

Warehouse Guiding Application Developer Manual

Team Bug Killers

Professor Quoc Viet Dang

Team:

Eric Le

Wangyang Xu

Yiran Luo

Yuxin Shi

- Software title: Warehouse Navigation Application
 - Affiliation: University of California, Irvine
 - Date: 12/5/2021
 - Version: 1.5

Table of contents:

Version History	3
Revision History	3
1. Software Architecture Overview	4
Main Data Types and Structures	4
Major Software Components	4
Module Interfaces API	5
Program Control Flow (User Input, Program Output, Backend)	6
2. Installation	7
System Requirements	7
Setup & Configuration	7
Uninstalling	7
3. Documentation of Packages, Modules, Interfaces	8
Description of Data Structures	8
Description of Functions and Parameters	9
void countRate()	13
void evolution()	13
Description of Input/Output Format	14
4. Development Plan and Timeline	15
Partition of Tasks	15
Team Member Responsibilities	15
5. Back Matter	17
Copyright	17
Error Messages	17
6. Version Changes	20
Beta Release Revision History - Feedback and Fixes -	20
Algorithm Improvements	20

Version History

Version	Comment	Date (mm/dd/yyyy)
1.0	Developer Manual	10/12/2021
1.1	Alpha Release	10/20/2021
1.2	Beta Release	11/15/2021
1.3	Beta Release(Modified)	11/21/2021
1.4	Beta Release 2	11/29/2021
1.5	Final Release	12/5/2021

Revision History

Version	Fixed Issues	Date (mm/dd/yy yy)
1.0	Add basic functions and load the map	10/06/2021
1.1	Add error handling and add coordinates to the map	10/12/2021
1.2	Add Brute Force and Instructions	10/20/2021
1.3	Support picking up items in four directions	11/15/2021
1.4	Add Genetic Algorithm	11/29/2021
1.5	Fix some minor bugs and allow user choose output file or not	12/5/2021

1. Software Architecture Overview

Main Data Types and Structures

Data Types

- List
 - Implemented with the Java ArrayList object
- Graph
 - Implemented with 2-D array where all index points are connected vertices

Future Data Types

- Graph
 - Implemented with an Adjacency Matrix -2-Dimensional Array-

Major Software Components

Modules

- | | |
|---|-------------------------------------|
| <input type="checkbox"/> ProjectFolder | // Project folder |
| <input type="checkbox"/> warehouse | // Project java package |
| <input type="checkbox"/> Item.java | // Warehouse item abstraction |
| <input type="checkbox"/> Main.java | // Program driver |
| <input type="checkbox"/> Vertex.java | // Vertex in a graph |
| <input type="checkbox"/> Coordinate.java | // Location data for a vertex |
| <input type="checkbox"/> Item2ItemPath.java | // Shortest path between two nodes |
| <input type="checkbox"/> PrimaryController.java | // UI function handling |
| <input type="checkbox"/> Graph.java | // Graph implementation |
| <input type="checkbox"/> BruteForcePath | // Brute force implementation |
| <input type="checkbox"/> TSP_GA | // Generic Algorithm implementation |

Module Interfaces API

API of major functions:

Class	Method Signature and Description
BruteForcePath	void findShortestPath(int graph[][]) Calculates shortest path using the brute force approach; is used by primaryController to store shortest path into shortestPathCoordIndices list member.
PrimaryController	void printFullPathInstructions() Constructs the user instructions to traverse the entire path and pick up the various items; then prints the instructions to console.
PrimaryController	void setCurrentOrderGraph4N() Constructs the graph that is used by the shortest path algorithms. It constructs the graph based on the 4 adjacent nodes per item shelf, and only accounts for a shelf once if multiple items share a shelf.
Main	public static void main() Implements the text-based UI and controls user-program interaction
TSP_GA	public ArrayList<Integer> solve(string filename) Runs Generic Algorithm iteration for the given time. Returns the order of items with the shortest path so far and export it into filename.
TSP_GA	void tourToInstructions(ArrayList<Integer> tour) convert the pickup order (tour) into a list of instructions and mark the corresponding path in the warehouseMatrix
PrimaryController	Void findPathsBruteForce() Calls methods in BruteForcePath.java to get the shortest path and order of items. Also displays the resulting path in the UI.
PrimaryController	Void findPathsGenericAlgorithm() Calls methods in TSP_GA.java to get the shortest path and order of items. Also displays the resulting path in the UI.

Chart 1.1: API of major functions

Program Control Flow (User Input, Program Output, Backend)

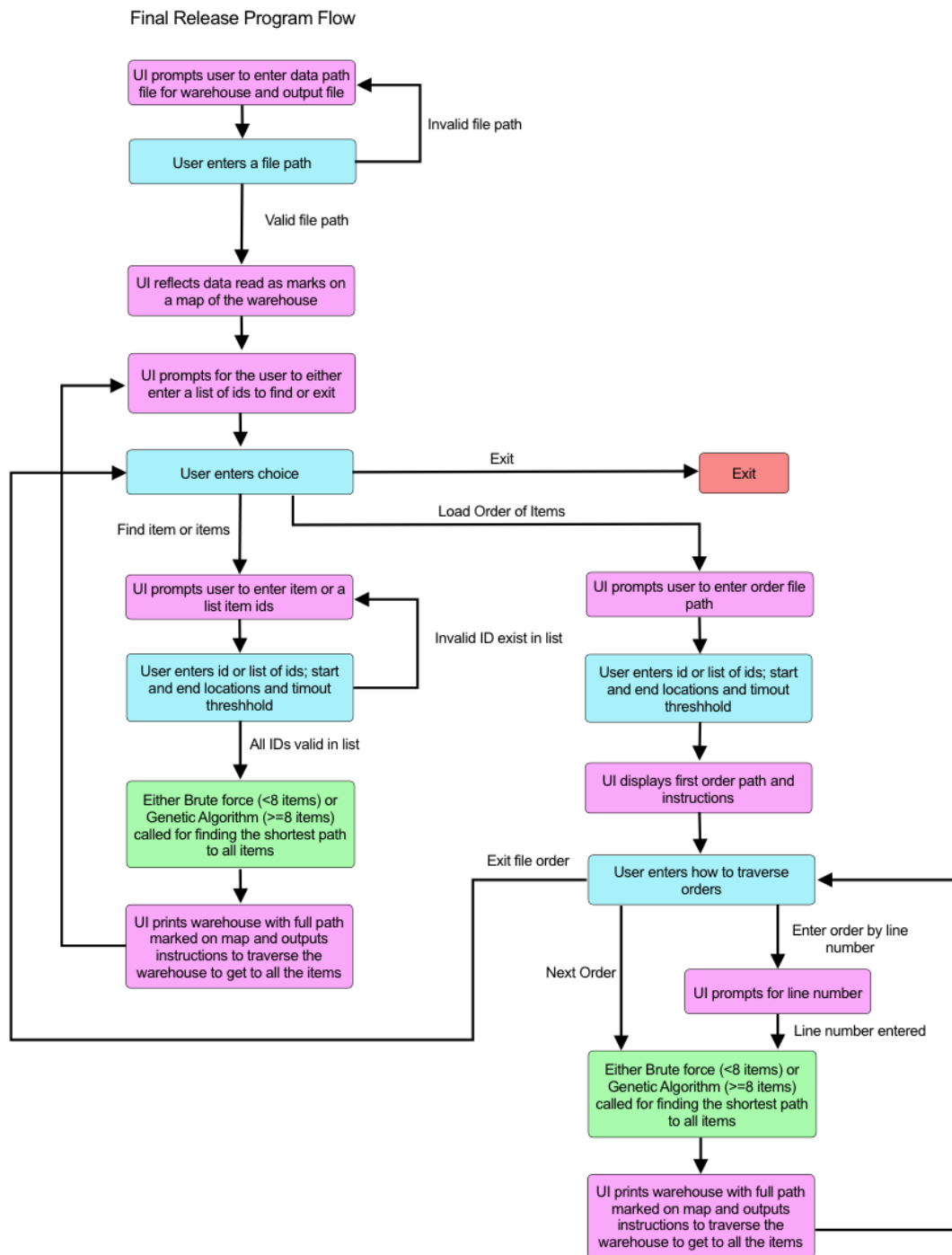


Figure 1.1: Final Release 2 Program Flow

2.Installation

System Requirements

MINIMUM:

OS: Windows 7 (32/64 bit)

Processor: Intel Core™ 2 Duo 2.0+ GHz or an equivalent AMD CPU

Memory: 4 GB RAM

Graphics: Not required

Storage: 50 MB available space

Java Runtime Environment (JRE): 17

RECOMMENDED:

OS: Windows 10 (64-bit)

Processor: Intel CPU Core™ i7 4790 4 GHz or an equivalent AMD CPU

Memory: 8 GB RAM

Graphics: Not required

Storage: 50 MB available space

Java Runtime Environment (JRE): 17

Setup & Configuration

1. Make sure you have properly installed Java Runtime Environment (JRE) 17 on your Windows computer. If you haven't, please follow the instructions on https://java.com/en/download/help/download_options.html.
2. Click [here](#) to download SmartWarehouseGuide.exe
3. Double click the executable file to run the application.
4. Enjoy using it!

Uninstalling

1. Simply delete SmartWarehouseGuide.exe in the directory where you downloaded it previously.

3.Documentation of Packages, Modules, Interfaces

Description of Data Structures

- **List:**
 - **ArrayList<Item> allItemsList:**
Holds all the Items loaded in from the warehouse data file and is used for setting up the graph representing the warehouse. It is a member of the PrimaryController.java module.
 - **ArrayList<Vertex> currentItem2ItemPath:**
Holds the shortest path between two vertices and is used to display the path to the user on the map and make the user traversal instructions. It is a member of the PrimaryController.java module.
 - **ArrayList<Item> currentOrderItems:**
Holds the Items that are added from the UI and is used to display the needed Items on the UI.
 - **ArrayList<Item> currentOrderItemsByShelf:**
Holds the Items that are added from the UI but is limited by unique shelves. Only the first item per shelf seen is stored in the list and used for shortest path calculations. Items that share the same shelf are added to the itemsOnSameShelfMap.
 - **ArrayList<Coordinate> currentOrderCoordinates4N:**
Contains the list of coordinates for all nodes that are to taken into consideration when generating the orderGraph. This includes unique shelves and 4 nodes per shelf.
 - **ArrayList<ArrayList<Integer>> currentLookupTable:**
Contains the list of the other 3 nodes that belong to an item per index. Each index represents the index of the node in currentOrderCoordinates4N.
 - **ArrayList<Integer> shortestPathCoordIndices:**
List that contains the coordinate indices of the item coordinate location in order of the shortest path. Each coordinate index can be formulated to get the index of the item in the currentOrderItemsByShelf list.
 - **ArrayList<Integer> shortestPathByID:**
List that contains the shortest path by ID, is used for display purposes.
 - **ArrayList<Vertex> path:**

List that contains the path between any two selected items, to determine the weights of the edges between two nodes.

- o **ArrayList<ArrayList<Item>> fileOrders:**
List that contains a list of item orders. These orders are read in from an input file.
- **Graph:**
 - o **Graph currentOrderGraph:**
Represents all the Items needed to be collected and the distances between them. It is a complete graph and is used for calculating the shortest path. Each item is abstracted as 4 separate nodes in the graph.
 - o **char[][] warehouseMatrix:**
Implemented as a 2-D array, this graph abstraction holds the printable character map as well as is used to calculate the shortest path between two vertices. Every index of the matrix is treated as a Vertex with edges to every adjacent vertex in cardinal directions. Note: this is not an adjacency matrix implementation.
- **HashMap:**
 - o **HashMap<Integer, ArrayList<Integer>> itemsOnSameShelfMap**
This map is used to obtain items on the same shelf. The key is the item index in currentOrderItems, with the value being a list of other item indices that share the same shelf.

Description of Functions and Parameters

- **ArrayList<Vertex> findBFSPath(char[][] graph, Coordinate src, Coordinate dest)**
 - o Takes the primaryController graph and two vertex Coordinates to compute the shortest path between the two vertices in the warehouse
 - o Implements Lee's Algorithm and returns the shortest path as a list of vertices, which later is later pointed to by the currentShortestPath list
- **ArrayList<Vertex> backtrackPath(Vertex q)**
 - o Helper function to iterate through a Vertex's ancestors to retrace the path determined by findBFSPath.
 - o Returns the path which findBFSPath returns
- **boolean isValid(int row, int col)**
 - o Helper to check if a Vertex's coordinates are within the bounds of a matrix

- **ArrayList<Vertex> findPathToItem**(Item start, Item finish)
 - Wrapper that calls findBFSPath in the PrimaryController module and returns the result
- **String findItemAndCallPath**(int id)
 - Check if an Item exists and if it does set currentShortestPath to the findBFSPath output.
 - Also calls and returns the result of makeUserInstruction
- **String makeUserInstruction**()
 - Parse the currentShortestPath into readable instructions for the user
- **void readAllItems**(String filePath) throws IOException
 - Reads the warehouse data file and uses it to populate the allItemsList
 - Throws file exceptions which are caught in main function
- **ArrayList<String> getFloatsFromString**(String raw)
 - Helper to extract the numbers from a String line of data
- **void setGraph**()
 - Set all the characters in the graph representing open space, shelves, items, the shortest path and the users locations
- **void printGraph**()
 - Print the graph to console
 - Prints the transpose and horizontal reflection of the matrix to print a nicer layout
- **boolean markItemInGraph**(int id)
 - Change the character in the graph for an Item based off id into a '\$' to represent an Item needed to be traversed to
- **void unmarkItemInGraph**(int id)
 - Revert a marked item's character back into an 'X'
- **Item getItemByID**(int id)
 - Item lookup in allItemsList based off Item id
- **boolean itemExist**(int id)
 - Check if Item exist in allItemsList
- **markPathOnGraph**()

- o Iterate through `currentShortestPath` and use the vertices Coordinates to mark the path on the map with 'P' characters
- **void unmarkPathOnGraph()**
 - o Iterate through `currentShortestPath` and use the vertices Coordinates to re-mark the path on the map with '.' characters
- **int isSharingShelf(Item item)**
 - o Helper to check if an item is sharing a shelf with an item already in `currentOrderItemsByShelf`
 - o Returns the index of the item it shares a shelf with
- **void setOrderItemsByShelves()**
 - o Construct the `currentOrderItemsByShelf` list as well as construct the `itemsOnSameShelfMap`.
- **void setLookUpTable ()**
 - o Construct the `currentLookupTable` list of list
- **void setCurrentOrderGraph4N()**
 - o Set the graph used for shortest path calculations
 - o Uses the four nodes of each unique item shelf when calculating the edge weights.
 - o Uses `findBFS` path to get the distance between each node
 - o Calls `setCurrentItems` by shelves before construction
- **void markFullPath()**
 - o Marks the full path traversal on the `warehouseMatrix`
- **String getShelfDirection(int pathX, int pathY, int itemX, int itemY)**
 - o Helper for `printFullPathInstructions` to return the relevant direction a shelf is to the user based on their warehouse location
- **void setShortestPathbyID ()**
 - o Dereferences the shortest path in terms `shortestPathCoordIndices` into the items by their ids and store into `shortestPathByID` list
- **void printFullPathInstructions ()**
 - o Construct and print the console the full direction list to traverse the entire shortest path to collect items
- **void findPathsBruteForce()**

- o Calls findShortestPath in BruteForcePath.java to get the shortest path based off of the currentOrderGraph4N.
 - o Before the shortest path calculations, calls setCurrentOrderGraph4N and setLookupTable
 - o Stores the shortest path in shortestPathCoordIndices and the cost in shortestPathCost
- **void printCurrentOrderGraph()**
 - o Prints the currentOrderGraph4N adjacency matrix to console for debugging purposes
- **void resetWarehouse()**
 - o Resets the warehouse matrix to original state after loading in the warehouse file. Also resets any list used in current order shortest path calculations.
- **void findPathGeneticAlgorithm()**
 - o Initiate the parameters in TSP_GA.java.
 - o Limit the running time within 60s, output the pick-up order of selected items and the overall distance
- **void init(int[] s, int[] e, ArrayList<Item> OrderItems, char[][] Matrix)**
 - o Initialize the main parameters in TSP_GA.java
- **void tourToInstructions(ArrayList<Integer> tour)**
 - o Call setInstructions() for each items to convert the pickup order into a list of instructions
 - o Mark the pickup items as '\$'
 - o Call setMatrix() to mark the path as 'P'
- **void setMatrix(char[][] warehouseMatrix, ArrayList<Vertex> path)**
 - o Make corresponding path as 'P' in warehouseMatrix
- **void setInstructions(ArrayList<Vertex> path)**
 - o Calculate and store the distance and directions in instructions according to the path
- **String getDirection(int x1, int y1, int x2, int y2)**
 - o Get and return the direction of adjacent node (x1,y1) and node(x2,y2)
- **String pickupDirection(Coordinate c1, Coordinate c2)**
 - o Get and return the pickup direction between worker's location and needed item

- **boolean checkNeighbors(Coordinate c1, Coordinate c2)**
 - check if the dest node is next to the source node. if true, the distance should be 0, no need BFS; if false, need BFS
- **ArrayList<Vertex> setBFSPath(char[][] warehouseMatrix, Coordinate c1, Coordinate c2)**
 - Find the path from c1 to c2 in warehouseMatrix
- **ArrayList<Integer> solve(int timeOut)**
 - Limit the running time within 60s, output the pick-up order of selected items and the overall distance
- **void initGroup()**
 - Randomly initialize the order of items and store them in oldPopulation
- **int evaluate(int[] chromosome)**
 - chromosome[] is a list of all items, exclude the start/end
- **void countRate()**
 - Calculate the cumulative probability of each individual in the population as part of the gambling wheel selection strategy
- **void evolution()**
 - save the best chromosomes from crossover
 - Two steps: select and then crossover/mutation
- **void selectBestGh()**
 - The individuals with the highest fitness in a generation population were selected and copied directly into the offspring
- **void copyGh(int k, int kk)**
 - Copy chromosome, k represents the position of the new chromosome in the population, and kk represents the position of the old chromosome in the population
- **void select()**
 - Wheel selection strategy, save the best chromosomes from crossover
- **void OXCross1(int k1, int k2)**
 - Crossover
 - Switch some items' order between k1 and k2 and store them into the newPolulation

- **void OnCVariation(int k)**
 - Mutation
 - Randomly switch items from one individual in several times
- **public static void main(String[] args)**
 - Driver function for the program
 - Controls the flow of the application utilizing the functions within the PrimaryController class
- **static void setTimeoutTime(PrimaryController primaryController, Scanner scanner)**
 - Helper for main to set dynamic timeouts
- **static void setStartAndEndLocations(PrimaryController primaryController, Scanner scanner, int[] start, int[] end)**
 - Helper function for main to set dynamic start and end locations

Description of Input/Output Format

- **Input Format:**
The user communicates with the application by typing in responses to the command line. The text is processed by the program and stored in the appropriate data structures in order to execute the applications functionality. The user enters data including:
 - file paths for the warehouse shelf and item locations, multiple orders and traversal instruction file output
 - integers for deciding what program actions they wish to take as well as item ids
- **Output Format:**
The UI communicates to the user through text-based prompts displayed in the users console. These prompts list and ask the user what program actions to take throughout application use. The UI outputs the following types of information:
 - An ascii map representing a top-down view of the warehouse
 - This map displays to the user shelf locations, open space, items to pick up, and the corresponding paths to collect orders
 - Text prompting for integer input for program actions and item input
 - Text prompting for file path input for reading and writing data to and from files

4. Development Plan and Timeline

Partition of Tasks

- Yuxin Shi: Responsible for the Graph module, updates in developer manual and timeline.
- Eric Le: Responsible for the structure of the system and the improvement in UI module
- Yiran Luo: Responsible for the ShortestPath module and updates in user manual
- Wangyang Xu: Responsible for implement Genetic Algorithm to find the shortest path
- Everyone's task: All of us will be responsible for the testing and improvement of the whole system. And there will be role exchanges between each other to make sure that everyone's ideas are considered and have a whole grasp of the system
- The timeline of the project can be divided into four stages: documentation stage(this stage has already been done by now), implementation stage(currently undergoing this stage and would finish in one or two weeks), improvement stage(estimated to be around one or two weeks); test stage(the rest of the time, at least one week)

Team Member Responsibilities

- Yuxin Shi(Manager):
 - Make sure the team starts quickly and remains focused during the activity.
 - Take care of time management.
 - Make sure all voices in the team are heard.
- Eric Le(Presenter):
 - Communicates team questions and clarifications with the teacher or other teams.
 - Ensures all team members have had a chance to respond before asking outside sources.
 - Ensures that everyone in the team agrees on what to ask if an outside source is needed.
 - Presents Conclusions of the team to the class, as requested.
- Yiran Luo(Reflector):
 - Guides consensus-building process; team must agree on responses to questions.

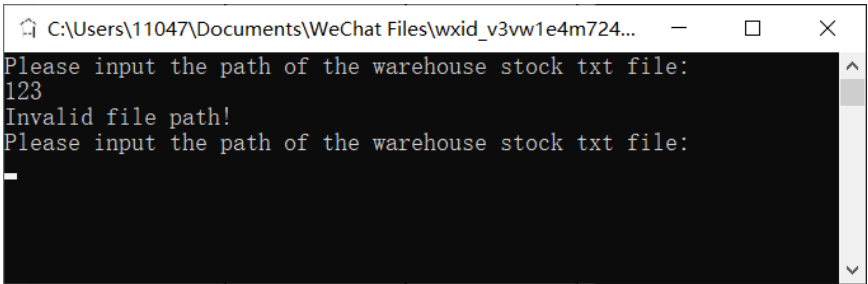
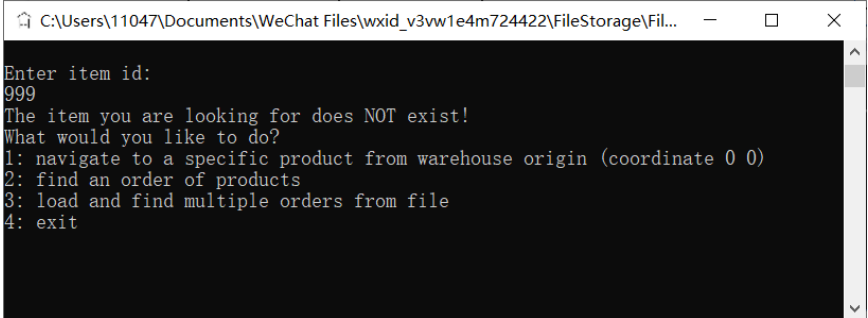
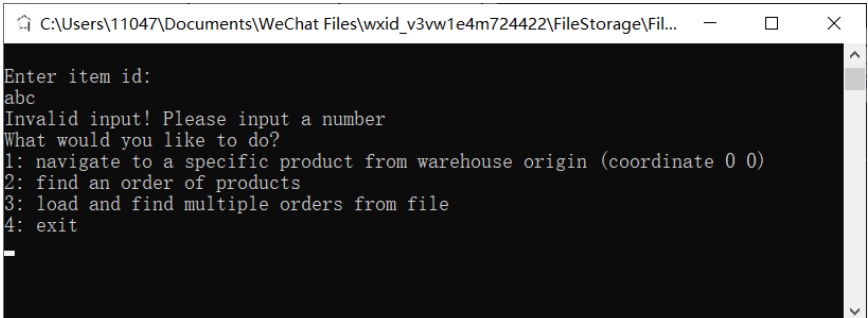
- Observes team dynamics and behavior with respect to the learning process.
 - Reports to the team periodically during the activity on how the team performs.
 - Be ready to report to the entire class about how well the team is operating.
- Wangyang Xu(recorder):
 - Records the names and roles of the group members at the beginning of each activity.
 - Records the important aspects of group discussions, observations, insights, etc.
 - The recorder's report is a log of the important concepts that the group has learned.

5.Back Matter

Copyright

© 2021, Bug killers. All rights reserved.

Error Messages

Error Input	Output	Screenshot(All based on Final Release)
Invalid file path of warehouse map	Invalid file path!	 A terminal window titled 'C:\Users\11047\Documents\WeChat Files\wxid_v3vw1e4m724...' displays the following text: 'Please input the path of the warehouse stock txt file:', '123', 'Invalid file path!', and 'Please input the path of the warehouse stock txt file:'.
Invalid item ID (not exist)	The item you are looking for does NOT exist!	 A terminal window titled 'C:\Users\11047\Documents\WeChat Files\wxid_v3vw1e4m724422\FileStorage\Fil...' displays the following text: 'Enter item id:', '999', 'The item you are looking for does NOT exist!', 'What would you like to do?', '1: navigate to a specific product from warehouse origin (coordinate 0 0)', '2: find an order of products', '3: load and find multiple orders from file', and '4: exit'.
Invalid item ID (not a number)	Invalid input! Please input a number.	 A terminal window titled 'C:\Users\11047\Documents\WeChat Files\wxid_v3vw1e4m724422\FileStorage\Fil...' displays the following text: 'Enter item id:', 'abc', 'Invalid input! Please input a number', 'What would you like to do?', '1: navigate to a specific product from warehouse origin (coordinate 0 0)', '2: find an order of products', '3: load and find multiple orders from file', and '4: exit'.

Invalid option number (any other input except '1' or '2' or '3')	Invalid input! Please input '1' or '2' or '3'.	<pre> C:\Users\11047\Documents\WeChat Files\wxid_v3vw1e4m724422\FileStorage\File\2021-... What would you like to do? 1: navigate to a specific product from warehouse origin (coordinate 0 0) 2: find an order of products 3: load and find multiple orders from file 4: exit 5 Invalid input! Please input '1' or '2' or '3' or '4'. What would you like to do? 1: navigate to a specific product from warehouse origin (coordinate 0 0) 2: find an order of products 3: load and find multiple orders from file 4: exit </pre>
Invalid file path of order file	Invalid file path!	<pre> C:\Users\11047\Documents\WeChat Files\wxid_v3vw1e4m724422\FileStorage\File\2021-... Please input the path of the order file: abc Invalid file path! Please input the path of the order file: </pre>
Wrong start or end location input	You can't start or end in one of item shelves	<pre> C:\Users\11047\Documents\WeChat Files\wxid_v3vw1e4m724422\FileStorage\File\2021-... Please enter the START point warehouse coordinates separated by a single space: 30 0 You can't start in one of item shelves Please enter the START point warehouse coordinates separated by a single space: </pre>
Start or end location either be negative or too large	please input the correct START location(only Positive Integer Number) & please input the START location within warehouse coordinate	<pre> C:\Users\11047\Documents\WeChat Files\wxid_v3vw1e4m724422\FileStorage\File\2021-... Please enter the START point warehouse coordinates separated by a single space: -1 -1 Please input a valid START location(only Positive Integer Numbers) Please enter the START point warehouse coordinates separated by a single space: 999 999 Please input the START location within warehouse coordinates of X:0-39 Y:0-24 Please enter the START point warehouse coordinates separated by a single space: </pre>

	es of X:0-39 Y: 0-24	
Wrong format of time limit	please input the correct time limit (only positive number)	<pre> Please enter the START point location seperated by a blank. 0 0 Please enter the END point location seperated by a blank. 0 0 Your start and end points are (0,0) and (0,0) Please enter the time limit to find the path in seconds 1.1.1 please input the correct time limit (only positive number) Please enter the time limit to find the path in seconds -123 please input the correct time limit (only positive number) Please enter the time limit to find the path in seconds 1.12 </pre>
Wrong input of items' size	please input the correct size (only positive integer number)	<pre> Type the size of the order: -1 please input the correct size (only positive integer number) Type the size of the order: 1.2 please input the correct size (only positive integer number) Type the size of the order: 5 please type id of products separated by blanks: (we will only accept the first 5 products) _ </pre>
Input items that are not placed in warehouse	please input the items that are placed in the warehous e	<pre> Type the size of the order: 1 please type id of products separated by blanks: (we will only accept the first 1 products) 32131231 pleaseinput the items that are placed in the warehouse please type id of products separated by blanks: (we will only accept the first 1 products) </pre>

6.Version Changes

Beta Release Revision History - Feedback and Fixes -

- **Brute Force:**

Problem: The brute force needed to include a timeout and return a path if a timeout occurred.

Solution: The brute force algorithm now includes a timeout that will return the current shortest path it has calculated up until the timeout threshold.

- **Genetic Algorithm:**

Problem: There needs to be details documenting the genetic algorithms initial population, max generations, etc in our testing documentation.

Solution: We have now included such details in our testing documentation

Algorithm Improvements

- **Brute Force:**

- Brute force has been altered to include a timeout which returns the algorithm prematurely in its recursion if the time threshold is met. The algorithm now will save the current shortest path up till that time.
- Brute force now supports dynamic start and end locations. Previously, we added back the starting node to the end of a path to create a hamiltonian cycle, now we add the ending vertex instead at the end of every path that will cover all the vertices.

- **Genetic Algorithm :**

- Genetic Algorithm also supports dynamic start and end locations.
- Genetic Algorithm reduces the running time by skipping the BFSsearch when adjacent pick-up items are located on the same shelf.