Lyfter

Group 20

A. Requirements Engineering

Non Functional Requirements:

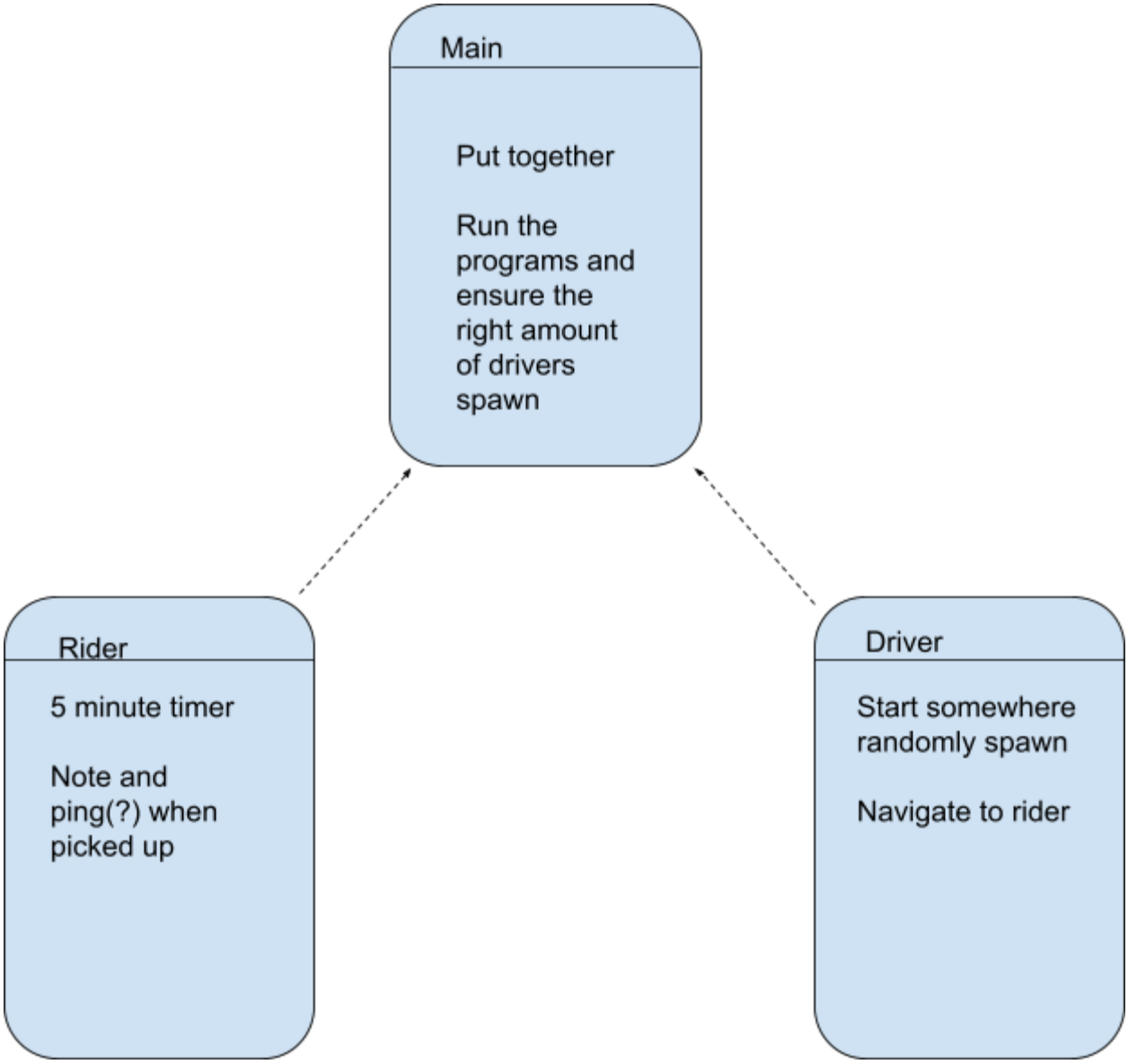
- Limit the region of the app to a specific city/town
- The time
- The law
- Use virtual environment
- Use python3
- Input = # of riders and locations in an x by y matrix, output = # of drivers

Functional Requirements

- A user should be able to log onto the app and based on their location and the number of drivers that guarantee a 5 minute or less wait time should be displayed
- Minimize # of drivers
- Carpooling/driver can pick up multiple riders in a single car (within a 5 minute wait time)
- Require the least number of drivers

B. System Modeling

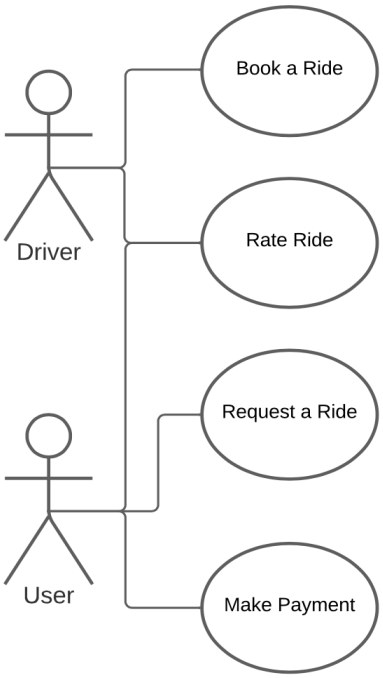
First/most basic model:



Grouping Riders Model:

								R6	R7
							R10	R9	R8
	R5	R1							
				R3		R4			
					R2				

Use Cases:



System	Ride - Calling System
Use Case	Request a Ride
Actors	User, Driver
Data	The User will use the system to request a ride from their destination that will transmit their starting point, their end point, estimated time of route, distance of car from riders.
Stimulus	The User will submit their ride request to the system.
Response	The System will determine whether there are any driver's nearby and notify whether the ride has been accepted including the details of the ride such as time and cost.
Comments	The user can only request for a ride with no additional stops along the way and should be able to cancel a ride before being confirmed with a driver.

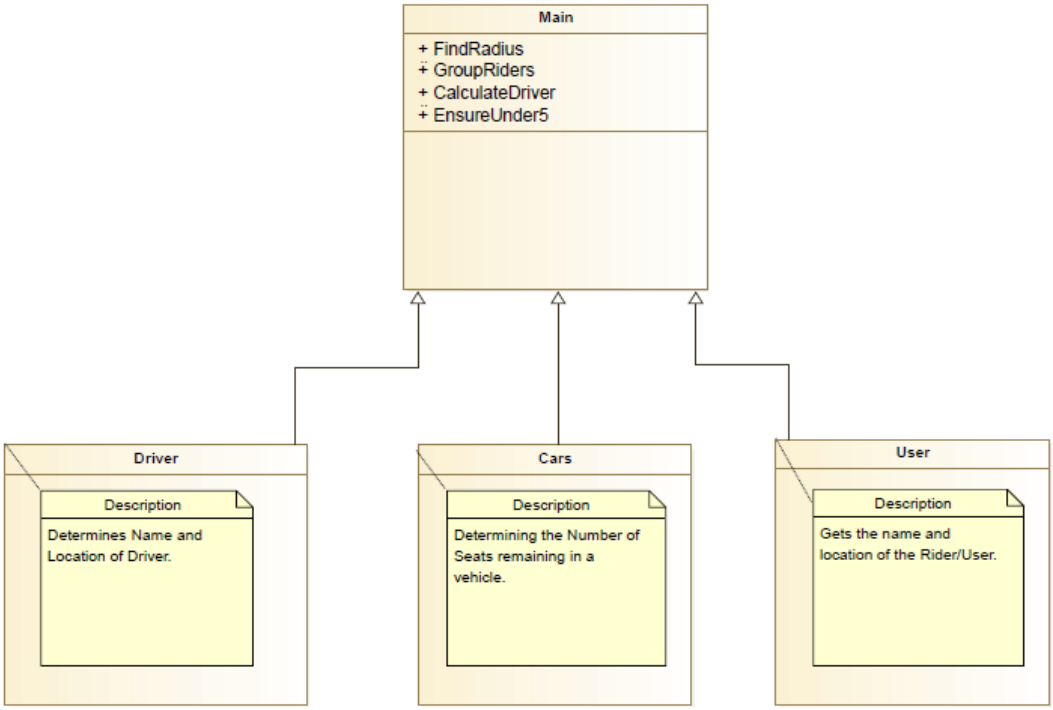
System	Ride - Calling System
Use Case	Book a Ride
Actors	User, Driver
Data	The Drivers will be given the information that a request for a ride from a user is nearby, the driver will then confirm that they want to take on this ride and pick up the rider.
Stimulus	Once the User sends the request for a ride the system will notify the riders.
Response	The Driver will determine whether they want to take this ride for the price and time then send their location to the rider.
Comments	Driver should only pick up rider if they are within 5 minutes of the rider in order to ensure that there is no complaint from the driver

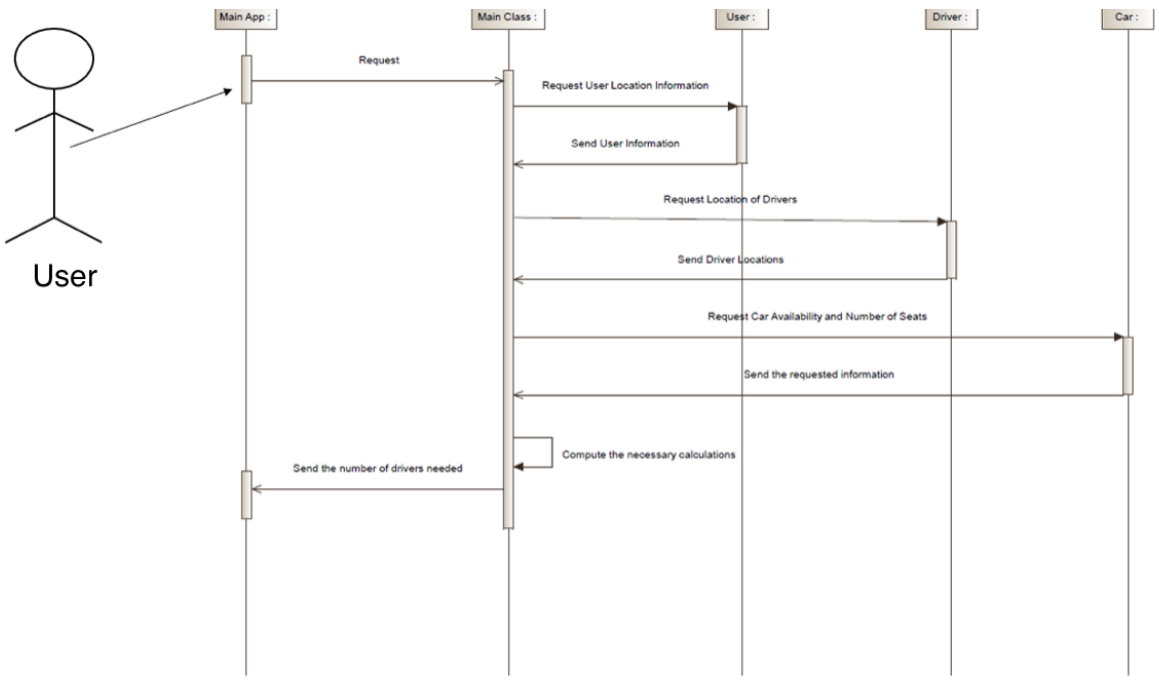
System	Ride - Calling System
Use Case	Rate a Ride
Actors	User, Driver
Data	After a ride is over the user and the driver should be able to give a rating of each other based on the ride. This 5 - star system should take into account personal experience of the user/driver and focus especially if the wait time was a problem for the ride.
Stimulus	Once the ride is finished the app will prompt the user and driver to submit a rating.
Response	The user and driver will each submit back a rating with any comments if provided.
Comments	If a rider experienced a bad or untimely ride it should be a red flag notified to the system immediately.

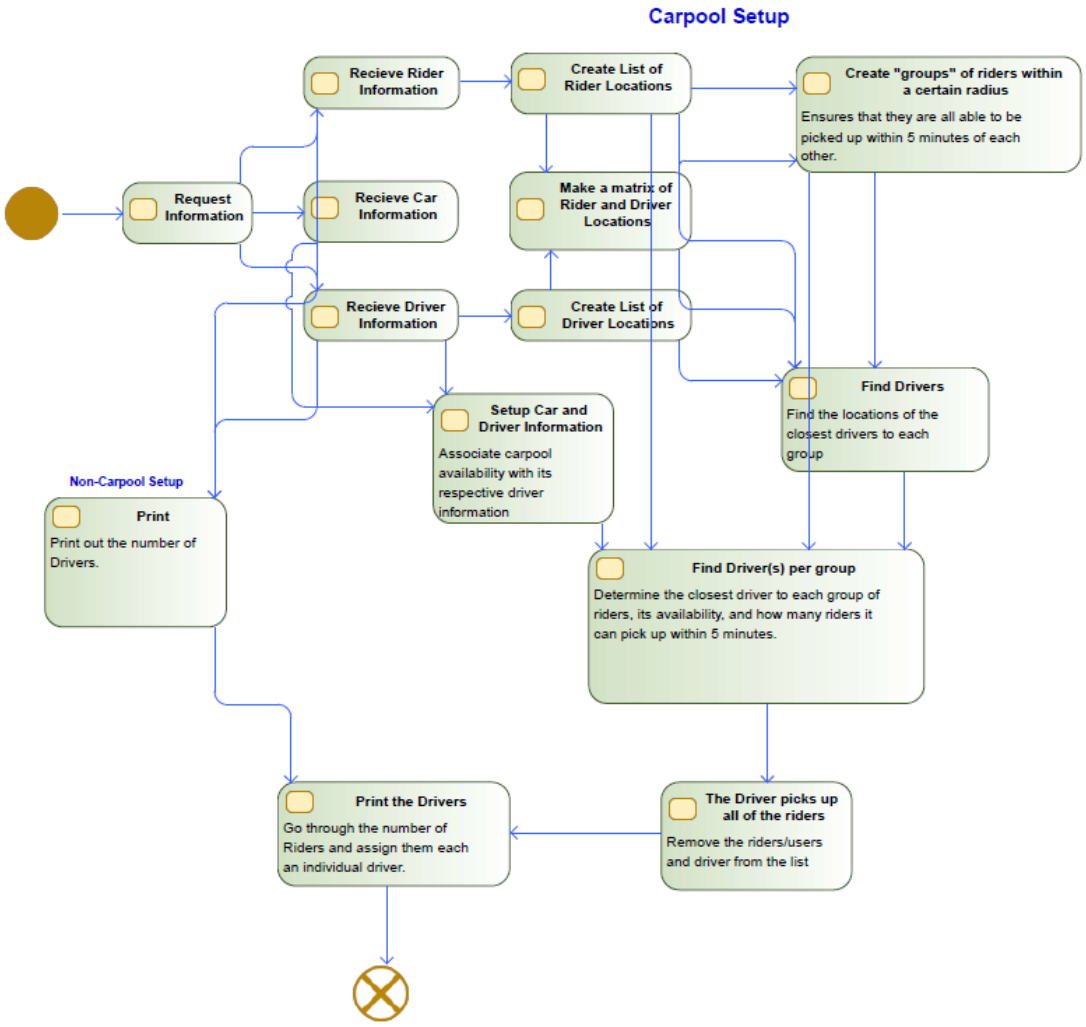
System	Ride - Calling System
Use Case	Make Payment
Actors	User, Driver
Data	Once the rider has finished the ride the system should be able to charge the cost of the ride to the driver plus any additional tip decided after the quality of the ride experienced by the rider.
Stimulus	After a ride has been confirmed the user should be alerted they are being charged the amount for the ride.
Response	The amount requested by the system should be given to the company.
Comments	The amount given from the user may not be just for the ride but also including tips for the driver and should be distributed back to the driver.

C. **Architectural Design**

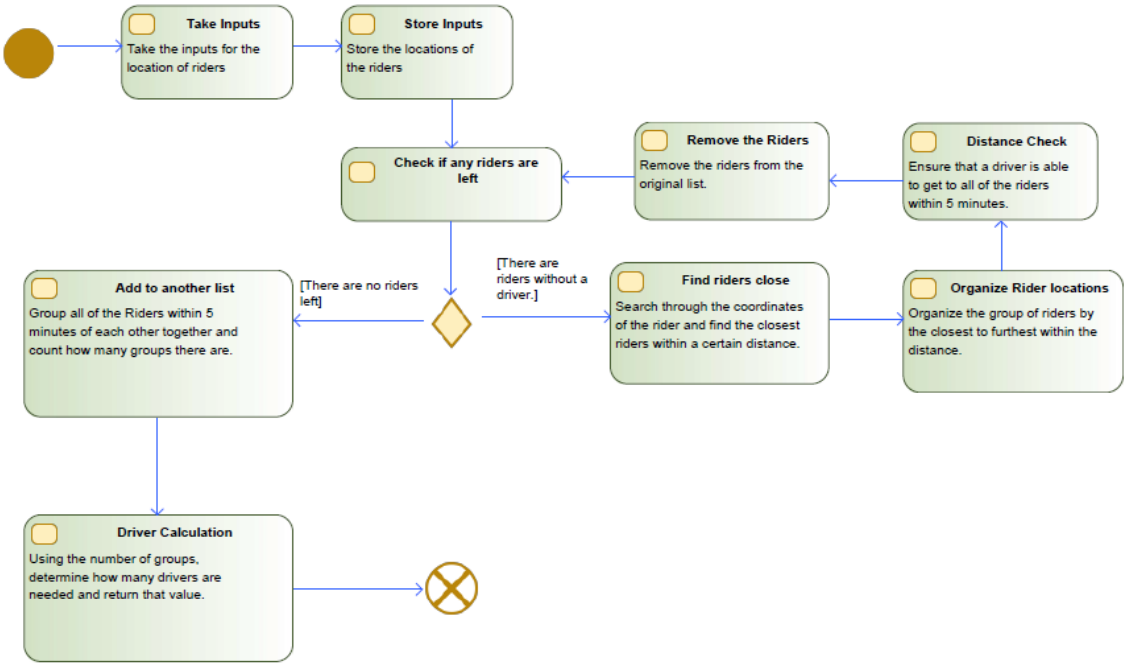
Original Design



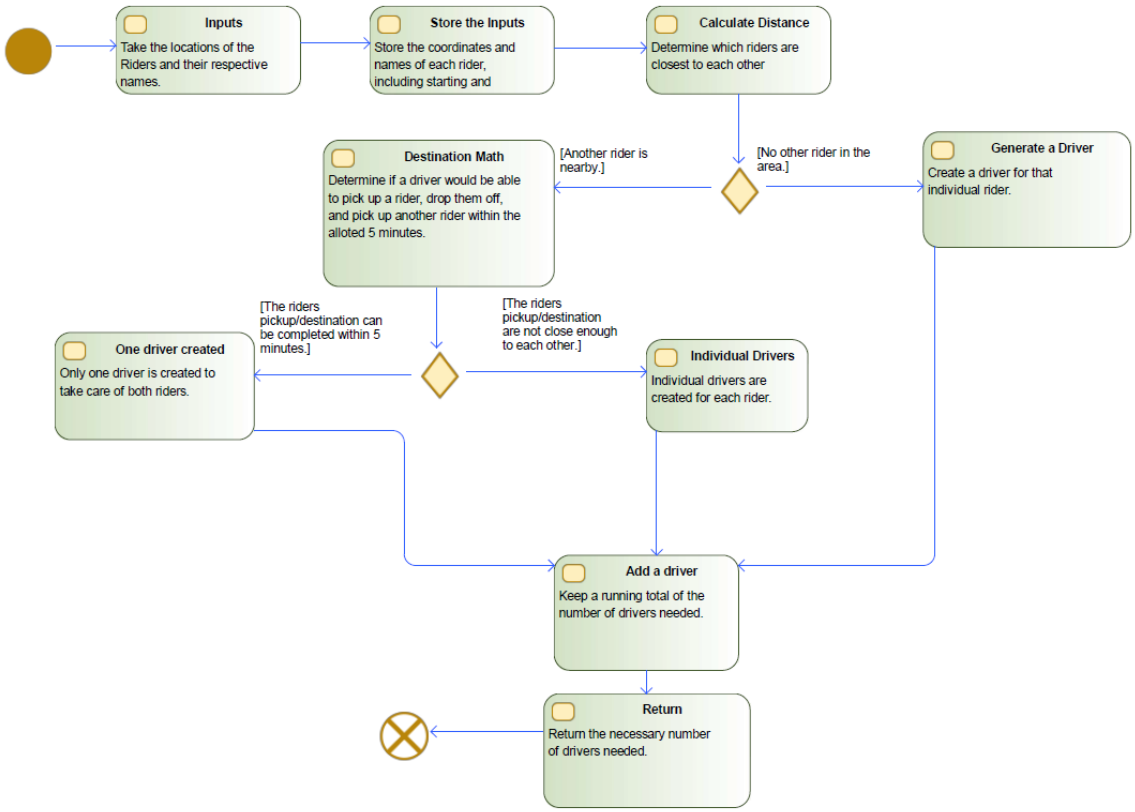

User
Main App :
Main Class :
User :
Driver :
Car :
Request
Request User Location Information
Send User Information
Request Location of Drivers
Send Driver Locations
Request Car Availability and Number of Seats
Send the requested information
Compute the necessary calculations
Send the number of drivers needed



Refactored Version:
Carpooling Activity Diagram

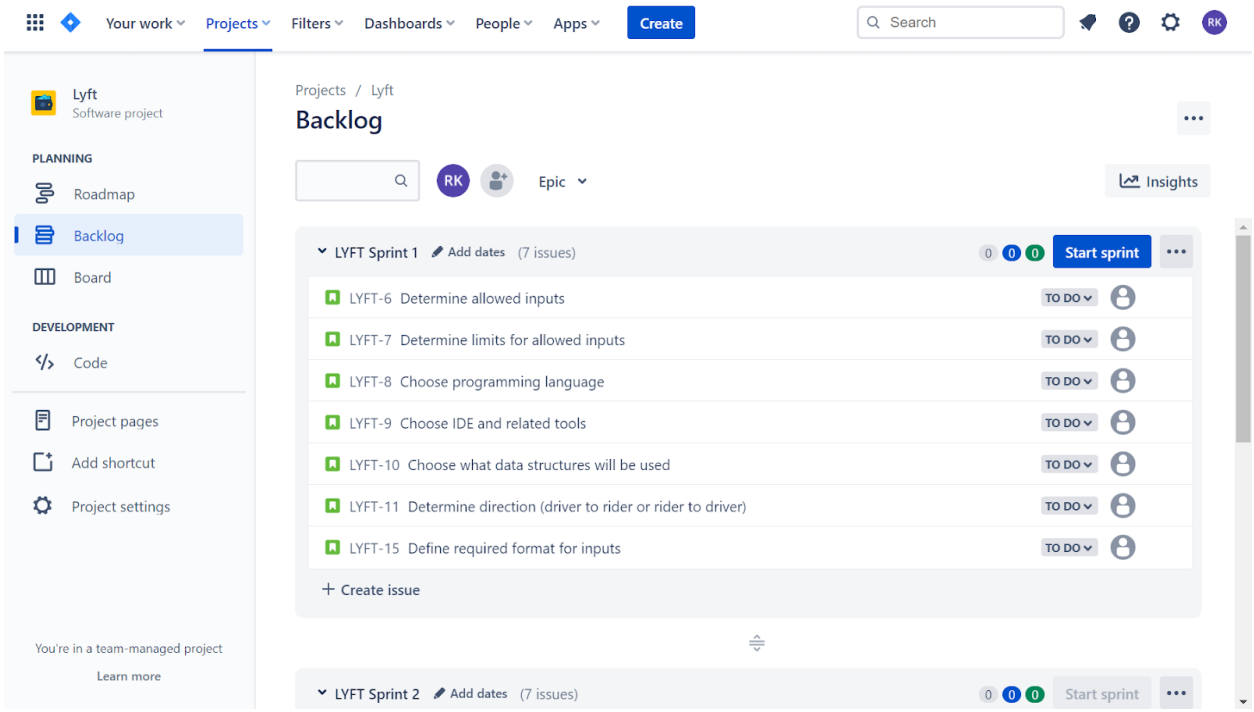


One Driver to One Rider Activity Diagram

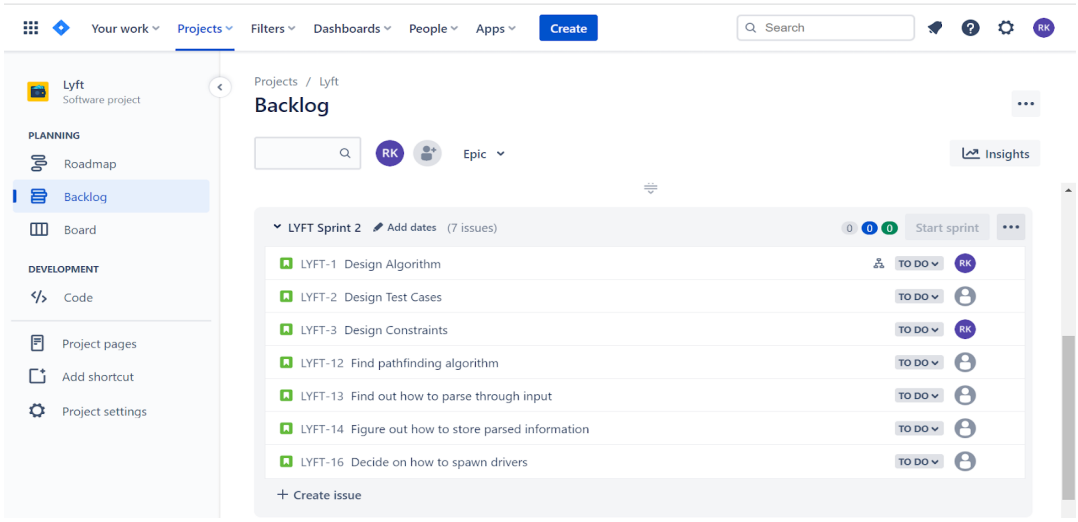


D. **Design and Implementation**

Jira Product Backlog:



We decided to make a txt file with user locations in “R0,2,3” format and implement the project in python3. There would be a 2D list (list of lists) or “grid” of vertical and horizontal streets with riders and drivers in the cells of the grid and moving through each cell takes 1 minute. Drivers find the riders and pick the rider closest to it that the driver can get to in 5 minutes or less.



In order to find the best optimized solution we gave the option to select whether car pooling or private car method was desired. The program will then prompt which user text file containing all of the randomized coordinates of riders.

We implemented a grouping algorithm to carpool and we made different rider txt files to test different cases. There are drivers spawning next to a rider so that the driver can reach the group of riders in 5 minutes or less through carpooling. We also have a 1 to 1 algorithm for private riding that needs the starting and ending points for each rider and if the driver can pick up and drop off a rider and still get to another rider in 5 minutes or less, the number of drivers can be optimized.

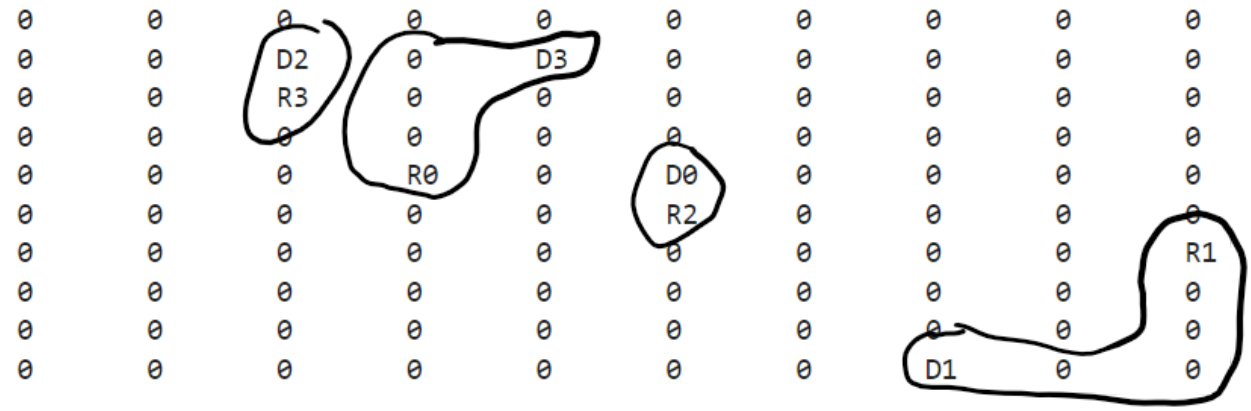
code in the github https://github.com/radical-teach/minor-project-group-20 (most recent version)

E. Software Testing

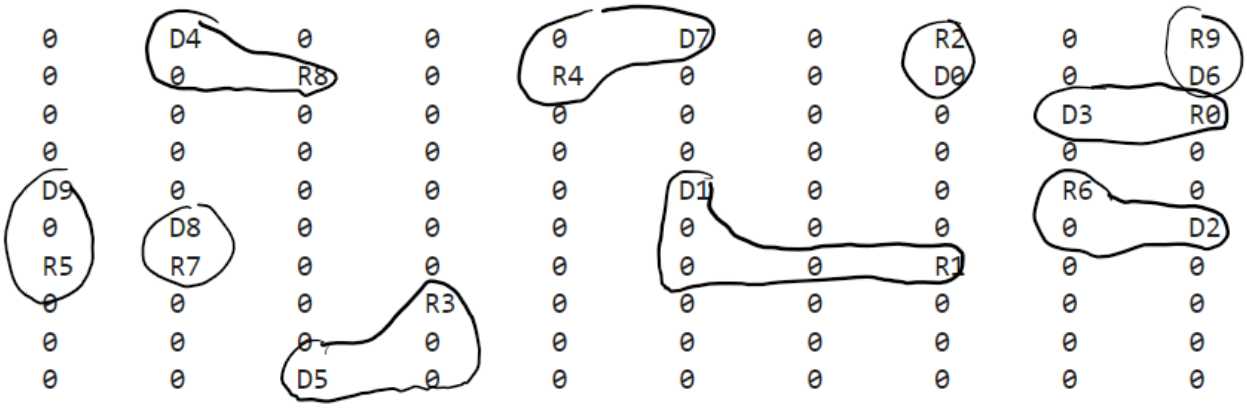
Driver Finds Closest Rider (Sprint 1)

Txt file	# of users	Random drivers	Drawing	Expected Output	Actual Output
users.txt	4	D0,4,5 D1,9,7 D2,1,2 D3,1,4	See Output1	D0 and R2 D1 and R1 D2 and R3 D3 and R0	D0 and R2 D1 and R1 D2 and R3 D3 and R0
testTen1.txt	10	D0,1,7 D1,4,5 D2,5,9 D3,2,8 D4,0,1 D5,9,2 D6,1,9 D7,0,5 D8,5,1 D9,4,0	See Output2	D0 and R2 D1 and R1 D2 and R6 D3 and R0 D4 and R8 D5 and R3 D6 and R9 D7 and R4 D8 and R7 D9 and R5	D0 and R2 D1 and R1 D2 and R6 D3 and R0 D4 and R8 D5 and R3 D6 and R9 D7 and R4 D8 and R7 D9 and R5

Output1:



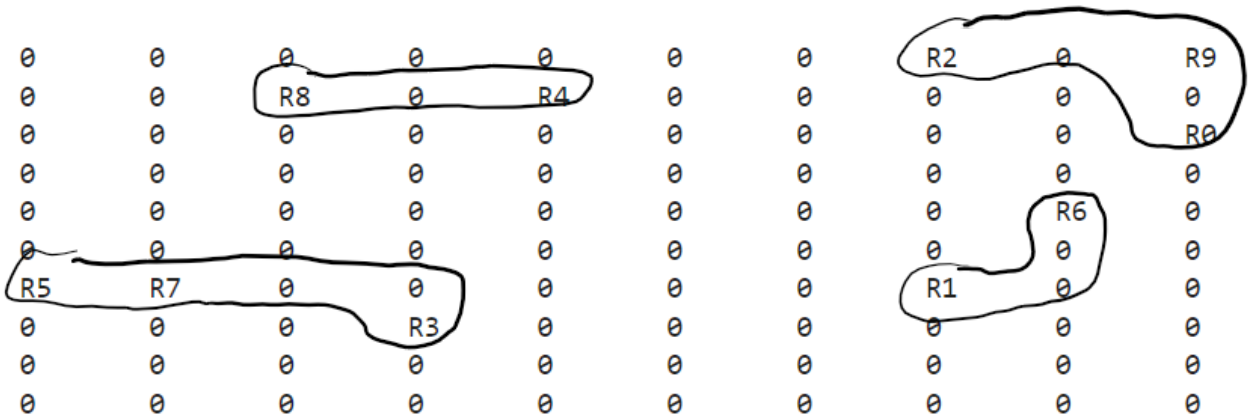
Output2:



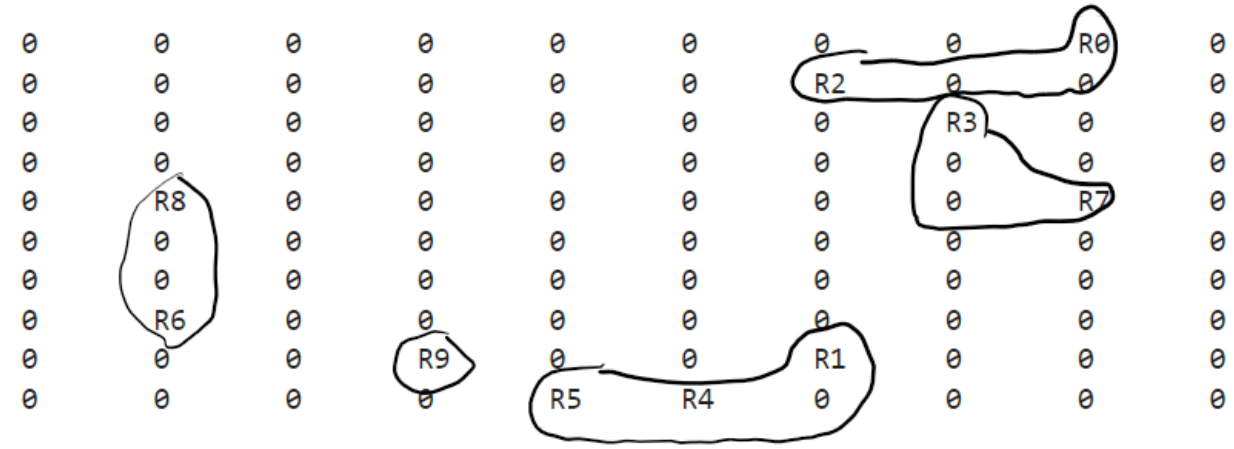
Carpooling

Txt file	# of users	Drawing	Expected Output	Actual Output
testTen1.txt	10	See Output3	4	4
testTen2.txt	10	See Output4	5	5
testFifty.txt	50		37	37
testHundred1.txt	100		83	83
testHundred2.txt	100		84	84

Output3:



Output4:



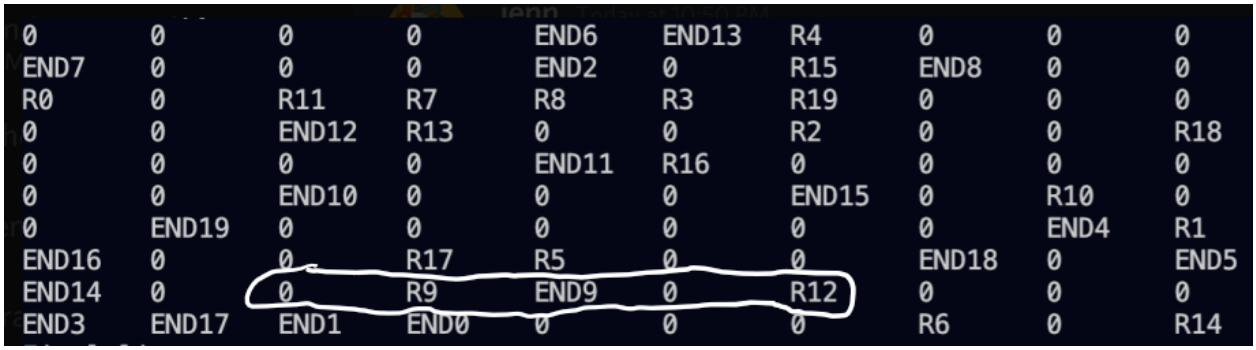
One Rider to One Driver

Txt file	# of users	Drawing	Expected Output	Actual Output
user.txt	5	See Output8	4	4
usersidk.txt	20	See Output9	19	19

Output8:

0	0	END5	END3	0
END1	R0	R2	0	0
0	0	END2	R1	0
R4	END4	R3	END0	0
0	0	R5	0	0

Output9:



0	0	0	0	END6	END13	R4	0	0	0
END7	0	0	0	END2	0	R15	END8	0	0
R0	0	R11	R7	R8	R3	R19	0	0	0
0	0	END12	R13	0	0	R2	0	0	R18
0	0	0	0	END11	R16	0	0	0	0
0	0	END10	0	0	0	END15	0	R10	0
0	END19	0	0	0	0	0	0	END4	R1
END16	0	0	R17	R5	0	0	END18	0	END5
END14	0	0	R9	END9	0	R12	0	0	0
END3	END17	END1	END0	0	0	0	R6	0	R14

F. Evaluation

Verification:

In order to verify the code, multiple tests were ran with as many variables as possible being changed. Some of the variables that were changed include the number of riders, locations of riders, and the size of the grid. Testing that would cause errors was also completed, ensuring that the expected error would occur.

Validation:

In order to fully validate the code, the requirements were consistently being compared to the results and the system. Ensuring that the requirements were all being fulfilled was one of the core parts of each Scrum, with a much more detailed version being done at the beginning and end of each Sprint.