Event Management System

# Project Details

Event management system is used to manage all the activity related to event. In any event many service providers work simultaneously and it is very hard to manage these providers. It is also important for event organizer that he has all the contacts details of these service providers so that he can contact them any time to plan an event at given time. To manage all these activities, we have developed this system. This system helps the event management company to manage their paper work online and they can also retrieve report of last event they have completed.

Following are the main module in this system.

* **Client Module** – It is customer who has to take initiative for inputting the values for any event.
* **Event management** – The events needs to be entered for the company to organize.
* **Sponsors management** – Some events cannot do without sponsors so their arrangement and management have to be done.
* **Service Providers management** – These are care taker of the event so work has to be allotted.

# GitHub URL

<https://github.com/manpreetrai1912/ProjectMain> (manpreet kaur)

<https://github.com/kuljeet42/Enterprise_project> (kuljeet kaur chatha)

<https://github.com/56fgfg/project_event_desc> (harinder kaur grewal)

<https://github.com/simranjit95/Enterpriseproject> (simranjit kaur)

# Team Members:

|  |  |
| --- | --- |
| Student ID | Student Name |
| C0742302 | Manpreet kaur |
| C0742139 | Kuljeet kaur chattha |
| C0742778 | Harinder kaur grewal |
| C0743554 | Simranjit kaur |

# NoSQL Database Information

Database Name: event\_management\_system

|  |  |  |
| --- | --- | --- |
| Collection Name | This Collection Details and #of Documents available | Team Member(s) worked on it |
| **client** | This collection contains client detail. | Kuljeet kaur chattha |
| **eventDesc** | This collection contains event description. | Kuljeet kaur chattha |
| **eventService** | This collection contains event services. | Manpreet kaur |
| **eventStaff** | This collection contains information about staff who managed a particular event. | Manpreet kaur |
| **serviceProvider** | This collection contains information about service providing in an event. | Harinder kaur grewal |
| **sponsor** | This collection contains sponsor information. | Harinder kaur grewal |
| **staff** | This collection contains information about staff. | Simranjit kaur |

Database backup file: Attach your database backup file here so that it can be restored and used.

# NodeJS Service Information

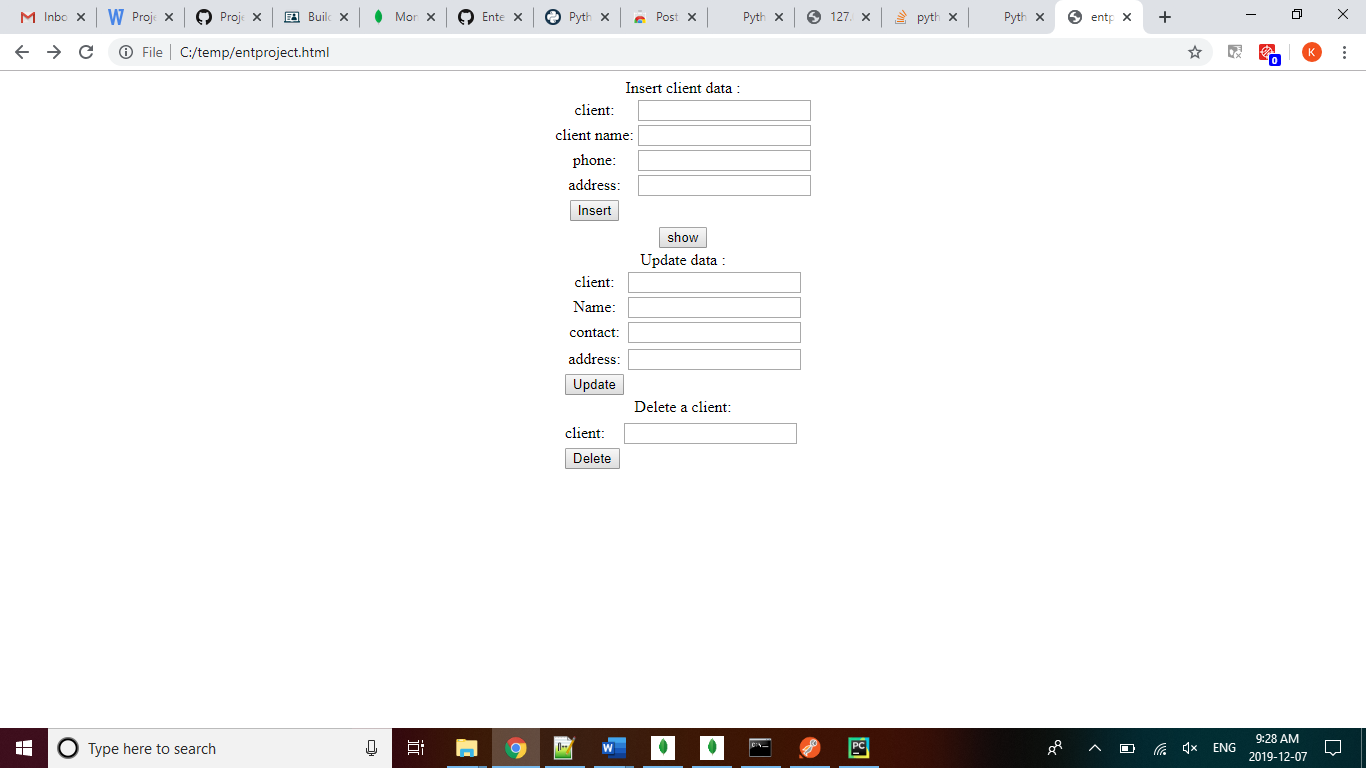
|  |  |  |
| --- | --- | --- |
| Service Path | Details | Team Member(s) worked on it |
| app.get('/api/project/eventService)  app.post('/api/insert1',(req,res)  app.put('/api/updateservice')  app.delete('/api/deleteservice) | Get Service to get the data from event service collection.  Post service to insert the data from event service collection.  Put service to update the data from event service collection.  Delete service to Delete the data from event service collection. | Manpreet Kaur |
| app.get('/api/staff/')  app.post('/API/as/st/')  app.put('/API/as/staf/update',)  app.delete('/API/staff/delete') | Get Service to get the data from staff collection.  Post service to insert the data from staff collection  Put service to update the data from staff collection.  Delete service to Delete the data from staff collection. | Simranjit Kaur |
| app.get('/api/client',)  app.put('/api/updateclient',)  app.post('/api/insert1')  app.delete('/api/delete1',) | Get Service to get the data from client collection.  Put service to update the data from client collection.  Post service to insert the data from client collection  Delete service to Delete the data from client collection. | Kuljeet kaur chattha |
| app.get('/api/project/eventdata)  app.post('/api/pr/newInsert',)  app.put('/api/pro/event/EID')  app.delete('/api/project/dataevent/new') | Get Service to get the data from service provider collection.  Post service to insert the data from service provider collection  Put service to update the data from service provider collection.  Delete service to Delete the data from service provider collection. | Harinder Kaur  Grewal |

Service Codes:



# Application or Website details

Client information



Project.js

﻿const express = require('express');

const app = express();

app.use(express.json());

var MongoClient = require('mongodb').MongoClient;

var url = "mongodb://localhost:27017/";

const bodyParser = require('body-parser');

app.use(express.static('public'));

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({ extended:true}));

app.post('/insert', (req, res)=> {

console.log("Trying to insert new client data..");

console.log("ClientId: " + req.body.client)

res.send("1 document inserted");

MongoClient.connect(url, function(err, db) {

if (err) throw err;

var dbo = db.db("event\_management\_system");

var myobj = {"clientId": req.body.clientId,"clientName" :req.body.clientName,"contactNo" :req.body.contactNo,"address":req.body.address};

dbo.collection("client").insertOne(myobj, function(err, res) {

if (err) throw err;

console.log("1 document inserted");

db.close();

});

});

});

app.get('/show',(req,res)=> {

MongoClient.connect(url, function(err, db) {

if (err) throw err;

var dbo = db.db("event\_management\_system");

dbo.collection("client").find({}).toArray(function(err,result) {

if (err) throw err;

console.log(result);

res.send(result);

db.close();

});

});

});

app.post('/update',(req,res)=> {

MongoClient.connect(url, function(err, db) {

if (err) throw err;

var dbo = db.db("event\_management\_system");

var myquery = { "client": req.body.client };

var newvalues = { $set: {"address":req.body.address } };

dbo.collection("client").updateOne(myquery, newvalues, function(err, res) {

if (err) throw err;

console.log("1 document updated");

db.close();

});

});

});

app.post('/delete',(req, res)=> {

MongoClient.connect(url, function(err, db) {

if (err) throw err;

var dbo = db.db("event\_management\_system");

var myobj = { "client": req.body.client };

dbo.collection("client").deleteOne(myobj, function(err, result) {

if (err) throw err;

res.send("1 document deleted");

console.log("1 document deleted");

db.close();

});

});

});

const port = process.env.PORT || 8081;

app.listen(port, () => console.log('Listening to port ${port}..'));

.

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" type="text/css" href="project.css">

</head>

<body>

<div align="center">

Insert client data :

<form action="http:/localhost:8081/insert" method="post"align="center">

<input type ="hidden" name ="\_method" value="PUT">

<div>

<table align="center">

<tr><td>

<label for="clientId">client:</label></td><td>

<input type="text" name="clientId"></input></td></tr>

</div>

<div><tr><td>

<label for="clientName">client name:</label></td><td>

<input type="text" name="clientName"></input></td></tr>

</div>

<div><tr><td>

<label for="contactNo">phone:</label></td><td>

<input type="text" name="contactNo"></input></td></tr>

</div>

<div><tr><td>

<label for="address">address:</label></td><td>

<input type="text" name="address"></input></td></tr>

</div>

<tr><td>

<div class="button">

<button type="insert">Insert</button></td></tr>

</table>

</div>

</form>

<div>

<form action="http://localhost:8081/show" method="get" align="center">

<input type ="hidden" name ="\_method" value="get">

<table align="center">

<tr><td>

<div class="button">

<button type="show">show</button></td></tr>

</table>

</div>

</form>

<div>

Update data :

<form action="/update" method="post" align="center">

<input type ="hidden" name ="\_method" value="PUT">

<div>

<table align="center">

<tr><td>

<label for="client">client:</label></td><td>

<input type="text" name="client"></input></td></tr>

</div>

<div>

<tr><td>

<label for="name">Name:</label></td><td>

<input type="text" name="name"></input></td></tr>

</div>

<div>

<tr><td>

<label for="contact">contact:</label></td><td>

<input type="text" name="contact"></input></td><tr>

</div>

<div>

<tr><td>

<label for="address">address:</label></td><td>

<input type="text" name="address"></input>

</td></tr>

</div>

<tr><td>

<div class="button">

<button type="update">Update</button></td></tr>

</table>

</div>

</form>

<div>

Delete a client:

<table align="center"><tr><td>

<form action="/delete" method = "post" align="center"></td><td>

<input type ="hidden" name ="\_method" value="DELETE"><td></tr>

<div>

<tr><td>

<label for="client">client:</label></td><td>

<input type="text" name="title"></input><td></tr>

</div>

<div><tr><td>

<button type="delete">Delete</button></td></tr>

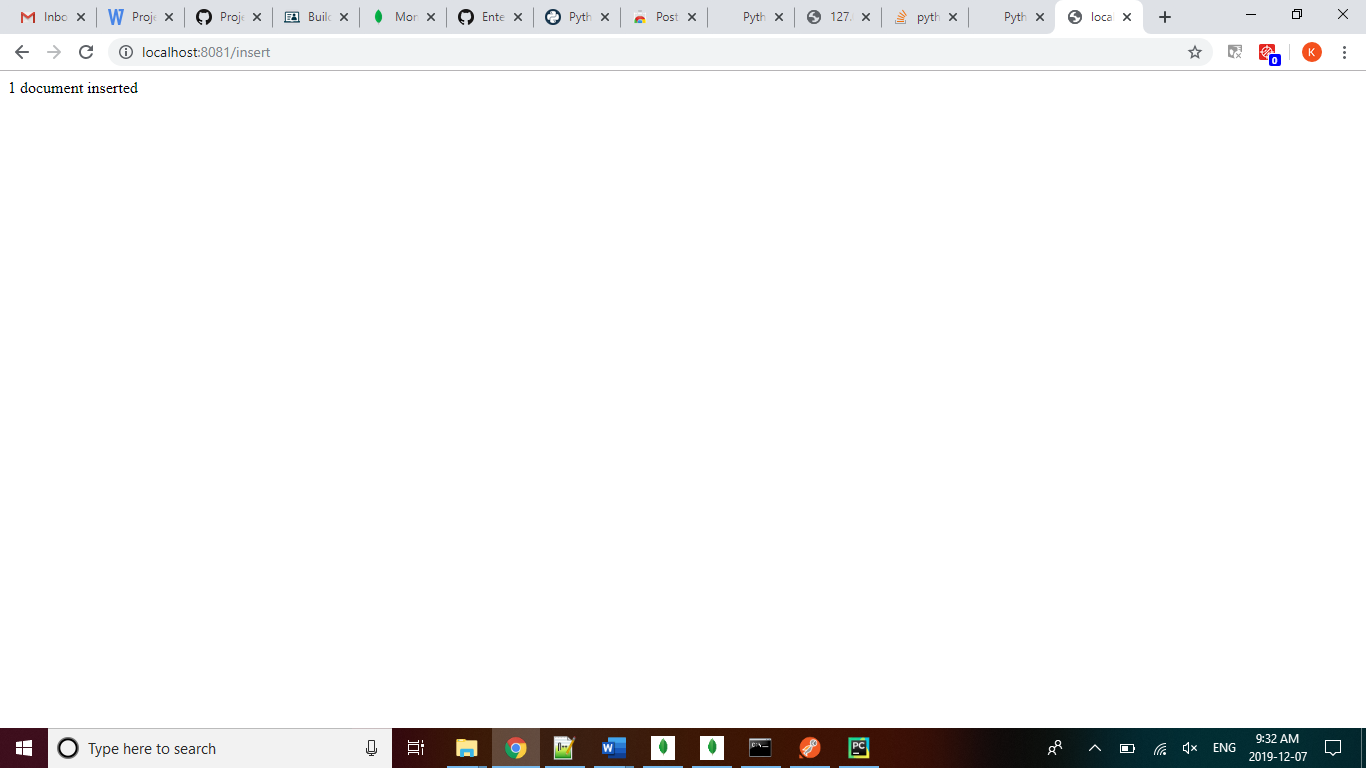
</table>

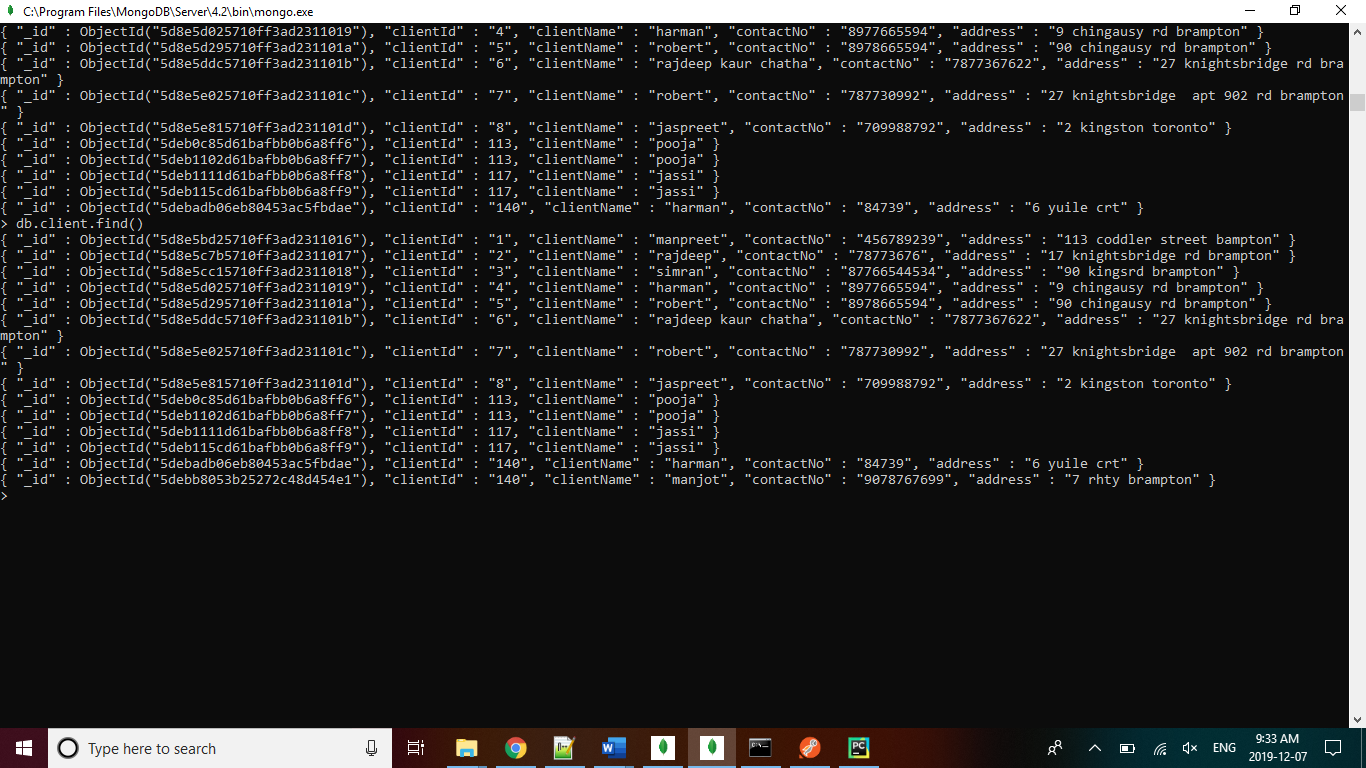
</div>

</form>

</head>

</html>





# How this application will be deployed into Cloud?

For example, we will use AWS CodeDeploy, following steps are required to deploy our application:

1. You will need to create a key pair to access your virtual machine with Amazon EC2 and name your key.
2. Click the home icon on the upper left corner of the AWS Management Console.  Find CodeDeploy under *Developer Tools*and click to open the AWS CodeDeploy Console.
3. In the AWS CodeDeploy Console, click Get Started Now.
4. Select *Sample Deployment* and click Next Step.
5. In this step, we will launch three EC2 instances using a pre-configured EC2 template.
6. Name your application and create deployment group.
7. Create service role and role name.
8. In this step, we will select a deployment configuration and then initiate the deployment to our three EC2 instances. By the end of this step, we'll have successfully deployed a live and running website, which we can visit online.
9. The deployment configuration lets you determine how many instances to simultaneously deploy your application revisions to and describes the success and failure conditions for the deployment. For example, using the default configuration (“One at a Time”), if you deploy your application to 3 instances, this configuration will deploy to one instance at a time.
10. Accept the *Default Deployment Configuration* and click Next Step.
11. Review the details of your deployment and click Deploy Now.
12. Once all three instances are completed, click View All Instances.
13. Click the instance ID for one of the instances you deployed to. This will take you to the EC2 dashboard where you can view the instance that you launched.
14. To verify whether your sample application deployed successfully, copy the address in the Public DNS field in the bottom panel, paste the address into your browser, and you will see your live web page.
15. In the EC2 console, the search bar is autopopulated with a search filter for the Instance ID.  Delete this filter and you will see all the instances launched by CodeDeploy.
16. Select the boxes of each Amazon EC2 instance to terminate. Select Actions,  Instance State, and click Terminate.
17. When prompted, click Yes, Terminate.

# Score Distribution

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Module | Total Marks | Student 1 | Student 2 | Student 3 | Student 4 |
| NoSQL Database | 8 |  |  |  |  |
| NodeJS Service | 8 |  |  |  |  |
| Application | 8 |  |  |  |  |
| Cloud Deployment Details | 8 |  |  |  |  |
| GitHub | 4 |  |  |  |  |
| Participation | 4 |  |  |  |  |
| Total | 40 |  |  |  |  |