Abertay University

CMP408 - Pi Quiz Application

Mairi MacLeod - 1700231

Introduction

This project combines a Python 3 Flask web application, a Raspberry Pi Zero and an AWS RDS POstgresql database.

The aim was to create a quiz application that could be answered using the Pi and the web application with data being stored on the cloud,

The main objectives of this project were:

> Create web application
  > Allows user to enter a username
  > Question answers randomised
  > High scores shown at the end
> Get input from Pi buttons to answer questions
> Use coloured LEDs to display answer
> Create database on AWS to store high scores and question

Methodology

AWS RDS Database

An RDS Postgresql instance was created in order to store users' names and scores. A database to store this information was chosen in order to allow all users of the application to have access to the latest up-to-date version of the quiz at all times. The database is comprised of two tables; HighScores and Questions. A database helper script titled createDB.py was created to make, populate and delete data from the tables as appropriate.The RDS instance requires both password and IAM authentication. To control access to the database a Virtual Private Cloud (VPC) was created that restricted ingress traffic. All data required for the authentication is imported from a separate python file,envs.py that is included in the gitignore, so that it would not be uploaded to GitHub and remain private

Flask Web Application

The web application is the heart of this project and connects all of the separate components together.
It is comprised of the class; QuizClass and the functions; index, getUsername, quiz and highScores.
QuizClass gets a question and corresponding answers from the database, shuffles the answers' order and returns them. Due to multiple functions being required to essentially do one thing the author decided to place them all together in a class. Index()renders a form for a user to enter their username. getUsername() takes this username and puts it in the database. quiz() displays the questions, answers and current score. It also checks the answers and updates the scores accordingly. highScores() displays the top ten high scores and updates the database with the users final score.

Raspberry Pi Zero

On the Pi there are four LEDs, three blue which represent incorrect answers and a green one that flashes when the user gets a correct answer. In order to get the Pi to be able to communicate with the LEDs the kernel object was created by using the command make on a C file containing driver code.

Project Highlights

Web application mostly works and displays the quiz questions, answers and accepts user inputs.

The RDS database and VPC are properly configured and are easily accessible via the createDB database helper script.

Future work

LEDs do not work in the script, the component configuration is fine it is just syntax issues in the python script. Buttons can be added in future to meet all of the original objectives

HighScores do not display after the quiz ends so that should also be fixed at a later date.

References:

Amazon Web Services (2020).Amazon Relational Database Service (RDS)[online] Available at:https://aws.amazon.com/rds/
Amazon Web Services Docs (2020) | Connecting to an Amazon DocumentDB Cluster fromOutside an Amazon VPC - Amazon DocumentDB. [online] Available at: https://docs.aws.amazon.com/documentdb/latest/developerguide/connect-from-outside-a-vpc.html | Raspberry Pi (2020).Kernel building[online]. Available at :https://www.raspberrypi.org/documentation/linux/kernel/building.mdd
SuperMairio (2020).Pi Quiz[online] GitHub. Available at:https://github.com/SuperMairio/Pi-Quiz