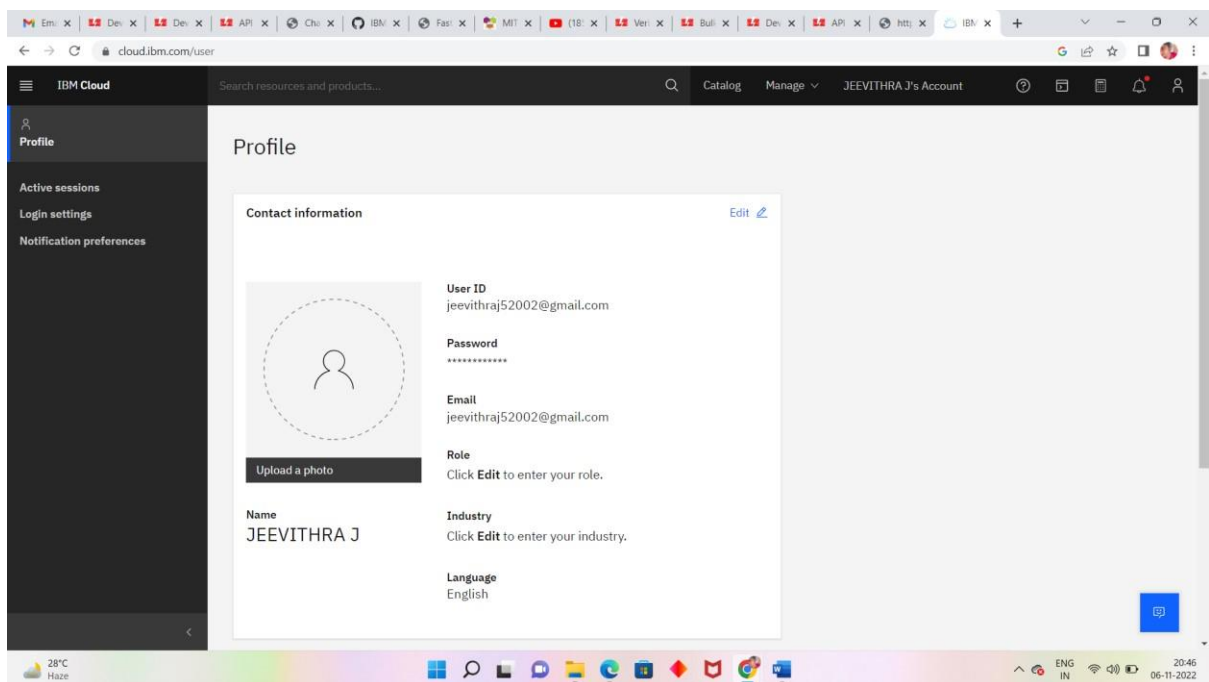
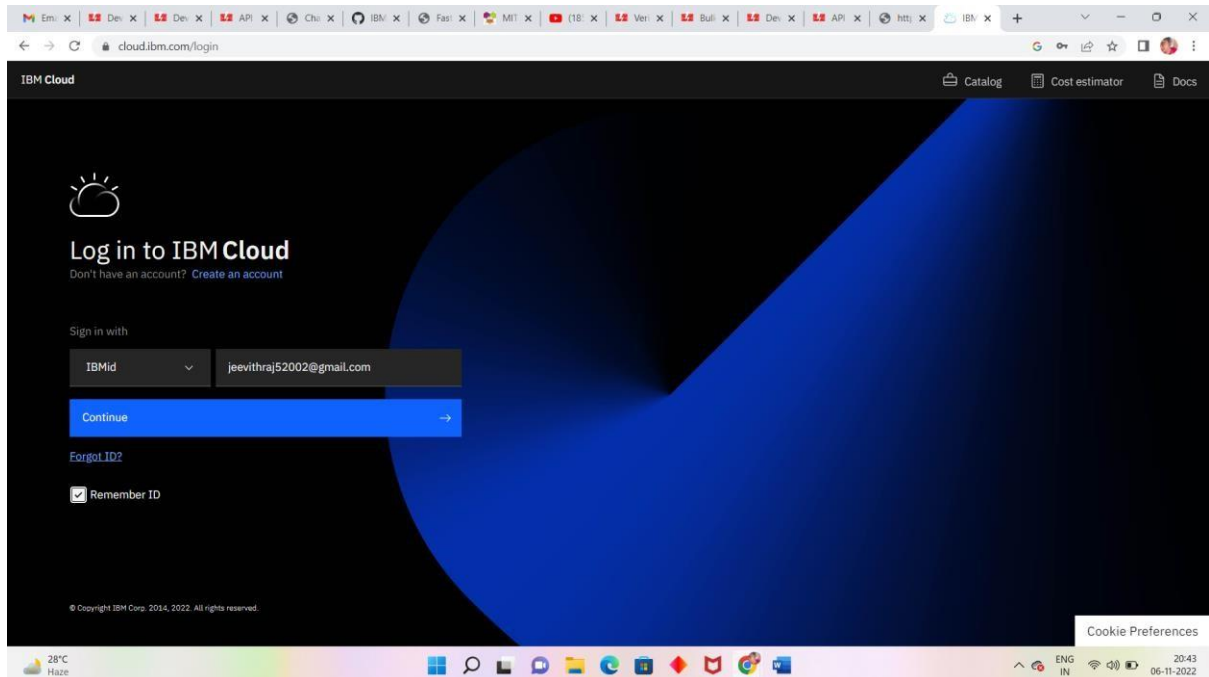
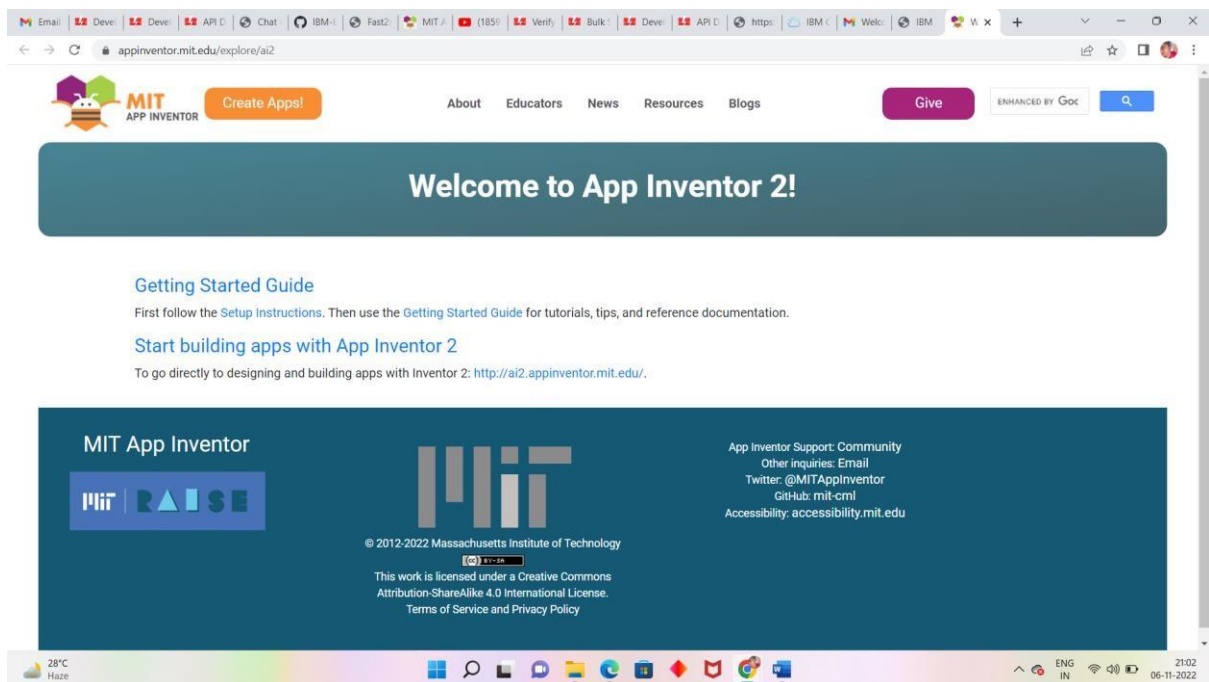


PRE REQUISITES:

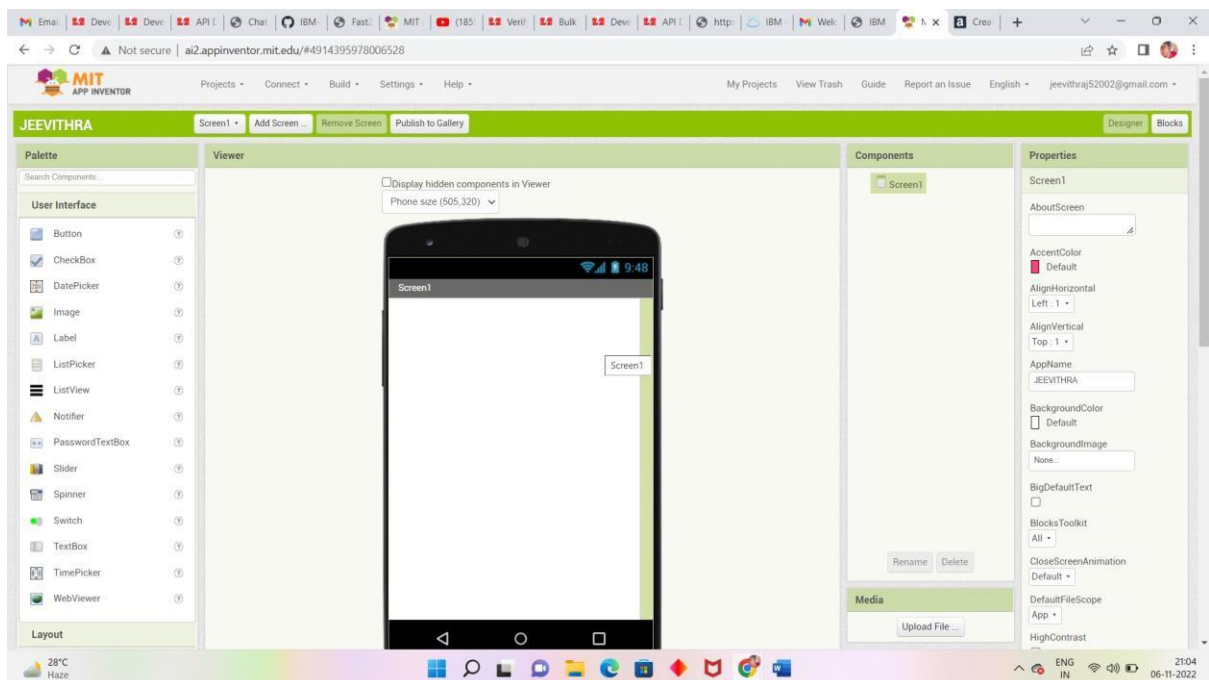
IBM CLOUD SERVICES:



MIT APP INVENTOR:

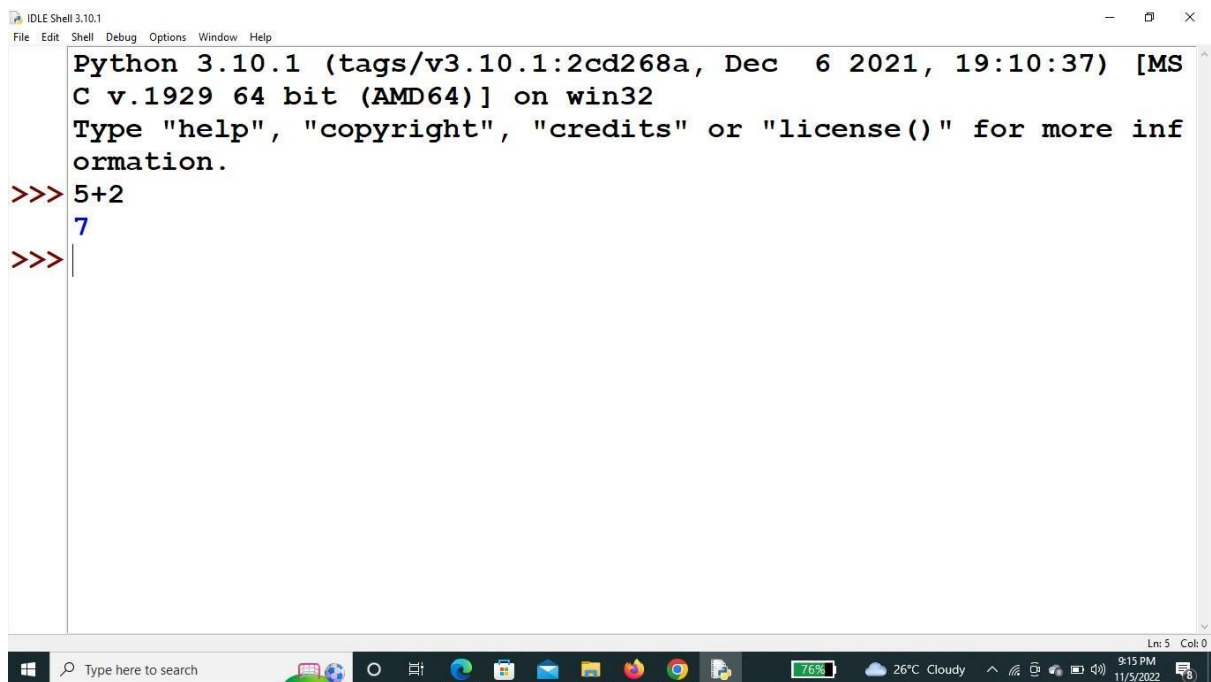


The screenshot shows the MIT App Inventor 2! homepage. At the top, there's a navigation bar with links for Email, Dev, API, Chat, IBM, Fast, MIT, (185), Verify, Bulk, Dev, API, and a search bar. Below the navigation bar is a large blue banner with the text "Welcome to App Inventor 2!". Underneath the banner, there's a section titled "Getting Started Guide" with a link to "Start building apps with App Inventor 2". Below this, there's a section titled "MIT App Inventor" with the MIT logo and the text "App Inventor Support: Community". To the right of the MIT logo, there's a list of links: "Other inquiries: Email", "Twitter: @MITAppInventor", "Git-Hub: mit-cml", and "Accessibility: accessibility.mit.edu". At the bottom of the page, there's a footer with the text "© 2012-2022 Massachusetts Institute of Technology" and "This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License. Terms of Service and Privacy Policy".



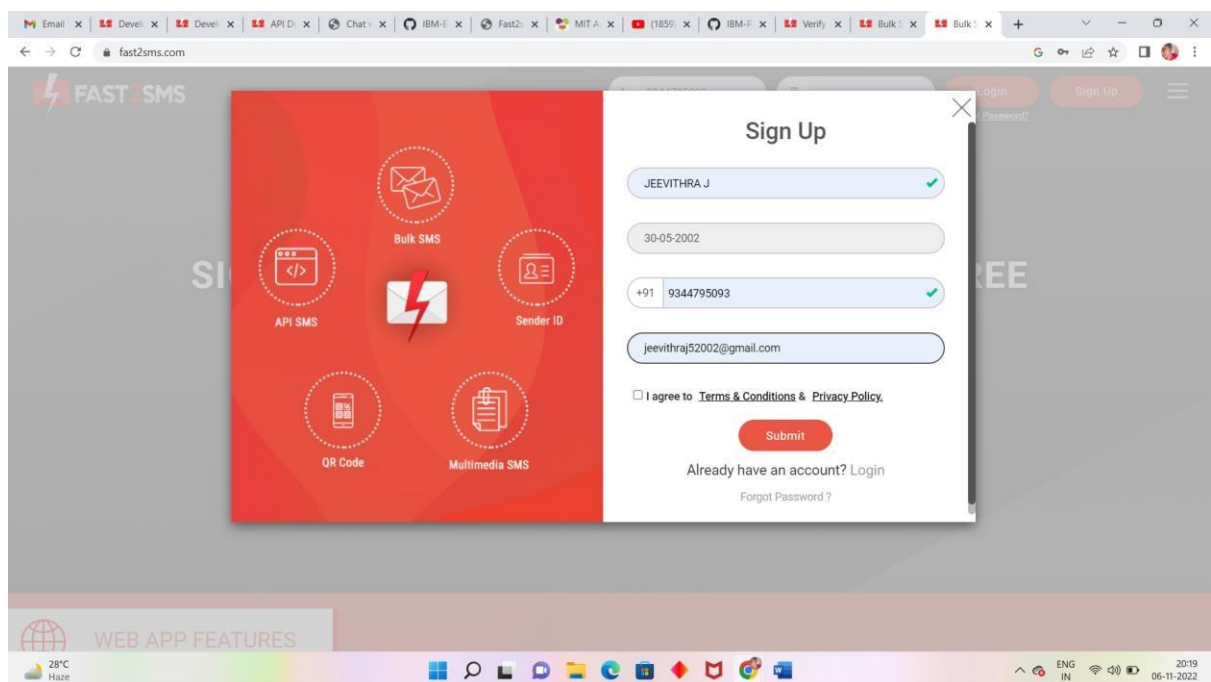
The screenshot shows the MIT App Inventor 2! IDE interface. The top bar includes a navigation menu with links for Projects, Connect, Build, Settings, and Help. Below the navigation menu, there's a section titled "JEEVITHRA" with a dropdown menu for "Screen1" and buttons for "Add Screen", "Remove Screen", and "Publish to Gallery". The main workspace is divided into four panels: "Palette" (containing a list of UI components like Button, CheckBox, DatePicker, Image, Label, ListPicker, ListView, Notifier, PasswordTextBox, Slider, Spinner, Switch, TextBox, TimePicker, and WebViewer), "Viewer" (showing a mobile app preview with a "Screen1" label), "Components" (showing a list of components for "Screen1"), and "Properties" (showing the properties for "Screen1", including AboutScreen, AccentColor, AlignHorizontal, AlignVertical, AppName, BackgroundColor, BackgroundImage, BigDefaultText, BlocksToolkit, CloseScreenAnimation, DefaultFileScope, and HighContrast). The bottom of the interface shows a Windows taskbar with various application icons and a system clock displaying 21:04 on 06-11-2022.

SOFTWARE:

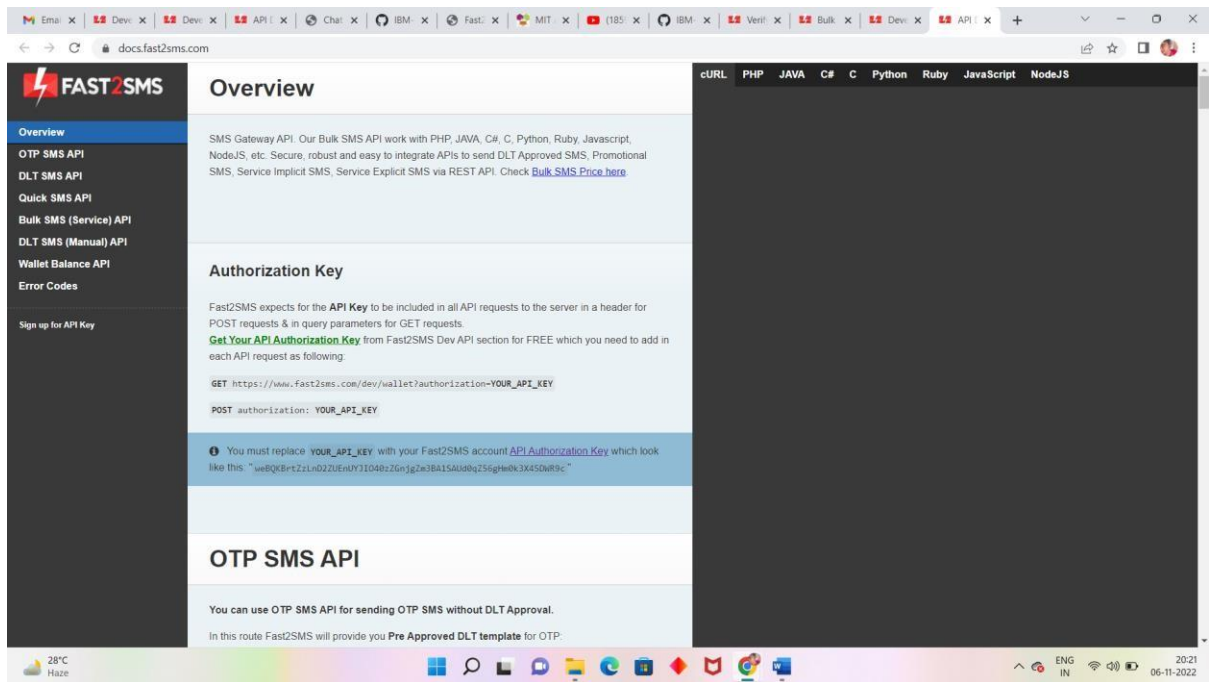


```
Python 3.10.1 (tags/v3.10.1:2cd268a, Dec 6 2021, 19:10:37) [MS
C v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more inf
ormation.
>>> 5+2
7
>>>
```

FAST2SMS:

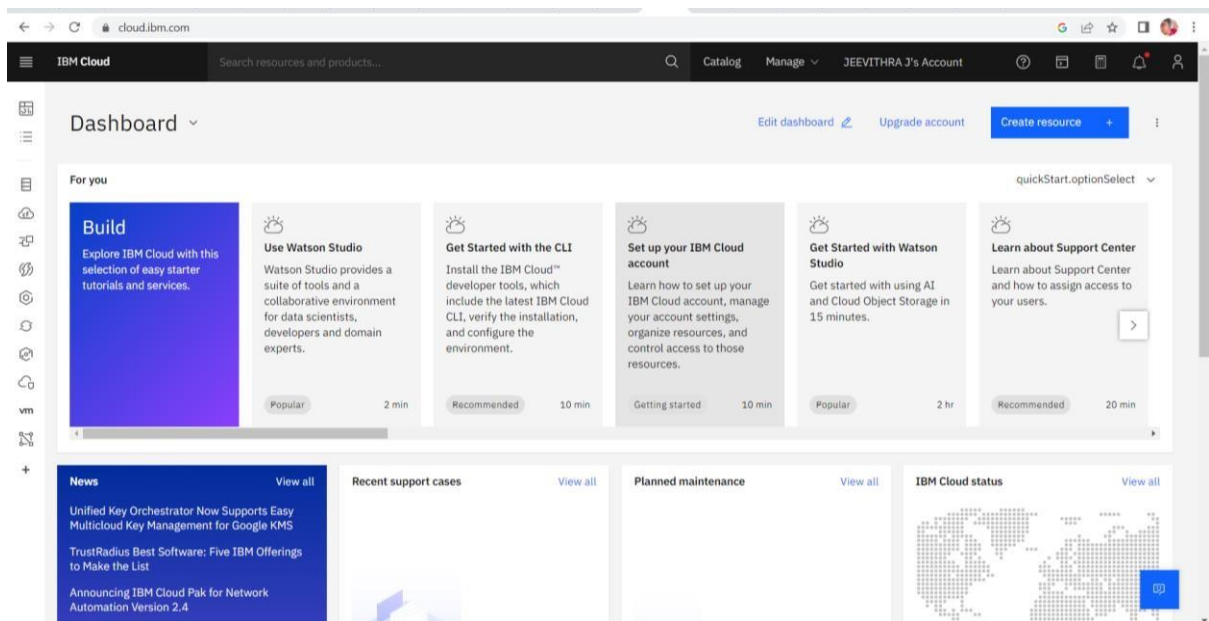


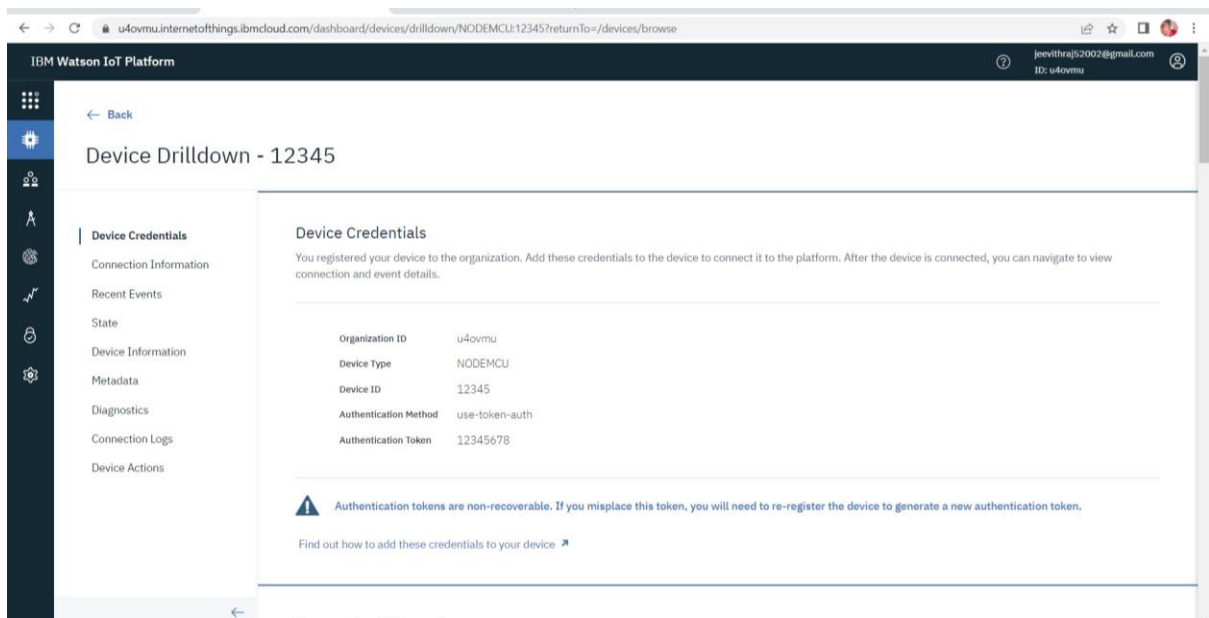
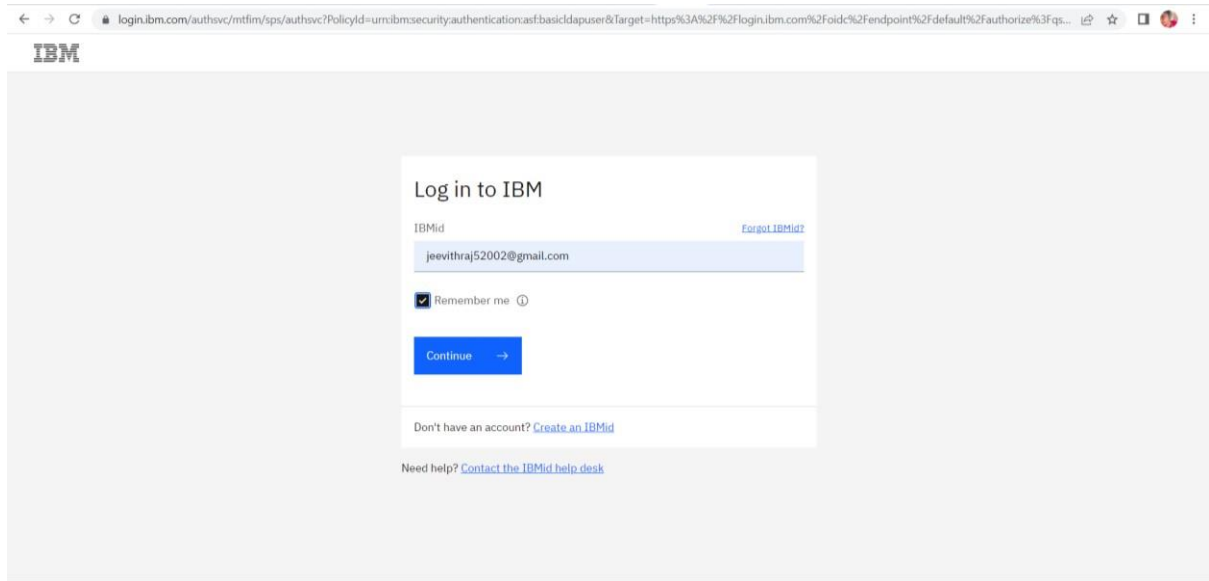
<https://www.fast2sms.com/dev/bulkV2?authorization=BaYHvngCRNXfljJFs0KhTLupe69zZy1G3SOMbPiDWVE2Ir7UcmvJsntUX3BV5RlZ1hmSjKicuPQW4w7D&route=q&message=HELLO&language=engli sh&flash=0&numbers=9344795093>



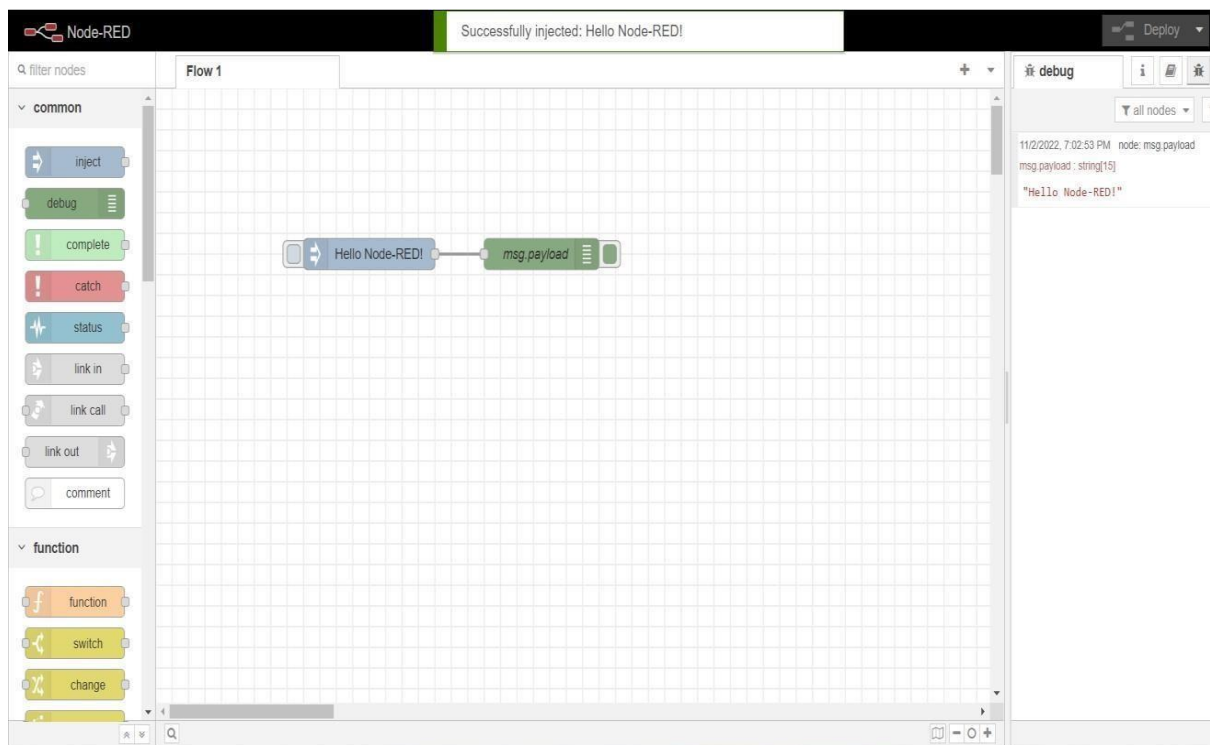
CREATE AND CONFIGURE IBM CLOUD SERVICES:

CREATE THE IBM WASTON IoT PLATFORM:





CREATE NODE-RED SERVICE:

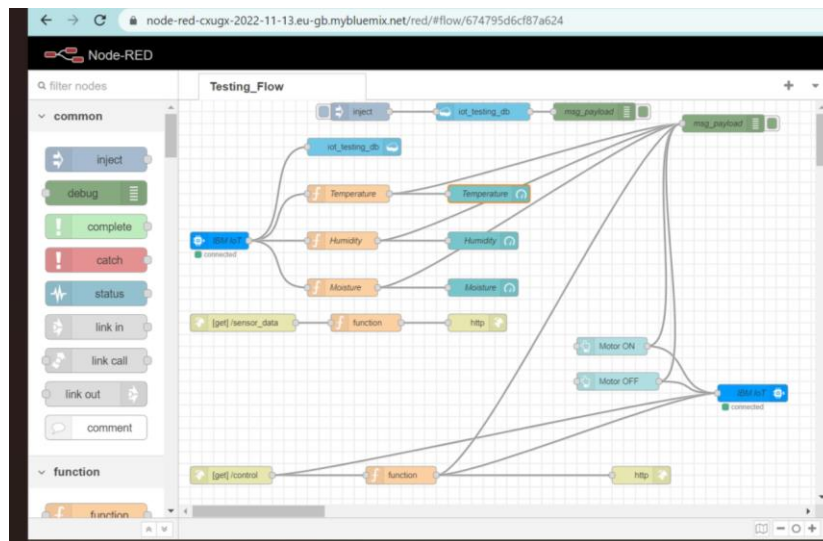


DEVELOP A PHYTHON SCRIPT TO PUBLISH AND SUBSCRIBE TO IBM IOT PLATFORM:

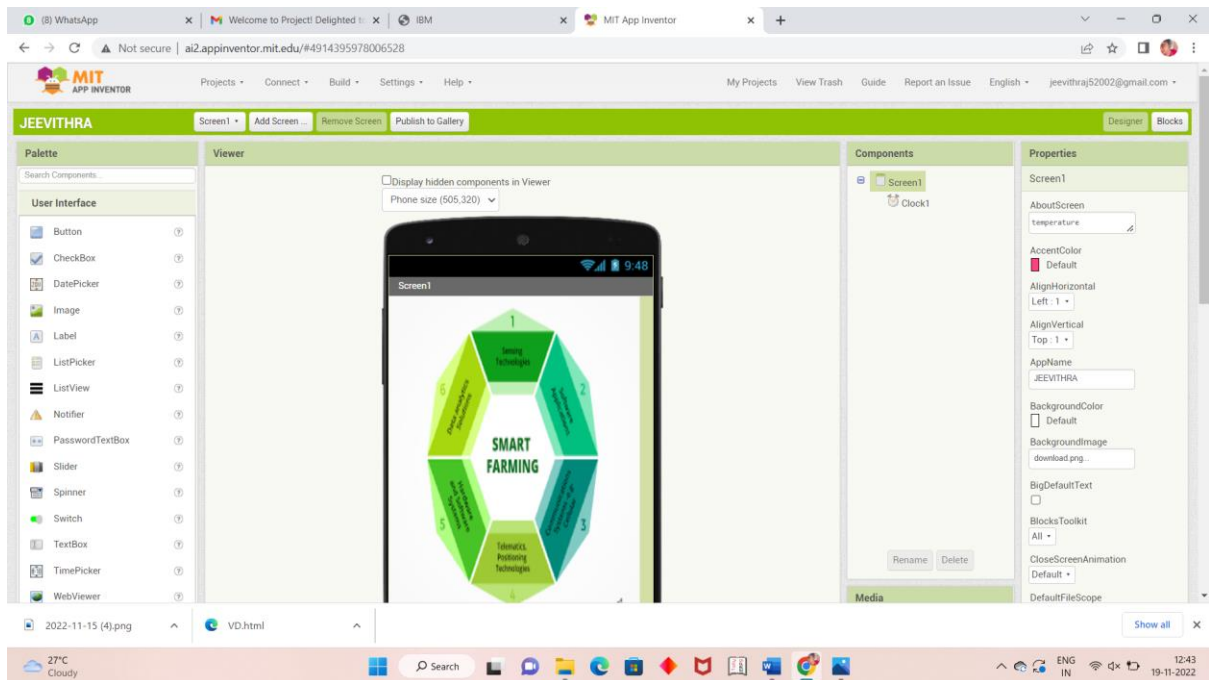
DEVELOP THE PYTHON CODE:

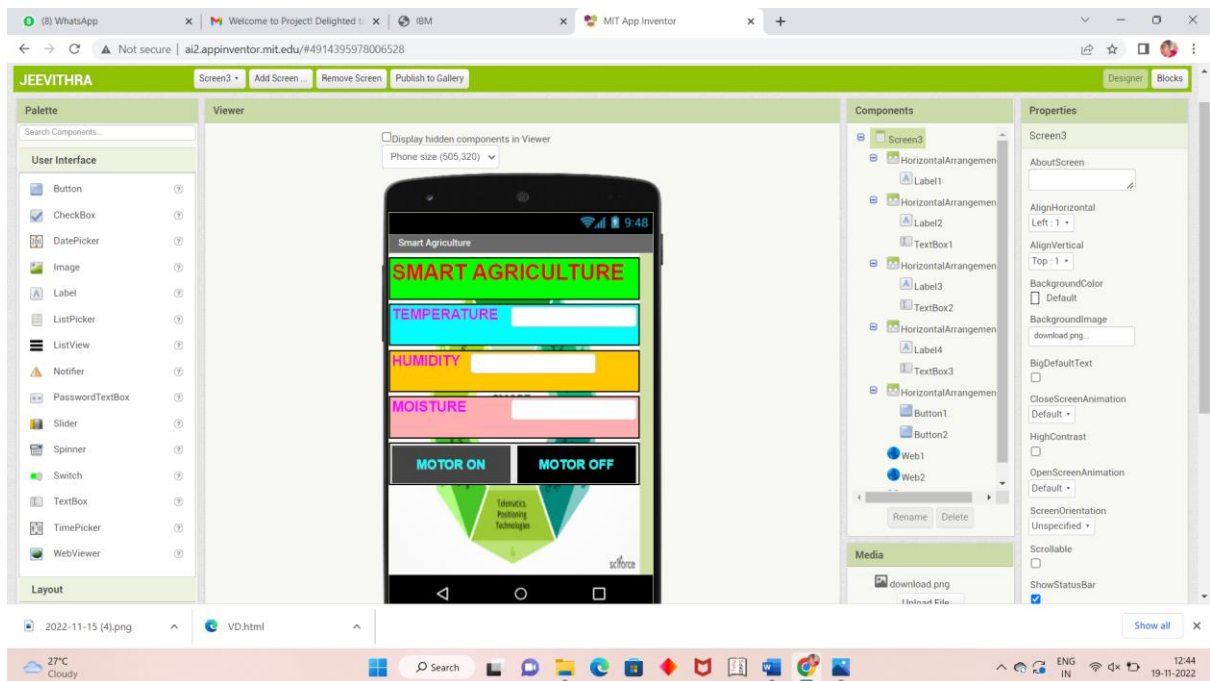
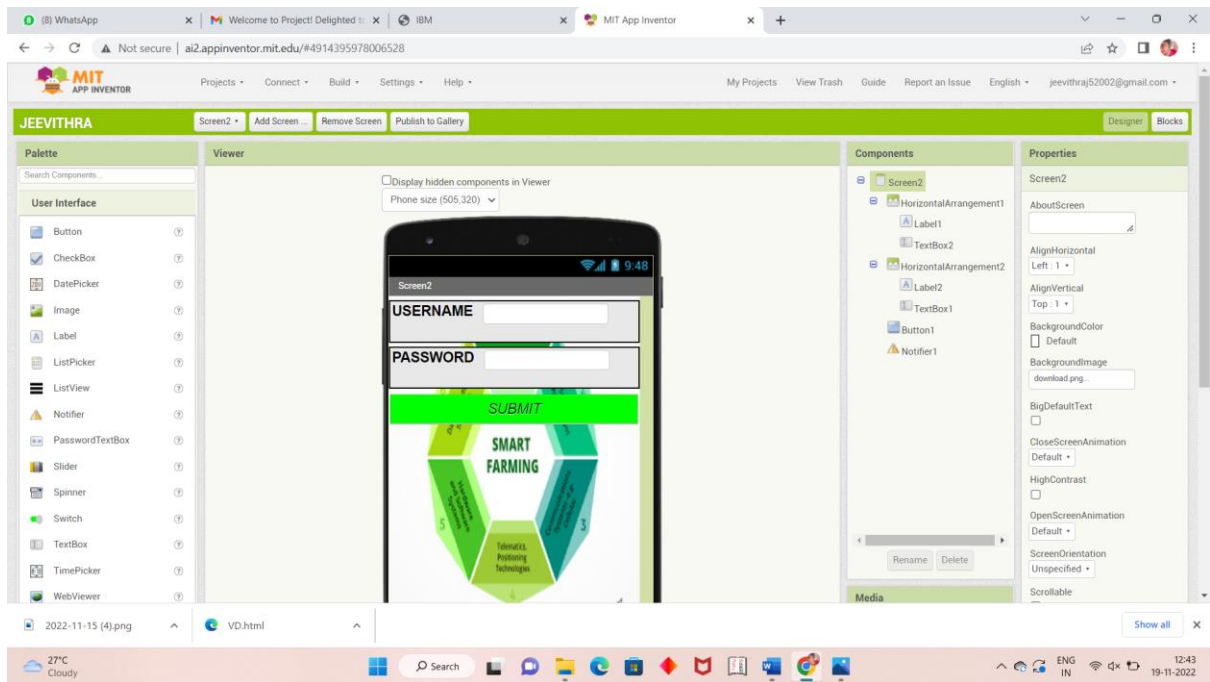
[illegible]

BUILD A WEB APPLICATION USING NODE-RED SERVICE:



DEVELOP A MOBILE APPLICATION:

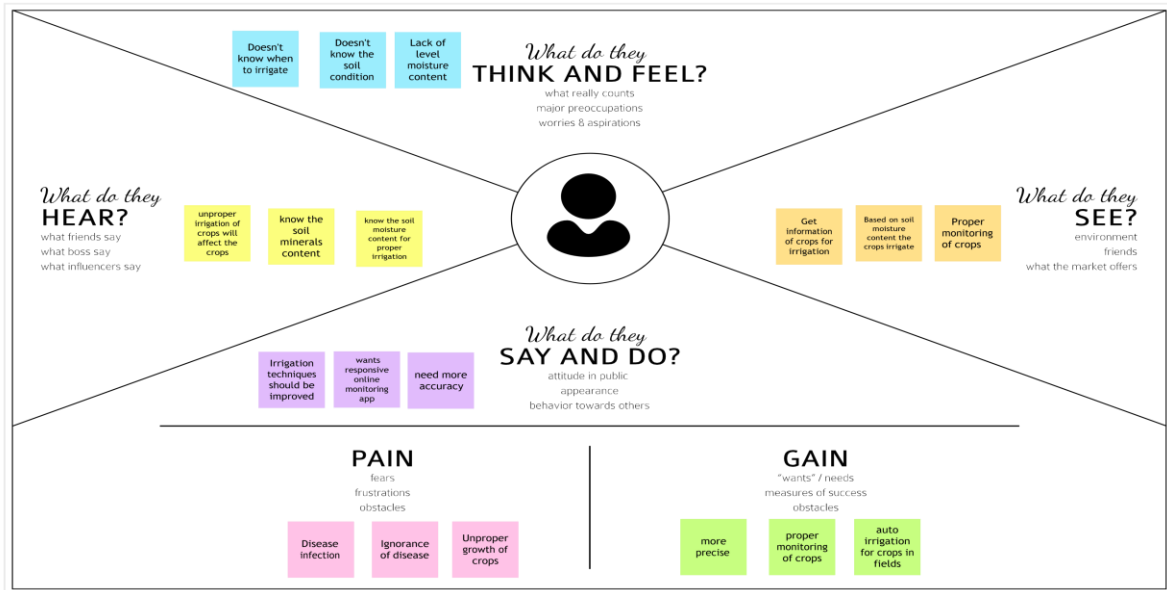




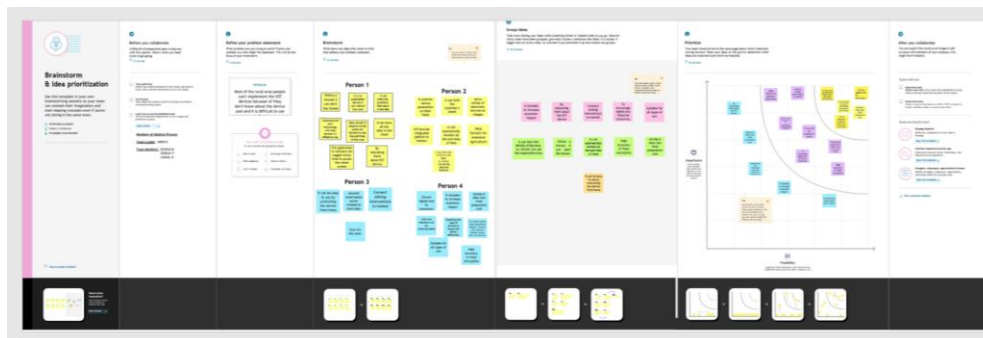
IDEATION:

EMPATHY MAP:

1 Empathy Map :

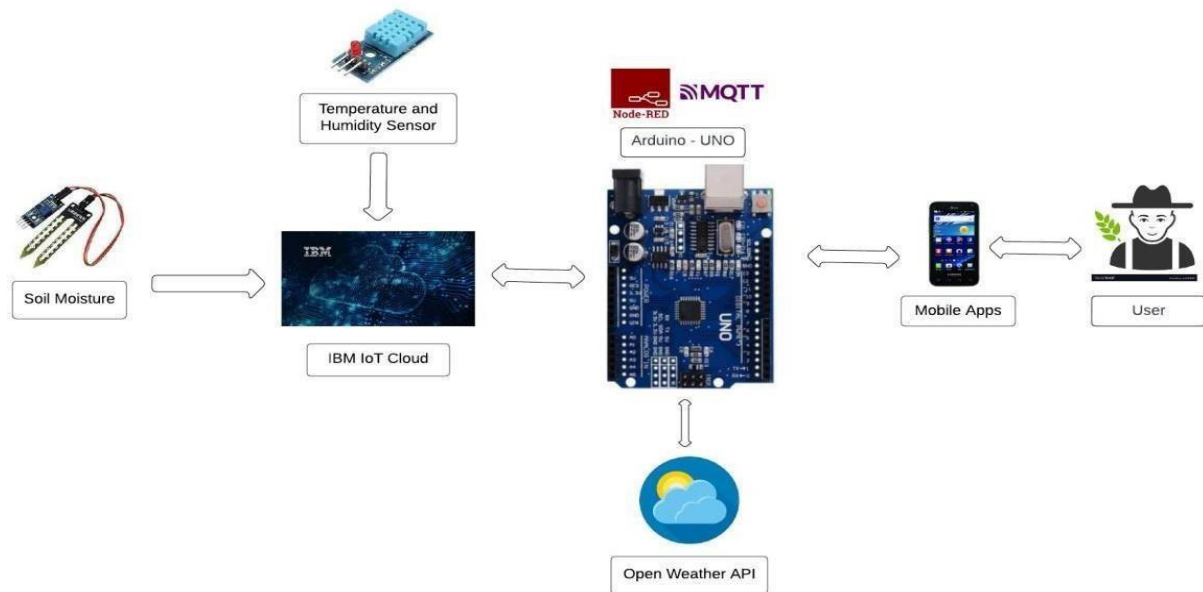


BRAINSTORM:



PROJECT DESIGN PHASE-I:

SOLUTION ARCHITECTURE:

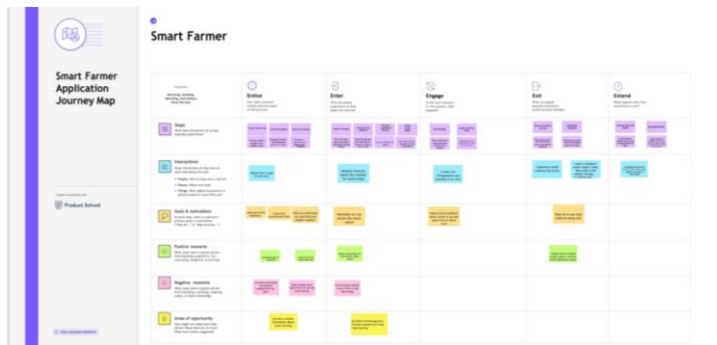


SOLUTION FIX:

<p>1. CUSTOMER SEGMENT(S) Who is your customer? I.e. working parents of 0-5 y.o. kids</p> <p>CS</p> <p>The customer for this product is a farmer who grows crops. Our goal is to help them, monitor field parameters remotely. This product saves agriculture from extinction.</p>	<p>6. CUSTOMER CONSTRAINTS</p> <p>CC</p> <p>What constraints prevent your customers from taking action or limit their choices of solutions? I.e. spending power, budget, no cash, network connection, available devices.</p> <p>Using a large number of sensors is difficult. An unlimited or continuous internet connection is required for success.</p>	<p>5. AVAILABLE SOLUTIONS</p> <p>AS</p> <p>Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? I.e. pen and paper</p> <p>The irrigation process is automated using IoT. Meteorological data and field parameters were collected and processed to automate the irrigation process. Disadvantages are efficiency only over short distances, and difficult data storage.</p>
<p>2. JOBS-TO-BE-DONE / PROBLEMS Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides.</p> <p>J&P</p> <p>The purpose of this product is to use sensors to acquire various field parameters and process them using a central processing system. The cloud is used to store and transmit data using IoT. The Weather API is used to help farmers make decisions. Farmers can make decisions through mobile applications.</p>	<p>9. PROBLEM ROOT CAUSE</p> <p>RC</p> <p>What is the real reason that this problem exists? What is the back story behind the need to do this job?</p> <p>Frequent changes and unpredictable weather and climate made it difficult for farmers to engage in agriculture. These factors play an important role in deciding whether to water your plants. Fields are difficult to monitor when the farmer is not at the field, leading to crop damage.</p>	<p>7. BEHAVIOUR</p> <p>BE</p> <p>What does your customer do to address the problem and get the job done? I.e. directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (I.e. Greenpeace)</p> <p>Use a proper drainage system to overcome the effects of excess water from heavy rain. Use of hybrid plants that are resistant to pests.</p>

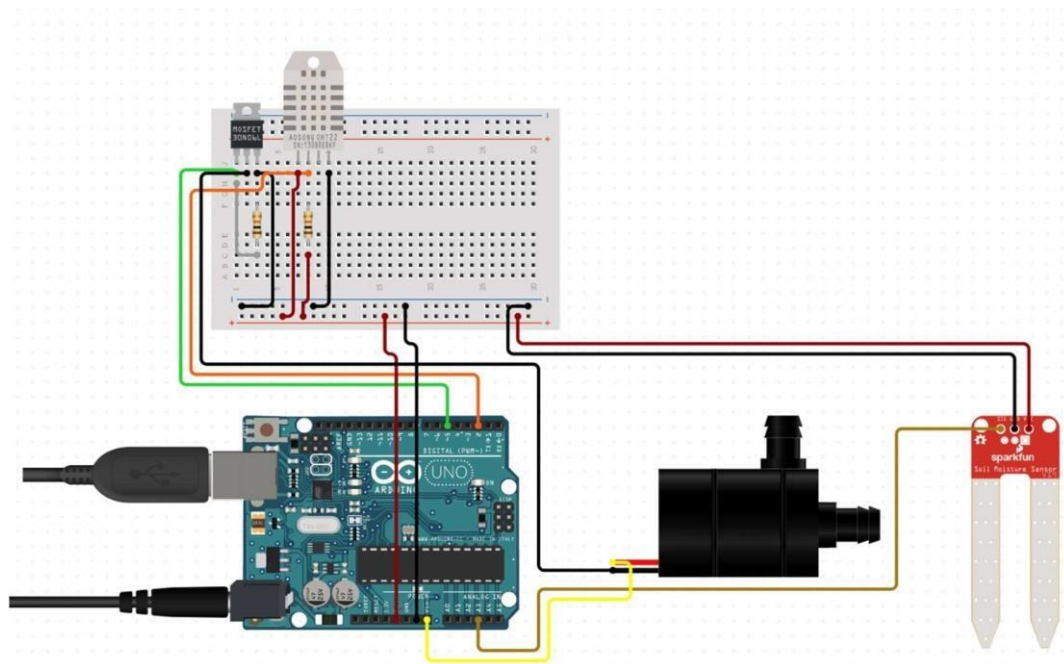
PROJECT DESIGN PHASE-II:

CUSTOMER JOURNEY:

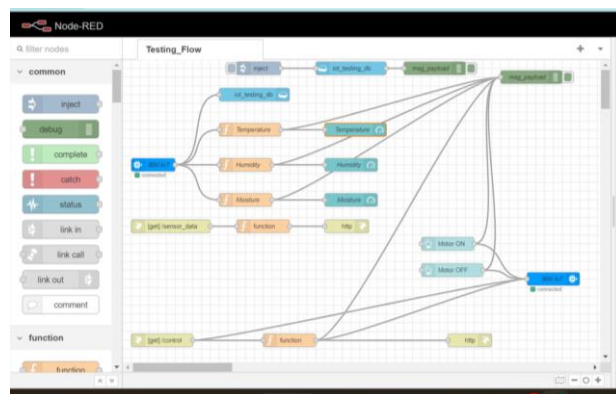
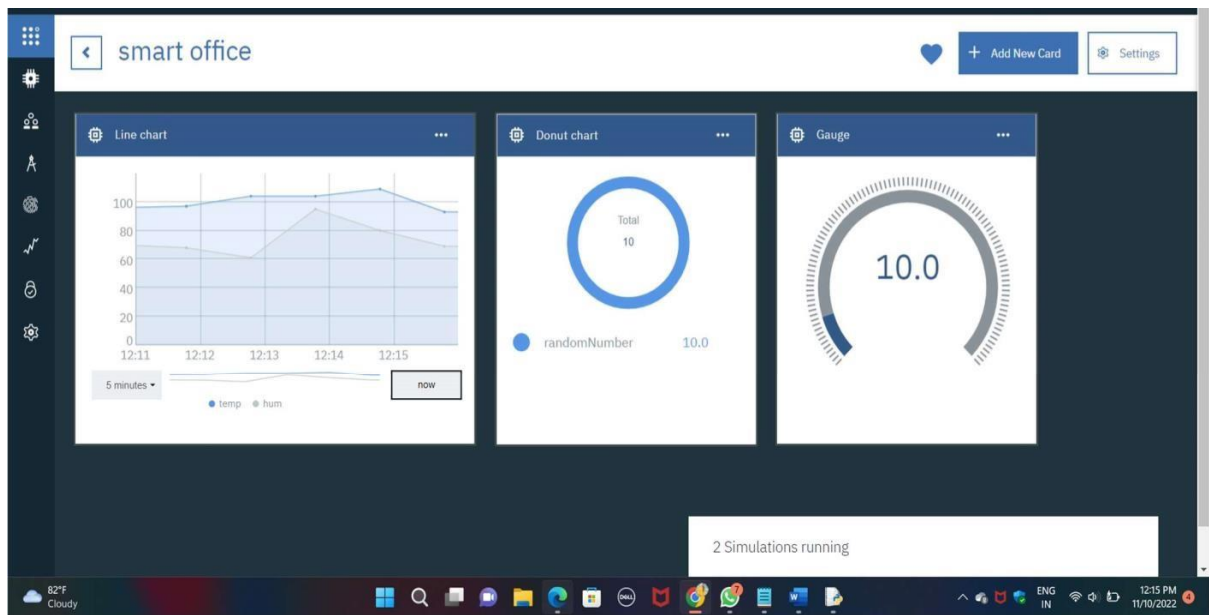


PROJECT DEVELOPMENT PHASE:

SPRINT-1:



SPRINT-2:



SPRINT-3:

MIT App Inventor interface showing the design and code for a Smart Farming application.

Design View:

- Screen3:** Contains three text boxes (TextBox1, TextBox2, TextBox3) and a button (Button1).
- Media:** Includes a download button and a file upload area.

Code View:

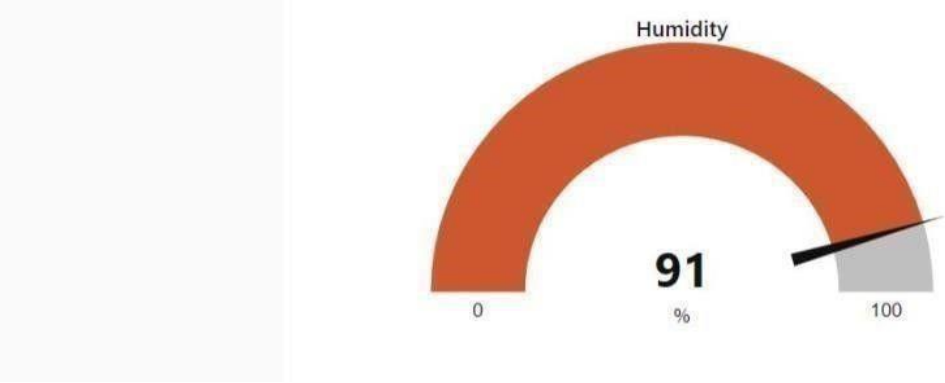
```
when Web1 GotText
do
  set TextBox1 Text to look up in pairs key Temperature pairs call Web1 JsonTextDecode jsonText get responseContent
  set TextBox2 Text to look up in pairs key Humidity pairs call Web1 JsonTextDecode jsonText get responseContent
  set TextBox3 Text to look up in pairs key Moisture pairs call Web1 JsonTextDecode jsonText get responseContent
  notFound not found
when Button1 Click
do
  set Web2 Uri to *
  call Web2 Get
```

Smart Farming Dashboard:

- Farming:** Displays Humidity and Temp line graphs. Humidity and Temp both show a decrease from 12:43:00 to 12:45:00, followed by an increase at 12:48:00. A "LIGHT OFF" button is present.
- Garden:** Displays Moisture level as a gauge showing 77 units.
- Switch Board:** Displays a "LIGHT ON" button.

SPRINT-4:

TEMPERATURE:



HUMIDITY:

MOISTURE:



