# Medical Quiz Application - README

## Table of Contents



## Introduction

This project is a Medical Quiz Application developed for the Raspberry Pi platform. It leverages Flask for the web interface, `gpiozero` for interacting with GPIO pins, and SQLite for database management. The application allows users to select between "Information" and "Quiz" modes:

- **Information Mode**: Users can click on different parts of a human skeleton image to learn about various body parts, associated medical implants, and view QR codes for additional resources.
- **Quiz Mode**: Users answer multiple-choice questions using physical buttons connected to the Raspberry Pi. Correct answers trigger specific LEDs corresponding to body parts, while incorrect answers cause all LEDs to blink.

## Project Structure

```
css
Code kopieren
medical_quiz/
├── app.py
├── initialize_db.py
├── check_database.py
├── quiz.db
├── templates/
│   ├── base.html
│   ├── mode_selection.html
│   ├── main.html
│   ├── information_content.html
├── static/
│   ├── styles.css
│   ├── images/
│   │   ├── skeleton.png
│   │   └── [other images]
│   ├── qr_codes/
│       ├── heart_qr.png
│       ├── ear_qr.png
│       └── [other QR code images]
└── README.md
```

## Setup Guide

### 1. Hardware Requirements

- **Raspberry Pi** (any model with GPIO pins)

- **LEDs**: 8 LEDs representing different body parts
- **Resistors**: 220Ω resistors for each LED
- **Buttons**: 4 momentary push buttons (normally open)
- **Breadboard and Jumper Wires**

## 2. Hardware Connections

### LEDs

- **GPIO Pins**:
  - Heart: GPIO18
  - Ear: GPIO27
  - Hip: GPIO22
  - Shoulder: GPIO8
  - Elbow: GPIO9
  - Knee: GPIO11
  - Foot: GPIO23
  - Arm: GPIO6
- **Connection**:
  - Connect the **anode** (+) of each LED to the respective GPIO pin via a 220Ω resistor.
  - Connect the **cathode** (-) of each LED to **GND**.

### Buttons

- **GPIO Pins**:
  - Button A: GPIO13
  - Button B: GPIO19
  - Button C: GPIO26
  - Button D: GPIO21
- **Connection**:
  - Connect one leg of each button to the respective GPIO pin.

- Connect the other leg to **GND**.

## 3. Software Installation

### a. Update System

```
bash
Code kopieren
sudo apt-get update
sudo apt-get upgrade
```

### b. Install Python and Pip

```
bash
Code kopieren
sudo apt-get install python3 python3-pip
```

### c. Install Required Python Libraries

```
bash
Code kopieren
sudo pip3 install flask gpiozero
```

### d. Install SQLite (if not already installed)

```
bash
Code kopieren
sudo apt-get install sqlite3
```

## 4. Setting Up the Database

Initialize the SQLite database by running:

```
bash
Code kopieren
python3 initialize_db.py
```

You can check the database contents using:

```
bash
Code kopieren
python3 check_database.py
```

# File and Function Descriptions

## `app.py`

The main application script that runs the Flask web server and handles GPIO interactions.

### Global Variables

- `leds`: A dictionary mapping body parts to their corresponding `LED` objects.
- `buttons`: A dictionary mapping options 'A', 'B', 'C', 'D' to their respective `Button` objects.
- `selected_mode`: Tracks the current mode ('information' or 'quiz').
- `current_question_index`: Keeps track of the current quiz question index.
- `question_list`: Holds the list of quiz questions loaded from the database.
- `body_part_info`: Contains information about each body part, including descriptions, implants, and QR code URLs.

### Functions

1. `get_db_connection()`
   - Establishes a connection to the SQLite database `quiz.db`.
2. `load_questions()`
   - Loads all quiz questions from the database into `question_list`.

3. `activate_led(body_part)`
    - Activates (blinks) the LED corresponding to the given body part for 3 seconds.
4. `blink_all_leds()`
    - Blinks all LEDs simultaneously for 3 seconds to indicate an incorrect answer.
5. `submit_answer_via_button(question_id, selected_option)`
    - Processes the answer submitted via a physical button.
    - Checks if the answer is correct, updates the database, and triggers LED feedback.
6. `handle_button_press(selected_option)`
    - Handles physical button presses.
    - Calls `submit_answer_via_button()` and advances to the next question.
7. `create_button_handlers()`
    - Sets up event handlers for each physical button.

## Flask Routes

1. `@app.route('/')`: `mode_selection()`
    - Renders the mode selection page.
2. `@app.route('/set_mode/<mode>')`: `set_mode(mode)`
    - Sets the application mode and redirects to the main page.
3. `@app.route('/main')`: `main_page()`
    - Renders the main page based on the selected mode.
4. `@app.route('/get_content/<body_part>')`: `get_content(body_part)`
    - Provides information about a selected body part in JSON format.
5. `@app.route('/get_current_question')`: `get_current_question()`
    - Returns the current quiz question in JSON format.
6. `@app.route('/submit_answer', methods=['POST'])`: `submit_answer()`
    - Handles answer submissions from the web interface (if used).

7. `@app.route('/cleanup')`: `cleanup()`
    - Placeholder route for GPIO cleanup (handled automatically by `gpiozero`).

### `initialize_db.py`

Script to initialize and populate the SQLite database with quiz questions.

## Functions

- `initialize_database()`
    - Creates `questions` and `responses` tables in `quiz.db`.
    - Inserts 30 predefined quiz questions into the `questions` table.

### `check_database.py`

Script to verify the contents of the database.

## Functions

- `check_database()`
    - Prints the total number of questions and responses in the database.

## Templates

- `base.html`
    - The base template containing common HTML structure and includes.
- `mode_selection.html`
    - Allows users to select between "Information" and "Quiz" modes.
- `main.html`
    - Displays the skeleton image and content area for information or quiz questions.
    - Contains JavaScript functions to handle user interactions and page updates.
- `information_content.html`
    - Displays detailed information about a selected body part, including descriptions, implants, and QR codes.

## Static Files

- `styles.css`
  - Contains CSS styles for the application.
- **Images (`/static/images/`)**
  - `skeleton.png`: Image of the human skeleton used in "Information" mode.
- **QR Codes (`/static/qr_codes/`)**
  - Contains QR code images for each body part (e.g., `heart_qr.png`, `ear_qr.png`).

---

# Expanding the Application

## Adding More Questions

To add new questions to the quiz:

1. **Update `initialize_db.py`**:
   - Add new question dictionaries to the `questions_data` list.
   - Each question should have the following keys:
     - `id`: Unique integer ID.
     - `question`: The question text.
     - `option_a`, `option_b`, `option_c`, `option_d`: The answer options.
     - `correct_answer`: The correct option ('A', 'B', 'C', or 'D').
     - `body_part`: The associated body part (must match keys in `body_part_info`).
2. **Reinitialize the Database**:
   - Run `initialize_db.py` to update the database:

     ```
     bash
     Code kopieren
     python3 initialize_db.py
     ```

3. **Update the Application**:
   - Restart the Flask application to load the new questions.

## Adding More LEDs and Body Parts

To add new body parts with corresponding LEDs:

1. **Hardware Setup**:
   - Connect the new LED to a GPIO pin with a 220Ω resistor.
   - Connect the LED's cathode to GND.
2. **Update `app.py`**:
   - **Add the LED to the `leds` Dictionary**:

     ```
     python
     Code kopieren
     leds = {
         # Existing LEDs
         'NewBodyPart': LED(GPIO_PIN_NUMBER),
     }
     ```

   - **Update `body_part_info`**:
     - Add a new entry for the body part with its description, implants, and QR code URL.

       ```
       python
       Code kopieren
       body_part_info = {
           # Existing body parts
           'NewBodyPart': {
               'description': 'Description of the new bod
       y part.',
               'implants': ['Implant1', 'Implant2'],
               'qr_code_url': 'static/qr_codes/new_body_p
       art_qr.png'
           },
       }
       ```

   - **Include the New Body Part in Templates**:

    - Update `main.html` to add a button for the new body part in "Information" mode.
    - Position the button appropriately over the skeleton image.
3. **Add QR Code and Images**:
    - Place the QR code image for the new body part in `static/qr_codes/`.
    - Update any other images or assets as needed.
4. **Add Questions Related to the New Body Part**:
    - Follow the steps in **Adding More Questions** to include questions associated with the new body part.

---

# Additional Information

## Hardware Setup

- **Button Configuration**:
    - The buttons are configured with internal pull-up resistors (`pull_up=True`).
    - They should be connected between the GPIO pin and GND.
- **LED Configuration**:
    - LEDs are controlled by setting the GPIO pin to HIGH (on) or LOW (off).
    - Ensure that the current-limiting resistor (220Ω) is used to prevent damage.

## Running the Application

Start the Flask application with root privileges to access GPIO pins:

```
bash
Code kopieren
sudo python3 app.py
```

Access the application in a web browser at:

```
arduino
Code kopieren
```

```
http://localhost:5000
```

## Dependencies

- **Python 3**
- **Flask**
- **gpiozero**
- **sqlite3**

Install dependencies using:

```
bash
Code kopieren
sudo pip3 install flask gpiozero
sudo apt-get install sqlite3
```

## Troubleshooting

- **Buttons Not Responding**:
  - Ensure buttons are correctly wired between the GPIO pin and GND.
  - Verify that the GPIO pin numbers in `app.py` match your wiring.
- **LEDs Not Lighting Up**:
  - Check the LED connections and ensure the anode and cathode are correctly connected.
  - Confirm that the GPIO pins are correctly specified in `app.py`.
- **Database Errors**:
  - Verify that `quiz.db` exists and contains the necessary tables.
  - Run `initialize_db.py` to recreate the database if needed.
- **Permission Issues**:
  - Always run the application with `sudo` to allow access to GPIO pins.

---

## How the Application Works

1. **Mode Selection**:
    - Users select between "Information" and "Quiz" modes on the home page.
2. **Information Mode**:
    - Users click on body parts displayed on a skeleton image.
    - The application fetches and displays information, implants, and QR codes related to the selected body part.
3. **Quiz Mode**:
    - The application displays multiple-choice questions.
    - Users answer by pressing physical buttons corresponding to options A, B, C, or D.
    - Correct answers trigger the LED of the related body part.
    - Incorrect answers cause all LEDs to blink.
4. **LED Feedback**:
    - LEDs provide immediate visual feedback on the user's answer.
    - The application loops through the questions indefinitely.

---

## Conclusion

This Medical Quiz Application combines hardware and software to create an interactive educational tool. By following this guide, you can set up the application, understand its components, and expand its functionality to suit your needs.