Fourth Wall

*Finbourne Technology*

*Group 26 - Project Plan*

Daniel Whelan

Adam Mulvihill

Xiaolei Zhang

Haojun Xing

Tamunotonye Harold Karibye

Daniel Madaghjian

**Table Of Contents**

[1. Project Goals and Objectives 2](#_Toc98175507)

[1.1 Background 2](#_Toc98175508)

[1.2 Objectives 2](#_Toc98175509)

[1.3 Goals 2](#_Toc98175510)

[2. Project Scope 3](#_Toc98175511)

[2.1 Project Deliverables 3](#_Toc98175512)

[2.2 Project Boundaries 3](#_Toc98175513)

[2.3 Product Backlog 4](#_Toc98175514)

[3. Project Approach 5](#_Toc98175515)

[3.1 Scrum Sprints 6](#_Toc98175516)

[4. Project Organisation 7](#_Toc98175517)

[4.1 Staff 7](#_Toc98175518)

[4.2 Staff Charts 8](#_Toc98175519)

[5. Risk 10](#_Toc98175520)

[5.1 Risk Analysis 10](#_Toc98175521)

[5.2 Risk Mitigation 10](#_Toc98175522)

[6. Project Controls 11](#_Toc98175523)

[7. Communication 12](#_Toc98175524)

[7.1 Client Communication 12](#_Toc98175525)

[7.2 Project Team Meetings 12](#_Toc98175526)

# 1. Project Goals and Objectives

## 1.1 Background

Our client Finbourne operates multiple independent environments containing client data. To protect the data, access to these environments is limited by a system where users who wish to access them must authenticate themselves and submit a request for elevated access before being able to view or modify the data or systems within each environment. The volume of these “break glass” requests has become a point of concern. The idea of our project is to develop a service that can run inside each environment in parallel with databases. The service will monitor the databases and surface data necessary for developers. A suitably authenticated API will be used to make the data available. The purpose of this is to allow access to data such as table statistics, index usage, and long running queries, without allowing access to the client data itself, and as more data is surfaced through the service, to reduce the number of necessary “break glass” requests.

## 1.2 Objectives

We aim to build a monitoring service that will be used to regularly poll a PostgreSQL database for table statistics, index usage, running queries, and open transactions, and to store what it gathers in blob storage. The service will also be able to take a query and run it through an Explain Analyse clause, returning the generated plan. We also aim to develop a suitably authenticated API that will allow access to the blob storage and the monitoring service. We have been asked by our client to use C# for development as it is the language they use and would allow for easy integration.

## 1.3 Goals

The goals of this project are:

* Develop a monitoring service in C# that will regularly poll a PostgreSQL database for table statistics, index usage, running queries, and open transactions, storing the data it gathers in blob storage.
* Develop an API with suitable authentication that allows access to the blob storage and monitoring service.
* (Stretch Goal) Develop a simple UI for the API for ease of use.

# 2. Project Scope

## 2.1 Project Deliverables

There are several deliverables for this project. We are required to deliver a Requirements Specification document that details our understanding of the clients requirements for the project. These requirements are to be represented visually as well through use cases and diagrams. This document is to be signed off by the client and submitted to the college.

We must keep video recordings of our meetings on a weekly basis, in which the work carried out by team members over the course of the week is to be described, as well as the work they plan to do next. These videos are to be submitted to the college.

We must submit a Software Design Specification document to the college which details the functionalities of our monitoring service and API, and our design choices. This document must also represent several of our design choices visually through diagrams

We are also required to submit a Project Plan to the college, which details how the project will be carried out, how work will be split up amongst the team members, and the time frames in which the work will be carried out. This document will also contain elements from both the Requirements Document and Software Design Specification in the document’s Appendix.

After development, Senior Fresh students will be required to write up a Software Development Report, which will provide an insight into the development of the software. This will include details of how the monitoring service and API operates. At the same time, Junior Sophister students will be required to write up a Management Report which will detail management steps taken during the project and how work was structured. Both of these documents must be submitted to the college.

A code bundle is to be submitted to the client, containing the source code for the monitoring service and API, along with a README file detailing what is in the code bundle, and a deployment file containing instructions on how to run the software and the required dependencies. A walkthrough video is also to be contained in the bundle, showing the main use cases of the software in operation.

Lastly, team members are required to individually write a two page reflective essay, reflecting on their experience with the project and their contribution to it. An Authors’ Declaration is to be signed by all team members detailing how much each member contributed, and submitted to the college.

## 2.2 Project Boundaries

In Scope

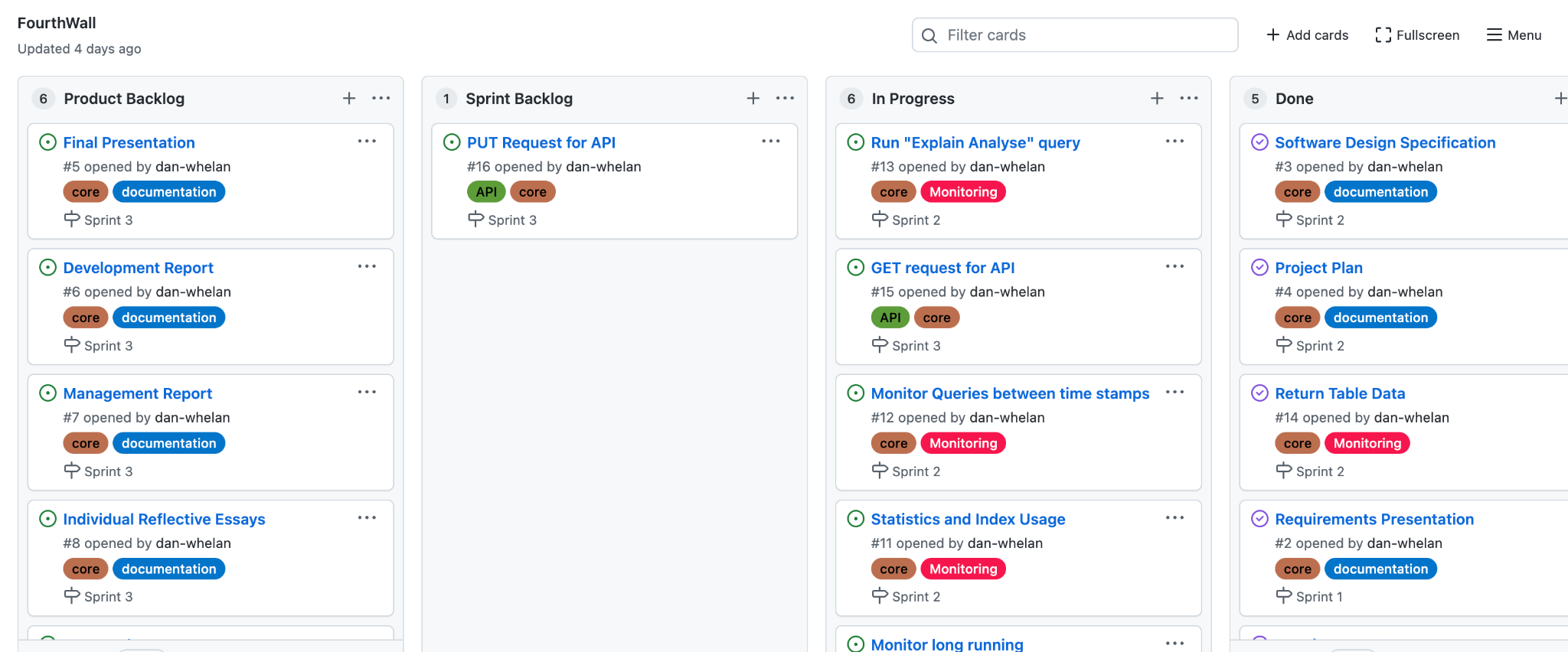
The achievable goals of this project that are within scope include the core deliverables, which include the code bundle, documentation, and a final presentation. The scope of the software includes a service that can monitor a PostgreSQL database for table statistics, index usage, running queries, and open transactions, storing what it gathers in blob storage, as well as an authenticated API that allows access to the data collected by the monitoring service.

Out of Scope

A proposed stretch goal was to develop a UI that could be used to interact with the API, as well as to think of other possible statistics that could be gathered outside of those proposed in the Requirements document. It is possible to get too preoccupied with trying to achieve this goal, which could hinder the project, as while a UI would help with ease of use, it does not contribute to the core functionality of the software.

## 2.3 Product Backlog

We are using Github Projects to keep track of our progress on the product backlog. A picture of its current state is provided below, along with a table of all the tasks so that the entire backlog can be seen.



|  |  |  |
| --- | --- | --- |
| **Task** | **Type** | **Status** |
| Requirements Document | Documentation | Done |
| Requirements Presentation | Documentation | Done |
| Return Table Data | Monitoring | Done |
| Project Plan | Documentation | Done |
| Software Design Specification | Documentation | Done |
| Statistics and Index Usage | Monitoring | Done |
| Monitor long running queries/transactions | Monitoring | Started |
| Analyse query performance | Monitoring | Done |
| Monitor Queries between timestamps | Monitoring | Started |
| Run “Explain Analyse” query | Monitoring | Not Started |
| Base Level API to contact Blobstore | API | Started |
| PUT request for API | API | Not Started |
| GET request for API | API | Started |
| Data Leaks & Security | Bug Fix / Improvements | Not Started |
| Development Report | Documentation | Not Started |
| Management Report | Documentation | Not Started |
| Final Presentation | Documentation | Not Started |
| Individual Reflective Essays | Documentation | Not Started |

# 3. Project Approach

When we received this project at first, we actually felt a little unfamiliar because it was not on our preference list. As we got to know the background of the project and the client's needs gradually, although this may not be something we are good at and familiar with, we became interested in it. After our investigation of various concepts in the project, as well as communicating with customers and understanding their requirements and problems to be solved, we began to plan and arrange the entire project with ease.

The client wanted us to use C# for development because it is the language they use and would allow for easy integration. Although it's not a requirement and none of us have ever touched it, after looking up documentation and tutorials, we decided to use it.

When deciding how to split the work, we thought the best way was to let people choose the parts they felt most confident in, so we pretty much chose and formed two small teams. This helps ensure that we produce high-quality work, and that everyone is more willing to research and explore what they are responsible for.

Next, we set up a table where all the work must be done, using GitHub Projects to record our progress and remind us of the following work that still needs to be done. And we created GitHub repositories to be able to share our current systems with our clients and with each other, allowing us to work on different parts of the project in real time. We also have a shared Google Drive folder so we can see each other's work, comment on any questions or issues we have, and collaborate on important documents.

## 3.1 Scrum Sprints

|  |  |  |
| --- | --- | --- |
| **Sprint Date:** | **Objectives:** | **Dates Set:** |
| Sprint 1:  24th Jan – 25th Feb | * Pick project * Explore required tech & background * Ascertain the challenges and capabilities of the product * Work on requirement doc & pre | * 25th Jan - Sprint planning * 28th Jan - Scrum * 2nd Feb - Client meeting & Sprint planning * 4th Feb - Scrum * 11th Feb - Scrum * 14th Feb - Client meeting * 15th Feb - Sprint review * 18th Feb – Scrum & Sprint retrospective * 20th Feb - Monitoring team meeting * 25th Feb – Scrum & Backlog refinement |
| Sprint 2:  28th Feb – 18th March | * Create scripts for testing with a mock database * Work to get the main monitoring features fully functional * Work on project plan and software design specification | * 28th Feb - Monitoring team meeting * 4th Mar - Scrum * 7th Mar - Client Meeting & Sprint planning * 14th Mar - Monitoring team meeting * Week8 - Client meeting * 18th Mar - Scrum & Sprint review |
| Sprint 3:  21st Mar – 1st April | * Work on API stuff * Test main functions * Attempt to create a simple UI * Connect all parts to get the product features fully functional | * 25th Mar - Scrum & Sprint planning * Week10 - Client meeting * 1st Apr - Scrum & Sprint review |
| Sprint 4:  4th – 14th April | * Clean up Code * Client Handover * Final Presentation * Work on development report and management report * Code bundle submission | * 4th Apr - Final Presentation * 8th Apr - Scrum & Sprint review * Week12 - Client meeting * 15th Apr - Scrum & Sprint retrospective |

# 

# 4. Project Organisation

## 4.1 Staff

|  |  |  |
| --- | --- | --- |
| **Name:** | **Prior Technical Skills** | **Prior Projects** |
| Adam Mulvihill | Java, C, HTML, CSS, JavaScript, ARM assembly, Prolog, Haskell | * Receiptless App with Orange Tribes |
| Daniel Madaghjian | Java, C, HTML, CSS, JavaScript, Excel | * Visual interface of information related to Covid-19 |
| Daniel Whelan | Java, C++, HTML/CSS, ARM Assembly, Python, Prolog, Git, Processing, JavaScript, Docker | * Development of LanguageNavigator App with Microsoft |
| Haojun Xing | Java, C, ARM assembly, XML, R, excel | * Visualising Covid 19 data with Processing |
| Tamunotonye Harold Karibye | Python, Java, C, JavaScript, Assembly, HTML, DynamoDB, Git | * Visualising Covid 19 data with Processing |
| Xiaolei Zhang | Java, C, HTML, Assembly, Python, JavaScript, Prolog, Haskell, Processing | * Improving the Wikidata Dashboard Web Application |

## 

## 4.2 Staff Charts

|  |  |  |
| --- | --- | --- |
| **Sprint Date:** | **Roles Appointed:** | **Work appointed:** |
| Sprint 1:  24th Jan – 25th Feb | * Product Owner: Daniel Whelan * Scrum Master: Harold * Team Members:   + Adam   + Daniel Madaghjian   + Haojun   + Xiaolei | * Team effort: Research, Requirement doc * Third Years: Textual work in requirement doc * Second Years: Diagrams in requirement doc * Monitoring team (Daniel Madaghjian, Harold, Xiaolei): Research, create scripts, start working on queries features * Adam: Learn to develop in C#, look up helpful tutorials * Daniel Whelan: Set up GitHub repo & Google drive, contact clients, learn to develop in C# * Haojun: learn to develop in C#, research about the database |
| Sprint 2:  28th Feb – 18th March | * Product Owner: Xiaolei * Scrum Master: Daniel Whelan * Team Members:   + Adam   + Daniel Madaghjian   + Haojun   + Harold | * Team effort: Software design spec * Third Years: Project Plan * Monitoring team (Daniel Madaghjian, Harold, Xiaolei): Keep working on monitoring features * API team (Adam, Daniel Whelan, Haojun): Understand monitoring part, get in touch with blob storage and API stuff |
| Sprint 3:  21st Mar – 1st April | * Product Owner: Adam * Scrum Master: Daniel Madaghjian * Team Members:   + Daniel Whelan   + Haojun   + Harold   + Xiaolei | * Team effort: demo prep * Monitoring team (Daniel Madaghjian, Harold, Xiaolei): Clean up monitoring codes, ensure the achievement of features, verify connection with API * API team (Adam, Daniel Whelan, Haojun): Work on API and storage |
| Sprint 4:  4th – 14th April | * Product Owner: Daniel Whelan * Scrum Master: Haojun * Team Members:   + Adam   + Daniel Madaghjian   + Harold   + Xiaolei | * Team effort: Final Presentation * Third Years: Management report * Second Years: Development report * Monitoring Team: Improve and streamline algorithm * API Team: Ensure secure connection and well tested |

# 5. Risk

## 5.1 Risk Analysis

|  |  |  |  |
| --- | --- | --- | --- |
| **Risk Element** | **Impact (1 to 5)** | **Likelihood (1 to 5)** | **Risk Factor (I\*L)** |
| Lack of experience with technologies - C#, REST API, PostGres etc. | 4 | 3 | 12 |
| Problems due to time restrictions | 4 | 5 | 20 |
| Difficulty accessing PostGres DB and Binary Large Object Storage through API | 5 | 3 | 15 |

## 5.2 Risk Mitigation

|  |  |
| --- | --- |
| **Risk** | **Measures to Reduce Risk** |
| Time Restrictions | This Project has a total scope of 12 weeks and as a result we have scoped the requirements appropriately. We split the team into two distinct sub-teams to tackle the two major elements of the project appropriately. This should make the workload more manageable per team member and hence improve the overall quality of the end product |
| Difficulty accessing PostGres DB and Binary Large Object Storage through API | As this area is new to all team members, we have spent time researching the topic. Our client has given us useful links and is available to help us. There are also some tools and libraries available in C# that help with contacting Binary Large Object Storage specifically that may aid in the overall final product. |
| Lack of experience | To mitigate this problem, we have dedicated time in the early stages of this project to learn these new technologies. Our monitoring service team researched the C# language and PostGres Database features in order to gain a better overall understanding. Our API team has been learning how to build REST APIs using DotNet and contacting Binary Large Object Storage devices using said APIs. Furthermore, our client has been guiding us through these new technologies and helping us gain a better knowledge of them. |

# 6. Project Controls

We will use the following tools to ensure our project runs smoothly:

* **Github**: We will be using github for our code repository. Our client has access to this so that they can follow our progress throughout the project. A simplified version of gitflow has been implemented where the main branch is protected and the development branch is used for new features and then merged into main.
* **Github Projects**: This is a web-based, Kanban-style, list-making application that is available through GitHub. Here we are able to keep track of Issues that appear in development. We have set up four distinct boards that will order these issues in terms of priority, whether they are in the sprint backlog or overall product backlog, and also whether they are in progress or completed.
* **Microsoft Teams**: This is a collaboration platform in which we can communicate and share files or notes with our team and client. We use teams to host our weekly meetings with our Demonstrator and our Client.
* **Google Drive**: This is where all our reports and meeting minutes can be shared amongst each other and are saved on the drive to ensure our documents are saved properly and can be accessed by each individual group member
* **Whatsapp**: This is the easiest form of communication amongst just the team members so we can organise meetings, discuss what has been done and remind each other of deadlines on a day-to-day basis.
* **Our sprint schedule**: This is an important schedule which features in this report to make sure we are keeping track of our time and using it wisely and also to make sure at the end of each sprint we have done what is outlined.
* **Zoom**: This is a video calling web application that lets us communicate within our team. Although we meet weekly in person we use Zoom to record our weekly scrum meeting summary and also some scrum meetings when people are unable to make it in person.

# 7. Communication

## 7.1 Client Communication

Our approach to client communication was to continue with an agile methodology with a meeting biweekly, usually on a Monday. However, there have been weeks where the client has been unavailable and hence we have scheduled some meetings three weeks apart rather than biweekly.

|  |  |
| --- | --- |
| **Date:** | **Purpose:** |
| 2nd February | * Initial first meeting - Establish requirements & introduce the team |
| 14th February | * Second Meeting - Sprint 1 review, Receiving Coding Standards & requirements sign off |
| 7th March | * Third Meeting – Mid-point Project Review & guidance relating to Binary Large Object Storage |
| 21st March | * Fourth Meeting – Sprint & Project Plan Review |
| 4th April | * Fifth Meeting – Final Touches & Testing Discussion |
| 11th March | * Sixth Meeting – Project wrap-up & Client Handover |

## 7.2 Project Team Meetings

|  |  |
| --- | --- |
| **Date:** | **Purpose:** |
| 26th Jan | Discuss Projects and form List of top five |
| 28th Jan | Discuss Project received & set up client meeting |
| 4th Feb | Discuss Research & Assign work for Requirements document |
| 11th Feb | Finalise Requirements Document & Requirements Presentation Practice |
| 18th Feb | Split Team into two sub-teams & Assign Jobs to individual team members |
| 25th Feb | Review Sprint Progress. Skeleton Monitoring System Done |
| 4th Mar | Assign Work for Reading Week & Prepare Questions for Client Meeting |
| 7th Mar | Debrief After Client Meeting & Assign Elements of Documentation to Team Members |
| 15th Mar | Finalise Documentation & Discuss Final Elements of Software to be Completed |
| 21st Mar | Client Meeting Debrief & Final Touches |
| 25th Mar | Presentation practice |
| 1st April | Presentation Practice |
| 8th April | Review & Code Cleanup Appoint Docs |
| 11th April | Sync up on Docs |
| 14th April | Code Bundle submission, Document submission, Wrap up |

# 