System Verification and Validation Plan for Software Engineering

Team #13, ARC
Avanish Ahluwalia
Russell Davidson
Rafey Malik
Abdul Zulfiqar

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

List of Figures

1 Symbols, Abbreviations, and Acronyms

symbol	description
T	Test

[symbols, abbreviations, or acronyms — you can simply reference the SRS (**?**) 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]

The software being tested is "Realm", an AR social platform. The platform allows users to place AR objects in the real world for others to see, and allows organizations to create AR tours to provide users high quality experiences.

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]

?

[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

[List any approaches you intend to use for SRS verification. This may include ad hoc feedback from reviewers, like your classmates (like your primary reviewer), or you may plan for something more rigorous/systematic. —SS]

[If you have a supervisor for the project, you shouldn't just say they will read over the SRS. You should explain your structured approach to the review. Will you have a meeting? What will you present? What questions will you ask? Will you give them instructions for a task-based inspection? Will you use your issue tracker? —SS]

[Maybe create an SRS checklist? —SS]

3.3 Design Verification Plan

[Plans for design verification —SS]

[The review will include reviews by your classmates —SS]

[Create a checklists? —SS]

- Internal Team Review
- Peer Review
- TA Review
- Requirements Coverage Check

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]

The primary method of validation for the implementation of the system will be the functional, non-functional, and unit tests described in this document. In addition to the static testing and code review specified in the non-functional tests, we will have a code walkthrough for each module to verify adherence to the design, and the completeness of the implementation. In these walkthroughs we will use our MIS as a checklist and go through module implementations checking off correctly implemented elements.

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]

[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 Realm Testing

1. **Name:** Validate AR Object Perspective Adjustment
   Test ID: Test-RI1
   Type: Functional, Manual
   Initial State: Realm interface is open, AR objects are displayed in the user's vicinity on the camera feed.
   Input/Condition: Tester changes their position and angle in relation to an AR object.
   Output/Result: The AR object adjusts perspective appropriately, reflecting the new camera position and angle.
   How test will be performed: The tester will move around the AR object, observing whether it adjusts in real time and maintains correct perspective relative to the user's view.

2. **Name:** Validate AR Object Clutter Management
   Test ID: Test-RI2
   Type: Functional, Manual
   Initial State: Realm interface is open, tester is in an area with overlapping AR object instances
   Input/Condition: Tester moves camera over a crowded area where multiple AR objects are present.
   Output/Result: The interface selectively displays a manageable number of AR objects without overwhelming the user's view.

How test will be performed: Tester observes the interface to confirm only a few objects are displayed at once, reducing clutter in the view.

3. **Name:** Validate AR Object Placement Accuracy
   Test ID: Test-RI3
   Type: Functional, Automated and Manual
   Initial State: Realm interface is open, test AR object instance is nearby
   Input/Condition: Test AR object instance is placed with a known alignment in the real world, and reference screenshots
   Output/Result: Test AR object appears in correct position and orientation as expected from the known alignment and reference screenshots, position also matches stored object instance data
   How test will be performed: Automated tests will compare object instance data, while a manual test will involve the tester visually confirming the position accuracy in the camera view.

4. **Name:** Validate Sub-Realm Selection Indicator
   Test ID: Test-RI4
   Type: Functional, Manual
   Initial State: Realm interface is open with a sub-realm selected.
   Input/Condition: Tester views the interface with a sub-realm selected.
   Output/Result: The current sub-realm is clearly indicated on the interface.
   How test will be performed: Tester observes the interface to ensure that the active sub-realm is always visually indicated.

5. **Name:** Validate Sub-Realm Selection Change
   Test ID: Test-RI5
   Type: Functional, Manual
   Initial State: Realm interface is open with a sub-realm selected, tester is in area with object instances from multiple sub-realms
   Input/Condition: Tester attempts to change the sub-realm using the interface controls.
   Output/Result: The interface updates to display the newly selected sub-realm, and the correct object instances appear.

How test will be performed: Tester selects a new sub-realm and observes if the displayed objects and interface reflect the new selection.

6. **Name:** Validate Object Placement Workflow Control
   Test ID: Test-RI6
   Type: Functional, Manual
   Initial State: Realm interface is open.
   Input/Condition: Tester attempts to access the object placement workflow via the provided control.
   Output/Result: Tester is successfully redirected to the object placement workflow.
   How test will be performed: Tester selects the object placement control and verifies redirection to the appropriate workflow screen.

7. **Name:** Validate Object Scanning Workflow Control
   Test ID: Test-RI7
   Type: Functional, Manual
   Initial State: Realm interface is open.
   Input/Condition: Tester attempts to access the object scanning workflow via the provided control.
   Output/Result: Tester is successfully redirected to the object scanning workflow.
   How test will be performed: Tester selects the object scanning control and verifies redirection to the appropriate workflow screen.

8. **Name:** Validate Nearby Tour Indication
   Test ID: Test-RI8
   Type: Functional, Manual
   Initial State: Realm interface is open, tester is near the starting point of a tour.
   Input/Condition: Tester moves within range of the tour start point.
   Output/Result: The interface displays a clear indication of the nearby tour and a link to the tour preview.
   How test will be performed: Tester observes if the indication and link appear when near the tour start point.

9. **Name:** Validate Hazard Warning
   Test ID: Test-RI9
   Type: Functional, Manual

Initial State: Realm interface is open, tester is approaching a real-world hazard.
Input/Condition: Tester moves closer to a hazard in real space.
Output/Result: Interface displays a clear warning when user approaches the hazard.
How test will be performed: Tester approaches a wall with the realm interface open, and verifies that a warning appears.

10. **Name:** Validate Offline Mode for Interactive Components
Test ID: Test-RI10
Type: Functional, Manual
Initial State: Realm interface is open and disconnected from the internet.
Input/Condition: Tester attempts to interact with various components of the interface in offline mode.
Output/Result: Interactive components function normally, but location-based features are disabled.
How test will be performed: Tester verifies the functionality of object scanning and the unavailability of object placement, maps, and other internet dependent features.

4.1.3 Object Placement Testing

1. **Name:** Validate Object Selection Stage
Test ID: Test-OP1
Type: Functional, Manual
Initial State: Tester has progressed object placement workflow to object selection stage
Input/Condition: Tester selects object from one of: inventory, new object scan, new prompt generation
Output/Result: Interface successfully proceeds to sub-realm selection step with the selected object
How test will be performed: Tester initiates object placement workflow and progresses to object selection step. They then use one of the object selection methods to select an object and validate that the interface moves on to sub-realm selection with the selected object. They select the option to return to object selection, and repeat the process for all object selection methods.

2. **Name:** Validate Sub-Realm Selection Stage
   Test ID: Test-OP2
   Type: Functional, Manual
   Initial State: Tester has progressed object placement workflow to sub-realm selection stage, and is part of multiple sub-realms
   Input/Condition: Tester selects multiple sub-realms for sharing the object instance.
   Output/Result: The system associates the selected sub-realms with the object instance upon placement.
   How test will be performed: Tester initiates object placement workflow and progresses to sub-realm selection step. They then select multiple sub-realms before placing the object, verifying that the correct sub-realms are associated by checking the object instance database entries.

3. **Name:** Validate Object Placement Stage
   Test ID: Test-OP3
   Type: Functional, Manual
   Initial State: Tester has progressed object placement workflow object placement stage
   Input/Condition: Tester rotates, resizes, and translates the object in real space, then confirms placement.
   Output/Result: Rotation, resizing, and translating are all functional, and the AR object is positioned accurately in real space with the correct orientation, aligning with the tester's intent.
   How test will be performed: Tester initiates object placement workflow and progresses to object placement step. Tester rotates, resizes, and translates the object, verifying visually that it appears in the intended location and orientation in the real-world view, and finally confirms the placement of the object.

4. **Name:** Validate Object Instance Storage
   Test ID: Test-OP4
   Type: Functional, Manual
   Initial State: Tester has just placed an AR object instance
   Input/Condition: Tester checks the AR object instance database
   Output/Result: The AR object instance that the tester placed is present with correct details, including object type, sub-realm(s), posi-

tion, and orientation.
How test will be performed: Tester completes the object placement workflow then checks the AR object instance database and verifies the presence and correctness of their newly created object instance.

5. **Name:** Validate Area Based Placement Limit
Test ID: Test-OP5
Type: Functional, Automated and Manual
Initial State: Test user has sufficient AR object instances recorded to reach the object placement limit of an area
Input/Condition: Tester attempts to place another object in the same area
Output/Result: System prevents additional placements once the area limit is reached, displaying a relevant warning.
How test will be performed: Automated test script creates AR object instance entries in database to reach limit. Tester manually attempts to place another object in selected area, and validates that they are prevented from doing so, and are presented with a warning.

6. **Name:** Validate Time Based Placement Limit
Test ID: Test-OP6
Type: Functional, Automated and Manual
Initial State: Test user has sufficient AR object instances recorded to reach the time-based object placement limit for a user
Input/Condition: Tester attempts to place another object within a short period
Output/Result: System restricts further placements once the time-based limit is reached, displaying a relevant warning.
How test will be performed: Automated test script creates AR object instance entries in database to reach limit. Tester manually attempts to place another object within a short period, and validates that they are prevented from doing so, and are presented with a warning.

7. **Name:** Validate Automated Retry for Failed Object Storage
Test ID: Test-OP7
Type: Functional, Automated
Initial State: System is running, with simulated conditions preventing initial object storage (e.g., network issues).
Input/Condition: User places an object, but initial storage attempt

fails due to simulated conditions.
Output/Result: System automatically retries object storage until success or retry limit reached.
How test will be performed: Simulate a storage failure on the initial attempt, monitoring logs to confirm retry attempts are made until storage is successful, validating the success scenario. Repeat this process, but simulate continued storage failure, and verify that the number of retries attempted is equal to the retry limit.

4.1.4 Database Testing

1. **Name:** Validate Periodic Database Backup
   Test ID: Test-DB1
   Type: Functional, Automated
   Initial State: System is running, database is available, periodic backup is set up.
   Input/Condition: Periodic backup run is completed.
   Output/Result: Automated monitor verifies that the database backup is present and correct.
   How test will be performed: An automated monitor will wait for periodic backups, then restore the database from the backup in a sandbox environment and check that the data is present as expected.

2. **Name:** Validate Database Encryption
   Test ID: Test-DB2
   Type: Functional, Automated
   Initial State: System is running, database is available.
   Input/Condition: Command to check encryption status is inputted into DBMS for all databases
   Output/Result: DBMS response shows that all databases are encrypted
   How test will be performed: Will depend on the database platform used, for example on SQL Server the following query would be ran, and the output would be checked:
   SELECT db_name(database_id), encryption_state
   FROM sys.dm_database_encryption_keys;

4.2 Tests for Nonfunctional Requirements

[The nonfunctional requirements for accuracy will likely just reference the appropriate functional tests from above. The test cases should mention reporting the relative error for these tests. Not all projects will necessarily have nonfunctional requirements related to accuracy. —SS]

[For some nonfunctional tests, you won't be setting a target threshold for passing the test, but rather describing the experiment you will do to measure the quality for different inputs. For instance, you could measure speed versus the problem size. The output of the test isn't pass/fail, but rather a summary table or graph. —SS]

[Tests related to usability could include conducting a usability test and survey. The survey will be in the Appendix. —SS]

[Static tests, review, inspections, and walkthroughs, will not follow the format for the tests given below. —SS]

[If you introduce static tests in your plan, you need to provide details. How will they be done? In cases like code (or document) walkthroughs, who will be involved? Be specific. —SS]

4.2.1 Usability Testing

1. **Name:** Validate Localization
   Test ID: Test-QS-U1
   Type: Non-Functional, Manual
   Initial State: App is open on settings page.
   Input/Condition: Language setting is changed to English, Mandarin Chinese, Hindi, Spanish, and French (The 5 most spoken languages in the world) in turn.
   Output/Result: Text in the app correctly changes to the selected language, with understandable translations.
   How test will be performed: Tester navigates to settings menu, selects one of the languages to be tested, and verifies that text in the settings page, realm interface, and maps interface are all correctly displayed in the selected language

2. **Name:** Validate User Intuitiveness and Satisfaction
   Test ID: Test-QS-U2
   Type: Non-Functional, Manual
   Initial State: App is open, and tester is logged in as a new user with

no prior experience using the app.
Input/Condition: Tester performs common workflows such as account setup, navigating between interfaces, creating AR objects, viewing and placing AR objects, and finding objects on the map, all without guidance or assistance
Output/Result: 80% of testers complete each task and report that the app is easy and satisfying to use, and rate each workflow as highly intuitive.
How test will be performed: A group of new users will perform specified workflows and complete a post-test survey rating the intuitiveness and satisfaction of their experience. Quantitative results should show that most users rate the app as highly intuitive and satisfying to use.

4.2.2 Availability Testing

1. **Name:** Validate Server Availability
   Test ID: Test-QS-A1
   Type: Non-Functional, Automated
   Initial State: Server is running, with monitoring tools actively tracking uptime.
   Input/Condition: Automated monitoring scripts track server uptime and downtime continuously over a one-week period
   Output/Result: Monitoring scripts log any downtime, with server uptime recorded at 99% or higher over the test period.
   How test will be performed: Monitoring scripts will check server availability at regular intervals and log any downtime events, ensuring the server meets the 99% availability criteria.

2. **Name:** Validate User Feedback on Server Availability
   Test ID: Test-QS-A2
   Type: Non-Functional, Manual
   Initial State: Server is running, and a test group of users has been granted access to the app.
   Input/Condition: Over a one-week period, users in the test group access the app multiple times per day, as they normally would, at varying times.
   Output/Result: No user complaints about server unavailability dur-

ing the test period. Users report no issues with app access.
How test will be performed: Test group participants will be surveyed at the end of the test period regarding any access issues they encountered. Any user-reported issues will be logged and reviewed to assess the server's availability from the user's perspective.

4.2.3 Maintainability Testing

1. **Name:** Validate API Error Message Clarity
   Test ID: Test-DI-M1
   Type: Non-Functional, Manual and Automated
   Initial State: System is running, with logging enabled for internal API calls.
   Input/Condition: Simulate the following types of system failures in internal APIs and observe the resulting error messages: database connection failure, invalid input data, service timeout.
   Output/Result: Error messages generated by the APIs clearly indicate the source and nature of the error in at least 90% of cases, helping developers quickly identify issues.
   How test will be performed: Automated scripts will be used to simulate common errors and log the resulting API responses. The resulting error messages will be manually reviewed to be evaluated on detail (e.g., error type, location, and possible causes), and clarity (accurate indication of what is causing error). Success is achieved if 90% of the messages help to identify the error source.

4.3 Traceability Between Test Cases and Requirements

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

Test-ID	Test Name	Requirements
Test-RI1	Validate AR Object Perspective Adjustment	RI-FR1.1
Test-RI2	Validate AR Object Clutter Management	RI-FR1.2
Test-RI3	Validate AR Object Placement Accuracy	RI-FR1.2
Test-RI4	Validate Sub-Realm Selection Indicator	RI-FR2.1
Test-RI5	Validate Sub-Realm Selection Change	RI-FR2.2
Test-RI6	Validate Object Placement Workflow Control	RI-FR3
Test-RI7	Validate Object Scanning Workflow Control	RI-FR4
Test-RI8	Validate Nearby Tour Indication	RI-FR5
Test-RI9	Validate Hazard Warning	RI-FR6
Test-RI10	Validate Offline Mode for Interactive Components	RI-FR7
Test-OP1	Validate Object Selection Stage	OP-FR2.1
Test-OP2	Validate Sub-Realm Selection Stage	OP-FR2.2
Test-OP3	Validate Object Placement Stage	OP-FR2.3
Test-OP4	Validate Object Instance Storage	OP-FR1
Test-OP5	Validate Area Based Placement Limit	OP-FR3.1
Test-OP6	Validate Time Based Placement Limit	OP-FR3.2
Test-OP7	Validate Automated Retry for Failed Object Storage	OP-FR1
Test-DB1	Validate Periodic Database Backup	DB-FR1
Test-DB2	Validate Database Encryption	DB-FR2
Test-QS-U1	Validate Localization	QS-U1
Test-QS-U2	Validate User Intuitiveness and Satisfaction	QS-U2
Test-QS-A1	Automated Server Availability Monitoring	QS-A1
Test-QS-A2	User Feedback on Server Availability	QS-A1
Test-DI-M1	Validate API Error Message Clarity	DI-M1

Table 1: Mapping of Tests to Requirements

5 Unit Test Description

This section will contain descriptions of the unit tests derived from the MIS, which will be completed in a later iteration of this document

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

- For the following statements, please indicate your level of agreement between Strongly Disagree, Disagree, Neither Agree nor Disagree, Agree, and Strongly Agree:
  - Navigating between interfaces is intuitive
  - Placing objects is intuitive
  - Scanning objects is intuitive
  - Generating objects by prompt is intuitive
  - Embarking on tours is intuitive
  - Creating tours is intuitive (ONLY FOR ORG USERS)
  - Managing tours is intuitive (ONLY FOR ORG USERS)
  - Changing user settings is intuitive
  - Interacting with other's objects is intuitive
  - Reporting other's objects is intuitive
  - Adding friends, and creating and managing sub-realms is intuitive
  - Using the app is generally satisfying
- Did you experience any service interruptions, including but not limited to excessive load times for elements like navigation and object placement, while testing the app?

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.

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?