Functional Specification

PROSPECTOR

Maximizing the Value of Bare Metal Servers

Fourth Year Project

James Hackett	20308896
Alexandru Dorofte	20414772
Supervisor	Stephen Blott

Table of Contents

1. Introduction

1.1 Overview

Prospector is a user management and infrastructure-as-a-service tool that enables easy, on demand deployment of jobs in the form of containers and virtual machines.

Users can sign up and create jobs on Prospector where they can choose to have their files stored on a network storage device and have those files mounted into containers or VMs when they are created. Jobs can be exposed to the internet if the user desires. The user will have an option to use common/useful recipes to deploy a job, such as a redis database or a web server.

The system will provide a platform for users to run their personal projects and services without the need for administrators to intervene, cutting the overhead of managing user jobs on a cluster.

Prospector will be written in Go and TypeScript using the Angular framework. It will use

Nomad to orchestrate jobs.

The name for our project came from the name given to the people who tried to extract maximum value out of bare land during the gold rush. We felt this was appropriate as we are trying to extract maximum value out of bare metal servers.

1.2 Glossary

Term	Definition
Docker	Docker is a platform to create and manage applications inside a virtual environment. Provides little isolation from the host operating system.
VM	A virtual machine is an environment that can be used to run operating systems. It offers complete isolation from the underlying operating system.
Nomad	Nomad is a cluster manager and scheduler that can be used to deploy and manage jobs on a cluster.
Job	A job is a container or virtual machine that is deployed on the cluster.
Traefik	Traefik is a reverse proxy and load balancer that can be used to route traffic and expose jobs to the internet.
Recipe	A recipe is a pre-configured job that can be deployed by a user.

2. General Description

2.1 Product / System Functions

Creating a user

A user can be created by an administrator or by the user themselves. There will be a CLI command and a signup form to allow each to sign up respectively.

Creating a job

A job can be created by a user or an adminstrator through a web UI. There will be options to specify the image or operating system to use.

Modifying a job

A job can be started, stopped, restarted, or deleted through the UI by a user, or via the CLI by an administrator.

Exposing a job to the internet

Jobs can be exposed to the internet at a subdomain of the project’s domain (prospector.ie (http://prospector.ie)).

Mounting a storage volume to a job

Users will be able to mount a file system into their jobs via a network file share.

Deleting a job

Jobs can be deleted by users and administrators via the web UI or via the CLI respectively.

2.2 User Characteristics and Objectives

Prospector is aimed at both technical and non-technical users. By providing a web UI, users unfamiliar with virtualised or containerised workloads will be able to easily create and manage their jobs, while more technical users will have a space to run their projects without the need for administrator intervention. In this way, we hope to cater to a wide range of users and provide a platform for them to run their projects.

2.3 Operational Scenarios

User creates an account and logs in

User can create an account by using the CLI or by signing up on the web UI. They will then be able to log in to the web UI.

Administrator configures a group of users

Admininistrator can create a group and assign users to that group. They can then impose limits on the group such as the number of jobs a user can have running at once, the amount of storage/compute and if they can expose their jobs to the internet. They can also create a namespace for the group in which resources are shared between users in the group.

Administrator imposes limits on users

Admin can select a user or user group within their dashboard and impose limits. These can be limiting the number of jobs a user can have running at once, the amount of storage/compute and if they can expose their jobs to the internet. Administrator can also view the status of all jobs.

User Configures a job

User will be presented with a dashboard where they can select to create their job. They will be presented with an option to select a container or VM. After filling in the required fields they can deploy their job. The user will then be presented with a dashboard where they can view the status of their job and change settings on their job.

User wants to host a website

User can create a job of type container and specify an image that will run a web server containing their web files. They can then use the UI to expose their job to the internet.

User wants to host a multi-service project

User can create a job of type virtual machine and configure it to their needs. After it is deployed the user will be able to ssh into the VM and work inside this environment to host their project.

User wants to modify and view status of their job

User can view the status of their jobs and change settings on their jobs. By viewing their running jobs and click on a deployed job to access the extended settings such as increasing compute or viewing performance metrics.

2.4 Constraints

Networking

There will be a lot of moving parts in this project, in both the management of the cluster and in running the jobs on the cluster. As we will have a frontend and backend, plus an LDAP server, network mounted storage devices, nomad and consul all running on the cluster, we will need to ensure that the network is configured correctly to allow all of these services to communicate with each other when needed, and are isolated from each other otherwise.

Hardware

As this project is designed to run user jobs of various types, the underlying hardware will need to be powered on and available around the clock. This may pose a challenge to operators running this on small scale clusters, but should not be an issue for larger clusters with many availability zones.

Security

If users decide to store sensitive information in Prospector, it will be important to ensure that the cluster is secure. This will involve ensuring that the network is configured correctly, and that the cluster is configured to only allow users to access their own jobs, storage allocations and other resources. We’ll also need to ensure that jobs are only exposed when explicitly requested by the user. This is closely tied with the networking constraint.

3. Functional Requirements

Job Management	
Description	The core feature of Prospector is management of jobs for users - creation, modification and deletion.
Criticality	High
Technical issues	Virtualisation and containerisation issues, particularly around networking and the persistence of those jobs
Dependencies	N/A

User Management	
Description	Users should be able to sign up and log in to Prospector. Administrators should be able to manage users
Criticality	High
Technical issues	Ensuring that user permissions are consistent between the UI and the deployment of their jobs will be a challenge
Dependencies	N/A

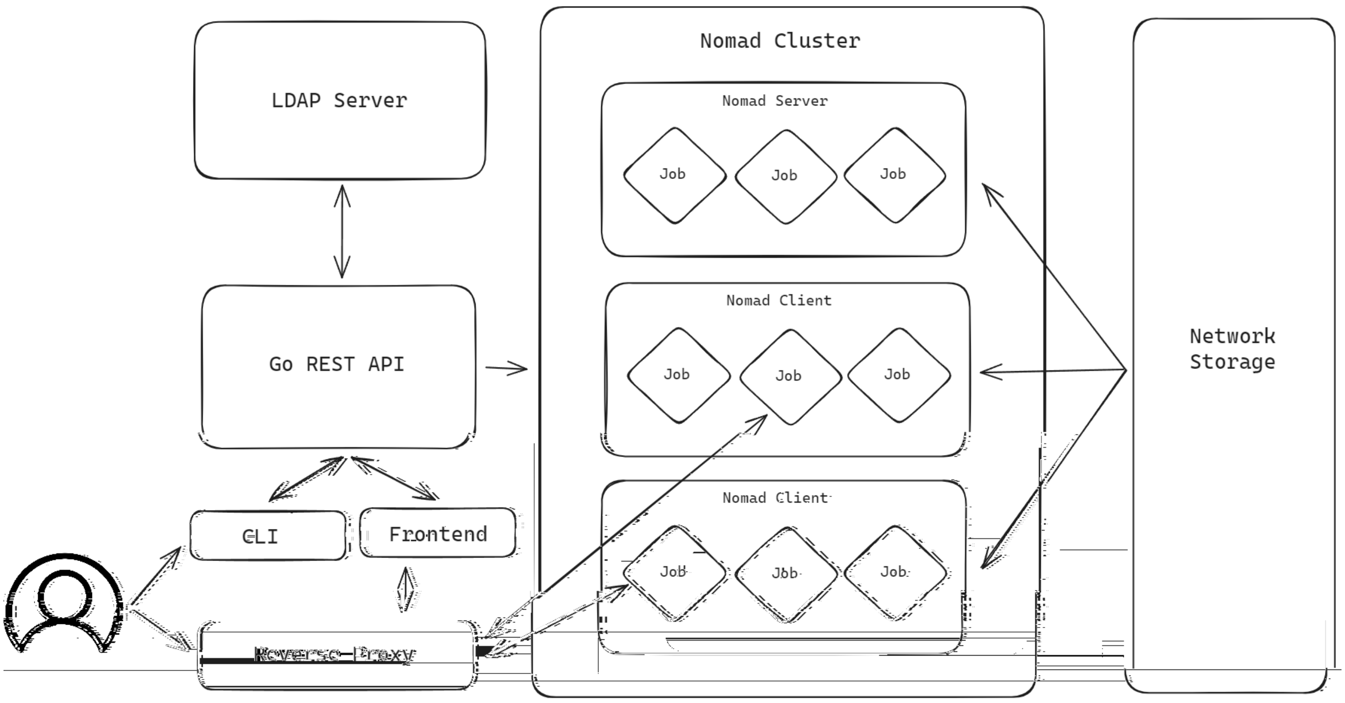
Networking	
Description	Jobs, whether exposed to the internet or not, will have to send and receive traffic with ease
Criticality	High
Technical issues	Ensuring that the network is configured correctly to ensure jobs are secure
Dependencies	Job management, user management

User Interface	
Description	Users should be able to interact with Prospector through a simple and intuitive UI. It should not hide any functionality
Criticality	High
Technical issues	Ensuring that the UI is consistent with the backend and that it is easy to use and understand
Dependencies	Job management, user management

Storage Management	
Description	Users should be able to mount storage volumes to their jobs. Administrators should be able to manage storage
Criticality	High
Technical issues	Ensuring that storage volumes are mounted correctly and are consistent across jobs
Dependencies	Job management

Job Recipes	
Description	Users should be able to select a recipe to deploy a job. Administrators should be able to manage recipes
Criticality	Low
Technical issues	Making a recipie as generic as possible to ensure a wide range of uses
Dependencies	Job management, storage management

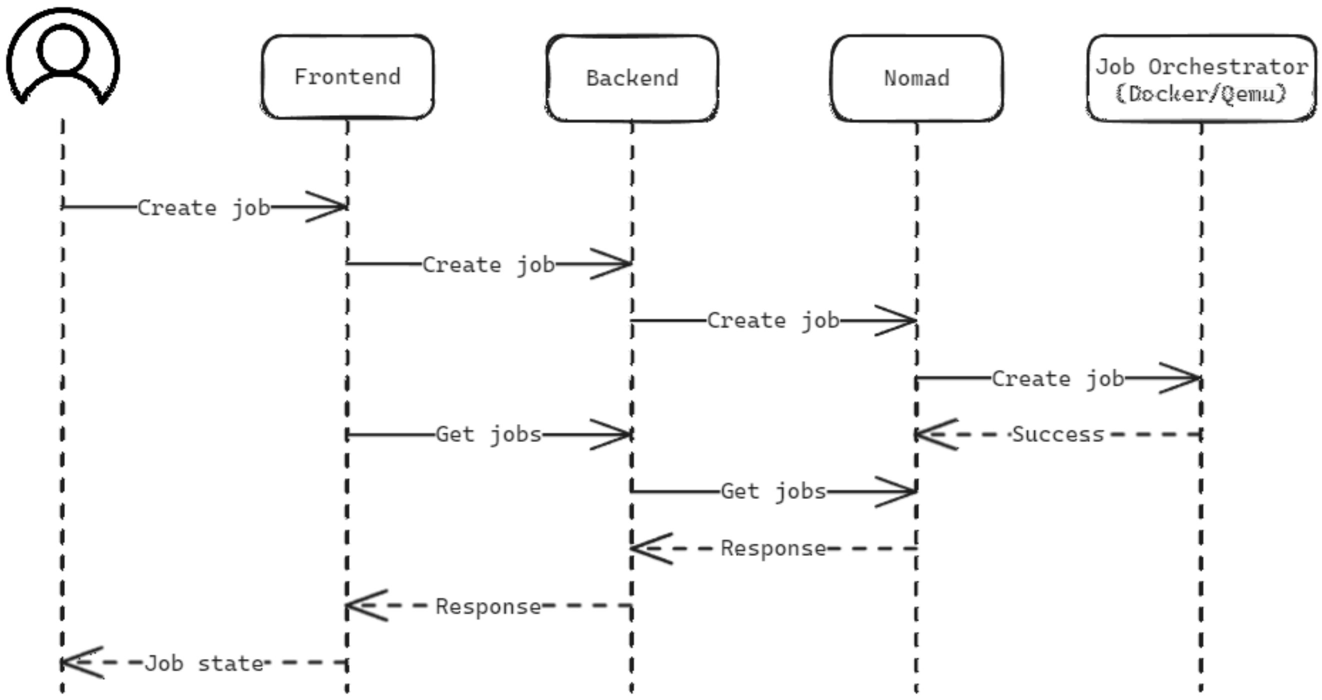
4. System Architecture



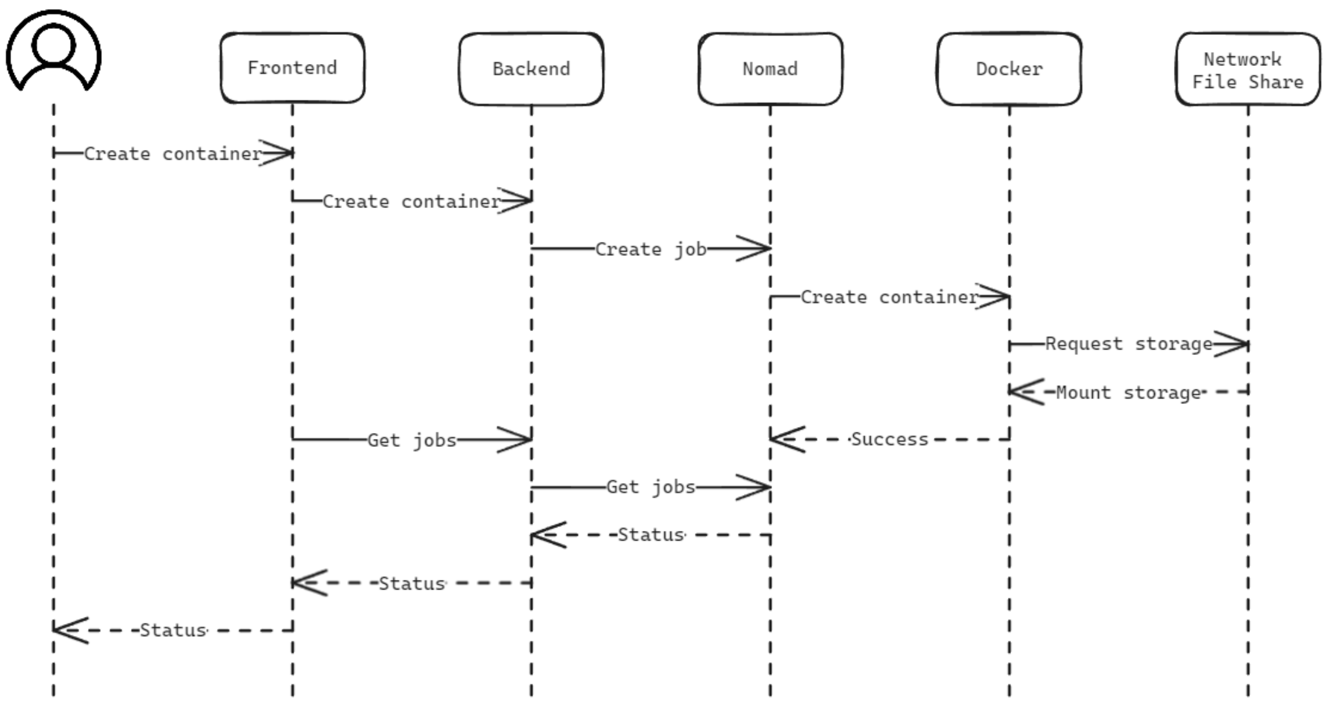
5. High-Level Design

Sequence Diagrams

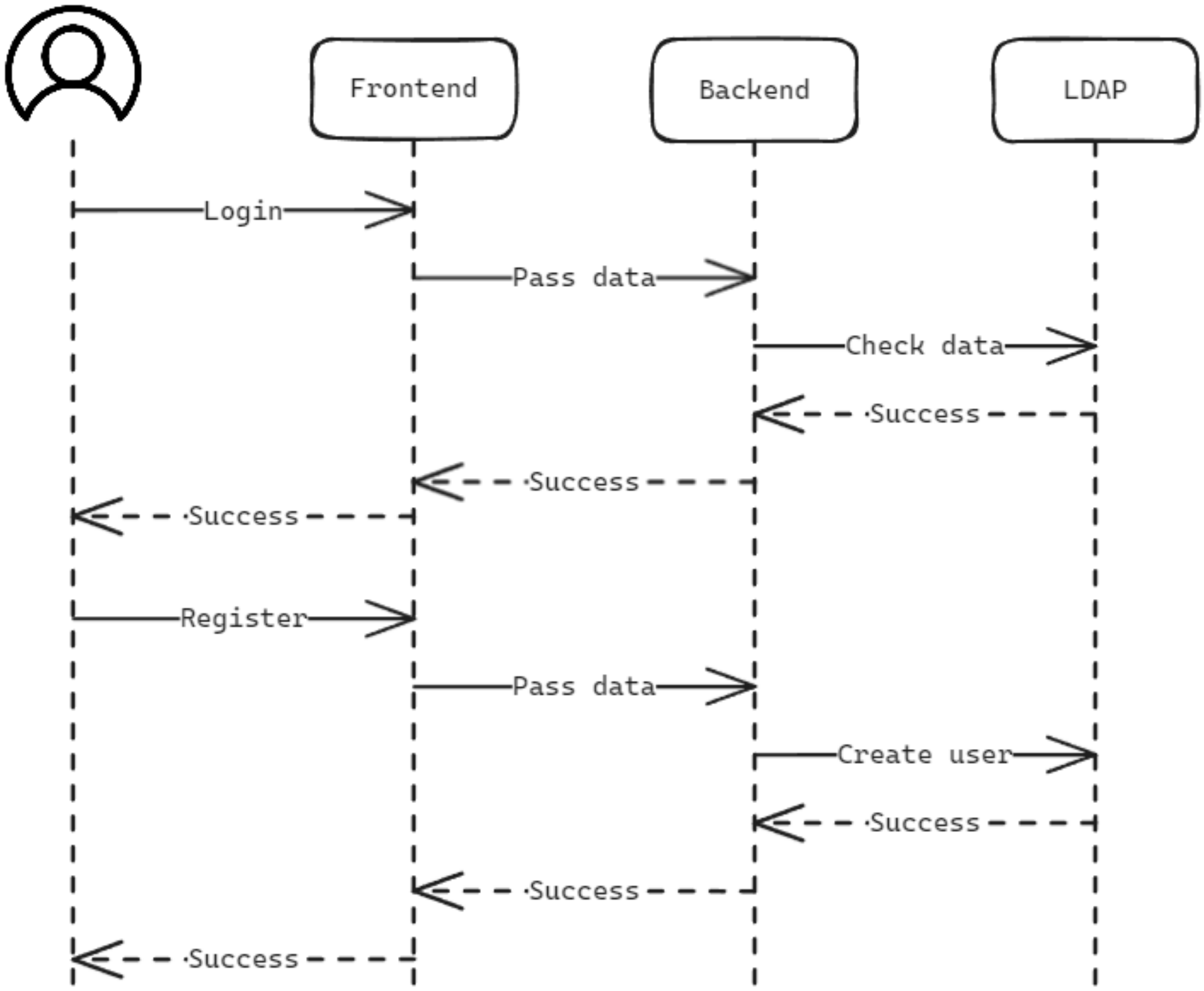
Generic job create



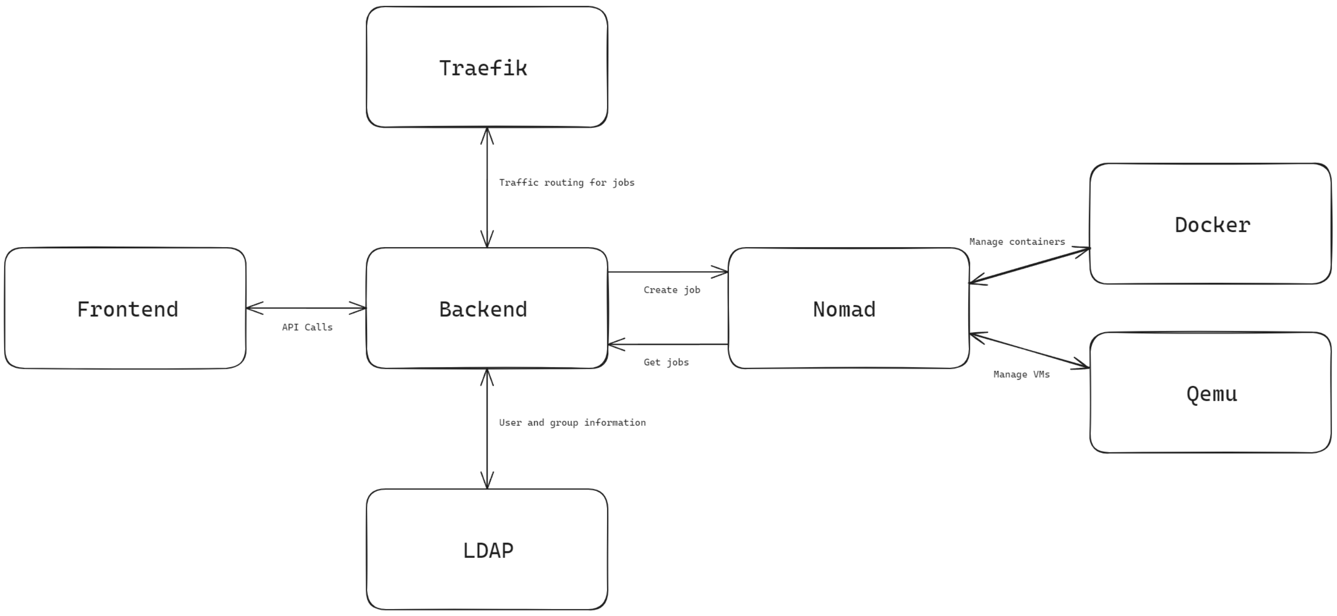
Job create container with storage



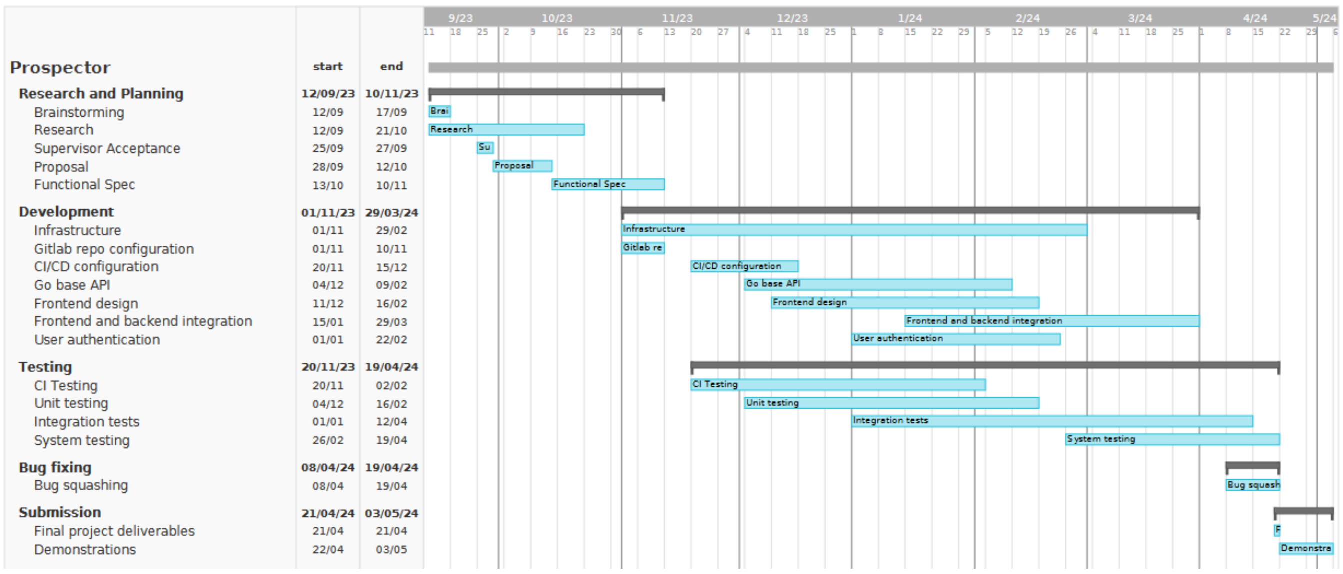
User login/create



Data flow diagram



6. Preliminary Schedule



Prospector	start	end
Research and Planning	**12/09/23**	**10/11/23**
Brainstorming	12/09	17/09
Research	12/09	21/10
Supervisor Acceptance	25/09	27/09
Proposal	28/09	12/10
Functional Spec	13/10	10/11
Development	**01/11/23**	**29/03/24**
Infrastructure	01/11	29/02
Gitlab repo configuration	01/11	10/11
CI/CD configuration	20/11	15/12
Go base API	04/12	09/02
Frontend design	11/12	16/02
Frontend and backend integration	15/01	29/03
User authentication	01/01	22/02
Testing	**20/11/23**	**19/04/24**
CI Testing	20/11	02/02
Unit testing	04/12	16/02
Integration tests	01/01	12/04
System testing	26/02	19/04
Bug fixing	**08/04/24**	**19/04/24**
Bug squashing	08/04	19/04
Submission	**21/04/24**	**03/05/24**
Final project deliverables	21/04	21/04
Demonstrations	22/04	03/05

7. Appendices

- Nomad (https://www.nomadproject.io/)
- Traefik (https://traefik.io/)
- Docker (https://www.docker.com/)
- Qemu (https://www.qemu.org/)