Development of a map-based web application showcasing the Dyfi Wildlife Centre

CS39440 Major Project Report

Author: Michael Male (mim39@aber.ac.uk)
Supervisor: Dr Edel Sherratt (eds@aber.ac.uk)

4th May 2020
Version: 0.2 (Draft)

This report was submitted as partial fulfilment of a BSc degree in Computer Science (includes Foundation Year) (G40F)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, U.K.

Declaration of originality

I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the regulations on Unacceptable Academic Practice from the University's Academic Registry (AR) and the relevant sections of the current Student Handbook of the Department of Computer Science.
- In submitting this work I understand and agree to abide by the University's regulations governing these issues.

Name Michael Male

Date 04/04/2020

Consent to share this work

By including my name below, I hereby agree to this project's report and technical work being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Name Michael Male

Date 04/04/2020

Acknowledgements

I am grateful to Mr Emyr Evans and Mr Thomas Faulkner of the Montgomeryshire Wildlife Trust for their support in the development of this project, and hope that the final product is useful to them.

I'd like to thank the Department of Computer Science for their provision of support and resources, particularly from a remote perspective during the COVID-19 pandemic in the second half of the project. Specifically, I'd like to thank my supervisor, Dr Edel Sherratt, for her help, support and encouragement during all stages of this project, as well as Mr Richard Shipman, who provided useful feedback in the mid-project demonstration that ensured it was on the right tracks.

Abstract

The Dyfi Wildlife Centre is a visitor centre run by the Montgomeryshire Wildlife Trust. It is situated on the Cors Dyfi Nature Reserve in Powys, Wales. The Trust had approached the University, with a request for a solution to assist in showcasing the reserve's work, and its place as an osprey conservation, engagement, and research project. They had procured an 86-inch touchscreen monitor, in the hopes of presenting an application that aids in achieving the aforementioned goal.

The software created in this project takes the form of a map-based web application. The frontend makes use of the Google Maps for JavaScript API, as well as HTML, CSS and Thymeleaf, to show visitors a map of the centre and its surroundings. The map has various markers, and filters, that can be clicked on, showing further information about the point of interest. This is backed up by a RESTful API backend, created using the Spring Framework, a model-view-controller framework using Java Enterprise Edition. An administration panel was also developed, involving authentication, and allowing the authenticated user to add, edit, and delete points of interest. Data was stored in a PostgreSQL relational database.

Planning and development of this project took the form of an agile approach, with ideas from both Kanban and Extreme Programming used and adapted to fit a single-developer project. A meeting with the customer took place prior to development, and user stories and prioritisation of tasks had branched out from that meeting. The concept of test-driven development was a driving force for the development of this project, with various testing suites including JUnit, HTMLUnit and Selenium utilised when creating test cases. A key part of this project was ensuring that the customer was kept up-to-date, and supplementary documentation was created for this project that provided a brief overview of the required setup and how to use the web app.

Contents

List of Figures

List of Tables

Chapter 1

Background & Objectives

1.1 Background

A number of key considerations were taken into place during the background planning stages of this project.

An initial meeting took place in February 2020 with the customer, the Montgomeryshire Wildlife Trust. The hardware implementation was discussed and it was understood that the customer would want to run the web application on an 86-inch touch screen monitor. They also expressed a preference for the web application to be run locally, and not published to a cloud service or the World Wide Web. Therefore, an important consideration of the project was enabling the web application to work using touch gestures, and use a responsive layout that would scale to a high resolution.

With the intention being that the application runs locally, therefore research had to be put into various technology stacks, and which web framework would work best for this kind of application. Research into this would have to take into account how much time needs to be allocated to performing spike work; ensuring a comprehensive understanding of the framework was achieved before work on the project began.

Another key consideration was the vendor for the maps API. A number of APIs were assessed for their usability as well as their licensing conditions.

In this section, the early investigative work into the project will be discussed, with an analysis of the steps taken to achieve a conclusion as to which technology stack to use.

1.1.1 Web Development Frameworks and Tools

Early on in the project it was decided that the use of a comprehensive web framework would be beneficial to the usability and production quality of the web application. This was opposed to a simple HTML 5, CSS, and JavaScript stack. Whilst it was inevitable that portions of each language had to be used, a more robust framework provided specific advantages, such as a package manager and cleaner code.

Three primary web frameworks were assessed, through the use of research, prior reading, and spike work, these being the following:

- A JavaScript-based framework, utilising the Node.js and Express backend frameworks and either Vue.js or React as the frontend.
- Django, a Python-based model-template-view framework.
- The Spring Framework, a Java-based model-view-controller framework.

Further research took place into the CSS frameworks, as well as an appropriate database management system. Discussion into these are expanded upon in their relevant sections.

1.1.1.1 JavaScript Frameworks

A full-stack JavaScript framework allowed for the benefit of the software being written in one programming language, which could reduce issues with the code's readability and maintenance. It would have also allowed for a single testing framework, such as Jest, a testing framework maintained by Facebook. The use of a runtime environment would be standard fare for a JavaScript framework, allowing sever-side scripting and dynamic web pages to be run outside of the usual web browser environment that JavaScript runs on. A popular JavaScript runtime is Node.js, which has been touted as a resource-efficient framework, a benefit for a project that is designed to run on a local computer.

Frontend frameworks were also looked at, with varying levels of spike work being put into them. React, a Facebook-maintained JavaScript library, and Vue.js, were both considered. The two frameworks are rather similar, such that they rely on sending data directly to the browser's Document Object Model, however Vue.js takes a declarative approach to HTML scripting, whilst React uses JSX, an HTML syntax extension to JavaScript.

Ultimately, a lack of familiarity with JavaScript, as well as the relative complexities of the frameworks, proved to be a deciding factor in not going ahead with a JavaScript-driven application. During research, it was deemed that a large amount of time would have to be dedicated to following tutorials and learning JavaScript, and ECMAScript 6, from scratch, and this would have taken too much time and risked a less complete final product.

1.1.1.2 Django

Django is an open-source framework based on the Python programming language. Its creators set the framework's primary philosophies as a quick approach to development, a view to not repeating the design and execution of concepts, and loose coupling - in this context being that the framework layers shouldn't be able to interface with eachother unless necessary.

Django utilises the object-oriented programming paradigm with Python, and utilises a model-template-view approach. In short, these are three distinct layers of a web application: the model consists of a data structure, the view consists of a representation of

information on a web browser, and a controller allows access to this data with meaningful requests.

Whilst Django appeared to be a good choice due to its highly cohesive pattern, testing of the setup proved to be difficult on occasion, with design patterns that were particularly unique to Django. Whilst Python is a language that is known to utilise concepts that are simple to understand for users with greater knowledge in other programming languages, utilisation of Python at this level required a higher level of expertise than was had at the time. It was decided that there would be a greater chance of success with the project if attention was placed more towards frameworks with a more familiar language, to avoid an inordinate amount of time being spent on the intricacies of specific programming languages and frameworks.

1.1.1.3 Spring Framework

The Spring Framework is an open-source framework, where its web application features are based upon Java Enterprise Edition, an enterprise specification of the Java programming language that has modules specifically tailored towards web services.

Spring provides many of the benefits that Django also provides, and the two are often compared against eachother. Similar to Django, Spring utilises a model-view-controller framework, and relies upon high cohesion. As Spring uses Java, it takes advantage of the object-oriented programming paradigm, and it is standard for classes to be written in such a way that they follow this concept. Spring is optimised to work with the Thymeleaf template engine, that provides server-side scripting and an interface between the controller and view layers.

Ultimately, a proficiency in Java from previous academic study and personal use made Spring an ideal choice for this project. Spring Boot, an addition to the platform that allows for automatic configuration of core dependencies, was also used, to reduce the amount of time spent studying elements of Spring that Spring Boot renders redundant. Spring also utilises Maven, a Java package manager, to provide a large number of dependencies; Spring Security was deemed a useful tool for an authentication layer, for example. An added benefit to Spring is its embedded Apache Tomcat server, that allows for the provision of an HTTP web server environment from opening the application, rather than having to take steps to deploy it into an existing web server environment.

1.1.1.4 CSS frameworks

It was decided in the planning stages of the project that it would be beneficial to use a CSS framework, rather than build a template with 'vanilla' implementations of HTML and CSS. CSS frameworks provide a large number of pre-built elements, many that are rather familiar to users, due to their prevalence in front-end web design. A review into three different CSS frameworks were performed.

Bootstrap, a framework initially developed for use with Twitter, provides elements that have been crafted with the User Experience at their forefront. It is used by a large variety of

web applications and websites, with the developers claiming it is 'the world's most popular front-end component libary.' Whilst this would have been a good choice for a familiar user interface, the framework was assessed to have less customisability, which posed an issue with a unique implementation where the main aspect is a single-page application. Bootstrap is also intended to be mobile-first, a feature that is not required, as the application is designed to run on a desktop computer.

Fomantic UI, a community fork of Semantic UI, which had seen a lull in development, is a framework that defines itself as using 'human-friendly HTML.' Classes within Fomantic use syntax from the English language, for example, a user interface with three buttons could be classed simply as `<div class="ui three buttons">`. While Semantic is quite an elegant interface, different frameworks were deemed more familiar to a user, and the User Experience aspect of this project was tailored towards people who may not have a great amount of technical knowledge.

The chosen CSS framework for the application was Materialize, a variation upon Google's Material Design language. Material Design is used in a large number of Android mobile phone applications, and on Google's services themselves. Similar to Fomantic, Materialize utilised the concept of human-friendly HTML, and was easy to integrate with Thymeleaf and Spring. Materialize utilises a twelve-column responsive grid system, which made it simple to create components that would scale with screen size. As the customer intends to run the application on a large monitor, this is an important design consideration.

1.1.1.5 Database Management System

A database management system was a key component of this project. Points of Interest had to be persistent, and, in later iterations, a database for users had to be created. Whilst an in-memory database management system, H2, was used during the early stages of the project, it was quickly settled upon that a server independent from the application should be created. The investigation into this settled on either using PostgreSQL, an SQL-compliant system written in C, or MongoDB, a NoSQL document-oriented database that takes a JSON-like approach to storage of data.

It was decided on to utilise PostgreSQL. Whilst there were various arguments for using MongoDB - a more readable schema, for example, studies have shown that PostgreSQL is generally faster in response times, and it is a popular implementation of SQL that sees good compatibility with Spring. A solid amount of background experience with PostgreSQL also contributed to the decision; whilst MongoDB has been described by its developers as similar to an object-oriented paradigm, the performance benefits of PostgreSQL negated any benefits of learning a new system.

1.1.1.6 Programming Tools

After the technology stack was decided, a decision had to be made as to the best programming tools for the task at hand. The Spring Framework has widespread support within IDEs. An extension, called Spring Tools 4, had been created by the developers, which provided Spring-related functionality to Eclipse, Microsoft Visual Studio Tools, and

Eclipse Theia. A large number of PostgreSQL clients existed, with some example being pgAdmin 4 and HeidiSQL. PostgreSQL could also be administrated via a command-line interface.

However, it was decided upon that the JetBrains' suite of software was to be used for development. JetBrains provides a free license to people with a University e-mail address, and its IntelliJ IDEA Java IDE has in-built support for Spring Boot projects, and also includes plugins for web development. DataGrip, a database IDE, was also used, as this allowed for a graphical representation of the database that resulted in the schema being easier to quickly understand. Various browsers, including Microsoft Edge, Google Chrome, and Mozilla Firefox, were used in order to test the web application on varying browsers. Mozilla's Firefox Browser Developer Edition was useful when debugging the web app, as it included useful tools pertinent to CSS and JavaScript debugging.

1.1.2 Mapping API

A crucial part of the application was the ability to present geographic data and information in a graphical format. A video had been provided by the customer, showing a map implementation on a similar nature reserve in Dorset. This used a satellite map with markers placed on the screen, and the intention was to take a similar approach.

As this application is web-based, a view towards a JavaScript API was adopted when assessing potential mapping solutions. The map must also take clickable markers and allow for quick downloading of satellite images. Two potential candidates were assessed:

- **OpenStreetMap** - A free and open-source map that allowed for users to request changes to be made, which has the potential to provide a more up-to-date map based on changes in geography. OpenStreetMap does not have an inbuilt satellite implementation or an API, however various free and paid sources are available and were assessed during this stage of the project.
- **Google Maps Platform** - Google provides a JavaScript API with a wide array of features attached to it, that uses data and imagery from Google Maps, hosted on Google's infrastructure. A potential hurdle in using Google Maps Platform, however, was its use of API keys and its pricing structure, an issue that is further discussed below.

1.1.2.1 OpenStreetMap

OpenStreetMap is defined as a community-driven repository of map data, that can be contributed towards in a 'wiki-like' manner. It did not, however, appear to provide a JavaScript API, but many implementations of OpenStreetMap have been realised in JavaScript, with Leaflet being one of them.

OpenStreetMap did not appear to host aerial imagery, and freeware JavaScript APIs only appear to have implemented a standard map interface. It was considered that this is not what the customer had wanted, and, whilst Mapbox, a closed-source implementation

of OpenStreetMap with aerial imagery and a free tier, was briefly assessed, a lack of customisability and the low-quality resolution of the aerial imagery were deciding factors in not going ahead with this approach.

1.1.2.2 Google Maps Platform

Google Maps Platform is one of the features offered with Google Cloud; a suite of cloud-based applications and APIs hosted centrally by Google. Google Maps is of course a popular mapping interface, and regular input from various countries' national mapping agencies allows it to be reasonably up-to-date. Google Maps Platform's main offering for non-mobile applications is the Maps JavaScript API, and a thorough amount of documentation has been provided for this.

Ultimately, a decision was made as to go ahead with using Google Maps Platform. The platform's high-resolution aerial imagery was key to this decision, along with the general reliability and uptime of Google's Cloud infrastructure. Unlike OpenStreetMap, the platform was not free, however includes a free tier that permits for up to USD $200 of free usage a month. After reviewing their pricing scheme it was decided that this would be enough for a locally-hosted application; the geo-coding API was not being used and a levy of USD $2 was placed on every one thousand static map requests, meaning that one hundred thousand requests a month would have to be made to exceed the free tier, which is not likely.

1.2 Software Development Process

1.2.1 Agile Development

1.2.1.1 Kanban

1.2.1.2 eXtreme Programming

1.2.2 Waterfall Model

1.2.3 Test-driven development

1.3 Analysis of key areas

1.3.1 Hosting

1.3.2 Authentication

1.3.3 Customer Requirements

Chapter 2

Design

You should concentrate on the more important aspects of the design. It is essential that an overview is presented before going into detail. As well as describing the design adopted it must also explain what other designs were considered and why they were rejected.

The design should describe what you expected to do, and might also explain areas that you had to revise after some investigation.

Typically, for an object-oriented design, the discussion will focus on the choice of objects and classes and the allocation of methods to classes. The use made of reusable components should be described and their source referenced. Particularly important decisions concerning data structures usually affect the architecture of a system and so should be described here.

How much material you include on detailed design and implementation will depend very much on the nature of the project. It should not be padded out. Think about the significant aspects of your system. For example, describe the design of the user interface if it is a critical aspect of your system, or provide detail about methods and data structures that are not trivial. Do not spend time on long lists of trivial items and repetitive descriptions. If in doubt about what is appropriate, speak to your supervisor.

You should also identify any support tools that you used. You should discuss your choice of implementation tools - programming language, compilers, database management system, program development environment, etc.

Some example sub-sections may be as follows, but the specific sections are for you to define.

2.1 Overall Architecture

2.2 Some detailed design

2.2.1 Even more detail

2.3 User Interface

2.4 Other relevant sections

Chapter 3

Implementation

The implementation should discuss any issues you encountered as you tried to implement your design. During the work, you might have found that elements of your design were unnecessary or overly complex; perhaps third-party libraries were available that simplified some of the functions that you intended to implement. If things were easier in some areas, then how did you adapt your project to take account of your findings?

It is more likely that things were more complex than you first thought. In particular, were there any problems or difficulties that you found during implementation that you had to address? Did such problems simply delay you or were they more significant?

You can conclude this section by reviewing the end of the implementation stage against the planned requirements.

Chapter 4

Testing

Detailed descriptions of every test case are definitely not what is required here. What is important is to show that you adopted a sensible strategy that was, in principle, capable of testing the system adequately even if you did not have the time to test the system fully.

Provide information in the body of your report and the appendix to explain the testing that has been performed. How does this testing address the requirements and design for the project?

How comprehensive is the testing within the constraints of the project? Are you testing the normal working behaviour? Are you testing the exceptional behaviour, e.g. error conditions? Are you testing security issues if they are relevant for your project?

Have you tested your system on “real users”? For example, if your system is supposed to solve a problem for a business, then it would be appropriate to present your approach to involve the users in the testing process and to record the results that you obtained. Depending on the level of detail, it is likely that you would put any detailed results in an appendix.

Whilst testing with “real users” can be useful, don’t see it as a way to shortcut detailed testing of your own. Think about issues discussed in the lectures about until testing, integration testing, etc. User testing without sensible testing of your own is not a useful activity.

The following sections indicate some areas you might include. Other sections may be more appropriate to your project.

4.1 Overall Approach to Testing

4.2 Automated Testing

4.2.1 Unit Tests

4.2.2 User Interface Testing

4.2.3 Stress Testing

4.2.4 Other types of testing

4.3 Integration Testing

4.4 User Testing

Chapter 5

Evaluation

Examiners expect to find a section addressing questions such as:

- Were the requirements correctly identified?
- Were the design decisions correct?
- Could a more suitable set of tools have been chosen?
- How well did the software meet the needs of those who were expecting to use it?
- How well were any other project aims achieved?
- If you were starting again, what would you do differently?

Other questions can be addressed as appropriate for a project.

The questions are an indication of issues you should consider. They are not intended as a specification of a list of sections.

The evaluation is regarded as an important part of the project report; it should demonstrate that you are capable not only of carrying out a piece of work but also of thinking critically about how you did it and how you might have done it better. This is seen as an important part of an honours degree.

There will be good things in the work and aspects of the work that could be improved. As you write this section, identify and discuss the parts of the work that went well and also consider ways in which the work could be improved.

In the latter stages of the module, we will discuss the evaluation. That will probably be around week 9, although that differs each year.

Appendices

The appendices are for additional content that is useful to support the discussion in the report. It is material that is not necessarily needed in the body of the report, but its inclusion in the appendices makes it easy to access.

For example, if you have developed a Design Specification document as part of a plan-driven approach for the project, then it would be appropriate to include that document as an appendix. In the body of your report you would highlight the most interesting aspects of the design, referring your reader to the full specification for further detail.

If you have taken an agile approach to developing the project, then you may be less likely to have developed a full requirements specification. Perhaps you use stories to keep track of the functionality and the 'future conversations'. It might not be relevant to include all of those in the body of your report. Instead, you might include those in an appendix.

There is a balance to be struck between what is relevant to include in the body of your report and whether additional supporting evidence is appropriate in the appendices. Speak to your supervisor or the module coordinator if you have questions about this.

Appendix A

Third-Party Code and Libraries

If you have made use of any third party code or software libraries, i.e. any code that you have not designed and written yourself, then you must include this appendix.

As has been said in lectures, it is acceptable and likely that you will make use of third-party code and software libraries. If third party code or libraries are used, your work will build on that to produce notable new work. The key requirement is that we understand what your original work is and what work is based on that of other people.

Therefore, you need to clearly state what you have used and where the original material can be found. Also, if you have made any changes to the original versions, you must explain what you have changed.

The following is an example of what you might say.

Apache POI library - The project has been used to read and write Microsoft Excel files (XLS) as part of the interaction with the client's existing system for processing data. Version 3.10-FINAL was used. The library is open source and it is available from the Apache Software Foundation [**?**]. The library is released using the Apache License [**?**]. This library was used without modification.

Include as many declarations as appropriate for your work. The specific wording is less important than the fact that you are declaring the relevant work.

Appendix B

Ethics Submission

This appendix includes a copy of the ethics submission for the project. After you have completed your Ethics submission, you will receive a PDF with a summary of the comments. That document should be embedded in this report, either as images, an embedded PDF or as copied text. The content should also include the Ethics Application Number that you receive.

Appendix C

Code Examples

For some projects, it might be relevant to include some code extracts in an appendix. You are not expected to put all of your code here - the correct place for all of your code is in the technical submission that is made in addition to the Project Report. However, if there are some notable aspects of the code that you discuss, including that in an appendix might be useful to make it easier for your readers to access.

As a general guide, if you are discussing short extracts of code then you are advised to include such code in the body of the report. If there is a longer extract that is relevant, then you might include it as shown in the following section.

Only include code in the appendix if that code is discussed and referred to in the body of the report.

3.1 Random Number Generator

The Bayes Durham Shuffle ensures that the psuedo random numbers used in the simulation are further shuffled, ensuring minimal correlation between subsequent random outputs [**?**].

```
#define IM1 2147483563
#define IM2 2147483399
#define AM (1.0/IM1)
#define IMM1 (IM1-1)
#define IA1 40014
#define IA2 40692
#define IQ1 53668
#define IQ2 52774
#define IR1 12211
#define IR2 3791
#define NTAB 32
#define NDIV (1+IMM1/NTAB)
#define EPS 1.2e-7
```

```
#define RNMX (1.0 - EPS)

double ran2(long *idum)
{
  /*-------------------------------------------------*/
  /* Minimum Standard Random Number Generator        */
  /* Taken from Numerical recipies in C              */
  /* Based on Park and Miller with Bays Durham Shuffle */
  /* Coupled Schrage methods for extra periodicity   */
  /* Always call with negative number to initialise  */
  /*-------------------------------------------------*/

  int j;
  long k;
  static long idum2=123456789;
  static long iy=0;
  static long iv[NTAB];
  double temp;

  if (*idum <=0)
  {
    if (-(*idum) < 1)
    {
      *idum = 1;
    }else
    {
      *idum = -(*idum);
    }
    idum2=(*idum);
    for (j=NTAB+7;j>=0;j--)
    {
      k = (*idum)/IQ1;
      *idum = IA1 *(*idum-k*IQ1) - IR1*k;
      if (*idum < 0)
      {
        *idum += IM1;
      }
      if (j < NTAB)
      {
        iv[j] = *idum;
      }
    }
    iy = iv[0];
  }
  k = (*idum)/IQ1;
  *idum = IA1*(*idum-k*IQ1) - IR1*k;
  if (*idum < 0)
  {
```

```
    *idum += IM1;
  }
  k = (idum2)/IQ2;
  idum2 = IA2*(idum2-k*IQ2) - IR2*k;
  if (idum2 < 0)
  {
    idum2 += IM2;
  }
  j = iy/NDIV;
  iy=iv[j] - idum2;
  iv[j] = *idum;
  if (iy < 1)
  {
    iy += IMM1;
  }
  if ((temp=AM*iy) > RNMX)
  {
    return RNMX;
  }else
  {
    return temp;
  }
}
```