Software Testing Plan

JabberJack

Team Member:

Sara Harris, Tyler Zimmerman, Jiasheng Yang, Gabriel Proctor

Team Mentor: Felicity H. Escarzaga

Client: Dr. Andy Wang

Northern Arizona University

Version 1.0

---

1 April, 2022

Contents

1.0 Introduction

JabberJack

Information is the foundation on which academia is built; knowledge and knowing is a part of learning and understanding and without ways to move information questions would never be answered. At Northern Arizona University there is a disconnect between information available and questions that need to be answered. This gap has caused confusion and questions to continue to be unanswered. The ChatterJack chatbot can solve this problem; the ChatterJack is a chatbot software that is able to answer questions about NAU and more specifically about the engineering college. The chatbot is a standalone software that can connect to an online database. This database is updateable through the online portal where approved faculty and staff can add, update, and delete information. The goal of these applications is to create an easy way for students, visitors, and faculty to access needed information.

The software is nearing the final stages of completion and in order to further development testing needs to be done. “Software testing is the process of evaluating and verifying that a software product or application does what it is supposed to do”[1]. Developers can create applications but they cannot test them since they know exactly how to interact with the application in order for it to do what it was designed to do. Testing in general is needed so that developers can ensure that modules work together and that users are interacting with the software properly. Testing can point out flaws within the software so that developers can fix it before releasing it to the users. Without testing there would be a lot of unhappy users and potentially lost revenue.

There are several tests that the ChatterJack software will need to be run through before its final software iteration. The Chatterjack will need to pass and go through three major testing phases: unit testing on all of its functions, integration testing on all of the major modules, and usability testing to ensure users ability to interact with the chatbot.

Natural language processing will require the most testing as it is the core of the chatbot's "understanding", there will also need to be unit testing done on both user authentication procedures as well as our database to make sure that everything is functioning in the way that is expected. Next the document will further describe what is needed for integration testing on the problem retrieval system, message generator, user interface, web portal, and the database. All of these components need to be able to work together so that the chatbot will function properly in production. Lastly, usability testing will be vital for the chatbot and web portal since it needs to be intuitive on both ends in order to be effective.

The natural language processing (NLP) module will require the most testing since it interacts the most with users. It will need to be able to understand words and questions that are unique since there is more than one way to ask a question. The nature of the chatbot will require more user testing than other applications since users must be able to interact with it without any instructions. It is also imperative that all modules are able to function together since without one module there is not a usable application.

This document will review the project's major testing sections: unit testing, integration testing, and usability testing. Testing is important because it allows developers to detect problems with the software early and be able to make needed improvements based on the results of these tests.

2.0 Unit Testing

JabberJack

Unit testing is a type of software testing that separates a program into different individual units, and then tests each part to make sure that associated control data, usage procedures, and operating procedures are fit for use. JabberJack will utilize unit testing to check whether each component of the ChatterJack chatbot can achieve the expected output and make it more clear about where bugs might be for future iterations of the project.

The natural language processing module of the project is written entirely in python because of this we can use unittest, a built-in python library, to create unit tests for all of the functions in the natural language processing module. Using unittest we can automate the testing of functions so that these tests can be run every time a change is made that may break the software. Using unittest we can ensure that the project's NLP will remain functional through multiple iterations.

As for the user authentication module we will implement some very basic unit testing. The web portal is written in basic PHP and to test user authentication we can just write some simple functions that will be able to test that the correct values are being grabbed from the database based on user input.

2.1 Natural Language Processing

The natural language processing system is the core of the ChatterJack chatbot and forms the chatbot's "understanding" of classes, people, organizations and interrogatives. This unit consists of two main classes; one is used for grabbing the interrogatives and subjectives, which is built by a python library named spaCy; another one is used for correcting the input and implementing fuzzy finding, which is built by a python library named sklearn and calculates the tf-idf. And all of them need to be tested to ensure that the chatbot is "understanding" input questions in a way that allows it to grab the correct answers needed for response.

By using unit testing, the team can ensure that the chatbot is extracting the correct information from user questions. Below table 2.1.1 shows all of the tests for grabbing intention class, inputs, expected outputs, and whether they passed or failed.

Function Tested	Expected	Outcome
`segmentSentence()`	Use token from spaCy segment the input sentence	**PASS**
`segmentList()`	Store each word in a list from token result	**PASS**
`interrogative()`	Grab the interrogative	**PASS**
`person()`	Grab the normal person using entity from spaCy	**PASS**
`nau()`	Grab the normal person using entity from spaCy	**PASS**
`expandPerson()`	Grab the specific name, eg: Dr. D	**PASS**
`classes()`	Grab the specific name using regular expression	**PASS**
`intention()`	Store all grabbing information into a dictionary and corresponding labels	**PASS**

Natural Language Processing Unit Tests Table 2.1.1

By using unit testing, the team can ensure that the chatbot is correcting the fuzzy input to the correct information stored in the database. Below table 2.1.2 shows all of the tests for correcting fuzzy input class, inputs, expected outputs, and whether they passed or failed.

Function Tested	Expected	Outcome
`tf()`	Calculate the tf-idf value of inputting and standard information	**PASS**
`compare()`	Store the inputting information and tf-idf value in a dictionary	**PASS**
`perCheck()`	By tf-idf value, judge whether correct the inputting information (person and organization)	**PASS**
`claCheck()`	By tf-idf value, judge whether correct the inputting information (class)	**PASS**

Natural Language Processing Unit Tests Table 2.1.2

2.2 User Authentication

User authentication exists so that the system is protected and so that the system can identify users. The management system for user authentication is written in PHP and utilizes the server's database to create and store user credentials. Users are required to request an account and from there are able to login and access certain elements of the database based on who they are. There are three major parts of the authentication system that need to be tested: registration, login, and logout. A manual test will be run to ensure that users are able to be registered, login to their accounts, and logout of their accounts.

To do this we will use a version of "black box testing", a type of testing that tests functionality rather than code; the code is not visible hence the name blak box. The team will test the functionality of the authentication system and not directly test the code.

Registration Flow Test

- Administration will create a new account by filling out the registration form
    - Expected : "Account Creation Successful" prompt, **SUCCESS**
    - Otherwise: **FAIL**

Login Flow Test

- Login using the credentials just created
    - Expected : Homepage display, **SUCCESS**
    - Otherwise : **FAIL**

Logout Flow Test

- Click the logout button present on the homepage and check if session is successfully cleared by trying to access the URL of another page
    - Expected : Return to login page if session is properly cleared, **SUCCESS**
    - Otherwise: **FAIL**

System Tested	Expected	Outcome
Register	"Registration Successful"	**PASS**
Login	Homepage display	**PASS**
Logout	Login page display	**PASS**

Authentication Unit Tests Table 2.2.1

By utilizing these three manual tests the team can tell where something is failing. If it fails in registration then there is a bug in the creation of a new user entry in the database. If it fails in login then there is an issue drawing user information from the database. If it fails in logout then the session is not properly cleared and needs to be refactored. When all tests pass then the authentication system works as expected. Completing these tests are necessary to ensure that user information is correctly being stored and drawn from the database; it is done to ensure that user information is being protected and properly handled.

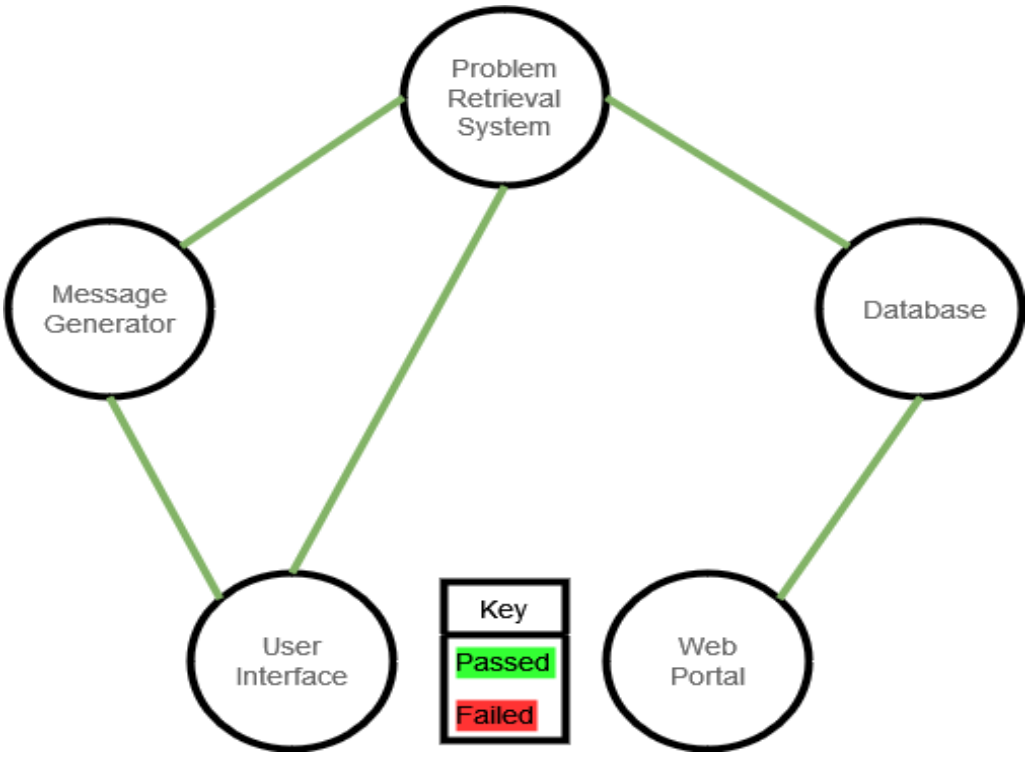
3.0 Integration Testing

JabberJack

Integration testing is another software testing method that combines each individual unit and tests them as a group. The purpose of integration testing is to evaluate whether the separate components of an application are able to function as a whole. Integration testing is necessary since separate pieces of an application may work properly on their own but when combined will fail; in order to ensure that an application is functional as a whole we employ integration testing.

The ChatterJack chatbot is used for answering questions from NAU students or staff quickly and accurately. The modules that integration testing needs to be used on for this project include the problem retrieval system, message generator, web portal, user interface, and database. All of these separate modules need to be able to communicate with each other to create a functioning chatbot system.

In figure 3.1 we can see all of the connections that are made between the 5 separate modules. Connections that have passed their testing are shown here in green and ones that have failed are in red. **The only test that has currently passed is the database integration with the web portal.**



Integration Tests Graph Figure 3.1 - CURRENTLY INCOMPLETE TESTING

3.1 Problem Retrieval System - Message Generator

The Problem Retrieval System (PRS) for the chatbot is the core component that combines the Natural Language Progress (NLP). The key idea for PRS is to grab the interrogatives and subjects, including classes, persons, and organizations. It should also handle all input and can use fuzzy finding by comparing the similarity of the input subjects and correct information stored in the database. In order to ensure that users receive back answers that make sense the PRS has a connection with the message generator.

To ensure that the system can handle fuzzy finding, the team will adjust the parameter of similarity between the input key information and correct information stored in the database. There will be a function named `checkSimilarity()` that is used to find the proper parameter so that it cannot return a wrong matching correct information. The team will test different miss typing key information; for example, the correct information is “Winfree”, the team will take notes of “winfrey”, “winfree”, “vinfrey”, and “wonfree”.

To ensure that NLP can always grab interrogatives and subjectives, the team asks as many different types of questions as the team. All the grabbing information will be stored in a dictionary. There will be a function named `grabIntention()`

that stores a set of dictionaries corresponding to different input questions. Input questions should refer to different types of interrogatives and subjects. If the key of the dictionary returns the grabbing information from the input question, and the value of the dictionary returns the corresponding label of the grabbing information, the test passes.

3.2 Message Generator - User Interface

The Message Generator for the chatbot is used for combining the key information that PRS retrieved from the database with sentence constituents to generate the answer, and then pass it to the User Interface. Here the message generator has connections with the problem retrieval system and user interface.

To ensure the connection between them, the team asks as many different types of questions as the team can so that every answer is accurate, readable, and reasonable. There will be a function named `correctAnswer()` that can return the corresponding answers according to the input question, then checking whether they are more like human conversation or not. Input questions should refer to different types of interrogatives and subjects. If answers are readable and accurate, the test passes.

3.3 User Interface - Problem Retrieval System

The user interface for the chatbot is where users can input their questions. Here the user interface has connections with the problem retrieval system and message generator.

To ensure the connection between the problem retrieval system and the user interface the team will check to see that the problem retrieval system is grabbing the proper information from an inputted question. This will make sure that the problem retrieval system is getting all of the information that is needed from the user interface.

There will be a function named `correctQuestionInfo()` that has hard coded questions with specific interrogatives and subjects. There will be a corresponding list of expected outputs. These questions will be passed to the problem retrieval system (PRS) and if the PRS returns the corresponding “expected” interrogatives/subjects, then the test passes. This will be run for multiple data points with differing interrogatives as well as subjects.

3.4 Web Portal - Database

The web portal is an essential part of the chatbot; this is where faculty and administrative users can add/update/delete information within the chatbot which will directly influence what questions it is able to answer. The Web portal only has a direct connection to the database and will need to have integration testing to ensure that these updates, additions, and deletions are going through correctly.

To ensure that the web portal can add an element to the database, a function named `addCheck()`, will query the database and collect the number of entities within the database and have a series of hardcoded elements to be added to the database. After which, utilizing the web portals "add" function, these hard coded elements will be added to the database. To pass, the database must have exactly the number of hard coded elements plus the number of existing elements and the elements added must be exactly the same as the hard coded elements.

To ensure the web portal can delete elements a function named `deleteCheck()` will have a series of hard coded database elements that will, first be queried to ensure their existence within the database, then these elements will be deleted with the web portal's "delete" function. The database will be re-queried to check if the hard coded elements exist within the database, if the query results contain no elements, the test passes, if elements are returned, the test fails.

To ensure the webportal can update information a function named `updateCheck()` will have a series of hard coded database elements. The function will first check that the hard coded elements already exist within the database, then record the values within the database. Then the hard coded elements will be passed to the web portal's "update" function. The database will be re-queried and the values of the elements will be recorded and compared to the values previously taken. In order for the test to pass these values must be different from the previous values and the new values must be equal to the hard coded elements.

If all functions return pass, then the webportal-database connection can be declared fully functional.

3.5 Database - Problem Retrieval System

The database is where all information pertaining to the questions the chatbot can answer will be stored. It has a connection to the problem retrieval system; this is the module that queries the database based on user questions to get the information stored within it.

To ensure that the database is correctly being queried by the problem retrieval system, a function named `retrievalCheck()`. It will have a series of hard coded questions to which some will have corresponding elements within the database and some that will not. A set hard coded database entries named expectedResponse will be contained within the function. These questions will be passed to the problem retrieval system. For each question being passed to the system, its response must be the same as the expected set of database entries.

4.0 Usability Testing

JabberJack

Usability testing implements the use of end users so that developers can examine how efficient a program is to use with a group of representative users. This is dones so that the development team can ensure that the ChatterJack is interacted with effectively and that the software is intuitive to all users. JabberJack separates users into three groups including general user, faculty, and administrator. The goal of usability testing is to see how users will interact with the software and based on the interactions the team can make changes to better suit the end users.

The team will separate the end user into three different groups, and each group has five to six end users who will participate in different testing tasks; no further information about the product will be given to them as it needs to be intuitive. Users will then be asked to think aloud as they interact with the product. Then the team will take notes on how users are using the product, what they think of the product, and will gauge how much users are enjoying the product.

4.1 General User

The general user, including NAU faculty, NAU student, visitor, or anyone who wants to know some information about NAU, will only be able to interact with the chatbot. This user is able to ask questions either by typing out the question into the user interface or by speaking aloud.

Five to six general end users will be asked to interact with the chatbot by asking questions; no further information about the product will be given to them as it needs to be intuitive. Users will then be asked to think aloud as they interact with the product. Then the team will take notes on how users are using the product, what they think of the product, and will gauge how much users are enjoying the product. Once the user is “done” working with the product they will be asked to rate several characteristics on a scale of 1 to 10. They will be asked to rate: Easability, enjoyment, and engagement. The user interactions with product should following these functionalities:

- User can ask the chatbot a question via text/speech input
- Receive a correct answer shown on the screen and read aloud it
- The conversation should feel like the human interaction
- It can handle the multiple different inputs for the same question
- Return “Sorry, answer not found” message when it cannot find the answer to the user’s question

4.2 Faculty

The faculty for this application can query or modify their own personal and course information. He/She cannot see or modify other faculty's information. They must get permission to get access to an account. When they login into the administration system it will link to the faculty manage page where they can manipulate data they have access to in the database.

Five to six faculty members will be asked to manage their own information. All selected faculty members should start the testing from registering. This testing should pay more attention to the simplicity of operation and design of the web portal. They will be asked a scale of 1 to 10 for both characteristics. The user interactions with product should following these functionalities:

- Faculty can request an account be made with given credentials
- Check their own register information, including the account type, password, and user name
- Login into the administration system
- Search and modify the information according to the selected table, PERSON or CLASS
- For personal information, they can modify their job, office room, office hour, and specific name
- For course information, they can modify class time, location, definition, and professor name
- Logout of their account

4.3 Administrator

The administrator for this application has all permissions to manage all Question and Answer (QA) pairs from the database, including searching, deleting, inserting, and modifying. They also must get permission when they login into the administration system.

Five to six administrators will be asked to manage all QA pairs of three tables from the database. All selected administrators should start the testing from registering. This testing should pay more attention to the simplicity of operation and

beauty of the web portal. They will be asked a scale of 1 to 10 for both characteristics. The user interactions with product should following these functionalities:

- Administrators need to request administrative access for an account
- Check their own register information, including the account type, password, and user name
- Login to the administration system
- Search, insert, modify, delete the information according to the selected table, PERSON, ORG, or CLASS.
- For search operation, if they do not type the specific searching information, there will be all the information displayed
- For insert operation, there should be the information shown on the text box
- For personal information, they can modify their job, office room, office hour, and specific name
- For course information, they can modify class time, location, definition, and professor name
- For organization information, they can modify its name, location, director, and how to go there
- Logout of their account safely

By testing the chatbot the team can receive needed user input for any aesthetics or design decisions. Using user testing the chatbot can become better suited specifically for the end users and can improve aspects of the chatbot for the final iterations of the project.

5.0 Conclusion

JabberJack

Software testing is an extremely important part of the process of software design and development. It allows users to get hands-on experience with the software while also testing its functionality and usability. Unit testing helps ensure that each little bit of the software is working and integration testing tests that each bit is working together properly.

The document has outlined a detailed plan in which it will run through multiple unit tests on the natural language processing unit of the project as well as the web portal. Each function that is necessary and vital will be tested to make sure that all parts are working the way they were intended. Then to put it all together integration testing will test all five connections between the parts to create a complete application. The ChatterJack application will then move through usability testing where users of each type will get a chance to try out the application for themselves and will be able to point out places where it could improve. Usability testing is vital as it can point out bugs and problems that the team previously was not able to see.

Software testing will allow the team to move forward with the design of the ChatterJack and catch bugs that were not previously apparent. This will improve the application by making it both more bug free as well as more user friendly. The Chatterjack chatbot will allow students, faculty and visitors to get answers to questions and to create a vital connection between people and information. The project will help Northern Arizona University establish physical chatbots on campus and will set a foundation for future chatbot projects on campus.

6.0 Reference

JabberJack

[1] IBM. 2022. Retrieved March 28, 2022 from https://www.ibm.com/topics/software-testing