Design Document

Kevin Niemiller

Use Case 2: Update Data by System

This diagram was created to show how the system updates Road Activity from the ODOT website. The system runs a loop every 5 minutes, and runs this loop forever. Everytime the loop executes, it executes an ImportData() routine which does a request to the ODOT website to pull the XML file in. This XML file includes all Road Activity. This design uses the Low Coupling GRASP pattern. It uses low coupling because the system does not care what information or the amount of information is being set from ODOT; it only cares that it is an XML so that the system can parse through this information.

Use Case 4: Filter Road Data

This diagram was created to show how the system allows the user to analyze road activity. The user navigates to the Road Activity page which shows all road activity. The user can then set filters for the category type, start date, and end date. Once the user selects these, the Analyze Road Activity sends this request to Road Activity who then sends back the roads (and road information) of what was selected. This diagram is based off of the Information Expert. It uses information expert because the Analyze Road Activity is getting\relying on the information from Road Activity who has all the information.

Use Case 5: Enter Roads Travelled

This diagram was created to show the relationship between TravelPaths, AddTravelPath, RoadName, and TravelPath. The TravelPaths shows all roads that have been added by the current user. The user makes an AddTravelPath request which looks up all possible roads to display to the user. Once the AddTravelPath information is inputted, then this information is submitted by the user which makes AddTravelPath create a TravelPath (with the road information). From here, this TravelPath (and it's information) can be displayed on the TravelPaths for that user. This diagram uses multiple GRASP patterns. This design uses High Cohesion because multiple steps through this Use Case rely on other classes and information within those classes. This design also uses the Creator pattern, because AddTravelPath creates a travel path that houses the new path and all the paths information (such as start mile, end mile, start time, end time, days of the week, etc.). This design also uses Information Expert since it relies on getting the Road Names so the user can pick one of these roads to enter as a road/path.

Use Case 6: Notification System

This diagram was created to show how the Notification System works which is an interaction between the TravelPathAlert and SMTP. The system runs a method every 11 minutes to check to see if there are any new alerts in the TravelPathAlert that the system has not notified the user of. If there is a new alert, this alert is sent from TravelPathAlert to the SMTP with the to and email body information. The SMTP already has the from and subject line. This design uses the Low Coupling GRASP pattern. It uses this pattern because there are two completely separate classes that are responsible for the notification system, and the TravelPathAlert

class only has to send the SMTP a signal to do everything that is necessary to send the email.

Use Case 10: Subscribe to Alerts for Travel Path

This diagram shows the interaction between the User Interface and the Subscription class. The user can subscribe to Travel Path Alerts from the User Interface. When the user does this, it sends its user Id and Boolean true to the Subscription class so the subscription class can subscribe the user to Travel Path Alerts. The design uses the Controller GRASP pattern because the user activates a method from the UI that the Subscription class will store the data to the database that indicates the user has subscribed for Travel Path Alerts.

Use Case 11: Unsubscribe to Alerts for Travel Path

This diagram shows the interaction between the User Interface and the Subscription class. The user can unsubscribe to Travel Path Alerts from the User Interface. When the user does this, it sends its user Id and Boolean false to the Subscription class so the subscription class can unsubscribe the user to Travel Path Alerts. The design uses the Controller GRASP pattern because the user activates a method from the UI that the Subscription class will store the data to the database that indicates the user has unsubscribed for Travel Path Alerts.

Use Case 12: View all Travel Paths

This diagram was created to show how the user interacts with the TravelPaths. The user navigates to the TravelPaths page and is able to view all travel paths that the user has entered. This design uses the Controller GRASP pattern. It uses this pattern because the user will select a page from the view and then the controller grants that request by loading the TravelPaths page along with all the travel paths for that user.

Use Case 13: Edit Travel Paths

This diagram was created to show the relationship between TravelPaths, EditTravelPath, RoadName, and TravelPath. The TravelPaths shows all paths that have been added by the current user. The user makes an EditTravelPath request which looks up all possible paths to display to the user. Once the EditTravelPath information is edited, then this information is submitted by the user which makes EditRoad update the TravelPath (with the road information). From here, this TravelPath (with it's information) can be displayed on the TravelPaths for that user. This diagram uses multiple GRASP patterns. This design uses High Cohesion because multiple steps through this Use Case rely on other classes and information within those classes. This design also uses Information Expert since it relies on getting the Road Names so the user can pick one of these roads to enter as a road/path if they are editing this information and also to call up the travel path the user is wishing to edit so that this information can be pre-populated to make it easier for the user to modify.

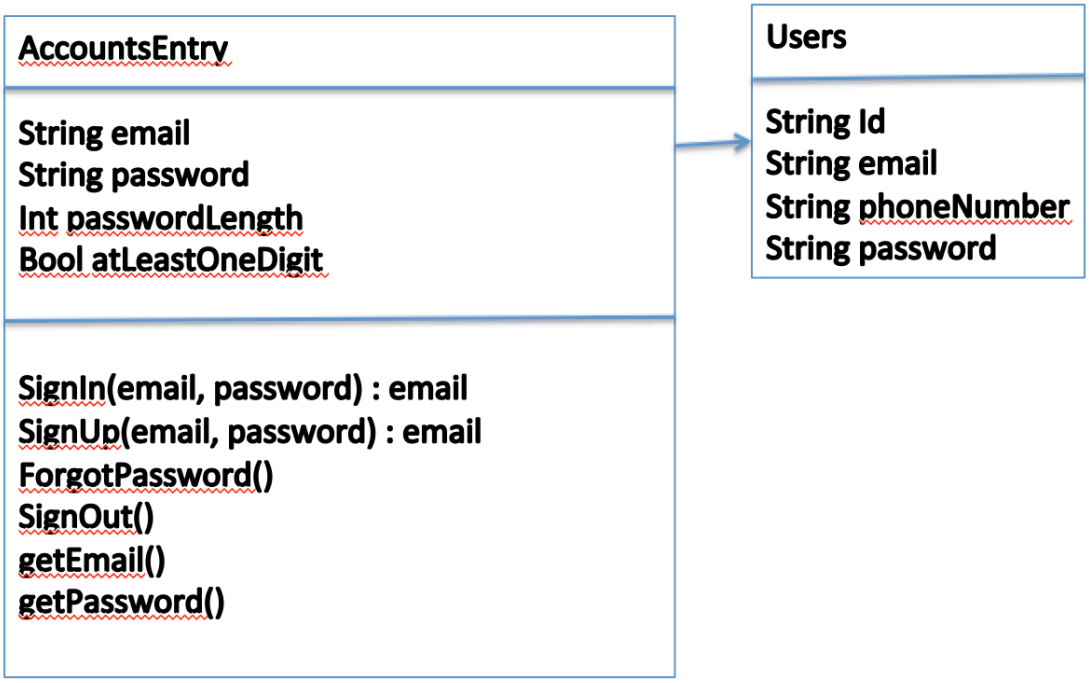
Use Case 14: Remove Travel Path

This diagram shows the interaction between the User Interface and Travel Paths. The user can delete a Travel Path by using the User Interface. When the user does this, the Travel Path Id is sent to the Travel Paths class to remove this Travel Path. The design uses the Controller grasp pattern. It uses this pattern because the user uses the UI to have the controller remove a Travel Path from the database.

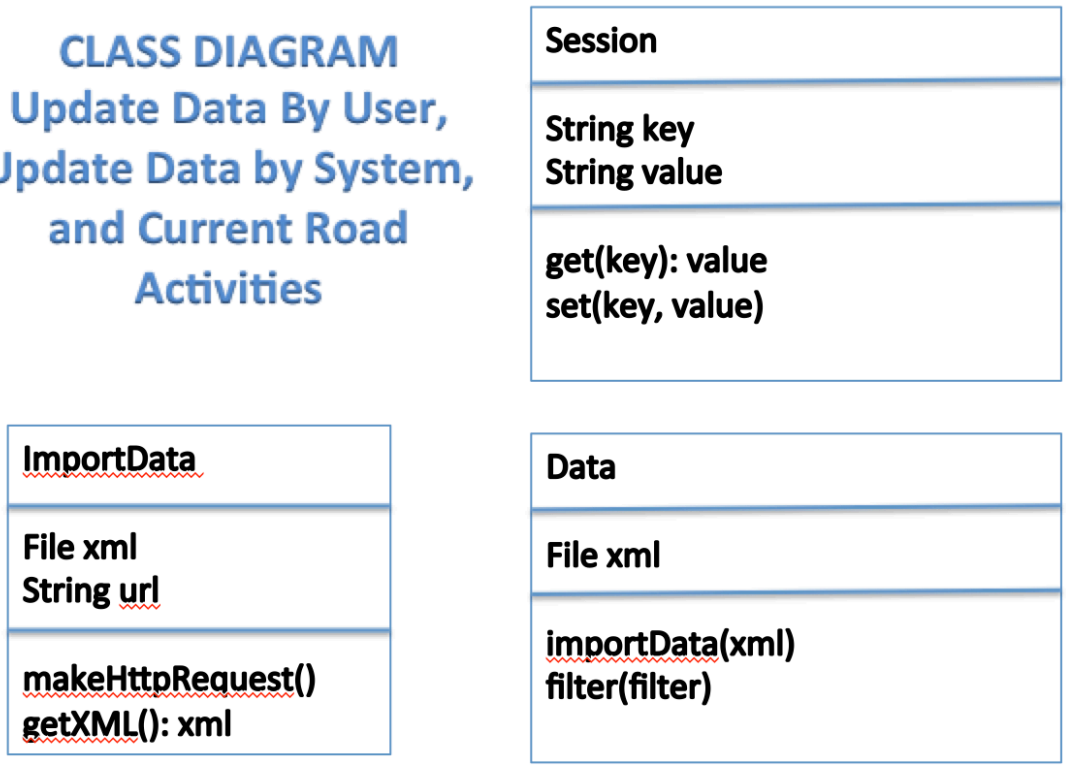
Use Case 15: Analyze Road Activity

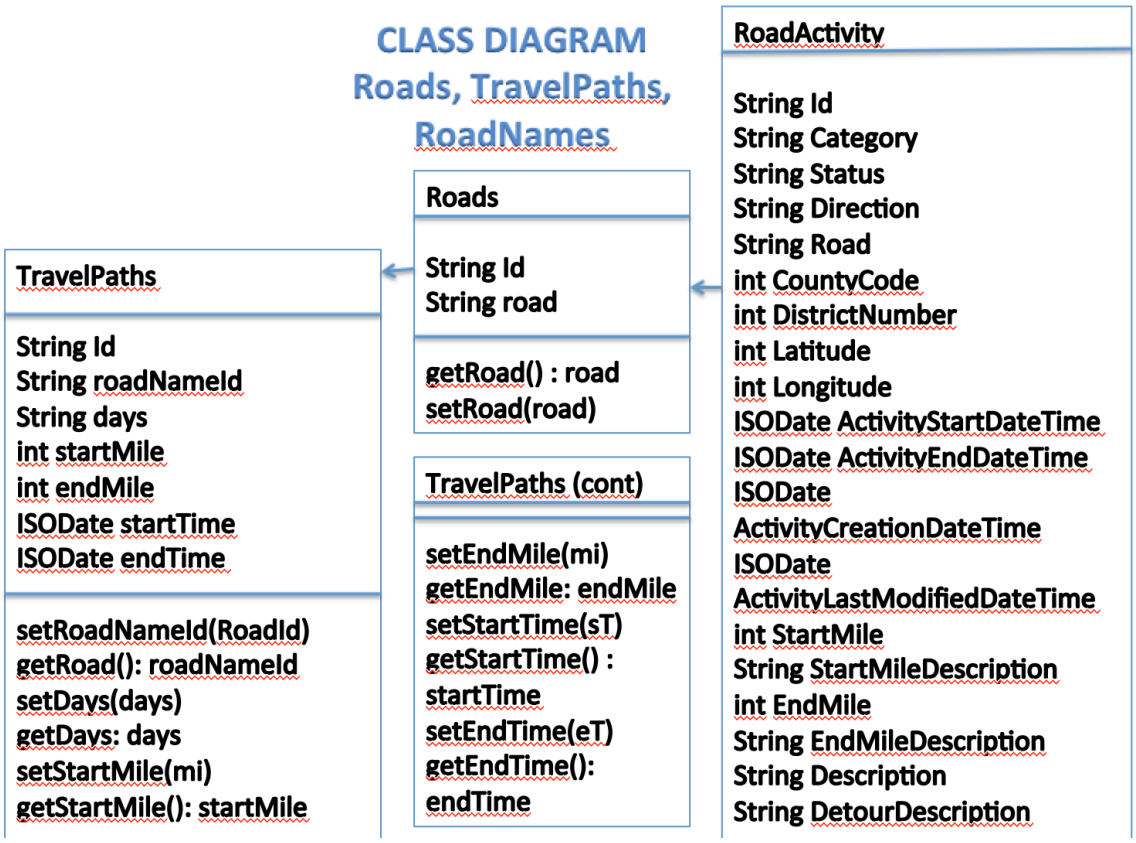
This diagram shows the interaction between the User Interface, add road to analyze, road names, and analyze roads class. The user can use the User Interface to add a road to analyze. When this is done, the road name selected by the user is pulled from the Road Names class and then the road is created in the Analyze Roads class. The design uses a couple different GRASP patterns. It uses the Information Expert pattern because it has to query the RoadActivity class to get the number of Accidents, Snow/Ice, Debris, Disabled Vehicle, etc. counts to provide to the UI. It also uses the Creator GRASP pattern because it creates a new instance of an AnalyzeRoad in order to analyze the data associated with it.

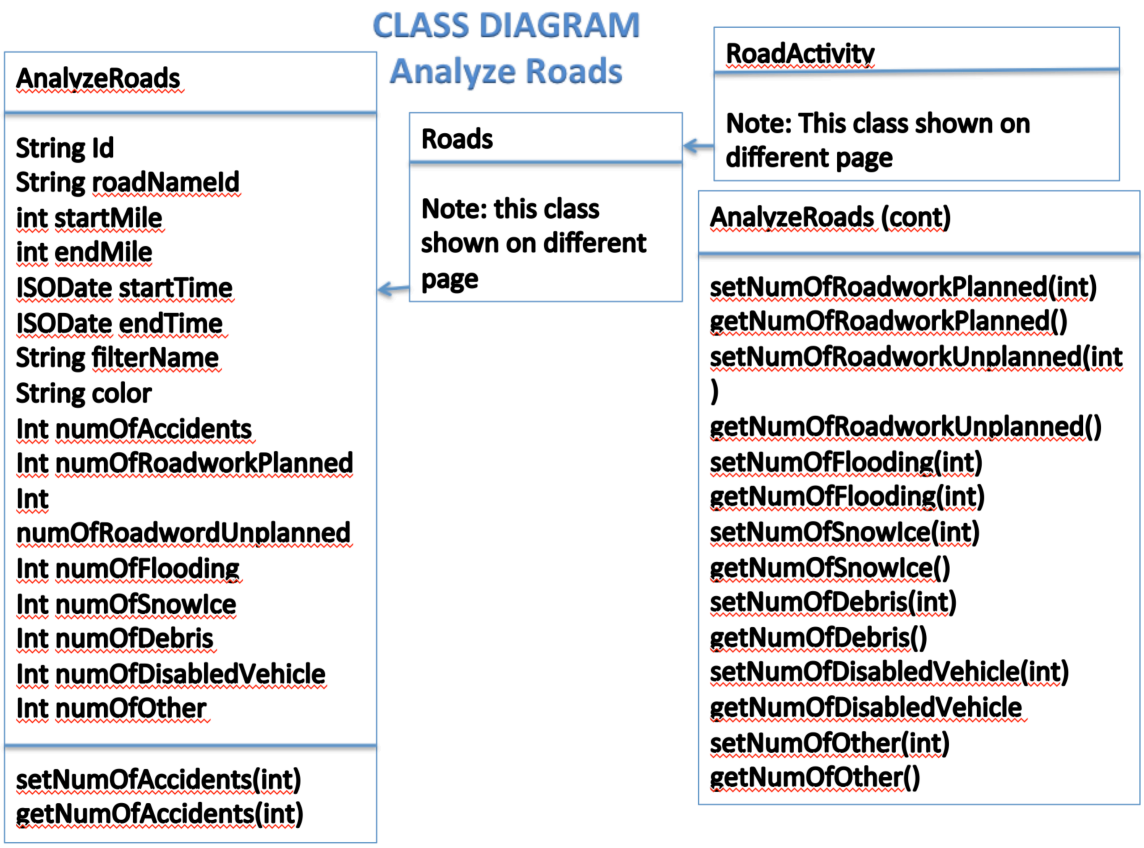
CLASS DIAGRAM
Accounts-Entry and Users

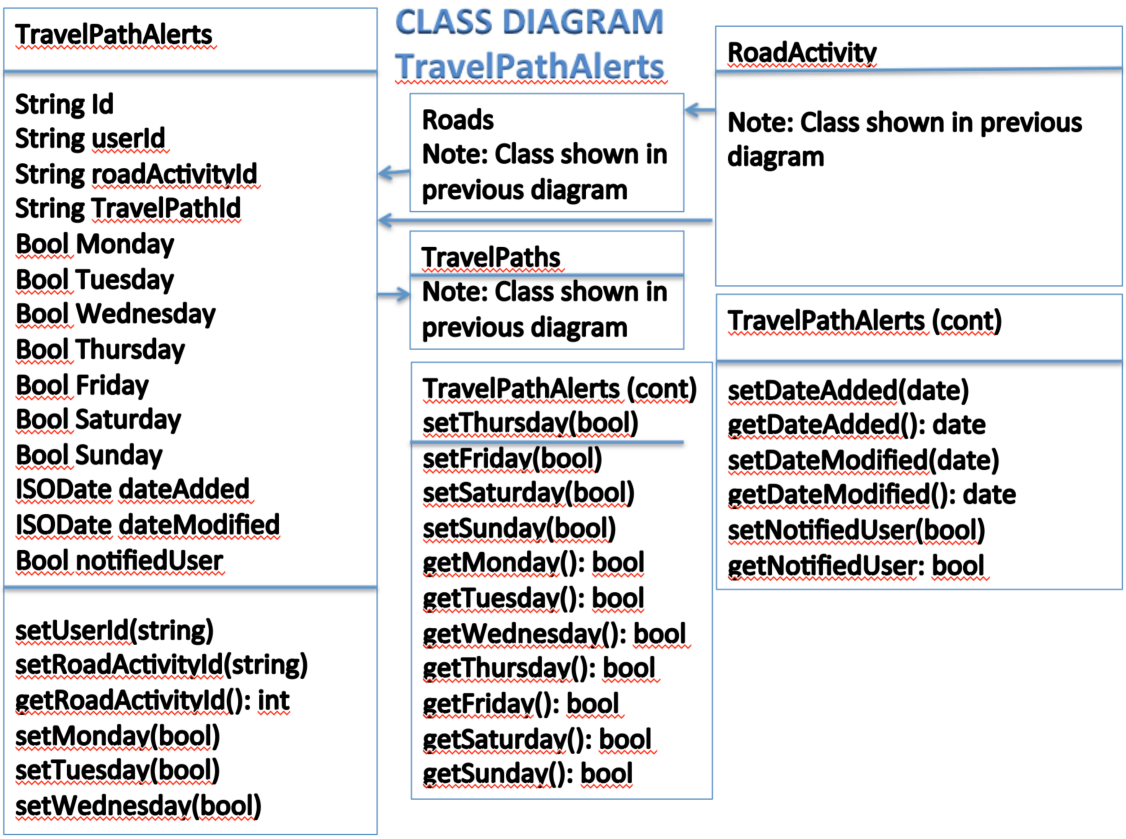


CLASS DIAGRAM
Update Data By User, Update Data by System, and Current Road Activities

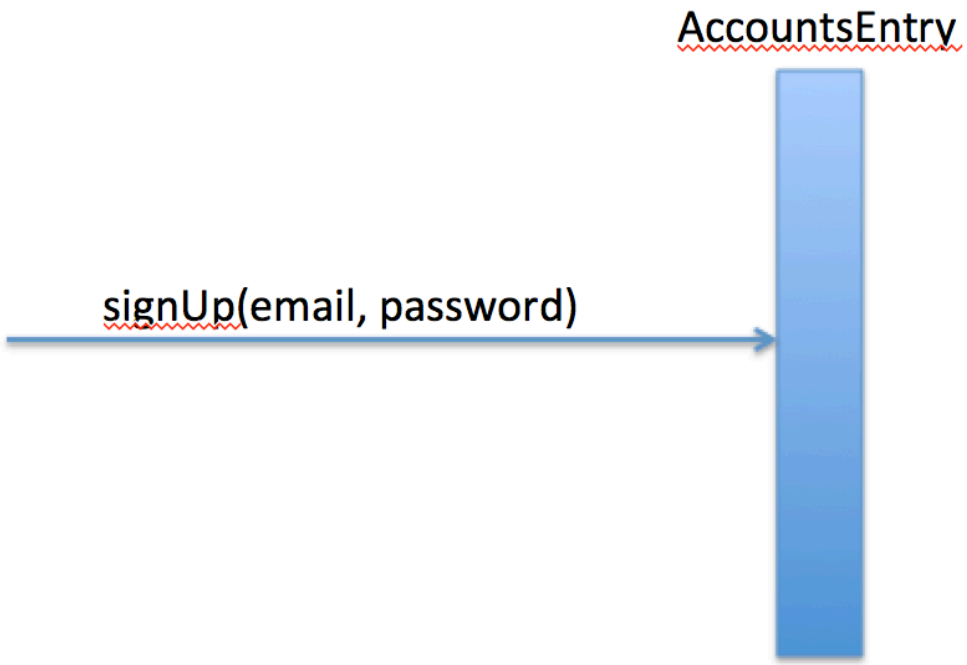



CLASS DIAGRAM
Roads, TravelPaths, RoadNames
RoadActivity
String Id
String Category
String Status
String Direction
String Road
int CountyCode
int DistrictNumber
int Latitude
int Longitude
ISODate ActivityStartDateTime
ISODate ActivityEndDateTime
ISODate ActivityCreationDateTime
ISODate ActivityLastModifiedDateTime
int StartMile
String StartMileDescription
int EndMile
String EndMileDescription
String Description
String DetourDescription
Roads
String Id
String road
getRoad() : road
setRoad(road)
TravelPaths
String Id
String roadNameId
String days
int startMile
int endMile
ISODate startTime
ISODate endTime
setRoadNameId(RoadId)
getRoad(): roadNameId
setDays(days)
getDays: days
setStartMile(mi)
getStartMile(): startMile
TravelPaths (cont)
setEndMile(mi)
getEndMile: endMile
setStartTime(sT)
getStartTime() : startTime
setEndTime(eT)
getEndTime(): endTime

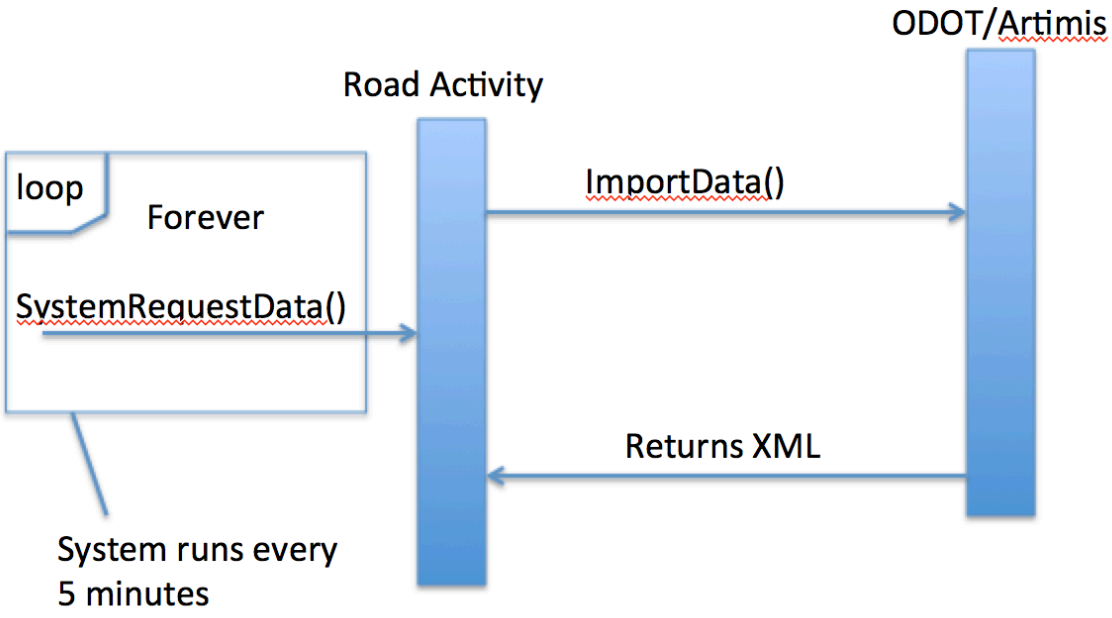

CLASS DIAGRAM
Analyze Roads
AnalyzeRoads
String Id
String roadNameId
int startMile
int endMile
ISODate startTime
ISODate endTime
String filterName
String color
Int numOfAccidents
Int numOfRoadworkPlanned
Int numOfRoadwordUnplanned
Int numOfFlooding
Int numOfSnowIce
Int numOfDebris
Int numOfDisabledVehicle
Int numOfOther
setNumOfAccidents(int)
getNumOfAccidents(int)
Roads
Note: this class shown on different page
RoadActivity
Note: This class shown on different page
AnalyzeRoads (cont)
setNumOfRoadworkPlanned(int)
getNumOfRoadworkPlanned()
setNumOfRoadworkUnplanned(int)
getNumOfRoadworkUnplanned()
setNumOfFlooding(int)
getNumOfFlooding(int)
setNumOfSnowIce(int)
getNumOfSnowIce()
setNumOfDebris(int)
getNumOfDebris()
setNumOfDisabledVehicle(int)
getNumOfDisabledVehicle
setNumOfOther(int)
getNumOfOther()


CLASS DIAGRAM
TravelPathAlerts
TravelPathAlerts
String Id
String userId
String roadActivityId
String TravelPathId
Bool Monday
Bool Tuesday
Bool Wednesday
Bool Thursday
Bool Friday
Bool Saturday
Bool Sunday
ISODate dateAdded
ISODate dateModified
Bool notifiedUser
setUserId(string)
setRoadActivityId(string)
getRoadActivityId(): int
setMonday(bool)
setTuesday(bool)
setWednesday(bool)
Roads
Note: Class shown in previous diagram
TravelPaths
Note: Class shown in previous diagram
TravelPathAlerts (cont)
setThursday(bool)
setFriday(bool)
setSaturday(bool)
setSunday(bool)
getMonday(): bool
getTuesday(): bool
getWednesday(): bool
getThursday(): bool
getFriday(): bool
getSaturday(): bool
getSunday(): bool
RoadActivity
Note: Class shown in previous diagram
TravelPathAlerts (cont)
setDateAdded(date)
getDateAdded(): date
setDateModified(date)
getDateModified(): date
setNotifiedUser(bool)
getNotifiedUser: bool

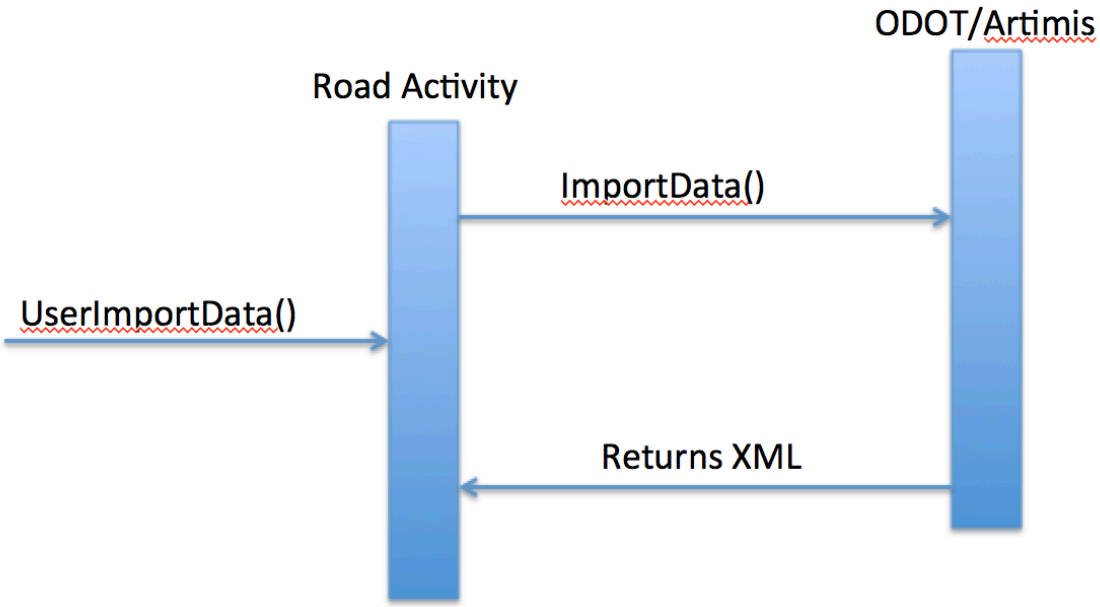
Interaction (Sequence) Diagram – Use Case 1: Sign Up



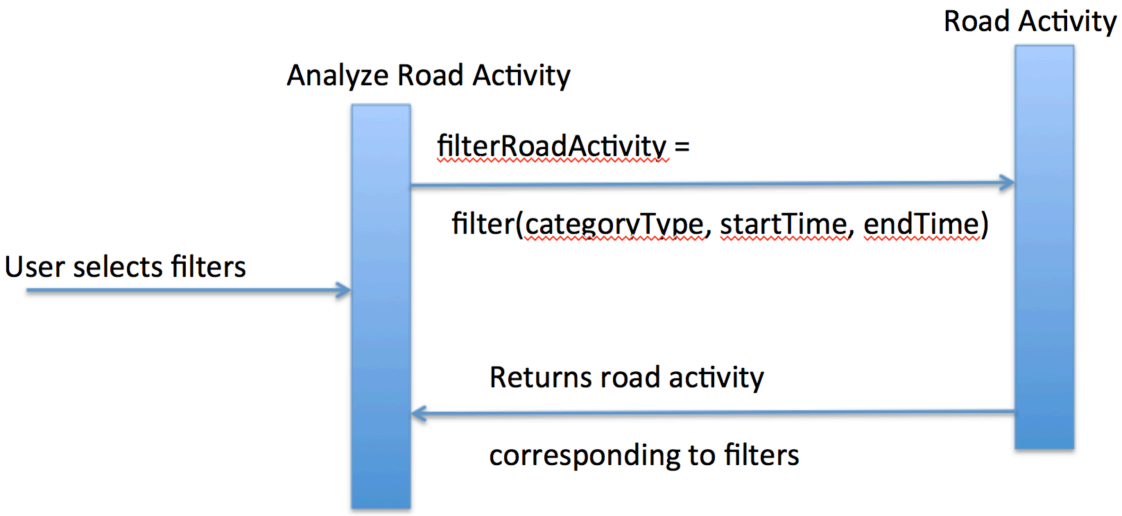
Interaction (Sequence) Diagram – Use Case 2: Update Data by System



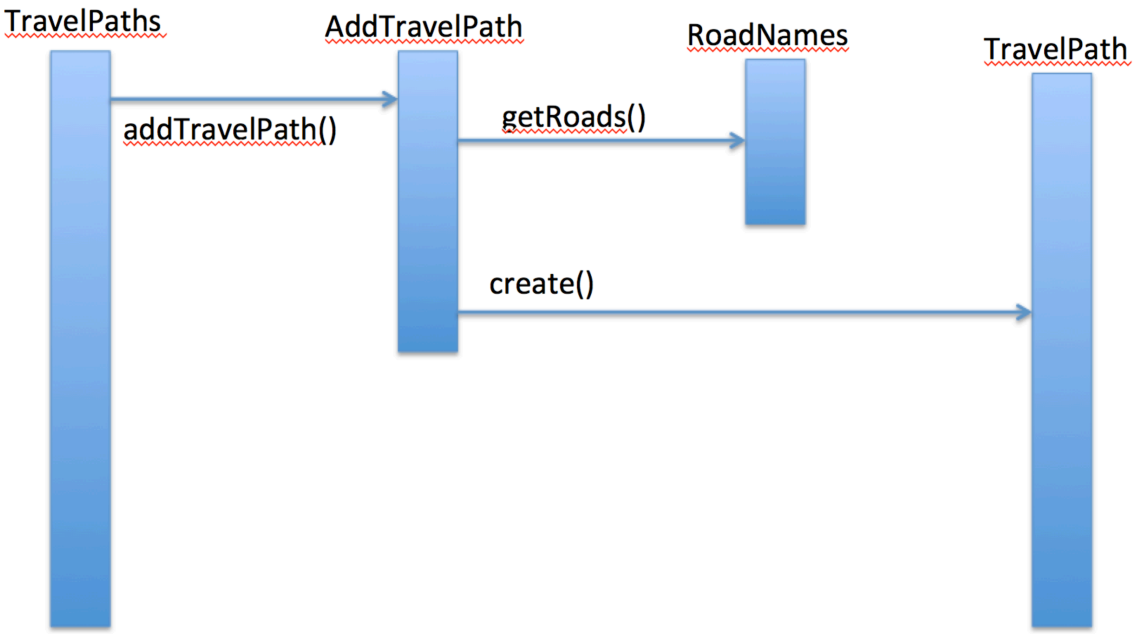
Interaction (Sequence) Diagram – Use Case 3: Update Data by User



Interaction (Sequence) Diagram – Use Case 4: Filter Road Activity

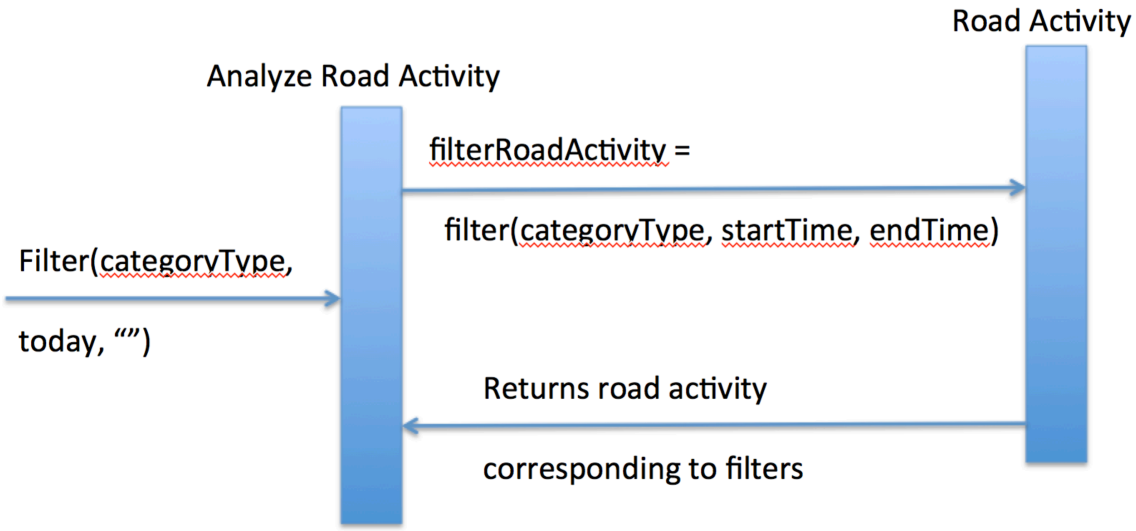


Interaction (Sequence) Diagram – Use Case 5: Enter Roads Travelled

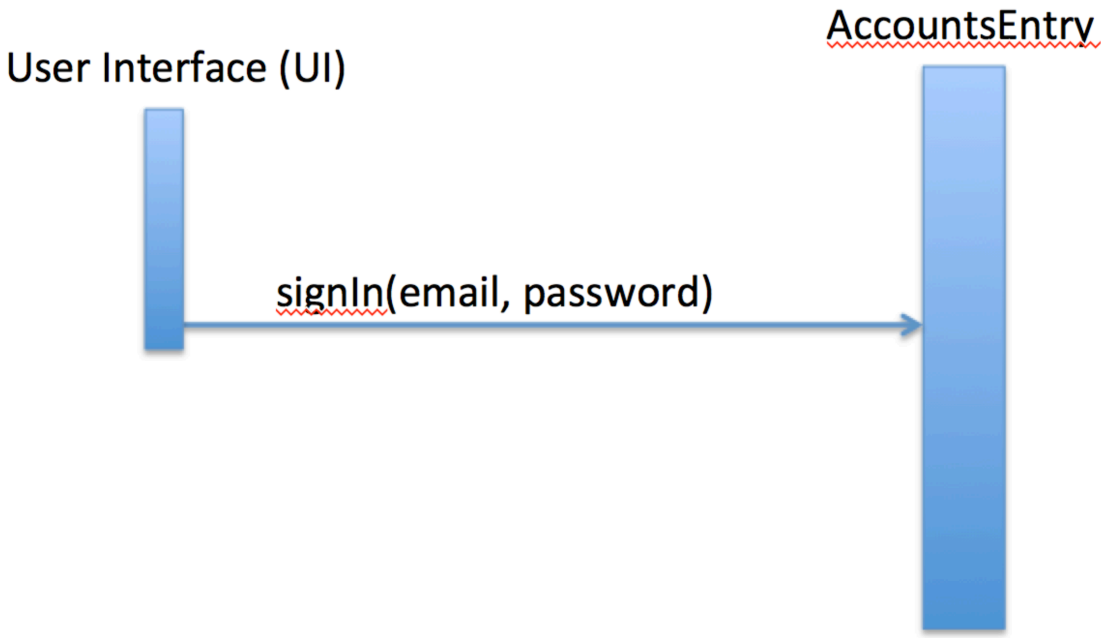


Interaction (Sequence) Diagram – Use Case 6: Notification System

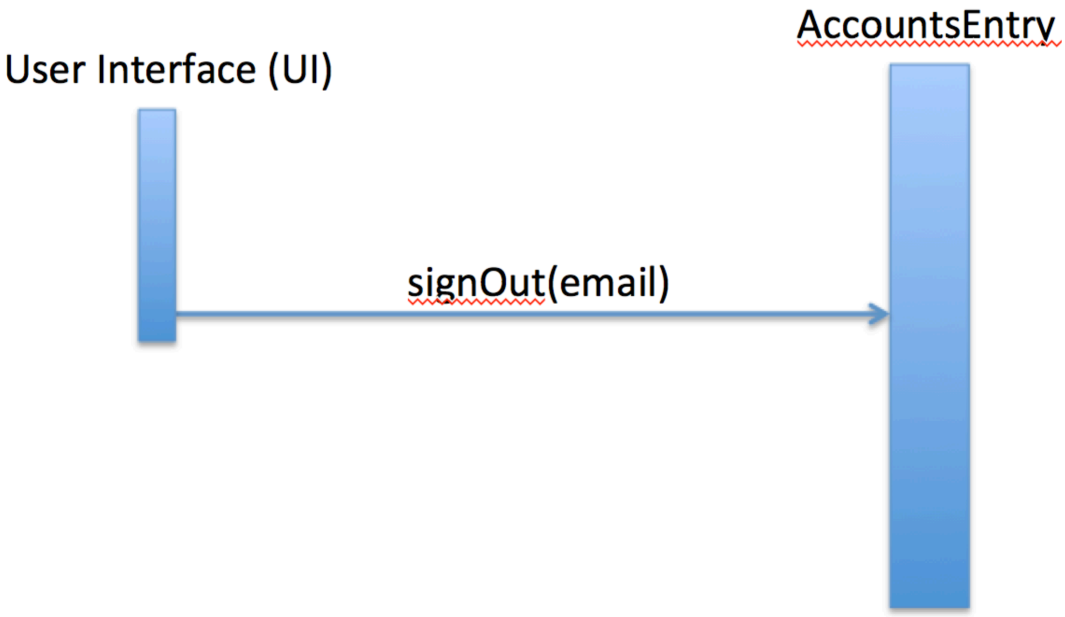
Interaction (Sequence) Diagram – Use Case 7: Current Road Activities



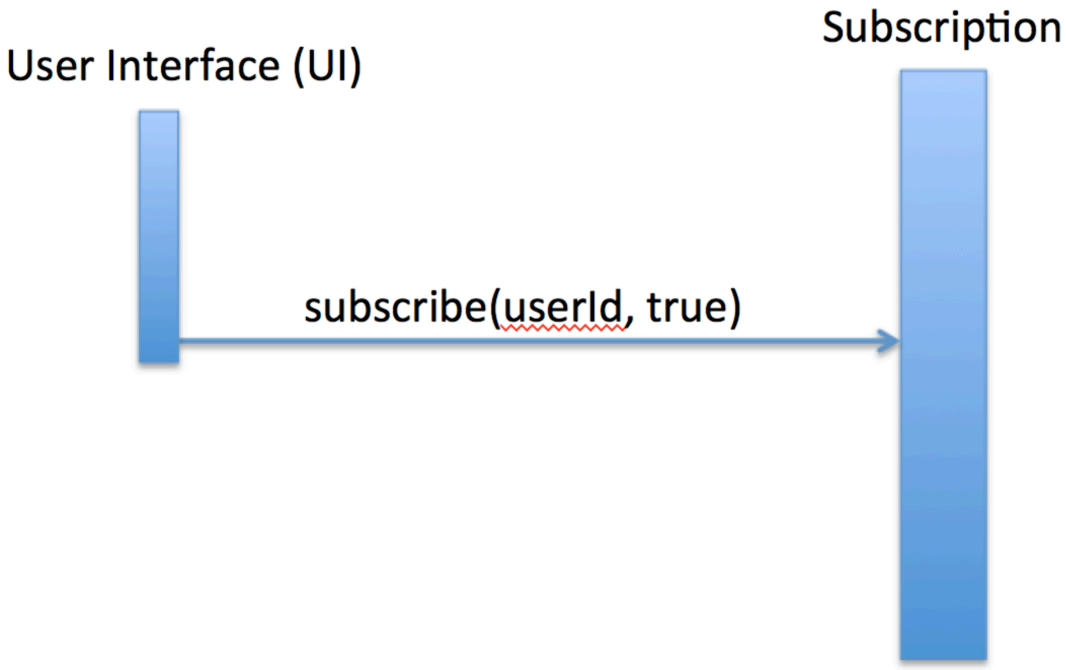
Interaction (Sequence) Diagram – Use Case 8: Sign In



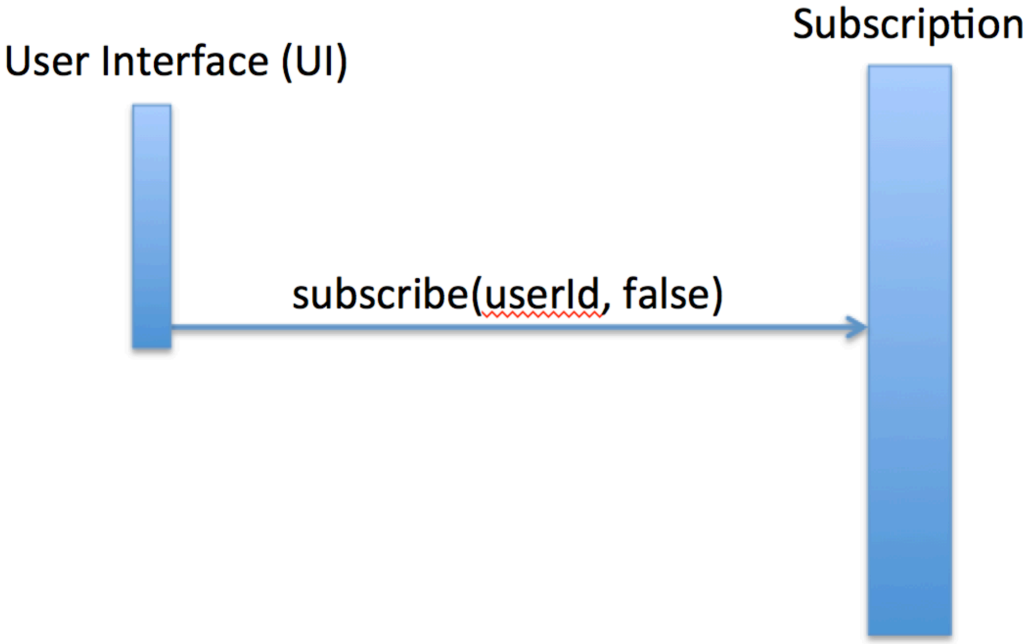
Interaction (Sequence) Diagram – Use Case 9: Sign Out



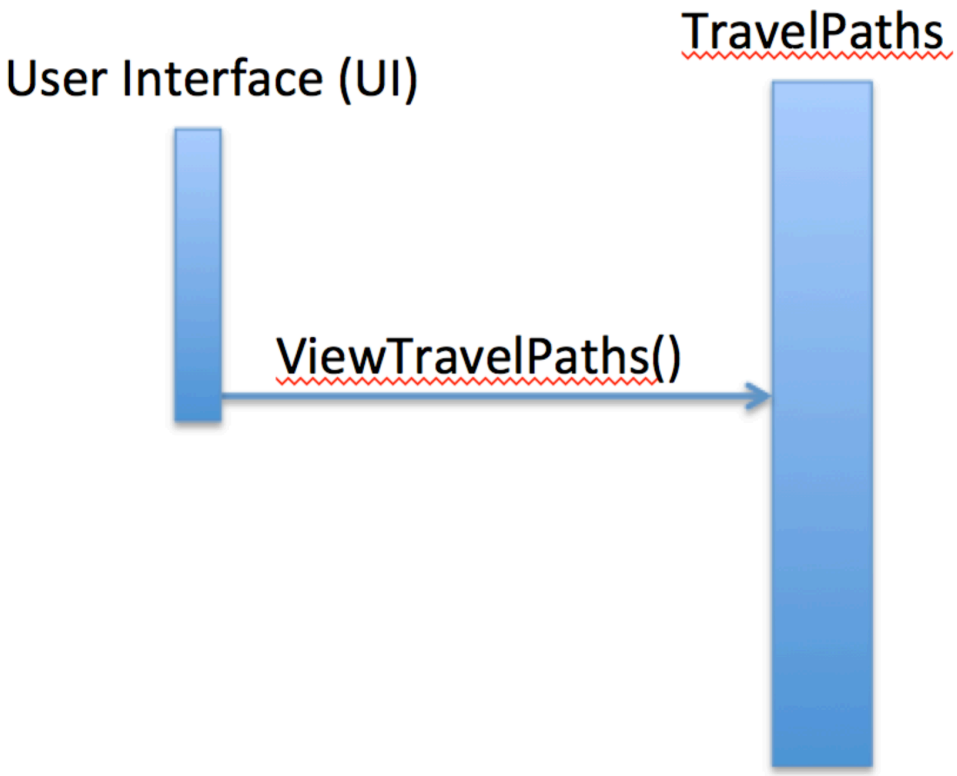
Interaction (Sequence) Diagram – Use Case 10: Subscribe to Alerts for Travel Path



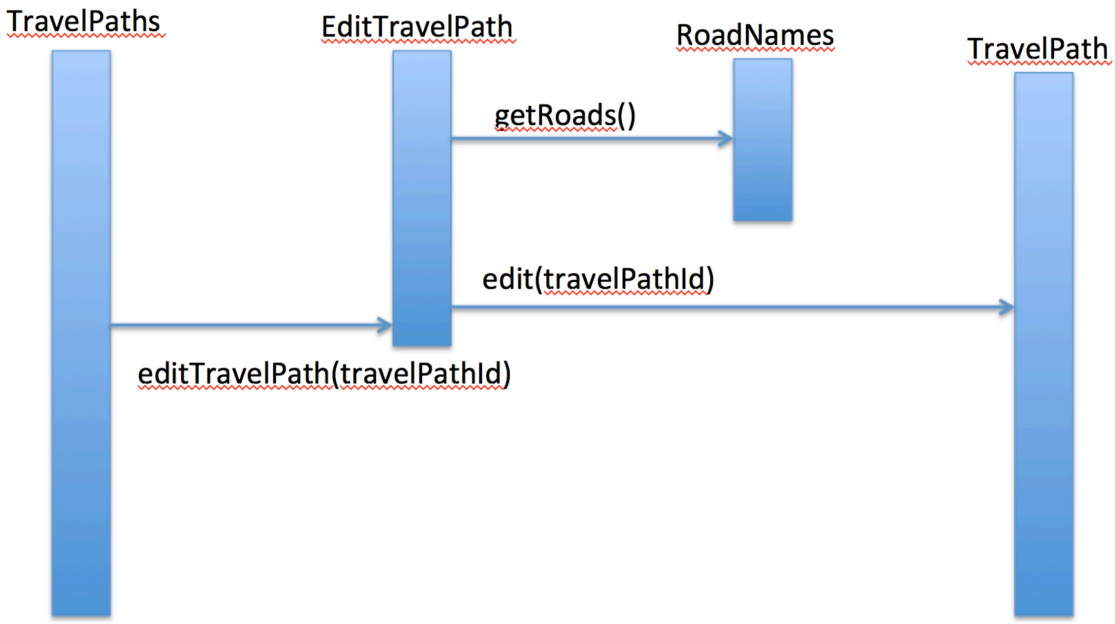
Interaction (Sequence) Diagram – Use Case 11: Unsubscribe to Alerts for Travel Path



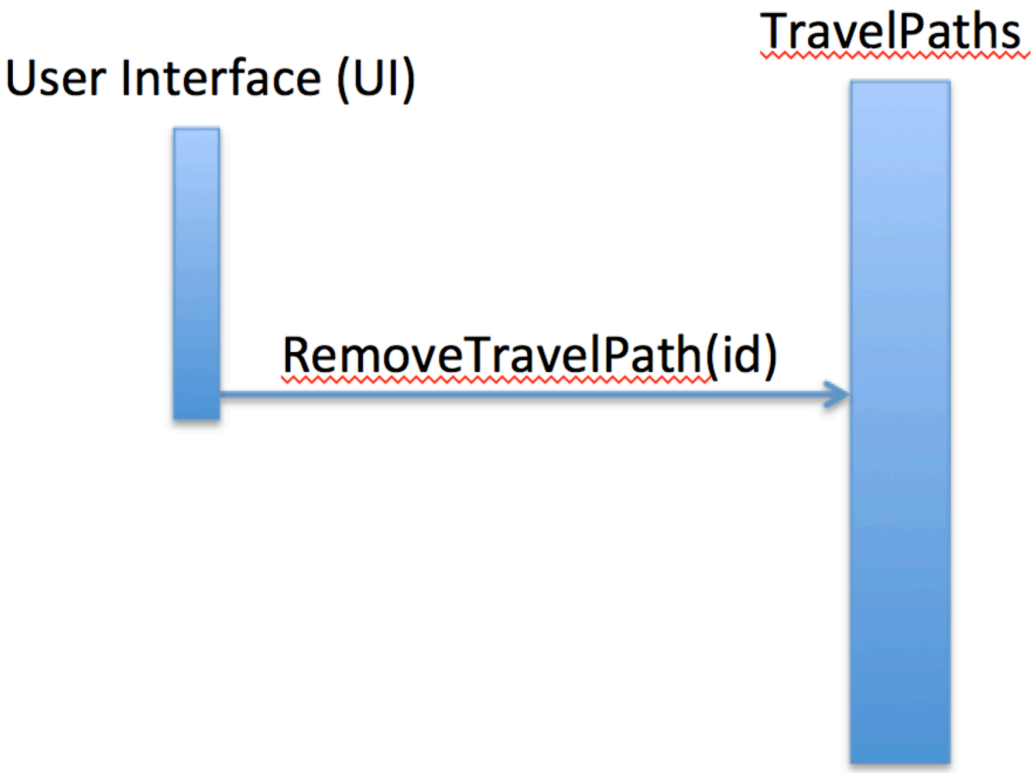
Interaction (Sequence) Diagram – Use Case 12: View All Travel Paths



Interaction (Sequence) Diagram – Use Case 13: Edit Roads Travelled



Interaction (Sequence) Diagram – Use Case 14: Remove Travel Path



Interaction (Sequence) Diagram – Use Case 15: Analyze Road Activity