: Driver & Rider's Profile and Rating

Pre-Conditions: User must have a registered account

Post-Conditions: Ride Confirmation

LOGIC:

1. Verify that the Rider is signed in
2. Enter the Schedule, Source & Destination Location or select a saved Favorite Schedule Route from the Favorite Schedule Route table
3. Based on the Schedule, Source & Destination Location entered and/or similar favorite schedule route, use Google Maps to filter the nearby drivers (based on latitude and longitude coordinates), fetched from the User database and/or Group database
4. If there are no rides available in that Schedule, Locations, then display "No rides available, please check later"
5. If Rides are available, then share the Driver's Profile and Rating to the User
6. If the User confirms the Ride then share the User's Profile and Rating to the Driver
7. Send confirmation to the User if the Driver confirms the Ride. Now in the UI, the Rider can continue tracking the exact location of the Driver using Google Maps's location tracking capabilities. Similarly, in the UI, the Driver can view his / her exact location and the Rider's exact location using Google Maps' location tracking capabilities.
8. If the driver rejects the request for the ride, then display "The driver cancelled your ride", and then the app will then start searching for nearby drivers or drivers that are a member of the same group as the Rider.

Signature:

Method Name: authorizeMember()

Class Name: User Permission Matrix Record **ID:** UserID

Clients (Consumers): User

Associated Use Cases: Join Group, Send Request to Group, Acknowledgement from group, Contact Group Member

Description of Responsibilities: Requesting to join a group that matches the User's Favorite Schedule Route.

Arguments Received: Group ID, Permission ID, User ID, User Profile and Favorite Schedule Route

Type of Value Returned: Permission, Group Details, Group Members

Pre-Conditions: User must have a registered account and up to date profile

Post-Conditions: Authorization

LOGIC:

1. Verify that the Group ID is valid in the Group table
2. Verify that the User A is a Group Admin for him or her to grant permission using the Group Database
3. Group Admin checks the (User requesting authorization) User B's Profile and Favorite Schedule Route using the User Database and Favorite Schedule Route Database
4. If Group Admin decides to grant permission to User B, he/she will grant access through the UI and a record is updated in the User Permissions Matrix table leading to User A receiving authorization
5. Otherwise, authorization for User A will be denied.

Signature:

Method Name: vendorRegistration()

Class Name: Interface **ID:** None

Clients (Consumers): User

Associated Use Cases: Account Registration, Registration Fees

Description of Responsibilities: A Vendor is registering his company to post advertisements for marketing

Arguments Received: User, Vendor Details, Payment

Type of Value Returned: Advertisement Lookup

Pre-Conditions: The User must be a representative of the vendor company

Post-Conditions: Advertisement is posted for Riders to view

LOGIC:

1. The Vendor Representative (denoted as Vendor User) enters First Name, Last Name, Email, Date of Birth, Phone Number, Driver's License, Driver's License Expiration Date, and Fingerprint Code for User Account Registration.
2. Validate the User data for errors. If there are no errors, then save the User data to the device cache. Otherwise, warn the Vendor representative about the fields that have errors.
3. The Vendor User updates the company information like Vendor Name, Street Address, Building Number, City, State, Country, Email, and Phone Number.
4. Validate the Vendor Lookup data for errors. If there are no errors, then save the Vendor Lookup data to the device cache. Otherwise, warn the Vendor representative about the fields that have errors
5. The Vendor User updates the Payment Card Details under the payment method option.
6. Validate the Payment Card Details data for errors. If there are no errors, then save the Vendor Lookup data to the device cache. Otherwise, warn the Vendor representative about the fields that have errors
7. The Vendor User initiates the Registration Fee Payment for the Advertisement
8. The Vendor User clicks on "Sign Up" button on main screen and can see advertisements in carpooling system
9. A Vendor User record is created in the User table.
10. A Vendor Lookup record is created in the Vendor Lookup table. This record references the User table.
11. A Payment Card Details record is created in the Payment Card Details table. This record references the User table.
12. The Bank completes the Registration Fee Payment Transaction and returns Confirmation Message
13. If successful, the Vendor User gets a Confirmation Message and the Vendor is now authorized to register Advertisements for the Rider to view. Otherwise, an error message is

thrown stating that the “Registration Payment Fee transaction failed”, and Vendor User is redirected to fix the Payment Card Details.

14. If the entire Vendor Registration is canceled or times out in 30 minutes, the Registration Fee Payment transaction is canceled, and the records associated with the given Vendor User will be deleted in the Payment Card Details, Vendor Lookup, and User tables.

REFERENCES:

Carson, Eric. “10 rideshare apps to crowdsource your commute”. *Tech Republic,* 10 April 2014, www.techrepublic.com/article/10-rideshare-apps-to-crowdsource-your-commute/. Accessed 25 July 2019

Russell, Robin. “Enrollment Growth Stays on Pace to Exceed Strategic Plan Goals”. *UTD News Center,* 18 November 2015, www.utdallas.edu/news/2015/11/18-31793_Enrollment-Growth-Stays-on-Pace-to-Exceed-Strategi_story-wide.html. Accessed 25 July 2019

Russell, Robin. “New Master Plan Pictures Possibilities for Future of the University”. *UTD News Center,* 25 February 2019, www.utdallas.edu/news/campus/new-master-plan-2019/. Accessed 10 June 2019

Silver, Jonathan. “Here’s how long Dallas commuters will be stuck in traffic this year”. *CultureMap LLC,* 7 March 2019, dallas.culturemap.com/news/city-life/03-07-19-dallas-worst-traffic-commute-times-america. Accessed 10 July 2019

PROJECT ACTIVITIES

The Project Activities table is responsible for summarizing the project activities that were repeatedly executed through the project. Deadlines, actual tasks, assignees, and peer reviewers are detailed in the section below.

Activity	Description
Research	Online Research was done individually to develop a possible project idea.
Brainstorming	After online research was conduct, project ideas were discussed, and the team finally decided on a project idea.
Build Diagram and Documentation	All diagrams were built in Visual Paradigm. All written documentation was developed in Microsoft Word.
Peer Review	Each task was assigned a up to 2 developers and up to 3 peer reviewers. Peer views are a form of check and balance.
Conference Call Meeting	Conference call meetings were established get quick feedback on the project, reviews, status reports, and provide scheduling to team members. Team members were encouraged to speak up about what processes need to be improved. All meetings had to be face to face because team members were in and outside of Dallas in different timezones.
Project Tracking	A spreadsheet was created to track the progress of tasks and sub-tasks.
Project Scheduling	Tasks are assignment each team member with a deadline. Up to two members would build the diagram and up to 3 people would provide a peer review as check and balance to ensure the work meets the requirements with high quality

ALLOCATION OF ACTIVITIES TO TEAM MEMBERS:

Allocation of Task Activities to Team Members			
Category	**Task Name**	**Resource Name**	**Reviewer Name**
Project Report Deliverables	Executive Summary	Kevin Sy, Amey Saste, Sanjana Buchala	All Team Members
	Problem statement	Kevin Sy, Monika Malik, Sanjana Buchala	All Team Members
	Objective	Kevin Sy, Sanjana Buchala	All Team Members
	Scope	Sanjana Buchala, Kevin Sy, Anamika Soni	All Team Members
	BPMN model / Choreography Diagram	Amey Saste, Kevin Sy	All Team Members
	Context Diagram	Anamika Soni, Kevin Sy	All Team Members
	Use-Case-Diagram	Anamika Soni, Kevin Sy	All Team Members
	Use-Case-Description	Amey Saste, Kevin Sy	All Team Members
	Class Diagram	Sanjana Buchala, Kevin Sy	All Team Members
	Sequence Diagram	Sanjana Buchala, Kevin Sy	All Team Members
	Data Dictionary	Monika Malik, Kevin Sy	All Team Members
	Functional	Kevin Sy and Monika Malik	All Team Members

	Specification Document		
	Non-Functional Specification Document	Kevin Sy, Amey Saste, Anamika Soni	All Team Members
	User Interface Design	Amey Saste,Anamika Soni, Monika Malik	All Team Members
	DataBase Design	Kevin Sy, Sanjana Buchala	All Team Members
	DataBase Constraints	Kevin Sy, Monika Malik	All Team Members
	Class Diagrams with Methods	Kevin Sy, Sanjana Buchala	All Team Members
	Software Design Contracts	Kevin Sy, Sanjana Buchala	All Team Members
Project Management Deliverables	Project Activities	Kevin Sy	All Team Members
	Task Allocated to Team Members	Kevin Sy, Monika Malik	All Team Members
	Planned Timeline	Kevin Sy	All Team Members
	Execution Timeline	Kevin Sy	All Team Members
	Meeting Minutes	Kevin Sy	All Team Members

PLANNED TIMELINE:

The Planned Timeline is designed to ensure smooth execution of the project. Sequential and parallel execution is intended to meet the theoretical deadlines. Development and peer reviews were executed as separation of duties to ensure higher quality of work.

Planned Timeline			
Week		**Deadline**	**Task (s) Completed**
May 27th, 2019	June 2nd, 2019	June 1st, 2019	• Group Sign-Up and formation
		June 1st, 2019	• Exchanged phone numbers and emails to create a WhatsApp group for collaboration
		June 1st, 2019	• Set up Skype for Business to establish conference calls
		June 1st, 2019	• Created a shared OneDrive to share documents and diagrams for the project
June 3rd, 2019	June 9th, 2019	June 5th, 2019	• View Sample Project Reports from previous semester
		June 7th, 2019	• Research world problems and list outs project idea
		June 9th, 2019	• Decide on a project idea as group
June 10th, 2019	June 17th, 2019	June 10th, 2019	• Get the project idea approved by professor
		June 14th, 2019	• Develop version 1 of the Product Name, Mission Statement, Executive Summary, Problem Statement, Scope, and Business Objectives

		June 17th, 2019	• Develop a rough draft of the smaller use cases before consolidating into a larger and more generalized (Team effort)
June 18th, 2019	June 24th, 2019	June 20th, 2019	• Develop finalized version of the more generalized Use Case Diagram
		June 23rd, 2019	• Develop version 1 the Functional and Non-Functional Specifications Document
		June 23rd, 2019	• Develop version 1 of the Context Diagram
		June 23rd, 2019	• Develop version 1 of the Database Diagram
June 25th, 2019	July 1st, 2019	June 27th, 2019	• Develop version 1 of the Data Dictionary
		June 27th, 2019	• Develop version 1 of the Database Constraints
		July 1st, 2019	• Finalize the Database Diagram in Visual Paradigm
		July 1st, 2019	• Finalize the Functional and Non-Functional Specifications Document
July 2nd, 2019	July 8th, 2019	N/A	• Practice homework problems and study for Midterm
July 9th, 2019	July 15th, 2019	July 11, 2019	• Develop version 1 of the Use Case Descriptions
		July 11, 2019	• Develop version 1 of the Data Model / Class Diagram without Methods
		July 13, 2019	• Finalize Use Case Descriptions
		July 13, 2019	• Finalize the Data Dictionary
		July 15, 2019	• Finalize the Data Model / Class Diagram without Methods
July 16th, 2019	July 22nd, 2019	July 18th, 2019	• Develop version 1 of the Complete Class Diagram with Methods
		July 20th, 2019	• Finalize Complete Class Diagram with Methods
		July 18th, 2019	• Develop version 1 of the Choreography Diagram
		July 20th, 2019	• Finalize Choreography Diagram
		July 21st, 2019	• Develop version 1 of the Sequence Diagrams
		July 21st, 2019	• Develop version 1 of the Mobile App User Interfaces using Adobe XD
July 23rd, 2019	July 29th, 2019	July 23rd, 2019	• Develop skeleton for the Final Report
		July 24th, 2019	• Finalize Sequence Diagrams
		July 24th, 2019	• Finalize User Interfaces
		July 26th, 2019	• Finalize all remaining diagrams with remaining details
		July 26th, 2019	• Develop the Software Design using contracts

		July 28th, 2019	• Finalize the Software Design using contracts
July 30th, 2019	August 4th, 2019	August 2nd, 2019	• Finalize project management data into the final report
		August 4th, 2019	• Finalize and submit the final report

EXECUTION TIMELINE:

The Execution Timeline below shows the actual execution of the tasks. Deadlines throughout the weeks fluctuated based on the other courses, internships, and shifting priorities.

Week		Task (s) Completed
May 27th, 2019	June 2nd, 2019	• Group Sign-Up and formation • Exchanged phone numbers and emails to create a WhatsApp group for collaboration • Set up Skype for Business to establish conference calls • Created a shared OneDrive to share documents and diagrams for the project
June 3rd, 2019	June 9th, 2019	• View Sample Project Reports from previous semester • Research world problems and list outs project idea • Decide on a project idea as group
June 10th, 2019	June 17th, 2019	• Get the project idea approved • Assign the project manager as Kevin Sy • Develop version 1 of the Product Name, Mission Statement, Executive Summary, Problem Statement, Scope, and Business Objectives • Develop a rough draft of the smaller use cases before consolidating into a larger and more generalized (Team effort)
June 18th, 2019	June 24th, 2019	• Develop finalized version of the more generalized Use Case Diagram • Develop version 1 of the Context Diagram • Develop version 1 of the Database Diagram
June 25th, 2019	July 1st, 2019	• Develop version 1 of the Data Dictionary • Develop version 1 of the Database Constraints • Develop version 2 of the Database Diagram
July 2nd, 2019	July 8th, 2019	• Practice homework problems • Study for Midterm
July 9th, 2019	July 15th, 2019	• Develop version 1 of the Use Case Descriptions • Develop version 1 of the Data Model / Class Diagram without Methods • Develop version 1 of the Functional and Non-

		Functional Specifications Document • Finalize Database Diagram to version in Visual Paradigm • Finalize Use Case Descriptions
July 16th, 2019	July 22nd, 2019	• Finalize and convert the Context Diagram to a version in Visual Paradigm • Finalize the Data Dictionary • Finalize the Data Model / Class Diagram without Methods • Develop version 1 of the Complete Class Diagram with Methods • Finalize the Functional and Non-Functional Specifications Document • Develop version 1 of the Choreography Diagram • Develop version 1 of the Sequence Diagrams • Develop version 1 of the Mobile App User Interfaces using Axure • Develop skeleton for the Final Report
July 23rd, 2019	July 29th, 2019	• Finalize all remaining diagrams • Develop and finalize the Software Design using contracts • Finalize project management data into the final report
July 30th, 2019	August 4th, 2019	• Finalize and submit the final report

MEETING MINUTES

Time Frame:	June 13, 2019: 9:30 PM to 10:30 PM CST
Meeting Type	Group Collaboration and Planning
Attendees	Amey Saste, Anamika Soni, Monika Malik, Sanjana Buchala, Kevin Sy
Summary of Meeting Points	• Get to know the team via Skype for Business • List out as many project ideas based on online research • Provide analysis of cost, benefits, and risks of each project idea
Conclusion	• Continue online research to generate new project ideas

Time Frame:	June 20, 2019: 9:00 PM to 10:00 PM CST
Meeting Type	Group Collaboration and Planning
Attendees	Amey Saste, Anamika Soni, Kevin Sy
Summary of Meeting Points	• Vote on the project idea • Brainstorm the App Name, Mission Statement, Problem Statement, and Scope

Conclusion	• Continue brainstorm the App Name, Mission Statement, Problem Statement, and Scope • Generate Use Case Diagrams

Time Frame:	June 22, 2019: 9:30 PM to 10:30 PM CST
Meeting Type	Group Collaboration and Planning
Attendees	Amey Saste, Monika Malik, Sanjana Buchala, Kevin Sy
Summary of Meeting Points	• Collaborate on the App Name, Mission Statement, Problem Statement, and Scope • Collaborate on smaller Use Case Diagrams
Conclusion	• Continue to collaborate on smaller Use Case Diagrams so that the team can create a more generalized Use Case Diagram later

Time Frame:	June 27, 2019: 7:30 PM to 9:30 PM CST
Meeting Type	Group Collaboration and Building Diagrams
Attendees	Sanjana Buchala, Kevin Sy
Summary of Meeting Points	• Review and revise the version 1 of the Context Diagram • Review and revise the version 1 of the Database Diagram
Conclusion	• Version 1 of the Context Diagram and Database Diagram finished

Time Frame:	July 9th, 2019: 9:30 PM to 10:30 PM CST
Meeting Type	Group Collaboration and Building Diagrams
Attendees	Monika Malik, Kevin Sy
Summary of Meeting Points	• Review and revise the version 1 of the Data Dictionary
Conclusion	• Version 1 of the Data Dictionary is finished. • The Data Dictionary will be revised again to be organized by Use Case

Time Frame:	July 10th, 2019: 9:00 PM to 10:30 PM CST
Meeting Type	Group Collaboration and Building Diagrams
Attendees	Anamika Soni, Kevin Sy
Summary of Meeting Points	• Review and revise version 1 of the general Use Case Diagram • Meeting communication done entirely through WhatsApp
Conclusion	• Version 1 of the general Use Case Diagram is finished for most part

Time Frame:	July 13th, 2019: 9:00 PM to 10:30 PM CST
Meeting Type	Group Collaboration and Building Diagrams

Attendees	Sanjana Buchala, Kevin Sy
Summary of Meeting Points	• Review and revise version 1 of the Data Model (or Class Diagram without the Methods) • Meeting communication done entirely through WhatsApp
Conclusion	• Version 1 of the Data Model Diagram is finished for most part

Time Frame:	July 22, 2019: 9:00 PM to 10:30 PM CST
Meeting Type	Group Collaboration and Planning for the User Interface Design
Attendees	Amey Saste, Anamika Soni, Monika Malik, Kevin Sy
Summary of Meeting Points	• Because Monika has some experience with Adobe XD for prototype design, the team is going to proceed with using Adobe XD prototyping the user interface. The team is going try out this for next day and tomorrow and will then finalize the design and will start working. • We will get one day for hands on experience with learning Adobe XD. • Anamika, Kevin, and Amey have reviewed the Use Case Descriptions. Anamika is going to review it more thoroughly and Amey will create 2 more use cases.
Conclusion	• Must learn Adobe XD within 2 days • Must complete the User Interface Designs • Must finalize the Use Case Descriptions • Keep the Data Dictionary consistent with the Use Case Descriptions

Time Frame:	July 24th, 2019: 9:30 PM to 10:30 PM CST
Meeting Type	Group Collaboration for User Interface
Attendees	Kevin Sy, Amey Saste
Summary of Meeting Points	Outlined the remaining UI that need to be designed
Conclusion	Delegated the remaining UI tasks to Monika Malik, Anamika Soni, and Amey Saste

Time Frame:	August 3rd, 2019: 9:00 PM to 11:00 PM CST
Meeting Type	Group Collaboration and Final Report
Attendees	Kevin Sy, Monika Malika, Anamika Soni, Amey Saste, Sanjana Buchala
Summary of Meeting Points	Review and revise the final report
Conclusion	Submit the final report