System Verification and Validation Plan for BrainInsight3D: 3D fMRI Visualization & Segmentation

Nada Elmasry

February 19, 2024

Revision History

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

Contents

1 Symbols, Abbreviations, and Acronyms

symbol	description
T	Test

See complete list of symbols and abbreviations at Elmasry (2024d)

2 Introduction

This document outlines the System Verification and Validation Plan (VnV) for BrainInsight3D: 3D fMRI Visualization & Segmentation project. The VnV plan outlines the verification and validation tests that need to be conducted in order for the software product to be have satisfied its requirements. Verification is done by testing individual components and multiple sub-components of the software products together to verify that they work correctly. Validation is done to verify that the finished software product meets the user needs and requirements.

the document is divided into five main parts, section3 presents the general information and objectives of the project, section4 discusses methods of testing to achieve project's objective, section5 discusses the system test descriptions including functional and nonfuctional requirements tests, and section6 discusses the unit testing of the systems components.

3 General Information

3.1 Summary

This document provides the software specification requirements(SRS) for the BrainInsight 3D project. The project offers 3D and 2D visualization and segmentation of fMRI brain scans with time and functional activity tracking in 2D through a web application that enables high-resolution visualization and segmentation.

3.2 Objectives

the objective of this document is to build confidience in the software correctness and usability by producing results comparable to current solutions with an easy to use interface for better interaction flow and usability. A secondary goal of this project is to make sure the project is extensible, this can be achieved by ensuring code quality and maintanibility to enable extensibility by multiple developers. Parts of the code will rely on external libraries for operations like image input and preprocessing, and bulding and hosting the user interface as a web application. While it would be beneficial to verify the correctness of these libraries, this is out of the scope of the project and

it will be assumed these external libraries will have been verified by their implementation teams.

3.3 Relevant Documentation

There exists multiple documents which are crucial for in-depth understanding of the software being tested. The documents are as follows:

- Problem Statement Elmasry (2024c): states the problem the project is trying to solve and introduces the project's features.
- Software Requirements Specification (SRS) Document Elmasry (2024d): States the assumptions, theoritical and implementation models, functional and nonfunctional requirements, and terminology that will be used throughout the project documentation
- Module Guide (MG) Document Elmasry (2024a): states the functionality and responsibilities of each module in the project.
- Module Interface Specification Document Elmasry (2024b): States the functional design of each module in the project.

4 Plan

This section includes the verification detailed plan for the documentation and software of the Braininsight3D project. Section 4.1 introduces the verification and validation team responsible for through implementation and validation of all aspects of the project's documentation and software components, sections 4.2, 4.3, 4.4 discuss the verification plan for the project's SRS, design, and VnV plans respectively. Sections 4.5, 4.6, and 4.7 discuss the software implementation verification plan, automated testing and verification tools, and software validation plan respectively. The primary aim of this section is to provide a complete description of the plans which will be used to ensure the software abides to its requirements and achieves its objectives.

4.1 Verification and Validation Team

Table 4.1 lists all VnV team members and their roles in the project.

Name	Document	Role	Description
Dr. Spencer Smith	SRS, VnV Plan, MG, MIS	Instructor/Expert Reviewer	Reviews all project's documentation, design, and functionality
Nada Elmasry	SRS, Vnv Plan, MG, MIS	Author/Developer	Author of all project's documents. Main developer and tester of the software product
Yi-Leng Cheng	SRS, VnV Plan, MG, MIS	Domain Expert Reviewer	Review all project's documents.
Waqar Awan	SRS	SRS Secondary Reviewer	Reviews the SRS document.
Kim Ying Wong	VnV Plan	Secondary Reviewer	Reviews the VnV Plan document.
Morteza Mirzaei	MG, MIS	Secondary Reviewer	Reviews the MG+MIS document.

4.2 SRS Verification Plan

The SRS document will be reviewed by the instructor, the domain expert reviewer, and the secondary reviewer to ensure that the documents fulfills the outlined requirements in the SRS checklist Smith (2022). The reviewers will provide their feedback through GitHub issues made by the author for SRS review. It is the responsibility of the author to incorporate the reviewers feedback and suggestions, and to address any possible issues in the SRS documentation.

4.3 Design Verification Plan

The software design will be designed with a focus on usability, accuracy, and performance. The design documentation will include the Module Guide (MG) plan and the Module Interface Specification (MIS) plan. The MG and MIS plans will be reviewed by the instructor, the domain expert reviewer,

and the secondary reviewer to ensure that the documents fulfills the outlined requirements in the MG checklist Elmasry (2024a)and MIS checklist Elmasry (2024b). The reviewers will provide their feedback through GitHub issues made by the author for MG and MIS review. It is the responsibility of the author to incorporate the reviewers feedback and suggestions, and to address any possible issues in the design documentation.

4.4 Verification and Validation Plan Verification Plan

The VnV plan will be continously updated throughout the project implementation phase. The VnV team will check whether the documented testing and verification process have been accomplished and the software functional and nonfunctional requirements have been fulfilled. The reviewers will provide their feedback through GitHub issues made by the author for the VnV plan review. It is the responsibility of the author to incorporate the reviewers feedback and suggestions, and to address any possible issues in the VnV documentation.

4.5 Implementation Verification Plan

In this section, we outline the plan for verifying the implementation of our software system. The verification process will ensure that the software meets the specified requirements and functions as intended.

4.5.1 Dynamic Testing

We will conduct a series of dynamic tests which include unit testing and integration testing. Due to the time constraints of the project, system testing and acceptance testing will not be conducted. The unit testing plan is detailed in Section 6 of the Test Plan document, where each module of the software will be tested individually to ensure correct behavior. Integration testing will ensure that multiple modules are well integrated and can communicate and operate together correctly.

4.5.2 Static Verification

In addition to dynamic testing, we will employ various static verification techniques to examine the source code. These techniques include:

- **Code Walkthroughs:** The development team will conduct regular code walkthroughs, where the source code is reviewed to identify potential issues and ensure adherence to coding standards.
- **Code Inspections:** Formal code inspections will be carried out to systematically examine the code for defects, compliance with design and coding guidelines, and other quality attributes.
- **Static Analyzers:** We will use static analysis tools such as ESLint to automatically analyze the code for common programming errors, code smells, and to enforce coding standards.

These static verification techniques will complement our dynamic testing efforts and help ensure the reliability and quality of the software implementation.

4.6 Automated Testing and Verification Tools

The project will be maintained using CI/CD pipeline implemented using Docker and GitHub actions. GitHub Actions offers flexibility and customization when building CI/CD pipelines which is cruical for our project as it have many overlapping components. The SRS document contains an introduction to the different components of the project Elmasry (2024d). GitHub actions can run tests by comparing expected outcomes with current outcomes from the software and directly build and deploy the software after each successful update. The individual tests will be written in Pytest library.

4.7 Software Validation Plan

Software Validation Plan is out of scope for BrainInsight3D due to the time constraints and limited resources of the project.

5 System Test Description

5.1 Tests for Functional Requirements

This sections presents the planned tests for the functional requirements for BrainInsight3D stated in SRS Document Elmasry (2024d) in Section 5.1.

There are six functional requirements for BrainInsight3D, R1 is related to input, R2 to R4 are related to image processing and visualization, and R5 and R6 are related to segmentation. Section 5.1.1 presents the tests for input verification, section 5.1.2 presents the tests for image processing and visualization, and section 5.1.3 presents the tests for image segmentation.

5.1.1 Input Testing

The purpose of the tests in this section is to ensure the input scan is non-corrupt and in the correct format in order to be viable to be used in the subsequent processing steps.

Input Scan Testing

1. test-input-scan-format

   Control: Automatic

   Initial State: System waiting to receive input scan

   Input: Input Scan in NIfTI format

   Output: A message signaling that the scan was read successfully or a detailed error message with the complete error log.

   Test Case Derivation: This test is conducted when the user uploads a new scan into the software application. The test ensures that the scan is in the correct format and non-corrupt before passing it to the preprocessing and visualization steps.

   How test will be performed: The input scan will be read by a library function, the function will return an output based on whether the scan was read successfully or not. The return message will then guide the system to print the success message or the failure message with the detailed error log.

5.1.2 Preprocessing and Visualization

1. test-image-processing-slices-2d

   Control: Automatic

   Initial State: Input Scan have been uploaded successfully in the supported format.

Input: Input Scans with different orientations, noise levels, and contrast levels.

Output: Processed scan slices with adjusted contrast and brightness levels, reduced noise, and the skull and neck details removed from the scan leaving only the brain area visible (region of interest).

Test Case Derivation: This test validates a crucial step in the visualization pipeline, the image preprocessing step. Failure of this test will lead to failure of both the visualization and segmentation steps. The test validates that the input scan has successfully passed through all the preprocessing steps before the scan is visualized in 2D or 3D space, or passed to the segmentation model.

How test will be performed: The input scan will be processed by the preprocessing function performing multiple operations as image registeration and smoothing to obtain the final result for visualization. If the function fails at any of the preprocessing steps, an error message will be printed highlighting the part where the error occurred along with the error log available from that processing step.

2. test-image-visualization-volume-3d

   Control: Automatic

   Initial State: Input scans have successfully passed through the preprocessing pipeline.

   Input: Preprocessed input scans.

   Output: Reconstructed 3D volume of the input scan 2D slices.

   Test Case Derivation: This test validates that the 3D volume has been successfully reconstructed from the 2D slices for visualization in the web interface.

   How test will be performed: The preprocessed input scans will be used as input to the 3D reconstruction function that will return the 3D volume if the reconstruction was successful, otherwise it will return an error log explaining where exactly in the reconstruction step did the error occur.

5.1.3 Area of Testing3

1. test-image-segmentation-model-load

Control: Automatic

Initial State: System waiting for segmentation model to load

Input: Segmentation Model file

Output: A message signaling if the segmentation model has been loaded successfully or an error message with the exact error log explaining where and why the error occurred.

Test Case Derivation: This test ensures that the segmentation model is successfully loaded and ready to receive input slices to perform the segmentation process.

How test will be performed: The model file will be uploaded in .h5 format which will be read by the machine learning library(PyTorch) and return a message indicating the success or failure of loading the model file.

2. test-image-segmentation-mask-result

   Control: Automatic

   Initial State: Segmentation model ready to receive input slice.

   Input: Processed slice image.

   Output: Mask image with the segmented brain areas.

   Test Case Derivation: This test case tests the performance of the segmentation model when receiving a preprocessed brain image slice for segmentation of the functional brain areas. The model should output a gray mask image with the segmented brain areas.

   How test will be performed: the user will upload the brain scans which will be read and processed then passed automatically by an automated script to the segmentation model. If the test is successful, the output of the test will be merged with the brain slice original image and visualized in the user interface.

...

5.2 Tests for Nonfunctional Requirements

This sections presents the planned tests for the functional requirements for BrainInsight3D stated in SRS Document Elmasry (2024d) in Section 5.2.

5.2.1 Accuracy Testing

1. test-id1

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input/Condition:

   Output/Result:

   How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

5.2.2 Usability Testing

test-id2

Type: Functional, Dynamic, Manual, Static etc.
Initial State:
Input:
Output:
How test will be performed:

5.2.3 Maintability Testing

5.2.4 Portability Testing

...

5.3 Traceability Between Test Cases and Requirements

[Provide a table that shows which test cases are supporting which requirements. —SS]

6 Unit Test Description

[This section should not be filled in until after the MIS (detailed design document) has been completed. —SS]

[Reference your MIS (detailed design document) and explain your overall philosophy for test case selection. —SS]

[To save space and time, it may be an option to provide less detail in this section. For the unit tests you can potentially layout your testing strategy here. That is, you can explain how tests will be selected for each module. For instance, your test building approach could be test cases for each access program, including one test for normal behaviour and as many tests as needed for edge cases. Rather than create the details of the input and output here, you could point to the unit testing code. For this to work, you code needs to be well-documented, with meaningful names for all of the tests. —SS]

6.1 Unit Testing Scope

[What modules are outside of the scope. If there are modules that are developed by someone else, then you would say here if you aren't planning on verifying them. There may also be modules that are part of your software, but have a lower priority for verification than others. If this is the case, explain your rationale for the ranking of module importance. —SS]

6.2 Tests for Functional Requirements

[Most of the verification will be through automated unit testing. If appropriate specific modules can be verified by a non-testing based technique. That can also be documented in this section. —SS]

6.2.1 Module 1

[Include a blurb here to explain why the subsections below cover the module. References to the MIS would be good. You will want tests from a black box

perspective and from a white box perspective. Explain to the reader how the tests were selected. —SS]

1. test-id1

   Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

   Initial State:

   Input:

   Output: [The expected result for the given inputs —SS]

   Test Case Derivation: [Justify the expected value given in the Output field —SS]

   How test will be performed:

2. test-id2

   Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

   Initial State:

   Input:

   Output: [The expected result for the given inputs —SS]

   Test Case Derivation: [Justify the expected value given in the Output field —SS]

   How test will be performed:

3. ...

6.2.2 Module 2

...

6.3 Tests for Nonfunctional Requirements

[If there is a module that needs to be independently assessed for performance, those test cases can go here. In some projects, planning for nonfunctional tests of units will not be that relevant. —SS]

[These tests may involve collecting performance data from previously mentioned functional tests. —SS]

6.3.1 Module ?

1. test-id1

   Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

   Initial State:

   Input/Condition:

   Output/Result:

   How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

6.3.2 Module ?

...

6.4 Traceability Between Test Cases and Modules

[Provide evidence that all of the modules have been considered. —SS]

References

Nada Elmasry. Braininsight3d module guide document, 2024a. URL https://github.com/NadaElmasry/3D-Image-Reconstruction-Segmentation/blob/main/docs/Design/SoftArchitecture/MG.pdf.

Nada Elmasry. Braininsight3d module interface specification document, 2024b. URL https://github.com/NadaElmasry/3D-Image-Reconstruction-Segmentation/blob/main/docs/Design/SoftDetailedDes/MIS.pdf.

Nada Elmasry. Braininsight3d problem statement, 2024c. URL https://github.com/NadaElmasry/3D-Image-Reconstruction-Segmentation/blob/main/docs/ProblemStatementAndGoals/ProblemStatement.pdf.

Nada Elmasry. Braininsight3d srs document, 2024d. URL https://github.com/NadaElmasry/3D-Image-Reconstruction-Segmentation/blob/main/docs/SRS/SRS.pdf.

Spencer Smith. Software requirements specifications checklist, 2022. URL https://github.com/smiths/capTemplate/blob/main/docs/Checklists/VnV-Checklist.pdf.

7 Appendix

This is where you can place additional information.

7.1 Symbolic Parameters

The definition of the test cases will call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

7.2 Usability Survey Questions?

[This is a section that would be appropriate for some projects. —SS]

Appendix — Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Lifelong Learning. Please answer the following questions:

Appendix — Reflection

[This section is not required for CAS 741 —SS]

The information in this section will be used to evaluate the team members on the graduate attribute of Lifelong Learning. Please answer the following questions:

1. What knowledge and skills will the team collectively need to acquire to successfully complete the verification and validation of your project? Examples of possible knowledge and skills include dynamic testing knowledge, static testing knowledge, specific tool usage etc. You should look to identify at least one item for each team member.

2. For each of the knowledge areas and skills identified in the previous question, what are at least two approaches to acquiring the knowledge or mastering the skill? Of the identified approaches, which will each team member pursue, and why did they make this choice?