Verification and Validation Report: SFWRENG 4G06 - Capstone Design Process

Team 17, DomainX

Awurama Nyarko
Haniye Hamidizadeh
Fei Xie
Ghena Hatoum

March 1, 2026

1 Revision History

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

2 Symbols, Abbreviations and Acronyms

symbol	description
T	Test

[symbols, abbreviations or acronyms – you can reference the SRS tables if needed —SS]

Contents

List of Tables

List of Figures

This document ...

3 Functional Requirements Evaluation

4 Nonfunctional Requirements Evaluation

4.1 Usability

4.2 Performance

4.3 etc.

5 Unit Testing

The following unit tests concern behaviour-hiding modules that are implemented by the tool. Modules implemented by external libraries are assumed to be tested themselves and therefore don't need to be unit tested.

5.1 Data Edit Module

5.2 User Authentication Module

5.3 User Role Access Module

5.4 User Page Module

5.5 Automated Metrics Module

5.6 Domains Page Module

1. `test_create_domain_success`
   Initial State: Create user and fixture of API client
   Input: Create payload containing domain name, description, and `creator_ids` with the creator and call POST
   Expected Output: Response returns status 201 for created, and querying for the domain object contains the same as in the payload
   Actual Output: Matches expected
   Result: Pass

2. `test_create_domain_missing_domain_name`
   Initial State: Create fixture of API client
   Input: Create payload with only description and call POST
   Expected Output: Response returns 400 bad request and informs of missing field
   Actual Output: Matches expected, data returns dictionary with `domain_name` and value of "This field is required"
   Result: Pass

3. `test_create_domain_missing_description`
   Initial State: Create fixture of API client
   Input: Create payload with only domain name and call POST
   Response returns 400 bad request and informs of missing field
   Actual Output: Matches expected, data returns dictionary with "description" and value of "Description field is required"
   Result: Pass

4. `test_list_domains`
   Initial State: Create fixture of API client and create two domain objects
   Input: Calling GET for domain
   Expected Output: Response 200 OK with list of domains matching the created domains
   Actual Output: Matches expected
   Result: Pass

5. `test_update_domain_sets_creators`
   Initial State: Create domain object with only `domain_name` and description and fixture of API client
   Input: Create two users and add them to a payload with updated domain name and description. Then call POST with the domain ID
   Expected Output: Response 200, domain should contain the new information and be linked to the two users
   Actual Output: Matches expected
   Result: Pass

6. `test_delete_domain`
   Initial State: Create fixture of API client and create a domain object
   Input: Call DELETE with the domain ID
   Expected Output: Querying for the domain ID returns None

Actual Output: Matches Expected
Result: Pass

5.7 Comparison Module

5.8 Configuration Module

5.9 Ranking Algorithm Module

5.10 Database Persistence Module

1. `test_domain_str_returns_name`
   Initial State: N/A
   Input: Directly creating a new domain object with `domain_name` "Alpha" and description "desc"
   Expected Output: Calling newly created object returns the domain name "Alpha"
   Actual Output: Returned "Alpha"
   Result: Pass

2. `test_get_domain_ID_returns_string_uuid`
   Initial State: N/A
   Input: Directly creating a new domain object with `domain_name` "Beta" and description "desc"
   Expected Output: Calling `get_domain_ID()` on new domain returns the UUID
   Actual Output: Method matches `domain_id` parameter and is instance of string
   Result: Pass

3. `test_defaults_and_blank_fields`
   Initial State: N/A
   Input: Directly creating a new domain object with `domain_name` "Gamma"
   Expected Output: Initializes domain object with default fields `published: False, paper_name: None, paper_url: "", category_weights:`
   Actual Output: Output matches expected
   Result: Pass

4. `test_unique_domain_name_constraint`
   Initial State: Domain object with `domain_name` "Delta" and descrip-

tion "one"
Input: Directly creating a new domain object with `domain_name` "Delta" and description "two"
Expected Output: Throws integrity error due to duplicate domain names
Actual Output: Matches expected
Result: Pass

5. `test_creators_many_to_many_assignment`
Initial State: Create two users with admin role
Input: Create two domain objects with separate names, add both creators to each domain.
Expected Output: Filtering in each domain's creators by the id of the creator, both creators should exist
Actual Output: Matches expected
Result: Pass

6 Changes Due to Testing

[This section should highlight how feedback from the users and from the supervisor (when one exists) shaped the final product. In particular the feedback from the Rev 0 demo to the supervisor (or to potential users) should be highlighted. —SS]

7 Automated Testing

8 Trace to Requirements

9 Trace to Modules

10 Code Coverage Metrics

Appendix — Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Reflection.

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

1. What went well while writing this deliverable?

2. What pain points did you experience during this deliverable, and how did you resolve them?

3. Which parts of this document stemmed from speaking to your client(s) or a proxy (e.g. your peers)? Which ones were not, and why?

4. In what ways was the Verification and Validation (VnV) Plan different from the activities that were actually conducted for VnV? If there were differences, what changes required the modification in the plan? Why did these changes occur? Would you be able to anticipate these changes in future projects? If there weren't any differences, how was your team able to clearly predict a feasible amount of effort and the right tasks needed to build the evidence that demonstrates the required quality? (It is expected that most teams will have had to deviate from their original VnV Plan.)