Verification and Validation

Testing

1. Testing Approach

To make the system more robust and increase the developing efficiency of team, a shared testing protocol should be followed within the teams during the whole developing procedure.

Note: In Sprint One, we focus more on the functionalities of the system other than its user experience. Therefore, for the automatic tests, we used Jest for functional tests. We will move to Cypress for UI testing in the next sprint.

Our frontend project uses React as framework, with the functions are rather isolated, and components can be separated. There are only five members in the team, therefore each developer needs to play a tester role. Because of the characteristics of this project, we design our testing approach with **three** steps.

Here's a table to give a quick vision of the whole procedure:

Testing Type	Who's in charge	Automation Degree	When to test	Failure handling	Test Cases	Testing result documentary	Note
Unit Testing	Developer	Manual/Automatic	After finishing a feature; before merging to the releasing branch	Developer fixes him/herself.	Testing Document Update it if necessary. (For user acceptance testing, tests should be conducted in different OS/browsers)	Travis CI has records, so don't need to keep it manually	
System Testing	Tester	Manual	After Merging into the releasing branch	Notice the developer on Kanban, record it in the document, set bug fixing due date according to its severity and priority.		System Testing Form (refer to below)	The tester should not be the same one to the feature's developer
User Acceptance Testing	Tester Team	Manual	Before releasing to the users and clients	Same as above		User Acceptance Testing Form (refer to below)	

1. Unit Testing

Before the test:

The developer is required to do unit tests after he/she finishing a function. All the requirements listed in user stories have their corresponding functions, and all the functions have their corresponding test cases.

When testing:

It's suggested to use the test cases listed in the Testing Document. The developer can also design their own tests. If any test cases are added, the testing document should also be updated.

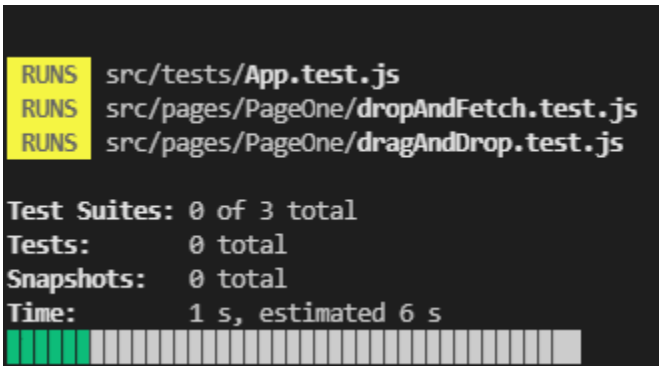
Tests can be conducted manually or automatically, depending on its **execution type** set in the testing document. If the function is tested automatically, a test file should also be uploaded within the function file folder.

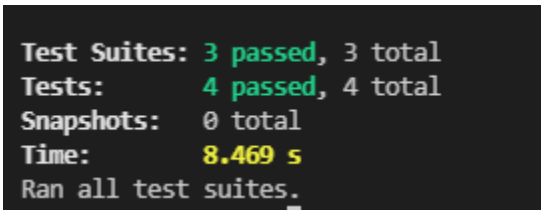
Example:

Original file name: <*filename*>.js

Corresponding test file name:<*filename*>.test.js

We use Jest to do the automatic tests. Run *npm test* and the results will be shown in the terminal like this:





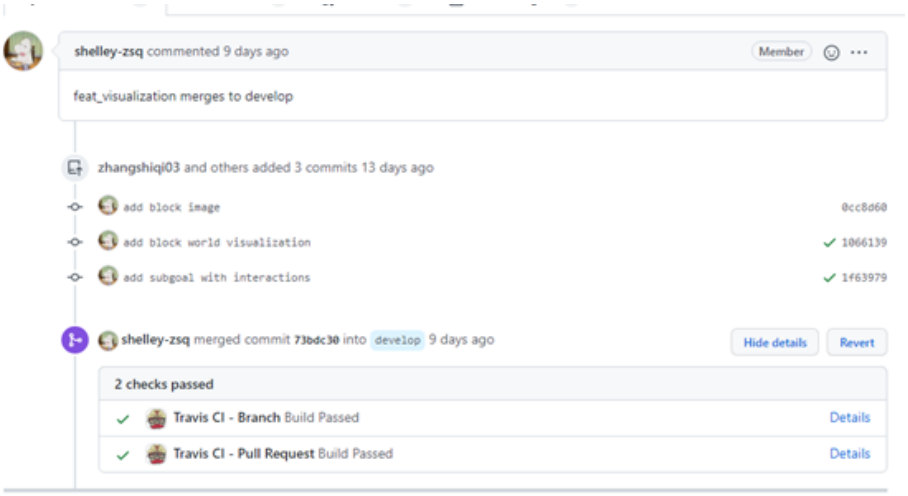
After the test:

The failures during unit tests don't need to be recorded. The developer can fix the bugs themselves.

When the code has passed the local tests, it is ready to merge.

Travis CI will automatically build the system and run all the tests whenever there are merge requests to the protected branches.

Example:



2. System Testing

After a new feature has merged into the **develop** branch, a system testing should be conducted by a tester to check if this feature can be operated correctly in the system, and if other functions are affected. The tester should be someone other than the feature's developer. If the tester finds a bug, it's encouraged that the tester creates a card in the to-do list on Kanban and mention the developer in it. After a system testing is done, the tester needs to fill in a testing result form, stating the bug, its severity, and its fixing due date. After the bug has fixed and merged, the tester should operate the system testing again and update the testing result form.

Note: as our features are rather small, it's not necessary to do a system testing every time a merge created. It can be done once there are 4-5 features merged.

In Sprint one, we did one system testing.

3. User Acceptance Testing

This is the last testing procedure before we release a version to the clients and users. The testing needs to be done by the whole testing team with as much as different OS/browsers. The testers should fill in a user acceptance testing form. Similar to the system testing approach, if any problem has been found, it should be recorded in the form and assign to developer team. The testing should be conducted repeatedly until all the tests have passed.

2. Testing Documents Sample

System Testing Form

Date:	**Tester:**
Environment:	

Severity	**Problem details**	**Fix before**
P0		
P1		
P2		
Notes: .		

User Acceptance Testing Form

Roles & Responsibilities		
Name	**Role**	**Responsibilities**

Test Results			
Test Case ID	**Result**	**Tester**	**Note**

Signature		
Name	**Signature**	**Date**

Reviewing

This table shows the reviewing protocol within the team

When to Review	In Charge	What to Review	Notice	References /Standards	N ote
Update a test case	QA Bojing Zhou	Check the coverage, if it has replication	mention QA in Testing Document when there are updates	Testing Document	
New merge request has created	Developer/Scrum Master	Code reviews, including coding styles, coding standards, logic to improve etc,.	When pulling a merge request, assign at least one reviewer on github	React Coding Standards	

To-do list changes	Scrum Master Shi qi ZHANG	Decide if a task should be re-prioritised in Kanban	Standup meetings	User stories	

Others

Version Control is used together with testing and reviewing.