**CA400 Final Year Project**

**Testing Documentation**

**DCU Campus Chatbot**

**Sam Wood - 16476066**

**Mark Agnew - 16315811**

**Project Supervisor: Gareth Jones**

**Table of Contents**

[**1. Schneiderman’s eight golden rules**](#_whe07b82ad8p) **3**

[**2. Python unit testing**](#_tqa41yn5q7n9) **6**

[**3. User Testing**](#_5bs8t0cyetlm) **9**

[3.1 The Survey](#_nsd4npbww8zc) 9

[3.2 Follow-up Interviews](#_ykmofqxqcj8w) 11

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

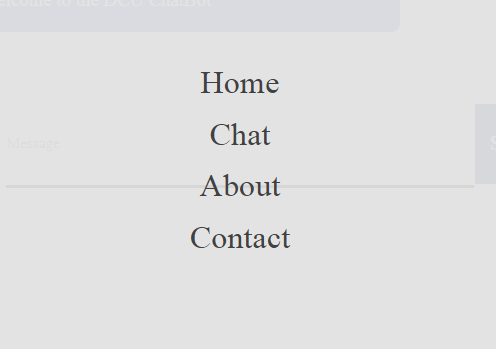
# 1. Schneiderman’s eight golden rules

Having learned all about Human Computer interaction and design in our third year HCI model, we were familiar with these 8 golden rules. Something we took away from that module is that these rules are a good way of analyzing a website so that it is appropriate for the use of our clientele.

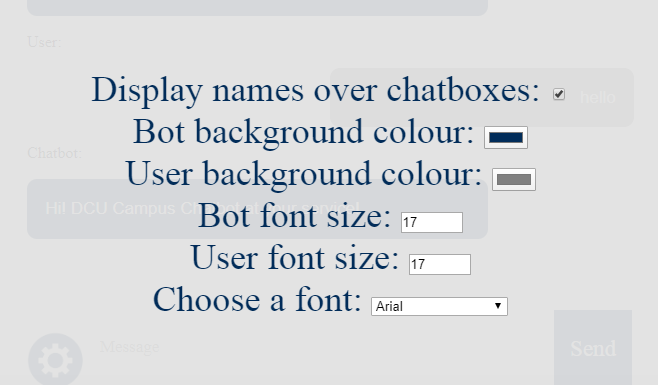
*Strive for consistency*

To ensure there was consistency in our user interfaces design we kept our pages colours similar throughout. For example, the colours of the button and top banner of our homepage are very similar to the writing on the DCU logo. Similarly, the white on navy design is seen on the chat page also, as the send button on the chat page is also this colour. With the writing detailing the apps functionality and writing of hints being of similar size makes it difficult for the user to become confused.

Also, the layout for our options menu and main menu is very similar. Both menus appear sliding from the left of the screen in the same fashion. This also maintains consistency. Please see the images below.



*Fig 1.1 Main menu*



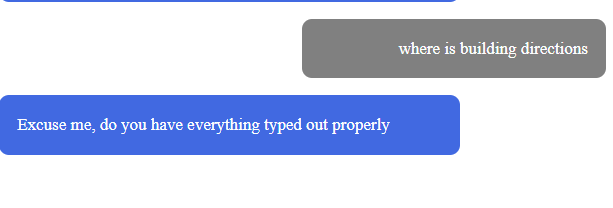
*Fig 1.2 Options menu*

*Enable frequent users to use shortcuts*

There is a higher demand for tasks to be completed quickly when there is an increase in a product’s use. For this reason we had to make sure that there were some shortcuts that make navigating easier for the user. Given that the majority of the applications usage is on the chat page, having the menu button in the top left corner of this page is an easy way to make shortcuts for the application.

*Offer informative feedback*

An important feature of the application is to let the user know what is going on at all times wherever they might be in the app. This feature is not absolutely difficult to implement given that our application is a system where the user says something which the app responds to in english. If the bot doesn’t know how to respond it will let the user know and ask to rephrase the query or ask something else. We feel this is an adequate representation of feedback for the user.

**

*Fig 1.3*

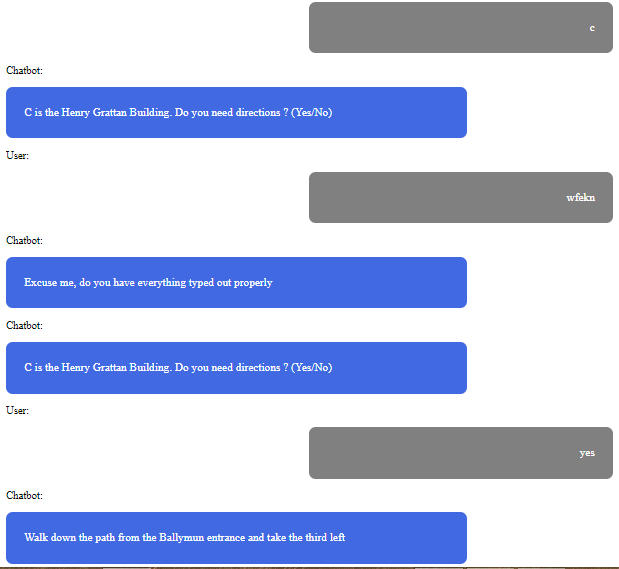
The user has not entered an appropriate query in the diagram above which has resulted in the bot asking for something else that it can understand.

*Design the dialogue to yield closure*

It is important to not keep the user guessing. In other words, to tell them what it is their action has just led them to do. This is similar to our last point. Whenever the user asks a question the response has just presented itself, the user is aware and thus there is no closure. This is also seen in the chat settings box when the user changes the response sections colour or the text-size. Both are done automatically and are perfectly apparent to the user.

*Offer simple error handling*

It is always beneficial to provide users with a simple, intuitive means of error-handling. One way we saw to this was with our traceback function. This function, written in python in our Chat.py file. This feature allows the user to respond to the second last message sent by the bot if their last message was misspelled or the wrong response for whatever reason. See images below for an example.



*Fig 1.4*

As you can see when the user incorrectly spelled the word ‘yes’, the traceback function recognises the ‘what’ response it replies to the user and checks the following query to see if it is compatible with the users original query. This reduces the work required by the user to begin their query again from scratch.

*Permit easy reversal of actions*

Again our product is a chat interface, thus once the query has not been interpreted, the most damage it can do is result in one of the what responses. Hence or otherwise, the user is immediately presented with a second chance to do what they had originally intended, all the way throughout their use of the application.

*Support internal locus of control*

This rule addresses the concept of the user feeling they are constantly in control. Given the options for the user ie. the settings menu and the fact the bot will not message the user unless spoken to (apart from the greeting message), we like to think the user would feel they are in control when using our DCU chatbot.

*Reduce short-term memory load*

Human attention is limited and we are only capable of maintaining around five items in our short-term memory at a time. Because of this, we have made the previous messages between the user and the chatbot scrollable. For example if the user has gotten their directions for their destination, they can scroll back up to it with ease if they forget what the message said.

# 

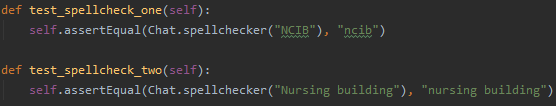
# 

# 

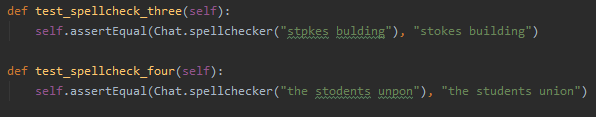
# 2. Python unit testing

The python code of this project handles both the front end of the project by using flask to display the correct html files. The other part of the python file helps to integrate the back end of the main application to the javascript contained within the chats webpage, it retrieves the users input and then using a function it tries to correct any spelling errors. It then passes that corrected message to the library used to interpret the aml files to see if it matches any patterns and if so provide the correct response. To test these various features of the python code we created three classes of unit test: first to make sure words that are misspelled are properly corrected, the second times the time taken to execute the programs different functions and lastly to ensure that the chatbot/aiml always provides the correct response for given queries.

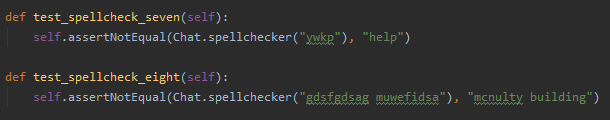
For the first class of test cases a selection of phrases the user may enter and ensure that they are incorrectly altered by the spell checker, phrases that are misspelled and ensure that they are corrected properly and finally test phrases so badly misspelled that they should not be corrected properly. Below are examples of each of these three cases.

**

*Fig 2.1 Two examples of correct spellings that should stay as they are*

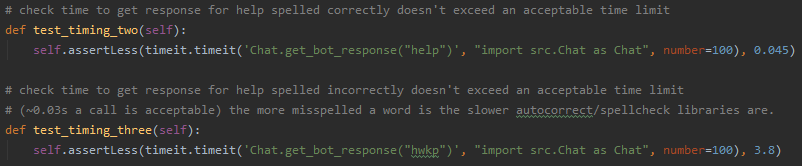
**

*Fig 2.2 Two examples of words misspelled but should be corrected.*

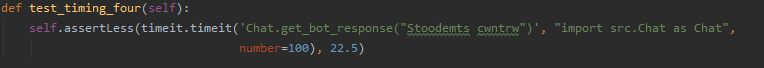
**

*Fig 2.3 Two examples of words badly misspelled that shouldn’t be corrected.*

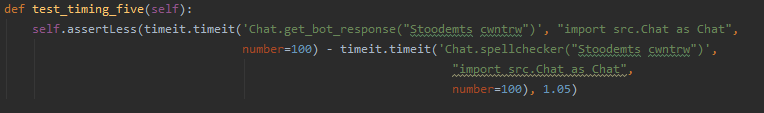
The second class of test cases focused on the execution time of the functions in the python code, this was to make sure that there was not too much of a delay on the python side of retrieving a response for the user. No response takes more than half a second but most of the time is spent in the execution correcting any misspelled words, this gives the user the feedback they need in a timely manner. Below are some examples of the unit tests we implemented.



*Fig 2.4 Time taken to respond to a correctly spelled query(top test) and an incorrectly spelled one (bottom)*

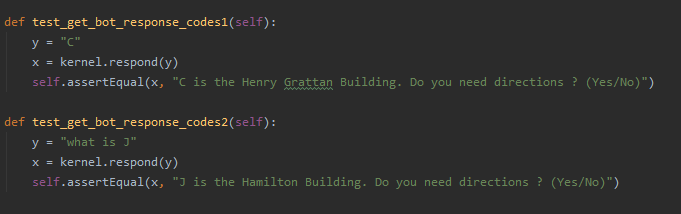
**

*Fig 2.5 The more misspelled a word or query is the longer it takes the bot to respond*

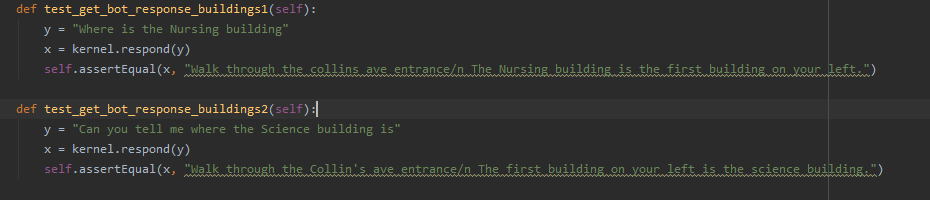
**

*Fig 2.6 measure execution time of a response minus the time to correct*

We have also implemented a class of tests which tests the responses of the bot. We created a series of tests for each feature. The first was for our building codes. This section consisted of a series of unit tests that were differently phrased to acquire the building codes. We also included a test where we tested the user responding ‘Yes’ to the reply and where directions are sent. Following this we started to test the building directions. This was also a series of tests where the queries were different phrases for getting directions to buildings on campus.

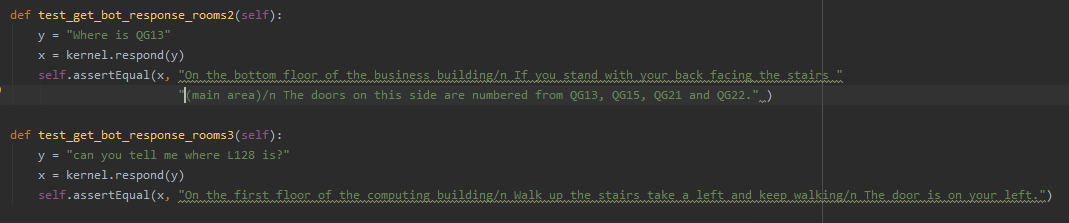


*Fig 2.7 Building code unit tests*

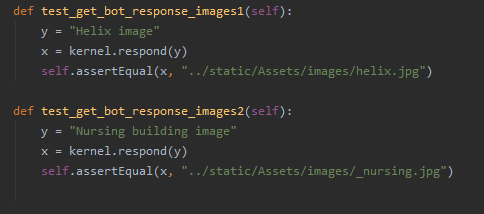


*Fig 2.8 Building directions tests*

This set of tests had shown us that our files for directions to the buildings and lecture rooms was limited to only one way of phrasing the query. As a result, we had to add to the AIML files so the user had a wider range of phrases to use this feature. A snippet of our tests for the lecture room file can be seen below. We also tested that when images of the buildings are requested that the correct address for retrieving the images is sent.



*Fig 2.9 Room direction tests*

**

*Fig 2.10 Image tests*

We also carried out several unit tests for the remaining files which were for greetings, frequently asked questions, basic chat and about buses called “test\_greeting”, “testFAQ”, “test\_basicchat”, ”test\_buses”.

# 3. User Testing

In order to carry out more testing we conducted user testing which involved getting users to try our web app and provide feedback via a voluntary survey and follow-up interview. This user testing would highlight any integration problems between the python code and the JavaScript in the HTML files, It would also highlight any issues with the UI design and compatibility of the website itself.

## 3.1 The Survey

We created a survey to gather feedback from users, the survey covered a few different areas of our project such as UI design, functionality, accessibility, compatibility and usefulness. In terms of testing it was used to spot any errors between the user accessing the website and the back-end code retrieving their message and returning a response. No user seemed to highlight any issues in this area, which we used as an informal method of integration testing. It showed that there was nothing major that would impact a users experience showing that the integration between the front and back-end of our code was working correctly, we needed this validation as it was hard to find a usable testing method to test across both the underlying JavaScript of the website and our python link to the interpreter.

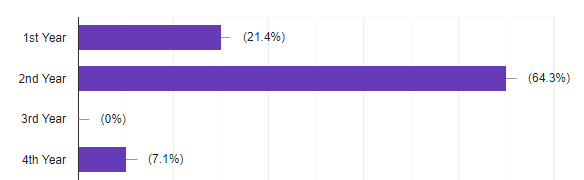
In order to gauge compatibility we asked users what device and browser they were using. Mobile devices were the most popular choice of device and google chrome the most popular browser, mobile devices and google chrome made up over 50% of those surveyed as shown below with the devices.



*Fig 3.1 user devices*

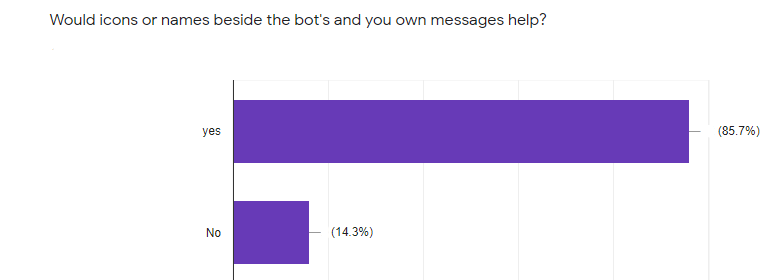
This highlights the usefulness of developing a web application that can be used both on mobile as well as the users own laptop or the desktops on campus.

To ensure we were testing our project on the correct demographic we also asked participants what year of their degree they were in just over 75% of these were either in 1st or 2nd year so it means the majority of feedback came from those that would be likely to be using it the most.

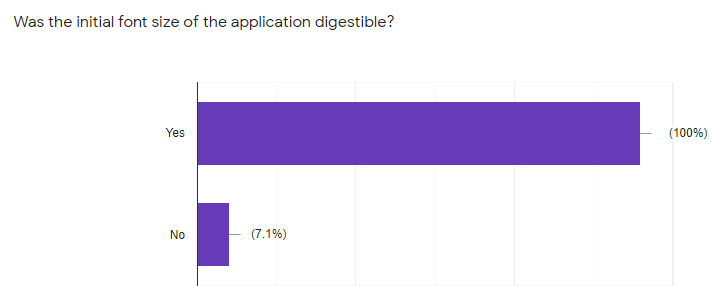


*Fig 3.2 users degree year*

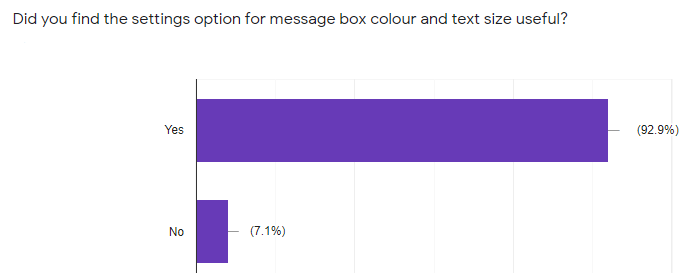
We also asked users about certain design features to make sure that they were the right choice for users, these features included labelling the messages as chatbot and user, font size, the ability to change settings and the splitting up of long replies. The responses to these questions are shown below.



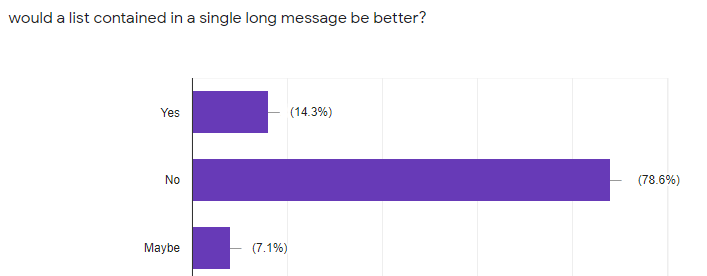
*Fig 3.3 Based on this feedback we implemented the names above each message with the option to show or hide the names also.*



*Fig 3.4*

**

*Fig 3.5*

**

*Fig 3.6 This was asked after asking users their opinions on the messages being split up(i.e the reply being a few separate messages)*

## 3.2 Follow-up Interviews

After the survey we then followed up with those who volunteered with an interview over zoom. We also went into greater detail in the areas discussed in the survey and asked new questions which couldn’t be asked in a survey, the enabled us to focus on what needed to be changed and gave us insight into the direction the project could go in the future and highlighted restrictions of changes we couldn’t make due to Covid-19.

The first person we interviewed had several visual impairments such as Severe Convergence insufficiency, photosensitivity, short sightedness and suffers from migraines. This person helped us to find aspects of the UI we needed to alter slightly in order to make the website more accessible to those with similar impairments. These aspects included the font type, size and colour in order to make the text more readable.

Another person we interviewed towards the end of the project due to exams taking place shortly after our survey, was happy with the design changes we had made and said that the only issues he had with queries were ones to do with directions to lecture rooms that we had not been able to include due to campus being closed due to Covid-19. So at this point between the user survey responses and our follow-up interviews we were happy that there were no major bugs. This interview also provided us with more ideas of the direction in which the project could go which we will mention in our technical guide. Also this person commented on the usefulness of such an application saying they would have used it in first year themselves had something such as this existed.