Project Development Phase
Sprint-1

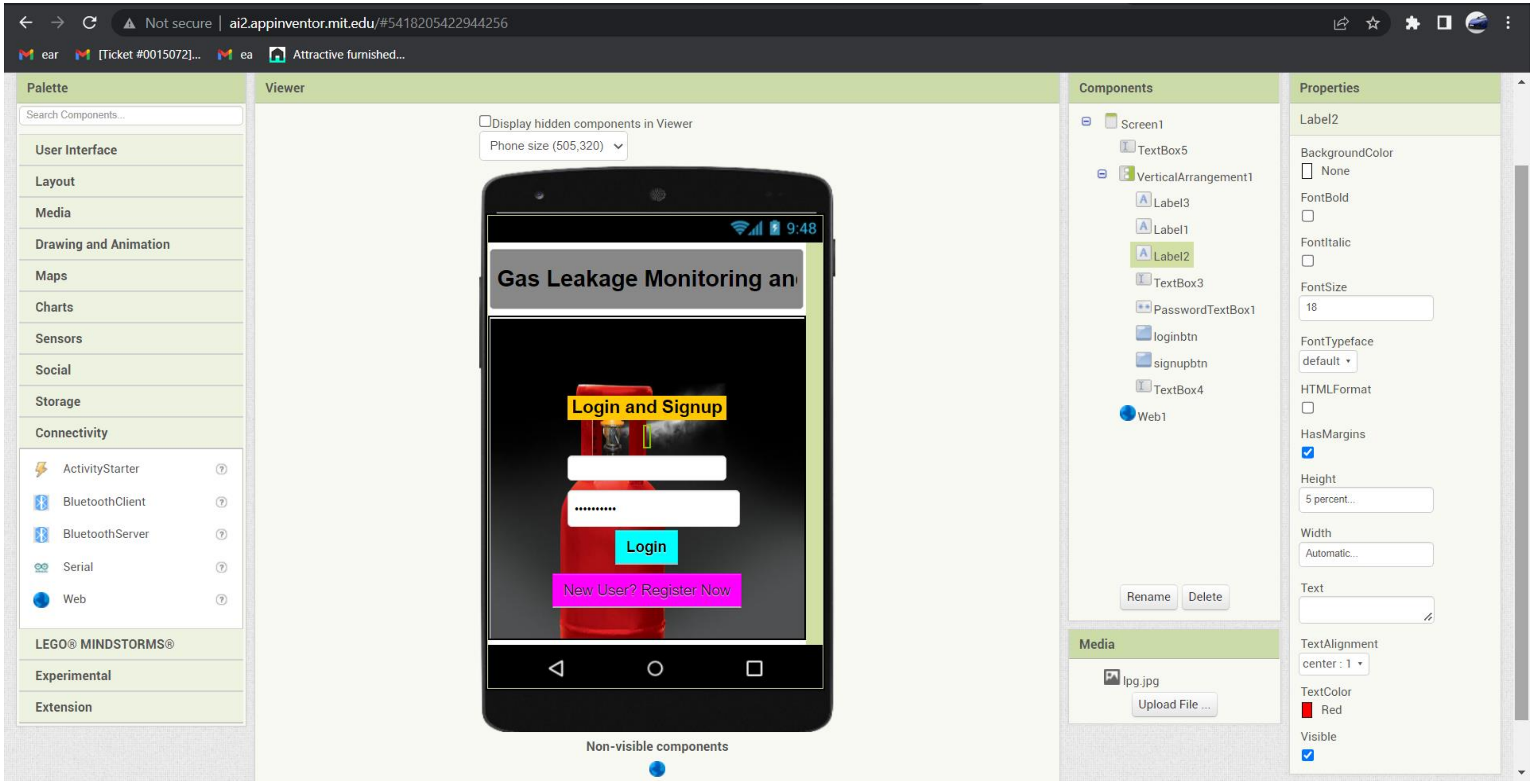
Date	2 November 2022
Team ID	PNT2022TMID35841
Project Name	Gas Leakage Monitoring and Alerting System

Sprint Target:

Sprint	**Functional Requirement (Epic)**	**User Story Number**	**User Story / Task**	**Story Points**	**Priority**	**Team Members**
Sprint-1	Registration	USN-1	As a user, I can start by registering my credentials so that I get notifications in case of a gas leakage	10	High	SUDARSHAN A R, B ADITYA, C PRADUMNA
		USN-2	As a user, I will receive confirmation once I have registered for the application.	5	High	SUDARSHAN A R, B ADITYA, C PRADUMNA
	Login	USN-3	As a user, I can log into the application by entering my credentials	5	High	SUDARSHAN A R, B ADITYA, C PRADUMNA

Introduction:

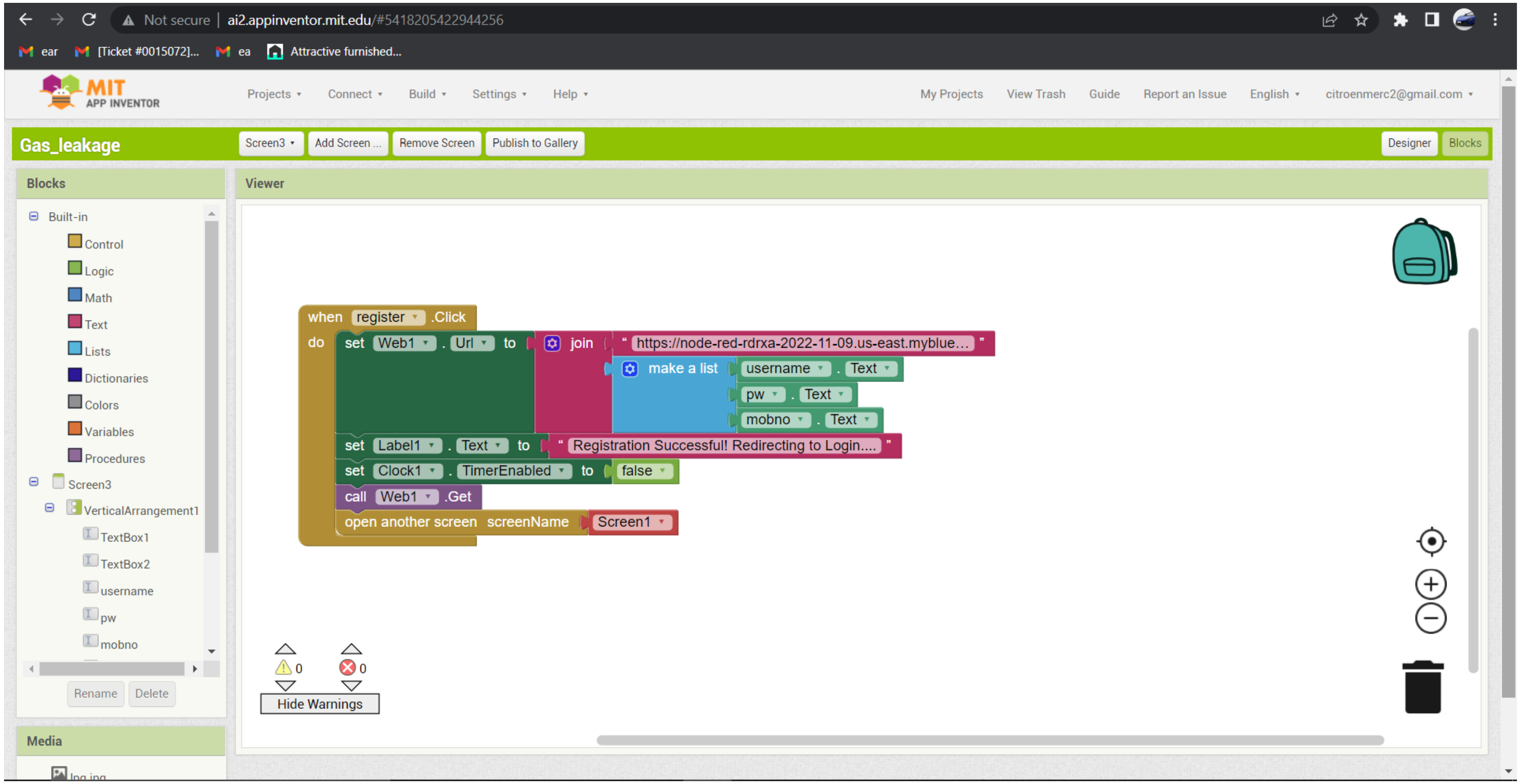
In this Sprint-1, we have created a Gas Leakage Monitoring and Leakage Detection application using MIT app inventor 2 and Node-RED API which allows user registration and login.



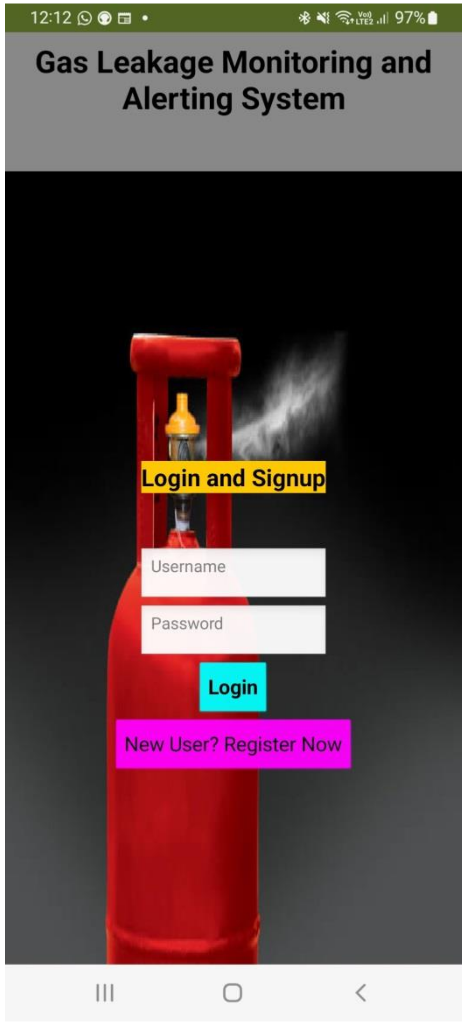
BackgroundColor: None
FontBold: ☐
FontItalic: ☐
FontSize: 18
FontTypeface: default
HTMLFormat: ☐
HasMargins: ☑
Height: 5 percent...
Width: Automatic...
Text:
TextAlignment: center : 1
TextColor: 🟥 Red
Visible: ☑

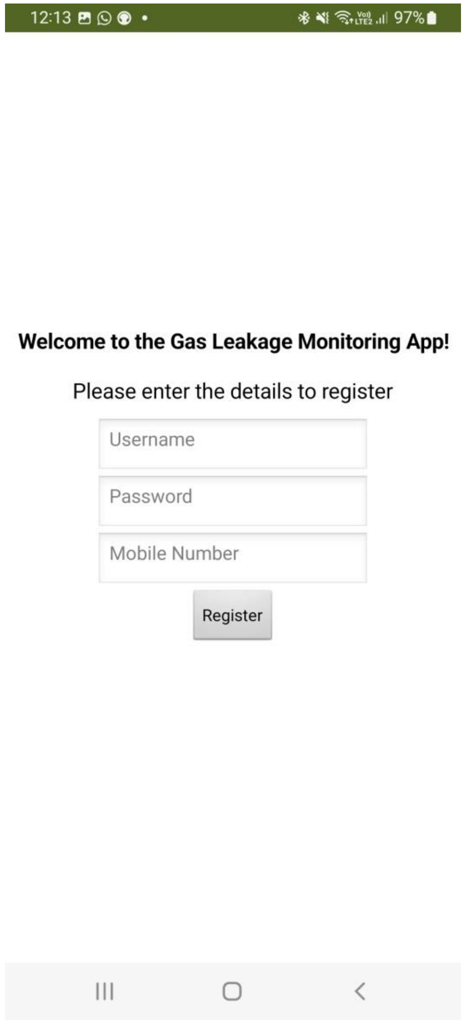
Registration/Signup

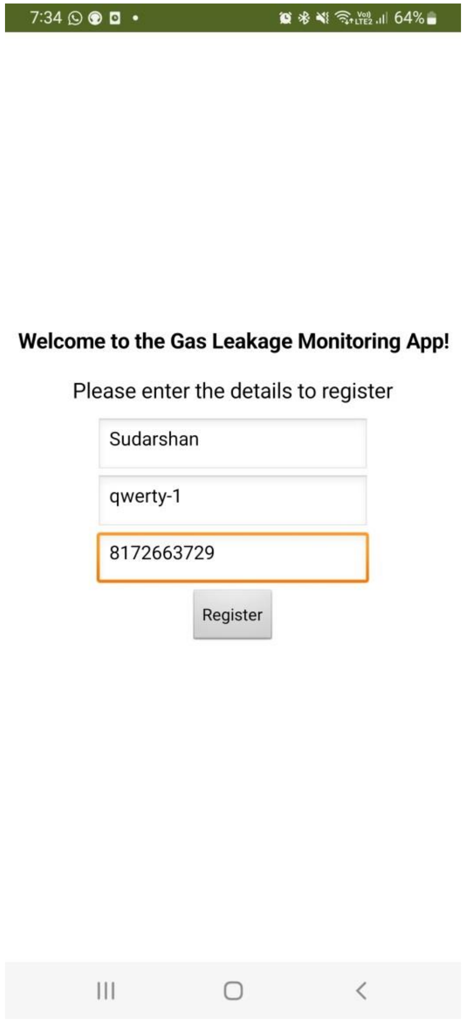
New users to the app can enter their username, password and mobile number to register.



```
when register .Click
do  set Web1 . Url to  join  " https://node-red-rdrxa-2022-11-09.us-east.myblue... "
                             make a list  username . Text
                                          pw . Text
                                          mobno . Text
    set Label1 . Text to " Registration Successful! Redirecting to Login.... "
    set Clock1 . TimerEnabled to false
    call Web1 .Get
    open another screen screenName Screen1
```




12:13 97%
Welcome to the Gas Leakage Monitoring App!
Please enter the details to register
Username
Password
Mobile Number
Register

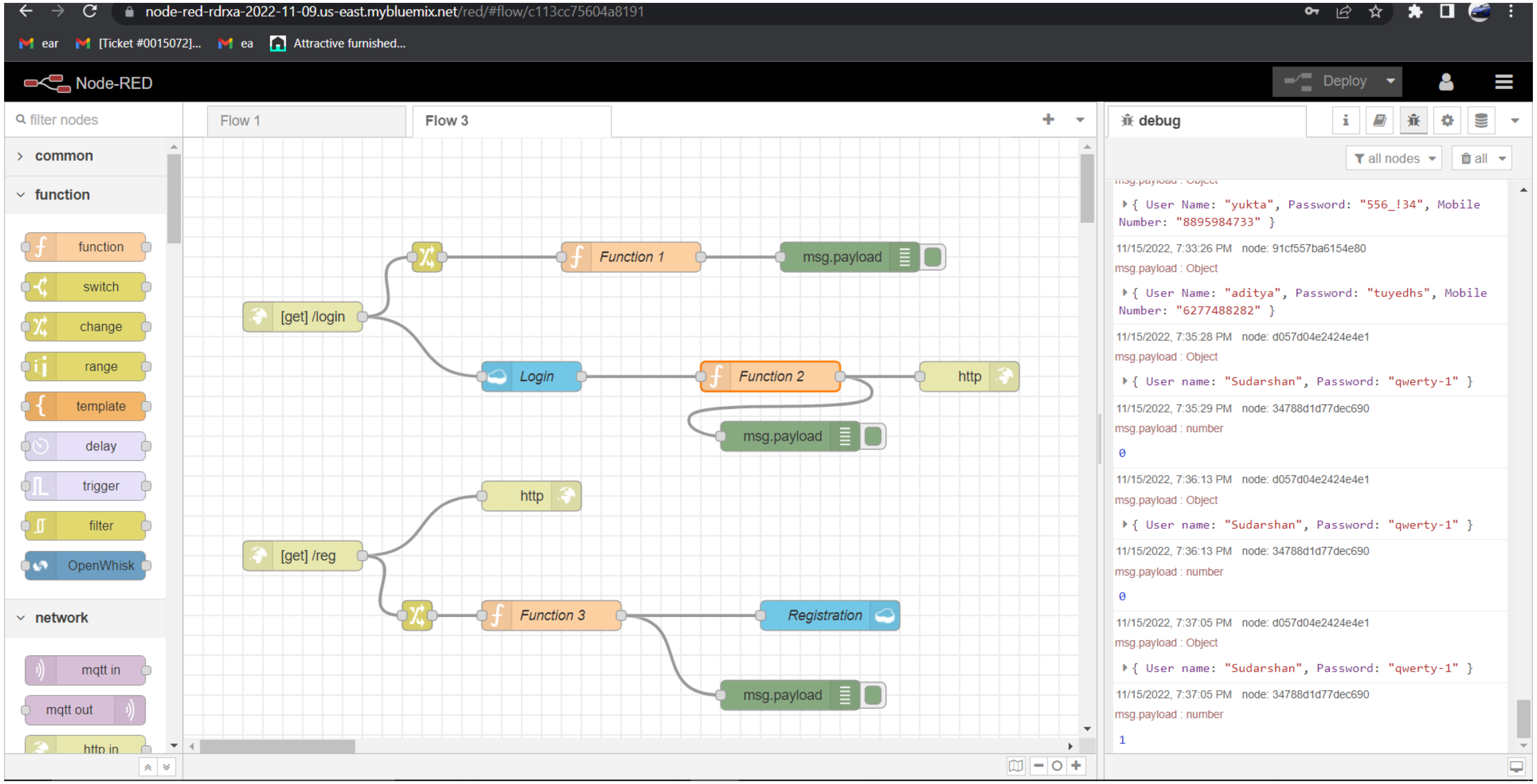

7:34 64%
Welcome to the Gas Leakage Monitoring App!
Please enter the details to register

Sudarshan

qwerty-1

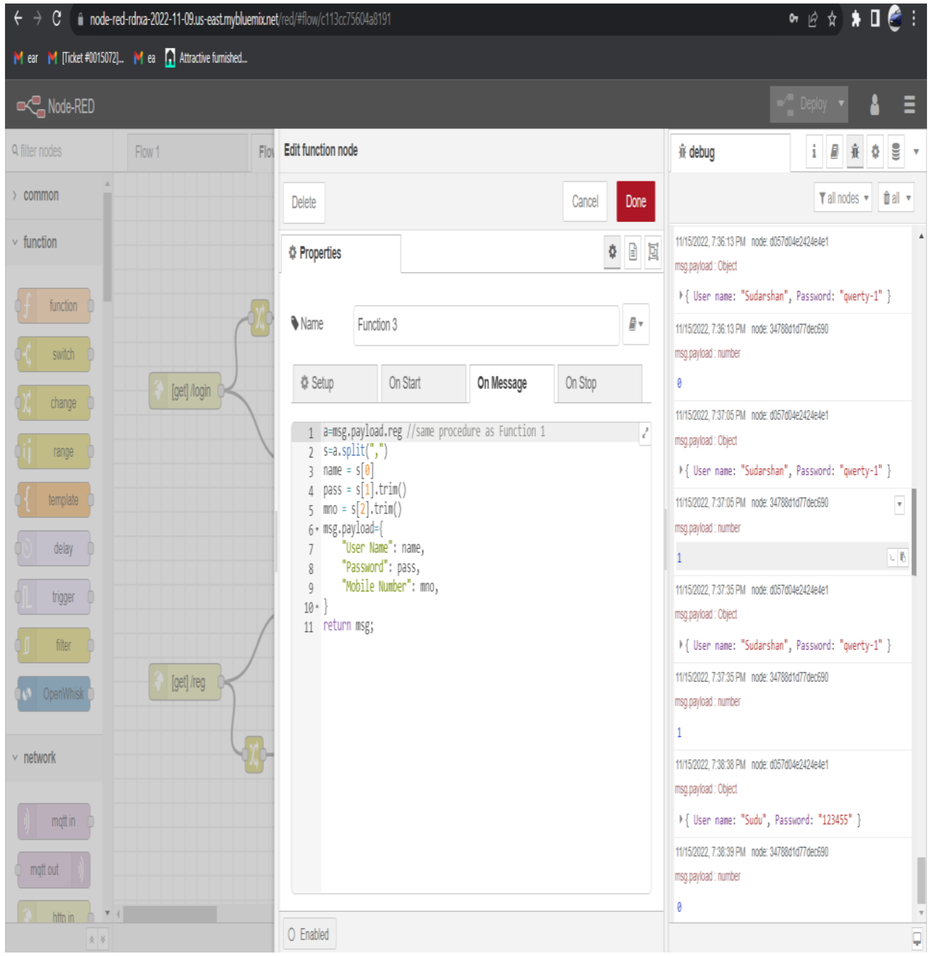
8172663729

Using Node-RED the entered details are stored in the Cloudant Database.



```
msg.payload : Object
▸{ User Name: "yukta", Password: "556_!34", Mobile
Number: "8895984733" }
11/15/2022, 7:33:26 PM   node: 91cf557ba6154e80
msg.payload : Object
▸{ User Name: "aditya", Password: "tuyedhs", Mobile
Number: "6277488282" }
11/15/2022, 7:35:28 PM   node: d057d04e2424e4e1
msg.payload : Object
▸{ User name: "Sudarshan", Password: "qwerty-1" }
11/15/2022, 7:35:29 PM   node: 34788d1d77dec690
msg.payload : number
0
11/15/2022, 7:36:13 PM   node: d057d04e2424e4e1
msg.payload : Object
▸{ User name: "Sudarshan", Password: "qwerty-1" }
11/15/2022, 7:36:13 PM   node: 34788d1d77dec690
msg.payload : number
0
11/15/2022, 7:37:05 PM   node: d057d04e2424e4e1
msg.payload : Object
▸{ User name: "Sudarshan", Password: "qwerty-1" }
11/15/2022, 7:37:05 PM   node: 34788d1d77dec690
msg.payload : number
1
```

Function 3 and the change node modify the message payload to store in proper format in the database.

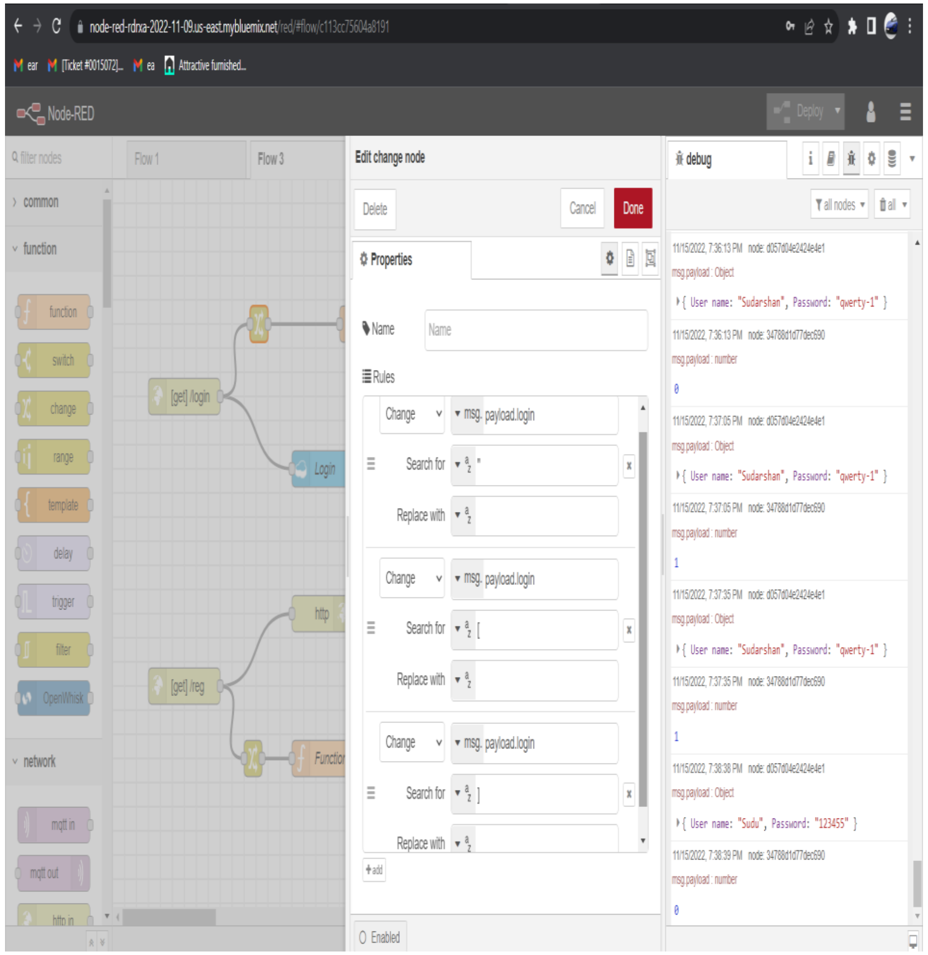


Edit function node

Name: Function 3

```
a=msg.payload.reg //same procedure as Function 1
s=a.split(",")
name = s[0]
pass = s[1].trim()
mno = s[2].trim()
msg.payload={
    "User Name": name,
    "Password": pass,
    "Mobile Number": mno,
}
return msg;
```

```
11/15/2022, 7:36:13 PM   node: d057d04e2424e4e1
msg.payload : Object
▸{ User name: "Sudarshan", Password: "qwerty-1" }
11/15/2022, 7:36:13 PM   node: 34788d1d77dec690
msg.payload : number
0
11/15/2022, 7:37:05 PM   node: d057d04e2424e4e1
msg.payload : Object
▸{ User name: "Sudarshan", Password: "qwerty-1" }
11/15/2022, 7:37:05 PM   node: 34788d1d77dec690
msg.payload : number
1
11/15/2022, 7:37:35 PM   node: d057d04e2424e4e1
msg.payload : Object
▸{ User name: "Sudarshan", Password: "qwerty-1" }
11/15/2022, 7:37:35 PM   node: 34788d1d77dec690
msg.payload : number
1
11/15/2022, 7:38:38 PM   node: d057d04e2424e4e1
msg.payload : Object
▸{ User name: "Sudu", Password: "123455" }
11/15/2022, 7:38:39 PM   node: 34788d1d77dec690
msg.payload : number
0
```



Edit change node

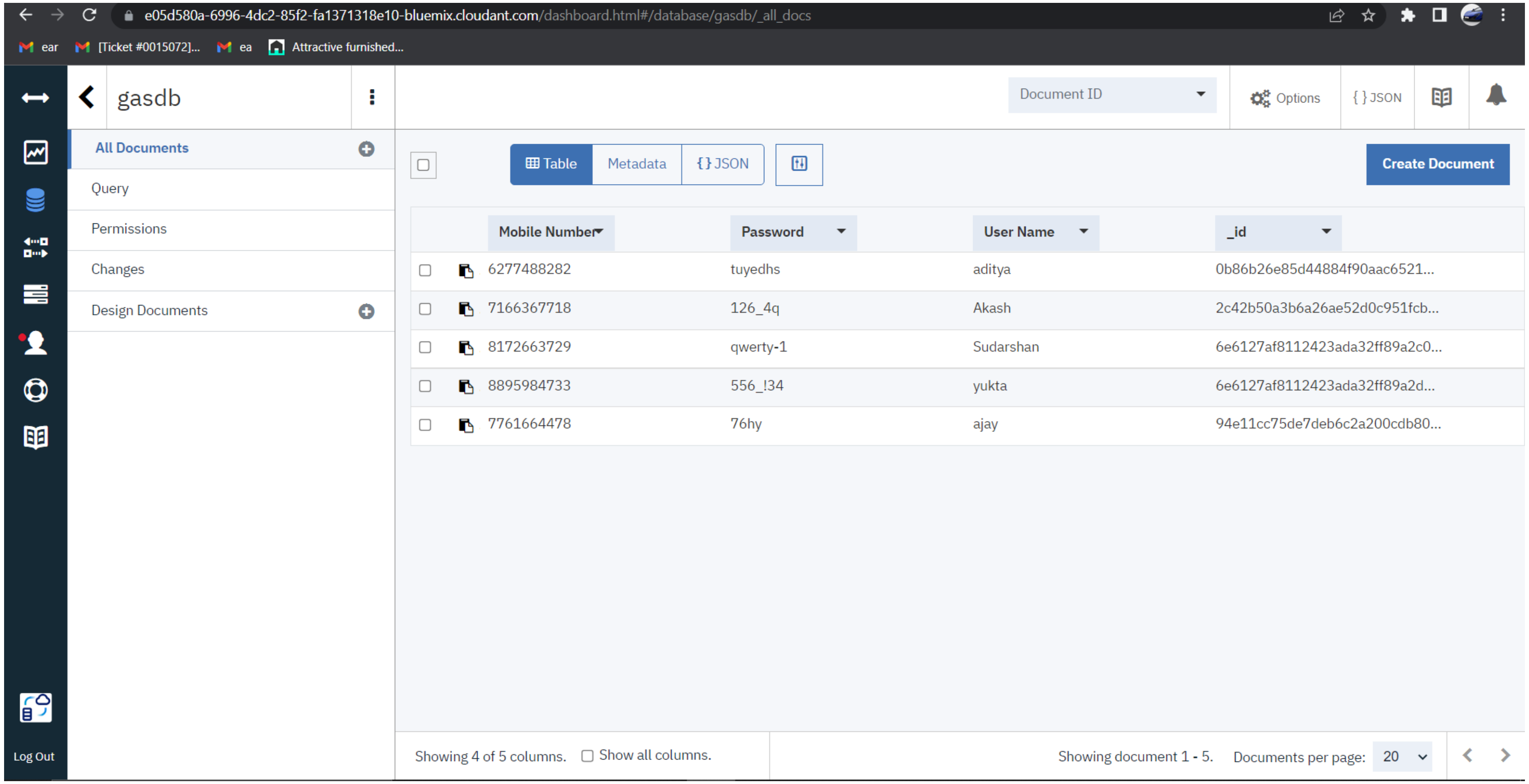
Name: Name

Rules

Change	msg. payload.login	
	Search for	"
	Replace with	
Change	msg. payload.login	
	Search for	[
	Replace with	
Change	msg. payload.login	
	Search for	]
	Replace with	

```
11/15/2022, 7:36:13 PM   node: d057d04e2424e4e1
msg.payload : Object
▸{ User name: "Sudarshan", Password: "qwerty-1" }
11/15/2022, 7:36:13 PM   node: 34788d1d77dec690
msg.payload : number
0
11/15/2022, 7:37:05 PM   node: d057d04e2424e4e1
msg.payload : Object
▸{ User name: "Sudarshan", Password: "qwerty-1" }
11/15/2022, 7:37:05 PM   node: 34788d1d77dec690
msg.payload : number
1
11/15/2022, 7:37:35 PM   node: d057d04e2424e4e1
msg.payload : Object
▸{ User name: "Sudarshan", Password: "qwerty-1" }
11/15/2022, 7:37:35 PM   node: 34788d1d77dec690
msg.payload : number
1
11/15/2022, 7:38:38 PM   node: d057d04e2424e4e1
msg.payload : Object
▸{ User name: "Sudu", Password: "123455" }
11/15/2022, 7:38:39 PM   node: 34788d1d77dec690
msg.payload : number
0
```

The updated database is shown below:



e05d580a-6996-4dc2-85f2-fa1371318e10-bluemix.cloudant.com/dashboard.html#/database/gasdb/_all_docs

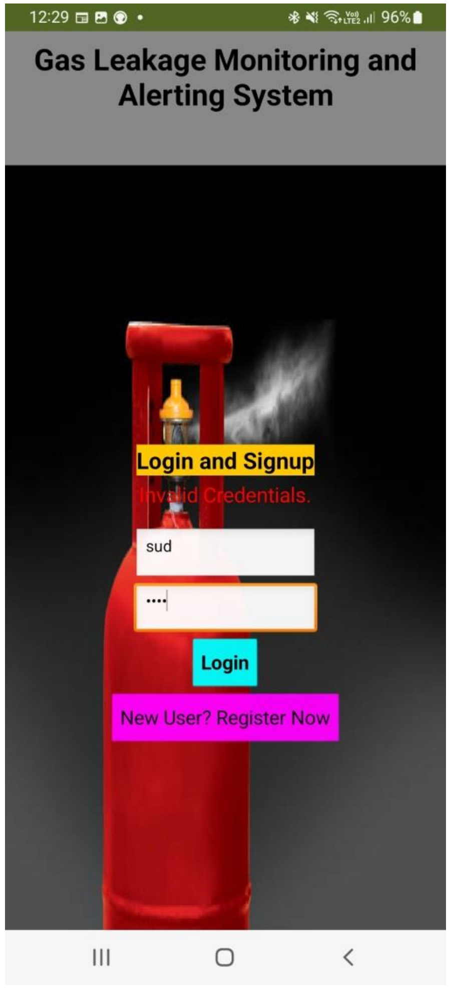
gasdb

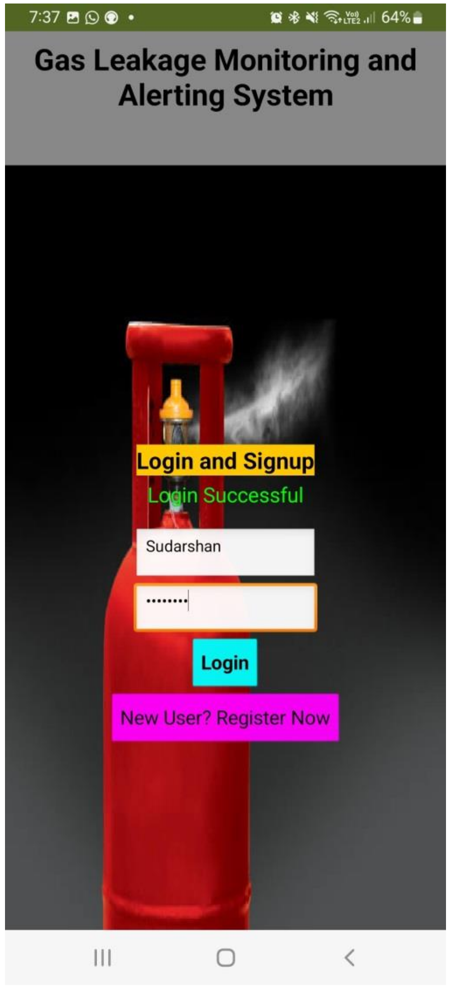
	Mobile Number	Password	User Name	_id
	6277488282	tuyedhs	aditya	0b86b26e85d44884f90aac6521...
	7166367718	126_4q	Akash	2c42b50a3b6a26ae52d0c951fcb...
	8172663729	qwerty-1	Sudarshan	6e6127af8112423ada32ff89a2c0...
	8895984733	556_!34	yukta	6e6127af8112423ada32ff89a2d...
	7761664478	76hy	ajay	94e11cc75de7deb6c2a200cdb80...

Showing 4 of 5 columns. Show all columns. Showing document 1 - 5. Documents per page: 20

Login

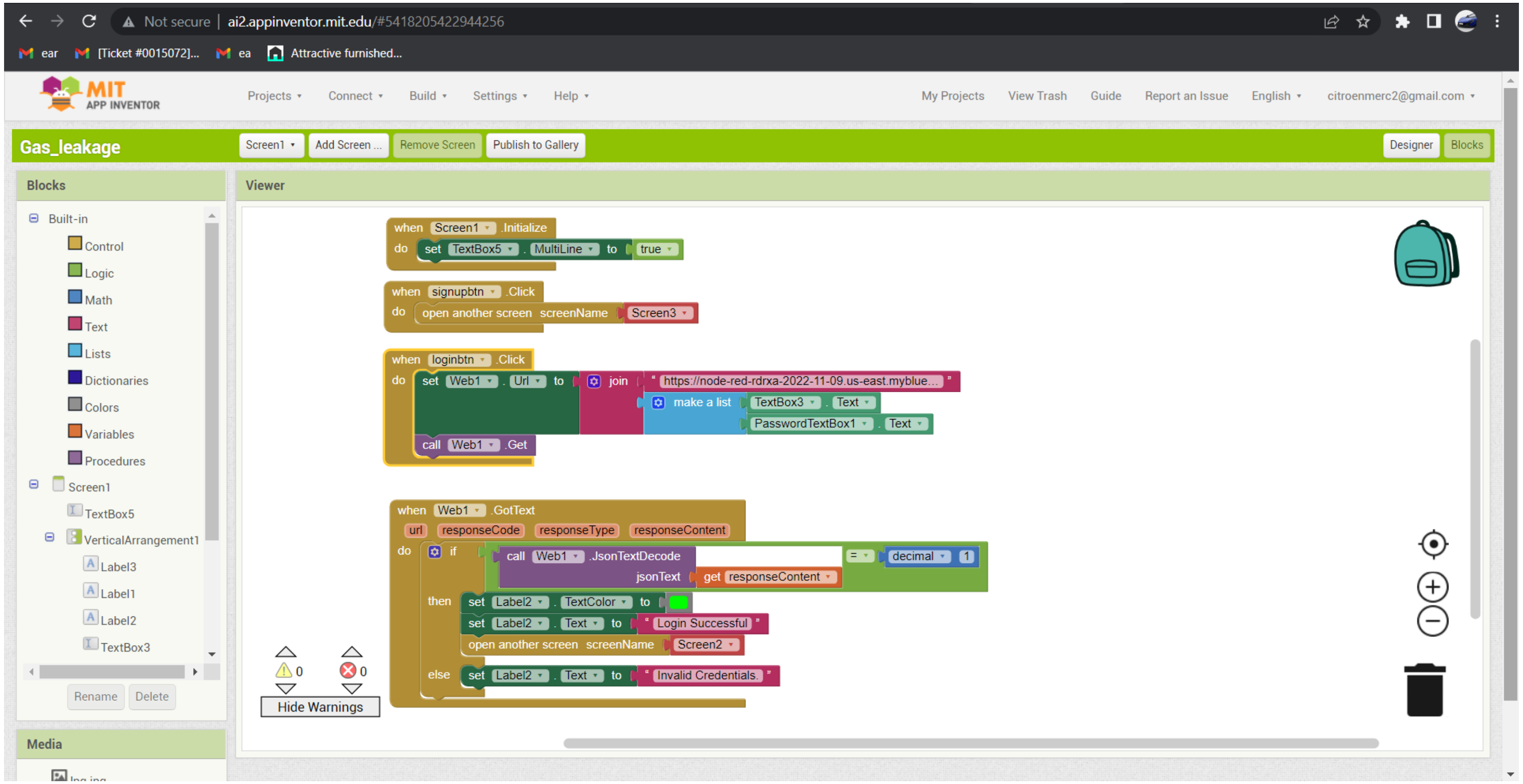
Once registered, the users can enter their newly created credentials to login. For now, they are redirected to a blank screen after successful login. We will be adding further details in this screen in future sprints. The first image shows invalid credentials entered. If correct credentials are entered, then the login is successful.





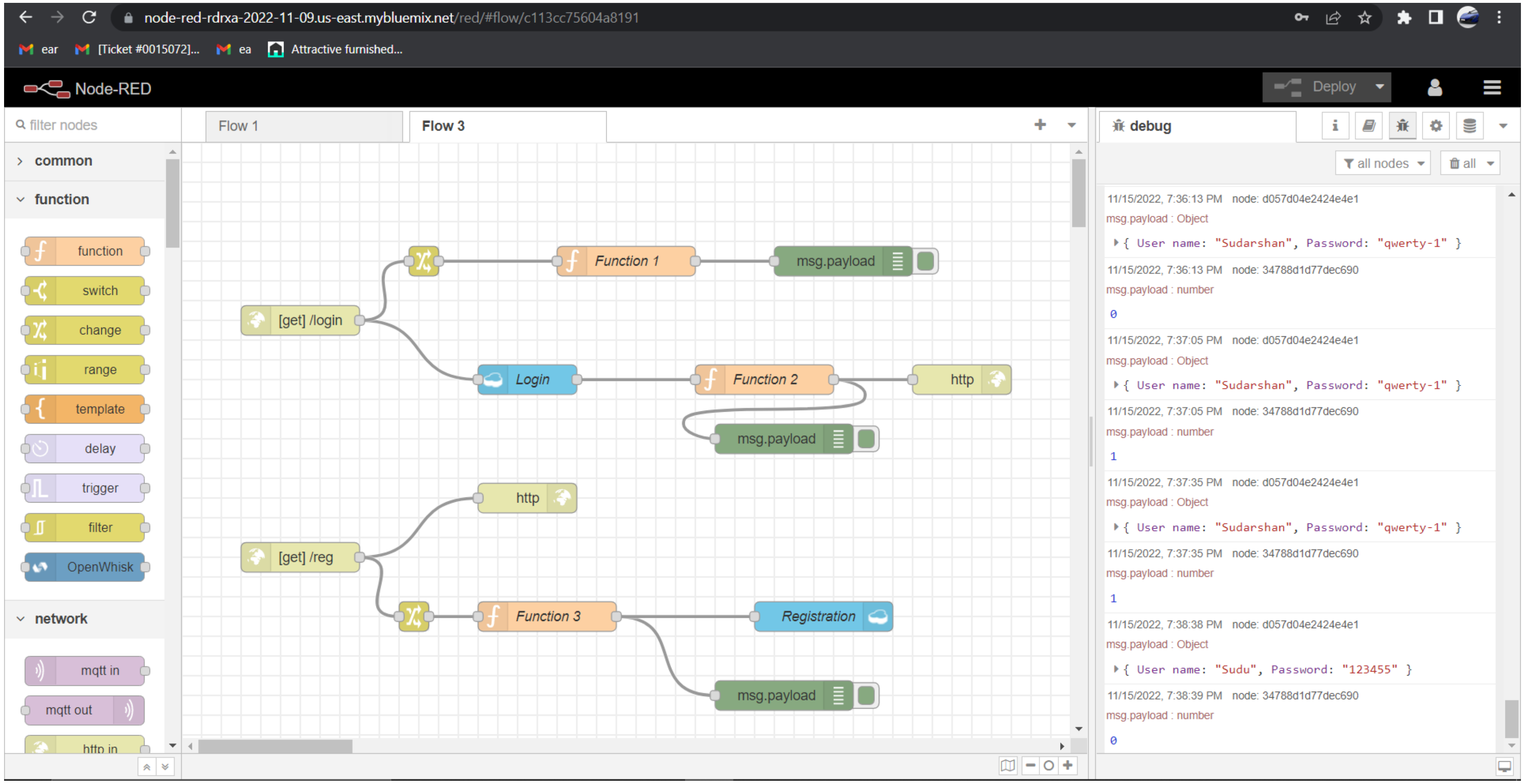
7:38
64%
Coming Soon!

For user login we used the following blocks in MIT App Inventor 2



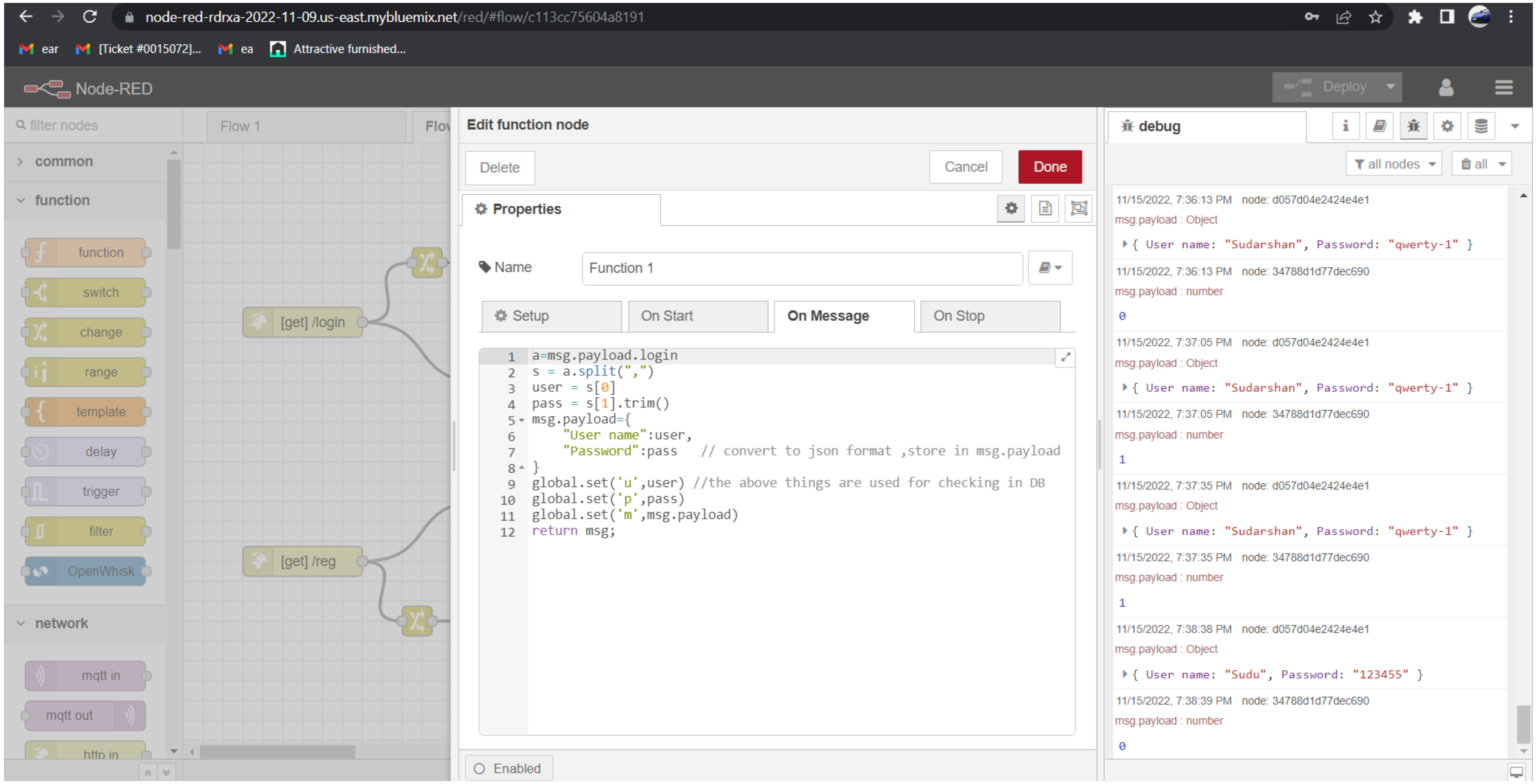
```
when Screen1 .Initialize
do  set TextBox5 . MultiLine to true

when signupbtn .Click
do  open another screen screenName "Screen3"

when loginbtn .Click
do  set Web1 . Url to join " https://node-red-rdrxa-2022-11-09.us-east.myblue... "
                           make a list  TextBox3 . Text
                                        PasswordTextBox1 . Text
    call Web1 .Get

when Web1 .GotText
 url  responseCode  responseType  responseContent
do  if    call Web1 .JsonTextDecode
                    jsonText  get responseContent   =   decimal  1
    then  set Label2 . TextColor to 🟩
          set Label2 . Text to " Login Successful "
          open another screen screenName  Screen2
    else  set Label2 . Text to " Invalid Credentials. "
```

The data entered during login is then sent to Node-RED where it is compared with the database stored in the cloud. If the credentials match, 1 is returned by Node-RED else 0 is returned.



```
11/15/2022, 7:36:13 PM   node: d057d04e2424e4e1
msg.payload : Object
▸{ User name: "Sudarshan", Password: "qwerty-1" }
11/15/2022, 7:36:13 PM   node: 34788d1d77dec690
msg.payload : number
0
11/15/2022, 7:37:05 PM   node: d057d04e2424e4e1
msg.payload : Object
▸{ User name: "Sudarshan", Password: "qwerty-1" }
11/15/2022, 7:37:05 PM   node: 34788d1d77dec690
msg.payload : number
1
11/15/2022, 7:37:35 PM   node: d057d04e2424e4e1
msg.payload : Object
▸{ User name: "Sudarshan", Password: "qwerty-1" }
11/15/2022, 7:37:35 PM   node: 34788d1d77dec690
msg.payload : number
1
11/15/2022, 7:38:38 PM   node: d057d04e2424e4e1
msg.payload : Object
▸{ User name: "Sudu", Password: "123455" }
11/15/2022, 7:38:39 PM   node: 34788d1d77dec690
msg.payload : number
0
```

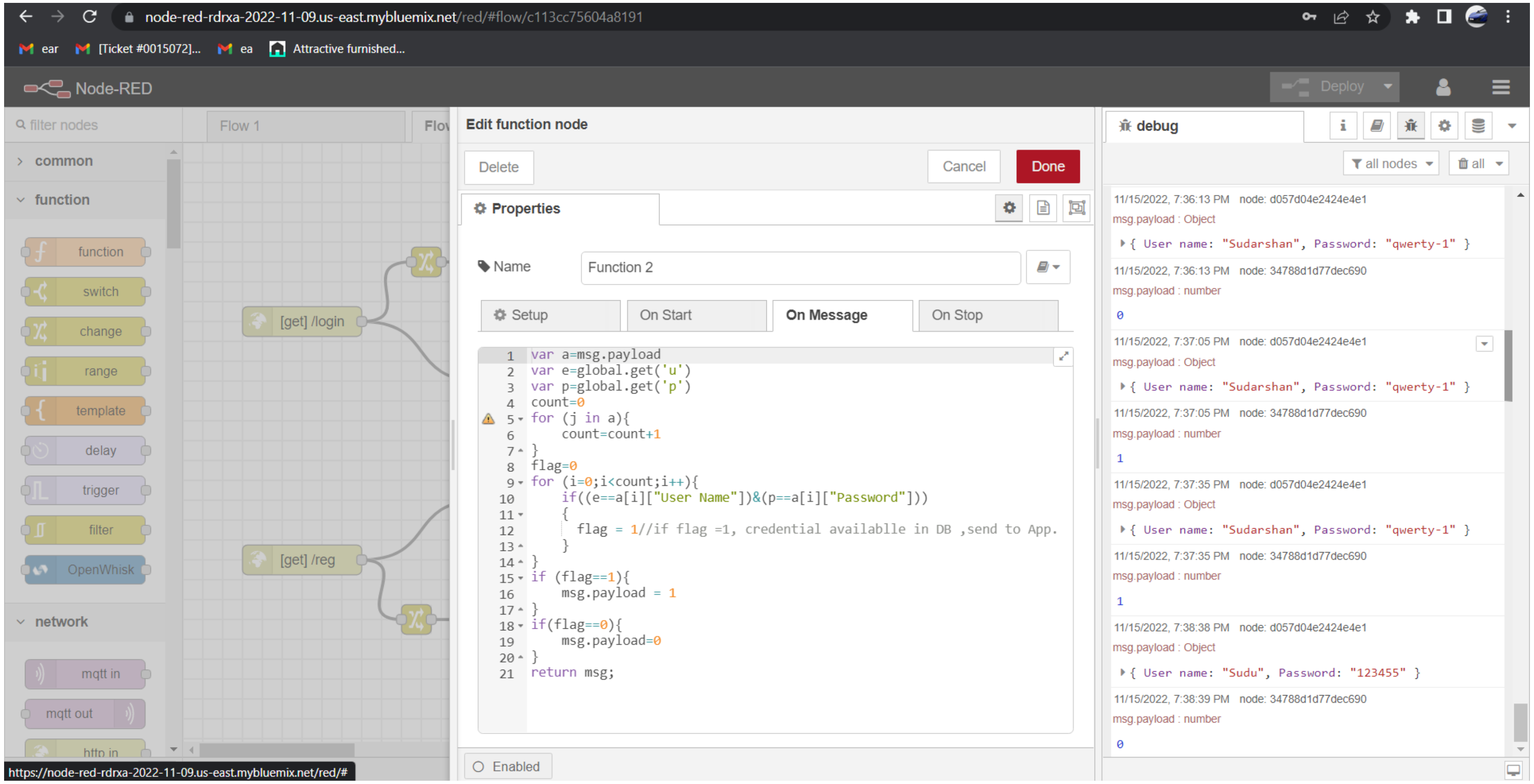
The functions used are shown below. Function 1 declares the received credentials as global so that it can be accessed by Function 2. Function 2 performs the comparison.



Name: Function 1

```
a=msg.payload.login
s = a.split(",")
user = s[0]
pass = s[1].trim()
msg.payload={
     "User name":user,
     "Password":pass   // convert to json format ,store in msg.payload
}
global.set('u',user) //the above things are used for checking in DB
global.set('p',pass)
global.set('m',msg.payload)
return msg;
```

```
11/15/2022, 7:36:13 PM   node: d057d04e2424e4e1
msg.payload : Object
▸{ User name: "Sudarshan", Password: "qwerty-1" }
11/15/2022, 7:36:13 PM   node: 34788d1d77dec690
msg.payload : number
0
11/15/2022, 7:37:05 PM   node: d057d04e2424e4e1
msg.payload : Object
▸{ User name: "Sudarshan", Password: "qwerty-1" }
11/15/2022, 7:37:05 PM   node: 34788d1d77dec690
msg.payload : number
1
11/15/2022, 7:37:35 PM   node: d057d04e2424e4e1
msg.payload : Object
▸{ User name: "Sudarshan", Password: "qwerty-1" }
11/15/2022, 7:37:35 PM   node: 34788d1d77dec690
msg.payload : number
1
11/15/2022, 7:38:38 PM   node: d057d04e2424e4e1
msg.payload : Object
▸{ User name: "Sudu", Password: "123455" }
11/15/2022, 7:38:39 PM   node: 34788d1d77dec690
msg.payload : number
0
```



Name: Function 2

```
var a=msg.payload
var e=global.get('u')
var p=global.get('p')
count=0
for (j in a){
    count=count+1
}
flag=0
for (i=0;i<count;i++){
    if((e==a[i]["User Name"])&(p==a[i]["Password"]))
    {
      flag = 1//if flag =1, credential availablle in DB ,send to App.
    }
}
if (flag==1){
    msg.payload = 1
}
if(flag==0){
    msg.payload=0
}
return msg;
```

```
11/15/2022, 7:36:13 PM   node: d057d04e2424e4e1
msg.payload : Object
▸{ User name: "Sudarshan", Password: "qwerty-1" }
11/15/2022, 7:36:13 PM   node: 34788d1d77dec690
msg.payload : number
0
11/15/2022, 7:37:05 PM   node: d057d04e2424e4e1
msg.payload : Object
▸{ User name: "Sudarshan", Password: "qwerty-1" }
11/15/2022, 7:37:05 PM   node: 34788d1d77dec690
msg.payload : number
1
11/15/2022, 7:37:35 PM   node: d057d04e2424e4e1
msg.payload : Object
▸{ User name: "Sudarshan", Password: "qwerty-1" }
11/15/2022, 7:37:35 PM   node: 34788d1d77dec690
msg.payload : number
1
11/15/2022, 7:38:38 PM   node: d057d04e2424e4e1
msg.payload : Object
▸{ User name: "Sudu", Password: "123455" }
11/15/2022, 7:38:39 PM   node: 34788d1d77dec690
msg.payload : number
0
```