Drexel Chatbot System Test Plan

Hoa Vu <htv27@drexel.edu>
Tom Amon <tpa27@drexel.edu>
Daniel Fitzick <dwf35@drexel.edu>
Aaron Campbell <ajc382@drexel.edu>
Nanxi Zhang <nz66@drexel.edu>
Shishir Kharel <sk3432@drexel.edu>

Version: 1.1

Revision History

Name(s)	Date	Comment(s)	Version
All	4/16/17	Preliminary Document Structure	0.1
All	4/23/17	Outlined document with notes and questions	0.2
All	4/25/17	Created test cases and linked the Test Case document	0.3
All	4/28/17	Created draft	0.5
All	4/30/17	Complete document	0.9
All	5/1/17	Group review and final revisions	1.0
All	5/2/17	Minor revisions	1.1

1. References

Project Plan v3.0:
https://github.com/DrexelChatbotGroup/DrexelChatbot/blob/master/docs/DC_ProjectPlan_v3.pdf
Requirements Specifications v1.1:
https://github.com/DrexelChatbotGroup/DrexelChatbot/blob/master/docs/DC_RequirementsSpecification_v1.pdf
Design Document v1.0:
https://github.com/DrexelChatbotGroup/DrexelChatbot/blob/master/docs/DC_DesignDocument_v1.pdf

2. Introduction

This document is used to define the details of the system test plan for the Drexel Chatbot project. It will describe the testing strategy the team will use to verify that the application meets the requirements listed in the Requirements Specification Document v1.1. This will include black box test cases of the entire product, as well as test cases of the backend, API, and database to ensure the entire system works as planned. This document will discuss software risk issues, and specify what features to be tested and what features to avoid.

If more tests are needed, all developers will agree on new tests and add them to the document. If anything related to the test plan changes during the execution and testing phase of the project, this document should be updated and re-approved.

3. Test Items

This plan aims to test the Drexel Chatbot system in four areas:

3.1 Front-end applications interface

These are the Web, mobile and SMS interfaces discussed in Design Document v1.0
The focus is on the functionalities of the applications themselves and not their integration with the backend. For example, the functionality of text boxes and buttons will be tested. Test cases will use the Drexel Chatbot web interface located at http://10.246.251.67:8080/chatbot as the main test tool.

3.2 API

The focus is on the functionality of the API itself and not their integration with the backend. For example, testing that the API returns the correct JSON format and error codes. Test cases will use the API itself, located here: http://10.246.251.67:8080/chatbot/api?query=, as the main test tool.

3.3 Database and information extraction

The focus is more on showing that data exists in the required areas, instead of showing the data is correct. Test cases of the database and information extraction module will use the stardog web application, located here: http://10.246.251.67:5820/chatbotDB#!/query, as the main test tool.

3.4 Backend

The focus is on testing the functionality of the backend system. Specifically, these following modules discussed in Design Document v1.0 will be tested: Main, GenericQuestionConstruction, GenericAnswerConstruction, GenericAnswerPopulation, and ErrorHandler. Test cases will use the Drexel Chatbot web interface located at http://10.246.251.67:8080/chatbot as the main test tool.

4. Software Risk Issues

Our system is using neural networks to find answer for questions, which requires retraining if new questions are to be answered by the system. As a result, answers may be different between before and after the retraining. This may cause a question that worked in a previous version to no longer work in a subsequent version. To mitigate this, if a testing question is found to fail, but overall question accuracy remains high, the development team must meet and decide to either:

- edit the question to a working one
- retrain the neural network to fix that question

5. Features to be Tested

5.1 Front-end applications interface

- Ease of Use of GUI **MEDIUM**
- Access to website **HIGH**

- Access to mobile application **HIGH**
- Access to SMS **MEDIUM**
- Text box functionality **HIGH**
- Send button functionality **HIGH**
- Chat window functionality **HIGH**
- Network error functionality **HIGH**

5.2 API

- Reading of a GET request **HIGH**
- Returning of a proper JSON structure **HIGH**
- Returning the proper status code on good and bad requests **HIGH**

5.3 Database and information extraction

- Access to database **HIGH**
- Existence of information relevant to supported topics in database **HIGH**

5.4 Backend

- Error Handling for invalid questions **HIGH**
- Error Handling for errors in generating an answer **HIGH**
- Expected results for certain questions of supported topics **HIGH**
- Speed test for the system **MEDIUM**

6. Features not to be Tested

- **Arbitrary questions:** The test cases will test the system with specific questions.
- **Correctness of the data in the database:** The test cases will not test the database to ensure the information's correctness. These tests are just to ensure that data exists under the expected column names. However, if any incorrect information is noticed, it will be recorded as a bug and fixed in the future.
- **Accuracy of the system:** The accuracy of the questions is automatically tested when the neural network is trained.

7. Approach

7.1 Overall Test Strategy

All test cases from Test Case Document v1.0 will be run. The tester will then compare their actual result with the expected result. If the results are not identical, the test will be reported as

Failed, and a bug report will be created. The bug report will contain the ID of the test and the incorrect output that was received. If the results are identical, the test is reported as Passed. Additionally, testing of the database, API, and front-end interfaces should be completed before attempting the tests for the Backend. This is because any errors found in the first three items may create ghost errors in the Backend tests, as described in Section 9.

7.2 Testing tools

Most test cases require no special tools, as they will be run on our system directly. However, the tests pertaining to the database will be completed using the Stardog web application. This requires minimal special training: the tester just needs to know where to type the test queries, where the "query" button is, and where the outputs are displayed.

The test tools for each item are as follows:
Front-end applications interface: http://10.246.251.67:8080/chatbot
Android application
SMS phone number
API: http://10.246.251.67:8080/chatbot/api?query=
Database and information extraction: http://10.246.251.67:5820/chatbotDB#!/query
Backend: http://10.246.251.67:8080/chatbot

7.3 Regression testing

Regression testing will be done at the beginning of the week following a change in the code being made. If further changes or bug fixes are added while the testing is ongoing, the testing will continue using the version of the previous week.

8. Item Pass/Fail Criteria

A test cases' priority is defined to be proportional to the priority of the requirement that it is testing. A test case for a priority 1 feature has HIGH priority. A test case for a priority 2 feature has MEDIUM priority, and a test for a priority 3 feature has LOW priority. If a test case spans multiple features, then its has the priority of the highest priority feature.
If all HIGH-priority test cases under an item and 50% of MEDIUM-priority test cases pass, then the item is considered to pass. Low priority test cases do not impact the result of testing an item. For the entire test plan to be considered complete, all items must pass.

9. Suspension Criteria and Resumption Requirements

If any HIGH-priority test of the web interface fails, then testing of the Backend will need to be delayed until the relating bug is fixed. If any HIGH-priority test cases of the Database or API item fails, then testing of the Backend system should not be attempted, as it will most likely cause ghost errors. In order for testing to resume, the bug that caused the failing test case must be resolved.

10. Test Deliverables

The deliverables of system testing is the System Test Plan document and the Test Cases document.

11. Remaining Test Tasks

Development will still be ongoing for some features after the test period has already begun. As such, some test cases should be ignored until the feature they test has been implemented. Specifically, these test cases are:
TC 1.3
TC 2.8 - 2.13
TC 4.10 - 4.15
See section 14 for details on when each test case should be included in the overall testing.

12. Environmental Needs

The Drexel Chatbot website can only be accessed from the Drexel Network. The tester needs to be in the Drexel network or using the Drexel VPN to run the tests.
For most of the test cases, the tester will only need a web browser to complete all the tests.

13. Responsibilities

Each team member will only perform the tests for their assigned item:

- Front-end applications interface: Shishir
- API: Hoa
- Database and information extraction: Aaron, Daniel
- Backend: Tom, Nanxi

The System Test Plan was prepared by everyone in the team.

Nanxi is the testing lead. As lead, she has final say on any changes to this document and any questions/concerns about testing. Everyone is responsible to provide input and collaborating, but she is ultimately in charge.

14. Schedule

We allocate 5 weeks to perform system testing and bug fixes, while we continue to develop the remaining features. Test cases for existing features (see section 11 for list of ignored tests) will be completed during the first week, and then regression testing on the test cases is scheduled to be done as specified in section 7.3. A release version of the software will be created on every Sunday, and that version will be tested throughout the week. Any bug fixes or new features developed during the week will be included in the version created the following Sunday.

When new functionality is implemented, more tests will be added to the regression testing. The schedule below illustrates when test cases should be included:

Task Name	Start
Carry out System Test Plan	Tue 5/2/17
Add test cases for SMS (TC1.3)	Wed 5/23/17
Add test case for on campus dining and Drexel policies (TC2.8 - 2.13, 4.10 - 4.15)	Wed 5/30/17

15. Planning Risks and Contingencies

Due to lack of people, the same group of developers who wrote the code will also be responsible for creating and running the test cases. In an effort to avoid members testing their own work, developers who coded certain features will not be involved in testing those sections. In addition, we will be seeking users who were not involved in the development to test the system.

Given our tight schedule, if testing surfaces too many bugs for high priority features, we will not develop certain low priority features, namely on-campus dining and Drexel policies, to increase development work on bug fixing. This will be decided in a group meeting and based on how much time remains until the end of the project.

Because the plan is to create a new version every week there is a code change, there will be at most 5 versions of the software tested. If new versions continually add new bugs, the schedule may need to be changed to increase the frequency of regression testing, perhaps to twice a week. This will be decided in a group meeting.

Testing for the final planned features (campus dining and Drexel policies) are scheduled to start just 6 days before the final product submission. The product will be reverted to not include these features if testing reveals too many bugs to be fixed in the time frame. This will be decided in a group meeting and based on severity of the bugs found.

16. Glossary

- Chatbot: An interface, usually text based, specializing in the mimicry of natural language conversation. AKA “artificial conversational entity.”
- Ghost bug: A bug that is caused by one unit’s reliance on another unit that is buggy.
- GUI: Graphic User Interface, a type of user interface that allows users to interact with the software through graphical icons (e.g. buttons, etc.).
- JSON: JavaScript Object Notation, a data-interchange format that is commonly used in exchanging data over the Internet.
- SMS: Short Message Service, the text messaging protocol of cellular telephones.
- Standard English: the language that can be understood by English-speaking high school graduates.
- Web API: an application programming interface (API) for either a web server.