Quality Assurance

- React Coding Standards
- Version Control
- Verification and Validation

React Coding Standards

NAMING CONVENTIONS

- Component's names should be written using the pascal case:

Header.js
HeroBanner.js
CookieBanner.js
BlogListing.js

- Non-components should be written using camel case:

myUtilityFile.js
cookieHelper.js
fetchApi.js

- Unit test files should use the same name as their corresponding file:

CookieBanner.js
CookieBanner.test.js

- The attribute name should be camel case:

className
onClick

- Inline styles should be camel case:

<div style={{font-size:'1rem'}}></div>

- Variable names should be camel cases. Variable names can contain numbers and special characters:

const variable = 'test';
let variableBoolean = true;

- CSS files should be named the same as the component:

CookieBanner.css
Header.css

- If a component requires multiple files (CSS, test) locate all files within component a folder
- Use .jsxor .tsx extension a for React components

BUG AVOIDANCE

- Use optional chaining if things can be null
- Use the guard pattern/prop types/typescript to ensure that the parameters you pass in are valid
- Create PURE functions and avoid side-effects
- Avoid mutating state when working with arrays
- Treat props as read-only. Do not try to modify them

ARCHITECTURE & CLEAN CODE

- No DRY violations. Create utility files to avoid duplicate code
- Follow the component/presentation pattern where appropriate. Components should follow the single responsibility principle
- Use Higher-Order Components where appropriate
- Split code into respective files, JavaScript, test, and CSS
- Create a index.js within each folder for exporting. This will reduce repeating names on the imports
- Only include one React component per file
- Favour functionless components
- Do not use mixins

- No unneeded comments
- Methods that are longer than the screen should be refactored into smaller units
- Commented out code should be deleted, not committed

CSS

- Avoid Inline CSS
- A naming convention is defined and followed (BEM, SUIT, etc..)

ES6

- Can you use spread operator be used instead?
- Can you use destructuring be used instead?
- Favour arrow functions
- Can the spread operator be used instead?
- Can the optional chain operator be used instead of an explicit null check
- Can nullish coalescing by used instead of an explicit null comparison

Version Control

On GitHub, we need to stick to a certain standards so that the teams could collaborate better.

There are two **protected branches**: Whenever there's a push or merge request, it requires the approve of scrum master and it requires more than one reviewer. And it will automatically run the tests.

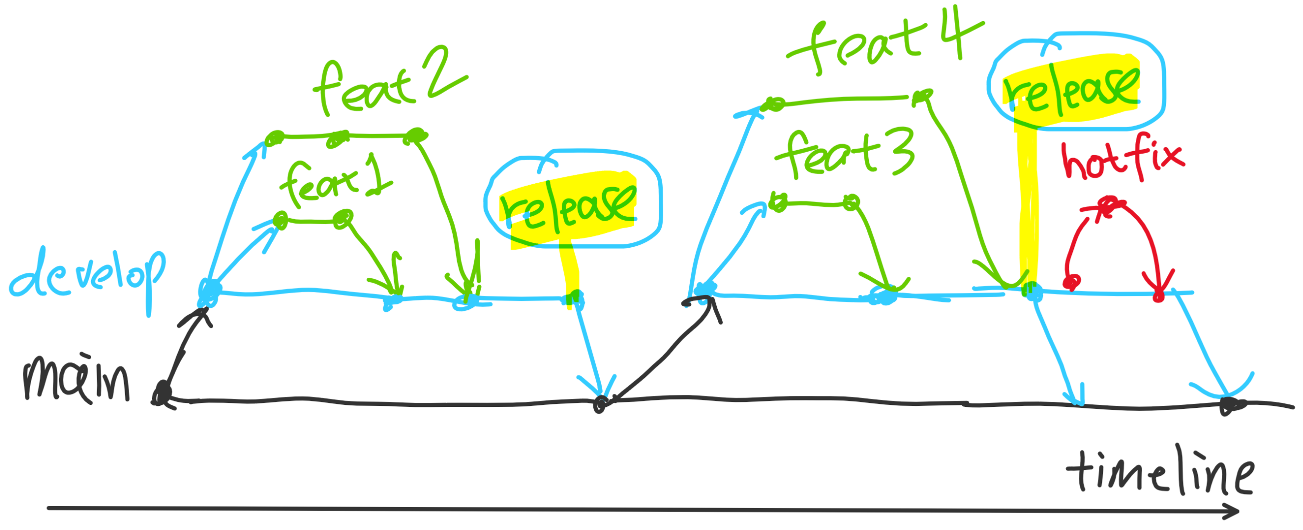
- **main** This branch is a **protected** branch. After the version on **develop branch** is released and there's no bug on it, the **develop** branch can be merged into main. **Only develop branch** can merge into this branch. This guarantees that the **main branch** is always pure and lack of bugs. After releasing, we merge **develop branch** into this **main branch**.

- **develop**This branch is the **default** branch. When using *merge* command, the default target branch is this one. At the end of each sprint, we release the version on this branch. Every merge into this branch should be runnable and tested in the local environment. The codes on this branch is updated in anytime. **Create** a new branch from this one every time you start a new task. This ensures your developing branches is always updated. After a while of launching, if there are not many problems on the current release, we can merge this branch into the **main branch** .

We use feature branches to code in our local environment. It's recommended that you create a new feature branch each time when you develop a new feature(task). If a few of features are mixed in one branch, it will be difficult when you want to launch one of them. For the purpose of readability, it's recommended that all the members use the format below to name your developing branch.

- **feat_XXX**
  - An example: *feat_homepageNavigator*
  - Create branches from **develop branch** *only*
  - When developing your feature, please don't fix others' bugs or modify codes which are not wrote by you. Otherwise it's easy to get conflicts if someone is also modifying the same file, and fixing conflicts is really time-consuming. If you find there's something wrong, create a task on Trello with some description and assign it to your team member if you know who's in charge of it. Note: please don't use code formatter for *the whole project*.
  - Only raise a merge request after your feature has completed and fully tested in your local environment. Commit and push them into your own branch if you're in the halfway.
  - After the current branch is merged into the **develop branch**, it's highly suggested that you create a new branch from **develop branch**, other than using the same old one.

- **hotfix_XXX**
  - We may not use this kind of branch much since we only have two sprints. Usually after releasing, if there's any problem online, we create this branch to fix a small bug and merge it into **develop branch**. Please don't use this to develop a feature or fix larger bugs.



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.

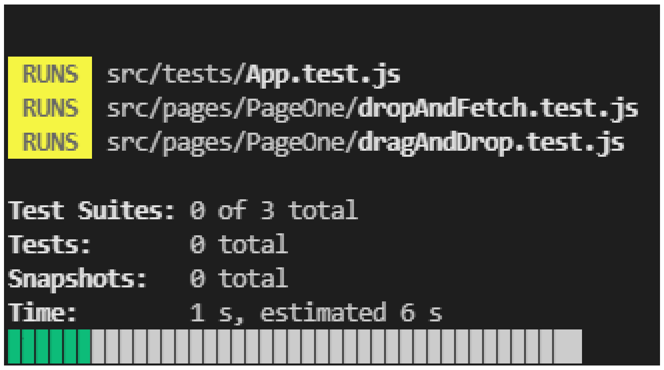
Tests can be conducted manually or automatically, depending on its **execution type** set in the testing document. If the function is tested automatically, a test file should also be uploaded within the function file folder.

Example:

Original file name: <*filename*>.js

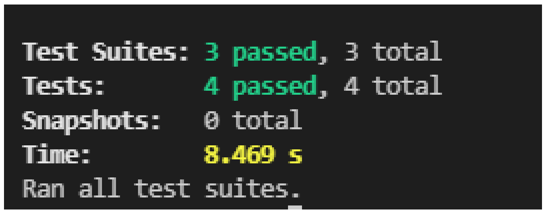
Corresponding test file name:<*filename*>.test.js

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



```
RUNS  src/tests/App.test.js
RUNS  src/pages/PageOne/dropAndFetch.test.js
RUNS  src/pages/PageOne/dragAndDrop.test.js

Test Suites: 0 of 3 total
Tests:       0 total
Snapshots:   0 total
Time:        1 s, estimated 6 s
```



```
Test Suites: 3 passed, 3 total
Tests:       4 passed, 4 total
Snapshots:   0 total
Time:        8.469 s
Ran all test suites.
```

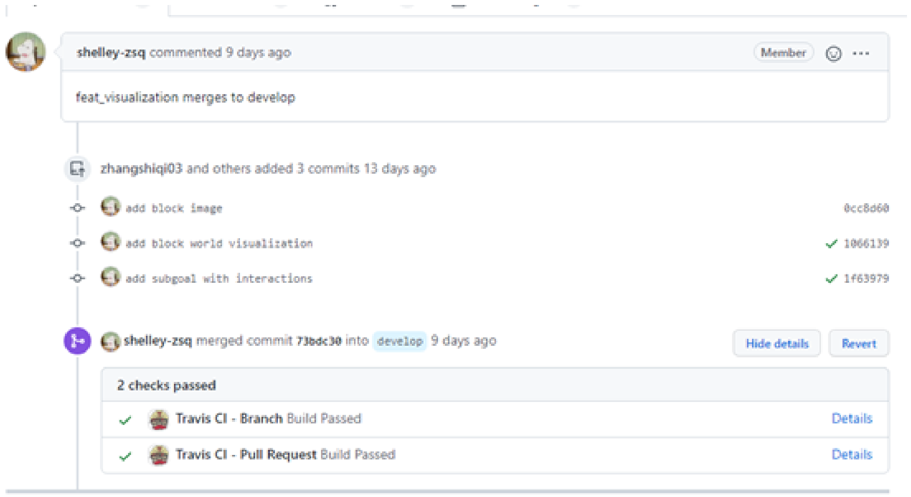
After the test:

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:



shelley-zsq commented 9 days ago

feat_visualization merges to develop

zhangshiqi03 and others added 3 commits 13 days ago

```
add block image                        0cc8d60
add block world visualization        ✓ 1066139
add subgoal with interactions        ✓ 1f63979
```

shelley-zsq merged commit 73bdc30 into develop 9 days ago

2 checks passed

✓ Travis CI - Branch Build Passed

✓ Travis CI - Pull Request Build Passed

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	Note
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.

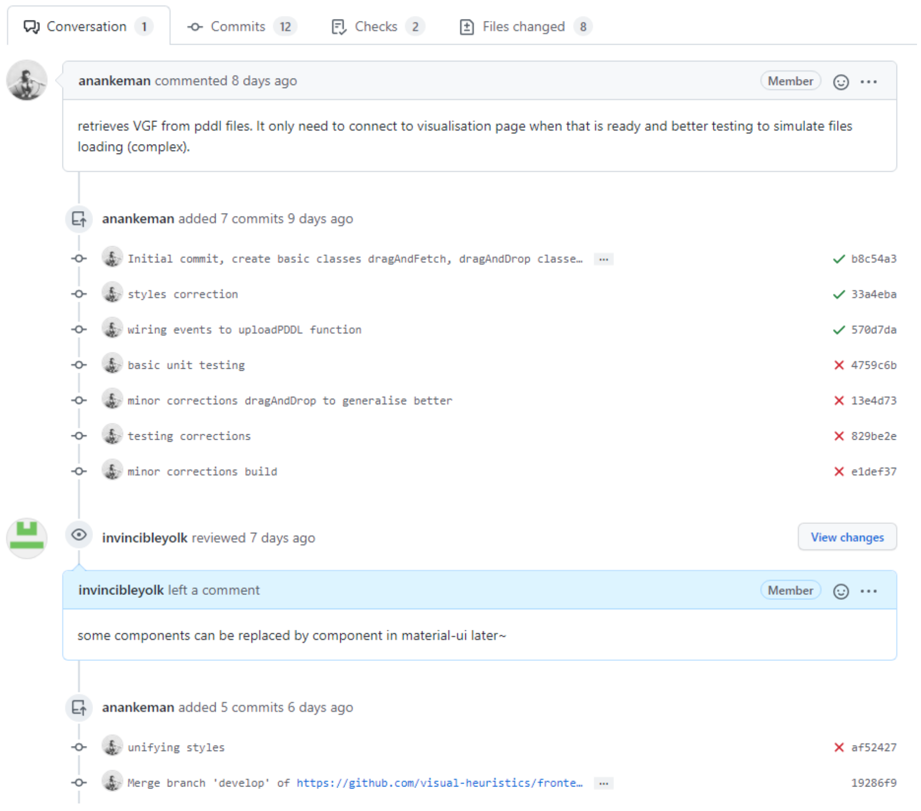
Reviewing

- Sprint One Reviewing
- Sprint Two Reviewing

Sprint One Reviewing

In this sprint, the team did some code reviews when pulling a merge request in github.

An example screenshot of one merge:



anankeman commented 8 days ago — Member

retrieves VGF from pddl files. It only need to connect to visualisation page when that is ready and better testing to simulate files loading (complex).

anankeman added 7 commits 9 days ago

```
Initial commit, create basic classes dragAndFetch, dragAndDrop classe…   ✓ b8c54a3
styles correction                                                        ✓ 33a4eba
wiring events to uploadPDDL function                                     ✓ 570d7da
basic unit testing                                                       ✕ 4759c6b
minor corrections dragAndDrop to generalise better                       ✕ 13e4d73
testing corrections                                                      ✕ 829be2e
minor corrections build                                                  ✕ e1def37
```

invincibleyolk reviewed 7 days ago — View changes

invincibleyolk left a comment — Member

some components can be replaced by component in material-ui later~

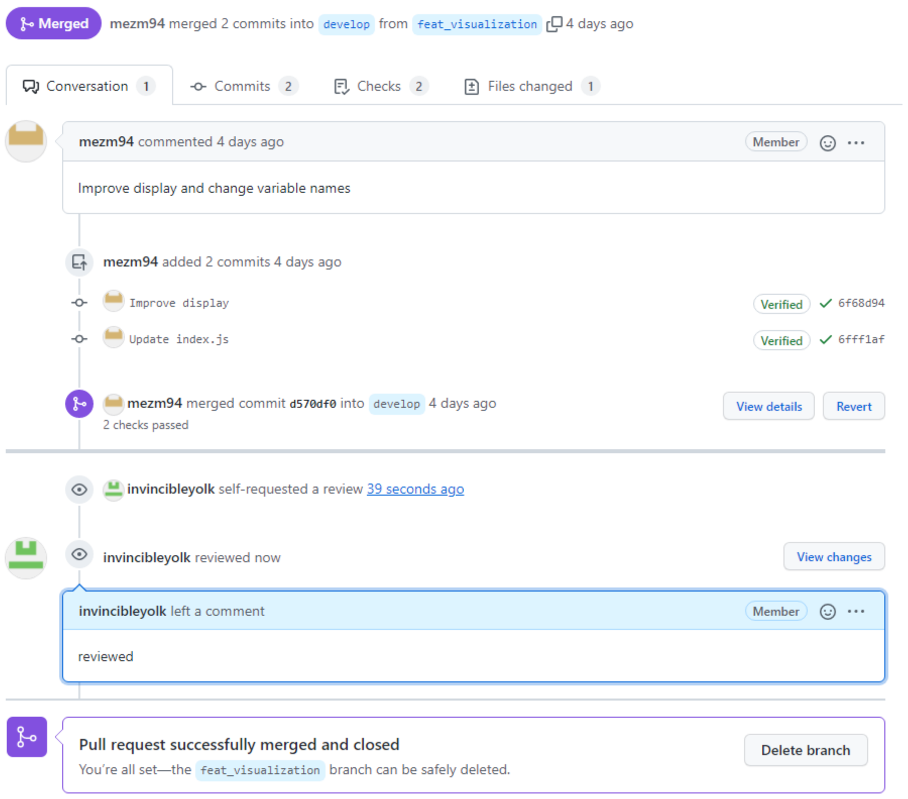
anankeman added 5 commits 6 days ago

```
unifying styles                                                          ✕ af52427
Merge branch 'develop' of https://github.com/visual-heuristics/fronte…     19286f9
```

Sprint Two Reviewing

In this sprint, the team did some code reviews when pulling a merge request in github.

An example screenshot of one merge:



Merged mezm94 merged 2 commits into `develop` from `feat_visualization` 4 days ago

Conversation 1 · Commits 2 · Checks 2 · Files changed 1

mezm94 commented 4 days ago — Member

Improve display and change variable names

mezm94 added 2 commits 4 days ago

- `Improve display` — Verified ✓ 6f68d94
- `Update index.js` — Verified ✓ 6fff1af

mezm94 merged commit **d570df0** into `develop` 4 days ago — View details · Revert

2 checks passed

invincibleyolk self-requested a review 39 seconds ago

invincibleyolk reviewed now — View changes

invincibleyolk left a comment — Member

reviewed

Pull request successfully merged and closed

You're all set—the `feat_visualization` branch can be safely deleted. — Delete branch

Testing

- Testing
- Sprint 1 Testing
- Sprint 2 Testing

Testing Document

COMP90082

Planimation

Team: PL-boxjelly

Bojing Zhou Bojing Zhou

Felipe Ramos Morales Felipe Ramos Morales

Shiqi Zhang Shiqi ZHANG

Xiaoyu Zhang XIAOYU ZHANG

Ziqi Meng Ziqi Meng

<students must use this table to track individuals' contributions to this document.

Every time you change this document, add the date you changed it, a description of your performed task and your name. For the version, please adopt the following format:

01.00-D<number> for draft versions related to Part 1 (any version before the final submission is considered draft). When your document is reviewed and finally ready to be submitted, change it to 01.00. For Part2, start with 02.00-D<number> and so on. This document should always be kept on GitHub>

(The updated modification for the sprint 2 is marked in green color)

Revision History

Date	Version	Description	Author
6/9/2021	01.00-D01	Initial draft, design functional test cases	Bojing Zhou
7/9/2021	01.00-D02	Functional test cases for US001	XIAOYU ZHANG
8/9/2021	01.00-D03	Functional test cases for US002	Felipe Ramos Morales
8/9/2021	01.00-D04	Functional test cases for US003	Shiqi ZHANG
9/9/2021	01.00-D05	Functional test cases for US006	Ziqi Meng
19/9/2021	01.00	Finish other test cases and entry data, final review and improvements on the document	Bojing Zhou
12/10/2021	02.00-D01	Test cases for US004, US005	XIAOYU ZHANG
14/10/2021	02.00-D02	Test cases for US011	Shiqi ZHANG
16/10/2021	02.00-D03	Test cases for US012	Ziqi Meng
17/10/2021	02.00-D04	Functional test cases for US013	Felipe Ramos Morales

20/10/2021	02.00	Test cases for US015, final review and improvements on the document	Bojing Zhou

Contents

1. Introduction

1.1 Proposal

The purpose of this document is to define and present the test cases for project Platimation and team boxjelly, covering the test cases for the system use cases.

1.2 Target Users

This document in mainly designed for those responsible for executing the test cases in this project [team members and SWEN90082 teaching team].

1.3 Conventions, terms and abbreviations

This section explains the concept of some important terms that will be used throughout this document. These terms are described in the following table, presented in alphabetical order.

Term	Description

2. Covered Requirements

This section lists the system requirements covered in the test cases.

2.1 Functional or Product Requirements

Requirement Identifier (User Story ID)	Requirement Name
US001	As a user, I could access the main interface for access to four sub-modules (including generating the visualisation from problem files, generating the visualisation from VFG file, accessing the user manual and accessing the demo).
US002	As a user, I could upload domain, problem, and animation PDDL files for generating the visualisation of the plan (i.e. solution) of this planning problem.
US003	As a user, I could upload a VFG file for generating the visualisation directly.
US006	As a user, I could view the animation of the visualisation of a particular planning problem on the visualizer page after uploading the files.
US007	As a user, I could check each step of the plan, the status of any step in the animation by selecting a particular step, and the detailed step information of the selected step on the visualizer page.
US008	As a user, I could check the subgoals of each step and all the steps corresponding to a certain subgoal.
US009	As a user, I could view the visualization of the final goal state.
US010	As a user, I could check the visualization status of the previous or next step.
US004	As a user, I could find a demo video or doc demonstration to learn how to operate this animation.
US005	As a user, I could find a user manual to help me use this web-based application.
US011	As a user, I could control the display of the animation, including play, pause, and reset.
US012	As a user, I could control the display speed of the animation.
US013	As a user, I could export the animation file.
US015	As a user, I could load Planimation from Visual Studio Code

3. Functional Test Cases

This section describes the test cases that cover the product requirements of the system.

3.1 US001: Access the homepage

3.1.1 TC01-1: Jump to pages successful

Test Type: Functional	**Execution Type:** Manual
Objective: Verify if the links can route to certain pages successfully.	

Setup: · None
Pre-Conditions: · None
Notes: [1] Click the buttons to perform page jumps. [2] Use forward, backward to perform page jumps. [3] Refresh first then perform page jumps. [4] Use fast 3G mode to perform page jumps. [5] Use slow 3G mode to perform page jumps. * Corresponding pages loaded successfully
Time constraint: Minimum: <1s Maximum: ~10s

3.1.2 TC01-2: Jump to pages unsuccessful

Test Type: Functional	**Execution Type:** Manual
Objective: Verify that the links do NOT route to certain pages successfully.	
Setup: · None	
Pre-Conditions: · None	
Notes: [1] Try offline mode to perform page jumps. * A 404-error page should show if it’s time out/the browser cannot load the page successfully	
Time constraint: Minimum: <1s Maximum: ~1min	

3.2 US002: Upload domain, problem, animation PDDL files

3.2.1 TC02-1: Upload domain file successful

Test Type: Functional	**Execution Type:** Manual/Automatic
Objective: Verify if uploading a Domain file is performed successfully.	

Setup: · None
Pre-Conditions: · None
Notes: [1] Upload the Domain file. [2] Cancel the chosen file then upload again. [3] Upload the same Domain file after uploading one. [4] Upload another Domain file after uploading one. *The file should be uploaded successfully and the latter one should overwrite the former.
Time constraint: Minimum: <1min Maximum: 2 min

3.2.2 TC02-2: Upload domain file unsuccessful

Test Type: Functional	**Execution Type:** Manual/Automatic
Objective: Verify that uploading a Domain file is NOT performed successfully.	
Setup: · None	
Pre-Conditions: · None	
Notes: [1] Try to upload a file other than pddl extension. [2] Try to upload an empty pddl file. [3] Try to upload a pddl file with some language characters other than English. [4] Try to upload the file when it’s offline. *The file should not be uploaded and there should be a warning for the user.	
Time constraint: Minimum: <1min Maximum: 2 min	

3.2.3 TC02-3: Upload problem file successful

Test Type: Functional	**Execution Type:** Manual/Automatic

Objective: Verify if uploading a Problem file is performed successfully.
Setup: · None
Pre-Conditions: · None
Notes: [1] Upload the Problem file. [2] Cancel the chosen file then upload again. [3] Upload the same Problem file after uploading one. [4] Upload another Problem file after uploading one. *The file should be uploaded successfully and the latter one should overwrite the former.
Time constraint: Minimum: <1min Maximum: 2 min

3.2.4 TC02-4: Upload problem file unsuccessful

Test Type: Functional	**Execution Type:** Manual/Automatic
Objective: Verify that uploading a Problem file is NOT performed successfully.	
Setup: · None	
Pre-Conditions: · None	
Notes: [1] Try to upload a file other than pddl extension. [2] Try to upload an empty pddl file. [3] Try to upload a pddl file with some language characters other than English. [4] Try to upload the file when it’s offline. *The file should not be uploaded and there should be a warning for the user.	
Time constraint: Minimum: <1min Maximum: 2 min	

3.2.5 TC02-5: Upload animation file successful

Test Type: Functional	**Execution Type:** Manual/Automatic

Objective: Verify if uploading an Animation file is performed successfully.
Setup: · None
Pre-Conditions: · None
Notes: [1] Upload the Animation file. [2] Cancel the chosen file then upload again. [3] Upload the same Animation file after uploading one. [4] Upload another Animation file after uploading one. *The file should be uploaded successfully and the latter one should overwrite the former.
Time constraint: Minimum: <1min Maximum: 2 min

3.2.6 TC02-6: Upload animation file unsuccessful

Test Type: Functional	**Execution Type:** Manual/Automatic
Objective: Verify that uploading an Animation file is NOT performed successfully.	
Setup: · None	
Pre-Conditions: · None	
Notes: [1] Try to upload an Animation file other than pddl extension. [2] Try to upload an empty pddl file. [3] Try to upload a pddl file with some language characters other than English. [4] Try to upload the file when it’s offline. *The file should not be uploaded and there should be a warning for the user.	
Time constraint: Minimum: <1min Maximum: 2 min	

3.3 US003: Upload VFG file

3.3.1 TC03-1: Upload VFG file successful

Test Type: Functional	Execution Type: Manual/Automatic
Objective: Verify if uploading a VFG file is performed successfully.	
Setup: · None	
Pre-Conditions: · None	
Notes: [1] Upload the VFG file. [2] Cancel the chosen file then upload again. [3] Upload the same file after uploading one. [4] Upload another file after uploading one. *The file should be uploaded successfully and the latter one should overwrite the former.	
Time constraint: Minimum: <1min Maximum: 2 min	

3.3.2 TC03-2: Upload VFG file unsuccessful

Test Type: Functional	Execution Type: Manual/Automatic
Objective: Verify that uploading a VFG file is NOT performed successfully.	
Setup: · None	
Pre-Conditions: · None	
Notes: [1] Try to upload a VFG file other than vfg extension. [2] Try to upload an empty vfg file. [3] Try to upload a vfg file with some language characters other than English. [4] Try to upload the file when it’s offline. *The file should not be uploaded and there should be a warning for the user.	
Time constraint: Minimum: <1min Maximum: 2 min	

3.4 US006: Visualise files

3.4.1 TC06-1: Parse VFG file successful

Test Type: Functional	**Execution Type:** Manual/Automatic
Objective: Verify if parsing the vfg file is performed successfully.	
Setup: · VFG file has uploaded	
Pre-Conditions: · None	
Notes: [1] Parse the file *The file should be parsed successfully and the page jumps to the Demo page	
Time constraint: Minimum: <1 min Maximum: 3 min	

3.4.2 TC06-2: Parse VFG file unsuccessful

Test Type: Functional	**Execution Type:** Manual/Automatic
Objective: Verify that parsing a VFG file is NOT performed successfully.	
Setup: · None	
Pre-Conditions: · None	
Notes: [1] Try to parse an empty vfg file. [2] Try to parse a vfg file with wrong format. [3] Try to parse a vfg file with correct format but empty values. [4] Try to parse the file when it’s offline. *The system should pop up a warning about the error and stays in the current page.	
Time constraint: Minimum: <1min Maximum: 2 min	

3.5 US013: Export animation file

3.5.1 TC13-1: Export animation file successful

Test Type: Functional	**Execution Type:** Manual/Automatic
Objective: Verify if exporting a VFG file is performed successfully.	
Setup: · At the animation page	
Pre-Conditions: · None	
Notes: [1] Export the VFG file. [2] Cancel the download then exporting again. [3] Export the VFG file when playing the animation. [4] Try to export the file when it’s offline. *The file should be downloaded successfully	
Time constraint: Minimum: <1min Maximum: 2 min	

3.5.2 TC13-2: Export animation file unsuccessful

Test Type: Functional	**Execution Type:** Manual/Automatic
Objective: Verify that exporting a VFG file is NOT performed successfully.	
Setup: · At the animation page	
Pre-Conditions: · None	
Notes: [1] Try to export a VFG file when there's no animation displayed *The file should not be downloaded and there should be a warning for the user.	
Time constraint: Minimum: <1min Maximum: 2 min	

4. Test Cases

This section describes the general test cases that will be related to more than one requirement or assumption / constraint, avoiding data replication.

4.1 TC07-1: Select plan steps

Test Type: Functional	Execution Type: Manual/Automatic
Objective: Verify if the displayed information is performed successfully when selecting.	
Setup: ·　A correct animation pddl file is already loaded	
Pre-Conditions: ·　None	
Notes: [1] Select steps online. [2] Select steps offline. *The corresponding step information, subgoals should be displayed, and the animation should be played.	
Time constraint: Minimum: <1s Maximum: 1s	

4.2 TC08-1: Select subgoals

Test Type: Functional	Execution Type: Manual/Automatic
Objective: Verify if the displayed information is performed successfully when selecting.	
Setup: ·　A correct animation pddl file is already loaded	
Pre-Conditions: ·　None	
Notes: [1] Select steps online. [2] Select steps offline. *The corresponding step information, subgoals should be displayed, and the animation should be played.	
Time constraint: Minimum: <1s Maximum: 1s	

4.3 TC10-1: Control animation

Test Type: Functional	Execution Type: Manual/Automatic
Objective: Verify if the displayed information is performed successfully when controlling.	

Setup: · A correct animation pddl file is already loaded
Pre-Conditions: · None
Notes: [1] Play the animation from the start. [2] Select a step before playing the animation. [3] Select a subgoal before playing the animation [4] Select a step after playing the animation. [5] Select a subgoal after playing the animation [6] Select a step while playing the animation. [7] Select a subgoal while playing the animation [8] Play the animation after the end [9] Pause the animation [10] Select previous step [11] Select next step *The corresponding step information, subgoals should be displayed, and the animation should be played/paused.
Time constraint: Minimum: <1s Maximum: 1s

4.4 TC04-1: Link to Demo

Test Type: Functional	**Execution Type:** Manual/Automatic
Objective: Verify if the link could jump to the demo url successfully when clicking.	
Setup: · None	
Pre-Conditions: · None	
Notes: [1] Click link online. [2] Click link offline. [3] Click link from navigation bar [4] Click link from homepage *The corresponding url should open in another page.	

Time constraint: Minimum: <1s Maximum: 1s

4.5 TC05-1: Link to User Manual

Test Type: Functional	**Execution Type:** Manual/Automatic
Objective: Verify if the link could jump to the user manual url successfully when clicking.	
Setup: · None	
Pre-Conditions: · None	
Notes: [1] Select steps online. [2] Select steps offline. [3] Click link from navigation bar [4] Click link from homepage *The corresponding url should open in another page.	
Time constraint: Minimum: <1s Maximum: 1s	

4.7 TC11-1: Control Animation

Test Type: Functional	**Execution Type:** Manual/Automatic
Objective: Verify if the animation is performed successfully when selecting different buttons (play, pause, and reset).	
Setup: · A correct animation pddl file is already loaded, the animation has shown	
Pre-Conditions: · None	

Notes: [1] Play the animation from the start. [2] Click play while playing the animation. [3] Click pause while playing the animation. [4] Click reset while playing the animation. [5] Click play while pausing the animation. [6] Click pause while pausing the animation. [7] Click reset while pausing the animation. [8] Play the animation after the end [9] Pause the animation *The corresponding step information, subgoals should be displayed, and the animation should be played.
Time constraint: Minimum: <1s Maximum: 1s

4.8 TC12-1: Control Speed of Animation

Test Type: Functional	**Execution Type:** Manual/Automatic
Objective: Verify if the displayed information is performed successfully when controlling its speed.	
Setup: · A correct animation pddl file is already loaded, the animation has shown	
Pre-Conditions: · None	
Notes: [1] Add speed while playing the animation. [2] Add speed while pausing the animation. [3] Minus speed while playing the animation. [4] Minus speed while pausing the animation. *The corresponding step information, subgoals should be displayed, and the animation should be faster/slower.	
Time constraint: Minimum: <1s Maximum: 1s	

4.9 TC15-1: Load Plugin on VS Code

Test Type: Functional	**Execution Type:** Manual/Automatic

Objective: Verify if the plugin is performed successfully in the VS Code.
Setup: · The plugin for VS Code has installed
Pre-Conditions: · None
Notes: [1] Input correct command on VS Code command palette.   *The corresponding interface should be displayed in the vs code panel.
Time constraint: Minimum: <1s Maximum: 1s

5. Entry Data

This section describes the entry data that will be used by more than one test case, avoiding data replication. These data are referenced by the test cases.

5.1 DATA VFG: <VFG Sample Data>

Description:

The test vfg data format in json format.

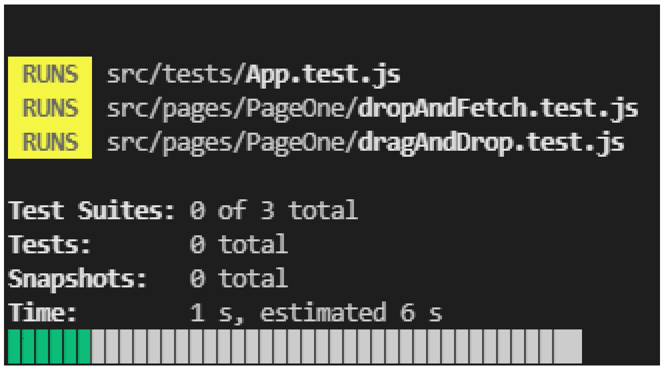
```
  {"visualStages": [
{
"visualSprites": [
{
"prefabimage":"img-block",
 "showname": true,
        "x":300,
        "y":82,
        "color":{ "r":1.0, "g":0.98, "b":0.8, "a":1.0},
        "width":80,
        "height":80,
        "name":"b",
        "minX":0.286,
        "maxX":0.357,
        "minY":0.091,
        "maxY":0.163
}
],
"stageName": "Initial Stage",
"stageInfo": "No Step Information"
"isFinal": "false"
}
  ],
  "subgoalPool": {
"m_keys": ["(on f g)", "(on c f)"],
"m_values": [
["f", "g"],
["c", "f"],
]
},
  "subgoalMap": {
"m_keys":[4, 5, 6],
"m_values":[
     [ "(on f g )" ], ["(on f g )" ],
]
  },
  "transferType":1,
  "imageTable": {
"m_keys":[
"img-claw",
    "img-block",
    "img-board"
],
"m_values": string[]
  },
  "message":""
}
```

Sprint 1 Testing

Unit Testing

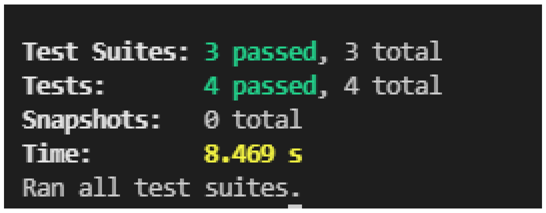
In this sprint, we focus more on the functionalities of the system other than its user experience. Therefore, for the automatic tests, we used Jest for functional tests. We will move to Cypress for UI testing in the next sprint.

Unit Testing in local workplace with Jest:



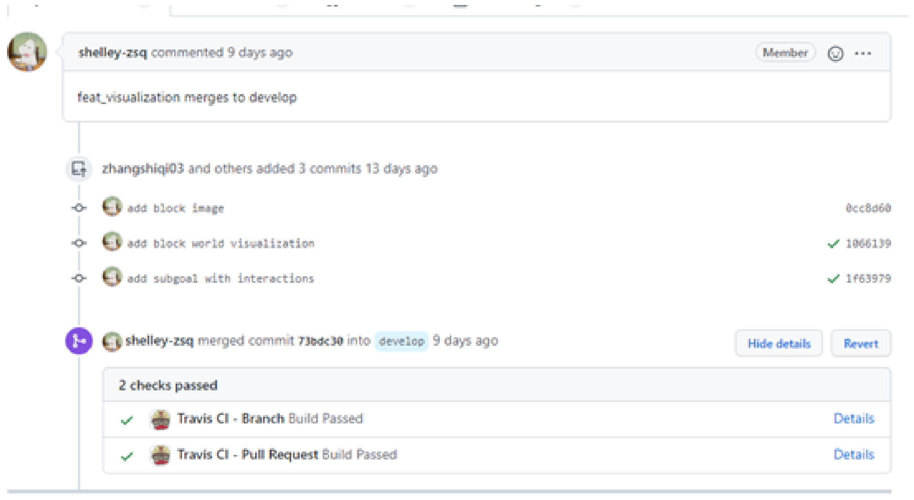
```
RUNS  src/tests/App.test.js
RUNS  src/pages/PageOne/dropAndFetch.test.js
RUNS  src/pages/PageOne/dragAndDrop.test.js

Test Suites: 0 of 3 total
Tests:       0 total
Snapshots:   0 total
Time:        1 s, estimated 6 s
```



```
Test Suites: 3 passed, 3 total
Tests:       4 passed, 4 total
Snapshots:   0 total
Time:        8.469 s
Ran all test suites.
```

Automatic Unit Testing on github with Travis:



shelley-zsq commented 9 days ago

feat_visualization merges to develop

zhangshiqi03 and others added 3 commits 13 days ago

```
add block image                          0cc8d60
add block world visualization          ✓ 1066139
add subgoal with interactions          ✓ 1f63979
```

shelley-zsq merged commit 73bdc30 into develop 9 days ago

2 checks passed

✓ Travis CI - Branch Build Passed

✓ Travis CI - Pull Request Build Passed

System Testing

In Sprint one, we did one system testing in total since all the features are rather small and independent. We will do more system testing in the next sprint.

The tester is assigned by QA leader

System Testing Form

Date:	**Tester:**
11 Sep 2021	Bojing Zhou

Environment:		
Windows 10, Google Chrome		
Severity	**Problem details**	**Fix before**
P0	✓ TC02-1 Error occurs when it's overtime, should catch the error and show an alert to user Felipe Ramos Morales	14 Sep 2021
P1	✓ TC07-1 The steps don't change when the subgoal changes Shiqi ZHANG	14 Sep 2021
P2	✓ Buttons layout needs adjusting Bojing Zhou	14 Sep 2021
	✓ Buttons spacing should be larger XIAOYU ZHANG	14 Sep 2021
	✓ Height of Animation page should be responsive Ziqi Meng	14 Sep 2021
Notes: 15 Sep 2021 Problems are solved. Has passed all the tests Bojing Zhou		

User Acceptance Testing

This test was conducted before the client meeting at the end of the sprint one on 15 Sep 2021

Resouces

The display of pages may vary in different operating systems and browsers, so in the user acceptance testing the team tested the system in different resources.

- Operating Systems:
  - OS X: 11.5
  - Windows 10
  - Linux: Ubuntu
- Browsers:
  - Chrome
  - Firefox
  - Safari

Test documentation

All test cases are documented within Confluence Testing Document.

Error Reporting

Failures or bugs are directly reported to the responsible developer and are fixed immediately.

User Acceptance Testing Form

Roles & Responsibilities		
Name	**Role**	**Responsibilities**
Bojing Zhou	QA leader	Test documentation work
Felipe Ramos Morales	Tester	Testing on Ubuntu, Firefox
Shiqi ZHANG	Scrum Master	Managing UAT test
XIAOYU ZHANG	Tester	Testing on OS X, Safari
Ziqi Meng	Tester	Testing on Windows, Chrome

Test Results

Test Case ID	Result	Tester	Note
TC01-1	Passed	Ziqi Meng	
TC01-2	Passed	Ziqi Meng	
TC02-1	Passed	Ziqi Meng	
TC02-2	Passed	Ziqi Meng	
TC02-3	Passed	Felipe Ramos Morales	
TC02-4	Passed	Felipe Ramos Morales	
TC02-5	Passed	Felipe Ramos Morales	
TC02-6	Passed	Felipe Ramos Morales	
TC03-1	Passed	Xiaoyu Zhang	
TC03-2	Passed	Xiaoyu Zhang	
TC06-1	Passed	Xiaoyu Zhang	
TC06-2	Passed	Xiaoyu Zhang	
TC07-1	Passed	Ziqi Meng	
TC08-1	Passed	Felipe Ramos Morales	
TC10-1	Passed	Xiaoyu Zhang	

Signature

Name	Signature	Date
Shiqi Zhang	Shiqi Zhang	15/9/2021
Xiaoyu Zhang	[illegible]	15/9/2021
Bojing Zhou	Bojing Zhou	15/9/2021

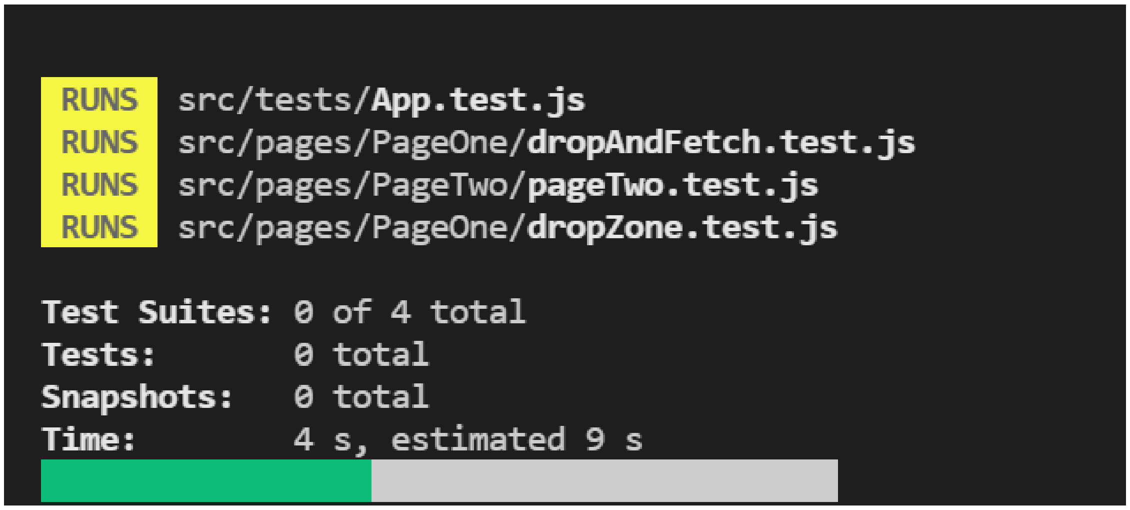
Felipe Ramos Morales	[illegible]	15/9/2021
Ziqi Meng	Ziqi Meng	15/9/2021

Sprint 2 Testing

Unit Testing

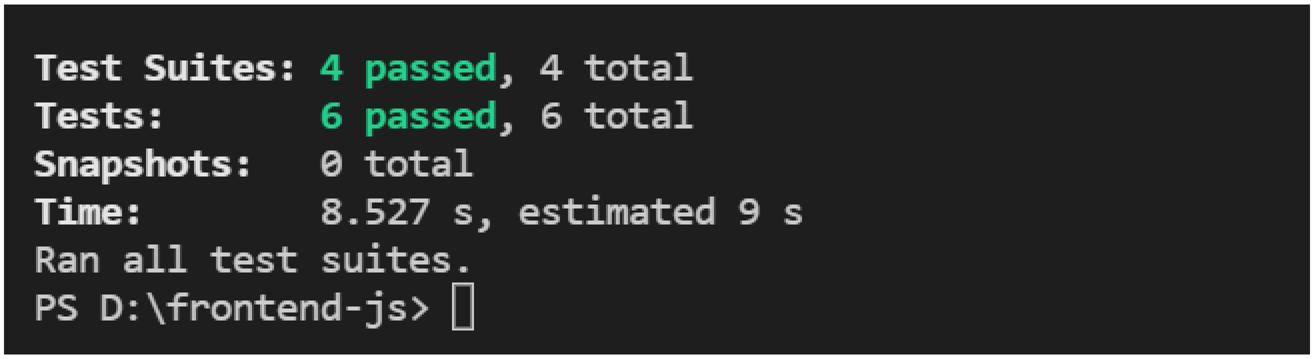
In the Sprint 2, we added more test cases. The tools for unit testing are Jest and Cypress. The two testing tools are orthogonal Jest is for smaller scope unit tests, and Cypress can be used for larger scope testing.

Unit Testing in local workplace with Jest:



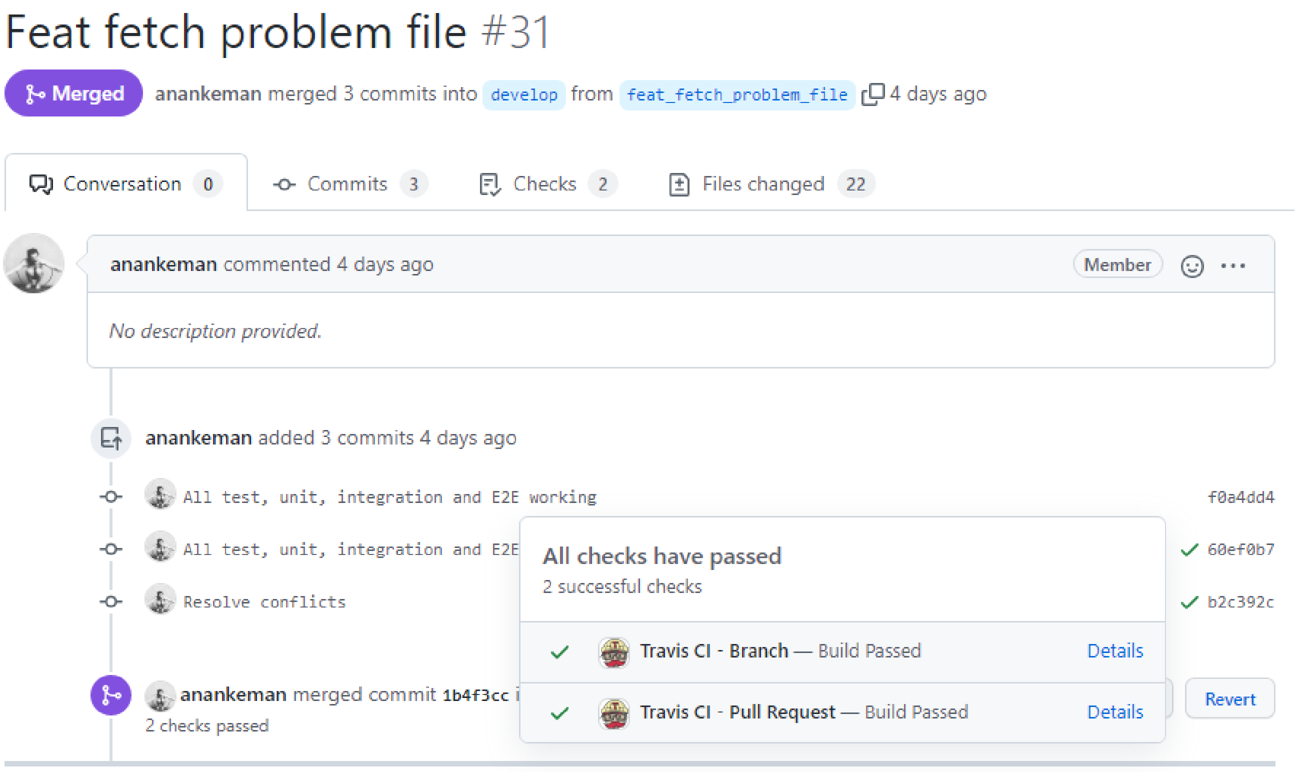
```
 RUNS  src/tests/App.test.js
 RUNS  src/pages/PageOne/dropAndFetch.test.js
 RUNS  src/pages/PageTwo/pageTwo.test.js
 RUNS  src/pages/PageOne/dropZone.test.js

Test Suites: 0 of 4 total
Tests:       0 total
Snapshots:   0 total
Time:        4 s, estimated 9 s
```



```
Test Suites: 4 passed, 4 total
Tests:       6 passed, 6 total
Snapshots:   0 total
Time:        8.527 s, estimated 9 s
Ran all test suites.
PS D:\frontend-js>
```

Automatic Unit Testing on github with Travis:



Feat fetch problem file #31

Merged anankeman merged 3 commits into `develop` from `feat_fetch_problem_file` 4 days ago

Conversation 0 | Commits 3 | Checks 2 | Files changed 22

anankeman commented 4 days ago — Member

No description provided.

anankeman added 3 commits 4 days ago

```
All test, unit, integration and E2E working                f0a4dd4
All test, unit, integration and E2E                        ✓ 60ef0b7
Resolve conflicts                                          ✓ b2c392c
```

All checks have passed
2 successful checks

✓ Travis CI - Branch — Build Passed — Details
✓ Travis CI - Pull Request — Build Passed — Details

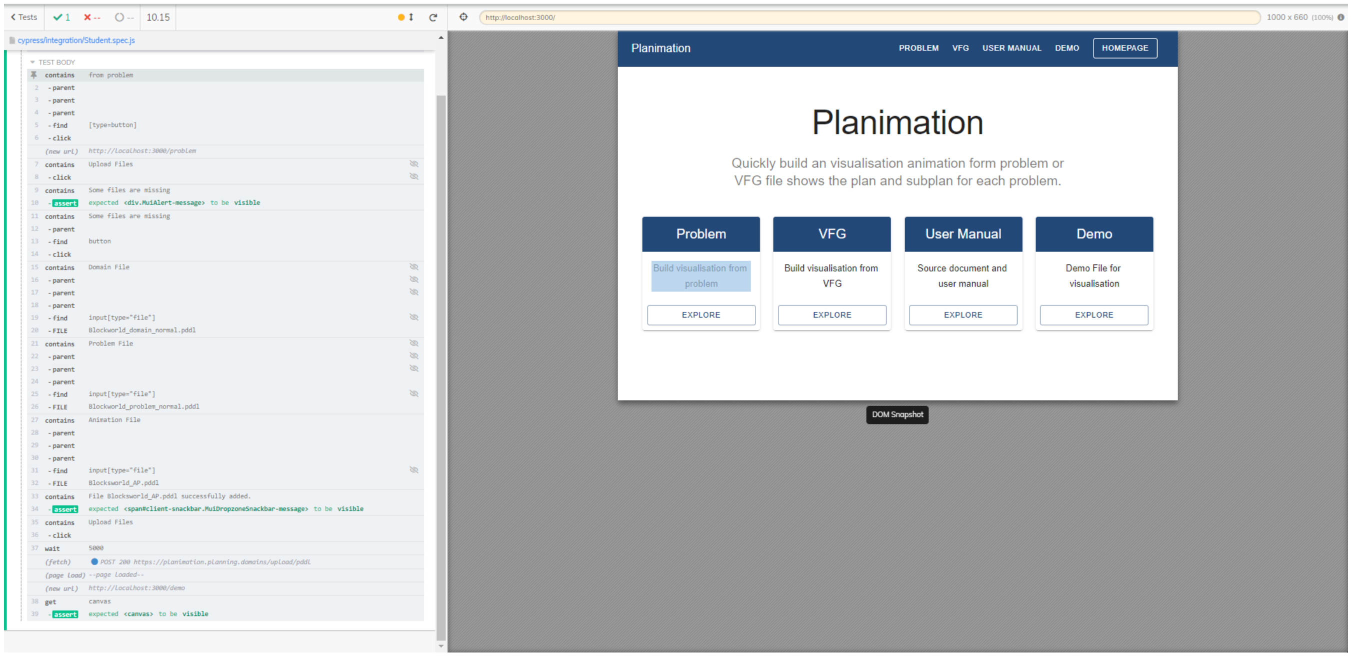
anankeman merged commit **1b4f3cc** i
2 checks passed

Revert

System Testing

In Sprint 2, we also did one system testing. Felipe Ramos Morales wrote system test cases with Cypress.

System testing with Cypress:



Tests ✓ 1 ✗ -- ○ -- 10.15

cypress/integration/Student.spec.js

TEST BODY

```
contains        from problem
 - parent
 - parent
 - parent
 - find         [type=button]
 - click
(new url)       http://localhost:3000/problem
contains        Upload Files
 - click
contains        Some files are missing
 - assert       expected <div.MuiAlert-message> to be visible
contains        Some files are missing
 - parent
 - find         button
 - click
contains        Domain File
 - parent
 - parent
 - parent
 - find         input[type="file"]
 - FILE         Blockworld_domain_normal.pddl
contains        Problem File
 - parent
 - parent
 - parent
 - find         input[type="file"]
 - FILE         Blockworld_problem_normal.pddl
contains        Animation File
 - parent
 - parent
 - parent
 - find         input[type="file"]
 - FILE         Blocksworld_AP.pddl
contains        File Blocksworld_AP.pddl successfully added.
 - assert       expected <span#client-snackbar.MuiDropzoneSnackbar-message> to be visible
contains        Upload Files
 - click
wait            5000
(fetch)         POST 200 https://planimation.planning.domains/upload/pddl
(page load)     --page loaded--
(new url)       http://localhost:3000/demo
get             canvas
 - assert       expected <canvas> to be visible
```

http://localhost:3000/
1000 x 660 (100%)
Planimation
PROBLEM
VFG
USER MANUAL
DEMO
HOMEPAGE
Planimation
Quickly build an visualisation animation form problem or VFG file shows the plan and subplan for each problem.
Problem
Build visualisation from problem
EXPLORE
VFG
Build visualisation from VFG
EXPLORE
User Manual
Source document and user manual
EXPLORE
Demo
Demo File for visualisation
EXPLORE
DOM Snapshot

The tester Felipe Ramos Morales is assigned by QA leader Bojing Zhou

System Testing Form

Date:	**Tester:**

15 Oct 2021			Felipe Ramos Morales
Environment:			
Windows 10, Firefox			
Severity	**Problem details**		**Fix before**
P0	None		17 Oct 2021
P1	✓ Didn't have animation between the initial stage and the first step Ziqi Meng		17 Oct 2021
P2	✓ Border styles of components in page2 XIAOYU ZHANG		17 Oct 2021
	✓ Subgoals with long text for display Ziqi Meng		17 Oct 2021
Notes: 18 Oct 2021 Problems are solved. Has passed all the tests Felipe Ramos Morales			

User Acceptance Testing

This testing is similar to the one in sprint 1.

This test was conducted before the client meeting at the end of the sprint two on 18 Oct 2021

Resouces

The display of pages may vary in different operating systems and browsers, so in the user acceptance testing the team tested the system in different resources.

- Operating Systems:
  - OS X: 11.5
  - Windows 10
  - Linux: Ubuntu
- Browsers:
  - Chrome
  - Firefox
  - Safari

Test documentation

All test cases are documented within Confluence Testing Document.

Error Reporting

Failures or bugs are directly reported to the responsible developer and are fixed immediately.

User Acceptance Testing Form

Roles & Responsibilities		
Name	**Role**	**Responsibilities**
Bojing Zhou	QA leader	Test documentation work
Felipe Ramos Morales	Tester	Testing on Ubuntu, Firefox
Shiqi ZHANG	Scrum Master	Managing UAT test
XIAOYU ZHANG	Tester	Testing on OS X, Safari
Ziqi Meng	Tester	Testing on Windows, Chrome

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

TC04-1	Passed	Felipe Ramos Morales	
TC05-1	Passed	Felipe Ramos Morales	
TC11-1	Passed	Ziqi Meng	
TC12-1	Passed	Ziqi Meng	
TC13-1	Passed	Xiaoyu Zhang	
TC13-2	Passed	Xiaoyu Zhang	
TC15-1	Passed	Felipe Ramos Morales	

Signature		
Name	**Signature**	**Date**
Shiqi Zhang	Shiqi Zhang	18/10/2021
Xiaoyu Zhang	[illegible]	18/10/2021
Bojing Zhou	Bojing Zhou	18/10/2021
Felipe Ramos Morales	[illegible]	18/10/2021

Ziqi Meng	Ziqi Meng	18/10/2021