4TB6: System Verification and Validation plan

Stonecap Solutions - Smart Serve

Max Turek *turekm*
Ryan Were *werer*
Sam Nusselder *nusselds*
Peter Minbashian *minbashp*
David Bednar *bednad*1

April 6, 2023

1 Revision History

Date	Developer(s)	Change
11/02/22	Max Turek Sam Nusselder Ryan Were Peter Minbasian David Bednar	Initial Draft
03/08/23	Ryan Were	Revised Test Cases During VnV Report
04/05/23	Sam Nusselder	Updated VnV Plan to address issues and feedback from instructors, Tas, and test users. Also ensured this document aligned with preceding documents such as the SRS document by updating the corresponding requirement numbers as some requirements were removed from the SRS document.

Contents

List of Tables

2 Symbols, Abbreviations and Acronyms

Refer to Definitions, Terms, Acronyms and Abbreviations table in SRS.

3 General Information

This document serves to detail the specific activities and tools that will be implemented in order to verify, validate and test the requirements and functionalities of our system Smart Serve.

3.1 Summary

Smart Serve aims to provide the user the creation of a cocktail that is completely automated from an order generated through the Web App accessed by the user. The Smart Serve system is comprised of both a hardware and software component, therefore our system testing will include the integration of these two components working successfully together. Unit testing will aim to test individual components separately.

3.2 Objectives

The test plan outlined in this document plans on focusing on these following objectives:

1. Enumerate and describe all relevant tools and activities in conjunction with the testing plan
2. Specify the testing environment
3. Specify the given conditions required for successful testing

3.3 Relevant Documentation

Relevant documentation includes the SRS.

4 Plan

The Plan Section will document the Verification and Validation Team and their given roles. Additionally, plans for the SRS, Design, Verification, Implementation, and Software plans are outlined along with the Automated Testing and Verification Tools.

4.1 Verification and Validation Team

Name	**Testing Team (Description)**
David	Hardware Verification (SRS and Implementation Verification around Hardware)
Max	Software Verification (Design Verification around Software)
Peter	Hardware Verification (Design and Implementation around hardware)
Ryan	Software Verification (SRS Verification around Software)
Sam	Software Verification (Implementation Verification around Software))
Timofey Tomashevskiy	General Verification (Contacted throughout both streams of design any verification plan)
Volunteers	Front-End Design Verification (Attempt to operate beta stages of Front-End design)

Table 1: Validation Team & Roles

4.2 SRS Validation Plan

All team members will be required to help contribute to testing to ensure that Smart Serve was built and is functioning as intended and that Smart Serve successfully fulfills the requirements set out in the SRS document. This will ensure that Smart Serve is the right system and can complete the required tasks as intended.

4.3 SRS Verification Plan

4.3.1 Reviewers

Reviewers will be contacted on a bi-weekly basis, to ensure that the work of the project stays within the original scope. Reviewers will be contacted on a rolling basis to ensure that feedback is being given consistently throughout development.

Opposing Sub-Team As the team is split into two development sub-teams (software and hardware) each team will review the progress of the others' work.

Supervisor The supervising TA for the project will be asked for specific feedback regarding causes of concern with the project.

4.3.2 Major Tasks Reviewed

Major tasks will be reviewed throughout the development and deployment of the project. These major tasks are selected based on whether said tasks are vital to the base functionality of the project. These tasks are reviewed both based on their functionality and the elegance of the implementation.

1. **Opening and Operating The Web App**
2. **Selecting a Drink**
3. **Pouring of a Drink**
4. **Completing Creation of a Drink**

4.4 Design Verification Plan

4.4.1 Software

Back-End Back-end verification will be done with the help of two separate reviewers: The Hardware Team and the Supervising TA. Reviewing will be done to ensure that the code being written is simple to follow, clean, and that the solution is elegant in nature. Reviewers will be contacted once the design of pseudo code is developed to ensure solutions are straightforward and efficient.

Front-End Front-end verification will be completed with the help of one reviewing team: Volunteers. With the spirit of the front end being simple to operate, it is the intent to have average university students, from differing backgrounds, review the front end to ensure it is easy to navigate. Reviewers will be asked to complete basic tasks on the beta version and asked for feedback with regard to the experience. Volunteers will be contacted whenever a major milestone or decision is being made.

4.4.2 Hardware

For Hardware design, the main reviewers will be the supervising TA and the Software team. Both sets of reviewers will review any aspect of the design before its actual creation. This is to ensure that the design of the functionality is as simple as possible while being able to pass appropriate tests and complete appropriate tasks.

4.5 Implementation Verification Plan

For implementing features into the design, all final implementations must successfully pass the tests outlined in Section 5 of the report.
Additionally, all code must make use of its appropriate linter and be reviewed by other group members in the format of a pull request to ensure well-structured code. This pull request must also include outputs of any aforementioned tests which would be applicable to the nature of the feature being implemented along with detailed comments when needed.

4.6 Automated Testing and Verification Tools

4.6.1 Automated Testing

Python: For Python **PyTest** will be used to conduct any automated testing needs.

JavaScript: For JavaScript **Gulp** will be used to conduct any automated testing needs.

4.6.2 Linters

Python: For Python **flake9** will be used as a linter to verify the integrity and structure of any Python code.

JavaScript: For JavaScript **Standard JS** will be used as a linter to verify the integrity and structure of any JavaScript code.

HTML: For HTML **HTML** will be used as a linter to verify the integrity and structure of any HTML code.

CSS: For CSS **CSSLINT** will be used as a linter to verify the integrity and structure of any CSS code.

4.7 Software Validation Plan

Due to the nature of the system, and the physical product which is used as the output, there is no external data that will be used to validate the functionality of the software.

To validate the functionality of the software, it must be observed that the system is properly communicating between its components to create the correct drink. This can be checked by observing the creation of the drink and determining if the correct amount/type of liquid is being dispensed, corresponding to the user input.

5 System Test Description

5.1 Tests for Functional Requirements

The subsections below walk through testing the systems functional requirements in a natural sequential flow. We first test the Ingredient Availability and cover the functional requirements related to those. Next the tests look at the requirements related to ordering, and then making a drink. All of these requirements can be seen in the SRS in section 3.1 Ordering Drink and 3.2 Making Drink.

5.1.1 Ingredient Availability

The area of testing in this subsection verifies how our system handles ingredient availability. This relates to requirements ODR2, ODR4, ODR7, and ODR8 in the SRS documentation. The tests run through two scenarios of different ingredient availability. The Web App is then checked to ensure that the system is properly communicating this information as detailed in the requirements. A final test is performed to ensure the operator is able to communicate the available ingredients and the dispenser location of these ingredients with the Web App.

Ingredient Communications Test

1. ST-FR-ICT-01

   Control: Manual

   Initial State: System is powered and ready.

   Input: Ingredients are all in inventory.

   Output: The operator has no notifications on the Web App relating to ingredients being out of inventory. The menu on a user account has all drinks being available.

   Test Case Derivation: This is the expected behaviour for our system when there are no errors present.

   How test will be performed: The operator will refill all previously empty vats. Then they will check the menu on a user account to verify that all drinks are now available.

2. ST-FR-ICT-02

   Control: Manual

   Initial State: System is powered and ready.

   Input: Operator inputs all ingredients available and dispenser location of each ingredient into the Web App.

   Output: The ingredients and dispenser location map is updated to match the inputted ingredients and dispenser locations added by the operator. The menu on a user account has all drink combinations possible available.

   Test Case Derivation: The expected output has been derived from the functional requirements listed in the SRS document. These are ODR7 and ODR8.

   How test will be performed: The operator will use the Web App and go to the ingredients and dispenser location map page. They will input all ingredients available and the corresponding dispenser location for each ingredient. Once saved, the operator will check to see if the ingredients and dispenser location map is updated properly and that the menu the user will see has all the correct possible drink combinations as options.

5.1.2 Drink Ordering

The tests below go over the functional requirements related to the processes of ordering a drink. The first test relates to testing the Web App and ensuring the user is able to scan the QR code which will successfully bring the user to a functioning We App. Tests are then performed to ensure the user is able to order a drink and see the order number in the queue of orders as well as the estimated time remaining until the drink is made. Two more tests follow this to ensure the user is able to change and/or cancel the order they previously placed.

Web App Up Test

1. ST-FR-WAUT-01

   Control: Manual

Initial State: System is powered and ready.

Input: Operator scans QR code with their phone.

Output: Operator successfully reaches the Web App. Web App is functional.

Test Case Derivation: As per functional requirement ODR1 in the SRS, the QR code should lead all users to website that is functional.

How test will be performed: The operator will scan the QR code on the frame of the system using their phone. They will then go to the link that pops up. They will verify that they can reach the Web App.

Drink Ordered Test

1. ST-FR-DOT-01

   Control: Manual

   Initial State: System is powered and ready.

   Input: Operator orders a drink on a user account.

   Output: Operator can view the order number in the queue of orders.

   Test Case Derivation: The expected output has been derived from the functional requirements listed in the SRS document. This is ODR3.

   How test will be performed: The operator will order a drink on a user account. They will then check the order number in the queue.

5.1.3 Drink Making

The tests below go over the functional requirements related to checking that a cup is present for pouring, and making drinks. A test is then performed to ensure that the system dispenses the correct proportions and the drink is made properly. Another test ensures that when a cup is filled, the user and operator receive and will continue to receive notifications until the full cup is removed so the next drink can begin to be created. These notifications will be cleared and the next drink will begin to be created once the full cup has been removed.

Cup Present Test

1. ST-FR-CPT-01

   Control: Manual

   Initial State: System is powered and ready, except no cup is present.

   Input: Operator orders a drink on a user account.

   Output: Smart Serve searches for a cup by doing a full rotation, waiting 8 seconds and repeating this to give the user 3 chances to place their cup.

   Test Case Derivation: This test proves that Smart Serve can handle the scenario where a drink is ordered and no cup is present.

   How test will be performed: The operator will order a drink on a user account. They will then check to ensure Smart Serve is performing as expected.

2. ST-FR-CPT-02

   Control: Manual

   Initial State: Initial State: System is powered and ready, except no cup is present. User has ordered a drink.

   Input: After 10 seconds operator supplies a cup into the pouring area.

   Output: Smart Serve begins making the drink ordered as soon as the cup is sensed. The drink is poured until completion.

   Test Case Derivation: This test case checks what happens when there is initially no cup after a drink is ordered, and then a cup is placed.

   How test will be performed: After completion of test ST-FR-CPT-01, the operator will place a cup correctly orientated in the pouring area. They will then check to ensure Smart Serve behaves as expected.

Drink Made Test

1. ST-FR-DMT-01

Control: Manual

Initial State: System is powered and ready. Operator has ordered a drink on a user account. A cup is present.

Input: System is dispensing a drink.

Output: Completed drink made with the correct proportions. The correct user is removed from the queue.

Test Case Derivation: If the system is ready, this is the only scenario where the user is removed from the queue.

How test will be performed: After completion of test ST-FR-CPT-02 the ordered drink will begin to pour. Once this process has completed, the operator will check that the corresponding user is removed from the queue as the drink has been completed. The operator will check if the proportions dispensed for the drink are correct, using a calibration apparatus. Lastly, the operator will also check the Web App statistics to ensure that these have been updated to include this completed order in the statistics.

2. ST-FR-DMT-02

   Control: Manual

   Initial State: System is powered and ready, cup is full in the pouring area.

   Input: Operator orders a drink on a user account while full cup is present.

   Output: Smart Serve checks to see if the cup is full and rejects it as Smart Serve will not pour a drink in a full cup. Smart Serve will continue checking every 8 seconds to see if an empty cup is present to pour into.

   Test Case Derivation: From requirement MDR6 and MDR7 in the SRS, the expected behaviour of our system when there is a completed drink in the pouring area and other drinks in queue, is to not pour the next drink until the completed one is removed.

   How test will be performed: After completion of test ST-FR-DMT-01, the operator will order another drink with a different user account.

They will note whether the system is dispensing another drink or not, as it should not be. After 20 seconds or so, the operator will remove the full drink and place an empty cup instead. Smart Serve is expected to complete the drink that was previously ordered.

3. ST-FR-DMT-03

   Control: Manual

   Initial State: System is powered and ready, cup is full in the pouring area. Another user order is in the queue.

   Input: Operator adds a cup to the pouring area while scanning from previous test is still occurring.

   Output: The system successfully scans the cups for the empty cup. It then pours the drink into the empty cup.

   Test Case Derivation: Full cups should never have another drink poured into them. The system should scan until it finds an empty cup and then pour it into that. This test covers the case where a full cup is present and there is an order in the queue. The queue order can be made assuming there is an empty cup in the second cup holder.

   How test will be performed: After completion of test ST-FR-DMT-02, the operator will add an empty cup to the pouring area. The operator will make sure that the order in the queue is only being poured into the empty cup, and that the full cup is scanned and skipped.

4. ST-FR-DMT-04

   Control: Manual

   Initial State: System is powered and ready, cup is half full in the pouring area.

   Input: Operator orders a drink on a user account.

   Output: Smart Serve checks to see if the cup is considered full and rejects it as Smart Serve will not pour a drink in a full cup. Smart Serve will continue checking every 8 seconds to see if an empty cup is present to pour into.

Test Case Derivation: From requirement MDR6 and MDR7 in the SRS, the expected behaviour of our system when there is a completed drink in the pouring area and other drinks in queue, is to not pour the next drink until the completed one is removed.

How test will be performed: After completion of test ST-FR-DMT-03, the operator will take out all previous cups in the pouring area and place a half full cup. The operator will then order another drink with a different user account. They will note whether the system is dispensing another drink or not, as it should not be. After 20 seconds or so, the operator will remove the half full drink and place an empty cup instead. Smart Serve is expected to complete the drink that was previously ordered.

5.2 Tests for Nonfunctional Requirements

The tests for non-functional requirements set out to cover all measurable non-functional requirements outlined in the SRS document. Tests are done manually, automated, with user test groups, or with administrator access.

The following test cases cover usability and performance.

5.2.1 Usability Requirements

1. ST-NFR-UR-01

   Control: Manual

   Initial State: Testers are logged into a user account on the Web App.

   Input: Testers are asked to navigate the web page and place an order. Testers will rate their navigating and ordering experience on a scale from 1 to 5: 1 - unusable, 2 - poor, 3 - satisfactory, 4- good, 5 - excellent.

   Output: The average score from the testers in the survey is greater than 3.

   How the test will be performed: A test group of individuals who are of legal drinking age will be equipped with a device and a user account to log into the Smart Serve Web App. The testers will be given 10 minutes to navigate the webpage and submit mock orders that will not be sent to Smart Serve.

2. ST-NFR-UR-02

   Control: Manual

   Initial State: Testers will approach Smart Serve and grab a cup.

   Input: Testers are asked to grab a cup. Testers will report if the machine was too high, too low, or fine by filling out a questionnaire.

   Output: More than half of the testers say the height is fine.

   How the test will be performed: A test group of individuals who are of legal drinking age will approach Smart Serve and grab a cup. Testers will report if the height of the machine was too high, too low, or fine on a questionnaire. The test will pass if more than half of the testers say that the height is fine.

5.2.2 Performance Requirements

1. ST-NFR-PR-01

   Control: Manual

   Initial State: System is powered and ready with an empty cup present.

   Input: Operator orders a drink on a user account.

   Output: The drink is made and ready within 60 seconds of input

   How the test will be performed: The operator will have already gone through the process of logging into the user account. The operator will order any drink on the menu and begin a timer at the same time of the order. The order will then be sent to and made by Smart Serve. The operator will wait until the user account receives their drink. The test will pass if the timer has recorded less than 60 seconds.

2. ST-NFR-PR-02

   Control: Manual

   Initial State: Testers are logged into a user account on the Web App.

Input: Testers are asked to navigate the Web App and perform many different operations. Testers will rate the responsiveness of the webpage on a questionnaire with a scale from 1 to 5: 1 - unusable, 2 - slow, 3 - satisfactory, 4- fast, 5 - instant.

Output: The average score from the testers in the survey is larger than 3.

How the test will be performed: A test group of individuals who are of legal drinking age will be equipped with a device and the Web App logged into a user account. The testers will be given 5 minutes to navigate the web page and will be told to try as many different actions as possible. The Web App will be disconnected from Smart Serve and will not send any commands to the machine. The testers will be asked to rate the responsiveness of the Web App on a questionnaire from 1-5 based on the criteria explained above. The test will pass if the average is over 3.

3. ST-NFR-PR-03

   Control: Manual

   Initial State: System is powered and ready with an empty cup present.

   Input: User orders a drink.

   Output: The order is added to Smart Serves internal database within 20 seconds of input.

   How the test will be performed: The test case has a user send an order to Smart Serve. Smart Serve will record the order details into a database. An administrative user will manually check the database to verify that each order has been recorded. The test will pass if the order is added to Smart Serves internal database within 20 seconds of input.

4. ST-NFR-PR-04

   Control: Manual

   Initial State: System is powered and ready with an empty cup present.

Input: Operator orders a drink consisting of a single type of alcohol mixed without anything on a user account.

Output: The drink must contain less than 1.1x the amount of expected alcohol.

How the test will be performed: The operator will have already gone through the process of logging into the user account. The operator will order a drink consisting of a single type of alcohol mixed without anything. The order will then be sent to and made by Smart Serve. The drink will be poured into a measuring cup to manually measure the content. The test will pass if the content of the measuring cup is less than 1.1x the ordered amount of alcohol.

5.3 Traceability Between Test Cases and Requirements

A table that shows which test cases are supporting which requirements.

Test Case	Requirement(s)
ST-FR-ICT-01	ODR2, ODR4, & ODR7
ST-FR-ICT-02	ODR7 & ODR8
ST-FR-WAUT-01	ODR1
ST-FR-DOT-01	ODR3
ST-FR-CPT-01	MDR3 & MDR6
ST-FR-CPT-02	MDR3 & MDR6
ST-FR-DMT-01	MDR2, MDR4, ODR10, & ODR11
ST-FR-DMT-02	MDR6 & MDR7
ST-FR-DMT-03	MDR2, MDR4, & ODR10
ST-FR-DMT-04	MDR6 & MDR7
ST-NFR-UR-01	UHR3
ST-NFR-UR-02	UHR1
ST-NFR-PR-01	PR1
ST-NFR-PR-02	PR2
ST-NFR-PR-03	PR3
ST-NFR-PR-04	PR4, & PR5

As can be seen above, not all of the requirements in the SRS document are tested for in the above test cases. Requirements that were difficult to test for or requirements that were not essential to the successful operation of Smart Serve (i.e. specific non functional requirements such as OER6, MSR1, MSR2, MSR3, MSR4, etc.) were left out of the test cases.

6 Unit Test Description

This section will be filled out once the MIS is completed.

6.1 Unit Testing Scope

This section will be filled out once the MIS is completed.

6.2 Tests for Functional Requirements

This section will be filled out once the MIS is completed.

6.3 Tests for Nonfunctional Requirements

This section will be filled out once the MIS is completed.

6.4 Traceability Between Test Cases and Modules

This section will be filled out once the MIS is completed.

7 Appendix

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.

Appendix — Reflection

Max Turek:

In order to successfully execute our VnV plan, I will need to acquire skills in the areas of automated and embedded testing. Although I have used pytest before, I know there are many other libraries of tests like Serenity that can be used to verify certain software requirements for our system. In regards to embedded testing, since most of that testing will involve someone acting as a user and running through a bunch of regression testing, all use cases of the system need to be specified. Therefore, I will need to systemically learn a framework or methods in order to have an extensive list of workflow uses of the Smart Serve system. My options to learn these skills will be to ask help from my group mates that are more familiar with these tools. Furthermore, there are lots of videos on YouTube or Udemy that can educate me in automated and embedded testing. Fortunately, in my previous job we used some testing tools and I know I could always reach out to my former supervisor for any advice.

Peter Minbashian:

In order to execute the outlined VnV plan I will have to learn how to properly conduct an efficient survey to understand the general perception of our design. With the spirit of the project revolving around a user-friendly design, it is imperative that we have a strong user experience. This will require communicating with a varying amount of people, most likely volunteers. It is imperative that strong questions are asked as it gives a stronger insight into the flaws of any design, so appropriate changes can be made Although simple, this is also still a skill I have yet to practice. To learn more, there are plenty of basic online tools, such as YouTube, which can be used. However, more specific tools can be used to gather a greater understanding of this skill, mainly research-specific institutes that detail effective survey creation. An example of this can be found on the site of the Pew Research Center which has written a guide to developing an effective survey.

Ryan Were:

To successfully complete the verification and validation of our project, I will need to acquire static testing knowledge. I have never participated in code reviews or walkthroughs, but I see them as a valuable tool to help remove any silly errors or make code more efficient before running it through dynamic tests. To acquire this skill, I can research this topic, learn the processes in detail and then try to run a code review meeting with our team. Another approach would be to ask other group members if this is an area they have some degree of expertise in, and then have them run a code review or similar static testing tool. I personally would like to take the first approach where I do my own research so I have a really good understanding before I am actively participating in it.

Sam Nusselder:

In order to successfully complete the verification and validation of our project, I will need to gain knowledge and skills in system and unit testing. It will be important to execute the tests needed to ensure proper functionality of Smart Serve in a robust and accurate way. When performing these tests it is important to follow the outlined tests with the correct inputs and outputs and determine if the result is desired. New tests will also need to be developed as Smart Serve is created to test different systems or components of the system. One of the ways to gain knowledge and skills in the area of system and unit testing is to do research and watch videos on testing. Another approach would be to talk to professors, TA's and fellow classmates to learn from any system or unit testing they were involved in. I will likely use the first approach as this will allow me to do extensive learning in this area.

David Bednar:

To successfully follow through with the outline VnV plan, I will need to learn how to create automated end to end test cases along with learning proper test group surveying methods. Learning to create automated test cases can be done using online resources and working with my groupmates to learn different testing methods. Automated testing will require a testing framework that is compatible with the web application. The web application will likely use javascript, so I will mostly likely learn the automated testing framework Selenium. I will learn proper group surveying techniques. I can do this by using online resources or approaching TA's that have performed these methods before.