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.

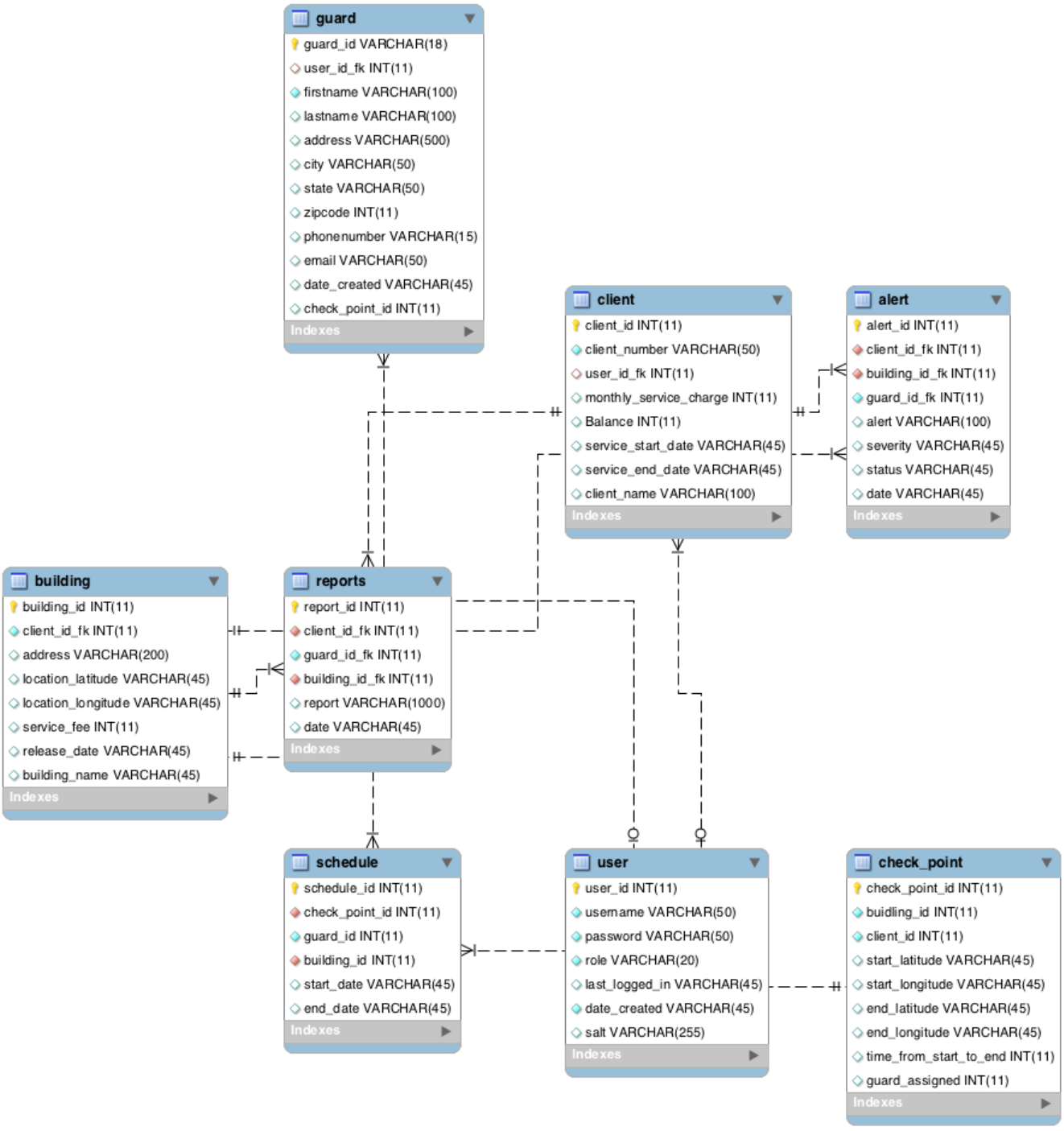
- Report

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`
--
```

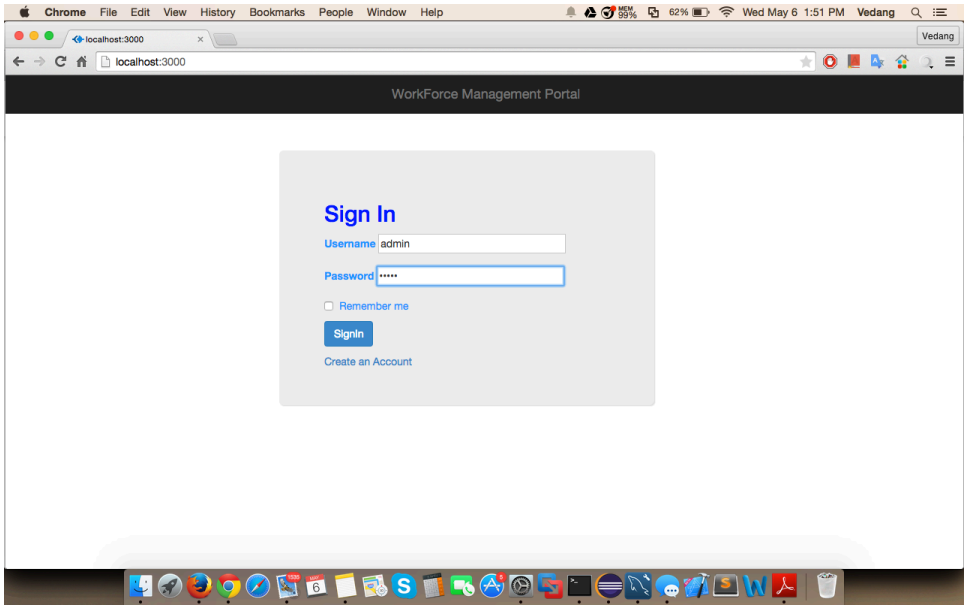
```
DROP TABLE IF EXISTS `guard`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `guard` (
`guard_id` varchar(18) NOT NULL,
`user_id_fk` int(11) DEFAULT NULL,
`firstname` varchar(100) NOT NULL,
`lastname` varchar(100) DEFAULT NULL,
`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`),
```

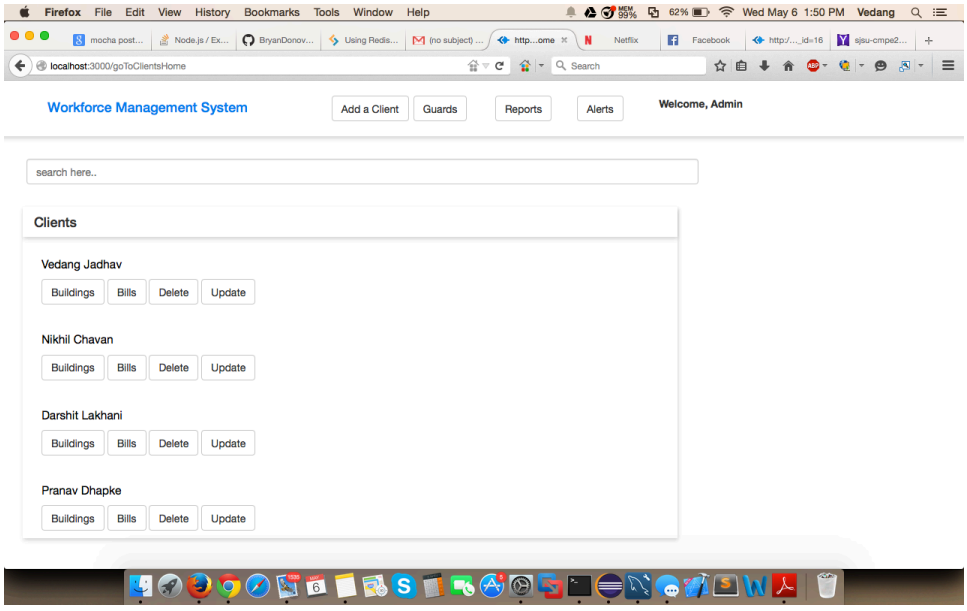
```
KEY `schedule_guard_id_idx` (`guard_id`),
KEY `schedule_building_id_idx` (`building_id`),
CONSTRAINT `schedule_building_id` FOREIGN KEY (`building_id`) REFERENCES `building` (`building_id`)
ON DELETE NO ACTION ON UPDATE NO ACTION,
CONSTRAINT `schedule_cp_id` FOREIGN KEY (`check_point_id`) REFERENCES `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

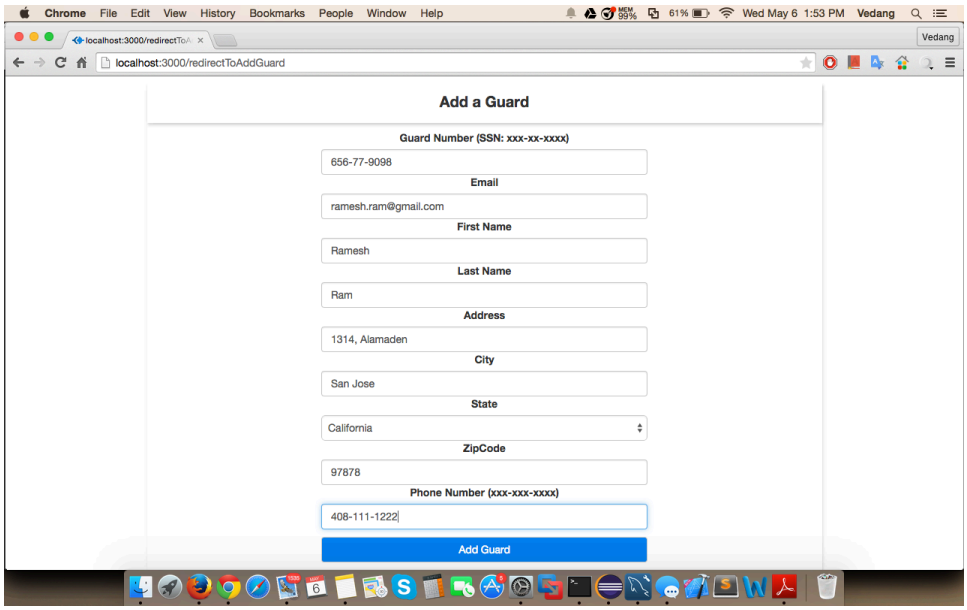
Admin Login



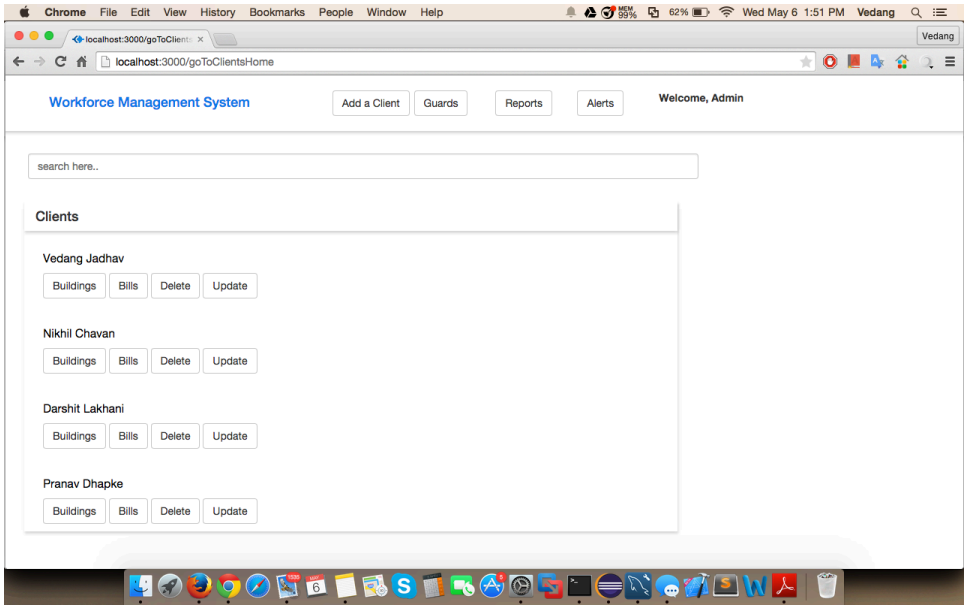
Admin Homepage



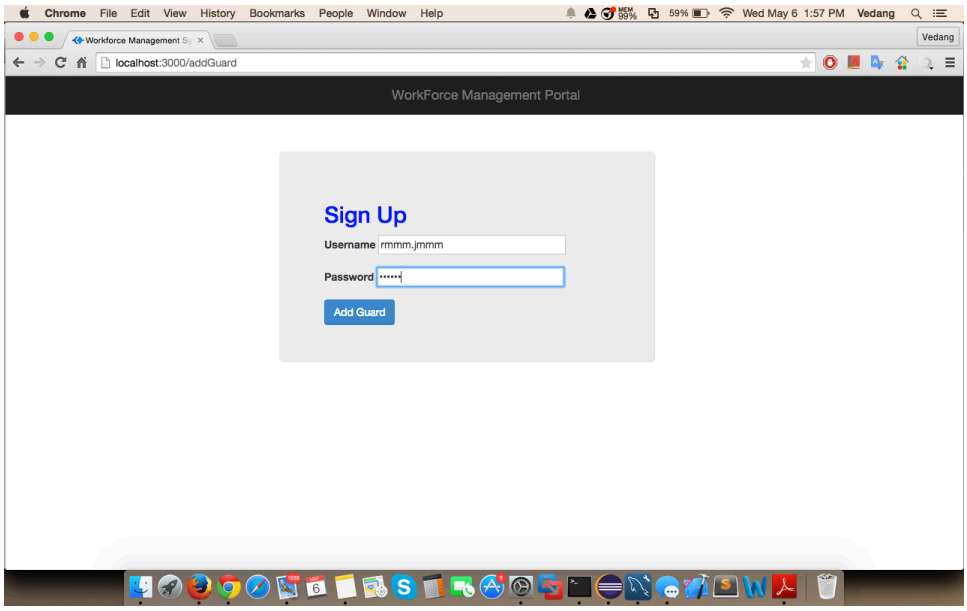
Add Guard



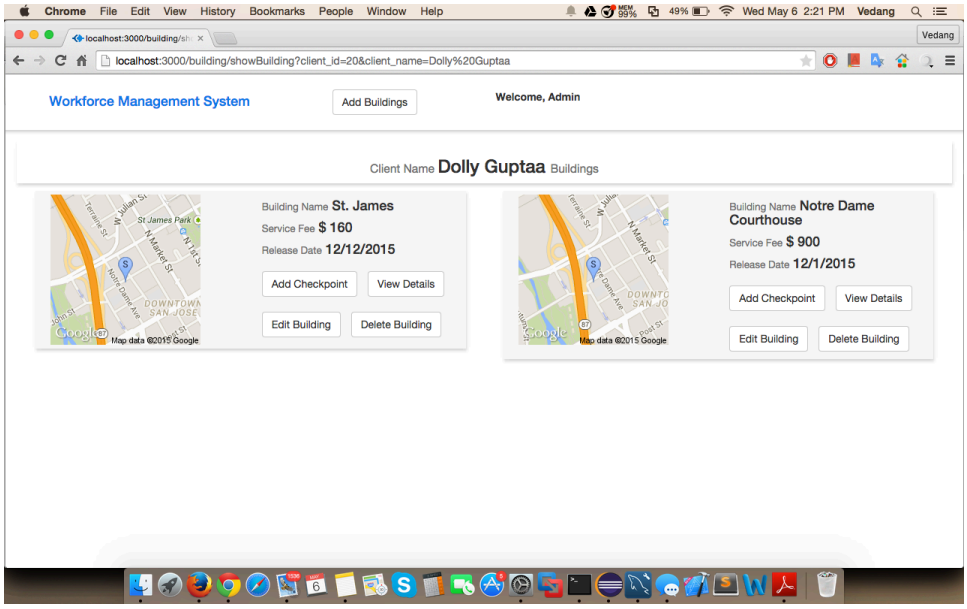
View Clients



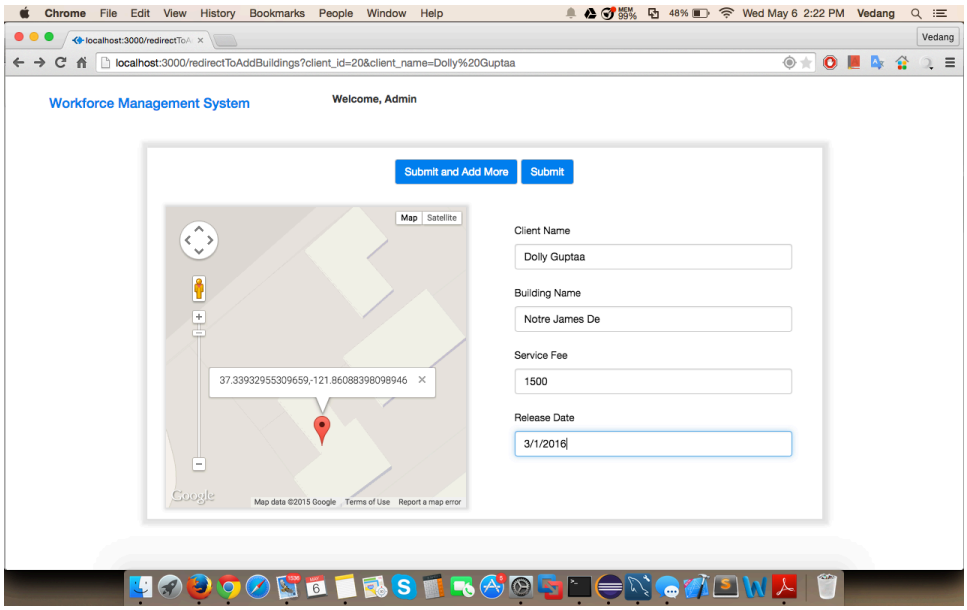
Add Guard



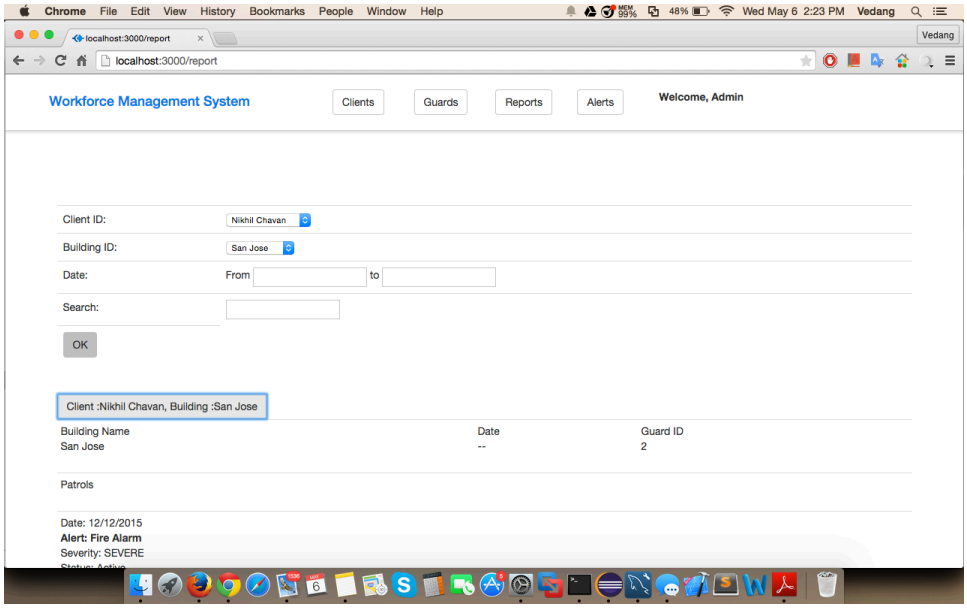
View Building



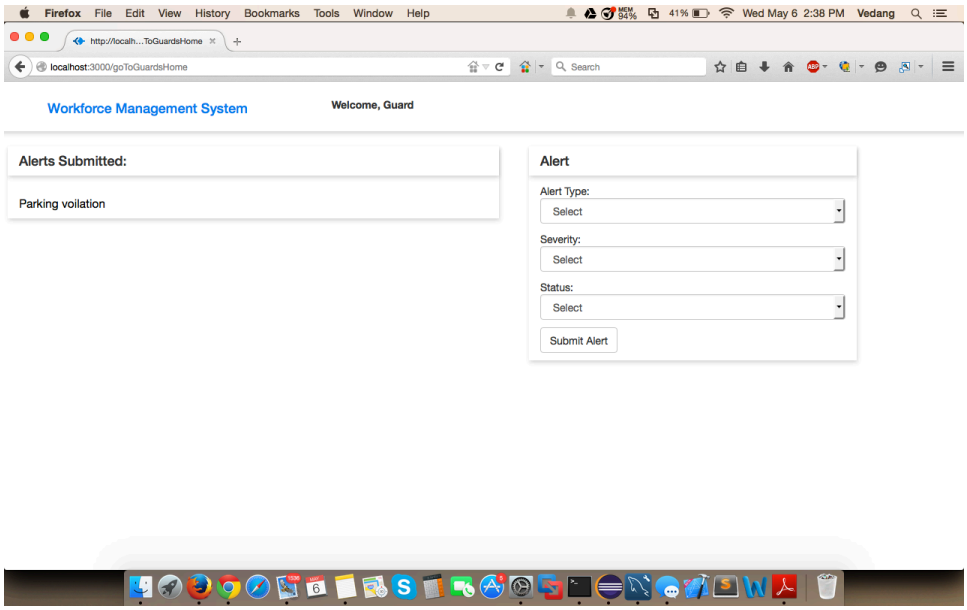
Add Building



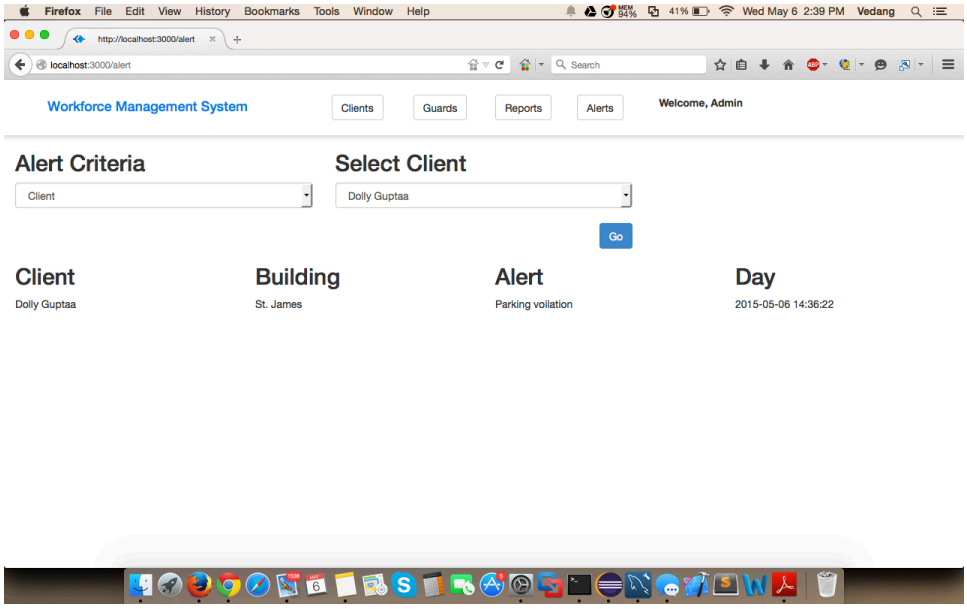
View Reports



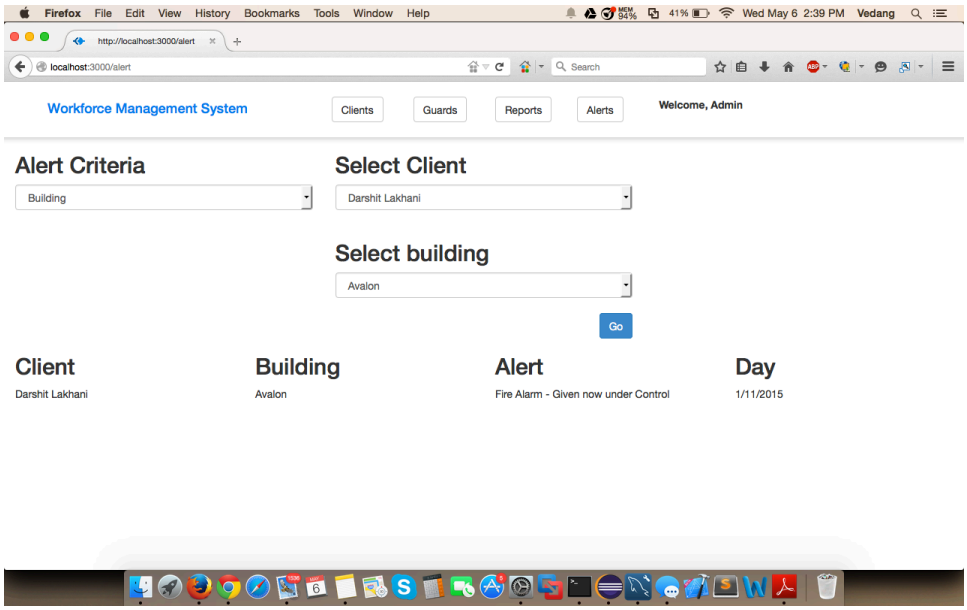
Guard Homepage



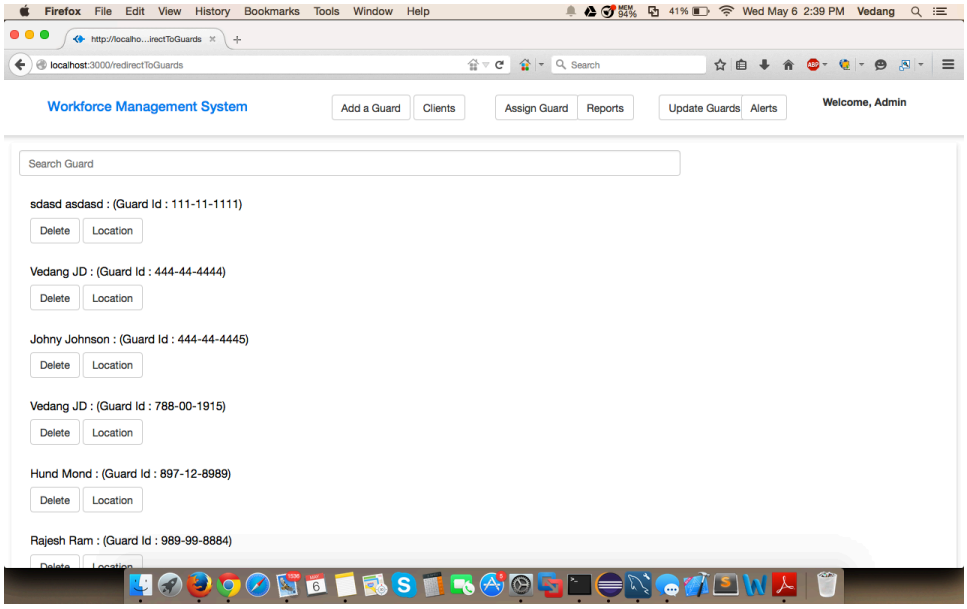
View alerts by Client



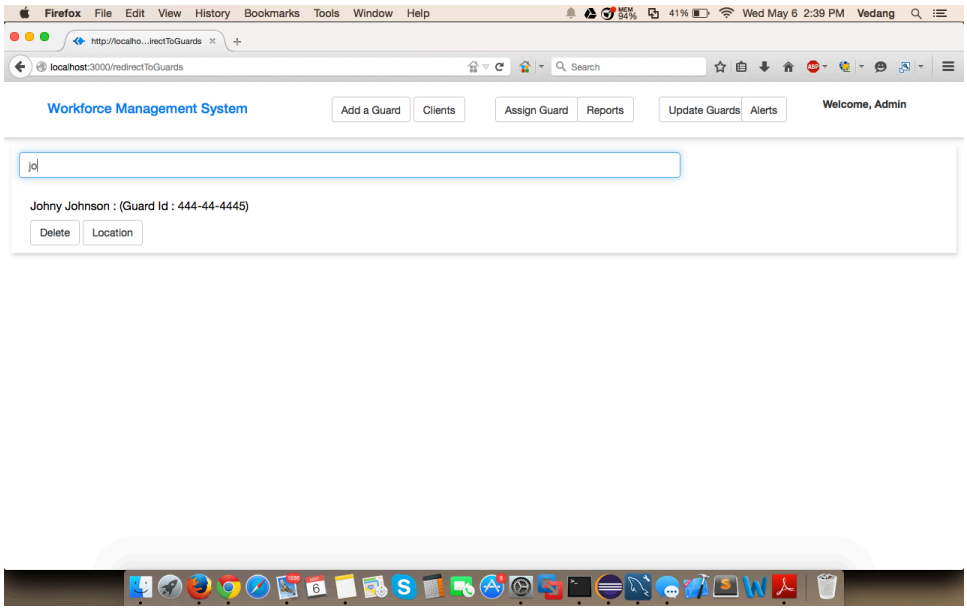
View Alerts by Building



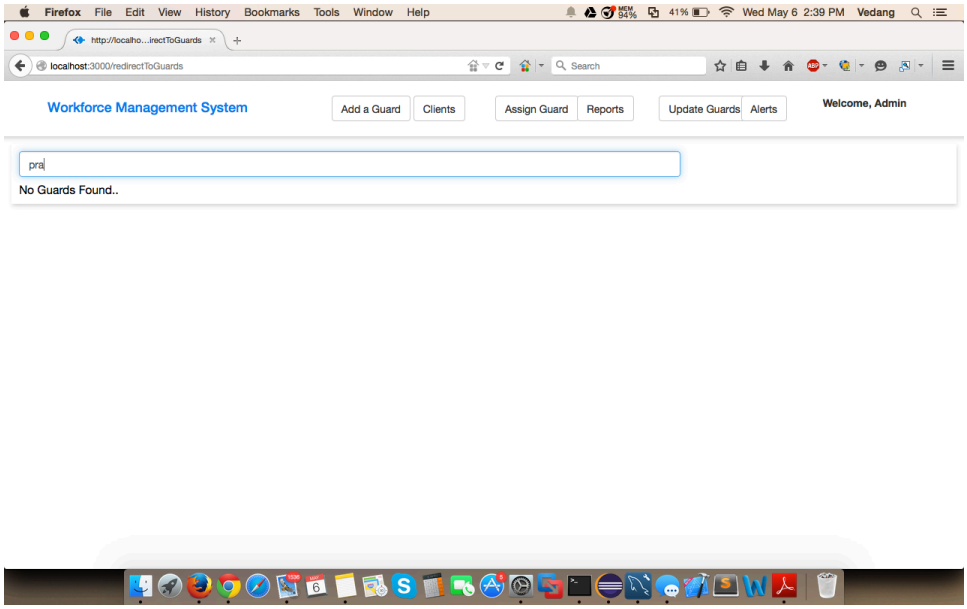
Search Guard



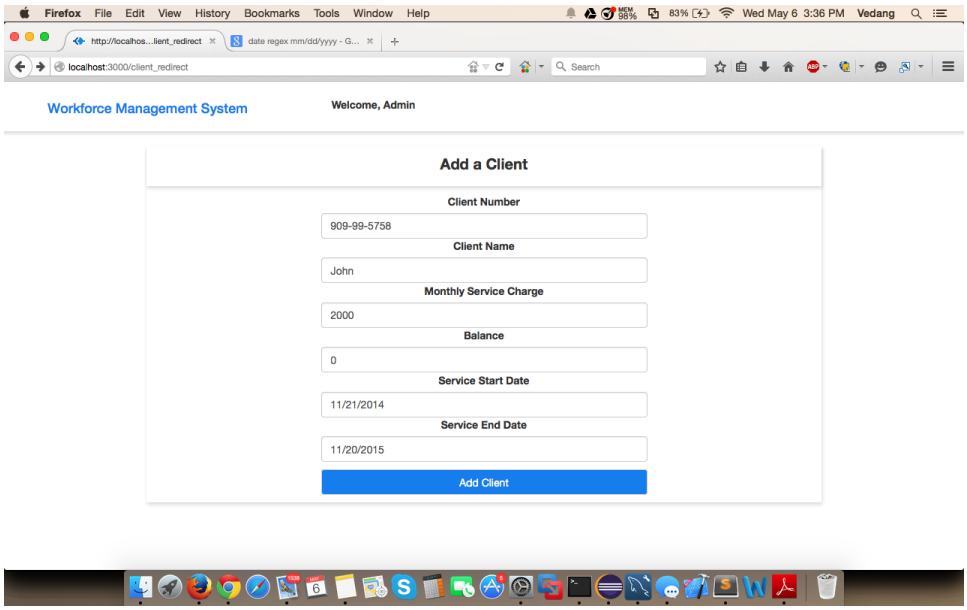
Search Guard



Search Guard

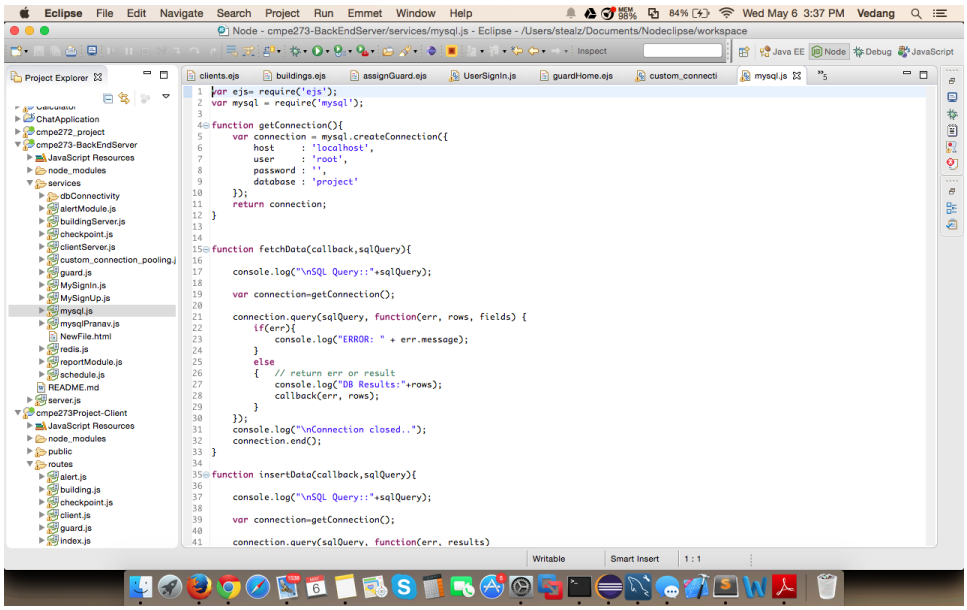


Add Client

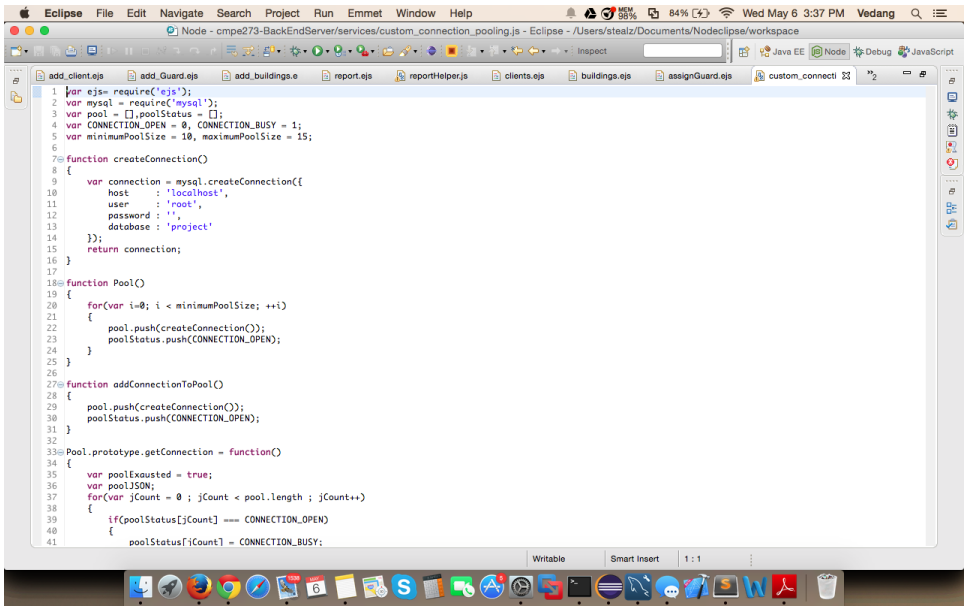


CODE LISTING

MySql

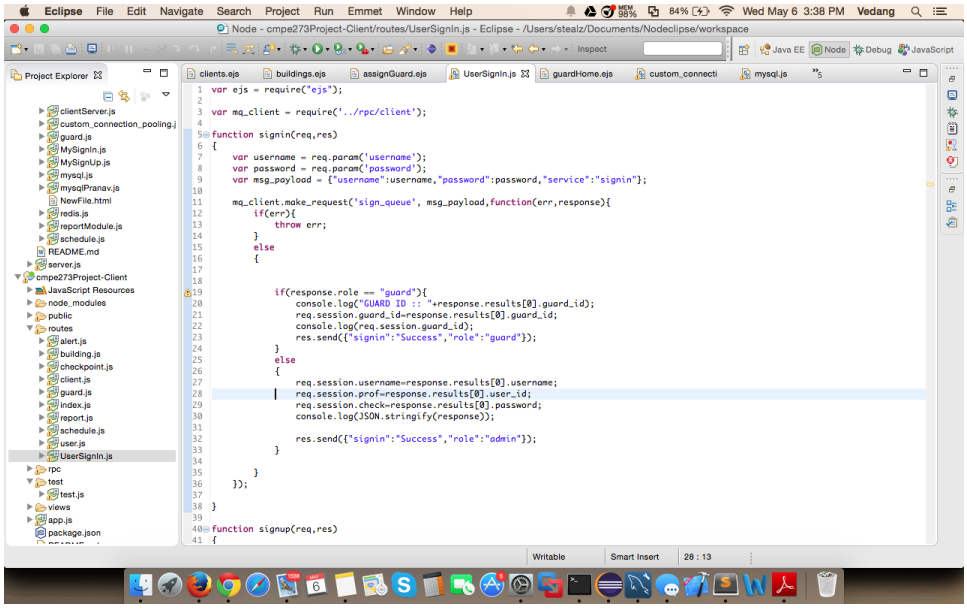


Connection Pooling:



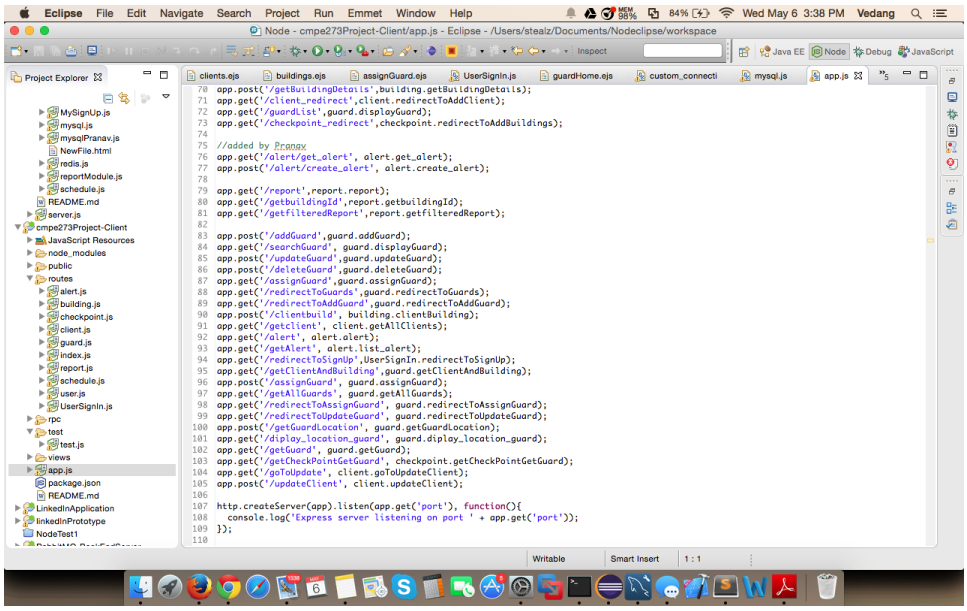
```
var ejs= require('ejs');
var mysql = require('mysql');
var pool = [],poolStatus = [];
var CONNECTION_OPEN = 0, CONNECTION_BUSY = 1;
var minimumPoolSize = 10, maximumPoolSize = 15;

function createConnection()
{
    var connection = mysql.createConnection({
        host     : 'localhost',
        user     : 'root',
        password : '',
        database : 'project'
    });
    return connection;
}

function Pool()
{
    for(var i=0; i < minimumPoolSize; ++i)
    {
        pool.push(createConnection());
        poolStatus.push(CONNECTION_OPEN);
    }
}

function addConnectionToPool()
{
    pool.push(createConnection());
    poolStatus.push(CONNECTION_OPEN);
}

Pool.prototype.getConnection = function()
{
    var poolExausted = true;
    var poolJSON;
    for(var jCount = 0 ; jCount < pool.length ; jCount++)
    {
        if(poolStatus[jCount] === CONNECTION_OPEN)
        {
            poolStatus[jCount] = CONNECTION_BUSY;
```

Session Object:

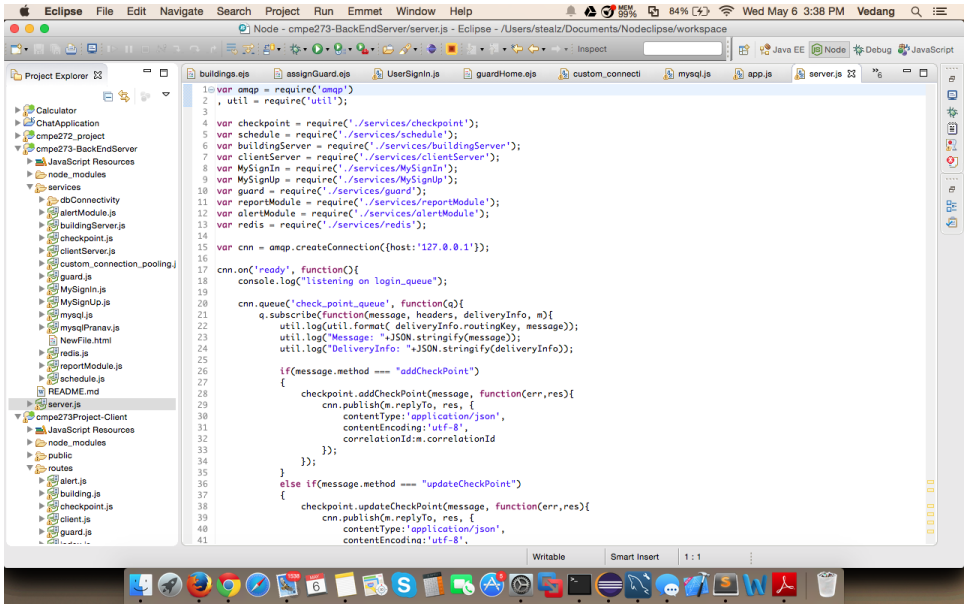


```
var ejs = require("ejs");

var mq_client = require('../rpc/client');

function signin(req,res)
{
    var username = req.param('username');
    var password = req.param('password');
    var msg_payload = {"username":username,"password":password,"service":"signin"};

    mq_client.make_request('sign_queue', msg_payload,function(err,response){
        if(err){
            throw err;
        }
        else
        {

            if(response.role == "guard"){
                console.log("GUARD ID :: "+response.results[0].guard_id);
                req.session.guard_id=response.results[0].guard_id;
                console.log(req.session.guard_id);
                res.send({"signin":"Success","role":"guard"});
            }
            else
            {
                req.session.username=response.results[0].username;
                req.session.prof=response.results[0].user_id;
                req.session.check=response.results[0].password;
                console.log(JSON.stringify(response));

                res.send({"signin":"Success","role":"admin"});
            }

        }
    });
}

function signup(req,res)
{
```

App.js

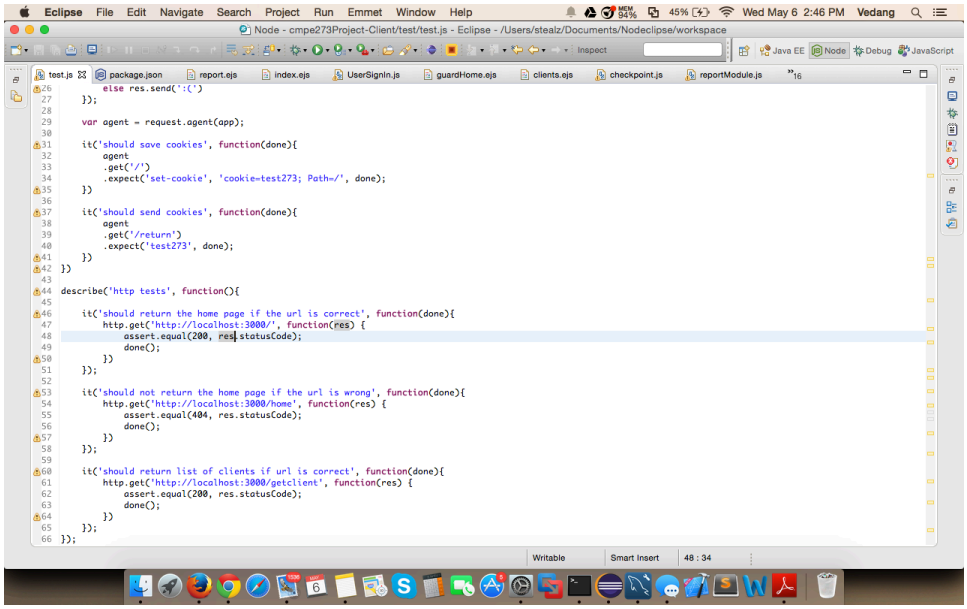


Server.js



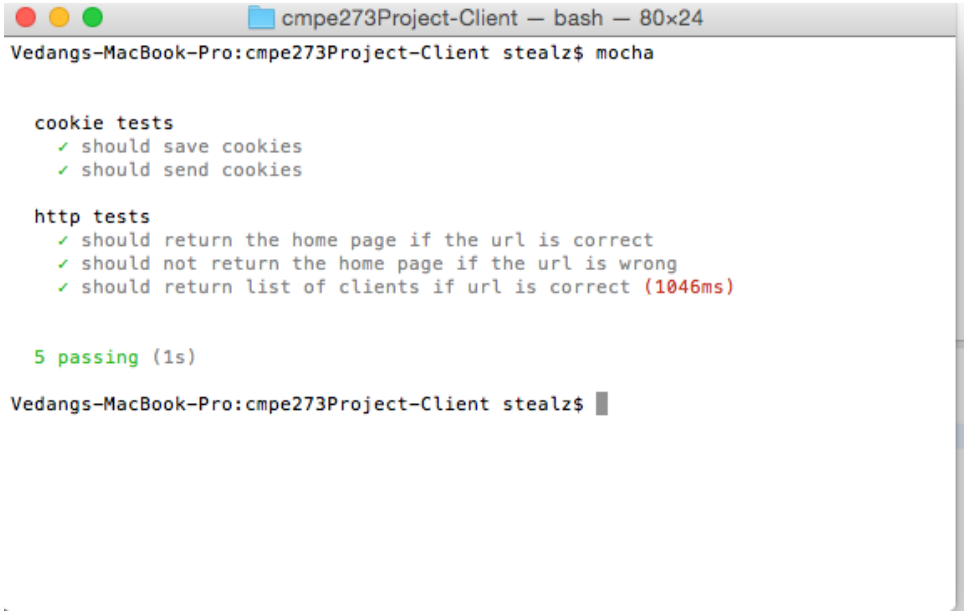
MOCHA TESTS

Mocha Test Code



```
    else res.send(':(')
});

var agent = request.agent(app);

it('should save cookies', function(done){
    agent
    .get('/')
    .expect('set-cookie', 'cookie=test273; Path=/', done);
})

it('should send cookies', function(done){
    agent
    .get('/return')
    .expect('test273', done);
})
})

describe('http tests', function(){

    it('should return the home page if the url is correct', function(done){
        http.get('http://localhost:3000/', function(res) {
            assert.equal(200, res.statusCode);
            done();
        })
    });

    it('should not return the home page if the url is wrong', function(done){
        http.get('http://localhost:3000/home', function(res) {
            assert.equal(404, res.statusCode);
            done();
        })
    });

    it('should return list of clients if url is correct', function(done){
        http.get('http://localhost:3000/getclient', function(res) {
            assert.equal(200, res.statusCode);
            done();
        })
    });
});
```

Mocha Test Results



```
cmpe273Project-Client — bash — 80×24
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:

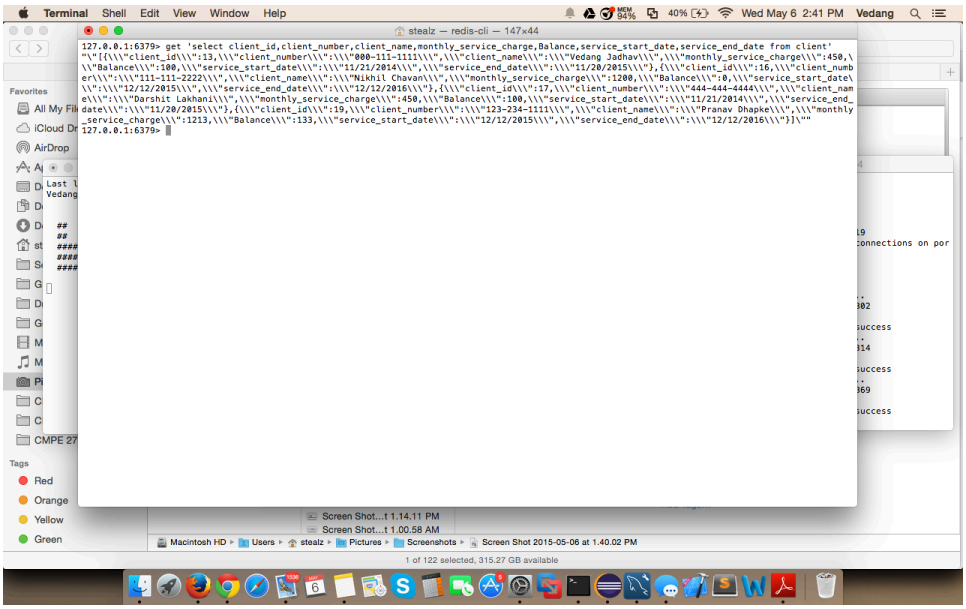
Connection pooling is used to improve the performance of the application. Connection pooling servers require less time to serve the request than a Connectionless-pooling server. Accessing database for each and every query is a time consuming and heavy task that degrades the performance. That is where 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 out application are checkpoint queue, signin queue, guard queue and many more that helps improve performance between the client and the server.

3. Radis Caching

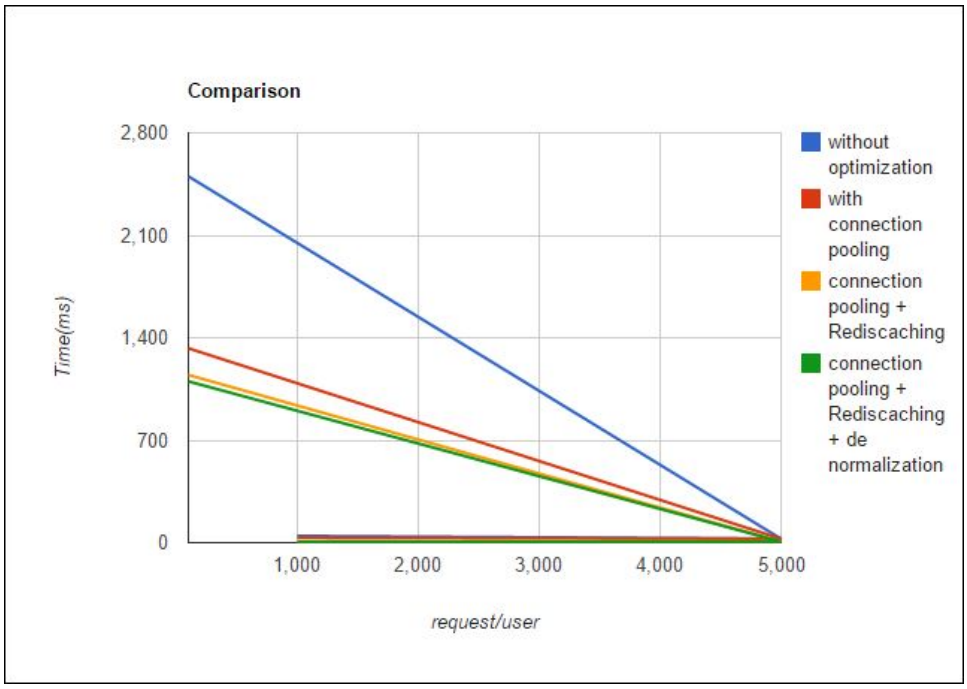
Redis Caching is used in our application to get large volume of data such as guard list, building list and client list. Implementation of this caching technique drastically improved the performance of the application. As Redis is an in-memory database, memory representation of complex data structure 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.



COMPARISION 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.