Automated Malnutrition Detection : Documentation

CONTENTS

PROGRAM MODULES

FRONTEND

Upon starting the application, the user will be prompted to register themselves or login with their credentials. The user can login/register as **ground personnel** or an **administrator.** (using a menu in the registration page)

After logging in, the user is redirected to the homepage, which contains a dashboard. The dashboard has the following components:

1. Calibrate: On this page, the user can update the calibration image used for height detection.

2. Patient registration: The user can register a new patient by uploading his/her details (name, age, gender, region). Once a patient is registered, his/her data can be used for identification in "upload data."

3. Upload data: This is the page where the image is uploaded. The user uploads the image taken of the patient's observations, along with identifying details of both the patient and the user.
   Once the picture is uploaded, it is processed in the backend, and the result of the image is then displayed on the screen.

4. View results: The user can enter the details of a particular patient and view all the results of that patient.

5. All results *(administrator only)*: The administrator can view the names of all the patients and can view the results of those patients as well.

BACKEND

There are 2 backend modules in our program: a **Flask** backend and a **MERN (MongoDB and Express)** backend.

1. **Flask backend**
   This backend is accessed exclusively by the MERN backend through API calls.
   It has 3 main functions:
   I. Image input: it receives base64 encoded strings of the observation image and the calibration image from the API call. It converts them into jpeg files which are then used for calculation and detection.

   II. Perform calculations: it executes the height detection algorithm (using checkerboard detection) and weight detection algorithm (using OCR)

III. Check patient status: from the detected height and weight, the Flask backend checks them against the corresponding WHO chart (determined by gender and age group), and returns the nutritional status and BMI of the patient.

2. **MERN backend**

It is accessed using the frontend through API calls. It is also connected to the MongoDB Atlas database via Mongoose.

The MERN backend has three main types of functions:

I. **User functions:**

- User registration: when administrator and ground personnel register for the first time, their data is stored in the MongoDB Atlas database.

- User login: The user logs in using their authentication details, and the authentication is verified using a JWT token.

II. **Patient functions:**

- Patient registration: When a patient is registered by the current app user, his/her details are stored in the MongoDB Atlas database.

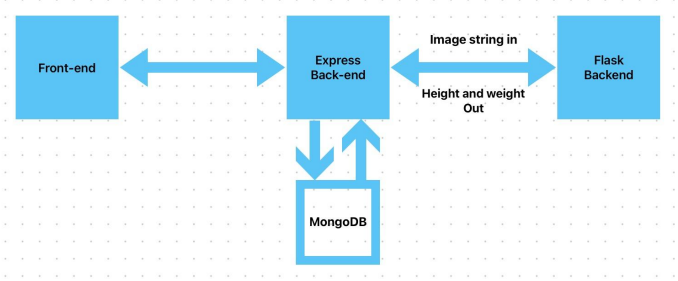
- Data retrieval (for one/multiple patients): Depending on the requirements (view result/all results) all data of a single patient/all patients can be retrieved from the database.

III. **Data functions:**

- Add data: the app-users can add new data for new patients.

- Retrieve data: All uploaded data (height, weight, nutrition status) of a particular patient can be retrieved as required.

Following is a picture of the communication flow between the frontend and the 2 backends during the running of the application.



Algorithms Used

1) Height-Detection

We can obtain the height using a reference object (here a checkerboard pattern) placed on the ground. Using CV, we can detect the reference object placed on the ground. cv2.findChessboardCorners() is a function of cv2 that helps in detecting the corners of a checkerboard.

There are two steps:-

a) Calculating the focal length of the camera

In this step, using the known actual distance and actual width of the reference object we can get the focal length of the camera using the formula:-

F = (P x D) / W
F = Focal Length
P = Apparent Width = Length of the Diagonal of the checkerboard.
D = Actual Distance
W = Actual Width

We are using a calibration image for this.

b) After calculating the focal length, we can apply the same above formula to calculate the actual distance of another image, as the focal length of a particular camera remains fixed.

2) Weight Detection

For weight detection, OCR (Optical Character Recognition) was used. Specifically, we made use of the PaddleOCR tool by PaddlePaddle for OCR.

Installation and setup instructions

1. Setting up the Anaconda environment:
    a. Install Anaconda using the link: https://docs.anaconda.com/anaconda/install/
    b. Make sure that the python version of your Anaconda installation is 3.8>= ,otherwise there will be errors due to certain dependencies.
    c. Create a development environment (which contains nodejs) using:
    `conda create -n <environment name> nodejs python==3.8.3`
    d. Activate the environment using:
    `conda activate <environment name>`

2. Setting up PaddleOCR within Anaconda:
    a. Follow the instructions given in this website to install PaddlePaddle: Installation on Windows via Conda-Document-PaddlePaddle Deep Learning Platform
    b. Install PaddleOCR using `pip install "paddleocr>=2.0.1"`
    c. In case of errors with the “protobuf” dependency (which might occur), install the version of “protobuf” which appears in the error message.

3. Setting up MongoDB (MongoDB Atlas was used in development)
    a. To set up MongoDB Atlas, follow the instructions given here: Get Started with Atlas — MongoDB Atlas
    b. Obtain the MongoDB connection string from the Atlas website. This string should be used in `backend/index.js` as the MongoDB connection string for the program (as the connection string will differ for each user)

4. Freeing up ports:
    Free up the following ports for the modules to run:
    - Frontend - 3000
    - Express backend - 3001
    - Flask backend - 8080

5. Running Flask backend module
    a. Navigate to the flask-backend folder
    b. In the folder, run: `python3 function_apis.py`

6. Running frontend and Express backend modules:
    a. If node_modules has not been installed in the modules: In each module, run
    `npm i`
    Ignore this step if node_modules is already present in these modules.
    b. Run the modules by using: `npm start`

Dependencies
Frontend module:

- react
- axios
- material-ui
- material-ui/x-date-pickers
- react-dom
- react-router-dom
- dayjs
- nodemon (for development - `devDependencies`)

Express backend module:

- axios
- bcrypt
- bcryptjs
- body-parser
- cors
- dotenv
- express
- express-async-handler
- jsonwebtoken
- mongoose
- nodemon (for development - `devDependencies`)

Flask backend module:

- Flask
- base64
- cv2 (`opencv` for installation)
- numpy
- paddleocr

References

1. https://www.who.int/childgrowth/standards/en/
2. https://github.com/PaddlePaddle/PaddleOCR
3. https://pypi.org/project/opencv-python/
4. https://docs.anaconda.com/anaconda/
5. https://material-ui.com/