Testing Report

F.1 Introduction

The aim of testing is to program the software as we planned and potentially prevent it from being broken. In general, testing is about ensuring quality. Test-driven development (TDD) is an overall approach that writes tests before implementing functionalities. This approach has been considered as our primary methodology to ensure the quality of the software. There are four major testing phases in this project, unit testing, integration testing, release testing and acceptance testing. Unit testing is responsible for individual pieces, ensuring the basic functionality of each component. Integration testing checks whether combinations of components work as expected. Release testing tests most of the possible interactions to ensure the stability of the whole system. Acceptance testing checks requirements and specifications by the customer, indicating whether the software is accepted. In addition to these testing phases, we apply continuous integration through the whole process of development for spotting errors early and improving efficiency. The following parts will introduce how these testing methods were deployed during the development and problems we met with respective remarks.

F.2 Unit Testing

As TDD instructs, developers in the team wrote unit tests for most basic software components before coding any actual functionalities. Therefore, unit tests work as the base of the whole project. By doing unit testing, the team has a clearer view of what features a component is expected to achieve. This can be illustrated by an example of a group in the team. At the early stage, TDD was not taken seriously by some of the team members. One group of two in the team did not follow the instruction of TDD and wrote code directly without writing any unit test. The component displays appropriately at first, but coding is painstaking as no clear plan was made—the group of two modified their design multiple times. After TDD was stressed to be vital, the group added unit tests for the component but found a title in it was wrong. Compared to human eyes and testing manually, automated unit testing helps design the code and prevents potential mistakes by checking components each time they are modified.

Specifically, unit testing in the project works for checking whether a fundamental component contains expected texts, buttons and testing whether functions inside a component run appropriately. Since we use React as the JavaScript library, Jest is the project's primary unit testing framework. React-testing-library is a testing utility that encourages good testing practices and simplifies testing processes such as rendering components and creating snapshots. It is possible to test a combination of several components as well. As fundamental components are already tested, mocking is utilised in testing combinations to avoid repeated tests. Tested basic components and third-party components will be mocked to avoid unnecessary rendering and unexpected errors.

All of the unit tests were firstly planned by documenting test plans in detail. Any failed case and modification were also recorded in a test log for future bug track convenience. Detailed test plans and logs can be viewed in Appendix G.

Examples of unit testing points are as follows:

1. Should contain specific text.
2. Should contain buttons.
3. A function should have been called after a button click event.
4. A subcomponent should have been called while rendering.
5. Functions should work as expected.

F.2.1 Test Plan for Unit Testing

Please refer to Appendix G.

F.2.2 Test Log

Please refer to Appendix G.

F.3 Integration Testing

Integration testing tests subsystems. In this project, scenes and huge combinations of multiple components are considered subsystems. Their interfaces were tested by jest snapshot, and their interactions were tested manually. Snapshot testing is a helpful tool to ensure a subsystem has not been modified. If any of the elements were changed by accident, the snapshot test would fail by comparing it to the old one. Integration testing was often conducted at the end of a sprint and may expose some bugs related to interaction. This is relatively helpful to check whether a subsystem works as the specification expected.

Examples of Integration testing are as follows:

1. Snapshot created and match with old one.
2. Test interactions manually in a subsystem to check them meet the requirement.

F.3.1 Test Plan for Integration Testing

Please refer to Appendix G.

F.3.2 Test Log for Integration Testing

Please refer to Appendix G.

F.4 Release Testing

Release testing is expected to be conducted by an individual quality assurance team that has not been involved in the system development. However, due to the team's small size, all the team members have done something related to the system. In this case, two members who focus more on user interface would take responsibility for release testing. They tested the software as a whole system manually to check whether the system achieves all the specifications and works without abnormal. Specifically, they took actions

to simulate the user stories we defined. Non-functional specifications were tested as well. Once it has been done, the software is ready for acceptance testing and external use.

Three strategies taken are as follows:

1. Performance driven.
2. Specification driven.
3. Scenario driven.

F.4.1 Test Log

Test for I Can Sort

Release testing time: 2021/3/26

Part 1. First open

WHEN FIRST OPEN THIS SOFTWARE, WHETHER IT WILL OPEN A DIALOG WINDOW TO ASK THE USER'S LEVEL.	**1**
In the dialog window, click the left button, whether it will jump to Tutorial module.	1
In the dialog window, click the right button, whether it will jump to Procedure module.	1

Part 2. Modules

IN ALL MODULES, CLICK THE SETTINGS BUTTON, WHETHER A SETTINGS WINDOW WILL POP UP.	**T**
In all modules, whether the progress bars will show the progress of the corresponding parts correctly.	T
In all modules, whether the recently accessed part will be stroked.	T
In the Settings button, when click GITHUB ADDRESS: I-CAN-SORT button, it will open a new browser window to the software's GitHub page.	T
In the Settings button, when click CONFIRM button, it will clear all history.	T
In all modules, click the Help button, whether a help window will pop up.	T
In Tutorial, click the Swap button, whether it goes to Swap page.	T
In Tutorial, click the Loop button, whether it goes to Loop page.	T
In Tutorial, click the Terminology button, whether it goes to Terminology page.	T
In Procedure, click the Bubble sort button, whether it goes to Bubble sort page.	T
In Procedure, click the Selection sort button, whether it goes to Selection sort page.	T
In Procedure, click the Insertion sort button, whether it goes to Insertion sort page.	T
In Procedure, click the Quick sort button, whether it goes to Quick sort page.	T

IN ALL MODULES, CLICK THE SETTINGS BUTTON, WHETHER A SETTINGS WINDOW WILL POP UP.	**T**
In Procedure, click the Merge sort button, whether it goes to Merge sort page.	T
In Procedure, click the Heap sort button, whether it goes to Heap sort page.	T
In Correctness, click the Tutorial button, whether it goes to Tutorial page.	T
In Correctness, click the Proof button, whether it goes to Proof page.	T

Part 3. Pages

IN ALL PAGES, CLICK THE HOME BUTTON, WHETHER IT GOES TO THE UPPER LEVEL MODULE CORRECTLY.	**1**
In Swap page, click the corresponding control buttons, whether the animation will act as it is told to do.	1
In Swap page, when the animation is playing, whether the code is highlighted correctly.	1
In Swap page, when the animation is playing, whether the values of variables show correctly.	1
In Loop page, when first get into this page, whether a dialog window pops up.	1
In Loop page, click the question marks, whether a dialog window pops up.	1
In Loop page, click the corresponding control buttons, whether the animation will act as it is told to do.	1
In Loop page, when the animation is playing, whether the code is highlighted correctly.	1
In Terminology page, when first get into this page, whether a dialog window pops up.	1
In Terminology page, when click left side catalogue buttons, whether it goes to corresponding catalogue page correctly.	1
In Terminology page, when click back and next buttons, whether the pages changes correctly.	1
In all subpages of Procedure, the Introduction part and the algorithm shows correctly.	1
In all subpages of Procedure, the shuffle button of Operation and Implementation part generates a random array correctly.	1
In all subpages of Procedure, the Export Quick Guide button exports a PDF file correctly.	1
In Tutorial page of Correctness, when click left side catalogue buttons, whether it goes to corresponding catalogue page correctly.	1
In all Tutorial pages of Correctness, when click back and next buttons, whether the pages changes correctly.	1
In Input page of Tutorial, when click the shuffle button, whether it will generate correct random inputs.	1
In Input page of Tutorial, when click the play button, whether it will put correct random inputs into the algorithm.	1
In Example page of Tutorial, when click any algorithm, it will run correctly.	1

F.5 Acceptance Testing

Acceptance testing is the test after release testing and is done with the clients. It aims to check whether the software meets stakeholders' expectations and receive feedbacks from them. Team 10 prepared checklist using similar strategy as release testing's and invited 6 stakeholders from year 2 and year 4 and Dr Heshan to do the acceptance testing. Team 10 showed the software to them and asked them to try the software freely. After trying the software, we discussed the problems with stakeholders and took notes of their comments and feedbacks.

F.5.1 Feedback from Stakeholders

During the acceptance test on 27th March, the stakeholders gave Team 10 valuable suggestions. Some of them are listed as follows.

1. In the Procedure module, most stakeholders think animation speed is too fast for stakeholders to understand, while some experienced stakeholders believe it is slow.
2. In the Implementation part of the Procedure module, some stakeholders find the quick guides (PDF files) helpful and would love to take a detailed look at them, while one user thinks it is unnecessary and complicated.
3. In the Correctness part, some stakeholders feel confused about what the software is doing. Most stakeholders with no algorithm correctness experience can not understand, especially the relationship among assertion, termination and correctness.
4. A user would like to use the keyboard to control the software. For example, press Enter to go to the next page or press Esc to close the pop-up window.
5. Most stakeholders do not prefer long sentences and paragraphs.
6. Most stakeholders would like more guides on how to use the software.
7. One thinks the user interface is plain and not so colourful.

Team 10 conducted a acceptance survey for further justification. In order to confirm with our supervisor, a acceptance checklist was also agreed to be rubrics. Please refer to Survey Result in Appendix F.5.3 to see the complete questionnaire. The acceptance checklist is provided in Appendix H. Here take some representative data to illustrate.

In the result of the survey, no stakeholder finds the software very difficult to use, but one says he/she may require a user guide. For the modules' user experience, the Tutorial and Procedure modules are rated 4.5 out of 5, while the Correctness module is only rated 3.8. Team 10 analyses the reason for this to be, most of the stakeholders have no experience in algorithm correctness. So Team 10 have to make it much clearer and emphasise the definitions and relationships of the professional terms.

Team 10 agrees on most of the suggestions and made improvement based on them. After the improvements below were done, an acceptance testing tested by Dr Heshan was passed. A corresponding acceptance checklist was signed, in which our supervisor said she was satisfied and accept the software to be released. The software was then released on the team's GitHub page officially.

F.5.2 Improvement

1. **First In page:** Extra information in explaining our software's purpose is added.
2. **Tutorial Swap:** The Swap page's introduction has been modified. Now it is clearly instructed to press the play button to learn swap.
3. **Procedure Operation:** Pressing Enter to input an array is now realizable. Now users can use both comma and white space to split two numbers when inputting.
4. **Procedure Implementation:** An "in increasing order" option is added to clarify the sorting order.
5. **Procedure Implementation:** Merge and Heap's pseudocode comments are added, informing users of checking the detail of merge and maxHeapify in quick guides.
6. **Animation:** The speed of animation is carefully adjusted to satisfy inexperienced users.
7. **Correctness Termination:** The animation for the terminating example is now changed. Now the two algorithms are both swap algorithms.
8. **More pop-up windows:** More pop-up windows are added in the First In page to inform the users how to utilize modules.
9. **Correctness Proof:** Extra information about the link between proof and assertions are added. It now stresses that this is an aid tool that is not theoretical enough.
10. **Selection sort:** One non-clear assertion is rewritten.
11. **Quicksort:** Quicksort is now presented recursively, being made consistent with merge and heap.

F.5.3 Survey Result

SURVEY STATISTICS (1 FOR DISAGREE, 5 FOR AGREE)		
1. I found the software unnecessarily complex		
Options	Percentage%	Subtotal
1	42.9%	3
2	57.1%	4
3	0.0%	0
4	0.0%	0
5	0.0%	0
Effective amount		7

2. I think the software is easy to use

SURVEY STATISTICS (1 FOR DISAGREE, 5 FOR AGREE)		
Options	Percentage%	Subtotal
1	0.0%	0
2	0.0%	0
3	14.3%	1
4	42.9%	3
5	42.9%	3
Effective amount		7

3. I need a user manual to use this software		
Options	Percentage%	Subtotal
1	28.6%	2
2	28.6%	2
3	0.0%	0
4	42.9%	3
5	0.0%	0
Effective amount		7

4. I understand the meanings of buttons		
Options	Percentage%	Subtotal
1	0.0%	0
2	0.0%	0
3	28.6%	2
4	14.3%	1
5	57.1%	4
Effective amount		7

5. I think the UI is consistent in this software		
Options	Percentage%	Subtotal
1	0.0%	0
2	0.0%	0
3	0.0%	0
4	14.3%	1
5	85.7%	6
Effective amount		7

6. I found the various functions in this software were well integrated		
Options	Percentage%	Subtotal
1	0.0%	0

SURVEY STATISTICS (1 FOR DISAGREE, 5 FOR AGREE)		
2	0.0%	0
3	0.0%	0
4	42.9%	3
5	57.1%	4
Effective amount		7
7. I think that I would like to use this software frequently		
Options	Percentage%	Subtotal
1	0.0%	0
2	14.3%	1
3	28.6%	2
4	42.9%	3
5	14.3%	1
Effective amount		7
8. I think the Tutorial module is useful		
Options	Percentage%	Subtotal
1	0.0%	0
2	0.0%	0
3	14.3%	1
4	42.9%	3
5	42.9%	3
Effective amount		7
9. I think the Process module is clear		
Options	Percentage%	Subtotal
1	0.0%	0
2	0.0%	0
3	14.3%	1
4	14.3%	1
5	71.4%	5
Effective amount		7
10. I think the Correctness module is useful		
Options	Percentage%	Subtotal
1	0.0%	0
2	0.0%	0
3	28.6%	2
4	57.1%	4

SURVEY STATISTICS (1 FOR DISAGREE, 5 FOR AGREE)		
5	14.3%	1
Effective amount		7
11. I really learned something about sorting algorithms by using this software		
Options	Percentage%	Subtotal
1	0.0%	0
2	0.0%	0
3	0.0%	0
4	28.6%	2
5	71.4%	5
Effective amount		7
12. I encountered confusing parts when using this software		
Options	Percentage%	Subtotal
1	14.3%	1
2	28.6%	2
3	14.3%	1
4	42.9%	3
5	0.0%	0
Effective amount		7
13. I will recommend this software to my friends		
Options	Percentage%	Subtotal
1	0.0%	0
2	0.0%	0
3	0.0%	0
4	57.1%	4
5	42.9%	3
Effective amount		7
14. Any suggestions		
Number	Answer	Time
6	All Good. Please keep developing!	2021-03-28T00:39:37+08:00
4	no so far	2021-03-27T15:47:11+08:00
3	no	2021-03-27T15:45:03+08:00

SURVEY STATISTICS (1 FOR DISAGREE, 5 FOR AGREE)		
2	no	2021-03-27T15:44:41+08:00