Improving the usability and accessibility of Fuzzy Logic software systems with a web-based approach

Submitted MAY 2014 in partial fulfilment of the conditions of the award of the degree MSci (Hons) Computer Science

Craig Knott
cxk01u

With Supervision from Jon Garibaldi

School of Computer Science and Information Technology
University of Nottingham

I hereby declare that this dissertation is all my own work, except as indicated in the text:

Signature ____________________

Date ______/______/______

Abstract

This project was both a practical implementation of an online fuzzy logic software system, and a research piece into the optimal design of a fuzzy logic system, and how fuzzy logic can be best introduced to novices of the field. This report details the implemented of an online system for the creation and evaluation of fuzzy logic sets and systems, using mostly JavaScript and JQuery, with an R back end to deal with the processing requirements of fuzzy logic.

In the end, it was found that the majority of participants much preferred a graphical user interface when working with fuzzy sets and systems. This is due to the ability to easily visualise the sets, and the structure of the system being produced, along with how these interfaces can be designed to aid the user. Specifically the advantages of the system presented in this report were the clean uncluttered interface, and the ay the tasks to be completed were structured in a logical order but still allowing the user full control over their experience.

Words in text | 19,013

Calculated with the TeXCount web service
`http://app.uio.no/ifi/texcount/online.php`

Contents

1 Introduction

Fuzzy logic is an ever expanding field, and as such, the tools we are using to work in this field should also be expanding. It is also important that the merits of fuzzy logic are made apparent to those other than the experts of this field, as this would help to produce more advanced control systems in the future.

Many software systems for working with fuzzy logic have already been produced, of which many different approaches have been attempted, and been successful to various degrees. Examples of such systems include: The MATLAB Fuzzy Toolbox[1], An R Package named FuzzyToolkitUoN[2], XFuzzy[3], and fuzzyTECH[4] (a more comphrehensive overview of these systems can be found in section 3.2).

These system are all worthwhile pieces of software, and they fulfil their main objective of allowing for the creation of fuzzy systems. However, whilst researching these systems as part of my second year group project at the University of Nottingham (and actually working on one, in the case of FuzzyToolkitUoN), I noticed that there were two key flaws that the majority of popular fuzzy software systems suffered from: difficulty of use, or difficulty of access (or even both).

The main objective of this project is to produce a software solution for the creation, manipulation, and inferencing of a fuzzy logic system, which is accessible online. With a specific focus on solving the issues that are faced by fuzzy logic software systems that are currently used (difficulty of access and use).

Many different techniques will be employed in solving these fundamental problems, to hopefully create a system that is as easy to use, and as easy to access, as possible. Some of these techniques will include: online access; the ability to work with multiple file types, for cross compatibility; an intuitive design; unrestricted navigation, giving the user complete control and freedom; a dedicated, unobtrusive help system, to offer help to those that need it, but not to bother those that do not; and to build it in a way that allows for future expansions. A comprehensive list of all aspects of the software system can be found in section 4.

It could be argued that *another* fuzzy logic software system is not necessary, as it has been demonstrated that there are already many systems available. However, the currently available software suffers from the key issues identified above, and this project aims to resolve these issues, and attempt to spread the influence of fuzzy logic to those other than experts in the field.

There will, however, be certain areas that this software system will *not* be focusing on, as these are not relevant to the question posed in this research. Namely, this project will not be focusing on higher levels of fuzzy logic; it will only be focused on type-1. This is because the leap in difficulty from type-1 to type-2 fuzzy logic is very large, and type-2 is simply not a concept that is suitable or appropriate to introduce beginners to. More on this topic, including a definition of both terms, can be found in section 8.3.

[1] `http://www.mathworks.co.uk/products/fuzzy-logic/`
[2] `http://cran.r-project.org/web/packages/FuzzyToolkitUoN/index.html`
[3] `http://www2.imse-cnm.csic.es/Xfuzzy/`
[4] `http://www.fuzzytech.com/`

2 Motivation

The motivation of this project is a simple one: to produce a fuzzy logic software system that is easy to access, and easy to use, to help promote the wider adoption of fuzzy logic. The problem with systems currently available is that they suffer from one of the two following pitfalls: difficulty of use or difficulty of access. This means that novices can find it very difficult to get into the field, the software available does not facilitate productive use, and even experts can be held back by the software they are using. Some specific issues include: locating systems to use, complex installation processes, cost to the user, unintuitive user interface, or a requirement of (a considerable amount of) prior knowledge.

As part of my second year group project at the University of Nottingham, I worked on an R Package called FuzzyToolkitUoN. The goal of this system was to expand upon work completed by the Intelligent Modelling and Analysis group[5], to facilitate the use of fuzzy logic within the R programming language [28]. Whilst working on this project, a large amount of research into existing fuzzy logic software systems was conducted, and it was during this research period that the two key commons flaws were noticed. Unfortunately, due to the nature of the R programming language, and the package being produced, FuzzyToolkitUoN also succumbed to one of these pitfalls - difficulty of use. Personally, I was frustrated with this, and that is one of the reasons for the birth of this project - remedying past mistakes.

As have been mentioned, the greater adoption of fuzzy logic would be extremely beneficial, as it adds a new level of reasoning that classical logic simply cannot. As such, another side goal of this project is to make a system that is as easy to use as possible, regardless of the skill of the user in both terms of knowledge of fuzzy logic, and of using computer software in general. This will mean a project that will not only be very easy to access and use, but also help novices to learn about what they are doing, as well as why they are doing it, to help them gain a greater understand of the field of fuzzy logic.

The project detailed in this dissertation will aim to implement a fuzzy logic software system, in a novel format (online), and to specifically avoid the common pitfalls observed of other similar systems. Being online, the system is already on the right path to solving the difficulty of access problem, as the users will be able to access the system from wherever they are, and on what ever platform (as it will not require any plugins, like Java, or Flash). This also means it is more accessible to the novice user, or the computer novice, as they need only navigate to a website to use the system; there is no complex download and installation process.

The project will be heavily influenced by the field of Human Computer Interaction, to ensure that the system is as user-friendly, and easy to pick up as possible. User interaction with a software system is extremely important, and the way systems are designed has a huge impact on how they are received by the user base. Simplicity is important in this design, because studies have shown that users lose more than 40% of their time to frustration, and that in most of these cases, the user ends up angry at themselves, angry at the computer, or feeling a sense of helplessness [14]; which is obviously not ideal for a system that is attempting to help the user learn.

[5] `http://ima.ac.uk/`

3 Background Information & Research

3.1 What is Fuzzy Logic?

Fuzzy logic is a "natural" way of expressing uncertain or qualitative information [1]. It is a form of logic that deals with approximate reasoning, as opposed to fixed, exact values, like those found in classical logic (where we may only have properties being true, or false). Instead of these strict truth values, fuzzy logic systems have a range of truth, between 0 and 1. This makes fuzzy logic much better for handling and sorting data, and is an excellent choice for many control system applications, due to the way it mimics human control logic. Lotfi Zadeh, who formalised fuzzy logic in 1965, states that the key advantages of fuzzy logic are that it allows us to make rational decisions in environments of imprecision, uncertainty, and partiality of truth, and to perform a wide variety of physical and mental tasks, without any measurements or computations [31].

In a classical set, the membership, $\mu_A(x)$ of x, of a set, A, in universe, X, is defined:

$$\mu_A(x) = \begin{cases} 1, & \text{iff x} \in \text{A} \\ 0, & \text{iff x} \notin \text{A} \end{cases}$$

That is, the element is either in the set, or not. In a fuzzy set, however, we have grades of membership, which are real numbers in the interval, $\mu_A(x) \in [0, 1]$. Every member of a set has a membership grade to that set, depicting how true the property represented by that set is, for the given member [30]. The traditional syntax for representing members of a fuzzy set is given below (although a full working knowledge of fuzzy logic theory is not necessary for this project).

$$A = \mu_A(x_1)/x_1 + ... + \mu_A(x_n)/x_n$$

The easiest way to observe the merits of fuzzy logic is to look at terms that we humans use in our everyday life, and attempt to map these as crisp functions. For instance, terms like "hot", "cold", "tall", and "short", are all terms that we understand very well, and use often. However, if we were asked to give *exact* values for tallness, or shortness, we would not be able to do so. At what cut-off point would a person change from being considered short, to being considered tall? Fuzzy logic helps to alleviate these impossible choices, by having varying degrees of membership, for certain properties. The example in figure 1 shows this using three linguistic variables to describe the height of a person. Instead of at one point being either tall, short, or medium height, we, at all times, belong to all properties, to a differing degree.

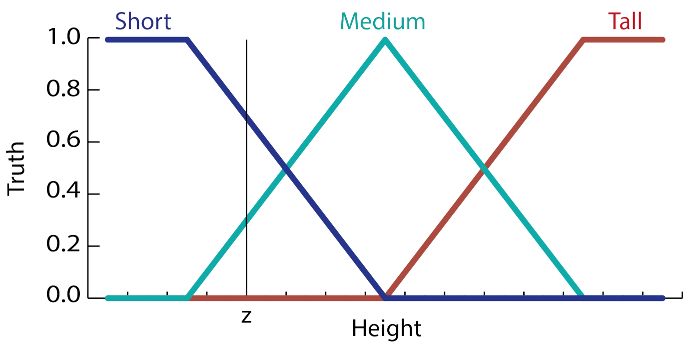


Figure 1: A fuzzy set depicting "height"

For instance, at the point labelled z, in the sets in figure 1, the membership to the "Short" set is 0.7, the membership to the "Medium" set is 0.3, and membership to the "Tall" set, is 0.0. This is, naturally, much more precise than simply saying we are "Short", "Medium", or "Tall".

3.2 Existing Systems

Fuzzy logic has been around for almost 50 years now, and, with the rising age of the computer, it would be alarming if no software systems for its usage were in circulation. Luckily, this is not the case, and there are many examples of software systems focusing on the use of fuzzy logic, of which many different approaches have been attempted, to varying degrees of success. In this section, a number of these software systems will be evaluated, to discern their positive and negative qualities, to help improve the design of the project presented in this report.

MATLAB Fuzzy Toolbox
The first system to be explored is MATLAB's fuzzy toolbox, an add-on for the MATLAB software suite, to work with fuzzy sets and systems. This toolbox provides everything required to create type-1 fuzzy sets and systems, with relative ease. The main advantage it has over most other systems is that it has a graphical user interface, which makes a task like working with fuzzy sets (that require a lot of visualisation and updating in real time) much simpler. There is also an extensive library of documentation and tutorials available for both MATLAB, and this specific toolbox, that help novices to get acquainted with the system. These things both help to make the system very easy to use, and novice friendly.

Unfortunately, these positives do not outweigh the major disadvantage of MATLAB, and it's fuzzy toolbox; which is that are pieces of proprietary software. This means that a novice to the field of fuzzy logic would have to invest a considerable sum of money, before they could even begin using the software. Whilst the system does have extensive documentation, and the user would be able to understand and use the system with relative ease, a piece of software does not require a large price tag to achieve this level of functionality and support. Another disadvantage of the MATLAB fuzzy toolbox is that is it not a dedicated piece of software, and is instead a limited subsection of the greater software of MATLAB. This means that the potential for extensibility is much less likely, as updates to the encompassing software would be deemed more important. It could even be argued that the installation of the MATLAB software, and then the installation of further software could be confusing to some novice users, which further alienates them. A final issue that has been observed is the excessive number of separate windows that are opened whilst working with the MATLAB fuzzy toolbox, as a new window is opened for each individual task that is being worked on (for instance, one for membership functions, one for rules, one for evaluations, which clutters the user's workspace rather rapidly).

FuzzyToolkitUoN
FuzzyToolkitUoN is an R-Package, produced by the Intelligent Modelling and Analysis group, at the University of Nottingham, that I personally worked on as part of my Second Year Group Project, at the aforementioned University. It provides functionality for the R Programming language to allow for work with fuzzy sets and fuzzy systems, including their evaluation. A major milestone for the project was it's official acceptance onto CRAN (The Comprehensive R Archive Network, an online library for R packages), in 2013. Being written in R, the package has access to the very powerful R processing tools, and graphics drawing capabilities, which help the user to visualise the system they are creating. Due to being hosted on CRAN, there is extensive documentation for every function in the package, including example usages, and explanations of all their parameters. This makes the usage of the package much simpler, because the user can easily access the documentation for any function they require, and there is a fully worked example for them to follow. Another advantage of the CRAN hosting is that any user with an R interpreter installed can access the package with only a few simple commands, helping the system to be much more accessible.

Unfortunately, there are downsides to working with the R language, the most prominent of which is the command line interface. Whilst the users have the features and functionality to plot the sets they are drawing, it can still be a cumbersome task, and does not promote ease of use. In a graphical user interface, the updating of graphs would be automatic, and the user could see their changes in real time, instead of having to modify a set, and then check what it looked like separately. Another issue with the package is that the descriptions of the functions, and the system itself, are specified so that a novice to fuzzy logic is not given the support they require; the system assumes that the user's knowledge of fuzzy logic is already somewhat sufficient.

XFuzzy (3.0)
Developed by the Institute of Microelectronics in Seville, Spain[6], XFuzzy (3.0), is a set of several tools, that cover the different stages of the creation of a fuzzy system. It allows for the construction of complex systems, whilst also offering flexibility. Each of the tools can be executed as an independent program, or, using the XFuzzy environment, can be integrated together as a graphical user interface, to ease the design process. As mentioned in the evaluation of the MATLAB fuzzy toolbox, a graphical user interface is a huge aid to the user in constructing a fuzzy system, as it allows them to see what changes they are making, and what effect these changes have. The software also runs on Java, which means it can be used on any operating system, as long as the Java Runtime Environment is installed, making it relatively easy to access.

There are, however, some issues with this software. These being that it is relatively unknown, is not updated frequently (the last update for version 3.0 was in 2012), and without using the graphical user interface, you are stuck using a command line interface, and must learn the system's bespoke language, in order to complete any tasks. The final disadvantage (which is also listed as an advantage, from a different view point), is the necessity for Java to be installed. This is an additional piece of software that this product is dependant on, and could further confuse the novice user with extra installations necessary.

fuzzyTECH
The final system that was observed was fuzzyTECH, another proprietary software, produced by INFORM[7]. It comes with a graphical user interface, making the interaction with the system much simpler than other software available. It also boasts that the application programmer does not require an understanding of fuzzy logic *or* programming to be able to use the system adequately - a feature that the software detailed in this report also hopes to boast. Another advantage of fuzzyTECH is that it produces source code that can be used on various hardware platforms (for instance, PCs and micro-controllers).

The main disadvantage of fuzzyTECH, just as with the MATLAB fuzzy toolbox, is that it is a pay-to-use piece of software. This greatly alienates the novice user, as they may not be willing to spend a large sum of money on a software they have never used, for a purpose they have never experienced before. There are also many different versions of fuzzyTECH, which would further confuse the novice user, as they may not necessary be aware of their specific needs, and thus be unsure as to which specific version would suit them best. In addition to this, fuzzyTECH has not been updated since early 2010, meaning the support that the novice user requires, may not be there - despite the systems extensive support functions.

[6] `http://www.imse-cnm.csic.es/`
[7] `http://www.inform-software.com/`

3.3 Platforms and Tools

This section introduces some potential platforms and tools that could have been used in the project, along with justifications for and against. The system itself consisted of two major parts, the web front end, and potentially some server back end (at the research stage of the project it had not yet been decided whether the application was to be entirely client-side or not). Both front end tools (including design frameworks and programming languages) and back end tools were explored and evaluated during this process to discern which would be the most appropriate for this project. Further to this, there was some research into graphing tools to be used on the website, as drawing fuzzy sets as graphs is an extremely powerful and intuitive way to represent them. A full list of the final decisions of tools to use, including their justifications, can be found in section 6.1.

3.3.1 System Back End

R-Node
R-Node[8] is a web front end that allows access to an R instance, providing full access to the R language, and the tools it provides. This means that there could be direct interaction with the FuzzyToolkitUoN package, alleviating a lot work on the back-end side of the project. This would greatly simplify the production process, as a brand new inference engine would not need to be produced. A particularly useful tool that FuzzyToolkitUoN provides is the drawing of graphs, which would be a huge usability boost to the system (as visualising fuzzy sets as graphics is extremely useful). FuzzyToolkitUoN is also currently under further development, so more features could easily be integrated into this project, if it were used for the back end. Unfortunately, the website for R-Node has been unresponsive for some time now, so finding documentation, or an actual download link is impossible. This shows that the tool is not very well maintained, and any issues discovered whilst using it would not likely be resolved in a reasonable time frame.

R-Shiny
R-Shiny[9] is an easy way to create web applications, using the R programming language. Like R-Node above, the advantages of using R would be the use of the FuzzyToolitUoN package, allowing for direct access to tools for the manipulation of fuzzy sets and systems. R-Shiny is currently in development, which means it lacks a lot of important features that would make its use much simpler, however, the development team are very active both in regards to responding to bug reports, and answering questions on their discussion board. R-Shiny allows for the creation of entire web pages using entirely R, or the use of just R as a back end, allowing for a much more dynamic web front end (created with HTML). Unfortunately, despite this more dynamic web front end, R-Shiny still does not provide an adequate way to create fully dynamic websites, which could be an issue for this project.

Node.js
Node.js[10] is a platform built on JavaScript that allows for the construction of network applications with ease. It is quick and easy to set up a basic server, however, can become quite confusing and difficult to set up a more complex one. Luckily, there is a lot of support available for Node.js, as it is becoming more and more popular among developers. This means that any issues that were encountered would hopefully be solvable within a reasonable time frame, and would not hinder the project too greatly. However, unlike the other back ends discussed, the use of Node.js would mean the construction of a new, bespoke, JavaScript based fuzzy inference system, which would be a significant increase to the work required.

---

[8] `http://squirelove.net/r-node/`
[9] `http://www.rstudio.com/shiny/`
[10] `http://nodejs.org`

3.3.2 Front End Programming Language

Naturally, the website would be built using HTML 5 and CSS 3.0, as they are the defacto standard languages for the construction and design of websites. However, the most appropriate language to provide the actual functionality to the website required some research.

JavaScript
JavaScript is an obvious choice when looking to provide functionality to a website, due to it's high level of integration with HTML. It is a client-sided programming language used to alter the content of a displayed document, or in this case, website. The main advantage of JavaScript is it's ability to directly and easily manipulate the content of a web page, by accessing the document object model. The basic semantics and syntax of JavaScript are very similar to many other languages, meaning adopting this language is very easy to both expert and novice programmers. It also provides advanced features and functionality, that help to produce extremely powerful systems with relative ease. Using JavaScript also allows access to other JavaScript tools, such as JQuery. The disadvantages of JavaScript are that there are security issues (although this is not an issue with the system proposed in this project), and there is potential that JavaScript can be rendered differently depending on layout engine, causing inconsistencies.

Adobe Flash (ActionScript)
Adobe Flash is a tool for the creation of graphics, animation, games, and rich internet applications, which require the Adobe Flash Player to be viewed. This allows for the creation of websites that are both aesthetically pleasing, and are interactive for the user, making the entire experience of using the website much more pleasing. However, HTML 5 is now at the stage where it can almost perform the same functionality as Flash, negating it's necessity. It also requires that the user have Adobe Flash Player installed, which, whilst free, is an extra unnecessary download that could stand to make the system more difficult to access (for optimal accessibility, the project would require no additional downloads).

3.3.3 Front End Design Framework

A front end framework, in the context of web development, is a standardised set of concepts, practices and criteria for dealing with the production of the front end of a website. There are many advantages to using pre-made front end frameworks for developing web applications, the most prominent of which are that: they generally look more aesthetically appealing than a single individual would be able to produce, they provide access to a large selection of easy to implement dynamic user interface elements, and many of them deal with browser cross-compatibility issues automatically.

Semantic
Semantic[11] is a web front end development system produced by Jack Lukic. The main advantages of semantic are that is it tag agnostic (meaning any tag with UI elements can be used), and that it has a variety of elements with real-time debug output, allowing the application programmer to see what effect their code has. It also has the advantage that it isn't Bootstrap, spoken about below, which is often cited as being overused; this would give the project a slightly more unique feel. Unfortunately, as this is a less well known, and less well developed framework, there are much fewer UI elements to choose from, in comparison to a framework like Bootstrap, and the entire framework is less documented, and there is less online help (due to it's lower adoption rate).

[11] `http://semantic-ui.com`

Twitter's Bootstrap
Twitter's Bootstrap[12] is an extremely popular web development front end (so popular in fact, that some individuals are beginning to complain about it's overuse [9]), which helps to produce responsive websites, that function both on desktop and mobile devices. It is designed so that anyone of any skill level, beginner to expert, can pick up the framework and begin creating websites with relative ease. It provides a framework that produces websites that easily and efficiently scale for a multitude of devices, such as phones, tablets, and desktops, helping to make them much more accessible (although, not entirely relevant for this project, as there are many logistical issues with attempting to create a fuzzy logic system from a mobile device). There is also extensive and comprehensive documentation available for Bootstrap, including live working examples, and ready-to-use code samples, making its adoption into any project extremely simple. As mentioned above, however, Bootstrap is so popular nowadays that many people find it's usage off putting, as a large number of websites are beginning to look like clones of one another. For this reason, if Bootstrap were to be used in this project, it would be important to ensure that some customisations to the basic styles were applied, so that the website looked unique.

HTML KickStart
HTML KickStart[13] was the final web front end that was evaluated. Like Bootstrap, this was a responsive and scalable framework that meant it was suitable for devices ranging from desktops, to mobiles (even if the application itself wasn't). It also comes with over 240 icons, which can be used to help the design and usability of elements on the website (as people generally associate certain icons with certain tasks), and the entire framework can be included with just two files (instead of Bootstrap that requires some level of configuration to ensure all possible features are included). On the other hand, this framework provided much fewer elements than other packages (like Bootstrap), and the documentation presented on the framework's web page was confusing and laid out poorly (which is ironic, as the website was built using HTML KickStart), which made searching for specific tools much more troublesome.

3.3.4 Graphing Tools

The drawing of fuzzy sets is extremely intuitive and powerful in a graphical format. For this reason, a suitable tool for the drawing of graphs on a website was required for this project. This would help to make the system considerably easier to use (as the user would be able to see their work as they were completing it), would take advantage of the graphical interface, and help to improve the general aesthetics of the system.

Directly from R
As mentioned in the back end section, there was potential to use the R language as the back end for the system. This meant that any R libraries could be imported and used for the front end. In the case of this system, the use of FuzzyToolkitUoN would allow for the work with fuzzy logic and fuzzy sets, without the need to write an entirely new client-sided inference system from scratch. Using FuzzyToolkitUoN to draw the graphs would promote uniformity between the back end and front end (if the back end did end up being R based), and would mean the implementation of drawing graphs would not need to be considered (as this would be handled by the R package). However, transferring graphical data from an R back end to a web front is a difficult task, and the graphs themselves would be simple, static images, where dynamic or interactive graph would be much preferred.

[12] `http://getbootstrap.com/`
[13] `http://www.99lime.com/elements`

Google Charts
Google Charts[14] is a tool for creating and displaying highly customisable graphs and charts on a web page. This package provides a large range of graphs and charts, all of which have interactive elements (helping to improve usability by a considerable margin, and reduce screen clutter). As this service is provided by Google, there is a large library of support available, and it can be assumed that the service is of a high quality. The only negatives of this tool are that the API isn't as simple to use as it potentially could be, and the returning of graphs can sometimes be slow.

Flot
Flot[15] was the final graphing tool that was evaluated. It is written, and accessed, through jQuery, which means any project using JavaScript can easily incorporate it. The project is in constant development, and bug reports filed by the users are taken into account, as well as their general suggestions. However, the package itself can be difficult to get started with, and there are much fewer graph types available, compared to other services.

[14] `https://developers.google.com/chart/`
[15] `http://www.flotcharts.org`

4 System Specification

As has been mentioned multiple times, the system to be produced is an on-line fuzzy logic inferencing and visualisation system, with the specific goal to be as user friendly, and as accessible as possible. In this section, the functional requirements (goals that the software must achieve), and non-functional requirements (constraints placed upon the system) of the system have been enumerated. This allows for the reader to see what functionality and features the system will have, but also provides a list of test criteria for later stages of the project.

4.1 Functional Requirements

In this section, the functional requirements of the project have been enumerated, so that they can be referred to at the evaluation stages of the project.

1. Users can manipulate membership functions
    - 1.1. Users will be able to create membership functions, of type...
        - 1.1.1. Gaussian
        - 1.1.2. 2-Part Gaussian
        - 1.1.3. Trapezoidal
        - 1.1.4. Triangular
    - 1.2. Users will be able to add membership functions to variables
    - 1.3. Users will be able to edit membership functions, including:
        - 1.3.1. Membership function names
        - 1.3.2. Any function parameters
    - 1.4. Users will be able to delete membership functions from variables
    - 1.5. Users will be able to access help on how to create membership functions
    - 1.6. Users will be able to see a plot of their membership functions
2. Users will be able to manipulate linguistic variables
    - 2.1. Users will be able to create linguistic variables
        - 2.1.1. Users will be able to create input variables
        - 2.1.2. Users will be able to create output variables
    - 2.2. Users will be able to edit the range of linguistic variables
    - 2.3. Users will be able to delete variables
    - 2.4. Users will be able to rename variables
    - 2.5. Users will be able to access help on how to create variables
3. Users will be able to manipulate system rules
    - 3.1. Users will be able to create rules for the system
        - 3.1.1. Users will be able to specify rule terms
        - 3.1.2. Users will be able to negate certain terms in a rule
        - 3.1.3. Users will be able to change the weight of a rule
        - 3.1.4. Users will be able to specify the connective to be used in the rule
    - 3.2. Users will be able to edit any previously constructed rules
    - 3.3. Users will be able to delete any previously constructed rules

3.4. Users can access help on how to create rules

4. Users will be able to manipulate system-wide parameters

    4.1. Users will be able to edit the system-wide parameters

        4.1.1. Name of the system
        4.1.2. Type of evaluation to use
        4.1.3. "And" method to use
        4.1.4. "Or" method to use
        4.1.5. Aggregation method to use
        4.1.6. Implication method to use
        4.1.7. Defuzzification method to use

    4.2. Users will be able to access help on what affect these changes make

5. Users will be able to perform file input/output on the system

    5.1. Users will be able to export their system to various formats, including

        5.1.1. MATLAB .fis file
        5.1.2. FuzzyToolkitUoN .fis file
        5.1.3. JSON Object file

    5.2. Users can access help on how to export files and what is supported

    5.3. Users will be able to import previously made systems of various formats, including

        5.3.1. MATLAB .fis file
        5.3.2. FuzzyToolkitUoN .fis file
        5.3.3. JSON Object file

    5.4. Users can access help on how to import files and what is supported

6. Users will be able to evaluate their system

    6.1. Users can provide a value for each input, and receive the output value

    6.2. Users can access help on how the evaluation process works

4.2 Non-Functional Requirements

In this section, a list of the non-functional requirements, that place constraints on the system, is provided.

Accessibility
This is, naturally, a huge goal for the system, as it was one of the reasons for it's conception. The proposed system is to be made available entirely on-line, allowing anyone with internet access, and a computer, to use the system. The system's accessibility will be further increased by it's lack of client-side dependencies, meaning the user is not required to download or install any additional software if they wish to use the software. The only potential issue with accessibility is if the server that is hosting the website was to stop functioning. This is, however, a potential issue that does not have a solution.

Usability and Operability
Due to the large range of potential users, the system will be designed in a way that is easy to navigate and use, regardless of the skill of the user. There will be a dedicated help system present in the system, that will allow the user to get help, whenever they need it, without having the leave the system and check some external documentation. The graphical user interface of the system helps to promote the ease of use, as the user is not expected to memorise a collection of commands, and instead need only press a few buttons to construct the system they wish to construct.

Maintainability
One of the eventual goals of the project is that the inference engine, that will initially be a bespoke JavaScript implementation, or using the FuzzyToolkitUoN implementation, can be swapped out, and any compatible engine be used. This means that this system would evolve into a web-front end, for any Fuzzy Logic back end; greatly expanding it's usefulness. As a result of this, and a result of good software engineering practices in general, the code will be kept as readable and modular as possible. The large quantity of JavaScript code that is required will be split into separate files, so that the maintaining programmer can easily identify issues. Each function will be commented in a JavaScript equivalent of JavaDoc, so that automated documentation can be produced, if necessary. This will hopefully ensure that new back-ends can be easily incorporated into the system, and any additions necessary will be quick and easy for the maintaining developers.

Quality
As this system is to be used externally, and will be a representation of both myself, and the University of Nottingham, there are several quality issues that must be addressed. The system must be built so that it is robust, and works as the user expects, but it must also contain as few bugs as possible. Any bugs that are identified should be reportable to the maintainer, and be fixed as soon as possible.

The quality of the code must also be considered. In this regard, the code will be written in as modular a way as possible, and useful comments will be provided to highlight the purpose of each function, and to illustrate any particularly complex code.

Resource Requirements and Constraints
As this system is aimed at any users of any skill level, no assumptions can be made on the level of hardware that the users will possess. For this reason, the system will be designed to use as few resources as possible. Fortunately, due to being a web-based system, the load of the system would be fairly minimal anyway, as it is mostly loading only JavaScript and HTML. The only true computation takes place when the system is evaluated, but this could take place on the server side, and thus would not be a concern for the user.

The ability to load and save files potentially causes a problem, but only if the user has very little hard drive space, and attempts to save an extremely large file from the system. Unfortunately, there is nothing that can be done about this, although even very large systems will have a relatively small file size.

Cross Platform Compatibility
Due to the project being a web-based system, it is difficult for it not to be cross platform compatible. However, there are still a few issues that may have to be dealt with, especially when looking at different browsers that the user may be using. For this reason, research into how the system performs on different browsers will be important, so that it can be assured that any user using any browser will have full access to the system, as it was meant to be.

Security

The issue of security is not one that need be discussed in great detail, as there are no security issues with the system to be produced. In a web-based system, there is potential for many security issues to be present, such as the storing of cookies without the users permission, or the tracking of what a user may be doing. However, the system being produced here is simply a tool for the users to create fuzzy systems in, and there is no useful information that tracking their movements would return. There is also no need to save cookies, as the system will have full file loading and saving functionality.

Reliability and Robustness

As the system is to be used by experts and novices alike, the system needs to be as robust and as reliable as possible. This is to ensure that the expert users can work uninterrupted, and that the novice users do not get confused, if something unexpected happens. This is why there will be rigorous testing of the system, by both novices, and by experts (both in regards to fuzzy logic, and in regards to the use of a PC), so that the system can be thoroughly bug-checked, and feedback can be gained on the usability of the system.

Documentation

The system will be fully documented through comments in the JavaScript code. This will be done using a style similar to JavaDoc, so that documentation pages can be generated easily. The system itself will have the extensive help system present on every page, so that any user confused with the system can be guided in the right direction. For these reasons, there will be no external documentation for the project produced (which actually helps the "ease of use" objective, as the user does not need to look through a large external document to find help).

Disaster Recovery

All of the files for the project are stored in a GitHub repository (which is, for the time being, private), so any issues involving loss of code are not a concern. There is potential for the system server to go down, and in this case, due care and attention will be dedicated to resolving this issue. The only other issue that requires disaster recovery is if the user enters their system, and closes the browser without saving. This is, however, out of my control, although a pop-up warning the user may be implemented.

5 System Designs

In this section, all of the design aspects of this system have been detailed and justified.

5.1 UI Design

As the purpose of this project was to aid in the accessibility and usability of fuzzy logic, to users of all skill levels, the design of the user interface was extremely important. The difference between a good piece of software, and a great piece of software, can easily be the interface they provide. In fact, the entire reason that FuzzyToolkitUoN is an inadequate piece of software for specifying fuzzy sets, is it's interface, and this is the exact thing this project aims to overcome. There were two main iterations to the user interface design, the first was a simple attempt to include all information on the page, in an easily viewable format. The second iteration was a refinement of the first stage, in which design principles were applied, and feedback was gathered from potential end users.

It is worth noting that having all the different tasks of constructing a fuzzy system (specifying inputs, specifying outputs, specifying rules, evaluating the system, and file input and output) are all distinct tasks, and there is no reason for them to be together at any point. This is why, regardless of design, these tasks are all separated and are on different pages, or tabs, of the website. Further to this, the designs presented below cover only the input variable creation page and the rule creation page, as the input creation page is an exact replica of the output variable creation page, and the remaining pages or tabs of the website do not require much level of design, as they are relatively small.

5.1.1 First Iteration

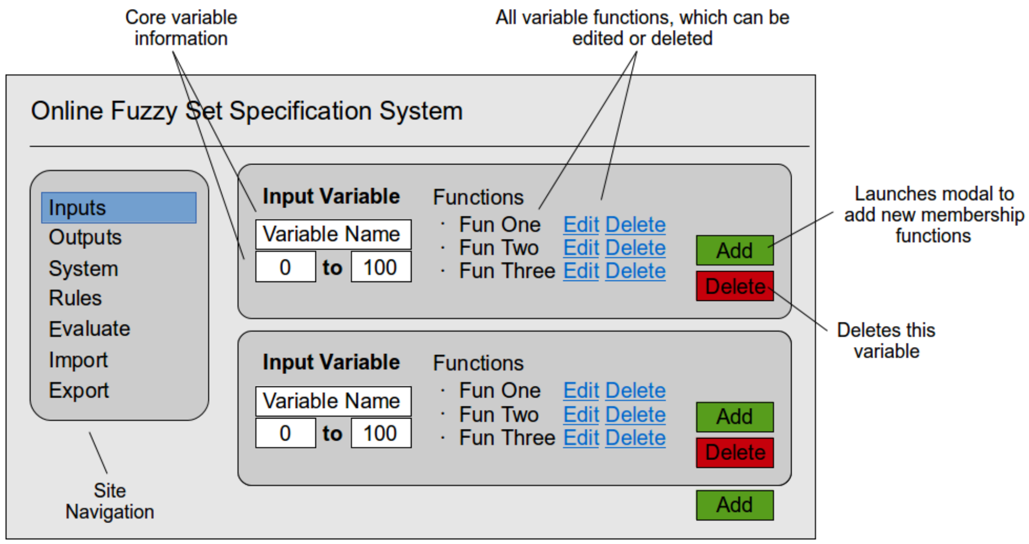


Figure 2: The first iteration of the design of the input creation page

The first iteration of the input creation page listed the navigation to the left hand side of the page, as this is a standard positioning for navigation [15], and the users would, most likely, be accustomed to this.

The separation of different tasks to different sections of the website is very important for the construction of a fuzzy system, as there are many individual elements, and a lack of separation of these would cause a great deal of confusion for the user, as there would be a huge number of elements on the screen at one time.

The design displays the core information of the variable on the far left (including name and range), as this is where the user's eye would fall first. The next set of information (the functions contained within the variable), are displayed further to the right, as the user would look to here after reading the initial information. The users have the option to edit or delete any function that they have created, granting them complete freedom over the system and everything they do within it. They also have the option to add a new membership function (which is a green button, to represent creation, which brings up a modal window to be used to construct the membership function), or delete the current variable (which is a red button, to represent destruction).

This design uses long horizontal representations of the variables, so a large number of them can be on the screen at the same time, as they take up little vertical space. This means the users can view many of their variables at the same time, and make edits as necessary. Colour is used sparingly throughout the page, so that it can be used effectively for highlighting important information or elements, so they are easily visible to the user.

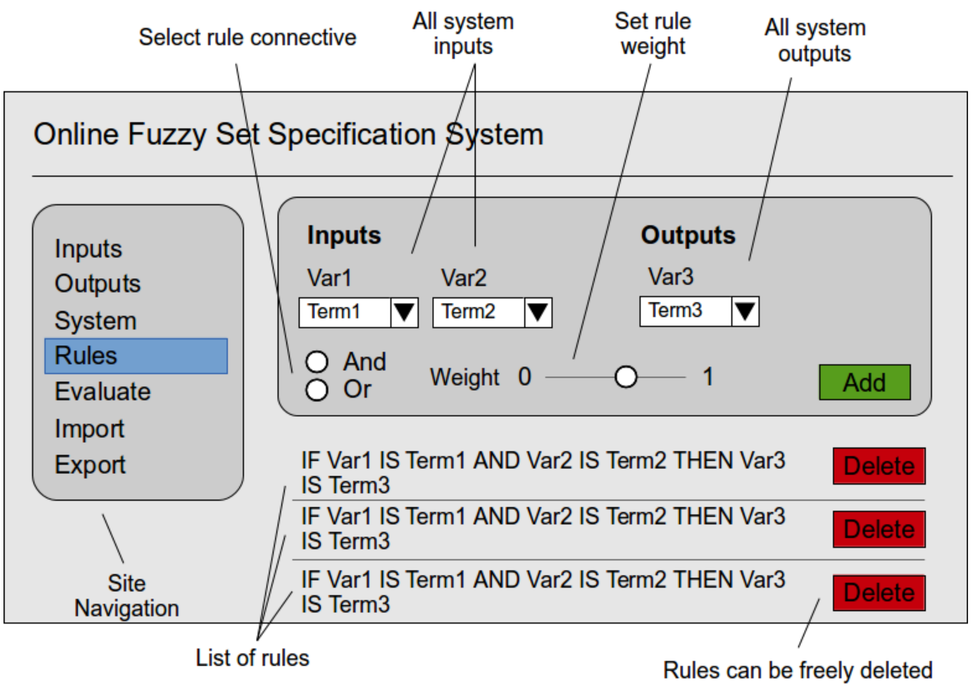


Figure 3: The first iteration of the design of the rule creation page

The rule creation page also follows a horizontal theme, which is especially effective for the rules, as there could potentially be a long list of them. The creation of the rules, and their displaying, are two clearly distinct segments to the page, which means there is no confusion between the two, and the process remains a simple one. In keeping with the standards of the previous page, the "Add" button is coloured green, to represent creation, and the "Delete" buttons present by each rule is coloured red, to represent destruction. These clear colour separations are useful to the user, as they do not even need to read a button to gain an understanding of what it will do [5], and they can be more careful to not make mistakes.

As can be seen, the navigation and header of the rule creation page are the same as those on the input creation page, which promotes a consistent style amongst the pages, and helps to remind the user they are still within the same system, even if they are completing a different task.

5.1.2 Second Iteration

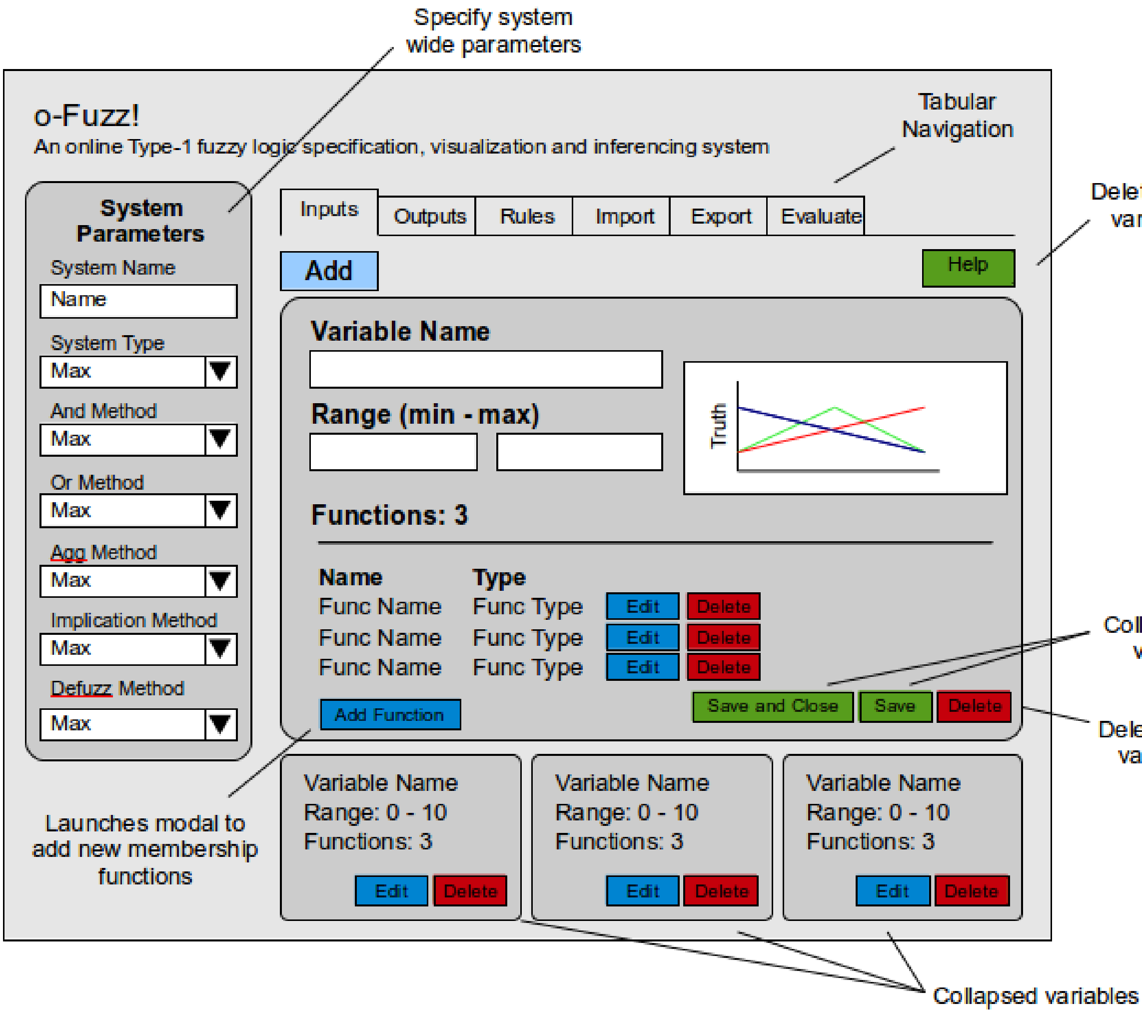


Figure 4: The second iteration of the design of the input creation page

The second iteration of the input creation page design includes many improvements over the first iteration. Specifically, this second iteration dealt with the issues the first iteration presented, and had well known and well documented usability heuristics and design principles applied to it, to ensure optimality.

One of the largest changes to the design was the re-working of the navigation to be entirely tabular, and to replace the old navigation area with system wide parameters segment. This tabbing helps to split the different tasks of the system in an intuitive manner, as most users are familiar with the concept of tabs, due to their popularity within internet browsers [10]. The system wide parameters are now displayed in place of the old navigation, which means they can be changed, regardless of the tab the user is on. This is especially convenient when the user comes to evaluate the system, as it makes tuning much quicker, and all permutations of parameters can be evaluated with ease.

The overall structure and displaying of the variables has also been drastically changed in this second iteration of design. The reason for this was that, whilst the old design promoted a large number of variables on the screen at any one time, this would severely increase the cognitive load on the user, and they could quickly become confused.

The new design works by having expandable and collapsible variables. When in the collapsed view, the variable takes up a much smaller space, displaying only key information, and in the expanded view, all of the information of the variable is present (like in the previous design). The ideal number of elements on a page is 7±2 [18], which can easily be exceeded with these new collapsed variables. However, the reason this design works, is because when the variable is collapsed, the user feels like it is completed, and they no longer need to concern themselves with it. So whilst this new design may look more cluttered, it significantly helps to reduce the cognitive load of the user, as they can create a variable, and then essentially forget about it.

To fit with the consistency of the other pages, the buttons for the editing and deletion of membership functions have also been changed. The edit button is now blue, which is the system wide colour for editing, and the delete buttons are now red, the system wide colour for deletion. This keeps the website consistent, and makes the functionality of these buttons easily identifiable.

Some other smaller changes include the moving of the "Add" button from the bottom right of the page, to the top left. so that it would not constantly move down the page as variables were added, as a moving button can be frustrating and confusing for some users. Another change is the addition of a short name for the software, with a longer tag-line underneath this, at the top of the page. This shortened title allows for users to refer to system with ease, allowing them to search for help for the system if necessary, and simply discuss the system with other potential users. The longer tag-line allows for a brief description of the system, so the user knows whether it will be able to accomplish the tasks they have set out to achieve. A graphical representation of each variable is now also displayed in the expanded view, allowing the user to quickly interpret how their system is progressing, and whether it looks as they expected. This is a very powerful tool, as users will be able to spot errors much sooner than if this were not present.

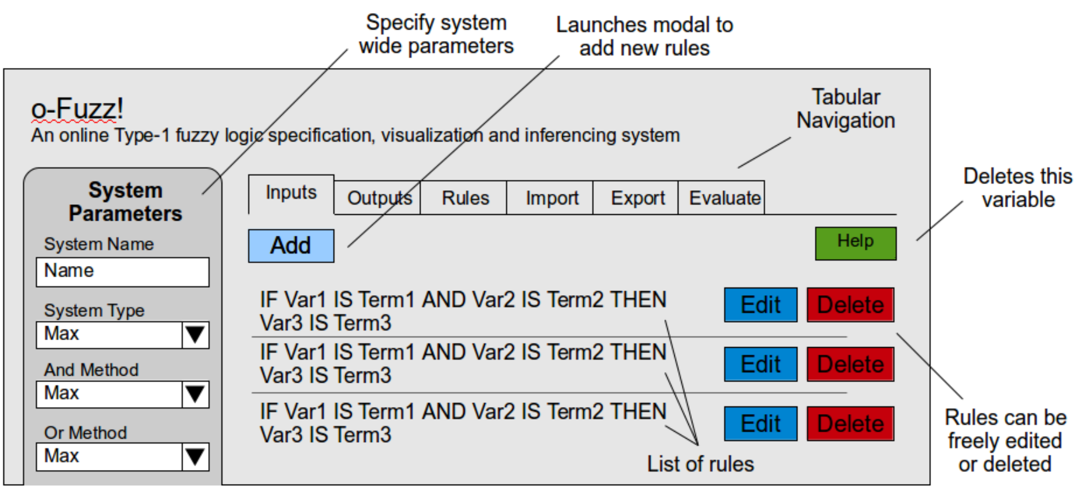


Figure 5: The second iteration of the design of the rule creation page

The rules creation page also received heavy changes in the second iteration. The first of these was the inclusion of certain features that were not present in the first design, but were necessary for the functioning of the program. For instance, the ability to edit rules once they had been created, the ability to create negated terms (for instance, "If food is *not* good") and the addition of the help button to the page.

However, the biggest change is the removal of the ability to create rules on this initial page. Instead, as with the creation of membership functions, this has been moved into a separate window that is launched when the user clicked the “Add” button. This reduces the clutter on the page, and means the rule creation functionality is only accessed when the user requires it (helping to reduce cognitive load on the user).

In addition, an extra feature was included in the form of a rule table that could be displayed if the user was using a system with two inputs, and one output (a relatively common set up for a fuzzy system). If this were the case, the table would be displayed, mapping the inputs to the outputs, in an intuitive graphical manner, an example of which can be seen in figure 6.

		Body Temperature		
		Low	Medium	High
Heart Rate	Low	Critical	Rising	Critical
	Medium	Critical	No Threat	Critical
	High	Critical	Rising	Critical

Figure 6: An example of a rule table that would be displayed if the user had two inputs (heart rate and body temperature) and a single output (urgency)

5.1.3 Heuristic Evaluation of Second Iteration

The new design also went through a heuristic evaluation [21], using the Usability Heuristics laid out by Jakob Nielson [20], and the Golden Rules for Design, laid out by Ben Shneiderman [25]. Both of whom are extremely well known and very influential in the field of Human Computer Interaction, and their heuristics define some of the most basic, but most important properties a user interface should possess.

1. Visibility of system status/Offering of informative feedback
   It is important that the users of a system are always updated as to what is happening within the system. This will be implemented through the use of JavaScript alert messages to alert the user to any errors they have made, or any important changes they have taken place.

2. User control and freedom/Support internal locus of control
   The user should be in full control of their experience of the system at all times, and thus a fluid and flexible navigation system is important. This is the purpose of the tabular layout of the website, as the user is free to travel to whichever pages they wish, in whichever order, which is specifically useful in the modern age where the culture and location of users has a large impact on how they navigate websites [12]. More details on the navigation of the system can be found in section 5.2.

3. Consistency and standards/Strive for consistency
   A good interface should also be consistent in it’s design, and follow platform standards. As you can see from the design of the input creator and the rule creator pages, the tabular layout is consistent throughout the website, and colours such as blue, green, and red have clearly defined purposes, which stay consistent throughout the as well.

4. Error prevention/ Help users recognize, diagnose, and recover from errors
   Unfortunately, with a system that is entirely dictated by user input, it is very difficult to prevent the user from making errors. However, measures will be put in place to ensure these errors do not affect the system. For instance, there is no way to enforce the user of a system to enter a number, but an informative error message will be displayed, telling the user this is not valid, and what should be done to rectify the issue (and, of course, not accepting the value into the system). Further to this, as there is no such thing as an "average" user [23], it is important that the website is designed in a way that it is usable by as many different types of users as possible (and of different skill levels), and thus error detection, prevention, and recover, are enormously important.

5. Recognition rather than recall/Reduce short-term memory load
   As mentioned multiple times already, the system is designed to reduce cognitive load on the user as much as possible. This is done by reducing the number of elements on the page at any one time, and by using modal windows for the creation of new membership functions and rules, so their creation and their display are distinct. This reduction in cognitive load is also extremely important in the case of designing a fuzzy logic system, as the task itself requires a significant level of concentration and thought, and thus reducing load in all other areas is a necessity [26].

6. Aesthetic and minimalist design
   To reduce any possible distractions for the user, the system is designed in a minimalistic fashion, using colour very sparingly, and sticking to neutral shades for background elements. The only colour used in the system is on the graphs drawn, and on the important buttons the user will be pressing. These buttons are essentially colour coded so the user is aware of their functionality, without even having to read them. Further to this, the use of modal windows to essentially hide functionality greatly helps to reduce clutter on the pages of the website, giving it a much cleaner look [6].

7. Help and documentation
   Due to the goal of being as easy as use as possible, the system is to be designed with a dedicated help system, built in. The advantage of this is that the user is able to access help without having to leave the application itself, meaning a minimised distraction time. Help is accessed via the large green help button present on every page, which is easily visible, and provides concise and helpful information for the user.

As the final stage of the design evaluation process, the "sins" of New Media Design [8] were also evaluated, to ensure an optimal design had been produced. The second iteration of design has taken these sins into account and has avoided those that were appropriate. Specifically, the website features no bulky borders, a generous use of margins, left alignment of elements (as opposed to centring), and an avoidance of a busy background, opting instead for a very simple white background. All of the aforementioned design choices could have easily caused distractions for the user, and damaged their enjoyment of the system.

5.2 Navigation/Control Flow Design

As of the second iteration of the design, the website has six main sections, enumerated below;

1. Inputs; The page used to construct input variables of the system
2. Outputs; The page used to construct output variables of the system
3. Rules; The page used to construct rules of the system
4. Evaluate; The page used to evaluate the system
5. Import; The page used to import files into the system
6. Export; The page used to export file from the system

The generally *expected* path for a user of the system to take, would be to launch the system to the input creation screen (the landing page), on which they would spent some time constructing their input variables. They would then move to the output variables, construct those, and then move onto the rules, and construct those. At this point, the path splits between evaluating the system, exporting it for later, or a mixture of the two.

Due to the ability to import files, an entirely new path is also available throughout this process, in which the user begins by importing a file, and then either editing it, or going straight to evaluating it.

The expected path through the system (beginning at the inputs page, as this is the landing page of the website), is represented in figure 7.

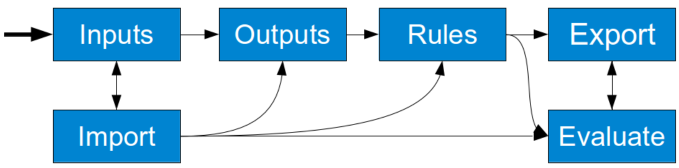


Figure 7: Expected path a user would take through the system

Of course, whilst the generally expected path through the system has been determined, this is not the only path through the system, and there is no guarantee that the user will follow this path. In fact, the user has the freedom to traverse the system in any path they please, as shown in figure 8. This is important for promoting freedom and a sense of control within the system. The user is free to decide how they wish to travel through the system, and this means they are free to traverse forward or backward through the system to make any alterations, or fix any mistakes they may have made.

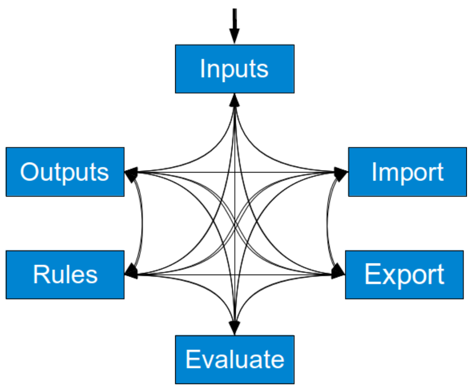


Figure 8: Potential paths that can be taken through the system

5.3 Internal Design

The user interface of a system is a huge part of the design process, and is extremely important, as this will be how the user will actually be interacting with it. However, this is not the only part of the system that requires designing; as the internal workings of the system also require a lot of thought.

For this project, a web-interface was to be used to interact with a back end that could deal the processing of a fuzzy logic system. By this point in the project's development, the software's implementation decisions had been made (which are detailed in section 6.1), and the decision of the back end to be used was the FuzzyToolkitUoN R Package, using the R-Shiny service, to interact between the front end and back end. The diagram in figure 9 shows how the interaction between these two systems is handled.

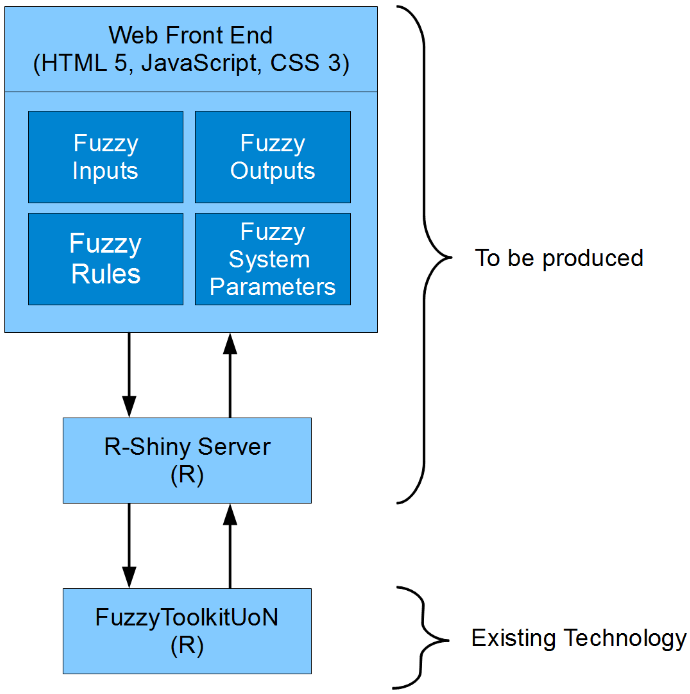


Figure 9: Interaction between the back end and front end of the system

As you can see, the web front end will be constructed using HTML 5, JavaScript, and CSS 3, and will allow for the creation and specification of fuzzy inputs, outputs, rules, and system parameters. Once a system has been specified, and the user chooses to evaluate the system, the information of the system is passed to the R-Shiny server. This then takes all the form elements from the web front end, extracts their values, and constructs a FIS file, within R. This can then have functions from the FuzzyToolkitUoN package applied to it, and the results of these functions can then be passed back to the web front end, for display to the user.

6 Software Implementation

In this section, the actual implementation of the software has been detailed, including: what tools were used in the implementation, how the software was implemented, and any issues that were encountered during the implementation process.

6.1 Key Implementation Decisions

As detailed in section 3.3, there were many different potential platforms and tools can could have been used in the implementation of the project. In this section, the final decision for which tool to use for each distinct section of the project has been detailed, along with their justifications.

6.1.1 System Back End - R-Shiny

R-Shiny is a new, up and coming R Package, that allows for the creation of web applications using the R programming language. The main reason that R-Shiny was selected as the back end was that it would be able to use the FuzzyToolkitUoN R package, and access all the fuzzy logic inferencing tools that were contained within it. R-Shiny was chosen over R-Node due to, at the time of implementation, R-Node not having a working website, and thus no ability to actually download it; which is not the sign of a well-maintained tool. Node.js was not chosen as this would require a brand new inference engine to be written in JavaScript, which would have taken a considerable length of time, and may not have been completed during the life time of the project.

Unfortunately, as detailed in section 6.5, this may have not been the greatest choice, as some issues were discovered with R-Shiny. However, the ability to use the FuzzyToolkitUoN inference system was a vital part of the project, and as R-Shiny continues to be developed, hopefully these issues will be resolved.

6.1.2 Front End Programming Language - JavaScript

JavaScript is a client-sided programming language used to dynamically alter the content of an HTML document. This the perfect tool to be used, as it has the ability to directly and easily manipulate the content of a web page, and thus dynamic interfaces can be constructed with ease. It also fulfils the goal of the system being easy to access, and use, as the user is not required to install any extra software, to begin using the system (like they would have to have done, if the other alternative, Adobe Flash, was used instead). Using JavaScript also means that other JavaScript libraries could be used within the system, and thus greater functionality provided. One other JavaScript library heavily used within the system is JQuery, which is a fast, small, and feature rich JavaScript library, that greatly simplifies document traversal and manipulation, and event handling.

6.1.3 Front End Design Framework - Twitter's Bootstrap

Despite complaints that Bootstrap is overused, it is still by far the best front end design framework available. This is due to ease of use, huge number of UI elements to choose from, and extensive documentation. Many of Bootstrap's feature were included in the implementation of the project, to heighten the user experience. This included the use of modal windows to reduce on-screen clutter, coloured buttons to dictate functionality, and tabbed panes, to help split the system into smaller segments. Due to the adoption of Bootstrap, the system is also fully responsive. This means that the system automatically scales to work on any screen resolution, from large, to small. This makes the system much more accessible to all users, and no strange UI bugs will occur if the user is using a different screen resolution to the one the system was designed on.

The only negative of using Bootstrap is that the modal window class does not function correctly when viewed on a mobile device. However, this is not a huge issue, as the system isn't truly designed to work on mobile devices, as this is an impractical interface for designing a fuzzy system.

6.1.4 Graphing Tools - Google Charts/Directly from R

Of the three potential candidates for graphic tools, Flot, was ruled out, due to it being difficult to use, and having a limited number of graph types available. The system instead utilised Google Charts for the majority of chart drawing. This is because Google Charts provide aesthetically pleasing, interactive graphs, in a large number of formats, and has extensive documentation. The graphs drawn to represent the fuzzy sets in the membership function creator, and those in the variable creator are all drawn using Google Charts.

Unfortunately, Google Charts does not support the drawing of three dimensional graphs, whereas FuzzyToolkitUoN does. When a fuzzy system of two inputs, and one output is defined (a common set up for type-1 systems), a three dimensional surface plot of the mapping between the two is an extremely effective way of visualising the system. For this reason, on the evaluation page, if the user has specified a system of two inputs and a single output, a surface plot will be displayed, directly from R.

6.2 Implementation Methodology

To help manage the implementation of such a large piece of software, the adoption of some methodology was necessary. It was decided that the best methodology would be an agile one, with heavy use of Kanban, using the tips laid out by Henrik Kniberg [11]. In order to accomplish this, at the beginning of the implementation stage, after the requirements specification had been detailed, the entire project was split into user stories. Each of these stories detailed a specific action that a user of the system would be able to accomplish, along with how long it should take to implement, how important it was, and a way of testing its completion. These stories were then organised onto a digital Kanban Board, using a service called Trello[16].

Each week, a set of tasks would be selected to be worked on for that week. The amount of tasks selected would be dependent on how much was completed, on average, in the weeks before, so that reasonable estimates could be made (obviously excluding the first few weeks). This ensured a decent portion of work was being completed per week, and that progress was constant. During the week, tasks would be selected from the available pool, prioritising those that were prerequisites of others, or had a high importance, and would then be worked on until completion. After the completion of a task, a new task would be selected, and work would begin on this. This was an extremely effective method of managing the implementation, as any small tasks that were necessary could be added to the board, and there was an assurance they would eventually be completed, and nothing would be overlooked. It has also been shown that it is much easier to reach goal,s if they have been written down [29], which a Kanban Board was the perfect tool for.

[16] `http://www.trello.com`

6.3 Detailed Description of the User Interface

In this section, each individual screen of the system has been displayed, along with a detailed explanation of why it is effective, and why it has been implemented as it has.

The first screen presented to the user upon launching the system (shown in figure 10), is a greyed out version of the input creation page, with a modal window displayed. This modal window gives a brief explanation of the purpose of the system, how to get started, the intended audience, and contact details for the maintainer. The purpose of this is so that any user that has navigated to the website can very quickly gain an understanding of what the system is for, how to use it, and as a result, whether or not they would want to, or need to.

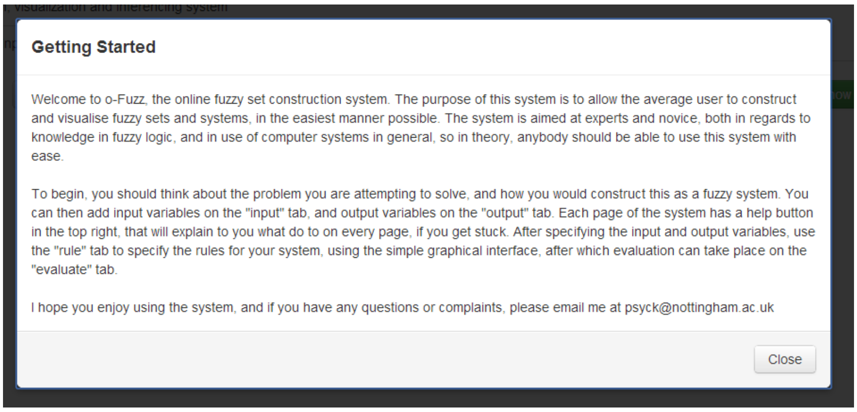


Figure 10: Screen shot of initial landing page modal window

After the user has finished reading the initial modal and closed it, they will be presented with the input variable creation screen. Inputs are generally the first thing a user will create in their system (as this is logically first), so this is a perfect landing page. The system wide parameters are also displayed on the left hand side of this page, which is where the eye of the user will be drawn to first, and thus noticed. A screen shot of this page, with two collapsed input variables, can be seen in figure 11. As you can see, all of the buttons on the page stand out, and can be easily located. This means the user's attention can cycle between the buttons easily, greatly increasing the speed at which they can accomplish their tasks. The help buttons, in green, are consistently displayed in the top right hand corner of the section they are providing help for, making them extremely easy to find, if help is required.

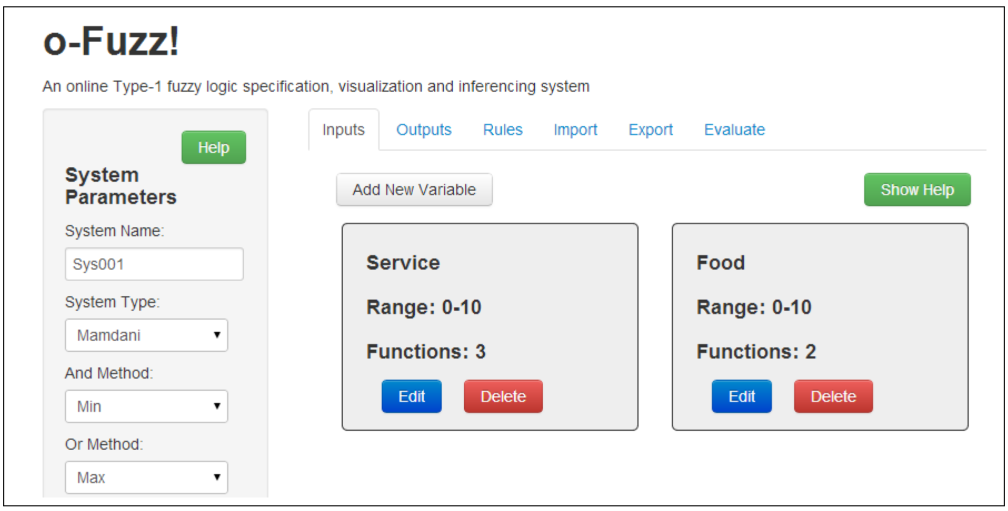


Figure 11: Screen shot of the input variable creation screen, with collapsed variables

Upon clicking the "Edit" button of a collapsed variable, the expanded view of it would be displayed, as shown in figure 12. In this expanded view, all the information of the variable would be displayed, such as the name, range, and any membership functions. These properties were now also editable, which they were not in the collapsed view. This means that to make a change, the user has to actively click a button, which decreases the chance of this happening by mistake. In this expanded view, the same colour scheme is observed as with the rest of the web site: blue representing editing or modifying, green representing a positive action, and red representing a negative action, like deleting. A plot of the membership functions of the variable can also be seen in the expanded view, which allows the user to visualise this variable, and all of the membership functions it contains alongside one another.

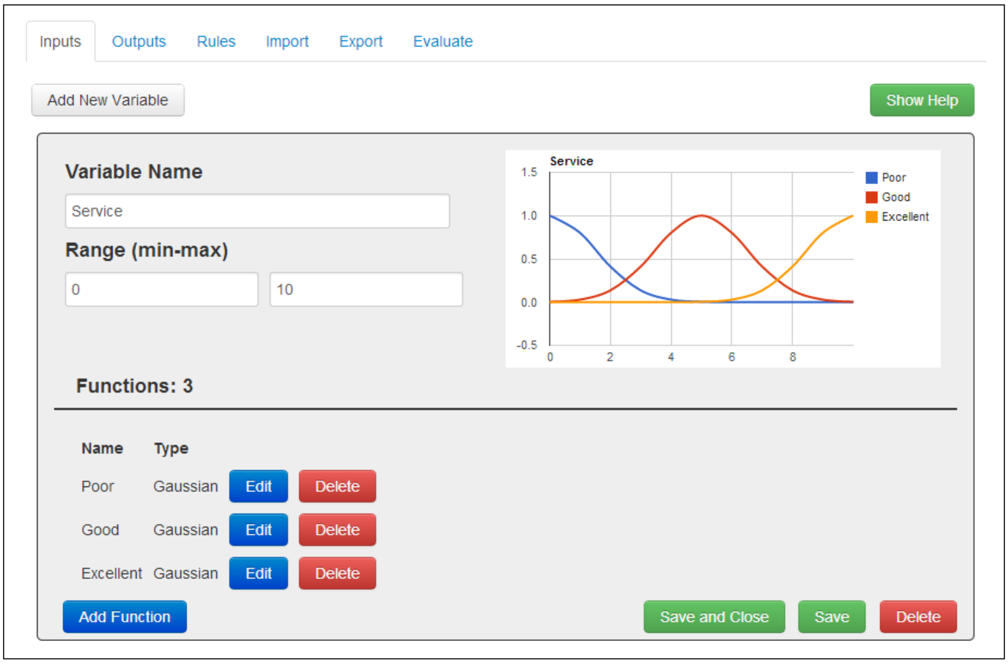


Figure 12: Screen shot of the input variable creation screen, with expanded variable

Upon clicking the "Add Function", or "Edit" buttons on the expanded view, the membership function creation window would be displayed (seen in figure 13).

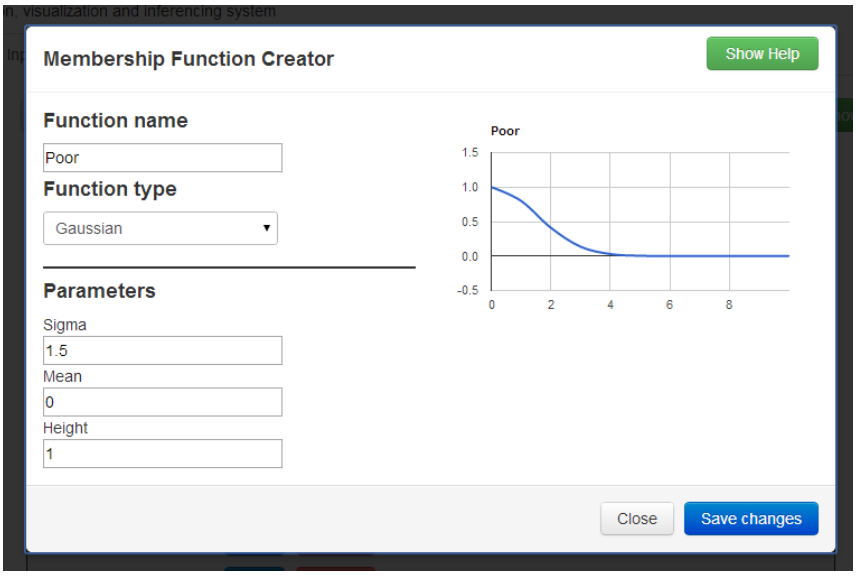


Figure 13: Screen shot of the membership function creation modal

The reasons for this being in a separate window are that it helps to split conceptually distinct tasks, reduces cognitive load on the user, and decrease clutter on the user interface. On this screen, the user is able to specify the parameters for a fuzzy set, and see how this set looks, once all parameters have been specified. This graphical representation is extremely useful for the user, as they can inspect their creation, and ensure it is as they expected. Note that the help button remains consistently placed in the top right hand corner.

After creating the input variables, and the output variables (whose interface is exactly that of the input variable, and thus has not been shown), the user would logically move onto the creation of rules. The interface for this, is displayed in figure 14. On this page, you can see that the consistent colour scheme has been kept, with the edit buttons being blue, the delete buttons being red, and the help button being green, and in the top right hand corner. On this screen, the rules that have been created will be displayed, in plain English, to the user, with the ability to edit and delete them. This is a much more intuitive way to display the rules than, for instance, FuzzyToolkkitUoN, which simply represents rules as a list of numbers. As this system is comprised of two inputs and one output, rule tables are also drawn for the user, to help them visualise all of their rules at once.

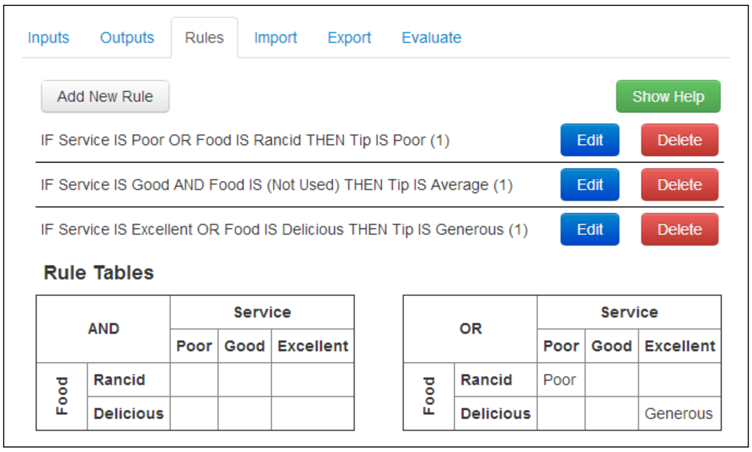


Figure 14: Screen shot of the rule management page

Like the creation of membership functions, the creation of rules takes place in a separate window (shown in figure 15). This is to help the user understand that these are distinct operations, reduce cognitive load, and reduce clutter on the interface. The window itself helps to make the task of creating rules as simple as possible, by only requiring the user to select a term for each variable, from a drop down box. The rule itself is also displayed at all times, so that the user knows exactly what they are creating.

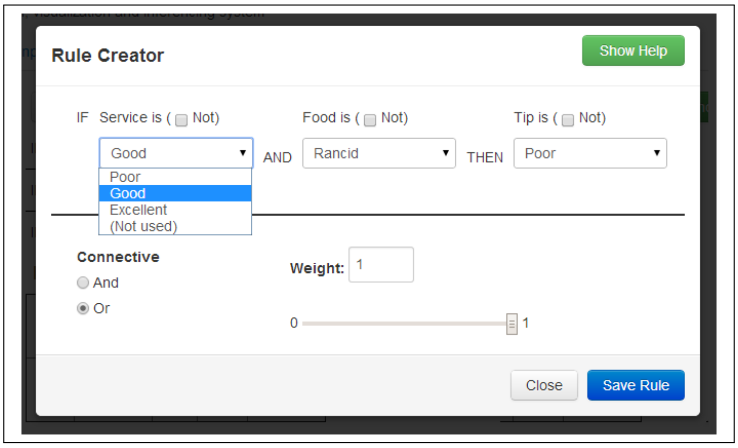


Figure 15: Screen shot of the rule creation modal

After the inputs, outputs, and rules have been specified, the user is finally ready to evaluate their system. This screen can be seen in figure 16. Each of the inputs, and their potential range, can be seen on the left, with the corresponding output on the right. In addition to this, because the system has two inputs and a single output, a three dimensional surface plot of the system (that is generated directly in R) is also displayed, so the user can easily visualise the mapping between all pairs of inputs, and their output.

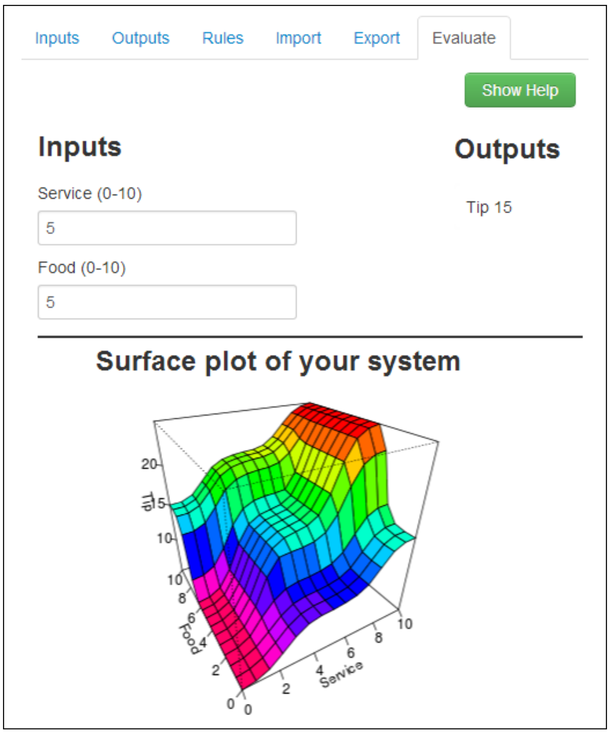


Figure 16: Screen shot of evaluation page, with surface plot

After evaluating their system, the user may wish to import another file to evaluate, or even save their current file, both of which they are able to do (in fact, they can do this at any point, due to the flexible navigation system). The interface for importing files is displayed in figure 17. This is the simplest of all pages of the website, and has a single button present. This makes the task of importing files very simple, because there are no other buttons to press. Once pressed, a file dialogue is launched, and the user can select a MATLAB FIS file, a FuzzyToolkitUoN FIS file, or an o-Fuzz JSON file to be imported. This is then loaded into the system, and also display on this screen, so the user can be assured this is the file they wanted, and that it loaded correctly.

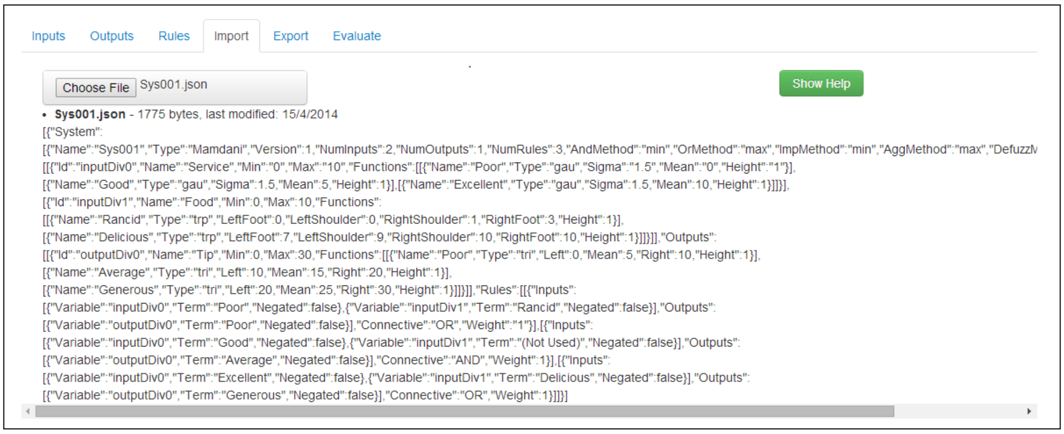


Figure 17: Screen shot of file import screen, with JSON file loaded

The final screen of the system is the file export page. Like the file import screen, this is an extremely simple page, making it very easy to use. This page has two buttons, one that generates a file to be exported (from three options), and another to actually download this file. Initially, the download button is disabled, until the user has selected a file type. After which, it will turn blue (indicating it is now usable), and the user will be able to download their file. This can be seen in figure 18.

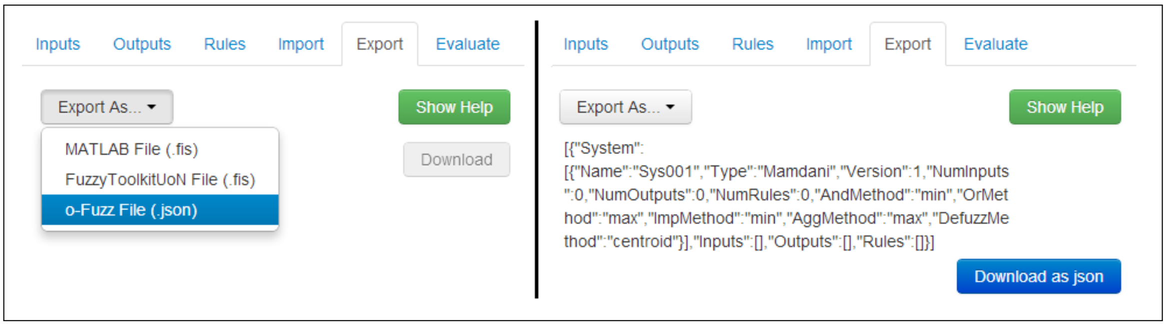


Figure 18: Screen shots of file export screen, before and after selecting a file type

As mentioned previously, help is available to the user at every stage of the system. This help is accessed by pressing the green "Help" button, present in the top right hand corner of every page. This consistent placement makes identification of the help button very easy, and means the user can quickly access help, regardless of where they are within the system (as they know where it can be found). When the help button is pressed, a pop over box appears, detailing help for the page the user is currently on (as shown in figure 19). This localised help system means the user does not have to trawl through some huge documentation to find help, and instead can simply click a button on the page they are on, to get help for that page. This makes finding help much quicker, and less intrusive.

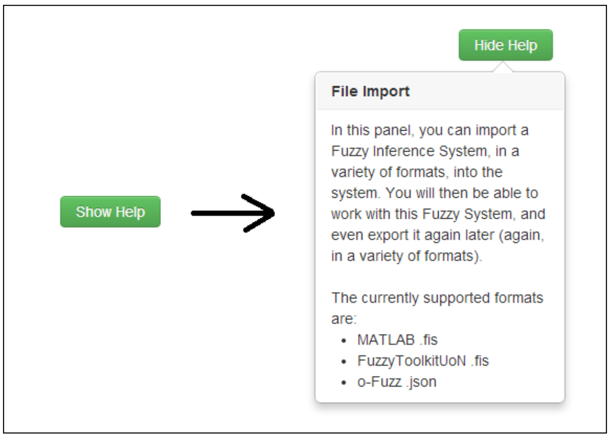


Figure 19: Screen shots of the help button on the import page, before and after being pressed

The last interface element to be mentioned, is that of errors. Errors are an important part of a user interface, as they give feedback to the user on mistakes they have made. It is important that errors are trapped, as to not cause unexpected actions from the system, which could cause confusion for the user, and reduce the overall quality of the system. When it came to actually writing the error messages, it was important to ensure they used positive language, and focused on what should be done to fix errors made, as opposed to condemning the user for making them [24]. Some examples of error messages within the system have been included in figure 20

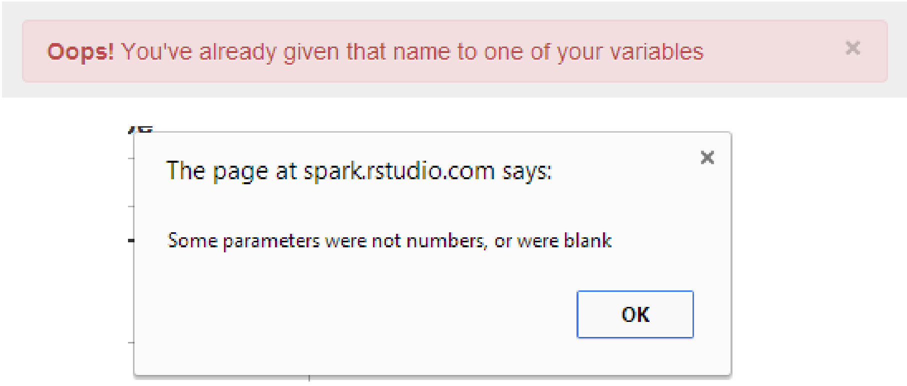


Figure 20: Example error messages of the system

In the first example (displayed when the user attempts to give two variables the same name), informal words, such as "Oops", are used to show that a mistake has been made, but this is not a huge issue, and the user should not worry about it (although the system will not allow them to progress until this has been rectified). It also clearly explains what the issue is, and thus how a resolution can be achieved. The second example (displayed when the user has not filled in all parameter boxes for specifying a membership function, or has not used numbers), is direct, and thus makes identifying issues quicker, and easier. It specifically references a segment of the page (the "Parameters" segment), which means the user can find the general area of the error, helping to reduce the time it takes to locate the error itself. The language used in the error message is extremely basic, using terminology that any user would understand, giving any user the ability to resolve the error.

6.4 Implementation of System Components

6.4.1 Web Front End

The front end of the project was written in HTML 5, with CSS 3 for styling (most of which was obtained from BootStrap). The dynamic interface and functionality of the system, were provided by a mixture of JavaScript, and JQuery. The purpose of the front end was to provide the user access to powerful back end inferencing tools, via an extremely simple user interface, to facilitate both novice and expert users. There were some 4,100 lines of JavaScript/JQuery across 10 individual files in the final copy of the software. Each file dealt with a very specific part of the system, as to help split up the huge code base into manageable chunks (for instance, one file would deal entirely with rules, whereas another would deal with file importing and exporting).

The front end is certainly a strong point with the system, as it has a clean, uncluttered interface, but also provided a wide range of functionality. Due to the modularity of code, adding new functionality is also extremely easy, and the consistent commenting through the files makes finding key functions (whether this be for debugging, or when extending the system) very easy, and understanding them equally so. Of course, with any interface, there is potential for improvement, to better suit users. For instance, the system could have had customisation possible, to help make the system appeal to different sets of users, or it could have had more support for keyboard short cuts. More detail on this can be found in section 8.2.

6.4.2 R-Shiny Back End

The back end of the system was constructed using R-Shiny, interacting with the FuzzyToolkitUoN R package, and thus was written entirely in R. This gave the system access to the powerful processing tools that R provides, and the visualisations that it can provide, if necessary. The purpose of the R-Shiny segment was to gather the data the user input into the front end, turn this into a format that could be used by the FuzzyToolkitUoN package, and then call the necessary functions for processing to take place.

The biggest issue with using R-Shiny is that it was a tool still in development, and there were some bugs present, and features missing. The biggest of these was the inability to properly handle dynamic user interfaces, such as the one this system required. This meant that, when it came to handling the user input, a hack had to be put in place for this to be dealt with correctly. This is covered in greater detail in section 6.5.

6.4.3 Front and Back End Interaction

The interaction between the front and back ends is extremely important for this system, as it meant that the user input could be dealt with on one side, and the processing of this input on another. The front end allowed for the user to fully specify their fuzzy system, including inputs, outputs, rules, and system parameters, and the back end would take these values, and evaluate the system, and return the appropriate results.

When the user navigated to the export tab, or the evaluate tab, the front end would query the arrays that were holding the system inputs, system outputs, and system rules, and combine these into a string, of the same format as a FuzzyToolkitUoN FIS file. This would then be stored in a text box, hidden from the view of the user. The back end of the system could then index into this hidden text box, and extract the value from it. After some parsing, this object would be converted into a FIS object that the FuzzyToolkitUoN package could handle, and then evaluation, or file exporting, could occur.

6.5 Problems Encountered

Unfortunately, as with most software systems, the implementation stage of this project was not without it's problems; the first of these being with the back end tool that was chosen, R-Shiny. As R-Shiny is a relatively new R Package, support for it is limited. This means that any issues that arise whilst attempting to use it, may not be documented, and a solution not found. The main issue that arose in this particular project, was the difficulty of creating a dynamic user interface, that may interact with R-Shiny. Normally, R-Shiny expects a static user interface, which allow for the server to index specific UI elements, and return their value. Unfortunately, that would not work for this project, as lots of dynamic UI elements were necessary, so that the user could construct the system exactly as they required. This meant it was impossible to list all UI elements in the server, as there could be any number. This problem was not discovered until a great deal of work had been carried out on the front end side, and at a point in the project life time at which restructuring was simply not possible.

A solution to this problem was eventually arrived at, although it was not ideal. Instead of attempting to return the value of every single UI element that could possibly be created in the system, a hidden call-back UI element was included. This call-back UI element was hidden from the user, but would be populated, using JavaScript to loop through all UI elements, with details of the system the user had created whenever server side interaction was required. This meant that the server was required to parse the input it received, but this was a trivial task.

Another issue that arose multiple times throughout the project, was the difficulty of working with such a large code base. By the end of the project, there was approximately 4,000 lines of code (split between R, and JavaScript), across 10 individual files. This could make debugging sometimes extremely difficult, as there was often a chain of function calls that would pass through multiple files, making pin-pointing exactly where errors occurred very troublesome.

Luckily, the Google Chrome JavaScript debugging functionality is very powerful, and would highlight the file that the error originated in. This meant that tracing the calls backward was much simpler, and errors be fixed much quicker. Google Chrome also allowed for the inspecting of variables being passed throughout the system, which meant values could be observed, and checked for validity at any point. To further alleviate the difficulties of working with such a large code base, every file had a list of functions present in it, at the top, so that offending functions could be found quickly. Further to this, every function had a JavaDoc style comment accompanying them, explaining what they did, what parameters they expected (including their type), and what they would return (if anything).

The final major issue that was encountered during the life time of the project, was the sub-standard implementation of FuzzyToolkitUoN, which was the back end for the system. Many bugs that are present in the system, are a direct result of bugs that are present in FuzzyToolkitUoN, and are thus not a controllable factor, and should not reflect poorly on the system itself. As a result of this, there was nothing that could be done to rectify these errors. The only thing that could be done, was to put error messages on the front end side of the system, that would explain issues of FuzzyToolkitUoN. Hopefully, as further work is completed on FuzzyToolkitUoN, some of these issues will be resolved, and the usability and robustness of the system will be increased considerably. In section 8.1, the potential for back end interchangeability is discussed, by using an API layer between the front end and the back end, which would have been a potential solution for this problem, if there had been enough time to do so.

7 Evaluation of the Project

Testing and evaluation were vital parts of the project, to ensure that the system contained all the functionality it should have, adhered to all the constraints place upon it and that it was actually usable. Three stages of testing were therefore conducted. The first of these was functionality testing, in which the functional requirements of the system would be evaluated, and the system would be tested to see whether or not it implemented this functionality. The next stage was non-functional testing, in which the non-functional requirements would be evaluated, to ensure the system abided by them. The final stage was user feedback testing, in which a group of participants would actually be using the system. In this stage, three systems would be evaluated: MATLAB, FuzzyToolkitUoN, and this project; in order to determine which provided a better user experience.

7.1 Functional Testing

In this section, each of the functional requirements laid out in section 4.1 have been evaluated in turn, to ensure the system meets them. Knowledge of the inner workings of the system is not actually necessary to understand these tests, as they simply check whether functionality is present, and are not concerned as to how the system actually implements it (this is known as black box testing [2]). A complete listing of all the tests conducted, and their results, can be found in appendix C.

7.2 Non-Functional Testing

In this section, each of the non-functional requirements laid out in section 4.2 have been enumerated, and the success to which they have been achieved has been detailed. The purpose of this is to evaluate how well the system has adhered to the constraints placed upon it.

Accessibility
This was one of the biggest goals for the system, as it was one of it's main reasons for conception. The problem with most fuzzy logic software systems currently is that they are difficult to use, or difficult to access. That is why the project proposed in this report has accessibility improvements as one of it's main goals. To accomplish this, the system is entirely web based. The advantage of this, is that the user can access the system from any computer, running any operating system. This cross compatibility greatly improves the outreach of the software, as no users will be unable to access it. Further to this, the system front end is constructed entirely in HTML, CSS, and JavaScript. This means that the user does not require the download of any additional software, or applications, to use this system (which would not be case if Adobe Flash, or Java had been used).

Usability and Operability
This is a goal of the system that is difficult to quantitatively measure. In order to actually test this goal, a set of participants of varying skill levels were asked to complete a list of tasks, using this system, and compare this to doing the same tasks in similar systems. The detailed results of these tests can be found in section 7.3. After the user feedback testing, it was found that the majority of participants preferred this new system, over the other two that were tested (MATLAB's fuzzy toolbox, and FuzzyToolkitUoN). The major reasons for this were the "waterfall" style to task completion (each task was laid out in a logical manner, and followed on smoothly from the last), and the clean, uncluttered user interface.

Maintainability
To aid with the eventual goal of the system back end being interchangeable with any back end that can process fuzzy logic, the system needed to be coded so that is was easily maintainable, and easily expandable. In order to achieve this, the code has been written in a modular fashion, split across multiple files. Each file deals with exactly one action (file manipulation, rule creation, evaluation, etc.), and thus adding new functionality should be relatively easy. Each function has been listed with the purpose of it, the parameters it takes (and their types), and the value it returns, if any. This means identifying the purpose of functions is extremely simple, and any maintainer will easily be able to interpret how the system works.

Quality
To ensure that this system reflects well both on myself, and the University of Nottingham, it was made to be of the highest quality possible. The system has been built with a vast quantity of error handling methods, so that the system should not crash under operation, and should function exactly as the user expects. Helpful, and positive, error messages accompany any error handling methods, so the user can rectify any mistakes they make (without feeling punished for making them).

Resource Requirements and Constraints
As the system is aimed at any level of user, no assumptions as to the level of hardware they may possess can be made. This means that the system must be built to be as light weight as possible, as to not overwhelm the user's device. Fortunately, the technologies used to build the front end (CSS, HTML and JavaScript) are relatively light weight, and the front end has minimal processing requirements. The main processing that takes place is on the server side, when inference occurs.

Cross Platform Compatibility
As mentioned previously, the system is built with entirely cross compatible languages, and only requires the user have an internet connection, and a web browser installed to be accessed and used. Aside from these, there are no compatibility issues the system presents, as it is entirely web based, and requires no additional resources to function. There is potential that the system would not work on extremely old operating systems, but all operating systems that are currently being supported by their developers should function correctly.

Security
In a web based system, there is potential for many security issues to be present. However, in this system, that is not the case, as it does not store any user data, and the only information stored on the user's computer are the fuzzy systems that they create. To give the user's peace of mind when downloading their files, the contents of it are displayed to them, before they are able to press the download button.

Reliability and Robustness
As the system will be used by both expert users, and novices, it is important that it is as robust as possible. For this, extensive error trapping has been implemented throughout the system, so that any incorrect values entered by the user are dealt with accordingly, and do not cause the system to crash. The reliability and robustness of the system are important so that the expert users are not inhibited whilst working, and novice users are not left confused (as they would not be able to tell the difference between a mistake they had made, and the system crashing). Testing of the system was carried out by both novice users, and expert users, to identify any bugs in the system, and any usability issues, details of which can be found in section 7.3.

Documentation
Large software documentation manuals are an extremely unintuitive way to find information on a software system. This fact stands even truer when looking at the novice audience, as a large documentation manual would simply be too daunting for them, and discourage them from using the system if they got stuck. To combat this, the system presented in this report does not have an external documentation. Instead, documentation is present in the system in the form of a help button being on each page. When clicked, a pop-up will be displayed, giving details on the workings of that specific page. This short explanation of how the page works is much more useful to the user, as it is concise, and easily locatable (always in the top right hand corner of the segment they are on, and always green). As far as documentation of the code, as has been mentioned, JavaDoc style comments will accompany every function written, that will describe what the function does, what parameters it takes (and their type), and what that function returns (if anything).

Disaster Recovery
During the lifetime of the project, the source code was stored in a private GitHub repository. This meant that the loss of code was not an issue, as it was backed up securely on the GitHub servers. As far as the system itself, due to the extensive error trapping that was present, the system was relatively robust, and would not crash due to user input. The only issue that remained, was if the user closed the system before saving their work. Unfortunately, this was not an issue that could be resolved without the implementation of cookies, which would have then raised a security concern. There was an attempt to include a pop-up, so that when the system was closed, the user was warned about the potential of losing their data, but this was not implemented successfully.

7.3 User Feedback Testing

An important part of evaluating the usability and accessibility of any software system is to have real world users attempt to actually use it [19]. In order to do this, several sessions were set up, and participants of various skills levels, both in terms of computers, and fuzzy logic, were invited along to complete a list of tasks using the software produced. So that comparative comments could be made, these same participants were also asked to complete the same list of tasks in two other similar software systems: FuzzyToolkitUoN, and MATLAB's fuzzy toolbox. After each task, the users would be asked for their feedback and opinions on the system, and how they felt it compared to the other systems.

There were a total of 23 participants in these studies, split into four main categories, based on their skill level in fuzzy logic, and using computers in general. These groups will be referred to throughout this section, and the table in figure 21 summarises the characteristics of the participants of each group.

Group	# of Members	Fuzzy Logic Skill	Computer Skill
1	7	Low	Low
2	5	Low	High
3	3	High	Low
4	8	High	High

Figure 21: Number of members in each group of the study, with their accompanying skill levels

Along with the user experience feedback that was recorded in these studies, the time taken for the user to complete the task list in each of the software systems, along with their favourite and least favourite systems was also recorded. The full results for this can be found in appendix B, and will be referred to throughout this section.

The tasks were designed to evaluate as many of the cross-compatible features of the systems as possible, so that they could be compared. The task itself was to implement the fuzzy tipper example, which is a popular first system for those learning fuzzy logic. The list of instructions given to the users can be found in appendix A.

7.3.1 Evaluation of FuzzyToolkitUoN

This test took place within the standard R environment, installed on a Windows 7 Machine. It was clear from the start that this would be the most difficult interface to use, and many of the novice computer users (and even a large portion of expert computer users) ran into trouble almost straight away. Many of the users could not conceptually separate the graphical user interface of the R environment, and the use of FuzzyToolkitUoN, and many of them would attempt to use the GUI when asked to perform certain tasks. The help provided by R was also extremely unhelpful in this endeavour. Some users attempted to access the help documentation, but this was about R itself, and offered no help for them using FuzzyToolkitUoN from within R. It was not until most users were shown the online documentation for FuzzyToolkitUoN on CRAN that they could begin. Most users could identify the correct function to use for each of the tasks, but some, such a "readFIS" and "writeFIS" were unusual to the user, and took longer to locate (as they expected "load" and "save"). Most users, regardless of skill level, would simply copy and paste the code from the "Examples" section of the documentation into R, and modify this to their needs. Whilst this did complete the task, the users admitted that they did not fully understand how these functions worked, or what they were doing. Towards the end of the task list, the documentation for the necessary functions began to diminish, and the users found the final few tasks much more difficult.

One of the greatest issues that users faced was the necessity to constantly reassign the FIS variable they had created, to itself. The purpose of this was to update their FIS variable with each function, but this concept was difficult to grasp for many users, and they claimed it was unintuitive. Throughout the majority of the tasks, users were confused, and this led many users to frustration. This fact has more impact knowing that it was the user interface that was being evaluated, and not the user, and yet they *still* felt like they were "failing".

After a few repetitions of the same actions (such as adding membership functions), most users finally began to understand better what they were doing, became more confident, and their progress sped up greatly. Unfortunately, when it then came to adding the rules to the system, much of their confidence was damaged again, as this was an extremely confusing process. Instead of using symbolic names, like they had been throughout the system so far, they were now expected to use numbers, which many users did not understand. The documentation for adding rules is also extremely long and confusing, and many users almost gave up.

Any mistakes made by the users (other than those users that had used R before) were, for them, irreparable, and external assistance was necessary. This essentially punishes the user for their lack of knowledge, and would often require them to start again, if assistance was not available. These mistakes were often not aided by the obtuse error messages provided by R. Throughout the usage of this software, users had many complaints, were often confused, and regardless of skill level, had difficulty using it.

7.3.2 Evaluation of MATLAB Fuzzy Toolbox

This test took place on an installation of MATLAB, with the Fuzzy Toolbox installed, on a Windows 7 machine. Immediately, users seemed more comfortable using this system than the command line interface of FuzzyToolkitUoN, this is because graphical interfaces are much more common in mass-market software, and the participants were much more accustomed to them. Unfortunately, when it came to actually using the interface, participants were still confused very frequently. The main reason for this was that there was such a large number of graphical elements on the screen at any one time, and it was very difficult to pick out which one was needed to complete the task at hand. Many UI elements that represented input boxes were disabled on the main screen (or at least appeared to be), so many users would believe certain tasks to be impossible. When users eventually realised that a mixture of the UI, and the menu were necessary to navigate the system, progress was made.

To solve the problem of splitting distinct tasks, MATLAB has a separate window for each part of the system (system parameters, inputs/outputs, rules, and evaluation). This makes sense from a design perspective, but has been implemented poorly in the case of MATLAB. This is because, with the opening of a new task, previous tasks would remain on the screen, in the background. Users very quickly were overwhelmed with the number of windows that were open, and assumed that all functionality should come directly from the first window they were presented with. They were confused with the concept of all these windows and would often ask whether or not they were allowed to close them, fearing that something bad would happen if they closed the wrong window.

Despite these flaws, the majority of users greatly preferred MATLAB, to FuzzyToolkitUoN, purely because it has a graphical user interface, regardless of how it was implemented. One important factor in this was the ability to rectify mistakes that had been made, something that was almost impossible in FuzzyToolkitUoN.

7.3.3 Evaluation of My Project

The final system that was evaluated was the project proposed in this report. The tests for this system took place using the Google Chrome web browser, on a Windows 7 machine. It is worth noting that all tests took place on the same machine, running the same software, so that this was not a factor. Upon launching this system, many users complemented it's appearance, and use of Bootstrap. They also appreciated how it was a lot less cluttered than MATLAB was, and there was more white space. This helped to not overwhelm the user, which is often the case when using a new piece of software for the first time. It was also extremely easy for the users to get started with the system, as it was launched on the exact page that they would start on. Many of the users commented on how they liked the tabbed navigation, as this helped to split up the tasks, and meant there were not multiple windows open, like in MATLAB. The "flow" of the system was also mentioned by many users. They felt that the way the system was designed made it extremely easy to go through the process of creating a fuzzy system, as all the actions they were required to take followed on from one another. For instance, the creation of a variable, and then within that, the creation of the membership functions. This nested structure was extremely useful, and helped the user to understand which parts of the system belonged where. The ordering of the tabs was also intuitive, as the user would move left to right through the tabs, to reach their final goal of evaluation the system. One negative of the tabbed navigation, however, was when the user was faced with a task they were unsure how to complete. They would sometimes attempt to just click on every single tab, and hope that an answer would be presented to them, instead of actually looking for one.

The error messages throughout the system were commended, as they were much more user friendly than those in both MATLAB, and FuzzyToolkitUoN. Users liked how they explained the error in such a way that a solution could easily be inferred. Another positive of the system, that was a result of being web based, was that many users were already away of certain short cuts that are present throughout the web. The most prominently observed of these, was the use of the tab key, to jump to the next input box. Many users, of differing skill levels, used this tab key method, which greatly speeded up the process of completing the tasks (many of which did not even comment on it, implying it is functionality they fully expected of a web system, and are familiar with using it). Some expert users, however, felt that there were not enough short cuts available to them. One that was specifically mentioned by some users was the ability to press the return key to submit the membership functions they had created. They felt as though, after using the keyboard to enter the parameters, and the tab key to navigate between input boxes, that having to use the mouse to then click the "Save Changes" button, was cumbersome.

7.3.4 Summary of Evaluations

The main two factors that were observed whilst the participants were completing the tasks were the speed at which they could do so, and the ease. Generally a faster completion time meant either a high level of understanding, or an easier piece of software to use. The graph in figure 22 shows a bar chart representing the average time taken for each group to complete the task list, in each software system, as well as the average for all participants. The data collected clearly indicates that using FuzzyToolkitUoN to complete the tasks was the most difficult (taking on average 35 minutes), which, after speaking to the participants, was a result of it's poor user interface, steep learning curve, and reliance on checking a huge documentation manual. Both of the graphical user interface systems faired much better, with MATLAB being the second fastest to use (with an average time of 18 minutes), and the software system proposed in this report taking on average 10 minutes (an improvement of 44% on MATLAB, and 71% on FuzzyToolkitUoN).

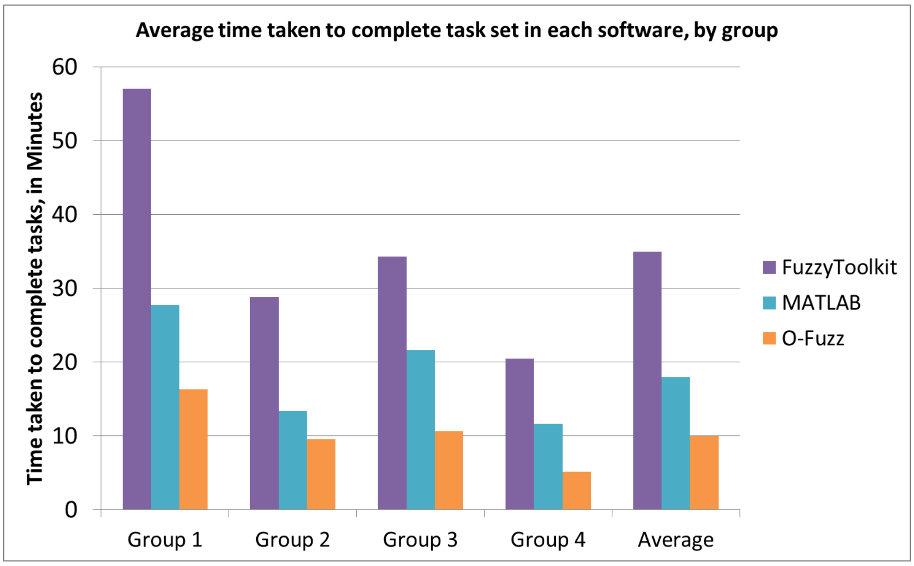


Figure 22: Bar chart of average time taken to complete the task set in the different software systems, by group

Whilst the time taken to complete the tasks was a strong indicator of the success of the software system, it was also important to ask the participants which system they *enjoyed* using the most. The results for this were conclusive, with 95.7% of participants (across all categories) claiming FuzzyToolkitUoN was the piece of software they enjoyed using the least. The main reasons for this were the command line interface being difficult to use, visualisations of the system difficult to access, the necessity to constantly refer to the large documentation manual, and that it was conceptually confusing for many computer novices.

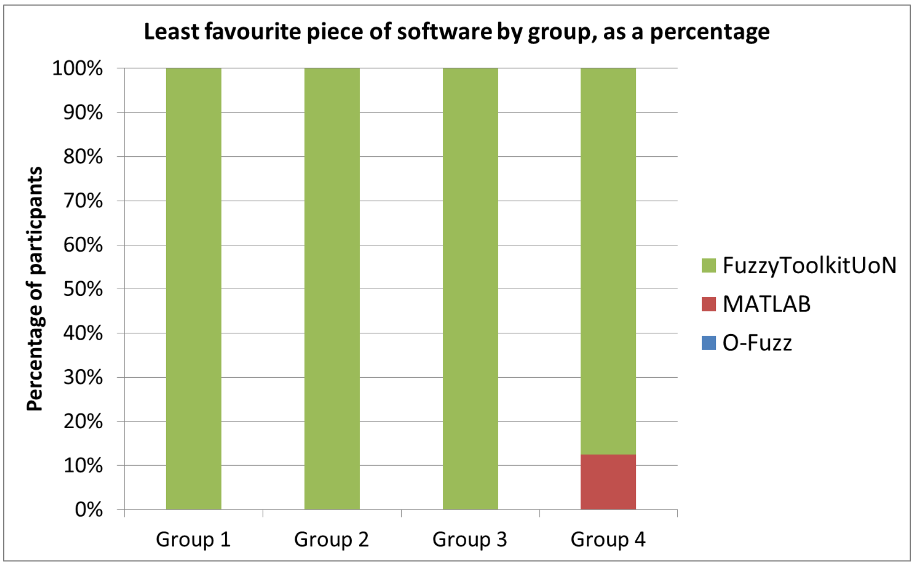


Figure 23: The percentage of users claiming each piece of software to by their least favourite

The piece of software that was most favoured by the test participants, was the system proposed in this report, o-Fuzz. Of the total participants, 65% claimed o-Fuzz to be their favourite software, with 30% saying it was MATLAB, and 4% saying FuzzyToolkitUoN (a single person). A decomposition of favourite software, across the separate groups, can be seen in figure 24

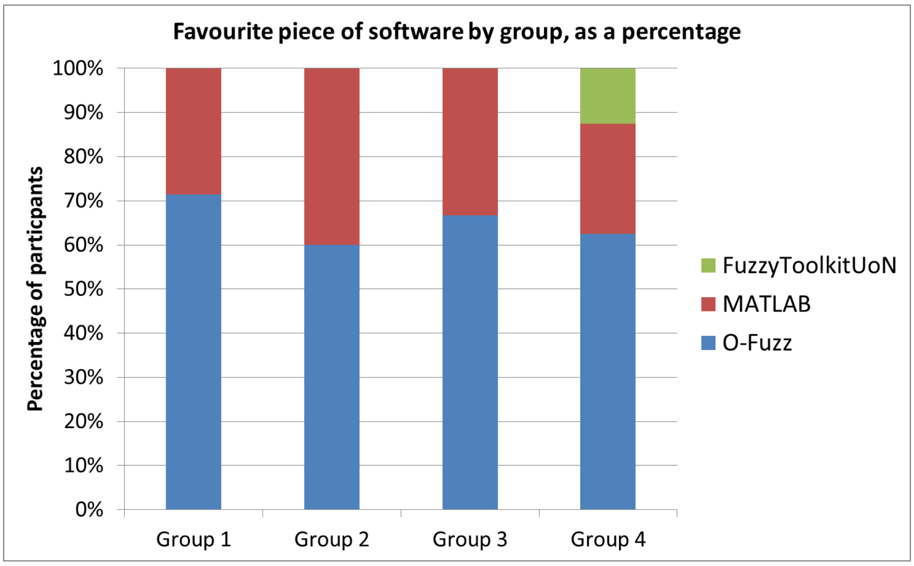


Figure 24: The percentage of users claiming each piece of software to by their most favourite

When questioned as to the preference of this new system, most users agreed that the interface was both extremely simple to navigate, and visually appealing. It was also mentioned that there was a natural flow to the software. By this, they meant that each task in the software lead onto the next very intuitively. For instance, the adding of a variable would be one distinct task, but the adding of membership functions would be a separate task nested inside this task. This meant that the conceptually distinct parts of the system were separate, but the system had a “waterfall” style flow to it. Many of the novice users also commended the system on it’s help system, specifically mentioning how this was much more intuitive than checking the huge documentation manual provided with FuzzyToolkitUoN.

It is worth noting that the single result for FuzzyToolkitUoN as favourite software came from a user that was extremely proficient in R, and specifically FuzzyToolkitUoN, and knew many commands from memory. This was also the same person that rated MATLAB as their least favourite piece of software, claiming that it was far too cluttered and aesthetically unappealing.

7.4 Successes and Limitations of the Project

Whilst getting user feedback is extremely important in evaluating a system, it is equally important for the creator of a software system to analyse the work they have completed. They are the only person that truly knows whether or not the software they have produced is what they were originally attempting to create. In this section, the successes and failures of the project have been listed, as seen by the project creator.

The biggest strength of the project is its adherence to its two main goals: ease of use, and ease of accessibility. The purpose of this project was to both be a practical implementation of a fuzzy logic software system and, more specifically, open up the field of fuzzy logic to novices of the area, and novices in using computers in general. In this regard, the project was a huge success. The feedback received from the user feedback evaluation strongly suggested that this newly produced system performed much better than existing systems, and that it was much easier for novice users to pick up and understand. Further adding to the the system’s accessibility, it was designed entirely in Bootstrap, meaning it was fully responsive, and would display adequately, regardless of the screen size of the user (in contrast to many systems nowadays that assume users have a certain level of space, which they may not always have). It was also more visually appealing as a result of the adoption of Bootstrap.

The system has also been designed in such a way, that it is hopefully easily expandable in the future, and open to modification. One goal would be, eventually, the ability to swap the back end that the system used. This would be a relatively simple goal, as the system has been created so that the majority of task are a front end affair, and adoption of “hot-swappable” back ends would only require the implementation of some API layer between the front end and back end. The code base itself should also be relatively easy to add to and modify, due to the JavaDoc style comments that accompany every function in the system, that detail what the functions does, what parameters it expects (and their types), and what value the function returns - if any.

The final main strength of this new software system is the way that it deals with mistakes the user makes. Some systems (such as FuzzyToolkitUoN) provide extremely obtuse error messages that make the rectification of errors very difficult to the novice user. The system specified in this report implements errors message that are concise, but also explain the problem, and are worded in such a way that a solution can easily be inferred by the user (regardless of their skill level).

As with any software project, there are obviously areas where extra work could have been completed to improve the system. The largest limitation of this new software system, is in fact, it's back end. Due to the relatively limited scope of the functionality of FuzzyToolkitUoN, the functionality for this new software system is also severely limited. Luckily, however, FuzzyToolkitUoN is still in development and extra functionality may become available, thus allowing extensions for this project. However, if extra time was allocated to this project, an API layer would have been implemented between the front end and any possible back end. This would then allow the back end to tell the front end what functionality it should expect, and the user would be able to swap the back end to whichever most closely met their needs.

Another limitation with the software, that is present in any large software systems, is the potential for some unidentified bug being present. Whilst the utmost has been done to ensure there are as few bugs as there can be, it is not always possible to claim there are none. This could have been helped, if Unit Testing and Test Driven Development were adopted towards the start of the project. This is the process of writing a test, and writing the code necessary to pass the test. This ensures that all code written is free of bugs, is as concise as possible, and integration and regression tests are extremely easy to perform[22] (as all the tests from previous work still remain).

The final limitation of the project, which was noted when testing the responsiveness of the software, is that the system does not load or display correctly on mobile devices. This is mostly due to Bootstrap, and how modal windows in Bootstrap (that are heavily used throughout the software) do not render correctly properly on mobile devices. Luckily, this isn't truly an issue, as the project was never intended to work on mobile devices, as this is an impractical medium in which to construct a fuzzy system, and thus no actions would be taken to amend this.

8 Further Work

Whilst the system produced can be considered a complete system, there are still definitely improvements that could be made, and functionality that could be added. In this section, the main extensions to the system have been detailed, along with their reasoning for not being in the initial release.

8.1 Back end Interoperability

The purpose of the system is to be a web based front end, to a back end that can perform fuzzy inference. This means that the system should not be reliant on the back end that it currently uses (R-Shiny/FuzzyToolkitUoN), and it could be interchanged with some other back end (potentially with greater functionality, or better performance). This is a result of the system being written almost entirely in JavaScript, making the majority of it's functionality a client-sided affair, and not relying on the back end. The only functionality that the back end truly provides, is the ability to save files to the user's machine, and to evaluate the fuzzy system created. This means that, with the implementation of a simple API layer between the front end and back end, the back end could easily be swapped out to any back end that the user would want. This would allow the user extra functionality within the system, making it much more useful for the expert user. This could include such things as different inference types (like TSK), different t-norms, and t-conorms for the inference methods, and different visualisation techniques. This would make the system much more useful, as it currently can only perform Mamdani inference, with a limited number of inference methods, due to the limited scope of the back end. The revised architecture of the system, taking these interchangeable back ends into account, can be seen in figure 25.

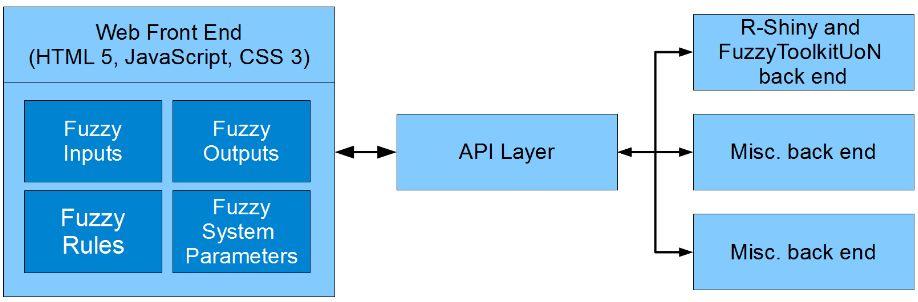


Figure 25: Interaction between the front end of the system and interchangeable back ends

8.2 More Usability Improvements

As a result of the user evaluation that was detailed in section 7.3, many potential improvements to the system were identified. Most of these were personal preferences mentioned by few participants, but others were mentioned by more participants, and some thought should go into these features, if any more work is to be completed on the system.

The first of these, noted by a handful of the participants with a high level of skill with computers, was the inability to press the return key, to save a membership function that had been created. Instead, the user was forced to click the "Save Changes" button. This was not a problem for the participants with a low level of skill with computers, as they generally use the mouse to navigate between input boxes. The expert users, however, were much more likely to use the tab key to navigate between the text boxes, and as a result, many of them expected that pressing the return key would save the membership function for them, so that they weren't required to use the mouse. This functionality *is* available, but requires the user to press the tab key twice more after their final parameter, and then press the space bar; a combination that was not noticed or used, by many users.

This leads onto the next possible improvement, of allowing more keyboard short cuts throughout the system. A good interface should be easy for novices to learn (which the results of the user feedback clearly indicate is true for this system), and efficient for experts to use. It has been found that, whilst most users do not use keyboard short cuts, they greatly increase the speed and efficient that tasks can be performed within a software system [13]. Currently, the user is required to use the mouse for the majority of the system, which is much slower than a keyboard could be, once the user had mastered the keyboard short cuts.

Another improvement that could be made, which would be useful for those wishing to learn how fuzzy logic works and for debugging purposes, would be the implementation of more visualisations of the inference process. Fuzzy inference is very easily representable graphically, and a display of these graphs on the evaluation page would be a great way for the user to learn how the inference process works so they could tweak their system accordingly, and gain a better understand of fuzzy logic as a whole. A mock up of how this could look is displayed in figure 26.

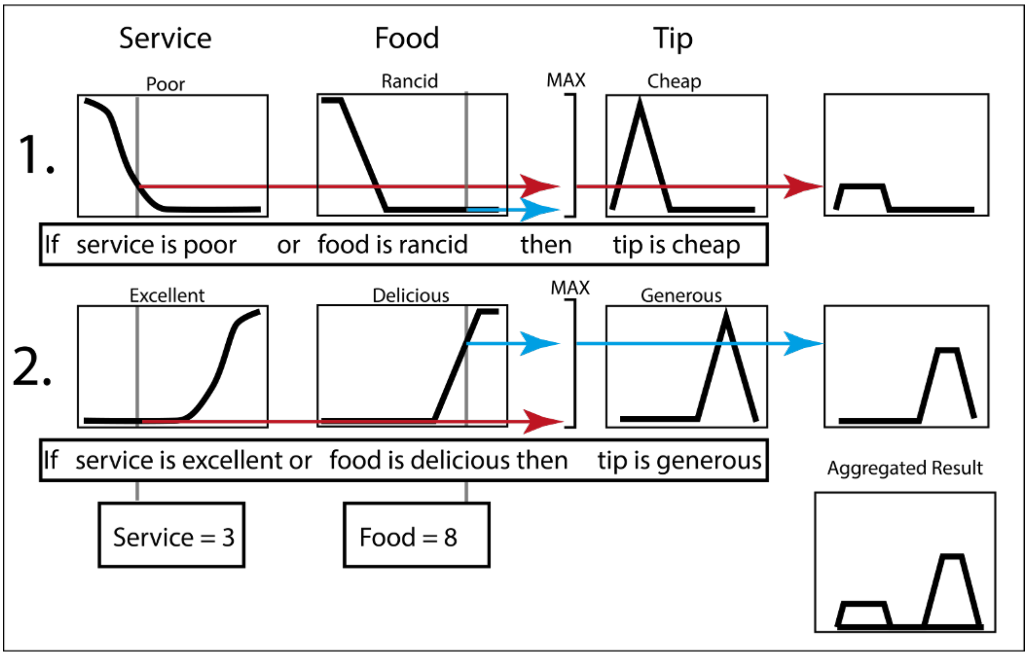


Figure 26: Example of a possible way to display the fuzzy inference process

The final improvement that could be made, would be the adoption of cookies within the system. Whilst this would present some security issues for the user, and would require a cookie agreement policy to be instated into the system, the addition of cookies could greatly increase user experience. The most obvious use of cookies would be the ability to auto-save the system the user is working on. This would mean that if they lost internet connection, or they accidentally closed the system, their work would not be lost, and they could continue working after restoring their work from a cookie.

8.3 Type-2 Fuzzy Logic Support

The purpose of a type-1 fuzzy logic system, is to model uncertainty in the terms that humans use to describe properties. Unfortunately, this is slightly hypocritical; fuzzy sets have a connotation of uncertainty, but the membership functions used are entirely certain once their parameters have been specified [16]. Type-2 fuzzy logic attempts to rectify this, in the sense that uncertainty is not only limited to the terms used, but also the membership functions defined [3]. This means that each membership function defined has a third dimension, which allows an additional degree of freedom, to directly model uncertainty [17]. An example type-2 fuzzy set can be seen in figure 27.

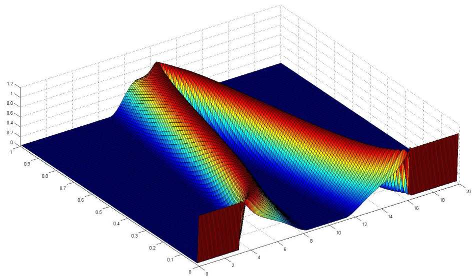


Figure 27: An example type-2 fuzzy set

As mentioned in the introduction to this report, type-2 fuzzy logic is not supported in this software. This is because the jump in knowledge between type-1 fuzzy logic (that *is* supported), and type-2, is extremely large, and would not be suitable for novice users. Other reasons include a lack of time to implement such a large feature, and that type-2 fuzzy logic is still not widely adopted in the field. If more time was allocated to the project, this could have been implemented alongside the type-1 system, however an entirely new back end would have been needed, as the current back end, FuzzyToolkitUoN, does not support type-2 fuzzy logic. Adding type-2 support to the system would have greatly improved it's usefulness and practicality, and could have easily made the tool a niche within the market, and the preferred software to use among experts. This would have been due to the ease of access, and ease of use of the system, which are features that most type-2 software systems currently available are lacking (such as Juzzy [27], which requires an installation of Java before it can be used).

8.4 Customisations

The ability to customise a website has been identified as one of the key factors for website success [7]. It has been found that users are more likely to return to a piece of software, if it is customised to their needs, and thus more personal. Whilst there is not a huge number of customisations that could be made with a system like this, there are still some, and these could potentially stand to increase usability. One of these could be the look and feel of the system, which is currently a minimalistic white design. Some users prefer to use a darker interface (especially when completing work in the evening), and thus a set of different colour schemes could be employed, which would both serve a practical purpose, and an aesthetic one. Another potential customisation could be the ability to switch to an "expert" mode, where help is less present in the system, more keyboard short cuts are supported, and there is less GUI to navigate. This would greatly increase the speed in which an expert user could complete their task within the system, as there would be less distractions.

9 Summary & Personal Evaluation

Personally, I feel as thought the project was a success, as it solved the two main issues it was conceived to solve, it met all the functional and non-functional requirements set out, and user feedback was almost entirely positive. Another reason that I believe the project was successful is that I have used the software myself, and found it a pleasurable experience. During the life time of the project, I enrolled in a module at the University of Nottingham known as *Fuzzy Sets and Fuzzy Systems*[4]. In this module, I was required to use FuzzyToolkitUoN to construct a system that would advise a doctor on how urgently a patient should be sent to the hospital, based on their heart rate and temperature. However, I found FuzzyToolkitUoN cumbersome to use, and instead used the prototype version of my dissertation software instead. This greatly sped up the process of completing this work, and gave me a unique experience to test my system as an end user.

This was the largest project that I had ever worked on by myself, but I felt as though I rose to the challenge, and employed several clever methods in order to help manage my time, and the work to be completed (such as a Kanban Board, and the use of a Gantt Chart). I did not always stick to the time scale set out by my Gantt chart, due to other work, and commitments, but the combination of the Gantt chart to track the "big-picture", and the Kanban board to track individual features made the management of the project much simpler.

One of the areas I feel as though was weaker within the project, was the research segment, or more specifically, research into software that could be used. Whilst multiple examples were evaluated against one another, this "research" was more of a superficial overview of the software, instead of an in-depth analysis. As a result of this, half way through the implementation, many issues began to arise with the tools chosen. The most prominent of these was the inability to properly construct dynamic user interfaces and have these interface with R-Shiny, and the numerous bugs and issues discovered whilst using FuzzyToolkitUoN. I also feel as though the testing process could have been much more rigorous, as numerous issues arose throughout the life time of the software development, and solving them was often time consuming, and cumbersome.

If I could work on this project again, from the beginning, there are several changes that I would make. The first of these would be a more rigorous evaluation of the tools that would be used, so that the major issues that arose with this project would not have done so. The other major change would be to implement Unit Testing and Test Driven Development from the start of the project, so that the code quality could be ensured throughout, and bugs easily identified. This includes integration, and regression testing, which would have made the addition of new features, and any extensions to the software, much simpler and easier to debug.

References

[1] Pedro Albertos and Antonio Sala. Fuzzy logic controllers, advantages and drawbacks. *IEEE transactions on control system technology*, 1998.

[2] Boris Beizer. *Black-box testing: techniques for functional testing of software and systems*. John Wiley & Sons, Inc., 1995.

[3] Oscar Castillo and Patricia Melin. Type-2 fuzzy logic. In *Soft Computing and Fractal Theory for Intelligent Manufacturing*, pages 33–46. Springer, 2003.

[4] The University Of Nottingham Module Catalogue. G53fuz fuzzy sets and fuzzy logic systems, 2014.

[5] Dianne Cyr, Milena Head, and Hector Larios. Colour appeal in website design within and across cultures: A multi-method evaluation. *International journal of human-computer studies*, 68(1):1–21, 2010.

[6] Webdesigner Depot. Interface design technique to simplify and declutter your interfaces, 2014.

[7] Wei-Shang Fan and Ming-Chun Tsai. Factors driving website success–the key role of internet customisation and the influence of website design quality and internet marketing strategy. *Total Quality Management*, 21(11):1141–1159, 2010.

[8] Kim Golombisky and Rebecca Hagen. *White Space is Not Your Enemy: A Beginner's Guide to Communicating Visually Through Graphic, Web & Multimedia Design*. CRC Press, 2013.

[9] Joshua Gross. Please stop using twitter bootstrap, 2013.

[10] Jeff Huang and Ryen W White. Parallel browsing behavior on the web. In *Proceedings of the 21st ACM conference on Hypertext and Hypermedia*, pages 13–18. ACM, 2010.

[11] Henrik Kniberg. Scrum and xp from the trenches. *Lulu. com*, 2007.

[12] Anett Kralisch, Martin Eisend, and Bettina Berendt. Impact of culture on website navigation behaviour. In *Proc. HCI-International*, 2005.

[13] David M Lane, H Albert Napier, S Camille Peres, and Anikó Sándor. Hidden costs of graphical user interfaces: Failure to make the transition from menus and icon toolbars to keyboard shortcuts. *International Journal of Human-Computer Interaction*, 18(2):133–144, 2005.

[14] Jonathan Lazar, Adam Jones, and Ben Shneiderman. Workplace user frustration with computers: An exploratory investigation of the causes and severity. *Behaviour & Information Technology*, 25(03):239–251, 2006.

[15] John D McCarthy, M Angela Sasse, and Jens Riegelsberger. Could i have the menu please? an eye tracking study of design conventions. In *People and computers XVI-IDesigning for society*, pages 401–414. Springer, 2004.

[16] Jerry M Mendel. Type-2 fuzzy sets: some questions and answers. *IEEE Neural Networks Society*, pages 10–13, 2003.

[17] Jerry M Mendel and RI Bob John. Type-2 fuzzy sets made simple. *Fuzzy Systems, IEEE Transactions on*, 10(2):117–127, 2002.

[18] George A Miller. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological review*, 63(2):81, 1956.

[19] Jakob Nielsen. The usability engineering life cycle. *Computer*, 25(3):12–22, 1992.

[20] Jakob Nielsen. Ten usability heuristics. ., 2005.

[21] Jakob Nielsen and Rolf Molich. Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 249–256. ACM, 1990.

[22] Michael Olan. Unit testing: test early, test often. *Journal of Computing Sciences in Colleges*, 19(2):319–328, 2003.

[23] Nikolaos Partarakis, Constantina Doulgeraki, Asterios Leonidis, Margherita Antona, and Constantine Stephanidis. User interface adaptation of web-based services on the semantic web. In *Universal Access in Human-Computer Interaction. Intelligent and Ubiquitous Interaction Environments*, pages 711–719. Springer, 2009.

[24] Ben Shneiderman. The future of interactive systems and the emergence of direct manipulation. *Behaviour & Information Technology*, 1(3):237–256, 1982.

[25] Shneiderman Ben Shneiderman and Catherine Plaisant. Designing the user interface 4 th edition. *ed: Pearson Addison Wesley, USA*, 2005.

[26] John Sweller. Cognitive load theory, learning difficulty, and instructional design. *Learning and instruction*, 4(4):295–312, 1994.

[27] Christian Wagner. Juzzy-a java based toolkit for type-2 fuzzy logic. In *Advances in Type-2 Fuzzy Logic Systems (T2FUZZ), 2013 IEEE Symposium on*, pages 45–52. IEEE, 2013.

[28] Christian Wagner, Simon Miller, and Jonathan M Garibaldi. A fuzzy toolbox for the r programming language. In *Fuzzy Systems (FUZZ), 2011 IEEE International Conference on*, pages 1185–1192. IEEE, 2011.

[29] Susan B Wilson and Michael S Dobson. *Goal Setting: How to Create an Action Plan and Achieve Your Goals.* Publisher - Amacom, 2008.

[30] Lotfi A Zadeh. Fuzzy sets. *Information and control*, 8(3):338–353, 1965.

[31] Lotfi A Zadeh. From computing with numbers to computing with words. from manipulation of measurements to manipulation of perceptions. *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, 46(1):105–119, 1999.

A User Evaluation Test Instructions

The purpose of this test is to construct the fuzzy logic tipper example, in three different software systems, so that the strengths and weaknesses of each of these systems can be identified. The tipper example is a simple fuzzy system that uses food quality (rated from 0 to 10) and service quality (rated from 0 to 10) as inputs, to determine how much of a tip should be left (between 0 and 30 percent).

Please follow the instructions below, and do not hesitate to ask for help if you are stuck, this is an evaluation of the user interface, and not of you. After each step, please explain your thought process, and how you found the task.

1. (**FuzzyToolkitUoN only**) Create a new **Fuzzy Inference System**
2. Add a new **input variable**, called **Service**, with range **0** to **10**
3. Add the following **membership functions** to the **Service** variable
   (a) A **Gaussian** function, named **Poor**, with parameters:
       Sigma 1.5
       Mean 0
       Height 1
   (b) A **Gaussian** function, named **Good**, with parameters:
       Sigma 1.5
       Mean 5
       Height 1
   (c) A **Gaussian** function, named **Excellent**, with parameters:
       Sigma 1.5
       Mean 10
       Height 1
4. Add a new **input variable**, called **Food**, with range **0** to **10**
5. Add the following **membership functions** to the **Food** variable
   (a) A **Trapezoidal** function, named **Rancid**, with parameters:
       Left Foot 0
       Left Shoulder 0
       Right Shoulder 1
       Right Foot 3
       Height 1
   (b) A **Trapezoidal** function, named **Delicious**, with parameters:
       Left Foot 7
       Left Shoulder 9
       Right Shoulder 10
       Right Foot 10
       Height 1

6. Add a new **output variable**, called **Tip**, with range **0** to **30**
7. Add the following **membership functions** to the **Tip** variable
   (a) A **Triangular** function, named **Cheap**, with parameters:
   Left 0
   Mean 5
   Right 10
   Height 1
   (b) A **Triangular** function, named **Average**, with parameters:
   Left 10
   Mean 15
   Right 20
   Height 1
   (c) A **Triangular** function, named **Generous**, with parameters:
   Left 20
   Mean 25
   Right 30
   Height 1
8. Add the following **rules** to the system
   (a) IF Service is Poor OR Food is Rancid, Then Tip is Cheap (Weight 1)
   (b) IF Service is Average, Then Tip is Average (Weight 1)
   (c) IF Service is Excellent OR Food is Delicious, Then Tip is Generous (Weight 1)
9. (**FuzzyToolkitUoN, and o-Fuzz only**) Please evaluate the following values:
   (a) Service score of 0, Food score of 0
   (b) Service score of 10, Food score of 10
10. Please change the **defuzzification method** of the system to **Bisector**
11. **Save** the inference system as a file to your hard drive
12. Now, read that same file back into the system

B Table of Results of User Evaluation

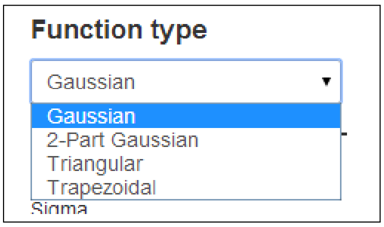
Group	**# of Members**	**Fuzzy Logic Skill**	**Computer Skill**
1	7	Low	Low
2	5	Low	High
3	3	High	Low
4	8	High	High

This table lists the time taken (in Minutes) for each participant to complete each task of the expirement. Task 1 was to follow the instruction list using FuzzyToolkitUoN, Task 2 was to follow the instruction list using the MATLAB Fuzzy Toolbox and Task 3 was to follow the instruction list using the new project created in this report. It also details the participants favourite, and least favourite software, of the three (OF standing for O-Fuzz, this project, FTU standing for FuzzyToolkitUoN, and M standing for MATLAB's fuzzy toolbox).

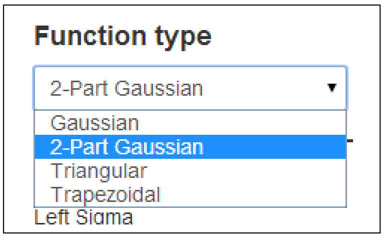
Participant	Group	Task 1	Task 2	Task 3	Most Liked Software	Least Liked Software
1	1	53	21	14	OF	FTU
2	1	45	23	15	OF	FTU
3	1	61	34	20	OF	FTU
4	1	70	36	21	M	FTU
5	1	63	31	15	OF	FTU
6	1	53	23	13	M	FTU
7	1	54	26	16	OF	FTU
8	2	35	13	9	OF	FTU
9	2	25	14	8	M	FTU
10	2	27	15	10	OF	FTU
11	2	26	11	11	OF	FTU
12	2	31	14	10	M	FTU
13	3	33	16	9	M	FTU
14	3	30	16	10	OF	FTU
15	3	40	33	13	OF	FTU
16	4	14	11	4	OF	FTU
17	4	28	12	6	OF	FTU
18	4	20	10	6	M	FTU
19	4	21	12	5	OF	FTU
20	4	12	13	6	FTU	M
21	4	25	10	4	OF	FTU
22	4	18	14	5	OF	FTU
23	4	16	11	5	M	FTU

C Complete Test Listing

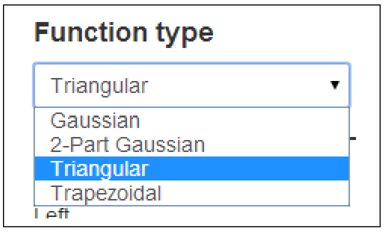
Test	**Expected Results**	**Actual Results**
Users will be able to create Gaussian membership functions	There will be an option to specify a Gaussian membership function	As expected



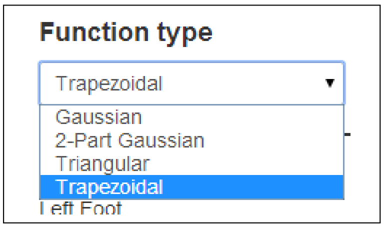
Users will be able to create 2-Part Gaussian membership functions	There will be an option to specify a 2-Part Gaussian membership function	As expected



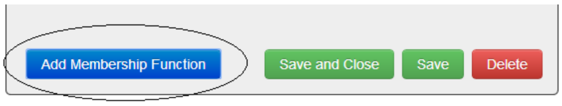
Users will be able to create Triangular membership functions	There will be an option to specify specify a Triangular membership function	As expected



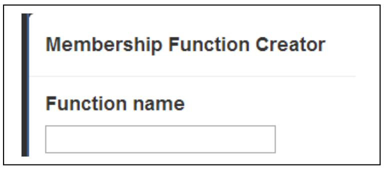
Users will be able to create Trapezoidal membership functions	There will be an option to specify a Trapezoidal membership function	As expected



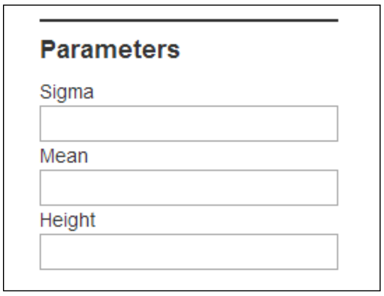
Users will be able to add membership functions to variables	There will be a button to add membership functions to variables	As expected



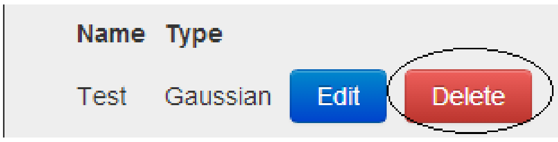
Users will be able edit membership function names	There will be a "function name" parameter box	As Expected



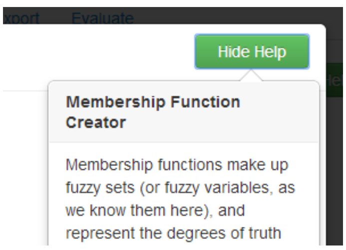
Users will be able edit membership function parameters	There will be input boxes to specify function parameters	As Expected



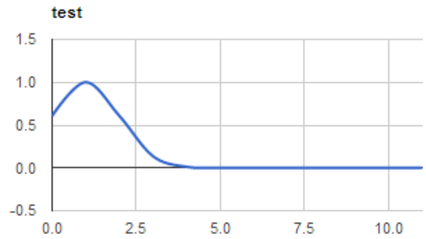
Users will be able to delete membership functions from variables	Each function will have a delete button	As Expected



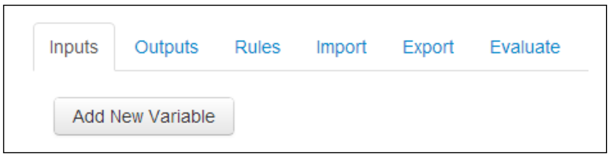
Users will be able to access help on how to create membership functions	There will be a show help button, that when pressed, will display an explanation of the window	As Expected



Users will be able to see a plot of their membership functions	A plot of the membership function will be displayed after all parameters have been given	As Expected



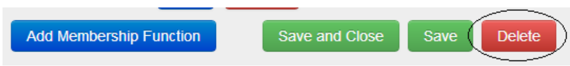
Users will be able to create input linguistic variables	There will be an “inputs” tab, with an “Add New Variable” button	As Expected



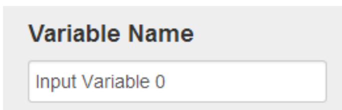
Users will be able to create output linguistic variables	There will be an “outputs” tab, with an “Add New Variable” button	As Expected

Users will be able to edit the range of linguistic variables	There will be input boxes for both minimum and maximum range	As Expected

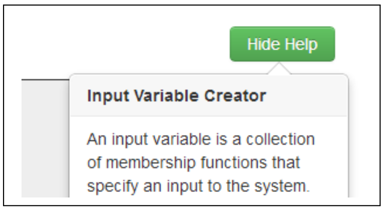
Users will be able to delete variables	There will be a button that deletes variables	As Expected



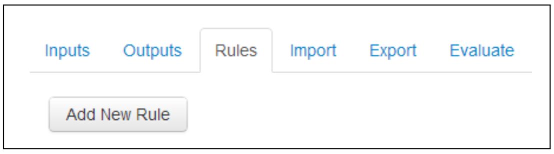
Users will be able to rename variables	There will be an input box to change the name of a variable	As Expected



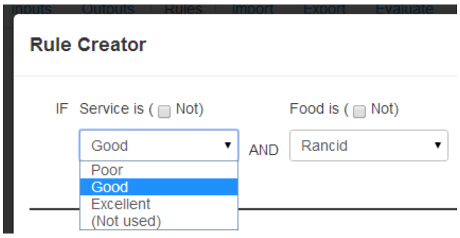
Users will be able to access help on how to create variables	There will be a show help button, that when pressed, will display an explanation of the page	As Expected



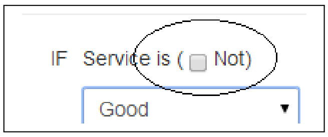
Users will be able to create rules for the system	There will be an "Add New Rule" button on the Rules page	As Expected



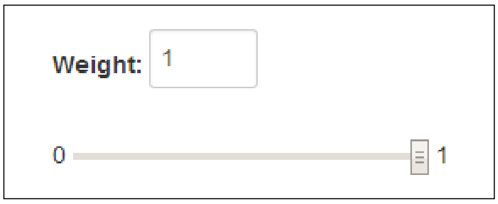
Users will be able to specify rule terms	There will be dropdown boxes for each variable in the system, that users can select terms from	As Expected



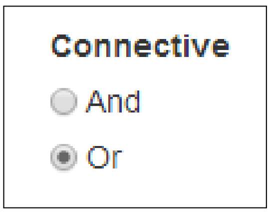
Users will be able to negate certain terms in a rule	There will be a checkbox above each term, symbolising negation	As Expected



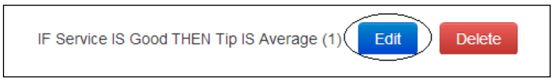
Users will be able to change the weight of a rule	There will be a slider, and input box, to change the rule weight	As Expected



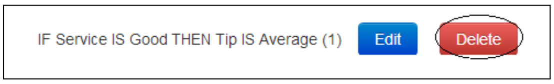
Users will be able to specify the connective to be used in the rule	There will be a set of radio buttons to toggle between connectives	As Expected



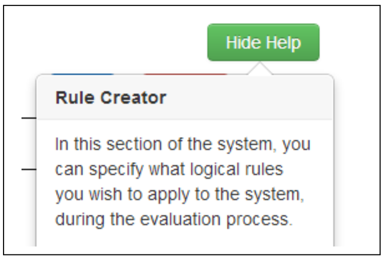
Users will be able to edit any previously constructed rules	There will be an edit button aside each constructed rule	As Expected



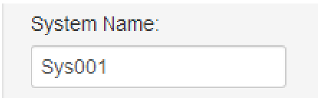
Users will be able to delete any previously constructed rules	There will be a delete button aside each constructed rule	As Expected



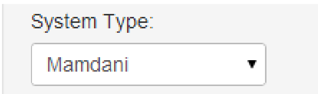
Users can access help on how to create rules	There will be a show help button, that when pressed, will display an explanation of the window	As Expected



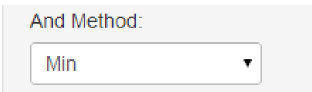
Users will be able to edit the name of the system	There will be an input box to edit the name of the system	As Expected



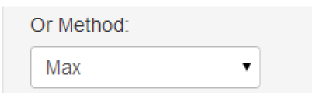
Users will be able to edit the type of evaluation to use	There will be an input box to edit the type of the system	As Expected



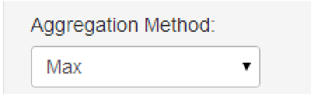
Users will be able to edit the "and" method to use	There will be an input box to edit the "and" method of the system	As Expected



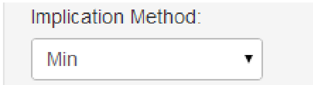
Users will be able to edit the "or" method to use	There will be an input box to edit the "or" method of the system	As Expected



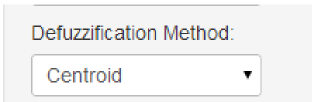
Users will be able to edit the aggregation method to use	There will be an input box to edit the aggregation method of the system	As Expected



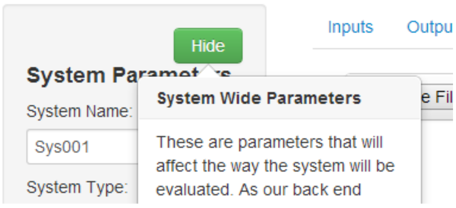
Users will be able to edit the implication method to use	There will be an input box to edit the implication method of the system	As Expected



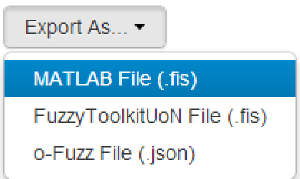
Users will be able to edit the defuzzification method to use	There will be an input box to edit the defuzzification method of the system	As Expected



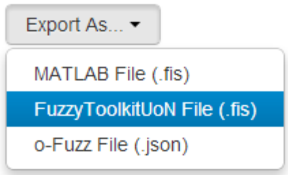
Users will be able to access help on what affect these changes make	There will be a show help button, that when pressed, will display an explanation of the parameters	As Expected



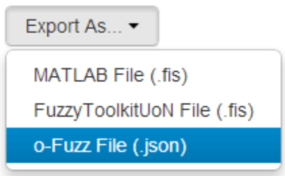
Users will be able to export their system as a MATLAB .fis file	On the export page, there will be an option to download a MATLAB .fis file	As Expected



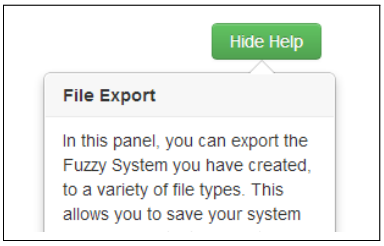
Users will be able to export their system as a FuzzyToolkitUoN .fis file	On the export page, there will be an option to download a FuzzyToolkitUoN .fis file	As Expected



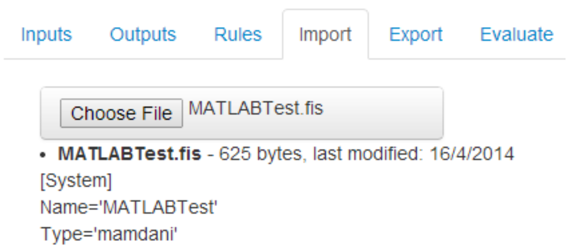
Users will be able to export their system as a JSON file	On the export page, there will be an option to download a JSON file	As Expected



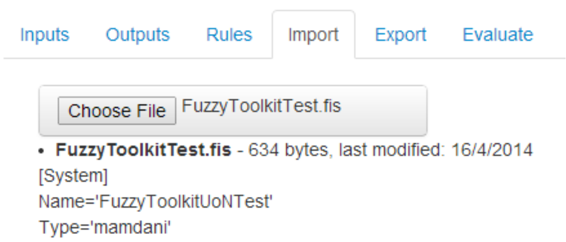
Users can access help on how to export files and what is supported	There will be a show help button, that when pressed, will display an explanation of the window	As Expected



Users will be able to import a MATLAB .fis file	Importing of MATLAB .fis files will be supported within the system	As Expected

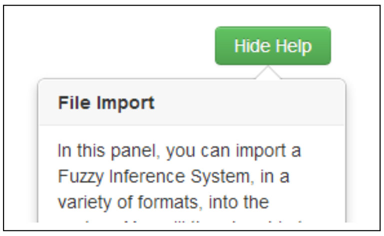


Users will be able to import a FuzzyToolkitUoN .fis file	Importing of FuzzyToolkitUoN .fis files will be supported within the system	As Expected

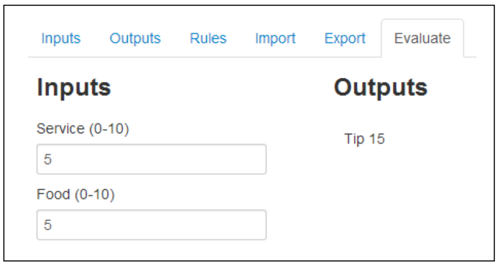


Users will be able to import a JSON file	Importing of JSON files will be supported within the system	As Expected

Users can access help on how to import files and what is supported	There will be a show help button, that when pressed, will display an explanation of the window	As Expected



Users can provide a value for each input, and receive the output value	An input box for each system input will be displayed, and an appropriate output displayed for each combination of input value provided	As Expected



Users can access help on how the evaluation process works	There will be a show help button, that when pressed, will display an explanation of the window	As Expected