



Manual and automated testing for BPM platform

Partner: TaskTrain

Website: <https://www.tasktrain.app/>

Project description: Students conducted exploratory testing, reported bugs, and built manual and automated test cases for a Business Process Management platform. Students also improved test reliability by refactoring automation and setting up CI/CD pipelines.

Screenshot of the TaskTrain home page.



PRODUCTPRICINGPROCEDURESPOSTS

TRY FREESIGN IN

New Feature AI-Powered Workflows

Does your team ignore important SOPs?

Get your team back on track by integrating your standard operating procedures into everyday workflow as actionable assignments.

SCHEDULE DEMO →WATCH DEMO

✔ No credit card required

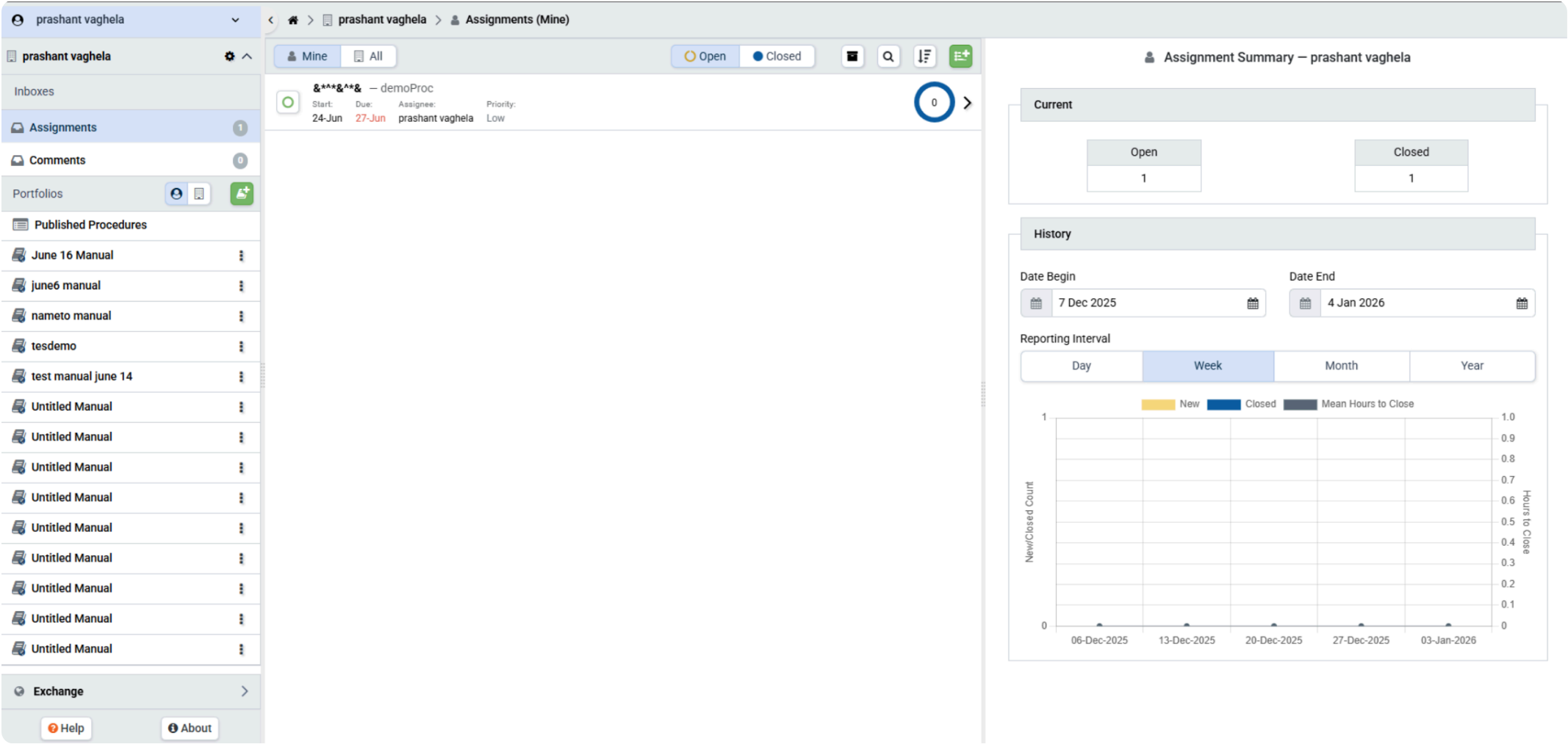
✔ Free Forever



How It Works

TaskTrain ensures your processes get put into practice.

Screenshot of a landing page of the TaskTrain application.



During the externship, engineers used tags to categorize tests.

```
// TESTS BEGIN
test.describe("Feature: Account Update Display Name", () => {
  test.describe("Scenario: Update Display Name", () => {
    test("Change display name to Arthur Dent @regression", async ({ page, navOrganization, organizationSettings }) => {
      // Click on Organization Settings button
      await navOrganization.manageOrganizationButton.click({ force: true });
      // Change Display Name to 'Arthur Dent'
      await organizationSettings.updateDisplayName("Arthur Dent");
      // Assert Display Name has been changed to Arthur Dent
      await expect(organizationSettings.displayNameInput).toHaveValue("Arthur Dent");
    });
  });
});
```

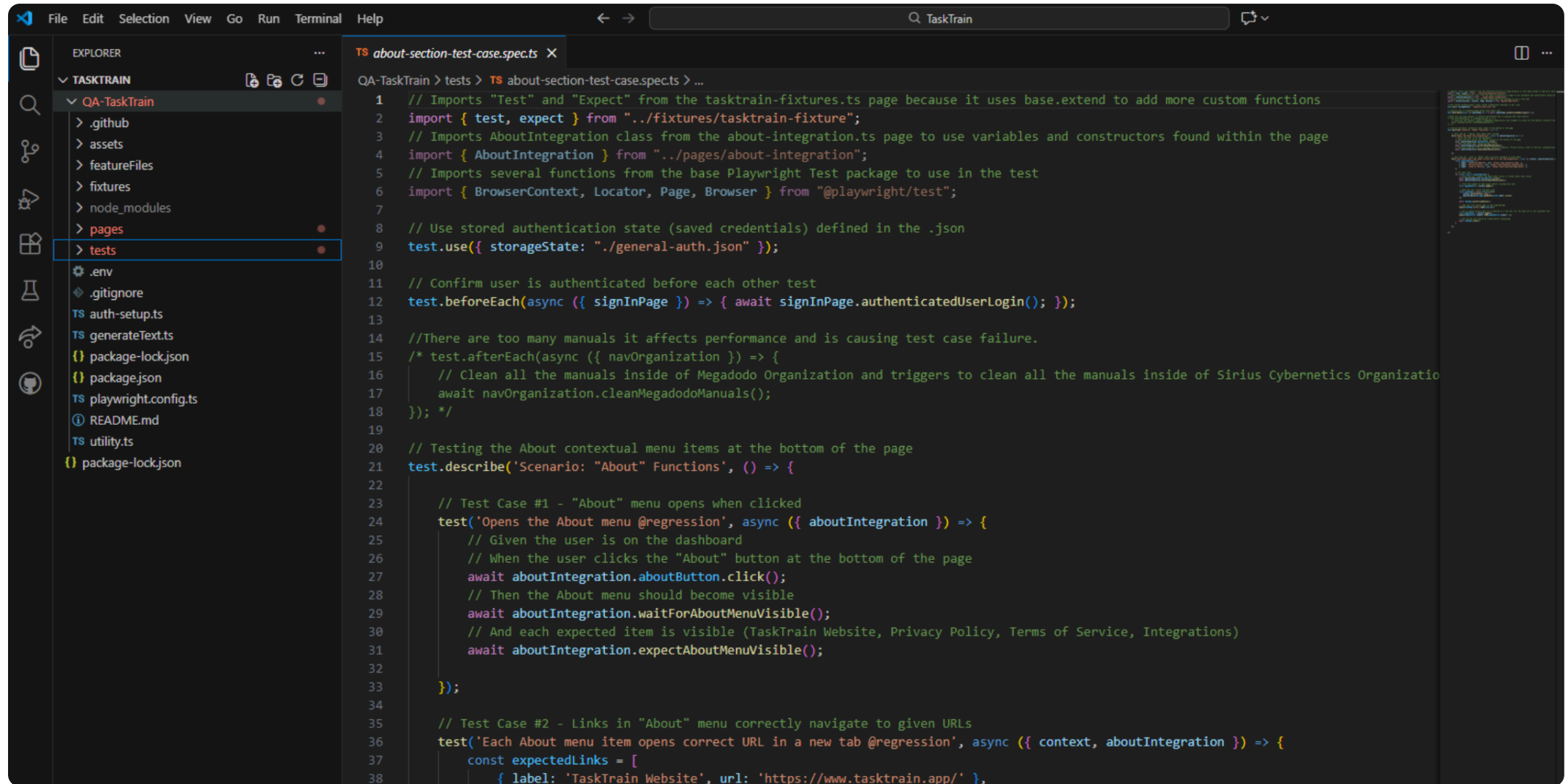
During the externship, engineers used tags to categorize tests.

```
    });  
    //smoke tests  
    test.describe('Feature: Assignment', () => {  
      test('Create Assignment @smoke @regression', async ({ page, navOrganization }, testInfo) => {  
        testInfo.annotations.push({ type: 'tag', description: 'smoke' });  
  
        await navOrganization.getProcedureName.click();  
        await navOrganization.actionMenu.click();  
        await navOrganization.procedurePublishButton.click();  
  
        await navOrganization.assignButton.click();  
        await navOrganization.assignmentName.fill('assignName');  
        await navOrganization.procedureAssignment();  
        await navOrganization.assignmentAssignButton.click();  
  
        await expect(page.getByText('Procedure assigned')).toBeVisible();  
      });  
    });
```

Engineers learned the value of abstracting reused code into Page Object Model classes or other data classes used to manipulate data.

```
TS dashboard-page.ts X
QA-TaskTrain > pages > TS dashboard-page.ts > ...
1  import { Locator, Page, expect } from "@playwright/test";
2
3  export class DashboardPage {
4
5      //Variables
6      readonly page: Page;
7      readonly homeIcon: Locator;
8      readonly releasesSection: Locator;
9      //Constructor
10     constructor(page: Page){
11         this.page = page;
12         this.homeIcon = page.locator("//li[@id='Home']/a[1]/span[1]");
13         this.releasesSection = page.locator('p-accordion-header.p-accordionheader:has-text("Releases")');
14     }
15
16     async clickHomeIcon() {
17         await this.homeIcon.click({force : true});
18     }
19
20     async releaseSection() {
21         await this.releasesSection.click({force : true});
22     }
23
24     async validateReleasesText(expectedText: string) {
25         const locator = this.page.locator('div.tt-article-title[title^="Release"]');
26         const actualText = await locator.textContent();
27         expect(actualText?.trim()).toBe(expectedText.trim());
28     }
29
30     async validateAnnouncementsText(expectedText: string) {
31         const actualText = await this.page
32             .getByRole('button', { name: 'Announcements' })
33             .textContent();
34         expect(actualText?.trim()).toBe(expectedText.trim());
35     }
36 }
37
38
```

TripleTen engineers used Visual Studio Code to write their automated tests and Page Object Model files.



The image shows a screenshot of the Visual Studio Code editor interface. On the left, the Explorer sidebar displays the file structure of a project named 'TASKTRAIN'. The 'tests' directory is selected, showing files like '.env', '.gitignore', 'auth-setup.ts', 'generateText.ts', 'package-lock.json', 'package.json', 'playwright.config.ts', 'README.md', 'utility.ts', and 'package-lock.json'. The main editor area displays the content of 'about-section-test-case.spec.ts'. The code is written in TypeScript and uses Playwright for testing. It includes imports for 'test' and 'expect' from 'tasktrain-fixtures.ts', and various Playwright classes like 'AboutIntegration', 'BrowserContext', 'Locator', 'Page', and 'Browser'. The test suite is defined with 'test.describe' and 'test' functions. The first test case, 'Opens the About menu @regression', checks if the 'About' menu opens when clicked. The second test case, 'Each About menu item opens correct URL in a new tab @regression', checks if clicking on menu items navigates to the correct URLs. The code is well-commented and uses async/await for asynchronous operations.

```
1 // Imports "Test" and "Expect" from the tasktrain-fixtures.ts page because it uses base.extend to add more custom functions
2 import { test, expect } from "../fixtures/tasktrain-fixture";
3 // Imports AboutIntegration class from the about-integration.ts page to use variables and constructors found within the page
4 import { AboutIntegration } from "../pages/about-integration";
5 // Imports several functions from the base Playwright Test package to use in the test
6 import { BrowserContext, Locator, Page, Browser } from "@playwright/test";
7
8 // Use stored authentication state (saved credentials) defined in the .json
9 test.use({ storageState: "../general-auth.json" });
10
11 // Confirm user is authenticated before each other test
12 test.beforeEach(async ({ signInPage }) => { await signInPage.authenticatedUserLogin(); });
13
14 //There are too many manuals it affects performance and is causing test case failure.
15 /* test.afterEach(async ({ navOrganization }) => {
16     // Clean all the manuals inside of Megadodo Organization and triggers to clean all the manuals inside of Sirius Cybernetics Organization
17     await navOrganization.cleanMegadodoManuals();
18 }); */
19
20 // Testing the About contextual menu items at the bottom of the page
21 test.describe('Scenario: "About" Functions', () => {
22
23     // Test Case #1 - "About" menu opens when clicked
24     test('Opens the About menu @regression', async ({ aboutIntegration }) => {
25         // Given the user is on the dashboard
26         // When the user clicks the "About" button at the bottom of the page
27         await aboutIntegration.aboutButton.click();
28         // Then the About menu should become visible
29         await aboutIntegration.waitForAboutMenuVisible();
30         // And each expected item is visible (TaskTrain Website, Privacy Policy, Terms of Service, Integrations)
31         await aboutIntegration.expectAboutMenuVisible();
32     });
33
34     // Test Case #2 - Links in "About" menu correctly navigate to given URLs
35     test('Each About menu item opens correct URL in a new tab @regression', async ({ context, aboutIntegration }) => {
36         const expectedLinks = [
37             { label: 'TaskTrain Website', url: 'https://www.tasktrain.app/' },
38             { label: 'Privacy Policy', url: 'https://www.tasktrain.app/privacy-policy' },
39             { label: 'Terms of Service', url: 'https://www.tasktrain.app/terms-of-service' },
40             { label: 'Integrations', url: 'https://www.tasktrain.app/integrations' }
41         ];
42         for (const link of expectedLinks) {
43             await aboutIntegration.clickLink(link.label);
44             await context.waitForURL(link.url);
45             expect(context.url(), `URL is ${link.url}`).toBe(link.url);
46         }
47     });
48 }
```

Configured CI/CD pipeline to execute Smoke tests on every Pull Request.



**Some checks were not successful**

2 failing checks



Smoke Tests on PR / test (pull_request) Failing after 4m

Smoke Tests on PR / test (push) Failing after 4m

**No conflicts with base branch**

Merging can be performed automatically.

Merge pull request

You can also merge this with the command line. [View command line instructions.](#)