Test Plan

1. Introduction (1):

a. A brief summary of the product being tested. Outline all the functions at a high level.

b. This test plan will cover the testing of the "space facts" Alexa skill. The skill will randomly pull from a bank of facts and present it to the user whenever it is called. In order to test the product, we will be conducting a variety of tests to ensure that the product is working as intended.

2. Objectives and Tasks (2):

The objective of this test plan is to efficiently and effectively test our Alexa skill, Space Facts, to determine if the skill is ready to be submitted for release. When our testing is complete, we should have met all of the testing recommendations made by Amazon. Testing would confirm we are given a space fact when the Alexa skill is called upon, validate the information we are being given is what we wanted, and make sure no bugs exist in the skill. The post-testing stage would include having Amazon determine if the skill is ready for production, as well as users giving us feedback on how the skill works. We would then conclude with the problem reporting stage where we would gather the feedback from all sources and analyze the efficacy of our Alexa skill.

3. Scope (2):

a. This section describes what is being tested, such as all the functions of a specific product, its existing interfaces, integration of all functions.
   - i. Space Facts (Alexa skill)
     1. The testing on this product will be to ensure that whenever the skill is being called, appropriate information is being returned. In this case, the appropriate information would be facts about space.
   - ii. Lambda Function (handler)
     1. The Lambda function acts like an API, therefore we need to ensure that the proper information is being outputted by the API, which would be the facts about space.
   - iii. Alexa skill and the Lambda Function
     1. Together, they will be tested together to ensure that the Alexa skill calls the Lambda function, and that the Lambda function provides the Alexa skill the proper information it needs in order to output the fact properly.
   - iv. Skill on Alexa Echo Dot
     1. This will be tested just to make sure that when run on an Alexa device, the skill still functions as expected.

b. List here how you will accomplish the items that you have listed in the "Scope". For example, if you have mentioned that you will be testing the existing interfaces, what would be the procedures you would follow to notify the key people to represent their respective areas, as well as allotting time in their schedule for assisting you in accomplishing your activity?

    i. Testing the Alexa skill would require some information about the information that is passed from the Lambda function to the Alexa skill so that we can send a simulated API result object to the Alexa skill to see if the results are as expected. This will allow us to do testing without having to have the Lambda function working. This can also allow tests to identify whether it is the Alexa skill or the Lambda function that is broken if something is to not work as intended.
    ii. Testing the Lambda function can be done in the AWS console by writing sample tests in the Lambda console and running the tests. We will give it input as we expect from an Alexa skill call and compare the return value to see if it is what we expect. The only information required about the Alexa skill would be about what is being passed from the Alexa skill call to the Lambda function (handler).
    iii. In order to test the combination of both the Alexa skill and the Lambda function, we would need to make sure the Alexa skill is able to properly call the Lambda function and that the Lambda function is able to properly send a response back to the Alexa skill with the information that it needs to give back to the user. To check the functionality of the combination, we can just check to see if the result that we got was expected or not.
    iv. To test the skill on an Alexa device, we would load the skill onto the Alexa and then vocally call the skill to see if it works. There wouldn't be much excessive testing as the Alexa skill itself is pretty basic.

c. List the test items (software/products) and their versions.
    i. Space Facts (Alexa skill) - version 1.0
    ii. Lambda Function (handler) - version 1.0
    iii. Alexa Echo Dot - 4th gen

d. Identify all software features and combinations of software features that will be tested.
    i. Space Facts (Alexa skill) - Getting facts from the skill.
    ii. Lambda Function (handler) - Ensure facts can be extracted from the fact bank.

e. Features not to be tested, Specify the reasons these features won't be tested.
    i. All of the features will be tested, as this skill just returns a random fact when it is called, there are no erroneous inputs that need to be tested.

4. Testing Strategy (5):

Describe the overall approach to testing. For each major group of features or feature combinations, specify the approach which will ensure that these feature groups are adequately tested. Specify the major activities, techniques, and tools which are used to test the designated groups of features. The approach should be described in sufficient detail to permit identification of the major testing tasks and estimation of the time required to do each one.

Unit testing ensures that the code is working fine and is tested by using interfaces, functions, and classes. Unit tests will be performed by the developers. When there is an integration of at least two modules, testers can test the functionality and behavior of the modules after integration. This can be performed with developers as well along with testers to write integration tests. After testing has been completed with an integrated system to evaluate, system testing will be done by the testers to make sure it is in compliance to the specified requirement. Performance and stress testing will follow to ensure robustness of the program. Finally, user acceptance testing will be done by the client for validation.

4.1. **Unit Testing**

Definition: Specify the minimum degree of comprehensiveness desired. Identify the techniques which will be used to judge the comprehensiveness of the testing effort (for example, determining which statements have been executed at least once). Specify any additional completion criteria (for example, error frequency). The techniques to be used to trace requirements should be specified.

- The purpose of unit testing is to validate each unit of the skill to ensure the space skill is working properly in the voice interface that recognizes the request from the user (Alexa developer) and the back-end that provides the functionality to fulfill the request (Lambda). The overall approach would be using white box testing given the internal structure/design/and implementation of the skill being available to the testers. Subsequently, test oracles can be used to decide whether a test case execution succeeds or fails. Some corrective actions may be necessary depending on the trend analysis. Unit testing should be done until the error quantity and the error frequency are minimal. This is especially important to test whether the two parts of the skills: skill invocation and the lambda has set up correctly. It is also important to test whether or not the endpoint is working as intended to tie the voice user interface with the back-end functionality. The trigger also has to be configured using the skill ID verification to ensure Lambda is linked to the skill. These test statements should be executed at least once in order to properly test the skill. From there, the space facts can be modified through the Lambda in which it sends the respective JSON output from the alexa sill invocation input. Traceability matrix can be used to trace the requirements to the tests in order to verify that the requirements have been fulfilled. An example of that would be bi-directional traceability to ensure that the invocation skill and the lambda has been properly linked and have appropriate tests to verify that.

Participants: List the names of individuals/departments who would be responsible for Unit Testing.

- Unit testing will be performed by the developers (Kayla & Bailey) to find bugs locally and quickly fix the bugs during development.

Methodology: Describe how unit testing will be conducted. Who will write the test scripts for the unit testing, what would be the sequence of events of Unit Testing and how will the testing activity take place?

- The developers will be conducted by the developers to ensure the alexa skill code performs as expected. In order to do unit testing, developers will work with a section of code that tests a specific function by writing test scripts for unit testing. The developers will write the units tests and run them every time the skill is called. Since unit tests are testing a single unit of code, it's not testing the user interface and using stub data to visualize the dependencies. Developers can run the units tests locally on their laptop. Most importantly in the sequence, unit testing can test parts of the unit without waiting for others to be completed due to the modular nature. AlexaUnit tests work better with a debugger in order to better identify and fix issues with the space fact skill. There are many unit test frameworks available, and Bespoken Tools is commonly used.The setup will require VS Code, NPM, and Bespoken CLI with the accompaniment of Google actions. The testing activity will be done through the command prompt using Bespoken tools. Outputs of the tests shows the summary of about the success of the tests and the basic code coverage info. When there is a bug, the debugger will be useful in fixing which line of code has been invoked after adding a breakpoint of the erroneous section. Furthermore, bst-proxy can be used to debug a realtime Alexa request if needed. It is also substantial to make sure to use a version control system to keep track of the test scripts. This can be done by cloning the sample skill to the local desktop and installing the node-modules dependency and then setting up the test framework from npm. Once the folder is created, respective custom unit tests can be written without interfering with the master branch.

4.2 System and Integration Testing

Definition: List what is your understanding of System and Integration Testing for your project.

- System and Integration testing is an essential part of software testing. This is the process when an individual software is combined and tested as a whole group, rather than by itself. We will be able to expose any bugs or defects within the Alexa Skill based on different interactions with the Skill.

Participants: Who will be conducting System and Integration Testing on your project? List the individuals that will be responsible for this activity.

- Kayla - responsible for performance and stress testing involving users
- Brian - System and Integration testing

Methodology: Describe how System & Integration testing will be conducted. Who will write the test scripts for the unit testing, what would be the sequence of events of System & Integration Testing, and how will the testing activity take place?

- The System and Integration testing will be performed in a variety of sequences. This can be handled in the Alexa Developer Console where we can provide JSON inputs and use the GoLang framework to read the test data. We will be able to compare the expected outputs with the actual outputs, otherwise known as assert statements. Keeping track of version control will also be an effective way to limit and source the bugs and exploits. All source code will be written in VS Code as the universal IDE.
- Kayla and Brian will handle the system and integration test scripts. The testing activity will take place in the Halo office, where the process involves denoting the skill that needs to be tested. Creating a JSON testing file where we can create a repeatable outcome based on the expected and actual outputs. We will be able to test the system end-to-end to ensure the code is working properly, and the outputs match the actual outputs. We will then retrace the test cases and re-test the defects. This process is repeated until the outcome is where it should be.
- Another example of system and integration testing would be doing tests on the alexa voice invocation. The testers can test different languages to see how the skills work. For example, the tester can compare results for the languages already configured within the system and compare it with a language that has not been inputted in the system yet. The facts provided and error handling should be consistent with the different languages.

4.3 Performance and Stress Testing

Definition: List what is your understanding of Stress Testing for your project.

- Stress testing allows us to confirm the robustness of an application by deliberately trying to overwhelm the program. For this Alexa skill, it will consist of testing the handling of multiple voice requests simultaneously from one or more individuals, requesting multiple facts, and requesting more facts than the ones available. The Alexa skill should respond to a user request within two seconds. We will also consider the online resources that we pull space facts from, taking into account that these resources may have innumerable amounts of facts. Our team will also test performance when thousands of users access the skill at once to determine scalability and reliability.

Participants: Who will be conducting Stress Testing on your project? List the individuals that will be responsible for this activity.

- Kayla - responsible for performance and stress testing involving users
- Kevin - responsible for performance and stress testing involving external space fact resources

Methodology: Describe how Performance & Stress testing will be conducted. Who will write the test scripts for the testing, what would be the sequence of events of Performance & Stress Testing, and how will the testing activity take place?

- Performance and stress testing involving user input, whether it be through voice or text, will be conducted on the Alexa Developer Console testing simulator. Kayla will write the JSON test scripts for a variety of space fact requests. These sample utterances are fed to the Alexa skill.
- There will also be an analysis for the average time of the response outputted with the given voice invocation. The optimal time for a response should fall within a couple seconds. There should also be a respective connection error message or error handling if the Alexa is unable to connect to the server, or if the command is invalid.

4.4 User Acceptance Testing

Definition:

- The purpose of acceptance tests is to confirm that the system is ready for operational use. During acceptance test, end-users (customers) of the system compare the system to its initial requirements.
- For this project, it will consist of beta testing the Alexa Skill: Space Facts. We plan to release the skill to a limited amount of users for use after which feedback will be collected. It will also consist of black box testing from our quality assurance team, where they will confirm expected functionality with the specifications.

Participants: Who will be responsible for User Acceptance Testing? List the individuals' names and responsibility.

- Bailey - responsible for black box testing
- Kevin - responsible for facilitating the collection of user feedback

Methodology: Describe how the User Acceptance testing will be conducted. Who will write the test scripts for the testing, what would be sequence of events of User Acceptance Testing, and how will the testing activity take place?

- User Acceptance Testing will be conducted through the Alexa Developer Console. Specifically, black box testing will be done on the testing simulator. Bailey will be writing the test plans and JSON scripts and carrying out the testing by interacting with the Space Facts skill through text input. The JSON input and output responses are shown on the simulator and will be compared with its expected output. Beta testing will be conducted through the Skill Beta Testing tool. We will be sending testing invitations to 50 individuals through email. Then, those 50 users are able to try out the space facts Alexa skill on their Alexa devices, and their feedback will be available on the developer portal. We can further leverage the utterance history and interaction path analytics to see where users ran into issues.

4.5 Automated Regression Testing

Definition: Regression testing is the selective retesting of a system or component to verify that modifications have not caused unintended effects and that the system or component still works as specified in the requirements.

Participants:

- Michelle - responsible for Automated Regression Testing

Methodology:

- Automated Regression Testing will be developed by Michelle using Mocha.js, a unit testing framework, and Chai.js, an assertion framework. Mocha will specifically be used to conduct automated unit testing, while Chai will determine if the expected results are what should be expected. Using Bespoken’s Virtual Alexa component, Michelle will be able to simulate Alexa voice commands to send JSON objects to our skill and receive an answer back for validation.

5. Hardware requirements (0.5)

Computers: Any working computer with microphone and device compatible with Alexa for voice input.
Other hardware: Requires proper CPU and a minimum of 16KB memory for protocol buffer. However, no hardware such as Amazon Echo required to test the skill.

6. Test schedule (1)

Estimate the time required to do each testing task. Specify the schedule for each testing task and test milestone. For each testing resource (that is, facilities, tools, and staff), specify its periods of use.

Type	**Schedule**	**Resources**
Unit Testing	Dec. 5 - Dec. 7 (3 days) Day 1: 35% method coverage Day 2: 70 % method coverage Day 3: 100% method coverage	*Facilities: Halo Office* *Tools: Bespoken, Mocha, BrainMacIntosh Alexa-Skill-Test-Framework*

		Staff: Michelle, Bailey
System and Integration Testing	Dec. 8 - Dec. 10 (3 days)	*Facilities: Halo Office* *Tools: Alexa Developer Console, GoLang* *Staff: Brian, Kayla*
Performance and Stress Testing	Dec. 11 - Dec. 13 (3 days)	*Facilities: Halo Office* *Tools: Alexa Developer Console* *Staff: Kayla, Kevin*
User Acceptance Testing	Dec. 14 - Dec. 18 (5 days) Day 1-3: 30 participants Day 4-5: 20 participants	*Facilities: Crescent Conference Room* *Tools: Alexa Developer Console - Analytics and Skill Beta Testing* *Staff: Bailey, Kevin*
Automated Regression Testing	Dec. 19 - Dec.21 (3 days)	*Facilities: The Hub* *Tools: Mocha, Chai, Bespoken* *Staff: Michelle, Brian*

7. Resources/roles & responsibilities (0.5)

Specify the staff members who are involved in the test project and what their roles are going to be (for example, Mary Brown (User) compile Test Cases for Acceptance Testing). Identify groups responsible for managing, designing, preparing, executing, and resolving the test activities as well as related issues. Also identify groups responsible for providing the test environment. These groups may include developers, testers, operations staff, testing services, etc.

Kobe - Project Manager, oversees the whole project
Brian - System and Integration testing and Automated Regression testing
Michelle - Unit testing and Automated Regression Testing
Kayla - System and Integration testing and Performance and Stress testing
Bailey - Unit testing and User Acceptance testing

Kevin - Performance and Stress testing and User Acceptance testing

8. Risks/assumptions (2)

Identify the high-risk assumptions of the test plan. Specify contingency plans for each (for example, delay in delivery of test items might require increased night shift scheduling to meet the delivery date).

One volatile aspect of the test plan revolves around our testing schedule. The schedule leaves little room for delays and any subsequent delays could result in more testing needing to be done on a given day, or having the schedule pushed back some.

Another volatile aspect of the test plan falls in the User Acceptance Testing section due to our test plan taking into account having 50 users who would conduct tests for us, and then submit their feedback to us. All 50 users may not send their feedback, nor test, but we set the number to 50 participants to account for nonresponsive participants.

Hardware failures is another potential issue as a microphone may not work or the entire system may fail, but this is unlikely.

9. Tools (0.5)

List the Automation tools you are going to use. List also the Bug tracking tool here.

- Mocha.js
- Chai.js
- Continuous Integration (CI)
- Code Cov (Code Coverage)
- Bespoken NLI
- NPM