SCHOOL OF INFOCOMM TECHNOLOGY

Diploma in Information Technology
Diploma in Cybersecurity & Digital Forensic
Diploma in Data Science

PROGRAMMING II

Year 2025/26 - Semester 2

ASSIGNMENT

Duration	:	2½ weeks (26 Jan 2026 to 11 Feb 2026)
Weightage	:	30% of total coursework
Individual/Team	:	Team of 2
Format	:	Programming - Class Implementation (10%) Basic Features (50%) Advanced Features (20%) Walkthrough Test (20%)
Cut-Off Date/Time:		Wednesday, 11 February 2026, 8.00 am

Penalty for late submission:

- 10% of the marks will be deducted for each day (inclusive of Saturdays, Sundays and public holidays) after the deadline.
- No submission will be accepted after 18 Feb 2026, 2359hrs.

There is a total of 14 pages (including this page) in this handout.

WARNING

If a student is found to have submitted work not done by him/her, he/she will not be awarded any marks for this assignment. Disciplinary action will also be taken.
Similar action will be taken for the student who allows other student(s) to copy his/her work.

Gruberoo Food Delivery System

In this assignment, you are to apply Object Oriented Programming to develop a simple ***Food Delivery System***. The assignment requirements described below are broken down into 2 stages of development, described in this document as '***Basic Features***' and '***Advanced Features***'. You are advised to do your programming progressively in these stages. Refer to the '***Grading Criteria***' to have an idea of how the different components are graded.

1. Background

Gruberoo is a food delivery application (app) that connects customers with restaurants, allowing them to order food and have it delivered to their chosen address. The system manages restaurants, food menus, customers, orders, payments, and delivery information. You have been tasked to develop a simplified version of the Gruberoo system using Object-Oriented Programming principles.

Restaurant personnel may sign up with their restaurant name and email address. Upon sign-up, a Restaurant ID is automatically assigned by the system to uniquely identify the restaurant. Restaurants can then list food items on their menus. Each restaurant has at least one menu. Each food item is uniquely identified by its item name and is described by a description and price. Restaurants may also create special offers, such as holiday discounts or delivery promotions. Gruberoo charges a fee of 30% for each order.

Customers sign up with their name and email address. After signing up, they can browse restaurants, search food items, and place orders. An order consists of multiple food items and includes a delivery date, time, and address. Upon submission, the system assigns a unique Order ID and records the order's creation date and time.

Customers then pay using their preferred payment method (Credit Card, PayPal, Cash on Delivery, etc.). Payment includes a standard $5.00 delivery fee. Once payment succeeds, the system notifies the restaurant. The restaurant may either confirm or reject the order.

- If rejected, Gruberoo refunds the customer, updates the order status and copies the order to a refund stack.
- If confirmed, Gruberoo updates the order status, and the restaurant begins preparing the order shortly before the delivery time.

Customers may still modify or cancel an order after payment, but only before preparation begins. Upon a successful cancellation, the restaurant refunds the payment. All rejected or cancelled orders are copied to the refund stack. All orders (delivered, rejected or cancelled) are kept in the system for 1 year, after which they are archived and removed.

Gruberoo continues to expand and is exploring new features such as:

- dividing restaurant menus into sub-menus
- allowing customers to customise food items (extra toppings, upgrades, etc.)
- notifying customers of new special offers
- allowing customers to create one-click favourite orders

A class diagram for the Gruberoo system is provided in Figure 1 to guide your implementation. A state diagram showing the states of the Order as it is processed is provided in Figure 2.

Here are some data from the respective files:

Restaurant ID	Name	Email
R001	Grill House	grillhouse@email.com
R002	Noodle Palace	noodlepalace@email.com
R003	Bento Express	bentoexpress@email.com

Table 1 – Sample Restaurant Data

Restaurant ID	Item Name	Description	Price ($)
R001	Chicken Rice	Steamed chicken with fragrant rice	5.50
R001	Beef Burger	Grilled beef patty with fries	9.80
R002	Spicy Ramen	House-special broth with chilli oil	11.20
R001	Caesar Salad	Romaine lettuce with house dressing	7.00

Table 2 – Sample Restaurant Food Item Data

Restaurant ID	Offer Code	Description	Discount Amount (%)
R001	PHOL	Public Holiday Discount	10
R003	DELI	Free Delivery for Orders over $30	-

Table 3 – Sample Special Offer Data

Name	Email
Alice Tan	alice.tan@email.com
Joseph Lim	joseph.lim@email.com
Wendy Ong	wendy.ong@email.com

Table 4 – Sample Customer Data

Order ID	Customer Email	Restaurant ID	…	Total Amount ($)	Status
1001	alice.tan@email.com	R003	. . .	28.4	Delivered
1002	joseph.lim@email.com	R001	. . .	33.3	Cancelled
1003	wendy.ong@email.com	R002	. . .	16.2	Preparing

Table 5 – Sample Order Data

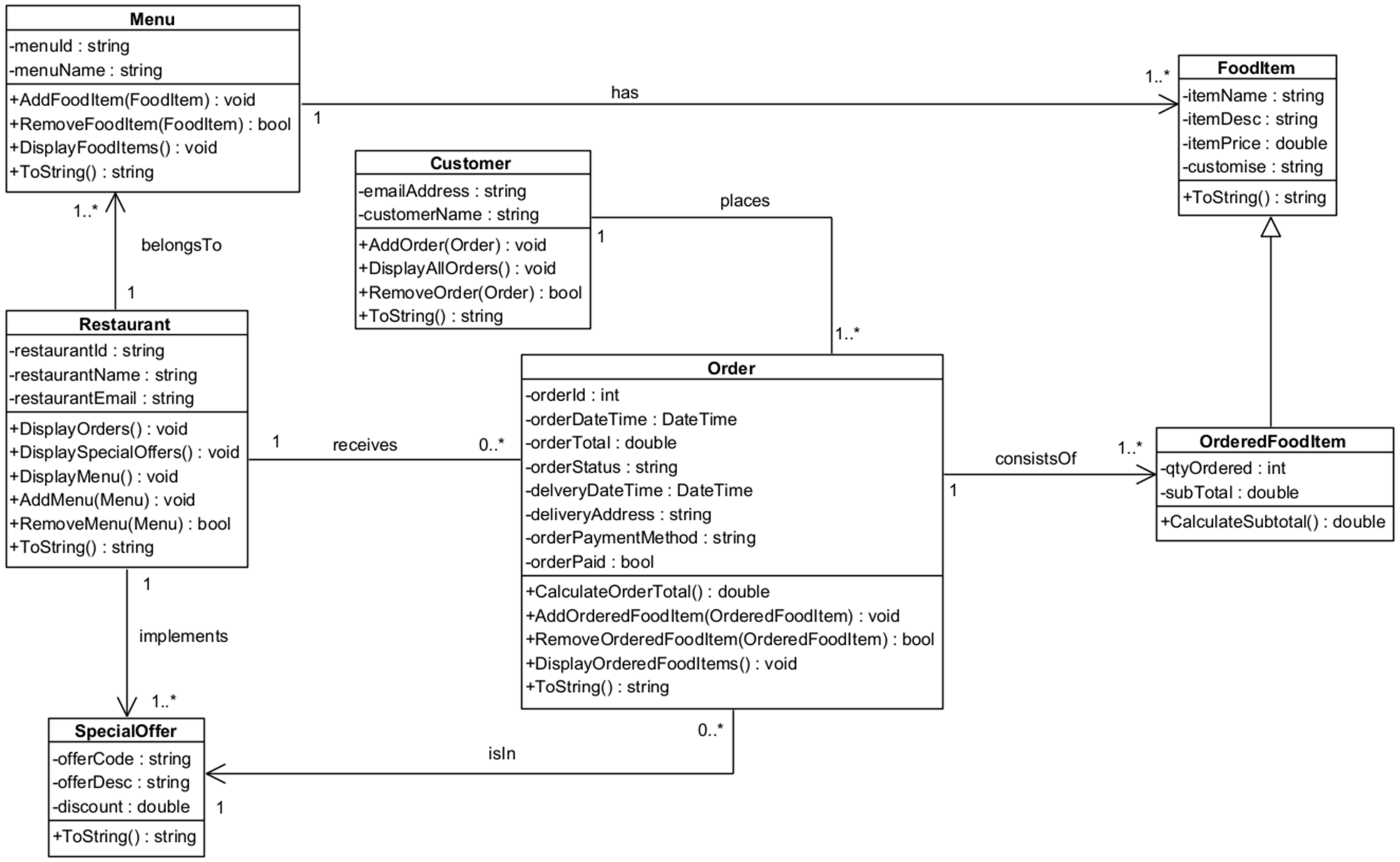

Figure 1: Class Diagram for Gruberoo System

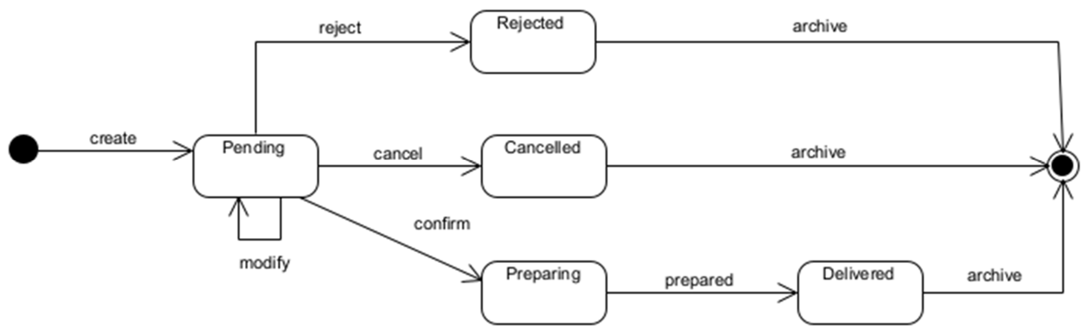

Figure 2 – States of an Order

2. Class Implementation – 10% Group

The team must divide the work when implementing the classes.

**Note it is highly recommended that teams use Git and GitHub to work together on the project and track their changes.*

3. Basic Features – 50% Individual

Your program is required to create Restaurants, Customers, Food Items, and Orders from the given data files at startup (e.g., restaurants.csv, fooditems.csv, customers.csv, orders.csv). After loading, the system will display a menu for the user to repeatedly perform the features described below until exit. When exit is chosen, the information in the queue and stack are saved to **new** files queue.csv and stack.csv respectively.
The features required are described below, together with sample output screens (user input are in **bold**).

```
Welcome to the Gruberoo Food Delivery System
15 restaurants loaded!
51 food items loaded!
20 customers loaded!
35 orders loaded!
```

Figure 3 – start of program

```
===== Gruberoo Food Delivery System =====
1. List all restaurants and menu items
2. List all orders
3. Create a new order
4. Process an order
5. Modify an existing order
6. Delete an existing order
0. Exit
Enter your choice:
```

Figure 4 – main menu

1) Load files (restaurants and food items)

- *load the **restaurant.csv** file*
- *create the Restaurant objects based on the data loaded*
- *load the **fooditems.csv** file*
- *create the FoodItem objects and assign them to their respective restaurants*

2) Load files (customers and orders)

- *load the **customers.csv** file*
- *create the Customer objects based on the data loaded*
- *load the **orders.csv** file*
- *create Order objects and place them into the Restaurant's Order Queue and the Customer's Order List.*

3) List all restaurants and menu items

- *display all restaurants with their details. For each restaurant, display its food items with name, description and price.*

```
All Restaurants and Menu Items
==============================
Restaurant: Grill House (R001)
  - Chicken Rice: Steamed chicken with fragrant rice - $5.50
  - Beef Burger: Grilled beef patty with fries - $9.80
  - Caesar Salad: Romaine lettuce with house dressing - $7.00

Restaurant: Noodle Palace (R002)
  - Spicy Ramen: House-special broth with chilli oil - $11.20
```

Figure 5 – list all restaurants and menu items (sample)

4) List all orders with basic information

- *display ALL orders with the required details:*
  Order ID, Customer Name, Restaurant Name, Delivery Time, Total Amount, and Order Status

```
All Orders
==========
Order ID   Customer     Restaurant      Delivery Date/Time    Amount    Status
--------   ----------   -------------   ------------------    ------    ---------
1001       Alice Tan    Bento Express   12/02/2026 12:00      $28.40    Delivered
1002       Joseph Lim   Grill House     13/02/2026 18:30      $33.30    Cancelled
1003       Wendy Ong    Noodle Palace   14/02/2026 19:00      $16.20    Preparing
```

Figure 6 – list all orders (sample)

5) Create a new order

- *prompt the user to enter Customer Email, Restaurant ID, Delivery Date/Time, Delivery Address*
- *display available Food Items and allow the user to select multiple items and quantity*
- *prompt the user to ask whether he wants to apply special requests (e.g. extra toppings). If [Y], to prompt the user to enter the special request. Only one special request is allowed.*
- *calculate the order total*
- *prompt the user to ask for payment. If [Y], to prompt the user to enter [CC] for Credit Card, [PP] for PayPal or [CD] for Cash on Delivery. If [N], exit this feature.*
- *save the payment method chosen*
- *update the status of order to "Pending"*
- *assign a new Order ID*
- *add the new Order to the Restaurant's Order Queue and the Customer's Order List*
- *append the new order to orders.csv*
- *display a confirmation message*

```
Create New Order
================
Enter Customer Email: alice.tan@email.com
Enter Restaurant ID: R001
Enter Delivery Date (dd/mm/yyyy): 15/02/2026
Enter Delivery Time (hh:mm): 12:30
Enter Delivery Address: 123 Main Street

Available Food Items:
1. Chicken Rice - $5.50
2. Beef Burger - $9.80
3. Caesar Salad - $7.00
Enter item number (0 to finish): 1
Enter quantity: 2
Enter item number (0 to finish): 2
Enter quantity: 1
Enter item number (0 to finish): 0
Add special request? [Y/N]: N

Order Total: $20.80 + $5.00 (delivery) = $25.80
Proceed to payment? [Y/N]: Y

Payment method:
[CC] Credit Card / [PP] PayPal / [CD] Cash on Delivery:  CC

Order 1004 created successfully! Status: Pending
```

Figure 7 – create a new order (sample)

6) Process an order

- *prompt the user to enter a restaurant ID*
- *Loop thru' the Order Queue and display each order for processing.*
  - *display the basic information of the order (order could be pending, cancelled or preparing)*
  - *prompt the user to enter [C] for confirming, [R] for rejecting the order, [S] for skipping or [D] for delivering the order .*
    - *If confirming, check that the order status is "Pending" and update the order status to "Preparing".*
    - *If rejecting, check that the order status is "Pending", update the order status to "Rejected", add the order to the Refund Stack and display refund message.*
    - *If skipping, check that the order status is "Cancelled" and move on to next order.*
    - *If delivering, check that the order status is "Preparing" and update the order status to "Delivered".*
  - *display a message indicating the outcome.*

```
Process Order
=============
Enter Restaurant ID: R001

Order 1004:
Customer: Alice Tan
Ordered Items:
1. Chicken Rice - 2
2. Beef Burger - 1
Delivery date/time: 15/02/2026  12:30
Total Amount:  $25.80
Order Status: Pending

[C]onfirm / [R]eject / [S]kip / [D]eliver: C

Order 1004 confirmed. Status: Preparing
```

Figure 8 – process an order (sample)

7) Modify an existing order

- *prompt the user to enter a Customer Email*
- *display all orders from the Order List that are "Pending"*
- *prompt the user to enter an Order ID*
- *display the items ordered, the delivery address and the delivery time*
- *display all available modifications: Items, Address, Delivery Time*
- *prompt user to select a modification*
- *accept updated data and apply changes (including updating of order total)*
- *prompt the user to pay if there is an increase in the order total*
- *display updated order details.*

```
Modify Order
============
Enter Customer Email: alice.tan@email.com
Pending Orders:
1004
1008
Enter Order ID: 1004
Order Items:
1. Chicken Rice - 2
2. Beef Burger - 1
Address:
123 Main Street
Delivery Date/Time:
15/2/2026, 12:30

Modify: [1] Items [2] Address [3] Delivery Time: 3
Enter new Delivery Time (hh:mm): 14:00

Order 1004 updated. New Delivery Time: 14:00
```

Figure 9 – modify an existing order (sample)

8) Delete an existing order

- *prompt the user to enter a Customer Email*
- *display all orders from the Order List that are "Pending"*
- *prompt the user to enter an Order ID*
- *display the basic information of the order*
- *prompt user to confirm deletion*
- *update the status of the order to "Cancelled", add the order to the Refund Stack and display refund message*
- *display a message indicating the outcome.*

```
Delete Order
============
Enter Customer Email: alice.tan@email.com
Pending Orders:
1004
1008
Enter Order ID: 1004

Customer: Alice Tan
Ordered Items:
1. Chicken Rice - 2
2. Beef Burger – 1
Delivery date/time: 15/02/2026  14:00
Total Amount:  $25.80
Order Status: Pending
Confirm deletion? [Y/N]: Y

Order 1004 cancelled. Refund of $25.80 processed.
```

Figure 10 – delete an existing order (sample)

- **Validations** (and feedback)
  - *The program should handle all invalid or unexpected input by the user, including:*
    - *invalid Order ID*
    - *invalid Restaurant or Customer*
    - *empty or incorrectly formatted values*
    - *invalid delivery date and time*
  - *If invalid input is detected, the program must provide appropriate feedback and re-prompt.*

IMPORTANT INSTRUCTIONS:

- *One student is required to implement features 2, 3, 5 & 7, and another is required to implement features 1, 4, 6 & 8.*
- *An Individual student without a team is required to implement features 1, 2, 3, 4, & 5.*

4. Advanced Features – 20% individual

The main menu shall have additional options to allow these advanced features to be chosen.

(a) Bulk processing of unprocessed orders for a current day

- *identify all orders with status "Pending"*
- *display the total number in the Order Queues with this status*
- *for each order in the queue:*
  - *attempt to process it*
    - *automatically set status to "Rejected" if its delivery time is less than 1 hour*
    - *otherwise automatically set status to "Preparing"*
- *display summary statistics:*
  - *number of orders processed*
  - *number of "Preparing" vs "Rejected" orders*
  - *percentage of automatically processed orders against all orders*

(b) Display total order amount

- *for each Restaurant*
  - *retrieve all successful orders (with status "Delivered")*
  - *compute and display total order amount (less delivery fee per order)*
  - *retrieve all refunded orders*
  - *compute and display total refunds*
- *after processing all restaurants, display:*
  - *total order amount*
  - *total refunds*
  - *final amount Gruberoo earns*

(c) Recommend an additional feature to be implemented (bonus marks are only awarded if the advanced feature is completed)

- *you may gain up to 5 bonus marks if you propose and successfully implement an additional feature per student. Examples include:*
  - *favourite orders*
  - *create an order with a special offer (order total to be re-calculated due to discount)*
  - *customer notifications*
  - *sub-menus for Restaurants*

You are to get APPROVAL from your tutor before implementing the advanced feature.

IMPORTANT INSTRUCTIONS:

- *A student is to implement feature (a), and another implements feature (b).*
- *An Individual student without a team can choose to implement either one of the features.*
- *Please note that you should implement the advanced features only AFTER all the basic features have been fully implemented and working.*
- *NO MARKS will be awarded for the advanced features if the basic features have NOT been fully implemented and working.*

5. ACTIVITY PLAN

Suggestions for Getting Started

There are many ways that you could complete this assignment. It is most important part to think about the entire assignment first so that it is easy to integrate the various parts.

a) Analysis

1. Understand the program specification and the requirements before attempting the assignment.
   e.g. the relationships between the classes
   the use of the attributes in each class

b) Program Design

2. Work out the User Interface needed to get the user input for suitable output.
3. Work out the main logic of the program using Object-Oriented programming techniques;

   i.e. use the inheritance and the association of the classes properly.
4. You are required to use suitable classes and methods appropriately for this assignment.

   Marks will be deducted for inefficient use of the classes/methods or improper use of classes/methods

c) Implementation & Testing

5. Determine the order in which the classes are to be implemented. (Certain classes need to be implemented before other classes can be implemented)
6. Implement the classes **ONE** at a time.
7. Test your program logic to make sure that it works.

   You must prepare test data to see that your program works correctly. All data entry should be validated, and illegal data entry should be highlighted to the user so that the user can enter correct data.

6. DELIVERABLES

You are to develop the project using the Console App. You should name your project as `Snnnnnnnn_PRG2Assignment` *(note: **Snnnnnnnn** is your Student Number)*.

You are required to submit your work in **TWO** stages:

Stage 1 – Due date: Wednesday, Week 16 (4 February 2026) @ 23:59 hours

- ALL the classes (source files) shown in the class diagram and at least 1 basic feature to your **Brightspace Stage 1 Submission**.

  In **EACH** of your source files, you **MUST** include a block comment at the top stating your **student number, student name,** and **partner name** as shown below:

```
//============================================================
// Student Number : S12345678
// Student Name   : Michael Jordan
// Partner Name   : Scottie Pippen
//============================================================
```

  Note: If a student does not submit his/her stage 1 work by the deadline, 20 marks will be deducted.

Stage 2 – Due date: Wednesday, Week 17 (11 February 2026) @ 08:00 hours

- ALL the classes (source files) that you have written for the whole assignment to your **Brightspace Stage 2 Submission**.

  In **EACH** of your source files, you **MUST** include a block comment at the top stating your **student number, student name,** and **partner name** as shown below:

```
//============================================================
// Student Number : S12345678
// Student Name   : Michael Jordan
// Partner Name   : Scottie Pippen
//============================================================
```

7. WALKTHROUGH TEST – 20% INDIVIDUAL

- You are to take the test on **11 Feb 2026 (Wed), 11am to 12:00pm.**

  The context of the test will be the assignment.

  There will be 5 Multiple-choice questions to assess concepts and 1 coding question to assess your ability to write/modify a feature that you have done for the assignment.

  Venues will be announced nearer the date.

8. GRADING CRITERIA

This assignment constitutes 30% of this module.

Performance Criteria for grading the assignment is as described below. Marks awarded will be based on **program code** as well as student's degree of understanding of work done as assessed during the **presentation**.

Grading criteria for the program is given below.

A Grade

- ♦ Program implements the *Basic Features* successfully and efficiently
- ♦ Program implements all the basic *input validations* successfully
- ♦ Program implements *Advanced Features* successfully and efficiently
- ♦ Program demonstrates good design with the correct use of methods
- ♦ Program provides strong evidence of good programming practice
- ♦ Program has been tested adequately

B Grade

- ♦ Program implements the *Basic Features* successfully
- ♦ Program implements some basic *input validations* successfully
- ♦ Program attempts to use methods
- ♦ Program provides sufficient evidence of good programming practice
- ♦ Program has been tested adequately

C Grade

- ♦ Program implements the *Basic Features* successfully
- ♦ Program provides some evidence of good programming practice
- ♦ Program has been tested adequately

D Grade

- ♦ Program implements the *Basic Features* successfully
- ♦ Program has been tested adequately
- ♦ Presentation is awarded at least a 'D' grade

NOTE

- *Evidence of good programming practice include appropriate use of methods, the use of meaningful variable names, proper indentation of code, appropriate and useful comments, adoption of standard naming conventions etc.*
- *Basic Input validation refers to the checking of the inputs entered by the user. e.g. invalid option, invalid date*