<http://www.qtsrms.com/message>

/\*

//Check if the message is from a registered user

Models.users.findOne({'simNu': msgFrom},function(doc){

knownSimNu = true;

//if user would like to opt-out of RMS membership

if (doc){

if (msgBody === 'RMS out'){

Models.users.findByIdAndRemove(doc.\_id).exec();

client.messages.create({

to: msgFrom,

from: process.env.Twilio\_no,

body: 'Hi '.concat(doc.fname)+'\xa0'.concat(doc.lname)+' your number is removed from RMS contact list'

});

}else{

//send a message to the user to instruct on how to opt out

client.messages.create({

to: msgFrom,

from: process.env.Twilio\_no,

body: 'Hi '.concat(doc.fname)+'\xa0'.concat(doc.lname)+' to opt-out, text: RMS out'

});

}

res.writeHeader(200,{'Content-Type': 'text/xml'});

res.end(twiml.toString());

}else{

//check if this is a register sensor simNu

Models.sensordb.findOne({'simNu': msgFrom},function(results){

knownSimNu = true;

if(results){

console.log('This is a sensor simNU');

res.writeHeader(200,{'Content-Type': 'text/xml'});

res.end(twiml.toString());

}else{

console.log('write to ErrCommNu');

res.writeHeader(200,{'Content-Type': 'text/xml'});

res.end(twiml.toString());

}

});

}

});

if (knownSimNu == false){

console.log('This is not a registered number');

res.writeHeader(200,{'Content-Type': 'text/xml'});

res.end(twiml.toString());

}

});

// }

//});

/\*

twiml.message('');

res.writeHeader(200,{'Content-Type': 'text/xml'});

res.end(twiml.toString());

<Response>

<Message>

</Message>

</Response>

\*/

// check if the message from a registered sensor

//});

/\*

//if the number not in the list then log the details into the Error Communication db

if(err||doc == null){

var item ={

simNu: msgFrom,

msgbody: msgBody};

var data= new Models.ErrComm(item);

data.save();

return

}

res.send(`

<Response></Response>

`);

});

});\*/

/\*

res.send(`

<Response>

<Message>

Hi ${doc.fname} ${doc.lname}, your contact number has been removed from RMS list.

</Message>

</Response>

`);

\*/

/\*

res.send(`

<Response>

<Message>

Hi ${doc.fname} ${doc.lname}. To opt out, please text 'RMS out'.

</Message>

</Response>

`);

\*/

1- clear form after success

$http.post('/users', userData)

.success(function (data) {

// Once complete, clear the form (except location)

$scope.formData.username = "";

$scope.formData.gender = "";

$scope.formData.age = "";

$scope.formData.favlang = "";

})

.error(function (data) {

console.log('Error: ' + data);

});

2- disable button if incomplete form

ng-disabled="addForm.$invalid" . This will prevent a user from clicking the Submit button unless the form is completely valid. In our case, since all fields have the attribute of required, this means that all fields will need to be populated before the button is enabled.

<button type="submit" class="btn btn-danger btn-block" ng-disabled="addForm.$invalid"> Submit</button>

3- <!-- Creates Form (novalidate disables HTML validation, Angular will control) -->

<form name ="addForm" novalidate>

4- listeners can be used with google map

google.maps.event.addListener(marker,'mouseover',function(e){

console.log('mouse over');

});

google.maps.event.addListener(marker,'mouseout',function(e){

console.log('mouse out');

});

5- <!--div ng-include="'app/views/pages/map.html'"></div>

6- onclick="setTimeout(myFunction, 2000);"