Test Report

1. Purpose

This document explains the various activities performed as part of testing of the Planiwiki website.

2. Application Overview

Planiwiki is an educational website targeted at Planimation users, which aims to make their learning experience of Planimation smoother and more structured. Its main features encompass:

- A centralized, up-to-date collection of **documentations**, for the functions and properties used for animation profiling in Planimation;
- A **gallery** of implemented **animations** for various domains, with an embedded PDDL Editor;
- Direction of users to **submitting formatted issues** on possible extensions of Planimation;
- A list of **external resources** for learning about PDDL and Planimation.

Each webpage can be viewed as a module, which are integrated to fulfil the purpose of the website. These modules include the Home, Documentations, Gallery, Suggestions, Reference, and 404 page.

3. Testing Scope

a) In Scope

Functional Testing for the following modules are in scope of testing:

- Home
- Documentations
- Gallery
- Suggestions
- Reference
- 404 page

b) Out of Scope / Not tested

- Performance Testing.
- Website's display on zooming in and out, and on different screen sizes.

4. Metrics

The metrics used for testing are whether the various workflows set up are passed or not.

Test	**Passed**
GitHub Action: "Deployment to Firebase Hosting" workflow	Yes
GitHub Action: "Frontend and Backend Testing" workflow	Yes
GitHub Action: "CodeQL" workflow	Yes

More detailed testing results can be found at:

- Frontend testing: Front-end Elaborated Test Cases (Sprint 2+)
- Backend testing: Backend Elaborated Test Cases (Sprint 2+)

5. Types of testing performed

This section is largely the same as the "Testing" section in the product report.

Frontend testing

Frontend testing mainly focused on button and link redirections, as well as the proper rendering of elements on the webpage.

Feature tested	Expected results
Button Redirection: • Home: Explore domains • Documentation: View original GitHub documentation • Function: Return • Gallery: View source code of animation profiles • Domain: Return, View Source Code	Successfully redirected to another intended webpage upon clicking on the button that has a valid link
Element display: • Buttons • Links • GIFs • Navigation bar • Introductory banner on Homepage • Guide on Homepage • Search bar • Animation guide collapsible	Accurately displayed on the web page with accurate size, location and displayed text if any
Layout:	Accurate layout

• Documentation: tables for functions & visual properties • Gallery: domain blocks	
Link redirection: • Documentation: tables for functions & visual properties • Gallery: domain blocks	Successfully redirected to another intended webpage upon clicking on the anchor element (with a valid link)
Search bar on Documentation and Gallery page	Clickable and typable
Animation guide collapsible	Can be opened and collapsed upon clicking
PDDL editor embedding on domain pages	The PDDL editor loads

Database security testing

Planiwiki does not allow any form of user input to the database, hence write access is not allowed for all users. Contributors who are owners of this Firebase project are able to modify the database, via the Firebase console or an admin SDK. Additionally, only contents to be displayed on the website are stored in the database, but not any private/sensitive information. The contents are then dynamically retrieved when a user accesses Planiwiki.

In the automated database security testing, we applied the same security rules from the Development Firestore database to the Emulator Firestore database, and ensured that:

- Read requests by a non-owner user are permitted; and
- Write requests by a non-owner user are rejected.

Integration testing

Integration testing has focused on webpage contents, search functionality, and 404 pages for incorrect URLs. In particular, we focused on the communication between frontend and backend, ensuring that data stored in the database is correctly mapped to the information displayed on the website.

Feature tested	Expected results
Search functionality (successful) • Documentation page • Gallery page	Functions and domains that are stored in the database remain on the page (under the correct section) upon entering their: • Full name • Partial name • Capitalized name • Lowercase name
Search functionality (unsuccessful) • Documentation page • Gallery page	Upon searching for non-existing functions or domains, no records will be listed on the page (for the relevant section).
Functions on Documentation page	Functions stored in the database are displayed on the Documentation page, under the correct section.
Visual properties on Documentation page	Visual properties stored in the database are displayed on the Documentation page, under the correct section.

Domains on Gallery page	Domains stored in the database are displayed on the Gallery page.
Domain details (successful)	Upon clicking into the URL of a domain (stored in the database), a page with the correct name, embedding link, and description, is displayed.
Function details (successful)	Upon clicking into the URL of a function (stored in the database), a page with the correct function name, description, YouTube video demo and optional parameters, is displayed. If the function is not functioning, a warning is displayed.
404 page: • Function • Domain	Upon navigation to the URL of a non-existing function or domain, users are redirected to the 404 error page.

6. Test Environment & Tools

- Browser: Google Chrome
- Server: Firebase Emulator Suite
- Database: Firestore
- Frameworks: Playwright (frontend and backend), JEST (backend)
- Website URL: PlaniWiki

7. Lessons Learnt

S. No.	Issues Faced	Solutions
1	The initial candidate frontend testing tool, Selenium Webdriver, was somehow difficult to set up.	We switched to using Playwright after more research and learning.
2	Initially, the frontend and backend test scripts had to be executed manually.	We set up a GitHub Action workflow to automate the test runs, which are triggered on every pull request.
3	There were lots of functions and domains, each having its individual page. Testing all at once would be too time-consuming.	We implemented a randomized selection mechanism, which chooses an individual domain/function’s page to check during each test run.
4	There were lots of repeated test operations (e.g. testing for button redirections).	We created helper test functions (e.g. in `functions/frontend-test/playwright-test-util.js`) to reduce repetitions and perform unit tests.
5	The precise locations where elements are displayed (e.g. whether some text is aligned) are difficult to be tested in code.	We covered these minor cases via manual testing.
6	Test data is different to what’s stored in the production database, but modification to the	We make use of the Firestore emulator, such that test data can be written into the

	production database is to be avoided.	emulated Firestore, rather than the production database.
7	Writing test data into the emulated Firestore using Admin-SDK is complicated.	We import the test data into the Firestore emulator when starting the emulator, which avoids the complexity related to the usage of admin-sdk.

8. Recommendations

- The website is best viewed on Google Chrome or Safari, on a standard screen size of 100%.

9. Best Practices

A repetitive task done manually every time was time consuming. This task was automated by creating scripts and run each time, which saved time and resources.

- Repeated unit tests (e.g. button redirection) were written as helper functions, which can be simply called in a larger suite of tests, on different elements.
- The frontend and backend test suites were automated as GitHub Action workflows, which ran fast and saved time.

10. Exit Criteria

1. All test cases should be executed – Yes
2. All GitHub Action workflows should be passed – Yes.
3. All manual tests should be passed – Yes.

11. Conclusion/Sign Off

As the Exit criteria was met and satisfied as mentioned in Section 10, this application is suggested to 'Go Live' by the Testing team.

Appropriate User/Business acceptance testing and client approval should also be attained before 'Go Live'.

12. Definitions, Acronyms, and Abbreviations

Planimation: A modular and extensible open source framework to visualise sequential solutions of planning problems specified in PDDL. This is the client's project.