



**A  
MINOR PROJECT REPORT  
ON**

**“AGROTECHGUIDE”**

**Submitted by:**

**TRIBHUVAN UNIVERSITY  
INSTITUTE OF ENGINEERING  
PACHIMANCHAL CAMPUS**

**Ankit Sapkota (PAS077BEI005)**

**Navraj Awasthi (PAS077BEI026)**

**Yogendra Baskota (PAS077BEI045)**

**Submitted to:**

**Department of Electronics and Computer Engineering  
Paschimanchal Campus, Institute of Engineering,  
Tribhuvan University  
Pokhara, Nepal**

**March, 2024**



## **PAGE OF APPROVAL**

**TRIBHUVAN UNIVERSITY**  
**INSTITUTE OF ENGINEERING**  
**PASHCHIMANCHAL CAMPUS**  
DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

The undersigned certify that they have read, and recommended to the Institute of Engineering for acceptance, a project report entitled “**AgrotechGuide**” submitted by Ankit Sapkota, Navraj Awasthi and Yogendra Baskota in partial fulfillment of the requirements for the Bachelor’s degree in Electronics, Communication and Information engineering.

---

Supervisor: Asst. Prof. Smita Adhikari  
Department of Electronics and Computer Engineering

---

Asst. Prof. Khem Raj Koirala  
Head of Department  
Department of Electronics and Computer Engineering

## **COPYRIGHT**

The author has agreed that the library, Pashchimanchal Campus, may make this report freely available for inspection. Moreover, the author has agreed that permission for extensive copying of this project report for scholarly purpose may be granted by the lecturers, who supervised the project works recorded herein or, in their absence, by the Head of Department wherein the project report was done. It is understood that the recognition will be given to the author of the report and to the Department of Computer and Electronics, Pashchimanchal Campus in any use of the material of this project report. Copying or publication or other use of this report for financial gain without approval of the Department and author's written permission is prohibited. Request for permission to copy or to make any other use of the material in this report in whole or in part should be addressed to:

Head of Department  
Department of Electronics and Computer Engineering  
Pashchimanchal Campus, Institute of Engineering  
Pokhara-16 Lamachaur  
Nepal

## **ACKNOWLEDGEMENT**

It gives us immense pleasure to express our deep sense of gratitude to our supervisor Asst. Prof. Smita Adhikari for her invaluable guidance, motivation, constant inspiration and above all for her ever co-operating attitude that enabled us in bringing up this project in the present form. We solely take the responsibility of any possible mistakes that may have occurred in preparing this report and we would like to welcome comments and queries during the submission of this report. Finally, we would like to thank all the staffs of Institute of Engineering Pashchimanchal Campus who helped us throughout our project.

## **ABSTRACT**

AgrotechGuide recommends the high-yielding crop tailored to the farmland's environmental conditions. Leveraging soil and environmental data analysis, it provides recommendations for the most suitable crop to grow. The system's machine learning model is trained on a dataset comprising 2200 labeled land profiles, each detailing its specific parameters and the most productive crop cultivated in that area. Users interact with the system through a user-friendly frontend developed on the Flutter framework. Userprovided land data, along with the hardware-provided environmental data, is fed to the model, which processes it to suggest the optimal crop choice. Furthermore, the application offers valuable cultivation guidance for the recommended crop. AgrotechGuide represents an innovative application of machine learning in crop recommendation. The future prospects of this project include training the model based on the collected user input to precisely tailor it to the Nepalese agricultural landscape.

## Table of Contents

PAGE OF APPROVAL .....	i
COPYRIGHT .....	ii
ACKNOWLEDGEMENT .....	iii
ABSTRACT .....	iv
Table of Contents.....	v
LIST OF FIGURE .....	vi
1. INTRODUCTION.....	1
Figure 1: BreadBoard .....	4
Figure 2: Jumper wire .....	5
3. LM35 Temperature Sensor.....	5
2. LITERATURE REVIEW .....	8
3. METHODOLOGY .....	9
WORKING: .....	10
Figure 6: Model Training .....	11
Figure 7: Flowchart .....	12
Figure 8: Hardware Implementation .....	13
4. Results and Discussion.....	25
4.2 Limitations: .....	31
5. Conclusion: .....	33
6. REFERENCES.....	34

## LIST OF FIGURE

Figure 1: BreadBoard .....	10
Figure 2: Jumper wire .....	11
Figure 3: LM35 .....	12
Figure 4: DHT11 .....	12
Figure 5: ESP8266 NODEMCU .....	13
Figure 6: Flowchart .....	19
Figure 7: Model Training .....	18
Figure 8: Hardware Implementation .....	20



# **1. INTRODUCTION**

## **1.1 Background**

In the current context, where national emphasis is placed on the production of agricultural goods within the country, the optimization and enhancement of production have become important element of study. While being centered around production, we can't ignore the fact that the geographical diversity within our country prevents the dominance of limited crops throughout the nation; apples thrive in the mountains, while mangoes flourish in the Terai region. Coffee finds its home in the west, whereas tea prevails in the east. Varied soil compositions and climatic conditions support a abundant varieties of crops across different regions. Achieving high yields in agricultural production necessitates the cultivation of a diverse range of crops suited to each specific locale. Thus, the identification of the most suitable crop for a given location becomes pivotal to meet the production potential.

In addressing this critical need for accurate crop selection tailored to specific locations, AgrotechGuide emerges as an important project.

## **1.2 Problem Statement**

The agricultural sector faces significant challenges in production optimization due to the inappropriate selection of crops. Due to the lack of an comprehensive crop selection mechanism, farmers are unaware about the potential cash crop that can grow in their farm. This results in suboptimal yields and hampers the overall productivity of the sector. A lack of comprehensive crop recommendation system in the geography diverse country like Nepal means an untapped economic potential.

## **1.3 Objectives**

The objective of AgrotechGuide is to recommend high-yielding crops to farmers based on their specific farm conditions, consequently maximizing productivity.

## **1.4 Application / Scope**

This project is purely intended for educational and research purpose and has commercial application. It could potentially refer to a specific product, service, or platform related to agriculture

## **1.5 System Requirement**

System Requirements are essential for the system to work efficiently.

### **1.5.1 Software Requirement**

Software Requirements of the projects are as follows:

#### **1. Flutter**

In the context of mobile app development, "Flutter" is an open-source UI software development toolkit created by Google. It enables developers to build natively compiled applications for mobile, web, and desktop platforms using a single codebase written in the Dart programming language. Flutter is known for its expressive and flexible user interface, allowing developers to create visually appealing and high-performance applications across various devices. The framework has gained popularity for its ease of use, hot reload feature for rapid development, and the ability to produce aesthetically consistent designs across different platforms. Flutter Consists Of Two Important Parts.

- An SDK (Software Development Kit): A collection of tools that are going to help you develop your applications. This includes tools to compile your code into native machine code (code for iOS and Android).
- A Framework (UI Library based on widgets): A collection of reusable UI elements (buttons, text inputs, sliders, and so on) that you can personalize for your own needs.

## 2. Dart

Dart is an open-source, general-purpose, object-oriented programming language with Cstyle syntax developed by Google in 2011. The purpose of Dart programming is to create a frontend user interface for the web and mobile apps. It is under active development, compiled to native machine code for building mobile apps, inspired by other programming languages such as Java, JavaScript, C#. Since Dart is a compiled language so you cannot execute your code directly; instead, the compiler parses it and transfers it into machine code. It supports most of the common concepts of programming languages like classes, interfaces, functions, unlike other programming languages.

## 3. Arduino IDE platform for NODEMCU

Arduino IDE is a powerful platform for embedded platforms which is rich in libraries so that coding microcontrollers is easier. Similar to many other libraries, libraries for nodeMCU can also be imported in this IDE such that programming the device becomes easier and more efficient.

## 4. Python

Python is a versatile, easy-to-read programming language known for its simplicity and extensive libraries. It supports multiple programming styles and is widely used across various domains, including web development, data analysis, and artificial intelligence.

### a. Scikit Learn

Scikit-learn is a powerful Python library for machine learning built on top of NumPy, SciPy, and matplotlib. It provides simple and efficient tools for data mining and data analysis, offering a wide range of algorithms for classification, regression, clustering, dimensionality reduction, and more. With its user-friendly interface and extensive documentation, scikit-learn is widely used in academia and industry for developing machine learning models.

### 1.5.2 Hardware Requirement

Hardware Requirements of the projects are:

#### 1. Breadboard

A breadboard is a solderless device for temporary prototype with electronics and test circuit designs. Most electronic components in electronic circuits can be interconnected by inserting their leads or terminals into the holes and then making connections through wires where appropriate.

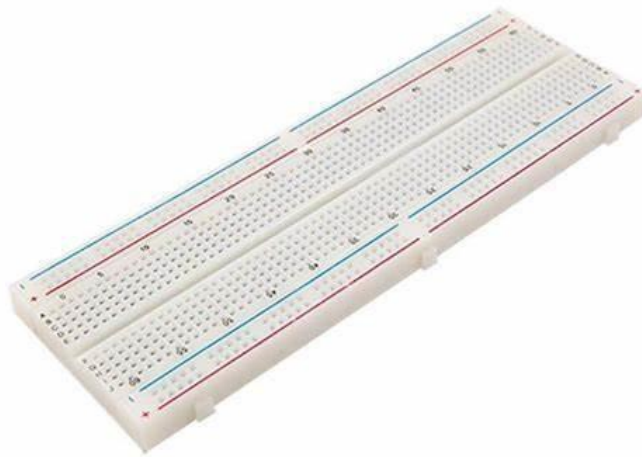


Figure 1: BreadBoard

## 2. Jumper wire

Jumper Wires are electrical wires with connector pins at each end. They are used to connect two points in a circuit without Soldering. You can use jumper wires to modify a circuit or diagnose problems in a circuit.

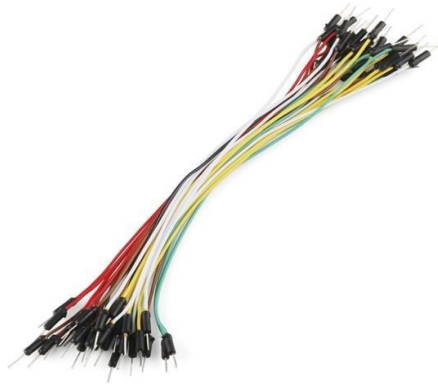


Figure 2: Jumper wire

## 3. LM35 Temperature Sensor

The LM35 is a temperature sensor integrated circuit that is commonly used to measure the ambient temperature of its surroundings. The primary task of the LM35 is to sense the temperature of its environment. It converts the surrounding temperature into a proportional analog voltage. The sensor has a linear output, with a 10 mV change in voltage per degree Celsius change in temperature.

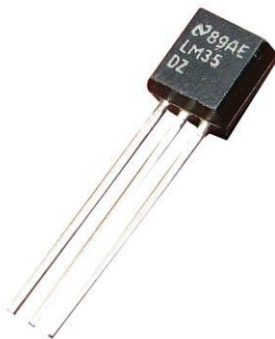


Figure 3: LM35

## 5.Humidity Sensor

A humidity sensor (DHT11) is like a small gadget that helps us understand how much moisture is in the air around us. It measures humidity, which is how much water vapor is in the air. Imagine it as a tiny device that can feel and tell us whether the air is dry or damp. It does this by sensing changes in the air that happen when it gets wet or dry. Humidity sensors are handy in various places, like in weather stations to predict rain, or in our homes to make sure the air isn't too humid or too dry, creating a more comfortable environment for us.

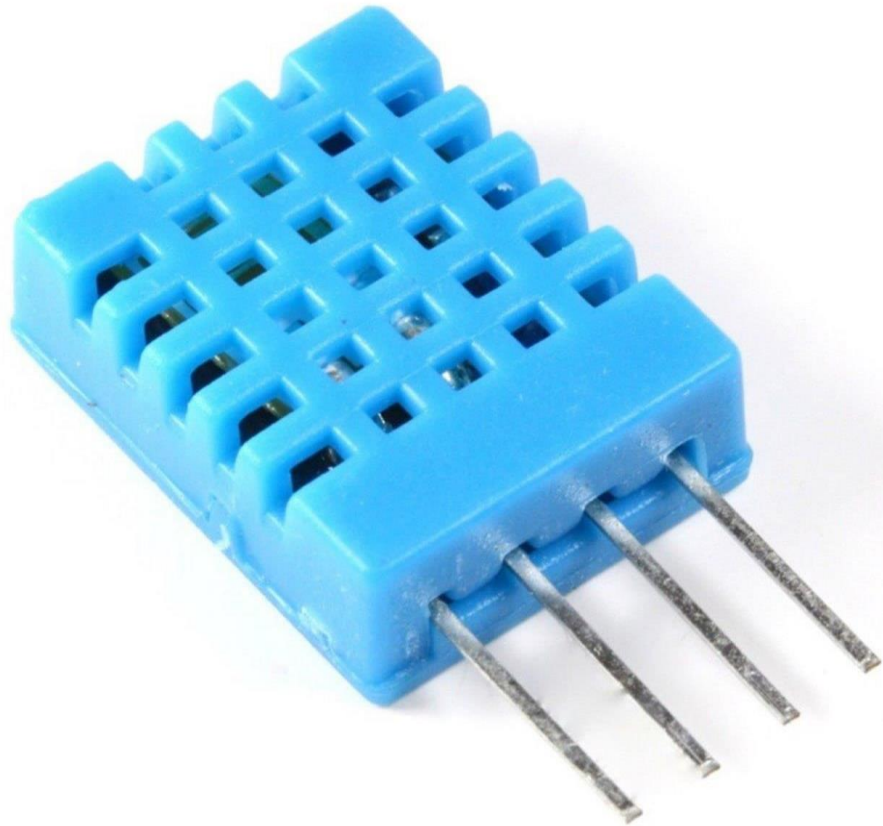
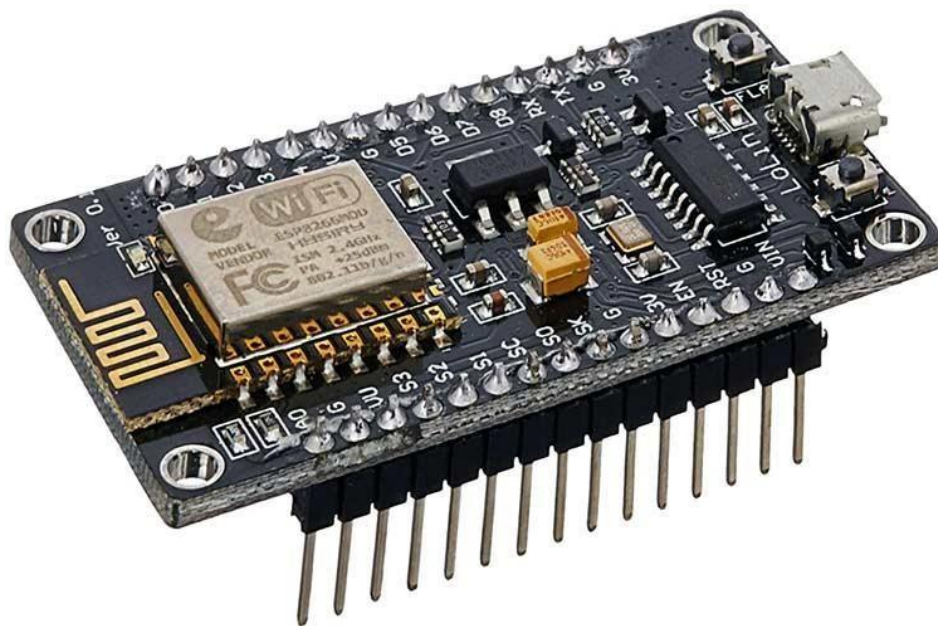


Figure 4: DHT11

## 6.The ESP8266 WIFI Module

The ESP8266 WIFI Module is a small electronic device that adds Wi-Fi capability to other electronics. It's like a magic chip that gives ordinary things, like lights or sensors, the power to connect to the internet. With this chip, gadgets can send and receive information over Wi-Fi, allowing them to communicate with each other or with you through your phone. It's commonly used in projects where you want devices to be connected and smart. The ESP8266 is like a tiny Wi-Fi superhero that makes everyday things smarter and able to share information wirelessly.



*Figure 5: ESP8266 NODEMCU*

## 2. LITERATURE REVIEW

- [1] Devkota, S. (2021, September). Prime Minister Agriculture Modernization Project (PMAMP), Nepal.

This research paper presents the Prime Minister Agriculture Modernization Project (PMAMP) in Nepal. PMAMP is a 10-year initiative with a budget of NPR 130 billion aimed at transforming traditional farming practices into modern, sustainable ones. It addresses challenges like labor shortages and rising costs while focusing on creating specialized crop regions, enhancing competitiveness, and generating employment. Farmers anticipate improved infrastructure, technology access, and increased productivity, leading to self-sufficiency in major crops. PMAMP emphasizes scientific land use and modern techniques adoption, offering hope for a prosperous and sustainable agricultural future in Nepal.

- [2] Patel, M., & Rane, A. (2023, April). Crop Recommendation System.

This research paper aims to help farmers worldwide by using machine learning to predict the best crops for their specific conditions, considering factors like soil nutrients, pH levels, humidity, and rainfall patterns. By employing user-friendly algorithms, the study provides accurate crop recommendations to optimize farming practices and increase productivity. Through precision farming techniques and sustainable approaches, farmers can mitigate risks and secure their livelihoods for the future.

- [3] Shrestha, M., Shrestha, M., Khanal, S., & Khanal, S. (2020, September). Future prospects of precision agriculture in Nepal. Archives of Agriculture and Environmental Science.

This research paper explores precision agriculture, a modern approach using technology to optimize farming practices for increased productivity and sustainability. It focuses on precise input application to reduce costs and risks while maximizing profitability, employing tools like GPS and remote sensors. Despite Nepal's agricultural importance, current practices underutilize resources. Precision agriculture offers potential to boost productivity and conserve the environment, but successful adoption requires technical expertise. This study examines precision agriculture's history, scope, and challenges in Nepal.



### 3. METHODOLOGY

The development of the crop recommendation system, AgrotechGuide, involved a systematic approach that encompassed various stages and considerations. The methodology followed for the project can be outlined as follows:

**3.1 Needs Assessment:** The first step was to conduct a comprehensive needs assessment to identify the key challenges and requirements related to crop selection. This involved reviewing existing literature, conducting surveys or interviews with potential users, and consulting agriculture professionals to gather insights and understand the target user's needs.

**3.2 Design and Conceptualization:** Based on the needs assessment, a conceptual design of AgrotechGuide was created. This involved defining the system's features, functionalities, and user interface. Collaborative brainstorming sessions and iterative design processes were employed to refine the concept and ensure that it aligned with the identified user needs.

**3.3 Technological Development:** Once the conceptual design was finalized, the technological development of AgrotechGuide commenced. This stage involved hardware and software development, incorporating the necessary sensors, data processing model, and user interface. Prototypes were built and tested to ensure the feasibility and functionality of the System.

**3.4 User Testing and Feedback:** User testing played a crucial role in evaluating the usability and effectiveness of AgrotechGuide, feedback from testing was used to identify areas for improvement and refine the design and functionality of the system.

**3.5 Validation and Performance Evaluation:** The developed AgrotechGuide prototype was subjected to validation and performance evaluations. This involved testing the device with preprocessed data, measuring its accuracy in crop recommendation, assessing the reliability of the model, and evaluating user satisfaction.

**3.6 Documentation and Reporting:** Finally, the methodology, findings, and outcomes of the project were documented in a comprehensive report. This included detailed

descriptions of the design process, development stages, and recommendations for future enhancements.

By following this methodology, the development of AgrotechGuide ensured a systematic and iterative approach to designing and creating an effective crop recommendation system. The methodology allowed for user-centered design, integration of advanced technologies, and rigorous testing and evaluation, ultimately leading to a functional prototype that addresses the challenges of crop selection.

## WORKING:

The operation of AgrotechGuide can be divided into three distinct stages:

### **Input:**

AgrotechGuide recommends crops based on user input, which comprises soil and environmental data. Environmental data is captured using hardware sensors, while soil data is provided directly by the user.

Environmental data collection: Two sensors LM35 for temperature and DHT11 for humidity is used to collect the environmental data. Data is feed via a ESP8266nodemcu.

Soil data collection: It is directly given by the user via User Interface.

### **Model Development:**

This segment is the core of our system development. It comprises of training data preparation and model training.

Data preparation: Training data is prepared in this phase. Dataset is checked for for various conditions like null value and suitable data is substituted in case of such. This phase is important for the proper training of model.

## Model Training:

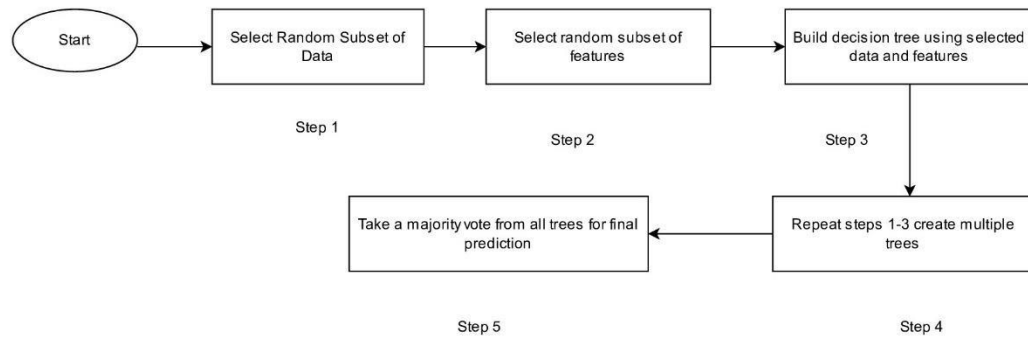


Figure 6: Model Training

We employ the Random Forest Algorithm to construct the data processing model. This algorithm generates ensembles of decision trees by randomly selecting subsets of parameters and building trees based on maximum gain achieved with the training data subsets. Furthermore, model hyperparameters are fine-tuned to enhance accuracy.

Subsequently, the model processes the input values from the frontend and returns the recommended crop based on those values.

## Output:

The generated output is transmitted back to the frontend application, where it is displayed to the user. In addition to crop recommendations, the application also provides essential cultivation information for the recommended crops.

## FLOWCHART:

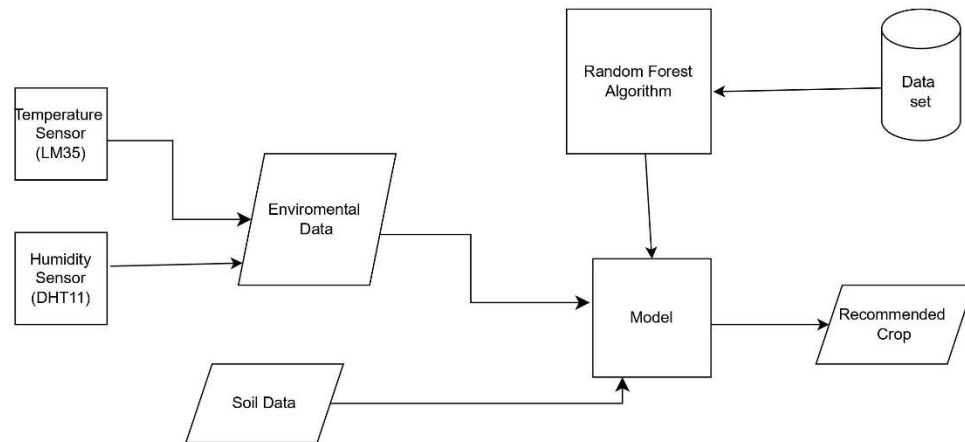


Figure 7: Flowchart

### 3.8 Hardware design:

The required components are ESP8266 Nodemcu, Temperature Sensor(LM35), Humidity Sensor(DHT11).

The simple hardware assembling can be shown as :

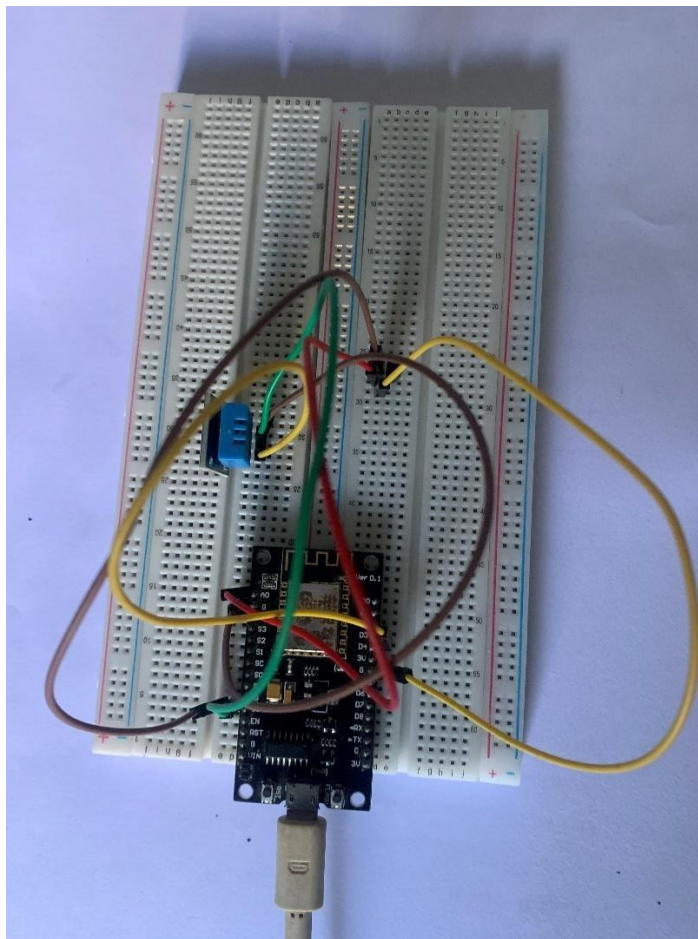


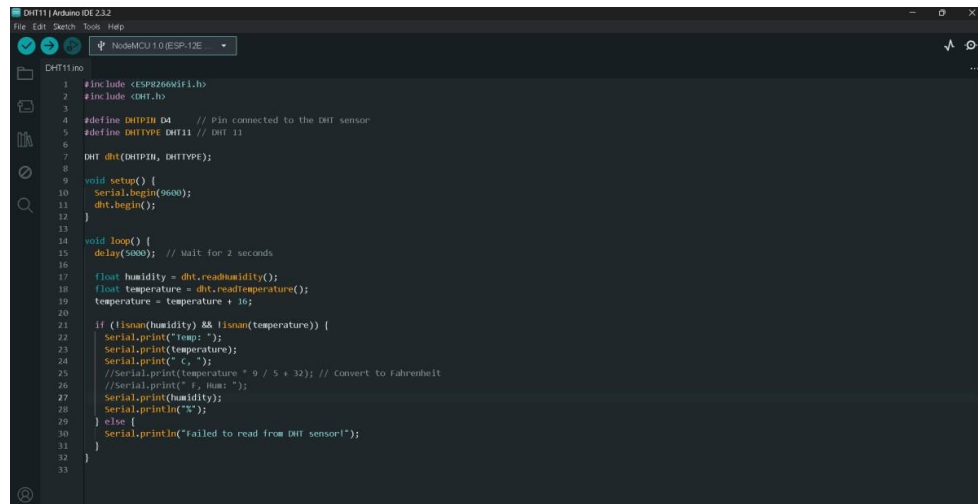
Figure 8: Hardware Implementation

## 3.9 Software implementation

### 3.9.1 Hardware

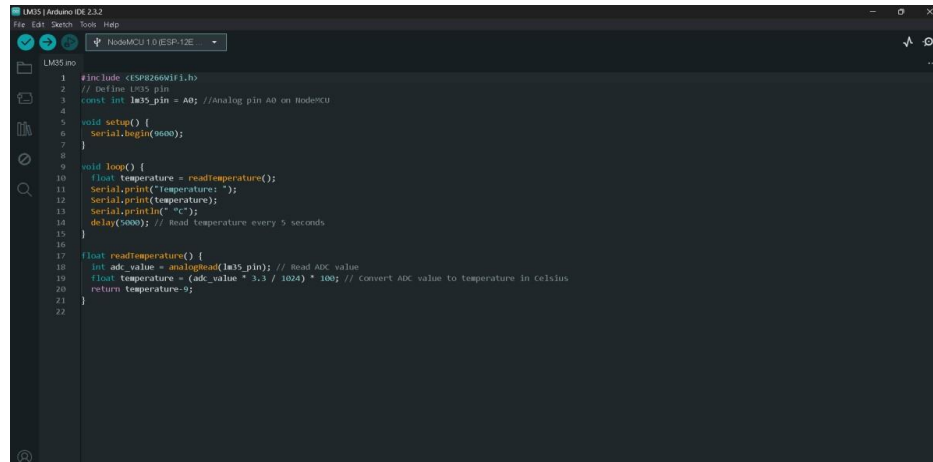
The software for the project is written through the Arduino Ide which consist of several parts . Some of the software test performed are :

Arduino IDE code for Humidity Sensor(DHT11):



```
DHT11.ino
1 #include <ESP8266WiFi.h>
2 #include <DHT.h>
3
4 #define DHTPIN D4 // Pin connected to the DHT sensor
5 #define DHTTYPE DHT11 // DHT 11
6
7 DHT dht(DHTPIN, DHTTYPE);
8
9 void setup() {
10   Serial.begin(9600);
11   dht.begin();
12 }
13
14 void loop() {
15   delay(5000); // Wait for 2 seconds
16
17   float humidity = dht.readHumidity();
18   float temperature = dht.readTemperature();
19   temperature = temperature + 32;
20
21   if (isnan(humidity) && isnan(temperature)) {
22     Serial.print("temp: ");
23     Serial.print(temperature);
24     Serial.print(" C. ");
25     //Serial.print(temperature * 9 / 5 + 32); // Convert to Fahrenheit
26     //Serial.print(" F. Hum: ");
27     Serial.print(humidity);
28     Serial.println("K");
29   } else {
30     Serial.println("failed to read from DHT sensor!");
31   }
32 }
33
```

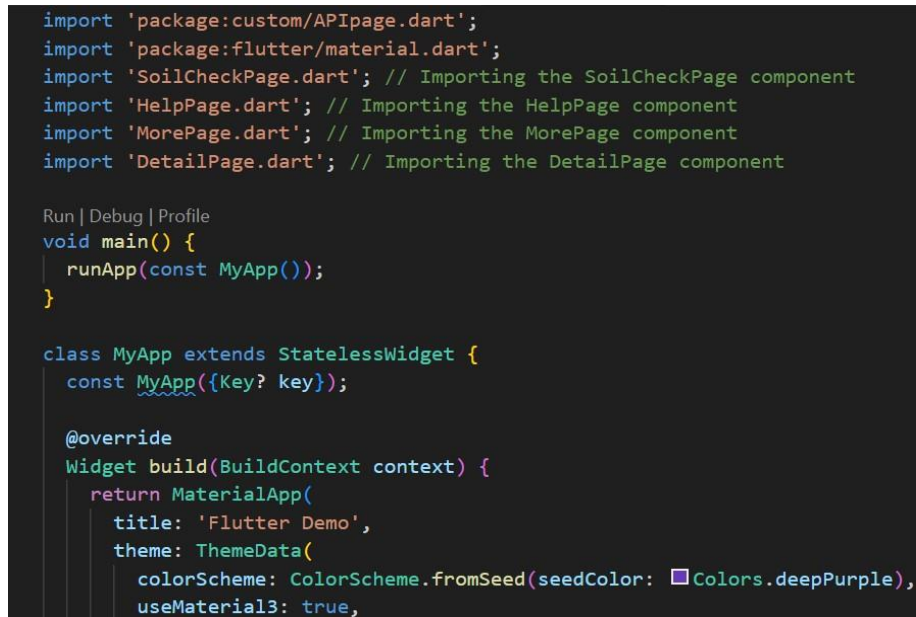
Arduino IDE code for Temperature Sensor (LM35):



```
1 #include <ESP8266WiFi.h>
2 // Define LM35 pin
3 const int lm35_pin = A0; // Analog pin A0 on NodeMCU
4
5 void setup() {
6   Serial.begin(9600);
7 }
8
9 void loop() {
10   float temperature = readTemperature();
11   Serial.print("Temperature: ");
12   Serial.print(temperature);
13   Serial.println(" °C");
14   delay(5000); // Read temperature every 5 seconds
15 }
16
17 float readTemperature() {
18   int adc_value = analogRead(lm35_pin); // Read ADC value
19   float temperature = (adc_value * 3.3 / 1024) * 100; // Convert ADC value to temperature in Celsius
20   return temperature-9;
21 }
22
```

### 3.9.2 Frontend

Main file in flutter:



```
import 'package:custom/APIpage.dart';
import 'package:flutter/material.dart';
import 'SoilCheckPage.dart'; // Importing the SoilCheckPage component
import 'HelpPage.dart'; // Importing the HelpPage component
import 'MorePage.dart'; // Importing the MorePage component
import 'DetailPage.dart'; // Importing the DetailPage component

Run | Debug | Profile
void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
    );
  }
}
```

## Soil Check

```
import 'dart:convert';
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;

class SoilCheckPage extends StatefulWidget {
  @override
  _SoilCheckPageState createState() => _SoilCheckPageState();
}

class _SoilCheckPageState extends State<SoilCheckPage> {
  final TextEditingController nController = TextEditingController();
  final TextEditingController pController = TextEditingController();
  final TextEditingController kController = TextEditingController();
  final TextEditingController temperatureController = TextEditingController();
  final TextEditingController humidityController = TextEditingController();
  final TextEditingController phController = TextEditingController();
  final TextEditingController rainfallController = TextEditingController();
  String soilStatus = '';
  String cropImage = '';
}
```

## Help page

```
import 'package:flutter/material.dart';

class HelpPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Help'),
      ), // AppBar
      body: SingleChildScrollView(
        child: Padding(
          padding: const EdgeInsets.all(16.0),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: <Widget>[
              Text(
                'Welcome to Our App!',
                style: TextStyle(
                  fontSize: 24,
                  fontWeight: FontWeight.bold,
                ), // TextStyle
              ), // Text
              SizedBox(height: 20),
              Text(
                'Check Your Soil:',
                style: TextStyle(
                  fontSize: 20,

```



## Detail Page

```
import 'package:flutter/material.dart';
import 'Buttons/RiceButton.dart'; // Import the RiceButton widget
import 'Buttons/MaizeButton.dart'; // Import the MaizeButton widget
import 'Buttons/ChickpeaButton.dart'; // Import the ChickpeaButton widget
import 'Buttons/MothbeansButton.dart'; // Import the MothbeansButton widget
import 'Buttons/KidneybeansButton.dart'; // Import the KidneybeansButton widget
import 'Buttons/PigeonpeasButton.dart'; // Import the PigeonpeasButton widget
import 'Buttons/MungbeanButton.dart';
import 'Buttons/BlackgramButton.dart';

class DetailPage extends StatelessWidget {
  const DetailPage({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Detail About Crops'),
      ), // AppBar
      body: Padding(
        padding: const EdgeInsets.all(8.0),
        child: Scrollbar(
          child: ListView(
            children: [
              RiceButton(), // Use the RiceButton widget here
              SizedBox(height: 8), // Add spacing between buttons
            ],
          ),
        ),
      ),
    );
  }
}
```

## Detail about rice

```
import 'package:flutter/material.dart';

class RiceButton extends StatelessWidget {
  const RiceButton({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return ElevatedButton(
      onPressed: () {
        Navigator.push(
          context,
          MaterialPageRoute(builder: (context) => RicePage()),
        );
      },
      style: ElevatedButton.styleFrom(
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(10), // Rectangular shape
        ), // RoundedRectangleBorder
      ),
      child: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Text('Rice'),
      ), // Padding
    ); // ElevatedButton
  }
}
```

Detail about maize:

```
import 'package:flutter/material.dart';

class MaizeButton extends StatelessWidget {
  const MaizeButton({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return ElevatedButton(
      onPressed: () {
        // Navigate to MaizePage when button is pressed
        Navigator.push(
          context,
          MaterialPageRoute(builder: (context) => MaizePage()),
        );
      },
      style: ElevatedButton.styleFrom(
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(10), // Rectangular shape
        ), // RoundedRectangleBorder
      ),
      child: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Text('Maize'),
      ), // Padding
    ); // ElevatedButton
  }
}
```

### 3.9.3 Backend

Backend: Software implementation of machine learning model is done at the backend.

a) Library inclusion

#### ▼ LIBRARY SPECIFICATION

```
[1]: #importing all the necessary library and frameworks
import numpy as np #used for number manipulation
import pandas as pd #data manipulation library
import matplotlib.pyplot as plt #library for data visualization
import seaborn as sns #statistical data visualization
from sklearn.preprocessing import LabelEncoder #for label encoding
from sklearn.model_selection import train_test_split #for splitting data
from sklearn.ensemble import RandomForestClassifier #random forest algorithm
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
import xgboost as xgb
from sklearn.model_selection import RandomizedSearchCV
from sklearn.ensemble import RandomForestClassifier
```

b) Fetching training data

## DATA ANALYSIS

```
[2]: # Loading CSV file into a DataFrame
df = pd.read_csv(r'C:\Users\User\Desktop\Agrotech\agrotech.csv')
df
```

```
[2]:
```

	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice
...	...	...	...	...	...	...	...	...
2195	107	34	32	26.774637	66.413269	6.780064	177.774507	coffee
2196	99	15	27	27.417112	56.636362	6.086922	127.924610	coffee
2197	118	33	30	24.131797	67.225123	6.362608	173.322839	coffee
2198	117	32	34	26.272418	52.127394	6.758793	127.175293	coffee
2199	104	18	30	23.603016	60.396475	6.779833	140.937041	coffee

### c) Labels

```
#Labels before encoding
class_labels = le.classes_
class_labels
#this later will be used to assign the machine learning output with the original labels

array(['apple', 'banana', 'blackgram', 'chickpea', 'coconut', 'coffee',
       'cotton', 'grapes', 'jute', 'kidneybeans', 'lentil', 'maize',
       'mango', 'mothbeans', 'mungbean', 'muskmelon', 'orange', 'papaya',
       'pigeonpeas', 'pomegranate', 'rice', 'watermelon'], dtype=object)
```

### d) Label are encode for the purpose of training

```
#encoding the labels into the integers for machine learning compatibility
le = LabelEncoder()
df['label'] = le.fit_transform(df['label'])
```

```
#corresponding encoded value for the labels
df['label']
```

```
0      20
1      20
2      20
3      20
4      20
..
2195    5
2196    5
2197    5
2198    5
2199    5
Name: label, Length: 2200, dtype: int32
```

### e) Splitting data for training and testing

#### BUILDING THE MODEL

```
[18]: #Splitting the data into training and test set of data
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.10,shuffle=True) # data splitted into 4 sets of datasets, two each for training and testing
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)

(1980, 7)
(220, 7)
(1980,)
(220,)
```

## f) Model training

```
\<<< /
[19]: #model atg is defined as the randomforest classifier
      #the model is made fit based on the training data
      atg_model = RandomForestClassifier()
      atg_model.fit(x_train,y_train)
```

```
[19]: ▼ RandomForestClassifier
      RandomForestClassifier()
```

## g) Hyperparameter tuning

### Hypertuning using csv

```
[25]: # Define the hyperparameter grid
      param_grid = {
          'n_estimators': [100, 150, 200],
          'max_depth': [10, 15, 20],
          'min_samples_split': [2, 5, 10],
          'min_samples_leaf': [1, 2, 4]
      }
```

```
[26]: # Instantiate RandomizedSearchCV
      rscv_model = RandomizedSearchCV(estimator=atg_model, param_distributions=param_grid, n_iter=10, cv=5, random_state=42)
```

```
[ ]: # Fit RandomizedSearchCV to the training data
      rscv_model.fit(x_train, y_train)
```

```
[ ]: # Get the best estimator
      best_model = rscv_model.best_estimator_
```

h) Processing input from the frontend application

```
app = Flask(__name__)

def predict_crop(n, p, k, temperature, humidity, ph, rainfall):
    # Load CSV file into a DataFrame
    df = pd.read_csv(r'C:\Users\User\Desktop\Agrotech\agrotech.csv')

    # Convert the labels into a Python list for machine learning
    class_labels = df['label'].unique().tolist()

    # Encode the labels into integers for machine learning compatibility
    le = LabelEncoder()
    df['label'] = le.fit_transform(df['label'])
    class_labels = le.classes_

    # Divide dataset into input 'X' and output 'Y'
    X = df.drop('label', axis=1)
    Y = df['label']
```

```
# Check if the trained model file exists
model_file = 'agrotech_model.pkl'
try:
    # Load the trained model from the file
    best_model = joblib.load(model_file)
except FileNotFoundError:
    # Split the data into training and test sets
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.10, shuffle=True)

    # Train a RandomForestClassifier
    atg_model = RandomForestClassifier()
    atg_model.fit(X_train, Y_train)

    # Define the hyperparameter grid
    param_grid = {
        'n_estimators': [100, 150, 200],
        'max_depth': [10, 15, 20],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4]
    }

    # Instantiate RandomizedSearchCV
    rscv_model = RandomizedSearchCV(estimator=atg_model, param_distributions=param_grid, n_iter=10, cv=5,
                                    random_state=42)

    # Fit RandomizedSearchCV to the training data
    rscv_model.fit(X_train, Y_train)
```



```

# Get the best estimator
best_model = rscv_model.best_estimator_

# Save the trained model to a file
joblib.dump(best_model, model_file)

# Taking user inputs
test_series = pd.Series(np.zeros(len(X.columns)), index=X.columns)
test_series['N'] = n
test_series['P'] = p
test_series['K'] = k
test_series['temperature'] = temperature
test_series['humidity'] = humidity
test_series['ph'] = ph
test_series['rainfall'] = rainfall

# Convert test_series into a DataFrame with a single row
test_df = test_series.to_frame().T

# Predict the recommended crop
output = best_model.predict(test_df)[0]
recommended_crop = class_labels[output]
print("Recommended Crop:", recommended_crop)
return recommended_crop

```

i) Sending the output label to the frontend application

```

@app.route('/api/send_data', methods=['POST'])
def checksoil():
    try:
        data = request.json
        n = float(data.get('N', 0))
        p = float(data.get('P', 0))
        k = float(data.get('K', 0))
        temperature = float(data.get('temperature', 0))
        humidity = float(data.get('humidity', 0))
        ph = float(data.get('pH', 0))
        rainfall = float(data.get('rainfall', 0))

        # Perform soil analysis using write_word function
        soilstatus = predict_crop(n, p, k, temperature, humidity, ph, rainfall)

        return jsonify({'Suitable crop is ': soilstatus})
    except Exception as e:
        return jsonify({'error': str(e)}), 500

if __name__ == '__main__':
    app.run(host='#ipaddress', port=5000)

```



## **4. Results and Discussion**

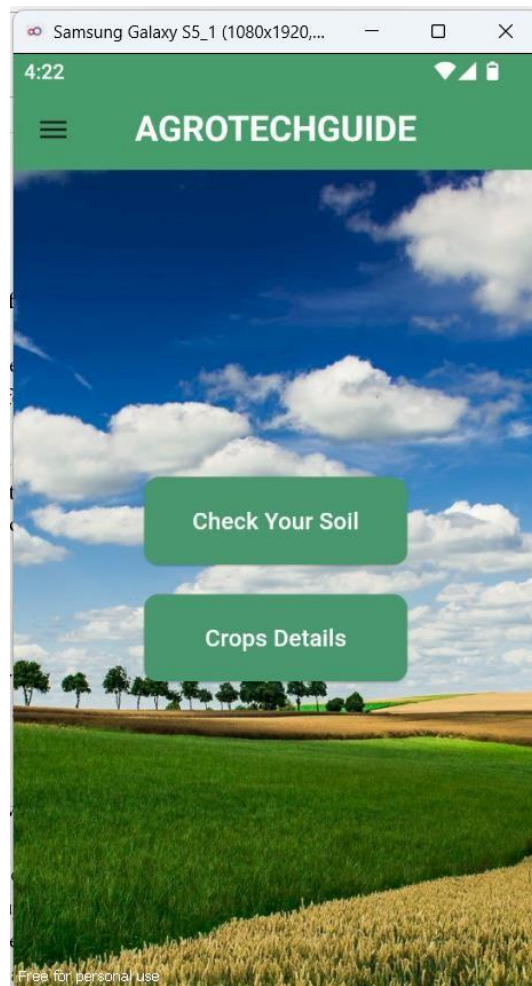
### **4.1 Results**

AgrotechGuide represents a significant milestone in the field of agriculture in Nepal by offering tailored crop recommendations based on environmental and soil data. This innovative system streamlines the process of crop selection for farmers, addressing the challenge of optimizing agricultural productivity in diverse geographical and climatic conditions. Through a meticulously designed user interface and integration of hardware sensors for data collection, AgrotechGuide ensures accessibility and usability for farmers. Its machine learning model, trained on extensive datasets of land profiles and crop performance metrics, delivers accurate recommendations, empowering farmers to make informed decisions about crop cultivation.

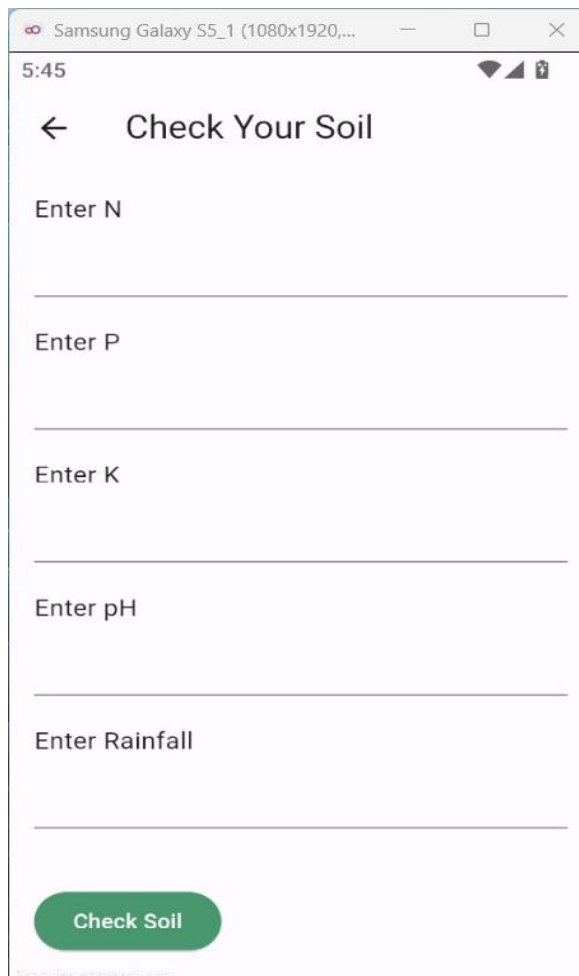
Furthermore, AgrotechGuide goes beyond mere recommendation by providing comprehensive cultivation guidance for the recommended crops. This additional feature equips farmers with essential knowledge on optimal planting times, and irrigation strategies, thereby fostering improved agricultural practices and sustainability. With its potential to enhance productivity and profitability in Nepalese agriculture, AgrotechGuide heralds a new era of data-driven decision-making and innovation in the agricultural sector, offering transformative benefits for farmers.

1) Home page:

This is the starting page of the application



2) Check your soil page :  
Input form to take soil data



The screenshot shows a mobile application interface on a Samsung Galaxy S5\_1. The title bar at the top indicates the device model and resolution. The app's status bar shows the time as 5:45 and standard Android icons for Wi-Fi, signal strength, and battery. The main screen has a light pink background and is titled 'Check Your Soil' with a back arrow icon. Below the title, there are five input fields, each preceded by a label: 'Enter N', 'Enter P', 'Enter K', 'Enter pH', and 'Enter Rainfall'. Each label is followed by a horizontal line representing the input field. At the bottom of the screen, there is a green rounded rectangular button with the text 'Check Soil' in white.

Samsung Galaxy S5\_1 (1080x1920,...)

5:45

← Check Your Soil

Enter N

Enter P

Enter K

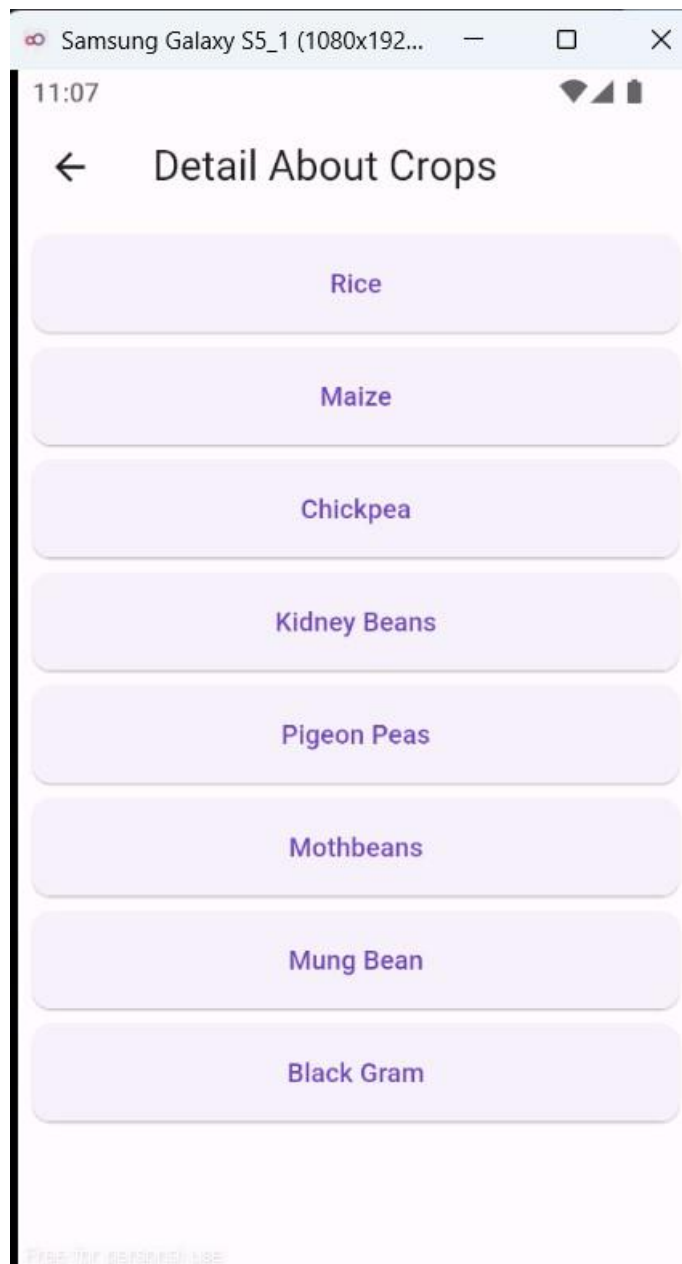
Enter pH

Enter Rainfall

Check Soil

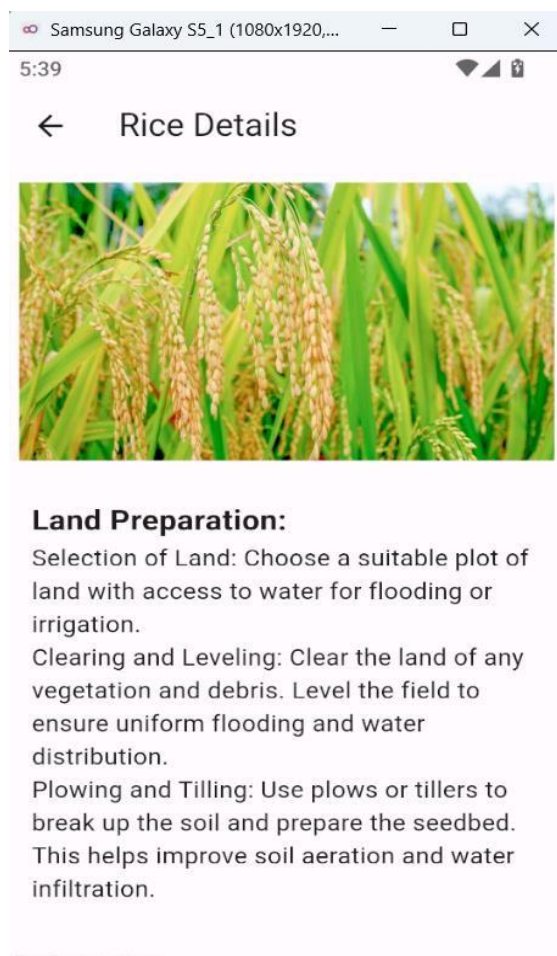
3 ) Detail page :

User is directed to this page when clicked Crop Detail



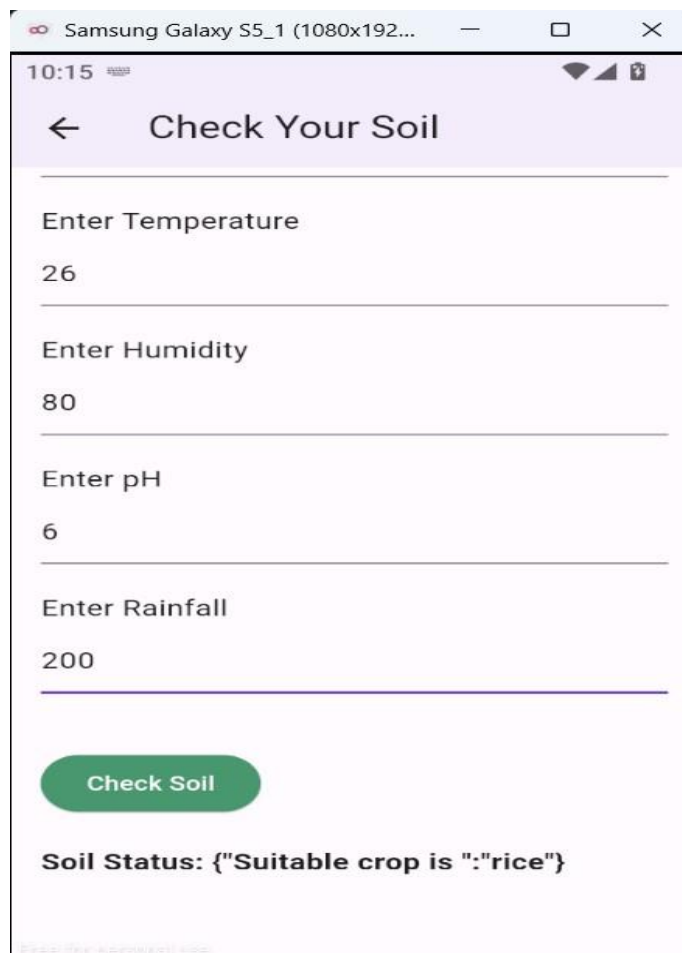
#### 4 ) Rice detail :

User is directed to this page when they choose rice inside the Crop Detail Page:



5) Output in app :

This is the demo output from the application



The screenshot shows a mobile application interface on a Samsung Galaxy S5\_1. The app has a title bar with a back arrow and the text "Check Your Soil". Below the title bar, there are four input fields with labels: "Enter Temperature", "Enter Humidity", "Enter pH", and "Enter Rainfall". The values entered in these fields are 26, 80, 6, and 200, respectively. Below the input fields is a green button labeled "Check Soil". At the bottom of the screen, the text "Soil Status: {"Suitable crop is ":"rice"}" is displayed. The status bar at the top shows the time 10:15 and various icons.

10:15

← Check Your Soil

Enter Temperature

26

Enter Humidity

80

Enter pH

6

Enter Rainfall

200

Check Soil

Soil Status: {"Suitable crop is ":"rice"}

Samsung Galaxy S5\_1 (1080x192...

10:17

← Check Your Soil

Enter Temperature

20

Enter Humidity

68

Enter pH

6

Enter Rainfall

87

Check Soil

Soil Status: {"Suitable crop is ":"maize"}

Free for personal use

#### 4.2 Limitations:

**Dependency on Data Quality:** The effectiveness of the recommendation system heavily relies on the quality and representativeness of the labeled data used for training. Inaccurate or biased data can lead to suboptimal recommendations and may not capture the full complexity of agricultural conditions.

**Limited Generalization:** Although random forest algorithms are robust and can handle various types of data, including categorical and numerical variables, they may struggle with generalizing to unseen or unusual scenarios. The model's performance may degrade when presented with data that significantly differs from the training dataset.

Limited Scope of Sensor Inputs: While sensor data can provide valuable insights into environmental conditions, they may not capture all relevant factors influencing crop growth and health. For instance, sensor inputs may not account for socio-economic factors, farmer preferences, or specific crop management practices.

### **4.3 Future plans**

Our future plans for the project involve several key initiatives aimed at enhancing its effectiveness and expanding its reach.

Continued Model Refinement: We aim to further train the model by incorporating feedback from users. By collecting and analyzing user feedback, we can identify areas for improvement and fine-tune the recommendation algorithms to better meet the needs of farmers and agricultural stakeholders.

Integration as a Testing Tool for Agriculture Technicians: Our goal is to integrate the crop recommendation system into the toolkit of agriculture technicians and extension workers. By providing them with access to our platform, we can empower them to make data-driven recommendations and provide targeted advice to farmers. This integration will enhance the efficiency and effectiveness of agricultural extension services and contribute to improved crop productivity and sustainability.

Partnerships with Agriculture-Based Software Companies: We plan to collaborate with agriculture-based software companies like Geo-krishi<sup>[4]</sup> to integrate our project with their existing platforms and solutions. By aligning our efforts with established industry players, we can leverage their technological infrastructure and user base to amplify the impact of our project.



## **5. Conclusion:**

AgrotechGuide project offers a promising solution to enhance agricultural productivity in Nepal through the application of machine learning. The system recommends crops based on soil and environmental data, aiding farmers in making informed planting decisions. This project underscores the potential of technology to optimize yields, with future prospects including model refinement and integration with agriculture-based software companies to better support Nepalese farmers.

## 6. REFERENCES

- [1][https://www.researchgate.net/publication/354533829\\_Prime\\_Minister\\_Agriculture\\_Mode\\_rnization\\_Project\\_PMAMP\\_Nepal?fbclid=IwAR0IxxjZYJ364RvztPAnOnPTauCKilb1fRSF5K3tyseC9u6VIcRd2aJHQ4u0](https://www.researchgate.net/publication/354533829_Prime_Minister_Agriculture_Mode_rnization_Project_PMAMP_Nepal?fbclid=IwAR0IxxjZYJ364RvztPAnOnPTauCKilb1fRSF5K3tyseC9u6VIcRd2aJHQ4u0)
- [2][https://www.researchgate.net/publication/370056714\\_Crop\\_Recommendation\\_System](https://www.researchgate.net/publication/370056714_Crop_Recommendation_System)
- [3][https://www.researchgate.net/publication/344376491\\_Future\\_prospects\\_of\\_precision\\_agr\\_iculture\\_in\\_Nepal](https://www.researchgate.net/publication/344376491_Future_prospects_of_precision_agr_iculture_in_Nepal)
- [4]<https://www.geokrishi.farm>