Connor Morrison

RASPBERRY PI WEB SERVER ARCHITECTURE

Contents

[Server Architecture 2](#_Toc169221977)

[Key Requirements 2](#_Toc169221978)

[Non-functional Requirements 2](#_Toc169221979)

[Hosted Projects ASRs 2](#_Toc169221980)

[Hardware 2](#_Toc169221981)

[Dependencies 2](#_Toc169221982)

[Server and Delivery 2](#_Toc169221983)

[Front-end 3](#_Toc169221984)

[Back-end 3](#_Toc169221985)

[External Dependencies 4](#_Toc169221986)

[Architecture 4](#_Toc169221987)

[Diagrams 4](#_Toc169221988)

[Questions 4](#_Toc169221989)

[Projects 5](#_Toc169221990)

[Portfolio 5](#_Toc169221991)

[Abstract 5](#_Toc169221992)

[Key Requirements 5](#_Toc169221993)

[Content 5](#_Toc169221994)

[Prompt Engineering 7](#_Toc169221995)

[Game Style Form 7](#_Toc169221996)

[Abstract 7](#_Toc169221997)

[Key Requirements 7](#_Toc169221998)

[Virtual Chatbot 8](#_Toc169221999)

[Abstract 8](#_Toc169222000)

[Key Requirements 8](#_Toc169222001)

[Biometric 2FA 8](#_Toc169222002)

[Abstract 8](#_Toc169222003)

[Key Requirements 8](#_Toc169222004)

# Server Architecture

## Key Requirements

#### Non-functional Requirements

* Availability
* Reliability
* Lightweight

### Hosted Projects ASRs

#### Portfolio Display

* Must be responsive suiting both mobile and web displays
* Must be dynamic, loading elements as the user scrolls
* Must be modular, new elements must be able to be created and recreated with ease.

#### Game-style Form

* Must submit data to a persistence database

#### Virtual Chatbot

* Must be lightweight as to not deplete the server’s resources

## Hardware

#### Raspberry Pi 4

* CPU – Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.8GHz
* RAM – 8GB LPDDR4-3200 SDRAM

## Dependencies

### Server and Delivery

#### Dynamic Domain Name Service – No-IP NameCheap

I plan to begin with No-IP’s free DDNS service for initial development with later planned integration to NameCheap’s DDNS service with custom IPs as this will improve accessibility and the user experience of the website giving a more professional experience to viewers.

#### Web Server Software – NGINX

NGINX was chosen for two main reasons, it’s highly lightweight architecture and for its popularity. The lightweight architecture was preferred due to the limited resources on the Raspberry Pi. The other reason was popularity, as I was inexperienced in working with NGINX it was crucial to have sufficient documentation and tutorials to learn as the project grew. One other main option was considered, Apache, this was rejected due to its heavyweight size that would be too demanding on the resources of a Raspberry Pi 4.

#### Containerisation – Docker & Docker Compose

Docker was chosen to containerise each microservice, the issue with this was from the container overhead, which may stress the CPU. However, it was decided to focus on using lightweight frameworks in each stage and implement restrictions on how much memory and CPU each container has access to. With Docker compose to manage the many containers used for the microservice architecture it is a sufficient choice.

### Front-end

#### Front-end Library – React

React was chosen as a front-end library due to the importance of modularity to future development, allowing for the creation of repeatably used assets. Also, React is a popular library with significant user support, this means solutions will be easier to research and work towards.

#### CSS Library – TailwindCSS + daisyUI

TailwindCSS was chosen as the most suitable CSS Library, this was for two main reasons. Tailwind offers more customisation over elements whilst still allowing to serve the practicality of a CSS Library, allowing for a more personal user experience. The other was that Tailwind can be combined with daisyUI to deliver predefined elements with ease. The main competitor considered was Bootstrap, however this was not chosen as apps developed with Bootstrap always look like a “Bootstrap app”, I personally wanted more customisability in my design choices than that, hence Tailwind.

### Back-end

#### Back-end Framework – Flask

Flask was chosen as a backend framework for its lightweight design and my Python language skills. Although Flask may take more work implementing what I need for each service as I develop this will be rewarded with a much more lightweight app which can be hosted on the Raspberry Pi.

#### Database – SQLite

This database was chosen due to its lightweight architecture however if the app were needed to scale it may be worth considering using a Redis database where appropriate to allow for frequently accessed data to be cached for quick access, although it may be more beneficial to add more Raspberry Pis to scale horizontally in this intense usage.

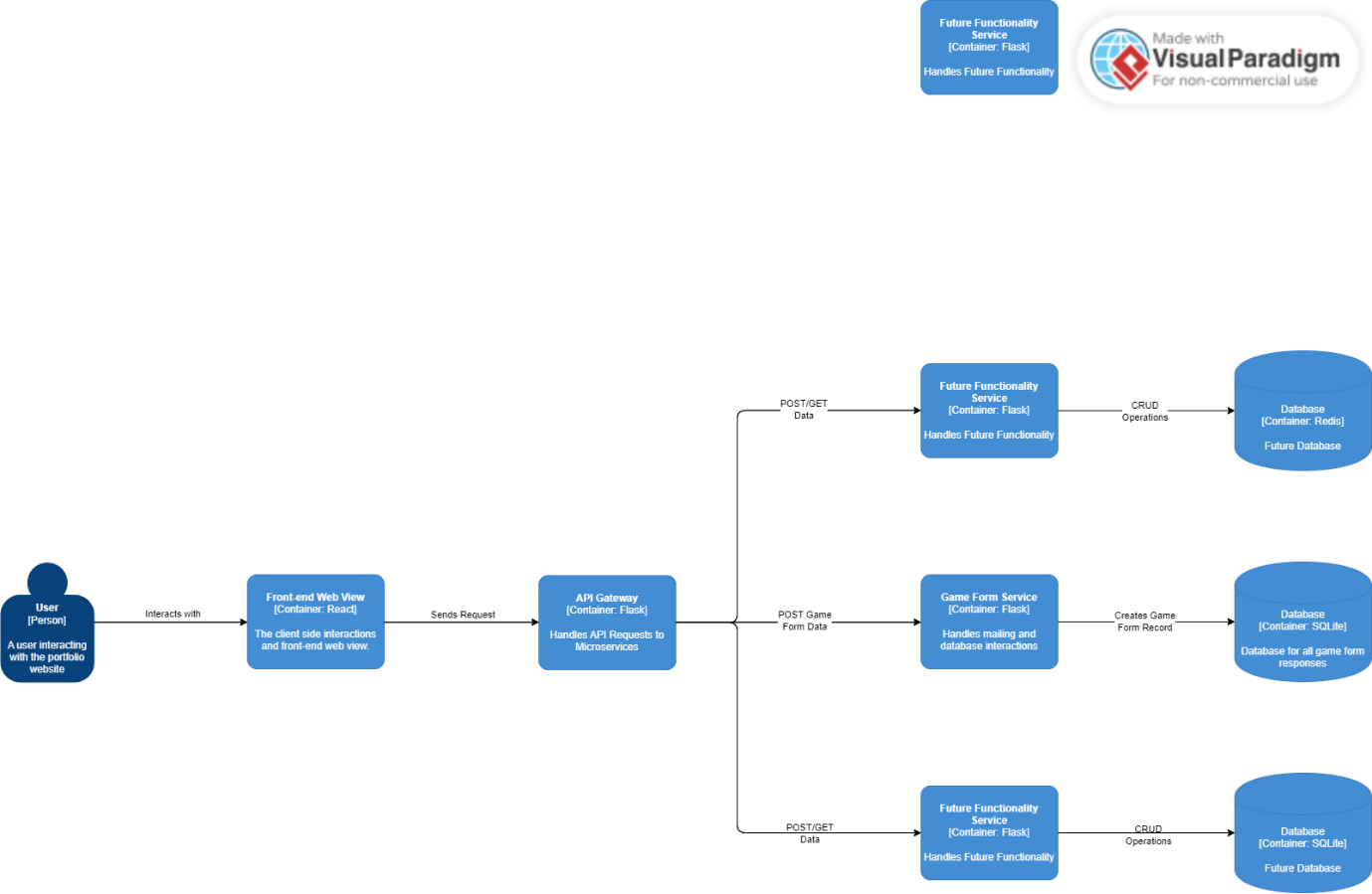
### External Dependencies

#### Chatbot - Azure AI Bot Service

The Azure AI Chatbot offers a free service for up to 10,000 messages (HTTP requests) per month and is always free. Another reason why this was chosen over hosting an open source Chatbot like Radas or Botpress is due to the resource demand these AI models would put on the Raspberry Pi.

## Architecture

### Diagrams



## Questions

* How to best address certain security concerns? Reference OWASP and Vulnerability Testing Tools???
  + Form input (XSS and SQL Injection)
  + Firewalls

# Projects

## Portfolio

### Abstract

The Portfolio project refers to the main static website, it will hold information about my professional profile.

### Key Requirements

#### Functional Requirements

* Must be modular, allowing for new sections to be added and dropped with ease.
* Must display all information in the Content section below
* Must be responsive suiting both mobile and web displays
* Must be dynamic, loading elements as the user scrolls

#### Non-functional Requirements

* Must have a sleek user interface

### Content

#### Work Experience

* Software Developer at Department of Employment and Workplace Relations (Feb 2024 – Present)
  + My role as a software developer at DEWR involved maintaining and extending the Workforce Australia website at https://www.workforceaustralia.gov.au/

#### Projects

* Raspberry Pi Web Portfolio (Jun 2024 – Present)
  + This is my custom-built web server which you are now viewing this page on. It was designed to host many projects to view as a portfolio.
* Gourmet - Recipe Sharing App (May 2024 – Jun 2024)
  + Gourmet, an innovative cooking application designed to transform how food enthusiasts create and enjoy recipes. This project, developed as part of the CSSE6400 course, leverages modern technologies to provide a seamless and interactive cooking experience.
* MenuScan (Mar 2024 – May 2024)
  + As part of my coursework in Web Information Systems (INFS3202) course at the University of Queensland, I developed "MenuScan," a Software as a Service (SaaS) platform designed to streamline the ordering process for restaurants, cafes, and coffee shops. This project was built using PHP with the CodeIgniter framework, following the Model-View-Controller (MVC) architecture. MenuScan allows businesses to create and manage digital menus, automatically generating unique QR codes for each table, enabling guests to access the menu and place orders directly from their smartphones.
* 2032 Brisbane Olympics Extended Reality Experience (Jul 2023 – Oct 2023)
  + While enrolled in DECO3801 at the esteemed University of Queensland, I had the privilege of collaborating with a talented group of developers to create an immersive Extended Reality (XR) experience. Our collective aim was to cultivate and heighten the excitement of ticket purchasers eagerly anticipating the forthcoming Brisbane 2032 Olympics in our immersive XR prototype.
* Gardens of the Galaxy (Jul 2023 – Oct 2023)
  + During my enrolment in the rigorous CSSE3200 program, I had the privilege of collaborating with an extensive team of approximately 40 skilled developers. Together, we undertook the formidable task of creating a sophisticated and distinctive Java-based game that embodied the pinnacle of quality and comprehensiveness.

#### Skills

* Python (Strength 5/5)
* Java (Strength 5/5)
* PHP (Strength 3/5)
* C# (Strength 3/5)
* HTML (Strength 5/5)
* CSS (Strength 4/5)
* JavaScript (Strength 4/5)
* SQL (Strength 4/5)
* RStudio (Strength 2/5)
* Linux
* Terraform
* Docker
* Amazon Web Services
* NGINX
* Microsoft SQL Server
* Git
* Figma
* Node.js
* Flask
* .NET
* Vue.js
* React
* React Native
* Bootstrap
* TailwindCSS

#### General Development Skills

* Cloud Computing
* Full-Stack Web Development
* RESTful API Development
* Containerisation
* Object Oriented Programming
* Testing
* Problem Solving
* Cyber Security

#### Education

* Bachelor’s degree in computer science from UQ (2021 – 2024)
* Certificate II in Information, Digital Media, and Technology (2019 – 2020)

#### Certification

* Google Cybersecurity Professional Certificate
* Mozilla JavaScript Professional Certificate
* OpenEDG Python Professional Certificate
* Docker Professional Certificate

#### Other

* Security Clearance – Negative Vetting 1

### UX Design

#### Requirements

* Animations
* Dynamic Elements
  + As the page loads more elements should appear
* Responsive Design
* Display
  + Home
  + Projects
  + Skills
  + Education
  + Contact + Links

#### Examples

* <https://cade.codes/>
  + Elements load progressively as the user scrolls
  + Size demonstrates importance of certain elements
* <https://www.benrogers.dev/#home>
  + Floating navbar
* <https://jackharris.dev>
  + Background Animation
  + Dynamic Job Role Text

## 

## Game Style Form

### Abstract

Think of a falling block/ball (addressed as the Selector) game that has buckets the player can direct the Selector into that have replies to form questions that can be sent back to the database. Once questions have been answered, users can input their name in a game like keyboard to add their “high score” to the board (every player gets the high score) (this “adding their score” submits to the data to the database).

### Key Requirements

#### Functional Requirements

* Must submit data to a persistence database
* Upon submission an email should be sent to me with the users replies

#### Non-functional Requirements

* Must have a fun colourful UI (think like Tetris I reckon)
* Must have an easy to use and practical control scheme
* Must seamlessly fit in with the user interface
* Mustn’t cause lag for the page

## Virtual Chatbot

### Abstract

A virtual chatbot to answer questions relating to my experience in computer science and my qualifications + experience.

### Key Requirements

#### Functional Requirements

* Must reply within 5 seconds of asking a question

#### Non-functional Requirements

# Future Projects

## Biometric 2FA

### Abstract

### Key Requirements

#### Functional Requirements

#### Non-functional Requirements

# Pre-Deployment Checklist

* Get Custom Domain
* Get a SSL certificate
* Dynamic Domain Name Service/Get Static IP
* Host NGINX Server
* Finish Static Website
* Finish Game Form
  + Interactive Game Form Front-End Implementation
  + Rate Limiting Security Integration
  + Player Decision Handler APIs
  + Database to store player data
* Integrate Chatbot into Website