Login

Primary Actor: The User (VT Student/Staff member)

Stakeholders/Interests: The User (VT Student/Staff member) must be able to securely authenticate into the App and manage the account.

Preconditions: The User (VT student/Staff member) has an account associated with the App and has valid login credentials.

Success Guarantees (Post-Conditions):

1. The User (VT student/Staff member) is successfully able to login and manage the Application accordingly.
2. The User (VT student/Staff member) must be able to sign off/log off the Application securely from their device.

Main Success Scenario:

1. The User (VT student/Staff member) opens the Application from a mobile phone or a website from PC.
2. If registered or newly registered, a VT student/staff member provides the credentials (username/email and password) to login.
3. The Application delegates user authentication to the VT service that hosts the account and authorizes the user to grant access to their account.
4. The user successfully opens their dashboard/profile.

Extensions (Alternate Scenarios):

1. VT student/Staff member enters invalid login details
    a. If Username is invalid, the App/Webpage requests to re-enter the username
    b. If Password is invalid, the App/Webpage requests to re-enter the password
    c. When both username and passwords are valid, then proceed to step 3 on Main Success Scenario
2. VT student forgets the password

a. If a VT student/Staff member forgets the password, the App/Webpage prompts to enter the associated email address to send a password change request link

Special Requirements: The App/Website must be able to reject user authentication/registration if the user is not a student/staff member of Virginia Tech University.

Technology and Data Variations List: The App/Website must use Google OAuth / Passport Authenticator to authenticate valid email IDs with @vt.edu domain. The App/Website must try to avoid crashing down.

Frequency of Occurrence: Occurrences of this functional requirement can be rare, as the devices can save the user credentials securely.

Use Case Diagram: Included with Register functional requirement.

Conceptual Class Diagram: Included with Register functional requirement.

Register

Primary Actor: The User (VT Student/Staff member)

Stakeholders and interests: The User (VT Student/Staff member) must be able to securely Register as a User of the Application or website.

Preconditions: The User is a registered Virginia Tech Student or a Staff Member and has a VT domain (@vt.edu) email and password setup with a Hokie ID.

Success Guarantees (Post-Conditions):

1. The User (VT student/Staff member) is successfully able to Register as a user and get the confirmation email.
2. The User (VT student/Staff member) is successfully able to create a profile and get the login credentials

Main Success Scenario:

1. The User (VT student/Staff member) opens the Application from a mobile phone or a website from PC.
2. If not registered as a user, VT student provides their associated Virginia Tech Email, confirms account and sets a password
3. If registered or newly registered, a VT student/staff member provides the credentials (username/email and password) to login.
4. The Application delegates user authentication to the VT service that hosts the account and authorizes the user to grant access to their account.
5. The user successfully opens their dashboard/profile.

Extensions (Alternate Scenarios):

1. VT student/Staff member enters invalid login details (not a valid email address domain @vt.edu)
    a. If email is invalid, the App/Webpage requests to re-enter their associated VT email
    b. If Password is invalid, the App/Webpage requests to re-enter the password associated with VT email

c. When both username and passwords are valid, then proceed to step 4 on Main Success Scenario

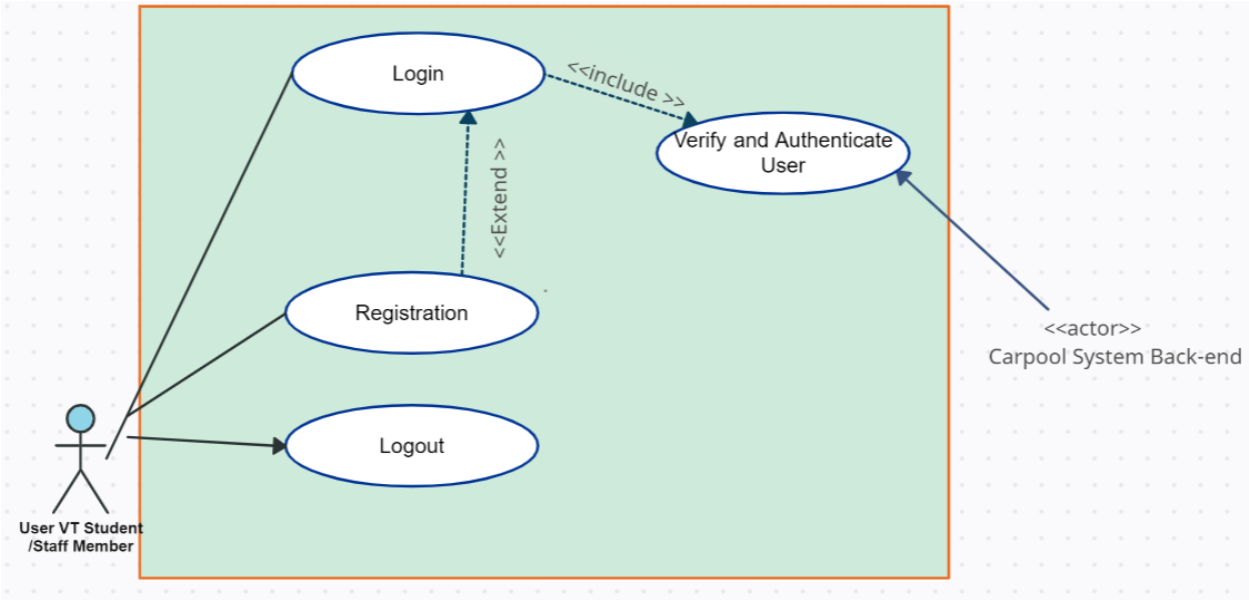
2. VT student enters invalid Hokie ID associated with the email.
    a. The system prompts the user to enter a valid Hokie ID number associated with their VT email

Special Requirements: The App/Website must be able to reject user authentication/registration if the user is not a student/staff member of Virginia Tech University.

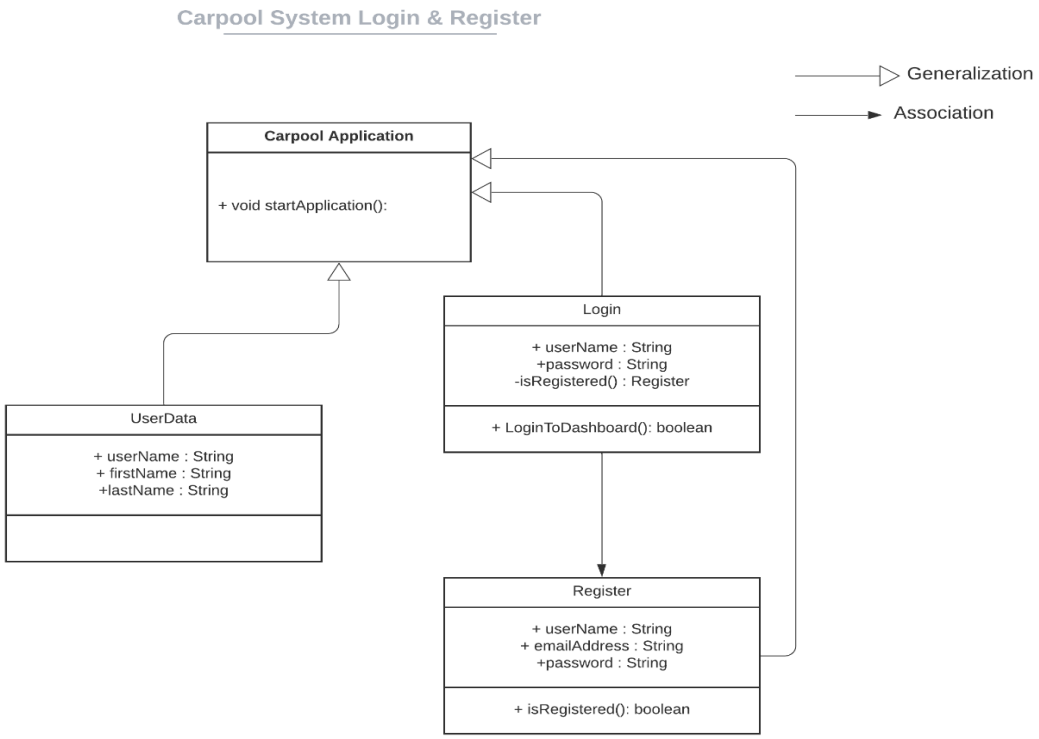
Technology and Data Variations List: The App/Website must use Google OAuth / Passport Authenticator to authenticate valid email IDs with @vt.edu domain.The App/Website must try to avoid crashing down, insecure login and unnecessary UI additions.

Frequency of Occurrence: There can only be one occurrence of this functional requirement as the users will only need to register once.

Use Case Diagram: Login and Register Functional Requirements



Conceptual Class Diagram:



Create/edit/delete profile & logout

Primary Actor: The User (VT Student/Staff member)

Stakeholders and interests: The User (VT Student/Staff member) must be able to Create, Edit or Delete their profile and must be able to securely log off from the device associated.

Preconditions: The User (VT Student/Staff member) must be successfully authenticated and logged in to view their dashboard.

Success Guarantees (Post Conditions):

1. The User (VT Student/Staff member) is able to make necessary changes i.e., creating a profile, updating their profile information or deleting the profile information. The App/Website must be able to save the changes made
2. The User (VT Student/Staff member) can successfully log off the application after saving the changes.

Main Success Scenario:

1. The User (VT Student/Staff member) is successfully logged in and can view the profile
2. The User (VT Student/Staff member):
    a. Creates a new profile if he/she does not have one
    b. Makes necessary changes to the already existing profile
    c. Deletes unnecessary content of the profile
    d. Saves the necessary changes and logs out successfully

Extensions (Alternate Scenarios):

1. The User (VT Student/Staff member) is not logged in with a registered account.
    a. App/Website proceeds to prompt user for account creation with VT email, confirm account and set password and continue to #1 of the Main Success Scenario.
2. The User (VT Student/Staff member) cannot make changes to the profile

a. An error report must be sent, the issue must be tackled by the server of the webpage/Application.

Special Requirements: In case of a website/App crash event, or if the user is unable to save the changes to the website/App, it must directly report the issue to the technical support and resolve.

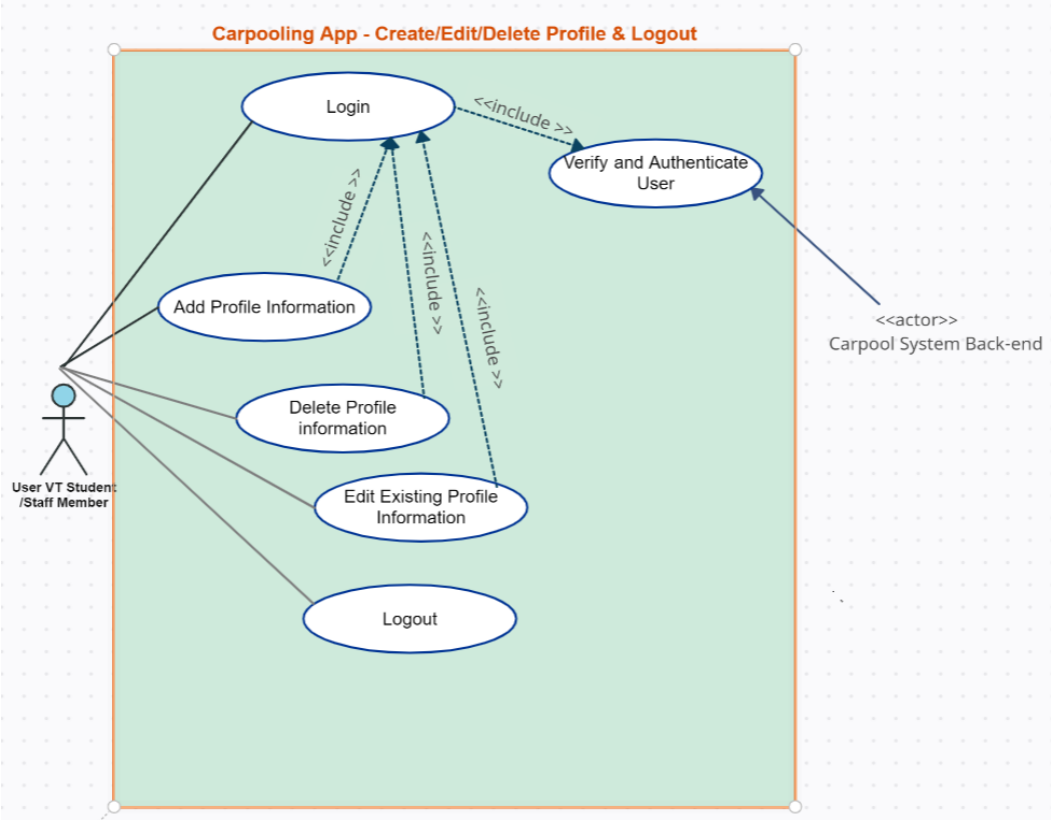
Technology/Data Variations List:

The App/Website must use Google OAuth/Passport authenticator to successfully authenticate a VT user (Student or Staff Member) and log in to their profile.

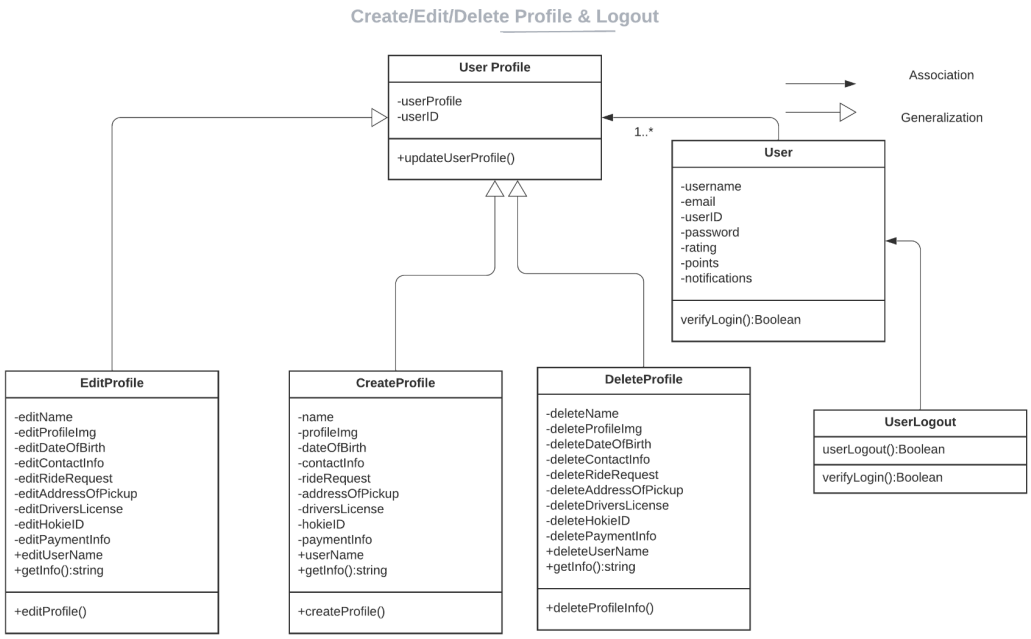
The App/Website must use Node.js with MongoDB to perform CRUD (Create, Update and Delete) operations on the user profile.

Frequency Of Occurrence: Creating and deleting the profile can only be done once. Editing the profile can occur multiple times.

Use Case Diagram:



Conceptual Class Diagram:



Ability to post a Trip Listing to the community

Primary Actor: The User (VT Student/Staff member)

Stakeholders and interests: The User (VT Student/Staff member) must be able to post a ride request (riders) or a driving destination (drivers) to the App/Website.

Preconditions:

1. The User (VT Student/Staff member) must be successfully authenticated and logged in to their dashboard.
2. The User (VT Student/Staff member) must have a completed profile with attached bank/Venmo accounts to process the payment
3. The Users, who are Drivers, must have their driver's license verified and updated.

Success Guarantees (Post Conditions):

1. The User (VT Student/Staff member) who is a Registered Driver must be able to post a Driving destination, a date, time and place of pickup to the App/Website's community of Virginia Tech users
2. The User (VT Student/Staff member) who is a Registered Rider must be able to post a Request for a particular destination, a date and time of availability to pickup to the VT App/Website's community of Virginia Tech Users.

Main Success Scenario:

1. The User (VT Student/Staff member) is successfully logged in and can view the profile
2. The User (VT Student/Staff member) who is also a Driver with a created profile with updated Hokie ID
    a. Uploads a copy of the driver's license and get it verified
    b. Posts a driving route, the destination, date and time of availability
    c. Saves the necessary changes.
    d. If ride is accepted, the User will be able to communicate with the rider
3. The User (VT Student/Staff member) who is also a Rider with a created profile with updated Hokie ID

a. Searches for the Users who posted trips to their required destination and requests if any ride is found.
b. Posts a Ride Request to the VT community of users, the destination and time of availability to pickup
c. If notified of the ride, the User gets to accept the ride and communicate with the driver

Extensions (Alternate Scenarios):

1. The User (VT Student/Staff member) is not logged in with a registered account.
    a. App/Website proceeds to prompt users for account creation with VT email, confirm account and set password and continue to 1 of Main Success Scenario.
2. The User (VT Student/Staff member) cannot make changes to the profile
    a. An error report must be sent, the issue must be tackled by the server of the webpage/Application.

Special Requirements: In case of a website/App crash event, or if the user is unable to save the changes to the website/App, it must directly report the issue to the technical support and resolve.

Technology/Data Variations List:

The App/Website must use Node.js with MongoDB to perform CRUD (Create, Update and Delete) operations on the user posts

Frequency Of Occurrence: A user can post multiple trips, so the functional requirement can occur multiple times.

Use Case Diagram: Included with Ability to Edit/Delete a Listing in the community Functional Requirement.

Conceptual Class Diagram: Included with Ability to Edit/Delete a Listing in the community Functional Requirement.

Ability to Edit or Delete a Listing in the community

Primary Actor: The User (VT Student/Staff member)

Stakeholders and interests: The User (VT Student/Staff member) must be able to edit/delete a ride request (riders) or edit/delete a driving destination (drivers) to the App/Website.

Preconditions:

1. The User (VT Student/Staff member) must be successfully authenticated and logged in to their dashboard.
2. The User (VT Student/Staff member) must have a completed profile with attached bank/Venmo accounts to process the payment
3. The Users, who are Drivers, must have their driver's license verified and Hokie ID updated.
4. The Users, who are Riders, must have their Hokie ID updated

Success Guarantees (Post Conditions):

1. The User (VT Student/Staff member) who is a Registered Driver must be able to edit an existing Driving destination i.e., can be able to edit the date, time and place of pickup to the App/Website's community of Virginia Tech users
2. The User (VT Student/Staff member) who is a Registered Driver must be able to delete an existing Driving destination
3. The User (VT Student/Staff member) who is a Registered Rider must be able to edit a Ride Request for a particular destination, i.e., the date and time of availability to pickup, the destination required to the VT App/Website's community of Virginia Tech Users
4. The User (VT Student/Staff member) who is a Registered Rider must be able to edit an existing Trip Request

Main Success Scenario:

1. The User (VT Student/Staff member) is successfully logged in and can view the profile

2. The User (VT Student/Staff member) who is also a Driver with a created profile with updated Hokie ID and Driver's License
    a. Edits an existing a driving route, the destination, date and time of availability to the community
    b. Saves the necessary changes.
    c. If notified and the request is accepted, the User will be able to communicate the trip details with the rider
3. The User (VT Student/Staff member) who is also a Rider with a created profile with updated Hokie ID and updated payment
    a. Searches for the Users who posted trips to their required destination and requests if any ride is found.
    b. Edits an existing Ride Request to the VT community of users. Can edit the destination and time of availability to pickup
    c. If notified of the ride, the User gets to accept the ride and communicate with the driver

Extensions (Alternate Scenarios):

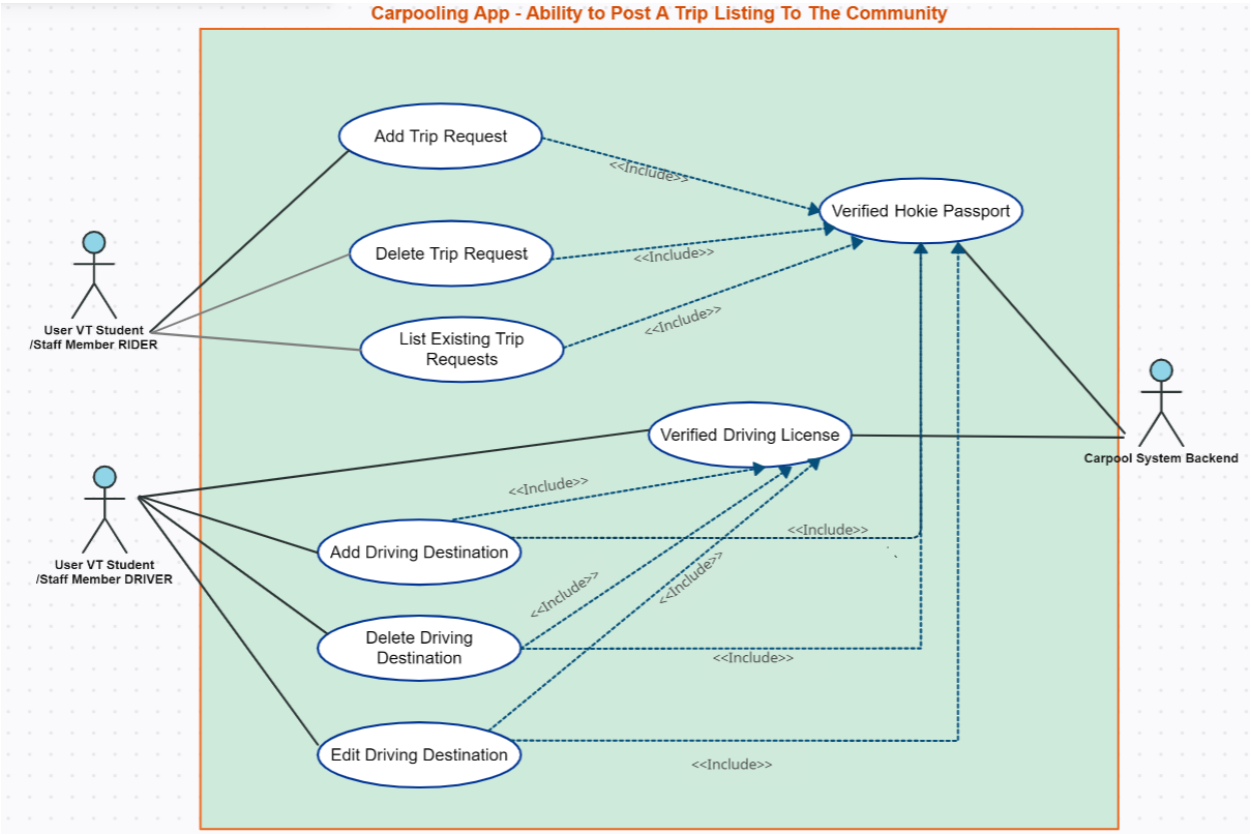
1. The User (VT Student/Staff member) is not logged in with a registered account.
    a. App/Website proceeds to prompt users for account creation with VT email, confirm account and set password and continue to 1 of Main Success Scenario.
2. Extension 2 - The User (VT Student/Staff member) cannot make edits necessary for the ride
    a. An error report must be sent, the issue must be tackled by the server of the webpage/Application.

Special Requirements: In case of a Website/App crash event, or if the user is unable to save the changes to the Website/App, it must directly report the issue to the technical support and resolve.

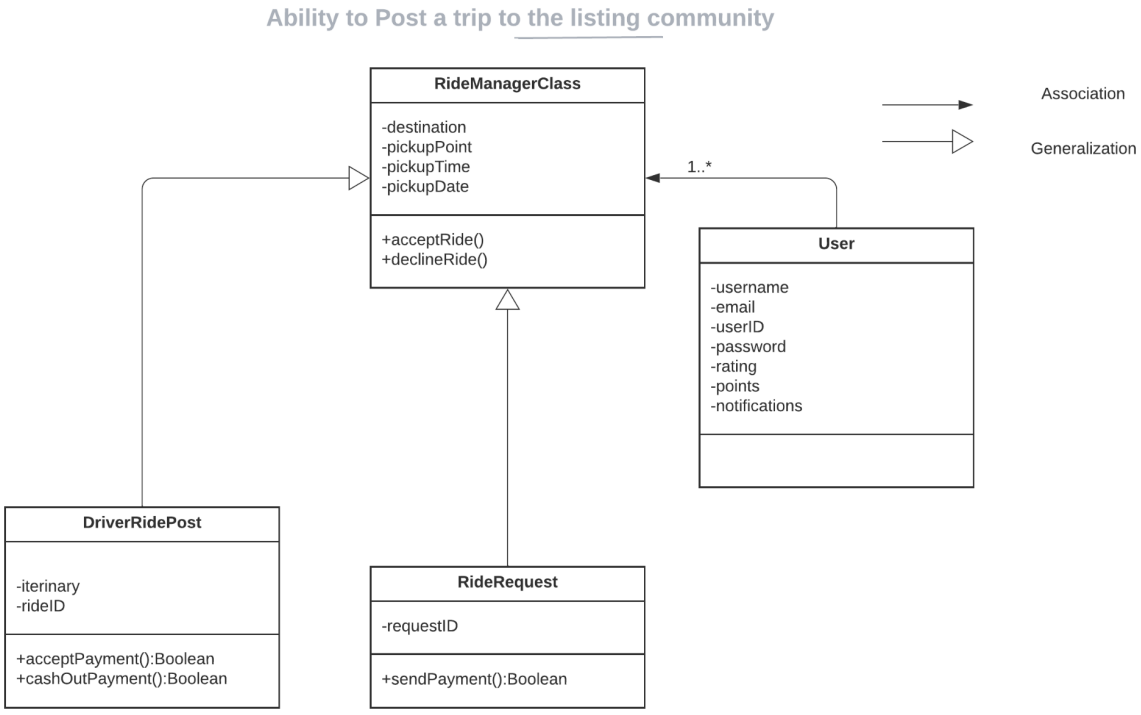
Technology/Data Variations List: The App/Website must use Node.js with MongoDB to perform CRUD (Create, Update and Delete) operations on the user ride requests or destination posts

Frequency Of Occurrence: A user can edit or delete multiple trips, so the functional requirement can occur multiple times.

Use Case Diagram: Add Listing & Edit/Delete Listing Combined



Conceptual Class Diagram:



Sorting Listings

Primary Actor: Driver

Stakeholders/Interests: Drivers must be able to view available listings based on various sorting algorithms.

Preconditions: The system must have pending listings that are awaiting a driver to be assigned.

Success Guarantee: All available listings are sorted based on the criteria the driver selects. One example is a driver sorts the listing by closest distance to pick up location and the system would order the listings from top match to worst match.

Main Success Scenario:

Driver initiates the sorting algorithm based on a specific criteria.

System considers each pending listing and begins defining the order.

Driver adds another filter to the sorting algorithm and step 2 is repeated.

System displays listings in order based on the filters the driver has chosen. The listing at the top will be the top match out of all listings.

Extensions:

1. Driver initiates the sorting algorithm based on a specific criteria.
    a. No listings are currently pending (No rides have been requested in the area).
        i. System responds to the lack of listings and displays a message informing the driver that no active listings are available.
2. System considers each pending listing and beings defining the order
    a. Two listings hold the same weight with respect to a specific filter.
        i. Since only one filter is established, the system will default to list the most recent listing first.
3. Driver adds another filter to the sorting algorithm and step 2 is repeated.
    a. Two listings hold the same weight with respect to a specific filter.

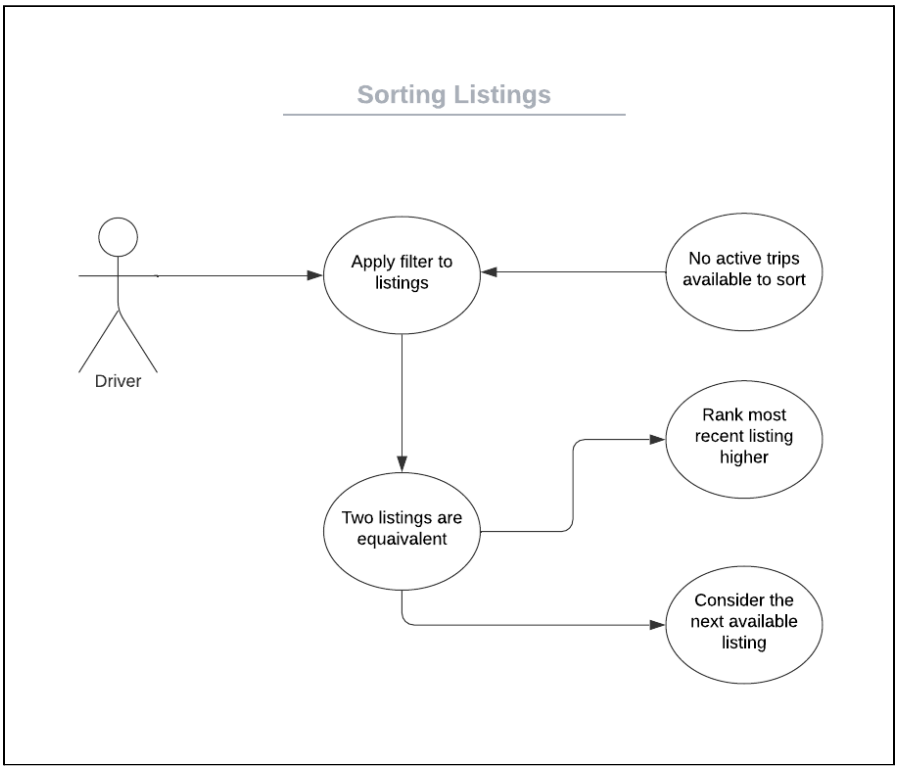
i. As multiple filters are established, the system will consider the next available filter to establish a hierarchy for the listings.

Special Requirements: The system will display listings within a few seconds of commencing the algorithm. The time required to sort will increase proportionally with the number of active listings. The more listings in the system the longer the sorting algorithm may take. If the system takes longer than 30 seconds a warning message will appear indicating that the sorting algorithm is running.

Technology/Data Variation List: The sorting algorithm implemented will have an efficiency of at least O(log n) to speed up system processing speeds.

Frequency of Occurrence: Occurrences of this functionality is very common as drivers will likely run filters before accepting any available listings.

Use Case Diagram:



Class Diagram: Shown in UML diagram for email confirmation

Transactions

Primary Actors: Rider & Driver

Stakeholders/Interests: Riders must be able to pay for their rides and drivers should be able to receive payment through the transaction functionality.

Preconditions: The rider must have already requested a ride by specifying pickup and drop off location and an available driver in the vicinity must have accepted the listing.

Success Guarantee: If the use case is successful then funds, either monetary or points, from the riders account will have been transferred to the driver. The driver picks up the rider and drops him or her off at their destination.

Main Success Scenario:

1. Rider has requested a ride and the driver has accepted the listing.
2. The total number of points are calculated for the trip.
3. Points are deducted from the rider's account and sent to the driver.
4. Driver picks up the rider and provides transportation to the destination.

Extensions:

1. Rider has requested a ride and the driver has accepted the listing.
    a. Rider cancels the trip before the transaction can occur.
        i. Transaction is canceled and the driver is informed.
2. The total number of points are calculated for the trip.
    a. Invalid destination.
        i. Rider requests a trip with a path that is not possible or requires an unacceptably large number of points.
3. Points are deducted from the rider's account and sent to the driver.
    a. Point balance is less than the amount due.
        i. Rider is given the option to add points with monetary funds
        ii. Transaction is canceled and the driver is informed.

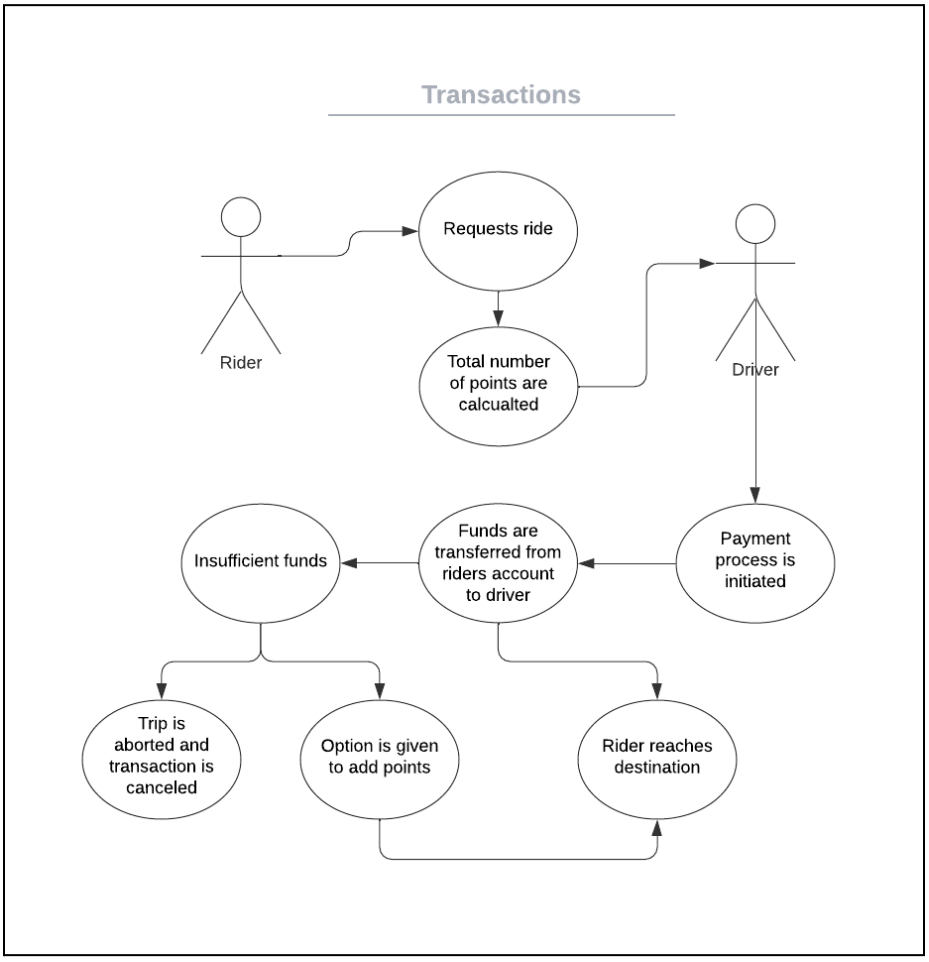
Special Requirements: The system is capable of handling a variety of payment

options, the major one being the point currency system. Furthermore, the system should also allow credit card, venmo, paypal, and other electronic payments. The system should also be able to handle refunds if any issues arise, such as a driver never showing up or showing up very late.

Technology/Data Variation: Electronic payment receipts and transaction histories will be made available to both the drivers and riders to maintain proper records of trips.

Frequency of Occurrence: The transaction functionality will have a very high occurrence frequency as a transaction is required for every single trip dispatched on the app.

Use Case Diagram:



Class Diagram: Shown in UML diagram for email confirmation

Forgot Password

Primary Actor: Driver & Rider

Stakeholders/Interests: Users must be able to reset the password to their account regardless of whether they are a driver or a rider.

Preconditions: User is locked out of their account from not remembering their login credentials.

Success Guarantee: The user is emailed a link to reset his or her password. Once this is done users will regain full access to their account and resume use of the service.

Main Success Scenario

1. User attempts to gain access to the service by logging in.
2. Credentials are invalid and access is not granted.
3. User clicks the forgot password button.
4. Users are asked to enter the email address associated with their account.
5. User is sent an email with a link to reset their account.
6. Passwords are reset and users are able to login.

Extensions:

1. Users are asked to enter the email address associated with their account.
    a. Invalid email address
        i. The email address passed to the app is not found in the database and a recovery email is not sent
2. User is sent an email with a link to reset their account.
    a. Invalid password
        i. User attempts to change the password to an invalid passphrase that does not meet the requirements or may have already been used.

Special Requirements: Users should not be able to reset the password to a

password already used in the past. Passwords should be at least a certain number of characters long and have a certain amount of complexity.

Technology/Data Variation: Passwords may not be pasted into the text field but rather typed.

Frequency of Occurrence: This functionality should not occur frequently, but rather only when necessary. Most users will rarely use this feature.

Use Case Diagram: Shown in the email confirmation use case diagram below.

Class Diagram: Shown in UML diagram for email confirmation

Email Confirmation

Primary Actor: Drivers & Riders

Stakeholders/Interests: Users will receive email notification and confirmation following actions taken on the app such as dispatchment of a trip of resetting a password.

Preconditions: An action must have taken place in the app to trigger this functionality. However, this functionality may also be triggered due to routine information updates.

Success Guarantee: Following an action on the app the user is sent an email containing all relevant information about the action and any next steps if user input is required.

Main Success Scenario:

1. User commences an action or operation on the app.
2. User receives an email for his or her records.
3. User input and next steps are taken if necessary.

Extensions:

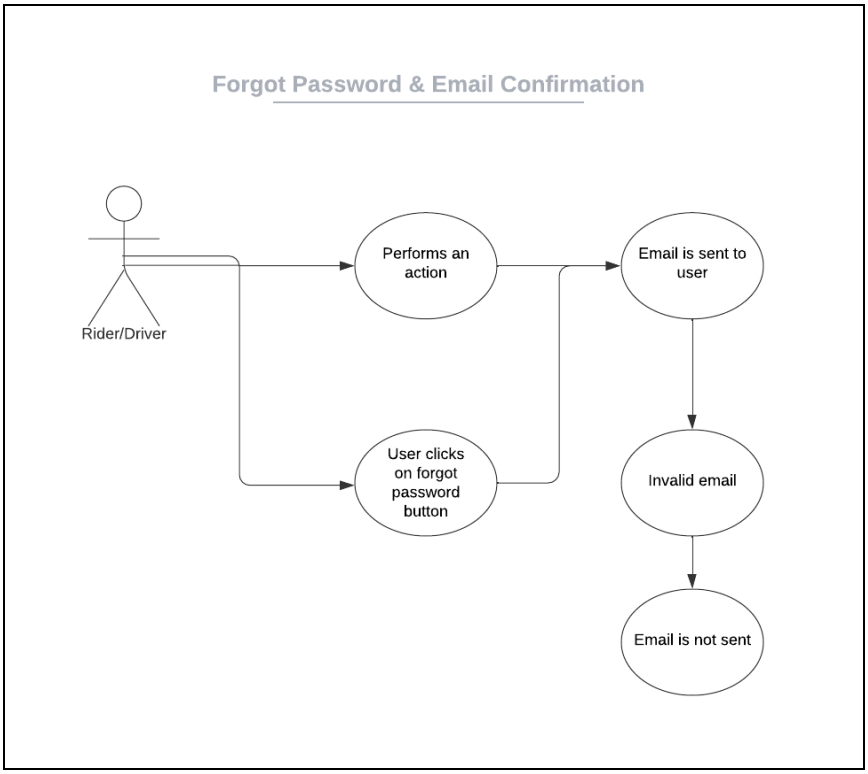
1. User receives an email for his or her records.
    a. Invalid email address
        i. User provided a invalid email address when signing up or email account has been deleted resulting in email not being sent

Special Requirements: The email will be sent to the user in a timely manner. For example, an email to reset a password will be sent within minutes and expire after a preset amount of time.

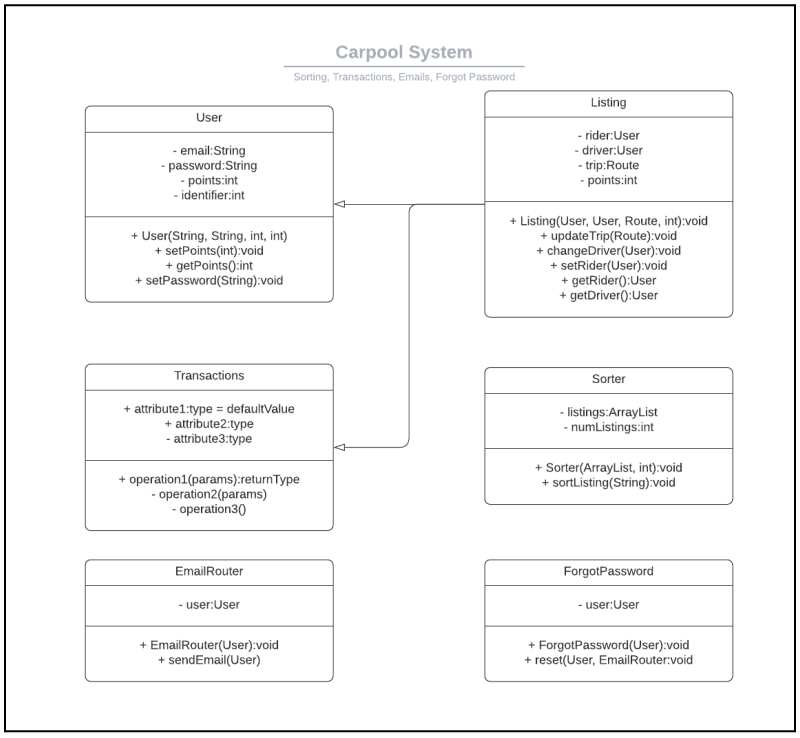
Technology/Data Variation: Access to the email is the only form of verification but 2 factor verification will be added at a later time to confirm the user's identity.

Frequency of Occurrence: This functionality will be very frequent as at the minimum an email will be sent as a receipt every time a trip is dispatched.

Use Case Diagram:



Class Diagram:



Trip Duration Estimate

Primary Actor: Rider

Stakeholders/Interests: Riders should be able to see how long it will take to reach their destination, to see if the ride is worth it and/or plan accordingly on when to start the trip.

Preconditions: The system must know where the trip starts, where it ends, and when the trip will start.

Success Guarantee: Rider is able to see trip duration after entering the destination on the app, before ordering the trip. Drivers can also see the trip duration before accepting to drive.

Main Success Scenario

1. Rider opens the app and sets the start location as his/her current location or another location to be picked up from.
2. Rider enters the destination.
3. App displays the trip duration estimate using previous trip data.
4. Rider and driver both can see the estimate and before choosing to ride/drive.

Extensions

1. Rider opens the app and sets the start location as his/her current location or another location to be picked up from.
    a. Rider is a confirmed VT student
    b. Location is within or near the campus
2. Rider enters the destination.
    a. Destination is within or near the campus.
3. App displays the trip duration estimate using previous trip data.
    a. Estimates are most accurate on common routes, like from the library to the nearest neighborhood.
    b. Estimates change based on the time of day and traffic patterns.
4. Rider and driver both can see the estimate and before choosing to ride/drive.
    a. Both parties have the option to decline the trip after seeing the

estimate.

b. Driver sees the trip only after the rider orders it.

Special Requirements

Estimates only works for locations within a 10 mile radius around campus since this service is only for Virginia Tech students. Service being available to far locations from campus will saturate the driver pool for those who need a ride for short trips.

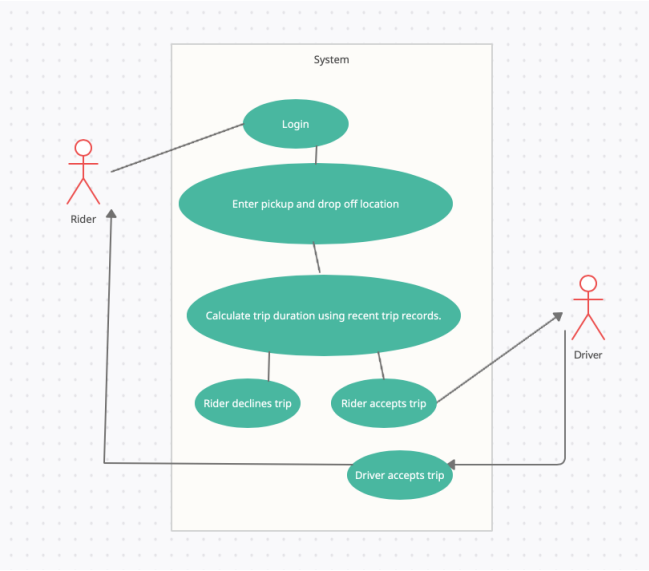
Technology/Data Variation

The algorithm will search recent trips with similar beginning and ending locations to give a proper estimate. This can be done with a LRU cache or a Dijkstra table. For trips without recent data, we will have an algorithm that takes two points to give an estimate using local traffic patterns.

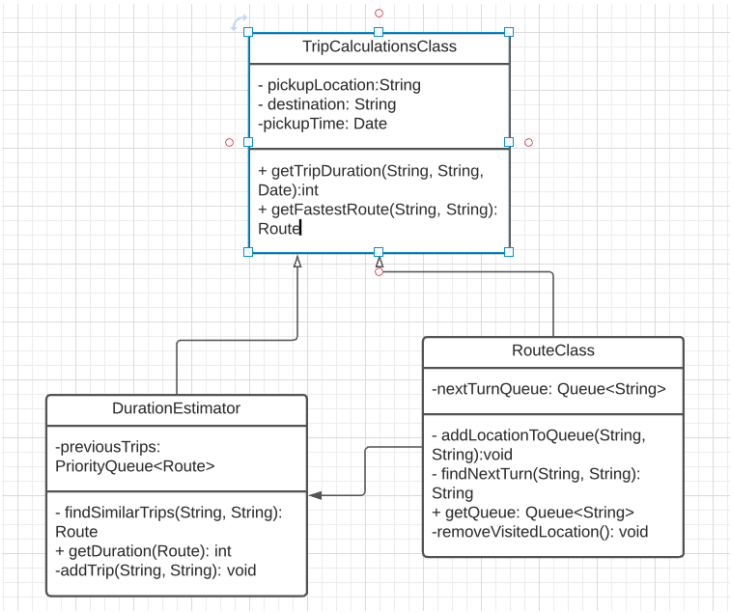
Frequency of Occurrence

Will be used for every trip since the estimate will always be given before the rider confirms the trip. Drivers will also get the estimate before each trip so they can decide if they have enough time to accept a trip for a certain amount of time.

Use Case Diagram:



Class Diagram:



Route Planning

Primary Actor: Driver

Stakeholders/Interests: Driver should get the shortest route to the rider's destination to have a shorter trip and save gas.

Preconditions: Destination is within range. Shortest routes must account for traffic accidents and local traffic.

Success Guarantee: Drivers get the shortest route to have the shortest route. Riders get the fastest route so they can get to the destination as fast as possible.

Main Success Scenario

1. Rider logs on the app and enters destination after entering pickup point.
2. Driver accepts the trip.
3. App gives the driver the fastest route based on time of day and traffic patterns to pick up riders.
4. App then gives the most efficient route again, but to the rider's destination.

Extensions

1. Rider logs on the app and enters destination after entering pickup point.
    a. Pickup and destination are both in range of campus.
2. Driver accepts the trip.
    a. Driver accepted the trip duration estimate.
3. App gives the driver the fastest route based on time of day and traffic patterns to pick up riders.
    a. Route is the shortest time wise, rather than the shortest distance.
4. App then gives the most efficient route again, but to the rider's destination.
    a. Route is the shortest time wise, rather than the shortest distance.

Special Requirements

Fastest routes can only be given to locations within range of campus, otherwise they will not be in the database to be able to calculate the fastest route.

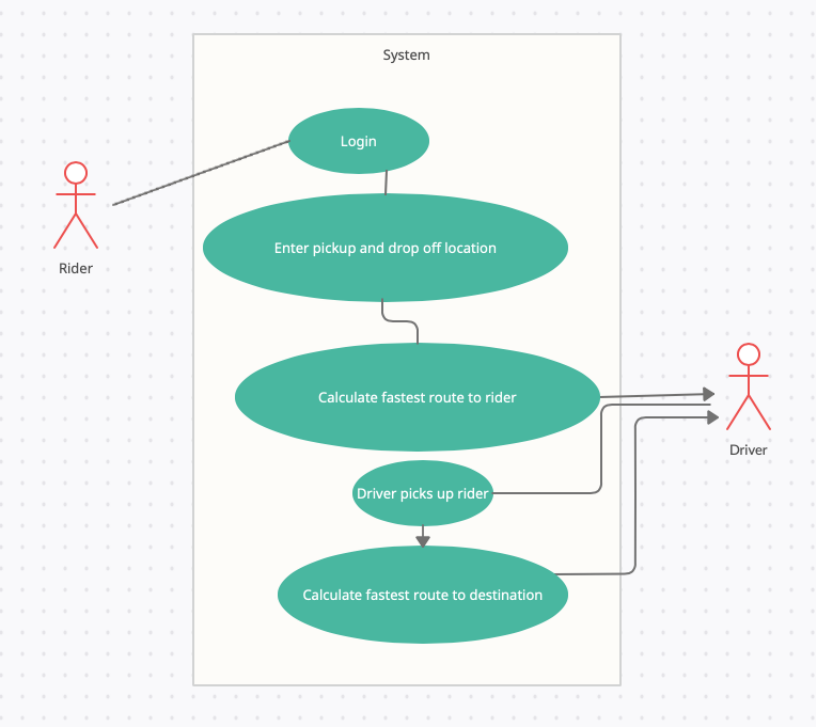
Technology/Data Variation

To calculate the fastest route, a Dijkstra Table will be used with the pickup and dropoff locations as the end points and common landmarks, like stores and classroom buildings in Blacksburg, as midpoints to calculate routes and determine which would be the fastest.

Frequency of Occurrence

The shortest route will be used at least twice for each trip. Once for the driver to get to the rider's pickup point and another time to get the rider to his/her destination.

Use Case Diagram:



Class Diagram: Same as the diagram used for trip duration estimate.

Pickup Scheduling

Primary Actor: Rider

Stakeholders/Interests: Riders should be able to choose a certain time of day for a ride rather than the current time. This is to have a higher chance of finding a ride for that time, especially if that is a very busy time that the number of drivers does not equate to the number of riders, even with ride shares.

Preconditions: Time selected for a future ride falls within a reasonable hour.

Success Guarantee: Success not guaranteed since drivers still need to accept the trip; however, riders are more likely to find a ride since drivers have more time to view the trip request and plan to make the trip at that time.

Main Success Scenario

1. Rider chooses a time for pickup in the future.
2. Rider enters the pickup location and destination.
3. Drivers can view this request from the time it is posted, all the way until the time of the pickup.
4. Once a driver accepts the request, the rider is notified they have a ride.

Extensions

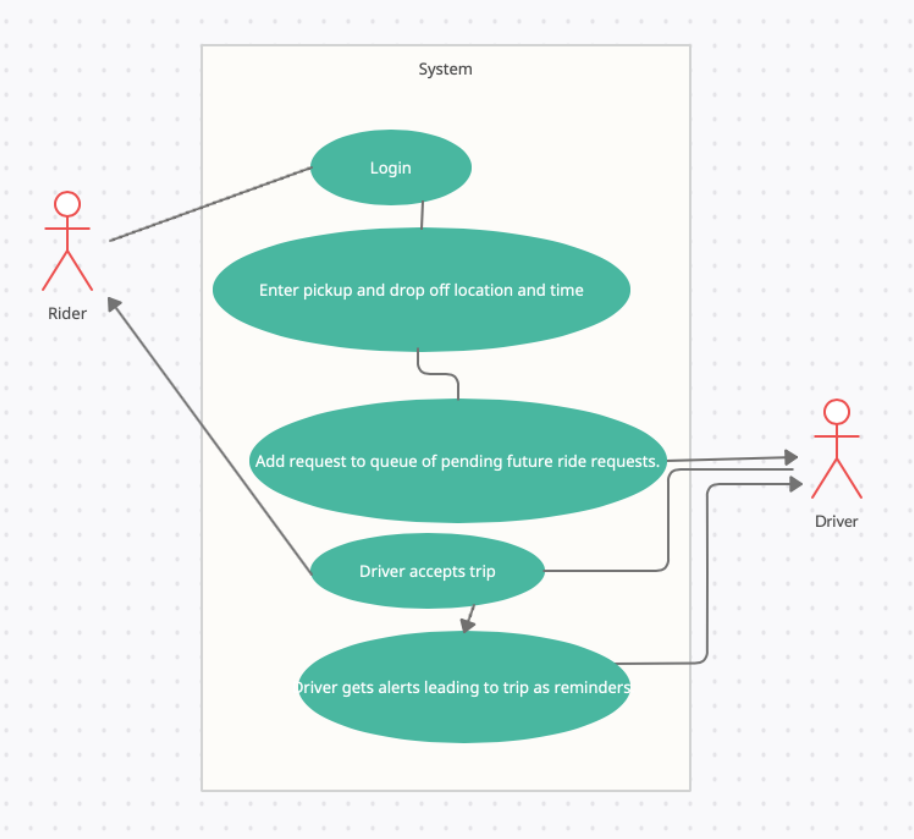
1. Rider chooses a time for pickup in the future.
    a. The time must be within a reasonable hour for a higher chance of getting a driver.
2. Rider enters the pickup location and destination.
    a. Pickup and destination must be within range of campus.
3. Drivers can view this request from the time it is posted, all the way until the time of the pickup.
    a. Requests will reach more drivers as the deadline for asked time draws nearer.
4. Once a driver accepts the request, the rider is notified they have a ride.
    a. The driver will be given an alert for the future ride so they do not forget about this prior commitment.

Special Requirements: Pickup times can only be scheduled for the future. Time for pickup must be at a reasonable hour.

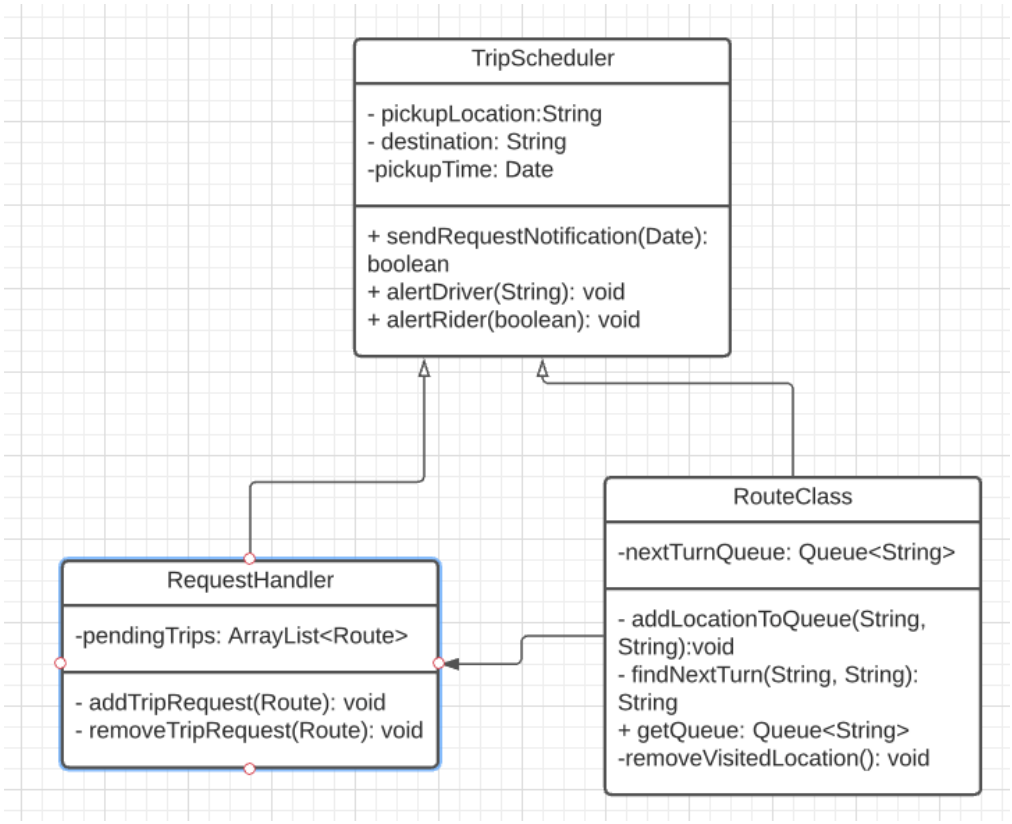
Technology/Data Variation: To keep track of future rides, the requests will be stored in a queue with the nearest time requests at the front.

Frequency of Occurrence: Assuming that most trips will be requested for the current time, the pickup scheduler will be used sparingly.

Use Case Diagram:



Class Diagram:



Trip Group Messaging

Primary Actor: Driver

Stakeholders/Interests: Driver should be able to effectively communicate ride details with all riders associated with this trip

Preconditions: The system must know where the trip starts, where it ends, and when the trip will start.

Success Guarantee: Driver is able to message all riders or one rider privately. Riders are able to receive messages and respond to the driver or the entire group.

Main Success Scenario:

1. Driver opens up chat for the specific ride listing with riders
2. Driver sends a message to selected recipients
3. All recipients are able to view and respond to the message
4. Read and delivered receipts are available for both parties to verify

Extensions:

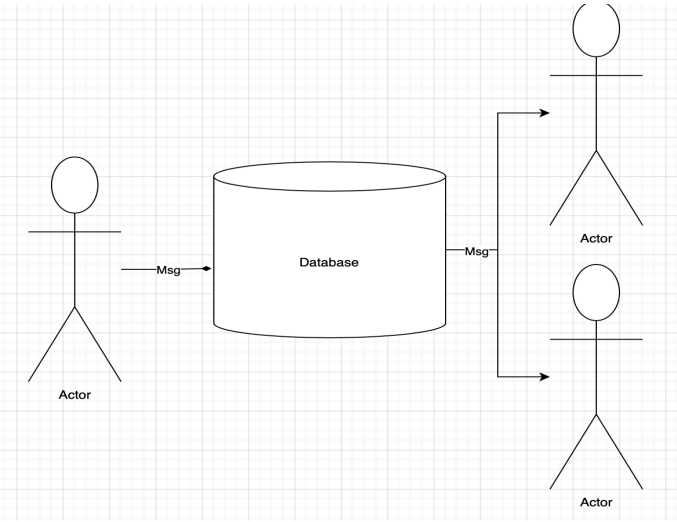
1. Driver opens up chat for the specific ride listing with riders
    a. Valid listing exists with riders and trip route
2. Driver sends a message to selected recipients
    a. Message meets community guidelines
    b. Message does not exceed maximum size
3. All recipients are able to view and respond to the message
    a. Message meets community guidelines
    b. Message does not exceed maximum size
4. Read and delivered receipts are available for both parties to verify
    a. Both parties have proper device privacy settings
    b.

Special Requirements: The system will allow messages to be sent both ways. If the sending of a message takes longer than 15 seconds It will be assumed that there was a network error. The request will time out and the user will have a chance to try again and be prompted for failure.

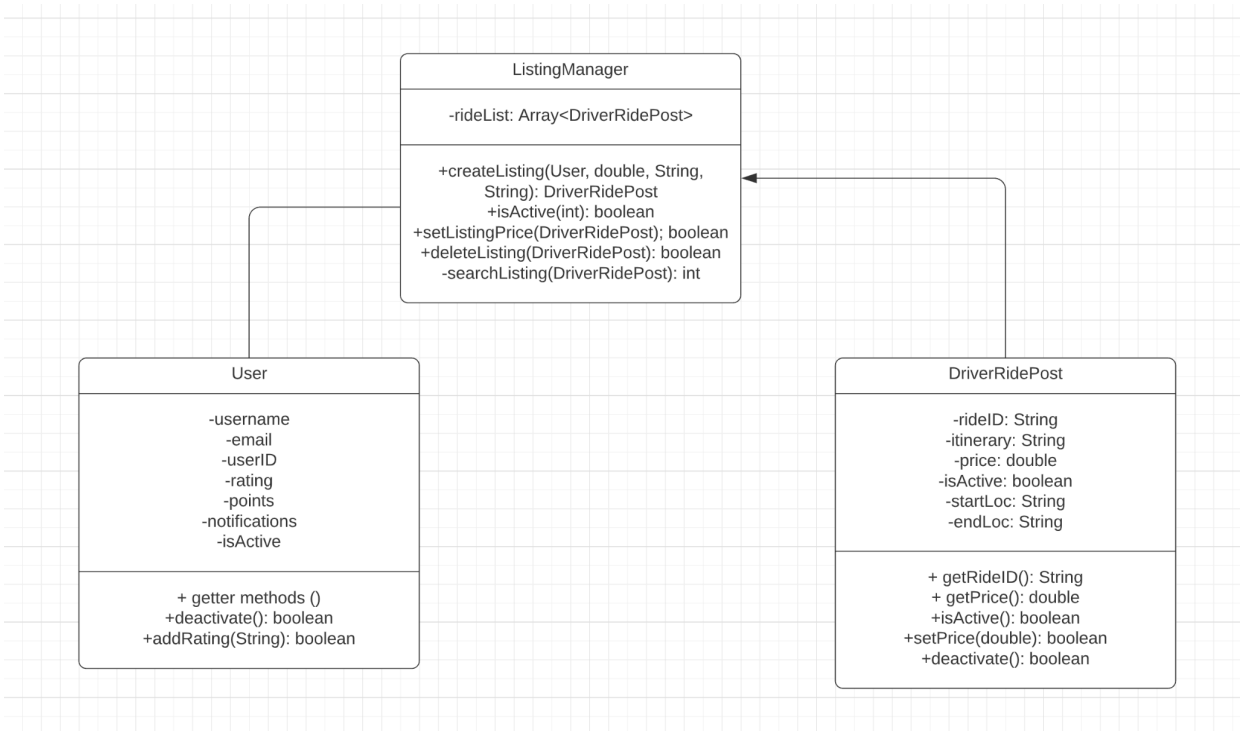
Technology/Data Variation List: The websockets technology will be used for instant messaging.

Frequency of Occurrence: Occurrences of this functionality is very common as drivers and rider will need to communicate to make their ride happen smoothly

Use Case Diagram:



Class Diagram:



Search/Filter Listings

Primary Actor: Rider

Stakeholders/Interests: Riders should be able to see how long it will take to reach their destination, to see if the ride is worth it and/or plan accordingly on when to start the trip. Riders should be able to search or filter according to their desired trip parameters.

Preconditions: The system must know where the trip starts, where it ends, and when the trip will start. The system should also know all the metadata associated with a trip listing.

Success Guarantee: Riders will be able to use the search/filter feature to look through all listings on the platform. The interface should report when a search/filter yields no results.

Main Success Scenario:

1. Rider open up to view current trip listings
2. Rider filters down listings to match desired start time
3. Rider searches for a friends listing by driver name
4. Rider accepts the listing offered by his friend

Extensions:

5. Rider open up to view current trip listings
    a. Listings are present on the platform
6. Rider filters down listings to match desired start time
    a. Listings are present on the platform with the desired start time
    b. Entered filtered parameters meet the requirements are valid
7. Rider searches for a friends listing by driver name
    a. The listing with the desired driver name exists
8. Rider accepts the listing offered by his friend
    a. Listing has space for another rider

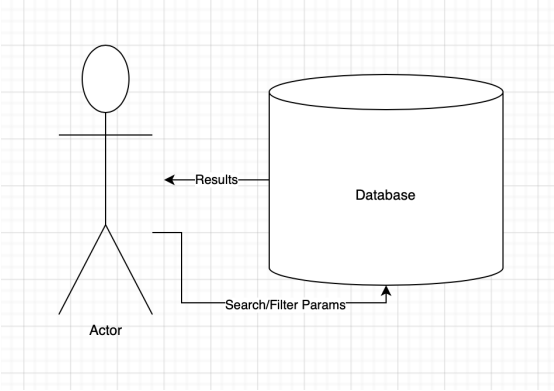
Special Requirements: The system will allow search and filtration of the rides database if there was a network error. The request will time out and the user will

have a chance to try again and be prompted for failure.

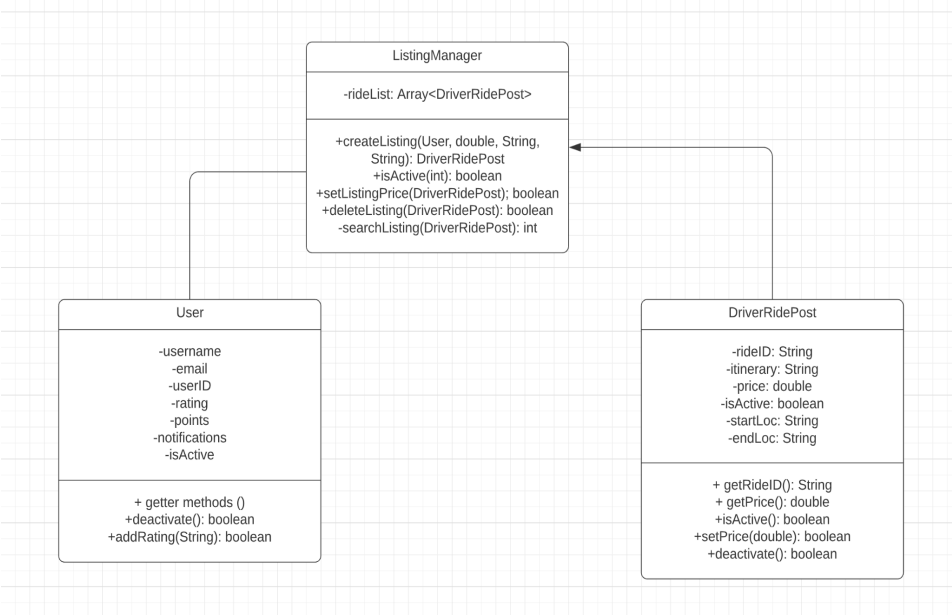
Technology/Data Variation List: An http api interface connected to a database will create this functionality.

Frequency of Occurrence: Occurrences of this functionality is very common as drivers and rider will need to communicate to make their ride happen smoothly

Use Case Diagram:



Class Diagram:



Share Trip

Preconditions: The system must know where the trip starts, where it ends, and when the trip will start. The system should also know all the metadata associated with a trip listing.

Success Guarantee: Riders will be able to share any listing on the platform. The interface should report when a share yields no results.

Main Success Scenario:

1. Rider open up to view current trip listings
2. Rider filters down listings to match desired start time
3. Rider searches for a friends listing by driver name
4. Rider uses share feature to send trip to a contact

Extensions:

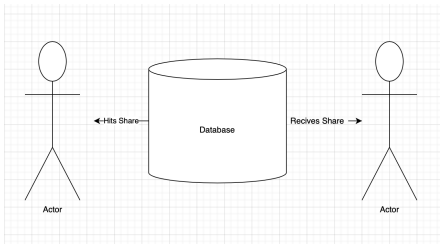
1. Rider open up to view current trip listings
    a. Listings are present on the platform
2. Rider filters down listings to match desired start time
    a. Listings are present on the platform with the desired start time
    b. Entered filtered parameters meet the requirements are valid
3. Rider searches for a friends listing by driver name
    a. The listing with the desired driver name exists
4. Rider uses share feature to send trip to a contact
    a. Contact is a valid receiver of share info

Special Requirements: The system will allow sharing of the rides database if there was a network error. The request will time out and the user will

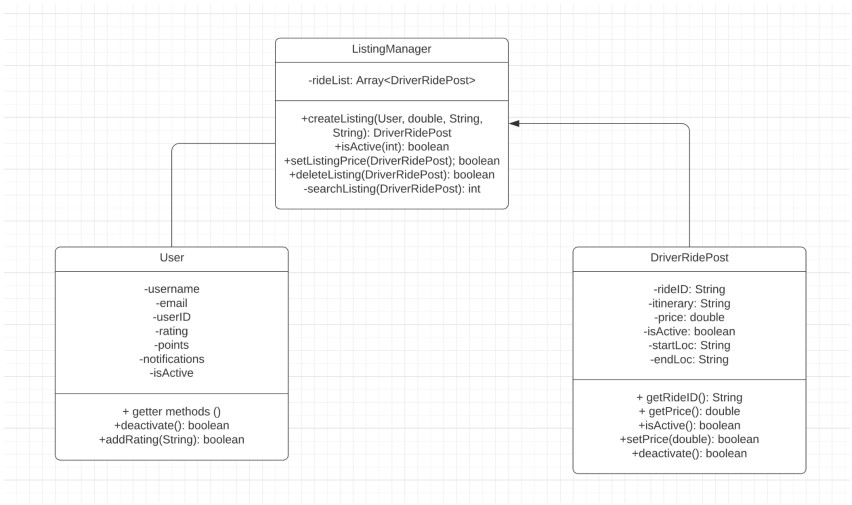
Technology/Data Variation List: An http api interface connected to a database will create this functionality.

Frequency of Occurrence: Occurrences of this functionality is very common as drivers and rider will need to communicate to make their ride happen smoothly

Use Case Diagram:



Class Diagram:



Listing Notifications

Preconditions: The system must know where the trip starts, where it ends, and when the trip will start. The system should also know all the metadata associated with a trip listing.

Success Guarantee: Riders will be able to use the search/filter feature to look through all listings on the platform. The interface should report when a search/filter yields no results.

Main Success Scenario:

1. Rider open up to view current trip listings
2. Rider filters down listings to match desired start time
3. Rider searches for a friends listing by driver name
4. Rider enables notifications for the listing
5. Device pushes notifications to the user

Extensions:

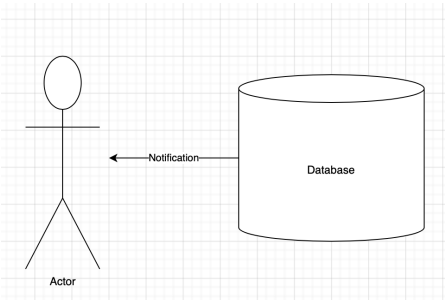
1. Rider open up to view current trip listings
    a. Listings are present on the platform
2. Rider filters down listings to match desired start time
    a. Listings are present on the platform with the desired start time
    b. Entered filtered parameters meet the requirements are valid
3. Rider searches for a friends listing by driver name
    a. The listing with the desired driver name exists
4. Rider enables notifications for the listing
    a. Rider has proper device settings
5. Device pushes notifications to the user
    a. Device has power and internet connections

Special Requirements: The system will allow push notifications if there was a network error. The request will time out and the user will

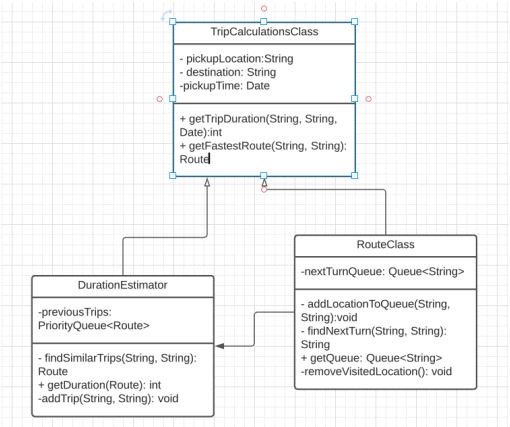
Technology/Data Variation List: An http api interface connected to a database will create this functionality.

Frequency of Occurrence: Occurrences of this functionality is very common as drivers and rider will need to communicate to make their ride happen smoothly

Use Case Diagram:



Class Diagram:



Maps routing

Primary Actor: The Driver (VT Student/Staff member)

Stakeholders/Interests: Drivers should be able to have directions generated promptly. Directions should be an optimal route to get both the driver and rider to their destination promptly. The directions should also take into account accidents, toll roads, and construction.

Preconditions: The riders have been picked up by the driver and the trip has not yet begun.

Success Guarantee: An optimal and safe route has been generated for the driver to follow.

Main Success Scenario:

1. Driver indicates the rider has been picked up
2. System generates directions (takes into account toll road, construction, and other preferences)
3. System displays directions on map and estimated time of arrival (ETA)
4. System updates display as ride progresses
5. Once destination is reached, system notifies driver and passenger

Extensions:

4. System updates display as ride progresses

    a. If delay occurs, system updates time ETA, reroutes for optimal directions, and notifies driver and rider

5. Destination is never reached due to rider/driver cancellation or car issues

    a. System opens up a prompt for rider and driver to enter detail issues so the system admin can determine the proper way to proceed.

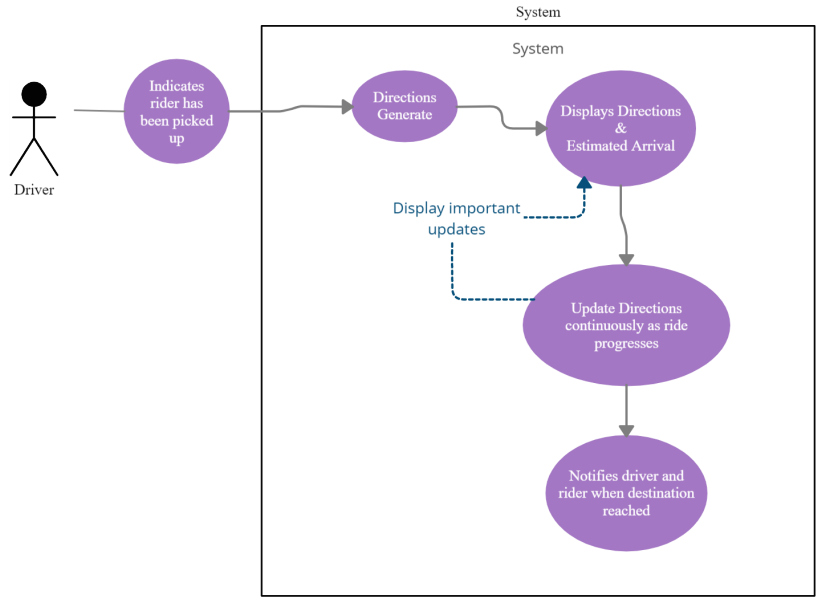
Special Requirements: Directions will be generated within 10 seconds (under ideal internet connectivity for the driver). Directions will attempt to be generated for up

to 1 minute before notifying the driver to attempt generating directions again.

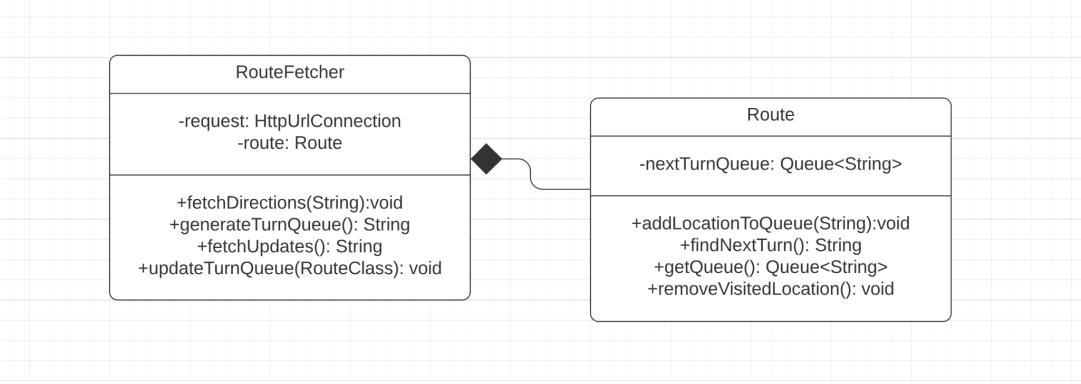
Technology/Data Variation List: Directions will be generated using Google Directions API.

Frequency of Occurrence: Every time a driver picks up a rider, directions for the trip will be generated.

Use Case Diagram:



Class Diagram:



Listing Price updates

Primary Actor: The Driver (VT Student/Staff member)

Stakeholders/Interests: Drivers should be in control of their listings. This means drivers should be able to update listing prices depending on whatever personal factors there may be.

Preconditions: Listing driver is attempting to update has already been published.

Success Guarantee: After submitting a listing price update, the database updates to display the correct pricing to all users. The driver's dashboard should also display these updates.

Main Success Scenario:

1. Driver selects listing to update from driver dashboard
2. System displays edit options
3. Driver selects "update listing price"
4. System prompts driver for updated price
5. Driver enters price and clicks submit
6. System validates price
7. System updates database with new price

Extensions:

5. Driver changes mind about updating price and decides they want to keep it the same

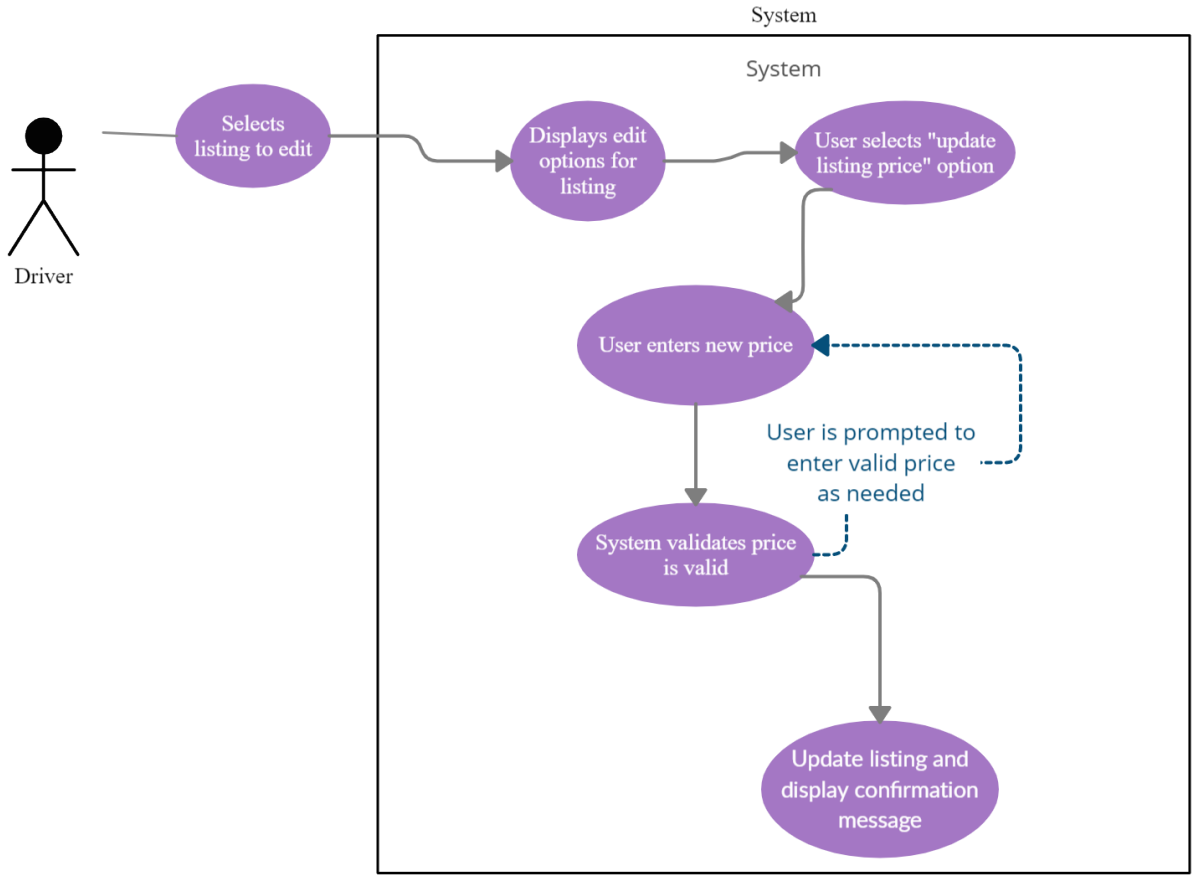
   a. Driver clicks the back button and the price update is canceled.

6. System fails to validate because user entered invalid price (i.e. negative value)

   a. System prompts user for price again and displays message about error
   b. Continue from step 5

Special Requirements: System updates should successfully complete within 10 seconds of submission.

Use Case Diagram:



Class Diagram: Same as UML Class Diagram for Delete Listing

Delete Listing

Primary Actor: The Driver (VT Student/Staff member)

Stakeholders/Interests: Drivers should be in control of their listings. This means drivers should be able to delete listings for whatever reason they may have.

Preconditions: Listing driver is attempting to delete has already been published.

Success Guarantee: After deleting a listing, the listing should no longer be in the database. This means riders will no longer be able to access the listing. Additionally, the driver should no longer be able to see the listing in their dashboard.

Main Success Scenario:

1. Driver selects listing to delete from driver dashboard
2. System displays edit options
3. Driver selects “delete listing”
4. System prompts driver asking if they are sure
5. Driver clicks “yes”
6. System removes listing from database

Extensions:

5. Driver clicks “no”

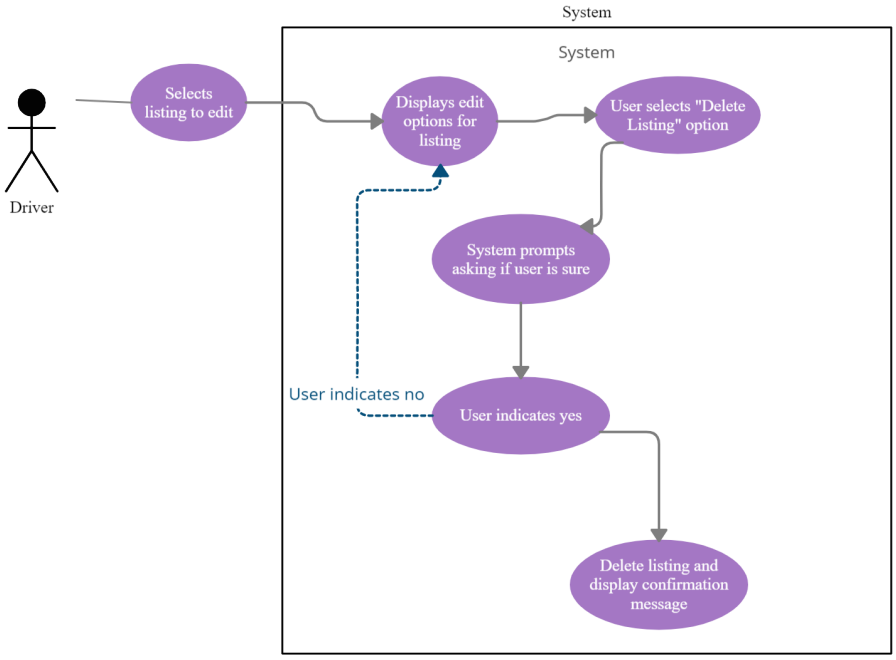
   a. System goes back to listing edit options page

6. System is unable to remove listing from database

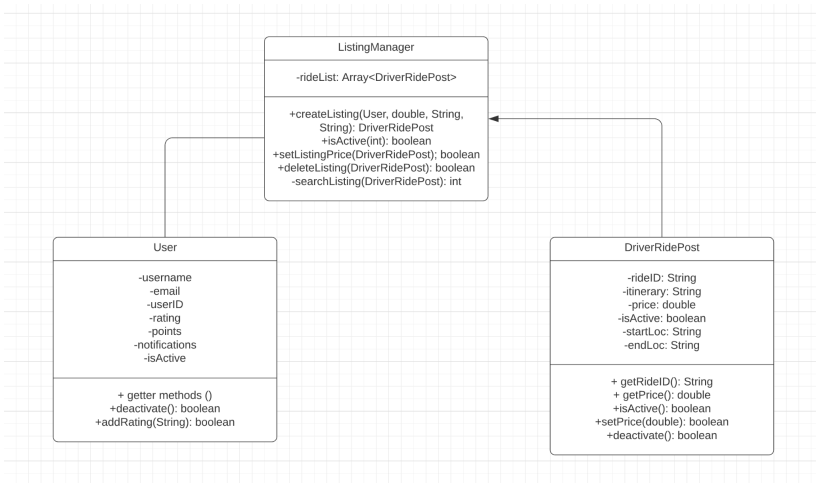
   a. System prompts user to try again and contact support if the issue persists
   b. System goes back to listings edit options page

Special Requirements: Delete listing operation should be complete within 10 seconds 90% of the time and 20 seconds 99.9% of the time.

Use Case Diagram:



Class Diagram:



Delete Profile

Primary Actor: The driver and rider (VT Student/Staff member)

Stakeholders/Interests: Individuals may want to delete their account for various reasons. We value privacy, so upon any profile deletion, we remove all their profile data from our database.

Preconditions: Account being deleted has to exist. User has to be logged in.

Success Guarantee: The profile, and all data relating to it, is deleted. The only thing the database keeps relating to the profile is ride history and rider/driver interactions to allow for proper customer support as needed.

Main Success Scenario:

1. User navigates to settings tab
2. User clicks delete account
3. System prompts user asking if they are sure
4. User clicks “yes”
5. System prompts user that their account will be deleted in 7 days and to contact a support if they change their mind
6. User is locked out of account
7. After 7 days, profile is deleted from the database

Extensions:

4. User clicks “no”

   a. System navigates back to the settings page

5. User changes their mind about deletion

   a. System admin reverts deletion process
   b. System sends password reset email to user so they can access their account again

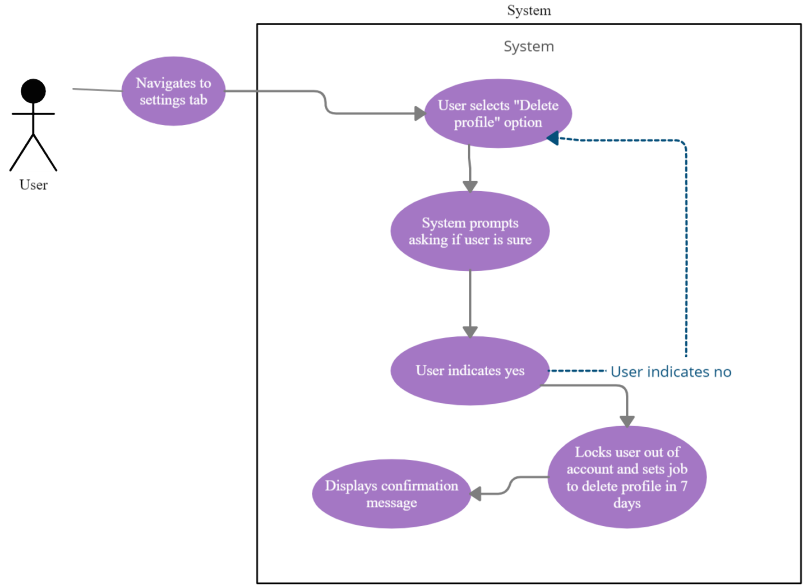
Special Requirements: System should prompt the account is being deleted within

30 seconds of clicking the “yes” button.

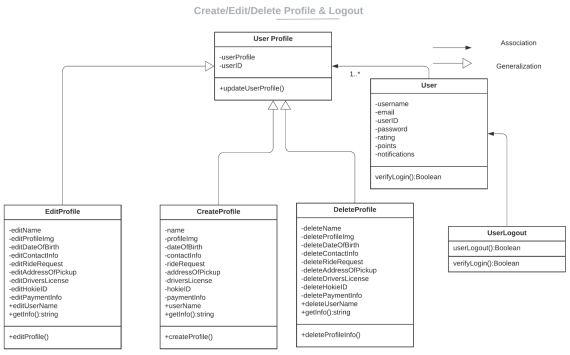
Technology/Data Variation List: Deletion of profiles doesn’t have to be very quick since the user experience isn’t impacted by long deletion times. The only thing that must be quick is the system response that it received the user’s request for account removal.

Frequency of Occurrence: Rarely. Many people who stop using the platform let their account stay alive and inactive.

Use Case Diagram:



Class Diagram:



Supplementary Specifications - Non-functional Requirements

Scalability/Compatibility and Reliability

The User Interface of the Application must be scalable and responsive. The Content should dynamically scale up or down and must fit to any screen size. The User Interface must be tested with repeatable and reproducible results each time. The content must be visible and consistent throughout the screen, it must prevent errors and issues that might rise while the user is interacting.

Security and Privacy

The system should protect and security of all users by withholding certain information until it is absolutely necessary. For example, drivers should not directly be given access to a riders phone number and other information but rather utilize an in app communication feature such as phone or video call. Furthermore, each user's account will be secured with a password and even 2 factor authentication if the user opts to enroll in that program. Drivers will not be able to monitor the users location after drop off and vice versa.

Performance & Maintenance

The system relies on many data structures to pull from previous trips and to also store information about future trips. To make sure the system maintains its speed and efficiency, these structures will have to be cleared periodically to remove least frequently used data to avoid extra, unnecessary searches. This is already what a PriorityQueue does so only frequently needed trip information will be saved. To keep the connection between the rider and the system, along with the driver and the system, fast we will have timeouts if the system does not respond within 20 seconds. This will allow the system a chance to reboot/reconnect without keeping the actors waiting for something that is not working. This is a way to prioritize the actors' time.

Database Backups

The system must be backed up continuously so all user data is safe and the user experience stays seamless. This is necessary to protect from fatal system bugs,

hacked user accounts, and other issues. The backups will take place every night. A cron job will be created in order to facilitate this. We will keep 30 days worth of backups before letting new backups override older ones. Backups must be completed within 1 hour of start time. We will be able to perform system wide resets to old images of the database. We will also be able to perform account resets to old images of individual accounts in case a user requests this (pending system admin approval).

Visual Impairment Accessibility

The system will need to have accessibility features for different users with different visual impairments. Some example features could be the ability to enlarge text or change the color settings to support color blindness. Another possible implementation would be to have dictation for completely blind users. This is our requirement in order to complete accessibility requirements for all the visually impaired users.