SmartMedKit

Enhancing Medical Supply Management with IoT

IoT System Development Workshop With Gili Kamma

Team: Eli Freid, Gil Morhaim, Shalev Kedar

21.06.2024

The Challenge We Face

Critical Gaps in Medical Supply and Emergency Response

The Challenge We Face

Critical Gaps in Medical Supply and Emergency Response

- **Frequent Stock Shortages:** Manual inventory tracking is unreliable, leading to frequent shortages of critical medical supplies.

The Challenge We Face

Critical Gaps in Medical Supply and Emergency Response

- **Frequent Stock Shortages:** Manual inventory tracking is unreliable, leading to frequent shortages of critical medical supplies.
- **Injury Analysis:** In a critical moment, not having instant access to accurate injury analysis can lead to improper treatment. Quick and precise injury identification and appropriate treatment instructions are vital for effective first aid.

Current Solutions: Why They Fall Short

Existing Approaches

Current Solutions: Why They Fall Short

Existing Approaches

Despite various existing solutions aimed at managing medical supplies and providing emergency response instructions, significant gaps and inefficiencies remain.

Current Solutions: Why They Fall Short

Existing Approaches

Despite various existing solutions aimed at managing medical supplies and providing emergency response instructions, significant gaps and inefficiencies remain.

- **Manual Inventory Management:** Most households, offices, and schools rely on manual inventory management or regular first aid kits, which is prone to human error and time-consuming. This method fails to provide real-time updates, leading to stock shortages.

Current Solutions: Why They Fall Short

Existing Approaches

Despite various existing solutions aimed at managing medical supplies and providing emergency response instructions, significant gaps and inefficiencies remain.

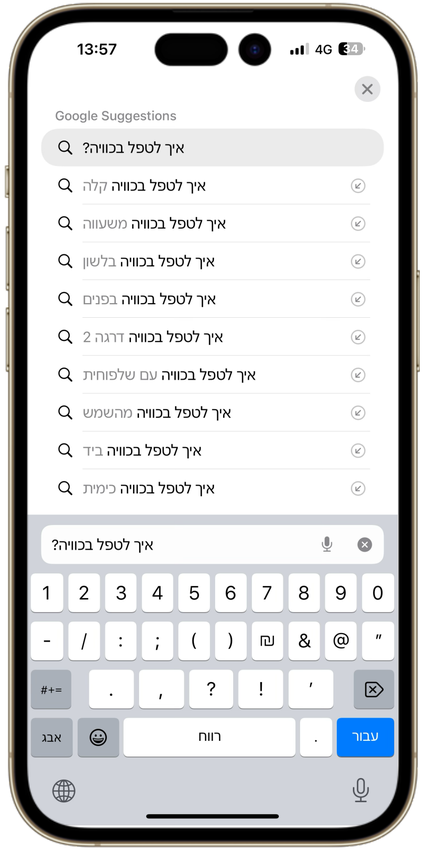
- **Generic First Aid Guides:** Many people turn to generic first aid guides or online searches for injury treatment instructions. These resources often provide broad, non-specific advice, which can lead to delays and improper treatment during emergencies.

Current Solutions: Why They Fall Short

Existing Approaches

Despite various existing solutions aimed at managing medical supplies and providing emergency response instructions, significant gaps and inefficiencies remain.

- **Generic First Aid Guides:** Many people turn to generic first aid guides or online searches for injury treatment instructions. These resources often provide broad, non-specific advice, which can lead to delays and improper treatment during emergencies.



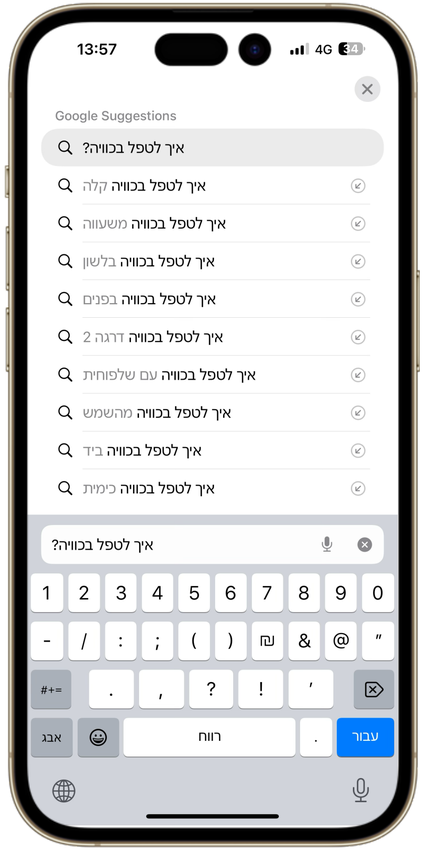
Current Solutions: Why They Fall Short

Existing Approaches

Despite various existing solutions aimed at managing medical supplies and providing emergency response instructions, significant gaps and inefficiencies remain.

- **Generic First Aid Guides:** Many people turn to generic first aid guides or online searches for injury treatment instructions. These resources often provide broad, non-specific advice, which can lead to delays and improper treatment during emergencies.

Current solutions are outdated and inefficient, unable to meet the demands of real-time medical supply management and accurate emergency response.



Imagine A World

Imagine a world where managing medical supplies in shared public spaces is effortless and efficient, and with a quick snap of your phone, you can get instructions for treating injuries.

Imagine A World

Imagine a world where managing medical supplies in shared public spaces is effortless and efficient, and with a quick snap of your phone, you can get instructions for treating injuries.

Introducing SmartMedKit

SmartMedKit: System Overview

Innovative and Integrated Medical Supply Management

SmartMedKit: System Overview

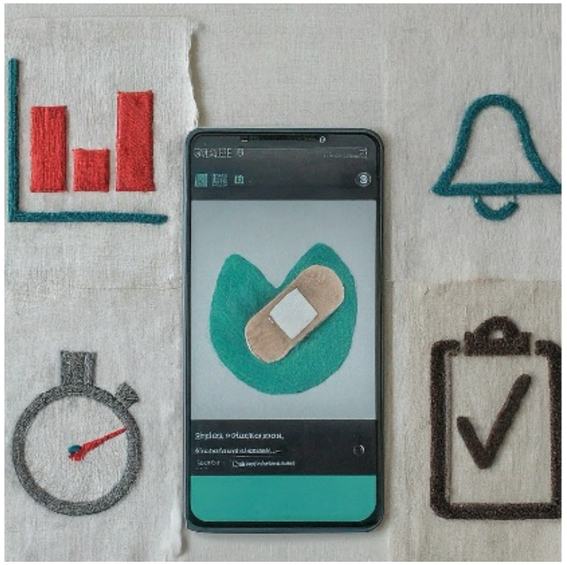
Innovative and Integrated Medical Supply Management

SmartMedKit is a comprehensive system designed to revolutionize how medical supplies are managed and how emergency responses are executed in shared public spaces.

The system integrates real-time inventory tracking with smart injury analysis, providing users with immediate, accurate information to manage supplies and treat injuries effectively.

Key Features of SmartMedKit

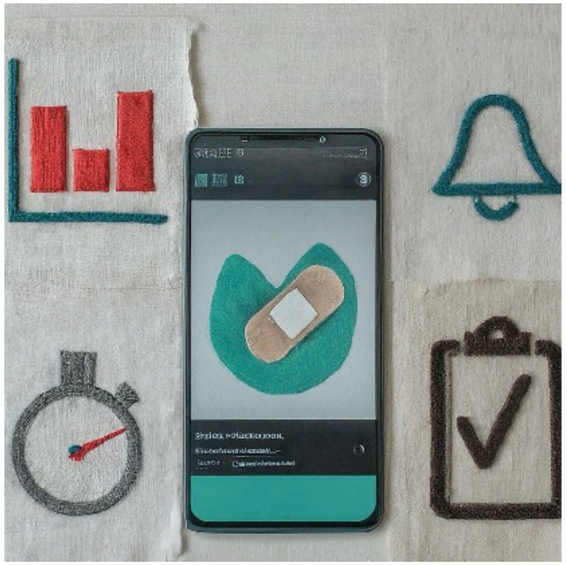
Transforming Medical Supply Management and Emergency Response



Key Features of SmartMedKit

Transforming Medical Supply Management and Emergency Response

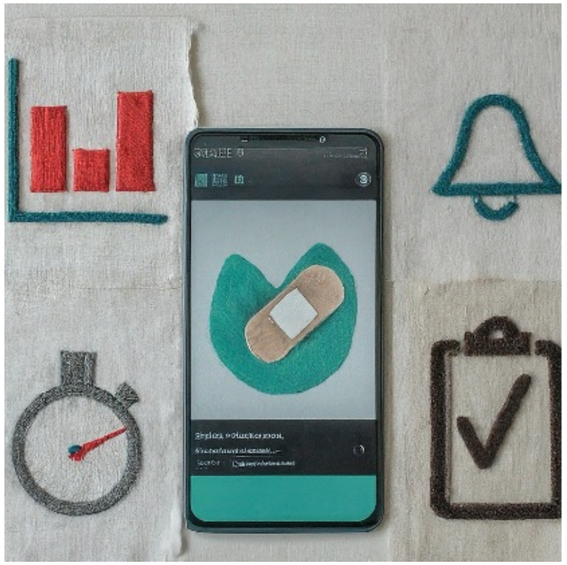
- **Injury Image Analysis:** With a quick scan of the QR code on the SmartMedKit, users can effortlessly access our intuitive web app. Upload an image of the injury, and our AI model will provide immediate treatment instructions.



Key Features of SmartMedKit

Transforming Medical Supply Management and Emergency Response

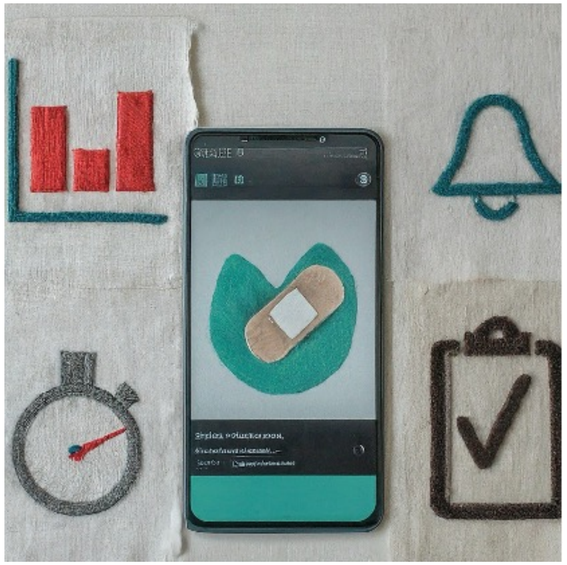
- **Injury Image Analysis:** With a quick scan of the QR code on the SmartMedKit, users can effortlessly access our intuitive web app. Upload an image of the injury, and our AI model will provide immediate treatment instructions.
- **Automated Inventory Management:** Say goodbye to manual stock-takes. SmartMedKit automatically tracks inventory levels in real-time and sends smart alerts when stock levels are low.



Key Features of SmartMedKit

Transforming Medical Supply Management and Emergency Response

- **Injury Image Analysis:** With a quick scan of the QR code on the SmartMedKit, users can effortlessly access our intuitive web app. Upload an image of the injury, and our AI model will provide immediate treatment instructions.
- **Automated Inventory Management:** Say goodbye to manual stock-takes. SmartMedKit automatically tracks inventory levels in real-time and sends smart alerts when stock levels are low.
- **Admin Dashboard:** From the administrator's perspective, SmartMedKit provides a comprehensive dashboard with valuable data analytics and a user-friendly interface. Gain insights into inventory usage trends, optimize supplies, and manage kits efficiently. Track supply status, view usage statistics, and analyze injury reports, all through an intuitive central platform.



Demo Time!

Experience the Future of Intelligent Health Assistance

Demo Time!

Experience the Future of Intelligent Health Assistance

SmartMedKit: System Architecture

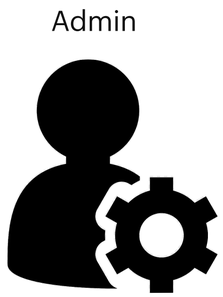
Seamlessly Integrated Components for Optimal Performance

SmartMedKit is built on a architecture that integrates various components to provide a seamless and efficient user experience. Here’s an overview of the key components and their interactions:

SmartMedKit: System Architecture

Seamlessly Integrated Components for Optimal Performance

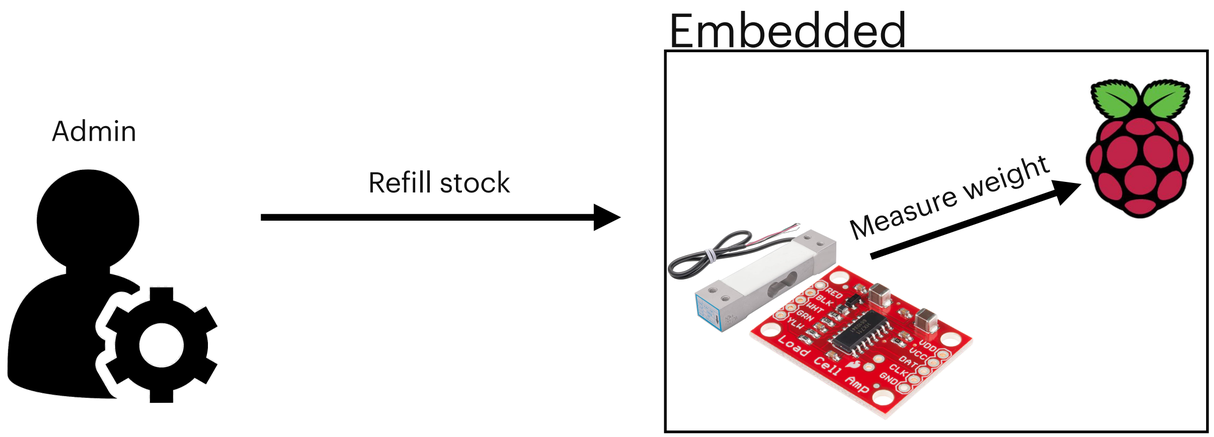
SmartMedKit is built on a architecture that integrates various components to provide a seamless and efficient user experience. Here’s an overview of the key components and their interactions:



SmartMedKit: System Architecture

Seamlessly Integrated Components for Optimal Performance

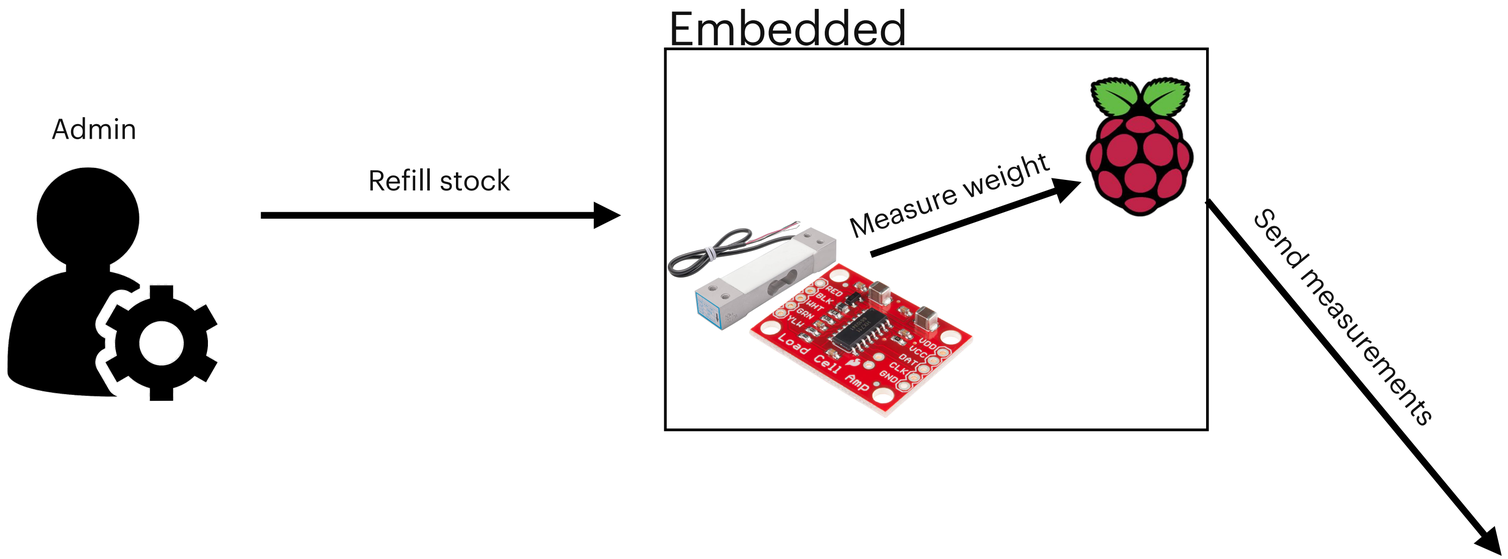
SmartMedKit is built on a architecture that integrates various components to provide a seamless and efficient user experience. Here’s an overview of the key components and their interactions:



SmartMedKit: System Architecture

Seamlessly Integrated Components for Optimal Performance

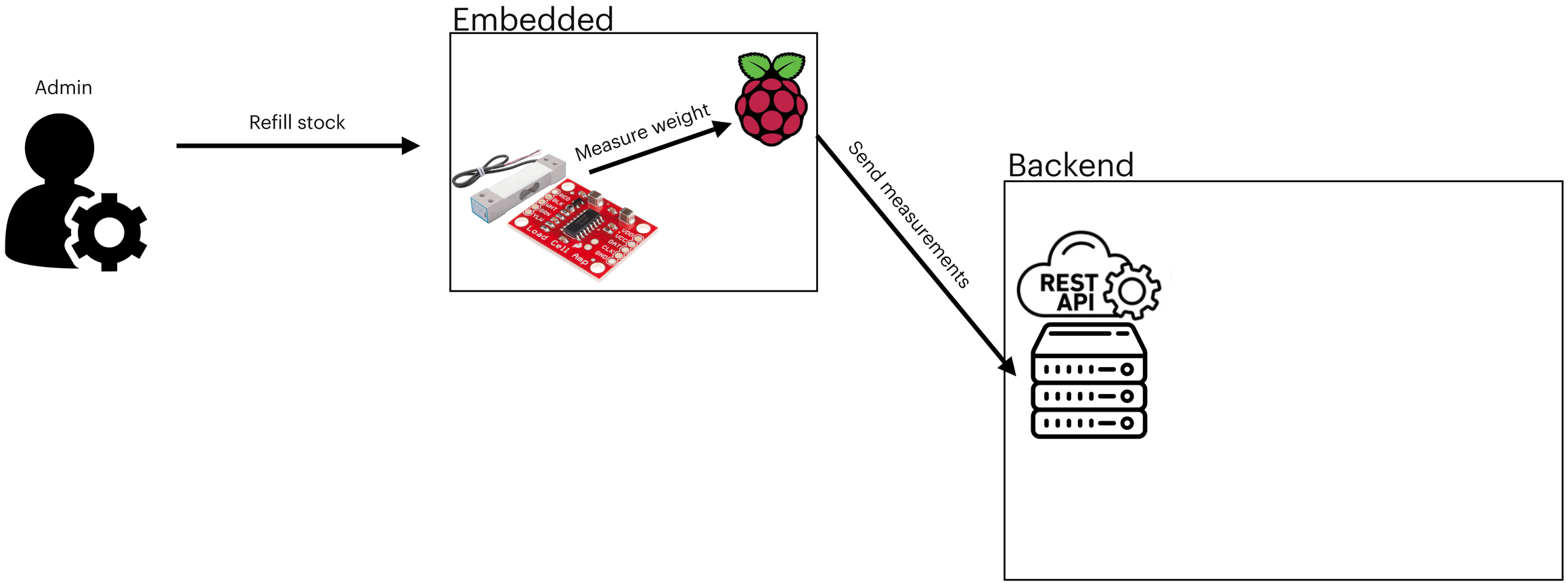
SmartMedKit is built on a architecture that integrates various components to provide a seamless and efficient user experience. Here's an overview of the key components and their interactions:



SmartMedKit: System Architecture

Seamlessly Integrated Components for Optimal Performance

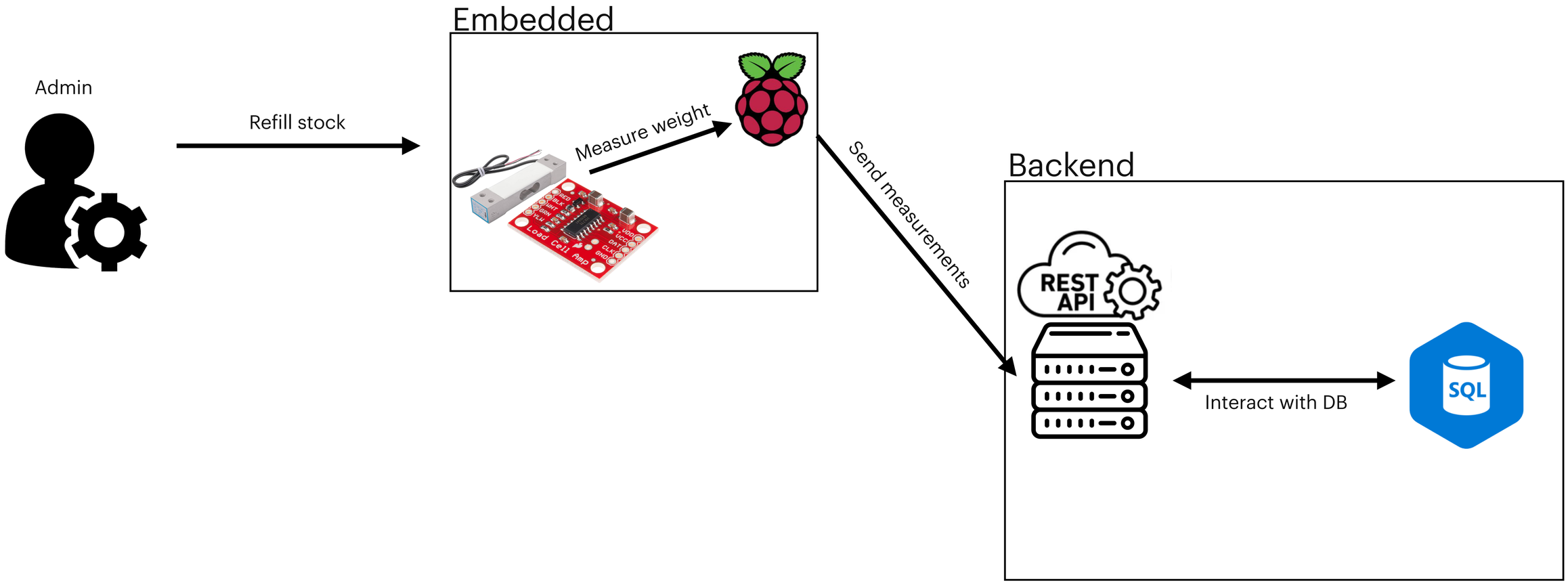
SmartMedKit is built on a architecture that integrates various components to provide a seamless and efficient user experience. Here's an overview of the key components and their interactions:



SmartMedKit: System Architecture

Seamlessly Integrated Components for Optimal Performance

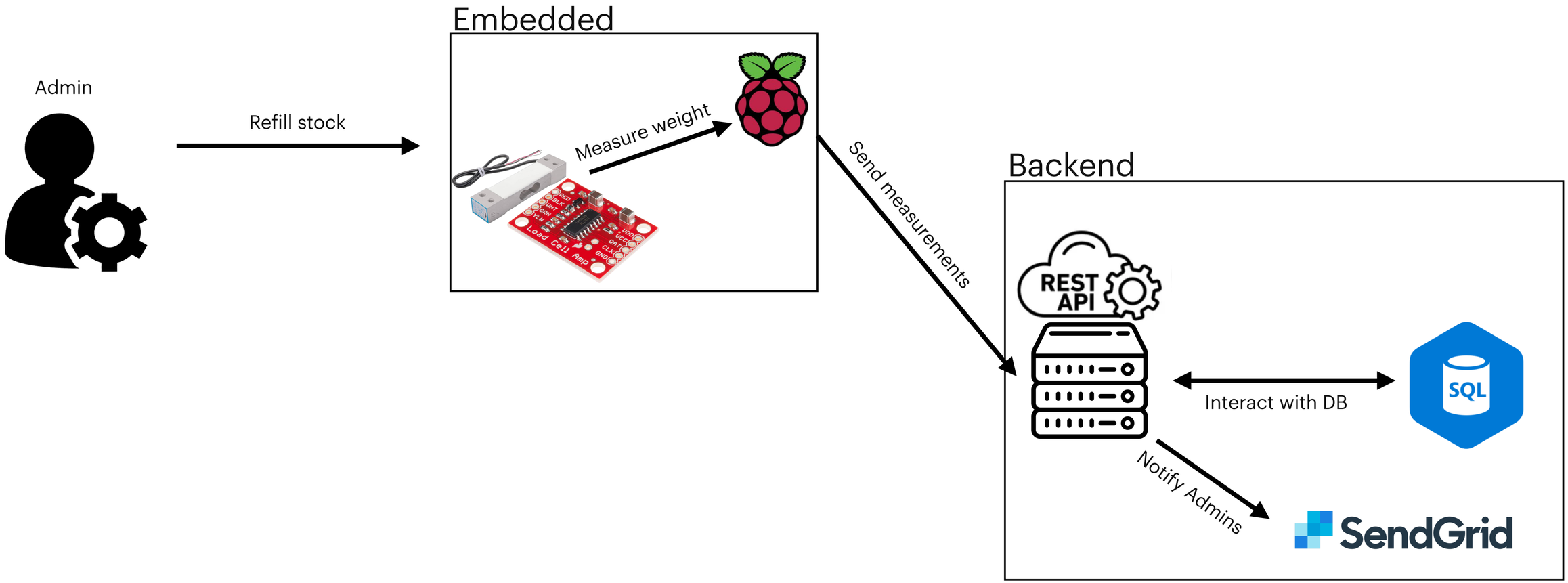
SmartMedKit is built on a architecture that integrates various components to provide a seamless and efficient user experience. Here's an overview of the key components and their interactions:



SmartMedKit: System Architecture

Seamlessly Integrated Components for Optimal Performance

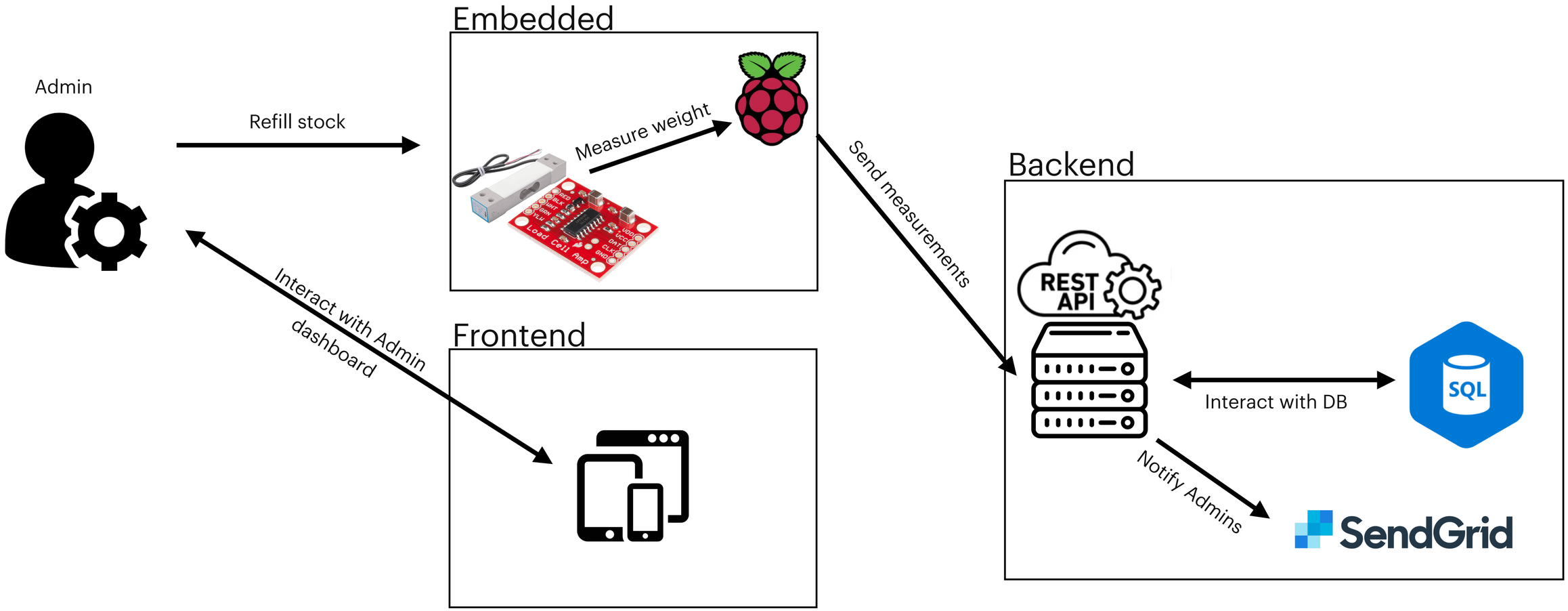
SmartMedKit is built on a architecture that integrates various components to provide a seamless and efficient user experience. Here's an overview of the key components and their interactions:



SmartMedKit: System Architecture

Seamlessly Integrated Components for Optimal Performance

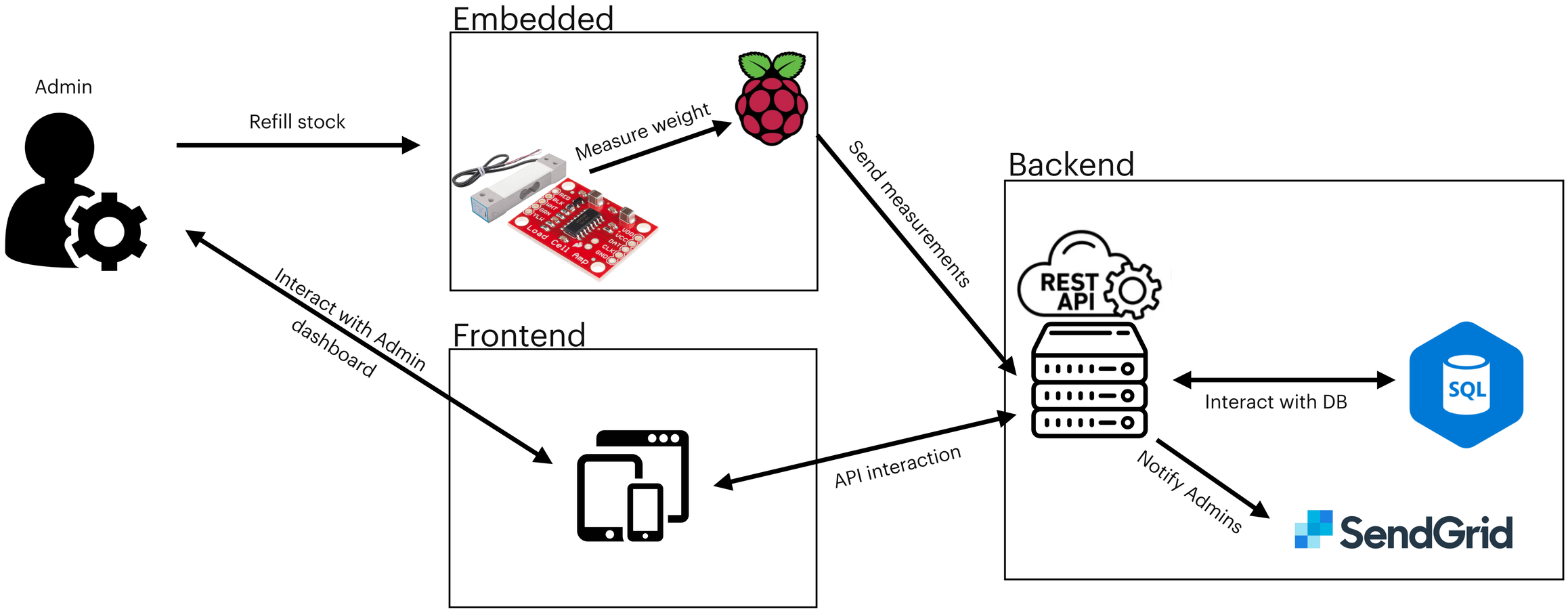
SmartMedKit is built on a architecture that integrates various components to provide a seamless and efficient user experience. Here's an overview of the key components and their interactions:



SmartMedKit: System Architecture

Seamlessly Integrated Components for Optimal Performance

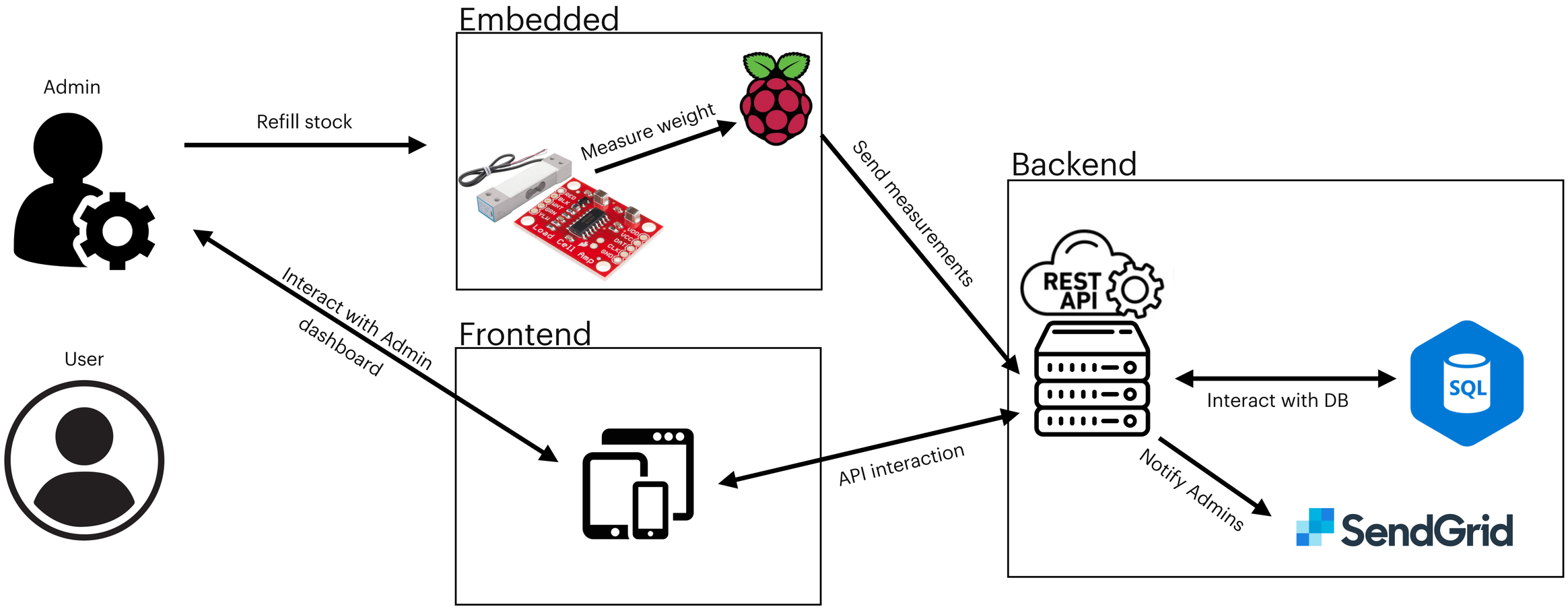
SmartMedKit is built on a architecture that integrates various components to provide a seamless and efficient user experience. Here's an overview of the key components and their interactions:



SmartMedKit: System Architecture

Seamlessly Integrated Components for Optimal Performance

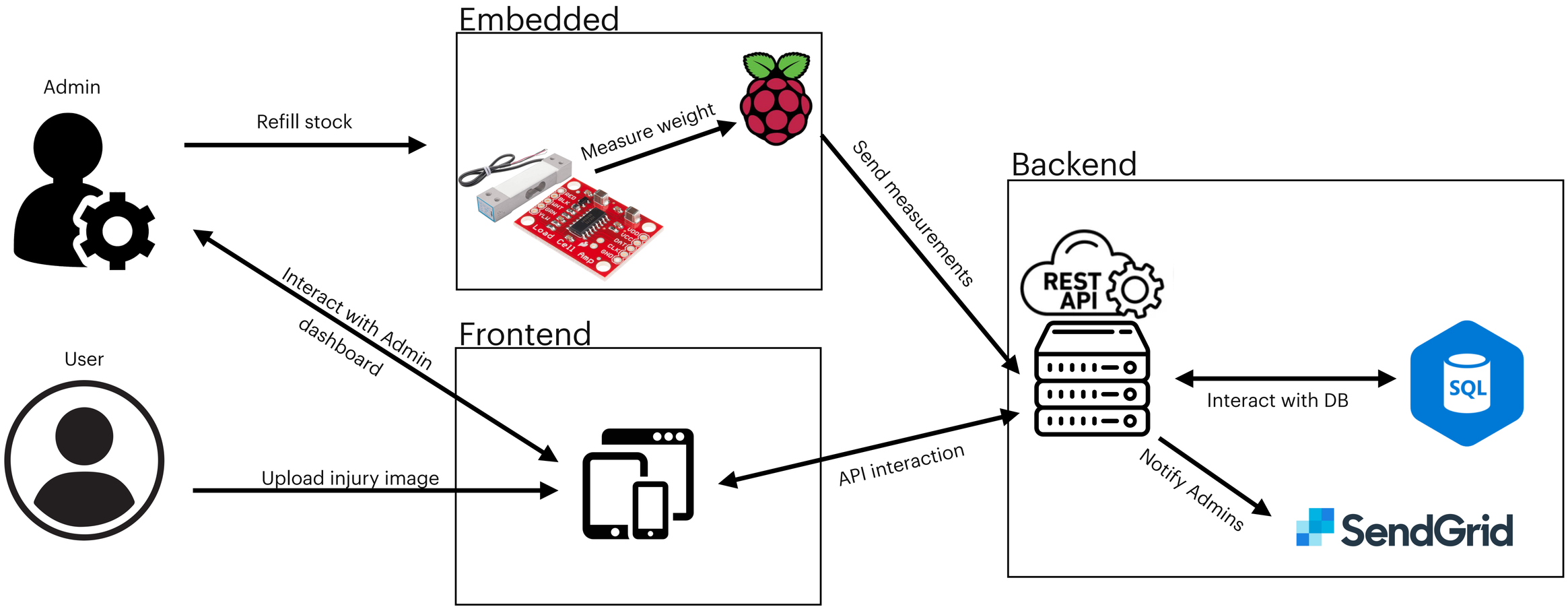
SmartMedKit is built on a architecture that integrates various components to provide a seamless and efficient user experience. Here's an overview of the key components and their interactions:



SmartMedKit: System Architecture

Seamlessly Integrated Components for Optimal Performance

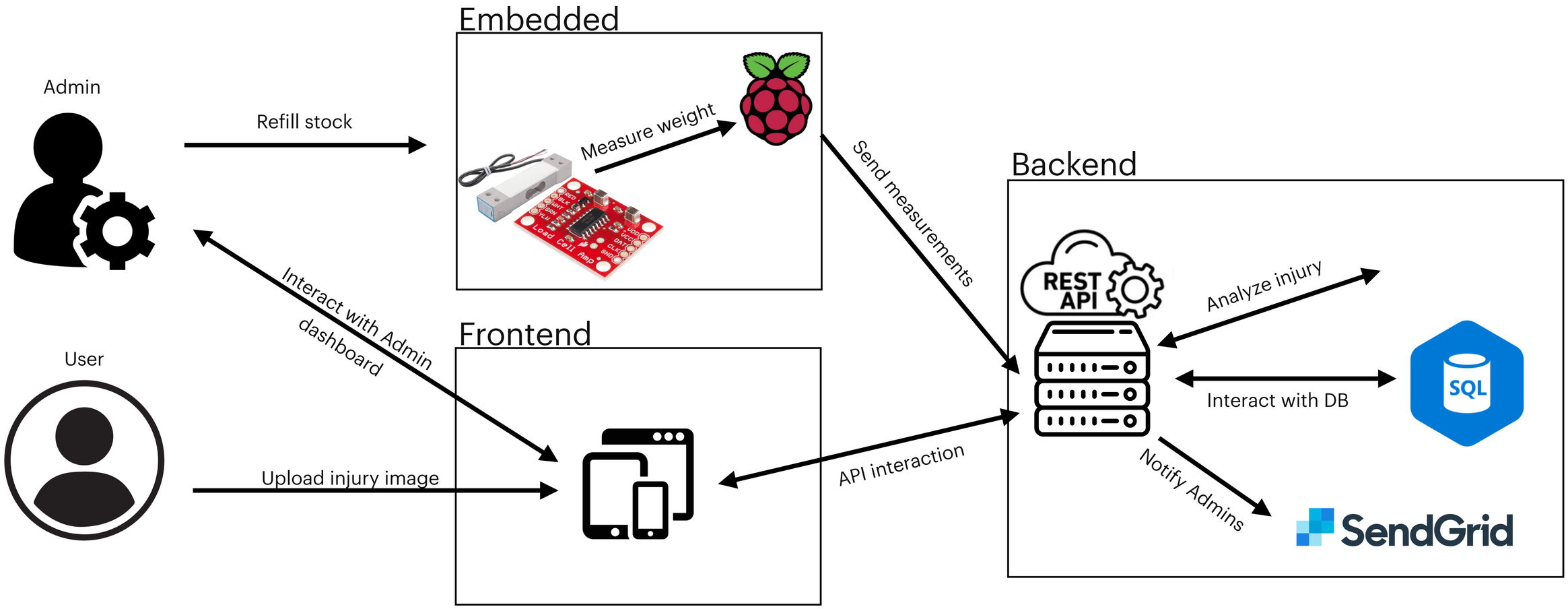
SmartMedKit is built on a architecture that integrates various components to provide a seamless and efficient user experience. Here's an overview of the key components and their interactions:



SmartMedKit: System Architecture

Seamlessly Integrated Components for Optimal Performance

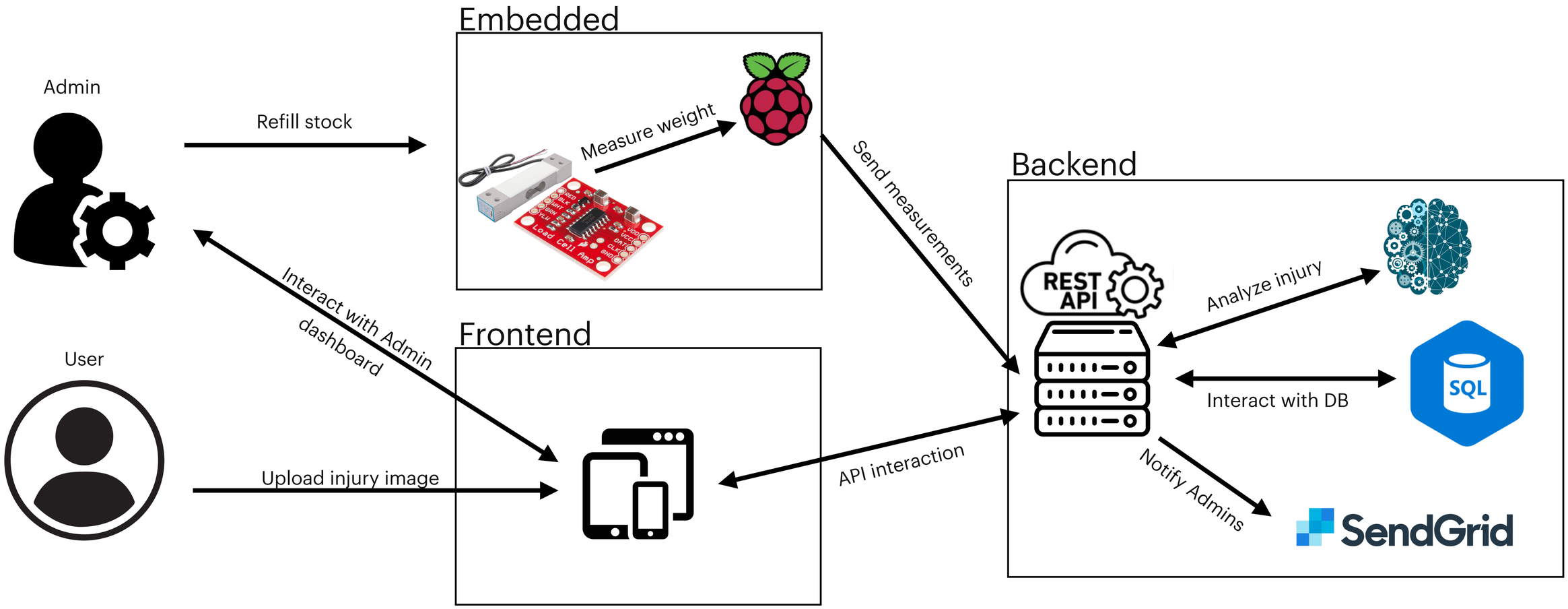
SmartMedKit is built on a architecture that integrates various components to provide a seamless and efficient user experience. Here's an overview of the key components and their interactions:



SmartMedKit: System Architecture

Seamlessly Integrated Components for Optimal Performance

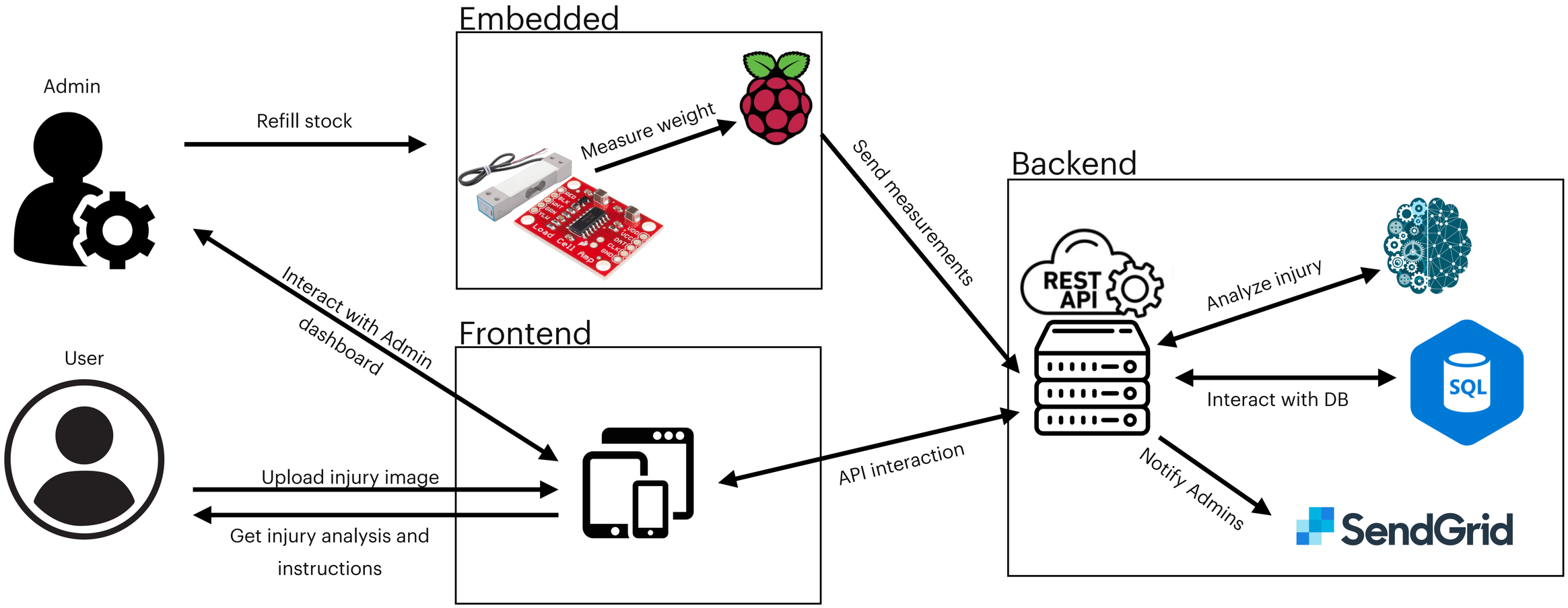
SmartMedKit is built on a architecture that integrates various components to provide a seamless and efficient user experience. Here's an overview of the key components and their interactions:



SmartMedKit: System Architecture

Seamlessly Integrated Components for Optimal Performance

SmartMedKit is built on a architecture that integrates various components to provide a seamless and efficient user experience. Here's an overview of the key components and their interactions:



Backend Tech Stack

Building the Core of SmartMedKit

Python & Flask: For developing the backend services and APIs.

SQLite: As the relational database for storing data.

Celery & Redis: For handling asynchronous tasks and notifications.

Docker: For containerization and easy deployment.

TensorFlow: For training, developing and deploying the AI model.

Google Cloud Platform: For hosting the Dockerized backend.

Our Machine Learning Model

Leveraging VGG16 for Injury Identification

Our Machine Learning Model

Leveraging VGG16 for Injury Identification

- **Model Overview:**

Our Machine Learning Model

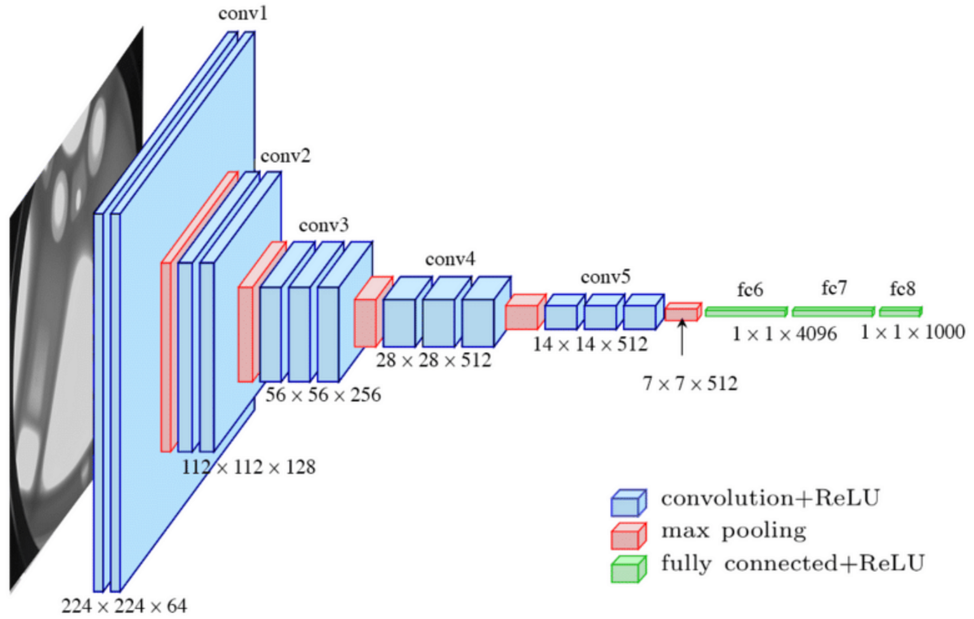
Leveraging VGG16 for Injury Identification

- **Model Overview:**
  - **VGG16 Model:** A type of neural network designed to process and classify images effectively. It is known for its performance in image recognition tasks.

Our Machine Learning Model

Leveraging VGG16 for Injury Identification

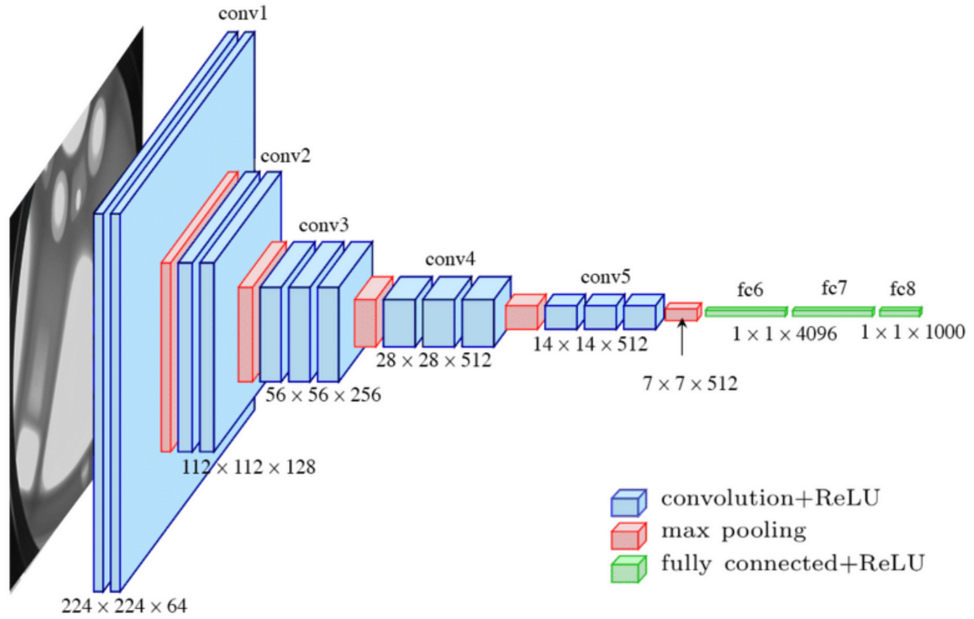
- **Model Overview:**
  - **VGG16 Model:** A type of neural network designed to process and classify images effectively. It is known for its performance in image recognition tasks.



Our Machine Learning Model

Leveraging VGG16 for Injury Identification

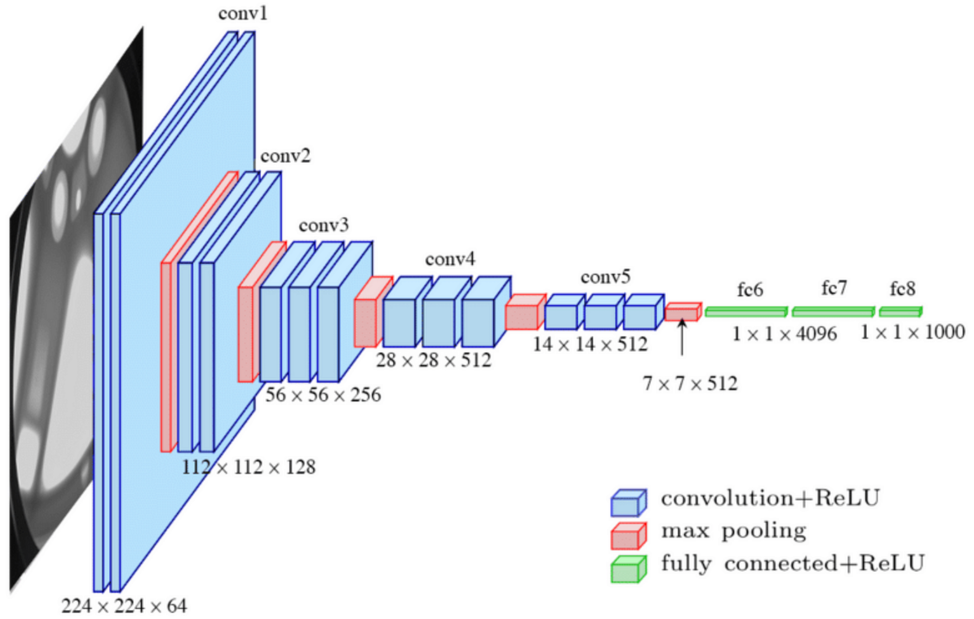
- **Model Overview:**
  - **VGG16 Model:** A type of neural network designed to process and classify images effectively. It is known for its performance in image recognition tasks.
  - **Application:** Used for distinguishing between burns and cuts in injury images.



Our Machine Learning Model

Leveraging VGG16 for Injury Identification

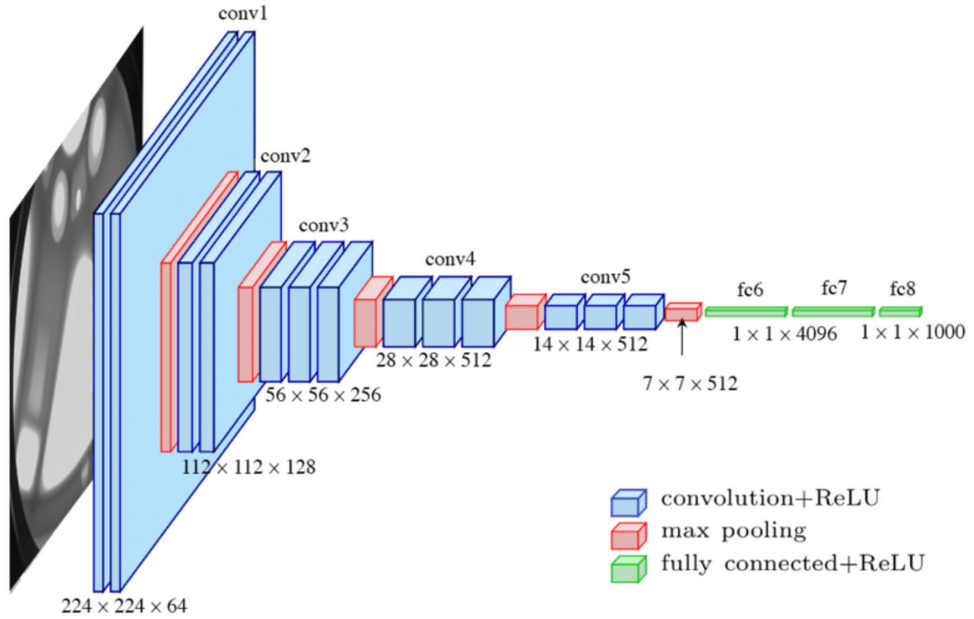
- **Model Overview:**
  - **VGG16 Model:** A type of neural network designed to process and classify images effectively. It is known for its performance in image recognition tasks.
  - **Application:** Used for distinguishing between burns and cuts in injury images.
- **Key Parameters:**



Our Machine Learning Model

Leveraging VGG16 for Injury Identification

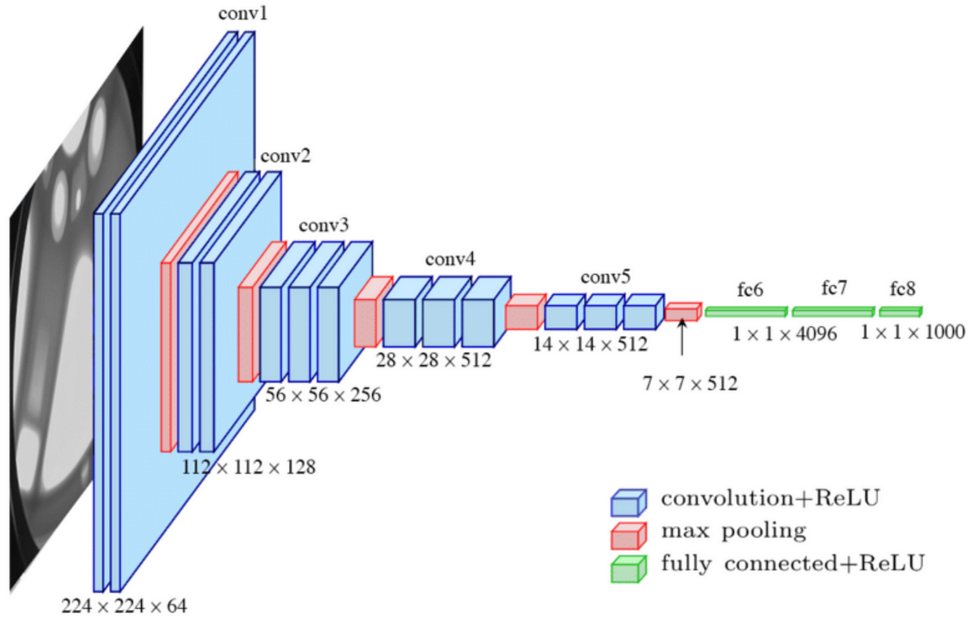
- **Model Overview:**
  - **VGG16 Model:** A type of neural network designed to process and classify images effectively. It is known for its performance in image recognition tasks.
  - **Application:** Used for distinguishing between burns and cuts in injury images.
- **Key Parameters:**
  - **Optimizer:** Adam - Adjusts the model's parameters during training to minimize the error.



Our Machine Learning Model

Leveraging VGG16 for Injury Identification

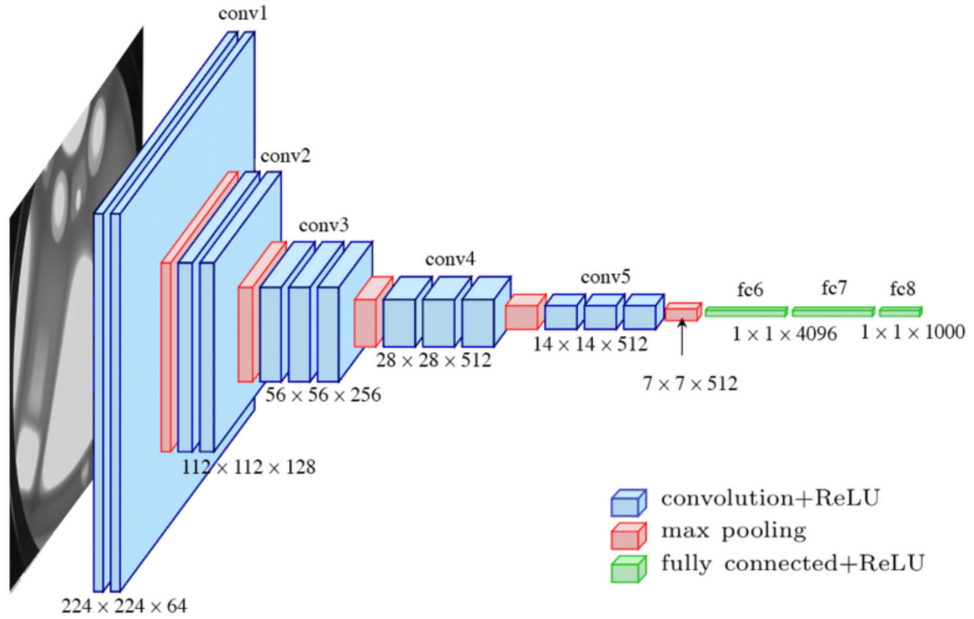
- **Model Overview:**
  - **VGG16 Model:** A type of neural network designed to process and classify images effectively. It is known for its performance in image recognition tasks.
  - **Application:** Used for distinguishing between burns and cuts in injury images.
- **Key Parameters:**
  - **Optimizer:** Adam - Adjusts the model's parameters during training to minimize the error.
  - **Learning Rate:** 0.00001 - Controls how much to change the model in response to the estimated error each time the model weights are updated.



Our Machine Learning Model

Leveraging VGG16 for Injury Identification

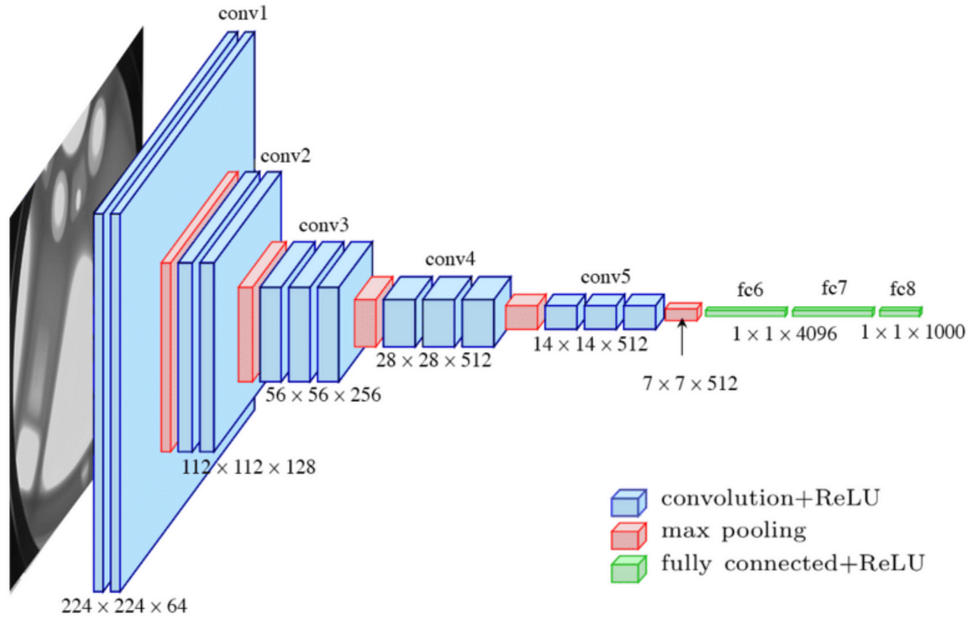
- **Model Overview:**
  - **VGG16 Model:** A type of neural network designed to process and classify images effectively. It is known for its performance in image recognition tasks.
  - **Application:** Used for distinguishing between burns and cuts in injury images.
- **Key Parameters:**
  - **Optimizer:** Adam - Adjusts the model's parameters during training to minimize the error.
  - **Learning Rate:** 0.00001 - Controls how much to change the model in response to the estimated error each time the model weights are updated.
  - **Loss Function:** Binary Crossentropy - Measures how well the model's predictions match the actual results.



Our Machine Learning Model

Leveraging VGG16 for Injury Identification

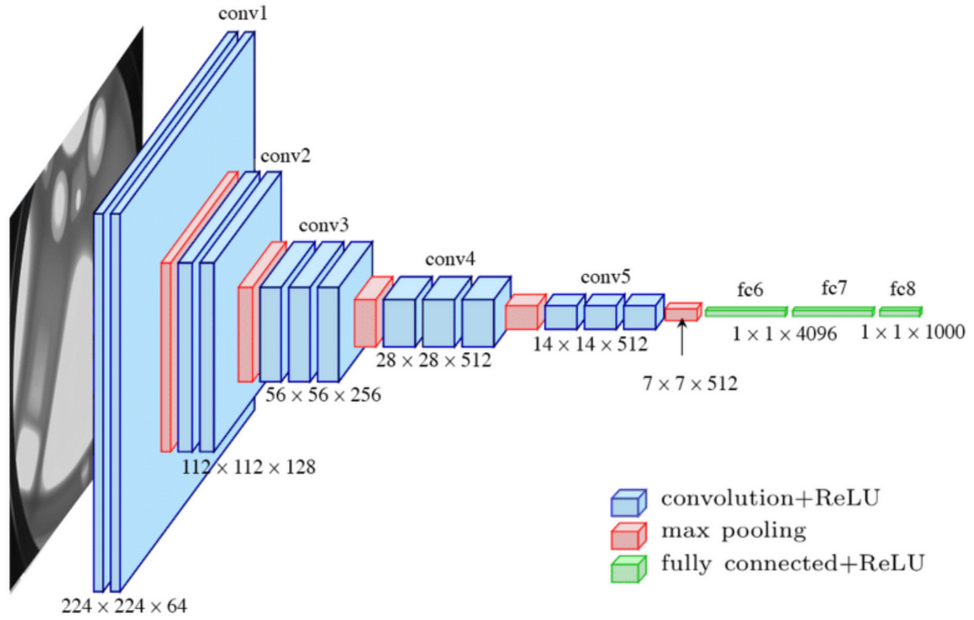
- **Model Overview:**
  - **VGG16 Model:** A type of neural network designed to process and classify images effectively. It is known for its performance in image recognition tasks.
  - **Application:** Used for distinguishing between burns and cuts in injury images.
- **Key Parameters:**
  - **Optimizer:** Adam - Adjusts the model's parameters during training to minimize the error.
  - **Learning Rate:** 0.00001 - Controls how much to change the model in response to the estimated error each time the model weights are updated.
  - **Loss Function:** Binary Crossentropy - Measures how well the model's predictions match the actual results.
  - **Epochs:** Up to 20 - Number of times the learning algorithm will work through the entire training dataset.



Our Machine Learning Model

Leveraging VGG16 for Injury Identification

- **Model Overview:**
  - **VGG16 Model:** A type of neural network designed to process and classify images effectively. It is known for its performance in image recognition tasks.
  - **Application:** Used for distinguishing between burns and cuts in injury images.
- **Key Parameters:**
  - **Optimizer:** Adam - Adjusts the model's parameters during training to minimize the error.
  - **Learning Rate:** 0.00001 - Controls how much to change the model in response to the estimated error each time the model weights are updated.
  - **Loss Function:** Binary Crossentropy - Measures how well the model's predictions match the actual results.
  - **Epochs:** Up to 20 - Number of times the learning algorithm will work through the entire training dataset.
  - **Data Augmentation:** Techniques like rotation, zoom, shear, and flips to increase dataset diversity and robustness.



Our Machine Learning Model

Leveraging VGG16 for Injury Identification

- **Model Overview:**
  - **VGG16 Model:** A type of neural network designed to process and classify images effectively. It is known for its performance in image recognition tasks.
  - **Application:** Used for distinguishing between burns and cuts in injury images.
- **Key Parameters:**
  - **Optimizer:** Adam - Adjusts the model's parameters during training to minimize the error.
  - **Learning Rate:** 0.00001 - Controls how much to change the model in response to the estimated error each time the model weights are updated.
  - **Loss Function:** Binary Crossentropy - Measures how well the model's predictions match the actual results.
  - **Epochs:** Up to 20 - Number of times the learning algorithm will work through the entire training dataset.
  - **Data Augmentation:** Techniques like rotation, zoom, shear, and flips to increase dataset diversity and robustness.
- **Advantages:**

Our Machine Learning Model

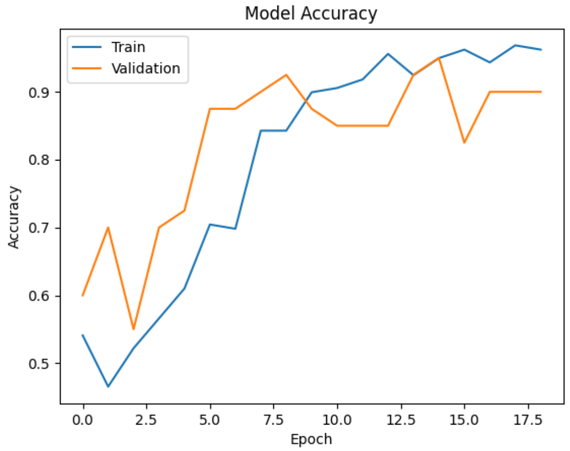
Leveraging VGG16 for Injury Identification

- **Model Overview:**
  - **VGG16 Model:** A type of neural network designed to process and classify images effectively. It is known for its performance in image recognition tasks.
  - **Application:** Used for distinguishing between burns and cuts in injury images.
- **Key Parameters:**
  - **Optimizer:** Adam - Adjusts the model's parameters during training to minimize the error.
  - **Learning Rate:** 0.00001 - Controls how much to change the model in response to the estimated error each time the model weights are updated.
  - **Loss Function:** Binary Crossentropy - Measures how well the model's predictions match the actual results.
  - **Epochs:** Up to 20 - Number of times the learning algorithm will work through the entire training dataset.
  - **Data Augmentation:** Techniques like rotation, zoom, shear, and flips to increase dataset diversity and robustness.
- **Advantages:**
  - **High Performance:** The VGG16 model provides accurate and reliable image classification.

Our Machine Learning Model

Leveraging VGG16 for Injury Identification

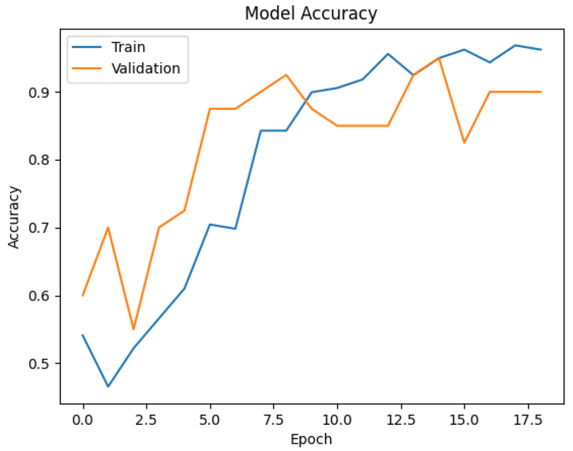
- **Model Overview:**
  - **VGG16 Model:** A type of neural network designed to process and classify images effectively. It is known for its performance in image recognition tasks.
  - **Application:** Used for distinguishing between burns and cuts in injury images.
- **Key Parameters:**
  - **Optimizer:** Adam - Adjusts the model's parameters during training to minimize the error.
  - **Learning Rate:** 0.00001 - Controls how much to change the model in response to the estimated error each time the model weights are updated.
  - **Loss Function:** Binary Crossentropy - Measures how well the model's predictions match the actual results.
  - **Epochs:** Up to 20 - Number of times the learning algorithm will work through the entire training dataset.
  - **Data Augmentation:** Techniques like rotation, zoom, shear, and flips to increase dataset diversity and robustness.
- **Advantages:**
  - **High Performance:** The VGG16 model provides accurate and reliable image classification.

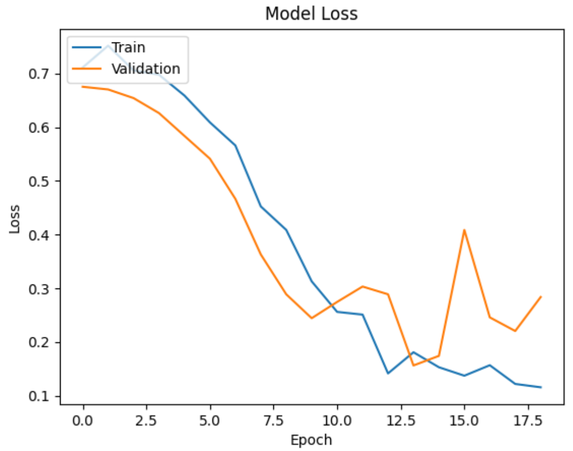


Our Machine Learning Model

Leveraging VGG16 for Injury Identification

- **Model Overview:**
  - **VGG16 Model:** A type of neural network designed to process and classify images effectively. It is known for its performance in image recognition tasks.
  - **Application:** Used for distinguishing between burns and cuts in injury images.
- **Key Parameters:**
  - **Optimizer:** Adam - Adjusts the model's parameters during training to minimize the error.
  - **Learning Rate:** 0.00001 - Controls how much to change the model in response to the estimated error each time the model weights are updated.
  - **Loss Function:** Binary Crossentropy - Measures how well the model's predictions match the actual results.
  - **Epochs:** Up to 20 - Number of times the learning algorithm will work through the entire training dataset.
  - **Data Augmentation:** Techniques like rotation, zoom, shear, and flips to increase dataset diversity and robustness.
- **Advantages:**
  - **High Performance:** The VGG16 model provides accurate and reliable image classification.

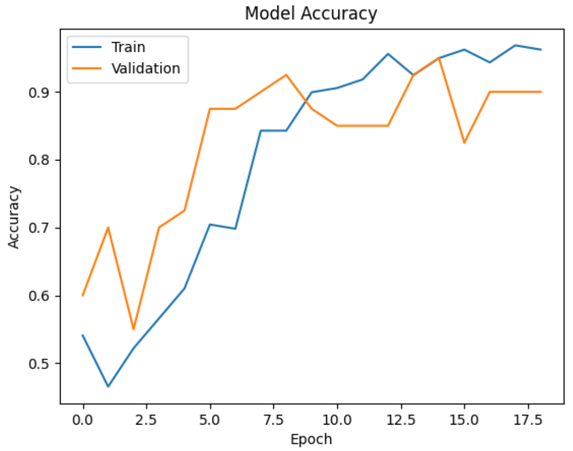


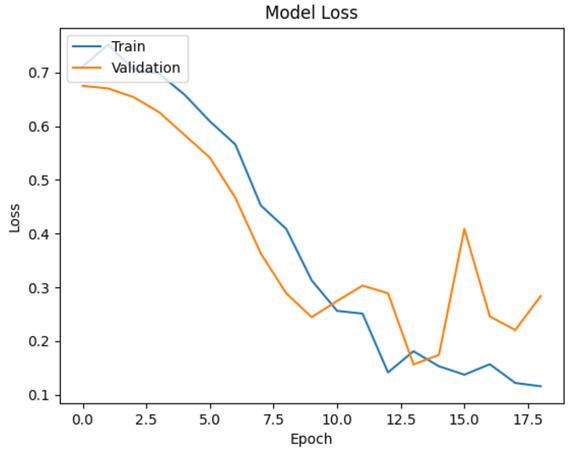


Our Machine Learning Model

Leveraging VGG16 for Injury Identification

- **Model Overview:**
  - **VGG16 Model:** A type of neural network designed to process and classify images effectively. It is known for its performance in image recognition tasks.
  - **Application:** Used for distinguishing between burns and cuts in injury images.
- **Key Parameters:**
  - **Optimizer:** Adam - Adjusts the model's parameters during training to minimize the error.
  - **Learning Rate:** 0.00001 - Controls how much to change the model in response to the estimated error each time the model weights are updated.
  - **Loss Function:** Binary Crossentropy - Measures how well the model's predictions match the actual results.
  - **Epochs:** Up to 20 - Number of times the learning algorithm will work through the entire training dataset.
  - **Data Augmentation:** Techniques like rotation, zoom, shear, and flips to increase dataset diversity and robustness.
- **Advantages:**
  - **High Performance:** The VGG16 model provides accurate and reliable image classification.
  - **Transfer Learning:** Utilizes pre-trained weights, speeding up the training process and improving accuracy with less data.





Backend API and Database

injury analysis
Operations on injury analysis

- POST `/injuries/analyze`
- GET `/injuries/`

kits
Operations on kits

- GET `/kit`
- POST `/kit`
- GET `/kit/{kit_id}`
- PUT `/kit/{kit_id}`
- DELETE `/kit/{kit_id}`

admins
Operations on admins

- GET `/admin`
- POST `/admin`
- GET `/admin/{admin_id}`
- PUT `/admin/{admin_id}`
- DELETE `/admin/{admin_id}`

kit_admin
Operations on associations between kits and admins

- GET `/kit_admin`
- POST `/kit_admin`
- DELETE `/kit_admin`

kit_compartments
Operations on kit compartments

- GET `/kit_compartments` Get all kit compartments
- POST `/kit_compartments` Create a new kit compartment
- GET `/kit_compartments/{kit_id}/{compartment_id}`
- PATCH `/kit_compartments/{kit_id}/{compartment_id}`
- DELETE `/kit_compartments/{kit_id}/{compartment_id}`

measurements
Operations on measurements

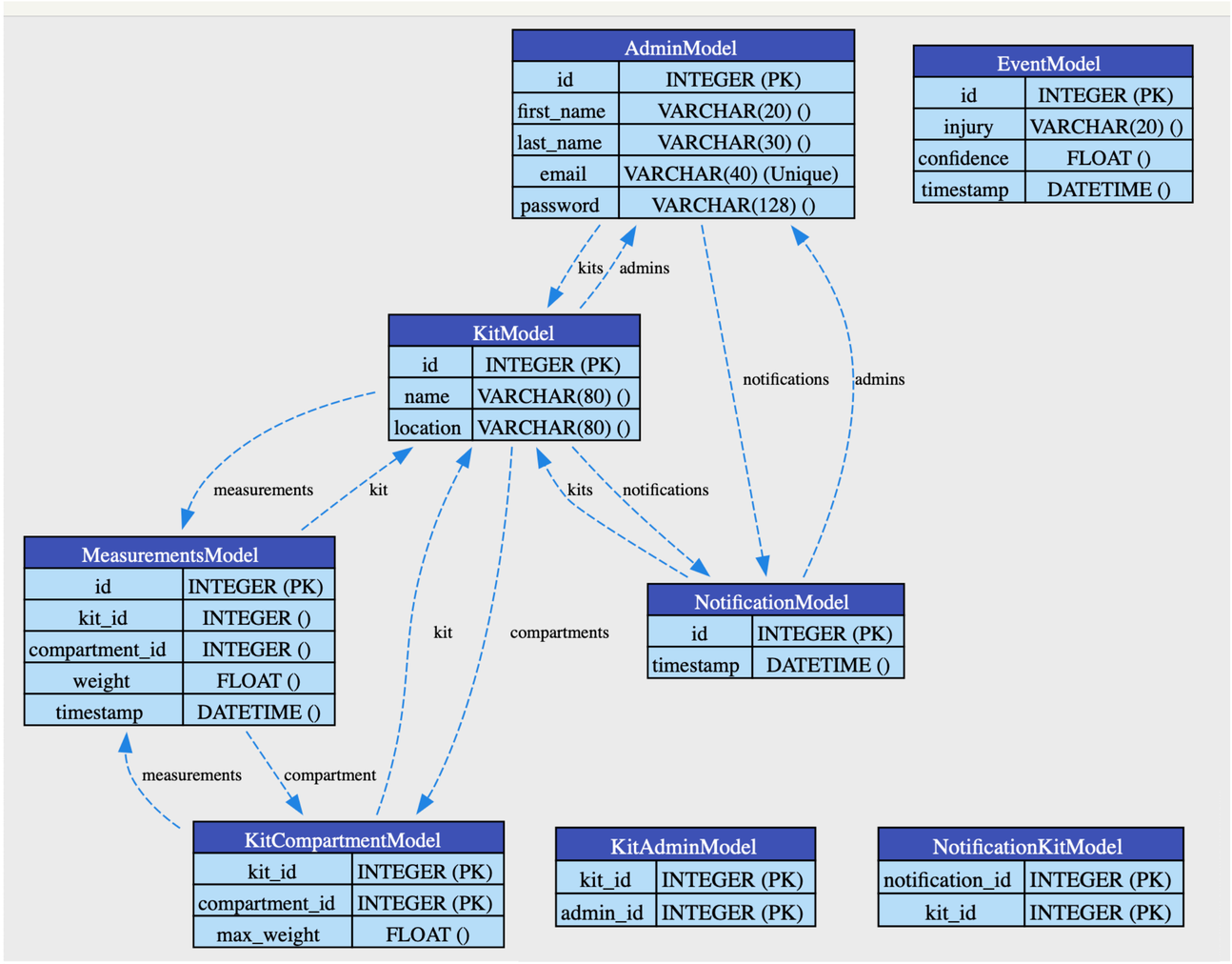
- GET `/measurements`
- POST `/measurements`

notifications
Operations on notifications

- GET `/notification`

Authorize
Authorization operations for admins

- POST `/admin/authorize`



Frontend Tech Stack

Creating an Intuitive User Experience

JavaScript: For dynamic and interactive web features.

Chart.js

react-chartjs-2: For analytics, statistics, and data visualization.

React: For building the user interface of the web application.

Bootstrap: For responsive design and rapid UI development.

React Router DOM: For client-side routing and navigation.

Axios: For making HTTP requests to the backend API.

Frontend Interface and User Experience

Navigating Through the SmartMedKit Web App

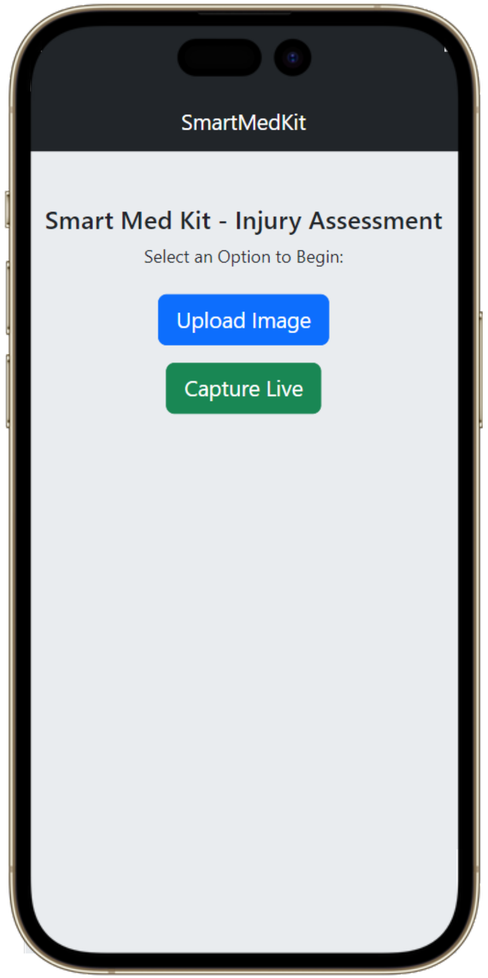
Frontend Interface and User Experience

Navigating Through the SmartMedKit Web App

Frontend Interface and User Experience

Navigating Through the SmartMedKit Web App

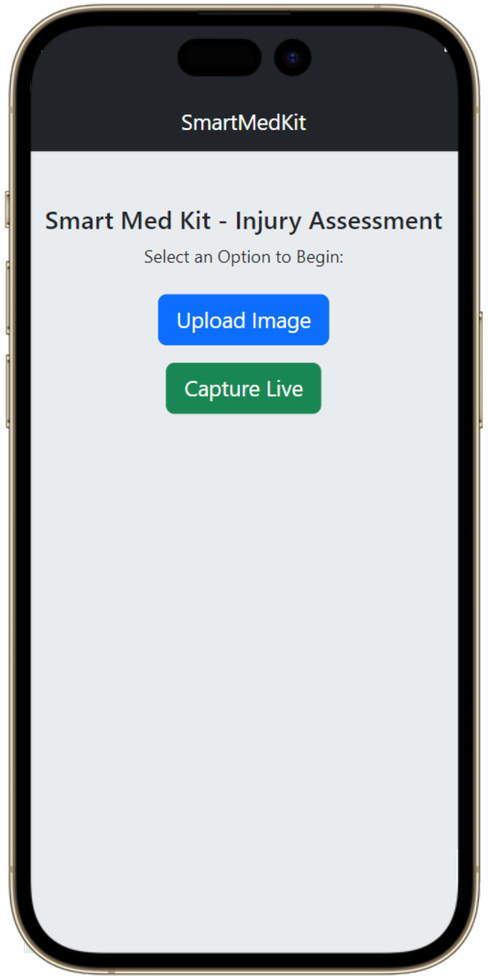
User

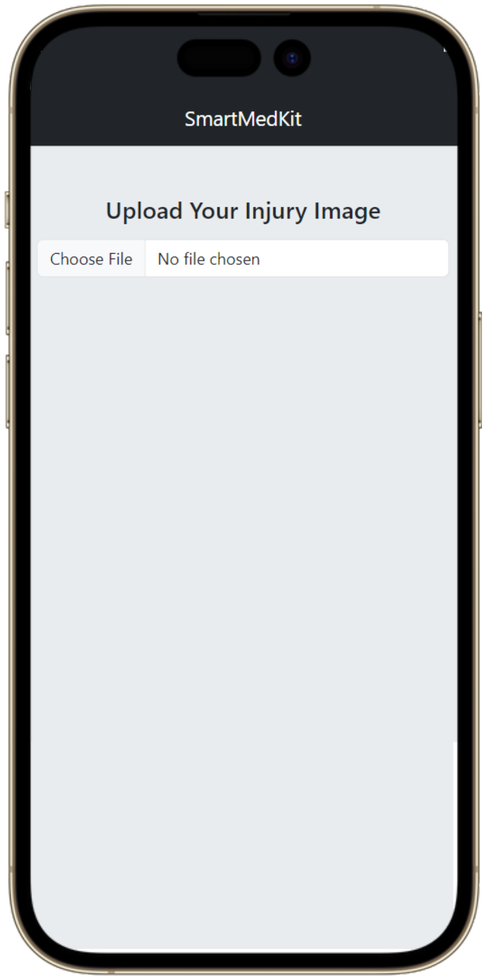


Frontend Interface and User Experience

Navigating Through the SmartMedKit Web App

User

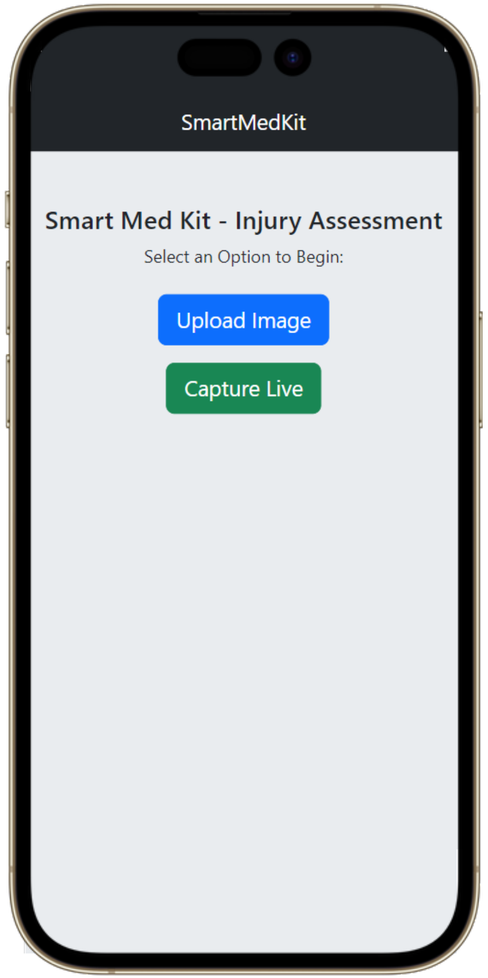


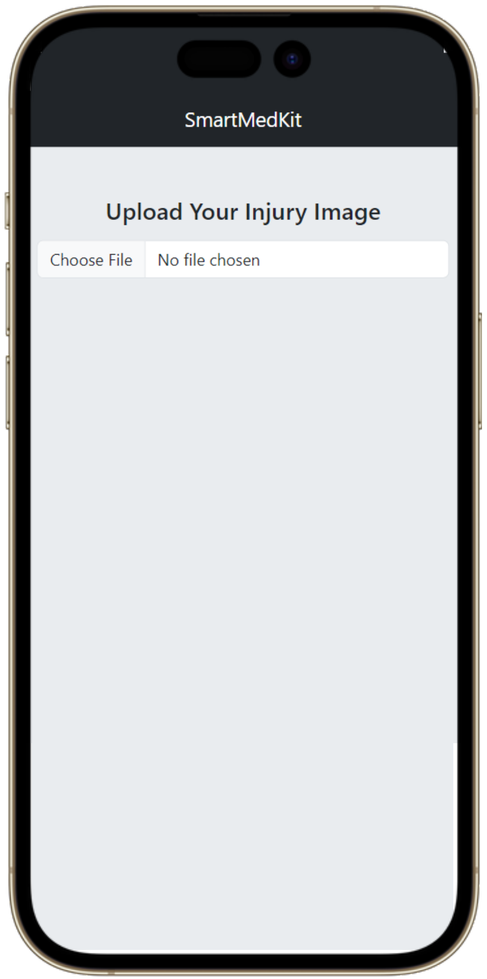


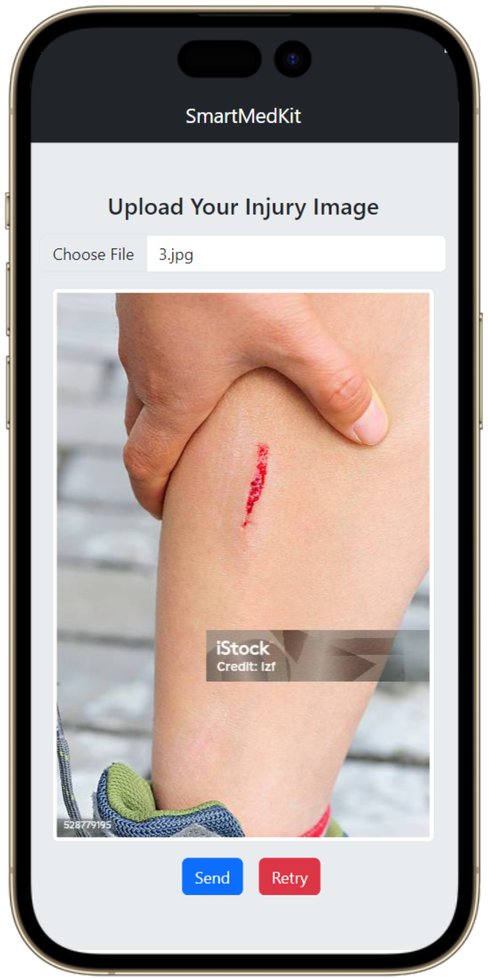
Frontend Interface and User Experience

Navigating Through the SmartMedKit Web App

User



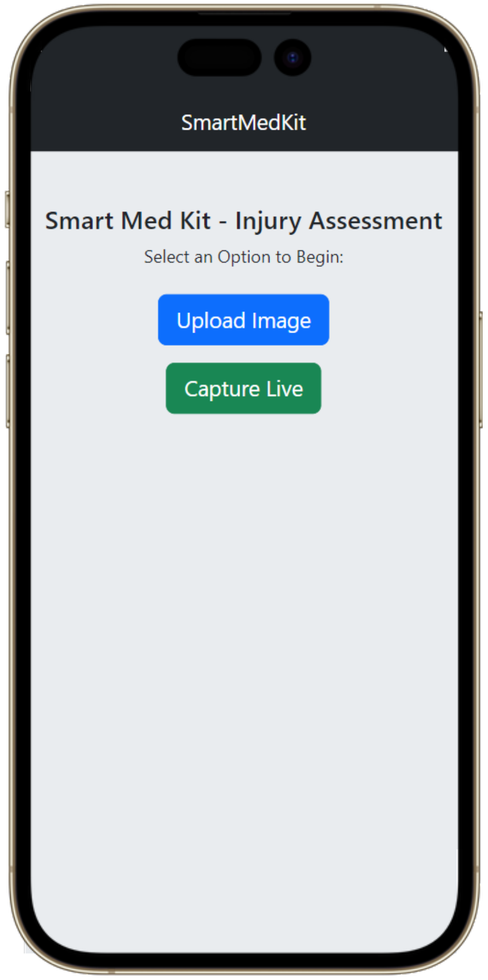


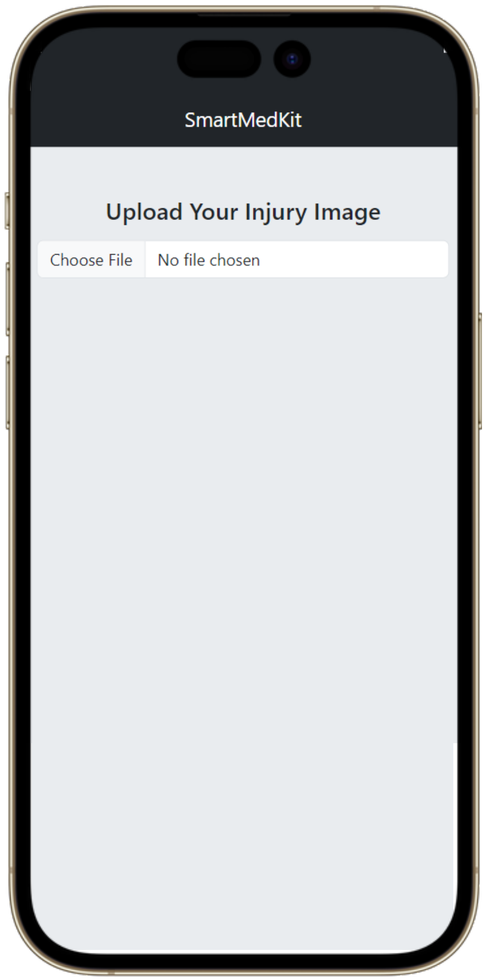


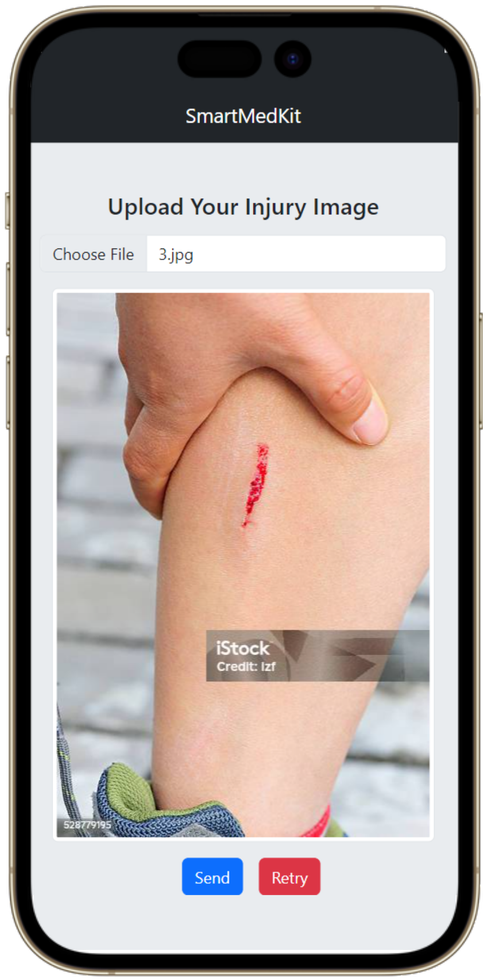
Frontend Interface and User Experience

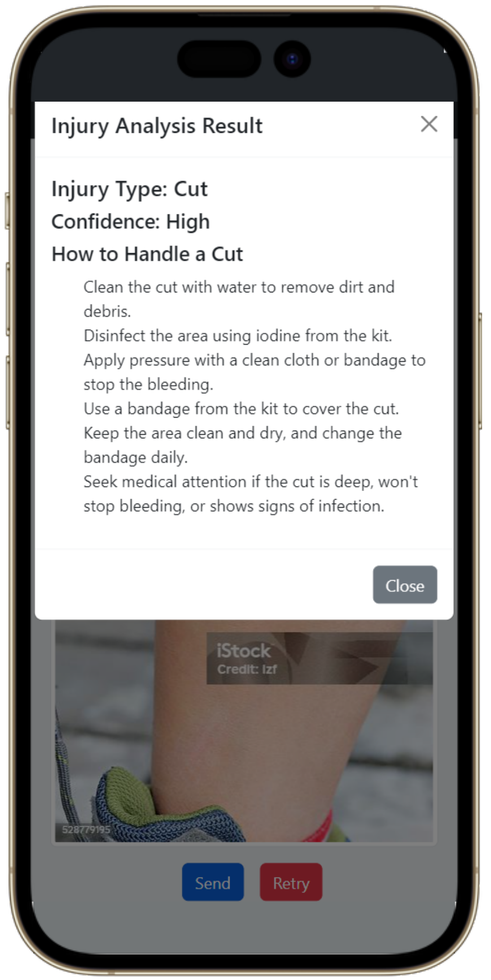
Navigating Through the SmartMedKit Web App

User









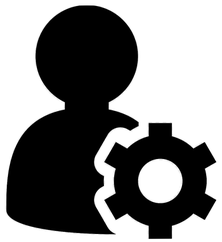
Frontend Interface and User Experience

Navigating Through the SmartMedKit Web App

Frontend Interface and User Experience

Navigating Through the SmartMedKit Web App

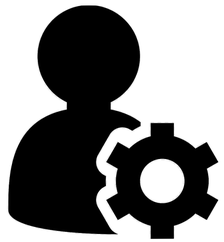
Admin

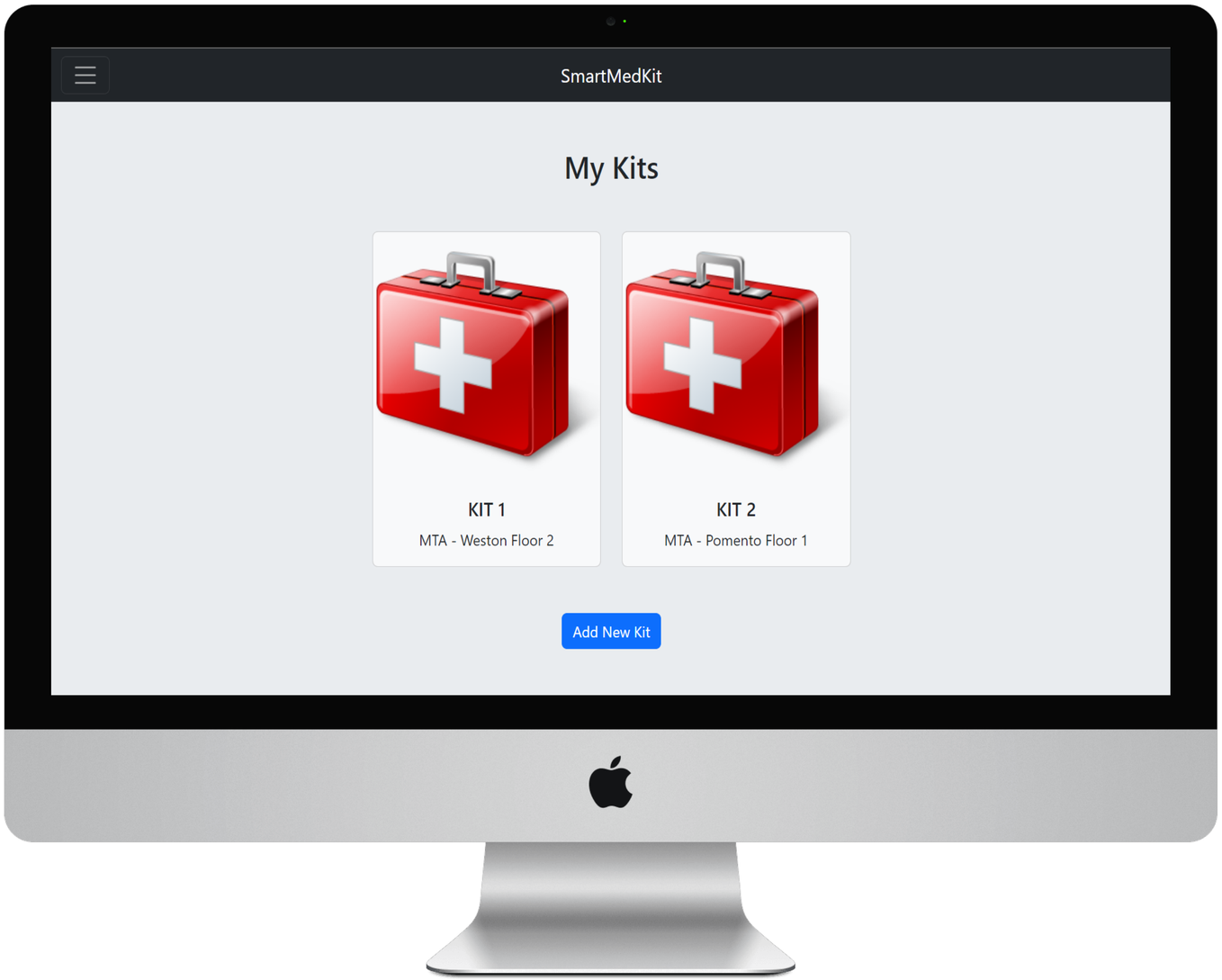


Frontend Interface and User Experience

Navigating Through the SmartMedKit Web App

Admin

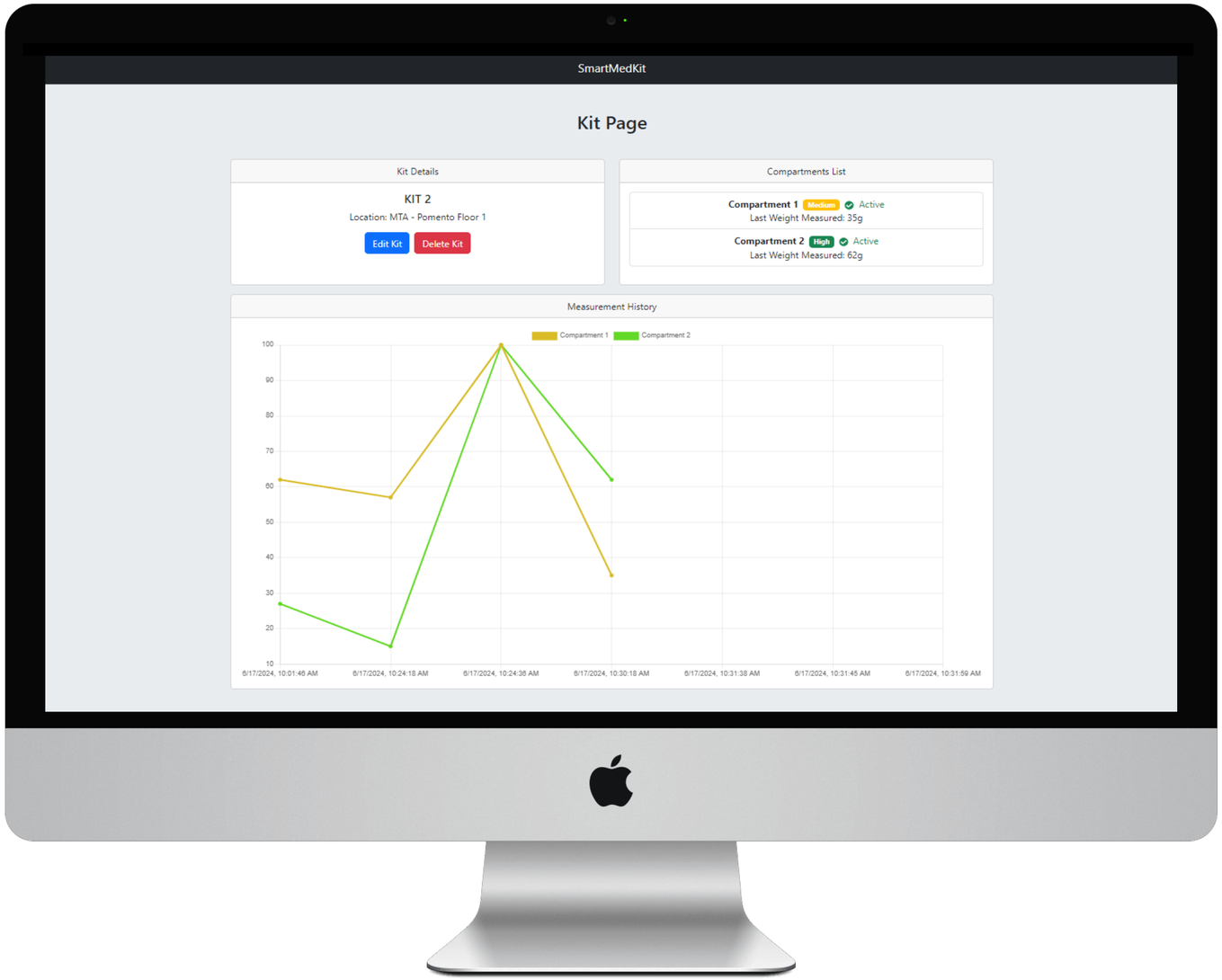




Frontend Interface and User Experience

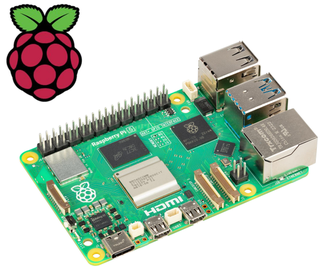
Navigating Through the SmartMedKit Web App

Admin

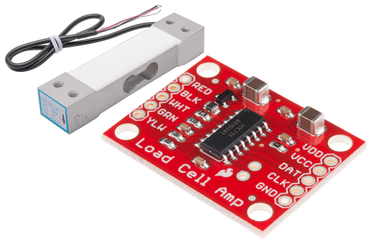


Embedded Tech Stack

Integrating Smart Hardware Solutions



Raspberry Pi: The core of our embedded system, chosen for its versatility and strong community support.



Weight Sensors (Load Cells) & HX711 Amps: For monitoring stock levels in real-time.

python™

Python: For scripting and controlling the hardware.

AUTODESK®
FUSION 360™

Fusion 360: For designing the physical aid kit, ensuring optimal functionality and integration.

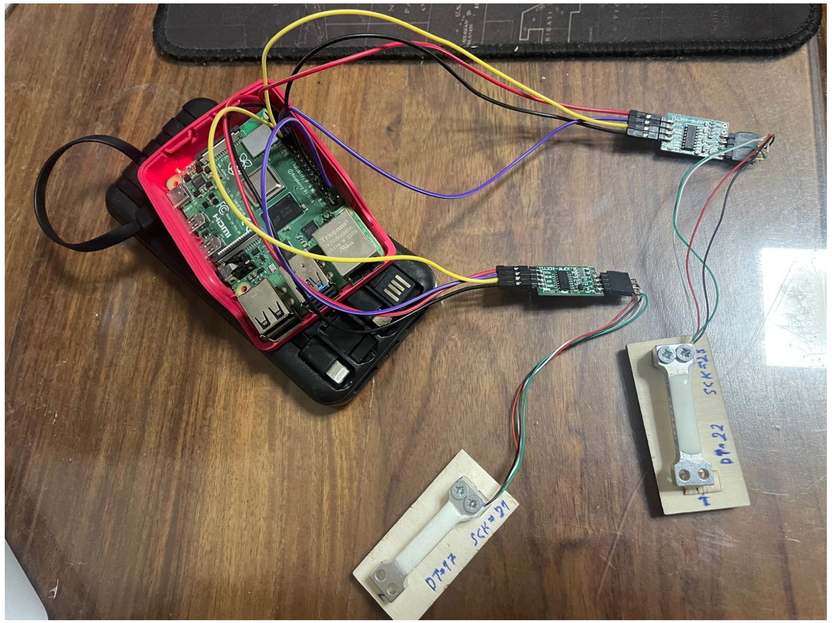
Embedded System Integration

Connecting Hardware to the SmartMedKit Backend

Embedded System Integration

Connecting Hardware to the SmartMedKit Backend

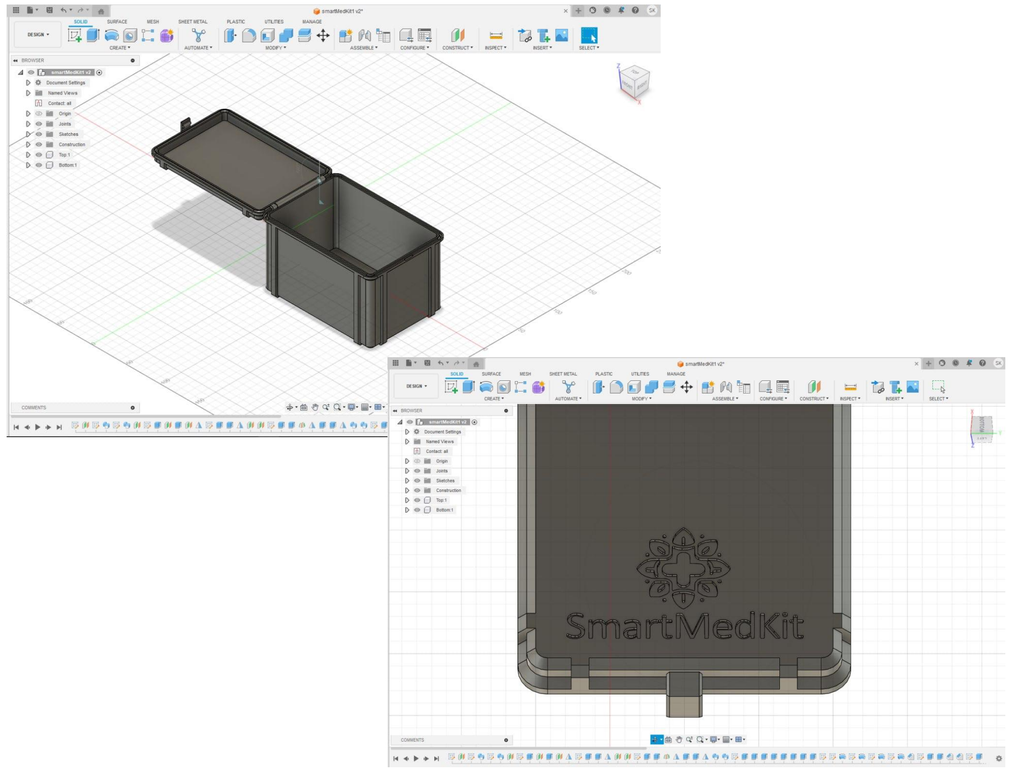
• **Soldering and Assembly:** Secure and reliable connections between components ensure consistent performance.



Embedded System Integration

Connecting Hardware to the SmartMedKit Backend

- **Soldering and Assembly:** Secure and reliable connections between components ensure consistent performance.



Embedded System Integration

Connecting Hardware to the SmartMedKit Backend

- **Soldering and Assembly:** Secure and reliable connections between components ensure consistent performance.
- **Weight Calibration**: Python scripts include calibration routines to ensure accurate weight measurements.

Embedded System Integration

Connecting Hardware to the SmartMedKit Backend

```
import RPi.GPIO as GPIO  # import GPIO
from hx711 import HX711  # import the class HX711
```

- **Soldering and Assembly:** Secure and reliable connections between components ensure consistent performance.
- **Weight Calibration**: Python scripts include calibration routines to ensure accurate weight measurements.

Embedded System Integration

Connecting Hardware to the SmartMedKit Backend

- **Soldering and Assembly:** Secure and reliable connections between components ensure consistent performance.
- **Weight Calibration**: Python scripts include calibration routines to ensure accurate weight measurements.

```python
import RPi.GPIO as GPIO  # import GPIO
from hx711 import HX711  # import the class HX711
```

```python
def setup_scale(dout_pin, pd_sck_pin, calibration_file):
    hx = HX711(dout_pin=dout_pin, pd_sck_pin=pd_sck_pin)
    err = hx.zero()
    if err:
        raise ValueError('Tare is unsuccessful.')

    ratio = load_calibration_data(calibration_file)
    if ratio is None:
        reading = hx.get_raw_data_mean()
        if reading:
            print('Data subtracted by offset but still not converted to units:', reading)
        else:
            print('invalid data', reading)

        input('Put known weight on the scale and then press Enter')
        reading = hx.get_data_mean()
        if reading:
            print('Mean value from HX711 subtracted by offset:', reading)
            known_weight_grams = input('Write how many grams it was and press Enter: ')
            try:
                value = float(known_weight_grams)
                print(value, 'grams')
            except ValueError:
                print('Expected integer or float and I have got:', known_weight_grams)

            ratio = reading / value
            save_calibration_data(ratio, calibration_file)
            print('Ratio is set.')
        else:
            raise ValueError('Cannot calculate mean value. Try debug mode. Variable reading:', reading)
    else:
        print('Loaded ratio from file.')

    hx.set_scale_ratio(ratio)
    return hx
```

Embedded System Integration

Connecting Hardware to the SmartMedKit Backend

- **Soldering and Assembly:** Secure and reliable connections between components ensure consistent performance.
- **Weight Calibration**: Python scripts include calibration routines to ensure accurate weight measurements.
- **Measurement Loop:** Periodically measures weight using the load cells and HX711 amps.

Embedded System Integration

Connecting Hardware to the SmartMedKit Backend

- **Soldering and Assembly:** Secure and reliable connections between components ensure consistent performance.
- **Weight Calibration**: Python scripts include calibration routines to ensure accurate weight measurements.
- **Measurement Loop:** Periodically measures weight using the load cells and HX711 amps.

```
try:
    GPIO.setmode(GPIO.BCM)  # set GPIO pin mode to BCM numbering
    hx1 = setup_scale(22, 23, CALIBRATION_FILE_1)
    hx2 = setup_scale(17, 21, CALIBRATION_FILE_2)

    print("Now, I will read data in infinite loop. To exit press 'CTRL + C'")
    input('Press Enter to begin reading')
    print('Current weight on the scales in grams is: ')
    while True:
        weight = hx1.get_weight_mean(20)
        print(weight, 'g')
        send_json(1, weight)
        weight = hx2.get_weight_mean(20)
        print(weight, 'g')
        send_json(2, weight)

        time.sleep(30)

except (KeyboardInterrupt, SystemExit):
    print('Bye :)')

finally:
    GPIO.cleanup()
```

Embedded System Integration

Connecting Hardware to the SmartMedKit Backend

- **Soldering and Assembly:** Secure and reliable connections between components ensure consistent performance.
- **Weight Calibration**: Python scripts include calibration routines to ensure accurate weight measurements.
- **Measurement Loop:** Periodically measures weight using the load cells and HX711 amps.
- **Data Transmission:** Collected data is sent to the backend API.

Embedded System Integration

Connecting Hardware to the SmartMedKit Backend

- **Soldering and Assembly:** Secure and reliable connections between components ensure consistent performance.
- **Weight Calibration**: Python scripts include calibration routines to ensure accurate weight measurements.
- **Measurement Loop:** Periodically measures weight using the load cells and HX711 amps.
- **Data Transmission:** Collected data is sent to the backend API.

```python
def send_json(scale_id, reading):
    # Round the weight to 2 decimal places
    weight = Decimal(reading).quantize(Decimal('0.00'))
    # Get the current timestamp
    timestamp = int(time.time())

    # Create JSON objects
    json1 = {
        "weight": float(weight),
        "timestamp": timestamp,
        "kit_id": 1,
        "compartment_id": scale_id
    }

    # Convert to JSON format
    json_str = json.dumps(json1)

    # Send POST requests
    url = "http://192.168.1.205:5000/measurements"
    headers = {'Content-Type': 'application/json'}

    response = requests.post(url, data=json_str, headers=headers)

    # Check the responses
    if response.status_code == 201:
        print("JSON was successfully sent.")
    else:
        print("Failed to send JSON.")
```

Current Status and Future of SmartMedKit

Where We Are Now and Where We’re Headed

Current Status

Future Enhancements

Current Status and Future of SmartMedKit

Where We Are Now and Where We’re Headed

Current Status

Model: Trained to distinguish between cuts and burns, Reliable but limited to initial dataset.

Future Enhancements

Current Status and Future of SmartMedKit

Where We Are Now and Where We’re Headed

Current Status

Model: Trained to distinguish between cuts and burns, Reliable but limited to initial dataset.

Future Enhancements

Improved Model: Trained on a larger and more diverse dataset for more accurate injury identification and treatment instructions.

Current Status and Future of SmartMedKit

Where We Are Now and Where We're Headed

Current Status

Model: Trained to distinguish between cuts and burns, Reliable but limited to initial dataset.

Inventory Tracking: Tracking stock levels based on weight sensors.

Future Enhancements

Improved Model: Trained on a larger and more diverse dataset for more accurate injury identification and treatment instructions.

Current Status and Future of SmartMedKit

Where We Are Now and Where We're Headed

Current Status

Model: Trained to distinguish between cuts and burns, Reliable but limited to initial dataset.

Inventory Tracking: Tracking stock levels based on weight sensors.

Future Enhancements

Improved Model: Trained on a larger and more diverse dataset for more accurate injury identification and treatment instructions.

Expiration Tracking: Adding capability to track expiration dates of items.

Current Status and Future of SmartMedKit

Where We Are Now and Where We're Headed

Current Status

Model: Trained to distinguish between cuts and burns, Reliable but limited to initial dataset.

Inventory Tracking: Tracking stock levels based on weight sensors.

Notifications: Email notifications for low stock.

Future Enhancements

Improved Model: Trained on a larger and more diverse dataset for more accurate injury identification and treatment instructions.

Expiration Tracking: Adding capability to track expiration dates of items.

Current Status and Future of SmartMedKit

Where We Are Now and Where We’re Headed

Current Status

Model: Trained to distinguish between cuts and burns, Reliable but limited to initial dataset.

Inventory Tracking: Tracking stock levels based on weight sensors.

Notifications: Email notifications for low stock.

Future Enhancements

Improved Model: Trained on a larger and more diverse dataset for more accurate injury identification and treatment instructions.

Expiration Tracking: Adding capability to track expiration dates of items.

Advanced Notifications: Integration with WhatsApp and other messaging services for customizable notification preferences.

Current Status and Future of SmartMedKit

Where We Are Now and Where We’re Headed

Current Status

Model: Trained to distinguish between cuts and burns, Reliable but limited to initial dataset.

Inventory Tracking: Tracking stock levels based on weight sensors.

Notifications: Email notifications for low stock.

Admin Dashboard: Basic stock, kit management and usage statistics.

Future Enhancements

Improved Model: Trained on a larger and more diverse dataset for more accurate injury identification and treatment instructions.

Expiration Tracking: Adding capability to track expiration dates of items.

Advanced Notifications: Integration with WhatsApp and other messaging services for customizable notification preferences.

Current Status and Future of SmartMedKit

Where We Are Now and Where We're Headed

Current Status

Model: Trained to distinguish between cuts and burns, Reliable but limited to initial dataset.

Inventory Tracking: Tracking stock levels based on weight sensors.

Notifications: Email notifications for low stock.

Admin Dashboard: Basic stock, kit management and usage statistics.

Future Enhancements

Improved Model: Trained on a larger and more diverse dataset for more accurate injury identification and treatment instructions.

Expiration Tracking: Adding capability to track expiration dates of items.

Advanced Notifications: Integration with WhatsApp and other messaging services for customizable notification preferences.

Advanced Admin Dashboard: Improved data searching and analysis tools, providing smart stock information and predictive insights based on historical usage.

Current Status and Future of SmartMedKit

Where We Are Now and Where We're Headed

Current Status

Model: Trained to distinguish between cuts and burns, Reliable but limited to initial dataset.

Inventory Tracking: Tracking stock levels based on weight sensors.

Notifications: Email notifications for low stock.

Admin Dashboard: Basic stock, kit management and usage statistics.

User Interface: intuitive user interface for uploading injury photos and receiving treatment instructions.

Future Enhancements

Improved Model: Trained on a larger and more diverse dataset for more accurate injury identification and treatment instructions.

Expiration Tracking: Adding capability to track expiration dates of items.

Advanced Notifications: Integration with WhatsApp and other messaging services for customizable notification preferences.

Advanced Admin Dashboard: Improved data searching and analysis tools, providing smart stock information and predictive insights based on historical usage.

Conclusion and Q&A

Wrapping Up and Open for Questions

Innovative Solution: SmartMedKit combines advanced technology with practical functionality to address critical needs in medical supply management and injury treatment.

Current Achievements: Effective injury identification, reliable inventory tracking, and user-friendly interfaces.

Thank you for your attention. We appreciate your interest in SmartMedKit and look forward to sharing our progress as we continue to innovate.