System Verification and Validation Plan for Software Engineering

Team 8 – Rhythm Rangers

Ansel Chen
Muhammad Jawad
Mohamad-Hassan Bahsoun
Matthew Baleanu
Ahmed Al-Hayali

November 4, 2024

Revision History

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

[The intention of the VnV plan is to increase confidence in the software. However, this does not mean listing every verification and validation technique that has ever been devised. The VnV plan should also be a **feasible** plan. Execution of the plan should be possible with the time and team available. If the full plan cannot be completed during the time available, it can either be modified to "fake it", or a better solution is to add a section describing what work has been completed and what work is still planned for the future. —SS]

[The VnV plan is typically started after the requirements stage, but before the design stage. This means that the sections related to unit testing cannot initially be completed. The sections will be filled in after the design stage is complete. the final version of the VnV plan should have all sections filled in. —SS]

Contents

List of Tables

[Remove this section if it isn't needed —SS]

List of Figures

[Remove this section if it isn't needed —SS]

iii

1 Symbols, Abbreviations, and Acronyms

symbol	description
T	Test

[symbols, abbreviations, or acronyms — you can simply reference the SRS (Author, 2019) tables, if appropriate —SS]

[Remove this section if it isn't needed —SS]

This document ... [provide an introductory blurb and roadmap of the Verification and Validation plan —SS]

2 General Information

2.1 Summary

[Say what software is being tested. Give its name and a brief overview of its general functions. —SS]

2.2 Objectives

[State what is intended to be accomplished. The objective will be around the qualities that are most important for your project. You might have something like: "build confidence in the software correctness," "demonstrate adequate usability." etc. You won't list all of the qualities, just those that are most important. —SS]

[You should also list the objectives that are out of scope. You don't have the resources to do everything, so what will you be leaving out. For instance, if you are not going to verify the quality of usability, state this. It is also worthwhile to justify why the objectives are left out. —SS]

[The objectives are important because they highlight that you are aware of limitations in your resources for verification and validation. You can't do everything, so what are you going to prioritize? As an example, if your system depends on an external library, you can explicitly state that you will assume that external library has already been verified by its implementation team. —SS]

2.3 Challenge Level and Extras

[State the challenge level (advanced, general, basic) for your project. Your challenge level should exactly match what is included in your problem statement. This should be the challenge level agreed on between you and the course instructor. You can use a pull request to update your challenge level (in TeamComposition.csv or Repos.csv) if your plan changes as a result of the VnV planning exercise. —SS]

[Summarize the extras (if any) that were tackled by this project. Extras can include usability testing, code walkthroughs, user documentation, formal proof, GenderMag personas, Design Thinking, etc. Extras should have already been approved by the course instructor as included in your problem statement. You can use a pull request to update your extras (in TeamComposition.csv or Repos.csv) if your plan changes as a result of the VnV planning exercise. —SS]

2.4   Relevant Documentation

[Reference relevant documentation. This will definitely include your SRS and your other project documents (design documents, like MG, MIS, etc). You can include these even before they are written, since by the time the project is done, they will be written. You can create BibTeX entries for your documents and within those entries include a hyperlink to the documents. —SS]

Author (2019)

[Don't just list the other documents. You should explain why they are relevant and how they relate to your VnV efforts. —SS]

3   Plan

[Introduce this section. You can provide a roadmap of the sections to come. —SS]

3.1   Verification and Validation Team

[Your teammates. Maybe your supervisor. You should do more than list names. You should say what each person's role is for the project's verification. A table is a good way to summarize this information. —SS]

3.2   SRS Verification Plan

The following approaches will be used for SRS verification:

- Formal reviews with the supervisor

- A checklist that will be given to the supervisor and any peer reviewers. It will also serve as a guide for the developers of the system

- Using feedback from grading to create new checklists and update existing checklists

- Ad-hoc reviews from peers and other teams in the course

This is the initial SRS checklist that reviewers will use. It will be updated as reviews are performed:

- ☐ Does each functional requirement have a detailed and accurate description, rationale and fit criteria?

- ☐ Is each requirement (both functional and non-functional) relevant and necessary?

- ☐ Are all functional requirements traceable to at least one use case?

- ☐ Are all fit critera unambiguous and verifiable?

- ☐ Have all issues opened by reviewers been closed?

3.3   Design Verification Plan

[Plans for design verification —SS]
    [The review will include reviews by your classmates —SS]
    [Create a checklists? —SS]

3.4   Verification and Validation Plan Verification Plan

[The verification and validation plan is an artifact that should also be verified. Techniques for this include review and mutation testing. —SS]
    [The review will include reviews by your classmates —SS]
    [Create a checklists? —SS]

3.5 Implementation Verification Plan

[You should at least point to the tests listed in this document and the unit testing plan. —SS]

[In this section you would also give any details of any plans for static verification of the implementation. Potential techniques include code walkthroughs, code inspection, static analyzers, etc. —SS]

[The final class presentation in CAS 741 could be used as a code walkthrough. There is also a possibility of using the final presentation (in CAS741) for a partial usability survey. —SS]

3.6 Automated Testing and Verification Tools

[What tools are you using for automated testing. Likely a unit testing framework and maybe a profiling tool, like ValGrind. Other possible tools include a static analyzer, make, continuous integration tools, test coverage tools, etc. Explain your plans for summarizing code coverage metrics. Linters are another important class of tools. For the programming language you select, you should look at the available linters. There may also be tools that verify that coding standards have been respected, like flake9 for Python. —SS]

[If you have already done this in the development plan, you can point to that document. —SS]

[The details of this section will likely evolve as you get closer to the implementation. —SS]

3.7 Software Validation Plan

[If there is any external data that can be used for validation, you should point to it here. If there are no plans for validation, you should state that here. —SS]

[You might want to use review sessions with the stakeholder to check that the requirements document captures the right requirements. Maybe task based inspection? —SS]

[For those capstone teams with an external supervisor, the Rev 0 demo should be used as an opportunity to validate the requirements. You should plan on demonstrating your project to your supervisor shortly after the scheduled Rev 0 demo. The feedback from your supervisor will be very useful for improving your project. —SS]

4

[For teams without an external supervisor, user testing can serve the same purpose as a Rev 0 demo for the supervisor. —SS]

[This section might reference back to the SRS verification section. —SS]

4   System Tests

[There should be text between all headings, even if it is just a roadmap of the contents of the subsections. —SS]

4.1   Tests for Functional Requirements

[Subsets of the tests may be in related, so this section is divided into different areas. If there are no identifiable subsets for the tests, this level of document structure can be removed. —SS]

[Include a blurb here to explain why the subsections below cover the requirements. References to the SRS would be good here. —SS]

4.1.1   Area of Testing1

[It would be nice to have a blurb here to explain why the subsections below cover the requirements. References to the SRS would be good here. If a section covers tests for input constraints, you should reference the data constraints table in the SRS. —SS]

Title for Test

1. test-id1

   Control: Manual versus Automatic

   Initial State:

   Input:

   Output: [The expected result for the given inputs. Output is not how you are going to return the results of the test. The output is the expected result. —SS]

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

How test will be performed:

2. test-id2

   Control: Manual versus Automatic

   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:

4.1.2 Area of Testing2

...

4.2 Tests for Nonfunctional Requirements

4.2.1 APR1 - Minimalist Layout

- **Test ID:** TAPR1

- **Type:** Static, Manual

- **Initial State:** User interface loaded

- **Input/Condition:** Visual inspection of layout

- **Output/Result:** Interface presents a minimalist layout with minimal distractions.

- **How test will be performed:** Manually inspect interface layout to ensure it follows minimalist design guidelines.

4.2.2 APR2 - High Contrast

- **Test ID:** TAPR2

- **Type:** Static, Manual

- **Initial State:** Interface set to default theme

- **Input/Condition:** Check for visual contrast

- **Output/Result:** All text and elements display high contrast for readability.

- **How test will be performed:** Perform a manual inspection of UI contrast using WCAG contrast standards.

4.2.3 APR3 - Intuitive Navigation

- **Test ID:** TAPR3

- **Type:** Dynamic, Manual

- **Initial State:** System interface opened

- **Input/Condition:** Navigate through different pages

- **Output/Result:** Users can easily navigate and locate functions within 3 clicks.

- **How test will be performed:** Manually navigate to different features and confirm efficient accessibility.

4.2.4 STR1 - Consistent Button Styles

- **Test ID:** TSTR1

- **Type:** Static, Manual

- **Initial State:** Interface loaded

- **Input/Condition:** Check for style consistency in buttons

- **Output/Result:** All buttons follow a consistent color and shape style.

- **How test will be performed:** Visually inspect all buttons to ensure they meet the style guidelines.

4.2.5 EUR1 - Tooltip Visibility

- **Test ID:** TEUR1

- **Type:** Dynamic, Manual

- **Initial State:** Interface loaded with tooltips

- **Input/Condition:** Hover over interactive elements

- **Output/Result:** Tooltips display with descriptive content.

- **How test will be performed:** Manually hover over elements to confirm tooltip visibility.

4.2.6 PIR1 - Customizable Color Themes

- **Test ID:** TPIR1

- **Type:** Dynamic, Manual

- **Initial State:** Interface with theme options

- **Input/Condition:** User switches between themes

- **Output/Result:** All themes display correctly with no visual errors.

- **How test will be performed:** Switch themes manually and verify consistent color scheme application.

4.2.7 LR1 - Initial Tutorial

- **Test ID:** TLR1

- **Type:** Dynamic, Manual

- **Initial State:** First-time user experience loaded

- **Input/Condition:** User accesses the system for the first time

- **Output/Result:** System displays an introductory tutorial.

- **How test will be performed:** Manually confirm tutorial launches on initial access.

4.2.8   LR2 - Tutorial Completion Time

- **Test ID:** TLR2

- **Type:** Dynamic, Manual

- **Initial State:** Tutorial in progress

- **Input/Condition:** Measure time for tutorial completion

- **Output/Result:** Users complete the tutorial within 5 minutes.

- **How test will be performed:** Track completion time for new users.

4.2.9   UPR1 - Friendly Feedback

- **Test ID:** TUPR1

- **Type:** Static, Manual

- **Initial State:** Error states are accessible

- **Input/Condition:** Trigger common errors

- **Output/Result:** System provides friendly, clear feedback.

- **How test will be performed:** Manually review error messages to confirm they are polite and helpful.

4.2.10   ACR1 - Access Control Verification

- **Test ID:** TACR1

- **Type:** Dynamic, Manual

- **Initial State:** User session active

- **Input/Condition:** Attempt unauthorized access to restricted data

- **Output/Result:** Unauthorized users are blocked from accessing data.

- **How test will be performed:** Attempt to access restricted areas with limited privileges.

4.2.11 ACR2 - Data Privacy Verification

- **Test ID:** TACR2

- **Type:** Static, Manual

- **Initial State:** Database access

- **Input/Condition:** Attempt to access personal data

- **Output/Result:** User data is secured and encrypted.

- **How test will be performed:** Review database encryption protocols and access policies.

4.2.12 PAR1 - Precision of Data Analysis

- **Test ID:** TPAR1

- **Type:** Dynamic, Automated

- **Initial State:** Analysis function active

- **Input/Condition:** Input test data with known precision values

- **Output/Result:** Analysis output matches expected precision.

- **How test will be performed:** Run automated precision tests on analysis output.

4.2.13 PAR2 - Rounding Accuracy

- **Test ID:** TPAR2

- **Type:** Dynamic, Automated

- **Initial State:** System loaded with rounding functions

- **Input/Condition:** Input values with decimals

- **Output/Result:** Values are rounded accurately according to specification.

- **How test will be performed:** Perform automated tests on rounding functions with pre-defined decimal values.

4.2.14   RAR1 - Fault Tolerance

- **Test ID:** TRAR1

- **Type:** Dynamic, Manual

- **Initial State:** Server operational with any state or load, e.g., idle, under little load (2 or fewer users interleaving requests less than once every 30 minutes), under intermediate use (2 or more users interleaving requests at least once every 30 minutes), or under strenuous use (4 or more users interleaving requests at least once every 5 minutes).

- **Input/Condition:** Server operational under any sequence of state transitions, e.g., from any of idle, under little, intermediate, or strenuous load, to idle, under little, intermediate, or strenuous load.

- **Output/Result:** Server operational with any state or load.

- **How test will be performed:** Schedule a 3-day monitoring period and use Ubuntu Server's `uptime` command to find the uptime across the monitoring period, allowing assessment and extrapolation of results from 3 days to 30 days. If possible, repeat the test with a 30-day (or longer) monitoring period. *Note*: Precise uptime metrics can only be achieved by formal checking which we do not have the expertise, time, or necessity for. A simple extrapolation result suffices for the given scope.

4.2.15   CR1 - Maximum Concurrent Users

- **Test ID:** TCR1

- **Type:** Dynamic, Load Test

- **Initial State:** System idle

- **Input/Condition:** Simulate multiple user logins

- **Output/Result:** System handles expected concurrent user limit.

- **How test will be performed:** Use a load testing tool to simulate concurrent users.

4.2.16   CR2 - Data Storage Capacity

- **Test ID:** TCR2

- **Type:** Static, Inspection

- **Initial State:** Database system active

- **Input/Condition:** Inspect data storage configurations

- **Output/Result:** System storage meets or exceeds required capacity.

- **How test will be performed:** Inspect database storage setup and ensure it meets capacity requirements.

4.2.17   EPER1 - Server Room Conditions

- **Test ID:** TEPER1

- **Type:** Environmental, Inspection

- **Initial State:** Server environment configured

- **Input/Condition:** Measure physical conditions in server room

- **Output/Result:** Server room meets temperature and humidity requirements.

- **How test will be performed:** Use physical monitoring devices to ensure environmental controls.

4.3 Productization Requirements

4.3.1 PRR1 - Production Readiness Verification

- **Test ID:** TPRR1

- **Type:** Dynamic, Manual

- **Initial State:** System deployed in staging environment

- **Input/Condition:** Perform a complete end-to-end run

- **Output/Result:** System operates without failure in a production-like setting.

- **How test will be performed:** Conduct end-to-end test in staging to verify production readiness.

4.3.2 MR1 - Ease of Code Updates

- **Test ID:** TMR1

- **Type:** Static, Manual

- **Initial State:** Source code repository active

- **Input/Condition:** Review update process

- **Output/Result:** Codebase is structured for easy updates.

- **How test will be performed:** Review code structure and modularity to ensure maintainability.

4.3.3 PR1 - Data Encryption Verification

- **Test ID:** TPR1

- **Type:** Static, Manual

- **Initial State:** Database system in use

- **Input/Condition:** Inspect database for encryption protocols

- **Output/Result:** All sensitive data is encrypted in storage.

- **How test will be performed:** Review encryption settings in database configuration.

4.3.4 AUR1 - Access Logs for User Sessions

- **Test ID:** TAUR1

- **Type:** Dynamic, Manual

- **Initial State:** System active with user sessions

- **Input/Condition:** Access user session logs

- **Output/Result:** Logs capture all user activities accurately.

- **How test will be performed:** Review session logs for accuracy and completeness.

4.3.5 CUR1 - Multilingual Support

- **Test ID:** TCUR1

- **Type:** Dynamic, Manual

- **Initial State:** System interface displayed

- **Input/Condition:** Switch to different language options

- **Output/Result:** System adapts to selected language without errors.

- **How test will be performed:** Switch languages manually and verify accurate translations.

4.3.6 LGR1 - Compliance with Copyright Laws

- **Test ID:** TLGR1

- **Type:** Static, Manual

- **Initial State:** Content library loaded

- **Input/Condition:** Inspect all music content for licensing

- **Output/Result:** All content has proper copyright attributions.

- **How test will be performed:** Check each music file and source for copyright compliance.

4.3.7   LGR2 - Adherence to Data Protection Regulations

- **Test ID:** TLGR2

- **Type:** Static, Manual

- **Initial State:** User data system in place

- **Input/Condition:** Inspect data management policies

- **Output/Result:** User data management meets legal requirements.

- **How test will be performed:** Review data management practices to confirm legal compliance.

4.4   Traceability Between Test Cases and Requirements

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

5   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]

5.1 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]

5.2 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]

5.2.1 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. ...

5.2.2 Module 2

...

5.3 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]

5.3.1 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:

5.3.2   Module ?

...

5.4   Traceability Between Test Cases and Modules

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

References

Author Author. System requirements specification. https://github.com/...,
2019.

6 Appendix

This is where you can place additional information.

6.1 Symbolic Parameters

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

6.2 Usability Survey Questions?

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

19

Appendix — Reflection

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

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. 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, Valgrind etc. You should look to identify at least one item for each team member.

4. 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?