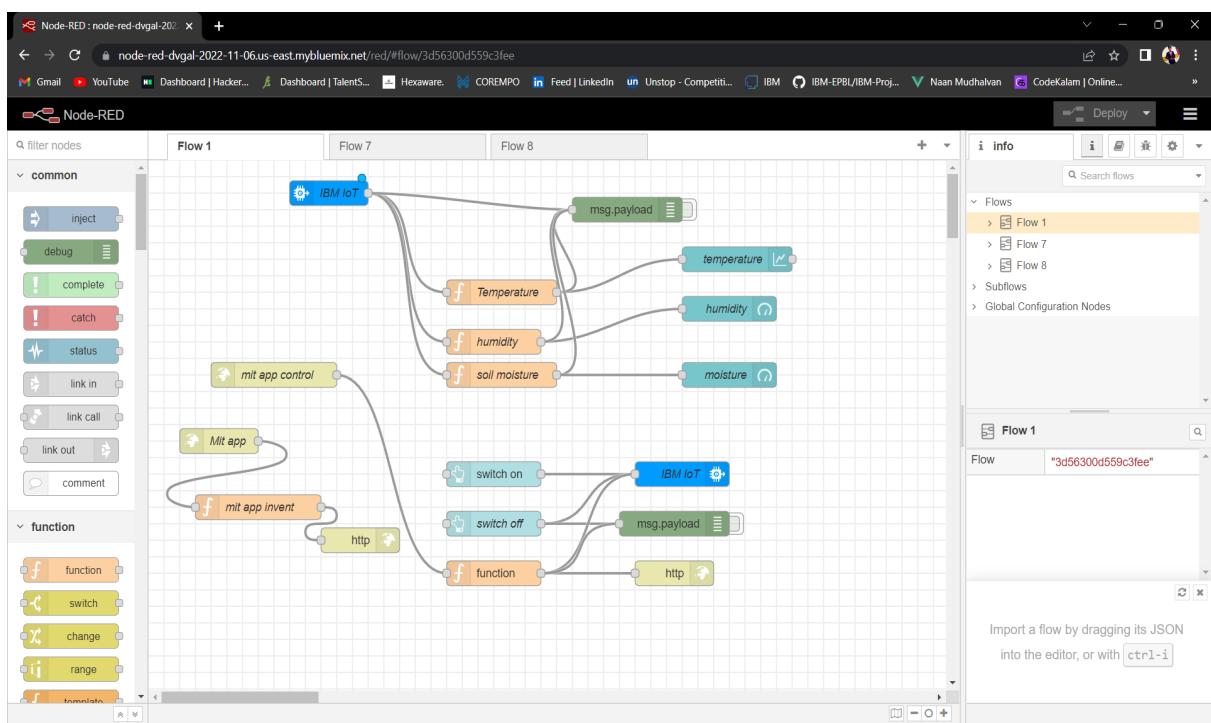


SPRINT 4

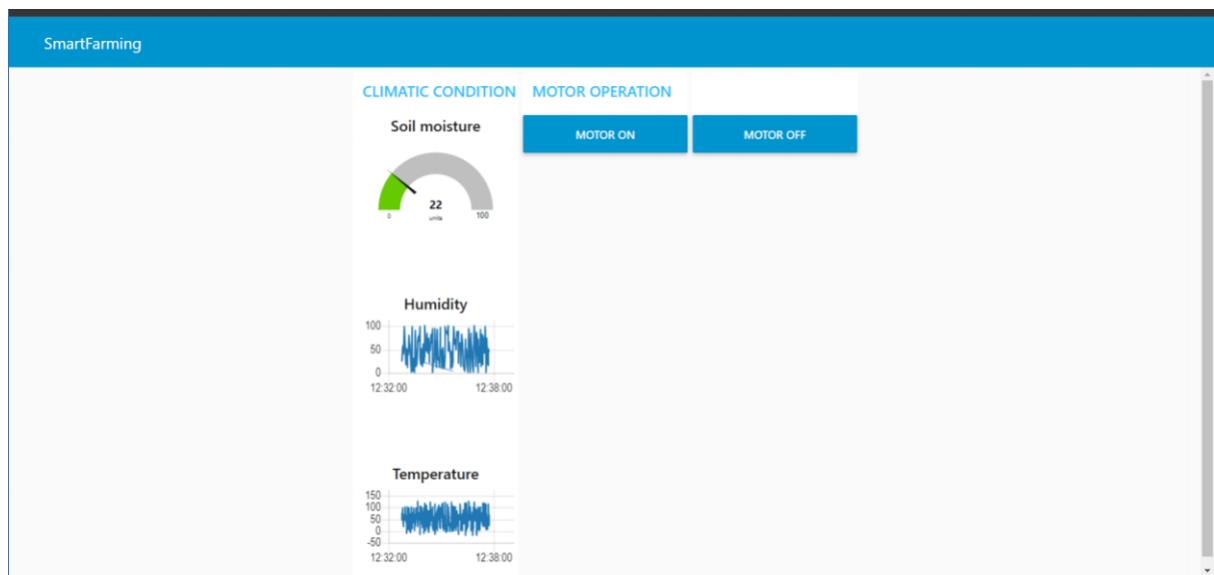
TEAM ID	PNT2022TMID33671
PROJECT NAME	Smartfarmer - IOT enabled smart Farming Application

BUILD A WEB APPLICATION USIND NODE-RED SERVICES:

STEP 1: NODE-RED



OUTPUT SCREENSHOT:



STEP 2:

IBM WATSON DEVICE PLATFORM

The screenshot shows the IBM Watson IoT Platform interface. The top navigation bar includes tabs for "Service Details - IBM Cloud", "IBM Watson IoT Platform", and a search bar. The main header says "IBM Watson IoT Platform". On the left, there's a sidebar with various icons. The main content area has tabs for "Browse", "Action", "Device Types", and "Interfaces". A "Search by Device ID" input field is present. A table lists a single device entry: "1234" (Status: Disconnected, Device Type: Node-red, Class ID: Device, Date Added: Nov 5, 2022 10:11 PM). Below the table, a section titled "Recent Events" shows a table of event logs. The log entries are as follows:

Event	Value	Format	Last Received
status	{"soil_moisture":34,"temperature":48,"humidity":...	json	a few seconds ago
status	{"soil_moisture":18,"temperature":87,"humidity":...	json	a few seconds ago
status	{"soil_moisture":40,"temperature":87,"humidity":...	json	a few seconds ago
status	{"soil_moisture":56,"temperature":4,"humidity":...	json	a few seconds ago
status	{"soil_moisture":59,"temperature":-9,"humidity":...	json	a few seconds ago

At the bottom right, a status bar shows "1 Simulation running".

OUTPUT SCREENSHOT:



STEP 3:

PYTHON SCRIPT

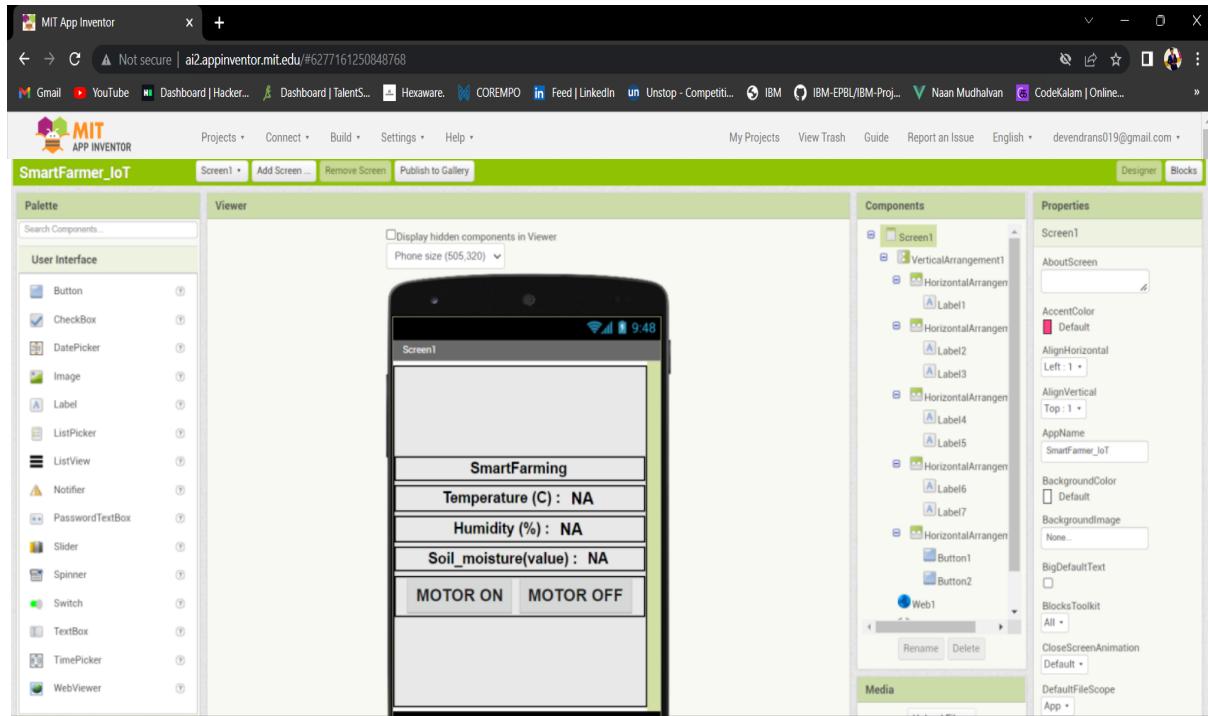
The screenshot shows a Windows desktop environment with two windows open:

- SmartFarmer.py - C:\python\Python37\SmartFarmer.py (3.7.4):** This window contains the Python code for the SmartFarmer application. It includes imports for `wiotp.sdk.device`, `time`, `os`, `datetime`, and `random`. It defines a configuration object `myConfig` with identity and auth details, and creates a device client `client` using the configuration. The script then enters a loop where it prints commands received from IBM IoT, handles motor control logic, and publishes random sensor data (soil moisture, temperature, humidity) to the platform.
- *Python 3.7.4 Shell:** This window shows the output of the Python script. It displays numerous `Published data successfully` messages, each containing a JSON object with `soil_moisture`, `temperature`, and `humidity` fields, indicating the periodic publishing of sensor data.

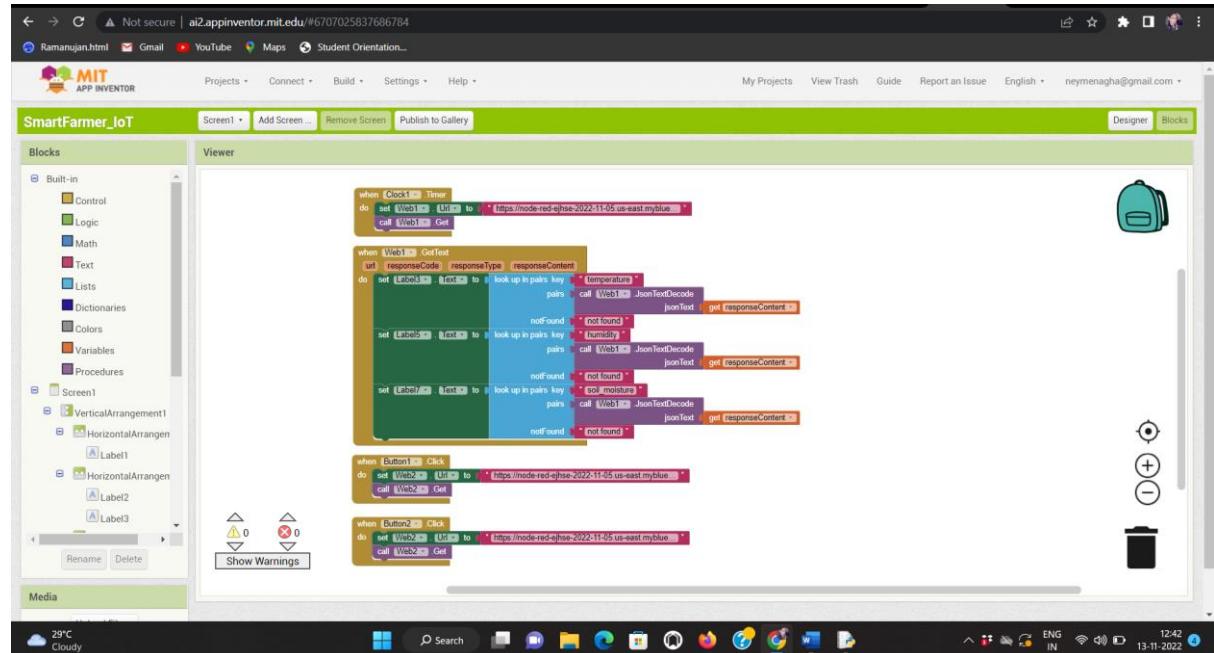
DEVELOP A MOBILE APPLICATION:

STEP 1:

MIT APP INVENTOR



BLOCKS:



MOBILE SCREEN:

