

code
sprint NATIONAL CONTEST
MT
COMPETITION

Final Round

Post-Secondary Competition 2025

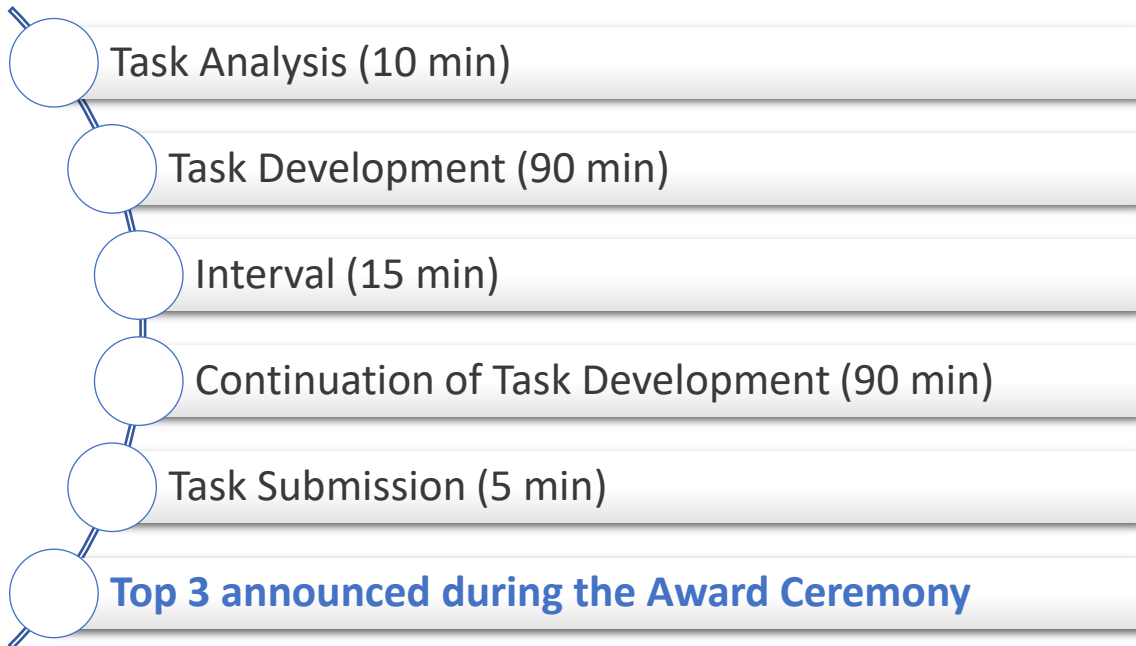


GOVERNMENT OF MALTA
MINISTRY FOR EDUCATION,
SPORT, YOUTH, RESEARCH
AND INNOVATION



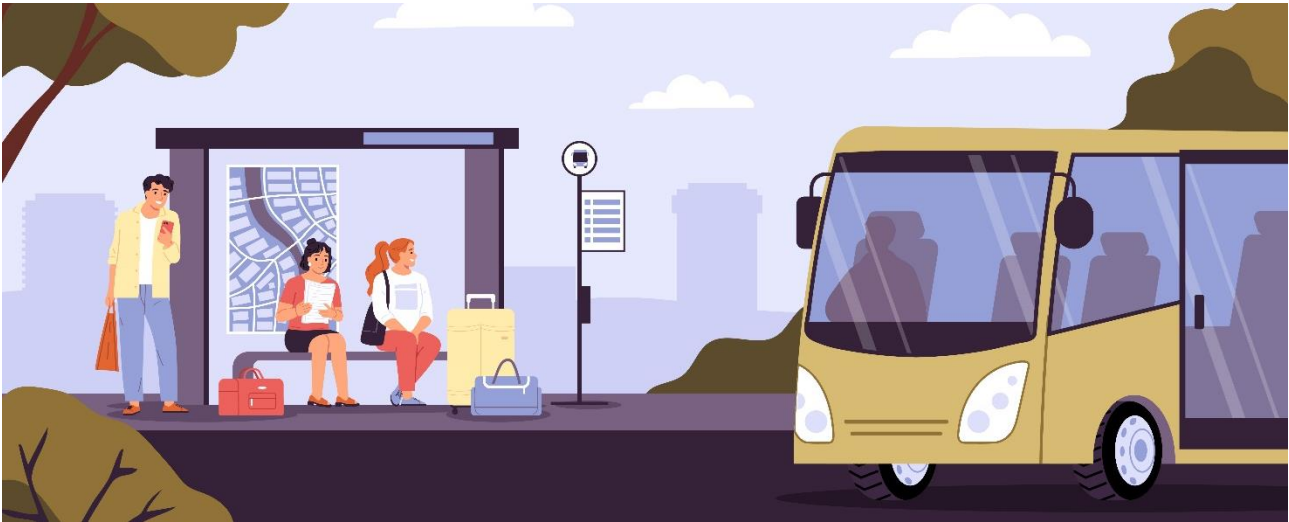
icecampus

Final Round Schedule



BusIt Malta – Track – Connect - Transfer!

In the bustling island of Malta, thousands of daily commuters, locals, and tourists rely on an efficient public transport system. The Malta Public Transport is investing in a next-generation solution called BusIt Malta, and you are the lead developer!




Your job is to develop a simulation of a bus route management system using Java. The system must allow adding new bus routes, searching for bus routes, searching routes by bus stops, finding transfer stops, and planning journeys, all using a user-friendly interface. You must apply good Object-Oriented Programming (OOP) practices, exception handling, and use appropriate data structures such as Arrays, ArrayLists or HashMaps.


Functionality #1: User Interface.

Create an interface with the following options:

=== BusIt Menu ===

1. Add a new bus route
2. Search for bus route by route code
3. Search for bus route/s by bus stop
4. Find transfer stops between two routes
5. Journey Planner (From → To)
6. Exit

 The program will stop only if the user chooses option 6.

 On every operation, the system should provide access to the relevant function and provide meaningful feedback or error messages.

Functionality #2: Predefined bus routes

The system should initialize with a few preloaded bus routes, as shown in Table 1 below. These routes must be **loaded from a file** when the program starts. The source of the data can be any file type of your choice, such as text file, object file, CSV file, JSON file, etc.

🔊 For convenience, a sample text file named *routes.txt* is provided and can be downloaded from www.codesprintmalta.edu.mt/routes.txt. This text file contains the predefined routes in CSV format.

Bus Route	Journey (Bus stops in sequence)
R001	Triton Fountain, City Gate, Upper Barrakka, Kastilja, Konkatedral, Republic Street, Mediterranean Conference Centre, Lower Barrakka, Waterfront, Park & Ride, Triton Fountain
R003	Valletta, Floriana, Hamrun, Santa Venera, Birkirkara, Iklin, Naxxar, Mosta
R037	Valletta, Floriana, Pietà, Msida, University, Mater Dei Hospital
R041	Valletta, Floriana, Marsa, Paola, Fgura, Zabbar, Kottonera
R045	Valletta, Floriana, Msida, Birkirkara, Balzan, Attard, Ta Qali, Rabat, Mdina, Dingli
R072	Marsa, Hamrun, Qormi, Luqa, Zebbug, Siggiewi, Ghar-Lapsi
R081	Paola, Tarxien, Gudja, Luqa, Kirkop, Zurrieq, Hal Far, Birzebbuga
R084	Fgura, Bormla, Birgu, Senglea, Zabbar, Marsascalea, Xghajra
R201	Luqa, Airport, Mqabba, Qrendi, Blue Grotto, Hagar Qim, Siggiewi, Rabat, Mtarfa
R212	Msida, Gzira, Sliema, St Julian, Pembroke, Swieqi, Bahar ic-Caghaq, Bugibba
R218	Bugibba, Qawra, St Pauls Bay, Xemxija, Mellieha, Cirkewwa
R223	Cirkewwa, Mellieha, Ghadira, Golden Bay, Mgarr, St Pauls Bay

Table 1: Predefined Bus Routes

Functionality #3: Add new bus route

1. This is a password-protected feature, accessible only with the password `LetMe!n`
2. The interface must allow the user to add a new bus route by entering a:
 - route code, such as `R101`, and
 - comma-separated list of bus stops in order, such as: `Valletta, Floriana, Msida, Birkirkara, Mosta`
3. The bus routes should start with the uppercase letter 'R' and followed by a three-digit number.
4. The system should not allow bus routes with the same route code.
5. Each bus route cannot have a duplicate bus stop.
6. No two bus routes should have the exact same sequence of bus stops. Each bus route must be unique! For example, `Marsa → Hamrun → Mosta`, and `Mosta → Marsa → Hamrun` are considered as two (2) unique routes.
7. Any newly added bus routes should be saved to the file that contains the predefined routes. This can be the provided `routes.txt` file or any other file format of your choice.
8. Return to BusIT Main Menu if password is incorrect, or bus route is invalid or not unique.

```
=== Bus Route Manager ===  
Enter password to continue: hello  
Access denied. Incorrect password.
```

Screenshot 1: Sample interface for Functionality 3

```
=== Bus Route Manager ===  
Enter password to continue: LetMe!n  
  
Enter route code (e.g., R101): A104  
Invalid route code format. It should start with 'R' followed by 3 digits (e.g., R105).
```

Screenshot 2: Another sample interface for Functionality 3

```
=== Bus Route Manager ===  
Enter password to continue: LetMe!n  
  
Enter route code (e.g., R101): R104  
Enter comma-separated list of stops: Valletta, Floriana, Hamrun, St Venera, Birkirkara  
Route R104 added successfully: Valletta → Floriana → Hamrun → St Venera → Birkirkara
```

Screenshot 3: Another sample interface for Functionality 3

Functionality #4: Search for bus routes by route code

This feature should support:

1. Searching for a bus route and, if found, display all its stops.
2. The search must perform an exact match, no partial matches!
3. Display an appropriate message if the bus route is not found.
4. Display an appropriate message if an invalid bus route is entered.
5. If the user types in the word 'all', the system should display all the bus routes and their stops in numerical order; for example, R301 comes after R288.

```
=== Bus Route Search ===  
Enter route code (e.g., R001) or 'all' to list all routes: R333  
X Bus route not found.
```

Screenshot 4: Sample interface for Functionality 4

```
=== Bus Route Search ===  
Enter route code (e.g., R001) or 'all' to list all routes: R072  
  
Route R072:  
Hamrun → Qormi → Marsa → Luqa → Żebbug → Siġġiewi
```

Screenshot 5: Another sample interface for Functionality 4

```
=== Bus Route Search ===  
Enter route code (e.g., R001) or 'all' to list all routes: all  
  
All Bus Routes and Stops:  
  
R001:  
Triton Fountain → City Gate → Upper Barrakka → Kastilja  
→ Konkatidral → Republic Street → Mediterranean Conference Centre → Lower Barrakka  
→ Waterfront → Park & Ride → Triton Fountain  
  
R003:  
Valletta → Floriana → Hamrun → Santa Venera  
→ Birkirkara → Iklin → Naxxar → Mosta  
  
R037:  
Valletta → Floriana → Pieta → Msida  
→ University → Mater Dei Hospital  
  
R041:  
Valletta → Floriana → Marsa → Paola  
→ Fgura  
  
R045:  
Msida → Birkirkara → Balzan → Attard  
→ Ta' Qali → Rabat → Mdina → Dingli
```

Screenshot 6: Another sample interface for Functionality 4

Functionality #5: Search routes by bus stop

This feature should support:

1. Searching for a bus stop to identify all routes passing through it in numerical order.
2. Display an appropriate message if a bus route is not found.

```
Enter a bus stop to search for: bugiba
X Sorry, no bus route was found serving the stop 'bugiba'.
```

Screenshot 7: Sample interface for Functionality 5

```
Enter a bus stop to search for: bugibba
Bus stop 'bugibba' is served by the following route(s):
- R212
- R218
```

Screenshot 8: Another sample interface for Functionality 5

Functionality #6: Find transfer stops between two routes

This feature should help users identify transfer points by finding common stops between two routes.

1. The user should enter two bus routes, and the system displays the common stops in alphabetical order.
2. Display an appropriate message if any one of the bus routes is not found.
3. Display an appropriate message if any one of the bus routes is invalid.
4. Display an appropriate message if bus routes are the same.

```
Enter the first bus route: R072
Enter the second bus route: r201

Common stops between R072 and r201:
→ Luqa
→ Siggiewi
```

Screenshot 9: Sample interface for Functionality 6

Functionality #7: From → To Journey Planner

This feature helps users identify the best bus route/s to reach their destination bus stop.

1. The user should enter the 'From' and 'To' bus stops. The system should then display all available bus route/s.
2. The results must include all the possible options, including 'Direct' routes and 'Transfer' routes, such as:
 - From Floriana ► To Iklin: Direct Route R003.
 - From Valletta ► To Mellieha: Transfer Route: R037 → Transfer to R212 at Msida → Transfer to R218 at Bugibba.
3. Routes should only include journeys where the 'From' bus stop comes before the 'To' bus stop along the route. *For example, a route from Marsa to Valletta would not exist if the bus travels from Valletta to Marsa.*
4. For each suggested bus route, generate a random time between 5 and 15 minutes to simulate the arrival time at the initial bus stop.
5. If there are multiple journey options, the system should display all available options in sequence, ordered by the earliest arrival time at the starting bus stop.
6. Display an appropriate message if either the 'From' or 'To' bus stop is not found.
7. Display an appropriate message if no route options are available.

```
From >> valletta
To >> University

 1 direct route found:
 R037 ⌚ arrives in 6 minutes at Valletta
-----
```

Screenshot 10: Sample interface for Functionality 7

```
From >> Floriana
To >> Msida

 2 direct routes found:
 R045 ⌚ arrives in 2 minutes at Floriana
 R037 ⌚ arrives in 6 minutes at Floriana
-----
```

Screenshot 11: Another sample interface for Functionality 7

```
From >> Marsa
To >> xemxija
✗ No bus routes available
-----
```

Screenshot 12: Another sample interface for Functionality 7


```
From >> Valletta
```

```
To >> MELLIEHA
```

```
☑ 1 Transfer Route found:  
☞ R037 ⌚ arrives in 8 minutes at Valletta  
☞ Transfer to R202 at Msida  
☞ Transfer to R208 at Bugibba
```

Screenshot 13: Another sample interface for Functionality 7

Functionality #8: Validation Processes

In addition to the validation processes mentioned in the previous features, your program must implement the following validations:

1. Empty user input should be ignored and not treated as invalid. The program should re-prompt the user to enter the requested data/option.
2. All user inputs should be case-insensitive to improve usability and prevent errors due to case mismatches.
3. Prevent any runtime errors due to invalid user actions.

Functionality #9: Modular Programming

- Structure your code using modular and object-oriented structure, using appropriate classes, methods, and Java conventions.
- Use ArrayLists, HashMaps, and/or HashSets where appropriate.
- Include Java naming conventions, indent and space your code consistently, and use comments when required.

Name the class containing the main method '**RunApp**'.

Name the file containing the Bus Routes '**routes**',
such as *routes.txt*, or *routes.json*, or *routes.csv*, or *routes.obj*, etc.

Submit your program in a folder named **BusIT_name_surname**
such as *BusIT_john_abela*

Assessment Rubric

Program Runs	User Friendly /Experience Interface	Proper use of Comments	Proper Conventions (Camel Case, meaningful var names etc.)	User Input
Suitable Prompts / Messages displayed	Name of Folder & Class/es	Adding Bus Routes	Search Bus Route	Display 'ALL' Bus Routes
Search for Route/s by Bus Stop	Find Transfer Stops Between Two Routes	To/From Available Direct Routes	To/From Available Transfer Routes	Load Pre-set Bus Routes from File
Save New Bus Routes in File	Sorting Displayed Routes in Numerical Order	Sorting Displayed To/From Routes by Arrival Time	Generating Random Arrival Time	Password Protected Feature
Proper use of Data Structures	Modular Code	Code Efficiency	Other Functions / Flow	
Validations				
Main Menu Options	Valid Route Codes	Route Code Already Exists (Adding new Routes)	Non-Duplicate Bus Stops in Sequence (Adding new Routes)	Unique Route Code (Adding new Routes)
Validations				
Same Route Codes (Finding Transfer Stops)	Ignore Empty Input	Avoid Runtime Errors	User Input Case Sensitivity	
Maximum Score: 66 + 2 for every extra feature				
0 – Not Satisfactorily 1- Partly Satisfactorily 2- Entirely Satisfactorily				

code sprint

NATIONAL CONTEST COMPETITION
MT

ORGANISED BY



GOVERNMENT OF MALTA
MINISTRY FOR EDUCATION,
SPORT, YOUTH, RESEARCH
AND INNOVATION



icecampus

POWERED BY TOP BRANDS

MAIN TITLE SPONSORS



COMMUNITY & EXPERIENCE SPONSORS



SUPPORTING SPONSORS



MEDIA PARTNERS

