Quick start Guide

Hethena - A reading device for the visually impaired

Setting up the Kubernetes cluster

start a k3d cluster using the terminal after ensuring that Docker, k3d and kubectl are installed using the command k3d cluster start <cluster-name>

Attached in the source code zip folder are 4 yaml files:
- deployment.yaml
- secrets.yaml
- service.yaml

We apply the secrets, deployment and service files on our cluster using the commands:
kubectl apply -f <path-to-deployment-file>
kubectl apply -f <path-to-secrets-file>
kubectl apply -f <path-to-service-file>

Once this is done open a browser and type in http://localhost/ to verify the cluster runs. It should show you a welcome screen displaying Hello World.
Now run 'ipconfig' to get the public IP address of your machine and note it down.
Now try running http://<public-ip-address>/ this should also be running.

The Kubernetes cluster has the code for the backend-app running on it as we containerized the backend-app using docker and set it up using Kubernetes cluster. If needed, you can take a look at the code for the backend-app by navigating inside the folder that says backend-app.

Setting up the mobile application

cd into the mobile-app directory

Next run the two commands in your terminal

1. npm install
2. npx expo start

To view the app, you will need to download the "expo go" application from the app store/google play. Then you can scan the qr code to access the app. The laptop and the mobile phone have to be using the same wifi.

Once you are able to access the app, login to the app using these credentials:
Username: jamie@gmail.com
Password: testing

Setting up the hardware

The esp-code folder in the source code files has the code to flash on the hardware. Go to the main/wpa2_enterprise_main.c file and make these 2 changes:

```
158  // Function to send an HTTP POST request
159  int send_http_request() {
160      const char *hostname = "192.168.230.152";
161      char post_data[256];
162
```

Replace hostname here with your own public IP address that we got when setting up Kubernetes.

```
293      wifi_config_t wifi_config = {
294          .sta = {
295              .ssid = "ONEPLUS_co_apsxar",
296              .password = "xxxxxxx"
297          },
298      };
```

Also replace the wifi configurations with your own ssid and password.

Once this is done you can open up a ESP-IDF terminal and flash the code to your esp device by running idf.py flash and idf.py monitor.

The setup is complete now you should be able to take a photo by clicking the button and the camera will then capture the image, send it to the Kubernetes cluster that processes it and generates audio in the user preferred language. This audio is then played on our mobile application.