

CMPT 276 Group 20

Macrohard Corp.
TrackMaster

Release 4

Release Date: 2024/07/17

Approved By:

Saadati Aryan, *Project Manager* _____

Bowen Jin, *CEO* _____

Sit Kwan Wai, *CFO* _____

James Isaac, *Chief of Staff* _____

Release History Page

- | | |
|---------------|--|
| Rel. 1 | <ul style="list-style-type: none">- 2024/06/07- Was composed of just the preliminary Requirements Spec. |
| Rel. 2 | <ul style="list-style-type: none">- 2024/06/19- Initial release of the user manual, providing comprehensive guidance on <i>TrackMaster's</i> features and functionalities |
| Rel. 3 | <ul style="list-style-type: none">- 2024/07/03- Initial release of the architectural design document as well as modules headers and source code. |
| Rel. 4 | <ul style="list-style-type: none">- 2024/07/17- Initial release of the detailed design document as well as modules implementation code and unit test cases. |

Release Table Page

Configuration Item:	Version	Number of Each Configuration Item in:			
	Rel. 1	Rel. 2	Rel. 3	Rel. 4	
<u>Documents</u>					
Requirements Spec	1	1	1	1	
User Manual		1	1	1	
Architectural Design Document			1	1	
Detailed Design Document				1	
Integration Report					
<u>Modules</u>					
Main.cpp			1	1	
Backup.h				1	
Backup.cpp				1	
Change.h			1	1	
Change.cpp				1	
FileNotOpenException.h				1	
FileNotOpenException.cpp				1	
FileOpenFailedException.h				1	
FileOpenFailedException.cpp				1	
InternalRequester.h			1		
InternalRequester.cpp					
LifeCycleController.h			1	1	
LifeCycleController.cpp				1	

RecordNotFoundException.h		1
RecordNotFoundException.cpp		1
Request.h	1	1
Request.cpp		1
Requester.h	1	1
Requester.cpp		1
PrintController.h	1	1
PrintController.cpp		1
Product.h	1	1
Product.cpp		1
ProductRelease.h	1	1
ProductRelease.cpp		1
ScenarioController.h	1	1
ScenarioController.cpp		1
testFileOps.cpp		1
UI.h	1	1
UI.cpp	1	1
unitTestMain.cpp	1	1

Macrohard Corp.
TrackMaster

Detailed 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 2024/07/17 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
Detailed Design Title Page	5
Document Version History	6
Table of Contents	7
Section 1: Discussion of Broad Projects	8
Section 2: Guide to Files	9
Section 3: Introduction to Unit Test Cases	10
Section 3.1: Functional Unit Test	
Section 3.2: Low Level Module Unit Test	

1.0) Discussion of Broad Projects

The decision to utilize a fixed length binary record table as the primary and sole database for *TrackMaster* presents several advantages throughout the project-wide implementation. This ensures data integrity since the structure of the records remains consistent across the project. Furthermore, it allows for more predictable space calculation and simplifies record retrieval, as the size of each object instance is uniform. Additionally, this method grants a more efficient data retrieval within a file through fast random access, favouring direct position calculations.

However, this approach also contains trade-offs. One significant drawback is limited flexibility in accommodating changes or additions to the data structure, as altering the fixed length table can be complex. Moreover, fixed length binary records will also require careful management of data structures and file pointers.

The records in the fixed binary files are stored in an unsorted linear set-up. This will allow for simplicity in storage and implementation, as files can be appended or updated without needing to maintain a specific order. This approach often allows for faster write operations due to its lack of need for rearranging existing data to maintain proper order. However, the lack of sorting can lead to slower read times when searching for specific files, as each file may need to be sequentially scanned via linear search. Our team at *Macrohard* has chosen this approach based on our anticipated data volume. For further details on our choice behind designing and implementing *TrackMaster*, please consult our Architectural Design Document.

2.0) Guide to Files

The individual files are not included in this document but rather attached to the end of this document as a list of alphabetically ordered sets of header files followed by their respective cpp implementation files, excluding the main.cpp file which is included at the top.

Each header file includes a comment block at the top that includes the revision history of the document and information on the classes housed within the module. Followed by each comment block there exists a class implementation which includes all of its method and function headers. Each of those headers are wrapped in comment blocks briefly explaining the purpose, the set of parameters, if any, and the return value. At the end of each class implementation you can find the set of private data members that exist within each class. For more information on the individual data members please refer to the Requirement Spec document.

Each cpp file includes all the function implementations with respect to the header files. Above each implementation there exists a comment block describing the purpose of the function/method, however the information is rather brief since a more detailed description of each exists in the header files. Within each implementation you will find lines of code comment in places where they were deemed necessary to include as a guide for the reader.

3.0) Introduction to Unit Test Cases

3.1) Functional Unit Test

This test case (unitTestMain.cpp) is for testing the functions of file: ScenarioController.h, the test case file can be found in alphabetical order after the attached main.cpp. 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:

- The program contains previous products.

Procedures:

Step 1: Create a change request according to section 4.1 in the User Manual.

Step 2: Update the change item according to section 4.5 in the User Manual.

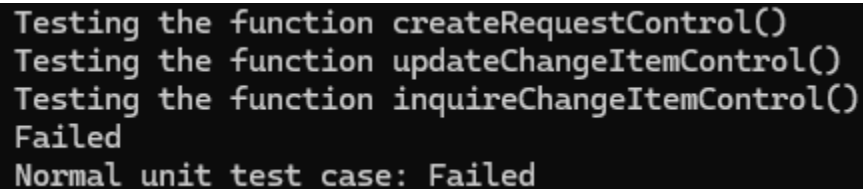
Step 3: Inquire on the change item according to section 4.6 in the User Manual.

Expected Results:

The program guides you through the processes of creating a request, updating change item, and inquiring about change item. The display in the inquiry process should also show the created and updated change item.

Actual Results:

As the program is currently not linked, the unit test case has failed, as verified by the user.

Output: The screenshot shows a terminal window with the following text: "Testing the function createRequestControl()", "Testing the function updateChangeItemControl()", "Testing the function inquireChangeItemControl()", "Failed", and "Normal unit test case: Failed".

3.2) Low Level Module Unit Test

This test case (testFileOps.cpp) is for testing the read and write functions of file: Requester.h, the test case file can be found in alphabetical order after the attached main.cpp. In the test, 3 requesters with errorless attributes information will be inserted, then 3 requesters objects will be read, and they will be compared for consistency.

Preconditions:

- An empty data file “Requesters.bin” exists in the same directory.

Procedures:

(Unlike previous test case, running the program will already automatically execute all steps)

Step 1: Initialize controller.

Step 2: Insert 3 Requester object files with errorless attributes information into data file.

Step 3: Go to the start of the file.

Step 4: Read 3 Requester object files from data file.

Expected Results:

An output of “File write and read back test: Passed”

Actual Results:

An output of “File write and read back test: Failed”

Output:

```
File write and read back test: Failed
```