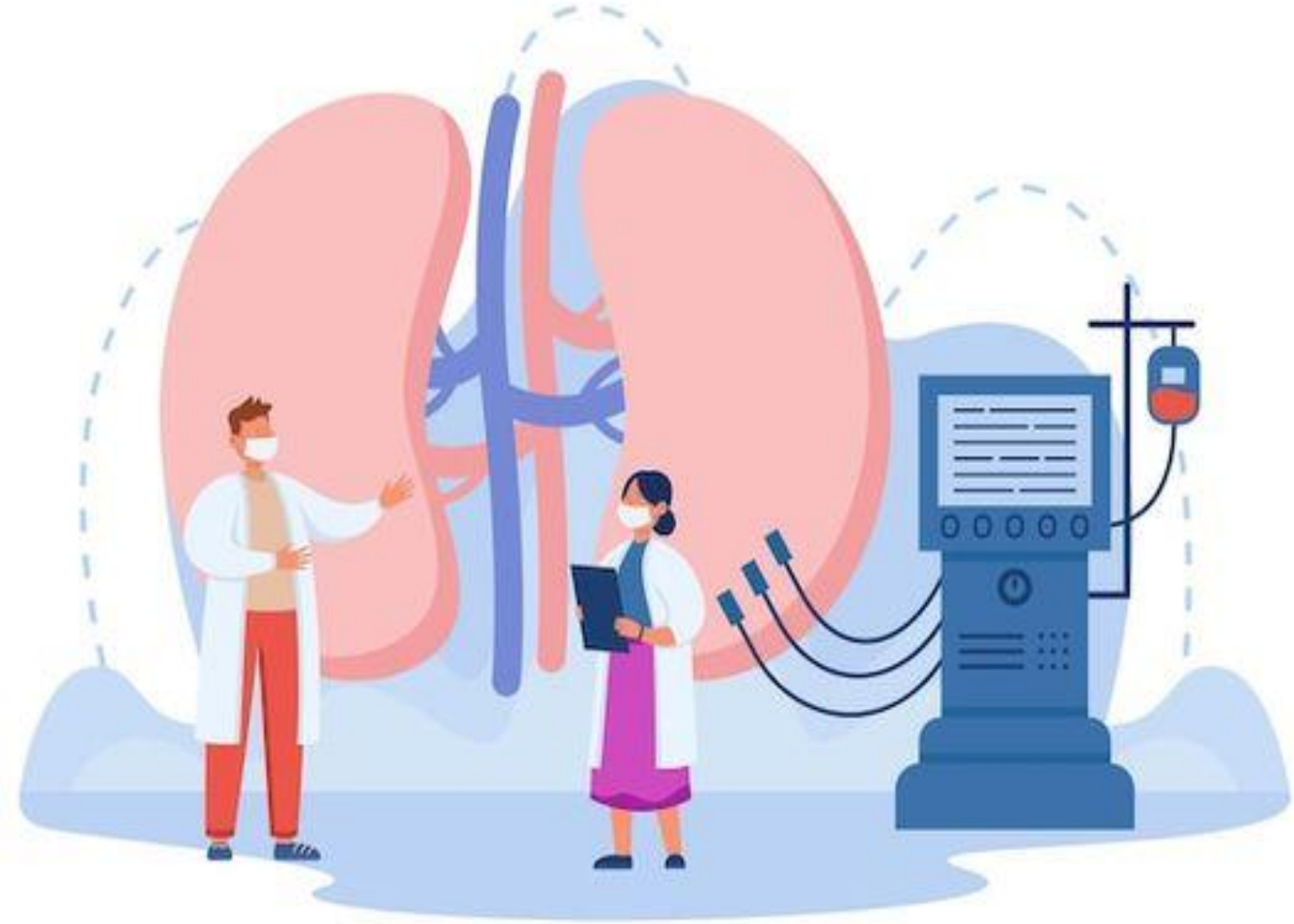
KidniFy

Your Kidney Care Companion

23-032



Outline

- ❑ **Our Team**
- ❑ **Introduction**
  - Background
  - Research Problem
  - Overall Problem
  - System Overview
  - Gantt Chart
  - Commercialization
- ❑ **Individual Components**
  - Background
  - Research Problem
  - Research Gap
  - Objectives
  - Methodology

Our Team

MS. WISHALYA TISSERA
SUPERVISOR

MR. SAMADHI RATHNAYAKE
CO-SUPERVISOR

DR. RANJAN MALLAWAARACHCHI
DOMAIN EXPERT

IT20154226
M.M.K.L MARASINGHE

IT20235260
D.R.N. SAMARAWILA

IT20226596
J.P.M.L. PERERA

IT20785192
W.B.M.A. ISURIKA

Research Problem

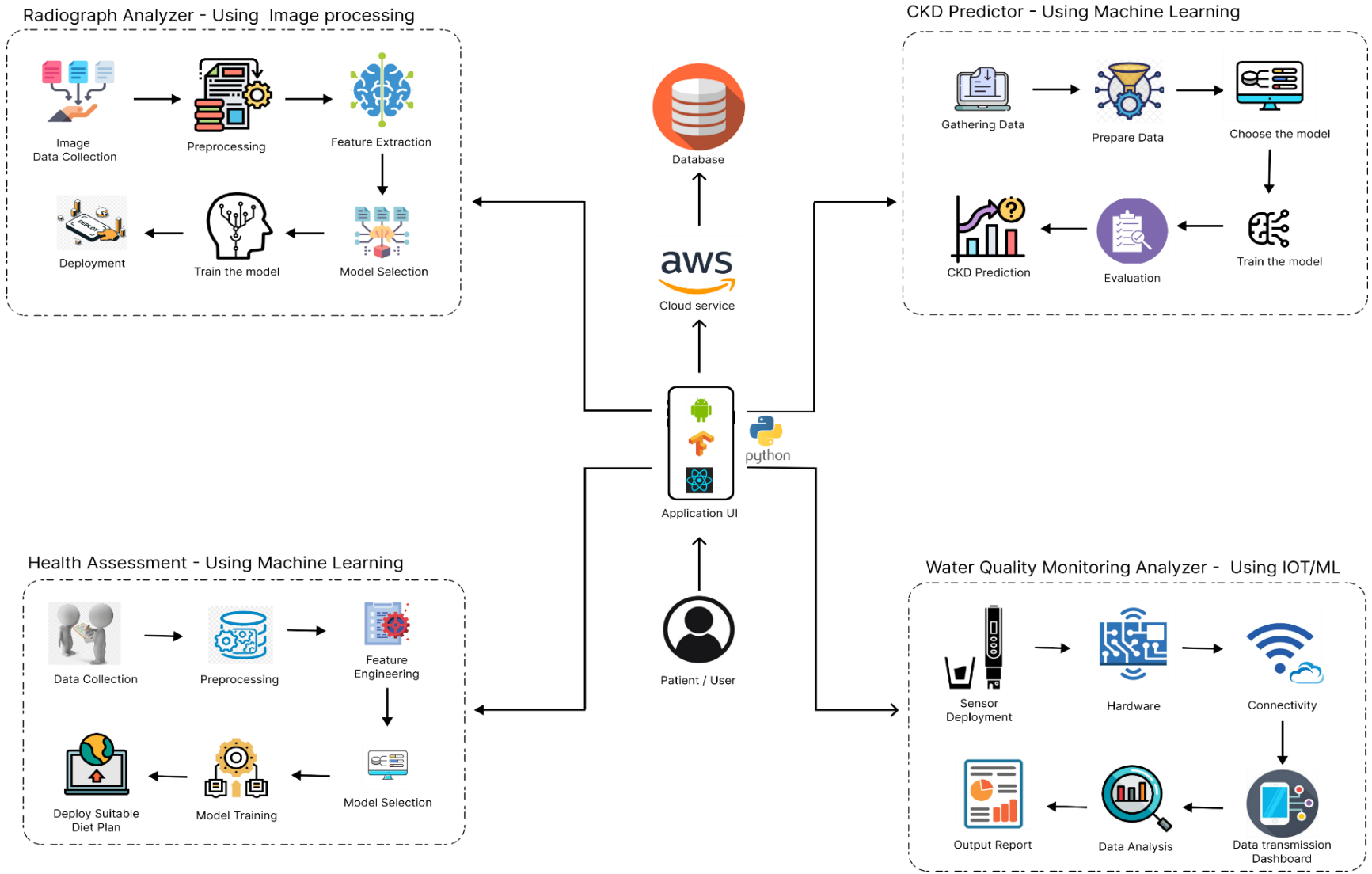
- How to develop an accurate ML model for kidney disease prediction?
- How to use image processing to analyze CKD patients' medical images?
- What are the efficient ways to test the water quality that affects kidney patients, combining IoT as well as ML?
- How to develop a personalized dietary prediction system for CKD patients?

Objectives

To develop a mobile app that is easy to use yet effective to

- Accurately predict about kidney health based on real time data.
- Get timely diagnosis of kidney disease using image processing
- Measure the quality of water with the use of an IOT device
- Analyse patient data and provide personalized diet plans

System Overview



Individual Components

IT20235260 | D.R.N. Samarawila

Specializing in Information Technology

Predictive analysis for risks of having a Kidney Disease

Background

- CKD is a global problem, identified as the 7th most common course of death in the world.
- In Sri Lanka prevalence of CKD also has been increased alarmingly in recent decades.
- Kidney diseases are major health concern, impacting a considerable amount of the population, affecting approximately 18% of the population and it is considerable value from a population.
- In addition to these main types of CKD, there are also other less common causes of kidney disease, such as kidney stones, tumors, and structural abnormalities.
- Identify a CKD in early stages and the importance of that.
- Use the modern technology to aware and to prevent from the CKD.

Research Problem

Detecting and treating kidney disease early is essential for preventing complications and improving patient outcomes.

1. How accurate are existing ML-based models for kidney disease prediction in the Sri Lankan population?

2. What factors may influence the performance of these models, such as age, gender, socioeconomic status, and geographic location?

3. What is the level of awareness among the general public in Sri Lanka about the risks and symptoms of kidney disease, and how can this be improved to facilitate early detection and treatment?

4. What are the current barriers to accessing timely and accurate diagnostic tools for kidney disease in Sri Lanka, and how can these be addressed?

- Despite the importance of the early detection and treatment there are significant challenges in identifying individuals at risk of kidney disease in Sri Lanka.

- One major challenge is the lack of access to timely and accurate diagnostic tools, particularly in rural areas where healthcare resource are often limited.

Research Gap

- This research gap highlights the need for a comprehensive solution that is accurate, easily accessible, and localized.
- Will increase the availability of kidney disease prediction tools for patients in Sri Lanka.
- Aims to address this research gap by providing an accurate prediction of a user's kidney health status using multiple machine learning models.
- Based on the patient's location, the application will display a list of the nearest medical facilities where they can go through the necessary tests.

Objectives

Main Objective

- Develop an effective solution for predicting a user's kidney health condition with high accuracy.

Sub Objectives

- Gathering the data and preprocessed that data.
- Train highly accurate ML model with collected data.
- Giving the prediction according to the real time test data.
- Giving the instructions and the recommend treatments
- Based on patient’s health condition and the current location, display a list of the nearest medical facilities where they can go through the necessary tests.

Completion of the Component

- Requirements Analysis
  - Collect data manually and create data set

Attributes

1. ID
2. Age
3. Gender
4. Diabetic
5. Family History
6. Obesity
7. Smoking
8. Alcohol

9.Prolong use of medications
10.Urinary Obstructions
11.Edema Symptoms
12. Urinary frequency stage
13.Urine color
14.Location

Target variable : Diagnose

Gathered data

id	age	gender	diabetic	family_history	obesity	smoking	alcohol	prolong_use_of_medication	urinary_obstructions	edema_symptoms	urine_frequency_stage	urine_color	location	diagnose
1	26	Male	No	No	Yes	Yes	Yes	No	Yes	No	2	2	Kurunegala	Risky
2	57	Female	Yes	Yes	No	No	Yes	Yes	Yes	Yes	0	1	Kurunegala	Risky
3	59	Male	Yes	No	No	No	No	Yes	Yes	Yes	0	0	Kurunegala	Risky
4	39	Female	No	Yes	No	Yes	No	No	Yes	Yes	1	0	Kurunegala	Risky
5	57	Male	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	2	1	Kurunegala	Risky
6	18	Female	Yes	No	No	No	Yes	Yes	Yes	No	2	1	Kurunegala	Risky
7	41	Female	No	No	Yes	Yes	Yes	Yes	Yes	No	0	1	Kurunegala	Risky
8	19	Male	No	Yes	Yes	Yes	No	Yes	No	Yes	2	2	Kurunegala	Risky
9	27	Male	No	No	Yes	Yes	No	Yes	No	Yes	1	2	Kurunegala	Risky
10	55	Female	Yes	No	No	Yes	No	Yes	Yes	Yes	1	2	Kurunegala	Risky
11	19	Female	No	No	Yes	Yes	Yes	No	Yes	No	0	2	Kurunegala	Risky
594	40	Female	No	Yes	Yes	No	Yes	Yes	No	Yes	0	0	Kurunegala	Not Risky
595	61	Male	No	No	No	No	No	No	No	No	1	0	Kurunegala	Not Risky
596	70	Male	No	Yes	No	Yes	No	No	No	No	0	0	Kurunegala	Not Risky
597	42	Male	No	Yes	No	No	Yes	No	Yes	Yes	0	0	Kurunegala	Not Risky
598	43	Male	Yes	Yes	No	Yes	Yes	Yes	No	No	0	0	Kurunegala	Not Risky
599	55	Female	Yes	No	No	No	No	No	Yes	No	0	0	Kurunegala	Not Risky
600	26	Male	Yes	No	Yes	Yes	No	Yes	No	Yes	0	2	Kurunegala	Not Risky

Import libraries

jupyter model Last Checkpoint: Last Tuesday at 18:58 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [111]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LogisticRegression

import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset
df = pd.read_csv('risk_of_ckd.csv')

df.head(10)
```

Out[111]:

	id	age	gender	diabetic	family_history	obesity	smoking	alcohol	prolong_use_of_medication	urinary_obstructions	edema_symptoms	urine_frequency_sta
0	1	26	Male	No	No	Yes	Yes	Yes	No	Yes	No	
1	2	57	Female	Yes	Yes	No	No	Yes	Yes	Yes	Yes	
2	3	59	Male	Yes	No	No	No	No	Yes	Yes	Yes	
3	4	39	Female	No	Yes	No	Yes	No	No	Yes	Yes	
4	5	57	Male	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	
5	6	18	Female	Yes	No	No	No	Yes	Yes	Yes	No	
6	7	41	Female	No	No	Yes	Yes	Yes	Yes	Yes	No	
7	8	19	Male	No	Yes	Yes	Yes	No	Yes	No	Yes	
8	9	27	Male	No	No	Yes	Yes	No	Yes	No	Yes	
9	10	55	Female	Yes	No	No	Yes	No	Yes	Yes	Yes	

Data preprocessing

jupyter model Last Checkpoint: Last Tuesday at 18:58 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

In [114]:
```
print(missing_values)
```

```
id                          0
age                         0
gender                      0
diabetic                    0
family_history              0
obesity                     0
smoking                     0
alcohol                     0
prolong_use_of_medication   0
urinary_obstructions        0
edema_symptoms              0
urine_frequency_stage       0
urine_color                 0
location                    0
diagnose                    0
dtype: int64
```

In [115]:
```
print(df.dtypes)
```

```
id                          int64
age                         int64
gender                     object
diabetic                   object
family_history             object
obesity                    object
smoking                    object
alcohol                    object
prolong_use_of_medication  object
urinary_obstructions       object
edema_symptoms             object
urine_frequency_stage       int64
urine_color                 int64
location                   object
diagnose                   object
dtype: object
```

In [119]:
```
# Create a LabelEncoder object
label_encoder = LabelEncoder()
```

jupyter model Last Checkpoint: Last Tuesday at 18:58 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

In [119]:
```
# Create a LabelEncoder object
label_encoder = LabelEncoder()

# Encode the "gender" column
df['gender_encoded'] = label_encoder.fit_transform(df['gender'])
df['diabetic_encoded'] = label_encoder.fit_transform(df['diabetic'])
df['alcohol_encoded'] = label_encoder.fit_transform(df['alcohol'])
df['urinary_obstructions_encoded'] = label_encoder.fit_transform(df['urinary_obstructions'])
df['smoking_encoded'] = label_encoder.fit_transform(df['smoking'])
df['obesity_encoded'] = label_encoder.fit_transform(df['obesity'])
df['edema_symptoms_encoded'] = label_encoder.fit_transform(df['edema_symptoms'])
df['prolong_use_of_medication_encoded'] = label_encoder.fit_transform(df['prolong_use_of_medication'])
df['family_history_encoded'] = label_encoder.fit_transform(df['family_history'])


print(df.dtypes)
```

```
id                                int64
age                               int64
gender                           object
diabetic                         object
family_history                   object
obesity                          object
smoking                          object
alcohol                          object
prolong_use_of_medication        object
urinary_obstructions             object
edema_symptoms                   object
urine_frequency_stage             int64
urine_color                       int64
location                         object
diagnose                         object
gender_encoded                    int32
diabetic_encoded                  int32
alcohol_encoded                   int32
urinary_obstructions_encoded      int32
smoking_encoded                   int32
obesity_encoded                   int32
```

jupyter model Last Checkpoint: Last Tuesday at 18:58 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [131]:
```
columns_to_drop = ['location','id']  # Replace with the actual column names you want to drop
df = df.drop(columns_to_drop, axis=1)
```

In [132]:
```
print(df.dtypes)
```

```
age                                int64
urine_frequency_stage              int64
urine_color                        int64
diagnose                          object
gender_encoded                     int32
diabetic_encoded                   int32
alcohol_encoded                    int32
urinary_obstructions_encoded       int32
smoking_encoded                    int32
obesity_encoded                    int32
edema_symptoms_encoded             int32
prolong_use_of_medication_encoded  int32
family_history_encoded             int32
dtype: object
```

In [74]:

In [ ]:
```
# # Select the features to scale (assuming numerical features)
# numeric_features  = ['blood_pressure', 'age', 'blood_sugar', 'albumin', 'hemoglobin', 'serum_creatinine', 'blood_urea_nitrogen'

# # Exclude non-numeric columns
# numeric_df = df[numeric_features]

# # Create a scaler object
# scaler = MinMaxScaler()

# # Scale the numeric features
# scaled_features = scaler.fit_transform(numeric_df.values)
# df[numeric_features] = scaled_features

# # Print the scaled dataset
# print(df)
```

jupyter model Last Checkpoint: Last Tuesday at 18:58 (autosaved)

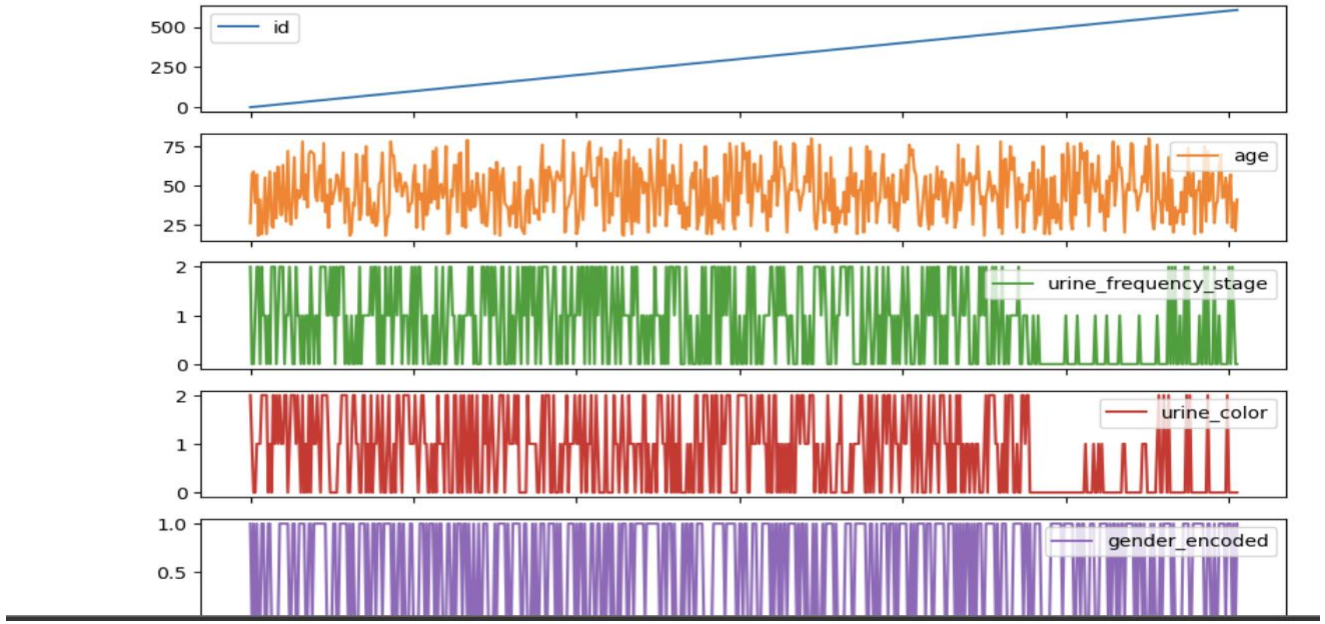
Edit View Insert Cell Kernel Widgets Help

Not Trusted

Run Code

```
# # Print the scaled dataset
# print(df)
```

In [129]:
```
# Plot all the columns
df.plot(subplots=True, layout=(len(df.columns), 1), figsize=(10, 20))
plt.show()
```



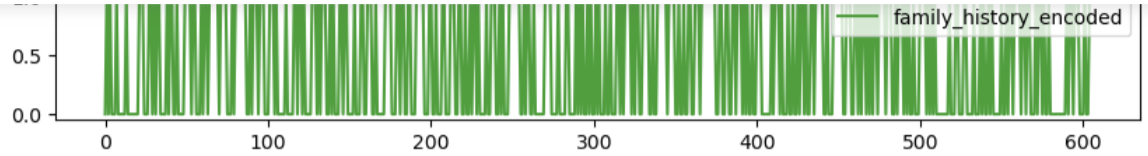
jupyter model Last Checkpoint: Last Tuesday at 18:58 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code



In [133]:
```
X = df.drop('diagnose', axis=1)  # Update 'target_variable' with the actual column name
y = df['diagnose']
```

In [134]:
```
print(df.dtypes)
```

```
age                                 int64
urine_frequency_stage               int64
urine_color                         int64
diagnose                           object
gender_encoded                      int32
diabetic_encoded                    int32
alcohol_encoded                     int32
urinary_obstructions_encoded        int32
smoking_encoded                     int32
obesity_encoded                     int32
edema_symptoms_encoded              int32
prolong_use_of_medication_encoded   int32
family_history_encoded              int32
dtype: object
```

In [135]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

In [136]:
```
# Create a LabelEncoder object
label_encoder = LabelEncoder()

# Encode the target variable
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)

# Build the linear regression model
model = LinearRegression()
```

Build the model and train

jupyter model Last Checkpoint: Last Tuesday at 18:58 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

In [136]:
```
# Create a LabelEncoder object
label_encoder = LabelEncoder()

# Encode the target variable
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)

# Build the linear regression model
model = LinearRegression()
model.fit(X_train, y_train_encoded)

# Predict the target variable for the test set
y_pred = model.predict(X_test)

# Calculate mean squared error (MSE)
mse = mean_squared_error(y_test_encoded, y_pred)

# Print the MSE
print("Mean Squared Error (MSE):", mse)
```

```
Mean Squared Error (MSE): 0.10486331057702959
```

In [137]:
```
from sklearn.metrics import r2_score

# Predict the target variable for the test set
y_pred = model.predict(X_test)

# Calculate R-squared
r2 = r2_score(y_test_encoded, y_pred)

# Print the R-squared
print("R-squared:", r2)
```

```
R-squared: 0.37468529061357825
```

In [138]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a logistic regression model
```

jupyter model Last Checkpoint: Last Tuesday at 18:58 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

```
# Predict the target variable for the test set
y_pred = model.predict(X_test)

# Calculate R-squared
r2 = r2_score(y_test_encoded, y_pred)

# Print the R-squared
print("R-squared:", r2)
```

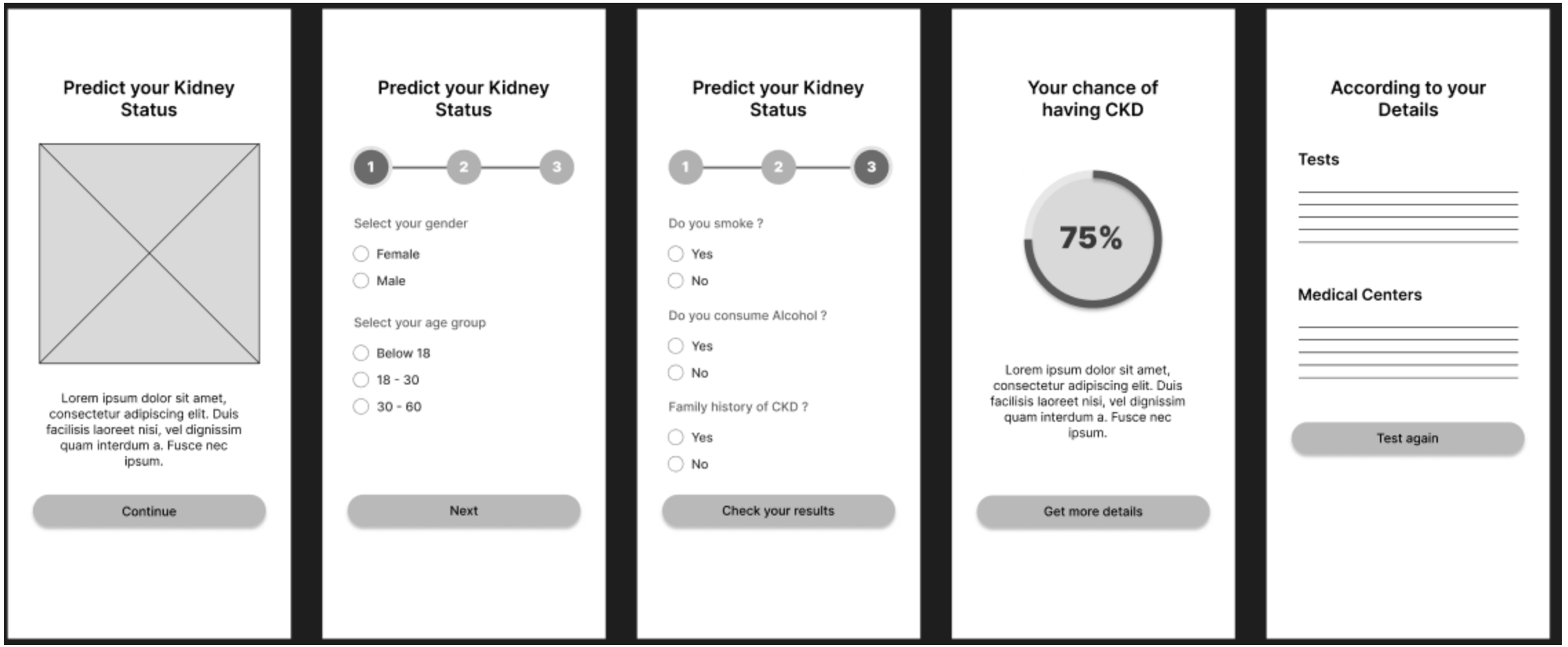
```
R-squared: 0.37468529061357825
```

In [138]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a logistic regression model
model = LogisticRegression()

# Fit the model on the training data
model.fit(X_train, y_train)

# Predict the target variable for the test set
y_pred = model.predict(X_test)

# Calculate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)

# Print the accuracy
print("Accuracy:", accuracy)
```

```
Accuracy: 0.8934426229508197
```

In [ ]:

Wireframes


Predict your Kidney Status
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis facilisis laoreet nisi, vel dignissim quam interdum a. Fusce nec ipsum.
Continue
Predict your Kidney Status
1
2
3
Select your gender
Female
Male
Select your age group
Below 18
18 - 30
30 - 60
Next
Predict your Kidney Status
1
2
3
Do you smoke ?
Yes
No
Do you consume Alcohol ?
Yes
No
Family history of CKD ?
Yes
No
Check your results
Your chance of having CKD
75%
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis facilisis laoreet nisi, vel dignissim quam interdum a. Fusce nec ipsum.
Get more details
According to your Details
Tests
Medical Centers
Test again

What is planned to do next

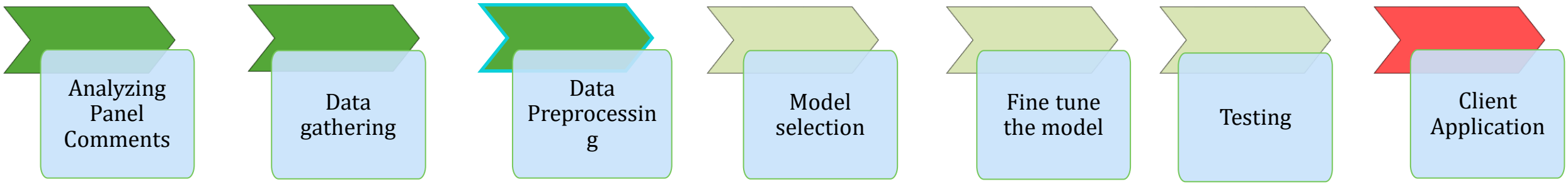
- Plan to get results accuracies for algorithms

Used Algorithm	Overall Accuracy
Linear regression	85.54%
Logistic Regression	89.17%
Random Forest	Plan to implement
Support Vector Machines (SVM)	Plan to implement
Naive Bayes	Plan to implement

- Recommend the nearest kidney treatment unit for a specific person where they can go through the necessary tests.

- Implementing the integration between the backend and frontend components of the mobile application.

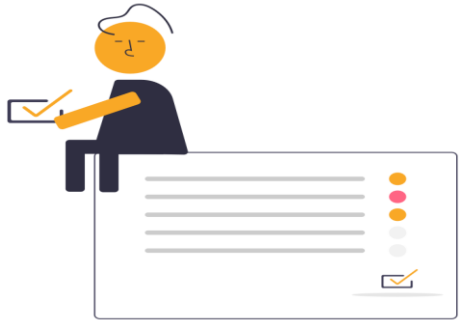
Overall Progress



Completed

To be started

Work In Progress



Tools and Technologies

Software technologies

- Python
- TensorFlow
- AWS / Azure
- React Native
- MongoDB
- Node JS

Requirement Analysis

Functional requirements

- Gathering the data and preprocessed that data.
- Train highly accurate ML model with collected data.
- Giving the prediction according to the real time test data.
- Giving recommendations for the treatments.

Non-functional requirements

- Accuracy
- Availability
- Performance
- Speed

References

[1] Gazi Mohammed Ifraz, “Comparative Analysis for Prediction of Kidney Disease Using Intelligent Machine Learning Methods,”National Library of Medicine, 2021.

[2] H. B. A. S. B. Abhijit V. Kshirsagar, "A Simple Algorithm to Predict Incident Kidney Disease," JAMA Network, 2008.

[3] A. Vijayalakshmi, “Survey on Diagnosis of Chronic Kidney Disease UsingMachine Learning Algorithms,”IEEE, 2020.

IT20154226 | M.M.K.L. Marasinghe

Specializing in Information Technology

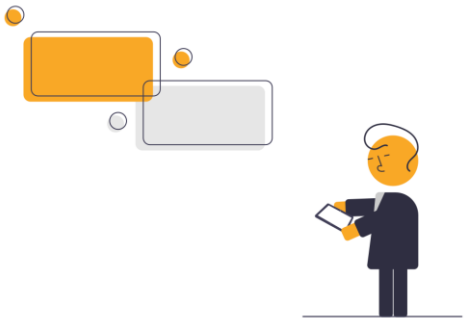
Kidney Disease Diagnosis using Image Processing

Background

- Types of image modalities
  - ❑ MRI, CT, Ultrasound, X-Ray
- Importance of medical imaging when diagnosing CKD patients
  - ❑ Accessibility, Point of Care analysis, Patient Education
- Available methods and architectures that can be used for analyze medical images
  - ❑ CNN
  - ❑ Deep Convolutional Autoencoders
  - ❑ Graph Convolutional Networks
  - ❑ Transfer Learning
  - ❑ Image Registration and Fusion

Research Problem

- How to use image processing to analyze CKD patients' medical images?
- What are the parameters to identify when analyzing medical images using image processing?
- How can image fusion techniques be utilized to integrate information from multiple imaging modalities?



Research Gap

- Developing a comprehensive, technology-driven approach to accurately diagnose kidney diseases using more advanced technologies.
- Automatic abnormality detection of kidney using CNN
- Multimodal Image Fusion to get more accurate predictions.
- Using standardization to keep track of the medical images uploaded by the user.

Objectives

- Identify parameters of the medical images
- Analyse medical images to diagnose the patient more accurately using image processing.
- Using standardization, create a visual representation of the progression of the kidneys over time.
- Using multimodal image fusion to get better accurate results.

Requirement Analysis

Functional requirements

- Analyse medical images to diagnose the patient more accurately
- Create a visual representation of the progression of the kidneys over time.

Non-functional requirements

- Accuracy
- Availability
- Performance
- Speed

Completion of the Component

- Gathering medical imaging dataset from internet resources
- Analyse the Data set
- Pre-process the Data Set
- Build the Model and Train the Model.

What's next ?

- Finalize the CNN model.
- Implement multimodal image fusion functionality to get more accurate results.
- Using standardization to track the progression of the kidney over time.
- Implementing the integration between the backend and frontend components of the mobile application.

Implementation

Loading the Image Dataset

```
import cv2
import glob

# Specifying the directory path containing the images
image_dir = 'D:\kidneyUS_images_14_june_2022\kidneyUS_images_14_june_2022/*.png'  # Update with the appropriate file extension ar

# Use glob to retrieve the file paths of all images in the directory
image_paths = glob.glob(image_dir)

# Create an empty list to store the loaded images
images = []

# Loading each image using OpenCV and handle loading errors
for image_path in image_paths:
    image = cv2.imread(image_path)

    # Checking if the image is loaded successfully
    if image is not None:
        images.append(image)
    else:
        print(f"Error loading image: {image_path}")

# Checking the number of successfully loaded images
num_loaded_images = len(images)
print(f"Successfully loaded {num_loaded_images} images out of {len(image_paths)}")
```

```
Successfully loaded 534 images out of 534
```

Resizing and splitting the image dataset

In [2]:
```
import cv2

# Specify the desired width and height for the resized images
desired_width = 1024
desired_height = 768

# Create an empty list to store the resized images
resized_images = []

# Resize each image using OpenCV
for image in images:
    resized_image = cv2.resize(image, (desired_width, desired_height))
    resized_images.append(resized_image)

# Check if the images are resized successfully
for i, image in enumerate(images):
    original_height, original_width = image.shape[:2]
    resized_height, resized_width = resized_images[i].shape[:2]

    if original_height != resized_height or original_width != resized_width:
        print(f"Image {i+1} was not resized correctly.")
    else:
        print(f"Image {i+1} was resized successfully.")
```

```
Image 1 was resized successfully.
Image 2 was resized successfully.
Image 3 was not resized correctly.
Image 4 was resized successfully.
Image 5 was resized successfully.
Image 6 was resized successfully.
Image 7 was resized successfully.
Image 8 was resized successfully.
Image 9 was resized successfully.
Image 10 was resized successfully.
```

```
# Check if any images were dropped during resizing
if len(images) == len(resized_images_success):
    print("All images were successfully resized.")
else:
    print(f"{len(images) - len(resized_images_success)} images were dropped during resizing.")
```

```
All images were successfully resized.
```

```
# Create an empty list to store the successfully resized images
resized_images_success = []

# Create a new list to store the indices of the successfully resized images
resized_indices_success = []

# Iterate over the original images and their corresponding resized images
for i, (image, resized_image) in enumerate(zip(images, resized_images)):
    if resized_image is not None:
        # If the resized image is not None, it indicates successful resizing
        resized_images_success.append(resized_image)
        resized_indices_success.append(i)
    else:
        print(f"Image {i+1} was not resized successfully and will be dropped.")

# Update the 'images' list with the successfully resized images
images = resized_images_success
```

```
from sklearn.model_selection import train_test_split
import random

# Define the labels
labels = ["1", "2", "3", "3", "4", "5"]

# Assign labels randomly to the resized images
random_labels = random.choices(labels, k=len(resized_images_success))

# Split the data into training, validation, and testing sets
train_images, test_images, train_labels, test_labels = train_test_split(resized_images_success, random_labels, test_size=0.2, rar
train_images, val_images, train_labels, val_labels = train_test_split(train_images, train_labels, test_size=0.2, random_state=42)

# Print the sizes of the resulting sets
print("Training set size:", len(train_images))
print("Validation set size:", len(val_images))
print("Testing set size:", len(test_images))
```

```
Training set size: 341
Validation set size: 86
Testing set size: 107
```

Pre-processing the input data for TensorFlow model training

In [6]:
```
import tensorflow as tf
import numpy as np

# Convert images to NumPy arrays
train_images = np.array(train_images)
val_images = np.array(val_images)
test_images = np.array(test_images)

# Convert images to float32 and normalize
train_images_tensor = tf.convert_to_tensor(train_images.astype(np.float32) / 255.0)
val_images_tensor = tf.convert_to_tensor(val_images.astype(np.float32) / 255.0)
test_images_tensor = tf.convert_to_tensor(test_images.astype(np.float32) / 255.0)

# Convert labels to TensorFlow tensors
train_labels_tensor = tf.convert_to_tensor(train_labels)
val_labels_tensor = tf.convert_to_tensor(val_labels)
test_labels_tensor = tf.convert_to_tensor(test_labels)
```

Defining the CNN Model

In [7]:

```
import tensorflow as tf
from tensorflow.keras import layers

image_height = 1024
image_width = 768
num_channels = 3

# Define the CNN model
model = tf.keras.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(768, 1024, 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(1, activation='sigmoid')
])
```

CNN model compilation for image classification

```
In [8]: model.compile(optimizer='adam',
                      loss='sparse_categorical_crossentropy',
                      metrics=['accuracy'])
```

Label Encoding and One-Hot Encoding for Classification Labels

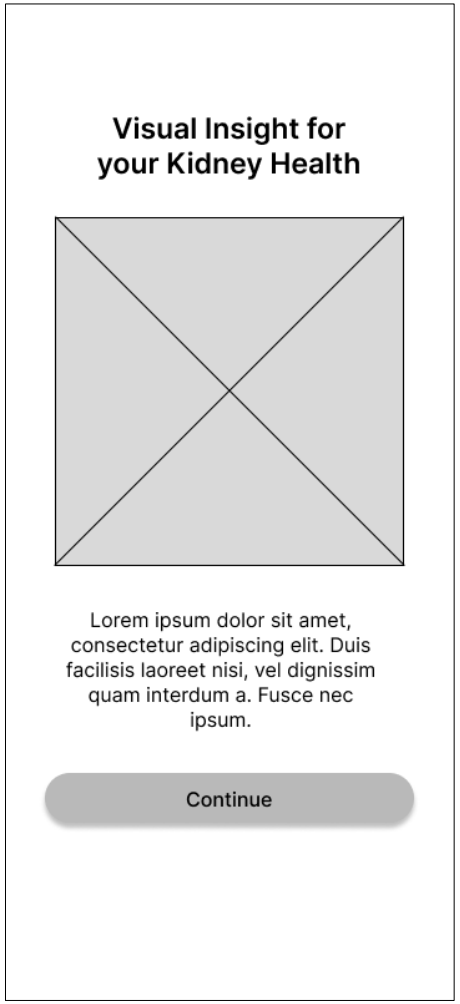
```
In [10]: import numpy as np
         import tensorflow as tf

         actual_labels = ['3', '4', '5', '2', '3', '3', '3', '3', '3', '5', '3', '3', '1', '5', '3', '1', '4', '3', '1', '1', '2', '3', '5

         # Create a dictionary to map the label names to integer values
         label_to_integer = {label: i for i, label in enumerate(set(actual_labels))}

         # Convert the actual labels to numerical values using the mapping
         train_labels = np.array([label_to_integer[label] for label in actual_labels])

         # Convert the numerical labels to one-hot encoded format
         num_classes = len(label_to_integer)
         train_labels_encoded = tf.one_hot(train_labels, num_classes)

In [12]: train_labels_encoded = tf.reshape(train_labels_encoded, (-1, num_classes))

In [13]: import numpy as np

         num_classes = len(np.unique(train_labels))from collections import Counter

         label_counts = Counter(train_labels)
         num_classes = len(label_counts)

In [14]: num_classes = len(set(train_labels))

In [15]: from collections import Counter

         label_counts = Counter(train_labels)
         num_classes = len(label_counts)

In [16]: model.add(layers.Dense(534, activation='softmax'))
```

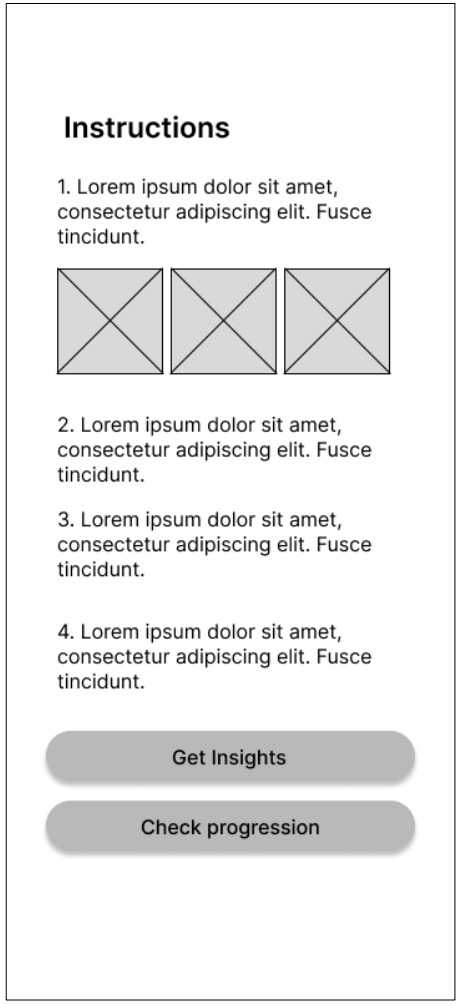
Training the model

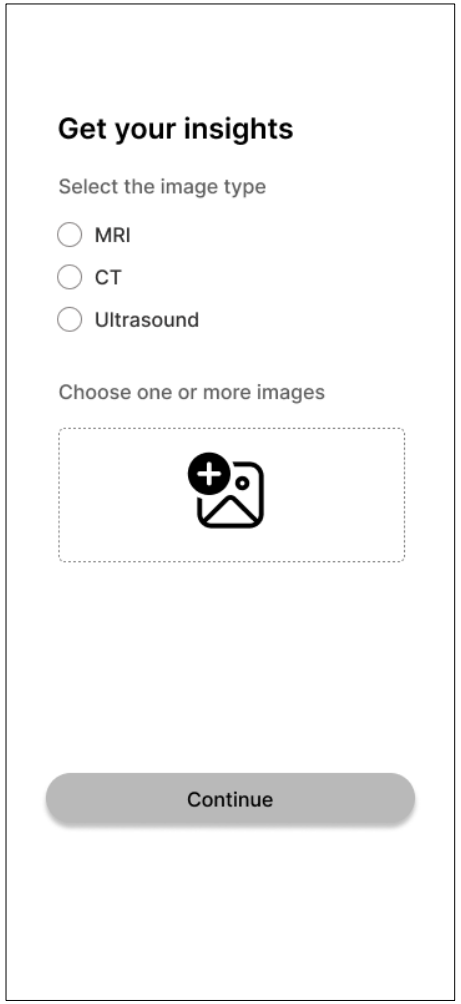
```
In [*]: history = model.fit(train_images_tensor, train_labels_encoded, epochs=10, validation_data=(val_images_tensor, val_labels_tensor))
```

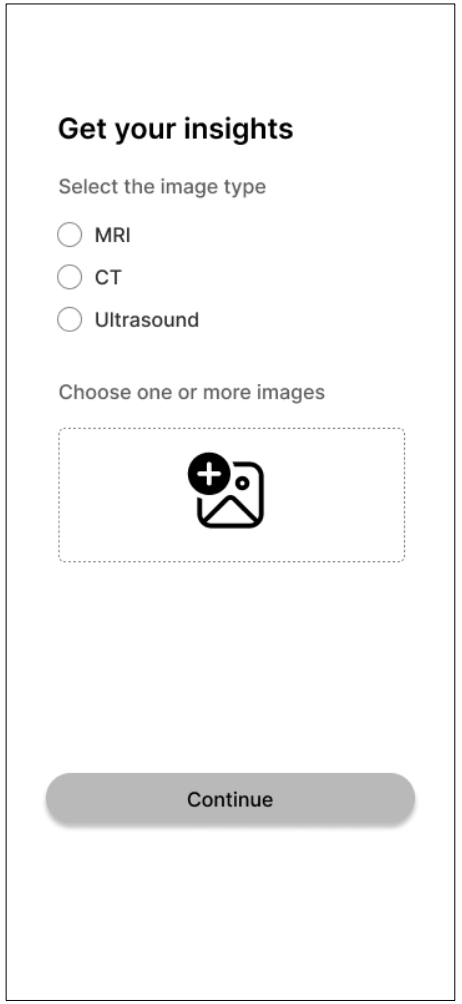
```
Epoch 1/10
```

Wire Frames


Visual Insight for your Kidney Health
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis facilisis laoreet nisi, vel dignissim quam interdum a. Fusce nec ipsum.
Continue


Instructions
1. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce tincidunt.
2. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce tincidunt.
3. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce tincidunt.
4. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce tincidunt.
Get Insights
Check progression


Get your insights
Select the image type
MRI
CT
Ultrasound
Choose one or more images
Continue


Get your insights
Select the image type
MRI
CT
Ultrasound
Choose one or more images
Continue

Your insights

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis facilisis laoreet nisi, vel dignissim quam interdum a. Fusce nec ipsum.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce tincidunt.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce tincidunt.

Save as PDF

Let's Check your CKD progression

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce tincidunt.

Choose an image

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce tincidunt.

Check progression

Your CKD progression

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce tincidunt.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce tincidunt.

References

[1] Siddharth Rajput, “Automated Kidney Stone Detection Using Image Processing Techniques,” IEEE, 2021.

[2] Ahmed Soliman, “Kidney segmentation from CT images using a 3D NMF-guided active contour model, “ IEEE, 2021 .

IT20785192 | W.B.M.A. Isurika

Specializing in Data Science

Personalized Health Assessment and Management

Background

Important of Diet plan for CKD patients

- ❑ Slowing the Progression of CKD
- ❑ Managing Fluid Balance
- ❑ helps regulate fluid intake to prevent fluid overload, thereby reducing the strain on the kidneys and managing blood pressure.

Research Problem

- Despite the critical role of personalized diet management in slowing the progression of Chronic Kidney Disease (CKD) and improving patient outcomes.
- Currently they use one leaflet for all patients.

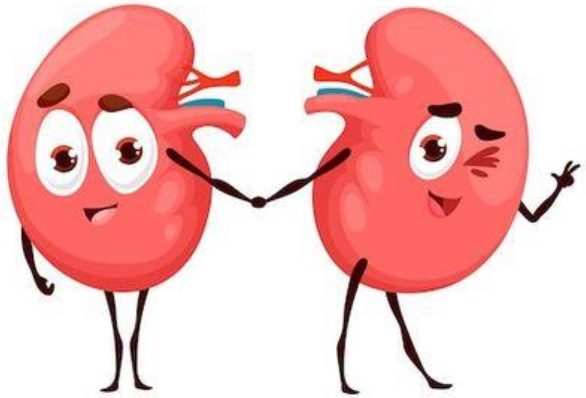
Research Gap

- Recent research has focused on Potassium level of blood to help manage the diet plans for CKD patients.
- Increase the accuracy of model and give proper diet plan
- Most studies have focused on short-term outcomes, such as changes in blood pressure or kidney function, rather than long-term outcomes such as financial side of patients.

What is Zone Attribute ?

- Existing dietary managements needs will be determined based on the ZONE Attribute

  - ✓ Safe - blood potassium level 3.5 - 5.0
  - ✓ Caution - blood potassium level 5.1 - 6.0
  - ✓ Danger - blood potassium level higher than 6.1



What tests they do ?

- "Serum Potassium Test" or "Potassium Blood Test"

Objectives

Main Objective : provide personalized recommendations to kidney patients regarding the most appropriate diet, exercise, and lifestyle changes based on their health condition.

Sub Objectives :

- ❑ Based on the Potassium level of the blood, categorize the patient into three zones.
- ❑ Categorize and prepare proper diet plan

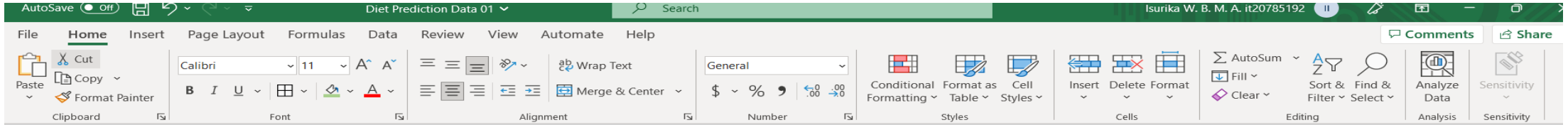
Completion of the Component

- Requirements Analysis
  - Collect data manually and create data set

- Attributes

ID	Serum Albumin (g/dL)
Gender	Potassium
Age	Calcium
Height(m)	Phosphorus
Weight(kg)	Sodium
Stage of CKD	Hemoglobin
Current Medications	Cholesterol

Data Set



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	ID	Gender	Age	Height(m)	Weight(kg)	Current Medications	Stage of CKD	Serum Albumin (g/dL)	Pottasium	Calcium	Phosphorus	Sodium	Himoglobin	Cholesterol	Zone
2	1	Male	47	1.2	51.7	FALSE	4	4.9	5.4	7	3.9	138.2	11.3	14	3
3	2	Male	42	1.5	73.4	TRUE	3	5.4	3.8	11	5.9	139.4	12.3	14.3	2
4	3	Male	38	1.2	53.7	FALSE	3	2.5	4.8	8.8	6.3	146.1	15.6	16.3	2
5	4	Male	86	1.5	70.3	TRUE	5	2.5	6.9	8.3	6.7	123.7	12.3	13.8	4
6	5	Male	67	1.4	73.4	FALSE	3	2.7	2.9	10.5	4.6	133.3	11.8	14.7	1
7	6	Male	43	1.3	66	FALSE	3	3.6	3.6	11.3	6.4	154.4	16	17.6	2
8	7	Male	75	1.6	73.2	TRUE	2	5	2.9	11.5	7	155.9	12	10.6	1
9	8	Male	69	1.5	78.4	FALSE	3	3	5.7	10.2	2.4	129.6	13.7	17.9	3
10	9	Male	59	1.3	66.5	FALSE	5	5.9	2.8	11.6	6.2	148.3	13.6	17.3	1
11	10	Male	80	1.5	49.7	TRUE	5	2.7	3.5	11.6	6.3	120.3	13.6	14.8	2
12	11	Male	61	1.3	71.4	FALSE	3	3	7	8.9	6.3	155.1	13.5	16	4
13	12	Male	46	1.3	66	FALSE	1	2.5	4.4	11.8	5.9	129.8	10.6	13.1	2
14	13	Male	66	1.3	79.1	TRUE	3	5.9	3.3	6.5	3.2	157.8	17.3	13.3	1
15	14	Male	46	1.6	64	FALSE	3	4.5	3.9	6.6	6.3	153.9	10.9	16.2	2
16	15	Male	73	1.3	71.2	FALSE	3	3.2	6.2	8.1	6.9	158.8	18.2	12.4	4

511	510	Female	39	1.1	118.78	TRUE	3	3.1	3.7	7.4	6	137	10.7	13.1	2
512	511	Female	55	1	101.1	FALSE	1	4.2	3.4	11.7	5.7	149.4	13.6	16.7	1
513	512	Female	47	1.2	95.96	TRUE	1	3.5	6.6	7.4	6.3	128.5	10.6	12.8	4
514	513	Female	40	1.6	108.64	FALSE	2	5.1	5.7	10.9	6	157.1	13.5	17.8	3
515	514	Female	53	1.6	76.91	FALSE	4	5.2	2.4	10.2	2.2	152.5	12.6	11.7	1
516	515	Female	61	1.3	61.33	FALSE	3	4.5	4.2	9.2	3.5	133.1	13	15.6	2
517	516	Female	47	1.3	59.17	FALSE	2	5.4	5.6	8.7	5.4	130.8	10.7	11.4	3
518	517	Female	61	1.6	103.46	FALSE	2	3.1	4.6	11.1	5.8	124.1	11.3	13.8	2
519															

Import libraries

Untitled30.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk

```
#import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import numpy

from sklearn.preprocessing import MinMaxScaler
from sklearn import metrics
from sklearn import svm
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, r2_score

import warnings
%matplotlib inline
warnings.filterwarnings('ignore')
```

```
# Assuming you have a dataset with 'X' as input features and 'y' as the target variable
data = pd.read_csv('Diet Prediction Data 01.csv')
data.head()
```

	ID	Gender	Age	Height(m)	Weight(kg)	Current Medications	Stage of CKD	Serum Albumin (g/dL)	Pottasium	Calcium	Phosphorus	Sodium	Himoglobin	Cholesterol	Zone
0	1	Male	47	1.2	51.7	False	4.0	4.9	5.4	7.0	3.9	138.2	11.3	14.0	3
1	2	Male	42	1.5	73.4	True	3.0	5.4	3.8	11.0	5.9	139.4	12.3	14.3	2
2	3	Male	38	1.2	53.7	False	3.0	2.5	4.8	8.8	6.3	146.1	15.6	16.3	2
3	4	Male	86	1.5	70.3	True	5.0	2.5	6.9	8.3	6.7	123.7	12.3	13.8	4
4	5	Male	67	1.4	73.4	False	3.0	2.7	2.9	10.5	4.6	133.3	11.8	14.7	1

Data Preprocessing

```
[145] #To view the last few rows,
      data.tail()
```

	ID	Gender	Age	Height(m)	Weight(kg)	Current Medications	Stage of CKD	Serum Albumin (g/dL)	Pottasium	Calcium	Phosphorus	Sodium	Himoglobin	Cholesterol	Zone
721	722	Female	35	1.4	82.76	False	3.0	6.3	3.5	9.1	3.4	147.0	13.2	15.2	2
722	723	Female	29	1.0	67.30	False	1.0	4.6	5.0	9.2	6.9	125.1	13.5	10.9	2
723	724	Female	61	1.1	100.45	False	3.0	4.4	6.5	9.9	2.7	139.8	14.2	10.5	4
724	725	Female	31	1.2	110.13	False	3.0	3.4	2.5	10.0	5.7	147.5	13.5	16.8	1
725	726	Female	44	1.6	95.65	False	3.0	4.0	6.1	6.7	2.3	140.3	13.1	14.8	4

```
[146] #get the column headings of the data set
      data.columns.values

array(['ID', 'Gender', 'Age', 'Height(m)', 'Weight(kg)',
       'Current Medications', 'Stage of CKD', 'Serum Albumin (g/dL)',
       'Pottasium', 'Calcium', 'Phosphorus', 'Sodium', 'Himoglobin',
       'Cholesterol', 'Zone'], dtype=object)
```


```
[146] #get the column headings of the data set
      data.columns.values

array(['ID', 'Gender', 'Age', 'Height(m)', 'Weight(kg)',
       'Current Medications', 'Stage of CKD', 'Serum Albumin (g/dL)',
       'Pottasium', 'Calcium', 'Phosphorus', 'Sodium', 'Himoglobin',
       'Cholesterol', 'Zone'], dtype=object)
```


```
[147] #Explore the data types of the columns
      data.dtypes

ID                      int64
Gender                 object
Age                     int64
Height(m)             float64
Weight(kg)            float64
Current Medications    object
Stage of CKD          float64
Serum Albumin (g/dL)  float64
Pottasium             float64
Calcium               float64
Phosphorus            float64
Sodium                float64
Himoglobin            float64
Cholesterol           float64
Zone                    int64
dtype: object
```

```
[148] #describe the data set
      data.describe()
```

	ID	Age	Height(m)	Weight(kg)	Stage of CKD	Serum Albumin (g/dL)	Pottasium	Calcium	Phosphorus	Sodium	Himoglobin	Cholesterol	Zone
count	726.000000	726.000000	726.000000	726.000000	716.000000	726.000000	726.000000	720.000000	726.000000	726.000000	726.000000	720.000000	726.000000
mean	363.500000	57.071625	1.344766	75.338802	3.001397	4.442287	4.770386	9.230139	4.429063	140.465840	13.429477	14.487361	2.442149
std	209.722436	17.026562	0.157983	19.197597	1.097358	1.177497	1.306648	1.587852	1.487213	11.533203	2.135800	2.254686	1.035288
min	1.000000	28.000000	1.000000	48.000000	1.000000	2.400000	2.400000	6.500000	1.900000	120.100000	10.500000	10.500000	1.000000
25%	182.250000	42.000000	1.200000	60.100000	2.750000	3.500000	3.625000	7.900000	3.100000	130.425000	11.700000	12.600000	2.000000
50%	363.500000	57.000000	1.300000	72.200000	3.000000	4.400000	4.800000	9.300000	4.400000	141.400000	13.100000	14.500000	2.000000
75%	544.750000	72.000000	1.500000	88.152500	3.000000	5.400000	5.900000	10.600000	5.700000	150.100000	14.500000	16.325000	3.000000
max	726.000000	87.000000	1.600000	119.690000	5.000000	6.500000	7.000000	12.000000	7.000000	160.000000	18.500000	18.500000	4.000000

```
[149] #To get an idea about the null values
      data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 726 entries, 0 to 725
Data columns (total 15 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   ID                     726 non-null    int64
 1   Gender                 726 non-null    object
 2   Age                    726 non-null    int64
 3   Height(m)              726 non-null    float64
 4   Weight(kg)             726 non-null    float64
 5   Current Medications    704 non-null    object
 6   Stage of CKD           716 non-null    float64
 7   Serum Albumin (g/dL)   726 non-null    float64
 8   Pottasium              726 non-null    float64
 9   Calcium                720 non-null    float64
 10  Phosphorus             726 non-null    float64
 11  Sodium                 726 non-null    float64
 12  Himoglobin             726 non-null    float64
 13  Cholesterol            720 non-null    float64
 14  Zone                   726 non-null    int64
dtypes: float64(10), int64(3), object(2)
```

```
[150] #Getting the null value count
      data.isnull().sum()
```

```
ID                     0
Gender                 0
Age                    0
Height(m)              0
Weight(kg)             0
Current Medications    22
Stage of CKD           10
Serum Albumin (g/dL)   0
Pottasium              0
Calcium                6
Phosphorus             0
Sodium                 0
Himoglobin             0
Cholesterol            6
Zone                   0
dtype: int64
```

```
#replace null values using mean and value counts
Df_dataset = data.apply(lambda x: x.fillna(x.mean())
                        if x.dtype =='float'
                        else x.fillna(x.value_counts().index[0]))
```

```
[153] from sklearn.preprocessing import LabelEncoder

      # Convert categorical variable using LabelEncoder
      label_encoder = LabelEncoder()
      data['Gender'] = label_encoder.fit_transform(data['Gender'])
      data['Current Medications'] = label_encoder.fit_transform(data['Current Medications'])
```

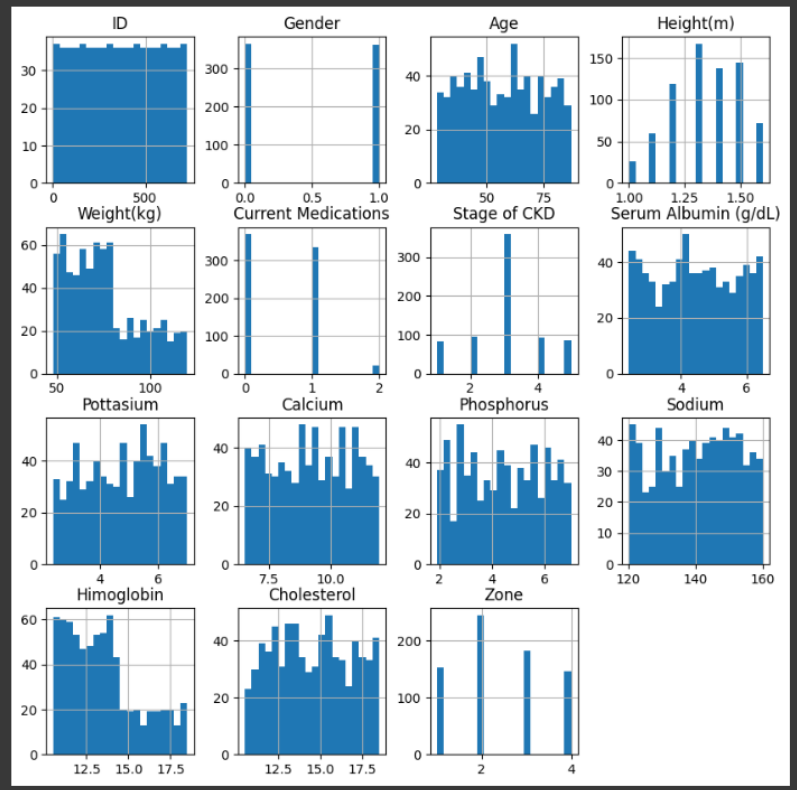
```
[154] data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 726 entries, 0 to 725
Data columns (total 15 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   ID                    726 non-null    int64
 1   Gender                726 non-null    int64
 2   Age                   726 non-null    int64
 3   Height(m)             726 non-null    float64
 4   Weight(kg)            726 non-null    float64
 5   Current Medications   726 non-null    int64
 6   Stage of CKD          716 non-null    float64
 7   Serum Albumin (g/dL)  726 non-null    float64
 8   Pottasium             726 non-null    float64
 9   Calcium               720 non-null    float64
 10  Phosphorus            726 non-null    float64
 11  Sodium                726 non-null    float64
 12  Himoglobin            726 non-null    float64
 13  Cholesterol           720 non-null    float64
 14  Zone                  726 non-null    int64
dtypes: float64(10), int64(5)
memory usage: 85.2 KB
```

```
[155] data
```

	ID	Gender	Age	Height(m)	Weight(kg)	Current Medications	Stage of CKD	Serum Albumin (g/dL)	Pottasium	Calcium	Phosphorus	Sodium	Himoglobin	Cholesterol	Zone
0	1	1	47	1.2	51.70	0	4.0	4.9	5.4	7.0	3.9	138.2	11.3	14.0	3
1	2	1	42	1.5	73.40	1	3.0	5.4	3.8	11.0	5.9	139.4	12.3	14.3	2
2	3	1	38	1.2	53.70	0	3.0	2.5	4.8	8.8	6.3	146.1	15.6	16.3	2
3	4	1	86	1.5	70.30	1	5.0	2.5	6.9	8.3	6.7	123.7	12.3	13.8	4
4	5	1	67	1.4	73.40	0	3.0	2.7	2.9	10.5	4.6	133.3	11.8	14.7	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
721	722	0	35	1.4	82.76	0	3.0	6.3	3.5	9.1	3.4	147.0	13.2	15.2	2
722	723	0	29	1.0	67.30	0	1.0	4.6	5.0	9.2	6.9	125.1	13.5	10.9	2
723	724	0	61	1.1	100.45	0	3.0	4.4	6.5	9.9	2.7	139.8	14.2	10.5	4

```
[156] data.hist(bins=20, figsize=(10, 10))
      plt.show()
```



```
[158] #split the dataset before balancing the class.
      X = Df_dataset.drop(columns=[ 'Zone'], axis 1)
      y = Df_dataset['Zone']
```

```
[159] X.columns.values
```

```
array(['ID', 'Gender', 'Age', 'Height(m)', 'Weight(kg)',
       'Current Medications', 'Stage of CKD', 'Serum Albumin (g/dL)',
       'Pottasium', 'Calcium', 'Phosphorus', 'Sodium', 'Himoglobin',
       'Cholesterol'], dtype=object)
```

```
[160] y.value_counts()
```

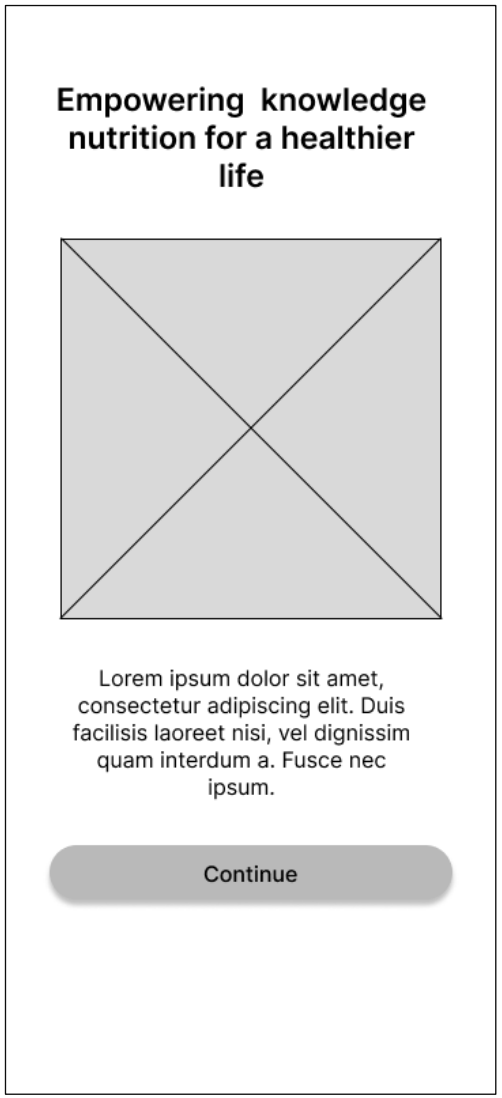
```
2    245
3    182
1    153
4    146
Name: Zone, dtype: int64
```

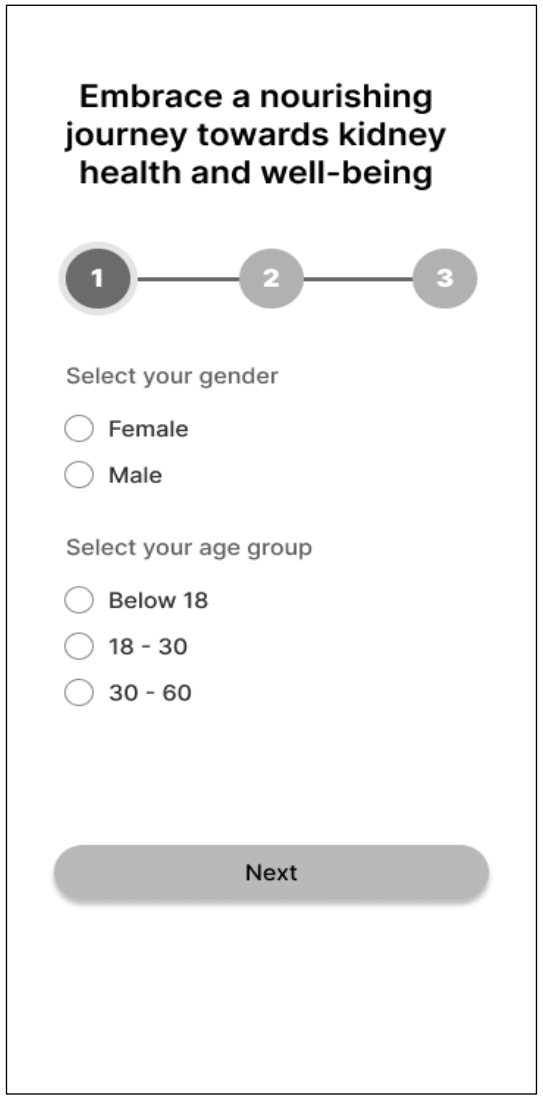
```
#delete missing values rows
data = data.dropna(axis=0)
```

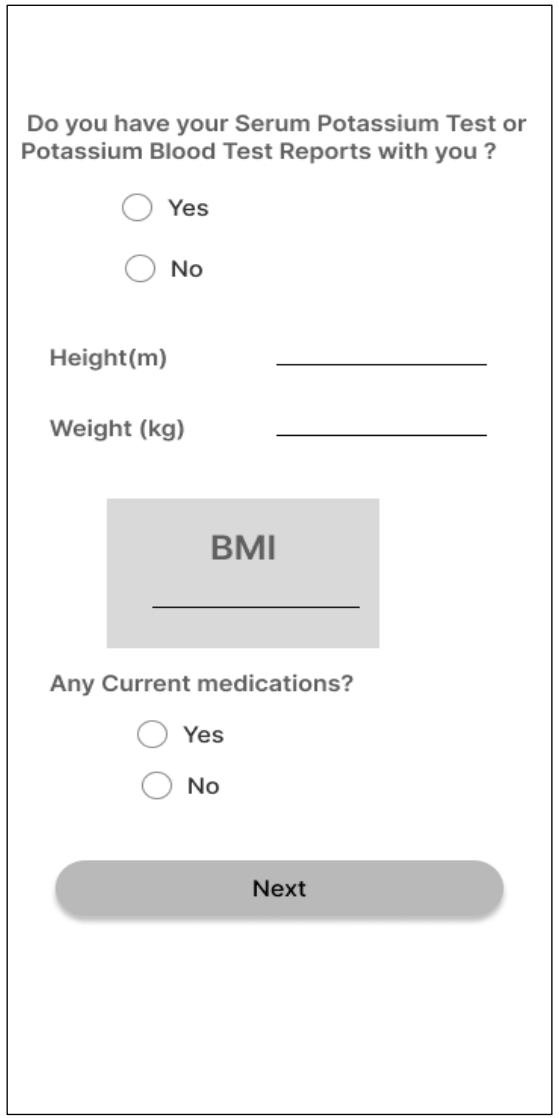
```
[163] # Split the data into train and test sets
      X_train, X_test, y_train, y_test = train_test_split(data.drop("Zone", axis=1), data["Zone"], test_size=0.25)

      # Create the random forest model
      rf_model = RandomForestClassifier(n_estimators=100, max_depth=10)

      # Fit the model to the training data
      rf_model.fit(X_train, y_train)

      # Predict the labels for the test data
      y_pred = rf_model.predict(X_test)

      # Calculate the accuracy of the model
      accuracy = accuracy_score(y_test, y_pred)
      print("Accuracy:", accuracy)

      # Calculate the R-squared value of the model
      r2 = r2_score(y_test, y_pred)
      print("R-squared:", r2)
```

```
Accuracy: 0.95
R-squared: 0.94269853783
```

Wireframes


Empowering knowledge nutrition for a healthier life
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis facilisis laoreet nisi, vel dignissim quam interdum a. Fusce nec ipsum.
Continue


Embrace a nourishing journey towards kidney health and well-being
1
2
3
Select your gender
Female
Male
Select your age group
Below 18
18 - 30
30 - 60
Next


Do you have your Serum Potassium Test or Potassium Blood Test Reports with you ?
Yes
No
Height(m)
Weight (kg)
BMI
Any Current medications?
Yes
No
Next

Serium Albumin (g/DL) _____

Pottasium _____

Calcium _____

Phosphorus _____

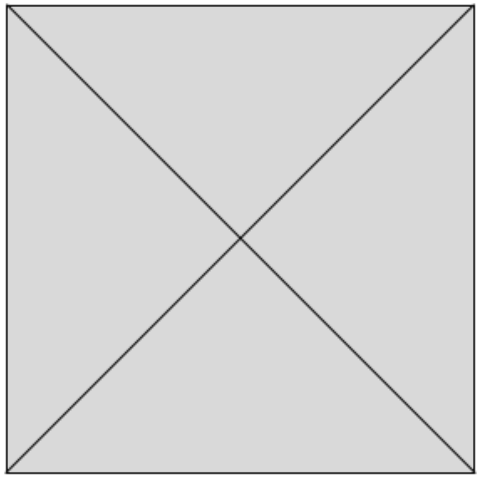
Sodium _____

Himoglobin _____

Check your results

Here's proper diet meal for you

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis facilisis laoreet nisi, vel dignissim quam interdum a. Fusce nec ipsum.

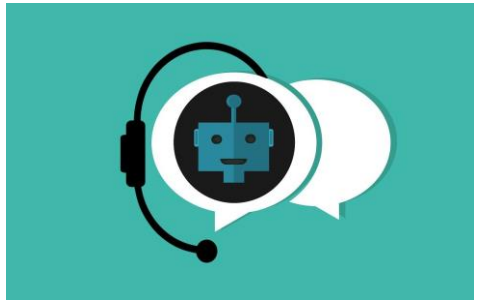


What we plan to do next ?

- Plan to get results accuracies for algorithms

Used Algorithm	Overall Accuracy
Random Forest	95.54%
Logistic Regression	89.17%
Decision Trees	Plan to implement
Support Vector Machines (SVM)	Plan to implement
Naive Bayes	Plan to implement

- Implement ChatBot give proper Diet plan for patients



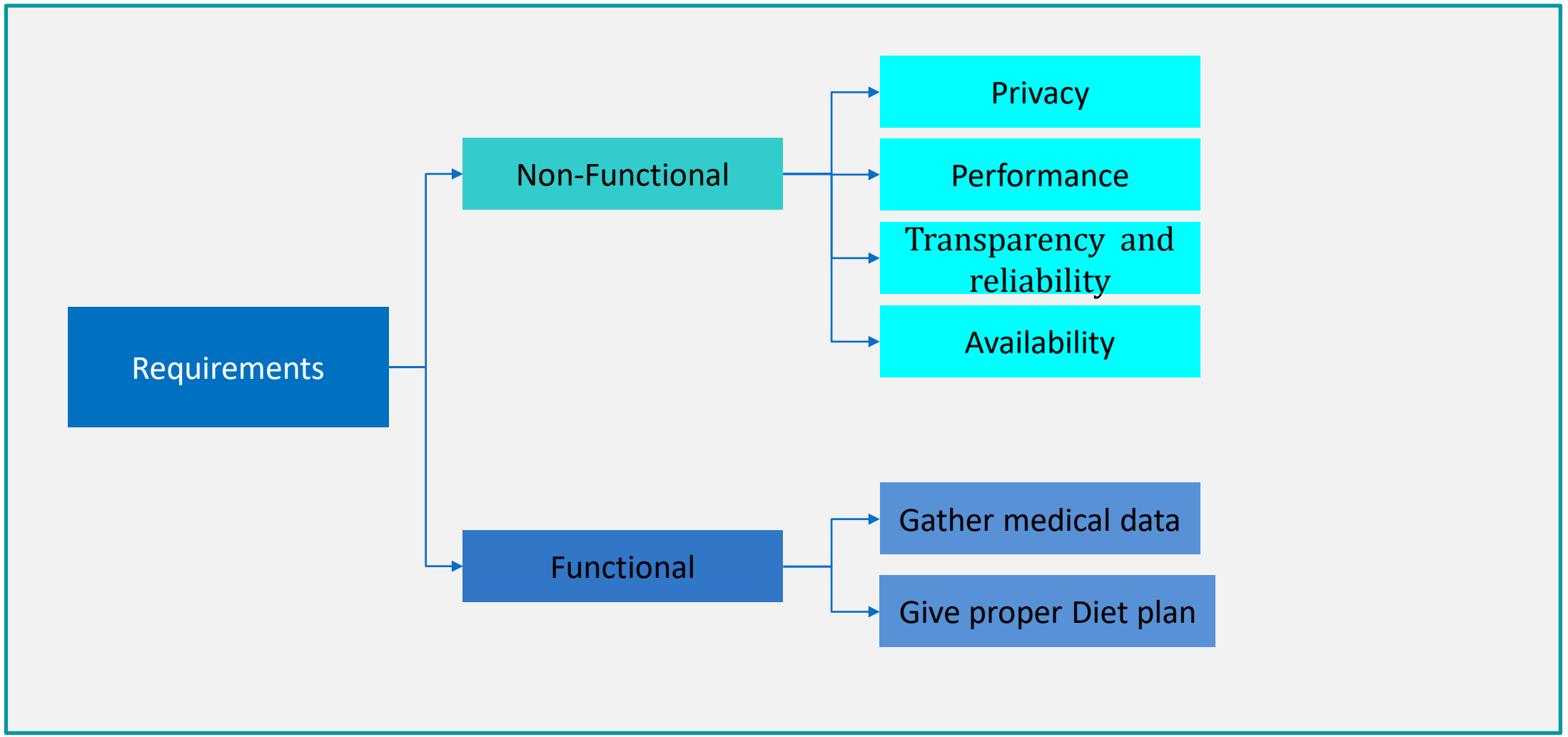
1. Natural Language Processing (NLP)

- Implementing the integration between the backend and frontend components of the mobile application.

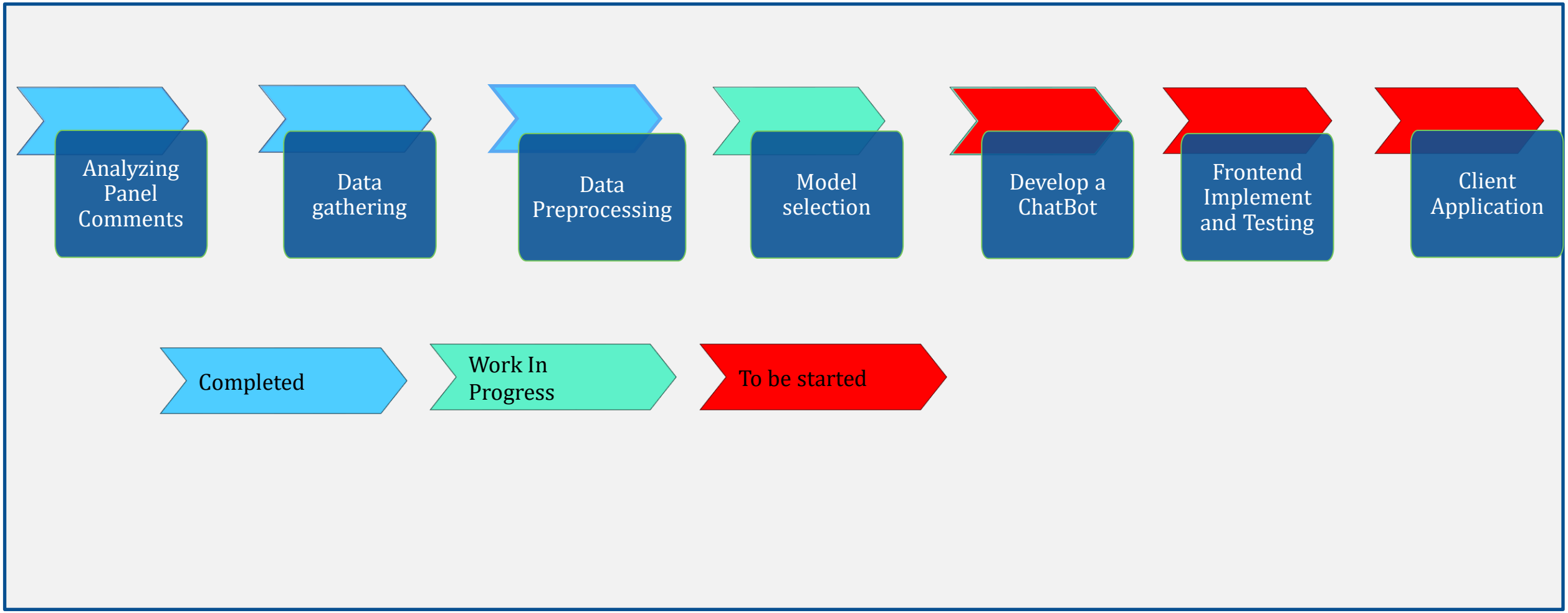
Tools and Technologies

- Python
- Pandas , NumPy
- TensorFlow
- MongoDB
- AWS
- NodeJS
- React Native

Requirements



Overall Progress



References

[10] N. Y. N. ,. R. R. S. S. P. K. Annapoorna B. A, "Prediction of chronic kidney disease and diet recommendation," IJARIIT, 2021.

[11] D. P. K. M.P.N.M. Wickramasinghe, "Dietary prediction for patients with Chronic Kidney Disease (CKD) by considering blood potassium level using machine learning algorithms," Research Gate, 2017.

[12] National Kidney Foundation," [Online]. Available: https://www.kidney.org/kidneydisease/howkidneyswrk.

IT20226596 | J.P.M.L. Perera

Specializing in Data Science

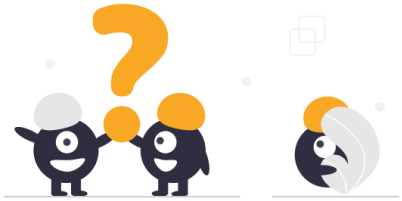
IoT & Machine Learning based Water quality Monitoring System for kidney patients

Background

- In Sri Lanka, prevalence of Kidney diseases (CKDu) have increased rapidly in recent decades.
- Most affected people are living in rural areas, and they have limited access to modern health care.
- Poor water quality has been identified as a possible contributor to kidney diseases.
- It is important to being aware of whether the daily used drinking water is harmful to kidneys or not.

Research Problem

- Lack of reliable and efficient real-time water quality monitoring system in Sri Lanka.
- Inadequate identification of sudden changes in water quality, impacting kidney patients' health.
- Absence of proper data management systems for recording, analyzing, and responding to water quality issues.
- Limited access to and understanding of water quality particularly for non-English speaking kidney patients.



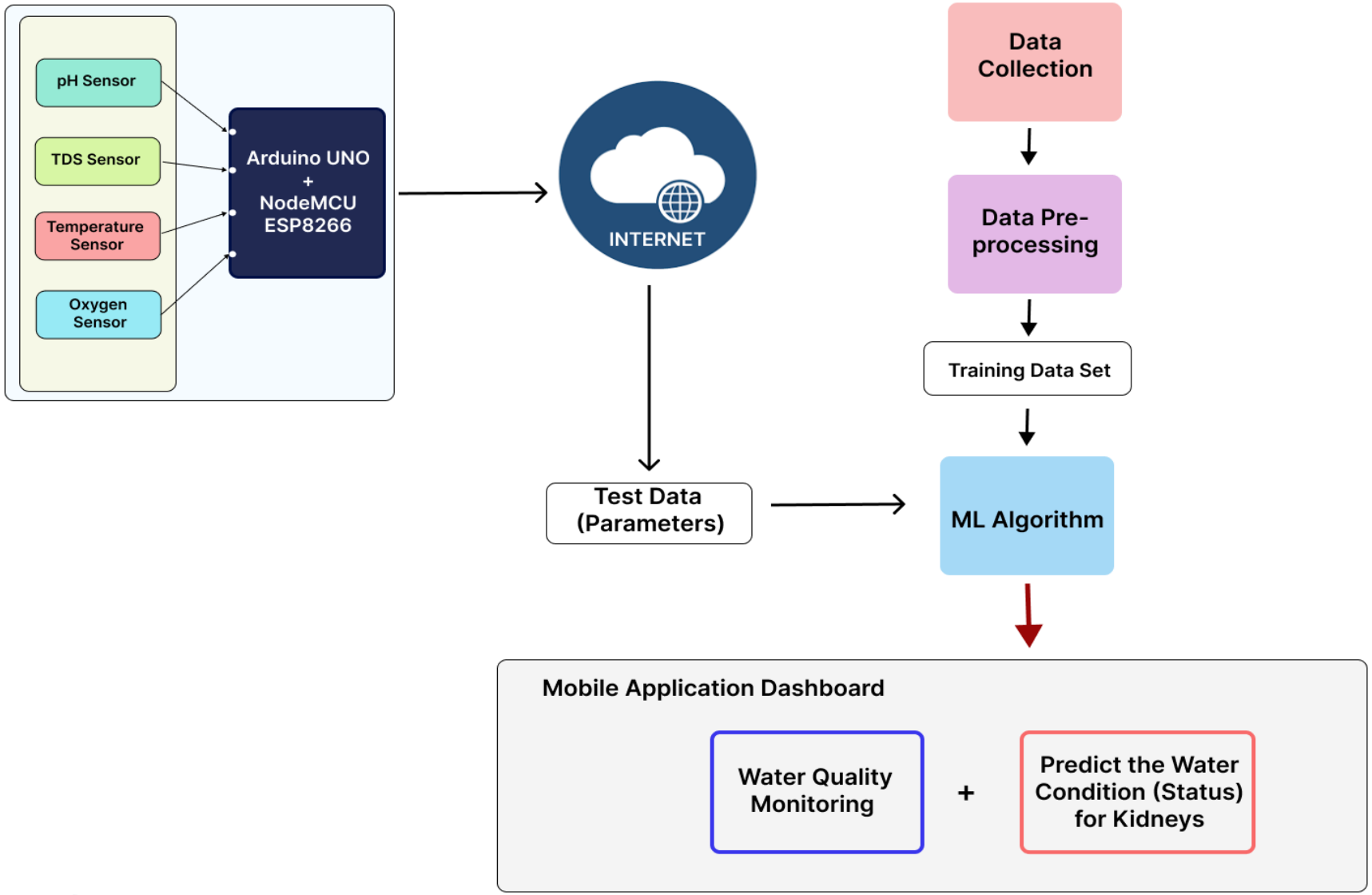
Research Gap

- The lack of a reliable and efficient real-time water quality monitoring system using IoT technology in Sri Lanka. Existing manual monitoring methods are unable to detect sudden changes in water quality.
- The absence of proper data management systems for recording, analyzing, and responding to water quality issues in Sri Lanka.
- Limited access to and understanding of water quality information, particularly for non-English speaking kidney patients in Sri Lanka.
- Insufficient integration of IoT and ML technologies for water quality monitoring and analysis in the context of Sri Lanka.

Novelty

- Implementing IoT technology specifically designed for monitoring water quality in Sri Lanka, which is currently absent or limited.
- Applying machine learning (ML) algorithms to analyze the collected water quality data and identify patterns, trends, and anomalies.
- Providing multilingual support to ensure that water quality information is available in local languages, improving accessibility and understanding for all users.
- Integrating IoT sensors and devices with ML algorithms to create a comprehensive solution for real-time monitoring, analysis, and management of water quality.

Component Overview



Completion of the Component

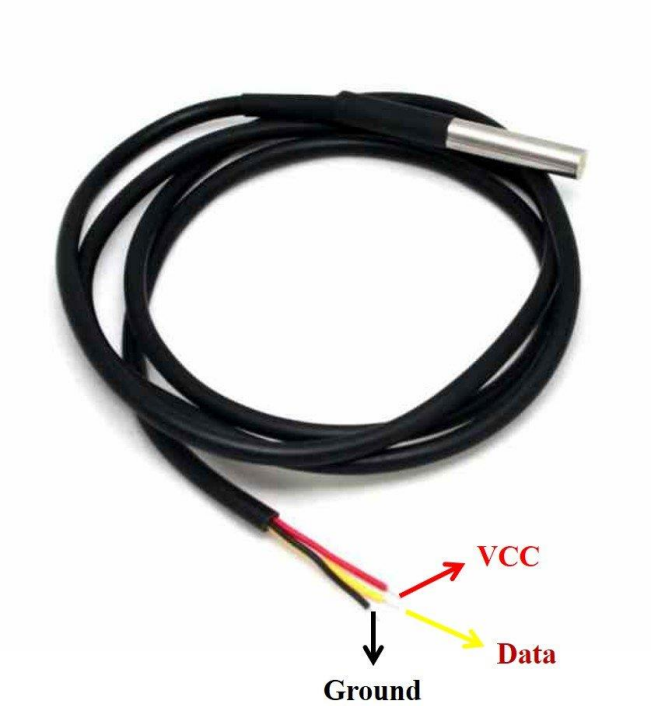
Hardware Solution – 50%

- Gathering Hardware Components
  - Arduino Uno Board
  - Sensors
  - Resistors
  - Jump wires
- Design the Circuits
- Programming the Arduino Board for test the Sensors
- Get values Separately from Sensors

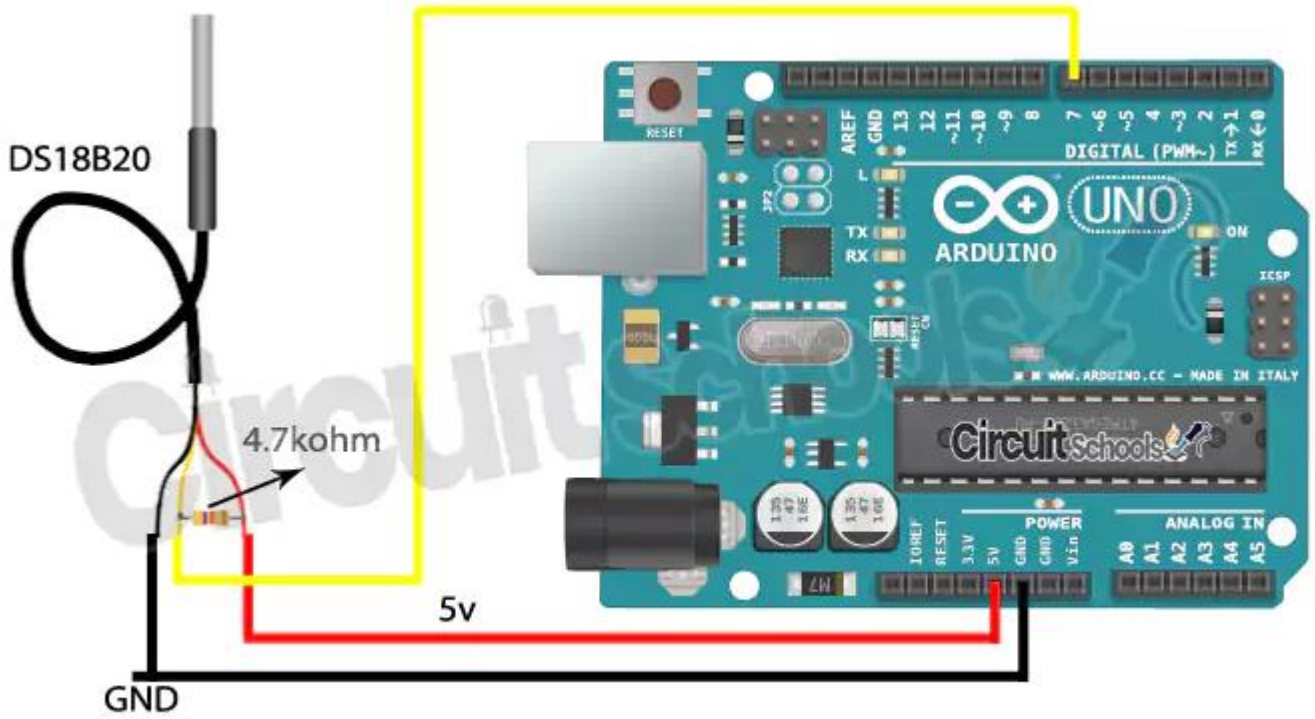
Implementation

Hardware Solution

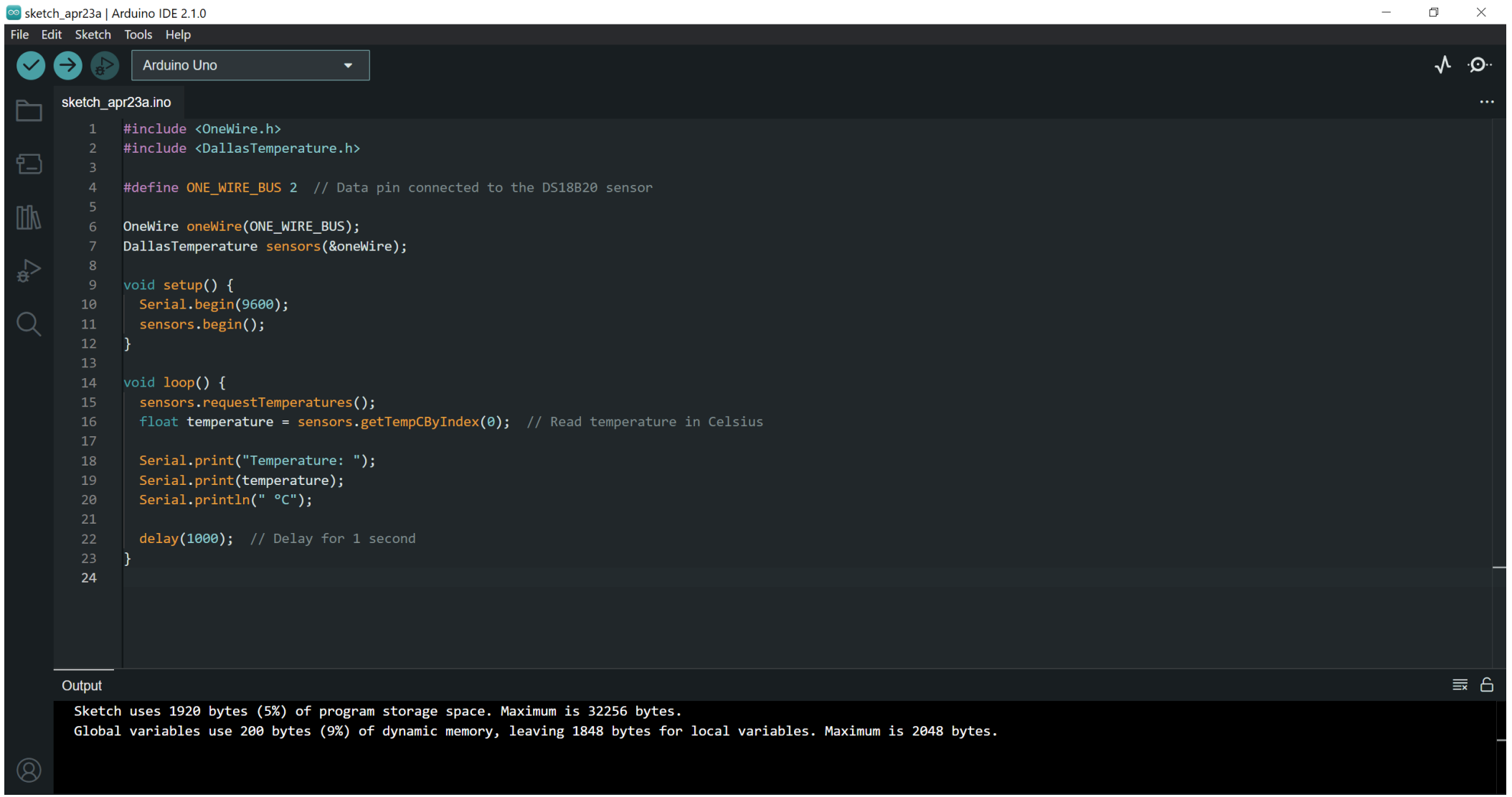
DS18B20 Waterproof Temperature Sensor



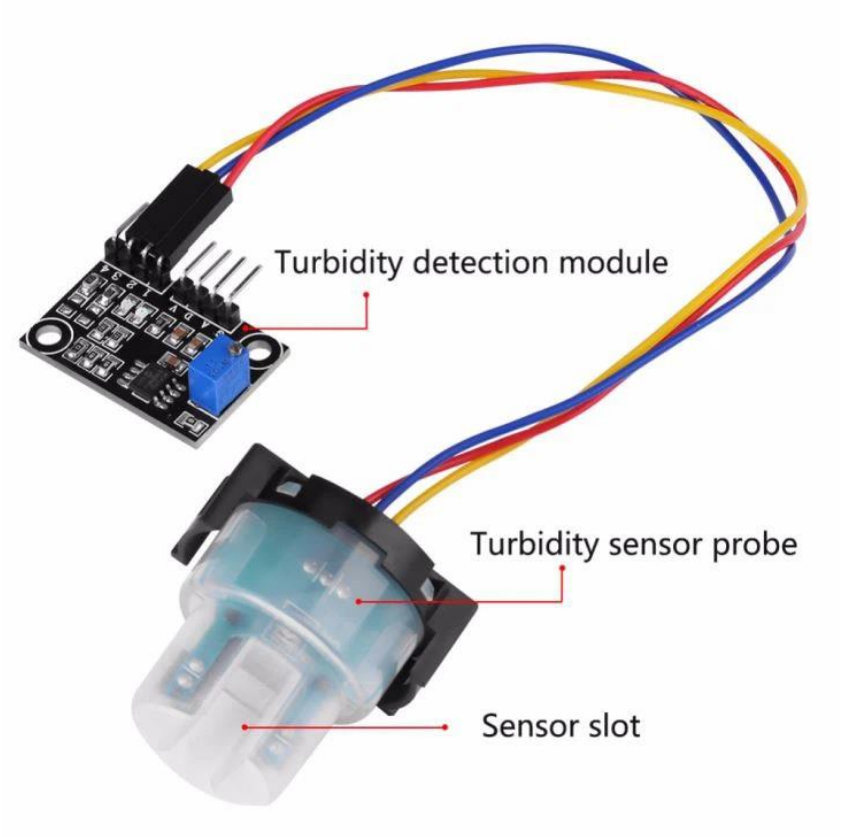
Circuit Diagram



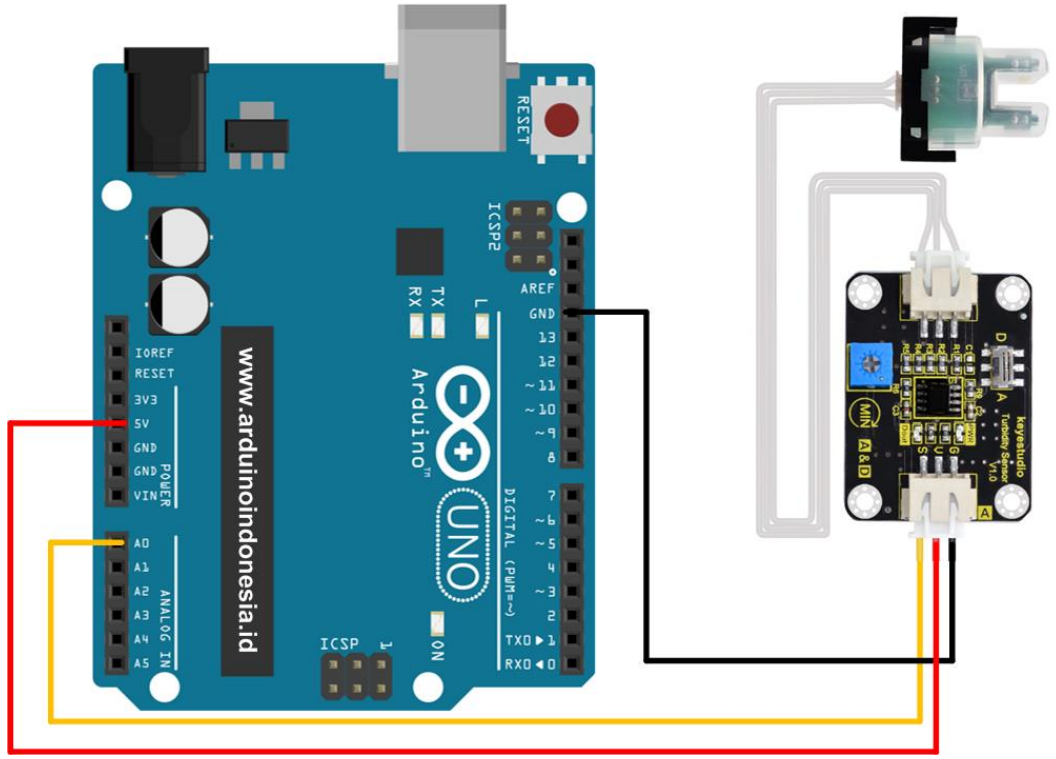
Config Temperature sensor in Arduino Uno



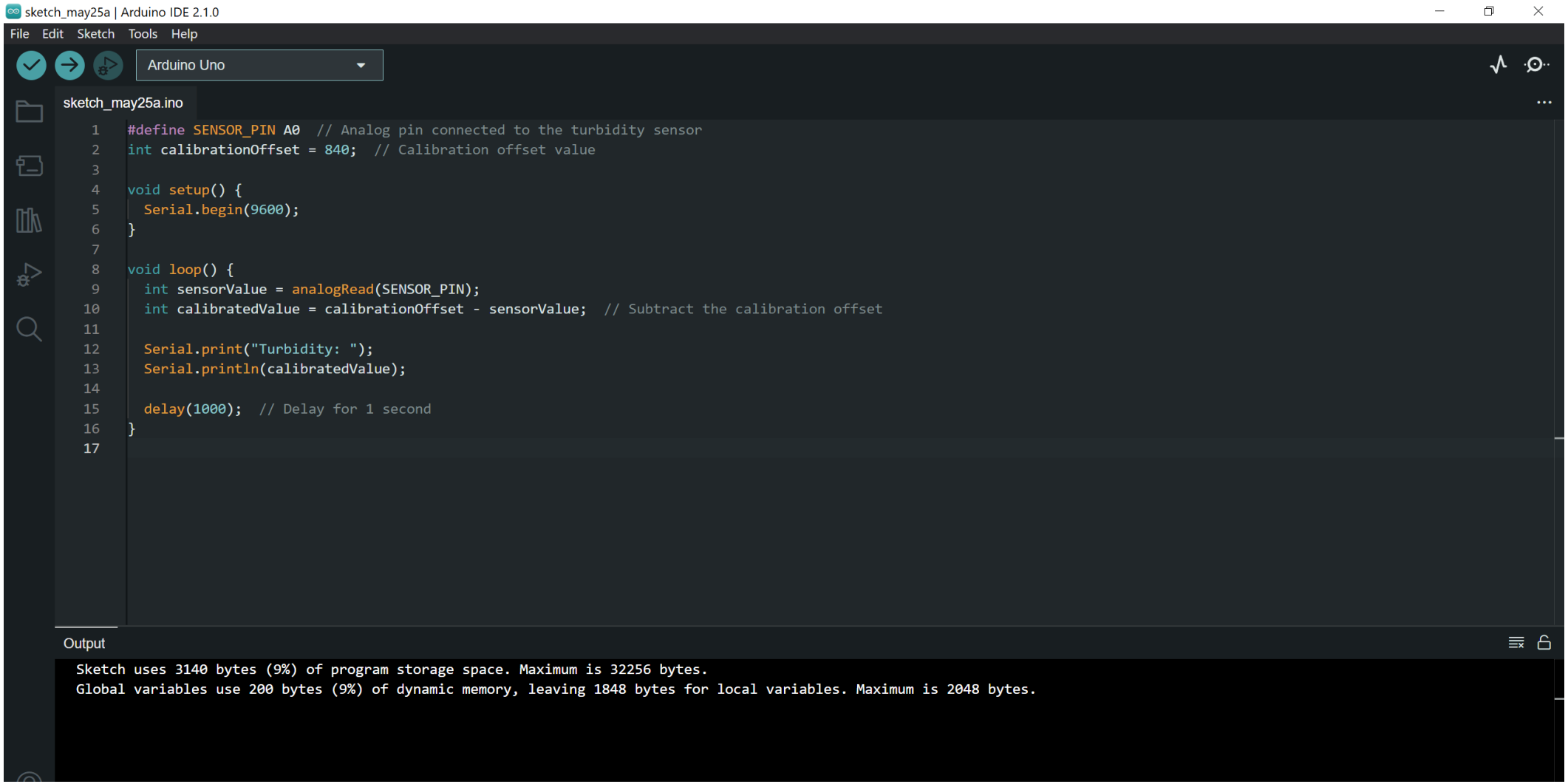
SENO189 Analog Turbidity Sensor



Circuit Diagram

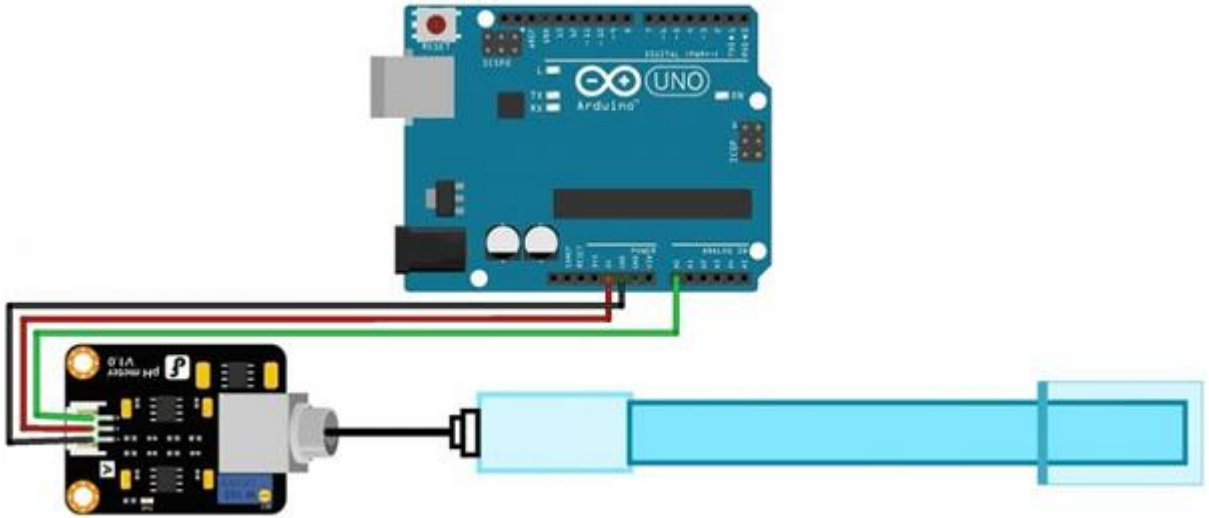


Config Turbidity sensor in Arduino Uno

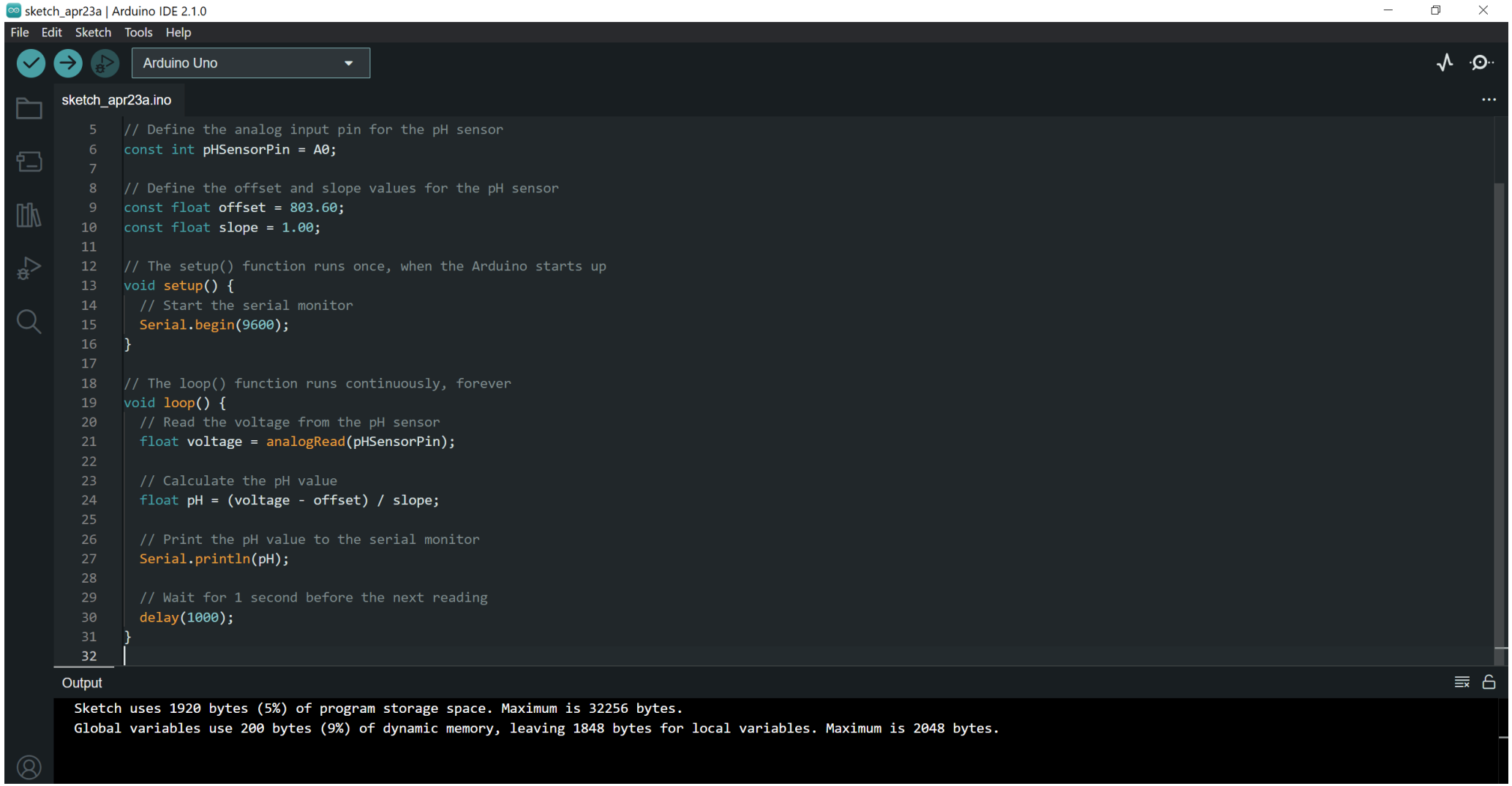


Analog pH Sensor with probe

Circuit Diagram



Config pH sensor in Arduino Uno



Completion of the Component

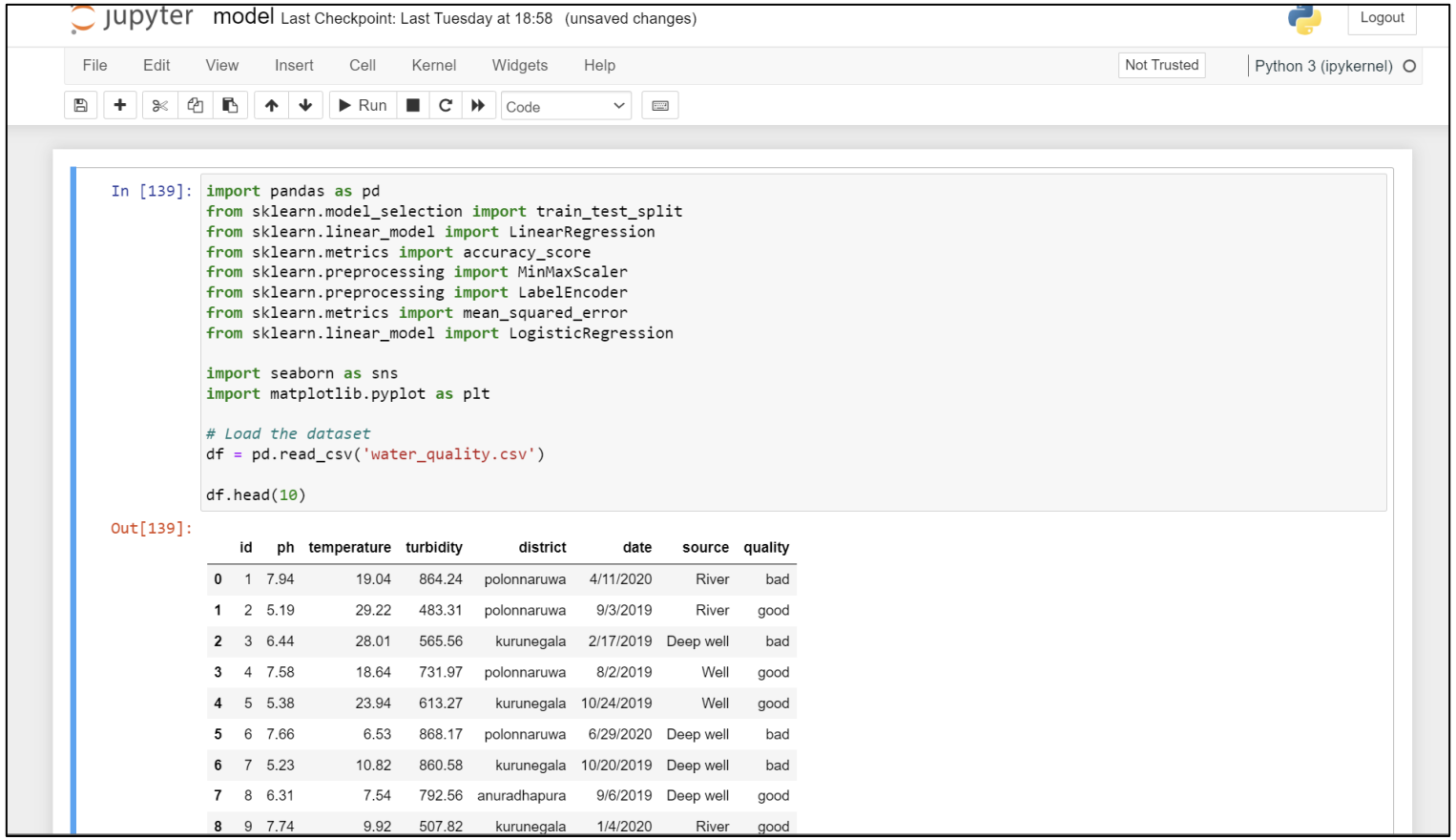
Software Solution – 50%

- Gathering Dataset from NIFS Kandy
- Analyse the Data set
- Pre-process the Data Set
- Build the Models and Train the Model
- Get the Accuracy

Implementation

Machine Learning (Software) Solution

- **Import Libraries & Load the Data set into Jupyter Notebook**



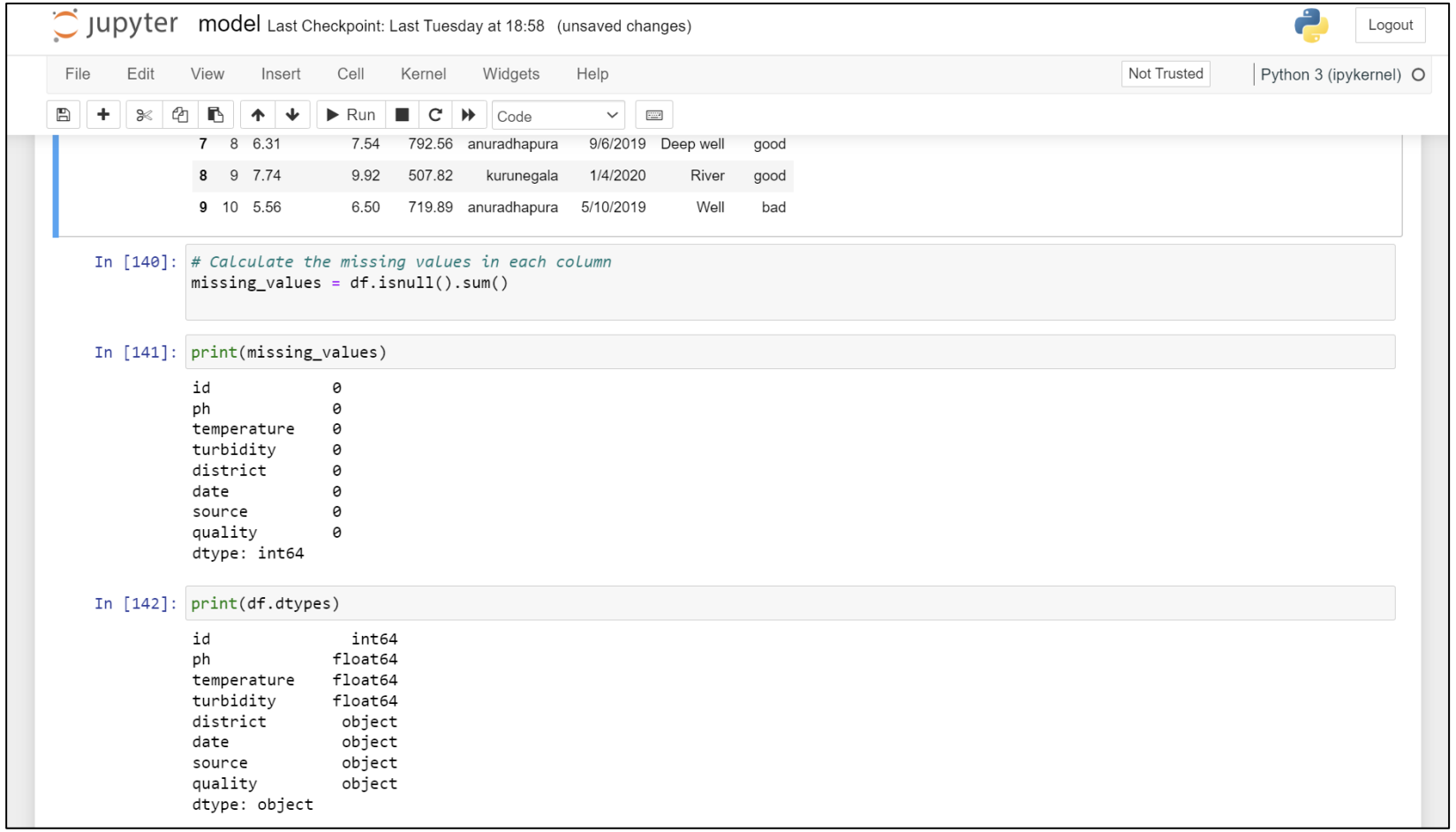
In [139]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LogisticRegression

import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset
df = pd.read_csv('water_quality.csv')

df.head(10)
```

Out[139]:

	id	ph	temperature	turbidity	district	date	source	quality
0	1	7.94	19.04	864.24	polonnaruwa	4/11/2020	River	bad
1	2	5.19	29.22	483.31	polonnaruwa	9/3/2019	River	good
2	3	6.44	28.01	565.56	kurunegala	2/17/2019	Deep well	bad
3	4	7.58	18.64	731.97	polonnaruwa	8/2/2019	Well	good
4	5	5.38	23.94	613.27	kurunegala	10/24/2019	Well	good
5	6	7.66	6.53	868.17	polonnaruwa	6/29/2020	Deep well	bad
6	7	5.23	10.82	860.58	kurunegala	10/20/2019	Deep well	bad
7	8	6.31	7.54	792.56	anuradhapura	9/6/2019	Deep well	good
8	9	7.74	9.92	507.82	kurunegala	1/4/2020	River	good

- **Checking Missing values & Data types**



- **Transform the Data types using Label Encoding**

jupyter model Last Checkpoint: Last Tuesday at 18:58 (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

In [143]:
```
# Create a LabelEncoder object
label_encoder = LabelEncoder()

# Encode the "gender" column
df['district_encoded'] = label_encoder.fit_transform(df['district'])
df['source_encoded'] = label_encoder.fit_transform(df['source'])

print(df.dtypes)
```

```
id                   int64
ph                 float64
temperature        float64
turbidity          float64
district            object
date                object
source              object
quality             object
district_encoded     int32
source_encoded       int32
dtype: object
```

In [144]:
```
# Drop specific columns
columns_to_drop = ['date','district']  # Replace with the actual column names you want to drop
df = df.drop(columns_to_drop, axis=1)
df.head()
```

Out[144]:

	id	ph	temperature	turbidity	source	quality	district_encoded	source_encoded
0	1	7.94	19.04	864.24	River	bad	2	1
1	2	5.19	29.22	483.31	River	good	2	1
2	3	6.44	28.01	565.56	Deep well	bad	1	0

- **Update the target variable and Split the data set into Train & Test datasets**

jupyter model Last Checkpoint: Last Tuesday at 18:58 (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted

Python 3 (ipykernel)

Run Code

In [149]:
```
X = df.drop('quality', axis=1)  # Update 'target_variable' with the actual column name
y = df['quality']
```

In [150]:
```
print(df.dtypes)
```

```
ph                 float64
temperature        float64
turbidity          float64
quality             object
district_encoded     int32
source_encoded       int32
dtype: object
```

In [151]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

- **Build the model, Train the dataset & Get the accuracy**

- **Logistic Regression**

```
In [154]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

          # Create a logistic regression model
          model = LogisticRegression()

          # Fit the model on the training data
          model.fit(X_train, y_train)

          # Predict the target variable for the test set
          y_pred = model.predict(X_test)

          # Calculate the accuracy of the model
          accuracy = accuracy_score(y_test, y_pred)

          # Print the accuracy
          print("Accuracy:", accuracy)

          Accuracy: 0.5217391304347826
```

- **Random Forest Classifier**

```
In [155]: from sklearn.ensemble import RandomForestClassifier

          # Split the data into training and test sets
          X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

          # Create a random forest classifier
          model = RandomForestClassifier(n_estimators=100, random_state=42)

          # Fit the model on the training data
          model.fit(X_train, y_train)

          # Predict the target variable for the test set
          y_pred = model.predict(X_test)

          # Calculate the accuracy of the model
          accuracy = accuracy_score(y_test, y_pred)

          # Print the accuracy
          print("Accuracy:", accuracy)

          Accuracy: 0.6521739130434783
```

What's Next ?

- Complete the IoT device
- Create the connectivity between device and mobile app
- Improve the ML model accuracy
- Give the Real time Analytical Quality Result of tested water sample through Our Mobile Application

References

[1] D. P. K. K. M.P.N.M. Wickramasinghe, "Dietary prediction for patients with Chronic Kidney Disease (CKD) by considering blood potassium level using machine learning algorithms," Research Gate, 2017.

[2] N. K. K. S. Moldobaeva Munara, "Recommending IoT based Real-time Water Quality Monitoring System in Malaysia," IEEE, 2022.

[3] A. J. N. B. J. D. P. K. Shashika Lokuliyana, "A Survey: IoT Enable Framework for Water Quality Measurement and Distribution," IEEE, 2018.

Commercialization

- Partner with healthcare providers to offer our app as a resource for their patients.

- Collect data from app users to identify patterns and trends in kidney disease prevalence and risk factors. Use this data to improve the app's features and functionality and to develop targeted marketing campaigns.

2023-04-29 At Kurunegala Teaching Hospital | Consultant Nephrologist Dr. Premil Rajakrishnan

SRI LANKA
INSTITUTE OF INFORMATION TECHNOLOGY
16th Floor, BoC Merchant Tower, No. 28, St. Michael's Road, Colombo 03

Date:28/04/2023 Your Ref : My Ref :2023-032

Dr. Premil Rajakrishna,
Consultant Nephrologist,
Teaching Hospital,
Kurunegala.

Dear Sir,

Certifying the project titled "KidniFy - A mobile based Chronic kidney Disease Patient care System Using ML and IoT" is conducting as a BSc in IT final year research project.

The Sri Lanka Institute of Information Technology (SLIIT) is the largest Degree Awarding Institute in the field of information Technology recognized by the University Grants Commission under the Universities Act. It was established in the year 1999 to educate and train Information Technology (IT) Professionals required by the fast-growing IT Industry in Sri Lanka.

This letter is to certify that the following students.
IT20154226 - M.M.K.L.Marasinghe
IT20226596 - J.P.M.L. Perera
IT20235260 - D.R.N. Samarawila
IT20785120 - W.B.M.A. Isurika

They are final year undergraduate students who conduct research entitled "KidniFy - A mobile based Chronic kidney Disease Patient care System Using ML and IoT" as partial fulfillment of the B.Sc. in Information Technology degree at Sri Lanka Institute of Information Technology (SLIIT). The students are conducting the research under the supervision of Ms. Wishalya Vanshanee Tissera

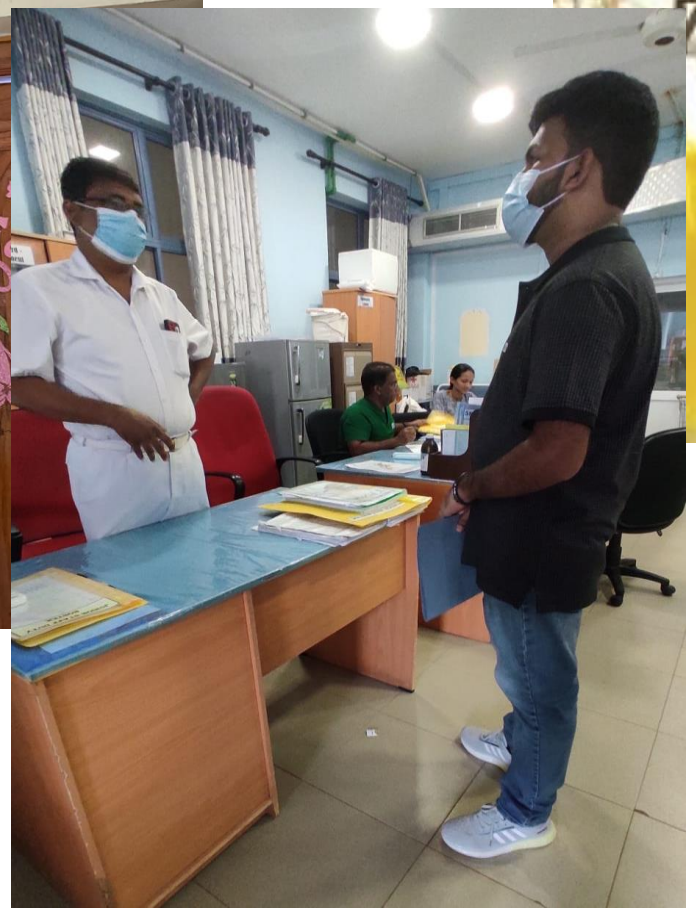
I kindly request your assistance in enabling these students to collect data from your organization to build their dataset for the research project. If you have any questions or require further clarification about the project, please do not hesitate to contact me.

Thank you for your cooperation

Dr. Jayantha Amararachchi
Assistant Professor/
Research Project Coordinator,
jayantha.a@sliit.lk
+94 11 754 4103

Tel: +94(0)11 2301904 - 5 Fax: +94(0)11 2301906 E-mail: info@sliit.lk URL: www.sliit.lk

2023-05-01 At Anuradhapura Teaching Hospital | Kidney Dialysis Unit



Thank You!

Q & A