## 1. Scope of Work

# 1.1 - Goal

Display meaningful sound data using visualisation on a web application that will assist the Australian Navy Medical Officer in identifying sailors that could be experiencing fatigue and other health issues due to extended exposure to large sounds while working aboard the vessel.

# 1.2 - Epics

The goal can be broken down into four epics:

* Gather sound data and research that will outline the potential affects sound will have on sailors over a period.
* Format data into a database to be displayed on web application.
* Create a web application that once finished can be easily migratable into the pre-existing web app used by the Australian Navy.
* Display the data on web application, while outlining sailors that could been experiencing fatigue due to large exposure to sizeable amounts of decibels.

## 2. Skills and Resources

# 2.1 - Task

This task requires the use of many different skills, of which we will either need to be already proficient at or develop proficiency at an early stage. These skills can be broken up into three distinct categories, namely: front-end skills, back-end skills and general skills.

# 2.2 - Skills

Front-end skills include:

* HTML and CSS for structuring and styling the web page(s) in which we will output our findings to.
* JavaScript for functionality over the web application
  + Familiarity with a JavaScript library for presenting the data (D3 – Data Driven Documents library).
  + Familiarity with React, a JavaScript library for building user interfaces across a web app.

Back-end skills include:

* SQLite or equivalent database engine library for setting the schema for our database, in which we will store sound data and resulting data after processing.
* Flask for hosting the web application and routing.
  + Python for constructing the Flask web app.

General skills include:

* Sound collection, including operation of a sound collecting device and withdrawing data from the device.
* Ability to comprehend, process and withdraw relevant information from the data resulting from the collection of sound.
  + This involves some level of understanding of the relationship between sound levels and fatigue levels.
* Project management skills, including utilising the scrum agile method to keep our project on track.

# 2.3 - Resources

Due to the fact that all of the group members have some background in the computer science/software field we are reasonably well equipped to handle most of the challenges that the task presents. All of our group members possess skills in the front-end and back-end categories, however, will need to develop our skills in the general category through the following resources:

* Scholarly articles and studies relating to the impact of sound on fatigue; through sound level, frequency and/or consistency.
* Mentorship from our mentor Matthew James from VGW.
* UWA and online material regarding agile methods, specifically the scrum method.

The task also requires access to both hardware and software tools, software tools include:

* JavaScript D3 library
* JavaScript React library
* Flask web app framework for Python

Hardware includes:

* Access to sufficient personal computing power to construct the system.
* Access to a sound recording device.

## 4. Project Acceptance

# 4.1 - Objectives

The purpose of this document is to outline the tests required to pass to determine whether the project is of an acceptable standard to deliver to the clients. The testing strategy is a combination of Unit tests - which will test individual code snippets, integration tests - which test that the systems components (backend, frontend) work correctly together, system tests – testing the system as whole, and acceptance tests – testing the end-user experience. The tests will be performed using a variety of technologies, such as code testing modules, and the Selenium testing framework.

# 4.2 – Document References

Requirement documents, which will be the product of the Sprint 1, are used to communicate the aims of the project in a clear, concise way that should reflect the needs of the client and stakeholders. The two main documents that will be focused on will be a Business Requirements Document (BRD), which details the high-level problems that our system should solve in relation to the customers’ needs. It will be concise and direct – in dot point form. We will also deliver a draft Functional Requirements Document (FRD), which defines how our system will behave to achieve the business requirements defined in the BRD.

Design documents will be delivered at the beginning of Sprint 2. Once the high-level business requirements have been finalised, we will detail how the system will be technically implemented. This will include a system architecture, as well as pseudocode examples of any algorithms and data structures.

There will be multiple stages of functional and non-functional testing before the final delivery of the project. Firstly, there will be code unit tests run on individual snippets. Secondly, there will be system tests, which test how components will work together. There will also be user acceptance testing, which tests the end-user experience compared to the requirements.

# 4.3 – Test Summary

The scope of work is broken down into four epics.

1. Gathering sound data that will provide insights into how sound will affect fatigue.
2. Formatting data into a backend database.
3. Creating a web application that will be reusable by Thales
4. Displaying the data on a web application with effective and clear visualisations.

![Diagram

Description automatically generated]()We will describe some high-level tests that will be applied to these systems.

# 4.4 – Testing Strategy

**4.4.1 - Test A**

|  |  |
| --- | --- |
| **System to be tested** | Backend database |
| **Testers** | Kese, Caleb |
| **Timeframe** | During development and upon integration |
| **Test location** | In development and production environments |
| **Test method** | Unit tests, schema tests |

Test A is a test of the backend data base system, to ensure that it obeys ACID properties, the schema is correct, and it is properly normalised. These tests will primarily be performed in a scripting language such as Python, where data can be inserted and retrieved to verify its integrity.

**Test Specification**

The database must:

1. Hold data regarding sound-level measurements across different locations
2. Be ACID compliant
3. Resilient to updates and deletions.

**Test Description**

|  |  |
| --- | --- |
| **Location of test** | Development server |
| **Means of control** | Automatically using a test driver |
| **Data** | Input: SQL Select, Insert, Update and Delete queries  Output: Returned data/logs |
| **Procedure** | Python script |

**Test Analysis Report**

The performance of the test will be determined by:

1. Whether all the SQL tests pass successfully
2. Whether the data returned is correct
3. Whether the database is ACID compliant.

**4.4.2 - Test B**

|  |  |
| --- | --- |
| **System to be tested** | Frontend webserver |
| **Testers** | Shane, Darby |
| **Timeframe** | During development and upon integration |
| **Test location** | In development and production environments |
| **Test method** | Unit tests, integration tests |

Test B is a test of the front-end web server system, to ensure that it is responsive, and all features are working as required. It will be performed primarily using Selenium browser-emulation tests.

**Test Specification**

The webserver must:

1. Be able to visualise sound data in a meaningful way.
2. Be able to slice the data on meaningful axis

**Test Description**

|  |  |
| --- | --- |
| **Location of test** | Development server |
| **Means of control** | Manual input  Automatic selenium tests |
| **Data** | Input: End-user behaviour, automated tests  Output: Visualisations |
| **Procedure** | Selenium test |

**Test Analysis Report**

The performance of the test will determined by:

1. How interpretable the client believes the visualisations are.
2. Whether the data returned is correct
3. Whether the automated tests pass

**4.4.3 - Test C**

|  |  |
| --- | --- |
| **System to be tested** | Sound collection |
| **Testers** | Aditi |
| **Timeframe** | During development and upon integration |
| **Test location** | In development and production environments |
| **Test method** | Unit tests, integration tests |

Test C is a test of the sound-collection system, to ensure that the devices we are using to measure the sound levels report timely and accurate data. This will be primarily conducted using a set of scripts that connect to the devices and compare them to other known sound levels. s

**Test Specification**

The sound collection system must be able to:

1. Accurate record sound levels over time

**Test Description**

|  |  |
| --- | --- |
| **Location of test** | Physical room, software emulation |
| **Means of control** | Manual sound level input |
| **Data** | Input: Varying sound levels in different rooms  Output: Accurate sound level readings |
| **Procedure** | Python scripts |

**Test Analysis Report**

The performance of the test will determined by:

1. How consistent the sound level measurements are
2. How accurate the measurements are

## 5. User Stories

# 3.1 - Sprint 2 Goals

For sprint 2, our goal is to have a skeleton of the final application up and running, with all the essential features working. Allowing us to present a working application to the client gives us time to incorporate their feedback before commencing sprint 3 and finalising the application. As a team, we have identified three overarching goals which will be complete during sprint 2, being:

1. Collect and store sample sound level recordings
2. Build out the back end of the application
3. Develop a dashboard that displays the sound levels in a meaningful way

# 3.2 - User Stories

# To help understand what needs to be completed for each of the goals identified for sprint 1, we have developed a number of user stories.

# The navy needs to store sound recordings on board the ships to help monitor the soldier's wellbeing

# The health officer needs to have access to the sound levels soldiers are exposed to onboard navy ships, helping them make better decisions to manage fatigue levels

# For the health officers to interpret the sounds readings, we need to display the findings in a dashboard

# Having sound data readily available on different locations onboard the ship will help schedule the soldiers to maximise their productivity levels

# The health officer will require the sound levels to be analysed on a person-person basis, a group basis, and the entire crew in order to understand how the level of noise influences fatigue

# Thales will need to integrate the sound monitoring software into their suite of offers before deploying it on the navy ships

# The majority of the user stories will be identified for sprint 3 as we flesh out functionality. We are focused on developing a frame in which we can add user features further down the track.

# 3.3 - Sprint 2 Tasks

We can further break down the three goals for sprint 2 into smaller actionable sub-tasks.

1. Collect and store sample sound level recordings
   1. Use a phone to collecting sample sound readings
   2. Collect samples in a variety of different environments (quiet, normal, loud, very loud)
   3. Collect samples from an environment where the noise levels spike. Helpful when testing whether we can pick up abnormal noise increases.
   4. Convert sound recordings into a format we can store and conduct our analysis from
2. Build out the back end of the application
   1. Carefully plan out the schema for the database
   2. We will be using flask and SQL lite for the backend
   3. Build the database with sample data collected from the above point
   4. Write functions that will be required to access and store data in the database
   5. The database will be running on a local server for sprint 2
3. Develop a dashboard that displays the sound levels in a meaningful way
   1. The dashboard will be built using Typescript and React
   2. Come up with a variety of visualisation in PowerPoint to give the client an indication of what they will be looking at.
   3. Check to ensure the visualisations satisfy the client's requirements
   4. Transform the visualisations into a dashboard
   5. Visualise the sample sound recordings in the dashboard

We have chosen to work on all major components of the application simultaneously, so as we develop the application and iteratively tweak the functionality and performance, all parts will still work together. For example, suppose the sound collection methodology must change for an unforeseen reason. In that case, we can quickly adapt the backend to handle the data, and the dashboard can be adjusted to use the data.