

## **Project Problem #1: OptimalDelivery**

OptimalDelivery is a logistics company that wants to move shipments from Producer to Consumer in a distance-optimal manner. They approach your team to help them design a route planning system that enables their fleet of T trucks to pick up from P (predetermined) pickup points, and deliver to D receiving points, so as to incur minimum gas costs, i.e., the fleet has to drive minimum distance collectively.

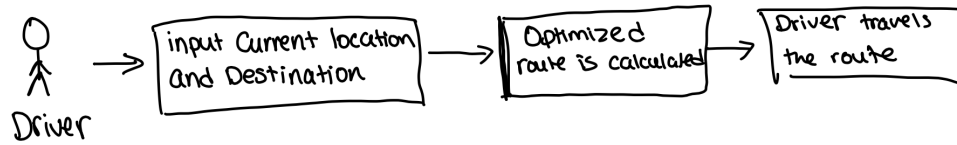
### **Requirements Engineering:**

- **User Requirements**
  - The truck drivers (users) will be provided optimal distances from the producer to consumer.
- **System Requirements**
  - New optimal routes will be generated constantly in case of an accident that makes a route unavailable. (using dijkstra's algorithm)
  - In the event of an accident that makes the optimal route unavailable the system will return the next best available route
  - Any employee authorized (i.e fleet manager, driver, etc) can view the trucks routes
  - Successfully blocks non employees from accessing truck routes
- **Functional Requirements**
  - Accept users input, current location and destination
  - The system generates a route from the Producer to the Consumer which will utilize the shortest distance, where shortest distance incurs minimal gas costs.
    - The optimal route is found by comparing the total distances of all the routes between current location and destination.
  - If a route is not available, a message will be displayed
  - This system should not give a false route, a route that doesn't exist or is unavailable
  - Access should only be given to those authorized views of the routes of drivers
- **Non-functional Requirements**
  - Organizational requirement
    - Users need to authenticate themselves by inputting their work id number
  - Efficiency Requirements
    - Make Certain the system is always able to output the shortest distance for fleet
      - Incur Minimum gas costs
  - Usability Requirements
    - Make certain system is able to be easily used and understood by OptimalDelivery
      - No Learning Curve, intuitive design
  - Performance Requirements

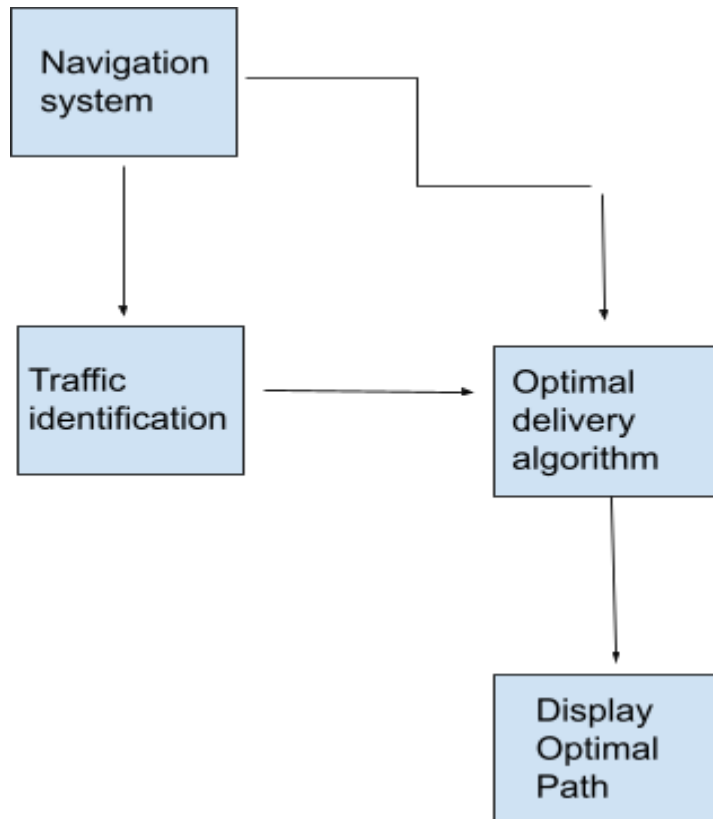
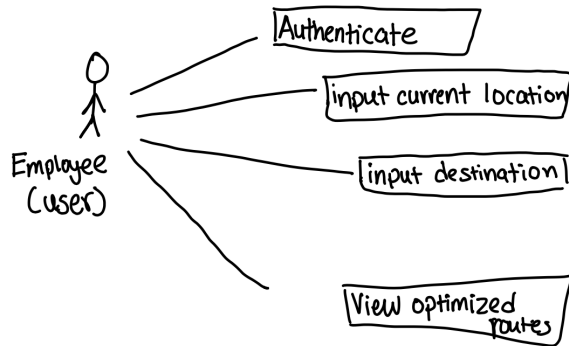
- Make certain that the system is able to run and output no matter the input size
  - With correct input, will always output shortest distance
- Environmental Requirements
  - Ensure that system is able to output the shortest distance possible
    - Shortest Distance = minimum gas costs
      - Minimum gas consumption = less environmental pollution
- Safety Requirements
  - Guarantee that routes give by system are safe roads to be driven on
    - Roads can handle traffic of fleet

### **System Modeling:**

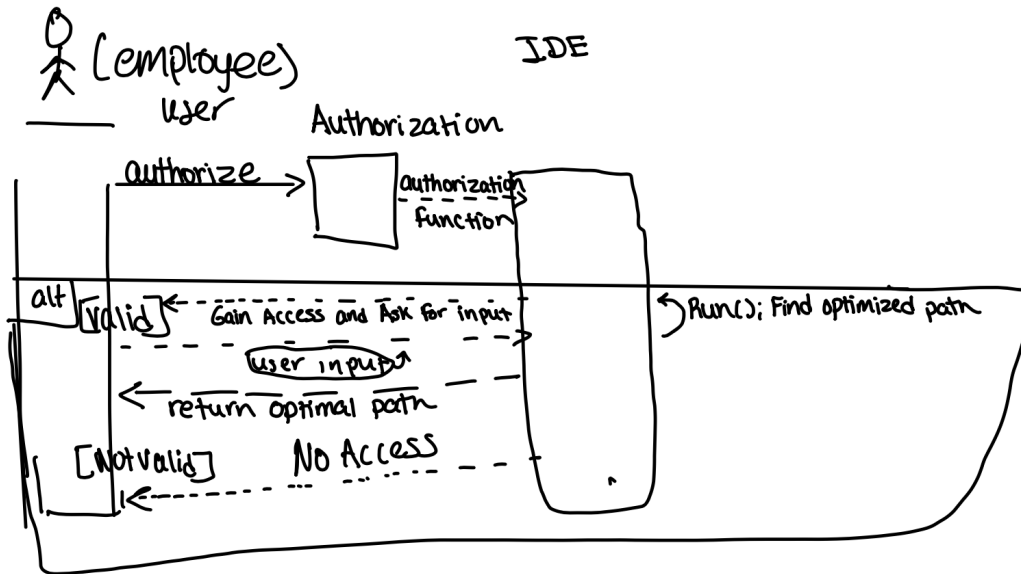
## User Requirements



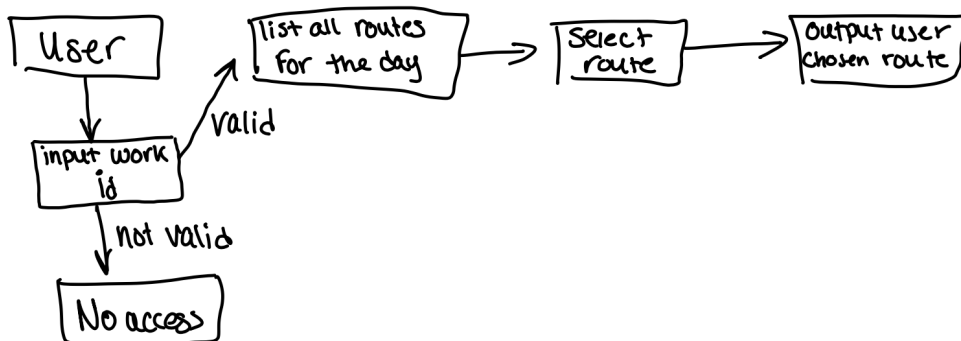
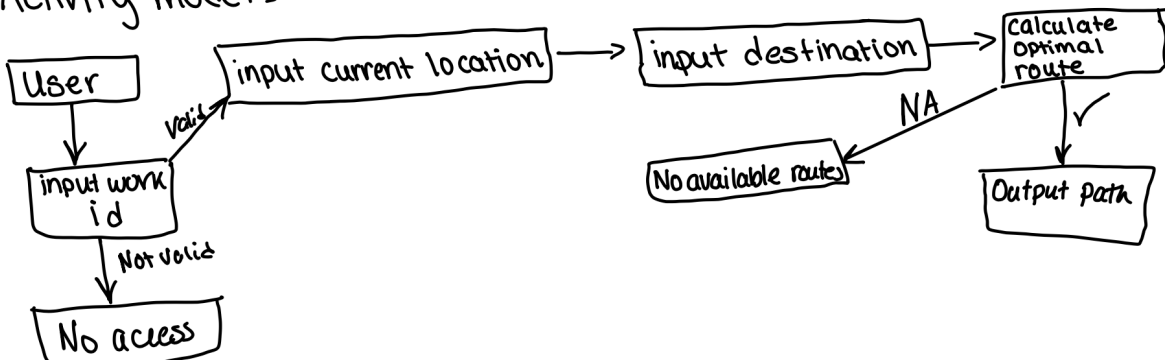
## Use cases:



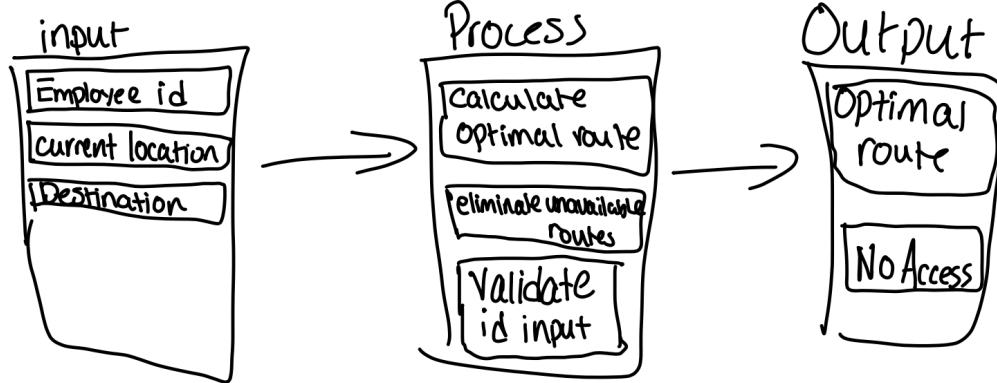
# Sequence Diagram



## Activity models



## Architecture

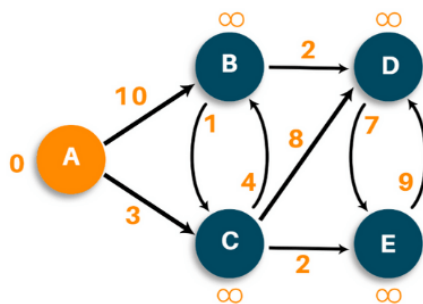


\* Only those authorized will get the optimal route; otherwise NO ACCESS

## Design and Implementation:

- Algorithm to code: Dijkstra's Algorithm
  - Provide prompt to the user which requests the possible routes into graph dictionary
  - The starting location(Producers) is entered into the given prompt followed by the user entering the Ending Location (Consumer)
  - Using the given coordinates in combination with the graph dictionary, the most direct path is calculated. The shortest distance that the algorithm will incur minimal gas costs.

## Dijkstra's Algorithm

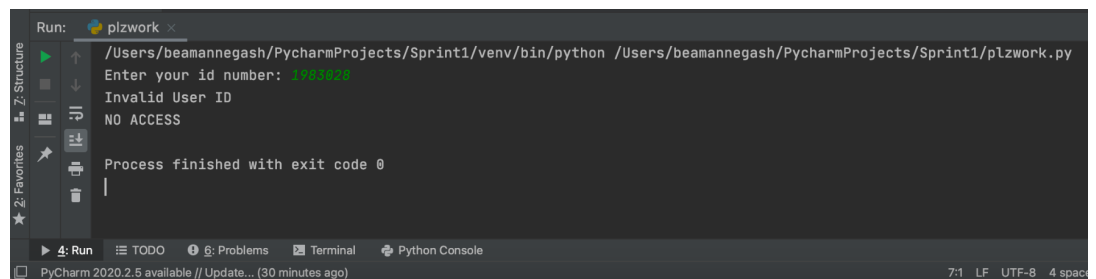


- Implement authorization function
  - Authorizes access to the system, by inputting worker id

- Implement random accidents, altering available optimal routes
  - Using a random number generator, randomly delete routes
  - This can happen at random
- Implementation of GUI
  - Prompt will come up asking for current location and destination
  - If path is available the route will be confirmed as the shortest possible distance and displayed
  - If the path is not available the GUI will relay a message stating that shortest path is not possible

### Software Testing:

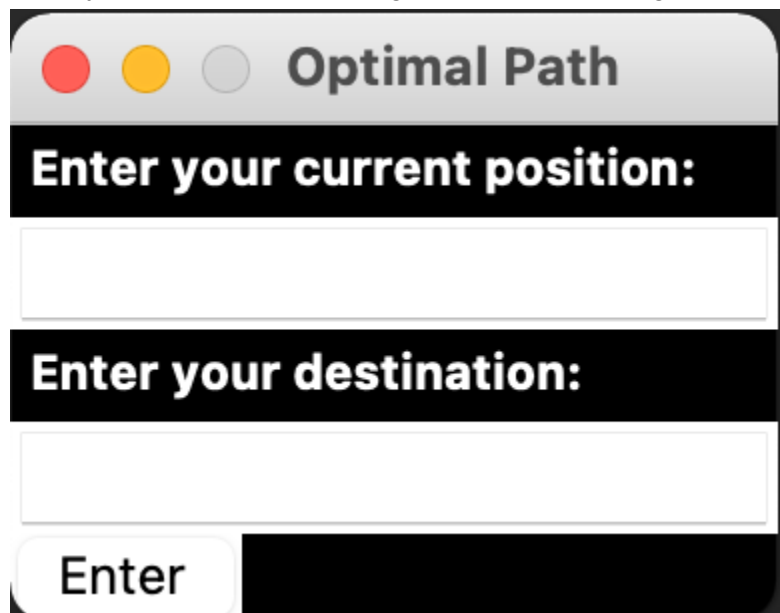
- Fed the system a list of valid and invalid work id numbers
  - The invalid work id numbers were given a “Invalid ID” and “NO ACCESS” message
  -



```

Run: plzwork x
/Users/beamannegash/PycharmProjects/Sprint1/venv/bin/python /Users/beamannegash/PycharmProjects/Sprint1/plzwork.py
Enter your id number: 1234567
Invalid User ID
NO ACCESS
Process finished with exit code 0
  
```

- If valid work ID number is inputted, GUI Opens up
- User is displayed GUI to enter starting location and ending location:



- 
- If User inputs a Valid Starting and Ending location, the optimal route will be displayed as well as the distance of that optimal route.

**Optimal Path**

**Enter your current position:**

a

**Enter your destination:**

h

Enter

**Optimal Path: ['a', 'c', 'd', 'e', 'h']**

**Distance Length: 13**

**Relaunch Program for New Starting/End Location**

- If the User inputs an invalid Starting or End Location, it will display the shortest path the user should take as well as the length of that most desired path

**Optimal Path**

**Enter your current position:**

a

**Enter your destination:**

z

Enter

**That Path is Not Reachable**

**Please Relaunch Program and**

**Enter a New Starting/End Location**

- To test the probability of a road accident leading to routes being unavailable we changed the probability from 1 in 103 to definite. The probability at 1 to 103 is too small to accurately so we guaranteed an accident in this portion of the project.

This allowed conclusions to be drawn based on the code's fault resistance displayed by the algorithm's effectiveness in unpredicted circumstances.

### **Evaluation:**

- We successfully implemented a GUI through Tkinter, a python library) which allows the user to input their starting location, and destination. This will allow users to input their locations through the use of an interface instead of through the IDE. In addition, we were able to simplify the code and make it much neater, allowing for easier comprehension of the code. Comments were added throughout the code to make better sense of what is happening in the GUI class and what elements impact the code. Our Dijkstra's Algorithm seems to work too, as by input a dictionary of routes, starting location, and end location, our program is able to output a route/distance.

### **Conclusion and Further Analysis:**

*In this segment we discussed potential improvements for a continuation of the OptimalDelivery project and assessed more of the real world factors and limitations of our project.*

Technically our approach would only minimize the total distances when we have a single truck that goes from a single pickup point and delivers to a single delivery point and certain other specific situations.

- I believe our solution is only necessarily optimal in the case that we have at least as many trucks as deliveries, Pick up points are far away from each other, delivery points are far away from each other, but pick up points and their corresponding delivery points are close to each other, and each truck starts off close to the pick up points (with as many trucks nearby the pickup point as the number of deliveries associated with that point).
- This is because our current setup only accounts for a single truck making a single delivery. If a truck were to perform multiple deliveries we then would need to account for what order the deliveries should be performed in. Additionally it would have to account for which trucks should perform multiple deliveries and how many.
- To complete the overall goal of minimizing collective distance, we would need to find the optimal partition for all the deliveries. This is the Vehicle Routing Problem with Deliveries and Pickups and is at least difficult to complete without brute force or use of heuristics which do not guarantee the optimal solution. There are  $T^D$  ways we can partition all the deliveries across each of the trucks. Dijkstra's algorithm with arrays is generally  $O(n^2)$  where  $n$  is the total number of nodes in our dictionary, and it would have to be implemented for each pickup point and delivery point in order to brute force every possible way to order the pickup's/deliveries. This makes it so that brute forcing it would likely take much longer than would be preferred, and may require more memory than we might have allotted for this project.



- If we were to continue we would need to decide if finding the optimal solution is worth the memory and time. Should we prioritize finding the optimal solution, or should we simply look for a good enough solution? Which may introduce the question of what is good enough, and may require us to reevaluate our requirements.

### Project Management:

Description	Impact	Probability	Severity	Mitigation
Team members may not submit code on time or are unresponsive	Deadlines for sprints will have to be pushed. Other team members may need to take up extra work to finish the project on time	Moderate	Serious	Check up periodically before sprint deadline on how everyone is progressing with their assignment
The estimated time for this sprint is underestimated	This would affect the quality of the sprint or it would change the schedule for production of the software	Moderate	Tolerable	Checking up periodically on the progress of each task will help the group adjust the deadlines faster if necessary
Group members do not have the requisite experience to work on their tasks	Team members would have to learn skills before starting to code their task	Moderate	Tolerable	Tasks should be started early so that team members can learn the skills required to complete the task