

MoneyGoWhere

[View on GitHub](#)

MoneyGoWhere User Guide



Hello Hawker!

Thank you for your interest in MoneyGoWhere!

You have just made the first step in making even more money from your business, nice!

This is the user guide for MoneyGoWhere. It is designed to help you, a hawker at NUS, learn how to use our system easily .

The purpose of this guide is to show easy-to-understand instructions on how to use the program .

There may be new words in the guide that you have never heard of or seen before, but do not worry! We have made sure all the harder words are linked to our glossary at the bottom of the guide. All you have to do is click on the difficult word, and your computer will lead you to what the word means. Easy!

What is MoneyGoWhere?

MoneyGoWhere is a computer program that helps you keep track of your sales.

It works like the *cashier machine* that you have been using, but is even better! With MoneyGoWhere, you can see how much money you make each day and what items you sell the most .

Now, you can make smarter business decisions, save time, and money. *Steady!*

"I have problems now, that's why I am reading the guide!"

"But I cannot find what I am looking for, how now?"

We understand that using a new system can be confusing and frustrating. That is why we used simple words and show examples in this guide.

We understand that you may be referencing this guide when business is booming - you can already barely cope, but you are unsure of how to use a feature .

Don't worry! The guide is written in such a way that you can find what you are looking for easily. Click on [hyperlinks](#) listed in the Table of Contents to jump to the particular section that you want to know more about.

Additionally, all words listed in our [glossary](#) will *look like this*, so you are able to easily identify between the various hyperlinks.

If you cannot find what you are looking for, you may call us at 9123 4567 so that we can help you as soon as possible.


How to use the Guide


This guide will bring you through:

1. How to use a *Command Line Interface (CLI)*
2. How to set up MoneyGoWhere
3. How to use our various *commands* to track items, add orders, and generate statistic reports

We will also provide some information on how your data is saved.

Before we begin, take note of the following information: This guide uses three different colored blocks and associated icons to indicate different things.

 This is a story block! Story blocks provide an example on how to use the many features of MoneyGoWhere.

 These blocks contain information that you should take note of, or additional details that you might be interested in.


 These are warning blocks. They are used to show you errors that you may encounter.

Table of Contents

- [Setting up MoneyGoWhere](#)
- [MoneyGoWhere Commands](#)
- [Style Features](#)
- [Summary of Features](#)
 - [Help](#)
 - [Item Features](#)
 - [Order Features](#)
 - [Statistics and Report](#)
- [Frequently Asked Questions](#)
- [Glossary](#)

Setting up MoneyGoWhere

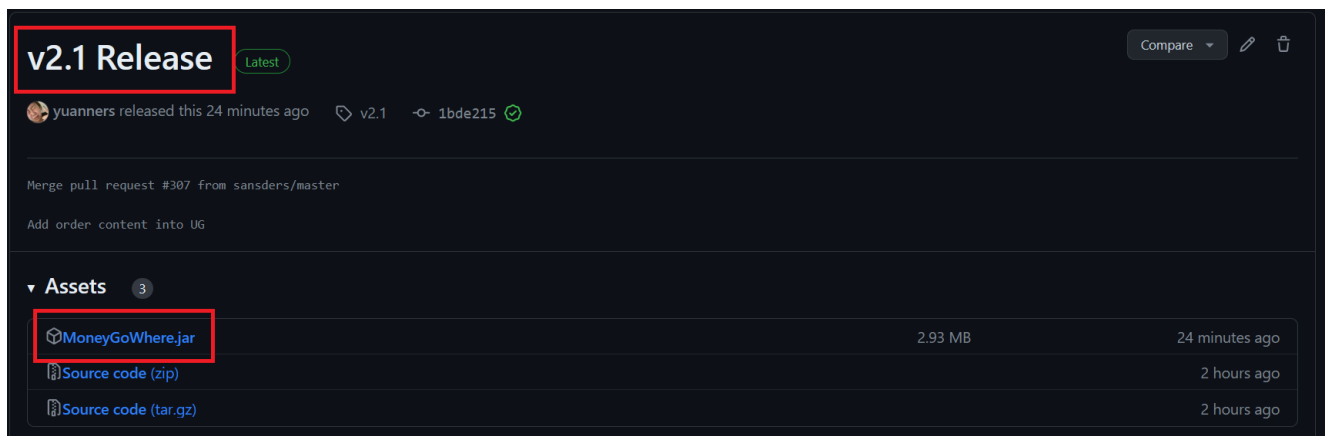
1. Before installing and using MoneyGoWhere, do note that Java 11 is required. If you do not already have it installed, you can do so [here](#).
 - For Windows users, download the x64 Installer. Ensure you are on the “Windows” page and download the correct version, in the red box.

Linux macOS Solaris Windows		
Product/file description	File size	Download
x64 Installer	141.14 MB	jdk-11.0.18_windows-x64_bin.exe
x64 Compressed Archive	158.85 MB	jdk-11.0.18_windows-x64_bin.zip

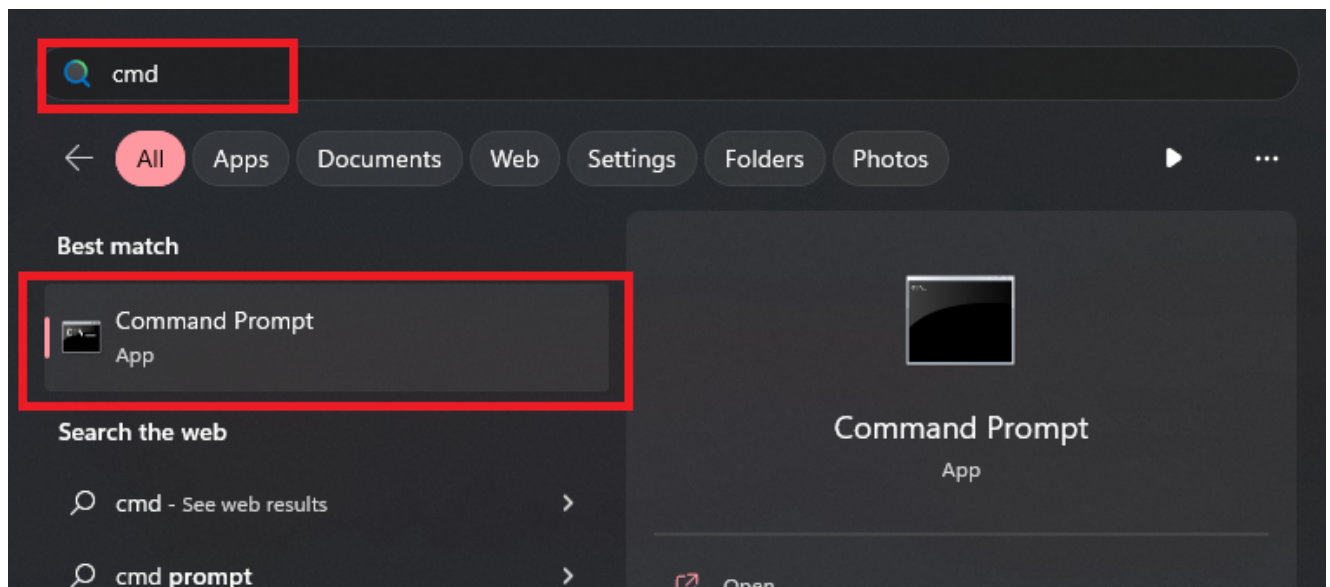
- For Mac users, download the ARM 64 DMG Installer. Ensure you are on the “macOS” page and download the correct version, in the red box.

Linux macOS Solaris Windows		
Product/file description	File size	Download
Arm 64 Compressed Archive	153.75 MB	jdk-11.0.18_macos-aarch64_bin.tar.gz
Arm 64 DMG Installer	153.24 MB	jdk-11.0.18_macos-aarch64_bin.dmg
x64 Compressed Archive	155.89 MB	jdk-11.0.18_macos-x64_bin.tar.gz
x64 DMG Installer	155.38 MB	jdk-11.0.18_macos-x64_bin.dmg

2. After installing Java 11, [please download the latest release of the MoneyGoWhere.jar file](#). Click on [MoneyGoWhere.jar](#) to automatically download the file. Ensure that it is the latest version (the one with the biggest number).

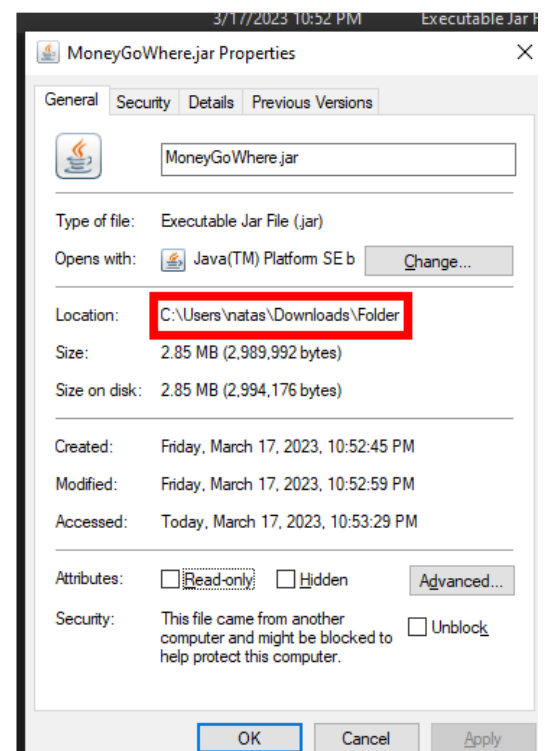
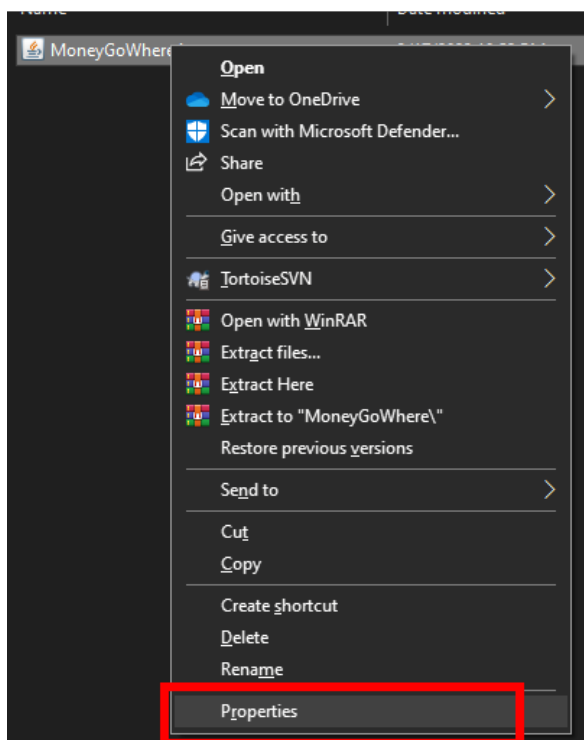


3. Find [MoneyGoWhere](#) in your Downloads folder.
4. Open [cmd](#) and navigate to the folder where the [MoneyGoWhere.jar](#) file is. If you are unsure of how to do so, follow steps 5 to 7. Otherwise, you may skip ahead to step 8.
5. Open a command [terminal](#) by entering [cmd](#) in your start menu and select the first [option](#) that appears.

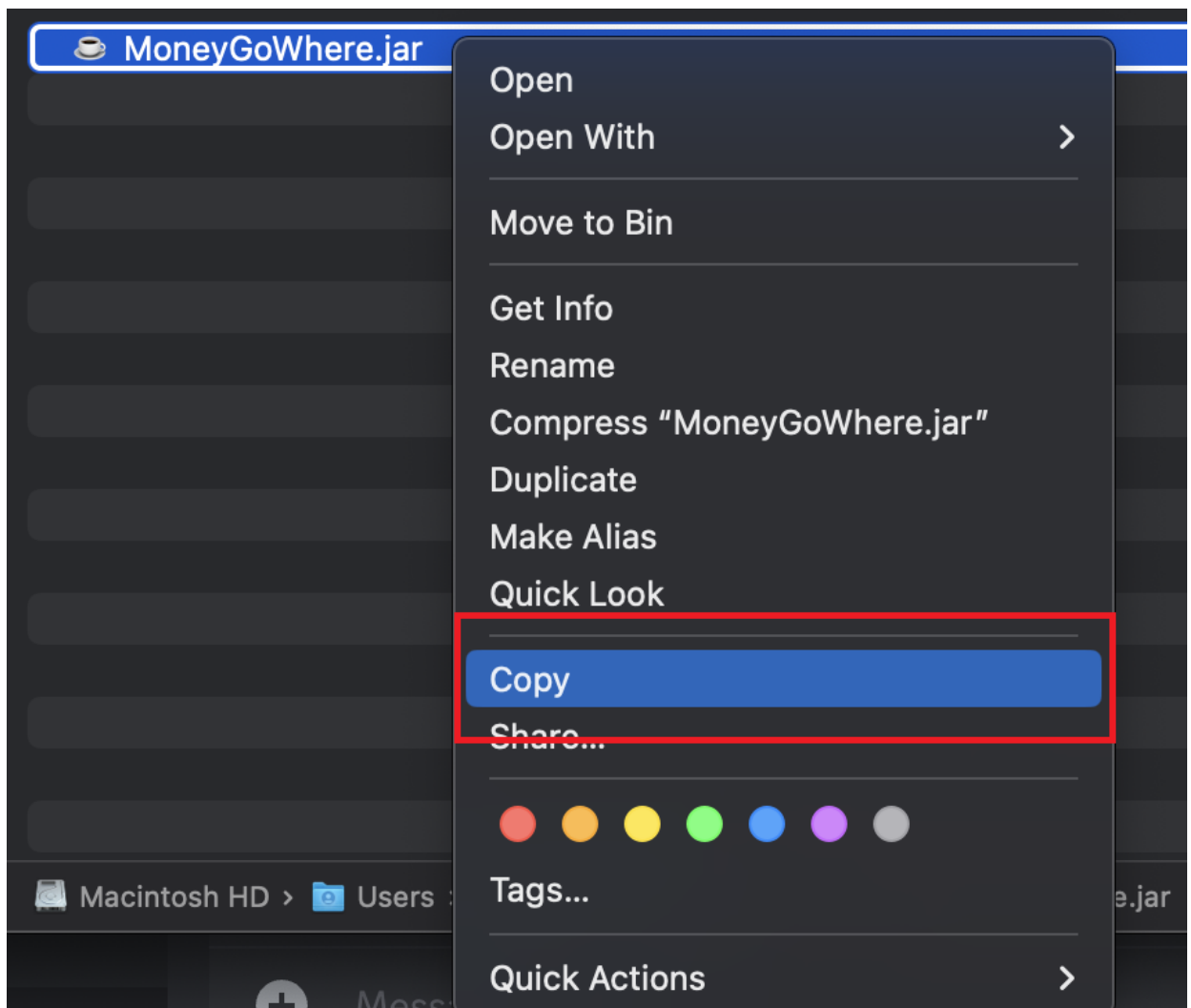


6. Get the file path of `MoneyGoWhere.jar` in your computer. To easily do so, you can do the following:

- For Windows users, right-click the file and select the “Properties” *option*. Then, copy the text under “Location”.



- For Mac users, right-click the file and select “Copy”.



7. In the CLI, navigate to the location of the folder by typing `cd` , type or paste the file path that you found from the previous step by pressing CTRL and V at the same time on your keyboard, then press enter.

- For example, if the file path that you have is `C:\Users\natas\Downloads\Folder` , then you should enter `cd C:\Users\natas\Downloads\Folder` into the CLI.

8. Use the *command* `java -jar MoneyGoWhere.jar` to launch MoneyGoWhere. If done correctly, you will see this on your first start up:



9. When you are done using the app, enter `exit` to shut the application down or simply close the window by clicking on the `x` on the top right hand corner of the CLI.

[Back to table of contents](#)

MoneyGoWhere Commands

📖 Meet John! He will be a character that is constantly referenced throughout the next few sections to better explain how the *commands* work. John is an aspiring hawker, and plans on running a new stall at The Deck in NUS.

💡 For new users, it is recommended to read the sections labelled as For New Users. These sections will bring you through using *command* in a guided, step-by-step manner. As you become more familiar with MoneyGoWhere and want to speed up using the application, we recommend reading sections labelled as For Experienced Users. In these sections, we provide examples on how to use each feature with just a single line of input.

💡 Even though the New User and Experienced User *command* are entered into the application slightly differently, they both work exactly the same. In other words, there is nothing that the Experienced User *command* can do that the New User *command* cannot.

[Back to table of contents](#)

Style Features

For the following sections, all examples on how to run *commands* are shown in images. There are four things to take note of:

1. Standard text printed by the application is written in white.
2. User input is signified and is written in *blue*.
3. Success messages and examples of expected output is written in *green*.
4. Error messages are written in *red*.

When you use the application, the text on your screen will be in white.

We have chosen to color-code our examples to make it easier for you to follow along and tell apart different parts of the examples.

Here is an example of what you can expect to see in this Guide.

This is an example of standard text printed by the MoneyGoWhere application.
This is where you enter your own input!
If you follow our examples, you should expect to see the same text on your screen.
However, if you enter something wrong, you should see text of this color.

For those reading the New User sections, do also note the following:

1. At any point, if you wish to exit from the *command*, you can do so with `/cancel`

For those reading the Experienced User sections, do also note the following:

1. Any words surrounded by `<>`, such as `<price>` are for you to fill in.
2. *Options* that are surrounded by `{ }`, such as `{-n "<name>"}` are optional.
3. All values have to be accompanied by *options* (begins with `-`, such as `-n` or `--price`).
Commands such as `/deleteitem delete -i 10` will be an invalid *command*.

[Back to table of contents](#)

Summary of Features

There are 10 features built into MoneyGoWhere, as described in the table below. More details are provided in their individual sections.

Name	Description
Help	Displays information about various <i>commands</i>
Item Features	
Add an Item	Adds an item to the menu
Delete an Item	Deletes an item from the menu
List All Items	Lists items in the menu
Update an Item	Updates an item in the menu
Find an Item	Finds an item in the menu, based on its name
Order Features	
Add an Order	Adds an order, with the <i>index</i> and quantity of each item
List All Orders	Lists all orders
Refund an Order	Refunds an order based on the unique order ID

Name	Description
Statistics	
Statistics and Reports	Generates a report based on various options

[Back to table of contents](#)

Help

This [command](#) displays information about the other [commands](#) available in MoneyGoWhere.

📖 John has just downloaded MoneyGoWhere. Unlike you, he does not know that MoneyGoWhere has a user guide and wants to find out more about the various [commands](#) offered by MoneyGoWhere. To find out more about the [commands](#) offered, he can use the `help commands`. Currently, all he knows is that he can do `help`, or `/help`.

💡 The help [command](#) does not accept any other input. Entering anything other than `help` or `/help` will cause an error.

For New Users

📖 As someone who is unfamiliar with the application, John wants to be guided step-by-step to use MoneyGoWhere. To view the [commands](#) available to him, he can use the `help` command.

Please enter a command:

help

There are 10 commands you can use as a new user in MoneyGoWhere.

For more details, please refer to the User Guide.

0. help
1. additem
2. listitem
3. deleteitem
4. updateitem
5. finditem
6. addorder
7. listorder
8. refundorder
9. /report {-r <type>} {-s <type>} {-y <year>} {-f <start-date> -t <end-date>}

To see commands for experienced users, use ``/help``.

For Experienced Users

📖 In the future, John develops a love for typing and now prefers to use just one line to complete a *command*. He can use the `/help command` to view how to use these single-line *commands*.

Please enter a command:

`/help`

There are 10 commands you can use as an experienced user in MoneyGoWhere. For more details, please refer to the User Guide.

0. `/help`
1. `/additem -n "<name>" -p <price>`
2. `/listitem`
3. `/deleteitem -i <index>`
4. `/updateitem -i <index> {-n <name>} {-p <price>}`
5. `/finditem -n "<name>"`
- 6a. `/addorder -i <index> -q <quantity>`
- 6b. `/addorder -I [<index>:<quantity>, ...]`
7. `/listorder`
8. `/refundorder -i <order ID>`
9. `/report {-r <type>} {-s <type>} {-y <year>} {-f <start-date> -t <end-date>}`

To see commands for new users, use ``help``.

Error Messages

1. Adding words or letters after the `command`

MoneyGoWhere does not allow for additional letters or words after the `help` `command` word.

Please enter a command:

`help aaa`

Error: This command is not recognised.

Solution: Only enter the `command` word, `help` or `/help`.

[Back to table of contents](#)

Item Features

Before you start taking orders, you will need to set up your menu. This must be done at least once.

Depending on the size of your menu, you may need to spend quite a bit of time to add all of your menu items into MoneyGoWhere. It is important that you do this before opening shop for the day, to ensure that your business can be run without any disruptions.


You may continue to add new items to the menu at any point of time.


Item Table of Contents

- [Add an Item](#)
- [List all Items](#)
- [Delete an Item](#)
- [Update an Item](#)
- [Find an Item](#)

Add an Item

If you need to add an item to the menu, you can use this [command](#). To add an item, you will need the item's name and price.

 John wants to run a Chicken Rice store. Currently, he only has two things to sell and has already set the cost: White Chicken Rice (\$4.50) and Egg (\$0.80). He will need to use the `additem` [command](#) to add these two items.

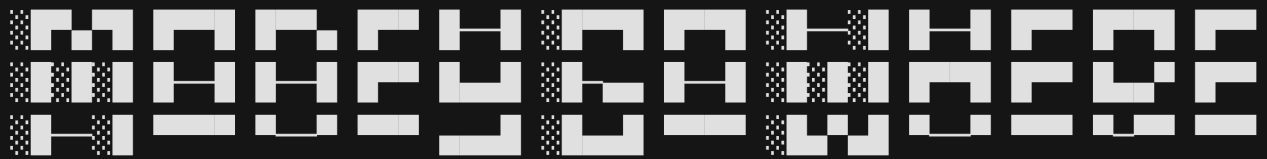
 There are restrictions for the name and price of items, as shown below:

Option	Description	Restrictions
<code>Item Name</code>	The name of the item.	Any input that is not empty and has less than 25 characters. It cannot only have numbers.
<code>Price</code>	The price of the item.	A number, with up to 2 decimal places . For example, <code>1</code> , <code>10.23</code> and <code>5.20</code> are valid inputs. However, <code>1.234</code> is not a valid input. Avoid entering a high number, ie. above <code>100,000</code> as it may cause unexpected results.

For New Users

This example will show you how to add an item step-by-step.

1. John starts the application for the first time. He sees the greeting and knows that he set up the application correctly. Now, he wants to add White Chicken Rice, which costs \$4.50 to the menu.



Welcome to MoneyGoWhere!

Please enter a command:

`additem`

Please enter the item's name or use "/cancel" to abort:

`White Chicken Rice`

Please enter the item's price or use "/cancel" to abort:

`4.50`

Item added successfully.

2. He sees that he is successful and feels happy that the application is easy to use. Now, he wants to add Egg, which costs \$0.80 to the menu.

Please enter a command:

`additem`

Please enter the item's name or use "/cancel" to abort:

`Egg`

Please enter the item's price or use "/cancel" to abort:

`0.8`

Item added successfully.

3. Lastly, John uses `listitem` to check that he has added the items correctly.

Please enter a command:

listitem

Index	Name	Price
0	White Chicken Rice	4.50
1	Egg	0.80

All items in the menu have been listed!

For Experienced Users

💡 To use the *command* in this manner, remember to add a `/` before the *command*, such as `/additem`.

Command Format

These are the different *command* formats that are accepted when adding a new item into the menu.

```
/additem --name "<item name>" --price <price>
```

```
/additem -n "<item name>" -p <price>
```

You may also interchange the location of the *options* in the *commands*.

For example, the following *command* is also valid.

```
/additem -p <price> -n "<item name>"
```

This example below will show you how to add an item in a single *command*.

1. John starts the application for the first time. He sees the greeting and knows that he set up the application correctly. Now, he wants to add White Chicken Rice, which costs

\$4.50 to the menu.



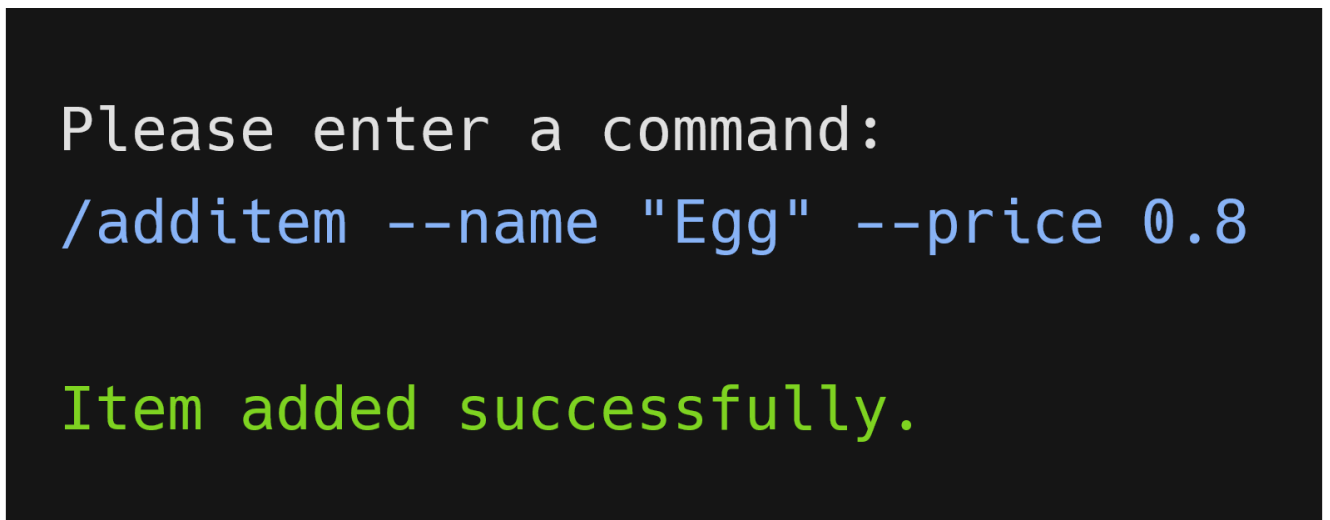
```

Welcome to MoneyGoWhere!

Please enter a command:
/additem -n "White Chicken Rice" -p 4.50

Item added successfully.
```

2. He sees that he is successful and feels happy that the application is easy to use. Now, he wants to add Egg, which costs \$0.80 to the menu.



```

Please enter a command:
/additem --name "Egg" --price 0.8

Item added successfully.
```

3. Lastly, John uses `listitem` to check that he has added the items correctly.

Please enter a command:

```
/listitem
```

Index	Name	Price
0	White Chicken Rice	4.50
1	Egg	0.80

All items in the menu have been listed!

Error Messages

! The next few examples are invalid inputs, designed to show off some error messages we have in place. This is not the full list of error messages. Additionally, these error messages will be shown for both New User and Experienced User *commands* if the mistake is made.

1. Name already exists

MoneyGoWhere does not allow for multiple items with the same name. If you try to put in an item with the same name, you will get the following error.

Please enter a command:

```
/additem -n "Vegetable" -p 2.8
```

Item added successfully.

Please enter a command:

```
/additem -n "Vegetable" -p 2.8
```

Error: Name already exists. Please choose a different name.

Solution: Use a different name.

2. Price has more than two *decimal places*

MoneyGoWhere does not allow you to enter a number with more than two *decimal places*. This is because there are no such denominations in real life.

```
Please enter a command:
```

```
additem
```

```
Please enter the item's name or use "/cancel" to abort:
```

```
Fish
```

```
Please enter the item's price or use "/cancel" to abort:
```

```
2.57325
```

```
Error: Price can have at most 2 decimal points.
```

```
Please enter the item's price or use "/cancel" to abort:
```

```
2.50
```

```
Item added successfully.
```

Solution: Limit the price to 2 decimal points.

[Back to table of contents](#)

List all Items

To see all the items you have entered in your menu, use this *command*!

📖 John has spent the past 10 minutes entering all the items he plans to sell. Happy with his work, he wants to look at all the items in the menu to make sure that everything has been entered correctly. To view the items in his menu, he will need the `listitem` *command*.

💡 This *command* uses only one word. Adding anything else after the *command* will cause MoneyGoWhere to not recognize the *command*.

1. John has added 10 items to the menu, and he wants to check that they are all correct. He uses the `command` `listitem` or `/listitem` to do so.

Please enter a command:

`listitem`

Index	Name	Price
-----	-----	-----
0	White Chicken Rice	4.50
1	Egg	0.80
2	Roast Chicken Rice	4.50
3	Chicken Hor Fun	5.00
4	Chicken Porridge	3.80
5	Dumpling Soup	5.00
6	Vegebatle	4.20
7	Chicken Liver	1.50
8	Fried Rice	5.30
9	Curry Chicken Rice	55.00

All items in the menu have been listed!

💡 Notice typos? Don't worry, it's intentional! These typos will be used in later sections.

Error Messages

❗ The next example is an invalid input, designed to show off the error messages we have in place. This is not the full list of error messages. Additionally, these error messages will be shown for both New User and Experienced User commands if the mistake is made.

1. Adding words or letters after the `command`.

Adding anything after `listitem` or `/listitem` will cause the *command* to be invalid.

Please enter a command:

`listitem a`

Error: This command is not recognised.

[Back to table of contents](#)

Delete an Item

If you decide to stop selling a particular item, you can remove it from the menu with this *command*.

💡 By using this *command*, you may potentially change the other *index* numbers of your other menu items. This may or may not affect the way you add items to orders. It is recommended that you use `listitem` to confirm if there are any changes.

To view more about listitem, [click here](#).

📖 As per the previous example, John currently plans to sell Fried Rice. However, after asking his friends, he realized that no one wants to buy Fried Rice from a Chicken Rice stall. The menu currently looks like this:

Index	Name	Price
0	White Chicken Rice	4.50
1	Egg	0.80
2	Roast Chicken Rice	4.50
3	Chicken Hor Fun	5.00
4	Chicken Porridge	3.80
5	Dumpling Soup	5.00
6	Vegebatle	4.20
7	Chicken Liver	1.50
8	Fried Rice	5.30
9	Curry Chicken Rice	55.00

So, he has decided to delete Fried Rice, which has an *index* of 8 from his menu using the `deleteitem` *command*.

💡 There are restrictions for the index, as shown below:

Option	Description	Restrictions
Index	The position of the item in the menu.	The entered number must be a valid <i>index</i> number from <code>listitem</code> . The entered number cannot be a negative number.

For New Users

- 1. John wants to delete Fried Rice, which has *index* number 8.

Please enter a command:

```
deleteitem
```

Please enter the item's index or use "/cancel" to abort:

```
8
```

```
Item deleted successfully.
```

For Experienced Users

Command Format

```
/deleteitem --index <index>
```

```
/deleteitem -i <index>
```

💡 To use the *command* in this manner, remember to add a `/` before the *command*, such as `/deleteitem`.

1. John wants to delete Fried Rice, which has *index* number 8.

Please enter a command:

```
/deleteitem -i 8
```

```
Item deleted successfully.
```

Error Messages

! The next example is an invalid input, designed to show off the error messages we have in place. This is not the full list of error messages. Additionally, these error messages will be shown for both New User and Experienced User commands if the mistake is made.

1. Invalid *Index*

You will not be able to delete an item if the *index* number of that item does not exist in the menu. In other words, you cannot remove something that does not exist!

```
Please enter a command:  
/deleteitem -i 10  
  
Error: Index does not exist.
```

Solution: Use a valid *index*.

[Back to table of contents](#)

Update an Item

Realised you entered the wrong name or price? You can update your items with this *command*!

📖 John has been happily using MoneyGoWhere to input all his items. However, he is a bit clumsy and entered a few wrong things. This is what the menu currently looks like:

Index	Name	Price
-----	-----	-----
0	White Chicken Rice	4.50
1	Egg	0.80
2	Roast Chicken Rice	4.50
3	Chicken Hor Fun	5.00
4	Chicken Porridge	3.80
5	Dumpling Soup	5.00
6	Vegebatle	4.20
7	Chicken Liver	1.50
8	Curry Chicken Rice	55.00

He wants to change two things. First, the word "Vegebatles" at *index*) number 6 is spelt wrongly. Also, he wants to increase the price of "Vegetables" to \$2.00. Next, the item "Curry Chicken Rice" should cost \$5.50. He can update these items with the `updateitem` *command*.

💡 There are restrictions for the *index*, name, and price of items, as shown below:

Option	Description	Restrictions
Index	The <i>index</i> number of the item.	Must be a valid <i>index</i> from <code>listitem</code> , cannot be a negative number. This <i>option</i> is compulsory.
Name	The new name of the item.	It cannot already exist in the menu. This <i>option</i> is not compulsory.
Price	The new price of the item.	It can have at most two <i>decimal places</i> . This <i>option</i> is not compulsory.

For New Users

- John wants to change spelling of the word "Vegebatle" to "Vegetable". To do so, he first enters the *command* `updateitem`, then confirms that he wants to change the name. He puts in the correct spelling, without quotation marks, and then indicates he does not want to change the price.

Please enter a command:

`updateitem`

Please enter the item's index or use "/cancel" to abort:

`6`

Would you like to update item name? (yes/no) or use "/cancel" to abort

`yes`

Please enter the item's name or use "/cancel" to abort:

`Vegetable`

Would you like to update item price? (yes/no) or use "/cancel" to abort

`no`

`Item updated successfully.`

- John wants to change the price of "Curry Chicken Rice" to \$5.50, instead of \$55. To do so, he first enters the `command`, indicates he does not want to change the name, then confirms he wants to change the price. He then puts in the correct price.

Please enter a command:

`updateitem`

Please enter the item's index or use "/cancel" to abort:

`8`

Would you like to update item name? (yes/no) or use "/cancel" to abort

`no`

Would you like to update item price? (yes/no) or use "/cancel" to abort

`yes`

Please enter the item's price or use "/cancel" to abort:

`5.50`

`Item updated successfully.`

- After updating, John checks to make sure that the items are now correct with `listitem`.

Please enter a command:

listitem

Index	Name	Price
0	White Chicken Rice	4.50
1	Egg	0.80
2	Roast Chicken Rice	4.50
3	Chicken Hor Fun	5.00
4	Chicken Porridge	3.80
5	Dumpling Soup	5.00
6	Vegetable	4.20
7	Chicken Liver	1.50
8	Curry Chicken Rice	5.50

For Experienced Users

Command Format

```
/updateitem --index <index> {--name "<name>"} {--price <price>}
```

```
/additem -i <index> {-n "<name>"} {-p <price>}
```

💡 To use the *command* in this manner, remember to add a `/` before the *command*, such as `/updateitem`.

1. John wants to change the name “Vegebatle” to “Vegetable”. The item is currently at *index* 6.

```
Please enter a command:  
/updateitem -i 6 -n "Vegetable"  
  
Item updated successfully.
```

2. Next, John wants to change the price of "Curry Chicken Rice", at *index* 8, to \$5.50 instead of \$55.

```
Please enter a command:  
/updateitem -i 8 -p 5.5  
  
Item updated successfully.
```

3. After updating, John checks to make sure that the items are now correct with `/listitem`.

Please enter a command:

```
/listitem
```

Index	Name	Price
0	White Chicken Rice	4.50
1	Egg	0.80
2	Roast Chicken Rice	4.50
3	Chicken Hor Fun	5.00
4	Chicken Porridge	3.80
5	Dumpling Soup	5.00
6	Vegetable	4.20
7	Chicken Liver	1.50
8	Curry Chicken Rice	5.50

Error Messages

! The next example is an invalid input, designed to show off the error messages we have in place. This is not the full list of error messages. Additionally, these error messages will be shown for both New User and Experienced User commands if the mistake is made.

1. Missing *Index*

Without indicating the *index*, MoneyGoWhere will not know what item you want to update.

Please enter a command:

```
/updateitem -n "Vegetable" -p 5.3
```

Error: Please include the item's index using: -i <index>

Solution: Include the *index*.

[Back to table of contents](#)

Find an Item

If you forget the [index](#) of an item, or want to see all items with a specific word in the name, you can use this [command](#).

📖 It's a new day and John wants to continue exploring MoneyGoWhere. However, his memory isn't the best, and he cannot remember how many items he has with the word "Chicken" in the name. So, he wants to find all of these items, with the `finditem` [command](#).

💡 There are restrictions for keyword, as shown below:

Option	Description	Restrictions
<code>Keyword</code>	The name or part of the item name you are searching for.	The keyword cannot be empty.

For New Users

1. John enters the [command](#), and then enters "chicken" to search for all items that have "chicken" in the name, regardless of whether the item name contains capital letters.

Please enter a command:

finditem

Please enter the keyword to search for or use "/cancel" to abort:

chicken

Index	Name	Price
0	White Chicken Rice	4.50
2	Roast Chicken Rice	4.50
3	Chicken Hor Fun	5.00
4	Chicken Porridge	3.80
7	Chicken Liver	1.50
8	Curry Chicken Rice	5.50

finditem completed!

For Experienced Users

Command Format

```
/finditem "<keyword>"
```

💡 To use the *command* in this manner, remember to add a `/` before the *command*, such as `/finditem`.

1. John enters the *command*, and then enters "chicken" to search for all items that have "chicken" in the name, regardless of capitalization.

Please enter a command:

```
/finditem "Chicken"
```

Index	Name	Price
0	White Chicken Rice	4.50
2	Roast Chicken Rice	4.50
3	Chicken Hor Fun	5.00
4	Chicken Porridge	3.80
7	Chicken Liver	1.50
8	Curry Chicken Rice	5.50

finditem completed!

Error Messages

! The next example is an invalid input, designed to show off the error messages we have in place. This is not the full list of error messages. Additionally, these error messages will be shown for both New User and Experienced User *command* if the mistake is made.

1. Not entering a keyword

Without a keyword, MoneyGoWhere will not know what to look for your menu.

Please enter a command:

```
/finditem
```

Error: Please specify the keyword to search for.

Solution: Include the word or letters you want to look for.

[Back to table of contents](#)

Order Features

If you have been successful in adding items to your menu, congratulations! You are now ready to start taking orders.

During business operations, you will be spending almost all of your time adding orders on MoneyGoWhere.

Therefore, it is a good idea to give this section a read if you want to familiarise yourself with adding orders.

Order Table of Contents

- [Add an Order](#)
- [List all Orders](#)
- [Refund an Order](#)

Add an Order

After adding your items to the menu, you can begin to take orders from your many customers with this [command](#).

Once an order has been added, the program will calculate the total cost of the order. You will then be asked to enter how much the customer gives you into the program, and MoneyGoWhere will calculate the change for you.

📖 A few days later, John's Chicken Rice stall is finally open! John is so happy that he has MoneyGoWhere as it is very easy and intuitive to use. After familiarizing himself with the system, John is ready to take an order from his hungry customers. His menu currently looks like this:

Index	Name	Price
-----	-----	-----
0	White Chicken Rice	4.50
1	Egg	0.80
2	Roast Chicken Rice	4.50
3	Chicken Hor Fun	5.00
4	Chicken Porridge	3.80
5	Dumpling Soup	5.00
6	Vegetable	4.20
7	Chicken Liver	1.50
8	Curry Chicken Rice	5.50

His first customer orders just one plate of Roast Chicken Rice, and the second orders two plates of White Chicken Rice, and two eggs. To add their orders, he will use the `addorder` [command](#).

💡 There are restrictions for the [index](#)/name and quantity of items, as shown below:

Option	Description	Restrictions
<code>Index/Name</code>	The index number or name of the item in the menu.	If you choose to enter an index number, it must be a valid index number from <code>listitem</code> . The index number cannot be negative.

Option	Description	Restrictions
		<p>You may also choose to enter the full or partial name of the item. However, the entered part of the name must be specific.</p> <p>In other words, if your menu contains items with names such as <code>Chicken Rice</code>, <code>Chicken Noodle</code>, and <code>Cereal</code>, then you at least have to enter <code>chicken r</code>, <code>ice</code>, or <code>rice</code> so that the program knows that you are trying to enter <code>Chicken Rice</code> and not any of the other two items.</p>
<code>Quantity</code>	The number of the specified menu item that the customer wishes to order.	The quantity cannot be negative, and avoid entering numbers larger than <code>10000</code> as it can cause the program to work incorrectly.

💡 As for payment, there are restrictions for the amount, and payment type:

Option	Description	Restrictions
<code>amount</code>	The amount paid by your customer.	Can have up to two decimal points, and has to be the same or greater than the cost of the order.
<code>type</code>	The type of payment.	The accepted inputs are <code>cash</code> , <code>card</code> , or <code>others</code> .

For New Users

1. John enters the `command` `addorder`.

He forgets the `index` of "Roast Chicken Rice", and instead chooses to search by name instead of `index`. Take note that the item name must be entered within `" "` marks.

Next, he enters the quantity and states that he has no more items to add. As the customer pays with cash, John then enters the payment information.

Please enter a command:

`addorder`

Please enter the item's name or index:

`"Roast chicken rice"`

Please enter the quantity of the item:

`1`

Do you have more items to add? (yes/no/cancel)

`no`

Subtotal: \$4.50

Order has been added successfully.

Please use `/pay -a <amount> -t <type>` or `pay` to make payment.

`pay`

Please enter payment type.

`cash`

Please enter amount to pay.

`4.5`

Order has been paid!

Order is added!

💡 When prompted to add more items, entering anything other than the provided inputs will cause MoneyGoWhere to interpret the input as `cancel`.

2. For the second customer, John repeats the same process.

He enters the *command*, then the *index* of "White Chicken Rice" and indicates the quantity.

He repeats the same process for adding "Egg" to the order, then completes the process by entering the payment details.

```
Please enter a command:
addorder
Please enter the item's name or index:
0
Please enter the quantity of the item:
2
Do you have more items to add? (yes/no/cancel)
yes
Please enter the item's name or index:
"egg"
Please enter the quantity of the item:
2
Do you have more items to add? (yes/no/cancel)
no

Subtotal: $10.60
Order has been added successfully.
Please use /pay -a <amount> -t <type> or pay to make payment.
pay
Please enter payment type.
card
Please enter amount to pay.
10.60
Order has been paid!

Order is added!
```

For Experienced Users

Command Format

There are two main ways in which an order can be entered using a single `/addorder` *command*. Listed below are the acceptable formats of the `/addorder` *command*.

1. To add a single item into an order, using the item *index*, by using the item name, or by using part of the item name, use any of these formats:

```
/addorder -i <index> -q <quantity>
```

```
/addorder --item <index> -q <quantity>
```

```
/addorder -i "<name>" --quantity <quantity>
```

```
/addorder --item "<name>" --quantity <quantity>
```

If the quantity is not specified, it will be set to 1.

2. To add multiple items into an order using the item *indexes*, by using the item names, or by using part of the item names, use any of these formats:

```
/addorder -I [<index>:<quantity>,"<name>":<quantity>,...]
```

```
/addorder --items [<index>:<quantity>,"<name>":<quantity>,...]
```

💡 To use the *command* in this manner, remember to add a `/` before the *command*, such as `/addorder`.

Here is an example of how the `/addorder` *command* may be used to add orders.

1. John enters the *command* `/addorder`. He uses the *index* of "Roast Chicken Rice", and enters the quantity.

As his customer pays with cash, John then enters the payment information with the `/pay` *command* and by entering the customer's payment details based on the instructions from the program.

```
Please enter a command:
```

```
/addorder -i 2 -q 1
```

```
Subtotal: $4.50
```

```
Order has been added successfully.
```

```
Please use /pay -a <amount> -t <type> or pay to make payment.
```

```
/pay -a 4.5 -t cash
```

```
Order has been paid!
```

2. For the second customer, John temporarily forgets the *index* of “White Chicken Rice”.

As he knows there is only one item with “White Chicken” in the name, he enters that as the first item, along with the quantity.

Then, he uses the *index* of “Egg”, along with the quantity. After adding both items, he proceeds to enter the payment.

This time, the customer chose to pay by card, which John enters into the application to reflect that mode of payment.

```
Please enter a command:
/addorder -I ["white chicken":2,1:2]

Subtotal: $10.60
Order has been added successfully.
Please use /pay -a <amount> -t <type> or pay to make payment.
/pay -a 10.60 -t card
Order has been paid!
```

Error Messages

! The next example is an invalid input, designed to show off the error messages we have in place. This is not the full list of error messages. Additionally, these error messages will be shown for both New User and Experienced User *commands* if the mistake is made.

1. Insufficient cash

This error message will appear when the customer attempts to pay an amount less than the total cost of their order.

```
Please enter a command:
/addorder -i 2 -q 2

Subtotal: $9.00
Order has been added successfully.
Please use /pay -a <amount> -t <type> or pay to make payment.
/pay -a 8 -t cash
Error: Insufficient amount. Payment amount must be more than or equals to subtotal.
```

Solution: Ensure that the customer pays more than or the exact cost of the order.

2. Paying with card

When your customers pay with their card, entering an amount different from the subtotal will be rejected.

```
Please enter a command:
```

```
/addorder -i 2 -q 2
```

```
Subtotal: $9.00
```

```
Order has been added successfully.
```

```
Please use /pay -a <amount> -t <type> or pay to make payment.
```

```
/pay -a 10 -t card
```

```
Error: Please input exact amount for card payment.
```

Solution: Enter the exact price of the order into the program.

[Back to table of contents](#)

List all Orders

After a long, busy, and profitable day, you can use this *command* to review all the orders you have entered.

This feature allows you to view the cost, order IDs, the items included in each order, and the time at which the orders were made.

📖 John's first day is a huge success! He has fed so many hungry students and could see them enjoying his delicious food. At the end of the day, he wants to take a look at all the orders he has taken over the day. He will use the `listorder` command. Note: Only the first two orders will be shown to reduce the length of the user guide.

💡 This *command* uses only one word. Adding anything else after the *command* will cause MoneyGoWhere to not recognize the *command*.

1. John uses the *command* `listorder` or `/listorder` to look through all his orders.

Please enter a command:

listorder

=====

Order 1

Order ID: 6e424149-9cbf-4666-a702-adddd1b25758

Order status: COMPLETED

Order time: 2023-04-05 12:55:50

1. Roast Chicken Rice x1

Subtotal: \$4.50

=====

Order 2

Order ID: 2dfa2483-4aa7-4bc7-8dc1-1ad1e2c97580

Order status: COMPLETED

Order time: 2023-04-05 12:56:51

1. White Chicken Rice x2

2. Egg x2

Subtotal: \$10.60

=====

All transactions have been listed!

Error Messages

! The next example is an invalid input, designed to show off the error messages we have in place. This is not the full list of error messages. Additionally, these error messages will be shown for both New User and Experienced User *commands* if the mistake is made.

1. Adding words or letters after the *command*.

As mentioned, adding anything after `listorder` or `/listorder` will cause the *command* to be an invalid error.

```
Please enter a command:
listorder a
Error: This command is not recognised.
```

[Back to table of contents](#)

Refund an Order

Did you enter the wrong item, or did a customer change their mind after paying for an item?

Don't worry! If the order has not yet been served or if you allow it, you can use this *command* to mark an order as refunded.

📖 After using MoneyGoWhere for a few days, John has encountered his worst nightmare: an indecisive customer! This customer wanted Curry Chicken Rice with Vegetables, but has now changed his mind. Not wanting to receive bad reviews about his newly opened stall, John decides to give in to his demands for a refund. John checked his list of orders using the `listorder` *command*, and found out that the order ID is 9a382d0f-81ee-4855-b29e-547e8e164f9a. John needs to refund the order, and can do so with the `refundorder` *command*.

💡 There are restrictions for the name and price of items, as shown below:

Option	Description	Restrictions
<code>Order ID</code>	A randomly generated order ID.	The entered ID must be a valid entered as shown exactly from the output of the <code>listorder</code> <i>command</i> .

For New Users

1. John uses `listorder` to look through the list of orders and gets the ID as `9a382d0f-81ee-4855-b29e-547e8e164f9a`. He then copies the ID.

2. Next, he uses the `command` `refundorder` and inputs the copied order ID from earlier, when asked to do so by the program.

```
Please enter a command:
refundorder
Please enter order ID to refund order.
9a382d0f-81ee-4855-b29e-547e8e164f9a

Order is refunded!
```

For Experienced Users

`Command` Format

```
/refundorder -i <order_id>
```

💡 To use the `command` in this manner, remember to add a `/` before the `command`, such as `/refundorder`.

1. John uses `/listorder` to look through the list of orders and gets the ID as `9a382d0f-81ee-4855-b29e-547e8e164f9a`. He copies the order ID from the `CLI`.
2. Next, he uses the `command` `/refundorder` and inputs the copied order ID from earlier.

```
Please enter a command:
/refundorder -i 9a382d0f-81ee-4855-b29e-547e8e164f9a
The order's status is now refunded!
```

Error Messages

❗ The next example is an invalid input, designed to show off the error messages

we have in place. This is not the full list of error messages. Additionally, these error messages will be shown for both New User and Experienced User *commands* if the mistake is made.

1. Wrong order ID.

If you entered an invalid or non-existent order ID, MoneyGoWhere will not recognise it.

Please enter a command:

```
/refundorder -i 9a382d0f-81ee-4855-b29e-547e8e164f9
```

Error: Invalid order ID.

Solution: Copy the order ID directly from `listorder` and paste it to ensure there are no errors. </br>

[Back to table of contents](#)

Statistics and Report

This important feature of MoneyGoWhere allows you to generate reports to view sales data. For now, there is only one way to generate reports.

📖 At this point, John has been running the stall for a few months. He's making so much money that it is hard for him to keep track of how much money he has made. He wants to see four things: 1. For the year 2023, which items bring in the most income? 2. For the month of January, which were ordered the most? 3. John wants to check how his income has changed over the year. 4. John wants to check his income for the first 10 days of March.

Command Format

```
/report {--rank <type>} {--sale <type>} {--year <year>} {--from <start-date> --to <end-date>}
```

```
/report {-r <type>} {-s <type>} {-y <year>} {-f <start-date> -t <end-date>}
```

💡 We understand that these *command* formats may be complicated. Here is a refresher on how such formats can be more easily understood.

1. Any words surrounded by `<>`.

2. *Options* that are surrounded by `{ }` are optional.
3. All values have to be accompanied by *options* (begins with `-`, such as `-n` or `--price`). *Commands* such as `/deleteitem delete -i 10` will be an invalid *command*.

💡 There are restrictions for *options*, as shown below:

Option	Description	Accepted Inputs
<code>rank</code>	Order items in the menu based on criteria.	<code>sales</code> or <code>popular</code>
<code>sale</code>	Shows your income based on criteria.	<code>daily</code> or <code>monthly</code>
<code>year</code>	The year you are trying to generate a report for.	A year in the format <code>YYYY</code> . Entering a negative year will cause the application to not work as intended
<code>from</code> and <code>to</code>	The start and end dates you are trying to generate a report for.	A date in the format <code>DD/MM/YYYY</code>

Don't worry if you find yourself still confused by the format. We admit, it is pretty complex. So, let's break it down:

Rank vs Sale

Of the two, only one can be present in your *command*.

- When generating your report, you can choose between the *options* `rank` or `sale`.
 - Rank orders the items on your menu based on either `sales` (monetary income) or `popular` (quantity ordered).
 - Meanwhile, Sale shows your income, either on a `daily` basis or `monthly` basis.

Year vs From/To

Of the two, only one can be present in your *command*.

- You can choose which dates to generate the report.
 - Year takes the format `YYYY`
 - From/To takes the format `DD/MM/YYYY`

📖 Now, we will show you how John was able to use this *command* to achieve his

goals.

1. John indicates the rank *option* to be sales , and the year to be 2023 .

Please enter a command:

```
/report -r sales -y 2023
```

Rank by sales		
Date: 01/01/2023 - 31/12/2023		

Rank	Name	Sales(\$)
-----	-----	-----
1	Roast Chicken Rice	1656.00
2	Chicken Hor Fun	1625.00
3	White Chicken Rice	1080.00
4	Chicken Liver	150.00
5	Egg	120.80
6	Dumpling Soup	110.00
7	Chicken Porridge	87.40
8	Vegetable	29.40
9	Curry Chicken Rice	11.00

2. John indicates the rank *option* to be popular , and the date range to be from 01 Jan 2023 to 31 Jan 2023 .

```
Please enter a command:
/report -r popular -from 01/01/2023 -to 31/01/2023

| ----- |
| Rank by quantity sold |
| Date: 01/01/2023 - 01/02/2023 |
| ----- |
| Rank | Name | Count |
| ----- | ----- | ----- |
| 1 | Dumpling Soup | 10 |
| 2 | Roast Chicken Rice | 5 |
| 3 | Vegetable | 4 |
| 4 | Chicken Porridge | 3 |
| 5 | Egg | 2 |
| 6 | Chicken Hor Fun | 0 |
| 7 | Chicken Liver | 0 |
| 8 | White Chicken Rice | 0 |
| 9 | Curry Chicken Rice | 0 |
| ----- |
```

3. Now, he wants to look at his monthly sales report for 2023. He indicates the `sale option` to be `monthly` and enters `2023` for the `year option`.

Note that the remaining months have been removed as they are all empty.

```
Please enter a command:
/report -s monthly -y 2023

| ----- |
| Monthly statistic for the year 2023 |
| Total sales: $282.30 |
| ----- |
| Month | Sales($) | Performance |
| ----- | ----- | ----- |
| Jan 2023 | 96.60 | |
| Feb 2023 | 88.50 | |
| Mar 2023 | 97.20 | |
| Apr 2023 | 0.00 | |
| ----- |
```

4. Lastly, he wants to check his daily sales report for the first 10 days of March 2023. He indicates the `sale option` to be `daily` and enters the date range `01/03/2023` to

10/03/2023 .

```
Please enter a command:
/report -s daily -f 01/03/2023 -t 10/03/2023

-----
| Daily statistic for the date range 01/03/2023 - 10/03/2023
| Total sales: $97.20
| -----
| Date       | Sales($) | Performance
| -----
| 01/03/2023 | 9.00     | |||||
| -----
| 02/03/2023 | 7.60     | |||||
| -----
| 03/03/2023 | 16.80    | |||||
| -----
| 04/03/2023 | 3.50     | ||| | |
|---|---|---|---|---|---|---|
| 05/03/2023 | 9.70     | |||||
| -----
| 06/03/2023 | 9.70     | |||||
| -----
| 07/03/2023 | 33.20    | |||||
| -----
| 08/03/2023 | 7.70     | |||||
| -----
| 09/03/2023 | 0.00     |
| -----
| 10/03/2023 | 0.00     |
| -----
```

Error Messages

! The next example is an invalid input, designed to show off the error messages we have in place. This is not the full list of error messages.

1. Generating a Montly Sale report with From/To

This happens when you try to generate a montly sale report without using the `year option`.

```
Please enter a command:
/report -s monthly -from 14/03/2023 -to 15/09/2023
Error: Use the [-y|--year] option instead of [-f|--from] and [-t|--to] for monthly sales report
```

Solution: When generating a monthly sale report, use the `year option`.

2. Wrong Date Format

This happens if you enter the wrong format for the date, such as using `-` instead of `/`.

```
Please enter a command:
/report -s daily -from 03/14/2023 -to 09/15/2023
Error: Date format provided in [-f|--from] and/or [-t|--to] is/are not recognised
```

Solution: Use the correct format of `DD/MM/YYYY`.

Frequently Asked Questions

Q: I'm switching to a new laptop. Do I need to re-enter all the items on my menu? And what about my transactions?

A: Don't worry! At MoneyGoWhere, we understand that you may change the laptop you use. When changing to a new laptop, simply copy the entire folder! This way, the application and all your data will stay the same across devices.

Q: Where is my save file?

A: Your menu and transaction list are stored in a folder called `datastore`. Inside it, you should see two files: `menu.json` and `orders.json`. However, all changes to your data should be made through the application itself, as we are not liable for any data loss as a result of tampering with those files.

Glossary

Term	Explanation
Hyperlink	A phrase or text that you can click.
Command Line Interface	A text-based interface. This means there are no icons to click and everything has to be typed.
Command	An instruction given to the program.
Terminal	A tool used for CLI programmes to be run. It accepts text input and outputs text.
Decimal Places	The number of digits behind a decimal point. For example <code>2.123</code> has 3 decimal places, while <code>4.20</code> has 2 decimal places.
Index	A set of ordered whole numbers used to indicate elements. For example, you can think about the different levels in a building.
Option	Used to specify instructions and change the behaviour of a command. In this application, options have a short-form and a long-form, ie. <code>-n</code> and <code>--name</code> .

`tp` is maintained by [AY2223S2-CS2113T-T09-2](#).

This page was generated by [GitHub Pages](#).