# UniGuideOnline

William Hadden - 6249537

October 6, 2022

## Introduction

This app provides functionality for a shared resource. In this case, papers at university. The idea is that users can see paper information generated by other users.

## Architecture

## 1 High level overview

This application is served by the interaction of two webservers. One webserver provides functionality for viewing papers while another provides functionality for creating papers. In order to store the papers persistently, a NoSQL database is used in order to retain heirarchy information (i.e how the papers depend on each other). To this end, there exists a CRUD API that takes care of these use cases to allow both to function. Further, there is a serveless function that is called by the api to take care of database interaction.

## 2 Justification of architecutre

Two benefit of splitting the user functionality is two fold. Load balancing: likely that there will be more visualising users than submitting users so can provision for this effectively.

Logically these are quite seperate concerns so it makes it easier to program that way In the future, will likely have references to either server using href keys

Allows effective and easy interaction with database/information, which is -in itself - customisable.

Don't have to have an API server running on the time managing requests. Can nicely scale and trigger cutom code only when it is required. (So saves lots of money). ALso means we can have simple interaction with the API Gateway (don't have to worry about what's going on in the background)

# 3 Techical Specifications

This application is provisioned soley with AWS. The webservers are provisioned using EC2 instances. Serverless functions as Lambda functions.

API as API Gateway.

# User Interaction

# Development story

Spent a lot of time trying to get to work. Basically was defeated by the IAM role that had to be assigned to certain resources would lead t me being eother locked out of certain functionality (i.e I couldn't see my functions on the AWS GUI) and the terraform destroy failing because the IAM user couldn't be deleted with it.

Clearly I was always going to need a nice way to interact with the differnet components. I decided to have a serverless function so that I didn't have to run another virtual machine which would be both expensive, slow and difficult to maintain (especiialy without CI/CD)

Would make the startup and development some much faster and nicer. However, learner labs don't support it

# Cloud specific information

# Future improvements

Not possible with learner lab so would need own account but would make this application x100 better

Creation of user accounts such that the priviliges for interacting with resources (such as papoers) can be restrivtd so only their owners can interact with them

Use of a CDN and load balancing would increase the reliability of the app and allow it to scale out

Would like to have another go at this but again, doens't really work with learner lab

Href into either through a authenication/user dialog to better increase the app

Might actually do this because it makes it nice