**Turitea Web Resources**

**Week 4 Report**

**(10.08.2019)**

|  |  |
| --- | --- |
| **To:** | Curtis Preston  Lance Gray  Xingyu Chen |
| **By:** | Prof. Amjed Tahir |
| **Paper:** | 159.356 Software Eng. Capstone Project |

Table of Contents

[1 Introduction 3](#_Toc16359573)

[2 Process 4](#_Toc16359574)

[2.1 Documenting Requirements 4](#_Toc16359575)

[2.2 Project Infrastructure 5](#_Toc16359576)

[2.3 Life Cycle Model & Project Plan 6](#_Toc16359577)

[2.4 Technology Selection 7](#_Toc16359578)

[2.5 Architecture / High Level Design 9](#_Toc16359579)

[2.6 Risk Management 11](#_Toc16359580)

[3 Quality Assurance 15](#_Toc16359581)

[3.1 Testing 15](#_Toc16359582)

[3.2 Issue Tracking 16](#_Toc16359583)

[4 Product 18](#_Toc16359584)

[5 References 19](#_Toc16359585)

[6 Appendices 20](#_Toc16359586)

[6.1 Use Case Diagrams 20](#_Toc16359587)

[6.2 Webpage Wire Frames 24](#_Toc16359588)

# Introduction

The basic scope of this project is to produce a Web Application to provide educational and scientific information about the ecology of the Turitea Stream in New Zealand. The Project Sponsor is Steve Trewick of Massey Universiy, and the software development team members include Curtis Preston (Team Leader), Lance Gray, and Xingyu Chen.

This report provides a brief summary the team's progress after four weeks.

# Process

## Documenting Requirements

The scope of this project includes creating a web application that provides the following in order of priority:

* General Information:
  + Textual and graphical information containing details about the Project Sponsor and the objectives of the website; and,
  + Textual and graphical information about the Turitea Stream.
* 3D Global Map:
  + Graphical image which provides a visually appealing 3D map of the planet.
  + Ability to navigate, rotate, and zoom-in to any location.
  + Ability to see and touch graphic data points (or pins) which provide ecological data at a specific global location; and,
  + Ability to add, edit and remove pins.
* Simple Blog for ecological data and images:
  + Provide a WYSIWYG rich text and graphic editor to provide Blog-type information;
  + Ability to add, edit and remove text, images, tables and graphs; and,
  + Editable only by the administrator (not open to general visitors).
* Feedback form:
  + Basic form for submitting feedback; and,
  + Allow visitors to submit their name, email address, and a feedback message.
* Administrator panel:
  + To allow an administrator user to add, edit and remove data;
  + Change website preferences;
  + Edit the 3D map;
  + Write Blog posts;
  + View visitor data; and,
  + Edit user credentials.
* Web-Developer panel:
  + A panel to manage and edit all other accounts.
* User access:
  + Any visitor of any age is permitted to use the web resources; however, the content should not include childish material that would only be suitable for a child;
  + Accounts are only required for administrators, not general visitors.

The information given above is summarised in the use case diagrams given in Appendix 6.1. The webpages associated with this web application shall be able to be rendered on any computer or mobile device.

Anything not specifically detailed above is excluded from this project. Also, all textual content, images and scientific data required for the web application, are to be supplied by the Project Sponsor.

## Project Infrastructure

In this project we are using **GitHub** for code management and sharing. At the heart of GitHub is **Git**. This is an Open Source application that manages and stores revisions of projects. **Git** allows the project team to download and edit a distributed clone of the software files, and then upload modifications. The project team is using the default **GitHub** issue tracker and a Kanban board for tracking current sprint tasks. The decision to use **GitHub** was fundamentally based on the fact that the team members were already familiar with using it, and liked its functionality.

**Tavis CI** will be used as our continuous integration tool for automated builds and testing since it seamlessly integrates with **GitHub**.

A **Docker** container will be used to deploy the web application. This is a lightweight and standalone executable package that packages up program code, tools, libraries, and all dependencies required so that the application can run quickly and reliably from one operating environment to another. The executable package (file) can then be launched using a shell script to start the Docker instance.

## Life Cycle Model & Project Plan

The Software Development Life Cycle model that will be employed is the **Agile Model**. The Agile model was developed by a consortium of people who met one weekend in 2001 to voice their frustrations by the current state of affairs with software development (Agile Manifesto, n.d.). As a result, the Agile Manifesto emerged which was short document that was instrumental in changing software development from then on. Since its creation, the Manifesto have been used by a wide variety of companies and organisations, and has proven itself as being a successful model for software development. It is for this reason, that its framework will be used with this project.

For this project **Scrums** will be used, which forms an integral part of the Agile framework. Scrums will be used to break the project work into actions that can be completed within timeboxed iterations (or Sprints). Although the project team is relatively small, it is still believed that Sprints can still be employed effectively. At this stage it is estimated that each Sprint cycle will take 2 weeks. Given the small size of the project team, the Scrum Master will be the project Team Leader for all Sprints.

A **Kanban** will be used to delegate tasks for each Sprint, and an issue tracker will be used to report on issues and their solutions for each task. **GitHub** has tools to integrate both of these features. A sample of the Kanban for the first Sprint is shown below:

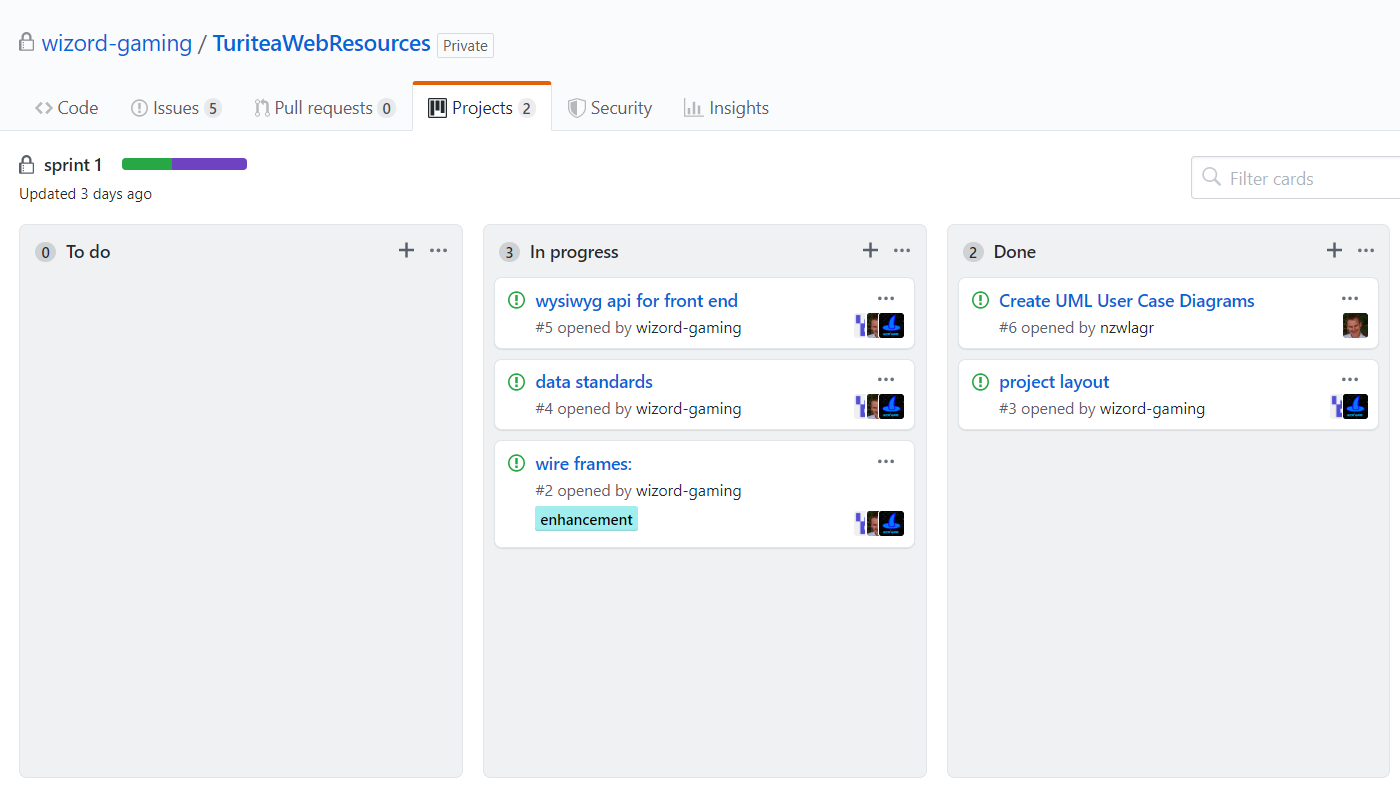


Figure 2.3.1: Sprint 1 KanBan (GitHub)

## Technology Selection

The **Go** Open Source programming language will be used for the backend of the web application. One of our team members has considerable experience with this programming language and expressed a desire to use it. The programming language has the following key benefits:

1. High-level compiled language;
2. Relatively simple programming language to learn;
3. Culture of keeping code simple, concise and efficient;
4. Native support for co-routines and channels;
5. Powerful network programming libraries (NET and officially maintained X/NET package);
6. Suitable for high-concurrency;
7. High-traffic modern server programming tasks;
8. Build-in testing and profiling libraries; and,
9. Contains numerous off-the-shelf packages and frameworks that can be used to accelerate the back-end development.

Due to its succinct syntax and high development efficiency, it is suitable for small teams for rapid development. Moreover, the language was developed and is maintained by Google. As a developer of the HTTP2.0 protocol, Google has extensive experience in today's network programming, and has designed good abstractions for the parts of the programming language that involve the network.

The frontend will be utilising web pages displayed using any HTML5 compliant web browser. The standard mark-up languages used with these browsers include **HTML**, **CSS** and **JavaScript**.Because these languages are universally used for other web applications, they will also be used to develop the frontend of this application. To ensure the frontend complies with the W3C standard, all web pages will be checked using the W3C validation website. The frontend will also include an advanced Style Sheet to allow the webpages to be displayed on a mobile device.

A number of different API's will be used to help accelerate the frontend development:

* **Cesium.js** is an Open Source JavaScript library, and it will be used to provide the 3D globes and maps, and the pins that will be attached it. Our team has already explored using this API, and have found it reasonably easy to use.
* **Plotly.js** is an Open Source JavaScript library that automates the generation of interactive graphs and charts within a webpage. We are still not sure yet whether or not this will required, as we may yet allow the administrator to add their own graphs and charts as pictures using a WYSIWYG API.
* We are still investigating a simple and flexible WYSIWYG API to allow the administrator to add and edit webpage Blog articles. At this stage we are still evaluating **Wiki.js**, **Trumbowyg**, **Summernote**, **SimEditor**, **CKEditor** and **TinyMCEz**.

**PostGreSQL** is an Open Source database, and will be used for data storage and data logic.

At this stage it is thought that all data files will be stored in one folder and all images in another. A file and image table will then be set inside a SQL database that maps URLs to filenames or relative paths.

## Architecture / High Level Design

The project will use a URL path and parameters with JSON to communicate between the front and backends of the web application. The **Ajax** JavaScript library will be used to send and retrieve data from the server asynchronously.

The current specification uses a JSON data format for the delivery of pin information (visible in **GitHub**). Other protocols should be abbreviated as the attribute name or attribute name of the various types of objects on the server side. The value is the corresponding value in JSON, such as passing the same type. Multiple values are transmitted as an array of objects. An array is only a homogeneous object. There must be no more than one type of object in the same JSON file. If a certain attribute value in the object is empty, or the value cannot be determined, the key value is ignored.

The current service path is as follows:

|  |  |
| --- | --- |
| **URL** | **What is it for?** |
| / | HTML files will be in the folder /. |
| /api | All dynamic content for the API HTML files. |
| /api/articles/ | Return *all* articles. |
| /api/articles/id | Return *single* article information for the selected id. |
| /api/pins/ | Return *all* pin data. |
| /api/pins?north=&sounth=&east=&west=; | Return *all* pin information in a given area. |
| /api/addPins?num=\*\* | Use the HTML Post method to upload a JSON file and set the value of parameter name **num** to the number of pins in the JSON file. The maximum value of **num** is 216-1. |

The interaction between the server and the persistence layer can be found in the SQL definition, or by using its binary representation for objects stored with files for easy access.

Server modules are passed between the use of defined type objects or immediate data, and services are accelerated through caching, asynchronous IO, and preloading techniques. For a user login system, an "almost" stateless server is created by using cookies on the browser side and caching their permissions on the server side.



Figure 2.5.1: Project High-Level Architecture

The database needs to store article data, pin data, user accounts, and maps for media files. An article may have multiple pins, and a pin may be linked to multiple articles. User accounts will have different permission levels, and the map for the media files will have to identify where the file is on the web and the type of media in order to identify what HTML tags to use and who uploaded it.

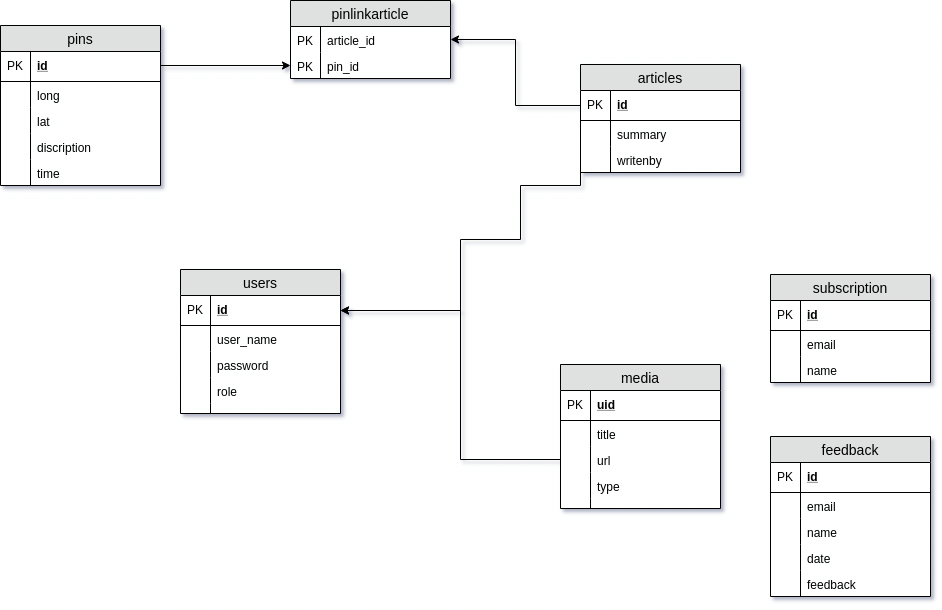


Figure 2.5.2: Database Relationship Map

## Risk Management

Prince2® (Projects IN controlled Environments) is a project management philosophy which is founded on the experience and contributions of countless individuals involved in thousands of projects (Murray, et al., 2009). This philosophy - and others suggested by authors such as Pinto (Pinto, 2013) - emphasises the importance of the following key sequential steps in minimising the risks associated with a project:

1. identify risks and threats for a specific project;
2. asses the identified risks and score them appropriately using a probably and consequence risk matrix;
3. develop a plan to respond to the identified risks; and,
4. implement the plan and audit its effectiveness.

The steps are iterative in nature in that they are repeated if new information becomes available about the project. In parallel to the steps, the philosophy emphasises the importance of continually communicating with the key stakeholders as pictured below:

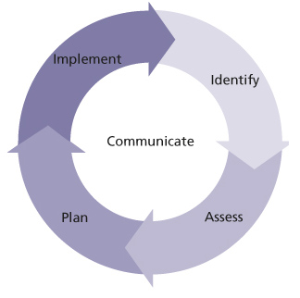


Figure 2.6.1: Risk Management Procedure (Murray, et al., 2009, p. 80)

Using the philosophy above, the following risk matrix was constructed which articulates the *consequence* and *likelihood* of the identified risk, and ultimately its impact:



Figure 2.6.2: Risk Matrix

Using the matrix above, the following risk plan has been developed:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Risk | Consequence | Likelihood | Impact |
| 1 | Team member leaves the course. | Major | Rare | Low |
|  | Controls:   * Ensure the effort of all team members is shared in a common repository. * Ensure equal distribution of workload. * Ensure regular and constructive communications occurs between all team members to encourage friendly comradeship and a willingness to remain engaged. * Team Leader to request a scope reduction from the course co-ordinator. * Team Leader to request a time extension. | | | |
| 2 | Team member becomes ill. | Moderate | Likely | High |
|  | Controls:   * Immunise team members against influenza. * Ensure the effort of all team members is shared in a common repository. * Ensure equal distribution of workload. * Team Leader to request a scope reduction from the course co-ordinator. * Team Leader to request a time extension. | | | |
| 3 | Poor productivity (Student Syndrome) | Moderate | Possible | High |
|  | Controls:   * Have regular and constructive communications. * Have short and frequent sprints (1-2 weeks) with clear objectives and strong leadership. | | | |
| 4 | Scope is incomplete, ambiguous or contradictory. | Moderate | Possible | High |
|  | Controls:   * Concerns to be communicated to the Team Leader immediately, and the Team Leader would then raise these concerns with the Project Sponsor and Course Co-Ordinator. * Have regular and frequent meetings with the Project Sponsor and Course Co-Ordinator. * Run project using sprints. At the conclusion of each sprint, a tangible result is to be presented to the project sponsor for review. | | | |
| 5 | Project scope changes. | Catastrophic | Possible | Very High |
|  | Controls:   * Change Request Form required for project scope changes. This form is to be approved only by the Team Leader and the Course Co-Ordinator. * Team Leader to request a time extension. | | | |
| 6 | Third-Party applications difficult to implement. | Moderate | Unlikely | Medium |
|  | Controls:   * Alternative applications are to be considered at the outset so that if a chosen one proves difficult to implement then an alternative can be considered. * Actively consult with other team members, the Course Co-Ordinator, or use on-line forums to seek expert advice or opinion on the operation of the application. | | | |
| 7 | Thirty-Party applications contain software bugs. | Moderate | Unlikely | Medium |
|  | Controls:   * Alternative applications are to be considered at the outset so that if a chosen one proves problematic then an alternative can be considered. | | | |
| 8 | Project file corruption or loss. | Catastrophic | Rare | Medium |
|  | Controls:   * Use on-line software repository (e.g. GitHub). | | | |

Figure 2.6.3: Risk Assessment & Controls Table

# Quality Assurance

Quality assurance (QA) is a methodology to ensure mistakes and defects are not present, or are at least tolerable. The role of each team member is to ensure that they communicate issues or concerns, and implement controls to provide adequate QA.

## Testing

Since this project is the development of a web application with a broad spectrum of users (general public, and an administrator), both **Acceptance Testing** for the user interface and **Unit Testing** for the backend will be employed.

**Acceptance Testing:**

Although Martin (2011) strongly recommends that an Acceptance Testing should be automated in effort to constrain project costs, the project development team has decided to use a manual testing approach (i.e. using real people to test the user interface) for the following reasons:

1. The web application is relatively simple to use, and so flaws would likely present themselves easily and quickly;
2. The project team is relatively small, and is concerned that the time required to implement an automated application such as **Robot Framework**, or **Selenium** may take too long; and,
3. The Project Sponsor concedes that he is still unsure with the overall functionality and behaviour of the Web Application which means any automated solution may quickly become obsolete.

With this approach the project team will rely extensively on a strong collaboration between themselves and the Project Sponsor during the development and testing phases. The final Acceptance Testing of the user interface will then be undertaken using three independent volunteers and the Project Sponsor himself.

**Unit Testing:**

Unit testing will be used to test and benchmark the backend server logic code and business code. Since the backend employs Google's Open Source programming language **Go**, these tests will be implemented using the programming language's in-built unit testing tools. In using these tools, we will endeavour to achieve a test coverage of 100%.

Since the **PostgreSQL** database tables are relatively simple, these will be tested directly using the **pgAdmin** application which is an Open Source administration and development tool for PostgreSQL.

The interaction between the browser and the server can be tested by simulating the interaction process using the simulated HTTP client first, and then testing the front end through the front-end access to reduce the complexity of the test.

## Issue Tracking

To ensure issues are tracked, and to facilitate free and open diaglogue between the team members, the team is using **GitHub**. **GitHub** allows it users to *create* and *track* issues. The issues are assigned to a user or users, and collectively the team can monitor and track progress on each issue. All tracking is time-stamped so users can also see how long an issue is kept open for. Below is a high-level extract of all the **GitHub** issues recorded so far for this project:

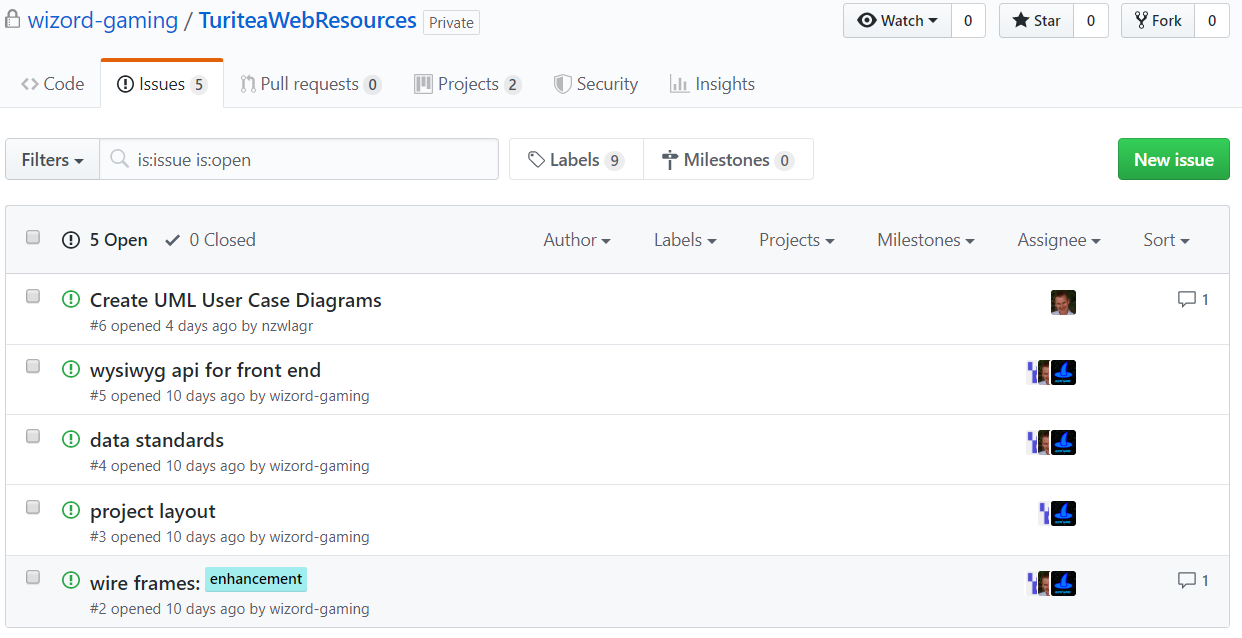


Figure 3.2.1: GitHub Issues (dated 10.08.2019)

It is intended that the **GitHub** issues tool will be used to identify new features, change requests, and report on software errors.

# Product

The following has been developed so far:

* A **GitHub** repository for storing all project files and creating issue logbooks;
* A **GitHub** project for creating a KanBans.
* A set of webpage Wire Frames developed using the **Wireframe Sketcher** application for consideration (see Appendix 6.2).
* A simple webpage with an integrated **Cesium.js** Javascript program demonstrating the potential use of the **Cesium.js** API.

# References

*Agile Manifesto*. (n.d.). Retrieved August 10, 2019, from Atlassian Agile Coach: https://www.atlassian.com/agile/manifesto

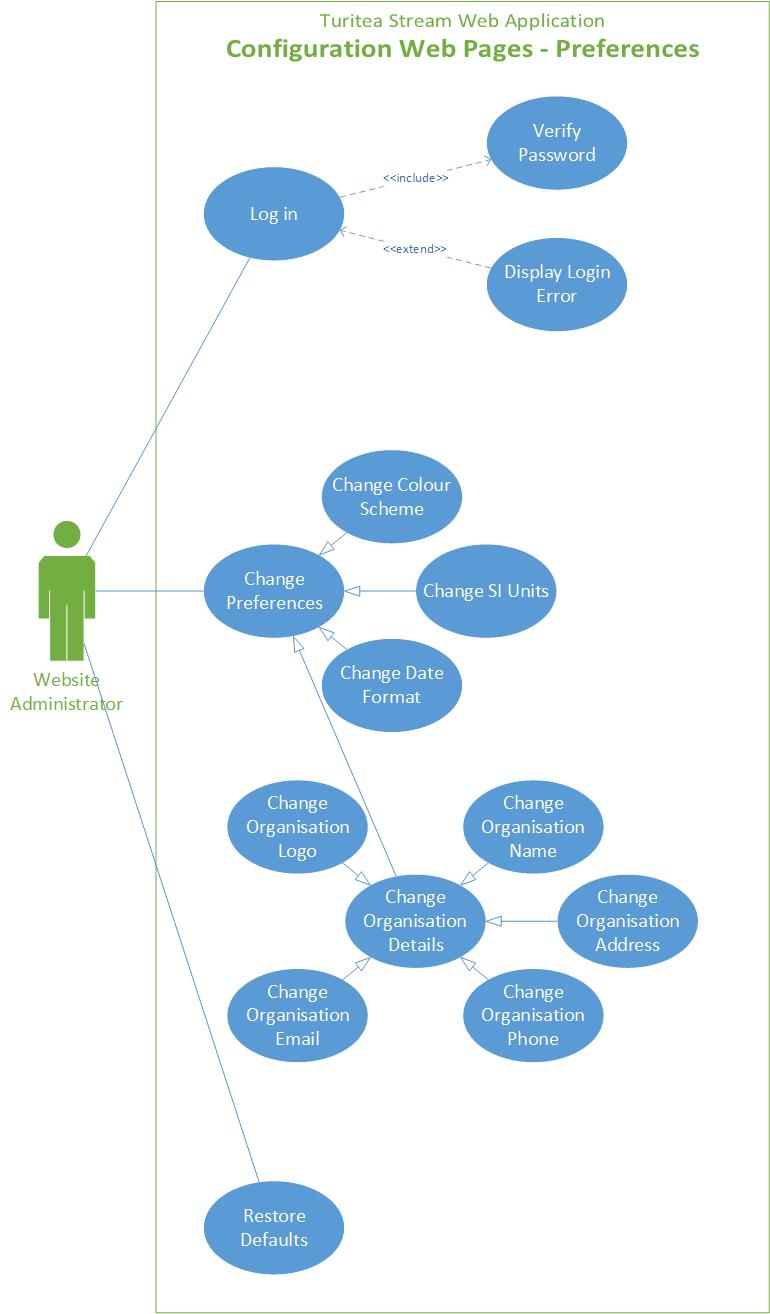
Martin, C. R. (2011). *The Clean Coder.* New Jersey: Pearson Education Ltd.

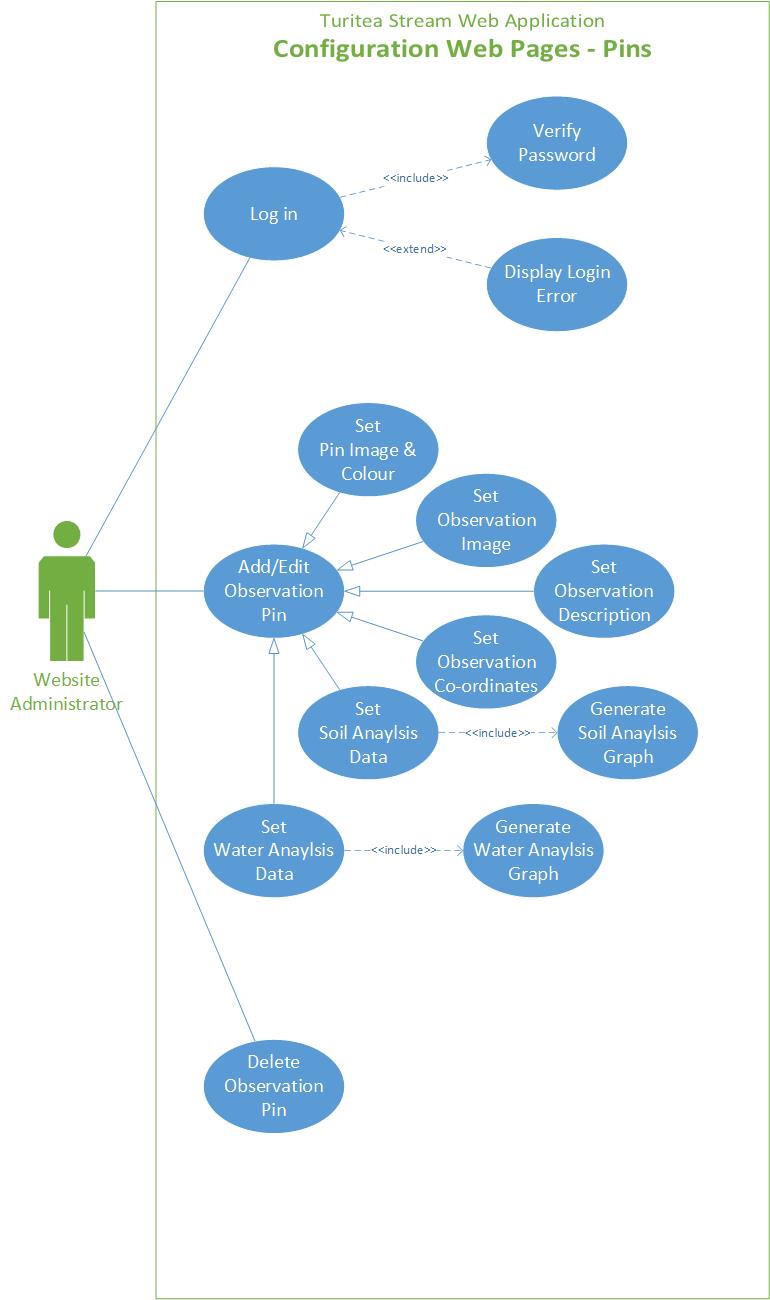
Murray, A., Bennett, N., Edmonds, J., Patterson, B., Taylor, S., & Williams, G. (2009). *Manageing Successful Projects With Prince2* (5 ed.). Norwich, England: The Stationary Office.

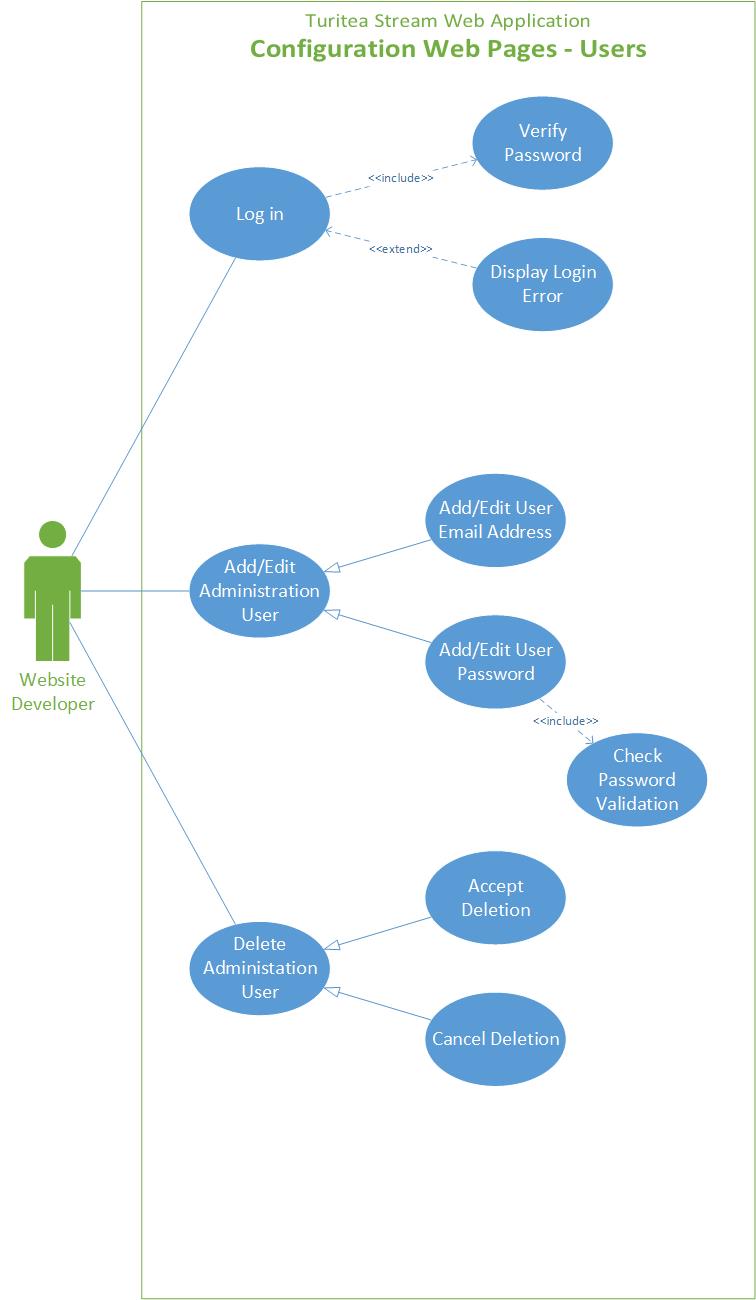
Pinto, J. K. (2013). *Project Management Achieving Competitive Advantage* (3 ed.). Essex, England: Pearson Education Limited.

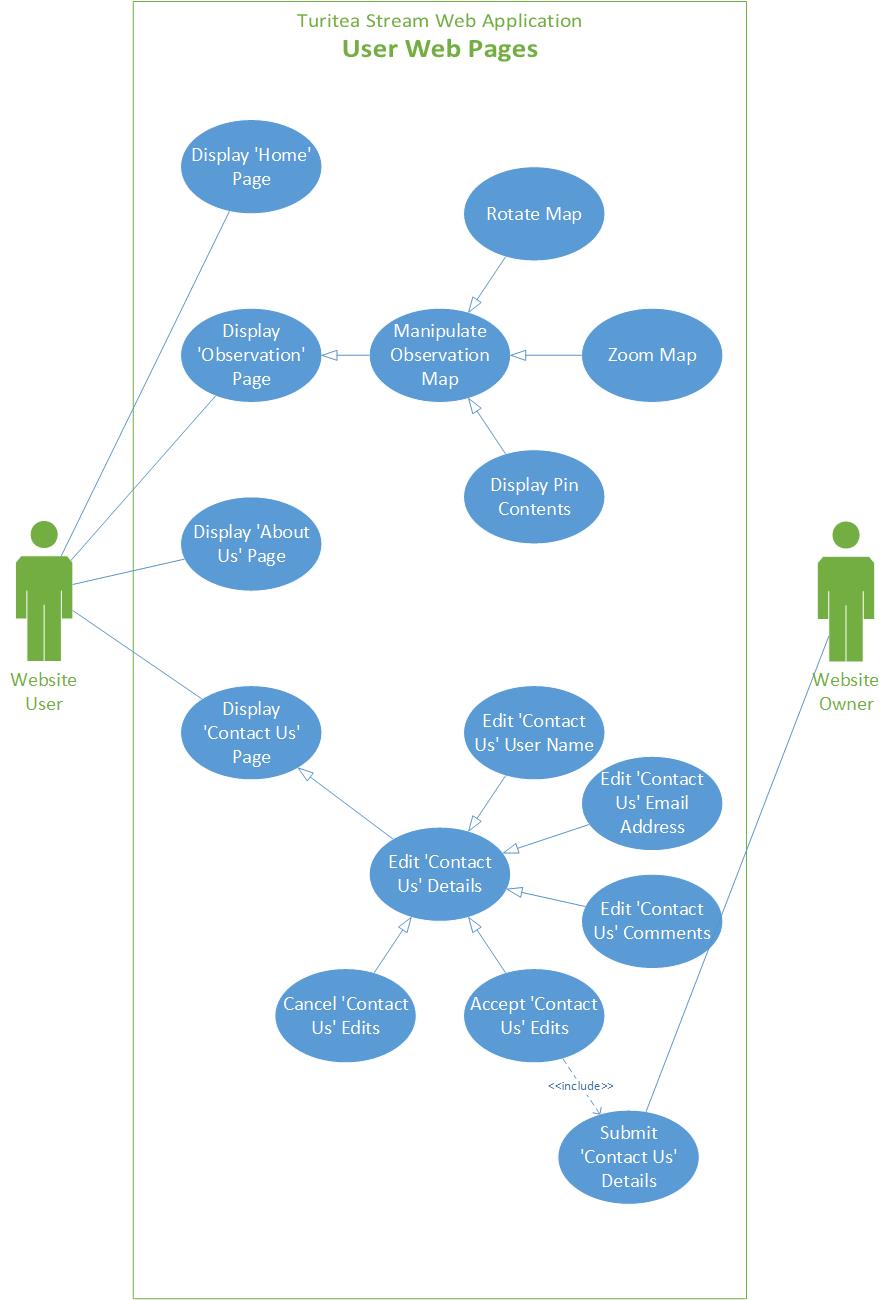
# Appendices

## Use Case Diagrams









## Webpage Wire Frames

