Smart Wheelchair for persons with disabilities in lower body and hearing impairements

Project ID

24 – 25J - 048

MEET THE TEAM

Ms. Dinithi Pandithage
Supervisor

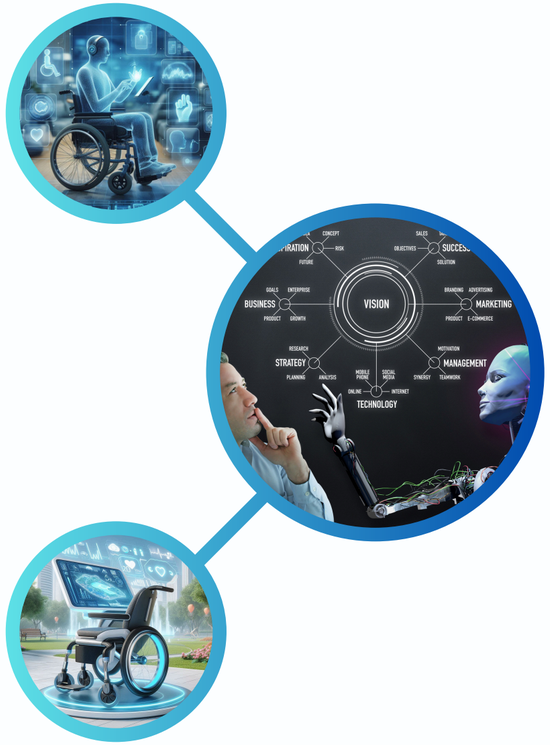
Ms. Shashika Lokuliyana
Co-supervisor

T.M.S.Thennakoon
Team Leader

H.G.K.D.Prematilaka
Member

S.N. Bentotage
Member

A.W.D.M. Samarawickrama
Member



INTRODUCTION

This research develops a smart wheelchair for individuals with disabilities and hearing impairments. It integrates LIDAR-based pathfinding for safe navigation, a health monitoring system to track vital signs, a personalized exercise plan for rehabilitation, and sign language navigation for intuitive control. The system enhances independence, safety, and accessibility.

Research Question

- How can we develop advanced mobility solutions that enhance independence for individuals with lower body disabilities, addressing their unique challenges?

- What innovative technologies can be integrated to improve safety, communication, and care for hearing-impaired users, considering their specific needs?

- How can we create comprehensive systems that not only reduce caregiver burden but also provide personalized health monitoring and support for individuals with disabilities?

MAIN OBJECTIVE

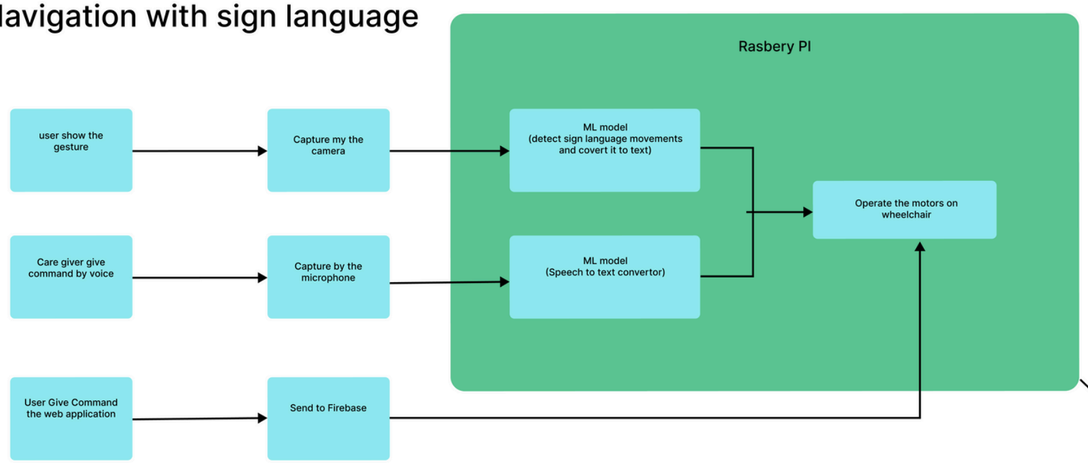
To create an intelligent smart wheelchair that improves mobility, accessibility, and quality of life for individuals with lower body disabilities and hearing impairments by integrating advanced navigation, real-time tracking, health monitoring, and personalized exercise features.

SUB OBJECTIVES

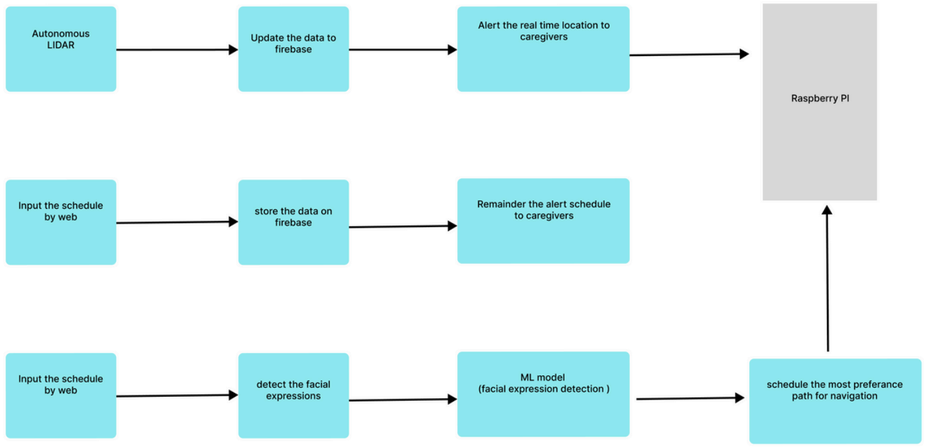
- Develop sign language control for hearing-impaired users to navigate the wheelchair.
- Implement pathfinding with real-time tracking and task reminders for enhanced mobility.
- Integrate a health monitoring system to continuously track and manage vital signs.
- Create an adaptive exercise plan with real-time motion detection for personalized physical therapy.

SYSTEM BLOCK DIAGRAM

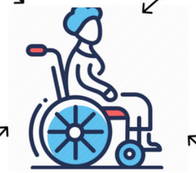
Navigation with sign language



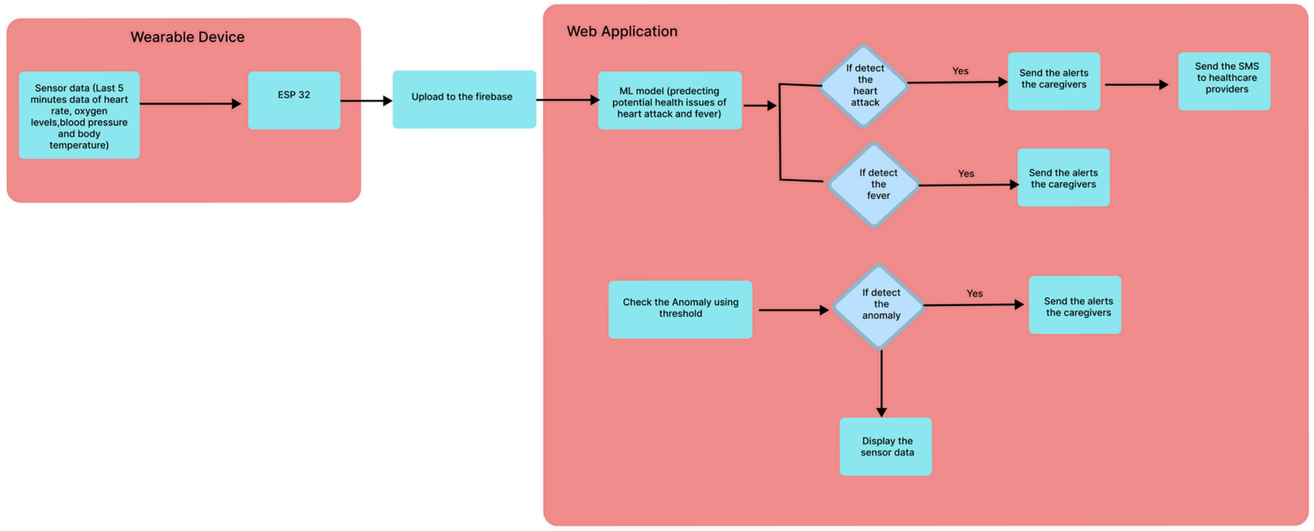
Pathfinding and Path planning



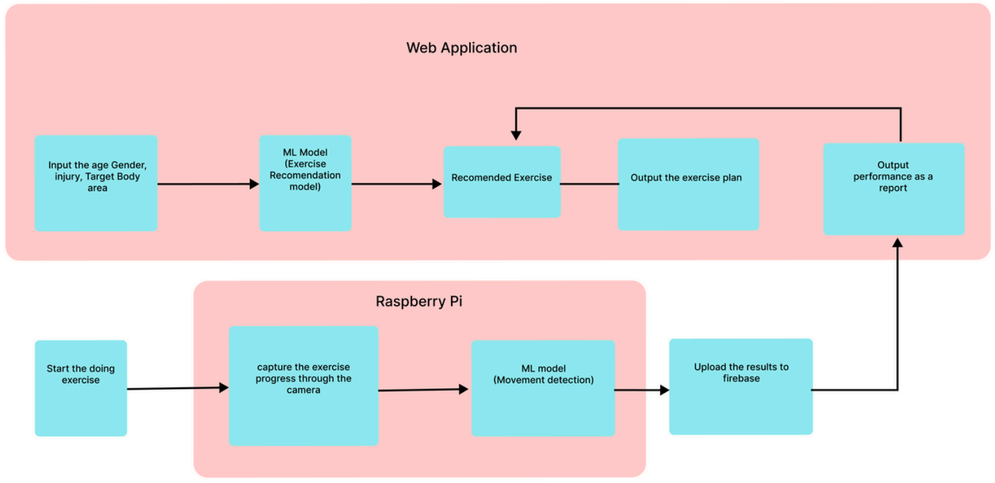
Smart Wheelchair



Real time Health Monitoring



Personalized Exercise Recommondation



H.G.K.D. PREMATHILAKA

BSc (Hons) in Information Technology

NAVIGATING WHEELCHAIR AND ENHANCING ACCESSIBILITY USING SIGN LANGUAGE-CONTROLLED FOR HEARING-IMPAIRED USERS

RESEARCH PROBLEM

How can wheelchair control systems be designed to effectively cater to hearing-impaired individuals, overcoming the limitations of traditional and voice-based control methods to enhance accessibility and independence?

HOW CAN WE OVERCOME THIS?

By integrating advanced computer vision techniques for real-time sign language recognition, we can develop a more intuitive and accessible control system for hearing-impaired users. This will enhance their ability to navigate and operate wheelchairs independently and safely.

RESEARCH GAP

System/Reference	Sign Language Recognition	Real-Time Gesture Detection	Culturally Relevant Gestures	Accessible for Hearing-Impaired
[1] Nair et al. (2021)	❌ No	✅ Yes	❌ No	❌ No
[2] Aizam et al. (2020)	❌ No	❌ No	❌ No	❌ No
[3] Patankar et al. (2023)	❌ No	✅ Yes	❌ No	❌ No
[4] Kengale et al. (2021)	❌ No	❌ No	❌ No	❌ No
[5] Deshmukh et al. (2021)	❌ No	❌ No	❌ No	❌ No
PROPOSED SYSTEM	✅ Yes	✅ Yes	✅ Yes	✅ Yes

TECHNOLOGIES, ALGORITHMS

TECHNOLOGIES

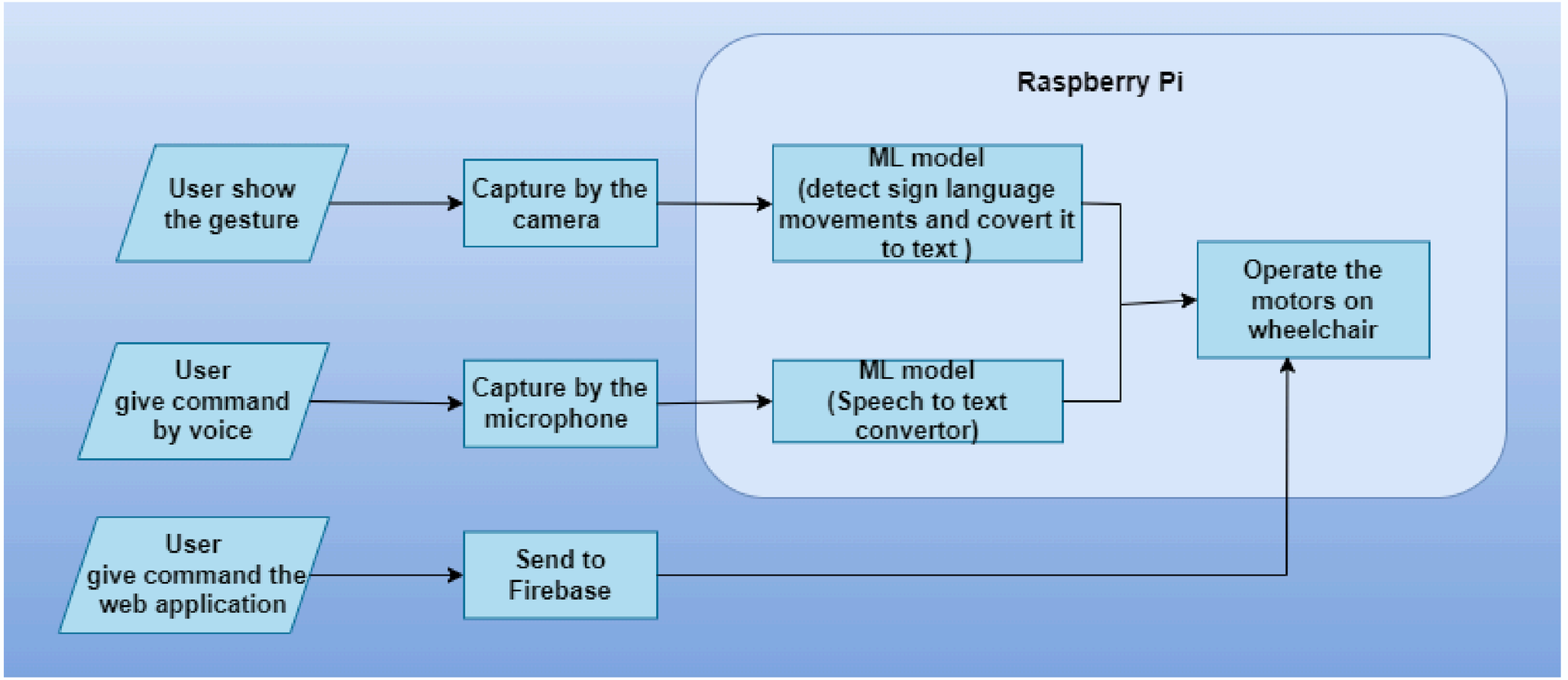
- **Python**: The primary programming language used for writing code.
- **OpenCV**: For real-time computer vision tasks, such as reading video frames and performing image processing.
- **NumPy**: Essential for numerical computations and manipulating data arrays.
- **WebSocket**:Real-time communication protocol for sending detection results from the server to a client.

ALGORITHMS

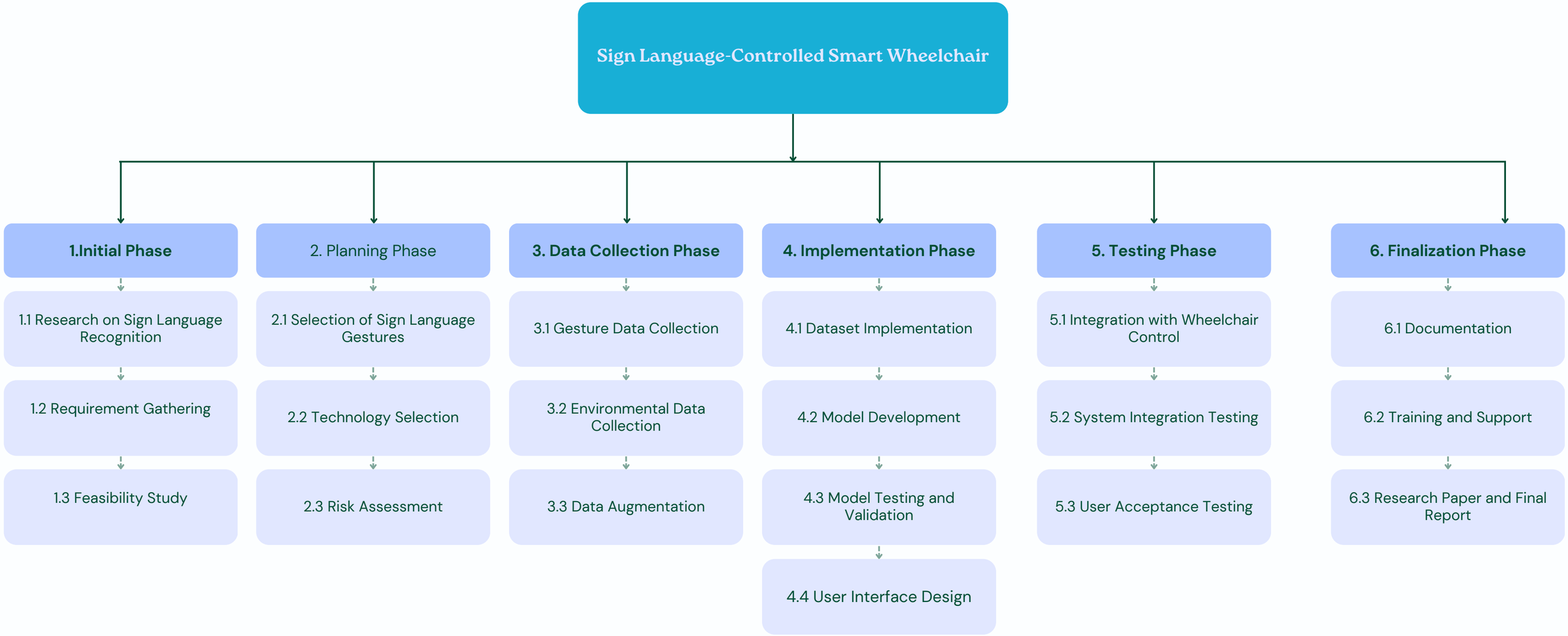
YOLO (You Only Look Once) Algorithm

- Type: Object Detection Algorithm
- Version Used: YOLOv5
- Purpose: Detects and localizes ASL gestures in real-time video frames by predicting bounding boxes and class labels in one pass.

COMPONENT BLOCK DIAGRAM



WORK BREAKDOWN STRUCTURE



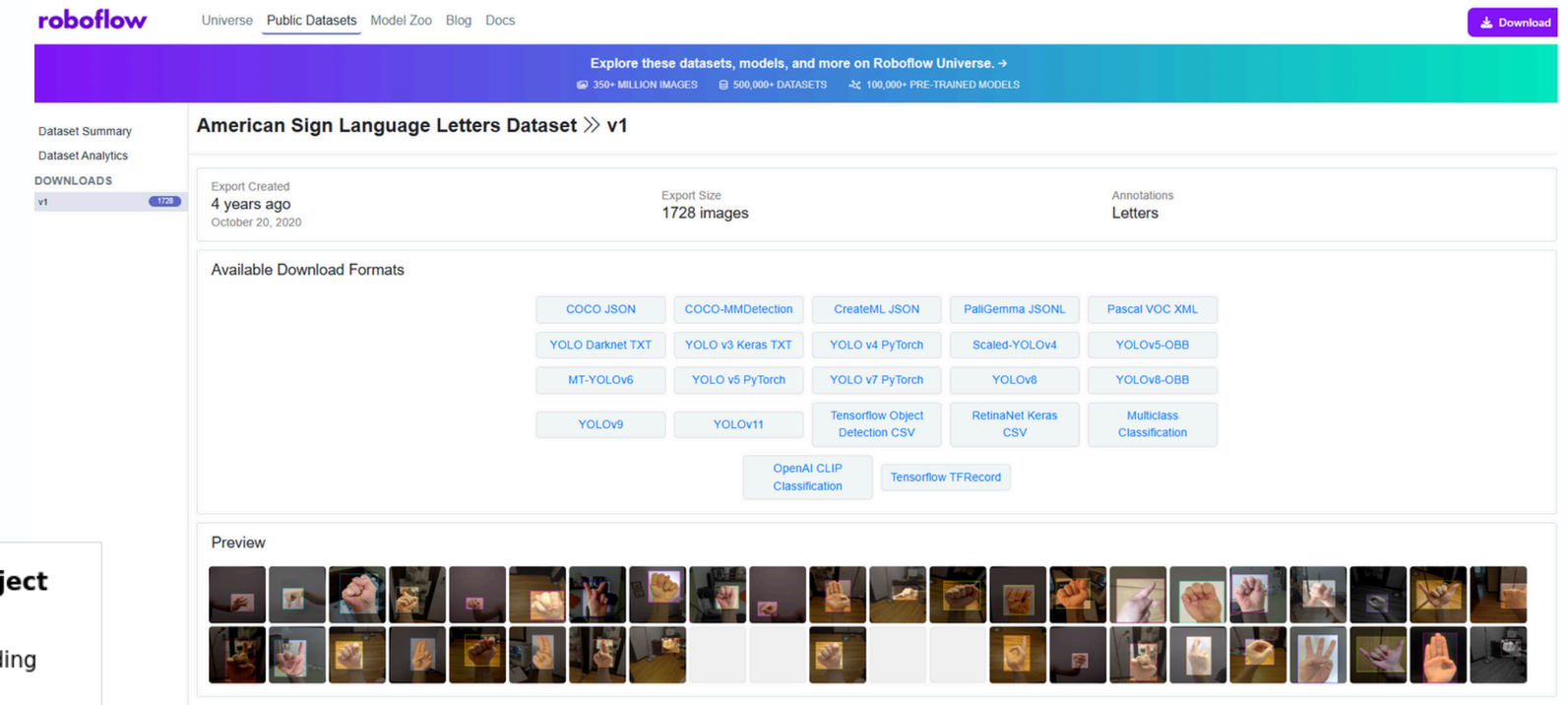
DATASETS

AMERICAN SIGN LANGUAGE GESTURE DATASET

American Sign Language Letters Object Detection Dataset - v1

Download 720 free images labeled with bounding boxes for object detection.

Roboflow



DATASET PRE-PROCESSING

```
!git clone https://github.com/ultralytics/yolov5
%cd ./yolov5
!  pip install -r requirements.txt
```

```
Cloning into 'yolov5'...
remote: Enumerating objects: 16977, done.
remote: Counting objects: 100% (172/172), done.
remote: Compressing objects: 100% (119/119), done.
remote: Total 16977 (delta 90), reused 109 (delta 53), pack-reused 16805 (from 1)
Receiving objects: 100% (16977/16977), 15.71 MiB | 19.52 MiB/s, done.
Resolving deltas: 100% (11619/11619), done.
/content/yolov5/yolov5
Requirement already satisfied: gitpython>=3.1.30 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line 5)) (3.1.43)
Requirement already satisfied: matplotlib>=3.3 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line 6)) (3.7.1)
Requirement already satisfied: numpy>=1.23.5 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line 7)) (1.26.4)
Requirement already satisfied: opencv-python>=4.1.1 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line 8)) (4.10.0.84)
Requirement already satisfied: pillow>=10.3.0 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (lin
Requirement already satisfied: psutil in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line 10)) (
Requirement already satisfied: PyYAML>=5.3.1 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line
Requirement already satisfied: requests>=2.32.2 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (l
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line
Requirement already satisfied: thop>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line 1
Requirement already satisfied: torch>=1.8.0 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line
Requirement already satisfied: torchvision>=0.9.0 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt
Requirement already satisfied: tqdm>=4.66.3 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line
Requirement already satisfied: ultralytics>=8.2.34 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt
Requirement already satisfied: pandas>=1.1.4 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line
Requirement already satisfied: seaborn>=0.11.0 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (li
Requirement already satisfied: setuptools>=70.0.0 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt
...
Requirement already satisfied: smmap<6,>=3.0.1 in /usr/local/lib/python3.10/dist-packages (from gitdb<5,>=4.0.1->gitpyt
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotl
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch>=1.8.0->-
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from sympy->torch>=1.8.0-
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Dataset

```
%cd /content
!curl -L 'https://public.roboflow.com/ds/RzcCBKoedH?key=DvQRgYMRpF' > roboflow.zip; unzip roboflow.zip; rm roboflow.zip
```

```
/content
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   910  100   910    0     0   3518      0 --:--:-- --:--:-- --:--:--  3513
100 22.3M  100 22.3M    0     0  27.0M      0 --:--:-- --:--:-- --:--:-- 27.0M
Archive:  roboflow.zip
 extracting: test/images/G7_jpg.rf.04faf434dd590a7bd02818b2b57a704f.jpg
 extracting: test/images/S6_jpg.rf.0b6e42445a56998369cdd3759c2cf3d4.jpg
 extracting: test/images/T24_jpg.rf.068e7b2424eac996c86bb6d9c38e083d.jpg
 extracting: test/images/K12_jpg.rf.1bd849de853e33f6262b6643555e098f.jpg
 extracting: test/images/J9_jpg.rf.213a9e356777a13d336e4988c33a93e1.jpg
 extracting: test/images/X9_jpg.rf.28187742b082579d85eda81cf98bf5bb.jpg
 extracting: test/images/B15_jpg.rf.0f0628552139144fc67c453e1f1b7b15.jpg
 extracting: test/images/W24_jpg.rf.1bc37c6fc4770b284edc6f9027eb6cf2.jpg
 extracting: test/images/S16_jpg.rf.138a4aa81180adcc24323c1634ed507e.jpg
 extracting: test/images/U5_jpg.rf.2be92300b664a1946dde8902dc50adbb.jpg
 extracting: test/images/V12_jpg.rf.0e68ec7ee88bd6f0f70fe5496e893068.jpg
 extracting: test/images/R5_jpg.rf.1e94c77f430ee342744dc9fce202c449.jpg
 extracting: test/images/Z18_jpg.rf.2cffbd9beaeb50a7d03751c3ce738e81.jpg
 extracting: test/images/Z16_jpg.rf.309328aaeb31736f8a93a570d6d4f140.jpg
 extracting: test/images/V10_jpg.rf.18af9b37cbe013f9a0640294fb71f0d6.jpg
 extracting: test/images/Q7_jpg.rf.07fdf6c096cd2a9be72b4de4a627935d.jpg
 extracting: test/images/Z27_jpg.rf.0a3e757898215b17678047cb5485c82c.jpg
 extracting: test/images/X20_jpg.rf.2a36699005b7e1881e5faa278c506b86.jpg
 extracting: test/images/T18_jpg.rf.334025dd29760be4f8918fd6d6c3f01c.jpg
...
 extracting: valid/labels/H18_jpg.rf.ff988df01210143b03fc11601dee2f5b.txt
 extracting: valid/labels/T8_jpg.rf.f1522d0cdad081aebc3e3f495a9e3087.txt
 extracting: valid/labels/T4_jpg.rf.d9d1cb8487bd3a4bc1851721b0c516c5.txt
 extracting: README.roboflow.txt
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Generate + Code + Markdown

DATASET TRAINING

Training

```
!python /content/yolov5/train.py --img 448 --batch 64 --epochs 500 --data /content/data.yaml --weights yolov5s.pt --workers 8
```

```
Streaming output truncated to the last 5000 lines.
  with torch.cuda.amp.autocast(amp):
     99/499      8.27G     0.0225   0.008996    0.01867        126        448:  42% 10/24 [00:08<00:16,  1.16s/it]/content/yolov5/tra
  with torch.cuda.amp.autocast(amp):
     99/499      8.27G    0.02286   0.009157    0.01938        156        448:  46% 11/24 [00:09<00:16,  1.25s/it]/content/yolov5/tra
  with torch.cuda.amp.autocast(amp):
     99/499      8.27G    0.02296   0.009098    0.01984        130        448:  50% 12/24 [00:11<00:14,  1.22s/it]/content/yolov5/tra
  with torch.cuda.amp.autocast(amp):
     99/499      8.27G    0.02303   0.009143    0.02007        151        448:  54% 13/24 [00:12<00:13,  1.25s/it]/content/yolov5/tra
  with torch.cuda.amp.autocast(amp):
     99/499      8.27G    0.02317   0.009116    0.02046        135        448:  58% 14/24 [00:12<00:10,  1.07s/it]/content/yolov5/tra
  with torch.cuda.amp.autocast(amp):
     99/499      8.27G    0.02307   0.009153    0.02052        143        448:  62% 15/24 [00:13<00:09,  1.03s/it]/content/yolov5/tra
  with torch.cuda.amp.autocast(amp):
     99/499      8.27G    0.02326   0.009197    0.02066        145        448:  67% 16/24 [00:14<00:07,  1.08it/s]/content/yolov5/tra
  with torch.cuda.amp.autocast(amp):
     99/499      8.27G    0.02333   0.009195    0.02069        128        448:  71% 17/24 [00:15<00:06,  1.09it/s]/content/yolov5/tra
  with torch.cuda.amp.autocast(amp):
     99/499      8.27G     0.0234   0.009211    0.02052        144        448:  75% 18/24 [00:16<00:05,  1.18it/s]/content/yolov5/tra
  with torch.cuda.amp.autocast(amp):
     99/499      8.27G    0.02349   0.009184    0.02008        143        448:  79% 19/24 [00:16<00:03,  1.26it/s]/content/yolov5/tra
  with torch.cuda.amp.autocast(amp):
     99/499      8.27G    0.02354    0.00921    0.02002        151        448:  83% 20/24 [00:17<00:03,  1.16it/s]/content/yolov5/tra
  with torch.cuda.amp.autocast(amp):
```

VALIDATION & ACCURACY

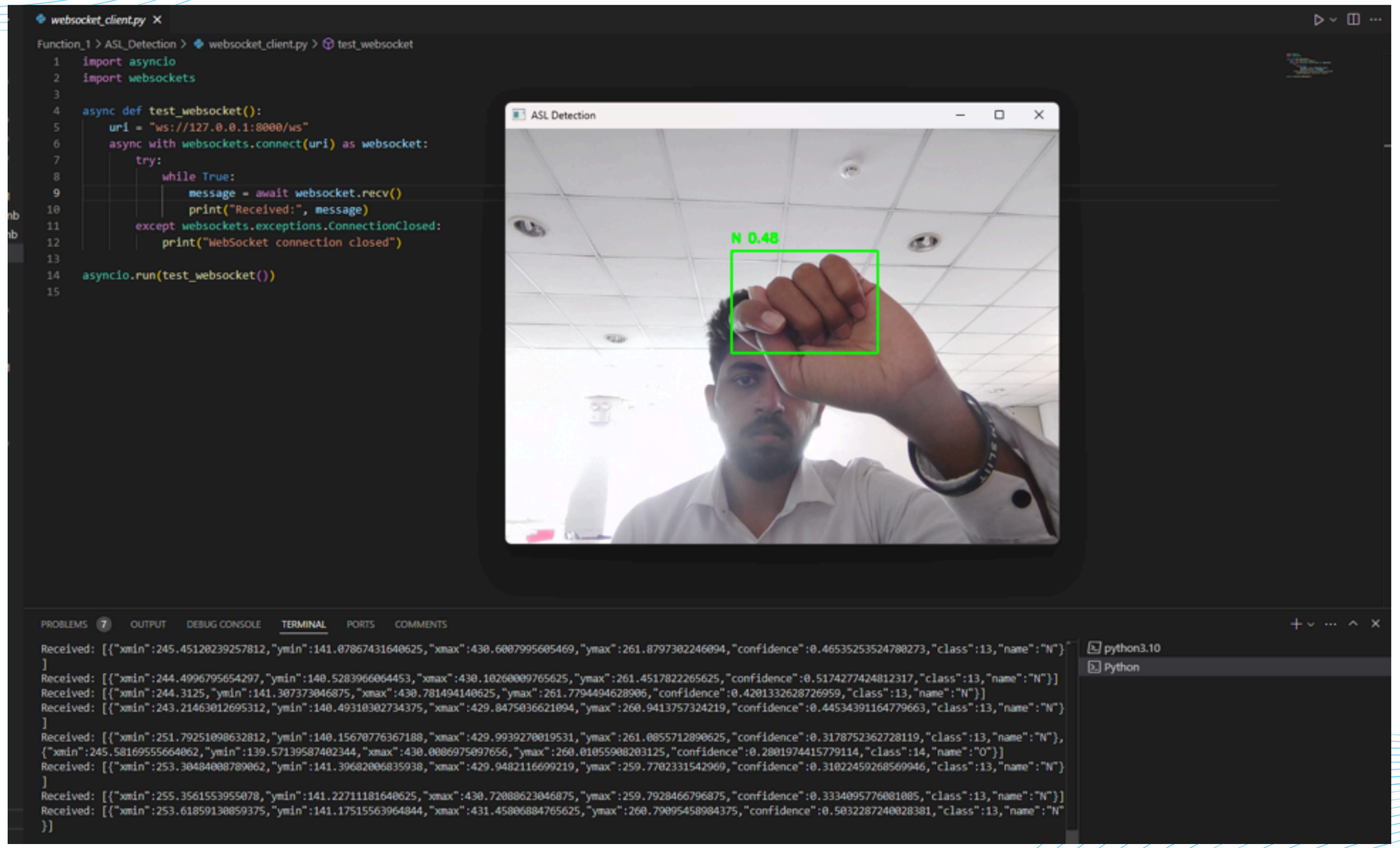
Validation

```
!python /content/yolov5/val.py --weights /content/yolov5/runs/train/exp/weights/best.pt --data /content/data.yaml --img 448
```

```
val: data=/content/data.yaml, weights=['/content/yolov5/runs/train/exp/weights/best.pt'], batch_size=32, imgsz=448, conf_thres=0.001, iou_thres=0.6, max_det=300, task=val, device=, workers=8, sing
YOLOv5 🚀 v7.0-371-g6629839d Python-3.10.12 torch-2.4.1+cu121 CUDA:0 (Tesla T4, 15102MiB)

Fusing layers...
Model summary: 157 layers, 7080247 parameters, 0 gradients, 16.0 GFLOPs
val: Scanning /content/valid/labels.cache... 144 images, 0 backgrounds, 0 corrupt: 100% 144/144 [00:00<?, ?it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 5/5 [00:05<00:00,  1.00s/it]
                   all        144        144      0.885      0.893       0.96      0.777
                     A        144          5      0.811      0.869      0.962      0.828
                     B        144          9          1      0.823      0.995      0.817
                     C        144          3          1      0.857      0.995      0.786
                     D        144          6          1      0.694      0.955      0.781
                     E        144          4          1      0.757      0.995      0.846
                     F        144          8      0.792          1      0.995      0.883
                     G        144          5      0.919          1      0.995      0.784
                     H        144          9          1      0.939      0.995      0.762
                     I        144          2       0.51        0.5       0.51      0.458
                     J        144          8      0.864          1      0.967      0.625
                     K        144          6      0.807      0.833      0.927      0.716
                     L        144          4      0.772          1      0.995      0.858
                     M        144          8      0.983          1      0.995      0.779
                     N        144          4      0.613          1      0.945      0.791
                     O        144          7      0.985          1      0.995      0.829
                     P        144          7          1      0.764      0.978      0.688
                     Q        144          4      0.981          1      0.995      0.796
...
                     Y        144          8          1      0.911      0.995      0.652
                     Z        144          4          1      0.971      0.995      0.846
Speed: 0.4ms pre-process, 6.0ms inference, 10.2ms NMS per image at shape (32, 3, 448, 448)
Results saved to yolov5/runs/val/exp2
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

TESTING PART



FUTURE IMPLEMENTATIONS

- **Motor Integration**: Interface motor controllers with Raspberry Pi for gesture-based speed and direction control with feedback mechanisms.
- **Raspberry Pi Setup**: Install necessary software and integrate sign language recognition for real-time gesture processing.
- **Camera Setup**: Configure and calibrate a high-quality camera for gesture recognition using OpenCV.
- **Adaptive Learning**: Use machine learning to personalize gestures and improve accuracy with a training mode.
- **Progress Monitoring & Updates**: Track performance, adjust sensitivity, and update the system based on user feedback.

REFERENCE

- [1]Nair, A., Bhatkar, A., Biju, B., Shetty, A., & Thomas, M. (2021). Design and Development of a Smart Wheelchair. 2021 International Conference on Nascent Technologies in Engineering (ICNTE 2021). Fr. C. Rodrigues Institute of Technology, Vashi, Navi Mumbai, India.

- [2]Aizam, M. A. M., Hadis, N. S. M., & Abdullah, S. (2020). IoT-Based Smart Wheelchair System for Physically Impaired Person. ESTEEM Academic Journal, Vol. 16, June 2020, 59-73. Universiti Teknologi MARA, Malaysia.

- [3]Patankar, N. S., Bhausaheb, G. S., Shaharam, D. A., Satish, P. S., & Deshmukh, T. R. (2023). IoT Based Hand Gesture Control Wheelchair for Disabled Person. Fourth International Conference on Smart Electronics and Communication (ICOSEC-2023).

- [4]Kengale, V., Bansod, K., Sure, C., Dalal, M., & Bawane, S. (2021). Designing of a Smart Wheelchair for People with Disabilities. Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV 2021).

- [5]Deshmukh, Y. S., Patankar, N. S., Bhausaheb, G. S., Shaharam, D. A., & Satish, P. S. (2021). IoT Based Hand Gesture Control Wheelchair for Disabled Person. Fourth International Conference on Smart Electronics and Communication (ICOSEC-2023).

- https://youtube.com/playlist?list=PLkz_y24mlSJYWpwFbU8fyaBSwihoVHiJz&si=1Ze5Yuqdj1_7mQ1b
- https://youtu.be/MdF6x6ZmLAY?si=VTgZBpcviJXa8fea

T.M.S. THENNAKOON

BSc (Hons) in Information Technology

Pathfinding with Real-Time Tracking and Task Reminder System.

The smart wheelchair's pathfinding system uses LIDAR for safe navigation and obstacle avoidance, with real-time tracking and task reminders. Caregivers receive alerts if the user deviates from the path, ensuring enhanced safety and support. Specially wheelchair capture the user's emotional state at each waypoint, allowing the system to factor in emotional preferences when planning the route.

Research Problem

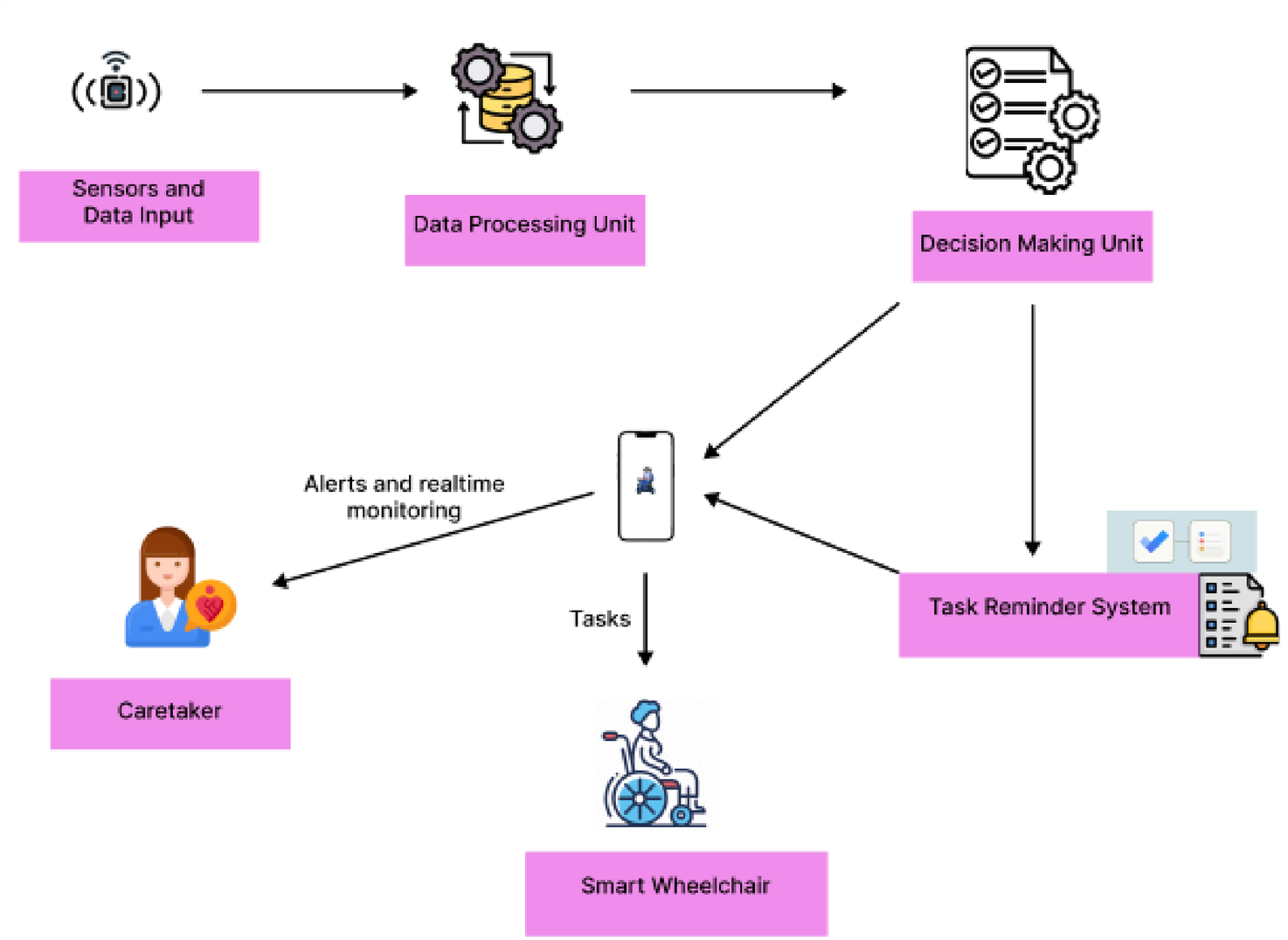
How can a smart wheelchair ensure safe navigation through complex environments, provide timely reminders for daily tasks, and allow caregivers to monitor the user's location and receive alerts if issues arise?

RESEARCH GAP TABLE

Comparison of Existing System

	LIDAR-BASED NAVIGATION	REAL-TIME TRACKING	TASK REMINDER SYSTEM	CAREGIVER ALERTS
[1]	✗	✗	✗	✗
[2]	✗	✗	✓	✓
[3]	✗	✗	✗	✗
PROPOSED SYSTEM	✓	✓	✓	✓

COMPONENT OVERVIEW DIAGRAM



Methodology

TECHNOLOGIES, ALGORITHMS

TECHNOLOGIES

- **Python**: The primary programming language used for developing the system, due to its versatility and extensive libraries for machine learning, data processing, and communication.
- **TensorFlow:** Utilized for machine learning and deep learning applications.
- **OpenCV**: For processing LIDAR data and enhancing obstacle detection, enabling the wheelchair to navigate complex environments in real time.
- **ROS (Robot Operating System):** Facilitates the integration of LIDAR and GPS, and supports the development of the pathfinding and tracking system.
- **Flask:** A lightweight web framework used to create APIs for the alert system, enabling communication between the wheelchair and the caregiver's mobile or web application.

ALORITHMS

- **A or D Pathfinding Algorithms:** :** To compute the optimal path for navigation, considering dynamic obstacles detected by LIDAR.
- **Geofencing Algorithms :** To define virtual boundaries and trigger alerts when the wheelchair crosses these boundaries or deviates from its planned path.
- **Dynamic Window Approach (DWA):** Used for local obstacle avoidance by considering the robot's velocity and acceleration constraints. DWA helps in making quick decisions to avoid collisions while following the planned path.

WORK BREAKDOWN STRUCTURE

Pathfinding with Real-Time Tracking and task reminder system.

- **Initial Phase**
  - 1.1 Research on pathfinding technology
  - 1.2 Requirement Gathering
  - 1.3 Feasibility Study
- **2. Planning Phase**
  - 2.1 Selection of environmental data
  - 2.2 Technology Selection
  - 2.3 Risk Assessment
- **3. Data Collection Phase**
  - 3.1 Environmental Data Collection🎯
  - 3.2 location and GPS Data Collection
  - 3.3 Data Augmentation
- **4. Implementation Phase**
  - 4.1 Dataset Implementation
  - 4.2 Model Development
  - 4.3 Integration with Wheelchair Control
  - 4.4 User Interface Design
- **5. Testing Phase**
  - 5.1 Model Testing and Validation
  - 5.2 System Integration Testing
  - 5.3 User Acceptance Testing
- **6. Finalization Phase**
  - 6.1 Documentation
  - 6.2 Training and Support
  - 6.3 Research Paper and Final Report

DATASET

FUNCTIONALITIES OF PATHFINDING, REAL-TIME TRACKING, AND TASK REMINDERS.

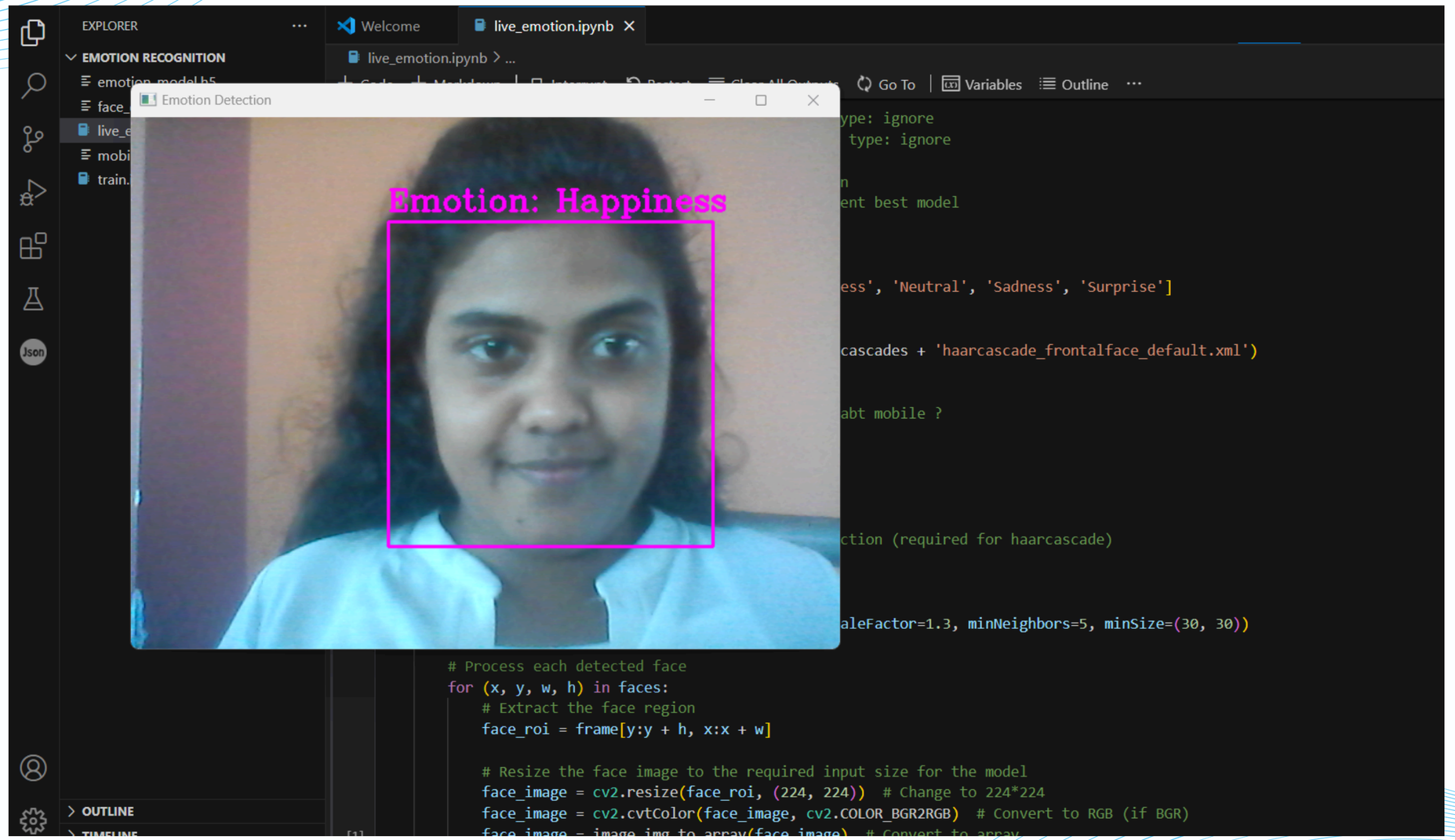
(FER2013)

DATASET PRE-PROCESSING

Data Loading and Pre Processing

```
INPUT_SIZE = (224, 224)
BATCH_SIZE = 80
INPUT_SIZE = (224, 224)

# Define FER2013 dataset paths
train_data_dir = 'fer2013/train'
val_data_dir = 'fer2013/val'

# Data generators
train_datagen = ImageDataGenerator(preprocessing_function=None, horizontal_flip=True)
test_datagen = ImageDataGenerator(preprocessing_function=None)

# Load data (change color_mode to 'rgb' since you're converting to RGB in preprocessing)
train_generator = train_datagen.flow_from_directory(
    train_data_dir, target_size=INPUT_SIZE,
    batch_size=BATCH_SIZE, class_mode='categorical', color_mode='rgb'
)

val_generator = test_datagen.flow_from_directory(
    val_data_dir, target_size=INPUT_SIZE,
    batch_size=BATCH_SIZE, class_mode='categorical', color_mode='rgb'
)

# Get one batch of images from the generator
x_batch, y_batch = next(train_generator)

# Check the shape of the first image in the batch
print(x_batch.shape)
```

CURRENT OUTPUT



FUTURE IMPLEMENTATIONS

- **Integrate SLAM toolbox :** for map drawing, detect wheel speed for accurate movement tracking, and use motor encoders for precise navigation.
- **Motor Integration**: Interface motor controllers with Raspberry Pi and LIDAR for autonomous navigation with avoiding the obstacles
- **Raspberry Pi Setup**: Install necessary software and integrate facial expression recognition for real-time path planning processing.
- **Camera Setup**: Configure and calibrate a high-quality camera for Facial expression recognition using OpenCV.
- **Data Augmentation :** Use machine learning to artificially increase the size and diversity of a dataset by applying transformations to existing data.

REFERENCE

- [1] S. Yang, The design and implementation of a scheduler and route planner for wheelchair users, University of British Columbia, 2006..
- [2] R.C.Simpson, Smart Wheelchairs, Department of Rehabilitation Science and Technology, University of Pittsburgh, Pittsburgh, PA, 2005
- [3] S. M. Mazharul Hoque Chowdhury, Sarnali Basak, Smart wheelchair for disable people, Department of Computer Science and Engineering Jahangirnagar University, 2019
- [4] Maryam Amur Khalfan Al Shabibi, Suresh Manic Kesavan, IOT for Smart Wheelchair, Department of Electrical and Communication Engineering, College of Engineering, National University of Science and Technology, Muscat, Oman

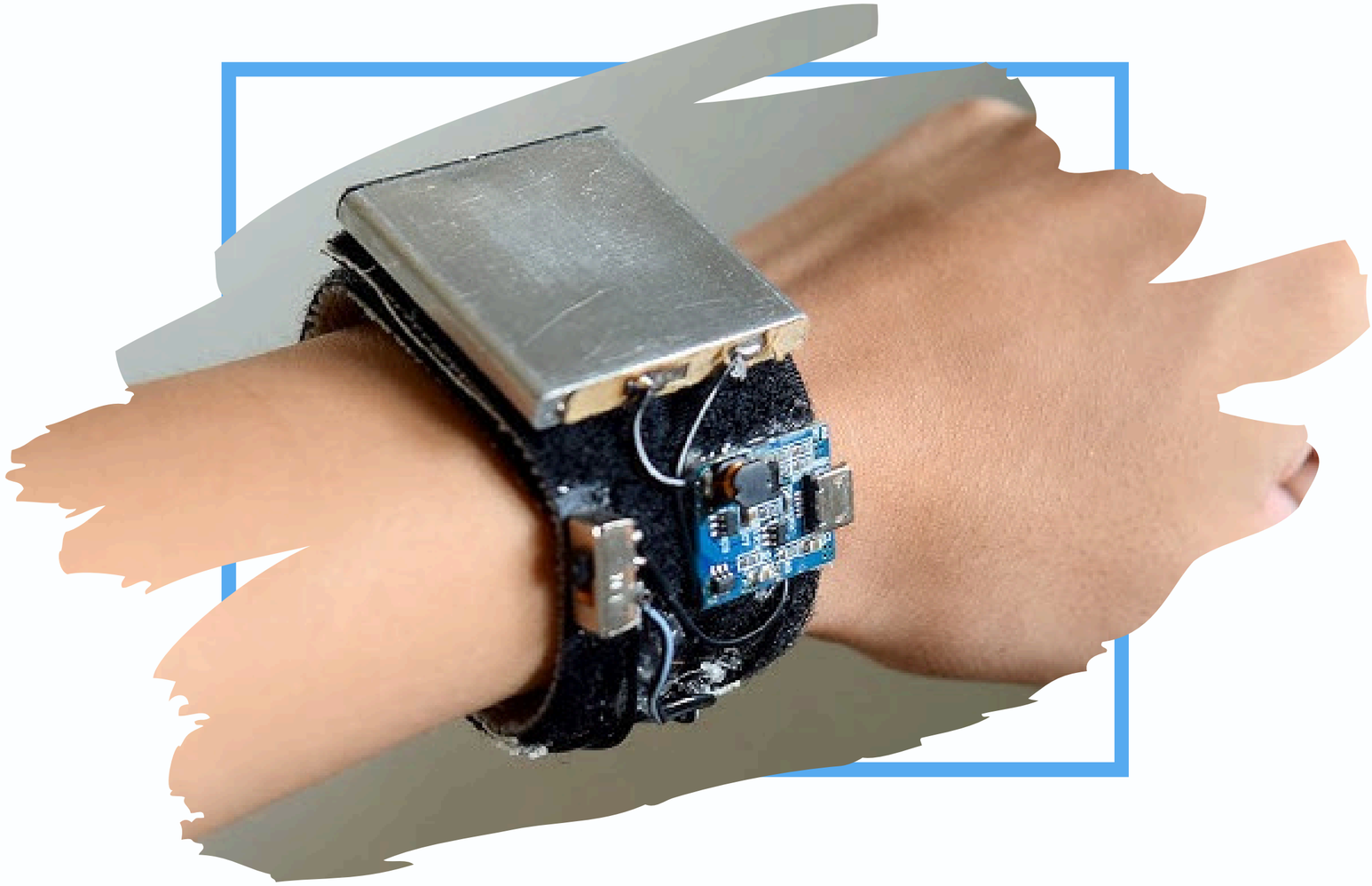
A.W.D.M.SAMARAWICKRAMA

1T21170652

BSc (Hons) in Information Technology

INTEGRATED HEALTH MONITORING SYSTEM

The 'Integrated Health Monitoring System' component continuously monitors the patient's vital health parameters such as heart rate, oxygen level, blood pressure and body temperature every 5 minutes. The system focuses on predicting potential health problems by continuously analyzing this collected data.



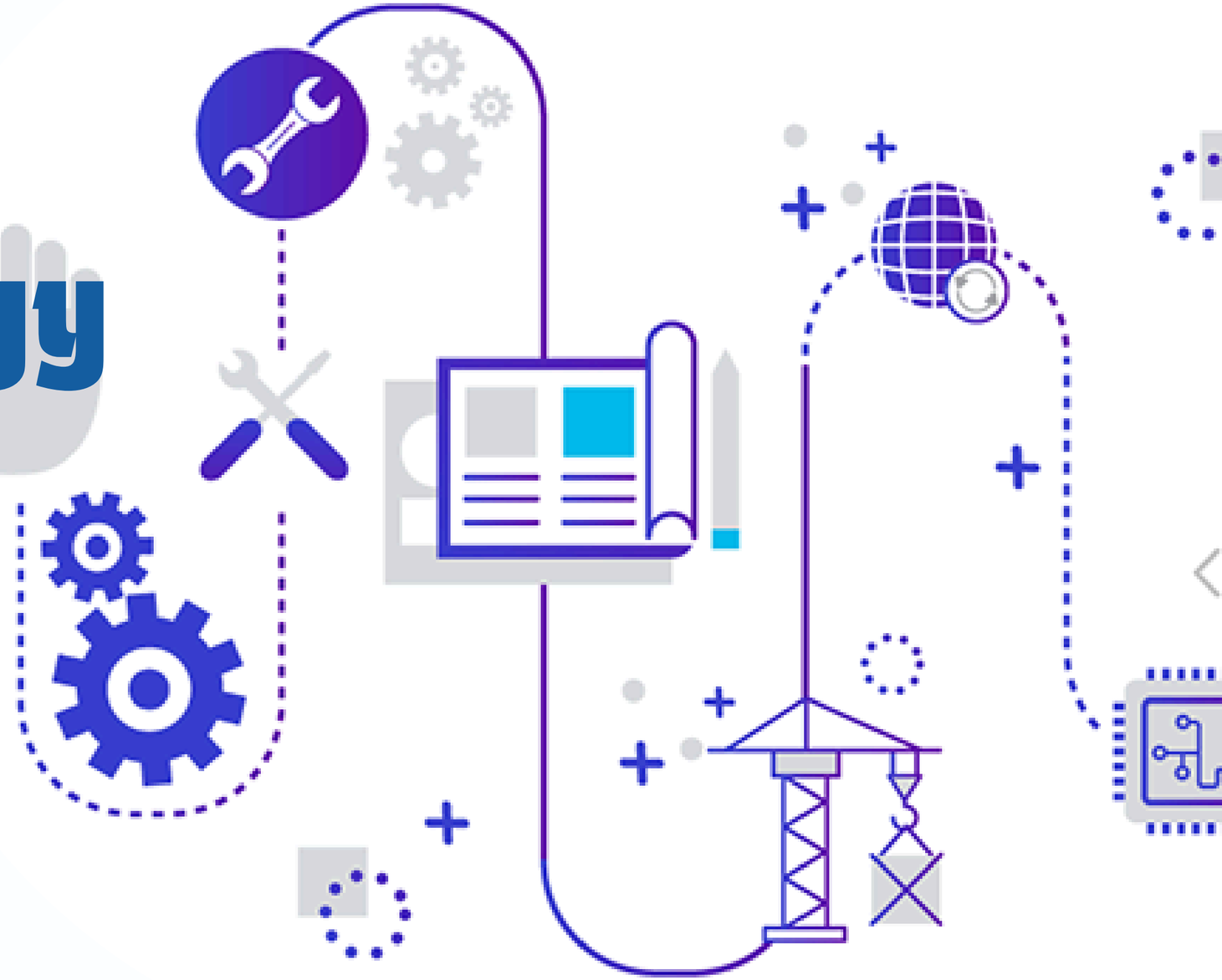
WHY IS IT IMPORTANT TO MONITORS THE PATIENT'S HEALTH PARAMETERS

Continuous monitoring helps identify health issues before they face any health issue and also its ensures better disease control and reduced hospitalizations.

RESEARCH GAP TABLE

System/Reference	Continuous Vital Monitoring	Predictive Health Analysis	Integrated with Wheelchair	Real-time Data Processing	Comprehensive Health Parameters
[1]	❌ No	❌ No	❌ No	❌ No	✅ Yes
[2]	✅ Yes	❌ No	❌ No	❌ No	❌ No
[3]	❌ No	❌ No	❌ No	❌ No	❌ No
[4]	✅ Yes	❌ No	✅ Yes	❌ No	❌ No
[5]	❌ No	❌ No	✅ Yes	❌ No	✅ Yes
PROPOSED SYSTEM	✅ Yes	✅ Yes	✅ Yes	✅ Yes	✅ Yes

Methodology



TECHNOLOGIES & ALGORITHMS

TECHNOLOGIES

- **Python**: The primary programming language used for writing code.
- **TensorFlow**: Utilized for machine learning and deep learning applications.
- **Pandas**: For managing tabular data with DataFrame.
- **NumPy**: For numerical computations and array operations.
- **Matplotlib**: For creating static, detailed plots.
- **Seaborn**: For high-level statistical visualizations.
- **XGBoost**: A library for scalable, efficient gradient boosting (used for model training and evaluation).
- **Scikit-learn**: For data preprocessing, splitting datasets, and evaluating models.
- **FastAPI**: A modern Python web framework for creating APIs quickly and efficiently.
- **Pickle**: For saving and loading trained machine learning models.

TECHNOLOGIES & ALGORITHMS

ALORITHMS

- **Data Generation Algorithms:** Generates synthetic data resembling real-world distributions by sampling values around a specified mean and standard deviation.

- **Machine Learning Algorithms:** Combines multiple decision trees iteratively to improve prediction accuracy and handle complex datasets efficiently.

- **Data Preprocessing Algorithms:** Converts categorical labels into numerical values to make them suitable for machine learning models.

- **Model Evaluation Algorithms:** Calculates the proportion of correct predictions to measure overall model performance and Summarizes prediction results into categories of true and false positives/negatives for error analysis.

THE DATASET

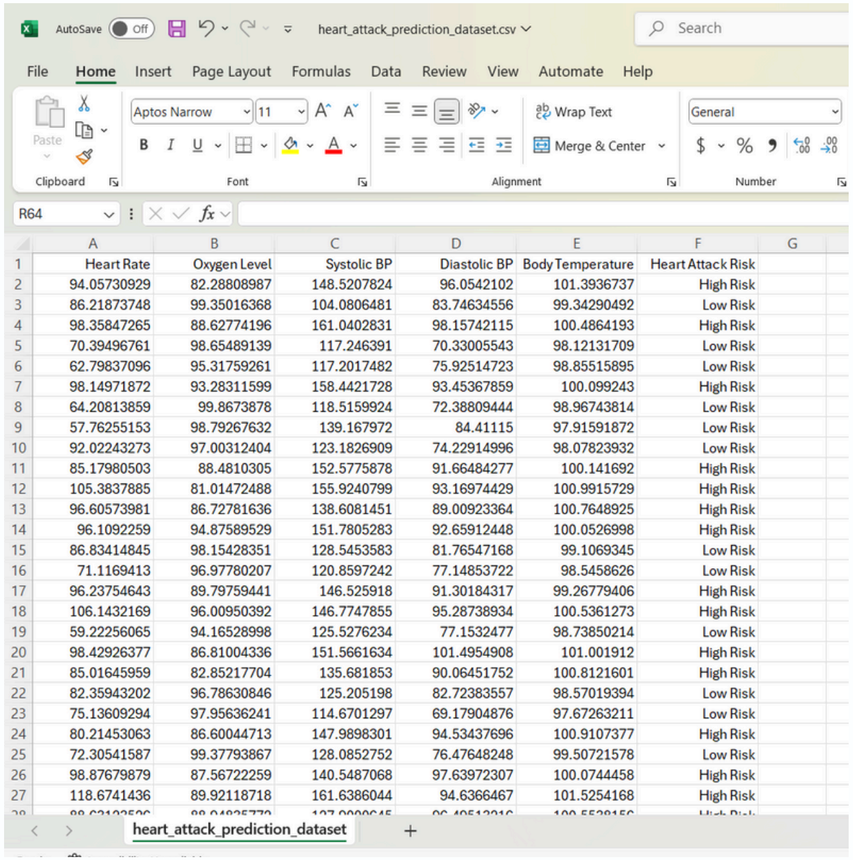
MONITORS THE PATIENT'S VITAL HEALTH PARAMETERS

OXYGEN LEVELS

HEART RATE

BLOOD PRESSURE

BODY TEMPERATURE



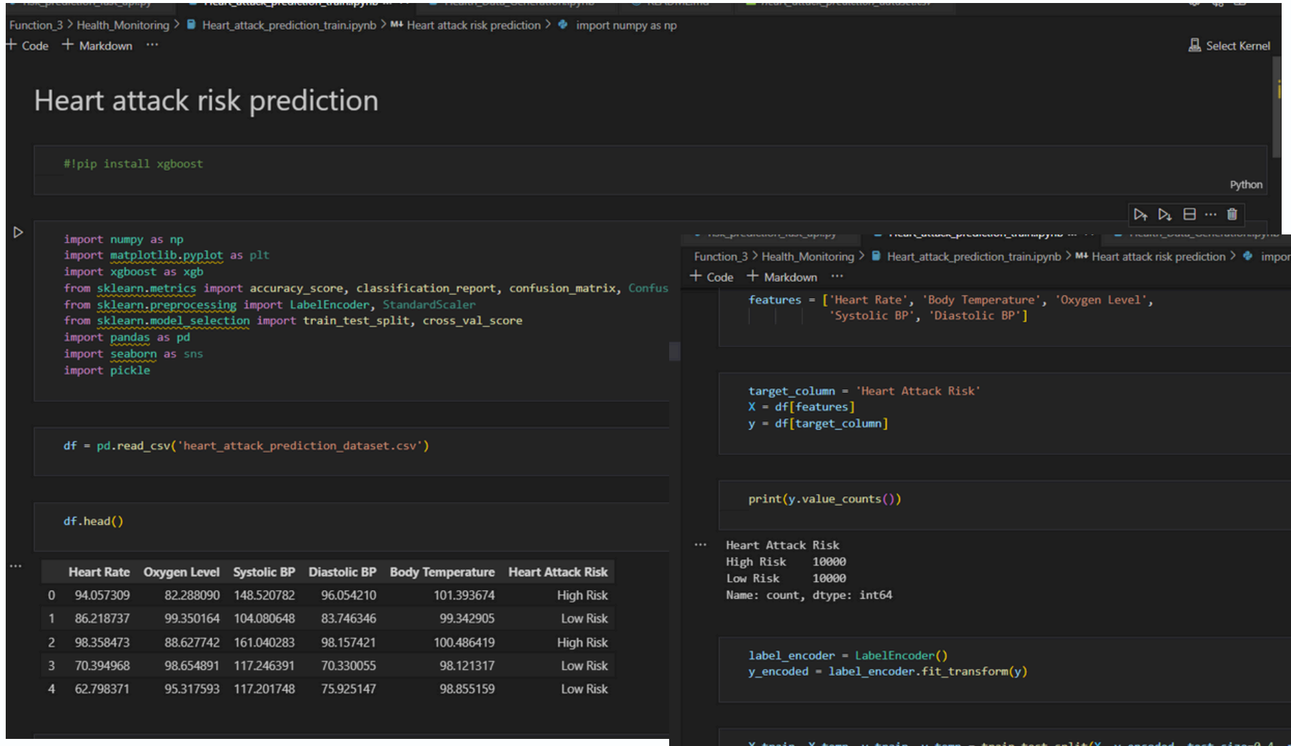
	A	B	C	D	E	F	G
1	Heart Rate	Oxygen Level	Systolic BP	Diastolic BP	Body Temperature	Heart Attack Risk	
2	94.05730929	82.28808987	148.5207824	96.0542102	101.3936737	High Risk	
3	86.21873748	99.35016368	104.0806481	83.74634556	99.34290492	Low Risk	
4	98.35847265	88.62774196	161.0402831	98.15742115	100.4864193	High Risk	
5	70.39496761	98.65489139	117.246391	70.33005543	98.12131709	Low Risk	
6	62.79837096	95.31759261	117.2017482	75.92514723	98.85515895	Low Risk	
7	98.14971872	93.28311599	158.4421728	93.45367859	100.099243	High Risk	
8	64.20813859	99.8673878	118.5159924	72.38809444	98.96743814	Low Risk	
9	57.76255153	98.79267632	139.167972	84.41115	97.91591872	Low Risk	
10	92.02243273	97.00312404	123.1826909	74.22914996	98.07823932	Low Risk	
11	85.17980503	88.4810305	152.5775878	91.66484277	100.141692	High Risk	
12	105.3837885	81.01472488	155.9240799	93.16974429	100.9915729	High Risk	
13	96.60573981	86.72781636	138.6081451	89.00923364	100.7648925	High Risk	
14	96.1092259	94.87589529	151.7805283	92.65912448	100.0526998	High Risk	
15	86.83414845	98.15428351	128.5453583	81.76547168	99.1069345	Low Risk	
16	71.1169413	96.97780207	120.8597242	77.14853722	98.5458626	Low Risk	
17	96.23754643	89.79759441	146.525918	91.30184317	99.26779406	High Risk	
18	106.1432169	96.00950392	146.7747855	95.28738934	100.5361273	High Risk	
19	59.22256065	94.16528998	125.5276234	77.1532477	98.73850214	Low Risk	
20	98.42926377	86.81004336	151.5661634	101.4954908	101.001912	High Risk	
21	85.01645959	82.85217704	135.681853	90.06451752	100.8121601	High Risk	
22	82.35943202	96.78630846	125.205198	82.72383557	98.57019394	Low Risk	
23	75.13609294	97.95636241	114.6701297	69.17904876	97.67263211	Low Risk	
24	80.21453063	86.60044713	147.9898301	94.53437696	100.9107377	High Risk	
25	72.30541587	99.37793867	128.0852752	76.47648248	99.50721578	Low Risk	
26	98.87679879	87.56722259	140.5487068	97.63972307	100.0744458	High Risk	
27	118.6741436	89.92118718	161.6386044	94.6366467	101.5254168	High Risk	

heart_attack_prediction_dataset

THE DATASET

Heart attack prediction dataset

DATASET TRAINING



Function_3 > Health_Monitoring > Heart_attack_prediction_train.ipynb > Heart attack risk prediction > import numpy as np

+ Code + Markdown ··· Select Kernel

Heart attack risk prediction

```
#!pip install xgboost
```

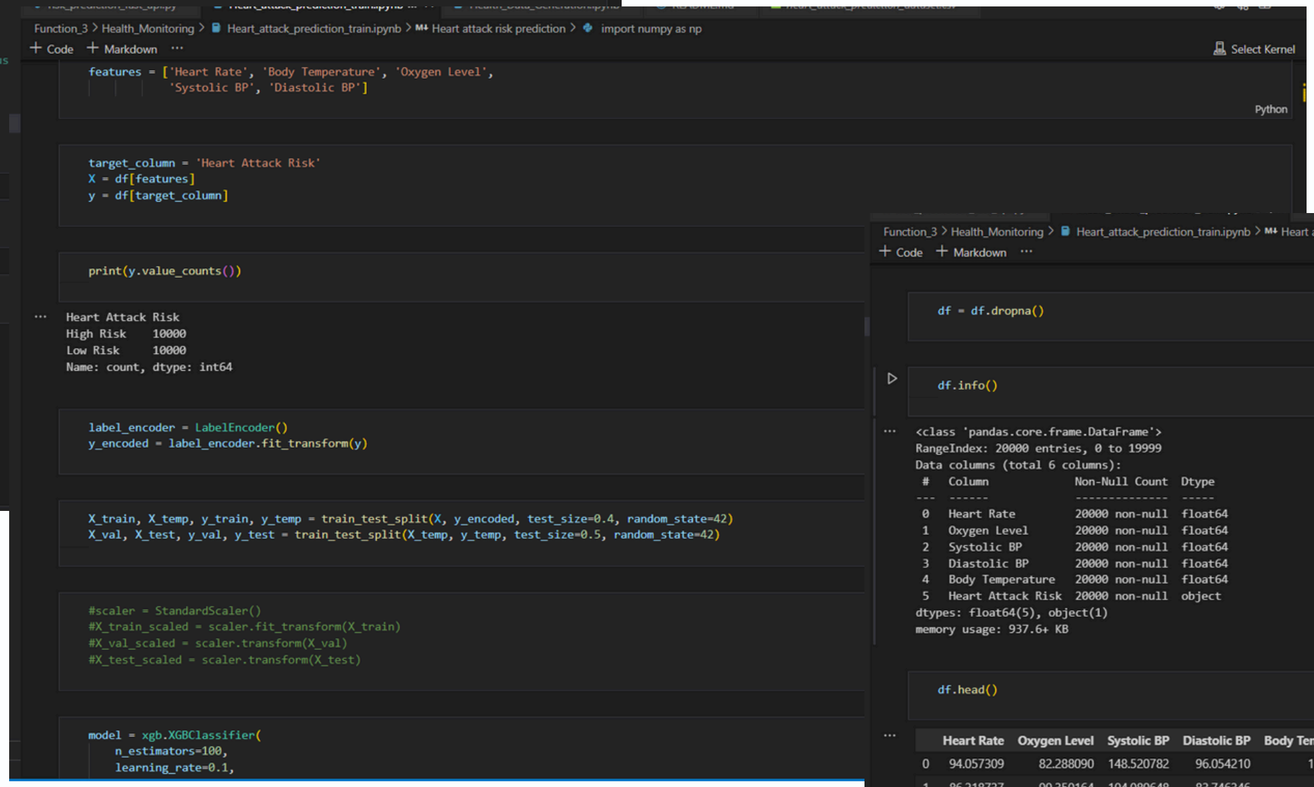
Python

```
import numpy as np
import matplotlib.pyplot as plt
import xgboost as xgb
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, Confus
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split, cross_val_score
import pandas as pd
import seaborn as sns
import pickle
```

```
df = pd.read_csv('heart_attack_prediction_dataset.csv')
```

```
df.head()
```

	Heart Rate	Oxygen Level	Systolic BP	Diastolic BP	Body Temperature	Heart Attack Risk
0	94.057309	82.288090	148.520782	96.054210	101.393674	High Risk
1	86.218737	99.350164	104.080648	83.746346	99.342905	Low Risk
2	98.358473	88.627742	161.040283	98.157421	100.486419	High Risk
3	70.394968	98.654891	117.246391	70.330055	98.121317	Low Risk
4	62.798371	95.317593	117.201748	75.925147	98.855159	Low Risk



Function_3 > Health_Monitoring > Heart_attack_prediction_train.ipynb > Heart attack risk prediction > import numpy as np

+ Code + Markdown ··· Select Kernel

```
features = ['Heart Rate', 'Body Temperature', 'Oxygen Level',
            'Systolic BP', 'Diastolic BP']
```

Python

```
target_column = 'Heart Attack Risk'
X = df[features]
y = df[target_column]
```

```
print(y.value_counts())
```

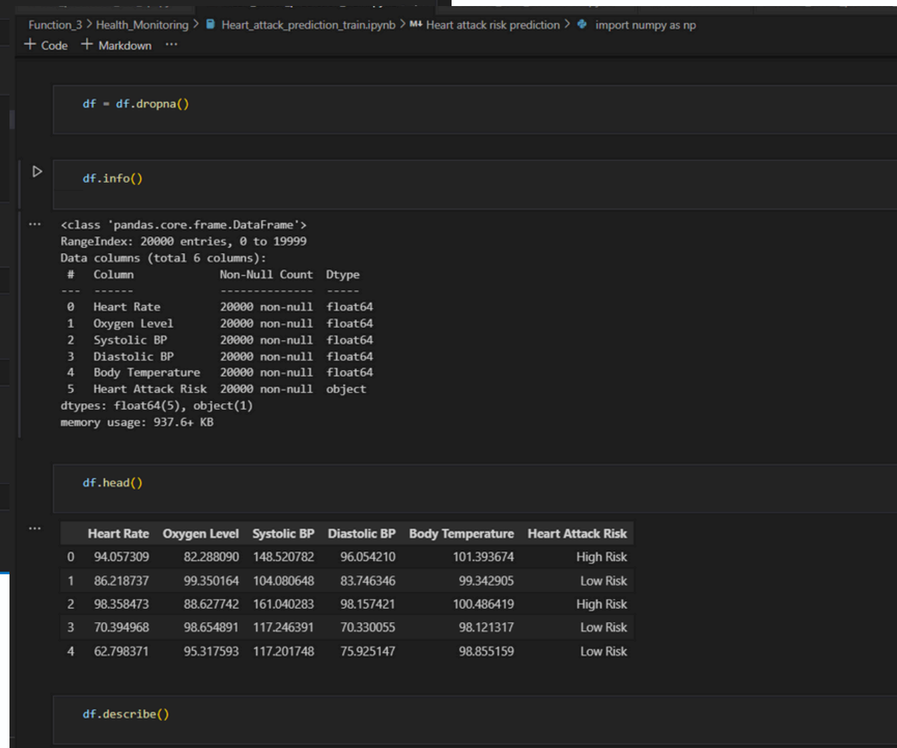
```
Heart Attack Risk
High Risk    10000
Low Risk     10000
Name: count, dtype: int64
```

```
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)
```

```
X_train, X_temp, y_train, y_temp = train_test_split(X, y_encoded, test_size=0.4, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)
```

```
#scaler = StandardScaler()
#X_train_scaled = scaler.fit_transform(X_train)
#X_val_scaled = scaler.transform(X_val)
#X_test_scaled = scaler.transform(X_test)
```

```
model = xgb.XGBClassifier(
    n_estimators=100,
    learning_rate=0.1,
```



Function_3 > Health_Monitoring > Heart_attack_prediction_train.ipynb > Heart attack risk prediction > import numpy as np

+ Code + Markdown ···

```
df = df.dropna()
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20000 entries, 0 to 19999
Data columns (total 6 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Heart Rate        20000 non-null  float64
 1   Oxygen Level      20000 non-null  float64
 2   Systolic BP       20000 non-null  float64
 3   Diastolic BP      20000 non-null  float64
 4   Body Temperature  20000 non-null  float64
 5   Heart Attack Risk 20000 non-null  object
dtypes: float64(5), object(1)
memory usage: 937.6+ KB
```

```
df.head()
```

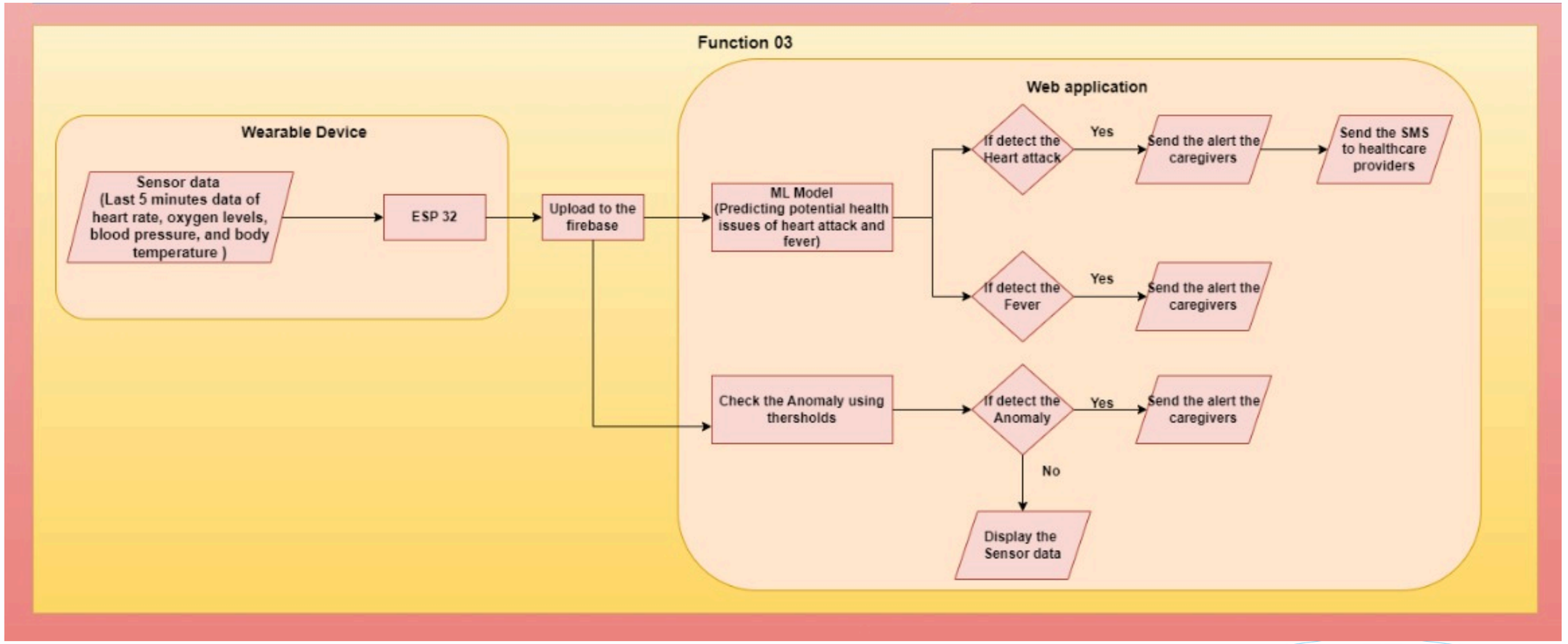
	Heart Rate	Oxygen Level	Systolic BP	Diastolic BP	Body Temperature	Heart Attack Risk
0	94.057309	82.288090	148.520782	96.054210	101.393674	High Risk
1	86.218737	99.350164	104.080648	83.746346	99.342905	Low Risk
2	98.358473	88.627742	161.040283	98.157421	100.486419	High Risk
3	70.394968	98.654891	117.246391	70.330055	98.121317	Low Risk
4	62.798371	95.317593	117.201748	75.925147	98.855159	Low Risk

```
df.describe()
```

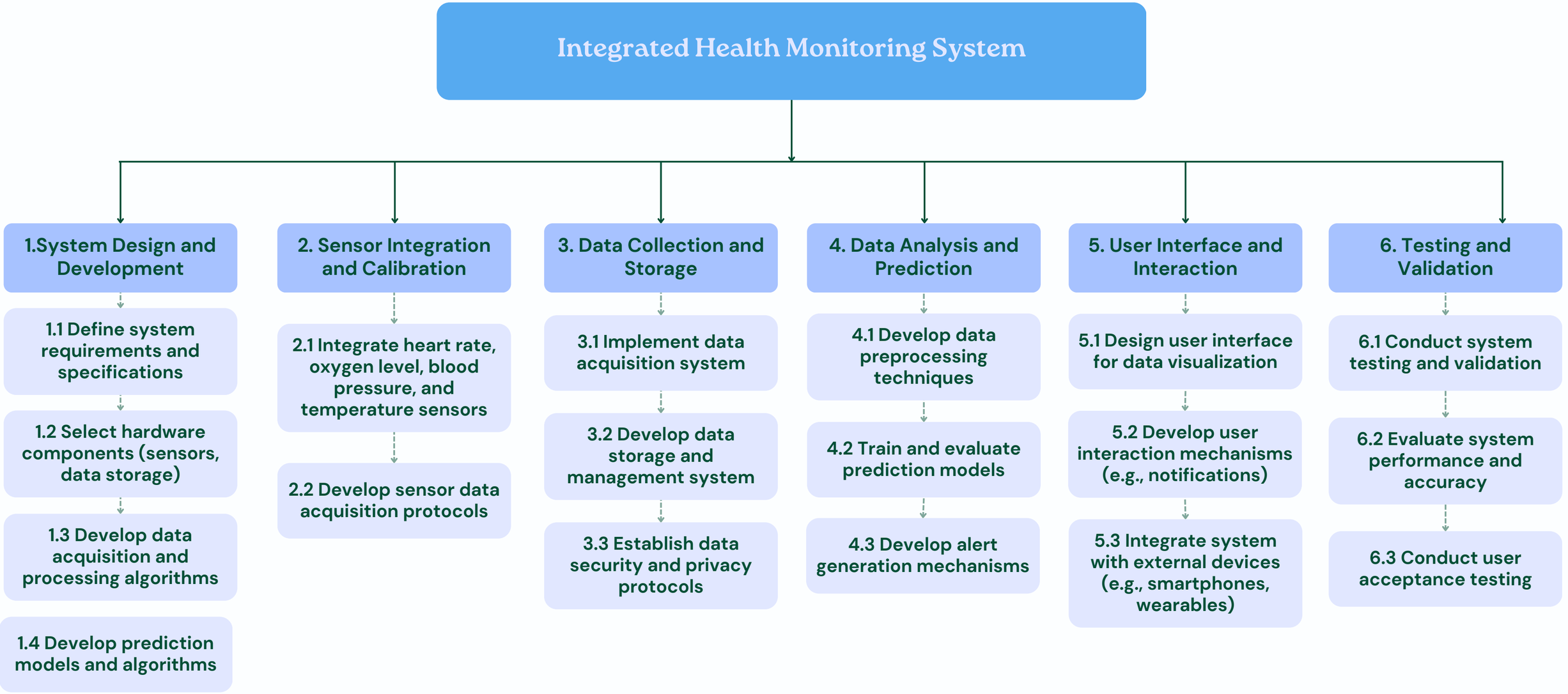
MODEL ACCURACY

```
...   Validation Accuracy: 0.99875

      Validation Confusion Matrix:
       [[1954    2]
       [   3 2041]]

      Validation Classification Report:
                    precision    recall  f1-score   support

                 0       1.00      1.00      1.00      1956
                 1       1.00      1.00      1.00      2044

          accuracy                           1.00      4000
         macro avg       1.00      1.00      1.00      4000
      weighted avg       1.00      1.00      1.00      4000
```

BLOCK DIAGRAM



WORK BREAKDOWN STRUCTURE



FUTURE IMPLEMENTATIONS

- **Improved Prediction Capabilities:** Integration with Real-Time Data Using wearable wrist band to provide continuous health data.

- **Enhanced User Experience:** Build a user-friendly web dashboard to visualize health trends and risk levels over time.

REFERENCE

- [1] D. S. D. S. Dr.Megha Singh, A Novel Electronic Wheel Chair Design using Artificial, 2024 5th International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV), 2024.

- [2] Mr. N.Ram Kumar, HEALTH MONITORING AND ALERTING SMART WHEEL CHAIR FOR PHYSICALLY DISABLED, RAVI KUMAR KANURI,A. PAVAN SADHGUNA,S.AVINASH,P. DIVYA SRI, 2022-2023.

- [3] D. S. P. Shripad Deshpande, Embedded system design for real-time interaction with smart wheelchair, 2016 Symposium on Colossal Data Analysis and Networking (CDAN), 2016.

- [4] F. Q. M. A. N. I. S. S. a. F. S. Zuhra Bano, Innovating an IoT-Integrated Wheelchair for Mobility and Health Monitoring in People with Different Abilities, Institute of Biomedical Engineering and Technology, LUMHS, Jamshoro, Pakistan, Published: June 30, 2023.

S.N. BENTOTAGE

BSc (Hons) in Information Technology

ADAPTIVE PERSONALIZED EXERCISE PLAN AND REAL-TIME MOTION DETECTION

This component integrate a system with **adaptive exercise plans** that are **personalized for each use**r to the wheelchair. First, there is a system that **recommends 5 exercises** for the **initial users** considering some factors like **age**, **gender** and their **injury**. Then, using a camera, we **monitor the exercises** in real time and give real time **feedback**. And also I generate a **progress report** of the exercises. Then based on that progress reports, I **adjust** the exercises for the next month.

RESEARCH PROBLEM

How can real-time monitoring and, adaptive feedback mechanisms be integrated into a wheelchair to improve the accuracy and effectiveness of upper body exercises for individuals with lower body disabilities and hearing impairments?

RESEARCH GAP TABLE

System/Reference	Upper Body Exercise Focus	Real-Time Motion Detection	Adaptive Feedback Mechanisms	Neck Exercise Specialization	Personalized Exercise Plans
Wilroy et al. [1]	✅ Yes	❌ No	❌ No	❌ No	✅ Yes
Sol et al. [2]	✅ Yes	❌ No	❌ No	❌ No	✅ Yes
Hwang and Jeon [3]	❌ No	❌ No	❌ No	❌ No	❌ No
Beomsoo and Doyoung [4]	❌ No	❌ No	❌ No	❌ No	❌ No
Lee et al. [5]	❌ No	❌ No	❌ No	❌ No	❌ No
PROPOSED SYSTEM	✅ Yes	✅ Yes	✅ Yes	✅ Yes	✅ Yes

TECHNOLOGIES, ALGORITHMS

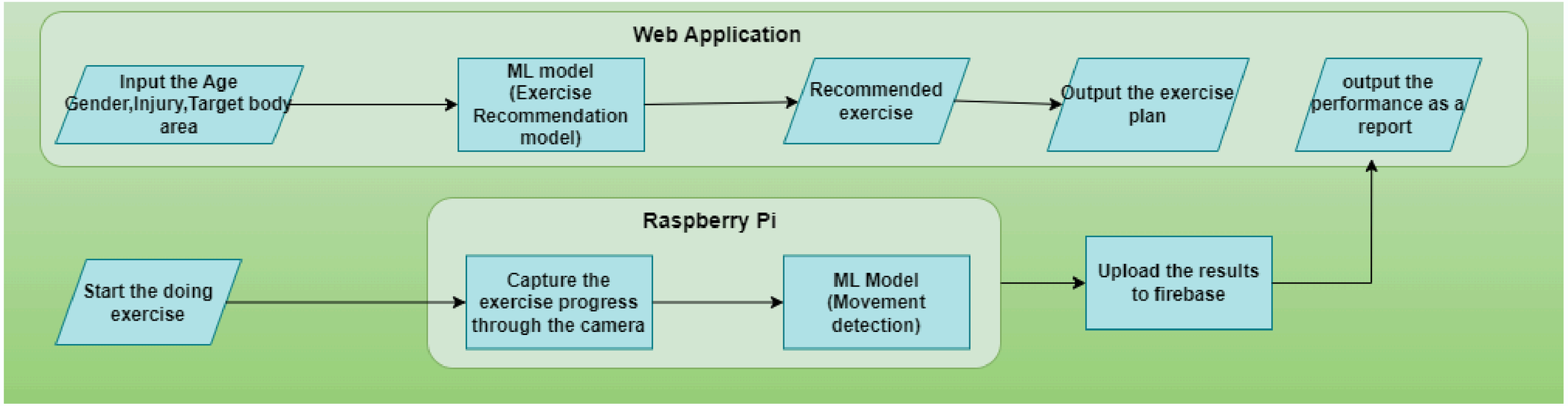
TECHNOLOGIES

- **Python**: Handles the algorithms and data processing.
- **OpenCV**: Framework for **real-time motion detection** and **image processing.**
- **Nearest Neighbors** : find the **closest number of data points** for our input data.
- **CNN**: For **understand** & **interpret** images & videos.
- **Augmentation & Albumentation**: **Increase the size** of the data set
- **FastAPI**: Check the implementation.

ALGORITHMS

- **Personalized Exercise Recommendation**: Tailors exercise routines based on user progress and needs.
- **Adaptive Motion Detection**: Analyzes user movements to detect deviations and provide real-time corrections.
- **Real-Time Feedback**: Algorithms that provide instant guidance and corrections during exercises.
- **Data Analysis and Reporting**: Processes and analyzes performance data to update exercise plans and generate progress reports.
- **Machine Learning Models**: Trained to recognize and analyze correct exercise forms and movements.

BLOCK DIAGRAM



WORK BREAKDOWN STRUCTURE

Adaptive Personalized Exercise Plan and Real-Time Motion Detection

1. Initial Phase
- 1.1 Research on Existing Technologies
- 1.2 Requirement Gathering ✏️
- Feasibility Study

2. Planning Phase
- 2.1 Define Real-time Motion Detection Requirements
- 2.2 Technology Selection (Sensors, AI Algorithms)
- 2.3 Develop a Personalized Exercise Framework
- Design Feedback Mechanism

3. Data Collection Phase
- 3.1 Collect Data on Upper Body Movements 🎯
- 3.2 Gather Information on Neck Exercises
- 3.3 Record User Interactions for Adaptive Feedback
- 3.4 Assemble Dataset for Machine Learning

4. Implementation Phase
- 4.1 Implement Motion Detection Algorithm
- 4.2 Build Adaptive Feedback System 🔍
- 4.3 Integrate Feedback Mechanism with Exercise Framework
- 4.4 Train the System with Collected Data

5. Testing Phase
- 5.1 Test Real-time Feedback Mechanism with Different Users
- 5.2 Validate the Accuracy of Motion Detection
- 5.3 Unit Testing of Adaptive Feedback 📝
- 5.4 System Integration Testing

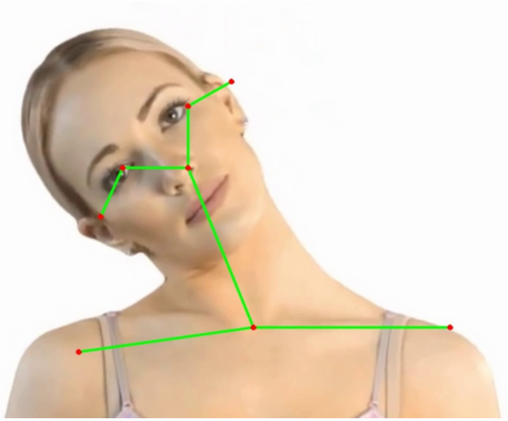
6. Finalization Phase
- 6.1 Optimize the Adaptive Exercise Plan System
- 6.2 Enhance the System Based on Testing Feedback
- 6.3 Prepare Final Reports 📝
- 6.4 Integration into Smart Wheelchair System

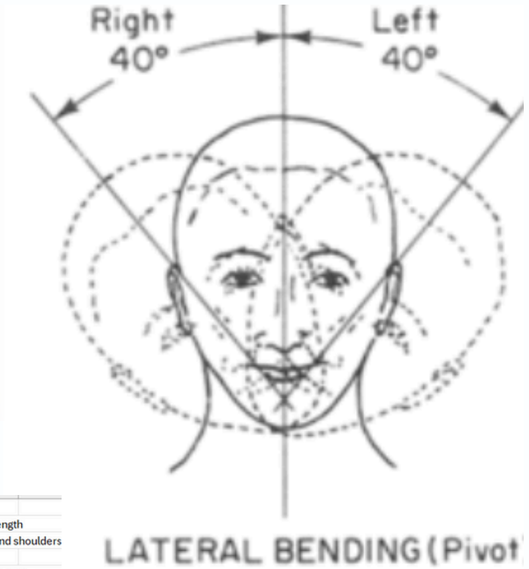
DATASETS

NECK EXERCISES

FRAMES TO DETECT THE EXERCISE MOVEMENTS

DATA ON EXERCISE EFFECTIVENESS



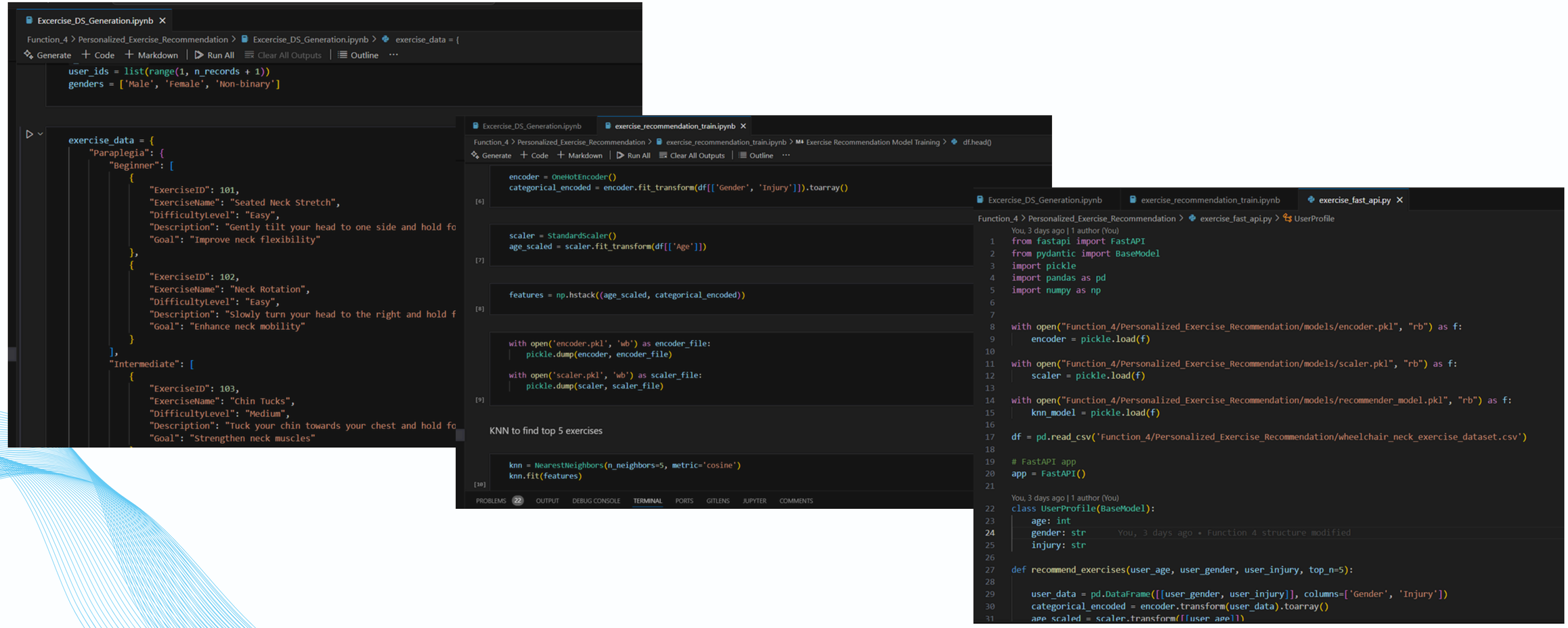


UserID	Age	Gender	Injury	Fitness Le	ExerciseID	Exercise N	Difficulty L	Descriptio	Goal
1	26	Female	Quadriples	Intermedia	106	Neck Isom	Medium	Press your	Increase neck strength
2	82	Male	Paraplegia	Advanced	108	Resistance	Hard	Use a resis	Strengthen neck and shoulders
3	32	Male	Multiple Sc	Advanced	115	Gentle Nec	Easy	Use your fi	Relieve tension
4	23	Male	Multiple Sc	Beginner	113	Seated Ne	Easy	Tuck your c	Strengthen neck stability
5	70	Non-binar	Spinal Cor	Beginner	107	Neck Flexi	Easy	Gently dro	Improve neck flexibility
6	28	Non-binar	Amputatio	Intermedia	112	Neck Stret	Medium	Use a towe	Improve flexibility
7	66	Female	Amputatio	Intermedia	112	Neck Stret	Medium	Use a towe	Improve flexibility
8	28	Female	Amputatio	Intermedia	112	Neck Stret	Medium	Use a towe	Improve flexibility
9	40	Female	Multiple Sc	Advanced	115	Gentle Nec	Easy	Use your fi	Relieve tension
10	62	Non-binar	Spinal Cor	Beginner	107	Neck Flexi	Easy	Gently dro	Improve neck flexibility
11	46	Non-binar	Amputatio	Beginner	110	Neck Exter	Easy	Look up to	Improve posture
12	48	Female	Amputatio	Beginner	110	Neck Exter	Easy	Look up to	Improve posture
13	87	Male	Quadriples	Beginner	105	Seated Sid	Easy	Place your	Relieve neck tension
14	38	Female	Spinal Cor	Beginner	107	Neck Flexi	Easy	Gently dro	Improve neck flexibility
15	83	Female	Paraplegia	Beginner	101	Seated Ne	Easy	Gently tilt y	Improve neck flexibility
16	79	Female	Spinal Cor	Intermedia	111	Seated He	Medium	Turn your h	Enhance neck mobility
17	70	Female	Quadriples	Beginner	105	Seated Sid	Easy	Place your	Relieve neck tension
18	20	Non-binar	Amputatio	Beginner	110	Neck Exter	Easy	Look up to	Improve posture
19	26	Male	Spinal Cor	Beginner	107	Neck Flexi	Easy	Gently dro	Improve neck flexibility
20	39	Non-binar	Multiple Sc	Intermedia	114	Seated Sho	Medium	Squeeze yo	Improve upper back strength
21	64	Male	Paraplegia	Intermedia	103	Chin Tucks	Medium	Tuck your c	Strengthen neck muscles
22	33	Female	Multiple Sc	Intermedia	114	Seated Sho	Medium	Squeeze yo	Improve upper back strength
23	70	Female	Quadriples	Intermedia	106	Neck Isom	Medium	Press your	Increase neck strength
24	48	Female	Paraplegia	Intermedia	103	Chin Tucks	Medium	Tuck your c	Strengthen neck muscles
25	22	Female	Multiple Sc	Beginner	113	Seated Ne	Easy	Tuck your c	Strengthen neck stability

wheelchair_neck_exercise_datase

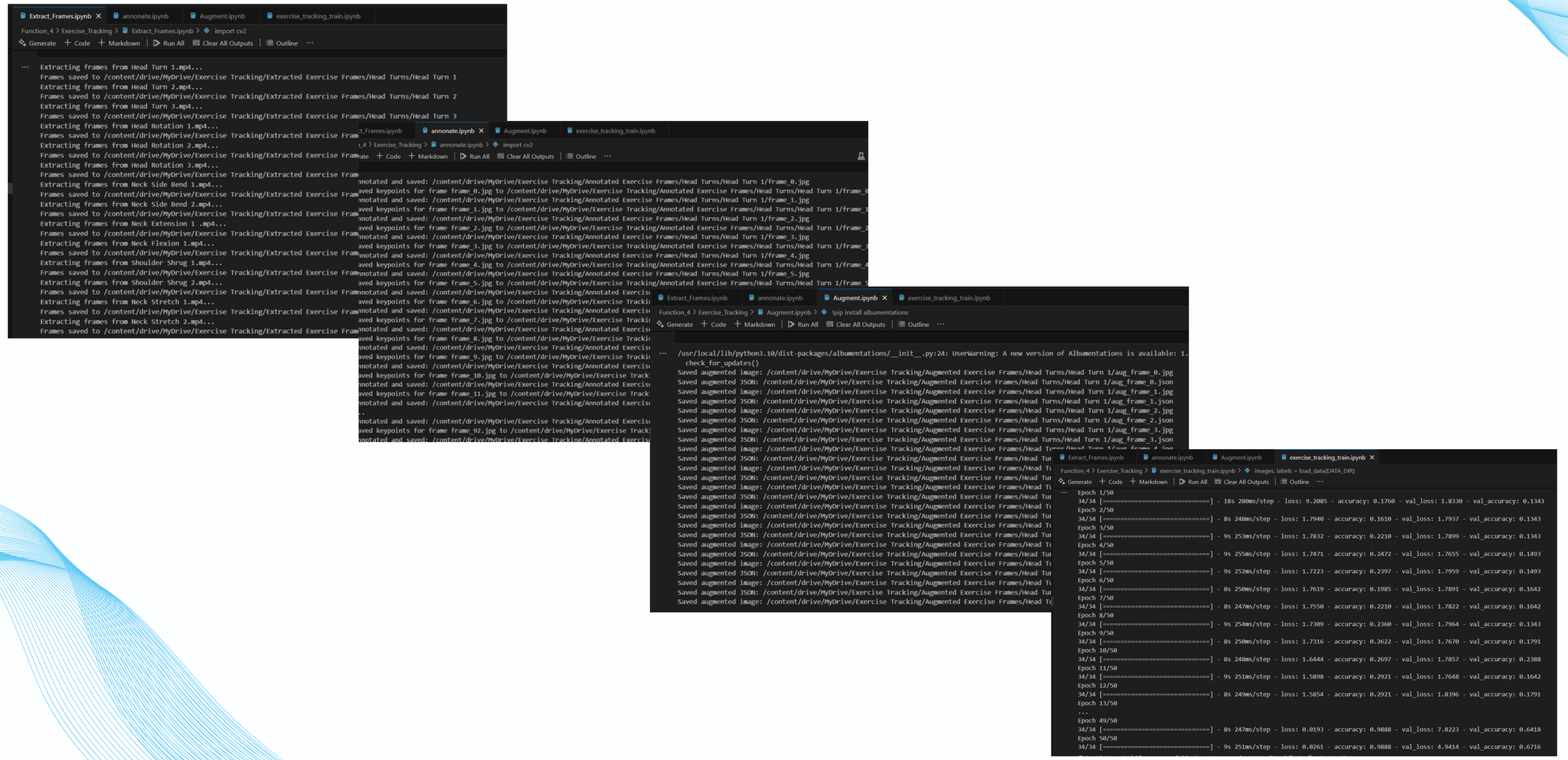
ALREADY IMPLEMENTED

Personalized Exercise Recommendation (with nearest neighbors)



ALREADY IMPLEMENTED

Exercise Tracking (using openPose and CNN)



MODEL ACCURACY

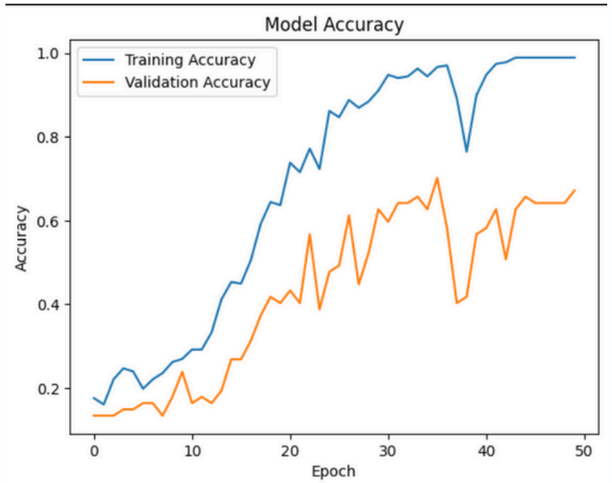
Exercise Tracking

Extract_Frames.ipynb | annonate.ipynb | Augment.ipynb | exercise_tracking_train.ipynb

Function_4 > Exercise_Tracking > exercise_tracking_train.ipynb > images, labels = load_data(DATA_DIR)

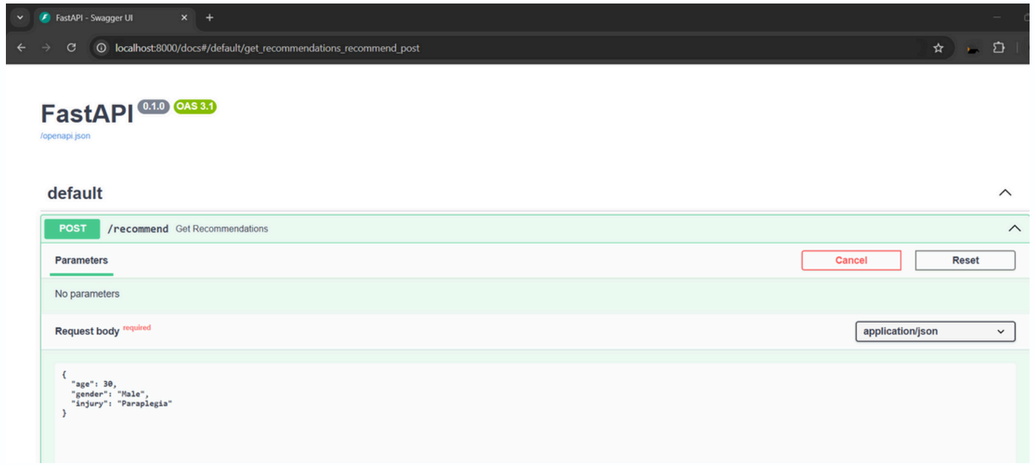
Generate | Code | Markdown | Run All | Clear All Outputs | Outline | ...

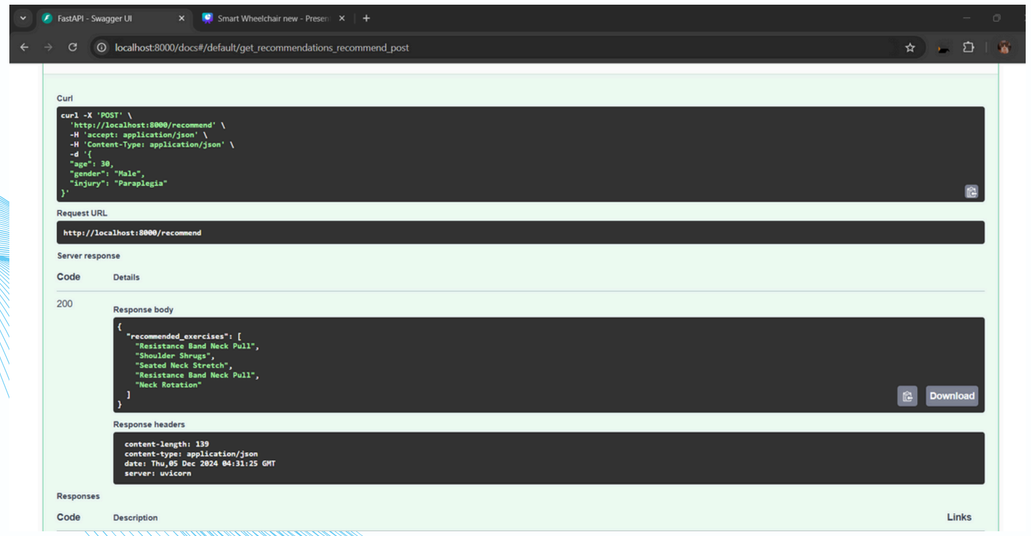
```
Epoch 1/50
34/34 [==============================] - 18s 280ms/step - loss: 9.2085 - accuracy: 0.1760 - val_loss: 1.8330 - val_accuracy: 0.1343
Epoch 2/50
34/34 [==============================] - 8s 248ms/step - loss: 1.7940 - accuracy: 0.1610 - val_loss: 1.7937 - val_accuracy: 0.1343
Epoch 3/50
34/34 [==============================] - 9s 253ms/step - loss: 1.7832 - accuracy: 0.2210 - val_loss: 1.7899 - val_accuracy: 0.1343
Epoch 4/50
34/34 [==============================] - 9s 255ms/step - loss: 1.7471 - accuracy: 0.2472 - val_loss: 1.7655 - val_accuracy: 0.1493
Epoch 5/50
34/34 [==============================] - 9s 252ms/step - loss: 1.7223 - accuracy: 0.2397 - val_loss: 1.7959 - val_accuracy: 0.1493
Epoch 6/50
34/34 [==============================] - 8s 250ms/step - loss: 1.7619 - accuracy: 0.1985 - val_loss: 1.7891 - val_accuracy: 0.1642
Epoch 7/50
34/34 [==============================] - 8s 247ms/step - loss: 1.7550 - accuracy: 0.2210 - val_loss: 1.7822 - val_accuracy: 0.1642
Epoch 8/50
34/34 [==============================] - 9s 254ms/step - loss: 1.7309 - accuracy: 0.2360 - val_loss: 1.7964 - val_accuracy: 0.1343
Epoch 9/50
34/34 [==============================] - 8s 250ms/step - loss: 1.7316 - accuracy: 0.2622 - val_loss: 1.7670 - val_accuracy: 0.1791
Epoch 10/50
34/34 [==============================] - 8s 248ms/step - loss: 1.6444 - accuracy: 0.2697 - val_loss: 1.7857 - val_accuracy: 0.2388
Epoch 11/50
34/34 [==============================] - 9s 251ms/step - loss: 1.5898 - accuracy: 0.2921 - val_loss: 1.7648 - val_accuracy: 0.1642
Epoch 12/50
34/34 [==============================] - 8s 249ms/step - loss: 1.5854 - accuracy: 0.2921 - val_loss: 1.8396 - val_accuracy: 0.1791
Epoch 13/50
...
Epoch 49/50
34/34 [==============================] - 8s 247ms/step - loss: 0.0193 - accuracy: 0.9888 - val_loss: 7.8223 - val_accuracy: 0.6418
Epoch 50/50
34/34 [==============================] - 9s 251ms/step - loss: 0.0261 - accuracy: 0.9888 - val_loss: 4.9414 - val_accuracy: 0.6716
```

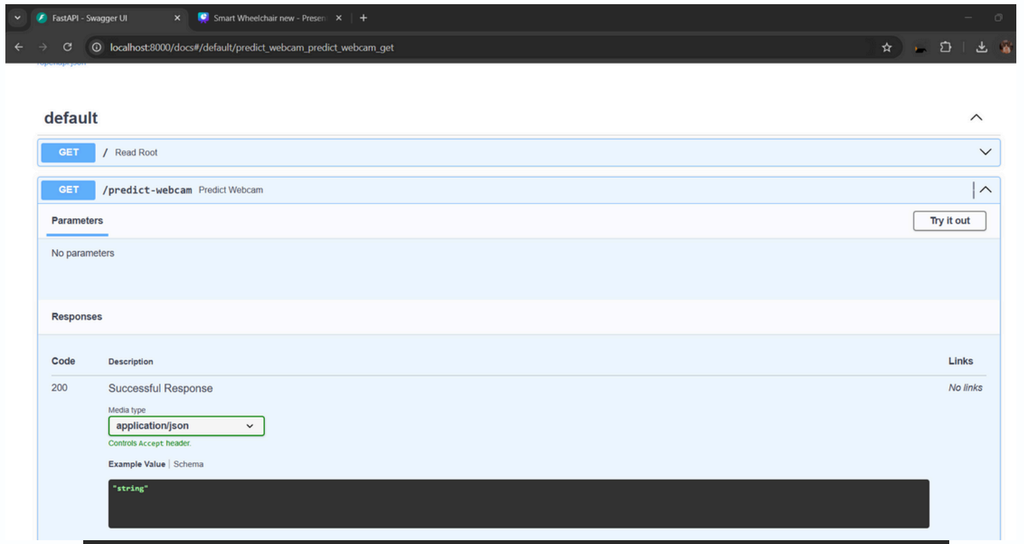


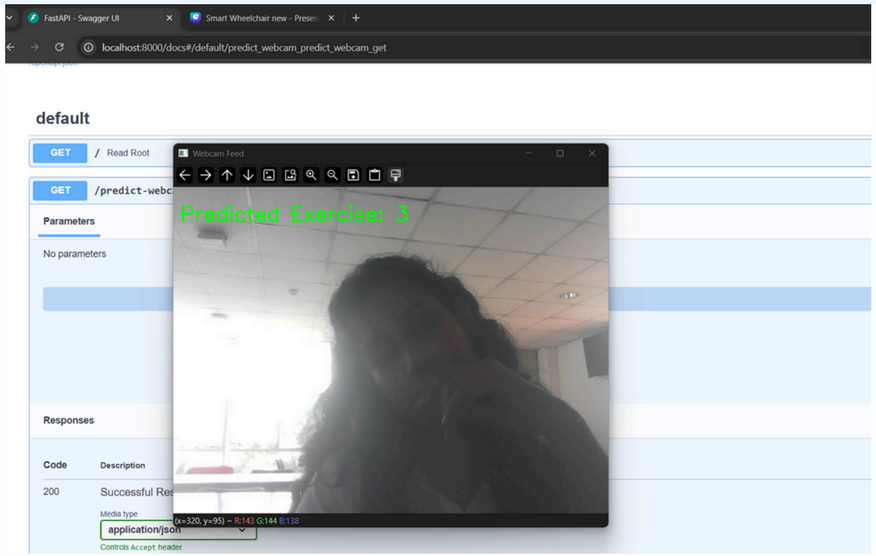
ALREADY IMPLEMENTED

API s









FUTURE IMPLEMENTATIONS

- **Improving the accuracy of the Exercise Tracking Model** (hope to change the data set)
- **Method of tracking the accuracy of the exercises**
- **Implement the logic for real – time feedback process**
- **Generate progress reports for the exercises**
- **Adjust the exercises according to the progress reports**
- **Integration with the wheelchair and web application**
- **Allow manual access to the care taker to alter the exercises**

REFERENCE

- [1] Wilroy, J. D., et al. (2020). "Correlates of adherence in a home-based, self-managed exercise program tailored to wheelchair users with spinal cord injury," Spinal Cord, vol. 59, pp. 55-62.

- [2] Sol, M. E., et al. (2021). "The effects of wheelchair mobility skills and exercise training on physical activity, fitness, skills and confidence in youth using a manual wheelchair," Disability and Rehabilitation, vol. 44, no. 16, pp. 4398-4407.

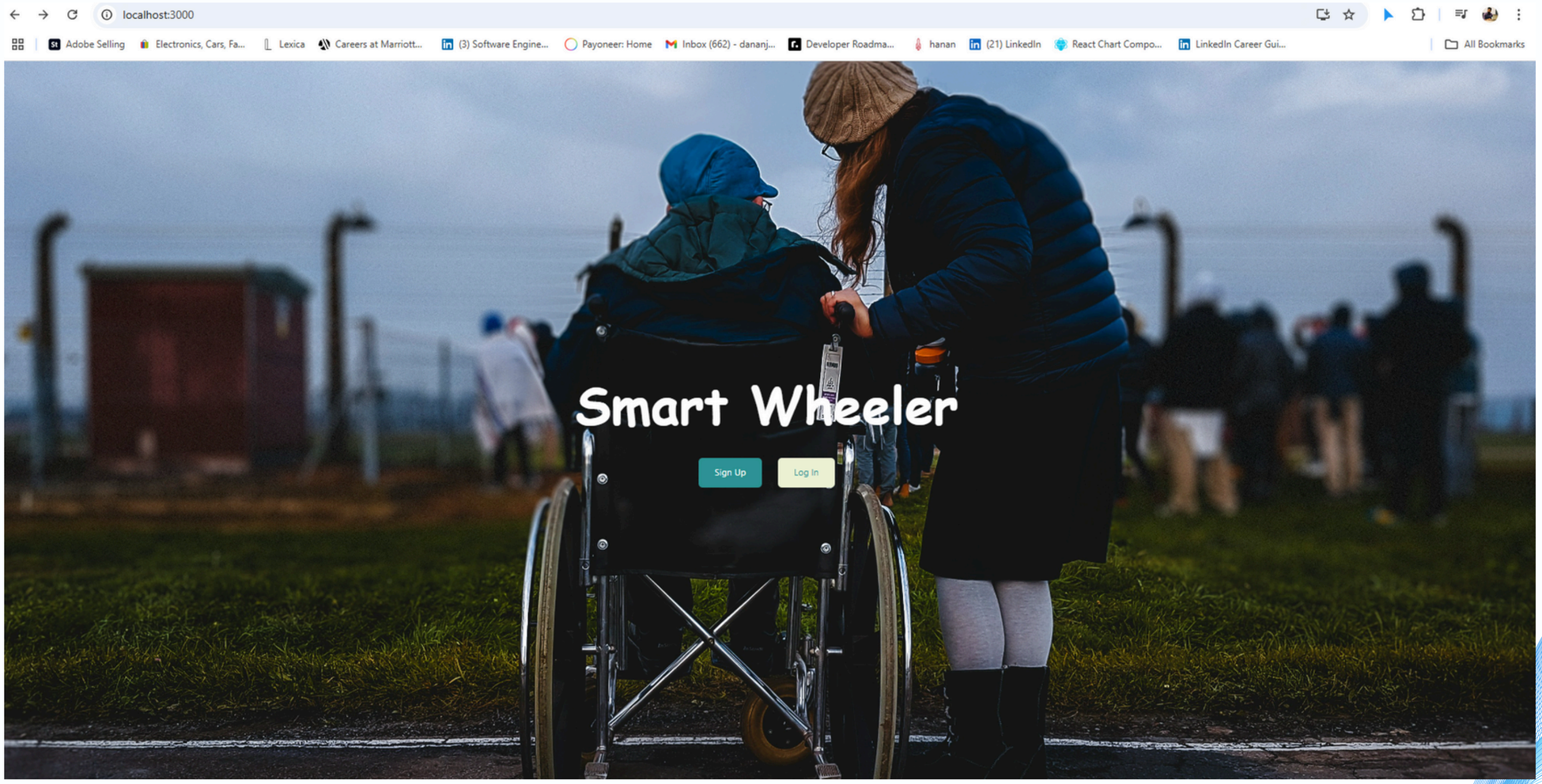
- [3] Hwang, B., and Jeon, D. (2012). "A wheelchair integrated lower limb exercise/rehabilitation system: Design and experimental results on the knee joint," IEEE/SICE International Symposium on System Integration (SII), pp. 164-169.

- [4] Lee, D., et al. (2013). "Development and preliminary testing of a novel wheelchair integrated exercise/rehabilitation system," IEEE International Conference on Rehabilitation Robotics (ICORR), pp. 1-6.

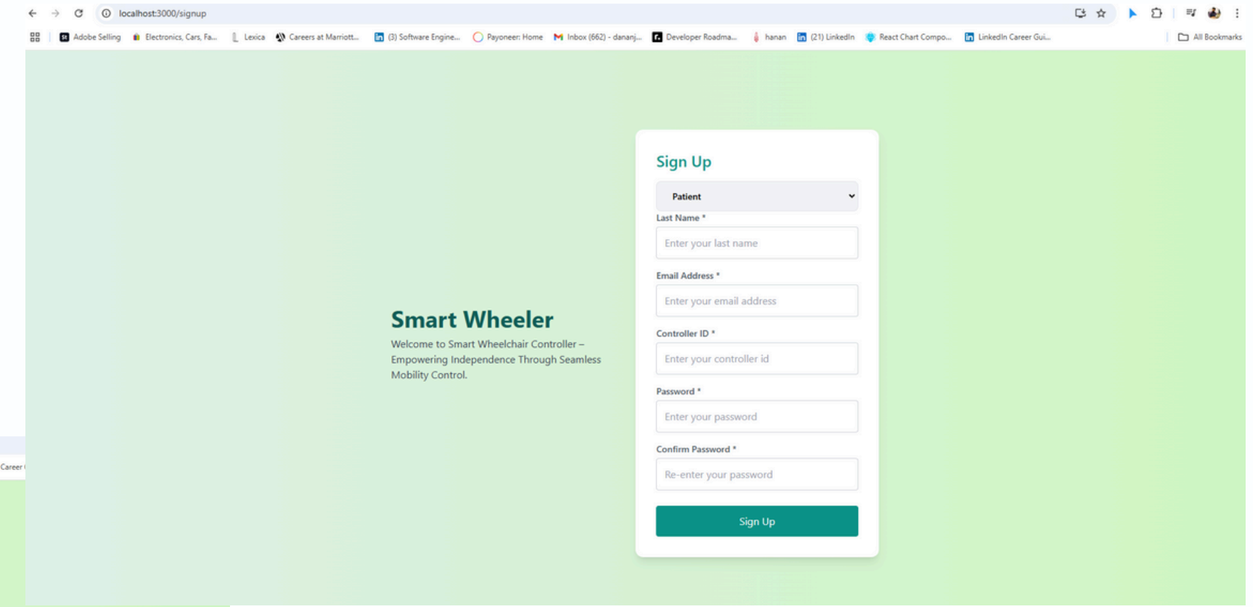
REFERENCE

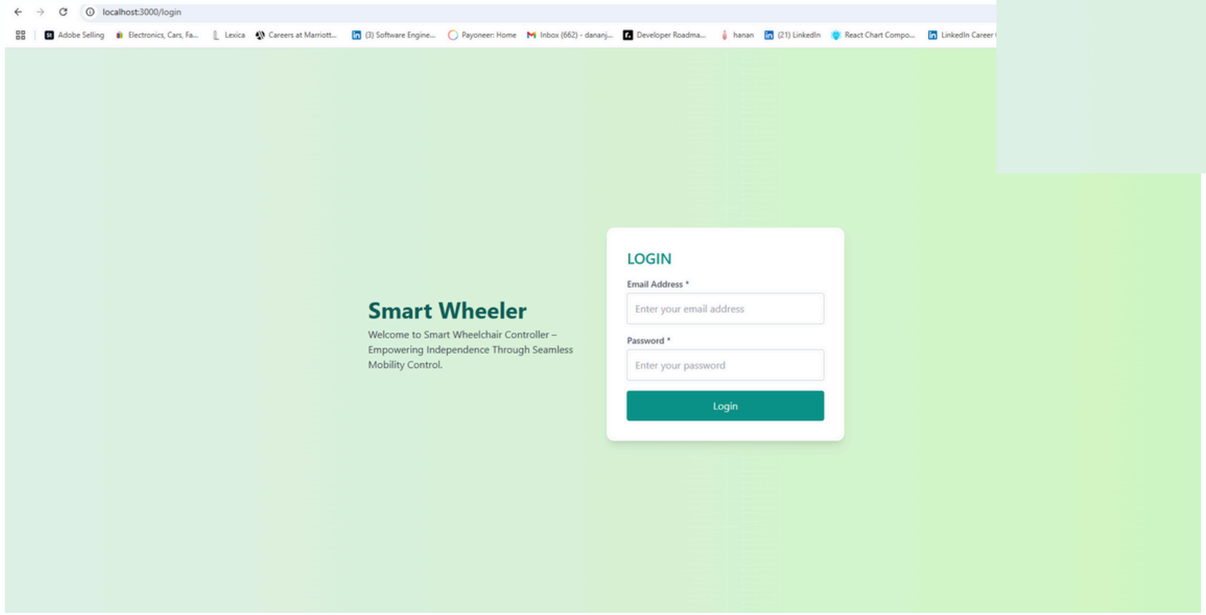
- https://youtu.be/JboZfxUjLSk?si=F6W51rMr7n8Tk135
- https://www.youtube.com/watch?v=Xhl_S_OZYEo
- https://www.youtube.com/watch?v=CxcKPnZCJ3k
- https://youtu.be/9jQGsUidKHs?si=fb_M1gs3oEeHOGlr
- https://youtu.be/0p0o5cmgLdE?si=cPreJOIyiJ36DgM0

WEB APPLICATION(CONTROL INTERFACE)

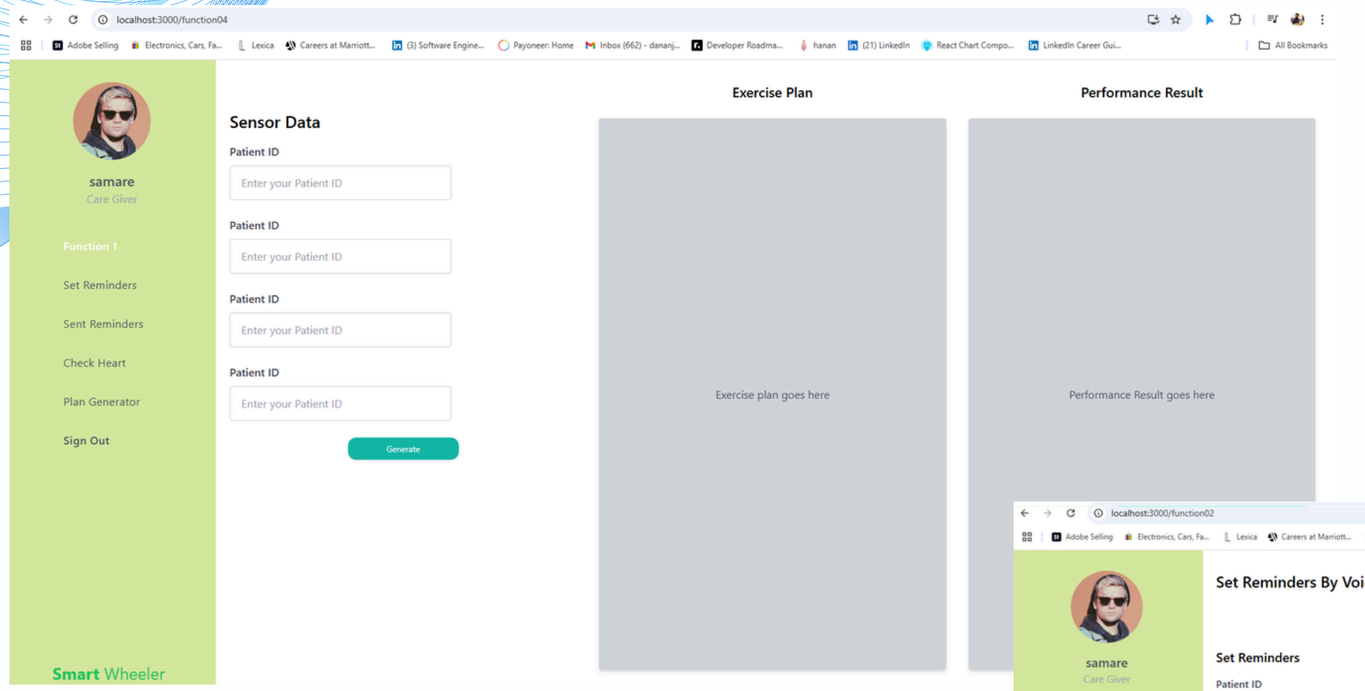


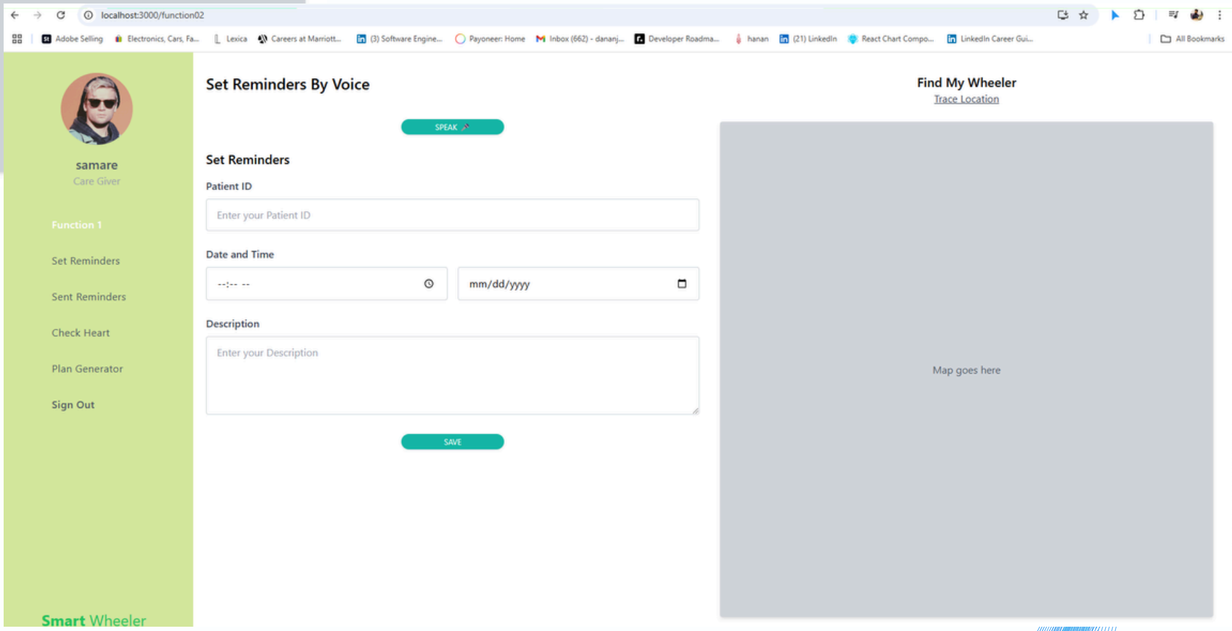
WEB APPLICATION(CONTROL INTERFACE)





WEB APPLICATION(CONTROL INTERFACE)





THANK YOU