CMPT 276 Group 205

Macrohard Corp.

TrackMaster

Release 3

Release Date: 2024/07/03

Approved By:

James Isaac, *Project Manager* _____

Bowen Jin, *CEO* _____

Sit Kwan Wai, *CFO* _____

Saadati Aryan, *Chief of Staff* _____

Release History Page

Rel. 1	- 2024/06/07
	- Was composed of just the preliminary Requirements Spec.
Rel. 2	- 2024/06/19
	- Initial release of the user manual, providing comprehensive guidance on *TrackMaster's* features and functionalities
Rel. 3	- 2024/07/03
	- Initial release of the architectural design document as well as modules headers and source code.

Release Table Page

Configuration Item:	Version	Number of Each Configuration Item in:	
	Rel. 1	Rel. 2	Rel. 3
Documents			
Requirements Spec	1	1	1
User Manual		1	1
Architectural Design Document			1
Detailed Design Document			
Integration Report			
Modules			
Main.cpp			1
Change.h			1
Change.cpp			
InternalRequester.h			1
InternalRequester.cpp			
LifeCycleController.h			1
LifeCycleController.cpp			
Request.h			1
Request.cpp			
Requester.h			1
Requester.cpp			
PrintController.h			1
PrintController.cpp			

Product.h	1
Product.cpp	
ProductRelease.h	1
ProductRelease.cpp	
ScenarioController.h	1
ScenarioController.cpp	
UI.h	1
UI.cpp	1

CMPT 276 Group 20

Macrohard Corp.

TrackMaster

Architectural Design Document

Version 1

Written by:

James Isaac _____

Jin Bowen _____

Saadati Aryan _____

Sit Kwan Wai _____

Approved by:

Quality Assurance Dept. _____

Document Version History

Version 1 - Original version released in 2024/07/03 by James Isaac, Jin Bowen, Saadati Aryan, and Sit Kwan Wai

Table of Contents

Content	Page
Title Page	1
Release History	2
Release Table	3
Architectural Design Title Page	5
Document Version History	6
Table of Contents	7
Section 1: Introduction	8
Section 2: Refinement of the Analysis Models	9
Section 2.1: Subclass Considerations	
Section 3: Architecture Presentation and Discussion	10
Section 3.1: Architecture Discussion	
Section 3.2: System Scenarios	
Section 3.3: Module Function List	
Section 4: Comparison with Alternate Design	27
Section 5: Initial Interface Design	28
Section 5.1: Enumerated “Status” in Change Module	
Section 5.2: Enumerated “Priority” in Request Module	
Section 6: Detailed Interface Design	29
Section 7: Black Box Test Cases	30
Section 7.1: Normal Test Case	
Section 7.2: Stress Test Case	
Section 7.3: Performance Test Case	

1.0) Introduction

TrackMaster is a software issue tracking system that helps software companies manage their bugs and features. This document is made to provide clarity as to how this software is designed and is going to be implemented.

Keep in mind the majority of the low-level design code is not included in this document, but is a part of a separate document that is included in this module. For further information not regarding design decisions and implementation techniques please refer to the Specification Document and the User Manual document.

2.0) Refinement of the Analysis Models

2.1) Subclass Considerations

The following calculations are made to decide between implementing subclass entities with roll-up or split. In the following, we will be calculating the total space needed to implement the subclass entities ‘Requester’ and ‘Employee’ (InternalRequester) for 1 requester comparing the two methods. Readers can refer to the object relationship diagram from the Requirements Spec for details.

Roll-up		Split	
Requester + Employee:		Requester:	
Attributes	**Bytes**	**Attributes**	**Bytes**
Email	30	Email	30
Name	15	Name	15
Phone	4	Phone	4
Department	15	TOTAL	49
TOTAL	64	+	
		Employee:	
		Attributes	**Bytes**
		Email	30
		Department	15
		TOTAL	45

Assume we have 100 requesters with 40 internal requesters employees, the total storage required for each method is as follows:

Roll-up = 64*100 = 6400 bytes
Split = 49*100 + 45*40 = 6700 bytes

From the calculations, it can be seen that the roll-up implementation has the least memory cost. As it also has the fastest access, it is decided that we will use the roll-up implementation design.

3.0) Architecture Presentation and Discussion

3.1) Architecture Discussion

The design architecture will be discussed in this section. More specifically this section serves as an introduction to the list of modules and a showcase as to how they collaborate with one another.

Call-ordering will primarily be centralized, handled by the various controller modules outlined above. This will allow for a maintainable solution, as control of sequencing during a use case will be primarily handled in a single module. Additionally, the control of sequencing and state management of use cases will be cohesively encapsulated in one module.

To handle the binary fixed-length I/O raw files for each recorded object, each object will encapsulate all necessary I/O file operations. Although each object includes functions tailored to handle its specific type, they share a common set of exported operations. These operations include initializing, shutting down, locating the beginning of a file, retrieving records, and storing records.

Each entity class can be instantiated as an object in memory and includes methods for writing the object to a file. Additionally, there are methods to retrieve records from the file and return them as objects. To avoid storing a large number of objects in memory simultaneously, there are methods to iterate through the file and retrieve one record at a time.

3.1.1) Overview of Modules

Module	**Files**	**Description**
Main	main.cpp	Handles the initialization and execution of the program.
User Interface	ui.h ui.cpp	Responsible for the main event loop, rendering UI components, such as menus and sub-menus, and calling user scenarios. **Note**: does not handle all user input, some will be handled by a control module.
Scenario Controller	scenarioController.h, scenarioController.cpp	Handles core use cases, such as creating a new request or a new product.
Printer Controller	printerController.h, printerController.cpp	Handles the logic to print reports.
Life Cycle Controller	lifeCycleController.h, lifeCycleController.cpp	Handles the startup and shutdown processes of the system.
Back up	backup.h backup.cpp	Handles logic to back up data.
Product	product.h, product.cpp	Manages RAM and Disk product data and behaviour.
Product Release	productRelease.h, productRelease.cpp	Manages RAM and disk product release data and behaviour.
Request	request.h, request.cpp	Manages RAM and disk request data and behaviour.
Change Item	changeItem.h, changeItem.cpp	Manages RAM and disk change item data and behaviour.
Requester	requester.h, requester.cpp	Manages RAM and disk requester data and behaviour.

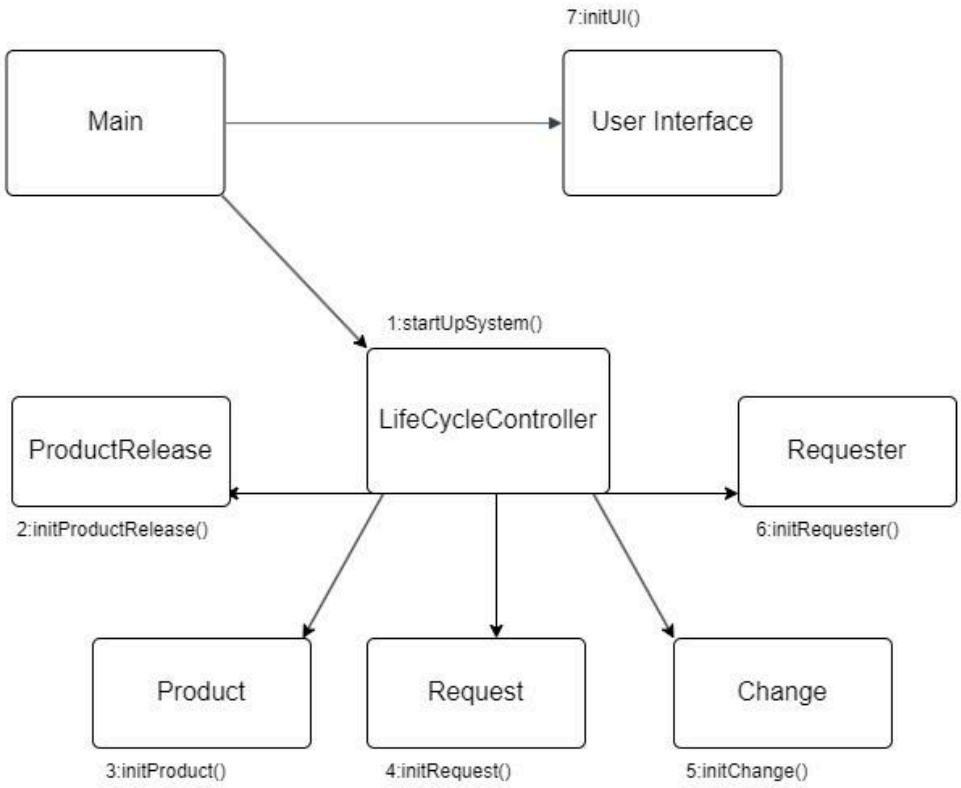
3.2) System Scenarios

3.2.1) Scenario Listing

This is a list of use case scenarios that the Trackmaster program must be capable of handling during its life cycle:

1. Startup
2. Create Request
3. Create Requester
4. Create Product
5. Assess New Change Items
6. Update Change Item
7. Inquire Change Item
8. Print Upcoming Release Changes
9. Print Completed Changes
10. Back up
11. Shutdown

3.2.2) Startup Call Trace



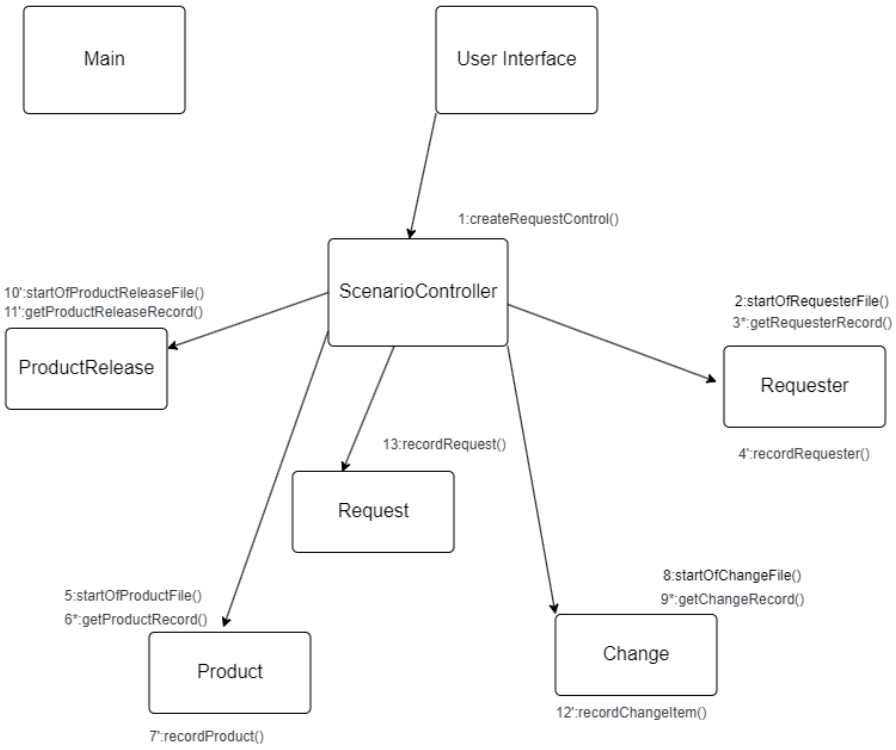
Scenario Description

Step 1: Main tells the Life Cycle Controller to start the system.

Step 2 - Step 6: Life cycle controller initializes the main program entities.

Step 7: Main initializes the UI to begin accepting user input.

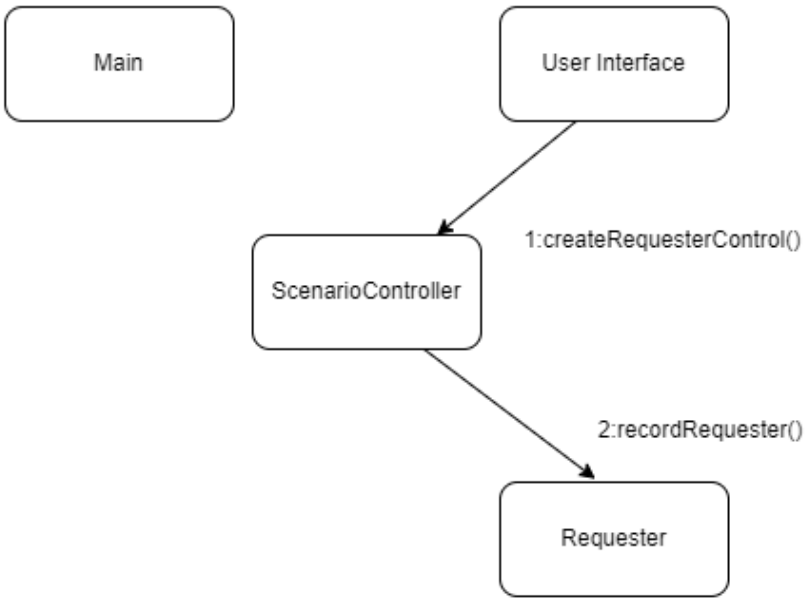
3.2.3) Create Request Call Trace



Scenario Description

Step 1: User Interface calls the Scenario Controller to start the Create Request use case.
Step 2 - Step 3: All requesters are displayed to the user to select one.
Step 4: If the relevant requester does not exist, create and record a new one.
Step 5 - Step 6: All products are displayed to the user to select one.
Step 7: If the relevant product does not exist, create and record a new one.
Step 8 - Step 9: All relevant change items are displayed to the user to select one.
Step 10 - Step 12: If no relevant change item exists, display all relevant product releases for the user to select one. Some additional information is asked of the user to create and record a new change item.
Step 13: The request is created and recorded.

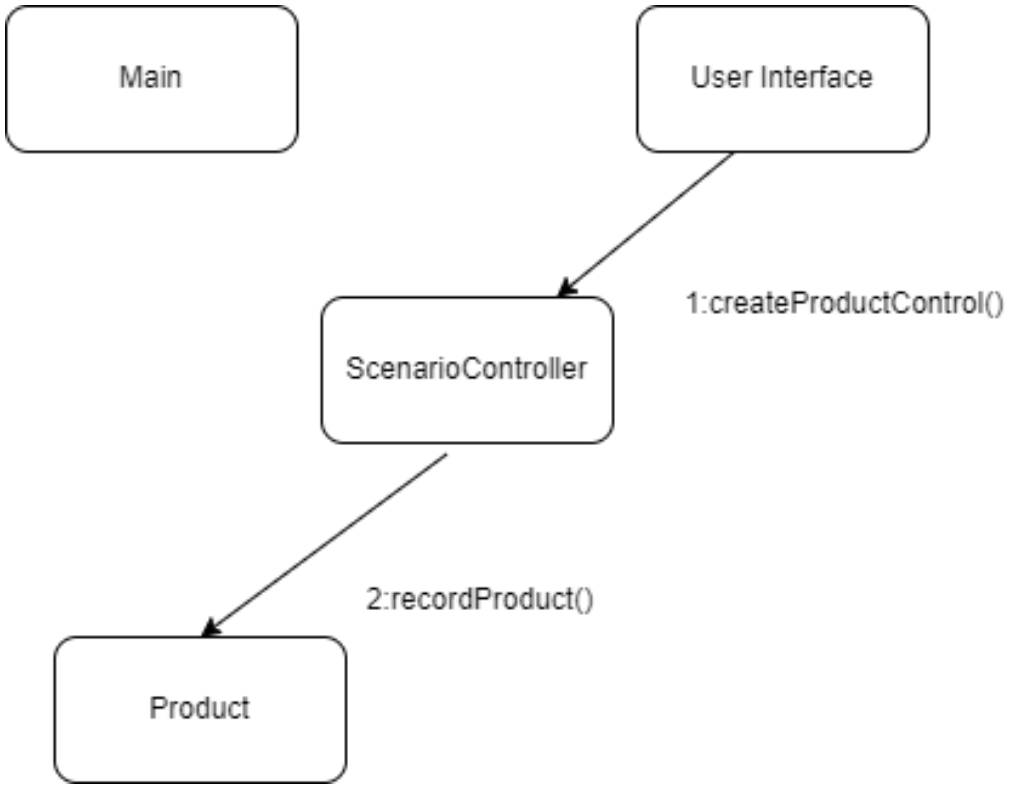
3.2.4) Create Requester Call Trace



Scenario Description

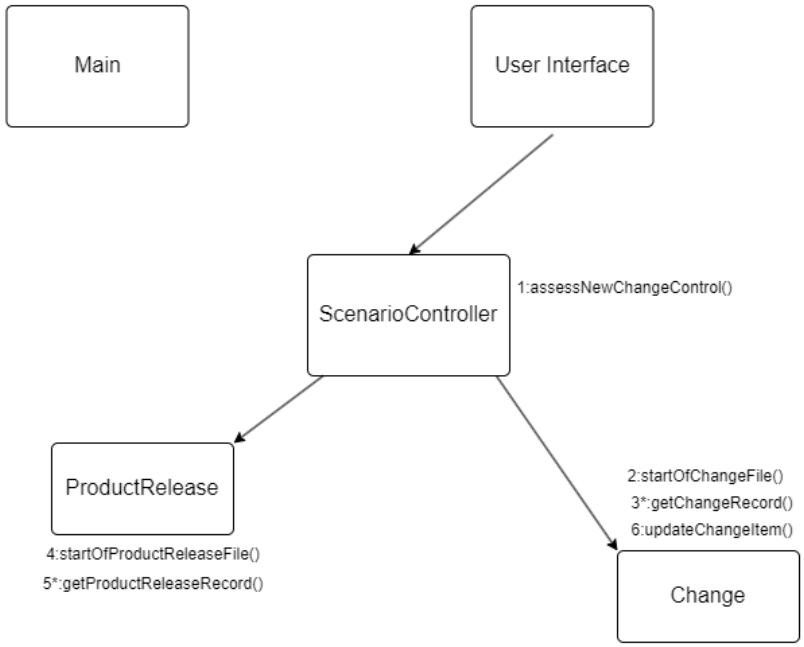
Step 1: User interface calls the scenario controller to start the create requester scenario.
Step 2: Scenario control prompts the user to input requester information, which is then used to create and record the requester.

3.2.5) Create Product Call Trace



Scenario Description
Step 1: User interface calls the scenario controller to start the create product scenario.
Step 2: Scenario control prompts the user to input product information, which is then used to create and record the product.

3.2.6) Assess New Change Items Call Trace



Scenario Description

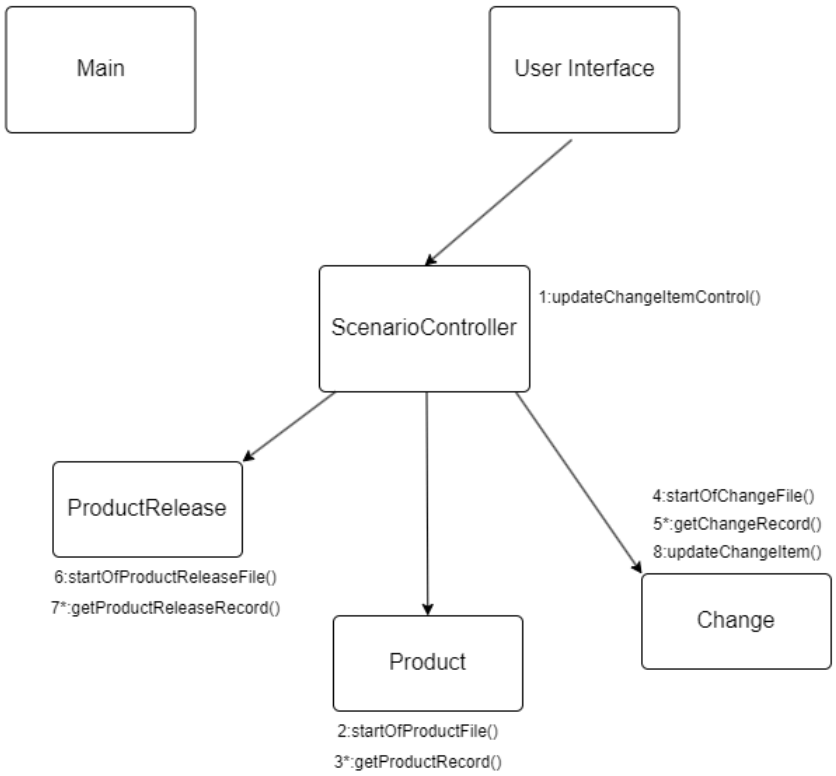
Step 1: User Interface calls the Scenario Controller to start the Assess New Change Items use case.

Step 2 - Step 3: All change items with the “open” status are displayed for the user to select one.

Step 4 - Step 5: All relevant product releases are displayed for the user to select one.

Step 6: Update the change item with new information.

3.2.7) Update Change Item Call Trace



Scenario Description

Step 1: User Interface calls the Scenario Controller to start the Update Change Item use case.

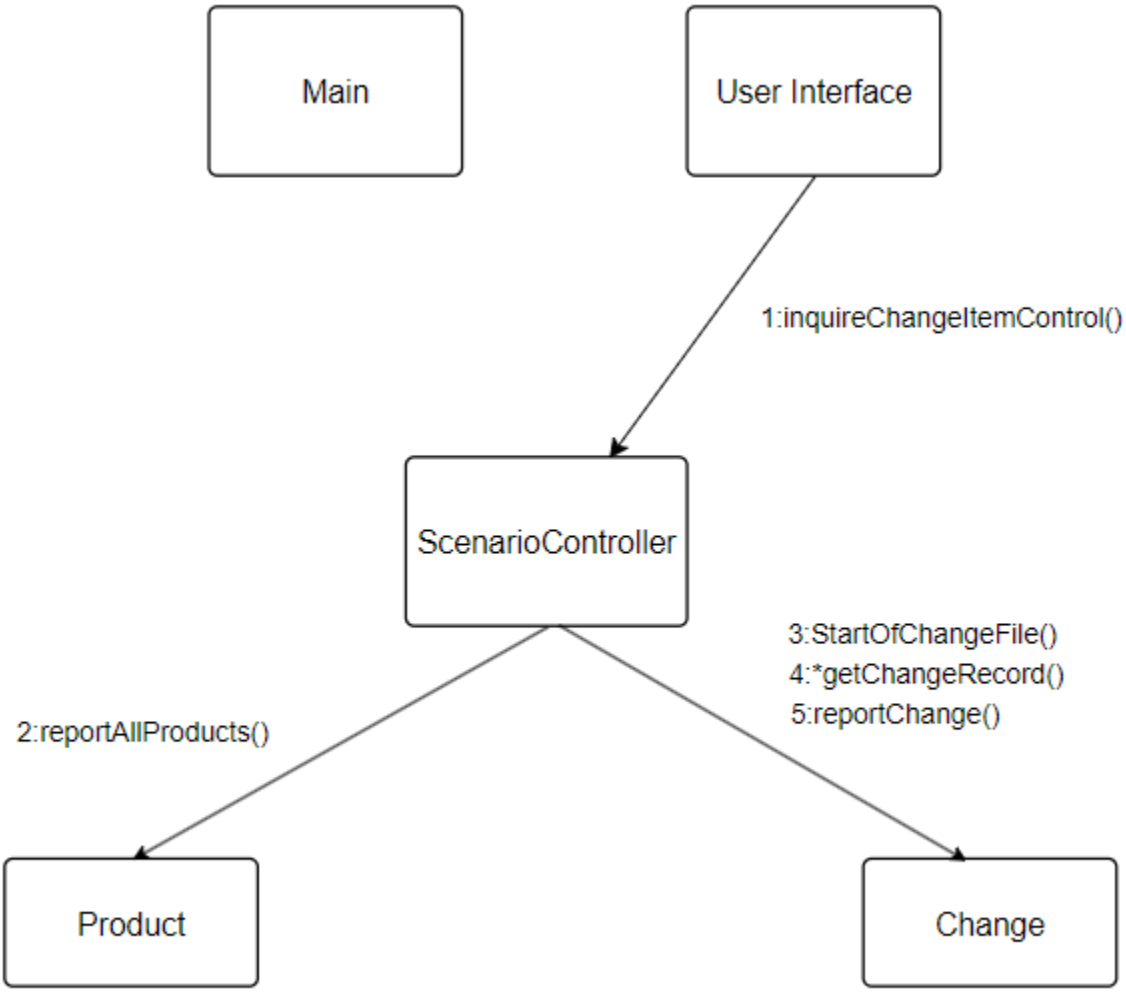
Step 2 - Step 3: All products are displayed to the user to select one.

Step 4 - Step 5: All relevant change items with the “assessed” status are displayed for the user to select one.

Step 6 - Step 7: All relevant product releases are displayed for the user to select one.

Step 8: Update the change item with new information.

3.2.8) Inquire a Change Item Call Trace



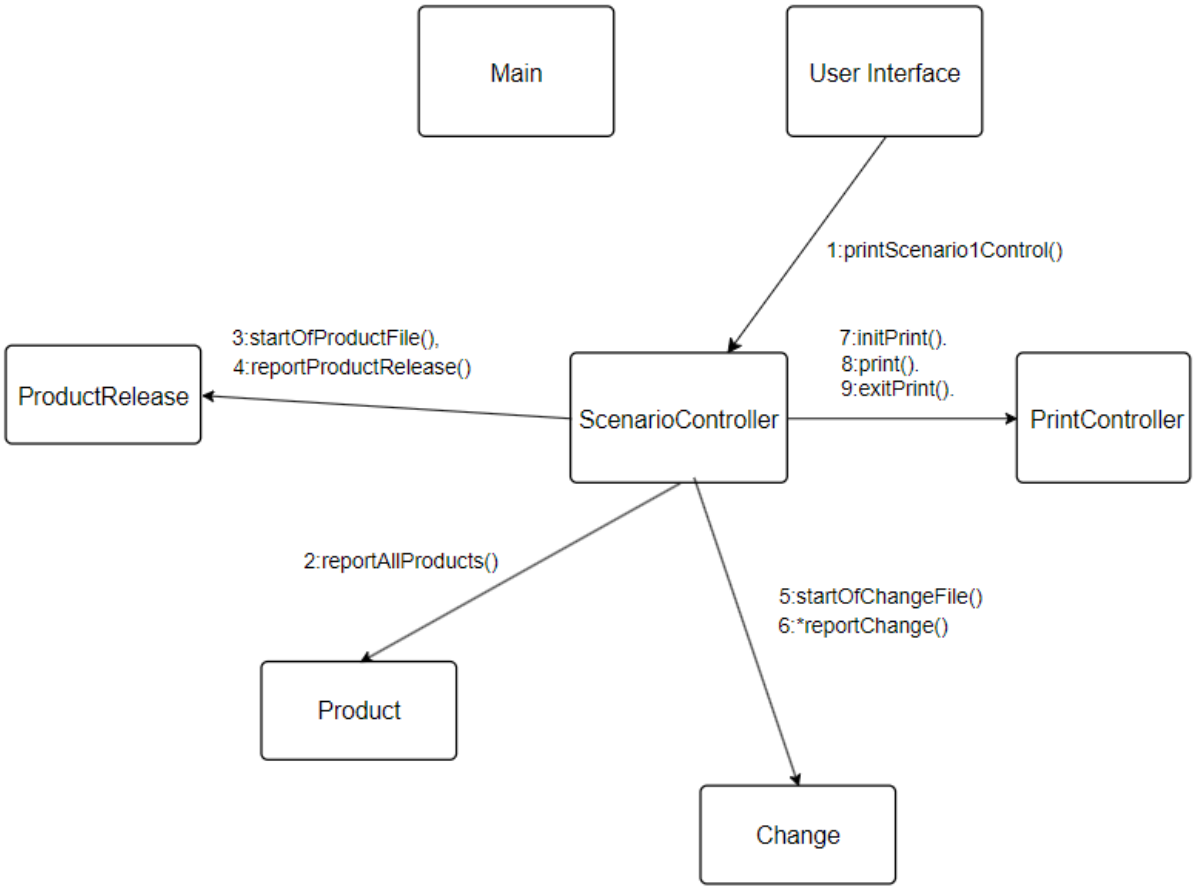
Scenario Description

Step 1: User interface calls the scenario controller to start the inquire change item scenario.

Step 2: Scenario control calls to get the list of all the products and prompts the user to pick one. .

Step 3: Scenario control searches the change data file for the appropriate change items, displays them to the user and prompts them to pick one, the change item gets displayed.

3.2.9) Print Upcoming Release Changes Call Trace



Scenario Description

Step 1: User interface calls the scenario controller to start the first print scenario.

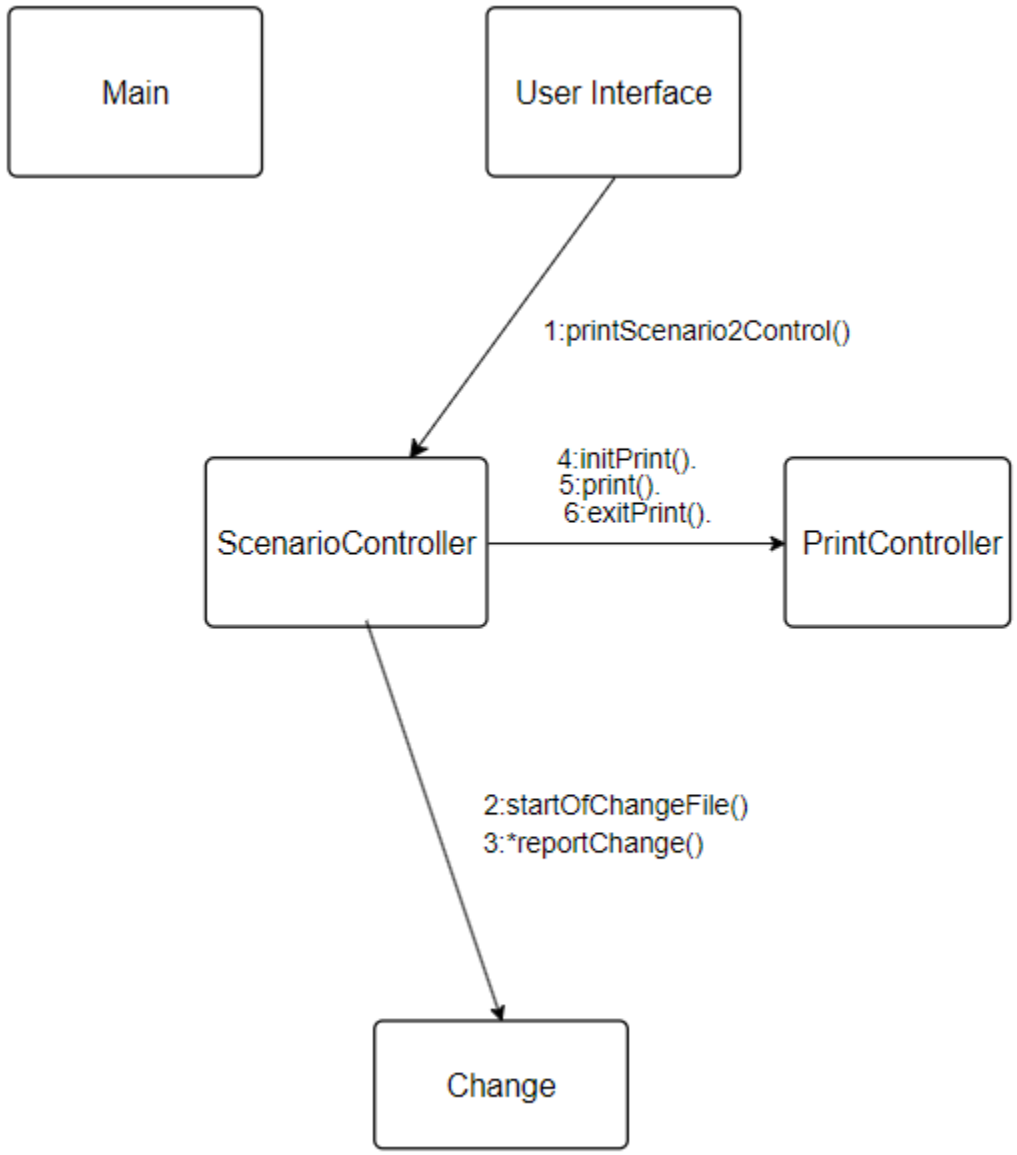
Step 2: Scenario control calls to product object to report all the products.

Step 3 - Step 4: Scenario control loops through the product release file and prompts the user to pick a release version.

Step 5 - Step 6: Scenario control uses the picked release version and finds corresponding change items.

Step 7 - Step 9: scenario control calls print on the change items.

3.2.10) Print Completed Changes Call Trace



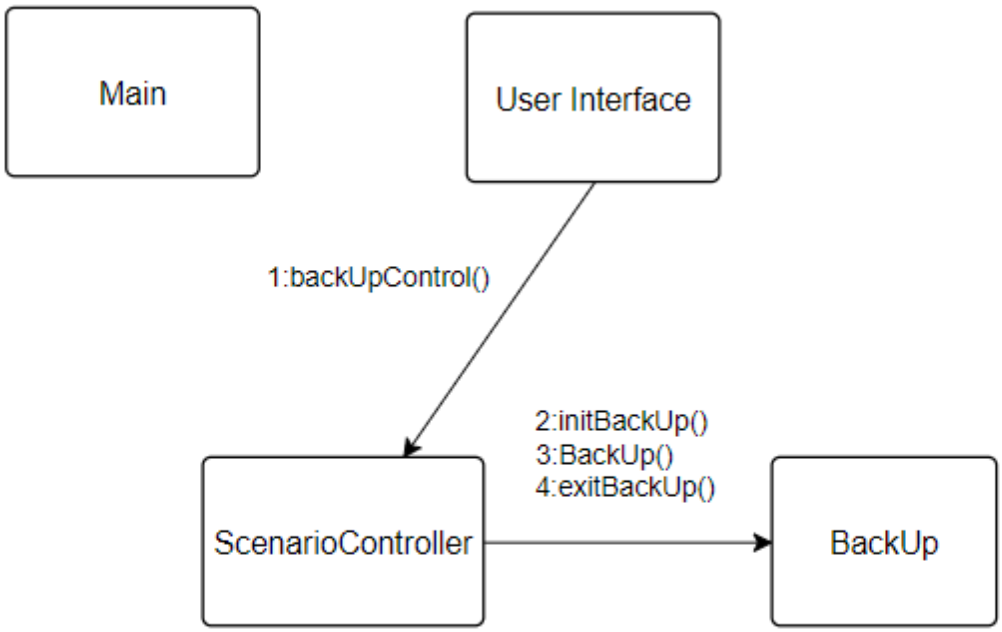
Scenario Description

Step 1: User interface calls the scenario controller to start the second print scenario.

Step 2 - Step 3: Scenario control calls to change object to report all the completed change items.

Step 4 - Step 5: scenario control calls print on the completed change items.

3.2.11) Backup Call Trace

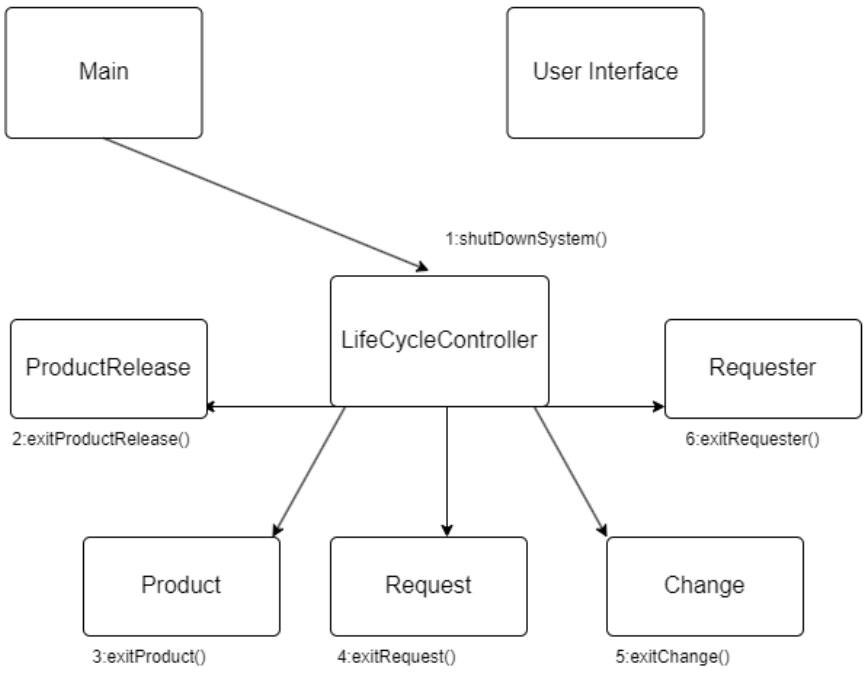


Scenario Description

Step 1: User interface calls the scenario controller to start the backup scenario.

Step 2 - Step 4: Scenario control calls the back up object.

3.2.12) Shutdown Call Trace



Scenario Description
Step 1: Main calls the Life Cycle Controller to shut down the system.
Step 2 - Step 6: Life Cycle Controller calls exit on main entities.

3.3) Module Function List

A listing of all modules and their exported functions and parameters. This list does not contain private data members or methods.

Main: N/A

Change:

- initChange()
- startOfChangeFile()
- getChangeRecord()
- recordChange()
- reportChange()
- reportAllChanges()
- exitChange()

InternalRequester:

- recordInternalRequester()
- reportAllInternalRequester()
- exitInternalRequester()
- InternalRequester()

LifeCycleController:

- startUpSystem()
- shutDownSystem()
- exitLifeCycleController()

PrintController:

- initPrintController()
- printProduct()
- printCompletedChangeItems()
- exitPrint()

Product:

- initProduct()
- startOfProductFile()
- getProductRecord()
- recordProduct()
- reportProduct()
- reportAllProducts()
- exitProduct()

ProductRelease:

- initProductRelease()
- startOfProductReleaseFile()
- getProductReleaseRecord()
- recordProductRelease()
- exitProductRelease()
- ProductRelease()

Request:

- initRequest()
- startOfRequestFile()
- getRequestRecord()
- recordRequest()
- reportRequest()
- reportAllRequests()
- exitRequest()

Requester:

- initRequester()
- startOfRequesterFile()
- getRequesterRecord()
- recordRequester()
- reportAllRequester()
- exitRequester()

ScenarioController:

- initScenarioController()
- createRequestControl()
- createRequesterControl()
- createProductControl()
- createProductReleaseControl()
- assessNewChangeControl()
- updateChangeItemControl()
- inquireChangeItemControl()
- printScenario1Control()
- printScenario2Control()
- backUpControl()

UI (User Interface):

- initUI()
- mainMenu()
- createMenu()
- updateMenu()
- printMenu()

Back Up:

- initBackUp()
- backUp()
- exitBackUp()

4) Comparison with Alternate Design

5) Initial Interface Design

5.1) Enumerated “Status” in Change Module

The Change module contains a public enum called “Status”. This is an exported enum that provides a structured representation of the different states that a change object can be in.

Since the status of a change object can only be one of the five constants (Open, Assessed, In Progress, Done, Cancelled), it prevents users from assigning arbitrary integer values to them accidentally.

Furthermore, enums provide an easier way to add more statuses or modify existing ones, by updating the enum declaration.

The enum is declared as public, allowing it to be accessible outside the class via Change instances. It cannot be declared as private as other parts of the program may interact with the status of the Change object. It also cannot be declared globally to ensure coupling with the functionalities of the Change class, as well as reducing namespace pollution.

5.2) Enumerated “Priority” in Request Module

The Request module contains a public enum called “Priority”, that defines different priority levels for Requests. In the Requirements Specification Document, the priority of a request is specified as an integer from 1 - 5, where 5 is the highest priority. The enum helps specify this in the program by assigning distinct names to these priorities:

1. LOW
2. MEDIUM
3. HIGH
4. VERY HIGH
5. CRITICAL

Since the priority of a request can only be one of the five constants, it will prevent accidentally assigning invalid values. Furthermore, due to its structured nature, it will allow for easier addition and modifications of priority levels in the future.

The enum class is encapsulated within the Request class to maintain tight cohesion between the priority and other attributes of the Request entity. It cannot be declared as global to prevent namespace pollution and to ensure it is directly associated with request instances.

6) Detailed Interface Design

The detailed interface design is not a part of this document, for further details please refer to the main and header modules included separately in this release

7.0) Black Box Test Cases

7.1) Normal Test Case

This test case is for testing if a user can inquire about a change item successfully. We will first need to create a change request, then update the change item. In the last step, inquiring about the change item will display all information related to the product and change.

Preconditions:

- Start in the main menu, after the launch of *TrackMaster*.
- The program contains previous products, requesters and change items. For test case purposes, we will use the requester Front Tussell, and the change regarding the product FreddyBear.

Start the program by clicking on the *TrackMaster* icon. You will be presented with the main menu:

```
=== MAIN MENU ===
1) Create
2) Update
3) Inquire
4) Print
5) Backup
0) Quit TrackMaster
ENTER selection [0-5]: 1
```

Step 1: ENTER **<1>** to enter the Create menu.

```
=== Create Menu ===
1) Create Request
2) Create Requester
3) Create Product
4) Create Product Release
0) Main Menu
ENTER selection [0-5]: 1
```

Step 2: Once in the create sub-menu, ENTER **<1>** to create a request.

```
Is this request coming from a customer or an employee?
ENTER a selection [c - customer/e - employee]
OR ENTER <0> to abort and exit to the main menu: c
```

Step 3: ENTER **<c>** to indicate the request coming from a customer.

```
=== Select Customer ===
  Name                  Email
1)Front Tussel          ft@sfu.ca
2)Mark Zuck             markzuck@metuh.com
3)Jeff Bozos            jeffb@amoozon.com
4)New Customer
ENTER selection [1-4] OR <0> to abort and exit to the main menu: 1
```

Step 4: ENTER **<1>** to select the customer 'Front Tussell'

```
ENTER the DATE of the request (YYYY-MM-DD) OR ENTER <0> to abort and exit
to the main menu: 2024-06-29
```

Step 5: ENTER the date of the request in the requested format..

```
=== Select Product ===
1)TakTak
2)FreddyBear
3)Cheeses
4)Rudolph
5)New Product
ENTER selection [1-5] OR <0> to abort and exit to the main menu: 2
```

Step 6: ENTER **<2>** to choose the product 'FreddyBear'.

```
=== Select Product Release ===
1) 1.0
2) 1.1
3) 1.2
4) 1.3
ENTER selection [1-4] OR <0> to abort and exit to the main menu: 4
```

Step 7: ENTER **<4>** to choose the release '1.3'.

```
=== Select a Priority ===
1) Lowest
2) Low
3) Medium
4) High
5) Highest
ENTER selection [1-5] OR <0> to abort and exit to the main menu: 5
```

Step 8: ENTER **<5>** to choose the priority 'Highest'.

```
=== Select Change Item ===
   Description                    ChangeID
1) Freddy's voice is squeaky      1234
2) New Change
ENTER selection [1-2] OR <0> to abort and exit to the main menu: 2
```

Step 9: ENTER **<2>** to create a new change.

```
ENTER a description for the new change (30 char. max): Freddy bites too
much.
```

Step 10: ENTER the description in the requested format.

```
Select a product release that the change is anticipated to be completed
for.
=== Select Product Release ===
1) 1.0
2) 1.1
3) 1.2
4) 1.3
5) Skip
ENTER selection [1-4] OR <0> to abort and exit to the main menu: 4
```

Step 11: ENTER **<4>** to choose the release '1.3'.

```
=== New Request Information===
Requester (customer): Front Tussell - ft@sfu.ca
Product: FreddyBear
Release: 1.3
Priority: 5 - Highest
Change ID: 1234
ENTER <1> to confirm OR <0> to abort and exit to main menu: 1
```

Step 12: ENTER **<1>** to confirm, and then you will be redirected to the main menu.

```
=== MAIN MENU ===
1) Create
2) Update
3) Inquire
4) Print
5) Backup
0) Quit TrackMaster
ENTER selection [0-5]: 2
```

Step 13: ENTER **<2>** to enter the Update menu.

```
=== Update Menu ===
1) Assess new change items
2) Update a change item
0) Main Menu
ENTER selection [0-2]: 1
```

Step 14: ENTER **<1>** to assess new change items.

```
=== Select Change Item ===
   Product        Description                  Status
1) FreddyBear     Freddy's voice is squeaky    Open
2) FreddyBear     Freddy Bites too much        Open
ENTER selection [1-2] OR ENTER <0> to abort and exit to main menu: 2
```

Step 15: ENTER **<2>** to select the recently added change item.

```
=== Select Status ===
1) Assessed
2) Canceled
ENTER selection [1-2] OR ENTER <0> to abort and exit to main menu: 1
```

Step 16: ENTER **<1>** to select 'Assessed'.

```
ENTER a new description for the change [max 30 characters, leave blank to
skip] OR <0> to abort and exit to main menu:
```

Step 17: Leave the field blank to indicate no changed description and ENTER.

```
Select a product release that the change is anticipated to be completed
for.
=== Select Product Release ===
1) 1.0
2) 1.1
3) 1.2
4) 1.3
5) Skip
ENTER Selection [1-4] OR <0> to abort and exit to main menu: 4
```

Step 18: ENTER **<4>** to choose the release '1.3'.

```
=== Assessed Change Item Information ===
Product: FreddyBear
Description: Freddy bites too much
Anticipated release: 1.3
Status: Assessed
Change ID: 1246
ENTER <1> character to confirm OR 0 to abort and exit to main menu: 1
```

Step 19: ENTER **<1>** to confirm the assessed change item, and then you will be redirected to the main menu.

```
=== MAIN MENU ===
1) Create
2) Update
3) Inquire
4) Print
5) Backup
0) Quit TrackMaster
ENTER selection [0-5]: 2
```

Step 20: ENTER **<2>** to enter the Update menu.

```
=== Update Menu ===
1) Assess new change items
2) Update a change item
0) Main Menu
ENTER selection [0-2]: 2
```

Step 21: ENTER **<2>** to update a change item.

```
=== Select Product ===
1) TakTak
2) FreddyBear
3) Cheeses
4) Rudolph
5) New Product
ENTER selection [1-5] OR <0> to abort and exit to the main menu: 2
```

Step 22: ENTER **<2>** to choose 'FreddyBear'.

```
=== Select Change Item ===
  Description                    Status            ChangeID
1)Freddy's voice is squeaky      Assessed          1235
2)Freddy Bites too much          Assessed          1246
ENTER selection [1-2] OR <0> to abort and exit to main menu: 2
```

Step 23: ENTER **<2>** to select the recently added change item.

```
=== Select Status ===
1) In Progress
2) Done
3) Canceled
4) Skip
ENTER selection [1-4] OR <0> to abort and exit to main menu: 1
```

Step 24: ENTER **<1>** to choose 'In Progress'.

```
ENTER a new description for the change (leave blank to skip) OR <0> to
abort and exit to main menu [max 30 characters]:
```

Step 25: Leave the field blank to indicate no changed description and ENTER.

```
Select a product release that the change is anticipated to be completed
for.
=== Select Product Release ===
1) 1.0
2) 1.1
3) 1.2
4) 1.3
5) Skip
ENTER Selection [1-4] OR <0> to abort and exit to main menu: 4
```

Step 26: ENTER **<4>** to choose the release '1.3'.

```
=== Updated Change Information ===
Product: FreddyBear
Description: Freddy bites too much
Anticipated release: 1.3
Status: In Progress
Change ID: 1246
ENTER <1> character to confirm OR 0 to abort and exit to main menu: 1
```

Step 27: ENTER **<1>** to confirm the updated change item, and then you will be redirected to the main menu.

```
=== MAIN MENU ===
1) Create
2) Update
3) Inquire
4) Print
5) Backup
0) Quit TrackMaster
ENTER selection [0-5]: 3
```

Step 28: ENTER **<3>** to enter the Inquire menu.

```
=== Inquire Menu ===
Select the product you wish to inquire a change item from:
1) TakTak
2) FreddyBear
3) Cheeses
4) Rudolph
0) Exit
ENTER Selection [0-4]: 3
```

Step 29: ENTER **<2>** to select ‘FreddyBear’.

```
Select a change item to inquire
=== Select Change Item ===
  Description                    Status            ChangeID
1)Freddy’s voice is squeaky      Assessed          1235
2)Freddy Bites too much          In Progress       1246
ENTER selection [1-2] OR <0> to abort and exit to main menu: 2
```

Step 30: ENTER **<2>** to inquire about the recently created and updated change item.

```
Change Item Report:

Product name: FreddyBear
ChangeID: 1246
First Reported: 2024-06-29
Statues: In Progress
Priority: 1
Anticipated release: 1.3
Description: Freddy Bites too much

Enter <1> to inquire about another change item.
Enter <0> to and go back to the main menu.
ENTER Selection: 0
```

Expected results: If the change item is created and updated successfully, the above screen should appear. This also reflects the success of inquiring about a change item.

Step 31: ENTER **<0>** to go back to the Main Menu.

7.2) Stress Test Case

This test case tests *TrackMaster* for using invalid input.

Preconditions:

- Start in the main menu, after the launch of *TrackMaster.*
- The program does not contain any previous data.

Start the program by clicking on the *TrackMaster* icon. You will be presented with the main menu:

```
=== MAIN MENU ===
1) Create
2) Update
3) Inquire
4) Print
5) Backup
0) Quit TrackMaster
ENTER selection [0-5]: 1
```

Step 1: ENTER **<1>** to enter the Create menu.

```
=== Create Menu ===
1) Create Request
2) Create Requester
3) Create Product
4) Create Product Release
0) Main Menu
ENTER selection [0-5]: 2
```

Step 2: Once in the create sub-menu, ENTER **<2>** to create a requester.

```
ENTER the EMAIL ADDRESS of the requester (Length: max 30)
OR ENTER <0> to abort and go back to the main menu: ExampleEmail@Email.ca
```

Step 3: ENTER an appropriate email for the requester as shown above.

```
ENTER the PHONE NUMBER of the requester (Length: Exactly 10)
OR ENTER <0> to abort and exit to the main menu: 911
```

Step 4: ENTER the phone number '911' for the requester.

```
Error: Input is invalid. Re-enter input.
Enter 0 to abort and return to the main menu.
```

Expected results: The above input error will be shown, as the phone number length must be exactly 10.

Step 5: ENTER **<0>** to go back to the main menu.

7.3) Performance Test Case

This test case tests *TrackMaster* for its response time for inquiring on a change item.

Preconditions:

- Start in the main menu, after the launch of *TrackMaster.*
- The program contains previous products, requesters and change items. For test case purposes, we will use the requester Front Tussell, the change regarding the product FreddyBear and change item with description ‘Freddy’s voice is squeaky.’
- A stopwatch is required to measure response time.
- There are 20 other change items in the database.

Start the program by clicking on the *TrackMaster* icon. You will be presented with the main menu:

```
=== MAIN MENU ===
1) Create
2) Update
3) Inquire
4) Print
5) Backup
0) Quit TrackMaster
ENTER selection [0-5]: 3
```

Step 1: ENTER **<3>** to enter the Inquire menu.

```
=== Inquire Menu ===
Select the product you wish to inquire a change item from:
1) TakTak
2) FreddyBear
3) Cheeses
4) Rudolph
0) Exit
ENTER Selection [0-4]: 2
```

Step 2: ENTER **<2>** to choose ‘FreddyBear’.

```
=== Select Change Item ===
  Description                    Status            ChangeID
1)Freddy's voice is squeaky      Assessed          1235
ENTER selection [1] OR <0> to abort and exit to main menu: 1
```

Step 3: ENTER **<1>** to select the change item. Press ‘start’ on the stopwatch at the same time.

```
Change Item Report:

Product name: FreddyBear
ChangeID: 1235
First Reported: 2024-06-29
Statues: Assessed
Priority: 1
Anticipated release: 1.3
Description: Freddy Bites too much

Enter <1> to inquire about another change item.
Enter <0> to and go back to the main menu.
ENTER Selection: 0
```

Step 4: Press ‘stop’ on the stop watch the moment it is displayed. This will give the response time of searching for the change item.

Expected results: The response time should be instantaneous.