Functional Specification

0. Table of Contents

1. Introduction

1.1 Overview

Panoptes is a web application that allows users to manage servers and any services they control all in a single place. The system will alow users to create and manage modules, which will be used to gather data and control external services, as well as optionally install services associated with those modules. Panoptes will also allow users to create role hierarchies which will limit what functions of modules a given user can access.

1.2 Business Context

As a product, Panoptes is aimed towards any business that requires system administration, and towards Homelab administrators running their own services.

From an enterprise perspective, Panoptes will be very useful due to system adminstrartors being able to easily create or install modules for their custom services, like any monitoring tools that they set up, without having to find an interface that has everything that they want.

They will also particularly benefit from the permissions system, which will allow them to give access to certain functions to more junior staff, while allowing senior staff access to more privileged functions.

Meanwhile, homelab administrators will benefit from the ability to create or install modules to monitor and control services they have set up themselves in a way that suits their niche use case.

1.3 Glossary

1.3.1 Panoptes

Panoptes is the name of the project - a modular web interface for managing servers and services.

Module - A user-created package that provides functionality for use cases not covered by the base software.

2. General Description

2.1 Product/System Functions

Using Panoptes you will be able to:

- Monitor and control services.
- Install user-created modules for control of custom or niche tools/services.
- Curate a list of tools and services to monitor on the same user interface.
- Implement user permissions to restrict or allow access to certain functions withen modules.

2.2 User Characteristics and Objectives

The primary user of Panoptes is an administrator of a server or fleet of servers. As Panoptes is primarily aimed towards people in technical professions, it is a reasonable assumption that users will have some degree of technical knowledge.

2.3 Operational Scenarios

2.3.1 Module Installation

The user wants to install a new module to monitor a web server they have set up. First the user will select the 'add module' button,from here they will see a list of approved user-created modules.

As well as that, the user can paste a URL to a repository containing the source code for a module they've eitjer wrote or found that suits their specific use case. After pasting in the required URL the user can click 'install' and the installation process begins.

2.3.2 Module Control

The user would like to configure what port their apache web server is running on. From our main menu they can select the module controlling their apache server.

From here they will see a summarised version of any important information regarding this server i.e port number, server load etc

After selecting the module they will see a list of functions and configurable settings. The user selects the setting for 'port number' and changes the number. They can then save and restart the server and observe it operating on a new port.

2.3.3 Function Permission Denied

The user is a junior adminstrator in an organisation. They wish to configure a setting within a module to fix an issue however they do not have access to it as their user account has been assigned the 'junior' role within the organisation.

The user must ask someone holding the 'senior' role or higher to configure the setting for them as only they have access to that specific function of the module.

2.3.4 User Login and Verification

User opens Panoptes and and is prompted to either log in or create an account. To log in, the user simply will enter their username and password which will be checked against our database and authorised using the OAuth 2.0 framework.

Similarly, if the user has not registered an account they will be prompted to enter a username and password. Once they have done this the system will check if the username and password are valid, create the account and add it to our database.

2.4 Constraints

2.4.1 Security Constraints

This web application will need to be developed in a way that prevents users and malicious actors from accessing data or functions that they should not be able to access.

2.4.2 Server Constraints

The server hosting Panoptes will need to support the same architectures that we have access to, due to the fact Docker requires systems to have the same archtecture. It is also important that the server is able to communicate using websockets, as they will be instrumental to the functioning of the web application.

2.4.3 Timing Constraints

Due to the limited amount of time available to complete this project, it will be important that we manage our time efficiently if we want to implement every feature.

3. Functional Requirements

3.1 User Authentication

3.1.1 Description

Panoptes must be able to verify that a user should have access to its features. This will be achieved by implementing user accounts using the OAuth 2.0 framework for authorisation and storing account information in our database.

3.1.2 Criticality

This is a critical feature as only authorised users should be able to access panoptes' functionality for security reasons.

3.1.3 Technical Issues

We dont foresee any technical issues with this feature.

3.1.4 Dependencies

This feature is not dependant on any others.

3.2 Hierarchy Management

3.2.1 Description

Within an organisation, it would be a security risk to allow all members of staff access to all functionality and configurations of a server.

To combat this we are implementing a hierarchy management system using roles. Roles will have a hierarchy and can be assigned to a user to restrict or give access to certain functionality within a given module.

3.2.2 Criticality

While this solves potential security issues it is not critical for panoptes to function.

3.2.3 Technical Issues

If a user has multiple roles, we need to ensure that they have cascading permissions. This is to say that they have the permissions of the highest priority role and if a permission is not defined in that role, it checks the next highest priority role.

3.2.4 Dependencies

This feature is dependant on user registration/authentication as roles will need to be assigned to user account.

3.3 Module Installation

3.3.1 Description

The core design of Panoptes revolves around the installation of custom modules. This feature involves taking code from a repository and spinning up a docker container for the module to exist in.

3.3.2 Criticality

This Feature is critical for panoptes to function as without it we will have no method of monitoring and controlling our services.

3.3.3 Technical Issues

We forsee issues in grabbing and running code from repositories. Security will also be a concern with community made modules.

3.3.4 Dependencies

This Feature has many dependencies. First of all, our security measures must be put in place. Secondly it will depend on container orchestration so that the docker containers that contain modules can be managed by the system. Lastly only an authenticated user can install modules which requires user authetication to work.

3.4 Module Control

3.4.1 Description

With a module installed a user should be able to monitor, control and configure said module through Panoptes' user interface.

3.4.2 Criticality

This feature brings the core functionality to Panoptes and is therefore of upmost criticality.

3.4.3 Technical Issues

Because modules can be so varied depending on the service it will be a complex task to ensure we can facilitate all kinds of required functionality.

3.4.4 Dependencies

This feature is dependant on the ability for modules to be installed in the first place.

It also depends on our container orchestration feature so panoptes can manage the container that the module lives in.

3.5 Container Orchestration

3.5.1 Description

Panoptes must be able to manage the containers used to run any modules installed on it. This will be done using the API provided by Docker.

3.5.2 Criticality

This is a critical aspect of our application, as without it the application cannot exist.

3.5.3 Technical Issues

One of the biggest technical issues will be keeping track of what containers are a part of this application and which ones are not. To help with this we will use labels to mark which containers are owned by the application.

3.5.4 Dependencies

This feature will not depend on any other components of the application.

3.6 Statistics Display

3.6.1 Description

Panoptes must have a framework that can be used to display statistics. In order to achieve this, we will build a library of components that can be easily dynamically generated with statistics embedded in them.

3.6.2 Criticality

This aspect of the application is very important for user experience, as although the system will function without it, users will find it very useful for browsing the statistics generated by modules.

3.6.3 Technical Issues

We do not foresee any technical issues with this feature.

3.6.4 Dependencies

This feature will not depend on any other components of the application.

3.7 Security

3.7.1 Description

As Panoptes will have a lot of access to the machine it runs on, it is important that the system is secure and cannot be used maliciously.

3.7.2 Criticality

Security is the most critical aspect of the application. The security of the application must protect from data and functions being accessed by people who shouldn't have access to them.

3.7.3 Technical Issues

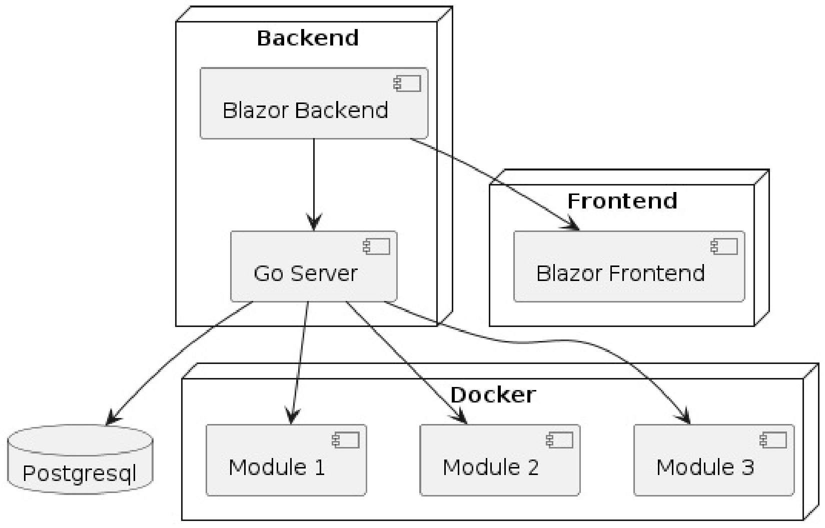
Installing community created modules is often a bad idea, so to mitigate security flaws all modules will be installed as docker containers, with only access to the resources required by it. Additionally, exposing the API that manages the containers is also a security risk, so to mitigate any security flaws arising from that, the application will function without the API being exposed.

3.7.4 Dependencies

This feature will not depend on any other components of the application.

4. System Architecture

4.1 Diagram



4.2 Go Server

The Go server takes requests from the Blazor frontend, and optionally from an external service, and uses those requests to interact with the Postgresql database, with Docker, and the containers it has set up.

4.3 Blazor Backend

The Blazor backend will handle all data processing between the Blazor frontend and the Go server. It will also handle any rendering that takes place for the Blazor frontend.

4.4 Blazor Frontend

The Blazor frontend will be what the user interacts with when using Panoptes. It will be used to display data, and also for the user to make any changes to to installed modules or configure the monitored services.

4.5 Postgresql

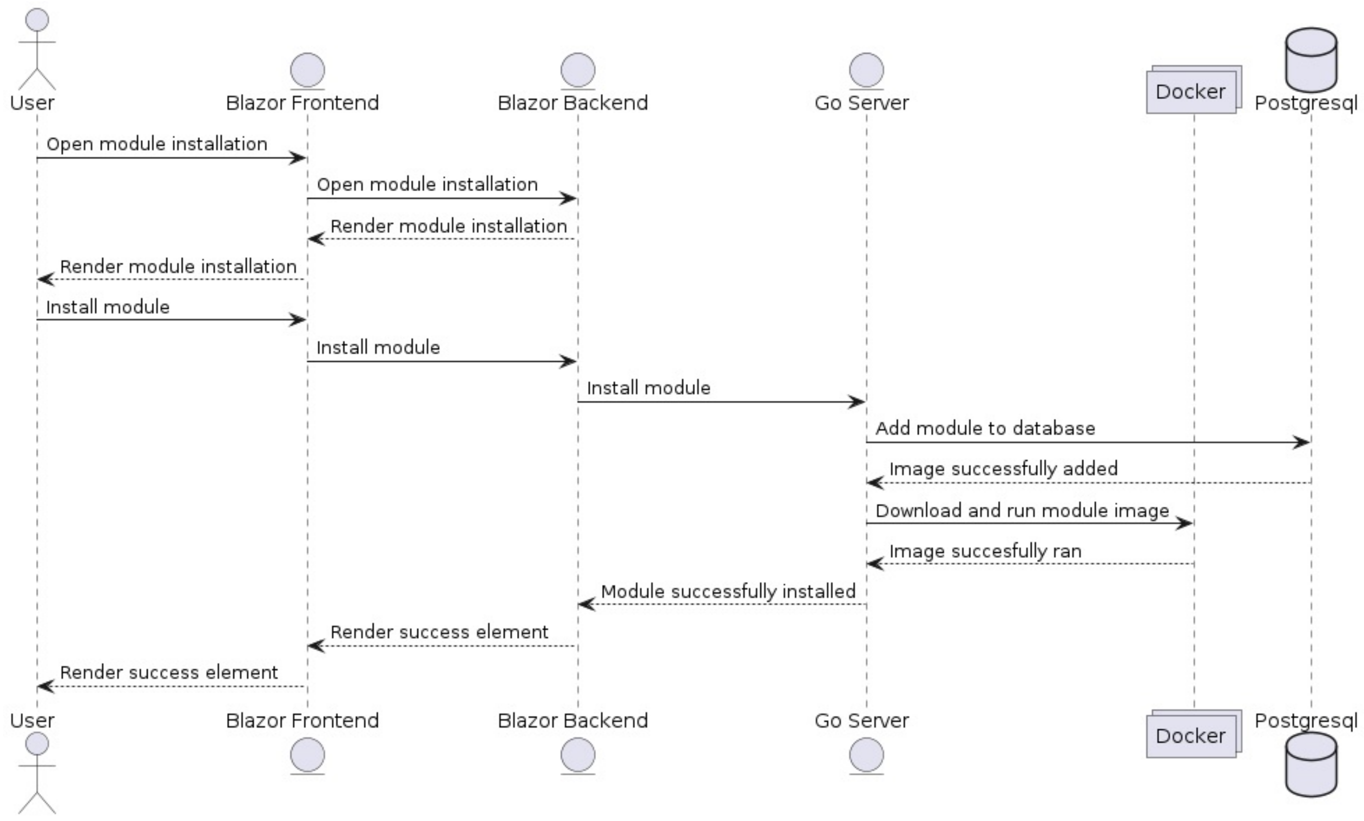
The Postgresql database will be used to store information such as user account details and module permissions.

4.6 Docker

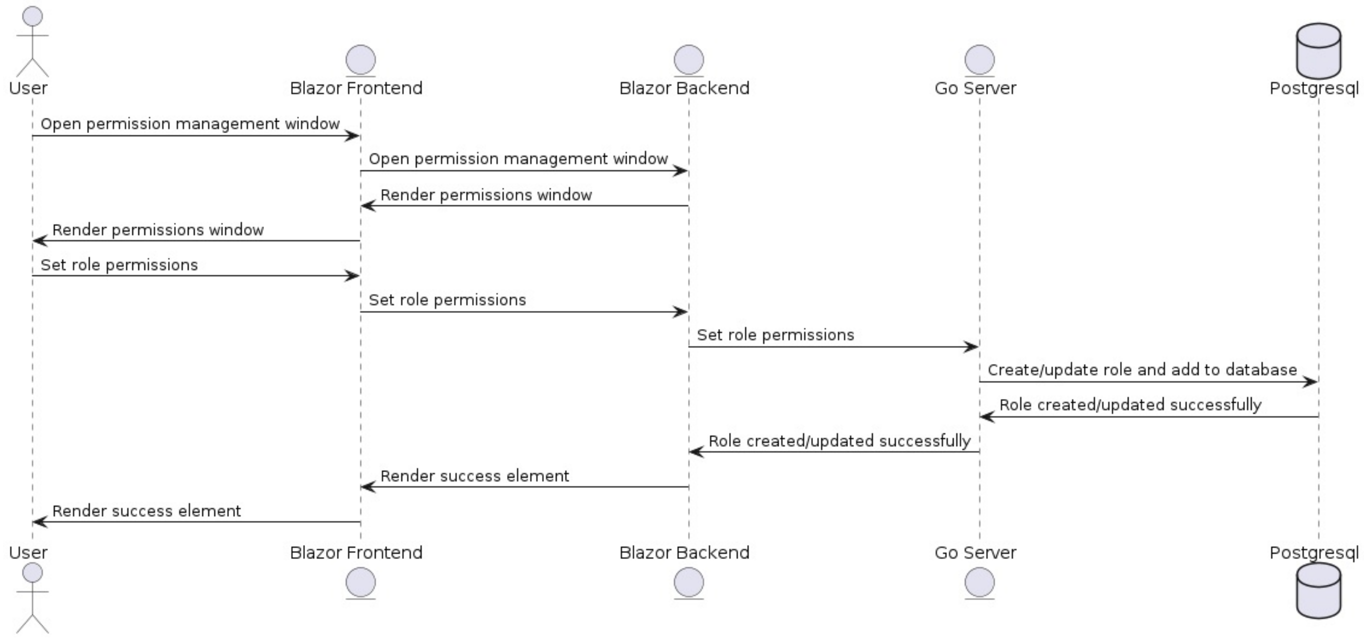
A Docker instance will have a series of containers managed by the Go server, which will be used to store modules and be interfaced with by the Go server.

5. High-Level Design

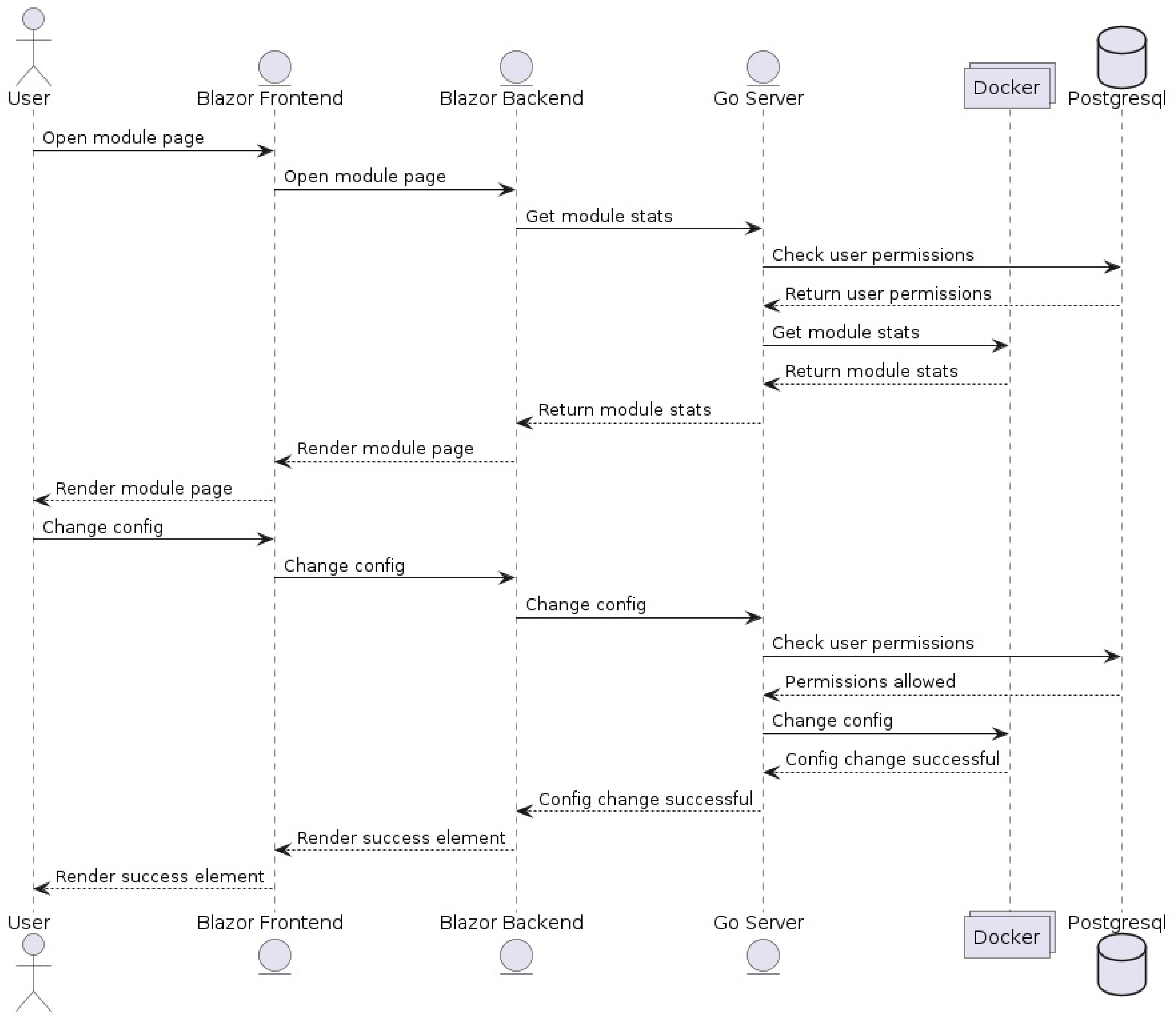
5.1 User Installs Module



5.2 User Creates/Updates Role Permissions


User
Blazor Frontend
Blazor Backend
Go Server
Postgresql
Open permission management window
Open permission management window
Render permissions window
Render permissions window
Set role permissions
Set role permissions
Set role permissions
Create/update role and add to database
Role created/updated successfully
Role created/updated successfully
Render success element
Render success element

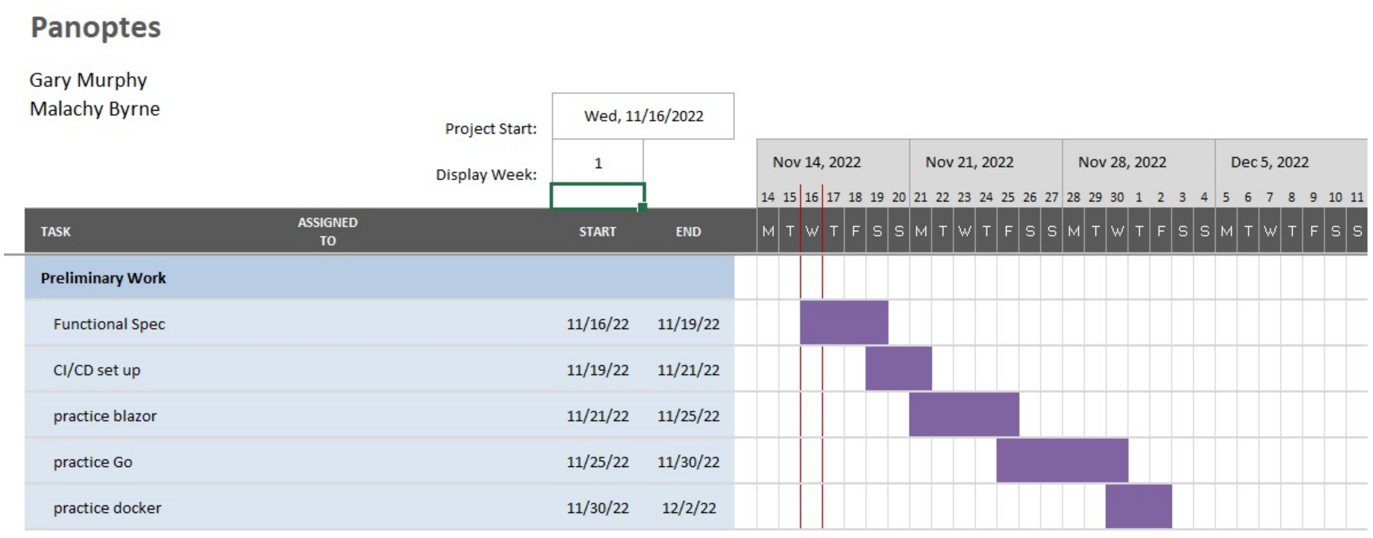
5.3 User Manages Module



6. Preliminary Schedule

We have set up our project into four distinct phases.

Phase three and four contain the bulk of the work. Phase one and two involve setting ourselves up for success by laying the foundations for phase three and four.

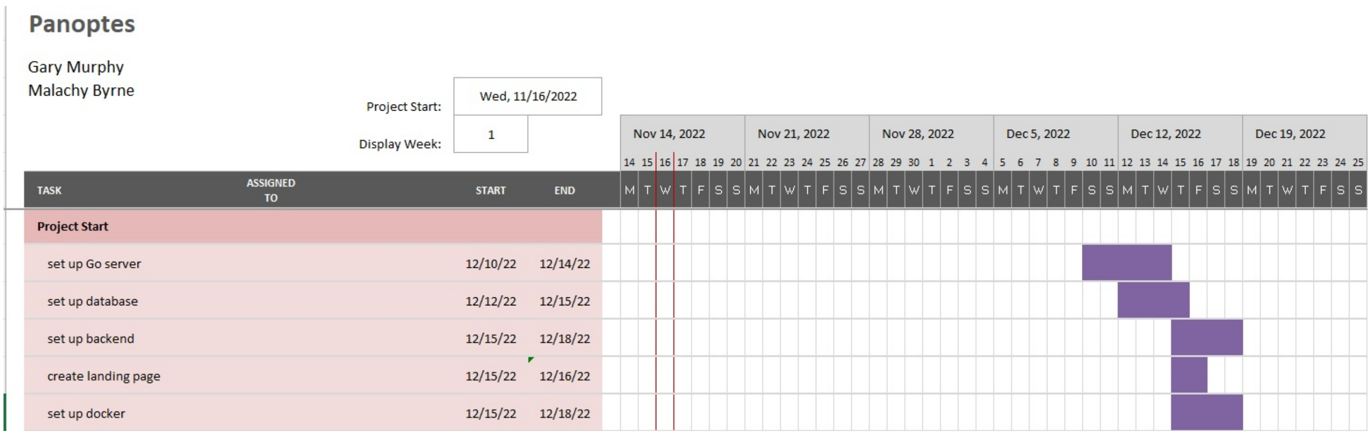

Panoptes

Gary Murphy
Malachy Byrne

Project Start: Wed, 11/16/2022

Display Week: 1

TASK	ASSIGNED TO	START	END
Preliminary Work			
Functional Spec		11/16/22	11/19/22
CI/CD set up		11/19/22	11/21/22
practice blazor		11/21/22	11/25/22
practice Go		11/25/22	11/30/22
practice docker		11/30/22	12/2/22



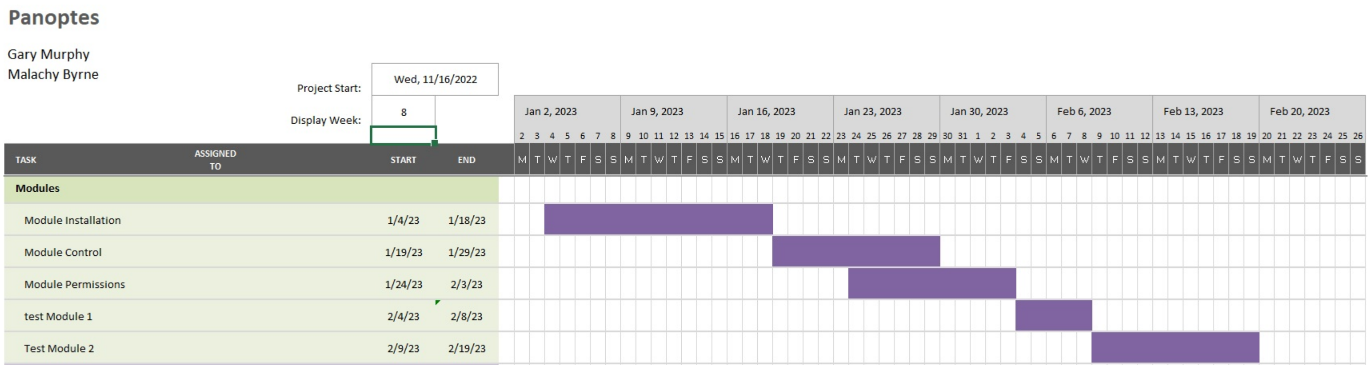
Panoptes

Gary Murphy
Malachy Byrne

Project Start: Wed, 11/16/2022

Display Week: 1

TASK	ASSIGNED TO	START	END
Project Start			
set up Go server		12/10/22	12/14/22
set up database		12/12/22	12/15/22
set up backend		12/15/22	12/18/22
create landing page		12/15/22	12/16/22
set up docker		12/15/22	12/18/22



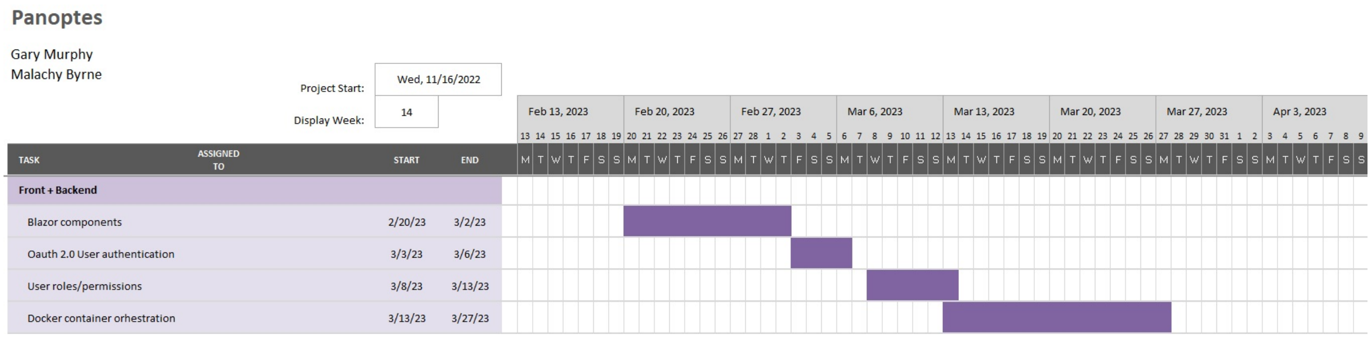
Panoptes

Gary Murphy
Malachy Byrne

Project Start: Wed, 11/16/2022

Display Week: 8

TASK	ASSIGNED TO	START	END
Modules			
Module Installation		1/4/23	1/18/23
Module Control		1/19/23	1/29/23
Module Permissions		1/24/23	2/3/23
test Module 1		2/4/23	2/8/23
Test Module 2		2/9/23	2/19/23



Panoptes

Gary Murphy
Malachy Byrne

Project Start: Wed, 11/16/2022

Display Week: 14

TASK	ASSIGNED TO	START	END
Front + Backend			
Blazor components		2/20/23	3/2/23
Oauth 2.0 User authentication		3/3/23	3/6/23
User roles/permissions		3/8/23	3/13/23
Docker container orhestration		3/13/23	3/27/23