



SAN JOSÉ STATE UNIVERSITY

CMPE 273 - 02: ENTERPRISE DISTRIBUTED SYSTEMS

Project Report

Workforce Management System

Submitted To:
Prof. Simon Shim

Date Of Submission
May 6th 2015

Submitted By
Team 4

Team Members

VEDANG JADHAV

NIKHIL CHAVAN

PRANAV DHAPKE

DARSHIT LAKHANI

MADHAV VASUDEV

DOLLY GUPTA

Project Architecture

In this project the Client is developed using AngularJS, HTML/CSS, JavaScript, JQuery and Bootstrap. Node.JS and Express.JS handles the server and the routing.. On the Server side the Node.JS server returns the results from the MySQL database to the RabbitMQ, which is then returned to the client.

Description of components in application DB Schema/strategy

- Client

Admin has access to the client's module, which allows the admin to add new clients into the system, list all the clients already in the system or delete a particular client from the database.

- Guard

In the Client side admin user has access to Guard module as well as Client module. An admin can add a new guard; he can delete an already existing guard or just update a particular guard's information. Admin can also search for guards and assign guards, which are available to the clients building.

- Building

After the client has been added by the admin, he can add buildings to a particular client. He can delete the building of a particular client as well. Admin will have to assign certain checkpoints for each building on which available guards will be assigned.

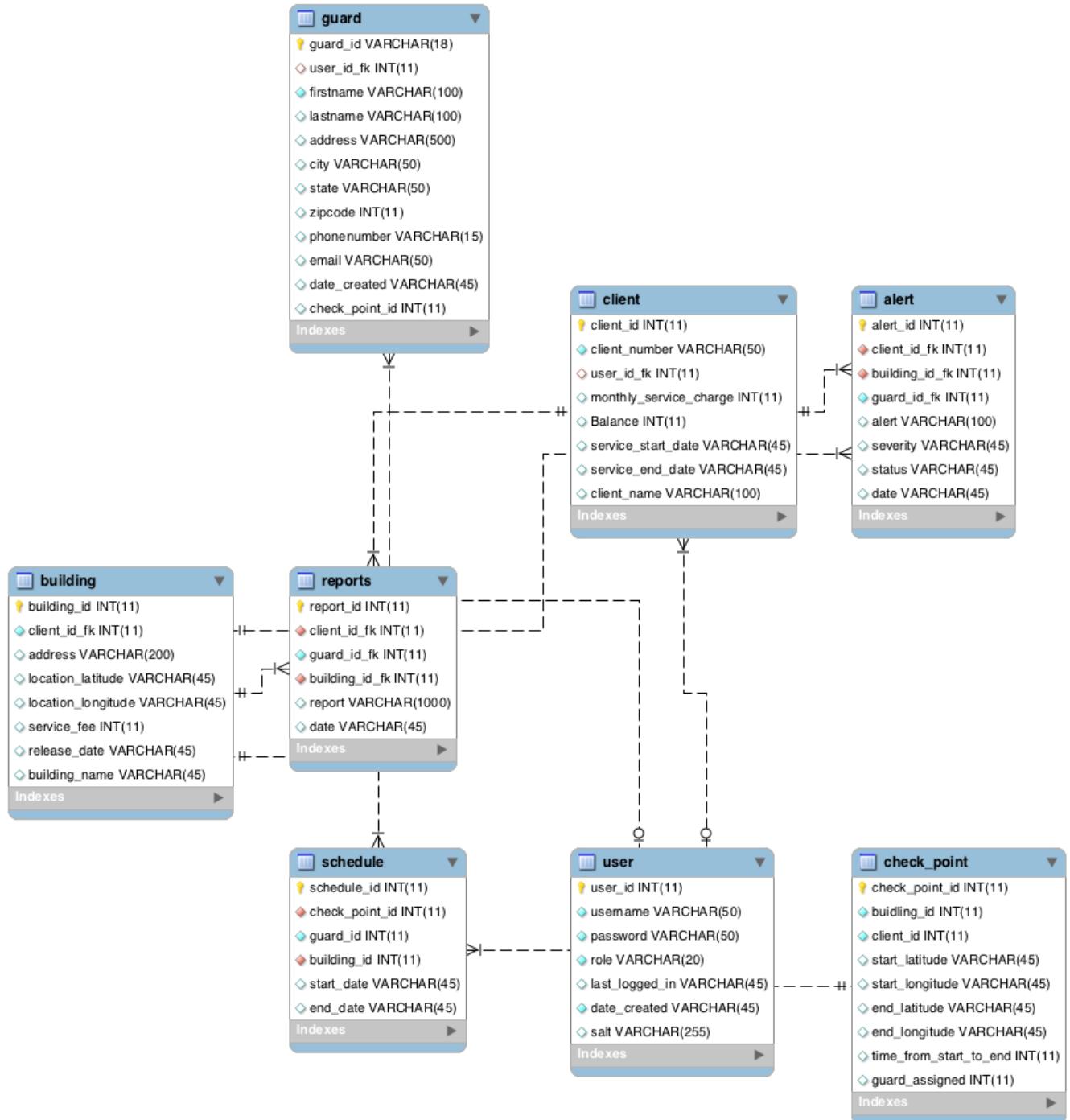
- Report

The guard for a checkpoint and the building he was assigned to used to generate reports. These generated reports will be available to the admin to view, which will be separate per clients and buildings. Admin will be able to search for these reports according to a particular client.

- Alert

Alerts module is responsible for creating new alerts, getting alerts by a particular guard or listing out all the alerts that have been submitted. Clients and admin will have the complete list of alerts that have been generated. The Guard is responsible for generating an alert. A guard can also view the alerts he has submitted.

ENTITY RELATIONSHIP DIAGRAM – DB SCHEMA



DB SCRIPT

```
--  
-- Table structure for table `alert`  
  
DROP TABLE IF EXISTS `alert`;  
/*!40101 SET @saved_cs_client    = @@character_set_client */;  
/*!40101 SET character_set_client = utf8 */;  
  
CREATE TABLE `alert` (  
  `alert_id` int(11) NOT NULL AUTO_INCREMENT,  
  `client_id_fk` int(11) NOT NULL,  
  `building_id_fk` int(11) NOT NULL,  
  `guard_id_fk` varchar(18) NOT NULL,  
  `alert` varchar(100) DEFAULT NULL,  
  `severity` varchar(45) DEFAULT NULL,  
  `status` varchar(45) DEFAULT NULL,  
  `date` varchar(45) DEFAULT NULL,  
  PRIMARY KEY (`alert_id`),  
  KEY `alert_client_fk_idx`(`client_id_fk`),  
  KEY `alert_building_fk_idx`(`building_id_fk`),  
  KEY `alert_guard_fk_idx`(`guard_id_fk`)  
) ENGINE=InnoDB AUTO_INCREMENT=14 DEFAULT CHARSET=latin1;  
/*!40101 SET character_set_client = @saved_cs_client */;  
  
--  
-- Table structure for table `building`  
  
DROP TABLE IF EXISTS `building`;  
/*!40101 SET @saved_cs_client    = @@character_set_client */;  
/*!40101 SET character_set_client = utf8 */;  
  
CREATE TABLE `building` (
```

```

`building_id` int(11) NOT NULL AUTO_INCREMENT,
`client_id_fk` int(11) NOT NULL,
`address` varchar(200) DEFAULT NULL,
`location_latitude` varchar(45) DEFAULT NULL,
`location_longitude` varchar(45) DEFAULT NULL,
`service_fee` int(11) DEFAULT NULL,
`release_date` varchar(45) DEFAULT NULL COMMENT 'release date is the date till when the building is to
be guarded according to the contract, specified by Mahesh in his slides.',
`building_name` varchar(45) DEFAULT NULL,
PRIMARY KEY (`building_id`),
KEY `client_id_foreign_key_idx`(`client_id_fk`)
) ENGINE=InnoDB AUTO_INCREMENT=29 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
```

-- Table structure for table `check_point`

--

```

DROP TABLE IF EXISTS `check_point`;
/*!40101 SET @saved_cs_client    = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;

CREATE TABLE `check_point`(
`check_point_id` int(11) NOT NULL AUTO_INCREMENT,
`buidling_id` int(11) NOT NULL,
`client_id` int(11) NOT NULL,
`start_latitude` varchar(45) DEFAULT NULL,
`start_longitude` varchar(45) DEFAULT NULL,
`end_latitude` varchar(45) DEFAULT NULL,
`end_longitude` varchar(45) DEFAULT NULL,
`time_from_start_to_end` int(11) DEFAULT NULL,
`guard_assigned` int(11) DEFAULT NULL,
PRIMARY KEY (`check_point_id`),
```

```

KEY `cp_building_fk_idx`(`buidling_id`),
KEY `cp_client_fk_idx`(`client_id`)
) ENGINE=InnoDB AUTO_INCREMENT=16 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

-- 
-- Table structure for table `client`
-- 

DROP TABLE IF EXISTS `client`;
/*!40101 SET @saved_cs_client  = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;

CREATE TABLE `client` (
`client_id` int(11) NOT NULL AUTO_INCREMENT,
`client_number` varchar(50) NOT NULL COMMENT 'client_number is the identifier with SSN Format
specified by Mahesh and client_id is the auto-increment Id',
`user_id_fk` int(11) DEFAULT NULL,
`monthly_service_charge` int(11) DEFAULT NULL,
`Balance` int(11) DEFAULT NULL,
`service_start_date` varchar(45) DEFAULT NULL,
`service_end_date` varchar(45) DEFAULT NULL,
`client_name` varchar(100) DEFAULT NULL,
PRIMARY KEY (`client_id`,`client_number`),
KEY `user_id_fk_idx`(`user_id_fk`),
CONSTRAINT `user_id_client_fk` FOREIGN KEY(`user_id_fk`) REFERENCES `user`(`user_id`) ON DELETE
NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB AUTO_INCREMENT=21 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

-- 
-- Table structure for table `guard`
-- 

```

```

DROP TABLE IF EXISTS `guard`;

/*!40101 SET @saved_cs_client    = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;

CREATE TABLE `guard` (
  `guard_id` varchar(18) NOT NULL,
  `user_id_fk` int(11) DEFAULT NULL,
  `firstname` varchar(100) NOT NULL,
  `lastname` varchar(100) DEFAULT NULL,
  `address` varchar(500) DEFAULT NULL,
  `city` varchar(50) DEFAULT NULL,
  `state` varchar(50) DEFAULT NULL,
  `zipcode` int(11) DEFAULT NULL,
  `phonenumber` varchar(15) DEFAULT NULL,
  `email` varchar(50) DEFAULT NULL,
  `date_created` varchar(45) DEFAULT NULL,
  `check_point_id` int(11) DEFAULT NULL,
  `checkpoint_assigned` int(11) DEFAULT NULL,
  PRIMARY KEY (`guard_id`),
  KEY `user_id_fk_idx`(`user_id_fk`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/*!40101 SET character_set_client = @saved_cs_client */;

-- 
-- Table structure for table `reports`
-- 


```

```

DROP TABLE IF EXISTS `reports`;

/*!40101 SET @saved_cs_client    = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;

CREATE TABLE `reports` (
  `report_id` int(11) NOT NULL AUTO_INCREMENT,

```

```

`client_id_fk` int(11) NOT NULL,
`guard_id_fk` int(11) NOT NULL,
`building_id_fk` int(11) NOT NULL,
`report` varchar(1000) DEFAULT NULL,
`date` varchar(45) DEFAULT NULL,
PRIMARY KEY (`report_id`),
KEY `building_id_fk_idx`(`building_id_fk`),
KEY `guard_id_fk_idx`(`guard_id_fk`),
KEY `client_id_fk`(`client_id_fk`),
CONSTRAINT `building_id_fk` FOREIGN KEY (`building_id_fk`) REFERENCES `building`(`building_id`) ON
DELETE NO ACTION ON UPDATE NO ACTION,
CONSTRAINT `client_id_fk` FOREIGN KEY (`client_id_fk`) REFERENCES `client`(`client_id`) ON DELETE NO
ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
```

```

-- Table structure for table `schedule`
--
```

```

DROP TABLE IF EXISTS `schedule`;
/*!40101 SET @saved_cs_client    = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
```

```

CREATE TABLE `schedule` (
`schedule_id` int(11) NOT NULL AUTO_INCREMENT,
`check_point_id` int(11) NOT NULL,
`guard_id` int(11) NOT NULL,
`building_id` int(11) NOT NULL,
`start_date` varchar(45) DEFAULT NULL,
`end_date` varchar(45) DEFAULT NULL,
PRIMARY KEY (`schedule_id`),
KEY `schedule_cp_id_idx`(`check_point_id`),
```

```
KEY `schedule_guard_id_idx`(`guard_id`),
KEY `schedule_building_id_idx`(`building_id`),
CONSTRAINT `schedule_building_id` FOREIGN KEY (`building_id`) REFERENCES `building`(`building_id`)
ON DELETE NO ACTION ON UPDATE NO ACTION,
CONSTRAINT `schedule_cp_id` FOREIGN KEY (`check_point_id`) REFERENCES `check_point`(`check_point_id`)
ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;
```

```
--
```

```
-- Table structure for table `user`
```

```
--
```

```
DROP TABLE IF EXISTS `user`;
```

```
/*!40101 SET @saved_cs_client    = @@character_set_client */;
```

```
/*!40101 SET character_set_client = utf8 */;
```

```
CREATE TABLE `user` (
```

```
 `user_id` int(11) NOT NULL AUTO_INCREMENT,
```

```
 `username` varchar(50) NOT NULL,
```

```
 `password` varchar(500) NOT NULL,
```

```
 `role` varchar(20) NOT NULL,
```

```
 `last_logged_in` varchar(45) DEFAULT NULL,
```

```
 `date_created` varchar(45) DEFAULT NULL,
```

```
 `salt` varchar(255) DEFAULT NULL,
```

```
 PRIMARY KEY (`user_id`)
```

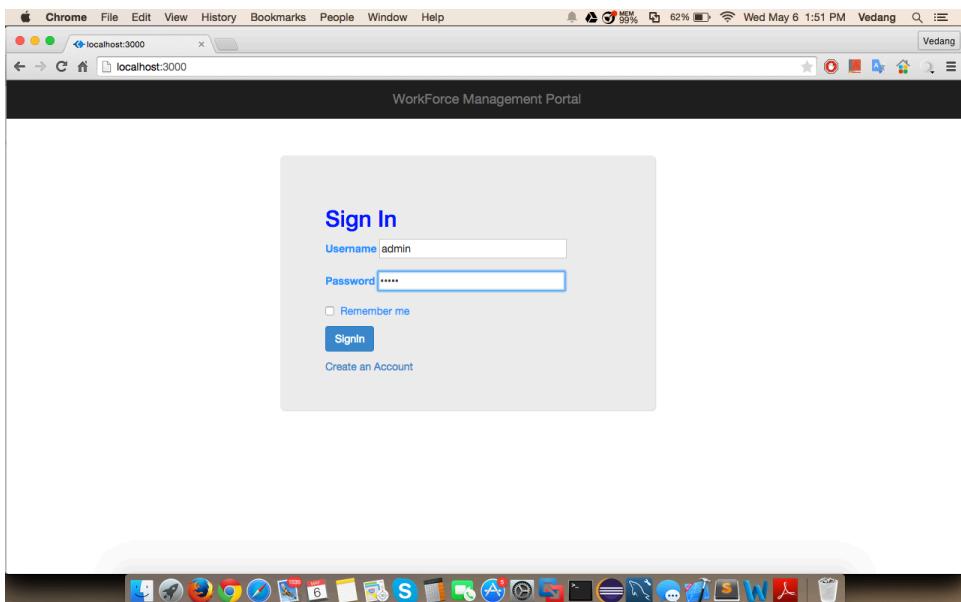
```
) ENGINE=InnoDB AUTO_INCREMENT=14 DEFAULT CHARSET=latin1;
```

```
/*!40101 SET character_set_client = @saved_cs_client */;
```

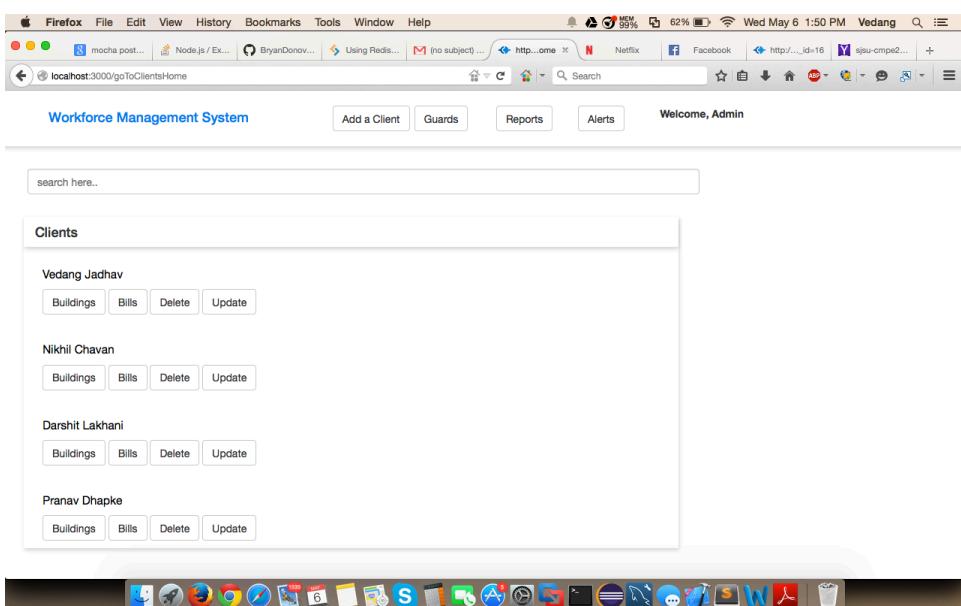
```
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;
```

SCREEN CAPTURES OF THE CLIENT APPLICATION

Admin Login



Admin Homepage



Add Guard

Guard Number (SSN: xxx-xx-xxxx)
656-77-9098
Email
ramesh.ram@gmail.com
First Name
Ramesh
Last Name
Ram
Address
1314, Alamaden
City
San Jose
State
California
ZipCode
97878
Phone Number (xxx-xxx-xxxx)
408-111-1222
Add Guard

View Clients

Workforce Management System

search here..

Clients

Vedang Jadhav

Nikhil Chavan

Darshit Lakhani

Pranav Dhapke

Buildings Bills Delete Update

Buildings Bills Delete Update

Buildings Bills Delete Update

Buildings Bills Delete Update

Add Guard

WorkForce Management Portal

Sign Up

Username: rmmm.jmmm

Password: *****

Add Guard

View Building

Workforce Management System

Welcome, Admin

Client Name **Dolly Gupta** Buildings

St. James
Service Fee \$ 160
Release Date 12/12/2015

Notre Dame Courthouse
Service Fee \$ 900
Release Date 12/1/2015

Add Checkpoint | View Details | Edit Building | Delete Building

Add Checkpoint | View Details | Edit Building | Delete Building

Add Building

The screenshot shows a web application titled "Workforce Management System". At the top, it says "Welcome, Admin". Below the title, there is a map interface with a red pin indicating a location. The map includes zoom controls and a "Map/Satellite" switch. To the right of the map, there are four input fields: "Client Name" (Dolly Gupta), "Building Name" (Notre James De), "Service Fee" (1500), and "Release Date" (3/1/2016). At the bottom left of the map area, there are links for "Map data ©2015 Google", "Terms of Use", and "Report a map error". Above the map, there are two buttons: "Submit and Add More" and "Submit". The browser's address bar shows the URL "localhost:3000/redirectToAddBuildings?client_id=20&client_name=Dolly%20Guptaa". The status bar at the bottom of the browser window shows the date and time as "Wed May 6 2:22 PM Vedang".

View Reports

The screenshot shows a web application titled "Workforce Management System". At the top, there are tabs for "Clients", "Guards", "Reports", and "Alerts", with "Reports" being the active tab. It also says "Welcome, Admin". Below the tabs, there are search fields for "Client ID" (Nikhil Chavan), "Building ID" (San Jose), "Date" (From [] to []), and "Search" (with a text input field). There is also an "OK" button. A message box at the top displays "Client :Nikhil Chavan, Building :San Jose". Below this, there is a table with three columns: "Building Name" (San Jose), "Date" (--), and "Guard ID" (2). Further down, there is a section titled "Patrols" with the following details: "Date: 12/12/2015", "Alert: Fire Alarm", "Severity: SEVERE", and "Status: Active". The browser's address bar shows the URL "localhost:3000/report". The status bar at the bottom of the browser window shows the date and time as "Wed May 6 2:23 PM Vedang".

Guard Homepage

The screenshot shows a Firefox browser window with the URL <http://localhost:3000/goToGuardsHome>. The title bar says "Workforce Management System" and "Welcome, Guard". The main content area has two sections: "Alerts Submitted:" containing "Parking violation" and "Alert" with dropdown menus for "Alert Type", "Severity", and "Status", plus a "Submit Alert" button.

View alerts by Client

The screenshot shows a Firefox browser window with the URL <http://localhost:3000/alert>. The title bar says "Workforce Management System" and "Welcome, Admin". The top navigation bar includes "Clients", "Guards", "Reports", and "Alerts". The main content area shows "Alert Criteria" (Client dropdown set to "Dolly Guptaa") and "Select Client" (dropdown set to "Dolly Guptaa"). Below this, a table displays a single alert entry:

Client	Building	Alert	Day
Dolly Guptaa	St. James	Parking violation	2015-05-06 14:36:22

View Alerts by Building

The screenshot shows a Firefox browser window with the URL <http://localhost:3000/alert>. The title bar says "View Alerts by Building". The page header includes "Workforce Management System" and navigation tabs for "Clients", "Guards", "Reports", "Alerts", and "Welcome, Admin".
Alert Criteria:
Building: Darshit Lakhani
Select Client:
Darshit Lakhani
Select building:
Avalon
Go

Client	Building	Alert	Day
Darshit Lakhani	Avalon	Fire Alarm - Given now under Control	1/11/2015

Search Guard

The screenshot shows a Firefox browser window with the URL <http://localhost:3000/redirectToGuards>. The title bar says "Search Guard". The page header includes "Workforce Management System" and navigation tabs for "Add a Guard", "Clients", "Assign Guard", "Reports", "Update Guards", "Alerts", and "Welcome, Admin".
Search Guard:
sdasd asdads : (Guard Id : 111-11-1111)

Vedang JD : (Guard Id : 444-44-4444)

Johny Johnson : (Guard Id : 444-44-4445)

Vedang JD : (Guard Id : 788-00-1915)

Hund Mond : (Guard Id : 897-12-8989)

Rajesh Ram : (Guard Id : 989-99-8884)

Search Guard

A screenshot of a Firefox browser window. The address bar shows 'localhost:3000/redirectToGuards'. The main content area displays a search result for 'jo'. The result is 'Johny Johnson : (Guard Id : 444-44-4445)'. Below the result are two buttons: 'Delete' and 'Location'. The browser's top bar includes the title 'Workforce Management System' and navigation buttons like 'Add a Guard', 'Clients', 'Assign Guard', 'Reports', 'Update Guards', 'Alerts', and 'Welcome, Admin'. The status bar at the bottom shows system information like battery level (94%), signal strength, and date/time (Wed May 6 2:39 PM). A Mac OS X dock is visible at the bottom of the screen.

Search Guard

A screenshot of a Firefox browser window, identical to the one above, showing search results for 'pra'. The results section is empty, displaying the message 'No Guards Found..'. The browser interface, including the top menu, search bar, and docked applications, is consistent with the first screenshot.

Add Client

Workforce Management System Welcome, Admin

Add a Client

Client Number
909-99-5758

Client Name
John

Monthly Service Charge
2000

Balance
0

Service Start Date
11/21/2014

Service End Date
11/20/2015

Add Client

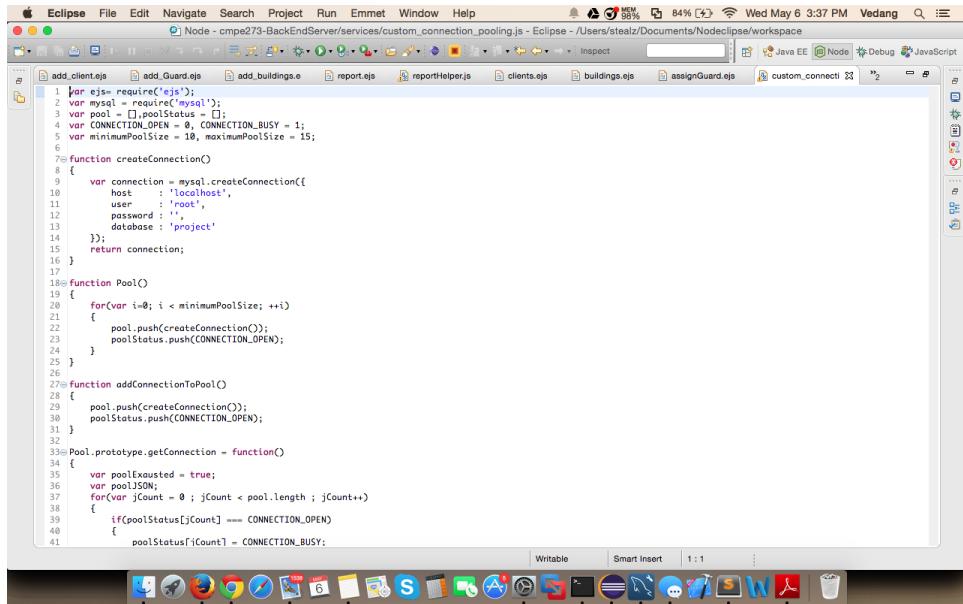


CODE LISTING

MySQL

```
1  for (ejjs = require('ejjs');  
2      var mysql = require('mysql');  
3      function getConnection()  
4      {  
5          var connection = mysql.createConnection({  
6              host : 'localhost',  
7              user : 'root',  
8              password : '',  
9              database : 'project'  
10         });  
11         return connection;  
12     }  
13     function fetchData(callback,sqlQuery){  
14         console.log("\nSQL Query::"+sqlQuery);  
15         var connection=getConnection();  
16         connection.query(sqlQuery, function(err, rows, fields) {  
17             if(err){  
18                 console.log("ERROR: " + err.message);  
19             }  
20             else{  
21                 console.log("DB Results:"+rows);  
22                 callback(err, rows);  
23             }  
24         });  
25         console.log("\nConnection closed..");  
26         connection.end();  
27     }  
28     function insertData(callback,sqlQuery){  
29         console.log("\nSQL Query::"+sqlQuery);  
30         var connection=getConnection();  
31         connection.query(sqlQuery, function(err, results)  
32         {  
33             if(err){  
34                 console.log("ERROR: " + err.message);  
35             }  
36             else{  
37                 console.log("DB Results:"+results);  
38             }  
39             callback(err, results);  
40         })  
41     }  
42 }
```

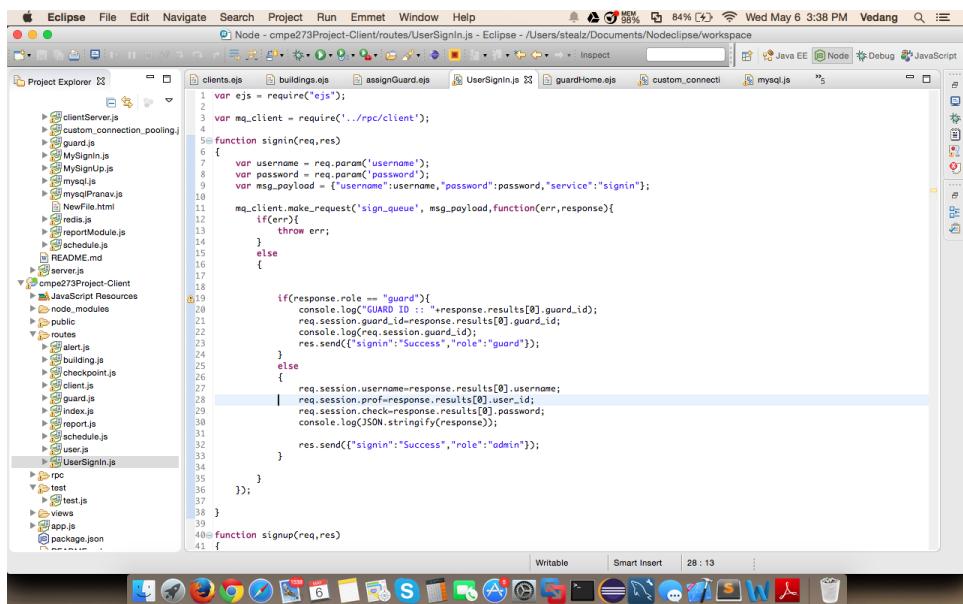
Connection Pooling:



A screenshot of the Eclipse IDE interface on a Mac OS X desktop. The title bar shows "Node - cmpe273-BackEndServer/services/custom_connection_pooling.js - Eclipse - /Users/stealz/Documents/Nodeclipse/workspace". The code editor displays a JavaScript file named "custom_connection_pooling.js". The code implements a connection pool for MySQL, defining a "Pool" object with methods to create connections and add them to a pool. It also includes a "getConnection" prototype method and logic to manage a pool of connections. The Eclipse interface includes toolbars, a project explorer, and various tabs for Java EE, Node.js, and JavaScript.

```
1 var ejjs = require('ejjs');
2 var mysql = require('mysql');
3 var pool = [];
4 var poolStatus = [];
5 var CONNECTION_OPEN = 0, CONNECTION_BUSY = 1;
6 var minimumPoolSize = 10, maximumPoolSize = 15;
7
8 function createConnection()
9 {
10     var connection = mysql.createConnection({
11         host : 'localhost',
12         user : 'root',
13         password : '',
14         database : 'project'
15     });
16     return connection;
17 }
18
19 function Pool()
20 {
21     for(var i=0; i < minimumPoolSize; ++i)
22     {
23         pool.push(createConnection());
24         poolStatus.push(CONNECTION_OPEN);
25     }
26 }
27
28 function addConnectionToPool()
29 {
30     pool.push(createConnection());
31     poolStatus.push(CONNECTION_OPEN);
32 }
33
34 Pool.prototype.getConnection = function()
35 {
36     var poolIsUsed = true;
37     var poolJSON;
38     for(var iCount = 0 ; iCount < pool.length ; iCount++)
39     {
40         if(poolStatus[iCount] === CONNECTION_OPEN)
41             poolStatus[iCount] = CONNECTION_BUSY;
```

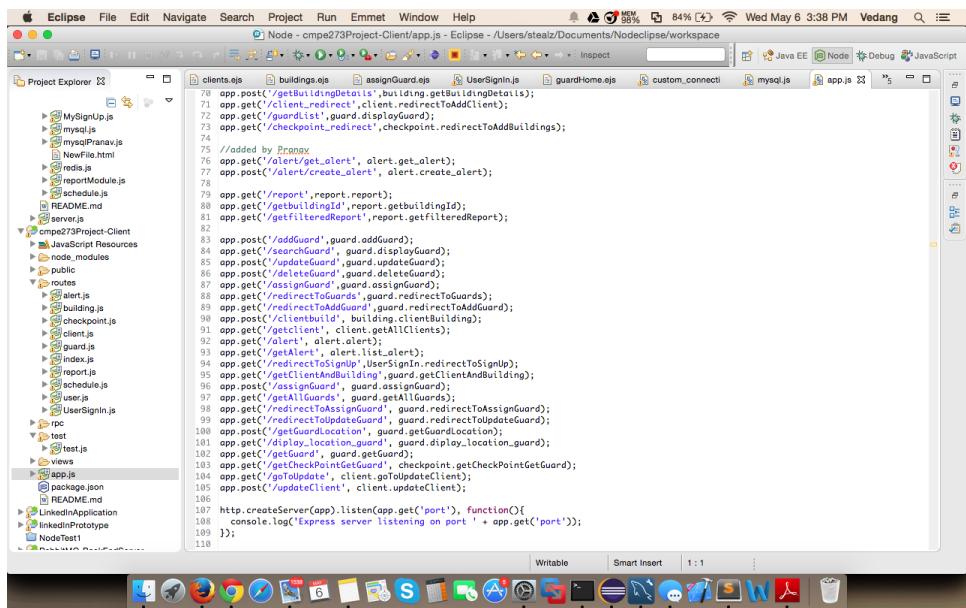
Session Object:



A screenshot of the Eclipse IDE interface on a Mac OS X desktop. The title bar shows "Node - cmpe273Project-Client/routes/UserSignIn.js - Eclipse - /Users/stealz/Documents/Nodeclipse/workspace". The code editor displays a JavaScript file named "UserSignIn.js". The code handles user sign-in logic, including validating credentials, interacting with a message queue client, and sending a response. The code uses ES6 syntax like arrow functions and template literals. The Eclipse interface includes toolbars, a project explorer, and various tabs for Java EE, Node.js, and JavaScript.

```
1 module.exports = (req, res) =>
2     let ejjs = require('ejjs');
3     let mq_client = require('../rpc/client');
4
5     function signin(req, res)
6     {
7         let username = req.param('username');
8         let password = req.param('password');
9         let msg_payload = {"username":username,"password":password,"service":"signin"};
10
11         mq_client.make_request("sign_queue", msg_payload, function(err, response){
12             if(err)
13                 throw err;
14             else
15                 {
16                     if(response.role == "guard")
17                         {
18                             console.log("GUARD", response.results[0].guard_id);
19                             req.session.guard_id=response.results[0].guard_id;
20                             console.log("Guard ID", req.session.guard_id);
21                             res.send({"signin":"Success","role":"guard"});
22                         }
23                     else
24                         {
25                             req.session.username=response.results[0].username;
26                             req.session.prof=response.results[0].user_id;
27                             req.session.check_response.results[0].password;
28                             console.log(JSON.stringify(response));
29
30                             res.send({"signin":"Success","role":"admin"});
31                         }
32
33                 }
34
35             });
36         });
37     }
38
39     function signup(req, res)
40     {
41 }
```

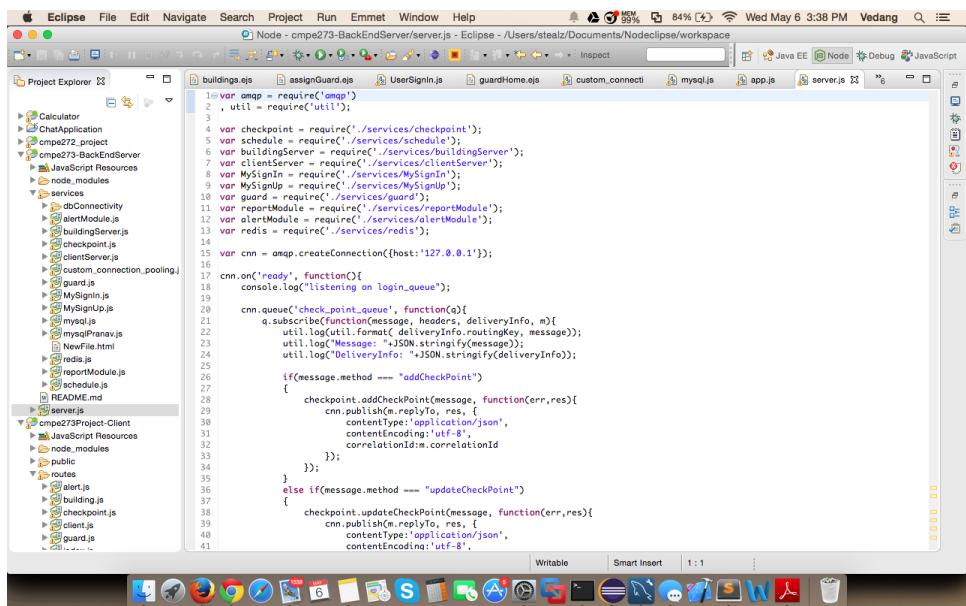
App.js



```
Project Explorer     clients.ejs     buildings.ejs     assignGuard.ejs     UserSignIn.js     guardHome.ejs     custom_connecti...     mysql.js     app.js     5%     Wed May 6 3:38 PM     Vedang     Java EE     Node.js     Debug     JavaScript

app.post('/getBuildingDetails', building.getBuildingDetails);
app.get('/client_redirect', client.redirectToAddClient);
app.get('/guardlist', guard.displayGuard);
app.get('/checkpoint_redirect', checkpoint.redirectToAddBuildings);
//added by Pranav
app.get('/alert/get_alert', alert.getAlert);
app.post('/alert/create_alert', alert.createAlert);
app.get('/report/report', report.report);
app.get('/getbuildingId', report.getBuildingID);
app.get('/filteredReport', report.getFilteredReport);
app.post('/addGuard', guard.addGuard);
app.get('/searchGuard', guard.displayGuard);
app.post('/updateGuard', guard.updateGuard);
app.post('/deleteGuard', guard.deleteGuard);
app.get('/clientAndBuilding', guard.getClientAndBuilding);
app.get('/redirectToAddGuard', guard.redirectToAddGuard);
app.get('/clientbuild', building.clientBuilding);
app.get('/getClient', client.getAllClients);
app.get('/alert/list_alert');
app.get('/redirectToSignUp', UserSignIn.redirectToSignUp);
app.get('/clientAndBuilding', guard.getClientAndBuilding);
app.post('/assignGuard', guard.assignGuard);
app.get('/redirectToAssignGuard', guard.redirectToAssignGuard);
app.get('/redirectToUpdateGuard', guard.redirectToUpdateGuard);
app.get('/getGuardLocation', guard.getGuardLocation);
app.get('/display_location_guard', guard.display_location_guard);
app.get('/getGuard', guard.getGuard);
app.get('/getCheckPoint', checkpoint.getCheckPointGetGuard);
app.get('/goUpdate', client.goToUpdateClient);
app.post('/updateClient', client.updateClient);
http.createServer(app).listen(app.get('port'), function(){
  console.log('Express server listening on port ' + app.get('port'));
});
});
```

Server.js



```
Project Explorer     buildings.ejs     assignGuard.ejs     UserSignIn.js     guardHome.ejs     custom_connecti...     mysql.js     app.js     server.js     6%     Wed May 6 3:38 PM     Vedang     Java EE     Node.js     Debug     JavaScript

var amqp = require('amqp');
var util = require('util');
var checkpoint = require('../services/checkpoint');
var building = require('../services/building');
var clientServer = require('../services/clientServer');
var MySignIn = require('../services/MySignIn');
var MySignUp = require('../services/MySignUp');
var guard = require('../services/guard');
var reportModule = require('../services/reportModule');
var alertModule = require('../services/alertModule');
var redis = require('../services/redis');

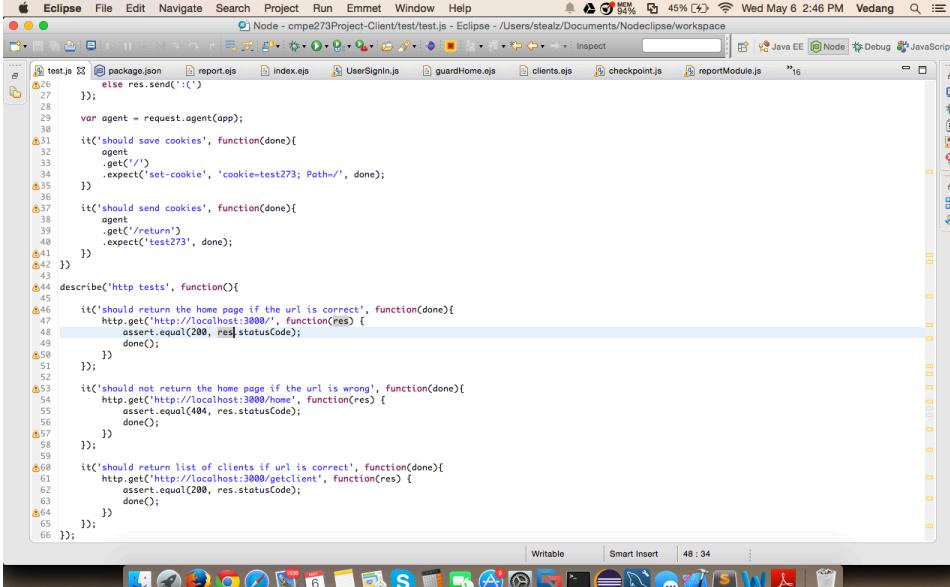
var cm = amqp.createConnection({host:'127.0.0.1'});

cm.on('ready', function(){
  console.log("listening on login_queue");
  cm.queue('check_point_queue', function(queue){
    queue.subscribe(function(message, headers, deliveryInfo, m){
      util.log(util.format(`deliveryInfo.routingKey, message`));
      util.log(`Message: ${JSON.stringify(message)}`);
      util.log(`DeliveryInfo: ${JSON.stringify(deliveryInfo)}`);

      if(message.method === "addCheckPoint")
      {
        checkpoint.addCheckPoint(message, function(err,res){
          cnn.publish(m.replyTo, res, {
            contentType:'application/json',
            contentEncoding:'utf-8',
            correlationId:m.correlationId
          });
        });
      }
      else if(message.method === "updateCheckPoint")
      {
        checkpoint.updateCheckPoint(message, function(err,res){
          cnn.publish(m.replyTo, res, {
            contentType:'application/json',
            contentEncoding:'utf-8',
          });
        });
      }
    });
  });
});
```

MOCHA TESTS

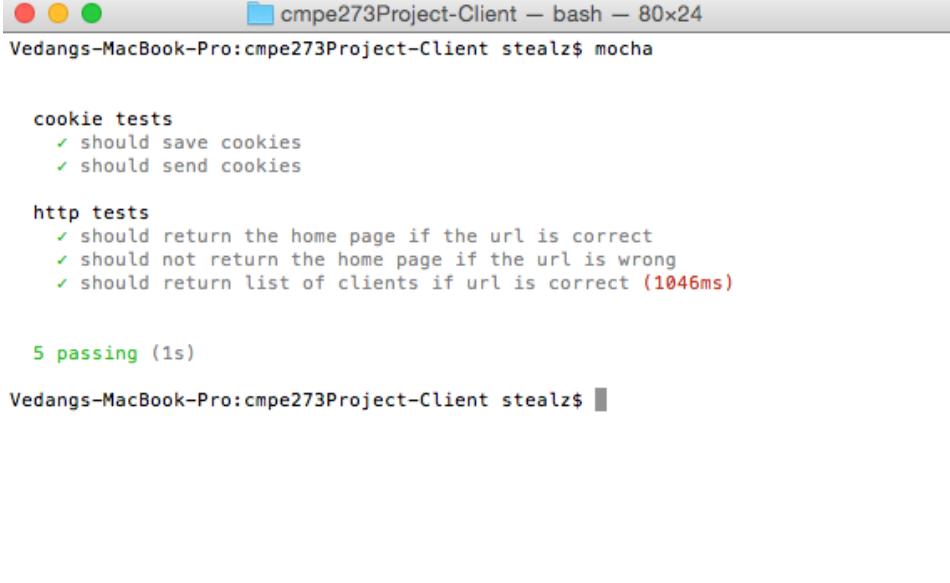
Mocha Test Code



```
test.js 26 package.json report.ejs index.ejs UserSignIn.js guardHome.ejs clients.ejs checkpoint.js reportModule.js 16
1 Node - cmpe273Project-Client/test/test.js - Eclipse - /Users/stealz/Documents/Nodeclipse/workspace
26
27     else res.send(':(')
28
29     var agent = request.agent(app);
30
31     it('should save cookies', function(done){
32         agent
33             .get('/')
34             .expect('set-cookie', 'cookie-test273; Path=/', done);
35     });
36
37     it('should send cookies', function(done){
38         agent
39             .get('/return')
40             .expect('test273', done);
41     });
42 });
43
44 describe('http tests', function(){
45
46     it('should return the home page if the url is correct', function(done){
47         http.get('http://localhost:3000/', function(res) {
48             assert.equal(200, res.statusCode);
49             done();
50         });
51     });
52
53     it('should not return the home page if the url is wrong', function(done){
54         http.get('http://localhost:3000/home', function(res) {
55             assert.equal(404, res.statusCode);
56             done();
57         });
58     });
59
60     it('should return list of clients if url is correct', function(done){
61         http.get('http://localhost:3000/getclient', function(res) {
62             assert.equal(200, res.statusCode);
63             done();
64         });
65     });
66 });

```

Mocha Test Results



```
Vedangs-MacBook-Pro:cmpe273Project-Client stealz$ mocha

cookie tests
  ✓ should save cookies
  ✓ should send cookies

http tests
  ✓ should return the home page if the url is correct
  ✓ should not return the home page if the url is wrong
  ✓ should return list of clients if url is correct (1046ms)

  5 passing (1s)

Vedangs-MacBook-Pro:cmpe273Project-Client stealz$
```

Performance tuning techniques

1. Connection pooling:

Connection pooling is used to improve the performance of the application. Connection pooling servers require less time to serve the request than a Connectionless-pooling server. Accessing database for each and every query is a time consuming and heavy task that degrades the performance. That is where connection pooling is best used. This algorithm creates a pool of min 10 Database connections and max of 20 Database connections. When a database query is executed, a connection from the pool is used instead of creating a new connection to fire the query. The connection is returned back to the pool after its execution to be used for another request.

2. Rabbit MQ

Rabbit MQ is an excellent message passing service that serves request from client to server and delivers response from server to client. We have created separate queues for most of the important modules such that the request from client to get alerts is put into get-alert_queue and response is delivered from the same queue. Other important queues that serve the purpose of our application are checkpoint queue, signin queue, guard queue and many more that helps improve performance between the client and the server.

3. Redis Caching

Redis Caching is used in our application to get large volume of data such as guard list, building list and client list. Implementation of this caching technique drastically improved the performance of the application. As Redis is an in-memory database, memory representation of complex data structure is much simpler. When the application first executes, the data is fetched. And for the requests that are recurring, the values are stored in keys.

```

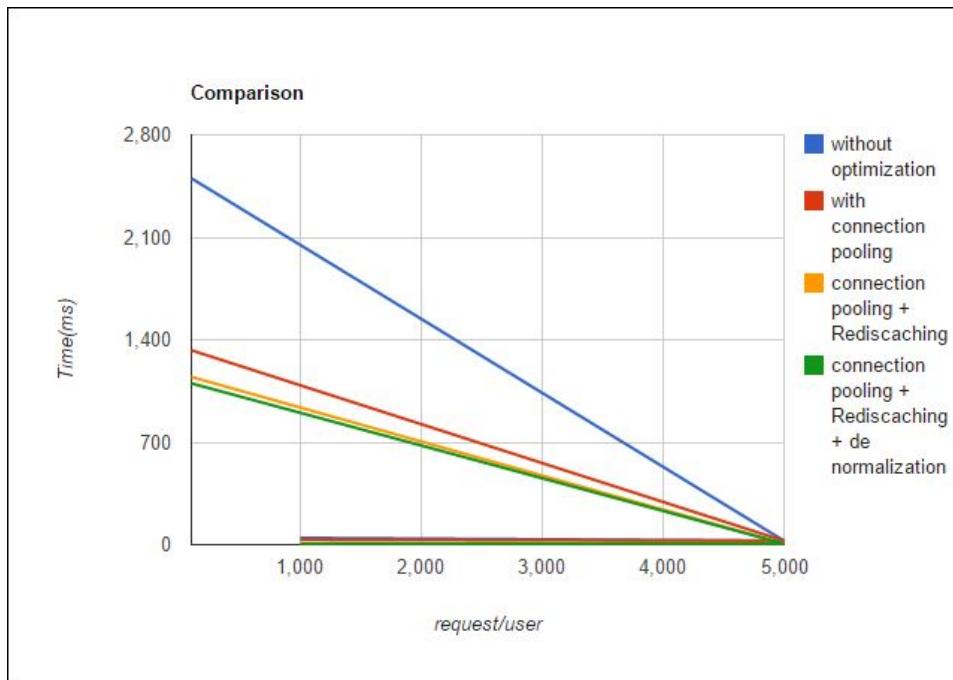
Terminal Shell Edit View Window Help
stealz — redis-cli - 147x44
127.0.0.1:6379 get 'select client_id,client_number,client_name,monthly_service_charge,Balance,service_start_date,service_end_date from client'
"{"client_id":13,"client_number":"000-111-1111","client_name":"Vedang Jadhav","monthly_service_charge":100,"Balance":100,"service_start_date":"2015-01-10 00:00:00+05:30","service_end_date":"2015-06-16 00:00:00+05:30","numbe
r":111-111-2222}, {"client_id":17,"client_number":444-444-4444,"client_name":Nikhil Chavan,"monthly_service_charge":1200,"Balance":100,"service_start_date
":2015-12-12 00:00:00+05:30,"service_end_date":2016-12-12 00:00:00+05:30}, {"client_id":19,"client_number":123-234-5555,"client_name":Pranav Dhapke,"monthly
service_charge":120,"Balance":100,"service_start_date":2014-11-21 00:00:00+05:30,"service_end_d
ate":2015-12-12 00:00:00+05:30}, {"client_id":139,"service_start_date":2015-12-12 00:00:00+05:30,"service_end_date":2016-12-12 00:00:00+05:30}]", "19 connections on port 6379"
success
success
success
success
success

```

COMPARISON TABLE

Details	Time for 5000 request in ms	Time for 1000 request in ms	Time for 100 users-1000 request per users in ms
Without any optimization technique	46	29	2503
Added connection pooling	35	26	1328
Added Redis caching	9	7	1146
Added DB de-normalization	7	11	1102

GRAPH COMPARING ABOVE RESULTS



CONTRIBUTIONS

1. VEDANG JADHAV

- Worked on design and analysis of the system.
- Worked on the Checkpoint Module.
- Worked on implementing caching functionality using Redis.
- Worked on implementing the connection pooling.
- Worked on integrating all the separate modules.
- Tested the integrated application.

2. NIKHIL CHAVAN

- Worked on design and analysis of the system.
- Worked on the Guard Module.
- Worked on implementing the connection pooling.

- Worked on the Wireframe Design of the application.
- Worked on the logging and plotting of graphs using Apache JMeter.
- Worked on integrating all the separate modules.

3. PRANAV DHAPKE

- Worked on design and analysis of the system.
- Worked on the Client Module.
- Worked on the DB schema and the normalization/de-normalization of the database.
- Worked on the Wireframe Design of the application.
- Implemented Session Handling.
- Worked on the prepared statement for Database queries.

4. DARSHIT LAKHANI

- Worked on design and analysis of the system.
- Worked on the Building Module.
- Worked on implementing caching functionality using Redis.
- Worked on test cases using Mocha and Supertest.
- Worked on integrating all the separate modules.
- Tested the integrated application.

5. MADHAV VASUDEV

- Worked on design and analysis of the system.
- Worked on the Alert Module.
- Worked on the DB schema and the normalization/de-normalization of the database.
- Worked on test cases using Mocha and Supertest.
- Worked on the prepared statement for Database queries.

- Worked on the logging and plotting of graphs using Apache JMeter.

6. DOLLY GUPTA

- Worked on design and analysis of the system.
- Worked on the Report Module.
- Worked on the Wireframe Design of the application.
- Populated the Database with mock data.
- Worked on the logging and plotting of graphs using Apache JMeter.
- Implemented the MD5 hashing algorithm.

OBSERVATIONS AND LESSONS LEARNT

- Since Node.Js is asynchronous, the database operations, which were interdependent, would not work as required. So, we added callback mechanism to accomplish a synchronous behavior.
- We learnt how to make an API framework, which would work as the guideline for the team members to follow.
- We learnt how to work in a large team, how to divide work and integrate all the work into one final application.
- We learnt how to write our test cases using mocha and supertest.