An Online tool to choose a Programming Language

Final Year Dissertation

Ronan Smith

BSc Computer Science

rs6@hw.ac.uk, H00189534

Supervised by Fairouz Kamareddine

Second Reader Hind Zantout

Acknowledgements

I would like to thank my supervisor, Fairouz Kamareddine for the constant support, encouragement and feedback I received throughout the duration of this project.

I would also like to thank my second reader, Hind Zantout, for her useful feedback and opinions on my deliverable 1 submission as well as her help in advertising my usability studies to the students on the Dubai campus.

Finally, I would like to thank all the participants who took part in the various studies that were run as part of this project.

Declaration

I, Ronan Smith, confirm that this work submitted for assessment is my own and has been expressed in my own words. At any point in this document where work of another author has been used in any form (e.g. ideas, equations, research data, programming code) this has been properly acknowledged with references.

Signed: 

Date: 23/04/2018

Abstract

This project looks at the design and creation of a website which can be used by computer programmers, mainly beginners in the area, to help them choose a programming language based on the domain that they want to program in.

A number of topics are included that may be visited when choosing a programming language, for example:

1. What is a good first programming language to learn?
2. What paradigms should be considered?
3. And what actually makes a language ‘good’ in the first place?

The website has been tested and evaluated at every stage of the development with constant user feedback and a number of different testing methods. It has also been compared against similar websites that are currently in place and evaluated based on how it compares to them.

Contents

1 Aims & Objectives

The ultimate aim of this project was to produce a website that can be used to help programmers decide on a programming language to use based on their needs. This website was to be aimed mainly at beginners and was to be simple to use, with lots of information on a large number of programming languages. To achieve the aim, the following objectives were set:

1.1 Carry out the relevant research on programming and programming languages

We will discuss the research has been carried out as part of this project to find out more of a background about programming and programming languages. This research involves summarising information from a number of sources, including articles, books and web blogs. The areas discussed answer, amongst others, the following questions:

- What makes a 'good' programming language?
- What is a good first programming language to learn?
- What are the paradigms you should consider when choosing a language?
- What are the most popular programming languages?

1.2 Widely explore similar or overlapping products that already exist

We shall explore the relevant online tools (websites) that are already out there including one website that fits the description *'an online tool to choose a programming language'* as well as websites that are used by computer programmers in general. These can be used for inspiration and as a benchmark to try to improve upon.

1.3 Build a solution that achieves the aim

The steps involved in designing and implementing the solution - *'an online tool to choose a programming language'* - will also be described in detail. The solution described is called 'LaTCh' (Language and Technology Chooser). The LaTCh website is designed to do at least the following things:

- Effectively provide users with information about a wide range of programming languages.
- Ask users to answer a few questions which will ultimately lead them to a programming language relevant to their needs.
- Be simple and easy to use but also full of information about programming and programming languages.
- Be welcoming to programmers of all different levels, from beginners to experts.

1.4 Evaluate the new solution and compare it with existing products

Regular testing was carried out throughout the duration of this project to evaluate that the product being built always met the requirements and always fitted the needs of the end user.

The full details of the testing and user feedback is discussed as well as the changes that were made at each prototype stage based on the results of testing. Potential users were also given the opportunity to compare and evaluate the new website in comparison to the similar websites mentioned as part of objective 1.2 and the results of this comparison will also be discussed.

1.5 Chapter Summary

- **Chapter 1:** The aims and objectives of this project are discussed.
- **Chapters 2 & 3:** All background research is discussed including research on programming and programming languages and the exploration of similar products.
- **Chapter 4:** All steps involved in designing and implementing the final product are discussed here.
- **Chapter 5:** The testing and evaluation of the website is the focus of this chapter.
- **Chapter 6:** Explains the 'Professional, Legal, Ethical and Social' considerations that were undertaken as part of this project.
- **Chapter 7:** Concludes the document by discussing how well the aims and objectives have been made and the future plans for the product.
- **Chapter 8:** References
- **Chapter 9:** Appendices

2 Literature Review

Computer programming can be a lot of fun, but also very challenging. An important aspect in programming is choosing the programming language one wants to use. Depending on what problem a programmer is trying to solve there may be hundreds or even thousands of possible languages to choose from. How can it be decided which one suits the situation best and how can a programmer be sure they are making a correct decision?

Computational problems that can be solved in finite time using step by step instructions can be called 'effectively computable'. According to the Church-Turing thesis (also known simply as the 'Church's Thesis'), "all [effectively] computable functions are computable by Turing Machines" [42]. In addition to this, all programming languages are based on the same foundations that Alan Turing's theoretical machine was based on. These points boil down to one important fact, discovered by Church Rosser and Alan Turing – in theory, any program written in one specific programming language can also be written in any other programming language one can think of, so in effect, all programming languages can solve all effectively computable problems. This is another reason why selecting a language is so difficult. Certain languages are better for certain tasks however. For example, it would be extremely difficult to model the lambda calculus (an abstract mathematical theory of computation [46]) in Java [47] but much easier in SML [48]. Although (in theory) it is possible, it would extremely difficult to build a web application in SML, but much easier in Java.

Choosing a programming language is a problem that depends very much on the context of the situation. The relevant details may include the experience of the programmer, the type of problem being dealt with (is it a logical problem? Does it deal with functions? Does it deal with objects?) and the problem itself – a new problem with little to no documentation or a well-defined problem which has been dealt with in many different ways before?

There isn't usually one specific answer, but the following sections discuss some areas that should be visited when choosing a programming language (or creating a tool to help do this) in much more detail.

2.1 What makes a good programming language?

This is a topic that is very much based on opinion and each programmer is likely to be bias towards certain languages based on their own previous experience and the reasons they generally use programming for. However, it is likely that most programmers would agree on the following statement; "a programming language is good if it helps us to write programs that are easy to read, easy to understand, and easy to modify" [1].

There are a huge number of ways we could categorise what makes a good programming language, but for the purpose of this project we will stick to the following 5; writability/readability, simplicity, definiteness, modularity and efficiency, described below [1].

2.1.1 Writability and Readability

A program that can be easily understood by the programmer **writing** it and others **reading** it is highly maintainable. This is something we like to see in a programming language, as maintainable programs are easy to extend in the future and can save a lot of development time for future programmers using that program code. Although the readability of a program is mostly controlled by how well the programmer actually writes it, this can be helped along by a programming language that is designed to be more readable in the first place. For example, let's picture a program that adds together the numbers from 1 to 10 and pictures the result. Figure 1 shows the source code for this program in MIPS Assembly [62] and figure 2 shows how the equivalent program would look in C [54].

```
      .file "adder.c"
      .section    .rodata
.LC0:
      .string     "The total is %d\n"
      .text
      .globl      main
      .type main, @function
main:
.LFB0:
      .cfi_startproc
      pushq %rbp
      .cfi_def_cfa_offset 16
      .cfi_offset 6, -16
      movq  %rsp, %rbp
      .cfi_def_cfa_register 6
      subq  $16, %rsp
      movl  $0, -4(%rbp)
      movl  $1, -8(%rbp)
      jmp   .L2
.L3:
      movl  -8(%rbp), %eax
      addl  %eax, -4(%rbp)
      addl  $1, -8(%rbp)
.L2:
      cmpl  $10, -8(%rbp)
      jle   .L3
      movl  -4(%rbp), %eax
      movl  %eax, %esi
      movl  $.LC0, %edi
      movl  $0, %eax
      call  printf
      movl  $0, %eax
      leave
      .cfi_def_cfa 7, 8
      ret
      .cfi_endproc
```

Figure 1 – MIPS Assembly program to add the numbers together from 1 to 10.

```
#include <stdio.h>
int main(){
      int i;
      int total = 0;
      for (i = 1; i < 11; i++){
            total = total + i;
      }
      printf("The total is %d\n", total);
}
```

Figure 2 – C program to add the numbers together from 1 to 10.

Of course, there are still uses for languages like MIPS Assembly (see chapter 2.5.5) in low-level programming and hardware manipulation, for example in operating system (OS) development and embedded systems. But for a programmer who has experience in high level programming and is working at a higher level, the block of C program code above would be much easier to read, understand and hence work with than the corresponding MIPS Assembly code, which is about 30 lines longer (depending on how you lay it out) and seems much more complicated to someone who is not an expert in Assembly.

2.1.2 Simplicity

Languages that are designed to be **simple** tend to be easier for programmers to use. "Simplicity is about reducing incidental complexity as much as possible in order to be able to focus on the complexity that is inherent to the problems we solve" [2]. In other words, having a programming language with a small number of easy to remember features allows the programmer to concentrate on what really matters – the program that is actually being written – rather than having to remember 200 or more useless extra features to remember. The Pascal [49] programming language was an early attempt at producing a small, simple and efficient alternative to the low-level languages that were already around. And in more recent times, we have seen languages like Python [50] and Swift [51] which are designed to be simple to pick up and learn.

2.1.3 Modularity

Modularity in programming means keeping sections of code relevant to a single task each. Writing code this way rather than having extremely long programs with all different functions stuck together enhances simplicity and in turn makes it easier for programmers to read and understand. This may be one of the reasons that Object-Oriented Programming is so popular.

Typically, it is seen as good programming practice to have high 'cohesion' and low 'coupling' in your programs. High cohesion means each block of code is designed for one

task and one task only, and low coupling means separate blocks don't rely on each other too much. As well as the benefits described above, writing programs in this way makes it much easier to change them in the future as separate code sections can be edited separately without having an effect on the others.

2.1.4 Definiteness (Having a definite philosophy)

A programming language with a **definite philosophy** – meaning it should be designed for a definite and specific reason as much as possible, can be seen as a highly useful language for the purpose it was built for. For example, the Visual Basic [52] language developed by Microsoft was designed to be easy to use for new programmers, and so is an ideal language to use in schools, however it would not be quite so useful for implementing huge industrial applications. Prolog [53] is great to use when working with logic but might not apply to other areas quite so well. Both are designed for a specific purpose and philosophy in mind which they work well for.

2.1.5 Efficiency

Efficiency has a huge part to play in deciding how good a programming language is. Compiled languages like C [54], C++ [55], Rust [56] and Ada [57] are ranked as some of the most efficient languages [28], due to the fact that they don't need to be compiled (translated from source code into assembly code – a potentially slow process) and certain manipulations can be carried out manually by the programmer such as memory allocation. If the programmer knows what he/she is doing, this can improve efficiency greatly. However, some programmers would prefer not to have to deal with the overhead of doing things themselves quite so much. We look to find a balance where the program is making a good use of system resources and performing at fast speeds but the programmer is also able to create the program code to an efficient standard. Just because a language needs to be compiled does not automatically mean it is inefficient, and Java is an example of a non-compiled language that is considered to be efficient.

2.2 A good first programming language to learn

If you plan on working as a programmer, the first language you learn can be a very important one. For young people, learning through a language that they enjoy may help to encourage them into going for a career in programming, whereas learning through a language that is difficult to work with may put them off. However, a first programming language can't just be chosen because it is fun to work with (at least not in education) but it must also teach you the fundamentals that you need to know and that you can apply to all the other programming languages you use in the future. If your first language gives you skills you can take into another language, then you are off to a good start. The subsequent sections (2.2.1 to 2.2.5) will discuss how programming is taught and learned throughout a learner's growth, from starting out to studying programming at university.

2.2.1 The Traditional Approach

The traditional approach to learning programming would be learning the basics of some large, general purpose language such as C, Algol [58] or Java and being given a set of tasks that work with number or symbol processing. This has its problems, however [4]:

1. General purpose languages are too big and have too many features. These can be difficult for a new programmer to remember and may make them feel inadequate.
2. There is not much of a visual side to general purpose languages, and so it is not so easy for the programmer to see and understand what the code they write is actually doing.
3. The tasks that students tend to work on don't really appear relevant to their everyday life and so are not as interesting and appealing to them.

2.2.2 Mini Languages

This is where ‘Mini Languages’ or ‘visual programming languages’ come in. A mini language is designed to have a small number of features and a small syntax with simple semantics [4]. At a simple level, this keeps things easy to remember for the programmer. Furthermore, mini-languages tend to be visually appealing, often created in a visual block format and often producing results that are interesting to look at.

Scratch [59] is an example of a mini-language and one that is used in primary and secondary schools for beginners in programming. It works well for new programmers as it is completely visual. Programs are ‘written’ by sticking together coloured blocks, like jigsaw pieces and the program itself usually involves moving a character, called a ‘Sprite’ around the screen. MIT App Inventor [5] is another example of a block programming system and this can be used to create very simple to very complex Android apps without seeing a single piece of source code. These are great ways to learn because they provide an interface where programmers can see the changes they make coming into action, and they can create interesting, working results very quickly. Some mini languages, like ‘Pocket Code’ [29] are so small they can literally be held in your pocket (as an application on your smartphone). Figure 3 below shows what block programming in MIT App inventor or Scratch may look like, and this is typically accompanied by another view, where the app developer can design the user interface by dragging components and dropping them onto the screen.

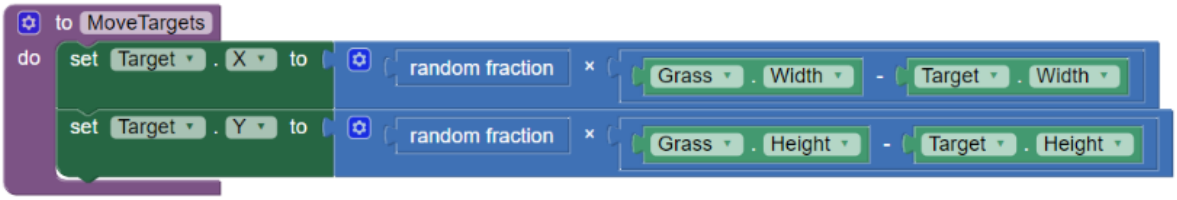


Figure 3 – A collection of blocks, created using MIT App Inventor.

2.2.3 Introduction to Source Code

Once source code starts being revealed to novice programmers, things can quickly become daunting and complicated for them. It is important to gradually introduce them to what source code looks like and to try to build a bridge between what they have learned in a Mini Language (see section 2.2.2), for example, and what it looks like under the hood. Visual Basic is a useful language at this stage as it allows the programmer to create an interface containing buttons, labels, text/input boxes and much more, simply through dragging and dropping items. Changing the view in the Visual Basic or Visual Studio IDE (Integrated Development Environment) can then allow them to start writing program code for each of the interface items in turn.

In figure 4, you can see what the design view might show in Visual Studio. Here you can drag and drop components onto the interface like buttons and labels. This is accompanied by another view where actual Visual Basic source code can be written that corresponds to the interface.

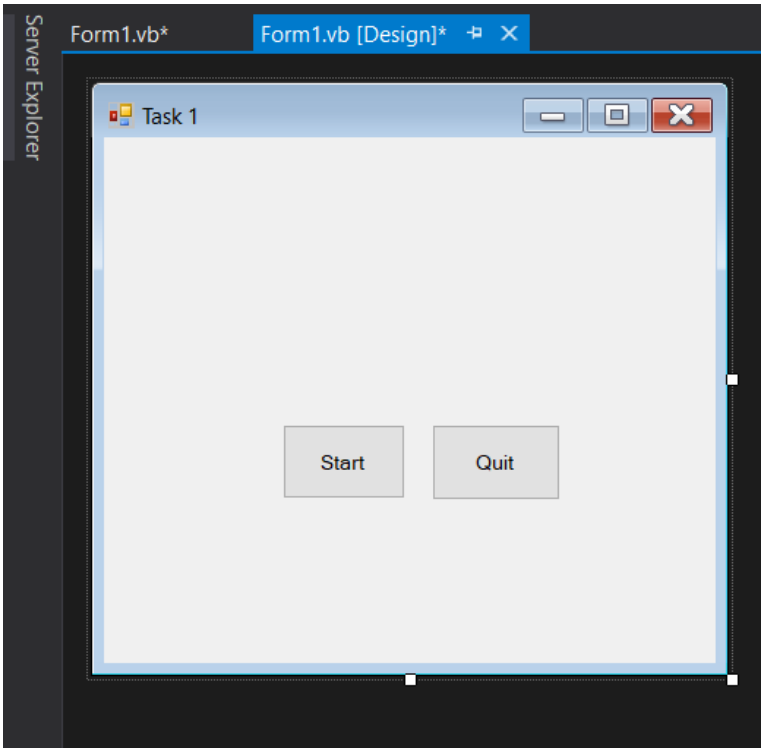


Figure 4 – The 'Design View' in Microsoft Visual Studio.

2.2.4 Programming at university

At the beginning of university things can become interesting for what programming languages should be taught as a programming introduction. The problem at this stage is that everyone has come from different backgrounds, countries and education systems, and so not everyone is at the same level of expertise. It could be easy for the university to look at industry and to decide to start teaching languages that are used in software development companies, however, this isn't necessarily the best idea. As Doug Grant said in 1992, "despite the apparent industry demand for students familiar with C and C++, these need not be the languages of choice for early computer science courses" [6]. This is still relevant today. It is not necessarily important to know a specific language that is used in industry, but it is important to learn the skills and principles needed to program in the real world.

2.2.5 There's no right answer

There is no standard way of doing things, and even the SQA (Scottish Qualifications Authority) have not set a specific language to work with in the new 'Curriculum for Excellence', feeling that it should be up to the teacher to decide based on their own experience and knowledge. They have, however, defined their own pseudocode reference language called SQARL (SQA Reference Language [20] – formerly called HAGGIS) to set the general idea of structure in programming. At university level in the United Kingdom, Java dominates introductory programming courses, although Python could be considered as easier to use [7] and potentially a better option. This may be because Java has been around for longer, or due to university lecturers having more experience with Java, or because there is more documentation out there for it. According to a study carried out by Philip Guo [30], Python is more popular in universities in the United States of America for introductory courses (see figure 5), but Java is also highly used there as well.

At the end of the day the specific language taught is not important, but what can be learnt from it.

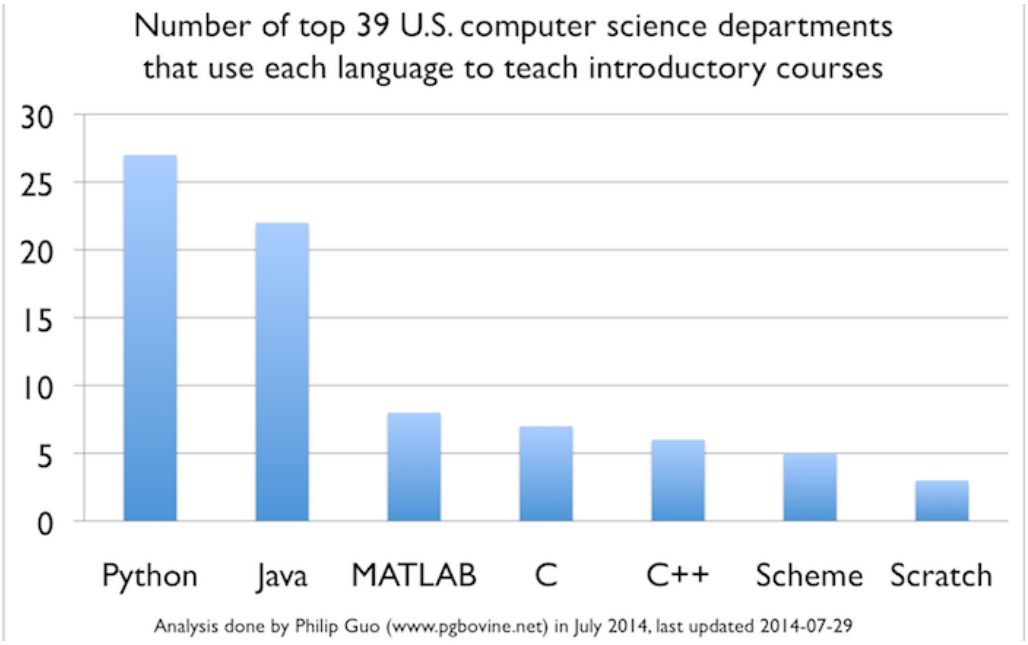


Figure 5 – Languages used for introductory programming courses at US universities.

2.3 Most popular programming languages

What are the most popular programming languages? Well, this isn't an easy question to answer straight off. Programming languages are tools, and one tool may be extremely useful in one situation, but not so useful in another. The popularity of a programming language depends largely on the domain (the type of problem) that it is being used for and possibly the paradigm (see chapter 2.4) it will be working in.

In 2015, most rankings of programming languages would see the "big 5" right at the top or around about the top. These were Java, C, C++, Python and C# [14], and you would often expect languages like JavaScript and PHP in the top 10 somewhere too. However, a more recent article discussing the top languages in 2017 according to GitHub [13] puts JavaScript right at the top with 2.3 million pull requests (requests to view code of that type using GitHub) in comparison with 1 million for its closest contender - Python - and shows that more

recently languages like C and C# have dropped in the rankings with languages such as Ruby [60], R [61] (a language mostly used for handling and visualising big data) and Swift [51] (Apple's relatively new programming language that can be used for creating mobile and desktop applications) climbing up the way.

In comparison, Jay Patel on the 'Coding Dojo Blog' [21] placed SQL at the top of the list of top 9 programming languages (see figure 6), and this didn't even appear in the GitHub list. This may be partly due to some sources considering SQL as more of a 'scripting language' than a 'programming language', but again it highlights that the best language is a matter of perspective and differs depending on the problem you are trying to solve. The list given at [21] was based on job postings on the popular job search website Indeed [22], and a graph representation is given below.

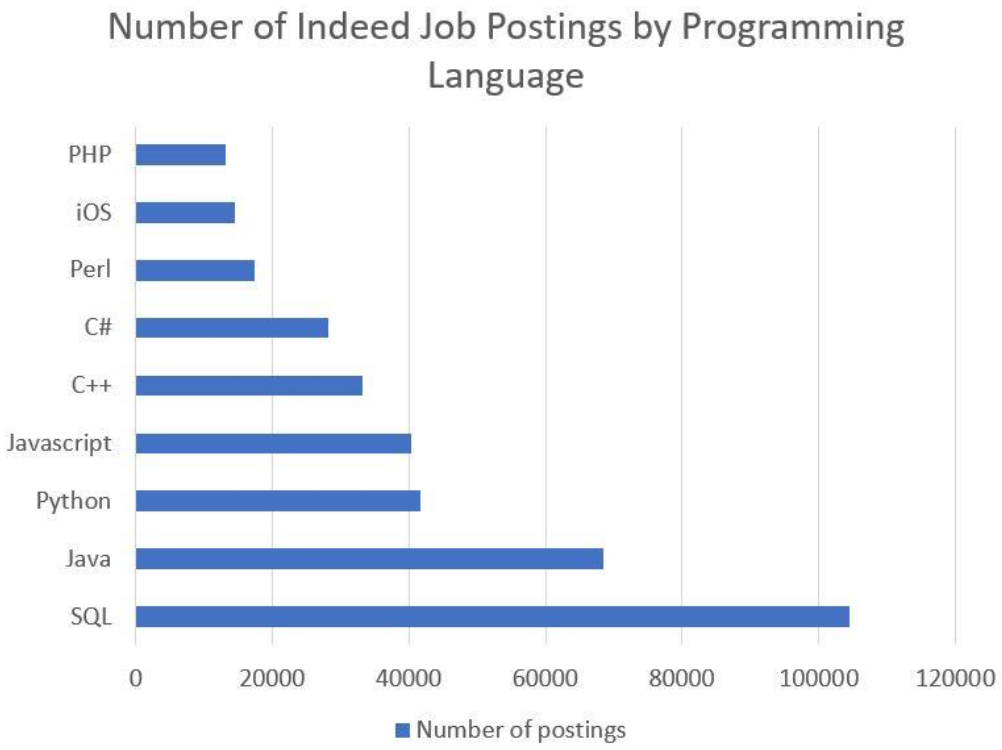


Figure 6 – The most popular programming languages based on Indeed job postings.

The most popular languages are always changing as time goes on, with more modern solutions constantly being worked on that can provide improvements on older languages. The language that you choose to use is up to the programmer themselves and should be chosen based on what they want to do with it, however, it is always useful to know what everyone else is using.

2.4 Programming Paradigms

A programming paradigm is a style, or "way" of programming [8]. Some languages may be better suited to solving different problems depending on what paradigm or paradigms they fit in to and some problems may be suited more to specific programming paradigms. The following sections will discuss 4 different paradigms; imperative, logical, object-oriented and functional.

2.4.1 Imperative Paradigm

In an imperative programming language, programs are written as a structured series of steps, which are executed in order, line by line, top to bottom, left to right. At the end of the sequence of steps a problem should have been solved or at least part of a problem and any change to the order of execution in an imperative program will change the results. This is one of the simplest paradigms and often one of the first taught in schools.

Some advantages of the imperative programming paradigm are as follows [9] [10]:

- **Efficient;** This is certainly true with fairly small programs as there is no need for function calls and a huge number of pointers around the program. The next instruction is always on the next line.
- **Close to machine code;** Low level programming like Assembly can be seen as imperative. Assembly languages are closely linked to machine code and the imperative paradigm is closely related to assembly code. This makes it easier to work with the imperative paradigm at low levels.

- **Iterative Process;** The process is made up of step by step instructions which should be fairly easy for the programmer to follow, especially in small programs.
- **Familiar;** The style of writing down instructions line by line is similar to the way the programmer might write down their tasks for the day in real-life and so it makes sense to them.

There are however, disadvantages [9] [10], some of which are listed here:

- **Debugging can be harder than other paradigms;** This is due to the fact that most variables in an imperative program are likely to be global and so they can be changed a lot over the course of execution, making them harder to keep track of.
- **Abstraction is limited;** Due to this, it is much more difficult to hide irrelevant data from view and thus ends up with a lot of mess on the screen.
- **Order is crucial;** This can be a problem as it means making changes to the lines of program code has to be done with much more thought.
- **Complex processes lead to "spaghetti code";** This is a way of describing code that gets out of control. Complex processes in a language following the imperative paradigm may not be as easy to break down into modules and therefore can get very complicated very quickly.

Some examples of languages that fit into the imperative programming paradigm are C, Fortran, Pascal and (non-object-oriented) Python.

2.4.2 Logical Paradigm

In the logical programming paradigm, programs are written in a more declarative way. This means instead of the traditional approach of writing line by line instructions in how to solve certain problems, a set of facts and rules is defined that can be applied to certain types of problem to try and solve them. The program is not told exactly how to solve certain problems but instead it applies the rules it has been provided with in order to find solutions. There are two main advantages to using the logical paradigm [9]:

- **The system solves the problem, not the programmer;** This keeps the actual programming of the system to a minimum and keeps things fairly simple.
- **Proving the validity of a given program is simple;** A program is made up of a set of facts and possibly a set of rules to connect facts. To run the program a series of queries can be written which will simply return the answer 'true', 'false' or the value of a variable.

Some disadvantages of the logical programming paradigm are as follows:

- **Programming in this paradigm is substantially different from most other paradigms** [11]; If the programmer has not worked with a logical language before it can be difficult for them to get into the right way of thinking to use this paradigm. Using a logical programming language requires a certain type of thinking which may not suit some programmers well.
- **The logical paradigm is rather deficient** [9]; It does not have a lot of the built-in characteristics that you would expect to see in object-oriented programming languages like Java or C++, for example.

The logical paradigm is great for dealing with logical formulae and is useful for creating tools such as expert systems; where a knowledge base is used to store facts and rules, and queries (structured questions based on the data) can be sent to the knowledge base through a means of communication called an inference engine. However, the logical programming paradigm does not fit well to general programming problems due to it being so different from other paradigms. Prolog is one of the most well-known logical programming languages but another example is Mercury [69], which can also be considered as a functional programming language (see chapter 2.4.4). It is important to point out that many languages actually cover a number of different paradigms rather than just the one.

2.4.3 Object-Oriented Paradigm

In the object-oriented paradigm, programs are defined through an abstraction of real-life objects. Everything that can be defined in some way to be its own entity is implemented as a 'class' with its own attributes and behaviours in an object-oriented program. Objects are kept as independent as possible but must also interact with each other to produce interesting results.

Some advantages or useful features of the object-oriented paradigm are as follows [9] [11]:

- **Inheritance;** The capability to allow classes to inherit from parent classes is a useful one. It gives a subclass (a class inheriting from a parent or superclass) the ability to inherit all (through extension), or some (through inheritance) of, the attributes of the superclass, allowing better control of structure of the class and less code repetition.
- **High amount of cohesion is possible;** The concept of objects allows programs to be written in a very cohesive way (i.e. tasks are split up into code fragments that are created for only that task). Highly cohesive programs can also be described as highly modular, which was a feature mentioned as important in chapter 2.1 'What makes a good programming language?'.
- **Encapsulation and Abstraction;** These are two features of object-oriented programming that keep the data that can be seen and accessed at any point in the program relevant and secure. This is useful as it stops parts of the program interfering with others; for example, one class changing another classes variable values.

The object-oriented (or OO) paradigm does have some disadvantages however, for example [12]:

- **Steep learning curve;** For people who are new to programming, or who haven't dealt in an object-oriented way of working before, the thought process involved in

object-oriented programming can be difficult to get their head around and may take a while to understand fully.

- **Suitability to general problems;** The object-oriented paradigm is only suitable for use on problems that can be broken down into distinct objects. Some problems just don't work with this paradigm and although they could probably be written in OO, they would not be very efficient.
- **Slower programs;** In general, an object-oriented program will be much larger than its imperative counterpart. This is due to the fact that everything is broken up into classes. The size of OO programs can cause them to be much slower as well as the fact that lots of function calls are necessary in comparison with an imperative program.

Object-Oriented programming languages such as Java and C++ are among the most popular in the world, numbers 3 and 6 respectively according to Github, sometimes referred to as the "Facebook for programmers" [13]. They are extremely useful and can be used to produce some outstanding pieces of software, but it should be remembered that they don't apply well to all types of problem.

2.4.4 Functional Paradigm

In the functional programming paradigm, "computation proceeds by rewriting functions, not by changing state" [11]. In other words, the functional paradigm has been built around a notion of not having any memory. There are no global variables for example (variables that can change throughout the scope of the program) and once the environment is decided, it cannot change unless it is overwritten. This is a very different approach to most other paradigms, where operations like assigning a value to a variable; an impossible situation in functional programming, are completely normal and expected.

This approach has a number of advantages, as described below [9]:

- **A High Level of Abstraction;** This means that irrelevant data is hidden from the programmer. At any point in programming with a functional language the irrelevant data that they don't need to know about is there but hidden from view. This is an advantage because it reduces the possibility of the programmer committing a number of different types of errors.
- **Lack of dependence on assignment operators;** With a lack of dependence on assignment operators, the order of execution in a program is no longer important. This is extremely useful as it means functional languages are ideal for use in parallel environments, where the order instructions are executed in tends not to be fixed. Parallel environments are becoming more and more common as well, with the rise of multi-core computers.
- **Referential Transparency;** It is easy to translate a program written in the functional paradigm back to mathematical functions and this makes it easy to carry out mathematical proofs and analysis in comparison to programs written in another paradigm, as well as making it easier to test for correctness against the original functions.

Some disadvantages of the functional paradigm are as follows [9]:

- **Efficiency;** Certain data structures such as arrays are not as easy to implement in functional programming languages and may have to be created with much less efficient code.
- **Lack of support for variables;** Due to the fact that global variables and assignment can't be used in functional languages, it can be difficult to solve a problem that contains many different variables. The programmer should go with a different paradigm if this is the case.

Functional languages, like SML New-Jersey are ideal for use in problems that contain a lot of functions. They work well with mathematical problems, due to how easy it is to translate

between mathematical functions and the functional paradigm, and they give you a different approach than other paradigms by not having global variables or state. The functional paradigm can be inefficient however and it should not be used to solve problems that contain a large number of variables.

2.5 High vs low-level programming

Computers store data in a base-2 numeral system called 'binary', where every possible value is stored using a combination of only two distinct values; 0 and 1. Binary representation is very effective for computers for a number of reasons, for example consider the following 2; First, it is simple to store. The values for 1 and 0 can be stored by transistor switches which can be made very small and are very cheap to mass produce. Second, the binary rules of arithmetic are simple (there are only four); 0+0=0, 0+1=1, 1+0=1 and 1+1=10 and this means they are easier to store and represent in hardware. The problem with binary representation comes only when some human needs to understand it. Binary representation, or machine code as it is also known is very difficult – almost impossible – for a human to read and understand right away and this is why programming languages are necessary. From low-level to high-level, programming languages provide more and more abstraction over machine code, becoming easier for a programmer to read and work with as they get higher. Here, I will discuss the different levels of programming with reference to some languages that are used at that level.

2.5.1 Assembly Code (Low-Level Programming)

Assembly has been mentioned before in chapter 2.1.1, where I highlighted that for many programmers, it is very difficult to read and understand at first glance when compared with a higher-level programming language like C, but it is much closer to machine code and therefore can be used for much more low-level tasks such as explicit memory allocation or direct hardware manipulation. Assembly languages are specific to the hardware they are written on. This means they cannot be freely moved around different computers without

thinking about the underlying hardware, in other words they are not portable. Assembly languages like ARM and MIPS are useful for tasks like building operating systems or compilers, reverse engineering (trying to understand someone else's compiled code) or building device drivers [15], as all these tasks involve communicating with and manipulating computer hardware.

2.5.2 C (System-Level Programming)

C and other system-level programming languages attempt to bridge the gap between low-level and high-level programming languages. They are an attempt to keep most of the low-level features of an assembly language available while making the syntax a bit easier to work with. C is much more portable than MIPS or ARM Assembly although it can still give different results when it is run on different hardware. C is a widely used language, making it a good one to learn. It also forms the basis of many of the high-level languages that are around today.

2.5.3 Java (High-Level Programming)

High-level programming languages like Java are about as far away from hardware as you can get in terms of abstraction. All memory allocation in Java is carried out automatically meaning the programmer does not need to think about the hardware at all. This has advantages as it allows the programmer to concentrate on the programming task in hand without having to worry about the extra overheads of hardware manipulation. However, as the compiled assembly code is produced automatically by Java Virtual Machine (JVM) it can be much less efficient than if a human programmer had written it themselves. Java is highly portable and can be used on any machine architecture that has JVM installed. JVM compiles Java source code into an intermediate language called Java Byte Code before compiling it to the relevant assembly language for the architecture it is running on. The underlying architecture, for example the processor that is being used in the computer the

Java code is being written on, is not something that has to even be considered by the programmer; everything is dealt with behind the scenes.

Java is ideal for high-level programming tasks such as mobile/web/desktop application development, but not so useful if you need direct control to hardware.

2.5.5 Other types of programming

It would be impossible to discuss every single type of programming in one document but here, some of the other types of programming, not discussed so far, are summarised:

- **Mark-up languages**, for example HTML [62] and XML [63] are used to control structure in documents databases, or applications. HTML is the most popular mark-up language in the world as it is the standard language for creating web pages on the world wide web. The reason it doesn't appear in chapter 2.3 'Most Popular Programming Languages' is because it is not technically considered to be a 'programming language' by most sources [31] (although for the purposes of this project, we will consider it as such. It is still a 'language' that you may want to choose).
- **Database Manipulation Languages** are used to implement and manipulate databases and are typically run within a database management system (DBMS). SQL (Structured Query Language) [64] has been one of the most popular for many years, however, as the need arises for databases to be more scalable, competitors such as MongoDB [65] and Neo4j [66] are arising.
- **Data Science** is another rising area of interest as big data becomes more and more relevant. In 2014, it was said by one source that "in the last five years, more scientific data have been generated than in the entire history of mankind" [32]. As we process more and more colossal amounts data, it is important that we can analyse and manipulate it. This is why languages like R are on the rise.

- **Scripting Languages** are small and quick. They aren't usually used to produce large-scale applications but are more likely used as small code sections in larger systems, for example JavaScript [67] embedded in HTML documents. They can also be used to carry out small, simple tasks like manipulating or searching text files. Server-side scripting is used at the back-end of websites, usually to interact with a database or some underlying application. PHP [68] is an example language used in this context.
- **Real-time embedded systems** rely "not only on the logical results of the computations [they carry out], but also the physical time when these results are produced" [32]. They need to complete their calculations in a set, usually short time. For example, they are used in safety critical systems, like a computer-controlled braking system in a car. Assembly languages, like ARM or MIPS and low-level, compiled languages like C and Ada are commonly used in these areas, due to their efficiency and speed. The best languages to use in safety-critical systems are discussed in "Languages for Safety-Critical Software: Issues and Assessment" [33].

3 Technical Literature Review

In this chapter three online resources are discussed that are commonly used by programmers. All the sites discussed have at least some features that have been included in the final LaTCh website. The specific resources discussed are Codecademy [16], Stack Overflow [17] and Best Programming Language for Me [18].

3.1 Codecademy

Codecademy is a website for learning programming languages. It provides lessons and tutorials for learning specific languages as well as tasks that one might need to do as a programmer such as setting up a live website. Codecademy also allows one to choose what to learn by category (programming, web development or data science) or by language, which is something that the LaTCh website implements and extends upon through the 'categories' section. Another interesting feature of Codecademy is that when a user selects an area of programming, it also tells them what the average salary is for someone who works in that area. A similar feature has been implemented in the LaTCh website for specific languages. See table 1 (section 3.4) for a summary of the Codecademy advantages and disadvantages.

3.2 Stack Overflow

Stack Overflow is an online community for programmers to come together and learn about programming, mostly through asking and answering questions that all other members of the system can get access to. Related questions usually appear from your Google (or other search engine) query but you can also search for questions by 'tag' which usually means the name of the programming languages that are involved in the question. This system also provides job listings for software developers. See table 2 (section 3.4) for the advantages and disadvantages of Stack Overflow.

3.3 Best Programming Language for Me

This website is designed specifically to allow users to choose a programming language based on what they need it for, a similar task to that of this project. Best Programming Language for Me has a simple interface and allows users to reach some language after answering a series of questions, getting closer to a specific answer after each question. See table 3 (section 3.4) for the advantages and disadvantages of Best Programming Language for Me.

3.4 Advantages and Disadvantages

Codecademy:

Advantages	**Disadvantages**
Lots of Detail; Codecademy provides lots of detailed information about each of the languages it has on its system, and it provides a range of tutorials or lessons that can take anything between 4 and 40 hours to complete.	**Only a small number of languages;** While this might be suitable for a tool like Codecademy, the LaTCh website aims to have information on a much larger set of languages.
User Accounts; Giving users the ability to create their own account can be useful as it allows them to keep track of what they have learned as well as increasing the level of their account as they progress.	**Lack of choosing a language by level;** Although there are lots of different levels of lessons on the website it is not always clear what areas of the website are better for beginners or experienced programmers for example. Further to this, if a user chooses to do a course in a language they already know, they can't skip the easy parts at the beginning and so this can be quite tedious to work through.
	Need an account to use the service; The fact you need to create an account to use any of the services on Codecademy can be frustrating for programmers as it is another set of login details to remember and it may put them off if they only want to use a small number of functions on the website.

Table 1 – Advantages and disadvantages of Codecademy.

Stack Overflow:

Advantages	Disadvantages
Widely used; Stack Overflow is very popular with programmers, meaning that most questions a user would ask can more than likely be answered by some other member.	**Not designed for beginners;** Beginners using the system and asking a question for the first time need to be prepared that some of the more experienced users aren't very friendly. Typically, a new user might get their questions downvoted and told that their questions don't make any sense.
No need to sign up; If a user simply wants to use this tool to view questions that have already been asked and answered, they don't need to create an account.	**No Anonymity;** On Stack Overflow a user must have an account to ask questions and their username is displayed when they ask a question. LaTCh provides an option to ask questions anonymously as it avoids embarrassment about the questions a user is asking – for example when others might see it as a simple concept.
Voting system; Users can 'vote-up' or 'vote-down' questions and answers, which helps a programmer to easily identify what the best answers are.	

Table 2 – Advantages and disadvantages of Stack Overflow.

Best Programming Language for Me:

Advantages	Disadvantages
Short number of questions lead to an answer; This is useful as it avoids the user getting bored going through a huge number of questions. Usually on this website, a user can have an answer after only 4 or 5 questions.	**Lack of tutorials on site;** To actually start learning a language a user has to follow a link to a different website.
Explanation; When one is given an answer to what language they should use, they also get a description of the language and some information of why it would be good for their situation.	**Static;** The webpages are static which means adding a new language might be more time consuming than if it were in a database for example. This is due to the fact that the database could dynamically update the structure of the website when a new language is added.
Well broken down; Getting from the start of the website to some programming language is made simple by the fact that at each stage languages are broken down into different types. For example, a breakdown at one stage might be mobile development, games development or desktop development, then if you selected mobile development, you would have the choice between iOS, Android or Windows.	**Only one feature;** The questionnaire is the only function of the website. It is fit for purpose but would be more interesting if it had added on features such as a discussion board.

Table 3 – Advantages and disadvantages of Best Programming Language for Me.

In summary, we have discussed three online resources that are used by programmers and due to what has been mentioned above a number of similar features have been added to the LaTCh website, for example:

- Detailed information pages about programming languages.
- A questionnaire function which tells users the best programming language to use for their task.
- An interface that is welcoming for both beginners and experienced programmers.
- A dynamic website that can be easily updated to accommodate new languages.

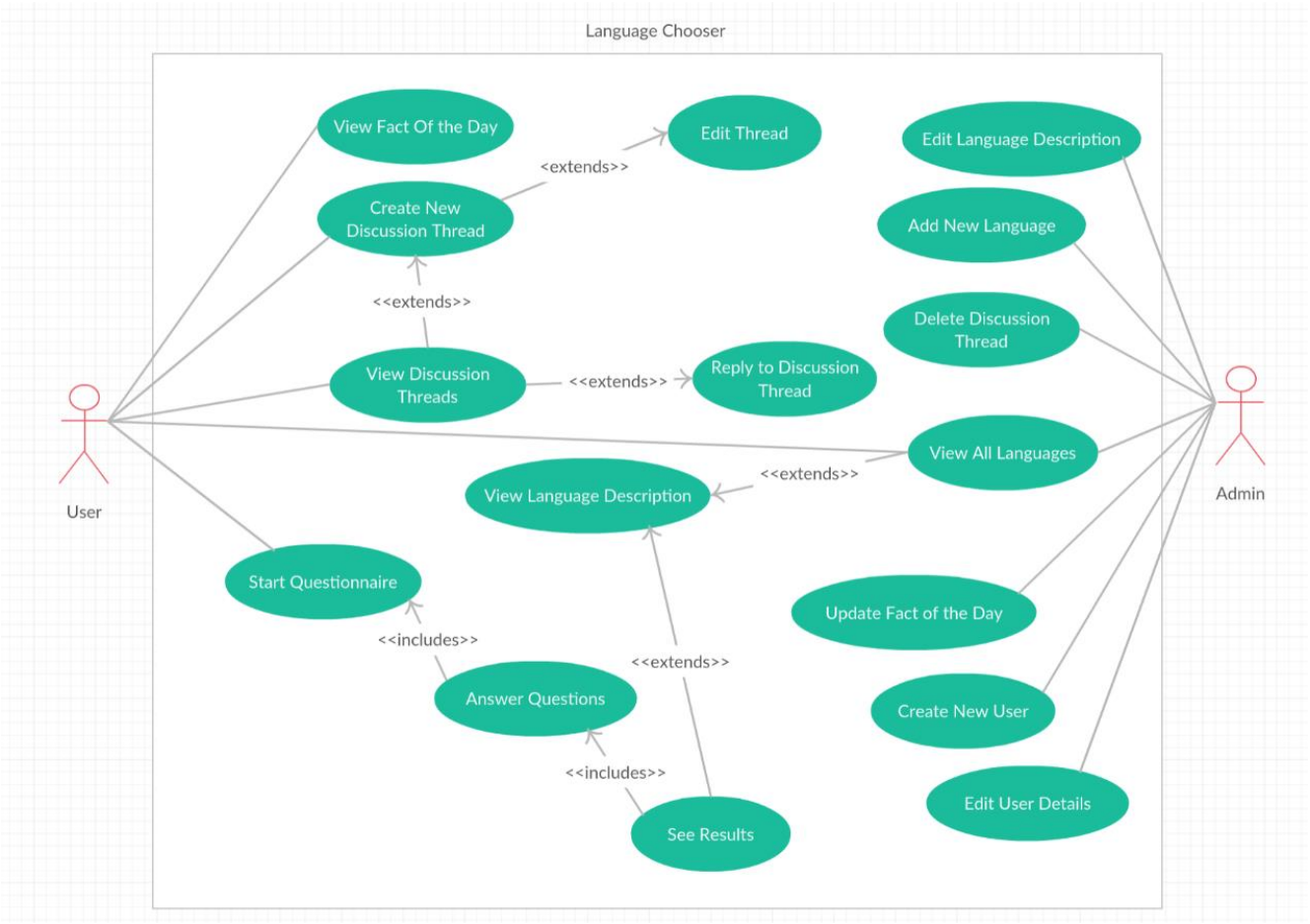
- Discussion board where registered users can discuss problems they are having and answer each other's questions, with the option to ask anonymously to avoid embarrassment or abuse.
- The option to create a user account, which can be used to post new threads and replies on the discussion board.
- A display of the average salary for each programming language.
- A 'beginner-friendliness' rating between 1 and 10 for each language which helps to direct beginners towards languages that they should learn.

LaTCh 'Language and Technology Chooser' aims to take the best features from these websites and make them better. For example, the questionnaire is similar to the Best Programming Language for Me questionnaire, but it gets you to an answer quicker and covers low-level programming as well as high-level programming. In addition, LaTCh includes more functionality than a simple questionnaire, with a categories section and discussion board also included. The discussion board is inspired by Stack Overflow and while it may not be as advanced as the professionally produced website, it is more beginner friendly, with the option to post threads anonymously and with the encouragement for a friendly community. It is aimed for LaTCh to one day in the future have interactive tutorials just like Codecademy although this was not possible during the time of this project.

4 Work Done as Part of the Project

4.1 Requirements

4.1.1 Use Case Diagram



The use case diagram defined in the early stages of this project proposed that there would be two actors – a ‘User’ and an ‘Admin’. Originally, the plan was to only have log in accounts for the Admin, however this was extended and now both have the option to log in.

Users can browse the website and use most of the features without logging in, but if they want to carry out certain functionalities such as posting a discussion thread, they will need to be logged in. This allows for administrative staff to monitor the posts more easily and reduce the risk of ‘spam’. Appendix 2 shows a full explanation of each use cases shown above.

4.1.2 Requirements List

These are the requirements defined for the project, with an added column to indicate if it was achieved (✓) or not (✕).

Requirements were classified using MoSCow analysis, where each requirement falls into one of 4 categories:

1. **M - Must have;** The final product must fulfil this requirement to be complete.
2. **S - Should have;** The final product should fulfil these requirements but does not necessarily have to in order to be successful.
3. **C - Could have;** This category covers requirements that are extra, optional requirements that could have been implemented if time allowed it.
4. **W – Would like to have;** Extra requirements that would have been fulfilled if there had been more time available to complete the project.

The requirements can also be described in one of the following two ways:

1. **Functional;** Requirements that refer to the specific functions and services that are expected of the completed system.
2. **Non-Functional;** Requirements that are not concerned with specific system functionality. Instead they deal with performance or external constraints such as availability and security.

Functional requirements have the letters FU at the start of their ID whereas non-functional requirements are marked with NF.

ID	Description	MoSCoW	Achieved?
FU-01	The system shall provide an information page about every language in its database.	M	✓
FU-02	The system shall provide a questionnaire-style function, whereby a user can answer a series of short, closed-end questions to come to one or more suitable answers.	M	✓
FU-03	The system shall provide the user with informative answers after they have completed their questionnaire which tells them a bit about the languages that are being advised to them and links them to tutorial or information sites.	M	✓
FU-04	The system shall provide a 'fact of the day' for its users. For example: "The first high-level programming language was FORTRAN. It was invented in 1954."	S	✓
FU-05	The system shall allow admin users to update the 'fact of the day'.	S	✓
FU-06	The system shall allow admin users to add and remove languages from the underlying database.	S	✓
FU-07	The system shall allow admin users to edit the description for programming languages in the database.	S	✓
FU-08	The system shall allow the user to see a comprehensive list of all programming languages in the database.	M	✓
FU-09	The system shall have a discussion board.	C	✓
FU-10	The system shall allow users to create their own discussion threads.	C	✓
FU-11	The system shall allow users to view other discussion threads.	C	✓
FU-12	The system shall allow users to reply to other users' discussion threads.	C	✓
FU-13	The system shall allow admin users to delete discussion threads.	C	✓
FU-14	The system shall allow a user to delete their own discussion threads.	C	✓
FU-15	The system shall allow a user to rate a language and leave a comment.	W	×
FU-16	The system shall provide its own language-specific tutorials for users to take part in.	W	×
FU-17	The system shall allow users to vote for their favourite programming languages.	W	×
FU-18	The system shall allow users to browse programming languages by popularity.	W	✓
NF-01	The system shall be constantly online and available (24/7).	M	✓
NF-02	The system shall run correctly on all the main Web Browsers (Chrome, Microsoft Edge, Firefox, Safari).	C	✓
NF-03	The system shall be usable and will adjust itself to work on all platforms including desktop, mobile and tablet.	C	✓
NF-04	The system shall be visually pleasing for the user.	M	✓
NF-05	The system shall be secure to access.	C	✓
NF-06	The system shall use https.	W	×
NF-07	The system shall be described as 'Secure' in a Google Chrome web browser, which means it passes all data in forms securely.	W	×
NF-08	The system shall be extendable i.e. it should be designed in such a way that new features can be added later.	M	✓
NF-09	The system shall be able to deal with multiple concurrent users.	M	✓

4.1.3 Discussion

Here, some of the requirements that were met are discussed as well as an explanation of why certain requirements were not met on the completion of this project. All of the 'Must have' 'Should have' and 'Could have' requirements have been achieved and all those not achieved are low priority 'Would have' requirements.

Requirement **FU-04** mentions the 'Fact of the Day' feature. This has been created and uses a PHP algorithm to update every day at midnight. When the page has loaded, the algorithm counts how many days have passed since the 31st December 2017. This number is then taken as the ID number for the fact in the database, which is then loaded in. The ID is taken 'modulo' the number of facts, so that after the highest ID number has been passed, it will return to ID=1 the next time. This feature was created to make the homepage more interesting and give the fact it appears as if it is being typed gives the page a bit of 'life'. The algorithm for this feature is shown in figure 7 (I have added more comments for explanation):

The 'fact of the day' is printed using 'echo' to an invisible area on screen, then printed out in a 'typewriter' style using JavaScript.

This feature, as well as many others including the login function required that I learned a lot about PHP which was challenging but also very interesting and rewarding at the same time (see chapter 5.6).

FU-15, **FU-16**, and **FU-17** are all features that ended up being beyond the scope of this project as they were low priority and there was little time to implement them. However, these are features that would definitely be added to a future version of the LaTCh website.

NF-04 says that the LaTCh user interface is visually pleasing for the user. This has received a 'tick' (✓) based on survey results where most users rated the interface well (discussed further in chapter 5.4 'Full Usability Study'). **NF-05** has received a 'tick' due to the fact that user information is stored securely on the MACS university servers and user passwords are encrypted using built in PHP hashing functions. However, **NF-06** and **NF-07**

receive a cross (×) because in order to mark a website as 'secure' on Google Chrome, it must have a security certificate and must use HTTPS Hypertext Transfer Protocol instead of the unencrypted version of HTTP. This is something I have no experience with and something that I could set up with more time.

```
<?php
     include "noOfFacts.php"; // counts the number of facts in the database

     $first_date=strtotime("December 31");
     $current_date = time(); // current date
     /* How many days have passed between $first_date and now? */
     $day = ceil(($first_date-$current_date)/60/60/24);

     $modulo_value = ($day%$no_of_facts) + 1; // 'plus one' to avoid 0 value
     $sql = "SELECT id, fact FROM facts WHERE id=" . ($modulo_value);
     $result = $conn->query($sql);

     if ($result->num_rows > 0) {
         // output data of each row
         while($row = $result->fetch_assoc()) {
              $fact = $row["fact"];
         }
         } else {
              echo "\nNo available fact for today...";
         }
         $conn->close();
         /* Make sure the characters are valid HTML. */
         echo htmlspecialchars($fact);

?>
```

Figure 7 – The 'fact of the day' PHP source code.

4.2 Project Planning

The Gantt chart below (figure 8) shows the main tasks of the project and the tasks they were planned to be completed by. This was not stuck to precisely due to other priorities interfering and some tasks taking longer or shorter amounts of time than estimated. However, this was always part of the plan, and the chart was designed to be as flexible as possible and used mainly as a reference. The most important thing to do was stick to the critical path and make sure the relevant tasks were completed before the main deadlines, shown by the two vertical lines (27th November 2017 for deliverable 1 and 23rd April 2018 for deliverable 2). The specific dates of each task are shown in appendix 4.

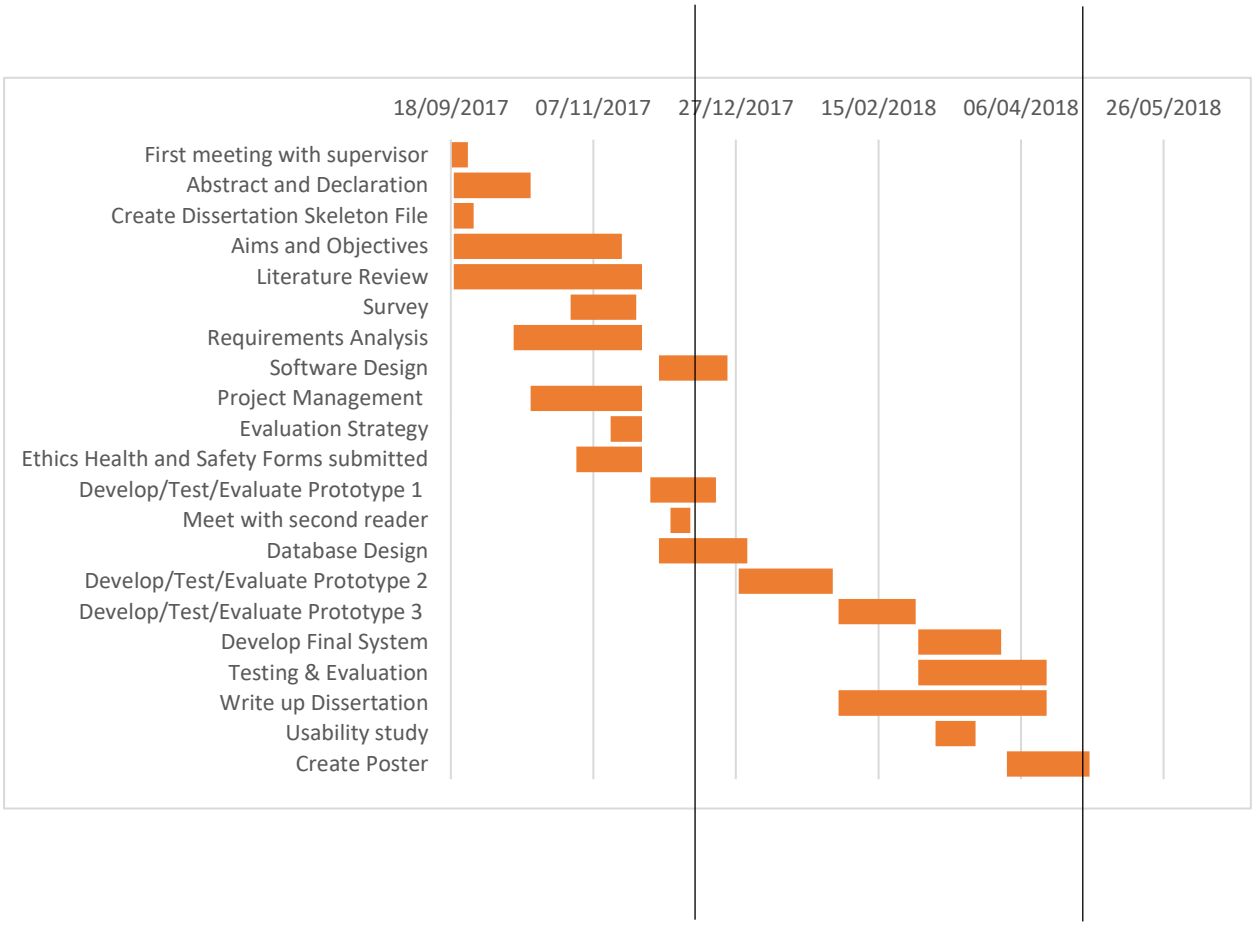


Figure 8 – Original project plan in the form of a Gantt chart.

4.3 Risk Analysis

Risk analysis is an extremely important part of most large software projects. A number of risks to this project were identified early on and these are summarised in the table in appendix 11. Each risk in the table has a description, to explain in more detail what it is, and an impact which describes if its impact on the project will be tolerable; a minor inconvenience, severe; problematic to the project but not causing it to fail, or catastrophic; which would cause the project to fail completely if it occurred. Each item in the table also has a likelihood; how likely it is to occur (high, medium or low) and a mitigation strategy which explains how I attempted to avoid running into this risk. It was important to keep the likelihood of high impact risks as low as possible.

4.4 Software Design

This project was undertaken with modern approaches to software development in mind. It was important to get an early working version of the website, and so the slow, traditional 'waterfall process' of software development was avoided. Software design was not as high a priority as it may have been in more traditional projects but it was still important to get some design ideas drawn up before getting started on the development. The diagrams shown in the subsequent sections were originally drawn using pen and paper and have been iteratively updated as the project has continued.

4.4.1 Mock-ups

The first part of the software design process for this project involved drawing up mock-ups, which would represent what the end-user interface (EUI) should look like. Since they were only mock-ups, they did not have to be stuck to strictly, but they were useful as a starting point.

It was important for the website to have a consistent structure all the way through, with navigation bars at the top and bottom of the pages. Large buttons would also be used, and boxes – which would later become drop-down boxes – would be used for listed items like threads in the discussion board and languages on the 'Languages' page.

The colour scheme is not shown in the mock-ups. This was to be simplistic and not overpowering, which is why black and grey are mainly used. The addition of the blue colour on button clicks and menu hovers was used to subtly add hints of colour and make the interface more interesting. The same colour of blue has been used throughout the website.

The questionnaire function is more of a standalone, separate function from the website itself, which is why it has a darker colour scheme. This is the function that changed the most, with its first design being webpages with buttons. It then became an interactive tree, then an interactive table. To see all the mock-ups and final EUI, go to appendix 9.

4.4.2 ER (Entity Relationship) Diagram

To make LaTCh more dynamic and changeable, there needed to be a database in the back end that could be updated by users. I chose to use a relational MySQL database, and below is the corresponding ER diagram (figure 9).

The database has been designed with extendibility in mind, for example the 'dBoard' table allows for multiple discussion boards to be created – only minor modifications would have to be made to the web application on top. Users are kept separate from admin users to allow them to have separate logon screens and to improve security, and there is a feedback table to allow users to give feedback on the LaTCh interface or functionality if they wish.

The cardinality of the relationships between the tables is as follows: one user can create zero or many threads, one language can have one main paradigm, one discussion board contains zero to many threads and one thread can have zero to many replies.

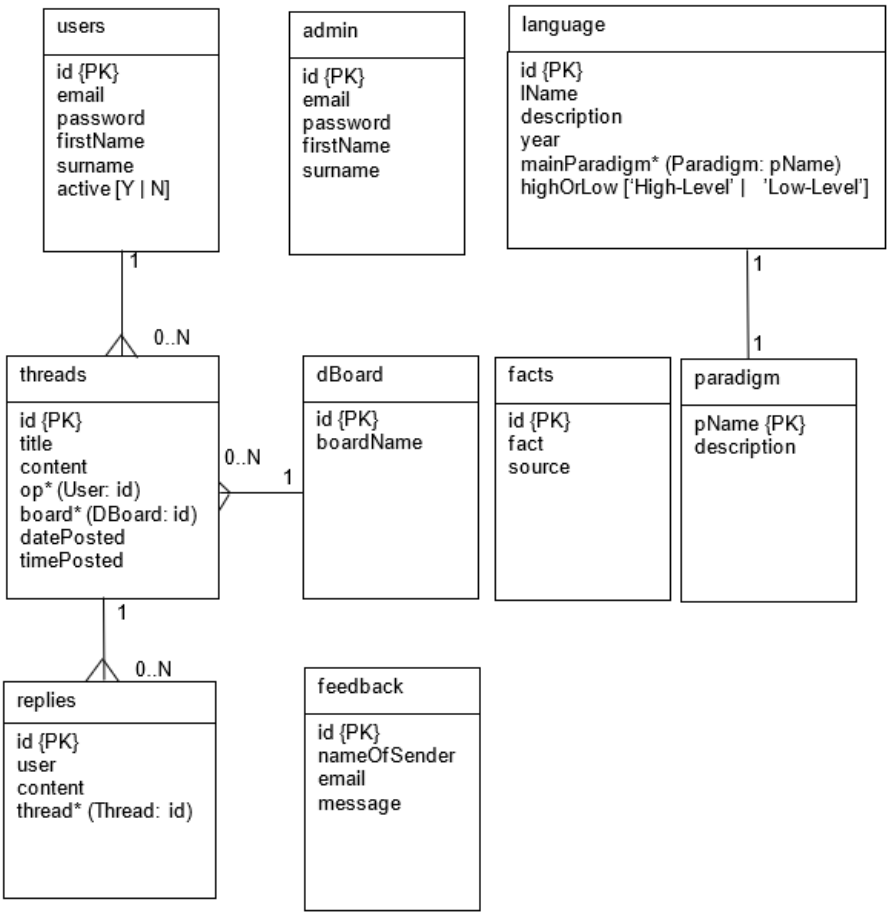


Figure 9 – ER diagram representing the underlying MySQL database.

4.4.3 Activity Diagram – Creating a Thread

The diagram below (figure 10) shows the activities a user must carry out to create a thread in the LaTCh discussion board. Each box shows an activity and each arrow shows the direction of flow between the activities. Activities correspond to web pages on the LaTCh website and each arrow represents hyperlink connections between pages. There are two sections (also called 'swim lanes') - 'Discussion Board' and 'LaTCh'. Activities are placed inside the section they belong to, with 'Discussion Board' being activities that belong specifically to that area of the website, and 'LaTCh' being activities outside the discussion board, but still on the website itself. Values inside [square brackets] are the options selected by the user at each point to flow in that direction. Arrows without a label in square brackets are carried out automatically when the previous activity is completed.

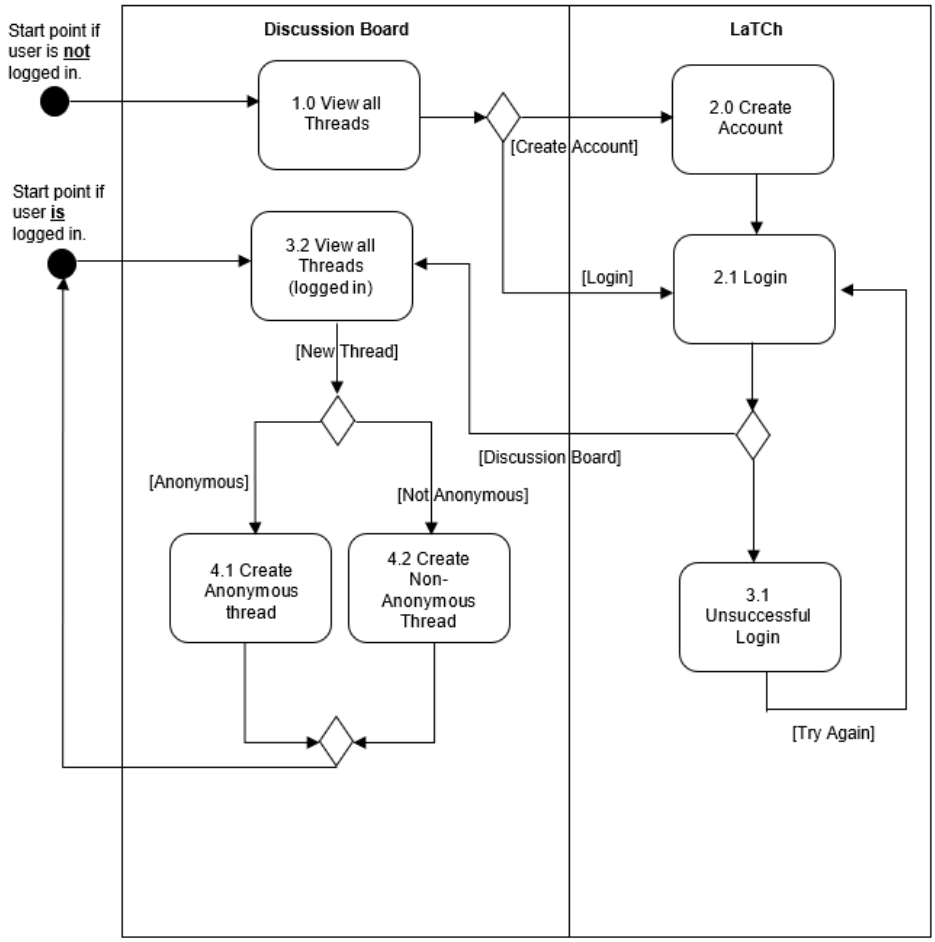


Figure 10 – Activity diagram representing a user creating a thread.

4.4.4 State-Machine Diagram – Discussion Board

The state-machine diagram shown below (figure 11) describes the differences in the functionality of the discussion board for guest users on the website, and users who have an account and who are logged in. The login screen is separate from the discussion board itself.

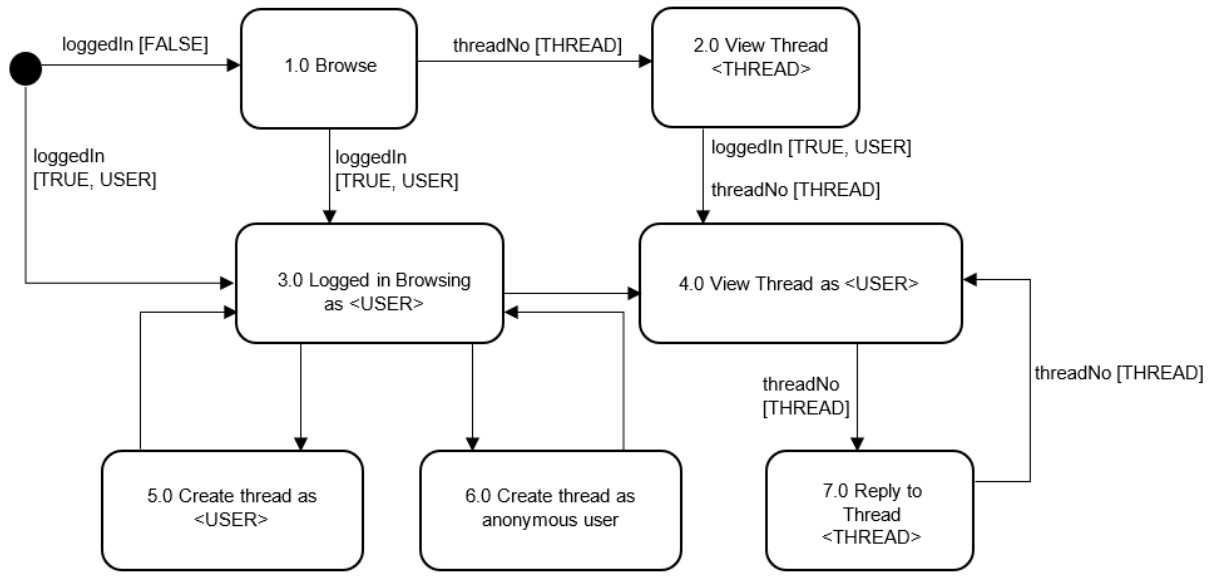


Figure 11 – State machine diagram which shows the functionality of the discussion board.

Each box shows a different **state**, the arrows between them show the state change that would have to take place to get to that state and the variables at each state change are also shown. Arrows without a label are traversed whenever the preceding state completes, whereas arrows labelled with variables can only be traversed when that variable exists in the state described. For example, an arrow with the label loggedIn [TRUE, USER] can only be traversed when a user **is logged in** to the account **USER**.

The discussion board has been designed to be accessible to all users. Guest users can browse the board and see all threads available on it at that time. They can also view a specific thread, including all replies that have been sent to it.

A user who is logged in can do all of those things too, but also has options to reply to threads and create their own ones. Users can write threads anonymously, or with their user number attached. The ● symbol shows the point where a user would begin.

4.4.5 'LaTCh' – Website name and logo

Part of the feedback received from the prototype 1 focus group (see chapter 5.1) was that the website needed a more interesting and catchy name, an alternative to the original name - 'Language Chooser'. The answer to this was 'LaTCh – Language and Technology Chooser', which gives you a catchy word as a name and an explanation of what the website does all in one. LaTCh allows you to choose a language (C, Java, Python etc.) or another technology (frameworks like Flask, development environments such as MatLab, etc.) to help you solve a problem. Below is the final collection of logos to be displayed on the website from prototype 3 onwards:

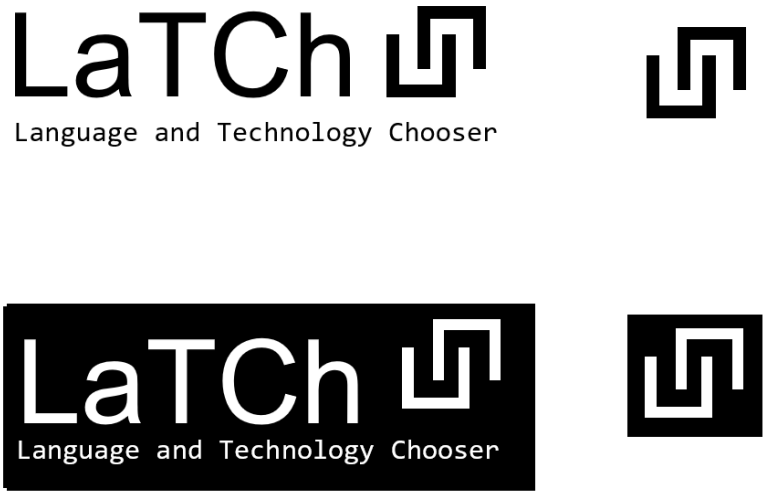


Figure 12 – LaTCh logos

The main purpose of having a catchy name and logo was to make the website more appealing to users and to hopefully help them remember it better – encouraging them to come back.

4.4.6 LaTCh Site Map

The site map below describes the structure of the website, including all the individual HTML/PHP pages. A table representation of this information, with the name of each webpage included is also shown in appendix 8.

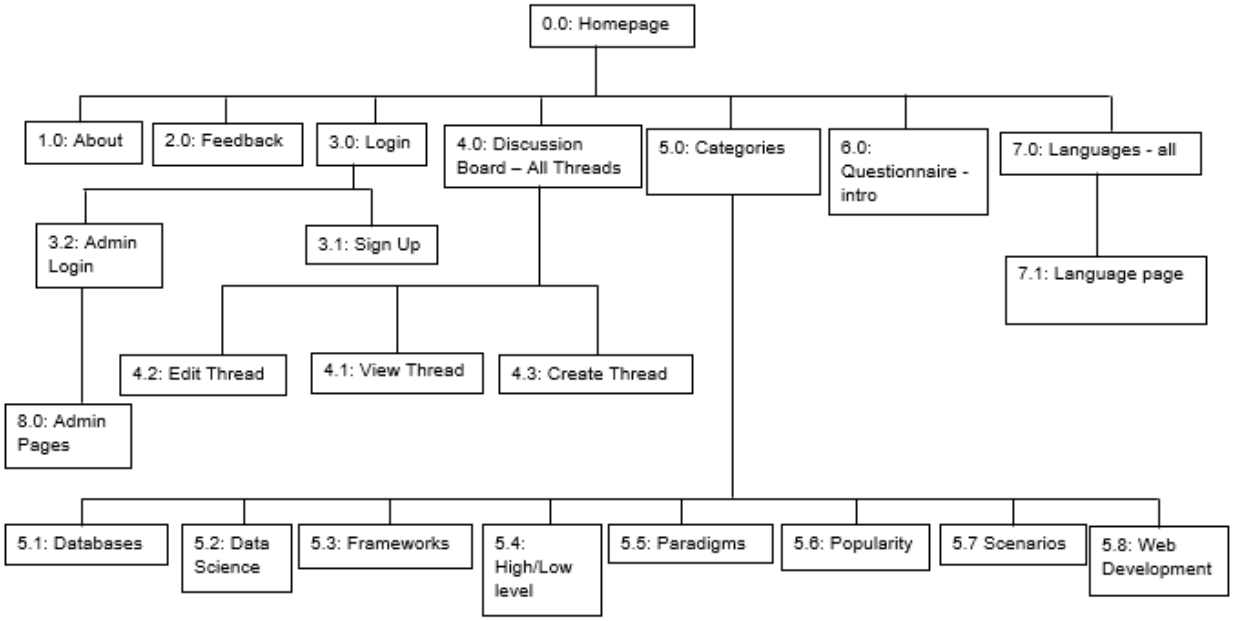


Figure 13 – Site Map. Describes the structure of the LaTCh website.

4.4.7 Questionnaire Decision Tree

One of the main features of this project is the questionnaire, which allows a user to find a programming language to suit their needs by answering no more than 3 questions. It does not answer a user's question or problem directly, but instead asks the user a few general questions to narrow down towards an answer.

The questionnaire has been designed based on a manually created decision. The tree was to be as shallow as possible, meaning users could get to an answer as quickly as possible.

There are 26 possible routes to an answer in this tree, with 18 different languages possible for an answer (for simplicity, I am including frameworks and database management systems as 'languages' in this context). The final tree is shown below:

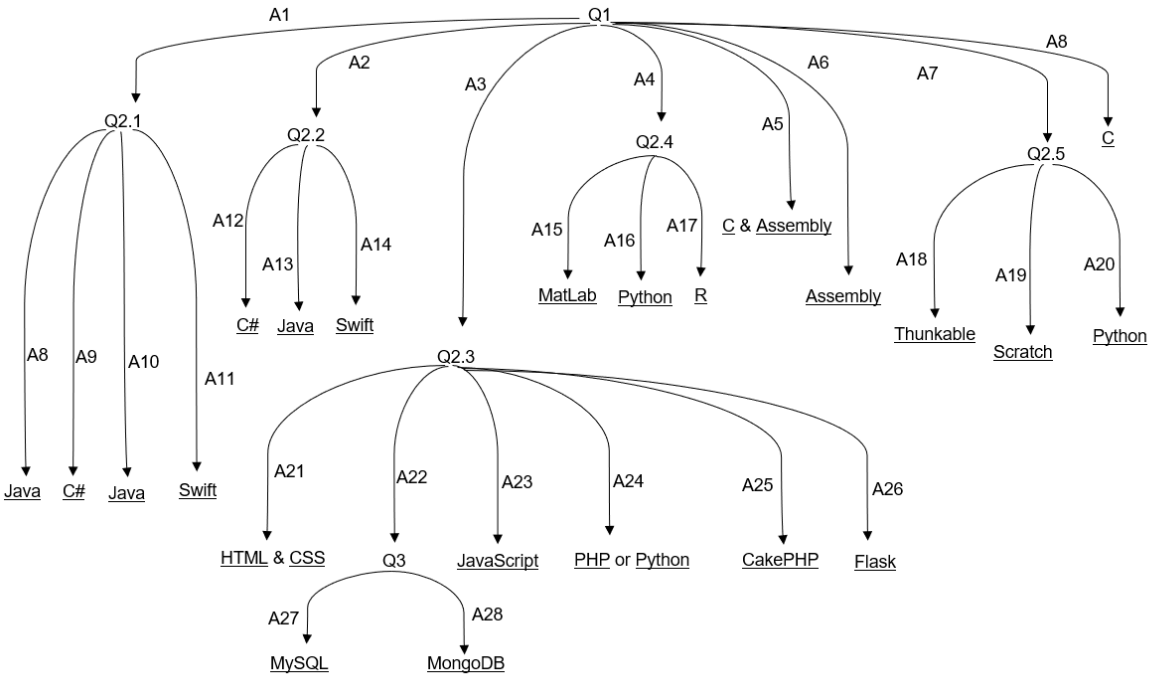


Figure 14 – Decision Tree which the questionnaire is currently based on.

Decision trees are commonly used in AI (Artificial Intelligence) decision making systems. This one has been used for this project to allow it to be extended in the future. With the foundations set just now, there is the possibility for an automated AI system to be set up that could be used to update the underlying decision tree and also the questionnaire intelligently, for example when a new programming language is added. See appendix 6 for the corresponding table which shows the questions Q1...Q3, and possible answers A1...A26.

4.5 Technologies used

This project required the use of a large number of different technologies, including programming languages like HTML5 [62], CSS3 and JavaScript [67] at the front end, and PHP [68] and SQL [64] (MySQL database management system) at the back end. For planning and version control, GitHub [34] and Trello [35] were used, and XAMPP [36] and PHPMyAdmin [37] were used to run a local version of the site – these were important tools, which allowed for a private development version of the website to be available as well as an online public version that was constantly up and running.

Throughout the project, a number of surveys were carried out in the form of questionnaires and usability studies, created using Survey Planet [19]. These surveys could be distributed/shared quickly through social media, making use of Facebook [39], LinkedIn [40] and Twitter [41] over the course of the project. This dissertation document was created using Microsoft Word [38]. Below (figure 15), we can see the system architecture of the LaTCh system, and some of the software that was in use during the development:

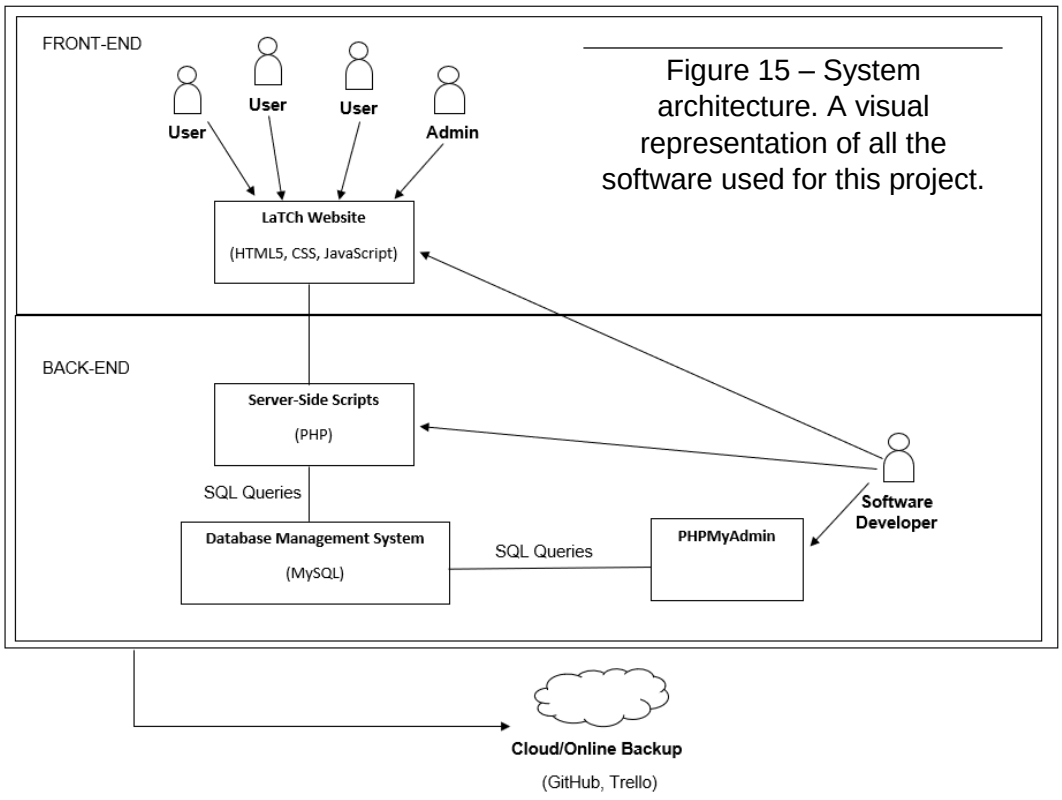


Figure 15 – System architecture. A visual representation of all the software used for this project.

It can be seen from the illustration that users (including admins) only have access to the website itself and this is their interface for interacting with the whole system, whereas the software developer has access to all the source code and every area of the LaTCh system.

4.6 Problems Encountered

This project has not been overly problematic, but there have been some challenges to overcome. One of the main issues was lack of experience. I had not used many of the technologies (such as PHP, JavaScript and MySQL) together on such a large scale, and I had never worked on an individual project of this magnitude before. I felt that these issues could be overcome with correct planning and strong organisation throughout the project.

Furthermore, there were different problems with the website itself as time went on. I questioned how I would layout the questionnaire throughout the project and had trouble deciding, with it first of all being static web pages with links between each question, then an interactive D3 JavaScript tree, then an interactive table. I had problems in deciding where the 'league table' system would fit in that I discussed in deliverable 1 and decided that this was not a necessary feature in the end.

Security was also a possible issue to begin with. I have been learning more and more about security on websites as this project has went on and have learned password hashing in PHP as well as how to protect against SQL and JavaScript injection.

Further to these issues was weather conditions and staff strikes. The closing of the university for two days due to heavy snow meant I lost two days worth of access to the university building. And staff strikes caused disruption to the normal university schedule.

4.7 Prototypes and Final System

For the development of the LaTCh website prototyping, a process used in agile software development [25], was used. This meant there would be a working version early in the project, allowing for testing to be constant and continuous throughout the development. It also meant lots of feedback could be received from potential users of the final system throughout the development process. See chapter 5 for all details of the prototype testing.

4.7.1 Prototype 1

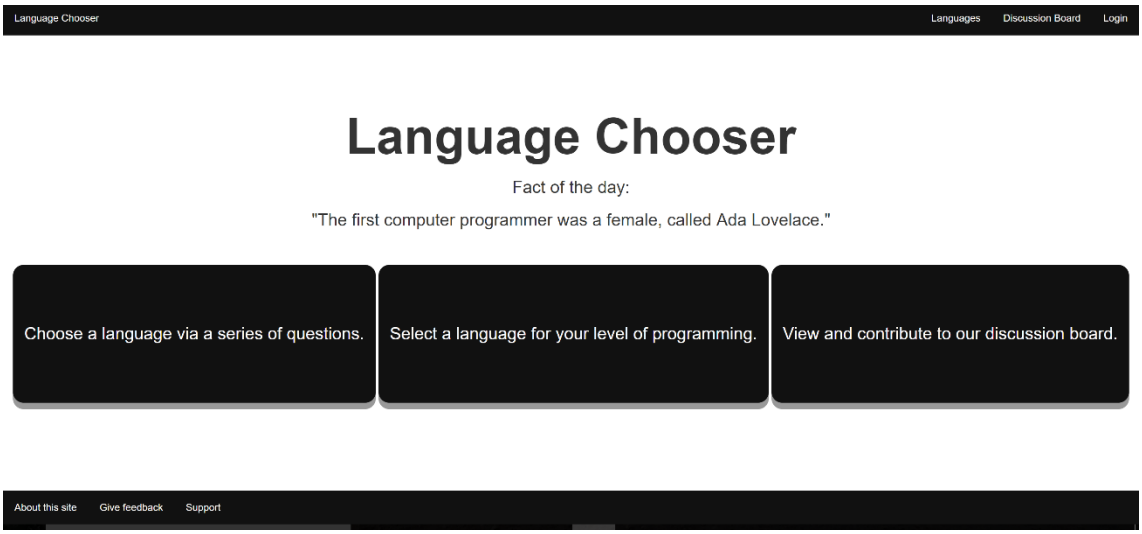


Figure 16 – The landing page on prototype 1.

The first prototype [23] was developed in the early stages of deliverable 2 – December 2017. The idea of this this prototype was to get an early working version of the website, with static web pages (no database connection) and the first version of the user interface. Working this way meant significant time did not have to be spent developing mock-ups and drawing up the design, but instead the website could actually be created early and then lots of feedback could be received from users trying out the real website (discussed in chapter 5.2).

Prototype 1 had limited features, including an early version of the questionnaire function. This function would allow users to answer a small number of questions to get to a resulting programming language and would later be designed based on a decision tree to make it as efficient as possible. The prototype also had pages for the login screen and discussion board, although these were not currently functional, and a 'Languages' page, which gave an overview of all languages on the website. Another feature was the 'Choose a language by level' option which gave you the ability to look at languages suitable for beginners, intermediate and expert programmers. This would be restructured to a 'Categories' section in prototype 3, and the 'beginner friendliness' of each language would be discussed in the overview page for that language. This would give the ability to rate languages on a scale of 1 to 10 for how beginner friendly they are rather than having only 3 distinct options (beginner, intermediate, expert) to choose from, and it gives beginners a good idea of what languages to work with. This prototype also displayed the 'Fact of the Day', but this was not updating automatically, and was instead being controlled manually in the background in a 'Wizard of Oz' fashion [26] – the function looked like it was working at the front end but was being controlled manually in the background.

This version of the website was up and running fairly quickly as it only required knowledge of HTML and CSS, which I did have experience with through high school and university.

More screenshots of the original interface are available in appendix 5. See chapter 5.2 to find out about the testing of prototype 1.

4.7.2 Prototype 2

The second prototype [24] had a number of changes from the original, which are mentioned in chapter 5.2. The main change was that the website would now connect to the MySQL database (described in chapter 4.4.2) through PHP, and the information on many of the pages could no be updated dynamically. The fact of the day was now being pulled from the database, with an algorithm calculating the number of days since 31st December being used.

This number would be used to select the id number of a fact – the fact of the day. See appendix 7 for screenshots of the interface.

The programming languages were now also being pulled from the database and listed on one large page. Having this database connection also allowed for the first version of the discussion board to be created, however, user login functionality would not be created until prototype 3, so all threads would currently be anonymous.

This is where the second version of the questionnaire was created, and a template was used from a JavaScript library called D3 to model it as an interactive tree [27]. This version brought many problems as I didn't have experience with the framework and not all browsers would run it properly. It would be changed to a HTML/CSS interactive table instead for the third prototype.

4.7.3 Prototype 3

As was the case with the previous prototype, this one was created with feedback from the previous testing being taken into account. Prototype 3 [44] was where most of the development work took place and involved improvements being made from the user interface at the front end all the way to the database at the back end. Chapter 5.3 discusses changes I made based on feedback at this stage.

Another adjustment in addition to those discussed in the testing section included adding a splash of colour to the home screen. This was done by adding a blue box around the fact of the day, which also now appears as if it is being typed, immediately drawing the users eye. The blue box subtly adds colour to the page, making it a more interesting interface without

interrupting the dark theme. The same shade of blue is consistently seen throughout the website.

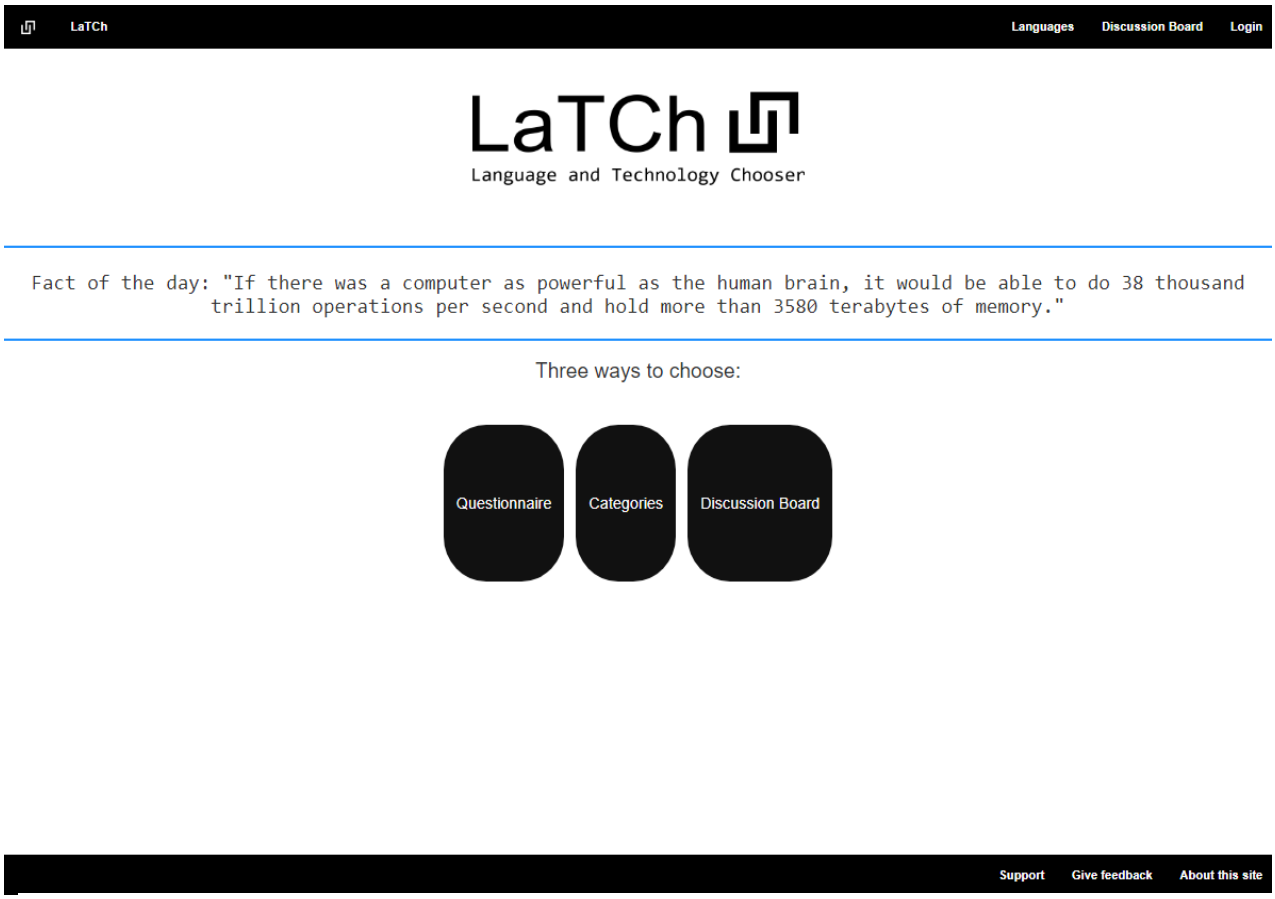


Figure 16 – The landing page on prototype 3.

The buttons on the homepage were reworded so the user would be able to understand their functionality more easily, and I added a drop-down box feature to listed items in the database. This can be seen on the languages page, discussion board, and inside some of the categories. The ‘Categories’ section was a new addition, and it was thought this would be an important feature to add as it allows a user to break down programming languages in so many different ways, hopefully appealing to as many users as possible. As well as this, overview pages have been added for every programming language, and these pages could be extended in the future. At the moment they give an overview of every language as well as information on the average salary developers get for that language.

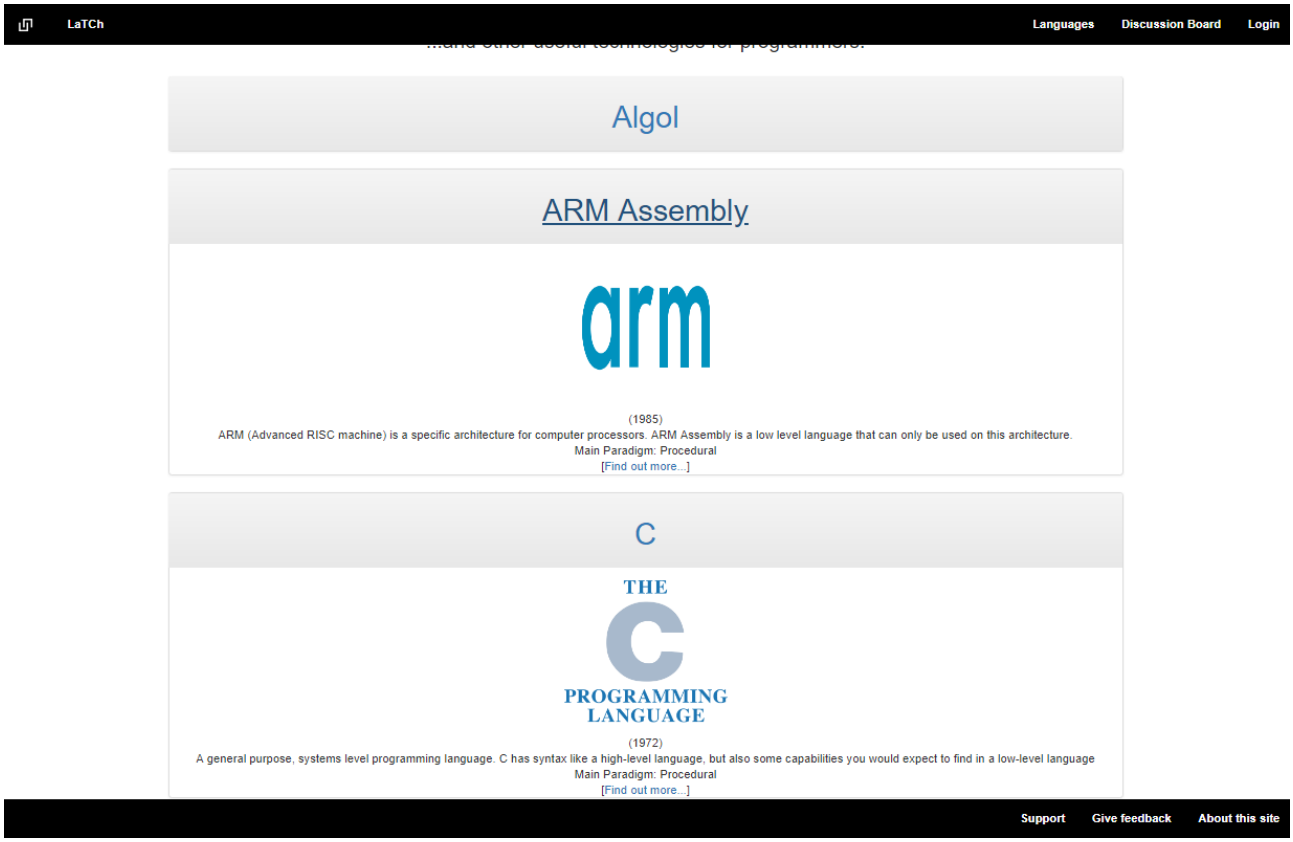


Figure 17 – Drop-down menus on prototype 3.

The ‘LaTCh’ name and logo appeared for the first time in prototype 3, as well as login functionality. There would now be two types of user – regular ‘users’ and ‘admin’ users. A regular user could login, then create their own threads and reply to threads that had already been created. They could edit and update their details, including their password, and they could make their threads anonymous if they wish. A change from prototype 2 was that a user would now need to have an account to create a thread, and this would reduce the likelihood of spam on the discussion board. An admin could now login to their own separate area of the website, which no other users have access to. This area was called ‘LaTCh – Admin’. From here they could interact with the underlying database, allowing them to add and edit user accounts, edit the fact of the day and add new facts, create other admin accounts, and add and edit programming languages.

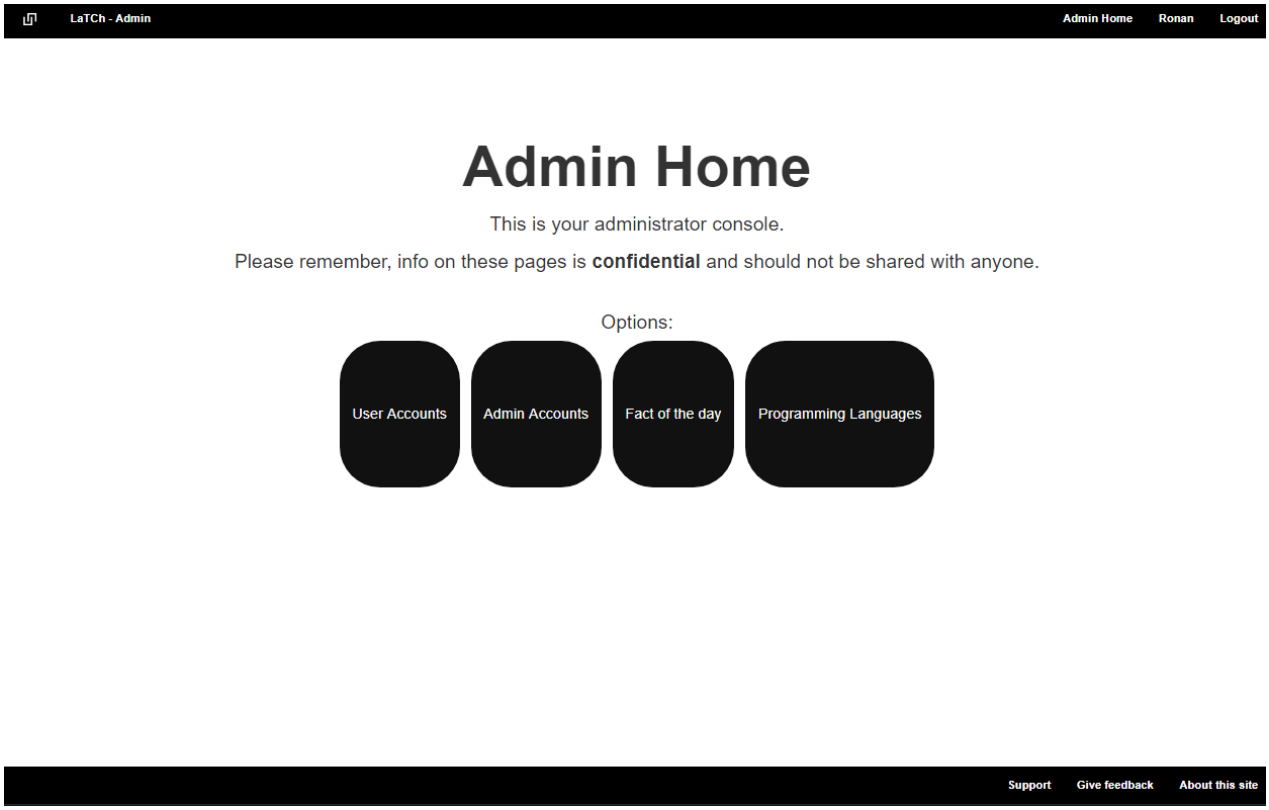


Figure 18 – ‘Admin home’ on prototype 3.

Slight adjustments had to be made to the database at this stage, including giving a language a ‘High-level’ or ‘Low-level’ value, for the categories section, and giving users an ‘active’ attribute so an admin could deactivate their account without deleting it, if necessary.

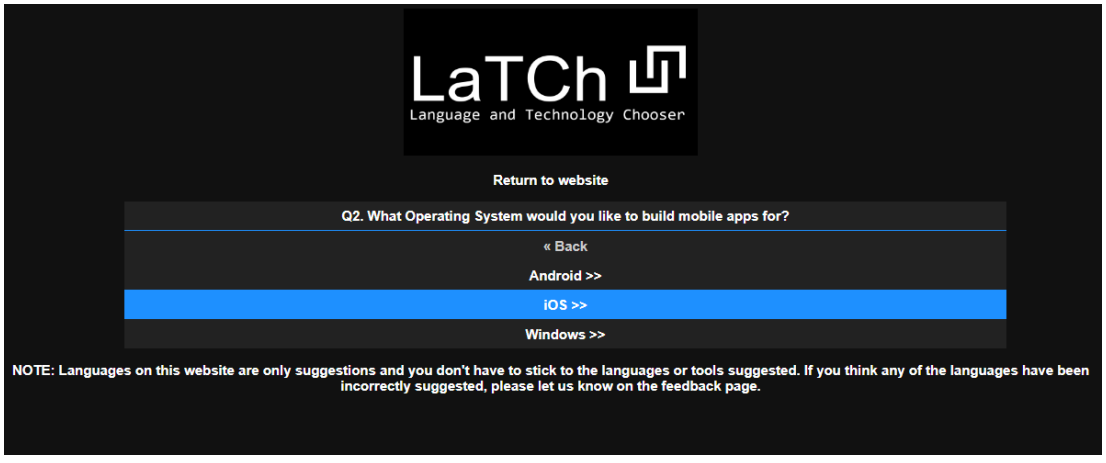


Figure 19 – The LaTCh Questionnaire.

In addition, LaTCh now had a more improved interface on mobile phones. The footer was forced to stick to the bottom of the page, and the navigation bar at the top would now form a drop-down menu. The questionnaire function was restructured and redesigned as an interactive table, 'separate' from the website itself. A user could run this version of the questionnaire without the need for JavaScript being enabled on their browser and it was also much more responsive. Chapter 4.4.7 discusses the decision tree used to structure the questionnaire function. And below, we can see some examples of the user interface on smaller screens:

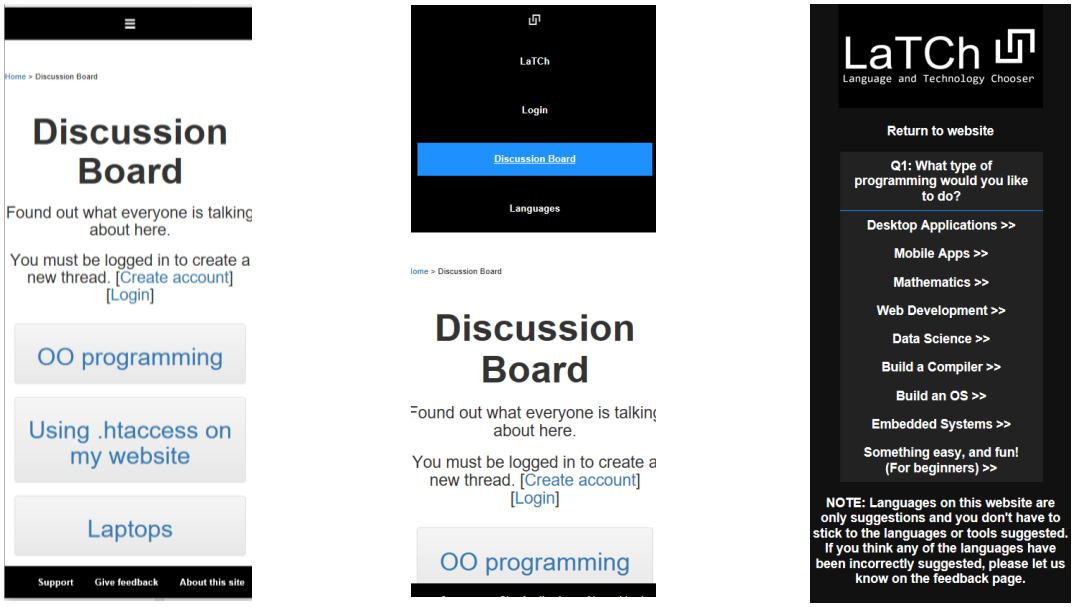


Figure 20 – The LaTCh interface on mobile screens.

4.7.4 Final System

The final version of the LaTCh website [45] has been developed based on feedback from all previous prototype stages as well as the original requirements. Since the feedback for prototype 3 was mostly good (see chapter 5.4), development of the final system mostly consisted of 'tidying up' the website, making it look nicer and more user friendly.

One of the first changes made from prototype 3 [44] was that the buttons were all made equal size. This was one of the main issues raised from the previous version of the website

as buttons would appear different sizes based on the text they were holding. The equally sized buttons appear much tidier than the previous versions and are shown below in figure 21.

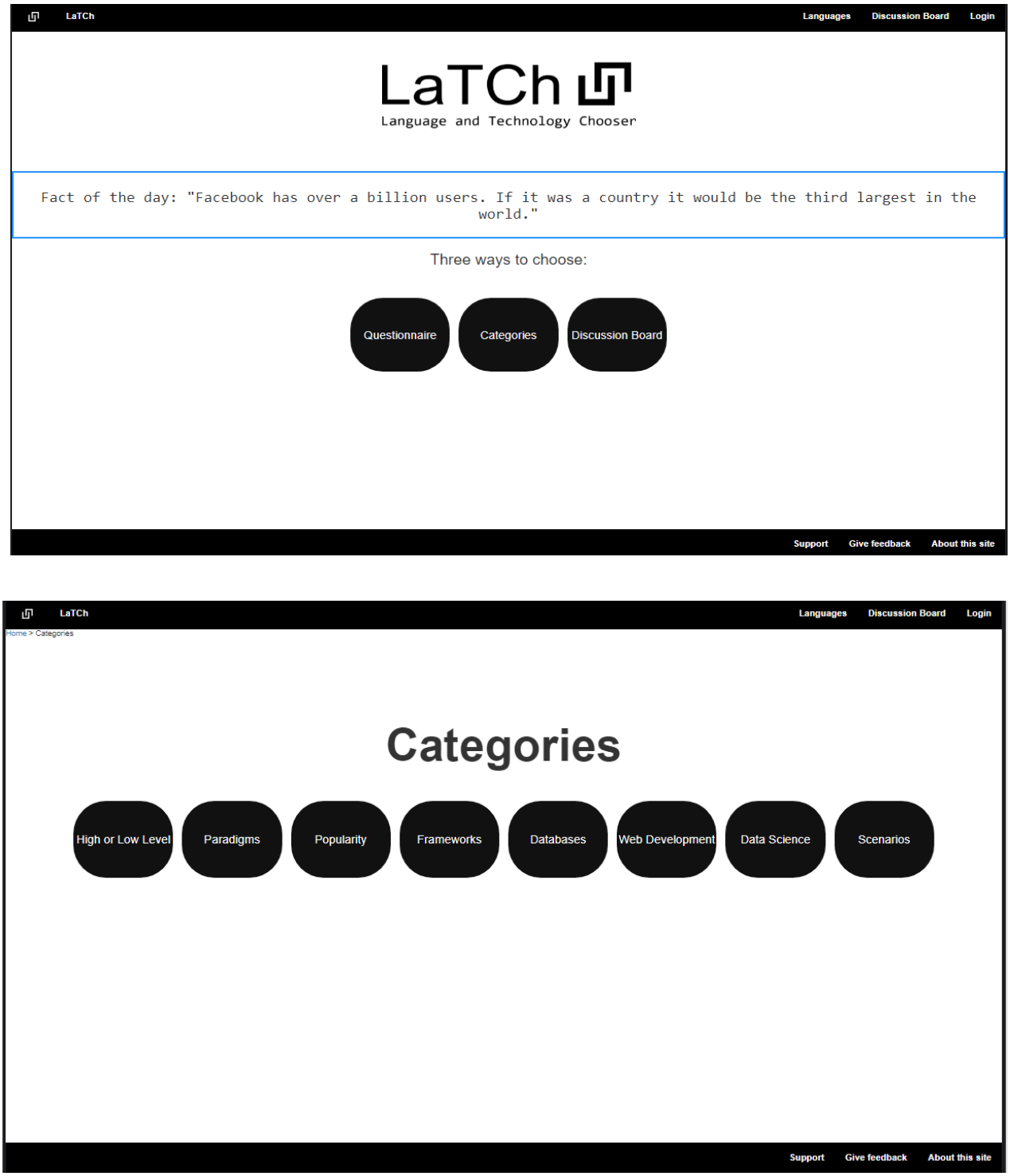


Figure 21 – Equally sized buttons on the home screen and the categories page.

Another issue raised from the usability study on prototype 3 was that there was very little automatic redirection when a user submitted a form. For example, when a user created a new account, they would be sent to a blank page with text on it saying that their form submission was a success, with a link back to the necessary page. If they entered an incorrect password when attempting to log in, they would be directed to a similarly unattractive empty page. This was unappealing for users and very unprofessional in terms of website design. This has now been amended using PHP '$_SESSION' variables so that whenever a user submits a form, they are automatically redirected to the relevant page and provided with the relevant message. This has been implemented on the discussion board for threads and replies, on the login and user creation screens, and on the admin pages for adding, editing and deleting data. An example of an unsuccessful login attempt is shown in figure 22.

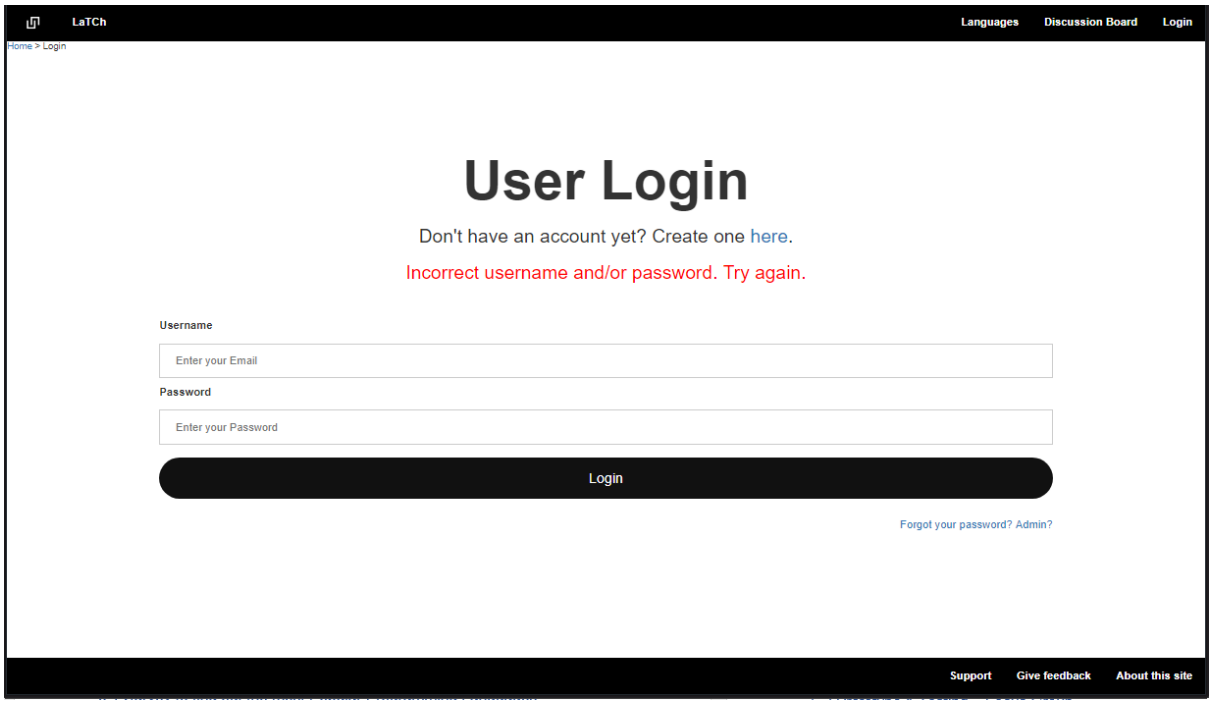


Figure 22 – Error message for unsuccessful login.

In addition to these issues raised, a small number of participants had mentioned that it is not easy to know what the buttons on the home screen actually do. To solve this problem, it has now been implemented for a small explanation to appear when a user hovers over each

of these buttons with the curser. This is a small and simple change that can make a huge difference for usability. One last attempt was made to add more colour to the website (as this was something that raised mixed opinions from test participants) by attempting to add a background. This idea was shelved, however as it did not fit with the websites dark, 'techy' colour scheme and did not adjust well on mobile screens.

The administrative abilities were also much improved. An 'admin' user can now add, edit and deactivate users and the fact of the day. This is done using a simple interface and items are sorted by their ID numbers to improve search. They can also create and delete other admin accounts, add, edit and delete programming languages, and delete user discussion threads (to prevent against 'spam' and inappropriate posts). An example of the user interface for manipulating the 'fact of the day' is shown in figure 23.

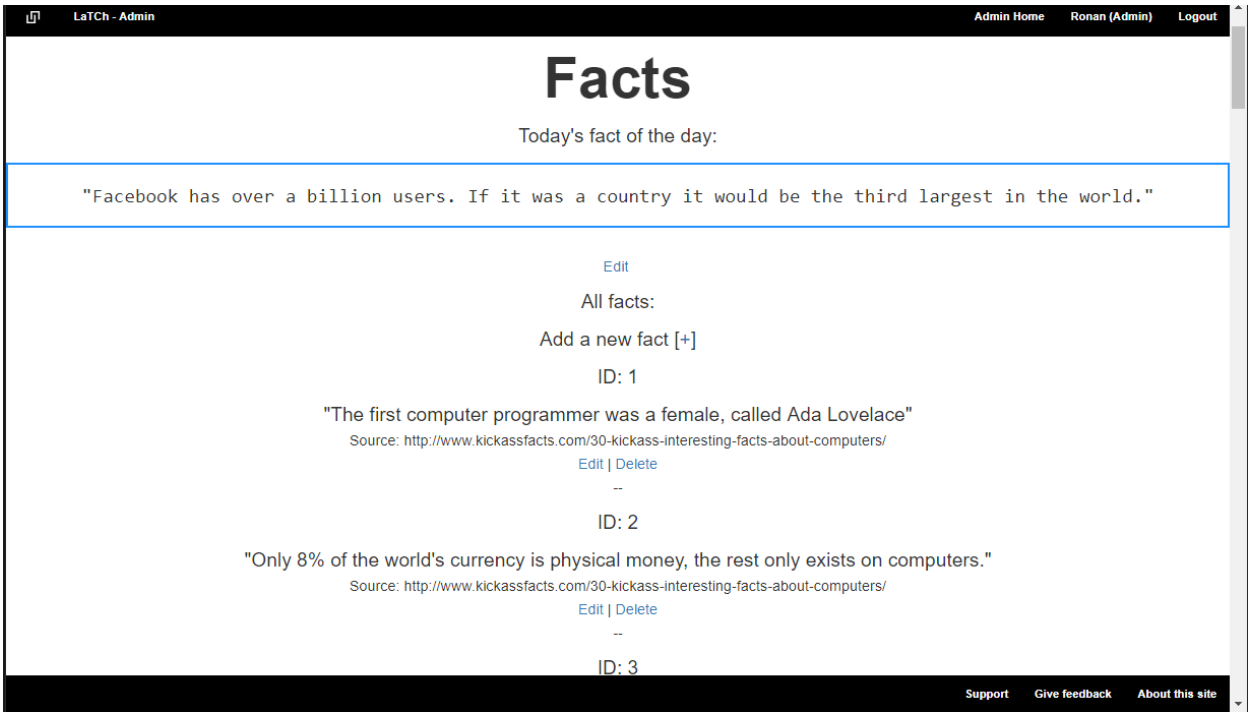


Figure 23 – 'LaTCh Admin' editing screen.

The LaTCh website is now in its final version for the purposes of this project but is ready to be extended in the future. Ideas for future development of LaTCh 'Language and Technology Chooser' are discussed in chapter 7.2 'Future Development'.

5 Testing and Performance assessment (Evaluation)

Testing was carried out before any of the implementation began. And during and after each prototype stage. The testing during each stage consisted of getting real-time feedback from potential users as different features were added, and the testing after each stage consisted of usability studies and a focus group. These are discussed in this chapter.

5.1 Survey to find out the most Popular Programming Languages

Between the 13th and 22nd November 2017 a survey was carried out to find out what the most popular programming languages are for programmers. The survey was aimed mostly at students, but also anyone with programming experience. After an initial pilot survey which I asked 4 of my colleagues to carry out, some minor changes were made to the wording of some of the questions to make them clearer and a new question was added at the end to discuss how likely programmers are to use a new programming language if given the choice. By the end, 29 people took part in the final survey, with 82.8% of them being under 25 and a mix of students between years 1 and 4.

The survey contained 4 scenario questions, each of which had 4 options of programming languages to use in that situation as well as an 'other' option for participants to give their own suggestions. This allowed me to get some concrete answers for languages as well as opinions on ones I may not have thought of. Participants were also asked to explain their answer for each scenario.

The next few questions looked for languages in terms of their type, for example object-oriented programming, server-side scripting and database manipulation. The top answers for these ones were Java, PHP, and SQL respectively, and most of these answers came down to the fact they are widely used, because the participant had more experience with them or because there is a wide range of documentation available for them.

Participants were then asked to name the first language they thought of for each paradigm and this provided a range of responses including a few languages I had never heard of before such as OCAML and F#. The final question asked how likely they would be to switch to a new language if it was better suited to their purpose and this resulted in an average rating of 3.71 out of 5, suggesting that most would be happy to switch languages.

For the full set of results on this survey see appendix 1.

5.2 Prototype 1 Testing – Focus Group

At the end of the first prototype [23], in December 2017, a focus group was carried out with 5 participants for around 1 hour, to take a look at the first version of the website and to have a browse. Since this was only an early version, participants were not given any specific tasks to carry out but they were asked to give as many opinions as they could on the user interface, the available functionality and the proposed future functionality. These were some of the comments received:

- Remove the need for a user to scroll on most pages by reducing the size of the buttons.
- Use some sort of ‘cascade’ system or breadcrumbs so the user always knows the path they have taken to get to a certain part of the website.
- Use a breadcrumbs-type system in the questionnaire as well.
- Come up with a more catchy and interesting alternative to the name ‘Language Chooser’.
- The text on the footer should be at the right or in the middle, rather than the left.
- Remove the shadow on the buttons.
- Buttons are too close together, especially on a small mobile screen.
- The simplicity of the website is good, but it could do with a bit more colour.

- The buttons shouldn't be disabled when you are on that page, for example you should be able to click on the 'about' button, even when you are on the 'about' page, and it should simply refresh the page.
- Have all questionnaire questions in one page – clicking an answer only opens a new section on the same page.
- Languages page – user interface could be more interesting. Perhaps have each language in a box? Also, there should be more information displayed about each language.
- Also, on the Languages page – instead of sorting languages by alphabetical order, they should be sorted into high-level and low-level, for example.

Most of these comments were taken into account for the development of prototype 2.

5.3 Prototype 2 Testing – Small Usability Study

Between the 7th and 19th February 2018, a small usability study was carried out to get some feedback on the progress of the website – currently called 'Language Chooser'. Since the website was still in the early stages of development, it was decided that it would only be released to a handful of individuals. The usability study had 7 participants and there was one more individual who tried the website out but did not complete the corresponding survey. 6 of the 7 survey respondents were students on the Edinburgh campus and 1 was from Dubai. I hoped to gain more feedback from Dubai students for my next usability study, which would take place after prototype 3 was complete.

This study consisted of five tasks that the participant was asked to complete on prototype 2 [24] and 5 follow up questions. They were asked to give as much feedback as possible regarding how usable the prototype was and what they did and didn't like about the user interface.

The full list of results from this usability study are shown in appendix 3.

This study was very useful in the preparation for prototype 3. Most participants seemed to like the interface, with 57% saying it was somewhat user friendly and 33% saying it was completely easy to navigate. However, the feedback received did make it clear that there were many improvements that could be made, with most problems being around the layout of listed items such as the languages and the responsiveness on mobile screens. Although the D3 version of the questionnaire (discussed in chapter 4.7.2) seemed like a good idea to begin with, feedback suggested that alternative ideas should be explored.

For prototype 3, the following changes were made to the website based on the feedback received:

- Highlighted buttons on the navigation bar to make them stand out more.
- Looked into ways to make it more responsive to small mobile screens.
- Re-structured questionnaire to make it more beginner friendly.
- Created a more detailed information page for each programming language.
- Gave a link to these pages at the end of the questionnaire.
- Added more line breaks in the HTML source code to avoid buttons getting stuck to the bottom navigation bar.
- Adjusted the interface for ‘Languages’ and the ‘Discussion Board’ with the aim of making the interface more interesting and appealing.
- Added in a feature that would automatically redirect users after submitting a form.

Feedback was also received on the security of the website and one participant even demonstrated SQL and JavaScript injection on it. This was a huge vulnerability which was removed for the next prototype by using SQL prepared statements instead of plan-text input and by sanitizing input data to remove unwanted or unexpected characters.

This is also the point where I decided on a new name for the system ‘LaTCh – Language and Technology Chooser’ (discussed in section 4.4 Software Design). For the third prototype, I would also implement user and administrative accounts.

5.4 Prototype 3 Testing – Full Usability Study

Between the 19th and 28th March 2018, a full usability study was carried out – testing the user interface and functionality of the third LaTCh prototype. The survey required participants to answer a few demographic questions before trying out tasks on the website and rating the pages on a scale of 1 to 10 for how much they liked it. All the corresponding charts can be found in appendix 10.

A total of 20 participants took place in this study, with a mixed level of ages, programming abilities and genders. Of the participants, 83.3% were between the ages of 16 and 24, 11.1% were over 45 and 5.6% were between the ages of 25 and 34. Of the testers, 35% of were either beginners or inexperienced with programming and 65% were experienced or expert programmers. In terms of genders, 75% of the participants in this study were male, with 20% being female (and 5% saying they would prefer not to say their gender). I feel that these demographics are not evenly balanced but are fairly representative of the group of users that would be expected to use the LaTCh website. Of the participants, 65% were Heriot-Watt University students, with 5% (only one person) currently working on the Dubai campus and the rest being based in Edinburgh.

All but one of the 'main' web browsers were covered in this study, with Safari, Google Chrome, Mozilla Firefox and Opera all being tested. Only Microsoft Edge was not tested, although this was tested during the website development.

As was done with the previous usability study, a pilot version was carried out before releasing the full study, however the pilot participants only raised a few minor issues which involved rewording some of the questions and this was simple to do.

In terms of the results, they were mostly good, with a complete mix of opinions and answers. All four of the webpages that were rated (between 1 and 10) had an average rating of around 7 – with the highest being the 'Questionnaire' function – which had an average rating of 7.94. The 'Discussion Board' received the lowest individual rating of 1,

which was given by a user who did not like the drop-down boxes. However, this feature still received an average rating of 7.72.

17 out of the 20 participants in this study (89.5%) said the website was either somewhat, or completely user friendly for beginners, with 68.4% of them saying it was completely user friendly. This suggests that the LaTCh website successfully completes the aim of being friendly for beginners.

As well as these statistics, there were plenty opportunity for users to give feedback in the form of opinions and this was extremely useful for the development of the final system. Most negative feedback referred to the buttons and suggested that they should be equally sized rather than variable. And as well as this there were a lot of participants unhappy with the fact that submitting forms – for example when creating a new user account – would take them to an unfriendly confirmation page. Some participants said that there could be more colour added to the website, although there were not enough participants with this opinion to make it a high-priority issue. Some users did like the dark colour scheme with the light blue and one even described it as 'cool' and 'techy'.

I feel that it would be useful to have gained more participants, although I still feel that this is a strong, representative group. Unfortunately, I did not gain enough feedback from Dubai students to make any worthy comparisons, although the one Dubai student who did participate seemed to dislike the website a lot more than most Edinburgh students. It would have been interesting to explore if this was a one off, or something that would have become a trend. Full details of this usability study are included in appendix 10.

5.5 Final system Testing – Site Comparison Survey

Between 16th and 18th April 2018, one final study was run to evaluate the LaTCh (Language and Technology Chooser) website. This was a comparison survey, which compared LaTCh to 'Best Programming Language for Me' (BPLFM), another website used to choose

programming languages and one discussed in chapter 3. There were 3 participants in this study – one beginner when it comes to computer programming and 2 experienced programmers.

Participants were asked to carry out 5 scenario tasks, where they would read a scenario and then act on the website as if they were in the scenario themselves. They were designed to be very specific so it could be explored how a user would answer specific questions on each website. Below, in figure 24, you can see how the sites compared against each other in terms of time taken.

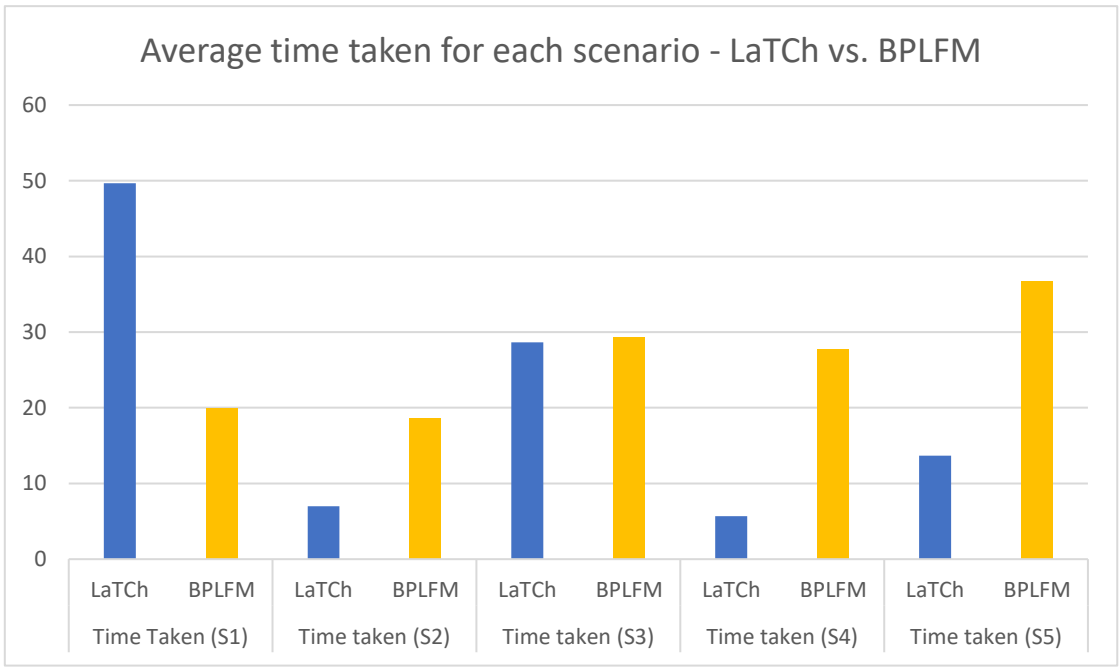


Figure 24 – Average time taken on each website for each of the 5 scenarios.

With exception to scenario 1, users completed each scenario more quickly on LaTCh than on BPLFM, highlighting that it is a better option for finding a quick answer. Furthermore, answers were found for all 5 scenarios on LaTCh, but only for scenarios 1, 2 and 3 on BPLFM. This is evidence that LaTCh covers a much wider area of programming, including low level programming. BPLFM mostly focuses on mobile and web development. In addition to this, participants were asked to give verbal feedback. Two of the three participants felt that they could easily get ‘lost’ in the BPLFM questionnaire when they took a wrong path but did

not highlight this issue with LaTCh. Participant 1, who was a beginner, did not like that there is jargon used in the LaTCh questionnaire without explanation and preferred the BPLFM approach where they back up any jargon with images. The same user felt that there was more variety on LaTCh and felt that they were always getting the same answers on BPLFM. All three participants felt that the first question on BPLFM was a bit confusing and made little sense. They all said they preferred the first question on the LaTCh questionnaire, despite the fact it has more options.

Participants all found answers on each of the websites when there was one, which suggests both BPLFM and LaTCh are fit for purpose in the context of the scenarios. They were asked to say which was their favourite for each scenario, which LaTCh received most votes for. The only scenario which a user marked BPLFM down as better was scenario 1 for participant 1. Participant 3 mentioned that the BPLFM questionnaire was much more complicated for some simple answers.

Finally, participants were provided with three statements and were asked to rate on a scale of 1-5 whether they completely disagreed with it (1) or completely agreed (5). Here are the statements and a summary of the responses:

- *Allowing users to post queries online anonymously allows them to ask questions more freely, with less chance of bullying or abuse*: Participant 1 completely agreed with this statement and participants 2 and 3 remained neutral, stating that it would probably reduce bullying and abuse but not get rid of it completely.
- *Having multiple features to choose a programming language is much better than having only one feature as it allows for all sorts of users to be accounted for:* This was mostly agreed with, with one participant mentioning that this could slow down the process, however.
- *It is better to find a programming language in a few, less specific questions than a larger number of more specific questions:* This was a statement that received two ‘3’

answers and a ‘4’, where participants mostly felt that this depended on the context that the user was in when looking for a language.

The main points raised from this study were that LaTCh ‘Language and Technology Chooser’ is better for finding an answer more quickly, even if it is not completely specific. It also provides a higher number of ways to search for languages, which makes it more flexible to different types of users. And finally, if a user cannot find an answer on BPLFM, it is difficult to realise that right away. It is easy to get lost in the maze of questions on this website in comparison with LaTCh and if a user can’t find a language there is no way out. On LaTCh there is always a way out – if there is not an explicit answer on the website already a user can go to the discussion board and ask other users for help.

Had there been more time available on this project, this is a survey that would ideally have been carried out with a much larger number of participants to allow for a much more in-depth analysis. The results and feedback for this survey, as well as the scenarios, are described fully in appendix 12. Each participant signed a consent form to provide their consent to taking part in the survey. This form is available in appendix 13.

5.6 Learning done for this project

To develop this website, I had a number of new skills to learn. This was a new and first-time experience in terms of carrying out an individual large-scale project and new experiences ranged from research and literature review writing to password hashing in PHP. I had to develop my CSS and HTML skills further and had to learn back-end scripting using PHP from Scratch. This included secure database connections and blocking against possible injection attacks. In figure 25 you can see how the PHP password_hash function was used to encrypt a user’s password. This would make a simple string like ‘abcdef’ appear in an unreadable format such as ‘$2y$10$0ogYQvpckDrx9uxNLH2V6eDkt-9qN7m.O7RNMIgh3yIO’ which can only be decrypted using the ‘salt’ key generated on creation. Even the same password for different users would be encrypted differently, which

improves the strength of this type of hashing. Figure 26 shows how a new user is created using ‘prepared statements’ to mitigate against SQL and JavaScript injection. These types of statement ‘sterilize’ any user input before it is allowed to reach the database.

```
$user_password  = $_POST['psw'];
$hashed_password = password_hash ( $user_password , PASSWORD_BCRYPT );
```

Figure 25 – PHP Password hashing using the ‘BCRYPT’ hash.

```
$sql = $conn->prepare("INSERT INTO users (email, password, firstName,
surname, active) VALUES (?, ?, ?, ?, ?)");
$sql->bind_param("sssss", $email, $hashed_password, $first_name,
$second_name, $active);
$sql->execute();
```

Figure 26 – PHP Password hashing using the ‘BCRYPT’ hash.

5.7 Test Data

Throughout the course of this project auto-generated test data was used to test the system. This data includes fictional users, discussion threads, fact(s) of the day and programming languages used to populate the database and give the website a more realistic feel.

This test data was created using an online mock data generator called “Mockaroo” [43]. All the test data in the database can be seen from the admin screens (except user passwords).

6 Professional, Legal, Ethical and Social Discussion

This project has been carried out **professionally** using proper planning techniques, discussed in chapter 4.2 'Project Planning'. Further to this, the project has been well-organised throughout, with weekly supervisor meetings being held and the source code has been backed up using GitHub [34] version control software.

Legal issues have been considered throughout this project, in particular during the running of surveys, used to test the product created at each prototype stage. In accordance with the Data Protection Act (1998), participants in all studies have been informed where small amounts of personal data have been recorded. They have given consent for this data being taken and have been informed that their answers can be removed from the survey at any time if they require. Participant's information and answers have been stored securely on the Survey Planet [19] website and have been kept as anonymous as possible. In line with the Copyright, Designs and Patents Act (1988), images in the LaTCh website have been checked for copyright and all pages with images not owned or created by myself have been noted with a disclaimer. I have not claimed any images to be my own if they are not. Furthermore, any work by other authors in this document has been correctly cited with references.

Ethical Approval was granted for the studies that were undertook as part of this project. Participants were given full details of what would be required of them in all focus groups, usability studies and other testing of the product before they took part and were informed that they were taking part on a voluntary basis and they could leave at any time if they required. All participants were over the age of 16.

Socially, the product described in this document has been aimed mostly at beginners in computer programming, although it is hoped that it can be extended in the future to be used as a tool for all programmers.

7 Conclusions

Selecting a programming language suitable for one's needs is not an easy task. When choosing a language there are a huge number of factors that could be taken into consideration and it is hoped that this project can help to simplify the selection process.

7.1 Meeting the aims and objectives

The ultimate aim of this project was to create a website that would help programmers choose a programming language to suit their needs. The website was to be mainly be aimed at beginners, with plans to extend it to work for all computer programmers in the future. The objectives defined to achieve the above aim were as follows:

1. ***Carry out relevant research on programming and programming languages:***
   This was done by exploring related articles, books and web blogs and the results are summarised in chapter 2 'Literature Review'. This chapter discusses some of the factors that should be taken into consideration when choosing a programming language as well as the different languages that are out there and the languages currently taught in different levels of education.
2. ***Widely explore similar or overlapping products that already exist:***
   Chapter 3 'Technical Literature Review' discusses online products with similar and overlapping aims to LaTCh 'Language and Technology Chooser'. We have explored these and summed up in tables the advantages and disadvantages of these products, also discussing some of the ways LaTCh improves and extends upon them in chapter 3 'Technical Literature Review'.
3. ***Build a tool that achieves our aim:***
   Through the process of this project, 'LaTCh Language and Technology Chooser' has been created – a website which helps programmers choose a programming language (or other technology tool) through three main features, described as follows:

- A **Questionnaire**, which asks the user a few simple questions to quickly and easily get them to an answer.
- A **Categories** section, which lets the user decide more freely. Languages are broken down into a number of different categories, which hopefully appeal to different types of user, and this allows them to more freely explore the languages LaTCh has to offer.
- A **Discussion Board** where users can interact with each other. This has been implemented to help users ask questions and can be used when opinions and deeper context are needed to help them choose a language.

Chapter 4 'Work done as part of the Project' discusses how this website was designed and created.

4. Evaluate the new solution and compare it with existing products:

The LaTCh website has been continuously evaluated throughout the duration of the project. Before the development of the website started, a survey was carried out to find out the most popular programming languages. This was used to give an indication of what languages should definitely be included on the site. Furthermore, constant evaluation was carried out through uninterrupted contact with possible users and testing at every prototype stage. Three prototypes of the website were created before the final system and each was tested in the following ways:

- Prototype 1 – Focus Group.
- Prototype 2 – Small Usability Study.
- Prototype 3 – Large Usability Study.
- Final System – Site Comparison Survey.

Each testing method was used to gain feedback for the next prototype, and all these testing methods are discussed in more detail in chapter 5 'Testing and Performance Assessment'.

7.2 Future Development

LaTCh 'Language and Technology Chooser' has been developed with future development in mind from the beginning. It does not aim to be a final, complete product, but instead looks to provide the foundations for a website that could be used by all levels of computer programmers (from beginner to expert) for a range of different tasks, and the underlying software has been written with extendibility in mind.

The first aim of the future LaTCh website would be to become an **online learning tool**. This would mean having tutorials for each language on-site - at the moment all tutorials being linked are on external websites. It would also be an aim to include more detailed information on each of the programming language overview pages and the website could eventually be extended to have every single modern programming language listed. Unfortunately, these tasks were simply impossible to complete in the time-scale of this project. Another idea to assist with online learning would be to give users their own online profile, where they could keep track of all the languages they have been learning and have links to pick up where they left off.

In the future, it would also be a priority to turn the LaTCh website into a more useful **source for information**. It was already mentioned that I would like to add more information to each language and perhaps this information could be pulled from multiple sources using Semantic Web technologies or maybe users could be allowed to update and add to the information in a similar style to Wikipedia. More discussion boards could be added so users could discuss their issues within specific domains or with programmers working at a similar level as themselves. Another idea, which was mentioned in the requirements for this project but unfortunately not implemented, would be to allow users to rate certain languages on a scale of 1 to 10. This could give other users an indication of how popular languages are, and users could leave a comment to explain their rating in a review-type system. It is possible

that adding the online learning and information features described here would make LaTCh more useful for experienced programmers as well as beginners.

Finally, it would be an aim to make the website more **intelligent** and **secure**. The intelligence would come from a dynamic decision tree that would update automatically when new programming languages were added to keep the questionnaire function of the website as efficient as possible. Currently, the foundations for this idea are there, with the questionnaire being built over a static decision tree, which has to be manually updated when necessary. In terms of security, care has been taken at all points to make sure this website is secure. However, there is always more that can be done. With more time, I would have liked to have learned how to display the website using HTTPS (the encrypted version of HTTP – Hypertext Transfer Protocol).

These ideas, along with others that may arise, could all help to extend this website from not only *an online tool to choose a programming language* but also more of a *programmers' assistant* which computer programmers of any level could use to lookup reliable information, discuss their issues with others and choose languages based on their problem area.

8 References

Most of my references were found using Google, Google Scholar or Heriot-Watt University's 'Discovery' service: discovery.hw.ac.uk.

1. Khedker, U.P., 1997. *What makes a good programming language*. Technical Report TR-97-upk-1, Department of Computer Science University of Pune, India.
2. Chris Webb (2015) *The Pursuit of Simplicity in Programming* [online]. Available from http://blog.mediumequalsmessage.com/simplicity-in-programming [Last accessed 9th October 2017].
3. Kate Springer, CNN Tech (2016) *Japan's Fukuoka poised to be the country's Silicon Valley* [online] Available from http://money.cnn.com/2016/11/16/technology/fukuoka-startup-city/index.html [Last accessed 9th October 2017].
4. Brusilovsky, P., Calabrese, E., Hvorecky, J., Kouchnirenko, A. and Miller, P., 1997. *Mini-languages: a way to learn programming principles*. *Education and Information Technologies*, *2*(1), pp.65-83.
5. MIT App Inventor: Available from http://appinventor.mit.edu/explore/ [Last accessed 9th October 2017].
6. Rosson, M.B., 1993. *How might the object-oriented paradigm change the way we teach introductory programming? ACM SIGPLAN OOPS Messenger*, 4(2), pp.313–314.
7. Murphy, E., Crick, T. & Davenport, J.H., 2016. *An Analysis of Introductory Programming Courses at UK Universities*., pp.The Art, Science, and Engineering of Programming, 2017, Vol. 1, Issue 2, Article 18.
8. Ray Toal, Loyola Marymount University. *Programming Paradigms* [online] http://cs.lmu.edu/~ray/notes/paradigms/ [Last accessed 11th October 2017].
9. Gary T. Leavens (1997) *Major Programming Paradigms* [online] Available from www.eecs.ucf.edu/~leavens/ComS541Fall97/hw-pages/paradigms/major.html [Last accessed 11th October 2017].

10. Curricular Linux Environment at Rice (CLEAR) *Lec02: Programming Paradigms* [online] Available from https://www.clear.rice.edu/comp310/f12/lectures/lec02/ [Last accessed 11th October 2017].
11. Gabbrielli, M. et al., 2010. *Programming Languages Principles and Paradigms*, London: Springer London.
12. The Saylor Foundation *Advantages and Disadvantages of Object-Oriented Programming* [online] Available from https://www.saylor.org/site/wp-content/uploads/2013/02/CS101-2.1.2-AdvantagesDisadvantagesOfOOP-FINAL.pdf [Last accessed 20th October 2017].
13. Matt Weinberger (2017) *The 15 Most Popular Programming Languages according to "Facebook for Programmers"* [online] http://uk.businessinsider.com/the-9-most-popular-programming-languages-according-to-the-facebook-for-programmers-2017-10/#8-c-8 [Last Accessed 20th October 2017].
14. Cass, S., 2015. The 2015 top ten programming languages. IEEE Spectrum, July, 20.
15. Quora (2016) "*In which fields assembly language is used?"* [online] Available from https://www.quora.com/In-which-fields-assembly-language-is-used [Last accessed 9th November 2017].
16. Codecademy [online]. Available from https://www.codecademy.com/ [Last accessed 14th November 2017].
17. Stack Overflow [online]. Available from https://stackoverflow.com/ [Last accessed 14th November 2017].
18. Best Programming Language for Me [online]. Available from http://www.bestprogramminglanguagefor.me/ [Last accessed 14th November 2017]
19. Survey Planet [online]. Available from https://app.surveyplanet.com [Last accessed 16th November 2017].

20. Scottish Qualifications Authority (2016) *Reference language for Computing Science question papers (summary)* [online]. Available from http://www.sqa.org.uk/files_ccc/Reference-language-for-Computing-Science-summary-Sep2016.pdf [Last Accessed 14th February 2018].
21. Patel, J. (2017) *The 9 Most In-Demand Programming Languages of 2017* [online]. Available from http://www.codingdojo.com/blog/9-most-in-demand-programming-languages-of-2017 [Last Accessed 14th February 2018].
22. Indeed: Job Search [online]. Available from https://www.indeed.co.uk/ [Last Accessed 14th February 2018].
23. Prototype 1: Language Chooser [online]. Available from http://www2.macs.hw.ac.uk/~rs6/LanguageChooser1 [Last accessed 20th February 2018].
24. Prototype 2: Language Chooser [online]. Available from http://www2.macs.hw.ac.uk/~rs6/LanguageChooser [Last accessed 20th February 2018].
25. Dingsoyr et al., 2010. *Agile Software Development Current Research and Future Directions / edited by Torgeir Dingsøyr, Tore Dybå, Nils Brede Moe.*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 95.
26. Rouse, M. *Wizard of Oz Prototyping* [online]. Available from http://searchcio.techtarget.com/definition/Wizard-of-Oz-prototyping [Last accessed 26th February 2018].
27. D3 Collapsible Tree [online] http://mbostock.github.io/d3/talk/20111018/tree.html [Last accessed 6th March 2018].
28. Elizabeth, J. (2017) *Java is one of the most energy-efficient languages, Python among the least energy efficient* [online]. Available from https://jaxenter.com/energy-efficient-programming-languages-137264.html [Last accessed 13th March 2018].
29. Harzl, A. et al., 2013. *A Scratch-like visual programming system for Microsoft Windows Phone 8*.

30. Guo, P. (2014). *Python Is Now the Most Popular Introductory Teaching Language at Top U.s. Universities* [online]. Available from https://cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popular-introductory-teaching-language-at-top-u-s-universities/fulltext [Last accessed 13th March 2018].
31. Romy, B. (2012) *Why HTML is Not a Programming Language* [online]. Available from https://ischool.syr.edu/infospace/2012/04/05/why-html-is-not-a-programming-language/ [Last accessed 13th March 2018].
32. Kopetz, H., SpringerLink & LINK, 2011. *Real-Time Systems Design Principles for Distributed Embedded Applications / by Hermann Kopetz.* 2nd ed., Boston, MA: Springer US, pp 2-12.
33. Brosgol, B., 2007. Languages for Safety-Critical Software: Issues and Assessment. *Companion to the proceedings of the 29th International Conference on software engineering*, pp.180–181.
34. GitHub [online] Available from https://github.com/ [Last accessed 15th March 2018].
35. Trello [online]. Available from https://trello.com/ [Last accessed 15th March 2018].
36. XAMPP [online]. Available from https://www.apachefriends.org/index.html [Last accessed 15th March 2018].
37. PHPMyAdmin [online]. Available from https://www.phpmyadmin.net/ [Last accessed 15th March 2018].
38. Microsoft Word [online]. Available from https://products.office.com/en-gb/word [Last accessed 15th March 2018].
39. Facebook [online]. Available from https://www.facebook.com/ [Last accessed 15th March 2018].
40. LinkedIn [online]. Available from https://uk.linkedin.com/ [Last accessed 15th March 2018].
41. Twitter [online]. Available from https://twitter.com/ [Last accessed 15th March 2018].

42. Copeland, B. Jack (2017) “The Church-Turing Thesis”, *The Stanford Encyclopedia of Philosophy* [online]. Available from https://plato.stanford.edu/archives/win2017/entries/church-turing [Last Accessed 26th March 2018].
43. Mockaroo [online]. Available from https://www.mockaroo.com/ [Last Accessed 28th March 2018].
44. Prototype 3: LaTCh ‘Language and Technology Chooser’ [online]. Available from http://www2.macs.hw.ac.uk/~rs6/LaTCh3/ [Last accessed 16th April 2018].
45. Final System: LaTCh ‘Language and Technology Chooser’ [online]. Available from http://www2.macs.hw.ac.uk/~rs6/LaTCh/ [Last accessed 16th April 2018].
46. Chattopadyay, A. et al. *Lambda Calculus* [online]. Available from https://brilliant.org/wiki/lambda-calculus/ [Last accessed 16th April 2018].
47. The Java Programming Language [online]. Available from https://java.com/en/ [Last accessed 16th April 2018].
48. Standard ML of New Jersey [online]. Available from https://www.smlnj.org/ [Last accessed 16th April 2018].
49. Pascal [online]. Available from https://www.techopedia.com/definition/3940/pascal [Last accessed 16th April 2018].
50. Python [online]. Available from https://www.python.org/ [Last accessed 16th April 2018].
51. Swift 4 [online]. Available from https://developer.apple.com/swift/ [Last accessed 16th April 2018].
52. Get started with Visual Basic [online]. Available from https://docs.microsoft.com/en-us/dotnet/visual-basic/getting-started/ [Last accessed 16th April 2018].
53. SWI-Prolog [online]. Available from http://www.swi-prolog.org/ [Last accessed 16th April 2018].

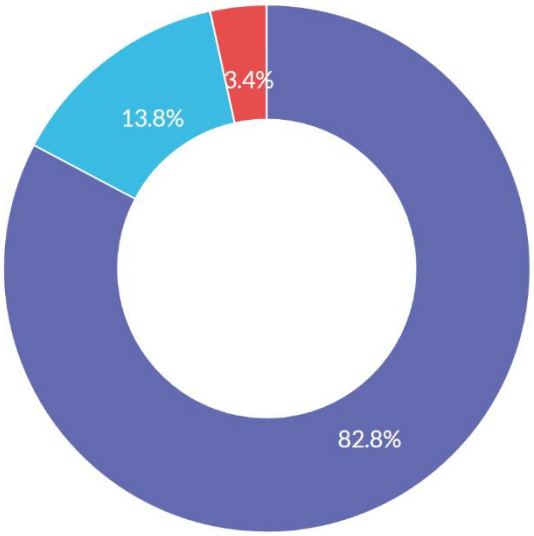
54. C Language Overview – Tutorials Point [online]. Available from https://www.tutorialspoint.com/cprogramming/c_overview.htm [Last accessed 16th April 2018].
55. C++ Language Tutorial [online]. Available from http://www.cplusplus.com/doc/tutorial/ [Last accessed 16th April 2018].
56. The Rust Programming Language [online]. Available from https://www.rust-lang.org/en-US/ [Last accessed 16th April 2018].
57. The Ada Programming Language [online]. Available from https://www.adacore.com/about-ada [Last accessed 16th April 2018].
58. Algol Computer Language [online]. Available from https://www.britannica.com/technology/ALGOL-computer-language [Last accessed 16th April 2018].
59. Scratch [online]. Available from https://scratch.mit.edu/ [Last accessed 16th April 2018].
60. Ruby – A Programmer's Best Friend [online]. Available from https://www.ruby-lang.org/en/ [Last accessed 16th April 2018].
61. What is R? [online] Available from https://www.r-project.org/about.html [Last accessed 16th April 2018].
62. HTML tutorial – w3Schools [online]. Available from https://www.w3schools.com/html/ [Last accessed 16th April 2018].
63. XML tutorial – w3Schools [online]. Available from https://www.w3schools.com/xml/default.asp [Last accessed 16th April 2018].
64. SQL Introduction – w3Schools [online]. Available from https://www.w3schools.com/sql/sql_intro.asp [Last accessed 16th April 2018].
65. MongoDB [online]. Available from https://www.mongodb.com/ [Last accessed 16th April 2018].
66. Neo4j [online]. Available from https://neo4j.com/ [Last accessed 16th April 2018].

67. JavaScript [online]. Available from https://www.javascript.com/ [Last accessed 16th April 2018].
68. PHP 5 Tutorial [online]. Available from https://www.w3schools.com/php/default.asp [Last accessed 16th April 2018].
69. The Mercury Project [online]. Available from https://www.mercurylang.org/ [Last accessed 17th April].

9 Appendices

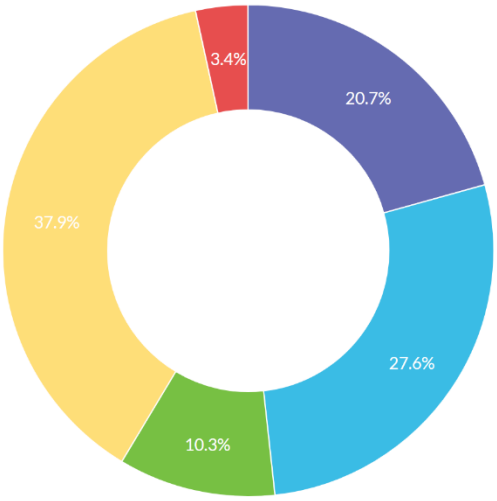
Appendix 1 – Survey results

Q1. Please select your age range from the list below.



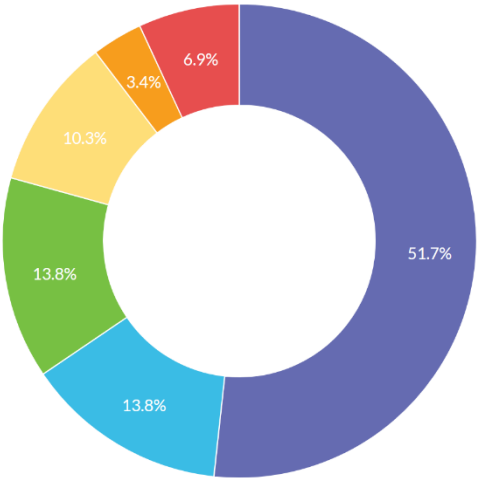
Choices	Totals
18-24 years old	24
25-34 years old	4
35-44 years old	0
45-54 years old	0
55+	0
Prefer not to say	1

Q2. If you are a student, please select your year of study.



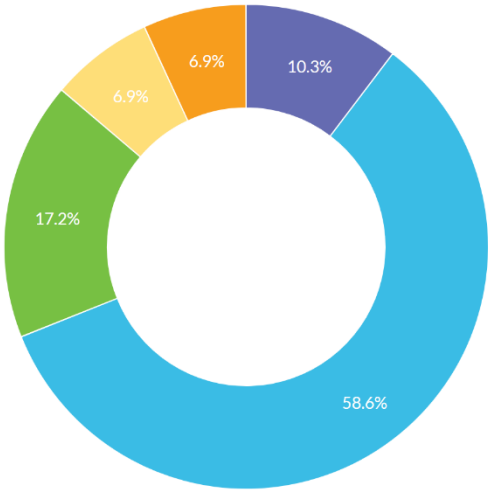
Choices	Totals
Year 1	6
Year 2	8
Year 3	3
Year 4	11
Year 5	0
I am not a student	1

Q3. If you are a student, please select the programme you are studying.



Choices	Totals
BSc Computer Science	15
BSc Information Systems	4
BSc Computer Systems	4
MEng Software Engineering	3
I am not a student	1
Other (please specify)	2

Q4. Scenario 1: You have been asked by a local high school to come in and teach their 4th year students a new programming language over a time period of around 3/4 weeks. They are relatively new to programming but have learned the basics through the visual programming language Scratch and they have also done some small websites with HTML. What language do you choose to teach the students? Be ready to explain your answer for the next question.



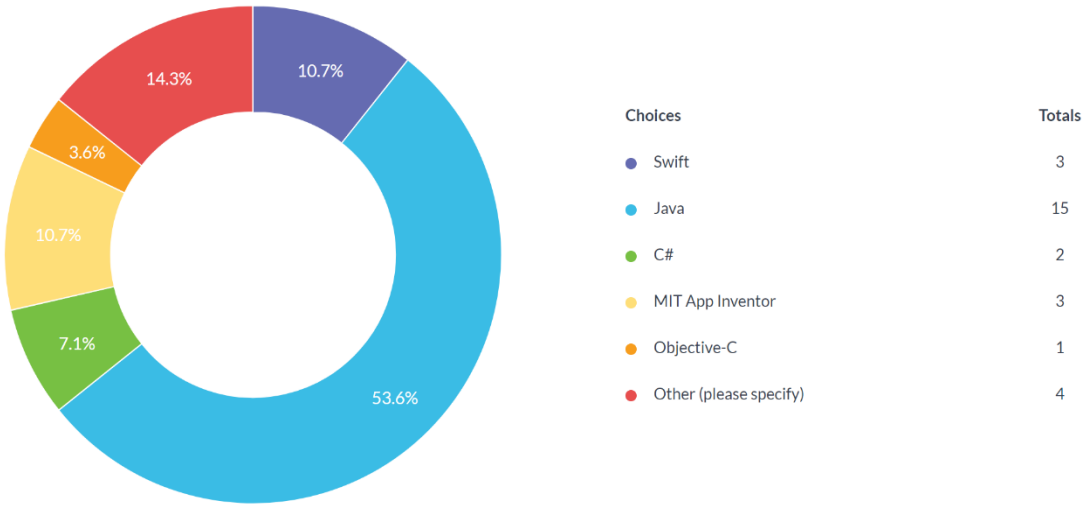
Choices	Totals
Visual Basic	3
Python	17
Java	5
PHP	2
Other (please specify)	2

Q5. Please explain the answer you gave for this scenario in the previous question.

Some common answers:

- (Python) Syntax is easy to understand.
- (Python) Easy to learn for beginners and can be applied to many different areas of software development.
- (Python) Forces them to think about code readability through indentation.
- (Java) Some people find it easier to teach the concept of object-oriented programming.
- (Java) Widely used and has good libraries.
- (Visual Basic) Simple syntax and the Visual Studio IDE makes things simpler for beginners than writing in a text file directly or in the terminal.

Q6. Scenario 2: You have just started working at a company that develops mobile apps. Your boss says that he wants you to take the lead on building a new app that can be used to track the users' fitness (counting steps, recording what they eat etc.). It's up to you which mobile platform you build the app for (between iOS, Android and Windows), but once it has been built the company will recreate it for other platforms if it is successful. What language would you choose to work in? Please be prepared to answer why in the next question.



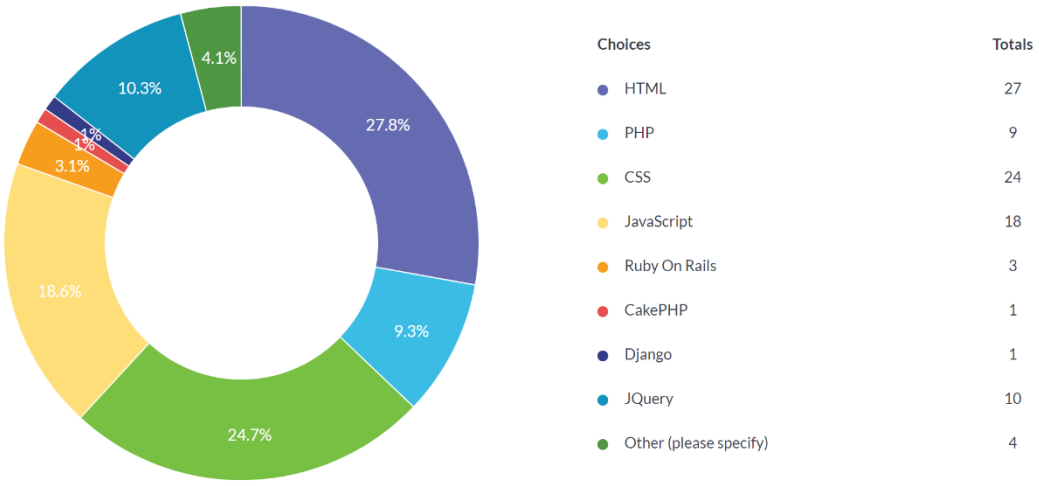
Q7. Please explain the answer you gave for this scenario.

Some common answers:

- (Python) It is a powerful language and is easy to use, it also has a huge amount of documentation to help you.
- (Swift) as this is currently used for iOS applications.

- (Java) Android has the largest market share.

Q8. Scenario 3: You are working part-time at a small cafe in Edinburgh to earn some extra cash while you study. The cafe manager finds out you study computing at university and asks if you could build a website for the cafe to extend their reach for customers. You gladly accept (this will be a great personal project to put on your CV). What languages/frameworks will you choose? You may choose multiple. Please be prepared to explain your answer in the next question.

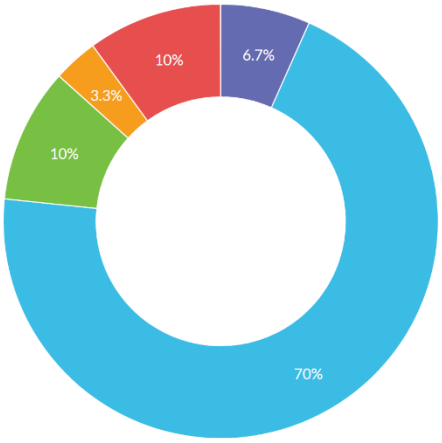


Q9. Please explain the answer you gave for this scenario.

Some common answers:

- (HTML, CSS, JavaScript) Widely used.
- (WYSIWYG Service Square Space) Only a small business and it allows the developer to offload website maintenance to less skilled users.
- (HTML) Not all browsers support all formats, so sticking with the classics is a good way to go.

Q10. Scenario 4: As part of a coursework task at university, you have to search a large text file for certain words and phrases and count the number of times each one occurs. You have also been told to do this using a scripting language. Which language would you use?



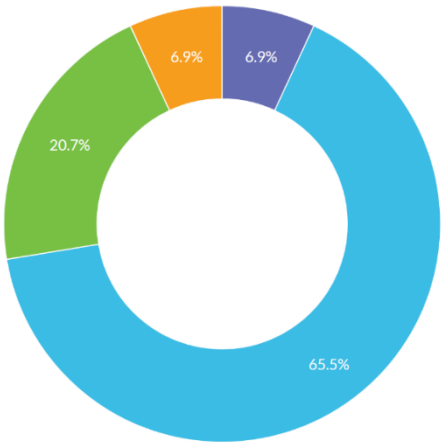
Choices	Totals
Bash	2
Python	21
JavaScript	3
Perl	0
Ruby	1
Other (please specify)	3

Q11. Please explain the answer you gave for this scenario.

Some common answers:

- (Python) Simple to use.
- (JavaScript) Allows you to use regexes to do this sort of task.
- (Bash) Easy to use in the command line.

Q12. What is your favourite language for object-oriented programming?



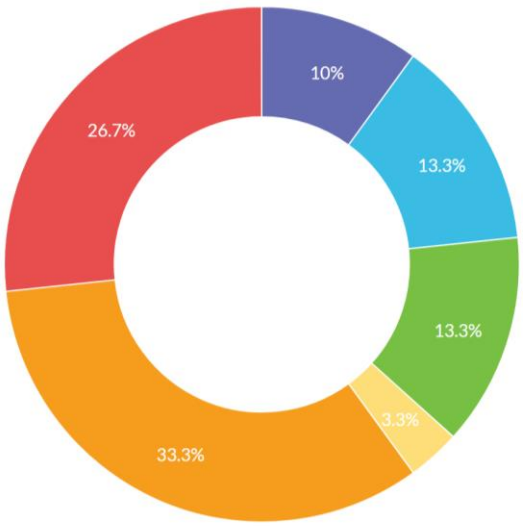
Choices	Totals
C++	2
Java	19
Python	6
Smalltalk	0
Other (please specify)	2

Q13. Please explain your answer to the question ‘What is your favourite language for object-oriented programming?’

Some common answers:

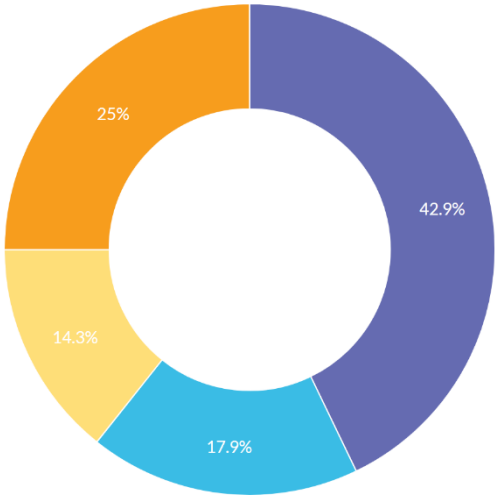
- (Java) Has good libraries and syntax.
- (Java) More experience with Java.
- (C++) Extra features such as async and await.
- (C++) Good low-level control.
- (Python) Good syntax

Q14. Have you worked with frameworks before for web development? If so, please select your favourite from the list below.



Choices	Totals
CakePHP	3
Ruby on Rails	4
JQuery	4
Django	1
I have not used any frameworks before.	10
Other (please specify)	8

Q15. What is your favourite language for server-side scripting?



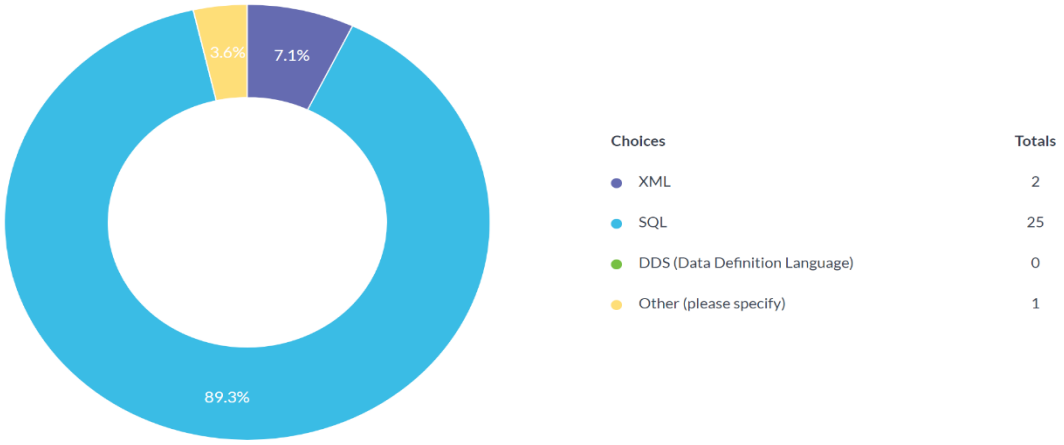
Choices	Totals
PHP	12
Java	5
Perl	0
Ruby	4
Other (please specify)	7

Q16. Please explain your answer to ‘What is your favourite language for server-side scripting?’

Some common answers:

- (PHP) Widely used.
- (Python) The Flask framework is useful.
- (Ruby) Experience.

Q17. What language would you prefer to write and manipulate databases with?



Q18. Please name the first programming language you think of for each of these paradigms. You can miss some out if you can't think of an example at this time. Object-Oriented, Procedural/Imperative, Declarative, Functional, Logical, Event-Driven, Parallel.

The most common for each paradigm has an asterisk* beside it.

Language	Object-Oriented	Procedural/ Imperative	Declarative	Functional	Logical	Event Driven	Parallel
Java	~~IIII~~ ~~IIII~~ ~~IIII~~ ~~IIII~~ III*	IIII	I	I		I	II
C++		I	I	I		II	I
Prolog			~~IIII~~ II*		~~IIII~~ ~~IIII~~ ~~IIII~~ ~~IIII*~~		
JavaScript	I					~~IIII~~ IIII*	
Python	II	~~IIII~~ IIII*		III	I		
SML			I	~~IIII~~ IIII*			
Haskell			I	IIII	I		II
ML			II	IIII			
C#	I	I				II	II
Basic		II					
SQL			~~IIII~~ I				
PHP				II			
C		III					~~IIII~~ I*
Cuda							I
F#				I			
Pascal		I					
Visual Basic	I					IIII	
OCAML				I			I
NodeJS						I	
Go							I
PDDL			I				
MatLab				I			
Erlang			I				
Event-ML						I	
Pascal		I					
Lisp					I		

Q19. If you are working on a new task and you discover that a language you have not learned before is much better suited for this task than languages that you already know, how likely are you to learn this new language from 1 – not likely at all to 5 – Extremely likely?

Average: 3.71

Appendix 2 – Use Case Textual Descriptions

Textual descriptions for each of the use cases in the use case diagram 'Language Chooser'.

Each textual description has a unique ID number, a name, actors who will be able to carry out the use case, a general description, a main flow of steps that will be involved in carrying out the use case, preconditions and postconditions (conditions that must be true before and after the use case). The main reason for the ID number is so that each use case can be uniquely identified. This ID will not change throughout the project, even if the name of the use case does.

ID: 1.0

Name: View Fact of the Day

Actors: User

Description: The act of the user seeing and reading the fact of the day that will be displayed on the website's homepage.

Main Flow:

1. The user navigates to the website homepage.
2. The fact of the day is displayed on screen.

Preconditions: The user must be on the website homepage.

Postconditions: The user now knows the fact of the day

ID: 2.0

Name: Create New Discussion Thread

Actors: User

Description: The act of the user creating a new discussion thread which can be used to post problems that you are having or questions you have for other users on the system.

Main Flow:

1. The user clicks on the 'Create new discussion thread' option.
2. The user enters details for the discussion thread:
    a. Subject
    b. Description/question
3. The user selects the 'Submit' option to make the thread active.
4. extensionPoint: Edit Thread

Preconditions: The user must be on the discussion threads page.

Postconditions: A new discussion thread has now been created that can be edited, viewed and replied to.

ID: 3.0

Name: View Discussion Thread

Actors: User

Description: The act of a user viewing some discussion thread that has been created by themselves or another user.

Main Flow:

1. The user will navigate to the thread that they want to view.
2. The thread information and replies will be displayed.
3. extension: Reply to Discussion Thread.

Preconditions: The user must be on the discussion threads page.

Postconditions: The user has now viewed the discussion thread they were interested in.

ID: 4.0

Name: Start Questionnaire

Actors: User

Description: The act of a user taking part in the questionnaire function of the website.

Main Flow:

1. The user selects the ‘Start Questionnaire’ button.
2. include(Answer Questions)

Preconditions: The user must be on the main page of the website.

Postconditions: The user has started the questionnaire.

ID: 5.0

Name: Edit Language Description

Actors: Admin

Description: The act of an administrator editing the description associated with a particular programming language in the system’s database.

Main Flow:

1. The administrator will navigate to the language they want to change the description for.
2. The administrator will type into an input form the new description that they want the language to have.
3. The administrator will select ‘Submit’.

Preconditions: The actor must have an admin account and must be logged in.

Postconditions: The language now has a new description.

ID: 6.0

Name: Add New Language

Actors: Admin

Description: The act of an administrator adding a new programming language to the database.

Main Flow:

1. The administrator will navigate to the admin languages page.
2. The administrator will select the ‘Add new language’ option.
3. The administrator will enter details about the language into an input form and select the ‘Submit’ option.

Preconditions: The actor must have an admin account and must be logged in.

Postconditions: A new language has been added to the database.

ID: 7.0

Name: Delete Discussion Thread

Actors: Admin

Description: The act of an administrator removing a discussion thread from the system. This may be done at request from a user or due to it being outdated.

Main Flow:

1. The administrator will navigate to the admin threads page.
2. The administrator will select the thread they wish to delete, and click the ‘delete thread’ option.

Preconditions: The actor must have an admin account and must be logged in.

Postconditions: The discussion thread has now been deleted.

ID: 8.0

Name: View All Languages

Actors: Admin, User

Description: The act of an actor viewing a summary of all languages in the database. The view will be slightly different for admin users and regular users.

Main Flow:

1. The actor will navigate to the ‘view all languages’ page.

2. extensionPoint: View Language Description.

Preconditions: A User must be on the website. An Admin must be on the website, have an admin account, and must be logged in.

Postconditions: The actor has viewed a list of all languages in the database.

ID: 9.0

Name: Update Fact of the Day

Actors: Admin

Description: The act of an admin manually updating the fact of the day.

Main Flow:

1. The actor will navigate to the administrator view of the fact of the day.
2. The actor will then select the 'edit fact of the day' option.
3. The actor will fill an input form with the new fact of the day and select 'submit.

Preconditions: The actor have an admin account, and must be logged in.

Postconditions: The fact of the day has been updated.

ID: 10.0

Name: Create New User

Actors: Admin

Description: The act of an administrator creating a new administrator account.

Main Flow:

1. The actor will navigate to the 'users' page.
2. The actor will select 'Create new User' option.
3. The actor will fill in a form with information about the new user and press 'submit'.

Preconditions: The actor have an admin account, and must be logged in.

Postconditions: A new admin user has been created.

ID: 11.0

Name: Edit User Details

Actors: Admin

Description: The act of an administrator updating the details for a particular admin user.

Main Flow:

1. The actor will navigate to the 'users' page.
2. The actor will select a user to edit.

3. The actor will fill in a form with the information they want to change and press ‘submit’.

Preconditions: The actor have an admin account, and must be logged in.

Postconditions: A user’s details have now been updated.

ID: 2.1

Name: Edit Thread

Actors: User

Description: The act of the user editing the details of a discussion thread that they have created.

Main Flow:

1. The user navigates to the thread they created.
2. The user edits the details of the thread through input forms and clicks ‘submit’.

Preconditions: The user must have created a discussion thread already.

Postconditions: The details of the thread have now been updated.

ID: 3.1

Name: Reply to Discussion Thread

Actors: User

Description: The act of a user posting their own comments in reply to a discussion thread that has already been created.

Main Flow:

1. The user selects ‘reply’.
2. The user enters their comment into a form and selects ‘submit’.

Preconditions: The user must be viewing a specific thread.

Postconditions: The user has now replied to the discussion thread.

ID: 8.1

Name: View Language Description

Actors: Admin, User

Description: The act of an actor viewing the description for a specific language in the database. The view will be slightly different for admin users and regular users.

Main Flow:

1. The actor selects a language to view.
2. The description will be shown on screen.

Preconditions: None

Postconditions: The actor has viewed the description for a particular language.

ID: 4.1

Name: Answer Questions

Actors: User

Description: The act of a user answering questions in the questionnaire function of the website.

Main Flow:

1. The user answers the current question to take them closer to a specific programming language.
2. The user repeats step one until a result is obtained.
3. include(See Results)

Preconditions: The user must have started the questionnaire already.

Postconditions: The user has answered question(s) on their problem and possibly been given a result.

ID: 4.2

Name: See Results

Actors: User

Description: The act of a user viewing the results of their answers to the questions they answered.

Main Flow:

1. The user completes the questionnaire.
2. A programming language or list of programming languages are displayed on the screen.
3. extensionPoint: View Language Description

Preconditions: The user must have started the questionnaire already.

Postconditions: The user has answered question(s) on their problem and possibly been given a result.

Appendix 3 – Prototype 2 Survey Results

Please select the Heriot-Watt University campus you are currently studying at.

Edinburgh	6 (86%)
Dubai	1 (14%)
Total	7

TASK 1: Start at the home page www2.macs.hw.ac.uk/~rs6/LanguageChooser. Please use the navigation bar at the top of the screen to go to the 'Languages' screen. On this screen, find the year that 'LISP' was released. Please enter the year in the box below as well as any comments you have about the user interface.

1958, nice UI, maybe highlight options, or draw attention to header better so people wont miss the links at the top
Not responsive at all, especially during JavaScript parts.
1958 I think it would be good if it was a drop down menu that shows it. Flexbox would be good for an alternative method for making it look nicer.
1958 - well displayed and consistent in every language, makes it really easy to follow.
1958. The Languages button should be over at the left.
1958
1958 hard to find language page...

TASK 2: From the homepage, navigate to the questionnaire function (through the 'Choose a language via a series of questions' button), which allows you to choose a language or set of languages via a series of questions. Try to find out at least one framework you could use for web development. Leave the answer you find and any comments you have below.

HTML. The tree isn't perfect, maybe open up the options to a question as well, so the user doesn't need to click twice to get options. Maybe reword some questions (i.e. high level/low level) or add a quick definition for terms like this.
Very bad experience when using mobile
CakePHP. I like this way of using a tree.
CakePHP - liked the functionality of the questions, very easy to follow and good that it can be reset when filling out the question.
CakePHP. When you arrive at the final node maybe give information about the framework.
HTML, the method for selecting the different branches is a bit cumbersome and a different method may be better
CSS found it by luck

TASK 3: Please use the navigation bar to get to the discussion board, and have a go at creating a new thread. It's up to you if you would like to enter an example question or something more like 'Test thread'. Please leave any comments you have on the interface here.

nice feature. Maybe ensure the page redirects after upload to the thread page.
I like the whole interface. An improvement would possibly to use AJAX to make it only a single page rather than have to return back to the page from another page.
Created Testing2 thread
Seems fine layout is nice
I got an error when I tried to submit my thread
Comments are in the thread

TASK 4: Find your thread in the discussion board and click on it. From here you should see more information on the thread and you should be able to reply to it. Please do this. For the reply, you can enter anything you like. However, in the thread number you should enter the number shown at the top of the screen. This thread number will be added automatically in later versions of the website. Again, please leave any comments you have here.

Works well, but add different sections and highlight them like question and comments as currently there is just two dashes. Even just add a box around comments etc so as it is hard to differentiate right now.
I like that you have thought ahead about the issue of the thread number as that was the only issue I could see. Otherwise it's very good.
Seems fine
Little hard to find without the direction in the question, highlight at bottom too. Redirect back to original page afterwards (although I figure you're working on this anyway). Send button is cramped at the bottom.

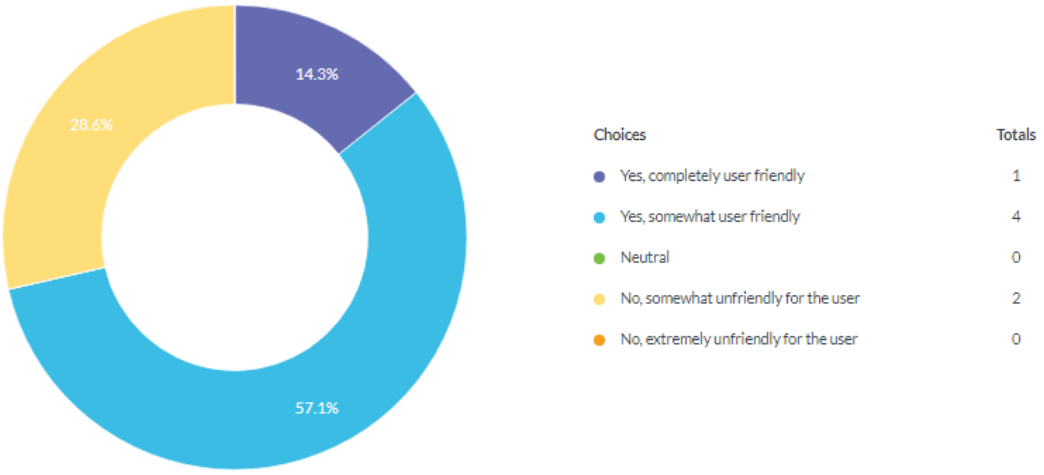
TASK 5: Please navigate to the 'Feedback' page using the navigation bar at the bottom of the page, and leave some feedback in the form provided. You do not need to enter your name or email address if you don't want to.

Little hard to find without the direction in the question, highlight at bottom too. Redirect back to original page afterwards (although I figure you're working on this anyway). Send button is cramped at the bottom.
I found doing this very easy

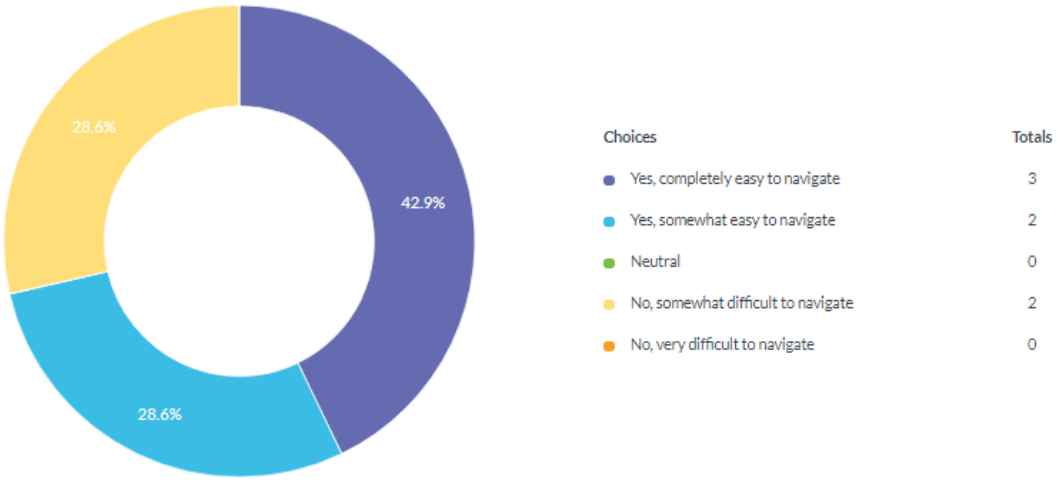
QUESTION 1: Do you have any comments on the current colour scheme for the website? (Black navigation bars with black buttons which turn blue when hovered over) Is there anything you would change about it or do you like it how it is?

UI and colour scheme is good, not too dark but well balanced.
It's ok
I like the colour scheme. Would maybe look nicer with bolder text in the nav bar. And highlight in the navbar too.
I would possibly use another colour rather than black, possible dark grey or a darker colour
Maybe make it a little more colourful.
The colour scheme is good, It is quite neutral and easy on the eyes.
black and white. I like colour

QUESTION 2: Would you consider the current design of the website to be user friendly?



QUESTION 3: Would you consider the current version of the website as easy to navigate?



QUESTION 4: Do you feel that there are any features missing from this website based on what you would use it for? Feel free to explore more before you answer this question.

Some code examples for the main languages in the questionnaire, like a code snippet of "hello world" for C, Java, Python, etc. -Add a page for each language in questionnaire like was in prototype 1, maybe with some links to tutorials etc
Responsiveness
Possibly embed YouTube tutorials onto webpage.
could possibly mention basic syntax?
Everything seems fine
not sure

I don't particularly do a lot of coding so I woudn't really have much use for the website.

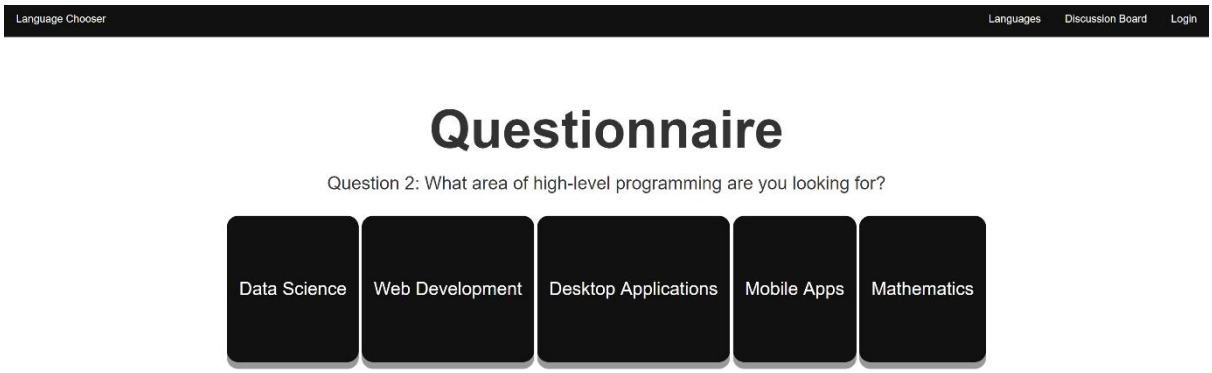
QUESTION 5: Finally, was there anything that you particularly liked about this prototype?

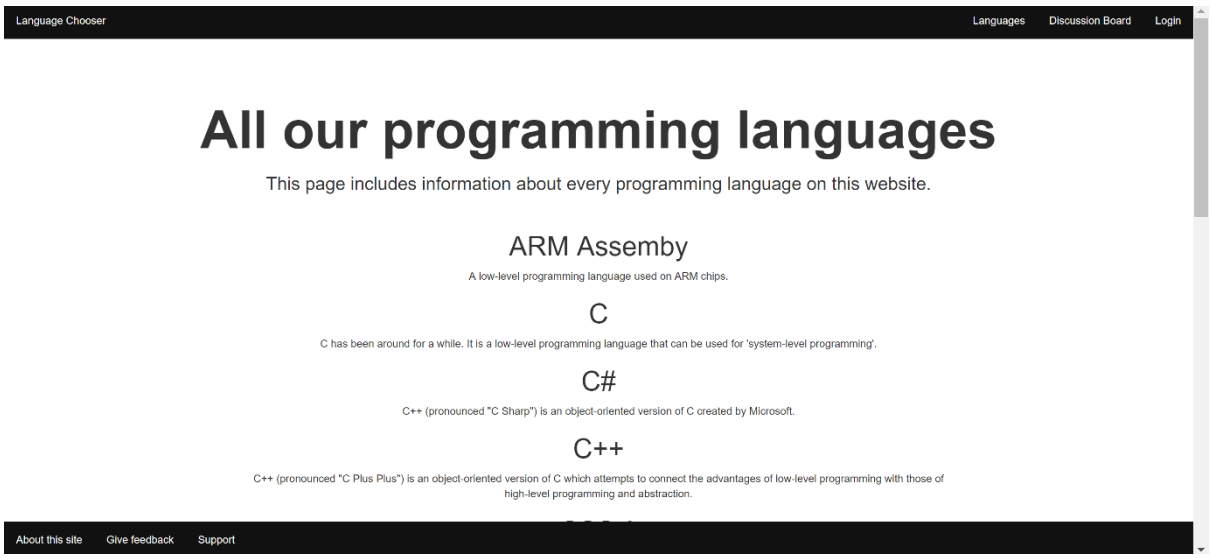
Clean UI, easy to use other than some little hiccups. The D3 tree is a nice idea, and when working correctly will be nice.
Color scheme
The simplicity of it. It's meant for people that may not necessarily have a lot of experience with computers or programming so that's good.
Liked the functionality of the question section and the simple navigation bar.
Discussion Board
I like that the prototype offers something different and it's something (which to my knowledge) hasn't really been done before
I learned new things which I appreciate. It could be made a valuable website that I will always use.

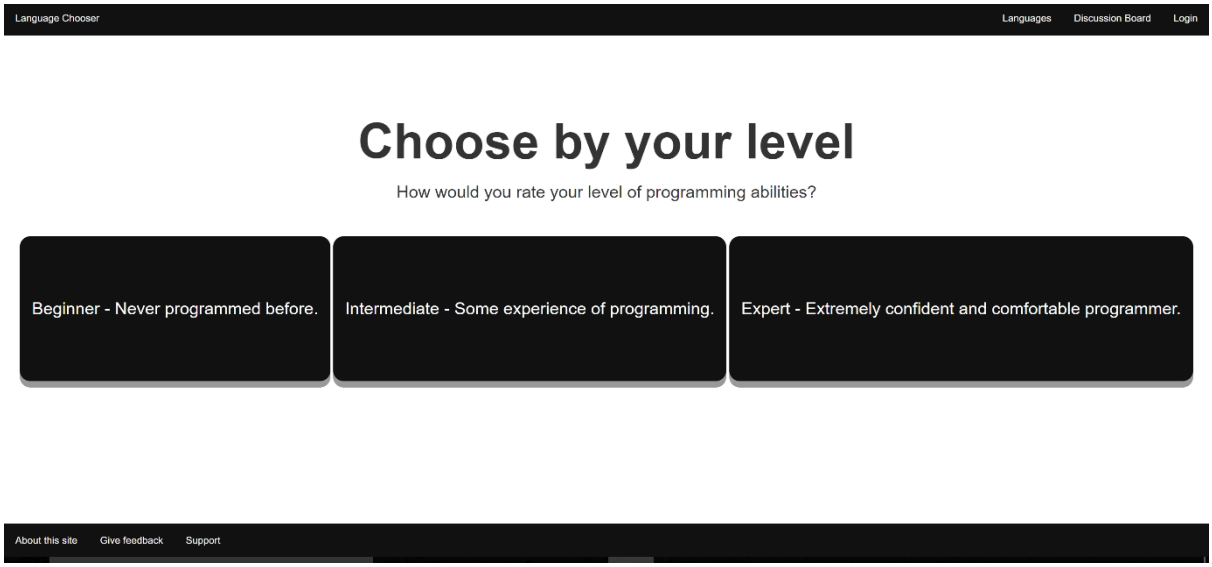
Appendix 4 – Project Plan in Table Form

Task	Start Date	End Date	Duration (days)
First meeting with supervisor	18/09/2017	24/09/2017	6
Abstract and Declaration	19/09/2017	16/10/2017	27
Create Dissertation Skeleton File	19/09/2017	26/09/2017	7
Aims and Objectives	19/09/2017	17/11/2017	59
Literature Review	19/09/2017	24/11/2017	66
Survey	30/10/2017	22/11/2017	23
Requirements Analysis	10/10/2017	24/11/2017	45
Software Design	30/11/2017	24/12/2017	24
Project Management	16/10/2017	24/11/2017	39
Evaluation Strategy	13/11/2017	24/11/2017	11
Ethics Health and Safety Forms submitted	01/11/2017	24/11/2017	23
Develop/Test/Evaluate Prototype 1 (Static Implementation)	27/11/2017	20/12/2017	23
Meet with second reader	04/12/2017	11/12/2017	7
Database Design	30/11/2017	31/12/2017	31
Develop/Test/Evaluate Prototype 2 (with database connection)	28/12/2017	30/01/2018	33
Develop/Test/Evaluate Prototype 3 (with decision tree)	01/02/2018	28/02/2018	27
Develop Final System	01/03/2018	30/03/2018	29
Testing & Evaluation	01/03/2018	15/04/2018	45
Write up Dissertation	01/02/2018	15/04/2018	73
Usability study	07/03/2018	21/03/2018	14
Create Poster	01/04/2018	30/04/2018	29

Appendix 5 – Prototype 1 screenshots







Appendix 6 –

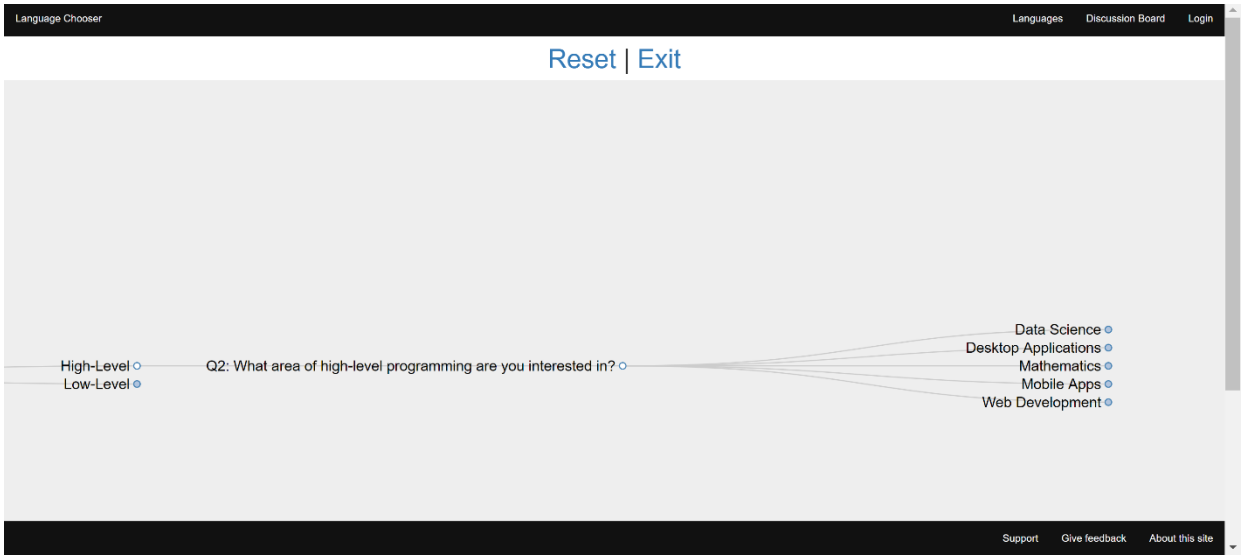
Questionnaire Decision Tree, underlying table with questions and answers

Q1	**What type of programming would you like to do?**
A1	Desktop Applications
A2	Mobile Apps
A3	Web Development
A4	Mathematics
A5	Build an OS
A6	Embedded Systems
A7	Something easy, and fun! (for beginners)
A8	Build a Compiler
Q2.1	**What Operating System would you like to build desktop apps for?**
A8	Windows
A9	Mac OS
A10	Linux
A11	Cross Platform
Q2.2	**What Operating System would you like to build mobile apps for?**
A12	Android
A13	iOS
A14	Windows
Q2.3	**What kind of web development would you like to do?**
A21	Build a static website
A22	Build a dynamic website with a database
A23	Client-side scripting
A24	Server-side scripting
A25	Build a dynamic website quickly with a framework
A26	Build a dynamic website quickly with a framework
Q3	**What type of database system would you like to use?**
A27	Relational
A28	Non-Relational
Q2.4	**What area of maths?**
A15	Function Manipulation
A16	Logic
A17	Matrix Manipulation
Q2.5	**What would you like to do?**
A18	Block programming – Create a mobile app
A19	Block programming – Create small animations
A20	Simple coding exercises

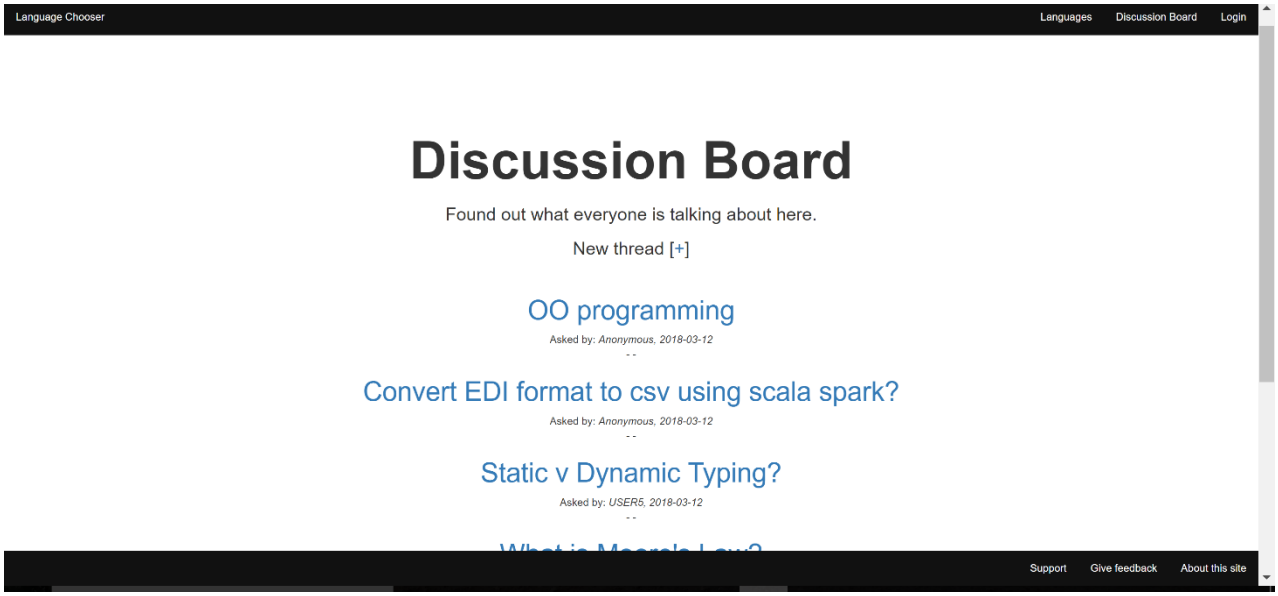
Appendix 7 – Prototype 2 screenshots

Home Page

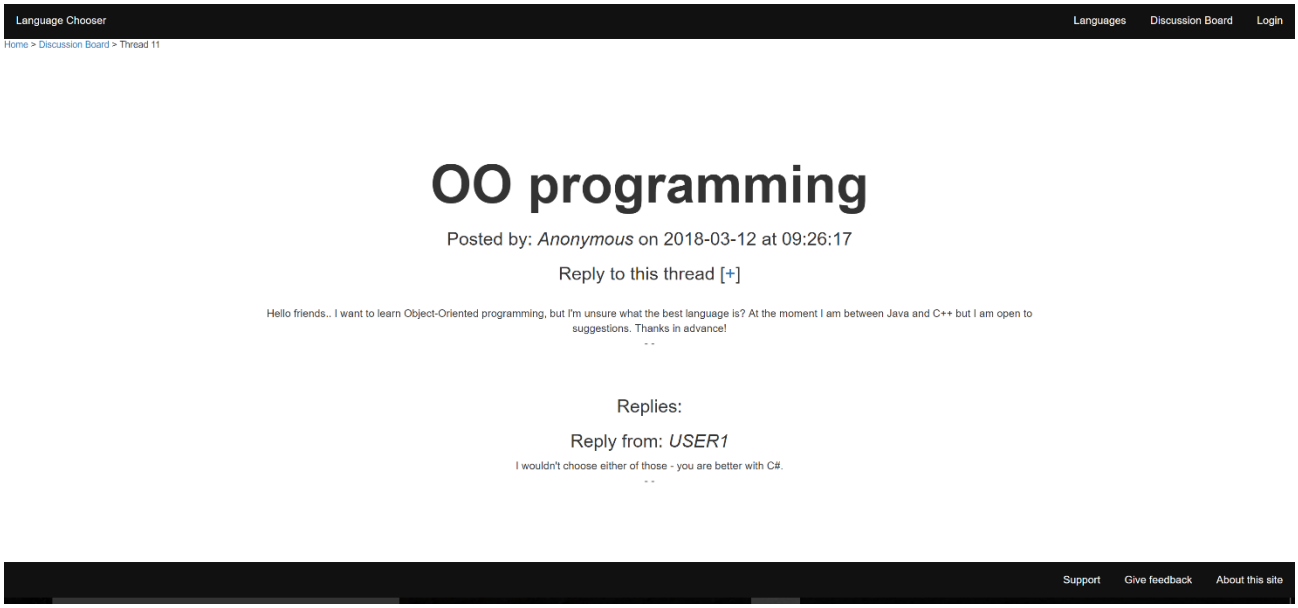
Questionnaire



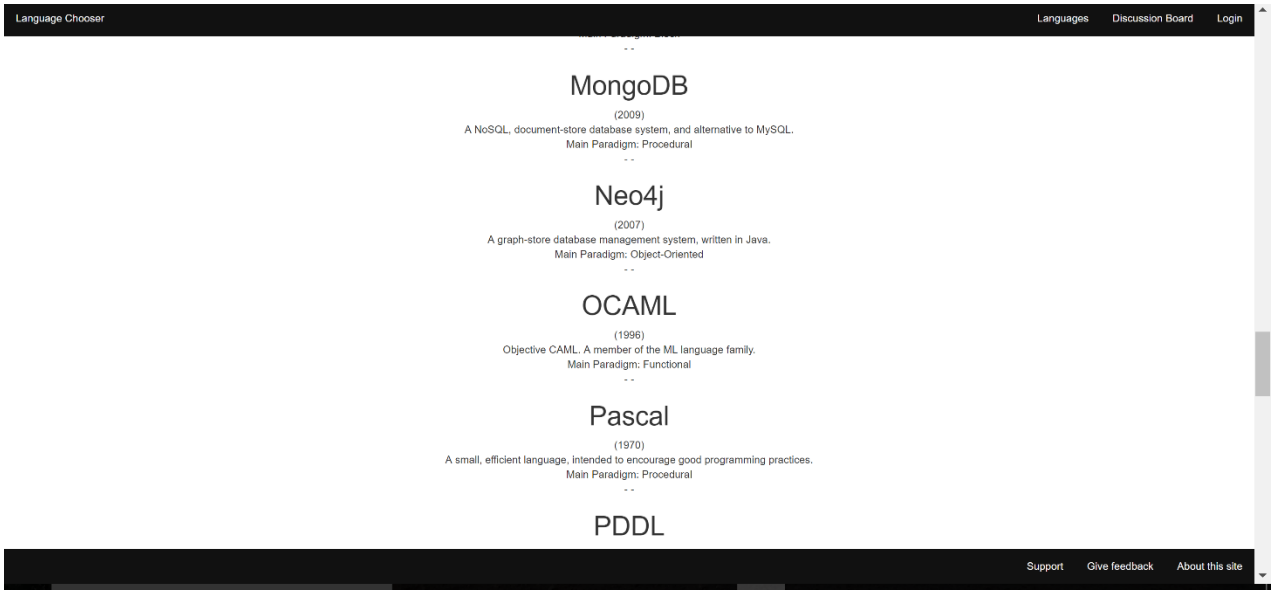
Discussion Board



Discussion Board – Viewing a Thread



Languages

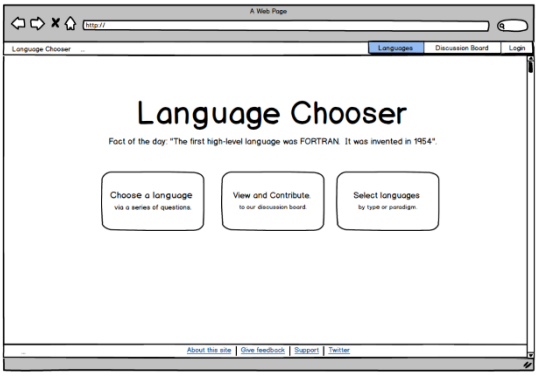

Language Chooser
Languages
Discussion Board
Login
MongoDB
(2009)
A NoSQL, document-store database system, and alternative to MySQL.
Main Paradigm: Procedural
Neo4j
(2007)
A graph-store database management system, written in Java.
Main Paradigm: Object-Oriented
OCAML
(1996)
Objective CAML. A member of the ML language family.
Main Paradigm: Functional
Pascal
(1970)
A small, efficient language, intended to encourage good programming practices.
Main Paradigm: Procedural
PDDL
Support
Give feedback
About this site

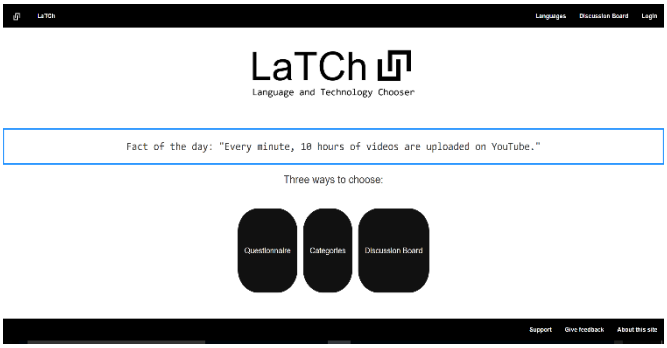
Appendix 8 – Site Map (table form)

ID Number	Name of page	Filename of page
0.0	Homepage	index.php
1.0	About	about.php
2.0	Feedback	feedback.php
3.0	Login	login.php
4.0	Discussion Board	allboards.php
5.0	Categories	categories.php
6.0	Questionnaire - intro	questions.php
7.0	Languages	all.php
8.0	Admin Home	Admin/home.php
3.1	Sign Up	signup.php
3.2	Admin Login	Admin/login.php
4.1	View Thread	DiscussionBoard/thread.php
4.2	Edit Thread	DiscussionBoard/editThread.php
4.3	Create Thread	DiscussionBoard/newThread.php
5.1	Databases	Categories/Databases.php
5.2	Data Science	Categories/DataScience.php
5.3	Frameworks	Categories/Frameworks.php
5.4	High/Low level	Categories/HighOrLow.php
5.5	Paradigms	Categories/Paradigms.php
5.6	Popularity	Categories/Popularity.php
5.7	Scenarios	Categories/Scenarios.php
5.8	Web Development	Categories/WebDev.php
7.1	Individual Language Page	<<Language>>.php

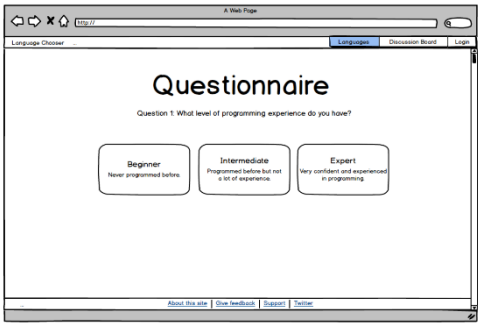
Appendix 9 – Initial website mock-ups & final EUI

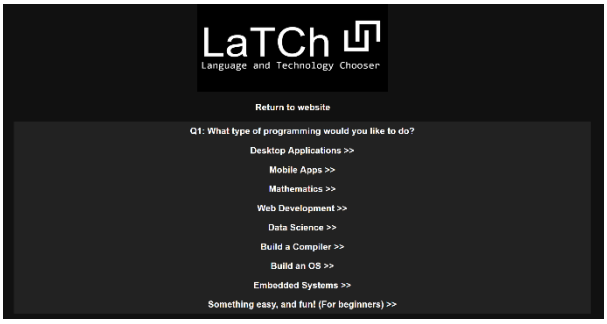
Index/Home screen/Landing Page:



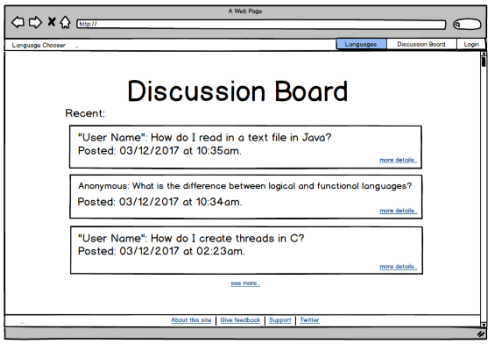


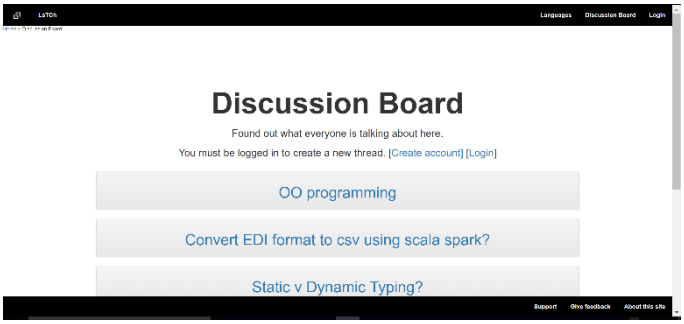
Questionnaire:



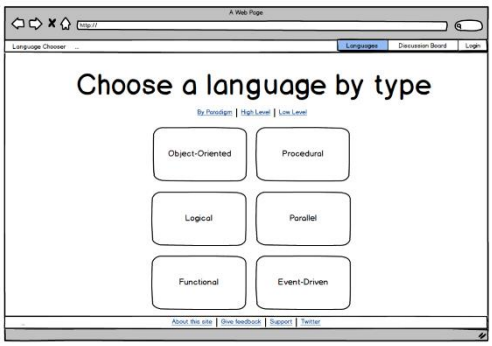


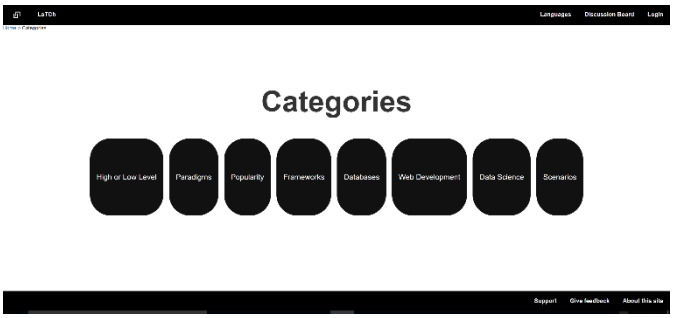
Discussion Board:





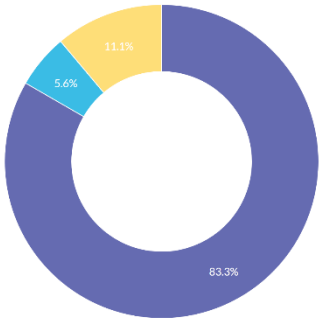
Choose a language by type – later extended to 'Categories'





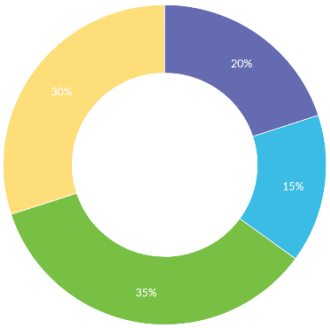
Appendix 10 – Full usability study; results

Q1 Please select your age range (you must be over 16 to complete this survey)
Multiple Choice



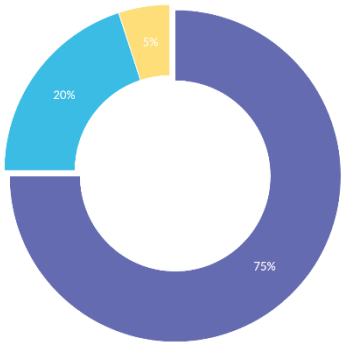
Choices	Totals
16-24	15
25-34	1
35-44	0
45+	2
Prefer not to say	0

Q2 Which of the following do you think describes you best when it comes to computer programming?
Multiple Choice



Choices	Totals
Beginner - I have never programmed before	4
Intermediate - I have tried out programming at home or at school but only a little	3
Experienced - I am studying programming at university/college	7
Expert - I am very confident in programming and do it as part of a university/college course or a job	6
Prefer not to say	0

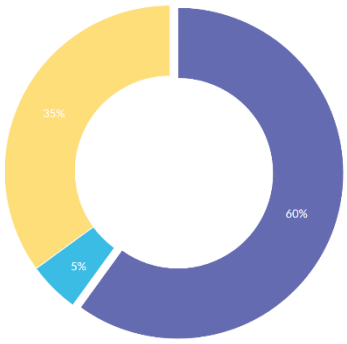
Q3 What would you consider to be your gender?
Multiple Choice



Choices	Totals
Male	15
Female	4
Other	0
Prefer not to say	1

Q4 If you are a Heriot-Watt University student, please select the campus you are currently studying at

Multiple Choice



Choices	Totals
Edinburgh	12
Dubai	1
Malaysia	0
I am not a Heriot-Watt student	7

Q5 TASK 1: FIND THE WEBSITE. Navigate to the LaTCh website at www2.macs.hw.ac.uk/~rs6/LaTCh/ , by entering the address into your web browser. You will need to have this page open as well as this survey...

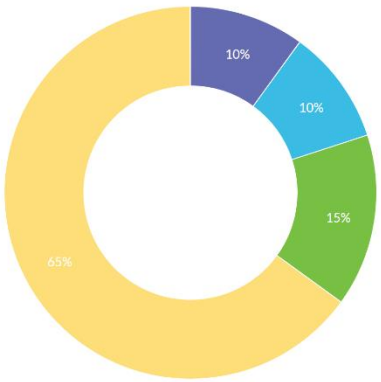
Essay

Popular Comments:

This task was easy
The URL is difficult to type in search bar.

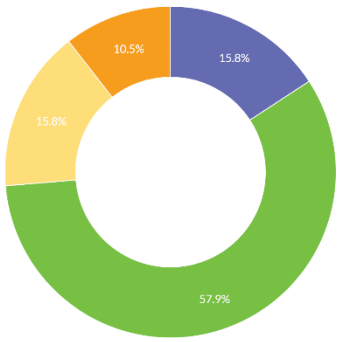
Q6 Which of the following devices are you testing the LaTCh website on?

Multiple Choice



Choices	Totals
Smartphone	2
Tablet	2
Desktop PC	3
Laptop	13
Other (please specify)	0

Q7 What internet browser are you using to carry out this usability study?
Multiple Choice



Choices	Totals
Safari (default on iPhone)	3
Microsoft Edge	0
Google Chrome	11
Mozilla Firefox	3
Other (please specify)	2

Q8 RATING 1: You will be asked to rate the user interface for certain pages on a scale of 1 to 10, with 1 meaning you really don't like the interface, and 10 meaning you loved it and found it easy to use...
Scale

3	7.72	10
Lowest	Average	Highest

Q9 TASK 2: QUESTIONNAIRE. From the homepage of LaTCh, please select the Questionnaire button in the midde of the screen (you may have to scroll on mobile), then select 'Begin Questionnaire'. Using this...
Essay

Popular comments:

Easy to complete.
The theme is a bit dark
A bit simple. Could be more complicated and informative for more experienced users.

Q10 RATING 2. Please rate the Questionnaire interface (shown above) from 1 to 10, with 1 meaning you really didn't like it and 10 meaning you really liked it. Additional comments are also appreciated.
Scale

2	7.94	10
Lowest	Average	Highest

Q11 TASK 3: CREATE A USER ACCOUNT AND LOGIN. Please return to the homepage of the LaTCh website by selecting 'LaTCh' in the top navigation bar. Select 'Login' at the top of the page and then select the...
Essay

Popular comments:

Easy task.
Should automatically redirect or give a more-friendly confirmation page when you create a new account.
Not secure shown in Chrome – could put users off using the system.

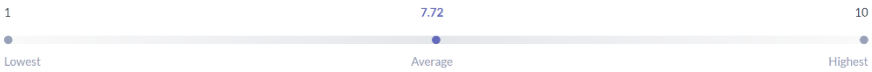
Q12 TASK 4: REPLY TO A THREAD. You will need to be logged in for this task. If you were unable to complete TASK 3, please say that you have skipped it in your comment below. If you are logged in to an...
Essay

Popular comments:

Should automatically redirect to thread after your reply has been sent.
Possibly have the reply on the same page.
Display date/time for replies.

Q13 RATING 3: Please rate the Discussion Board interface, including the thread and reply pages, on the same scale as before. Include any additional comments you feel may be useful.

Scale



Q14 TASK 5: CATEGORIES. Please return to the homepage (this can be done from anywhere on the website by clicking 'LaTCh' in the top menu bar). From here, find the 'Categories' page (button in middle of...

Essay

Popular comments:

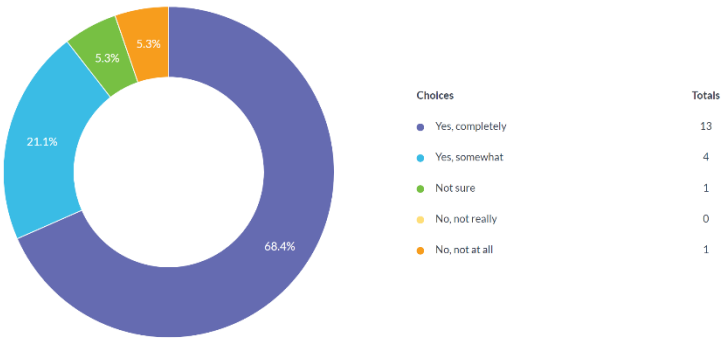
Buttons should all be a fixed size.
Not sure about buttons
More categories should be added.
Useful, especially for beginners.

Q15 RATING 4: Please rate the interface for the categories pages on a scale of 1 (worst) to 10 (best). Do you think this is a useful feature?

Scale

Q16 This website was created with the idea of being user friendly to all levels of programmers - including complete beginners. Do you feel LaTCh achieves this aim? Please leave additional comments if you...

Multiple Choice



Q17 Finally, is there anything you think is missing from this website that could be added? You are encouraged to have more of a browse around the LaTCh website if you wish.

Essay

Popular comments:

Return buttons should be added, for example inside a category.
More colour. Buttons should be a fixed size.
Search function.
Sub-forums to break up the discussion board into separate boards.
Expand by adding online learning resources.

Appendix 11 - Risk analysis

Risk	Description	Impact	Likelihood	Mitigation
Time Constraints	Not enough time to complete all of the defined requirements	Tolerable	High	Prioritised requirements using MoSCoW analysis.
Legal Issues	Issues relating to using someone else's source code or images and not making it clear when I am using someone else's work.	Catastrophic	Low	Made sure all work owned by another author was properly cited and didn't use someone else's work without their permission.
Ethical Issues	Not receiving the correct ethical approval for the project	Catastrophic	Low	Made sure the necessary ethical approval for the project was carried out before carrying out any surveys.
Compatibility	The product may not be compatible over all available browsers and platforms.	Tolerable	Low	Tested the product on as many browsers and platforms as possible; at least all the main browsers.
Workload	Inability to balance workload between this project and	Severe	Medium	Kept to the project plan as much as possible and adjusted it

	others on the course.			when necessary. Organised the available time and prioritised important tasks.
Data Loss	Corruption or loss of source code or other project related data.	Catastrophic	Low	Regularly backed-up all important project data both on an external USB drive and online.
Low survey response rate	Not enough participants in the survey to achieve interesting results.	Severe	Low	Advertised the survey as much as possible through email and social media.
Problems with working off-campus	Inability to work from home due to having different software installed or a different setup to systems like MySQL.	Severe	Medium	Set up all necessary software on personal and university machines as early as possible.
Not sticking exactly to the project plan	Not completing tasks on time or in the planned order.	Tolerable	Medium	Allowed the plan to be changeable. Made sure all the main tasks were completed on time.

Appendix 12 – Site Comparison Survey (BPLFM vs. LaTCh)

Scenarios:

SCENARIO 1: You have been asked by a local high school to come in and teach their **fourth-year students** a new programming language over 3 or 4 weeks. They are relatively new to programming but have learned some of the **basics** through the visual programming language **Scratch** and they have used **HTML** to create small websites. What language would you use for this task? Please use BPLFM and LaTCh to provide possible answers for this scenario and note them below.

SCENARIO 2: You have just started **working for a company** that develops **mobile apps**. Your boss says that she wants you to lead a project to create a new app that will help a user track and plan their holiday information including flights, hotel bookings, itinerary, etc. She has also mentioned that it should be created as an **Android** app to begin with. What programming language do you use for this project? Please use LaTCh and BPLFM to answer this question and note your answer(s) below.

SCENARIO 3: You have been asked by a local restaurant if you can create a **website** for them to advertise their business and to manage their bookings and provide customers with information. What programming language(s) do you use? Please use LaTCh and BPLFM for this task. Again, please leave your answers in the list below.

SCENARIO 4: You are in the process of creating your own, brand new programming language that you would like to eventually turn into a very powerful alternative to the current options. You need to create a **compiler** to translate this language into machine code. Please attempt to find an answer on LaTCh and BPLFM. What language do you use to create the compiler?

SCENARIO 5: With the rise of parallel computing, you have decided it might be a good idea to learn a programming language that is designed specifically to work with the **parallel paradigm**. Please use LaTCh and BPLFM to find a possible programming language for parallel programming. Once again, leave your answer below.

Time taken for each scenario:

Participant	Time Taken (S1)		Time taken (S2)		Time taken (S3)		Time taken (S4)		Time taken (S5)		
	LaTCh	BPLFM	LaTCh	BPLFM	LaTCh	BPLFM	LaTCh	BPLFM	LaTCh	BPLFM	Average:
P1	110	30	8	17	47	24	4	30	22	55	34.7
P2	18	14	6	14	20	35	9	35	11	30	19.2
P3	21	16	7	25	19	29	4	18	8	25	17.2
Average:	49.66667	20	7	18.66667	28.66667	29.33333333	5.666667	27.66666667	13.66667	36.66667	

Each User's preferred site for each scenario (L – LaTCH, B – BPLFM, L/P – Both):

Favourites	P1	P2	P3
S1	B	L	L
S2	L	L/B	L
S3	L	L	L
S4	L	L	L
S5	L	L	L

Which site(s) the participant found an answer on for each scenario:

Answer on	P1	P2	P3
S1	L/B	L/B	L/B
S2	L/B	L/B	L/B
S3	L/B	L/B	L/B
S4	L	L	L
S5	L	L	L

Statements for Likert scale:

1. Allowing users to post queries online anonymously allows them to ask questions more freely, with less chance of bullying or abuse.
2. Having multiple features to help you choose a programming language is much better than having only one feature as it allows for all sorts of users to be accounted for.
3. It is better to find a programming language in a few, less specific questions than a larger number of more specific questions.

Likert Scale Ratings:

Likert:	P1	P2	P3
1	5	3	3
2	5	4	4
3	4	4	3

Appendix 13 – Site Comparison Survey Consent Form

Consent Form – Site Comparison Survey

LaTCh (Language and Technology Chooser) vs. BPLFM (Best Programming Language for Me).

Consent to Act as a Subject in an Experimental Study

Principal Investigator: *Ronan Smith*

Description: The purpose of this study is to compare two similar websites and to find out, based on a few scenarios, which one (if any) works better than the other. You will be asked to carry out a few scenario tasks, where you will imagine you are the user being described. You are asked to attempt to solve each of the scenario problems twice – once using LaTCh and once using BPLFM. You will be timed on these tasks. After each scenario, you will be asked to note down which website you felt helped you solve the task the best. At the end of the survey you will be asked to indicate how much you agree with a few statements about the LaTCh website on a scale of 1 to 5.

There are minimal risks for you to participate in this study. All personal information will be kept confidential in a secure filing cabinet or in password-protected computer directories in accordance with the provisions of the Data Protection Act 1998. Please remember that there are no right or wrong answers. It is the usability of the systems mentioned that is being tested, not you.

You are free to decline to participate in this study. Should you decide to participate, you are free to end your participation at any time. Such a decision by you will not adversely affect or alter your status with the investigator in any way. You are also free to withdraw your data at any time after taking part. If you wish to do this, please contact Ronan Smith via email (rs6@hw.ac.uk). If you withdraw your data will be removed and destroyed.

Participation voluntary consent: I certify that I have read the preceding and that I understand its contents. Any questions I have pertaining to the research have been answered satisfactorily by the investigator. My signature below means that I have freely agreed to participate in this study.

Date: _____ Subject Signature: _____

.....

Investigator's certification: I certify that I have explained to the above individual the nature and purpose, the potential benefits, and possible risks associated with participation in this research study, have answered any questions that have been raised, and have witnessed the above signature.

Date: _____ Investigator Signature: _____ Subject No. _____