

**Take Home A Robot**

**CarpoolBot Step-by-Step Guide**

UiPath

## Getting started

### **Install the free version of UiPath Studio**

Use this link to get started:

### <https://www.uipath.com/community>

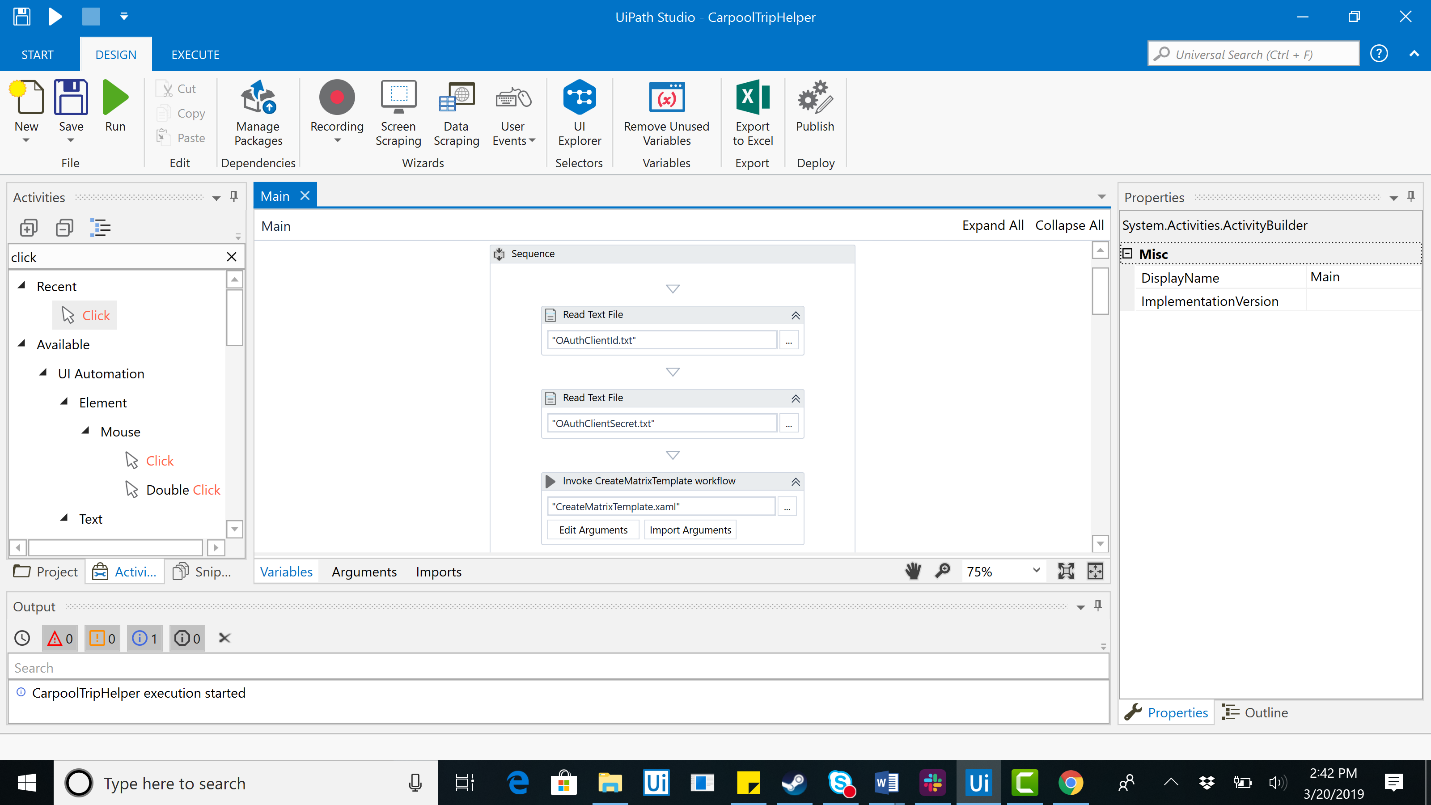
Once you have UiPath Studio Community or Trial edition installed, you are ready to go!

Overview: This bot is for a large group that wants to carpool. It will do two things at random (making sure no one has to be the bad guy)

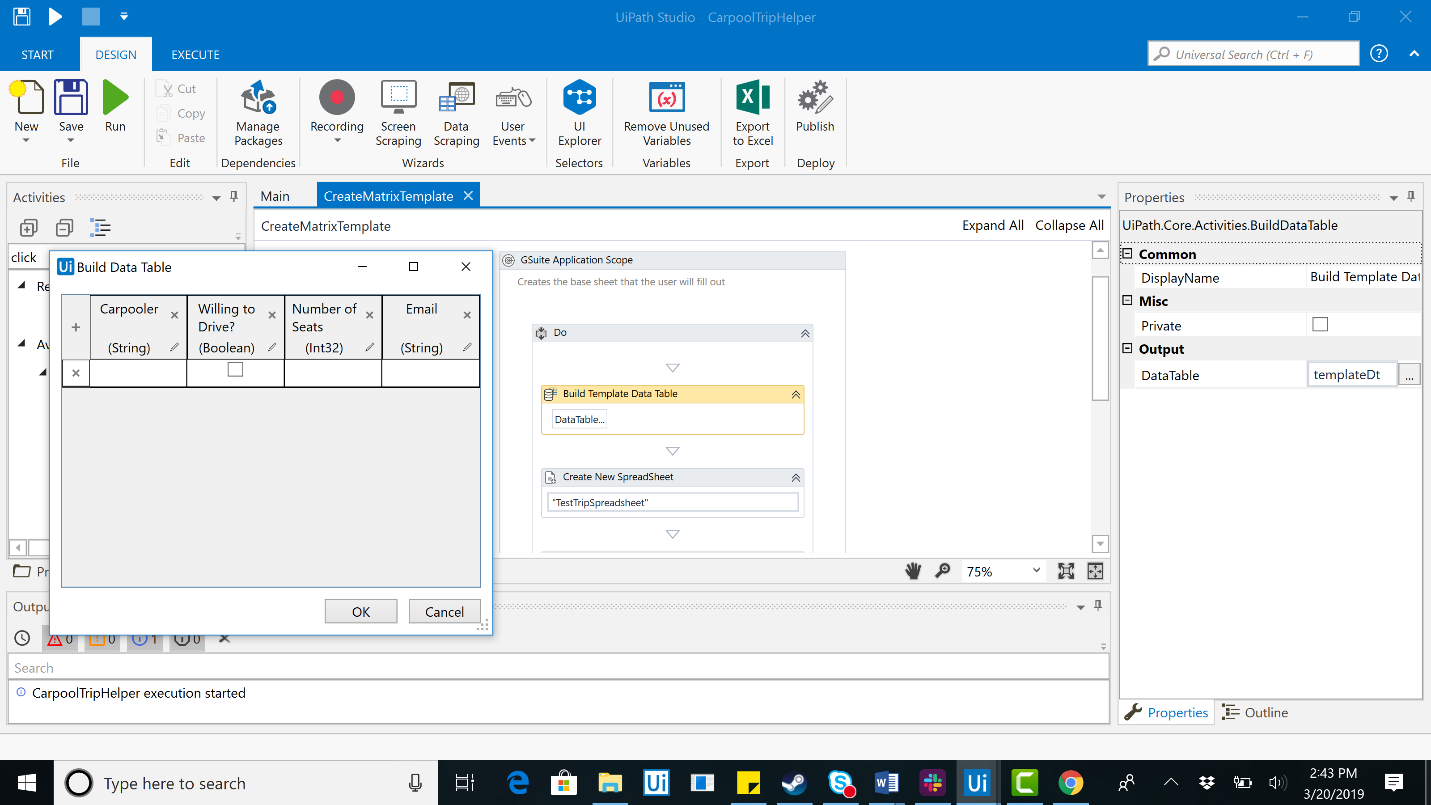
1. The bot will assign people to cars
2. The bot will assign people to bring specified items

Prerequisites:

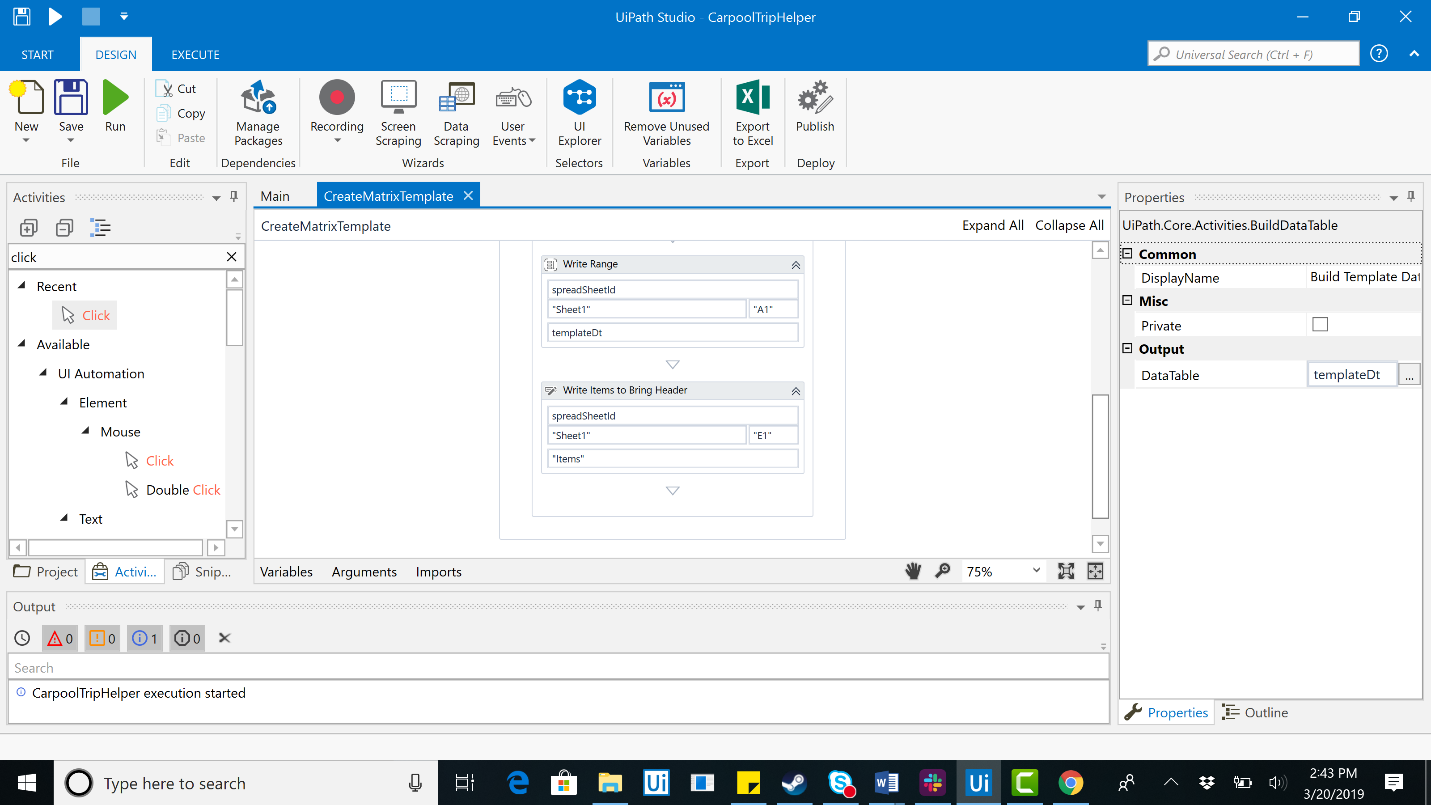
1. A google e-mail
2. Set up google sheet API with ‘OAuth Client Id’ following the instructions on this link: <https://integrations.uipath.com/docs/gsuite-application-scope>
3. Once you have enabled your google sheets API, you will have an OAuthClientID and OAuthClientSecret. Save each separately as a text file. The first step is to read both and save the output.



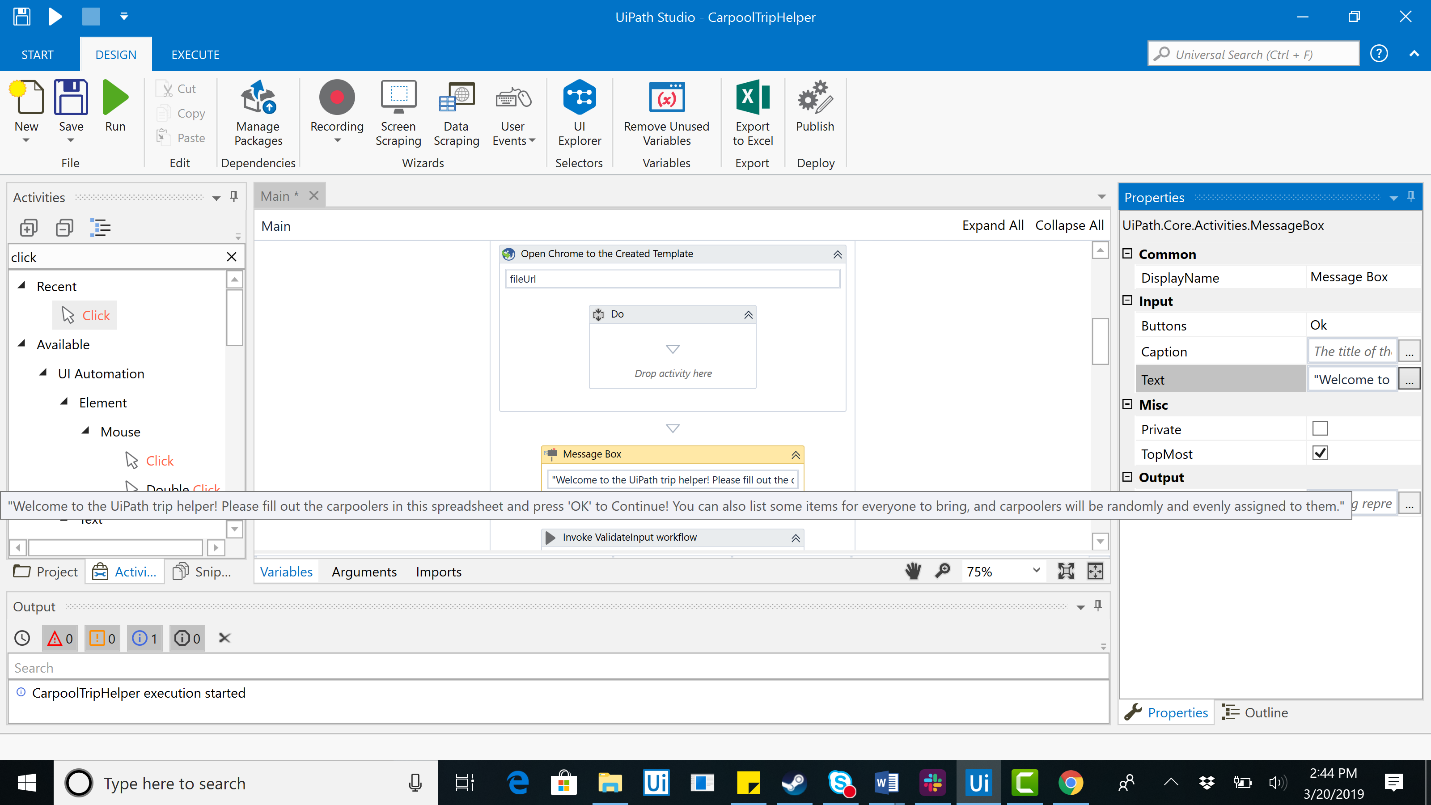
1. Next you are going to create a sequence called CreateMatrixTemplate. You are preparing the datatable to be entered into Google Sheets. Create columns for Carpooler, Willing to Drive, Number of Seats, and Email. Next, you are going to create a spreadsheet to write this on. In our example we named it TestTripSpreadsheet.



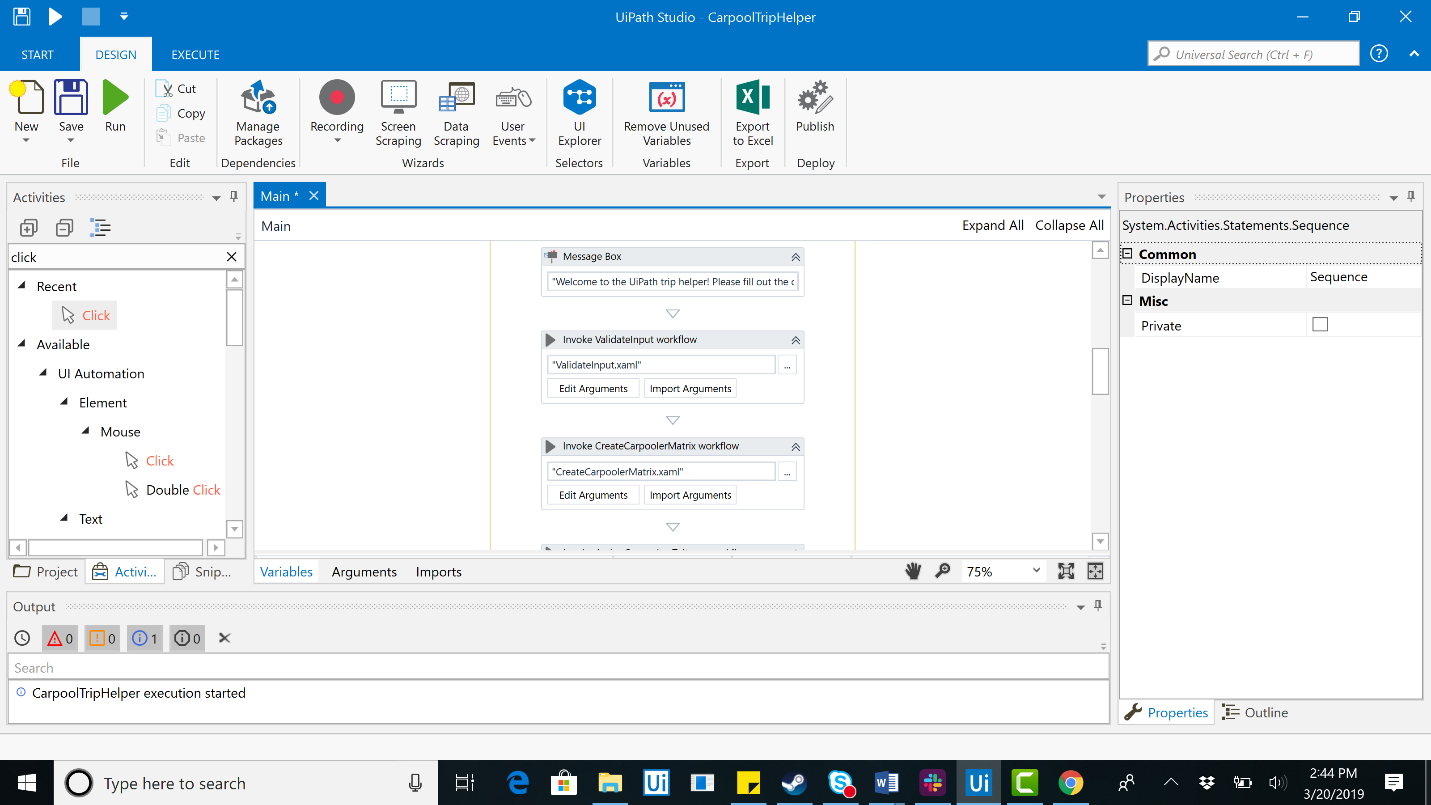
1. Next you will write this range to Google Sheets. Finally, add ‘Items’ as a header so that there is a place to store all items the party is hoping to bring.



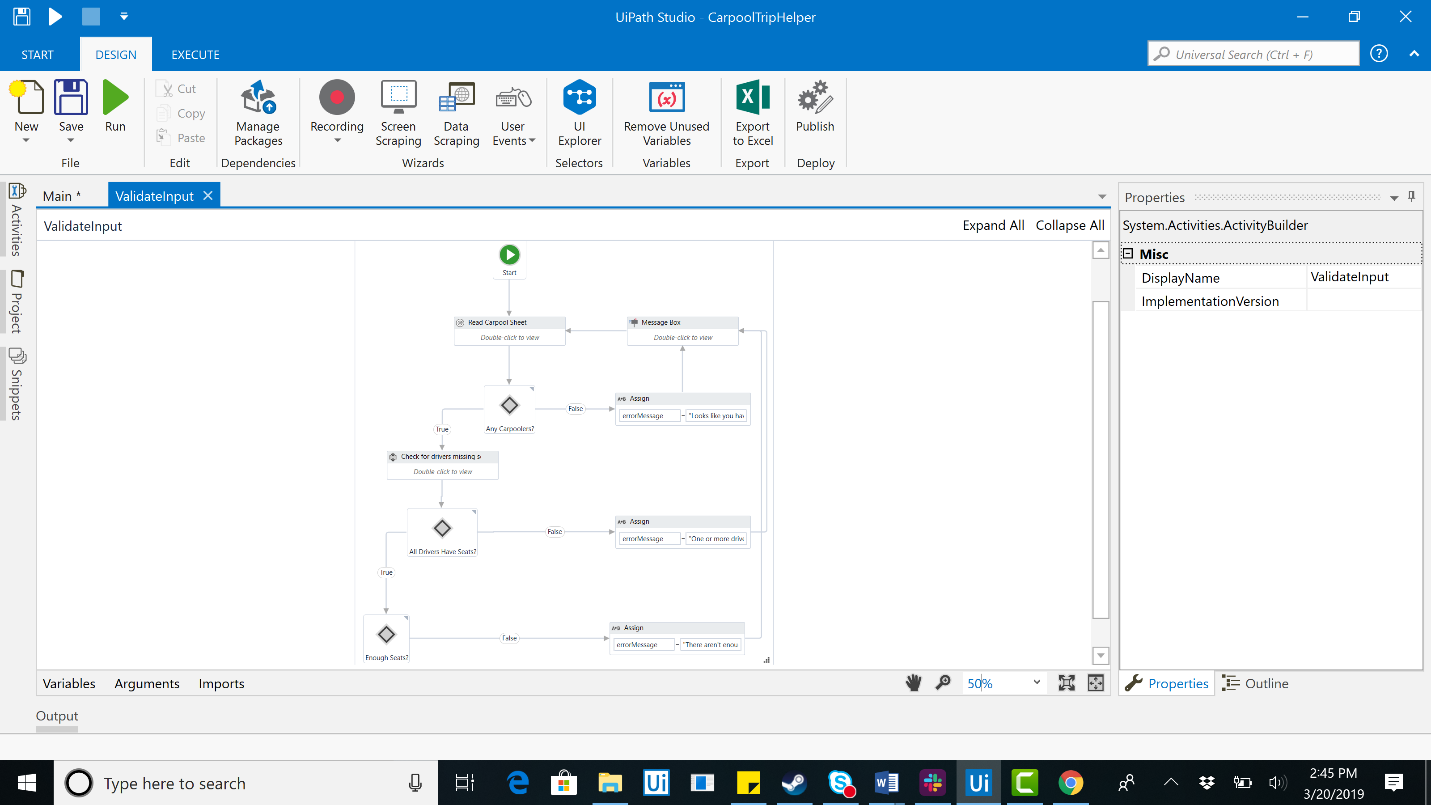
1. When this is completed, go back to Main. Drag in an Open Browser activity and input fileURL. This value should be an output argument from the Create New Spreadsheet in previous step. As you can see, once the browser is open there will be instructions for the user in the form of a message box.



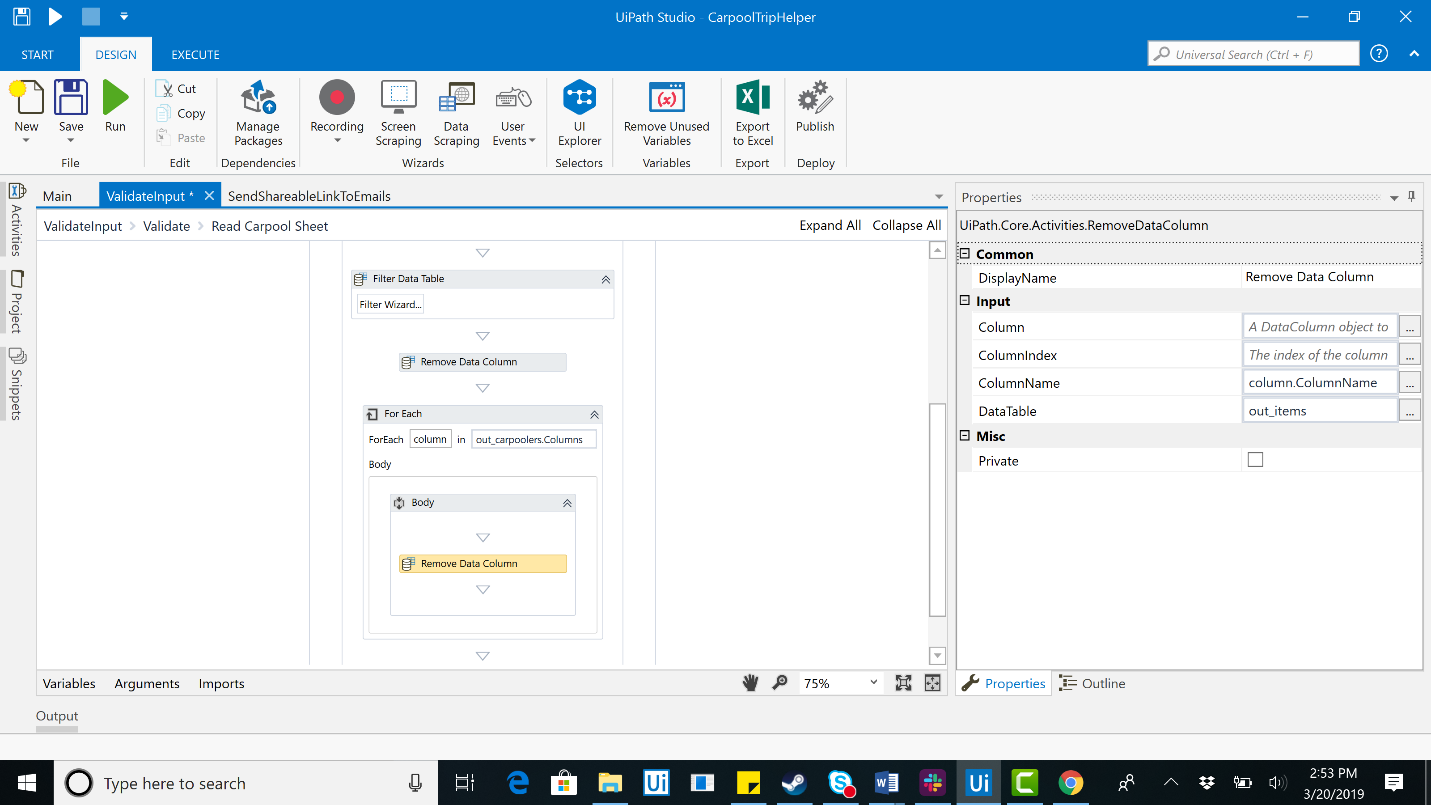
1. The next workflow in Main is ValidateInput. This is to ensure the values entered support a well organized roadtrip. Create this xaml file now.



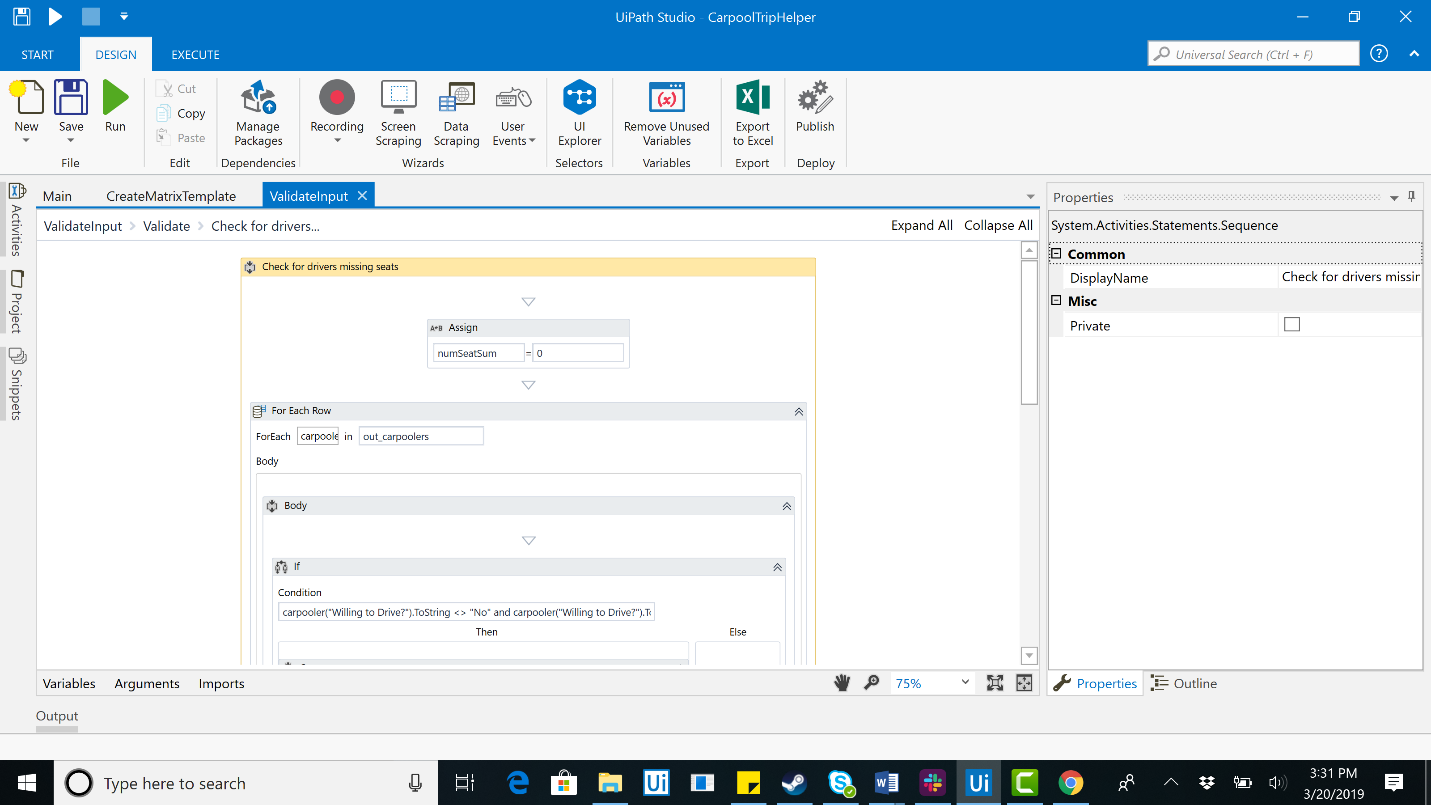
1. The three decisions the bot will have to make are ‘Any Carpoolers?’, ‘All drivers have seats?’, and finally ‘Enough Seats?’. If a condition is not met, a message box will instruct the user to update the data accordingly.



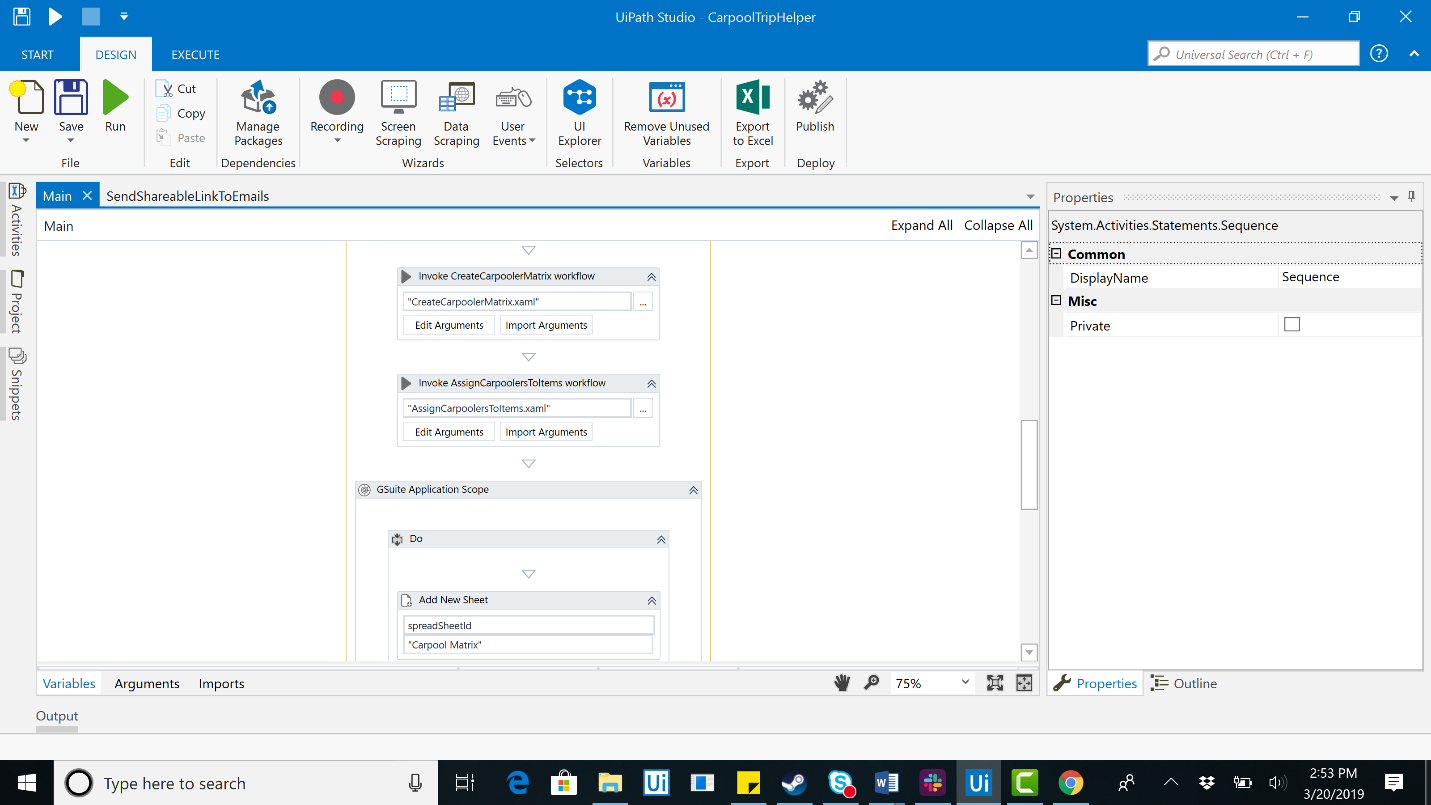
1. First read the range for all the data. You are going to filter all the data for two DataTables. First, one containing Carpoolers, Willing to Drive, and Seats. Next, you will get the DataTable for Items. These two DataTables are going to be your out arguments. The first condition in the flow decision is going to be ‘out\_carpoolers.Rows.Count > 0’. If no carpoolers have been entered, it will ask the user to enter them.



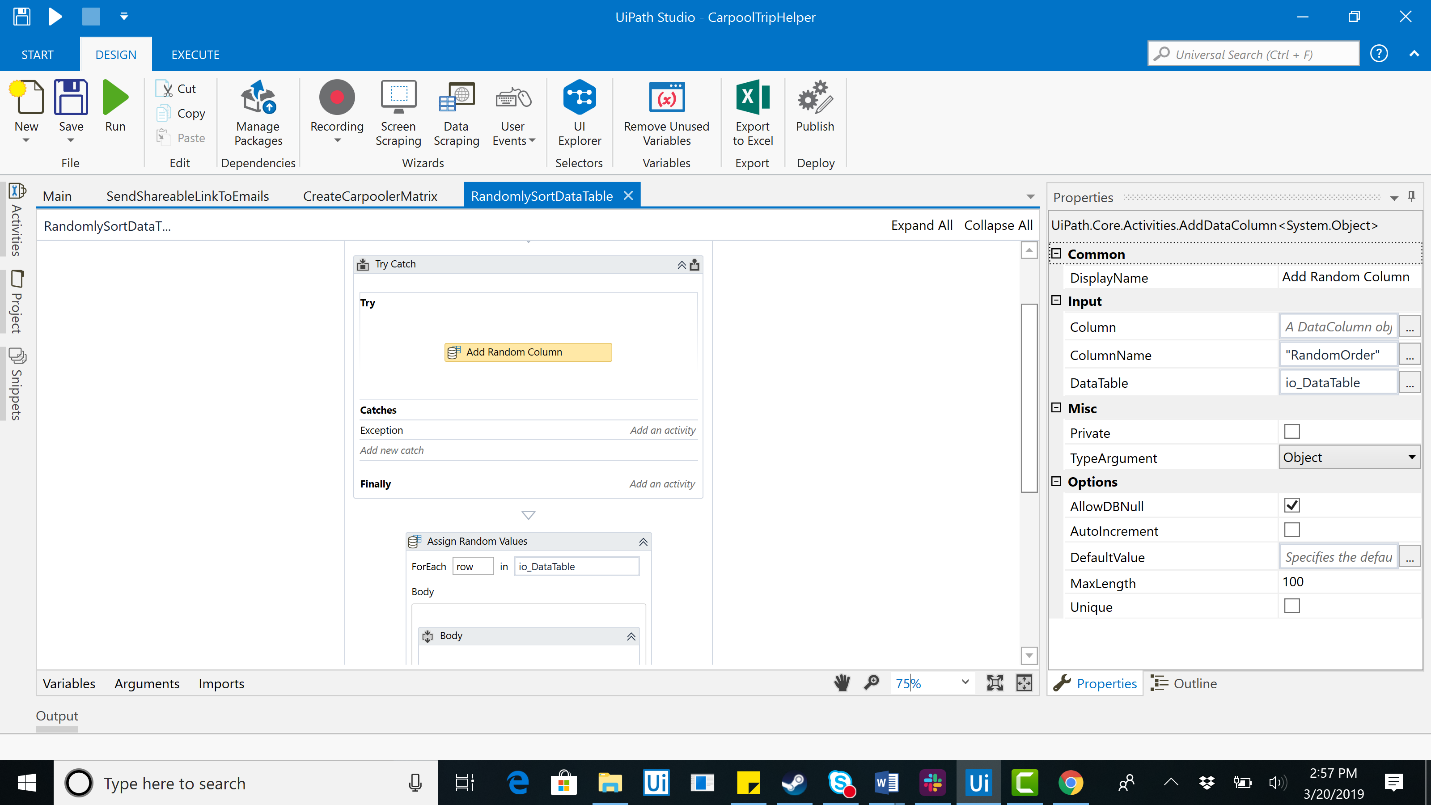
1. The next step is to ensure that the All Drivers Have Seats condition is met. Add a sequence called Check for Drivers Missing Seats. Iterate through each row in out\_carpoolers and pass it through this condition ‘carpooler("Willing to Drive?").ToString <> "No" and carpooler("Willing to Drive?").ToString <> ""’. After this if numSeats > 0 is met and ‘numSeatSum >= out\_carpoolers.Rows.Count’ is satisfied then it will be considered validated and we can continue to the next step in the Main.xaml.



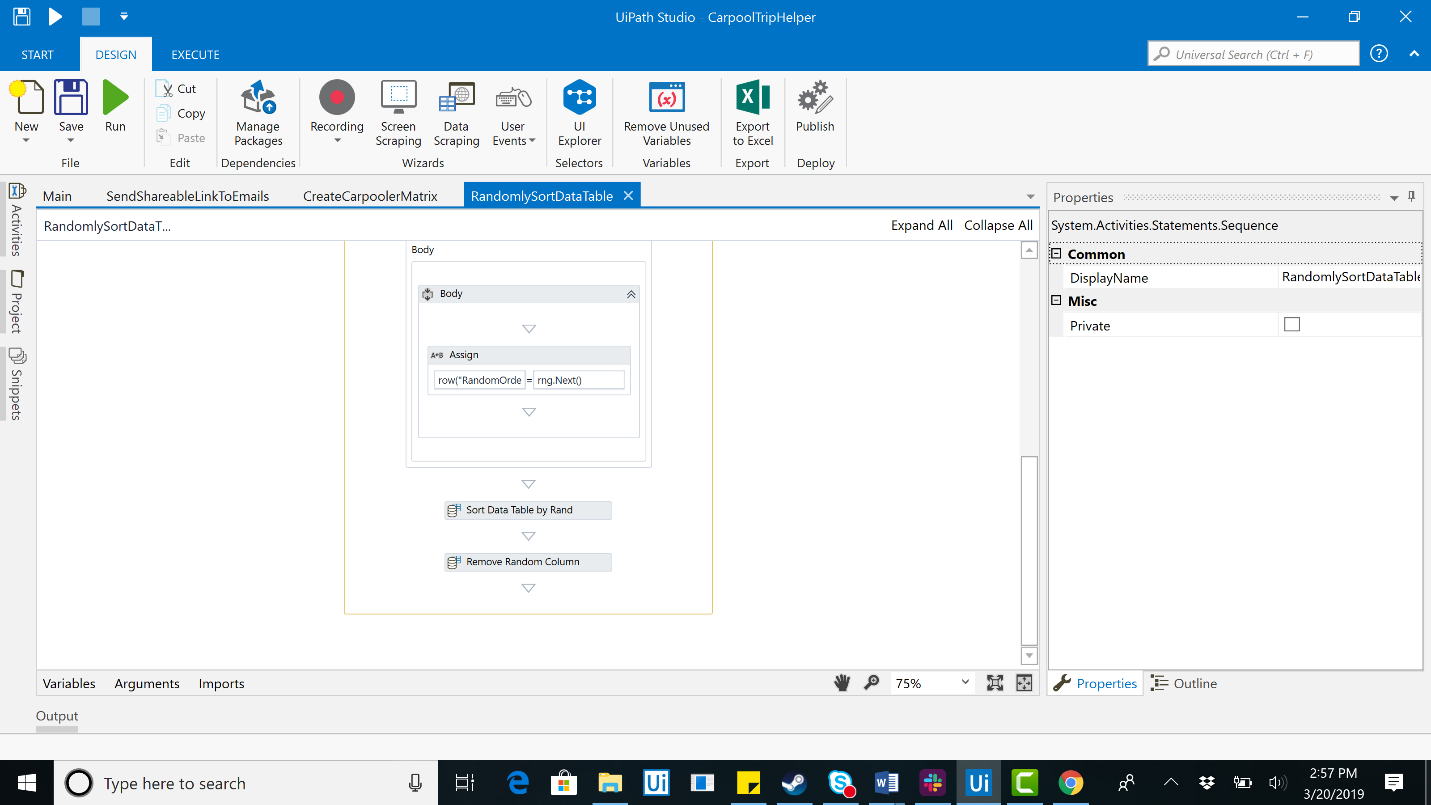
1. Now we are going to CreateCarpoolrMatrix. This is a separate sheet that considers all the entered data and outputs and the spreadsheet of who is in which car as well as the items assigned to each person.



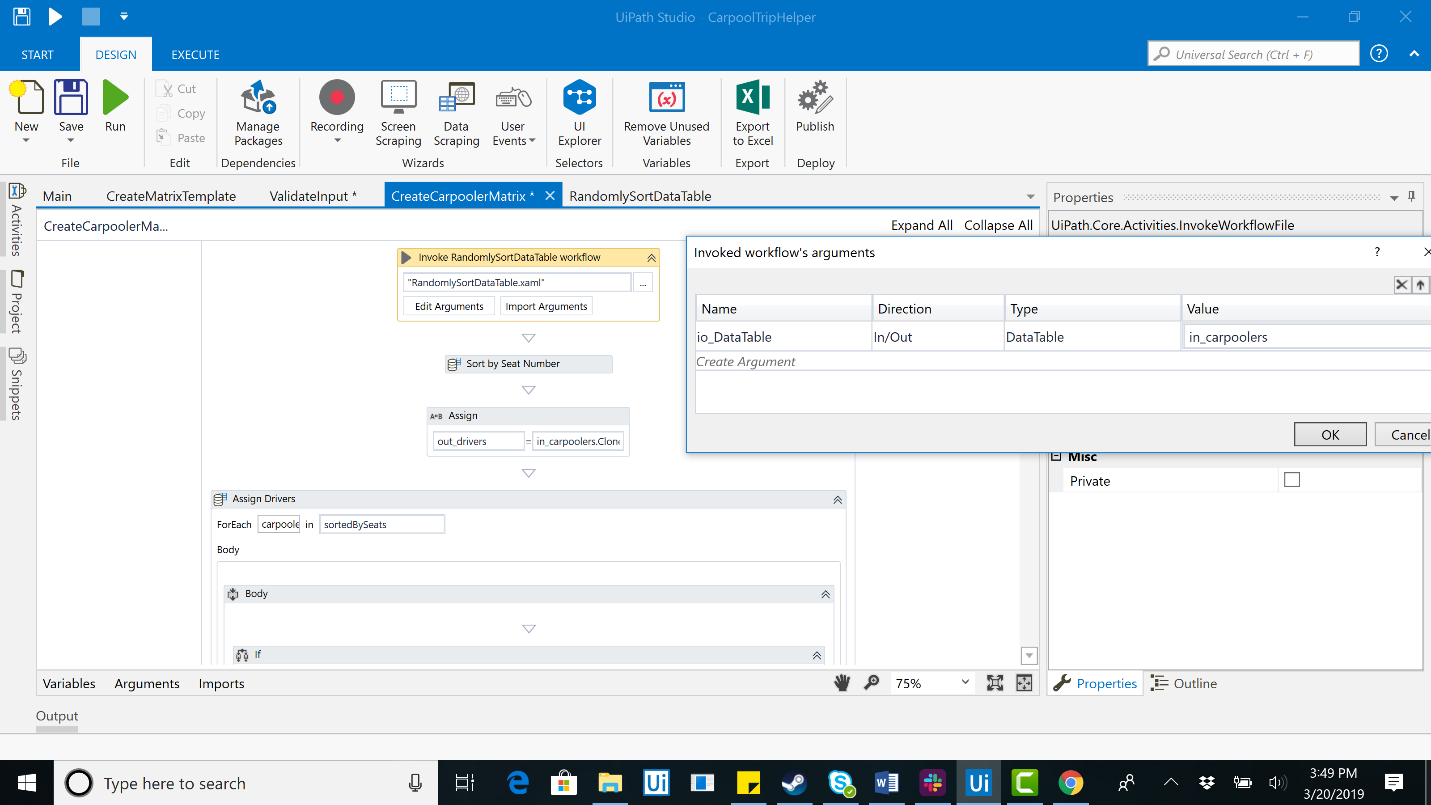
1. The first xaml file we encounter is the RandomlySortDataTable.xaml. Now that we are assigining people, we need to be fair and thus need a random way to sort. Let’s build this before completing CreateCarpoolerMatrix. First add a row whichever DataTable you will be randomly sorting.



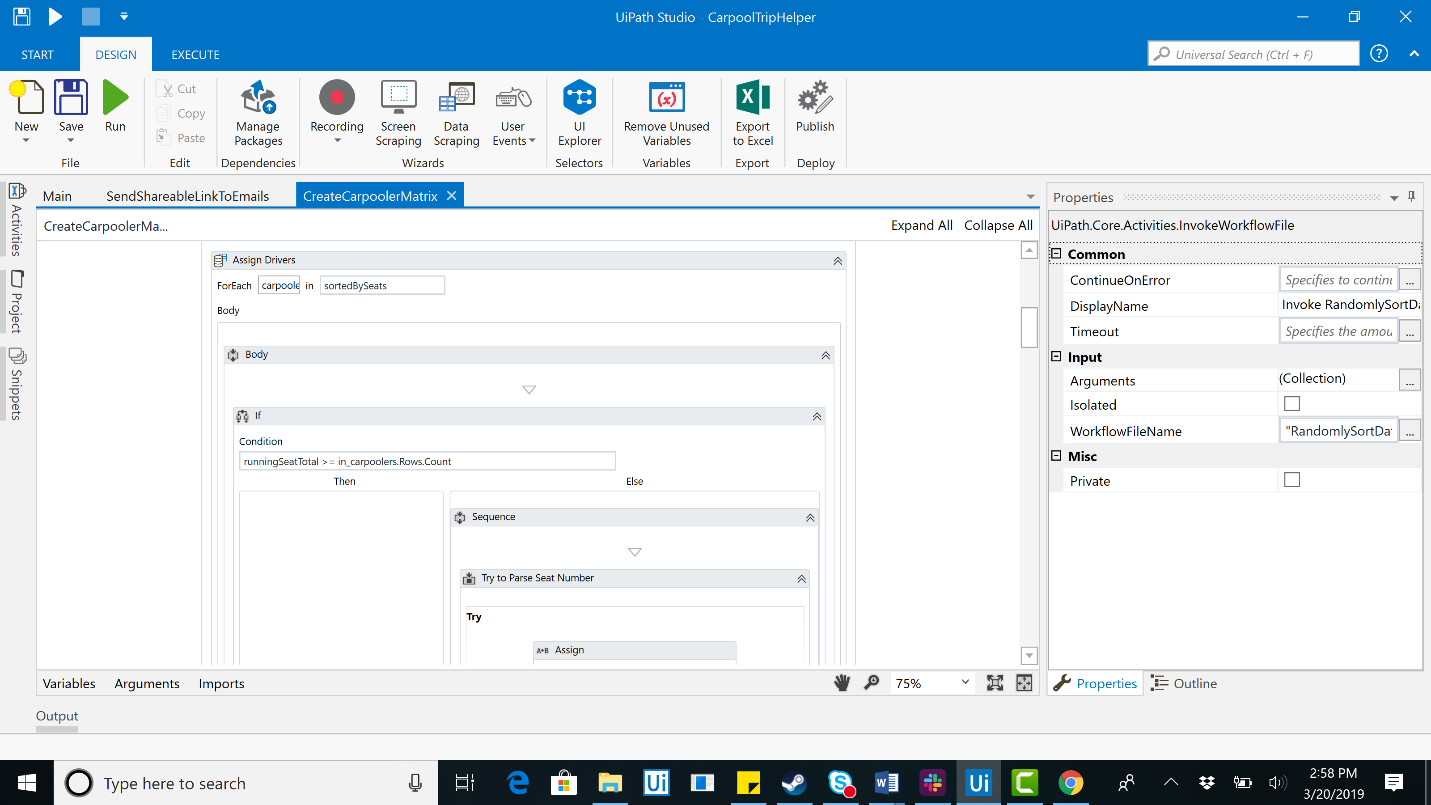
1. Next, add a number value for each row in the column. Finally, sort that column and remove the random row as it is unnecessary after a random sort has been achieved.



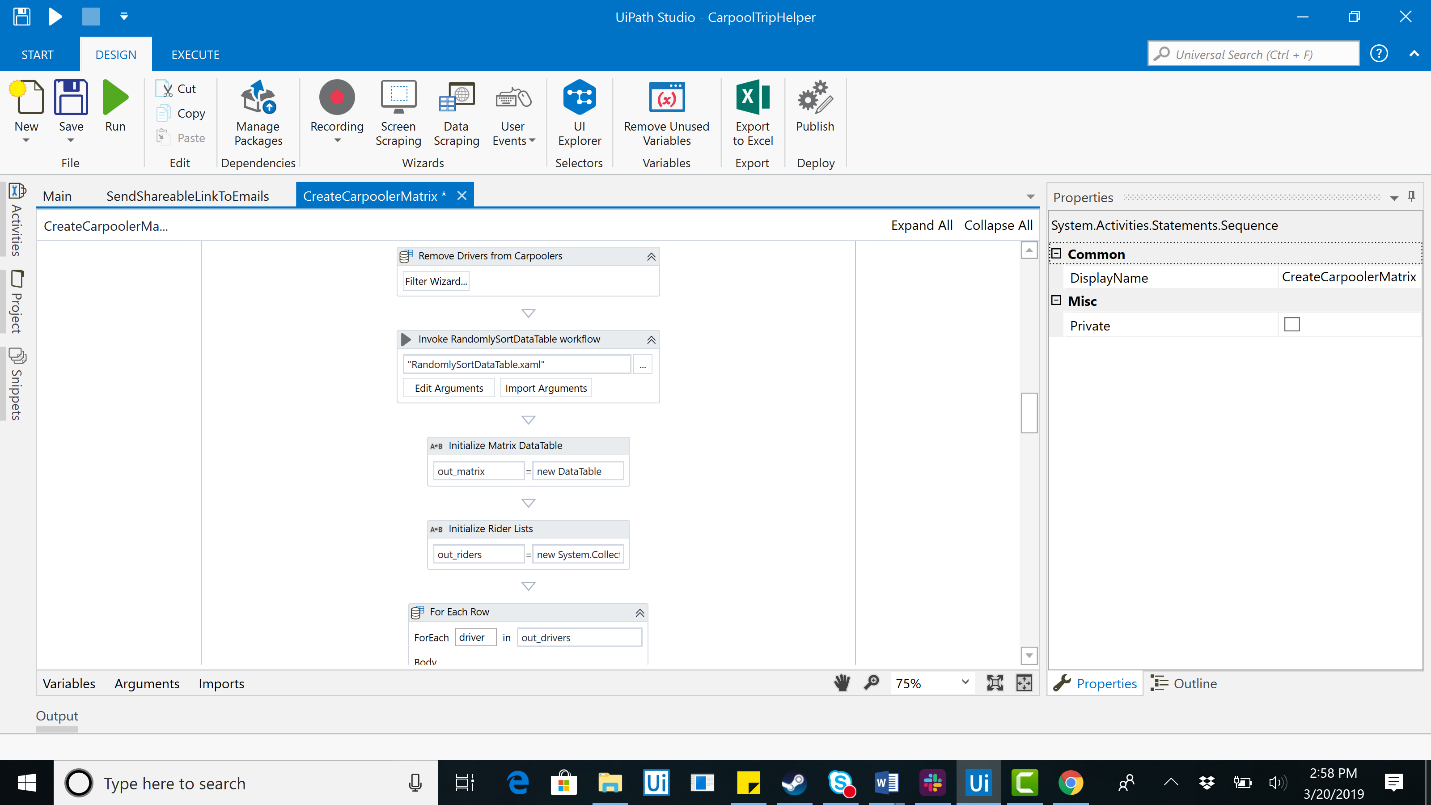
1. With this completed, let’s return to CreateCarpoolerMatrix after having passing the ‘carpoolers’ DataTable through our sorting module. To complete this module, we need a DataTable called out\_drivers to keep track of who will be driving and a DataTable called sortedBySeats which is a our carpoolers DataTable sorted by Number of Seats.



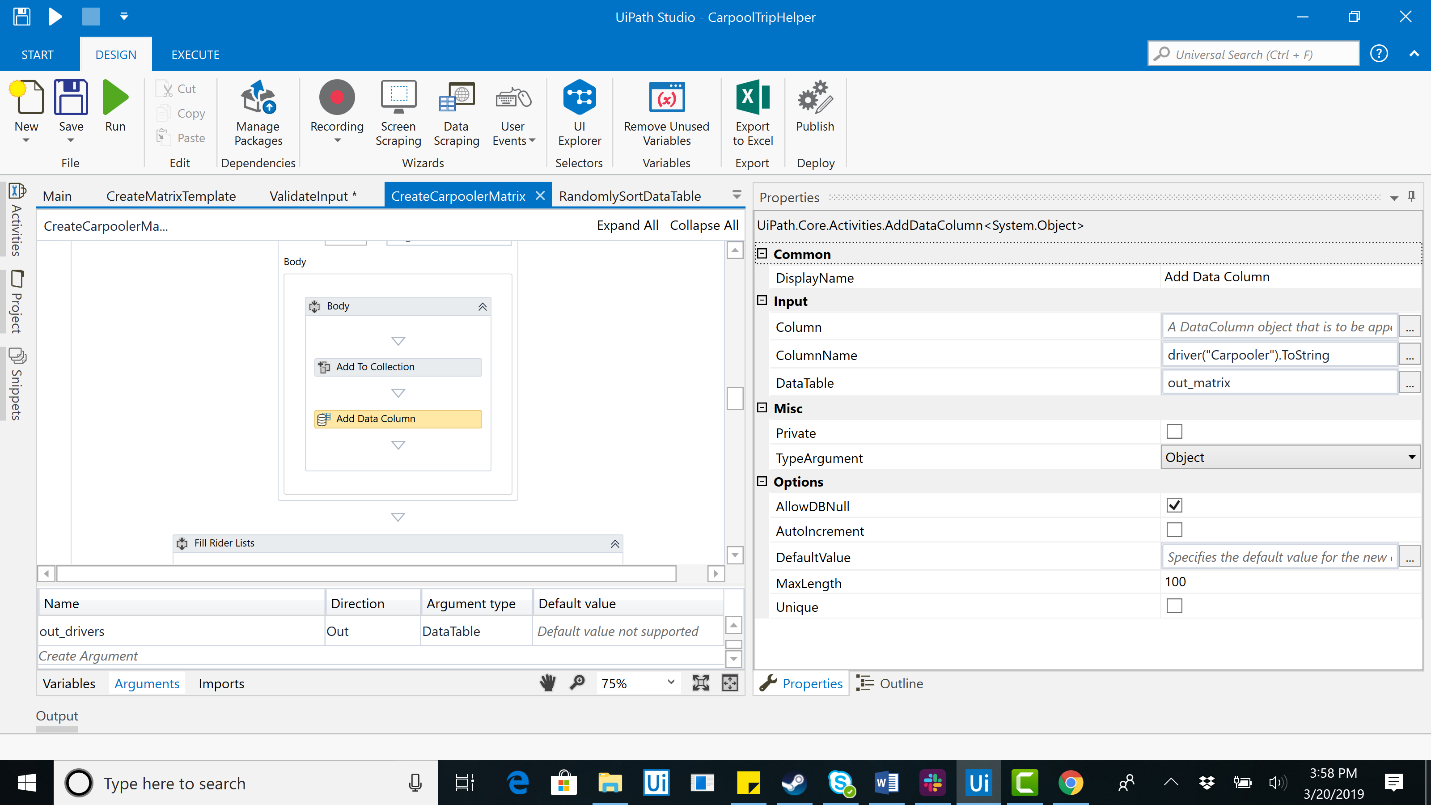
1. Now, let’s assign the driver. Use a for each row for the sortedBySeats DataTable. If ‘runningSeatTotal >= in\_carpoolers.Rows.Count’ then we assign runningSeatTotal to ‘runningSeatTotal + Integer.Parse(carpooler("Number of Seats").ToString)’. Otherwise we assign ‘carpooler("Willing to Drive?")’ to ‘No’.



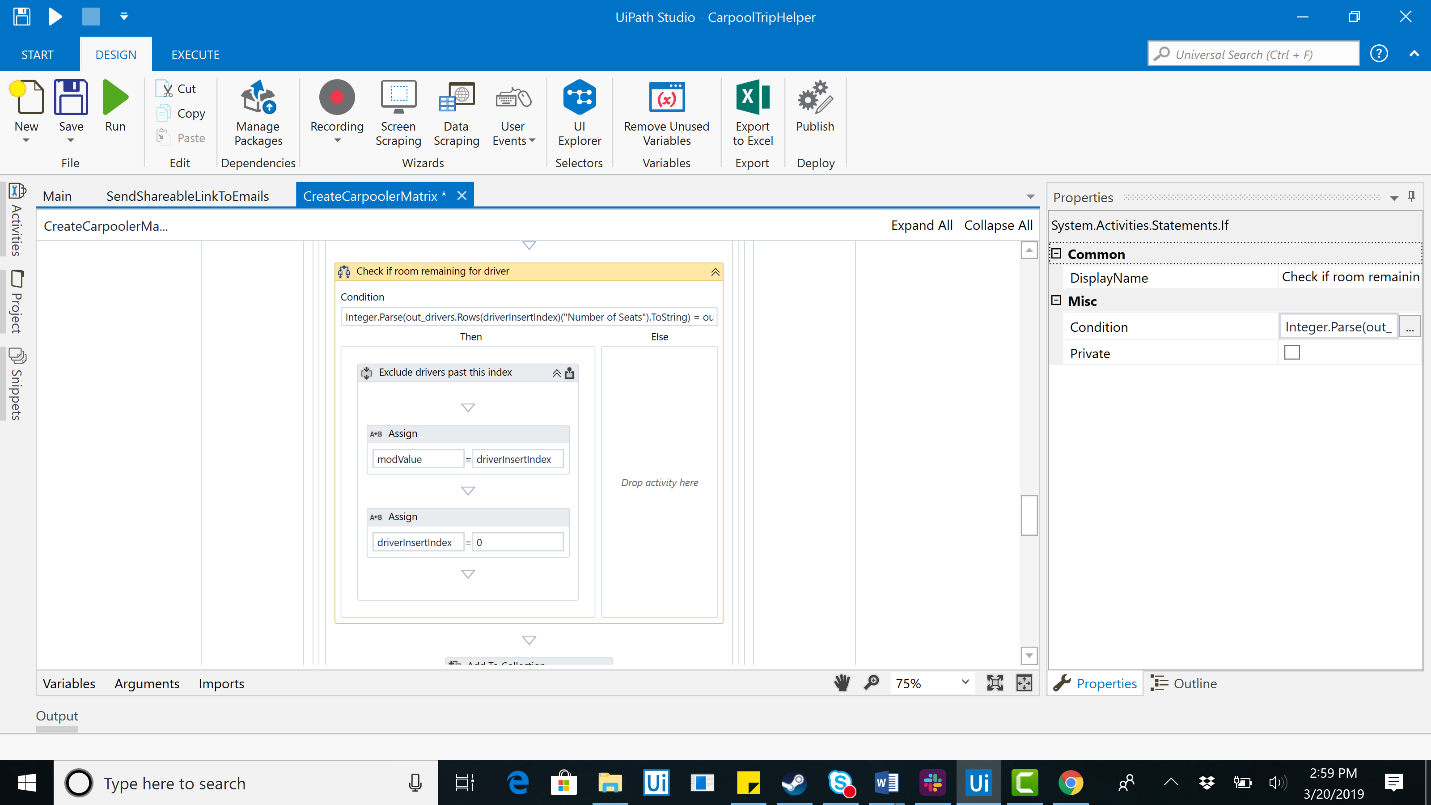
1. Next we need to remove the drivers from carpoolers as they will not need a seat and then sort the carpoolers randomly using the the module we built previously. Next let’s create an out\_matrix DataTable and assign out\_riders to new System.Collections.Generic.List(Of System.Collections.Generic.ICollection(Of String))’.



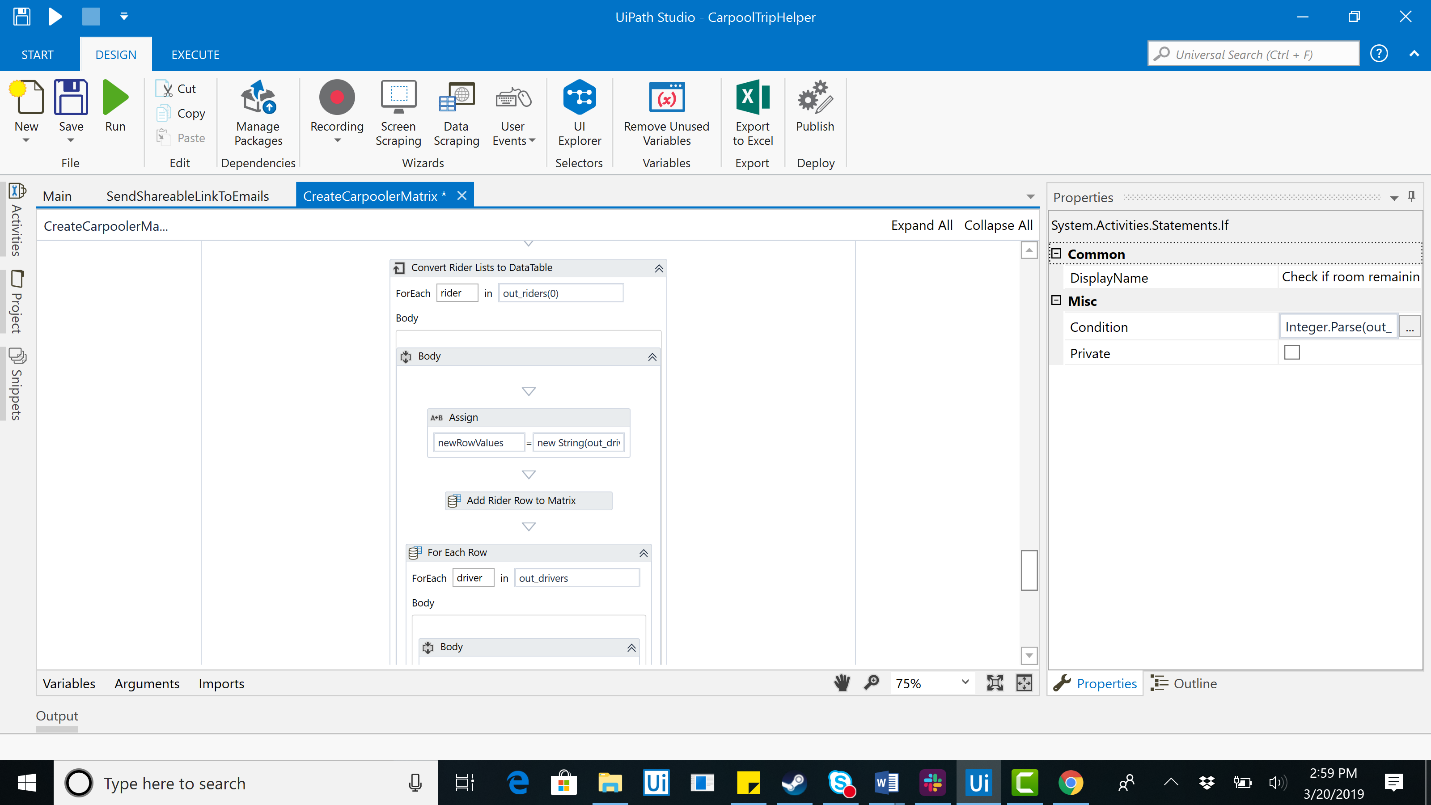
1. Next, for each driver we need to add them to the collection and then Add Data Column for driver(“Carpooler”).ToString for the out\_matrix DataTable we just created.



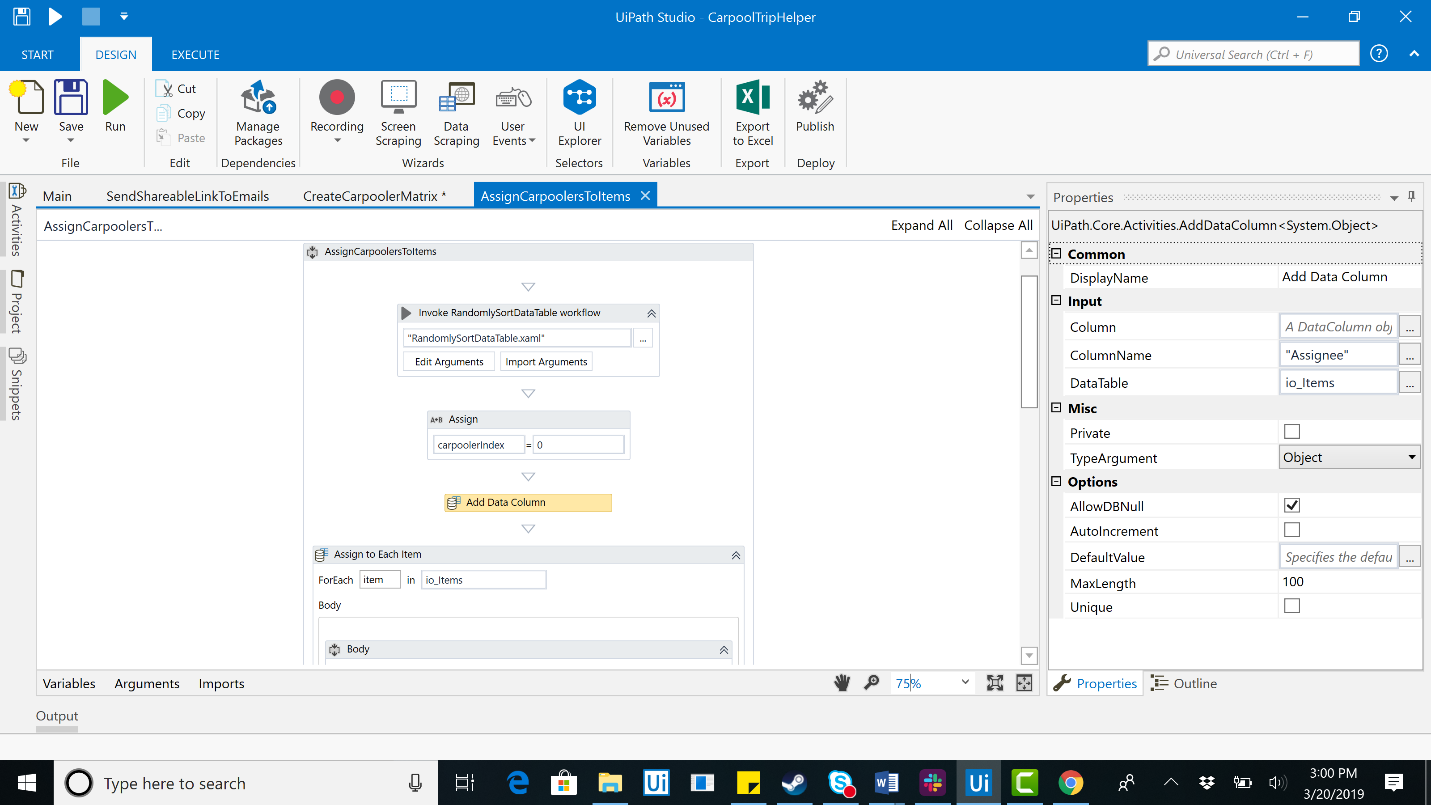
1. Next, we need to Check if room remaining for driver. If the condition ‘Integer.Parse(out\_drivers.Rows(driverInsertIndex)("Number of Seats").ToString) = out\_riders(driverInsertIndex).count – 1’ is satisfied we assign modValue to ‘driverInsertIndex’ and then driverInsertIndex to 0.



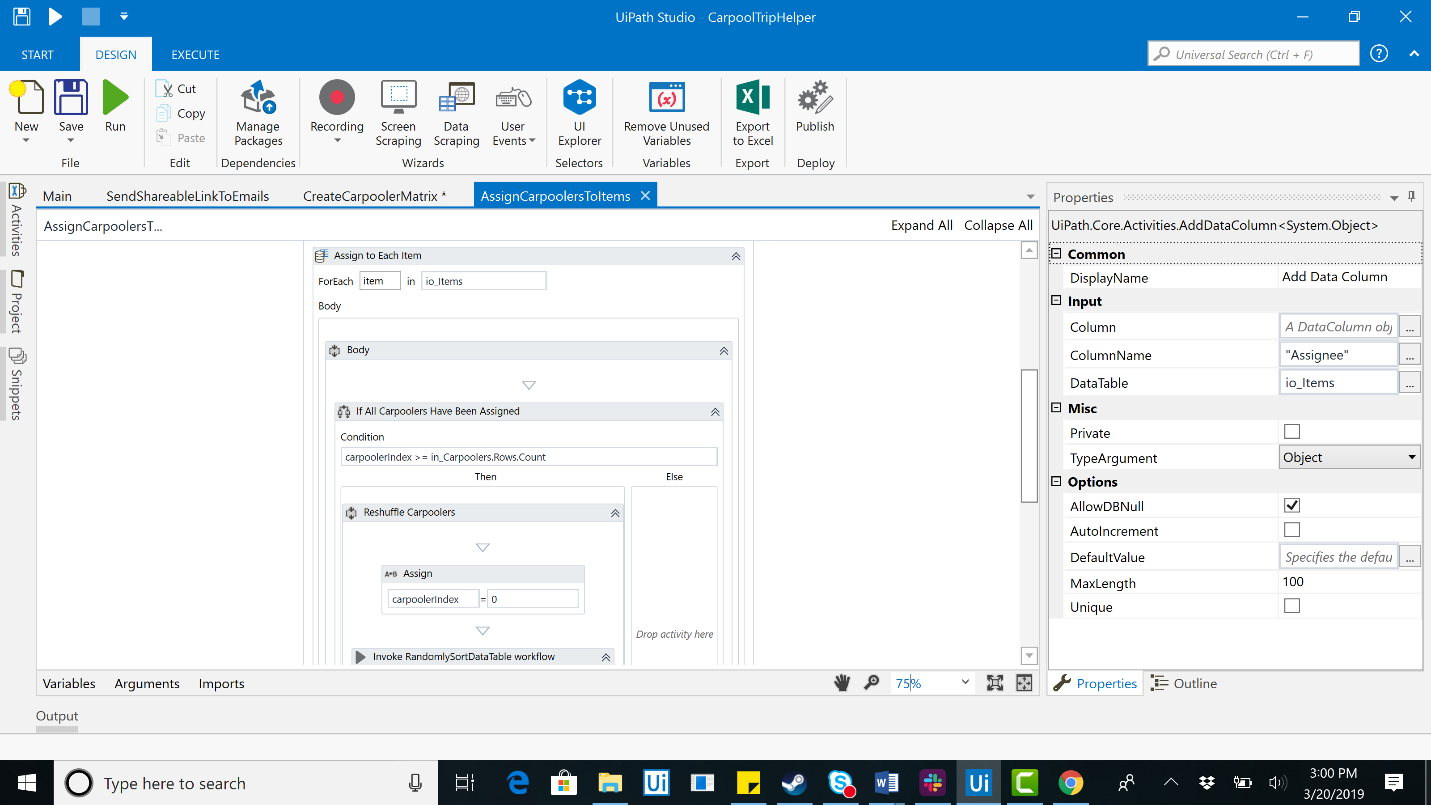
1. Finally, we convert riders lists to DataTable. For each rider in out\_riders(0), the newRowValues will be set equal to ‘new String(out\_drivers.Rows.Count - 1){}’ and we Add Rider Row to Matrix.



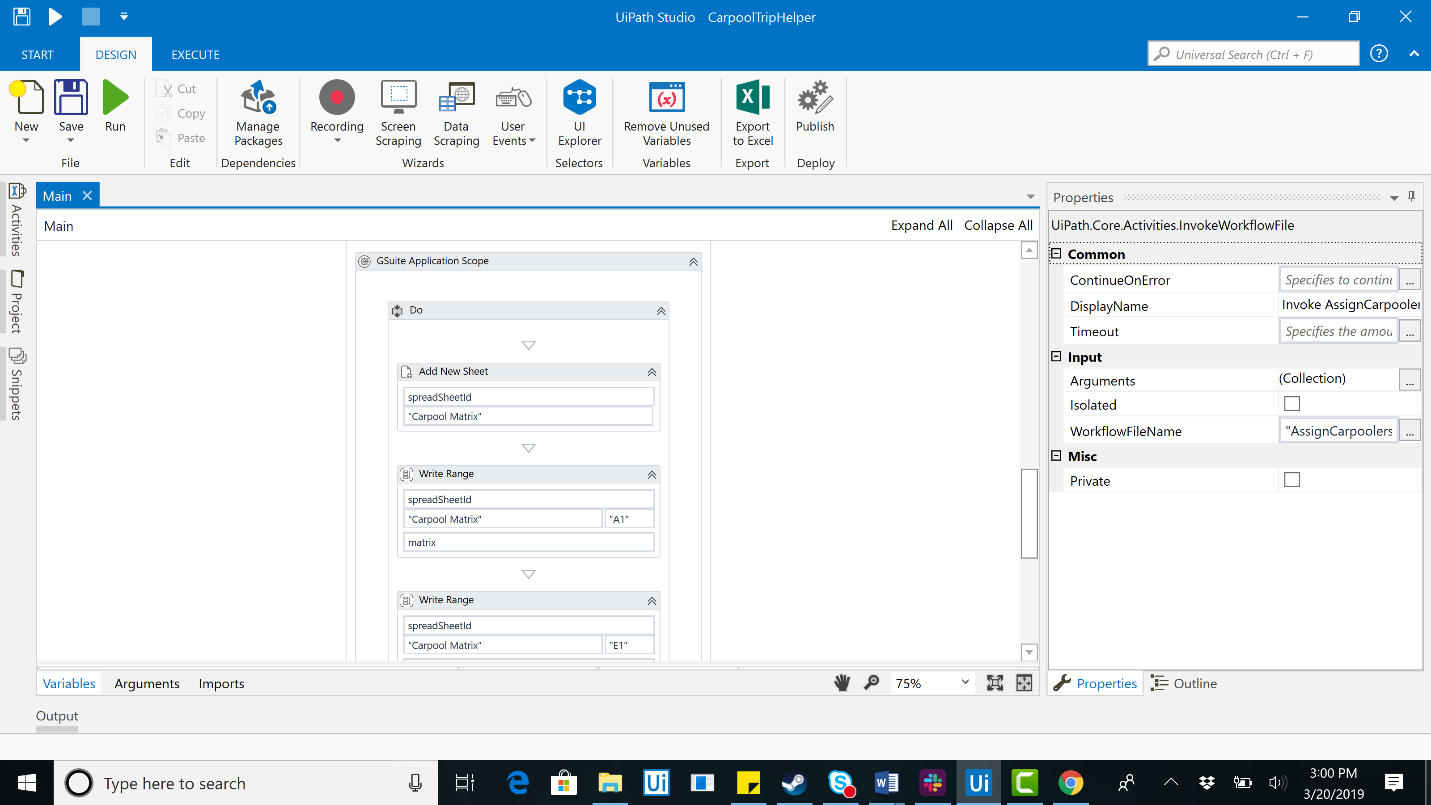
1. With this module completed, let’s do the AssignCarpoolersToItems module. We will again RandomlySortDataTable for in\_carpoolers. Next, set the carpoolerIndex to 0. Then, Add Data Column ‘Assignee’ to DataTable io\_items.



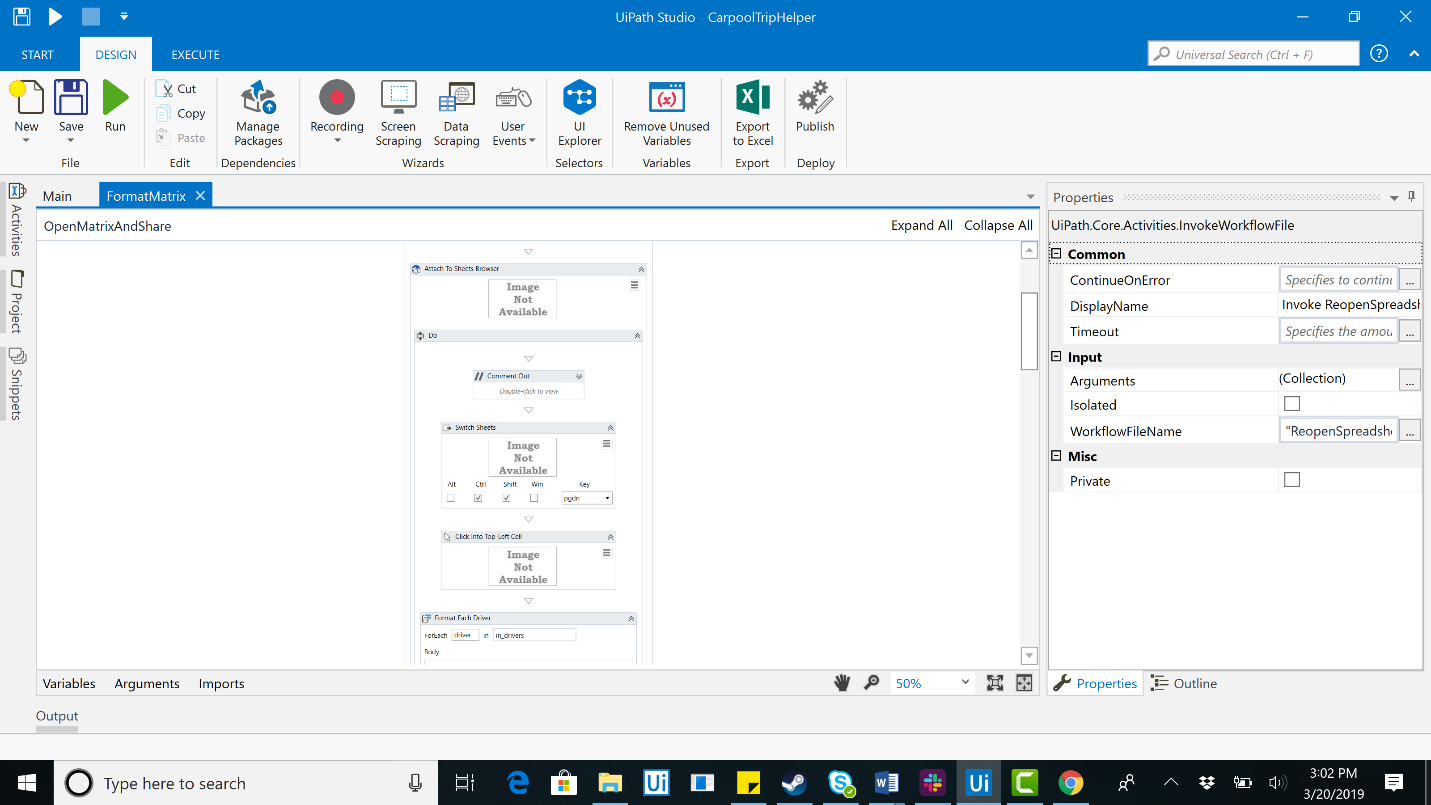
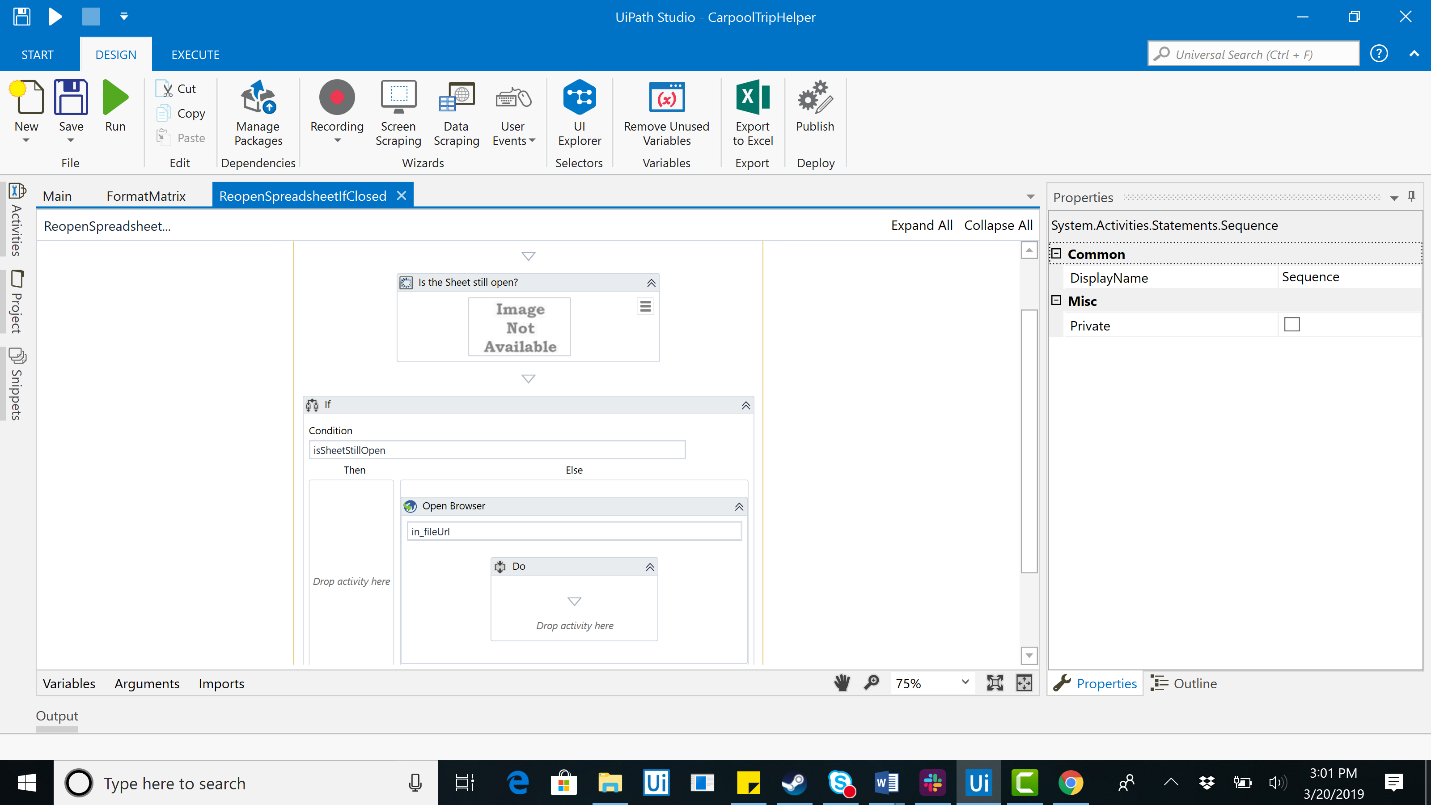
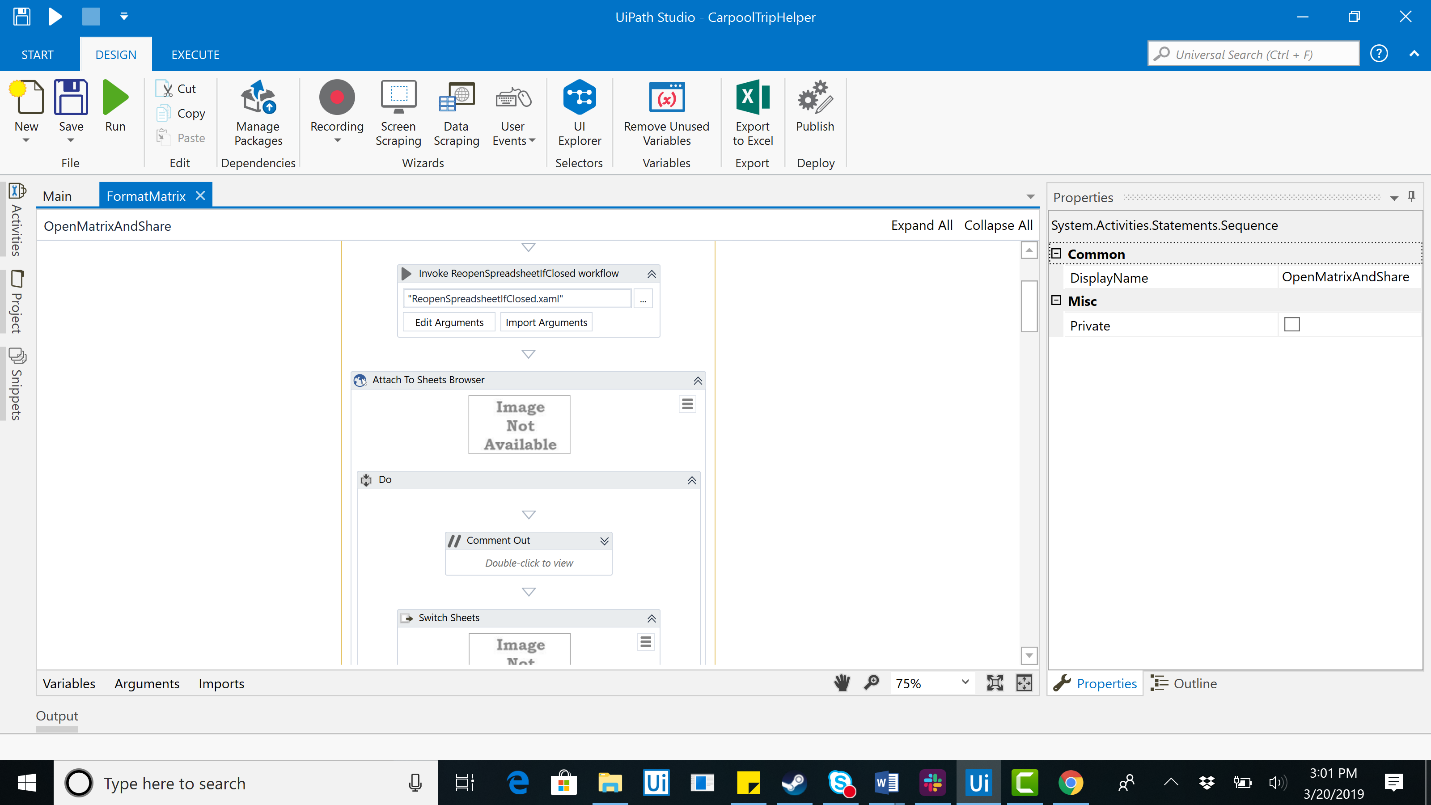
1. For each item in the collection, we check ‘carpoolerIndex >= in\_Carpoolers.Rows.Count’ and if the condition is satisfied we set carpoolerIndex to 0, randomly sort in\_carpoolers, then we set item(“Assignee”) to ‘in\_Carpoolers.Rows(carpoolerIndex)("Carpooler").ToString’ and carpoolerIndex to ‘carpoolerIndex + 1’.



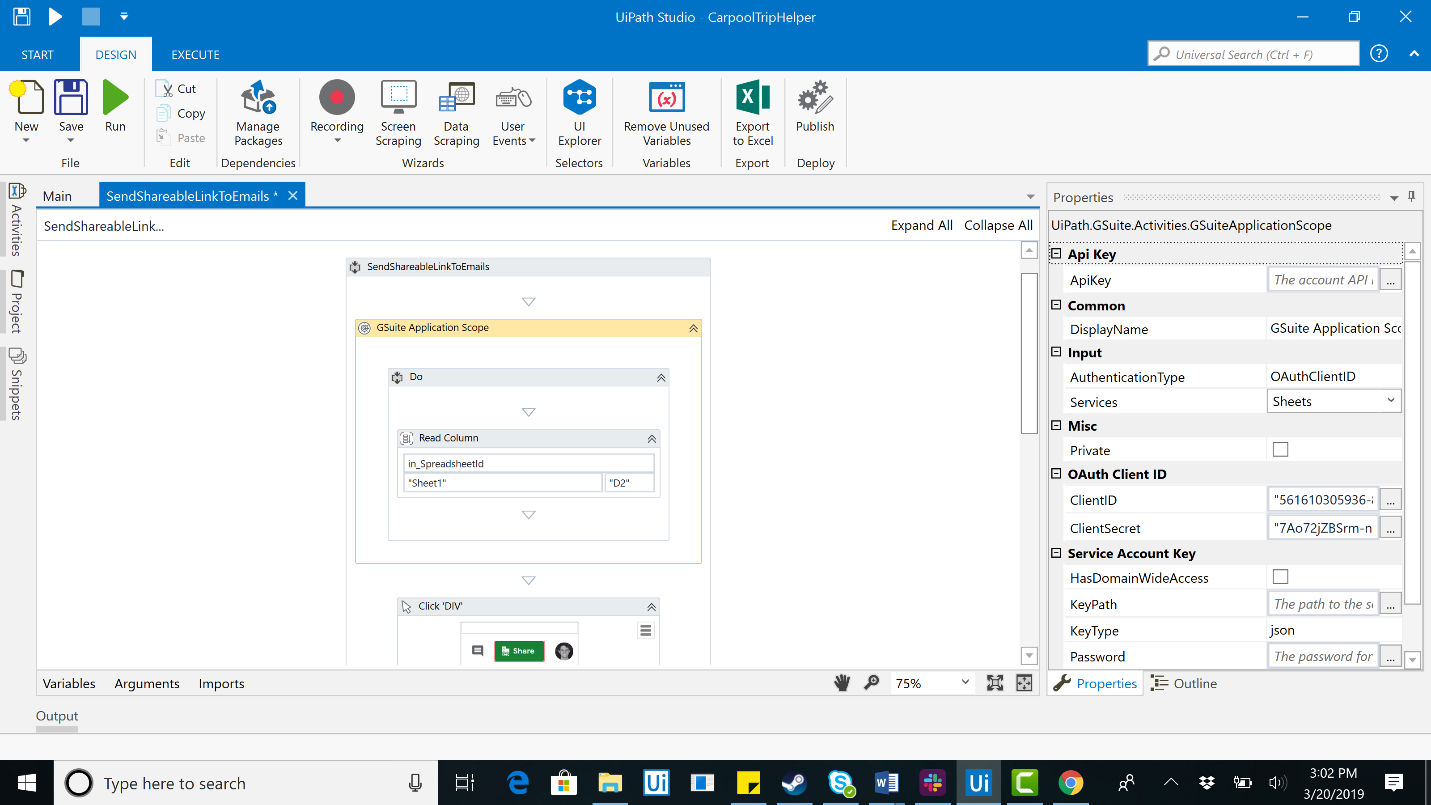
1. Now that Carpoolers have been assign, let’s return to Main.xaml. In the Google Application Scope, let’s Add New Sheet called Carpool Matrix. This is where the final sheet will be stored. Next, let’s Write Range to that sheet with the matrix we just created. Finally, let’s write the items to the sheet as well.



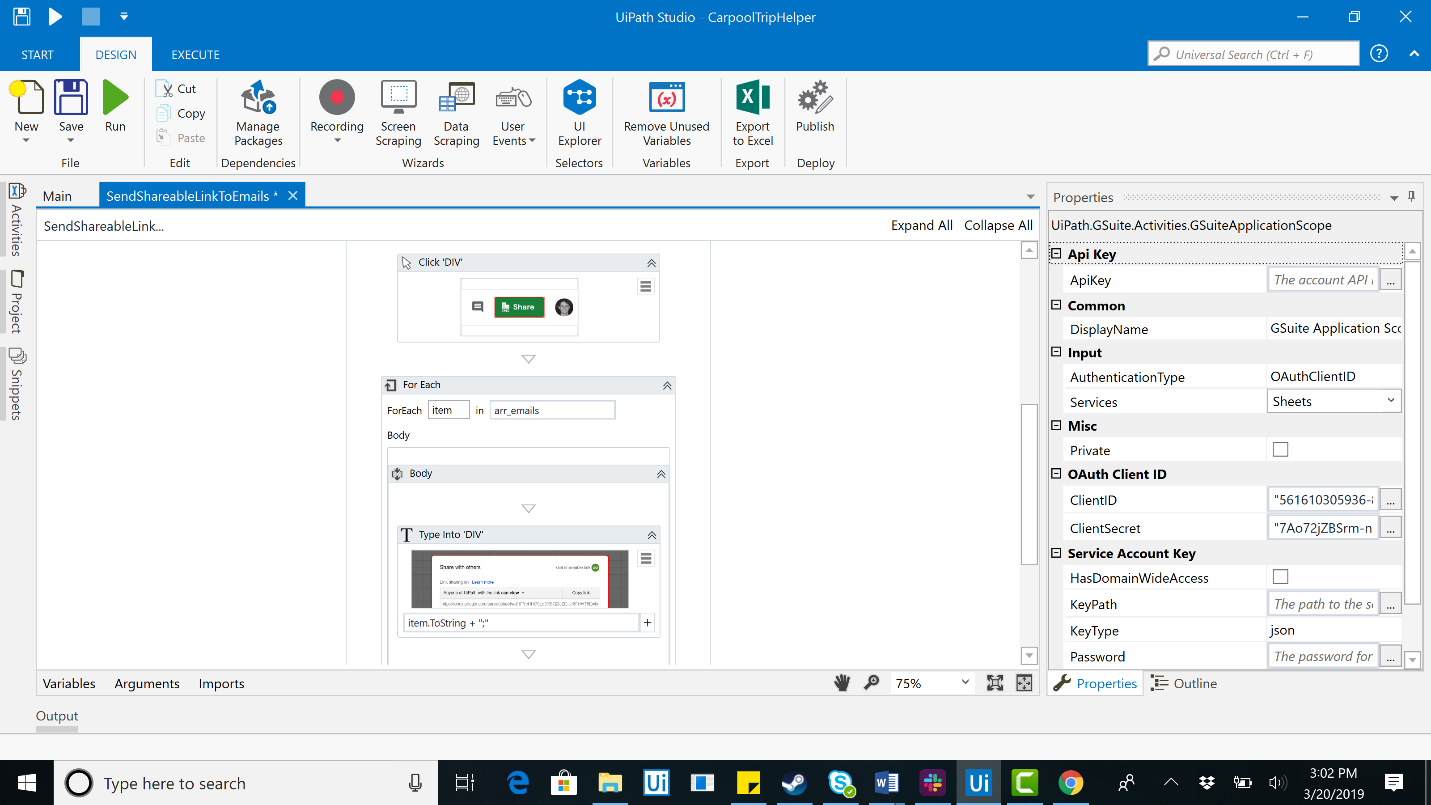
1. Now all we have left is to format and share. To format, we will send a series of hotkeys so long as the condition that the sheet is open has been satisfied. The following screenshots depict this. If you are unsure the hotkeys to format in Google Sheets, please reference the code.



1. The final step is to send a shareable to all relevant carpoolers. The user who entered the carpooling information should have included any e-mails they wanted sent. First you will read the column for e-mails in the original spreadsheet.



1. Next you will iterate through each of them e-mails and type them into the share modal. Finally, just click share.



The robot has been completed. You will no longer have to be the bad guy and be at fault for assigning people bad seats or items to bring!