
LANDSLIDE
GUARDIAN

# INTRODUCTION


**The Landslide Guardian empowers communities with an early warning system based on linear regression prediction model using real time data and a chatbot providing basic first aid information and mental support.**

**This comprehensive approach aims to significantly reduce casualties, minimize damage, and provide mental support during critical moments.**

# PURPOSE & BENEFITS

- **Purpose of our project is to predict the landslide severity and make sure people safe from landslide and ensure mental support.**

- **Benefits of our system is there is emergency chatbots, warning system  if there is a landslide.**

# TECHNOLOGIES USED

- **FRONTEND: REACT.JS, CSS, HTML**
- **BACKEND : PYTHON IN ML**
- **HARDWARE  :  ARDUINO UNO , SIM 800L, MOISTURE SENSOR**

# WHAT LANDSLIDE GUARDIAN DO

### LANDSLIDE SEVERITY PREDICTION

Using real time data(via API & Iot) from the location predicting landslide risk.

### WEATHER BROADCAST

Live weather broadcast

### CHAT BOT

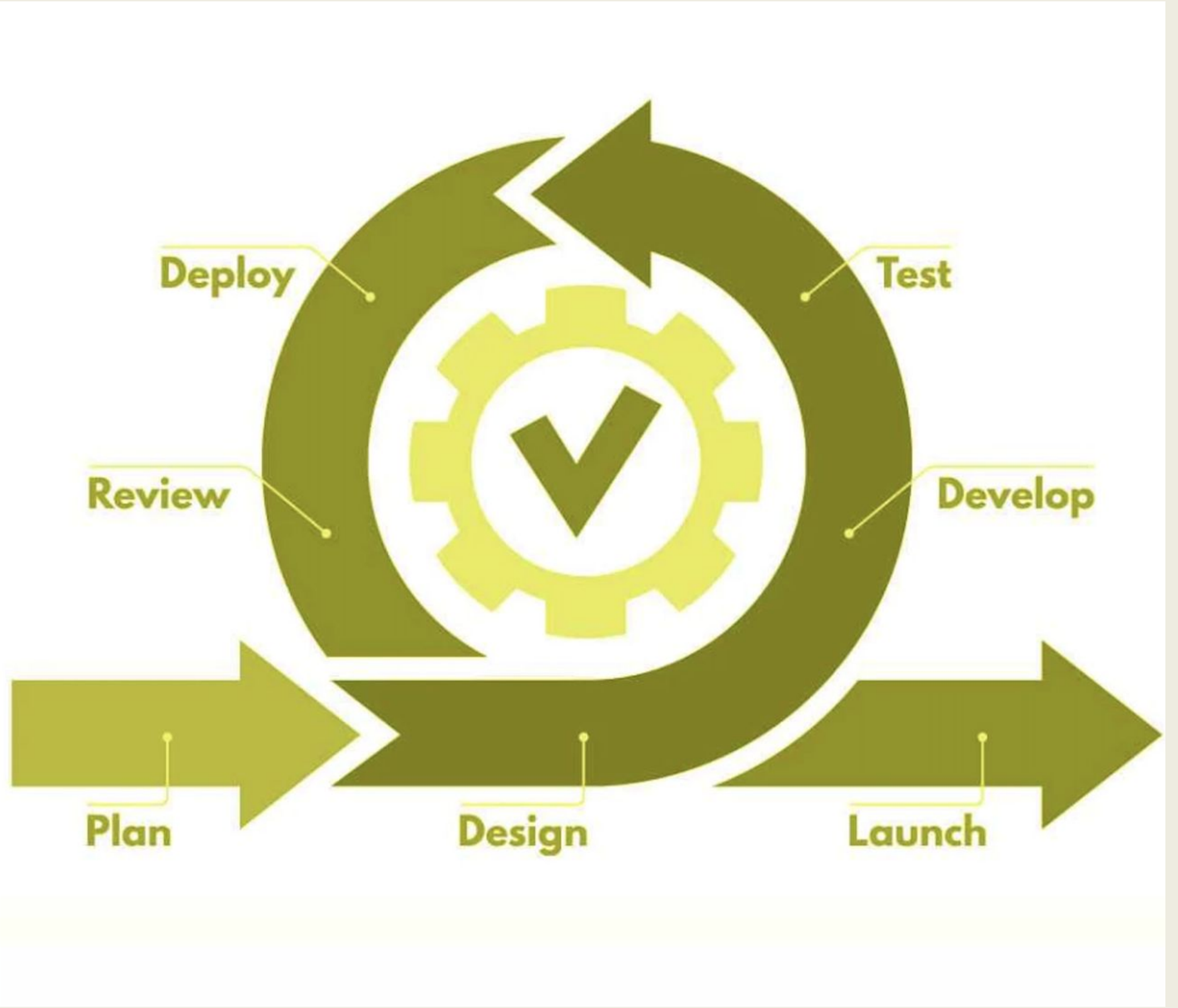
Chatbots are available for basic first aid information

### WARNING SYSTEM

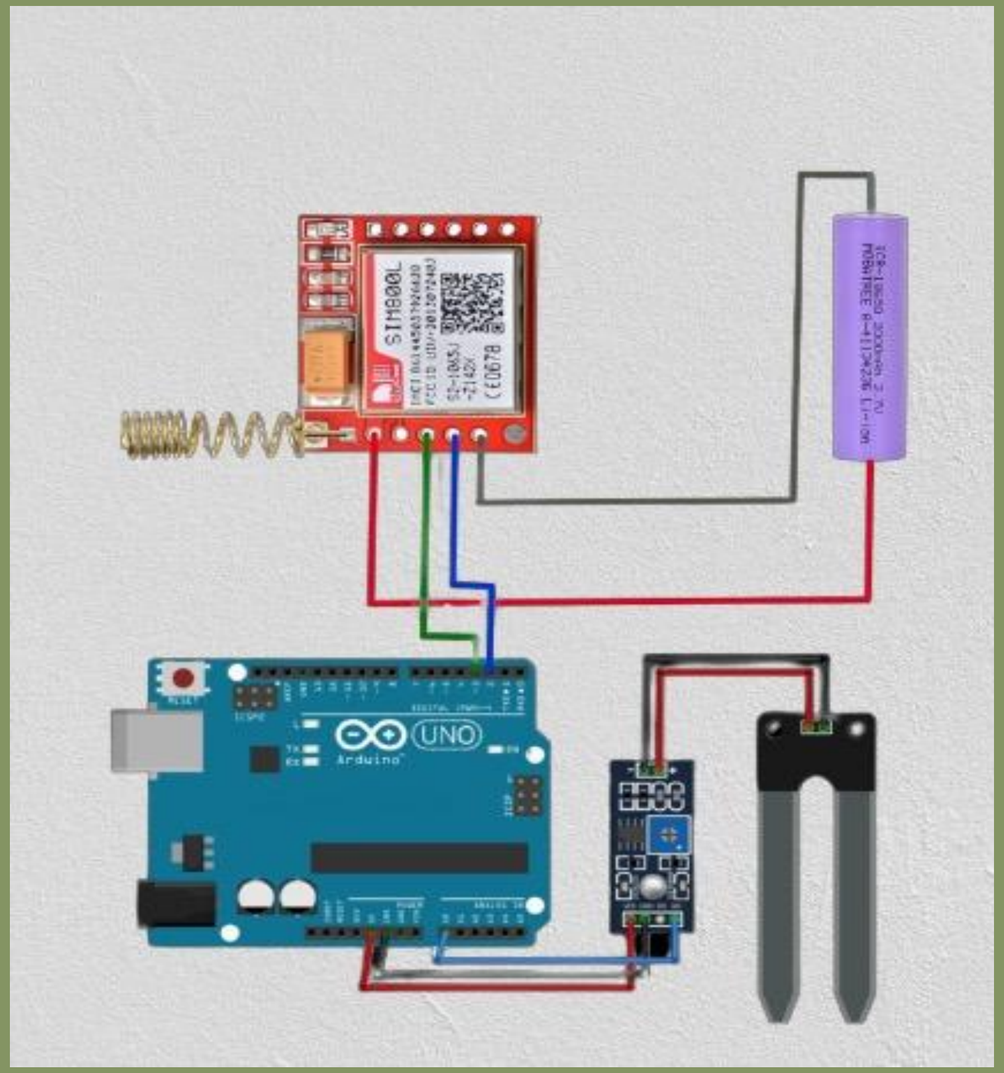
Give warnings about landslides

# DESIGN METHODOLOGY

## Agile

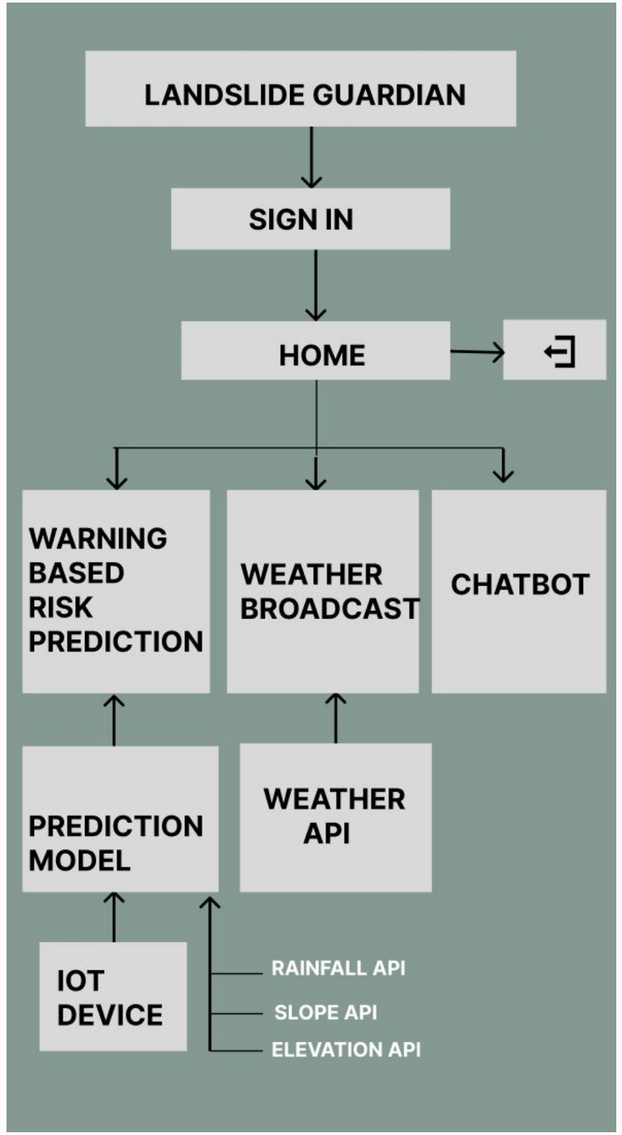


# ARCHITECTURE




HARDWARE

# ARCHITECTURE




SOFTWARE

# ALGORITHM

**IOT:**

1. *Start*

2. *Setup Initialization*
   - Initialize GPRS serial communication.
   - Initialize serial monitor communication.
   - Print "Setup initialized."
   - Wait for 1 second.

3. **Main Loop**
   - Repeat indefinitely:
     1. *Read Moisture Sensor Data*
        - Read moisture sensor value.
        - Print moisture value.

     2. *Check GPRS Module Availability*
        - If data is available from GPRS module, read and print it.

     3. *Send AT Commands to GPRS Module*
        - Send AT commands to initialize and configure the GPRS module.
        - Print status messages.
        - Wait for appropriate delays between commands.
        - Display data from GPRS module after certain commands.

     4. *Connect to Remote Server*
        - Send command to connect to the remote server.
        - Wait for connection status.
        - Print connection status.
        - If connection fails, exit loop.

     5. *Send Data to Remote Server*
        - Send command to initiate data sending.
        - Construct and send HTTP GET request with moisture value.
        - Print constructed request.
        - Send End Of Text character to end data sending.
        - Print "Data sent to server".

     6. *Close Connection*
        - Send command to close the connection.
        - Print "Closed connection."

4. *Stop*

.

# ALGORITHM

**Function to fetch real_time_soil_moisture :**

1.Define the fetch_data function to fetch data from the ThingSpeak API and parse the response as JSON.
2.Define the preprocess_data function to extract from field and convert soil moisture data.
3.Define the channel ID, read key, and field number.
4.Call the fetch function to get new data following preprocess_data function.
5.Check if the fetched data is valid (i.e., not None).
6.Raise an error if no valid data is fetched.
7.Return the soil moisture value.

**function get_weather_data :**

1.Fetches weather data using the OpenWeatherMap API,
2.Set the API key.
3.Input the location.
4.Construct the URL for the API request, including the location, API key, and units (metric) and make API Request
5.Using the request library send a GET request to the API URL and check the status code of the response to ensure it is 200 (OK).
6.Process API Response to get JSON data.
7.Extract rainfall(mm) information:Retrieve rainfall data for the last hour, defaulting to 0 if not present.
8.Exract the elevation(meter) and slope(Degree) data of the location
9.Handle any key errors or missing data by setting default values.
10.Return the extracted rainfall,elevation and slope values.

# ALGORITHM

**function Predicting_Landslide_Severity:**

1.Import neccessary libraries such as pandas, numpy, and sklearn
2.Load the dataset(csv file) using pandas.
3.Define the feature matrix X with relevant columns: 'Elevation', 'AAP(mm)', 'RiverDIST(m)', 'Slop(Degrees)', 'Soil Moisture'.
4.Define the target vector Y with the 'Landslide Risk' column.
5.Split the data into training and testing sets using train_test_split with 70% training and 30% testing.
6.Initialize the LinearRegression model.
7.Initialize the Ridge regression model with specified alpha(regularization strength), max_iter, and tolerance (for optimization) parameters.
8.Fit the model on the training data (Xtrain,Ytrain).
9.Calculate the training and testing accuracy of the Ridge regression model.
10.Use model to make predictions on the testing set Xtest.
11.Calculate and print the Mean Squared Error (MSE) and R-squared score for the predictions.
12.Call the get_soil_moiture() function to get the current soil moisture.
13.Call the get_weather_data() function to get the current rainfall,elevation and slope.
14.Prepare New Data Point:[elevation, rainfall, 0, random_slope, soilMoisture].
15.Use the trained linear regression model to predict landslide severity for the new data point.
16.Return the predicted landslide severity.

# ALGORITHM

**function Warning&Monitoring_system:**

1.Set the risk_value as  Predicting_Landslide_Severity
2.Categorize risk_value into warning levels based on defined thresholds.
3.Use the risk_value predicted by the model to get the corresponding warning level.
4.Implement Monitoring System to predict the risk levels for new data points.
5.Categorize the predicted risk values into warning levels.
6.Return the warning levels for the new data points.

# ALGORITHM

**chatbot**

. 1. Data Preparation
- Import necessary libraries
- Download required NLTK data
- Initialize WordNetLemmatizer
- Load intents from JSON file
- Tokenize and lemmatize words, ignore specified characters
- Remove duplicates and sort words and classes
- Save words and classes using pickle

2. Training Data Preparation
- Initialize training data
- Create bag of words for each document
- Shuffle and split training data into features and labels

3. Model Training
- Build and compile a neural network model
- Define learning rate schedule
- Train the model with specified parameters
- Save the trained model

4. Chatbot Prediction and Response
- Load words, classes, and trained model
- Define helper functions for tokenizing, bag of words, predicting class, and getting response
- Continuously read user input, predict intents, and generate responses

# RESULT

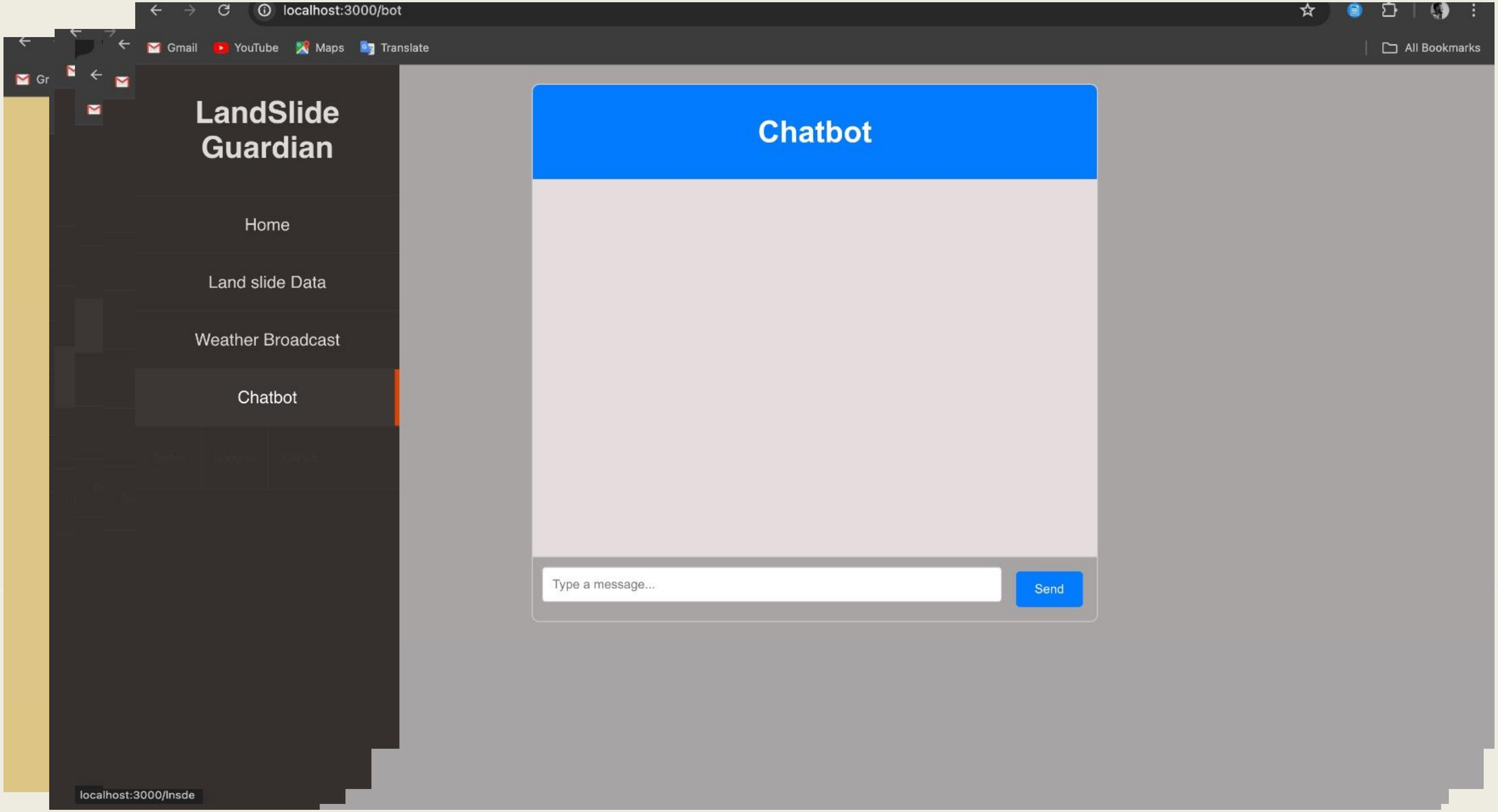
## HARDWARE

- Moisture detection  using IOT

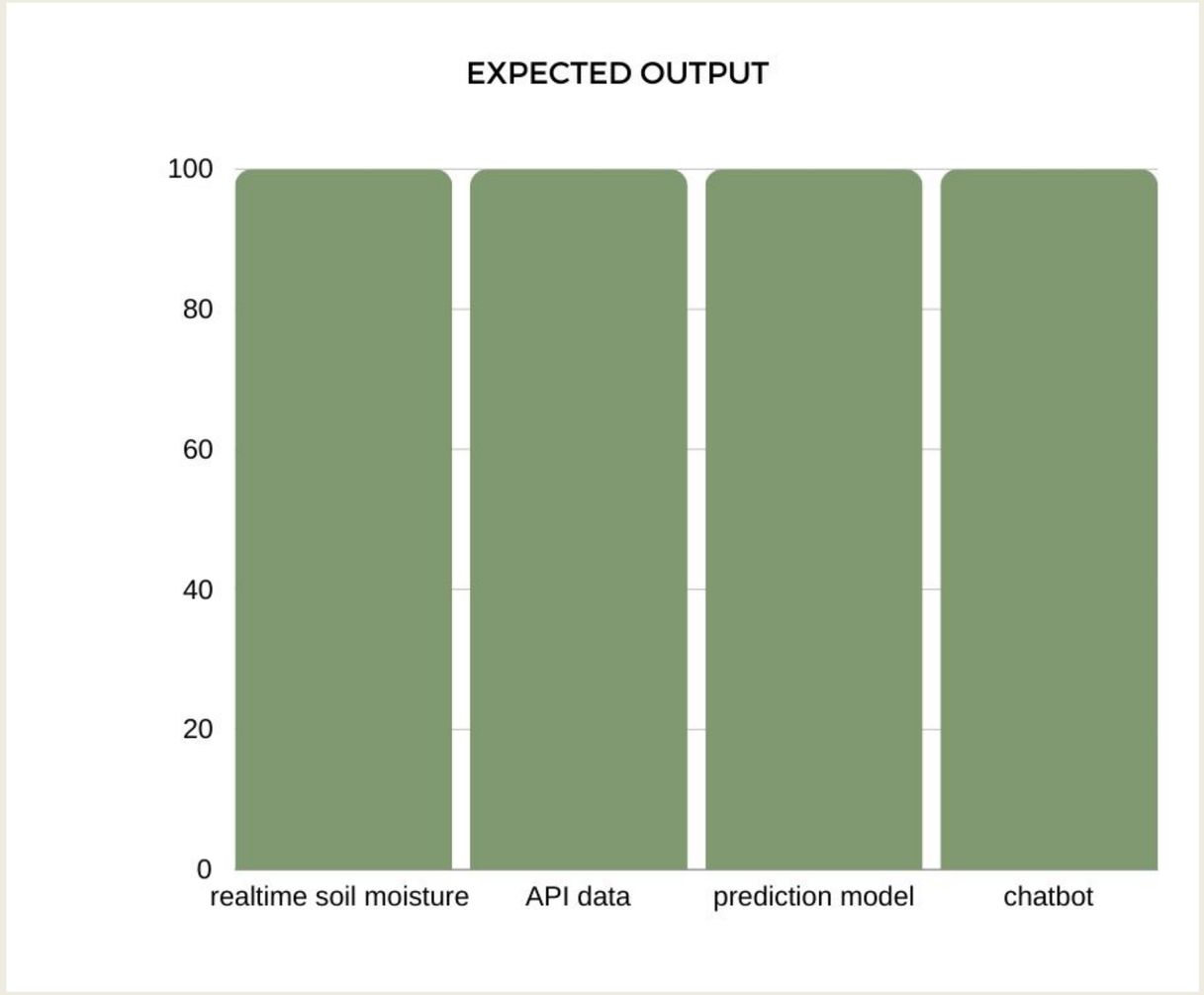
## SOFTWARE

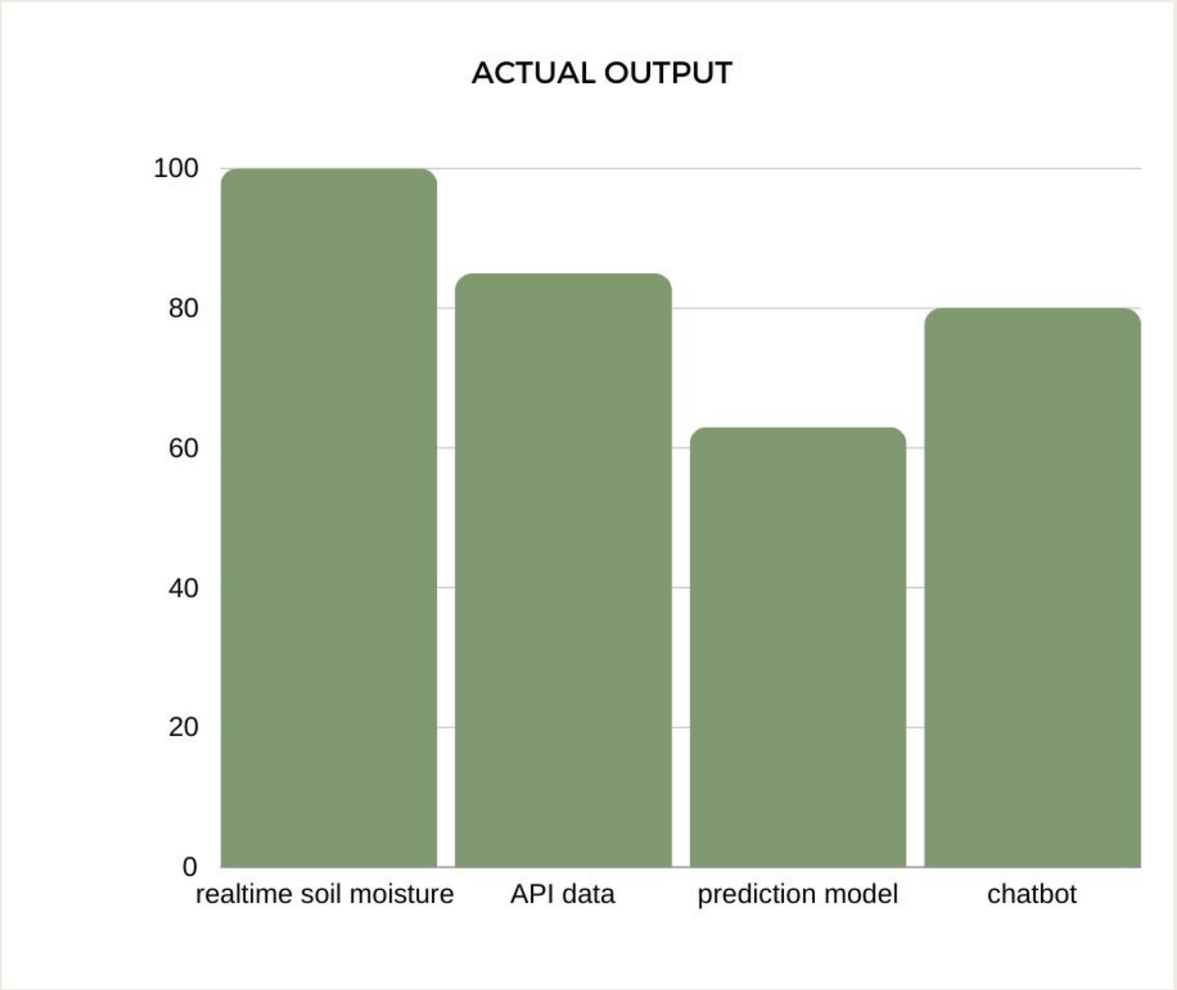
- Live Weather broadcast , so that we can get the details of weather
- Real-time data risk prediction
- Warning System and Monitoring System
- Chatbots provided basic first-aid information and mental support

# SCREENSHOTS



# PERFORMANCE EVALUATION

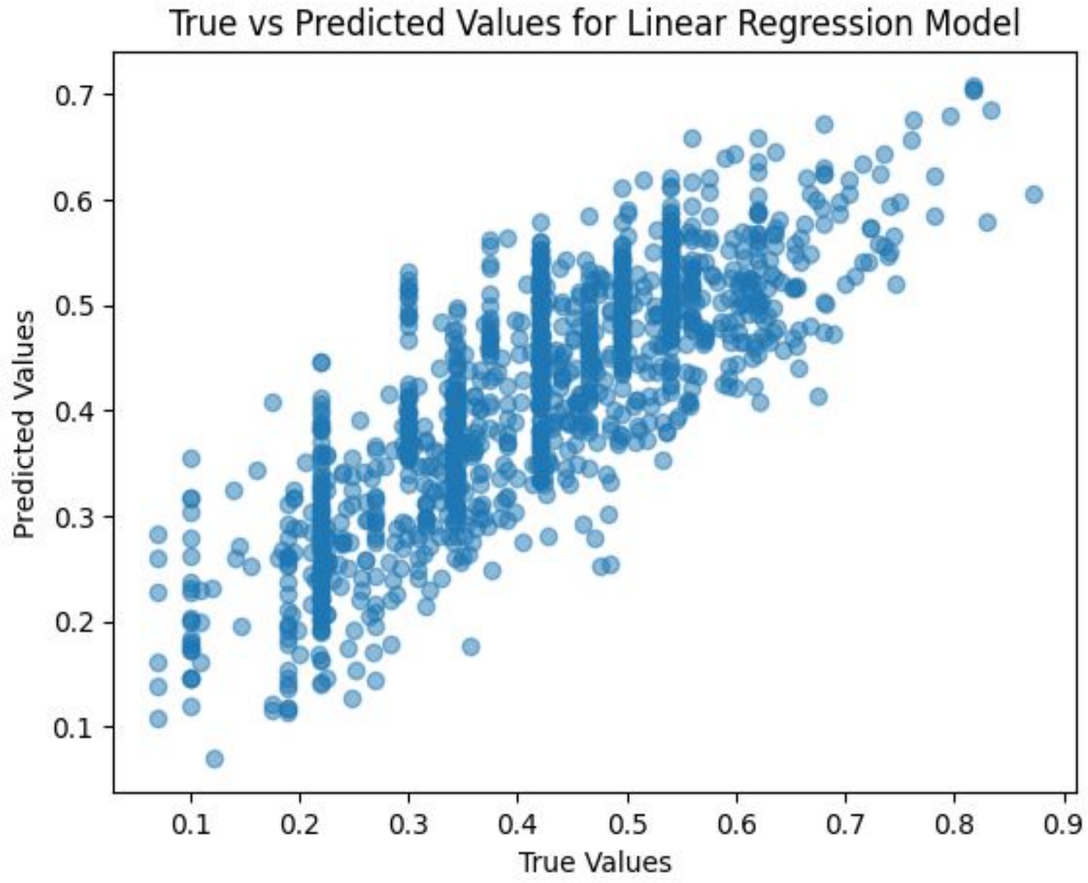


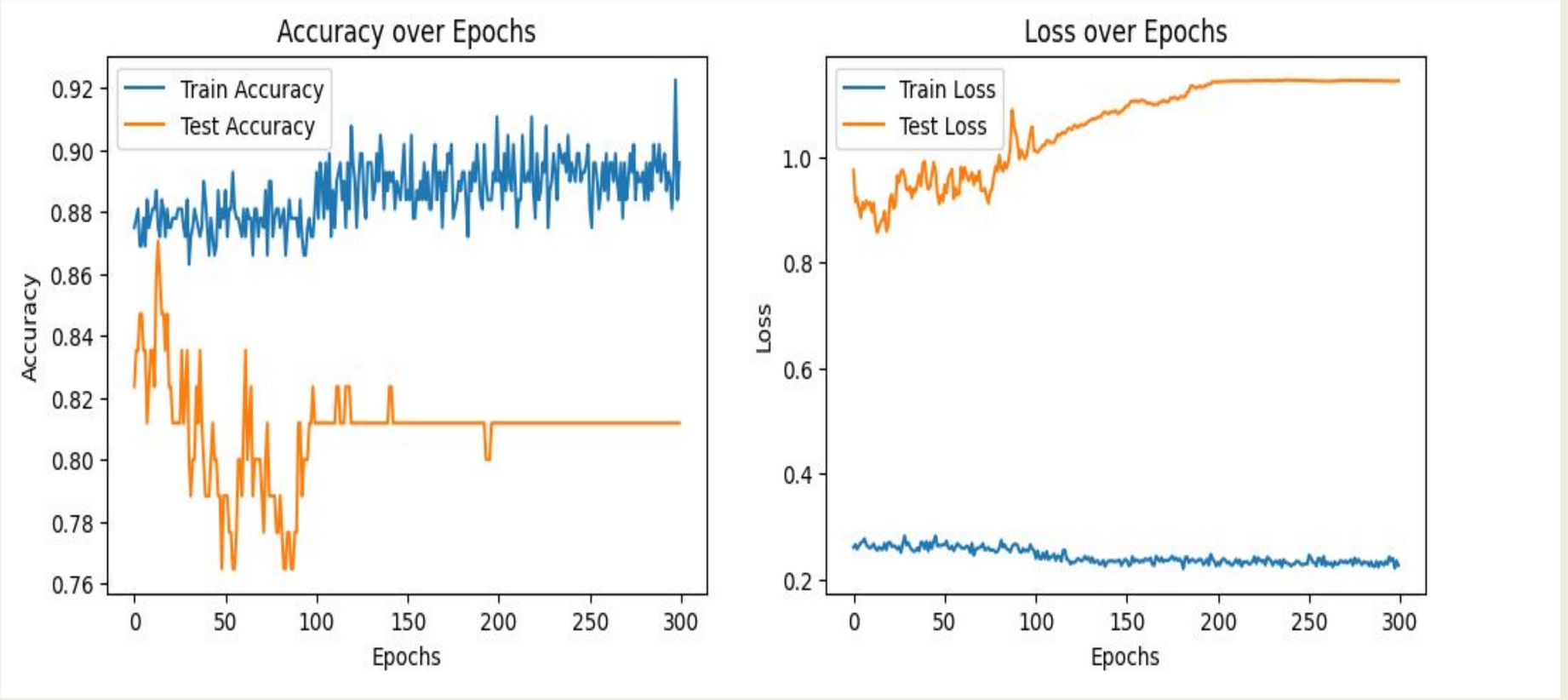
# PERFORMANCE EVALUATION

# PEDICTION SYSTEM

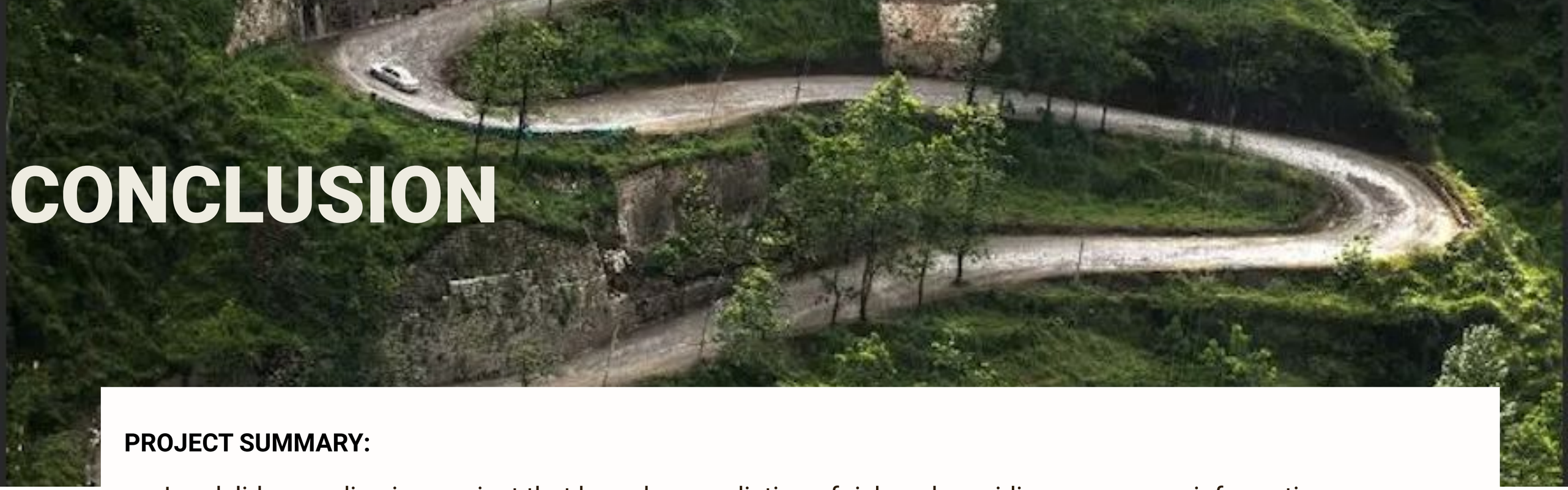
```
Train accuracy: 64.94 %
Test accuracy: 66.53 %
Mean Squared Error: 0.69 %
R-squared Score: 66.53 %
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
  warnings.warn(
Predicted Landslide Severity: [0.29]
```



# PERFORMANCE EVALUATION OF CHATBOT




```
Training Accuracy: 89.28571343421936
Test Accuracy: 80.0000011920929
Training MSE: 0.09163563332075923
Test MSE: 0.24820825075858263
Training RMSE: 3.027137811873771
Test RMSE: 4.98205028837107
```



# CONCLUSION

**PROJECT SUMMARY:**

- Landslide guardian is a project that based on prediction of risk and providing emergency information on landslide.
- For this project we used iot device integrated with machine learning and a website for user access.

**FUTURE DIRECTIONS:**

- In future we would like to make some improvisations such as alarm system, safe route map etc.

# OUR TEAM

**GUIDE : MR SAVYAN (ASST.PROF CSE)**

**MEMBERS:**

**ROHINIKANTH**

**PRAVEENKUMAR**

**ASWIN**

**ARCHA**