**Evaluation – 1**

***Deploying NodeJS application in GCP VM***

*19CSE445 – Cloud Computing*

1. The client and server code must be written in HTML and NodeJS respectively. Client-side validation not required. The NodeJS server must check the Name, Email, Web Address, and Comments fields are not empty.

**HTML:**

<!DOCTYPE html>  
<html>  
<head>  
 <meta name="viewport" content="width=device-width, initial-scale=1">  
 <style>  
 \* {  
 box-sizing: border-box;  
 }  
 h2 {  
 display: block;  
 font-size: 1.5em;  
 margin-top: 0.83em;  
 margin-bottom: 0.83em;  
 margin-left: 0;  
 margin-right: 0;  
 font-weight: bold;  
 font-family: Arial;  
 }  
 input[type=text], select, textarea {  
 width: 100%;  
 padding: 12px;  
 border: 1.5px solid black;  
 resize: vertical;  
 }  
 input[type=email], select, textarea {  
 width: 100%;  
 padding: 12px;  
 border: 1.5px solid black;  
 resize: vertical;  
 }  
 input[type=link], select, textarea {  
 width: 100%;  
 padding: 12px;  
 border: 1px solid black;  
 resize: vertical;  
 }  
 label {  
 padding: 12px 12px 12px 0;  
 display: inline-block;  
 color: blue;  
 font-family: Arial;  
 font-weight: bold;  
 }  
 input[type=submit] {  
 background-color: darkgray;  
 color: black;  
 padding: 12px 20px;  
 border: 1px solid black;  
 border-radius: 4px;  
 cursor: pointer;  
 float: right;  
 font-family: Arial;  
 font-weight: bold;  
 }  
 .container {  
 border-radius: 0px;  
 background-color: white;  
 border: 2px solid black;  
 margin-left: 30%;  
 margin-right: 30%;  
 margin-top: 0%;  
 border-top: white;  
 }  
 .header{  
 border-radius: 0px;  
 background-color: white;  
 border: 2px solid black;  
 margin-left: 30%;  
 margin-right: 30%;  
 margin-top: 0%;  
 }  
 #box1 {  
 padding: 10px;  
 height: 40px;  
 line-height: 30px;  
 background-color: rgb(48,154,252);  
 }  
 .col-25 {  
 float: left;  
 width: 25%;  
 margin-top: 6px;  
 }  
 .col-75 {  
 float: left;  
 width: 75%;  
 margin-top: 6px;  
 }  
 .row {  
 padding: 0px 20px;  
 }  
 #btn.row{  
 padding-top: 10px;  
 padding-right: 200px;  
 padding-bottom: 10px;  
 }  
 .row:after {  
 content: "";  
 display: table;  
 clear: both;  
 }  
 h1 span {  
 background: red;  
 color: #fff;  
 }  
 </style>  
</head>  
<body>  
<div class="header">  
 <div id="box1">  
 <h1></h1>  
 </div>  
</div>  
<div class="container">  
 <form action="http://localhost:8081/" method="GET">  
  
 <div class="row">  
 <div class="col-25">  
 <h2>Contact Us</h2>  
 </div>  
 </div>  
 <div class="row">  
 <div class="col-25">  
 <label for="fname">Name</label>  
 </div>  
 <div class="col-75">  
 <input type="text" id="fname" name="fname" placeholder="">  
 </div>  
 </div>  
 <div class="row">  
 <div class="col-25">  
 <label for="email">Email</label>  
 </div>  
 <div class="col-75">  
 <input type="email" id="email" name="email" placeholder="">  
 </div>  
 </div>  
 <div class="row">  
 <div class="col-25">  
 <label for="link">Web Address</label>  
 </div>  
 <div class="col-75">  
 <input type="link" id="link" name="link" placeholder="">  
 </div>  
 </div>  
 <div class="row">  
 <div class="col-25">  
 <label for="subject">Comments</label>  
 </div>  
 <div class="col-75">  
 <textarea id="subject" name="subject" placeholder="" style="height:100px"></textarea>  
 </div>  
 </div>  
 <div id="btn" class="row">  
 <input type="submit" value="Submit">  
 </div>  
 </form>  
</div>  
</body>  
</html>

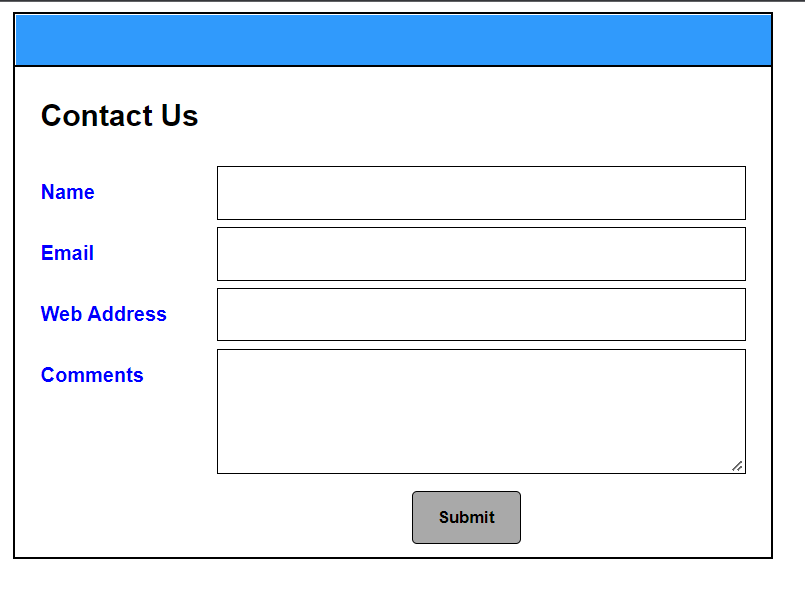
**Client.js (Accessing html file thorugh server):**

var http = require("http");  
var path = require("path");  
var fs = require('fs');  
  
http.createServer(function (req, res) {  
 fs.readFile(\_\_dirname + '/index.html', function (err,data) {  
 if (err) {  
 res.writeHead(404);  
 res.end(***JSON***.stringify(err));  
 return;  
 }  
 res.writeHead(200);  
 res.end(data);  
 });   
}).listen(8080);  
  
// Console will print the message  
***console***.log('Server running at 8080');

**Server.js (Validation of added record):**

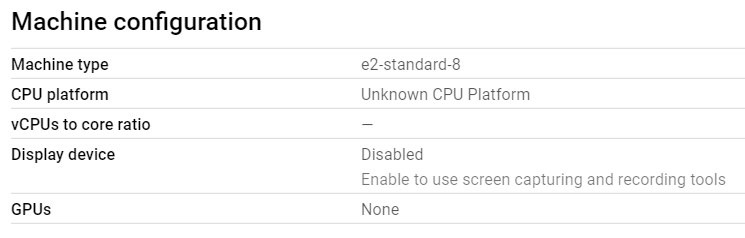
var http = require("http");  
var gow = require("url");  
  
http  
 .createServer(function (request, response) {  
 response.writeHead(200, { "Content-Type": "text/html" });  
 var q = gow.parse(request.url, true).query;  
 console.log("true");  
 let id = q.fname;  
 let mail = q.email;  
 let link = q.link;  
 let comments = q.subject;  
 var numbers = /^[0-9]\*$/;  
 var letters = /^[a-zA-Z]\*$/;  
 var email\_val = /^([a-zA-Z0-9\_\.\-])+\@(([a-zA-Z0-9\-])+\.)+([a-zA-Z0-9]{2,4})+$/;  
 var txt = "Welcome " + q.fname;  
 var dummy = "Errors: ";  
 response.write('<h1 style="color:darkblue" align="center">Hi,' + txt+'</h1>');  
 response.write('<h1 style="color:red"> \t' + dummy+'</h1>');  
  
  
 if (!(id !== "" && id !== undefined && id.match(letters))) {  
 response.write(  
 "<p>1. Name must contain characters only or should not be null</p>"  
 );  
 }  
 if (!(mail !== "" && mail !== undefined && mail.match(email\_val))  
 ) {  
 response.write("<p>2. Invalid Mail Format or should not be null</p>");  
 }  
 if (!(link !== "" && link !== undefined && link.match(letters))) {  
 response.write(  
 "<p>3. Invalid address or should not be null</p>"  
 );  
 }  
 if (!(comments !== "" && comments !== undefined && comments.match(letters))) {  
 response.write(  
 "<p>4. Comment cannot be null</p>"  
 );  
 }  
 response.end();  
 })  
 .listen(8081);  
// Console will print the message  
console.log('Server running at 8081');

**Screenshots:**

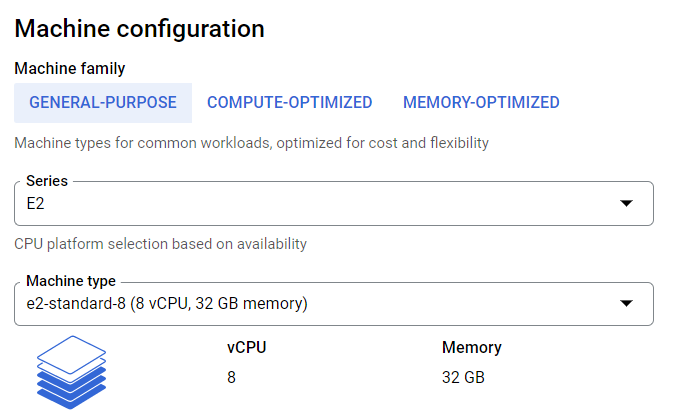
****

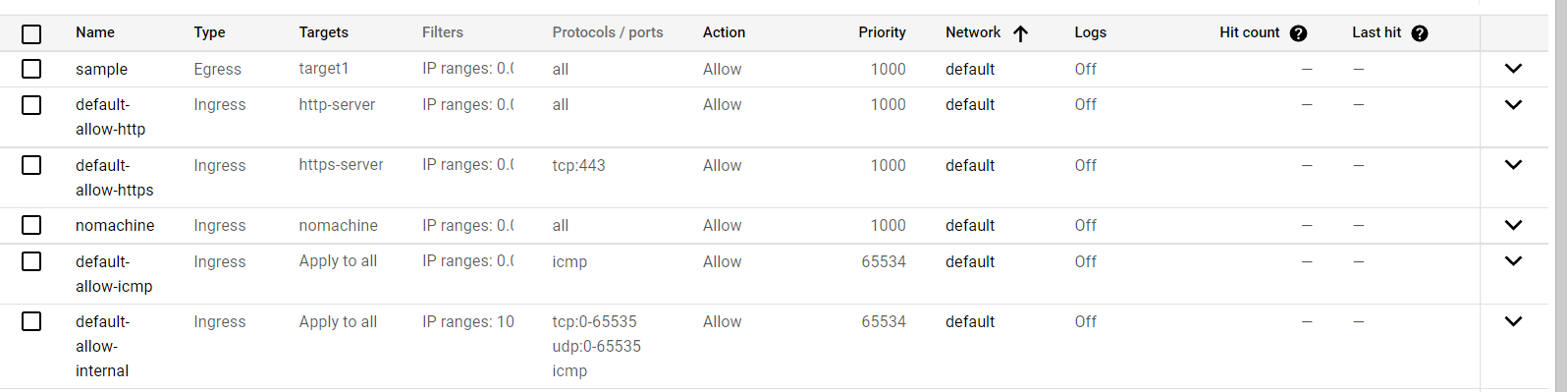
****

1. Deploy the designed client and server application in your favourite cloud platform (Any one of GCP / AZURE) with following requirement.
   * Choose the machine configuration with at-least 8 vCPUs with 32GM of main memory [5 Marks] [CO2]

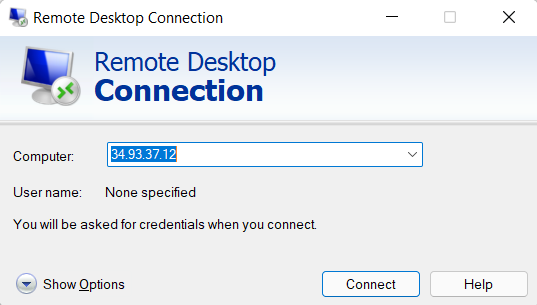


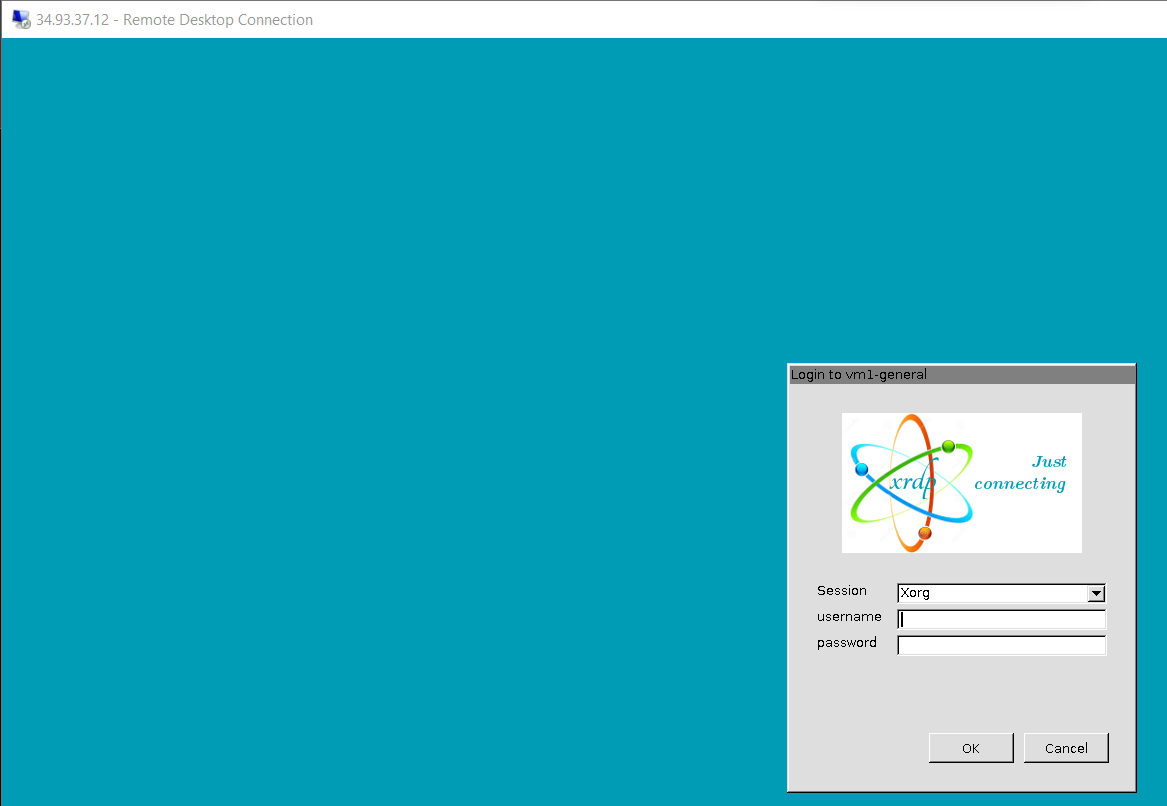
* + Must use Ubuntu 18.04 LTS as operating system [5 Marks] [CO2]

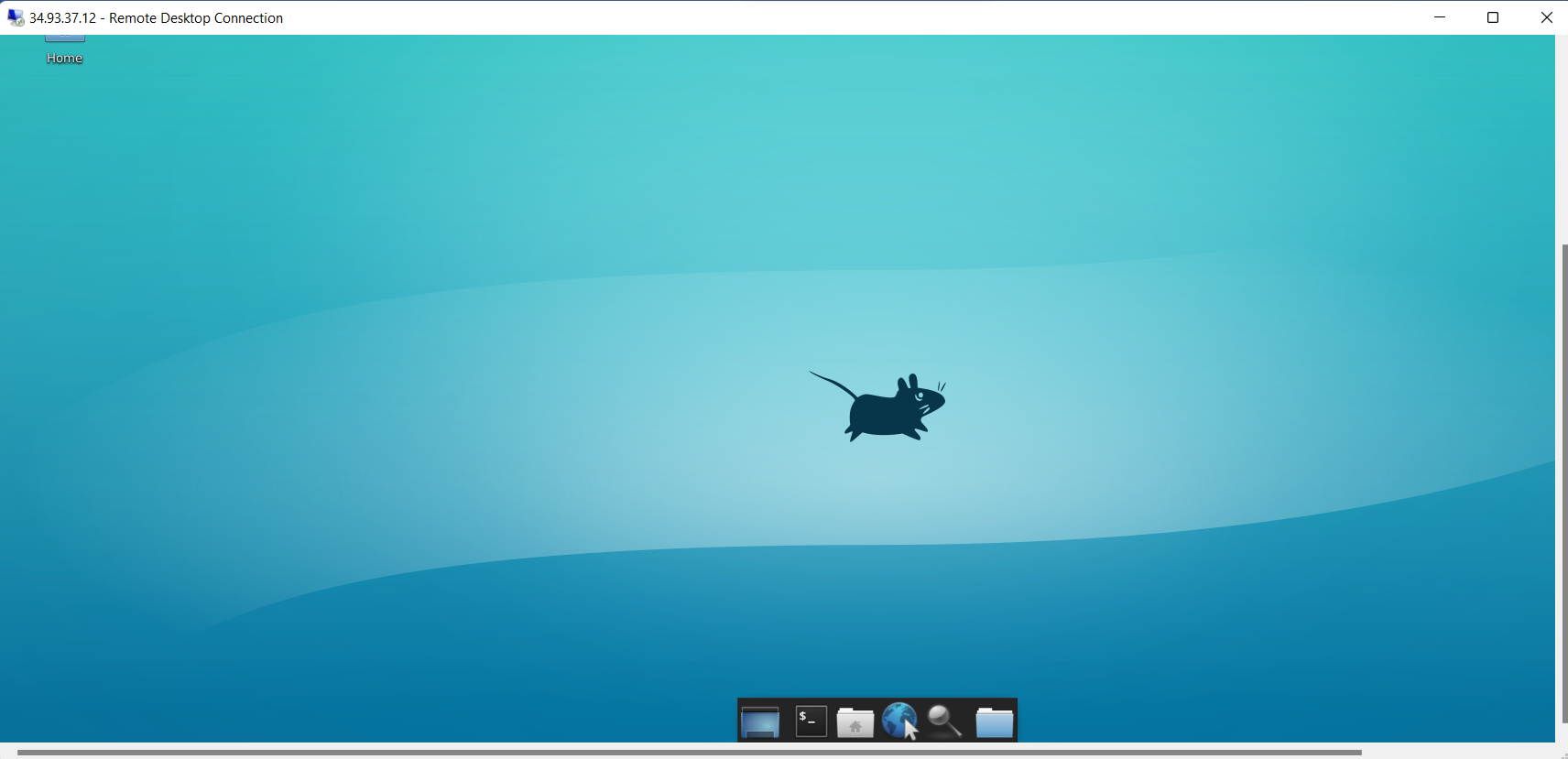


* + Set appropriate firewall configuration to enable public access to node server
  + ****
  + Set up GUI for the Ubuntu 18.04 LTS operating system and use RDP to execute NodeJS program and show demo.

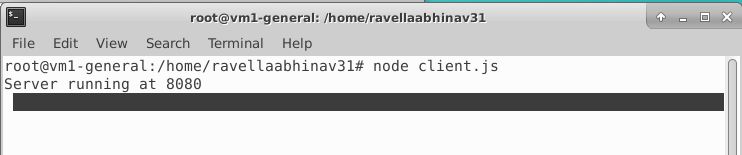
**External IP:** [**http://34.93.37.12/**](http://34.93.37.12/)

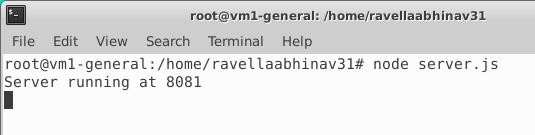
****

****

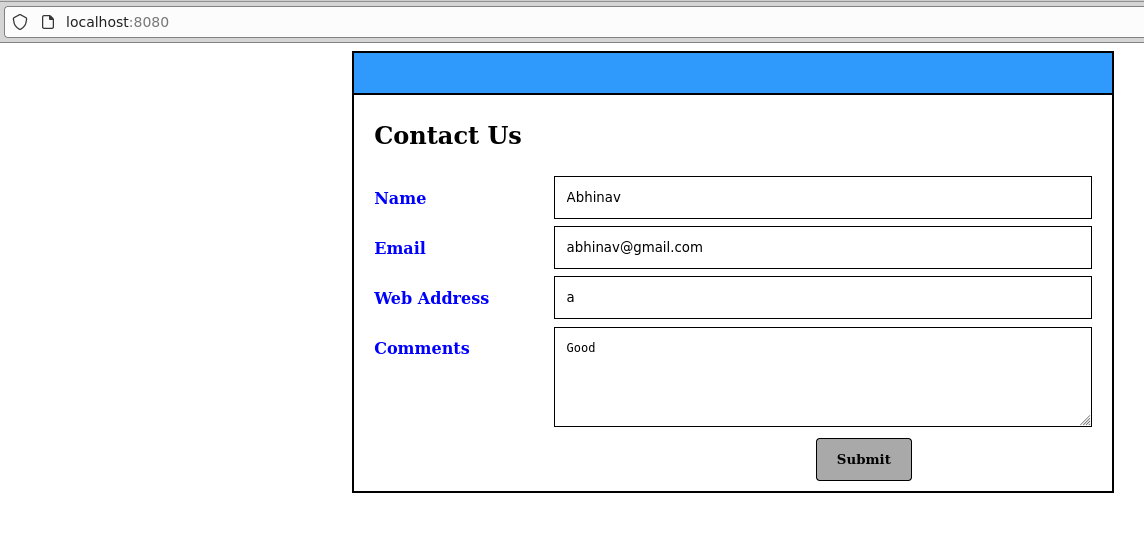
****

**Command Line:**

****

****

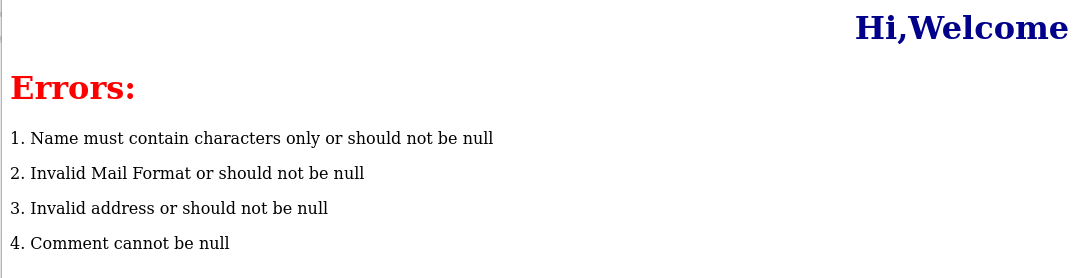
**Output:**

****

**If no errors:**

****

**If errors arises (Case: Form is empty):**

****