**Requirement Specification**

**Hardware Description**: The selection of hardware is very important in the existence and proper working of any software. When selecting hardware, the size and requirements are also important.

Minimum Requirements:

Processor : INTEL Core i3

RAM : 3GB

Hard Disk Drive : 250GB

The proposed System is developed on:

Processor : INTEL Core i5

RAM : 4GB

Hard Disk Drive : 500GB

**Software Description:**

Operating System : Windows 7 Service Pack 1

Front- End : MVC.NET AND ANGULAR 7

Back- End : MS SQL SERVER 2012 EXPRESS & SQL SERVER MANAGEMENT STUDIO

Editor : Visual Studio 2017

**Project Planning Document**

A project management project plan is one of the most fundamental parts of managing a project. They serve as the roadmap for the project and guide it along its way to completion. They may seem complex and time-consuming, however in the long-term their benefits do not go unnoticed.

**Visual Studio project files**

**RegistrationController.cs**

using EcaseMain.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Threading.Tasks;

using System.Web.Http;

using System.Web.Http.Cors;

using SendGrid;

using SendGrid.Helpers.Mail;

using System.Web;

using System.Diagnostics;

using System.Net.Http.Headers;

using System.IO;

using System.Web.Script.Serialization;

namespace EcaseMain.Controllers

{

[EnableCors(origins: "\*", headers: "\*", methods: "\*")]

public class RegistrationController : ApiController

{

ecaseEntities ecaseEntity = null;

public RegistrationController()

{

ecaseEntity = new ecaseEntities();

}

[HttpPost]

public IHttpActionResult Register(RegistrationDto registrationDto)

{

var currentRecord = ecaseEntity.registrations.Where(x => x.email == registrationDto.email).ToList();

if(currentRecord.Count > 0)

{

return Ok("false");

}

string token = Guid.NewGuid().ToString().Substring(0, 10);

registration register = new registration

{

firstname = registrationDto.firstName,

lastname = registrationDto.lastName,

email = registrationDto.email,

location = registrationDto.location,

password = registrationDto.password,

accounttype = registrationDto.accountType,

status = true,

token = token,

createdon = DateTime.Now,

updatedon = DateTime.Now

};

ecaseEntity.registrations.Add(register);

ecaseEntity.SaveChanges();

return Ok(register);

}

[HttpGet]

public IHttpActionResult GetAllLawyers()

{

var lawyerList = ecaseEntity.registrations.Where(x => x.accounttype == "2").ToList();

var LawyerSpecaialityList = new List<LawyerSpecialityDto>();

foreach (var lawyer in lawyerList)

{

string specialityName = "";

if (lawyer.speciality != null && lawyer.speciality != "")

{

var specialityId = Convert.ToInt32(lawyer.speciality);

specialityName = ecaseEntity.specialities.First(x => x.id == specialityId).name;

}

LawyerSpecialityDto lawyerSpecialityDto = new LawyerSpecialityDto {

Location = lawyer.location,

Id = lawyer.id,

Status = Convert.ToBoolean(lawyer.status),

FirstName = lawyer.firstname,

LastName = lawyer.lastname,

SpecalityName = specialityName

};

LawyerSpecaialityList.Add(lawyerSpecialityDto);

}

return Ok(LawyerSpecaialityList);

}

[HttpGet]

public IHttpActionResult GetAllClients()

{

var clientList = ecaseEntity.registrations.Where(x => x.accounttype == "1").ToList();

return Ok(clientList);

}

[HttpPost]

public IHttpActionResult ChangeStatus(int id)

{

var record = ecaseEntity.registrations.First(x => x.id == id);

if (record.status == true)

{ record.status = false; }

else

{ record.status = true; }

ecaseEntity.SaveChanges();

return Ok(record);

}

[HttpPost]

public IHttpActionResult ChangePassword(ChangePasswordDto changePasswordDto)

{

registration record = null;

try

{

record = ecaseEntity.registrations.First(x => x.email == changePasswordDto.email && x.password == changePasswordDto.oldPassword);

record.password = changePasswordDto.newPassword;

ecaseEntity.SaveChanges();

return Ok(record);

}

catch(Exception ex)

{

return Ok(record);

}

}

[HttpGet]

public IHttpActionResult GetMemberByEmail(string email)

{

registration record = null;

try

{

string base64String = "";

string specialityName = "";

record = ecaseEntity.registrations.First(x => x.email == email);

if (record.speciality != "" && record.speciality != null)

{

int specialityId = Convert.ToInt32(record.speciality);

specialityName = ecaseEntity.specialities.First(x => x.id == specialityId).name;

}

var filePath = HttpContext.Current.Server.MapPath("~/UploadFile/" + record.image);

if (File.Exists(filePath))

{

byte[] fileData = File.ReadAllBytes(filePath);

base64String = Convert.ToBase64String(fileData);

record.image = base64String;

record.speciality = specialityName;

}

return Ok(record);

}

catch(Exception ex)

{

return Ok(record);

}

}

[HttpGet]

public IHttpActionResult GetCases(string email)

{

registration record = null;

try

{

record = ecaseEntity.registrations.First(x => x.email == email);

int speciality = Convert.ToInt32(record.speciality);

var caseList = ecaseEntity.cases.Where(x => x.specialityid == speciality ).ToList();

var specailityName = ecaseEntity.specialities.First(x => x.id == speciality).name;

List<CasesRegistrationDto> casesRegistrationDtoList = new List<CasesRegistrationDto>();

foreach (var cas in caseList)

{

var clientRecord = ecaseEntity.registrations.First(x => x.id == cas.clientid);

CasesRegistrationDto casesRegistrationDto = new CasesRegistrationDto

{

id = cas.id,

title = cas.title,

description = cas.description,

clientName = clientRecord.firstname + " " + clientRecord.lastname,

specialityName = specailityName,

approvedLawyerId = Convert.ToInt32(cas.approvedlawyerid)

};

casesRegistrationDtoList.Add(casesRegistrationDto);

}

return Ok(casesRegistrationDtoList);

}

catch (Exception ex)

{

record = null;

return Ok(record);

}

}

[HttpPost]

public IHttpActionResult UpdateProfile(RegistrationDto registrationDto)

{

registration record = null;

try

{

record = ecaseEntity.registrations.First(x => x.email == registrationDto.email);

record.firstname = registrationDto.firstName;

record.lastname = registrationDto.lastName;

record.location = registrationDto.location;

record.speciality = registrationDto.speciality;

record.certificateno = registrationDto.certificateNo;

record.aboutme = registrationDto.aboutMe;

ecaseEntity.SaveChanges();

return Ok(record);

}

catch (Exception ex)

{

return Ok(record);

}

}

[HttpPost]

public IHttpActionResult UpdateImage(string emailId)

{

var record = ecaseEntity.registrations.First(x => x.email == emailId);

HttpResponseMessage response = new HttpResponseMessage();

var httpRequest = HttpContext.Current.Request;

if (httpRequest.Files.Count > 0)

{

foreach (string file in httpRequest.Files)

{

var postedFile = httpRequest.Files[file];

var fileName = Guid.NewGuid().ToString().Substring(0, 6) + postedFile.FileName;

record.image = fileName;

var filePath = HttpContext.Current.Server.MapPath("~/UploadFile/" + fileName);

postedFile.SaveAs(filePath);

byte[] fileData = File.ReadAllBytes(filePath);

if (fileData == null)

throw new HttpResponseException(HttpStatusCode.NotFound);

var base64String = Convert.ToBase64String(fileData);

ecaseEntity.SaveChanges();

return Ok(base64String);

}

}

return Ok(response);

}

}

}

**LoginController.cs**

using EcaseMain.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Threading.Tasks;

using System.Web.Http;

using System.Web.Http.Cors;

using SendGrid;

using SendGrid.Helpers.Mail;

namespace EcaseMain.Controllers

{

[EnableCors(origins: "\*", headers: "\*", methods: "\*")]

public class LoginController : ApiController

{

ecaseEntities ecaseEntity = null;

public LoginController()

{

ecaseEntity = new ecaseEntities();

}

[HttpPost]

public IHttpActionResult Login(RegistrationDto registrationDto)

{

registration loginStatus = null;

try

{

loginStatus = ecaseEntity.registrations.First(x => x.email == registrationDto.email && x.password == registrationDto.password);

if(loginStatus.status == false)

{

return Ok("Inactive");

}

return Ok(loginStatus);

}

catch(Exception ex)

{

return Ok(loginStatus);

}

}

[HttpPost]

public IHttpActionResult ForgotPassword(string emailId)

{

registration loginStatus = null;

try

{

loginStatus = ecaseEntity.registrations.First(x => x.email == emailId);

string password = loginStatus.password;

SendMessageAsync(emailId, password).Wait();

return Ok("Done");

}

catch (Exception ex)

{

return Ok(loginStatus);

}

}

[HttpPost]

public IHttpActionResult AdminLogin(RegistrationDto registrationDto)

{

registration loginStatus = null;

try

{

loginStatus = ecaseEntity.registrations.First(x => x.email == registrationDto.email && x.password == registrationDto.password && x.accounttype == "0");

return Ok(loginStatus);

}

catch (Exception ex)

{

return Ok(loginStatus);

}

}

private async Task SendMessageAsync(string toEmailId, string password)

{

Response response = null;

try

{

var client = new SendGridClient("SG.U1pTRAPJT3GvU2uffDFVKA.9Lt--zn\_a3ivLs\_U-M0MU80bxXR\_SZlCEToyMlh6lok");

var from = new EmailAddress("baljeetcodeapp@gmail.com");

var to = new EmailAddress(toEmailId);

string link = "Your current password is "+ password;

var msg = MailHelper.CreateSingleEmail(from, to,

"Reset Password", "Reset Password", link);

response = await client.SendEmailAsync(msg).ConfigureAwait(false);

}

catch (Exception e)

{

var statusCode = response?.StatusCode.ToString() ?? "None";

}

}

}

}

**RegionsController.cs**

using EcaseMain.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Threading.Tasks;

using System.Web.Http;

using System.Web.Http.Cors;

using SendGrid;

using SendGrid.Helpers.Mail;

namespace EcaseMain.Controllers

{

[EnableCors(origins: "\*", headers: "\*", methods: "\*")]

public class RegionsController : ApiController

{

ecaseEntities ecaseEntity = null;

public RegionsController()

{

ecaseEntity = new ecaseEntities();

}

[HttpGet]

public IHttpActionResult GetAll()

{

List<region> regionList = new List<region>();

regionList = ecaseEntity.regions.ToList();

return Ok(regionList);

}

[HttpGet]

public IHttpActionResult GetById(int regionId)

{

var region = ecaseEntity.regions.First(x => x.id == regionId);

return Ok(region);

}

[HttpPost]

public IHttpActionResult Create(RegionsDto regionsDto)

{

region region = new region

{

name = regionsDto.name,

createdon = DateTime.Now,

updatedon = DateTime.Now

};

ecaseEntity.regions.Add(region);

ecaseEntity.SaveChanges();

return Ok(region);

}

[HttpPost]

public IHttpActionResult Delete(int regionId)

{

var region = ecaseEntity.regions.First(x => x.id == regionId);

ecaseEntity.regions.Remove(region);

ecaseEntity.SaveChanges();

return Ok(region);

}

[HttpPost]

public IHttpActionResult Update(RegionsDto regionsDto)

{

var region = ecaseEntity.regions.First(x => x.id == regionsDto.id);

region.name = regionsDto.name;

region.updatedon = DateTime.Now;

ecaseEntity.SaveChanges();

return Ok(region);

}

}

}

**SpecialitiesController.cs**

using EcaseMain.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Threading.Tasks;

using System.Web.Http;

using System.Web.Http.Cors;

using SendGrid;

using SendGrid.Helpers.Mail;

namespace EcaseMain.Controllers

{

[EnableCors(origins: "\*", headers: "\*", methods: "\*")]

public class SpecialitiesController : ApiController

{

ecaseEntities ecaseEntity = null;

public SpecialitiesController()

{

ecaseEntity = new ecaseEntities();

}

[HttpGet]

public IHttpActionResult GetAll()

{

List<speciality> specialityList = new List<speciality>();

specialityList = ecaseEntity.specialities.ToList();

return Ok(specialityList);

}

[HttpGet]

public IHttpActionResult GetById(int specialityId)

{

var speciality = ecaseEntity.specialities.First(x => x.id == specialityId);

return Ok(speciality);

}

[HttpGet]

public IHttpActionResult GetByIds([FromUri] int[] specialityId)

{

var specialitiesList = ecaseEntity.specialities.Where(x => specialityId.Contains(x.id));

return Ok(specialitiesList);

}

[HttpPost]

public IHttpActionResult Create(SpecialitiesDto specialitiesDto)

{

speciality speciality = new speciality

{

name = specialitiesDto.name,

createdon = DateTime.Now,

updatedon = DateTime.Now

};

ecaseEntity.specialities.Add(speciality);

ecaseEntity.SaveChanges();

return Ok(speciality);

}

[HttpPost]

public IHttpActionResult Delete(int specialityId)

{

var speciality = ecaseEntity.specialities.First(x => x.id == specialityId);

ecaseEntity.specialities.Remove(speciality);

ecaseEntity.SaveChanges();

return Ok(speciality);

}

[HttpPost]

public IHttpActionResult Update(SpecialitiesDto specialitiesDto)

{

var speciality = ecaseEntity.specialities.First(x => x.id == specialitiesDto.id);

speciality.name = specialitiesDto.name;

speciality.updatedon = DateTime.Now;

ecaseEntity.SaveChanges();

return Ok(speciality);

}

}

}

**CasesController.cs**

using EcaseMain.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Threading.Tasks;

using System.Web.Http;

using System.Web.Http.Cors;

using SendGrid;

using SendGrid.Helpers.Mail;

namespace EcaseMain.Controllers

{

[EnableCors(origins: "\*", headers: "\*", methods: "\*")]

public class CasesController : ApiController

{

ecaseEntities ecaseEntity = null;

public CasesController()

{

ecaseEntity = new ecaseEntities();

}

[HttpPost]

public IHttpActionResult Create(CasesDto casesDto)

{

@case caseObj = new @case

{

title = casesDto.title,

description = casesDto.description,

clientid = casesDto.clientid,

specialityid = casesDto.specialityid,

approvedlawyerid = casesDto.approvedlawyerid

};

ecaseEntity.cases.Add(caseObj);

ecaseEntity.SaveChanges();

return Ok(caseObj);

}

[HttpPost]

public IHttpActionResult HireLawyer(int caseId, int lawyerId, string emailId)

{

var caseRecord = ecaseEntity.cases.First(x => x.id == caseId);

caseRecord.approvedlawyerid = lawyerId;

ecaseEntity.SaveChanges();

var caseTitle = caseRecord.title;

var clientRecord = ecaseEntity.registrations.First(x => x.id == caseRecord.clientid);

var clientName = clientRecord.firstname + " " + clientRecord.lastname;

SendMessageAsync(emailId,caseTitle, clientName).Wait();

return Ok(caseRecord);

}

private async Task SendMessageAsync(string toEmailId, string caseTitle, string clientName)

{

Response response = null;

try

{

var client = new SendGridClient("SG.U1pTRAPJT3GvU2uffDFVKA.9Lt--zn\_a3ivLs\_U-M0MU80bxXR\_SZlCEToyMlh6lok");

var from = new EmailAddress("baljeetcodeapp@gmail.com");

var to = new EmailAddress(toEmailId);

var msg = MailHelper.CreateSingleEmail(from, to,

"Hired", "Hired", "You are hired by "+ clientName +

"on the case : "+ caseTitle);

response = await client.SendEmailAsync(msg).ConfigureAwait(false);

}

catch (Exception e)

{

var statusCode = response?.StatusCode.ToString() ?? "None";

}

}

}

}

**QuotesController.cs**

using EcaseMain.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Threading.Tasks;

using System.Web.Http;

using System.Web.Http.Cors;

using SendGrid;

using SendGrid.Helpers.Mail;

namespace EcaseMain.Controllers

{

[EnableCors(origins: "\*", headers: "\*", methods: "\*")]

public class QuotesController : ApiController

{

ecaseEntities ecaseEntity = null;

public QuotesController()

{

ecaseEntity = new ecaseEntities();

}

[HttpGet]

public IHttpActionResult GetQuotedCaseList(int clientId)

{

registration record = null;

try

{

List<CasesQuotesDto> casesQuotesDtoList = new List<CasesQuotesDto>();

var caseList = ecaseEntity.cases.Where(x => x.clientid == clientId).ToList();

foreach(var cas in caseList)

{

CasesQuotesDto casesQuotesDto = new CasesQuotesDto();

casesQuotesDto.title = cas.title;

casesQuotesDto.description = cas.description;

casesQuotesDto.id = cas.id;

casesQuotesDto.approvedLaywerId = Convert.ToInt32(cas.approvedlawyerid);

if(cas.approvedlawyerid != 0)

{

var lawrecord = ecaseEntity.registrations.First(x => x.id == cas.approvedlawyerid);

casesQuotesDto.LawyerName = lawrecord.firstname + " " + lawrecord.lastname;

}

casesQuotesDto.specialityName = ecaseEntity.specialities.First(x => x.id == cas.specialityid).name;

List<Quoted> quotedList = new List<Quoted>();

var QuoteList = ecaseEntity.quotes.Where(x => x.caseid == cas.id).ToList();

foreach(var quote in QuoteList)

{

Quoted quoted = new Quoted();

quoted.id = quote.id;

quoted.description = quote.description;

quoted.duration = quote.duration;

quoted.price = quote.price;

quoted.lawyerid = Convert.ToInt32(quote.lawyerid);

var lawyer = ecaseEntity.registrations.First(x => x.id == quote.lawyerid);

quoted.lawyername = lawyer.firstname + " " + lawyer.lastname;

quoted.emailId = lawyer.email;

quotedList.Add(quoted);

}

casesQuotesDto.Quoteds = quotedList;

casesQuotesDtoList.Add(casesQuotesDto);

}

return Ok(casesQuotesDtoList);

}

catch (Exception ex)

{

record = null;

return Ok(record);

}

}

[HttpPost]

public IHttpActionResult Create(QuotesDto quotesDto)

{

quote quote = new quote

{

description = quotesDto.description,

price = quotesDto.price,

duration = quotesDto.duration,

lawyerid = quotesDto.lawyerId,

caseid = quotesDto.caseId

};

ecaseEntity.quotes.Add(quote);

ecaseEntity.SaveChanges();

return Ok(quote);

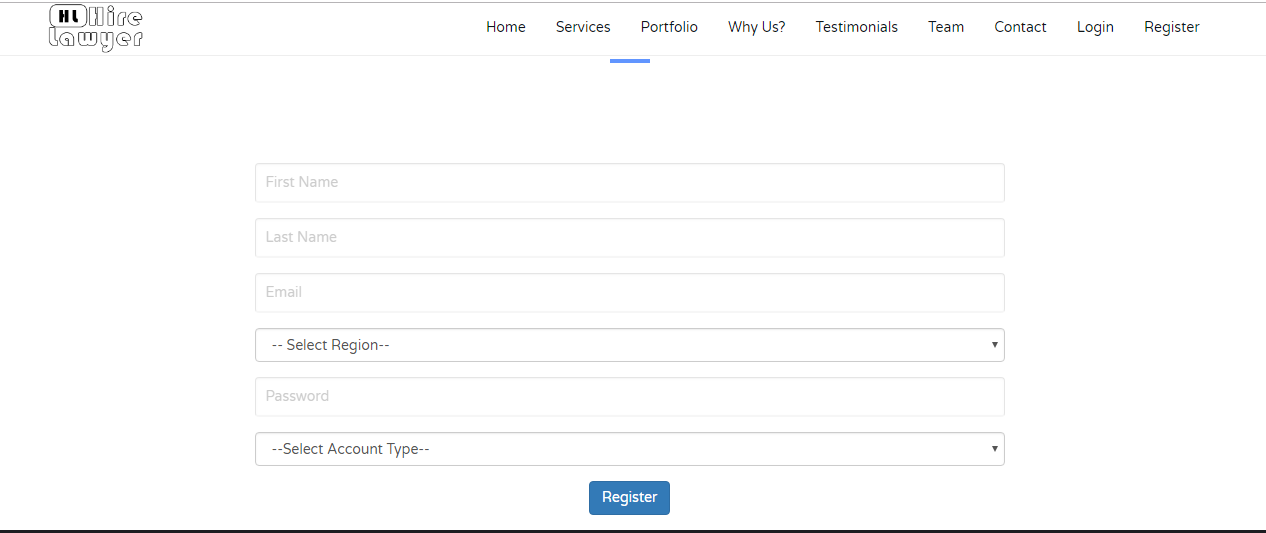
}

}

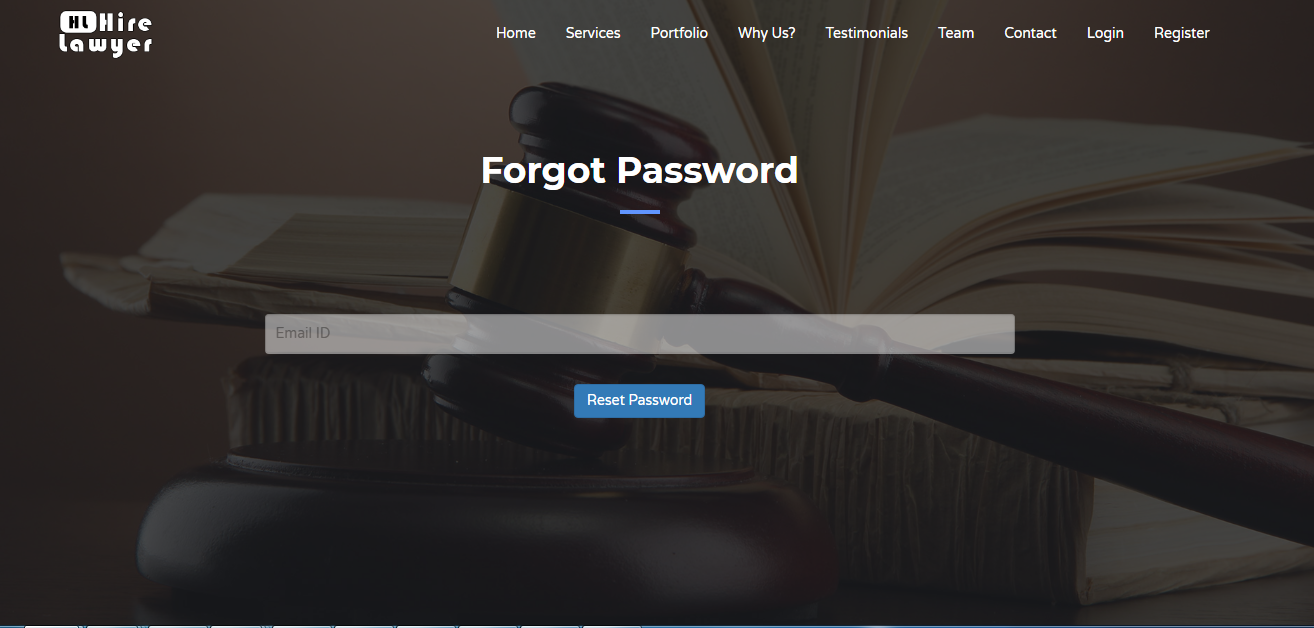
}

**Testing Document**

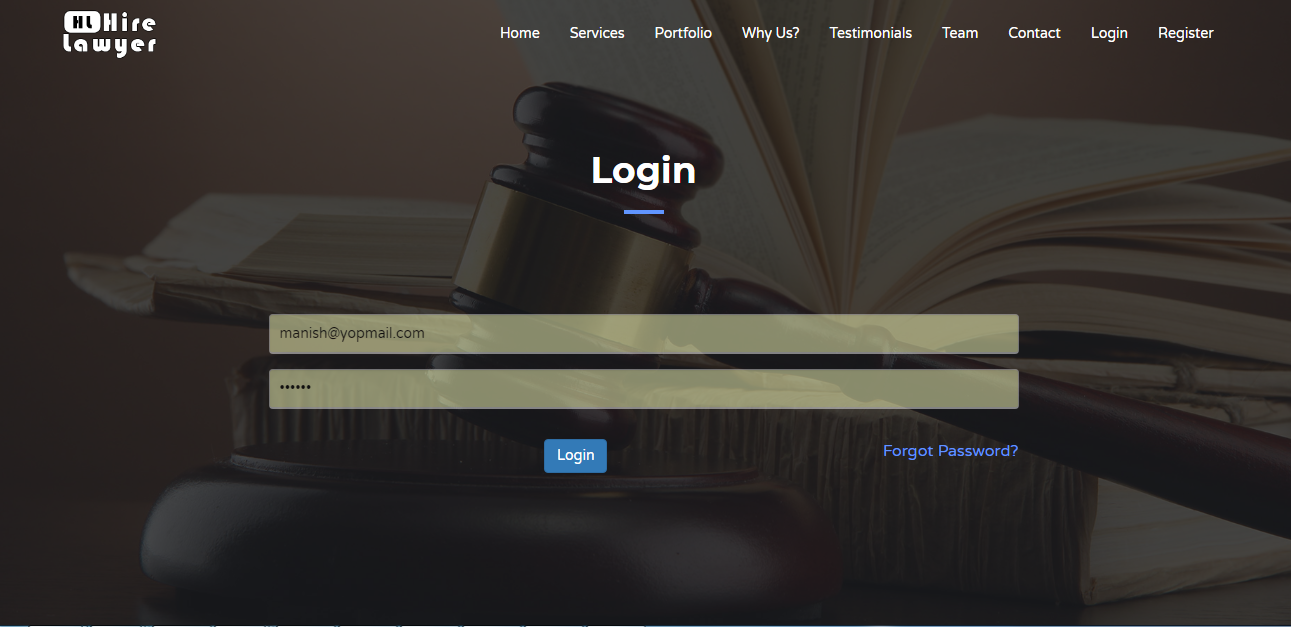
REGISTRATION SCREEN

****

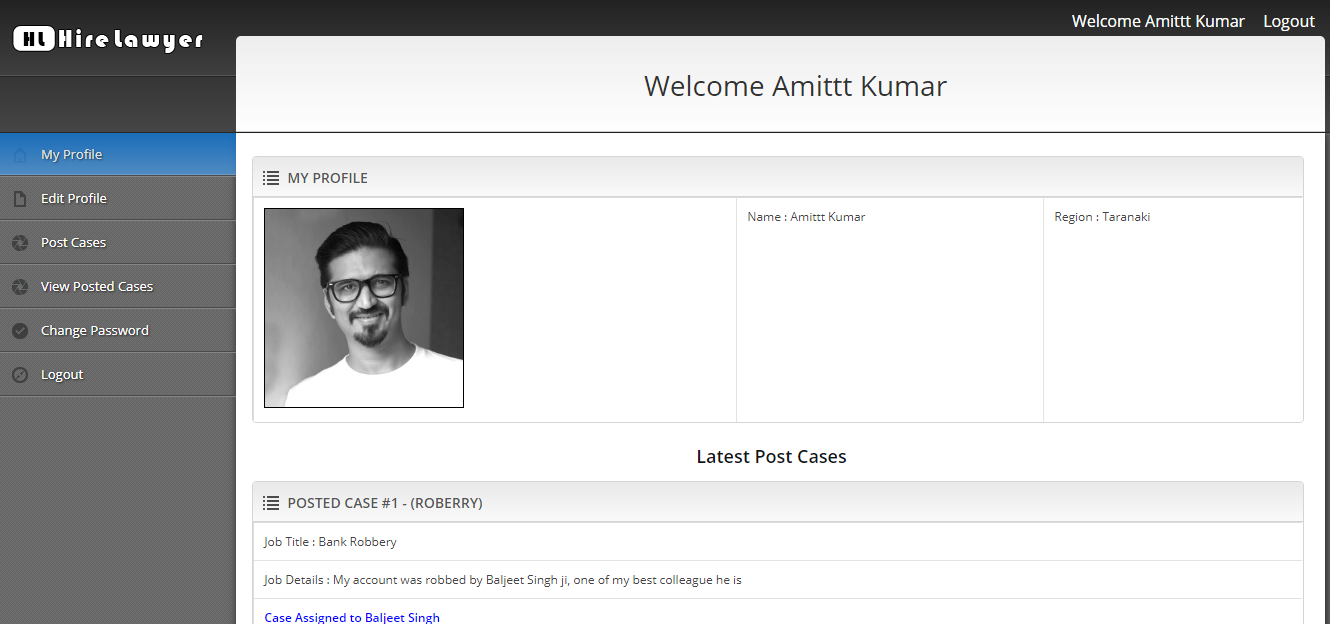
FORGOT PASSWORD SCREEN



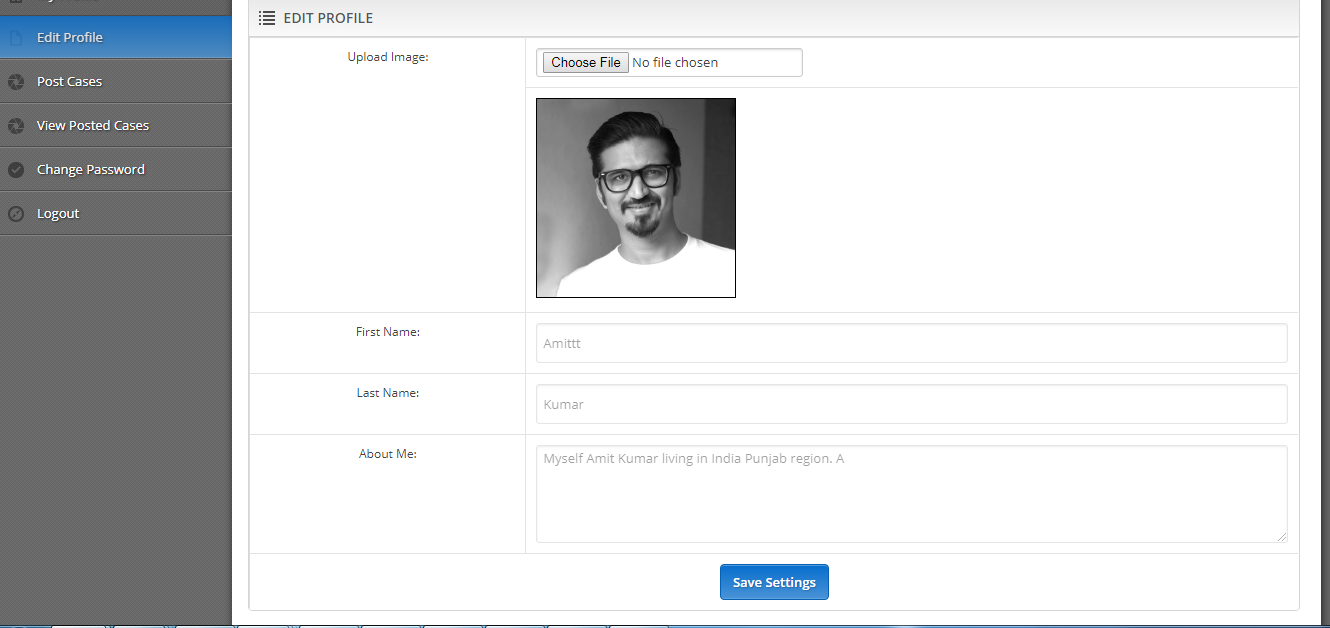
LOGIN SCREEN



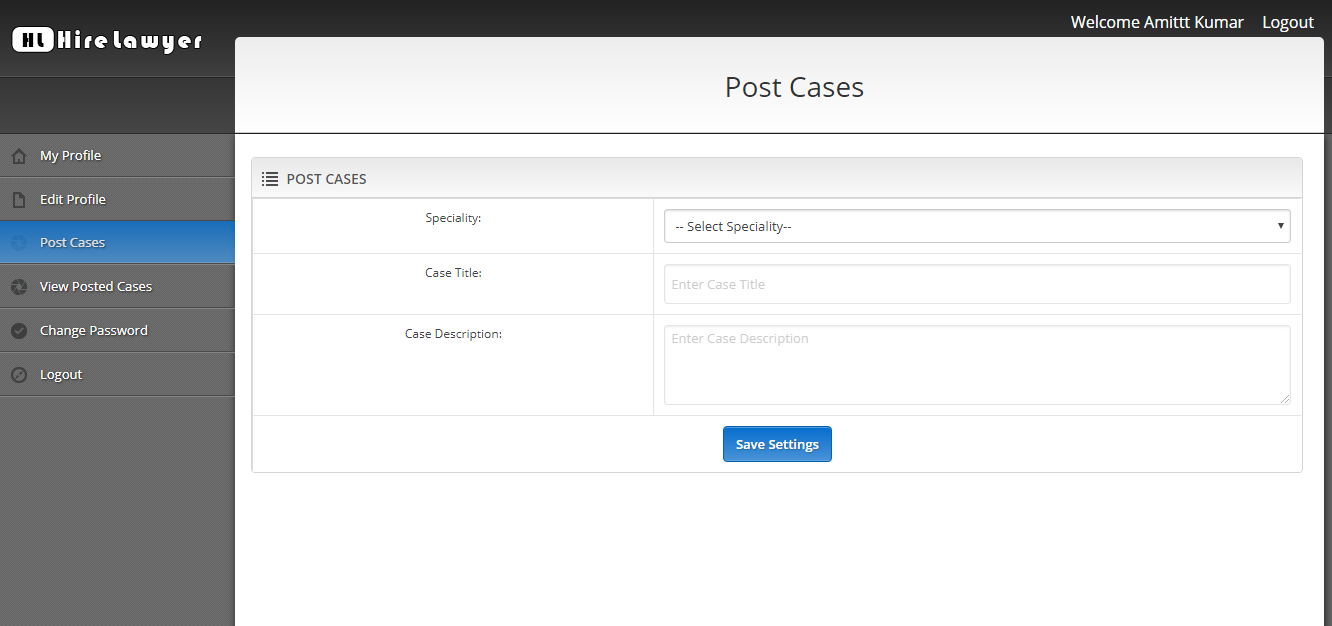
CLIENT PROFILE SCREEN



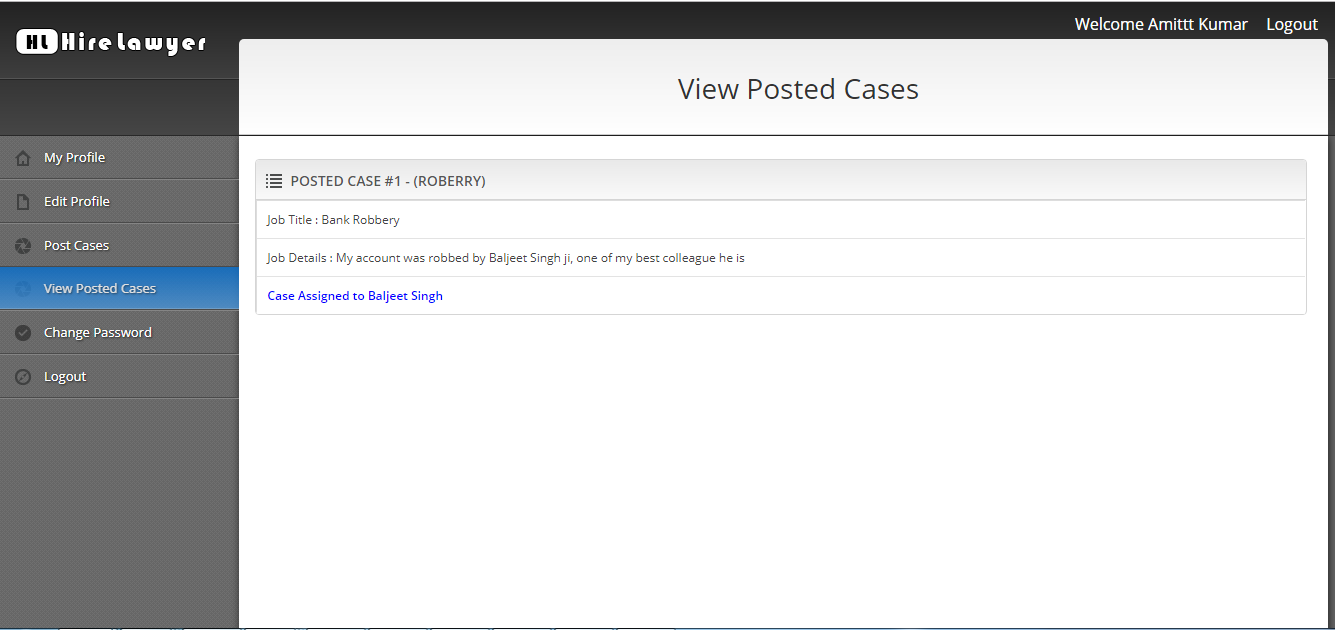
EDIT CLIENT PROFILE SCREEN



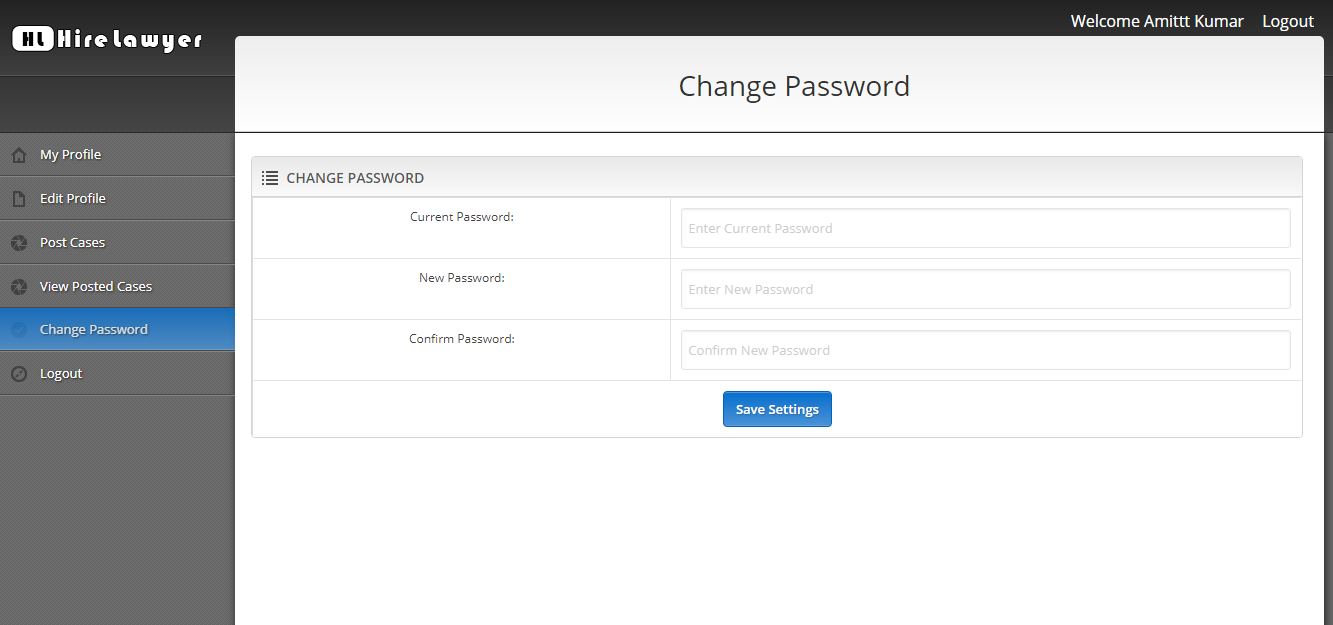
CLIENT POST CASES SCREEN



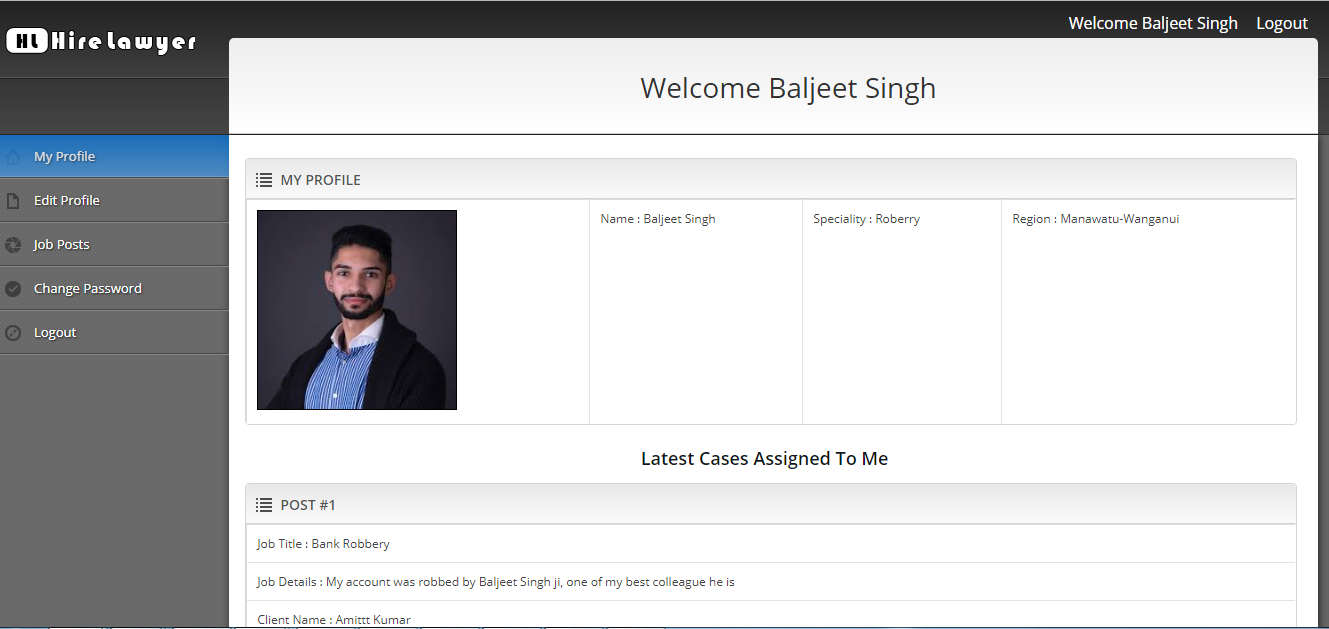
CLIENT VIEW POSTED CASES SCREEN



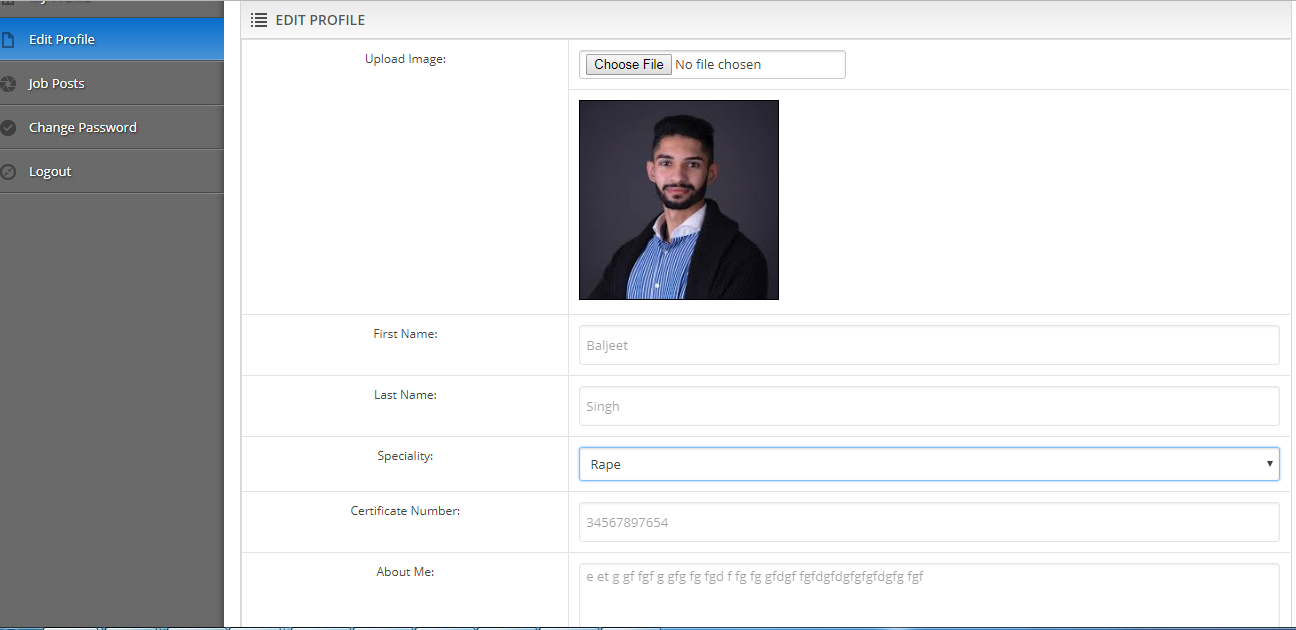
CLIENT CHANGE PASSWORD SCREEN



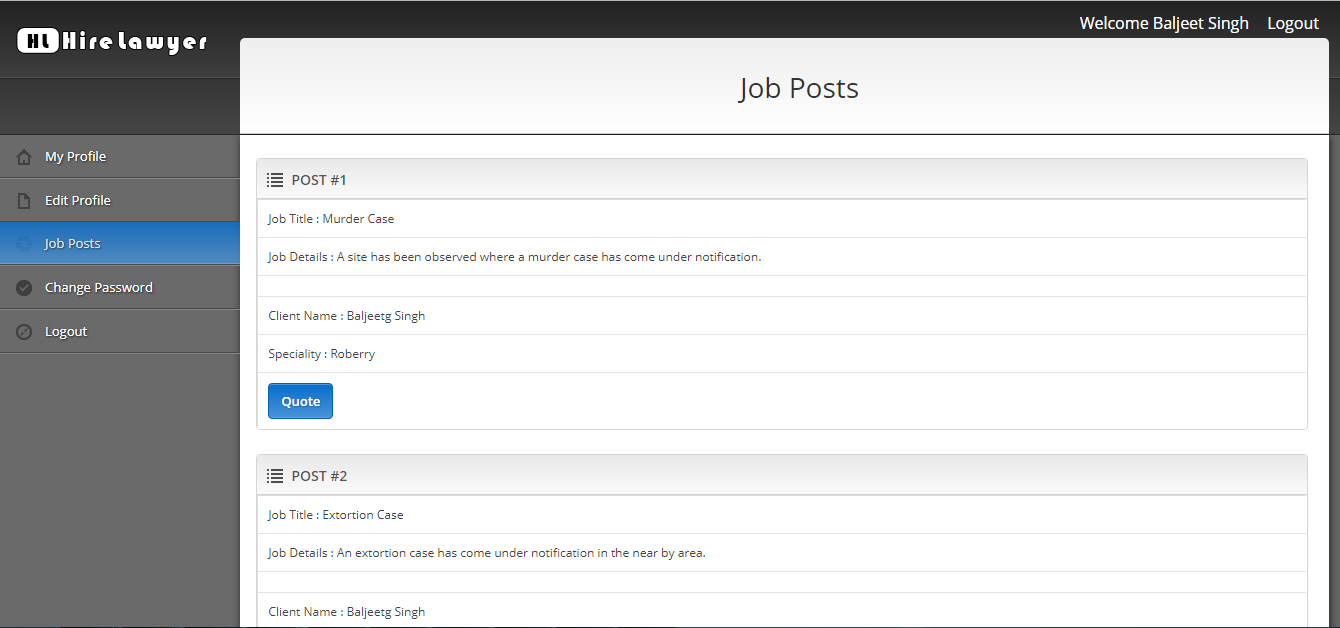
LAWYER PROFILE SCREEN



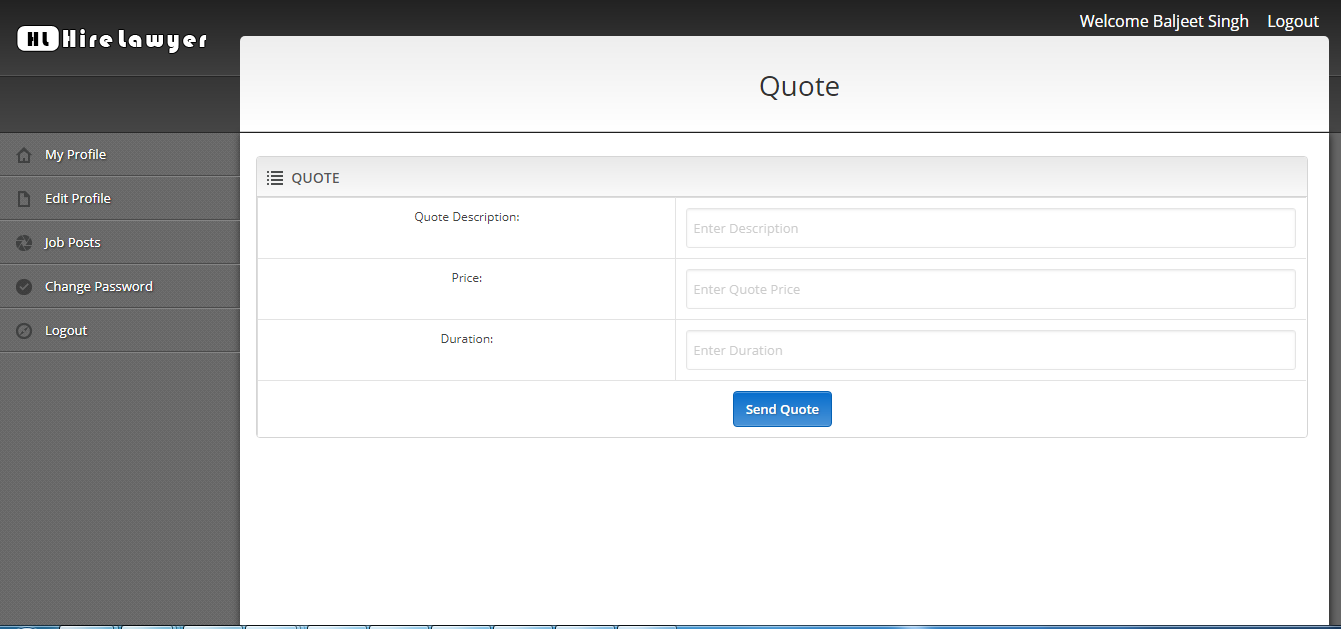
LAWYER EDIT PROFILE SCREEN



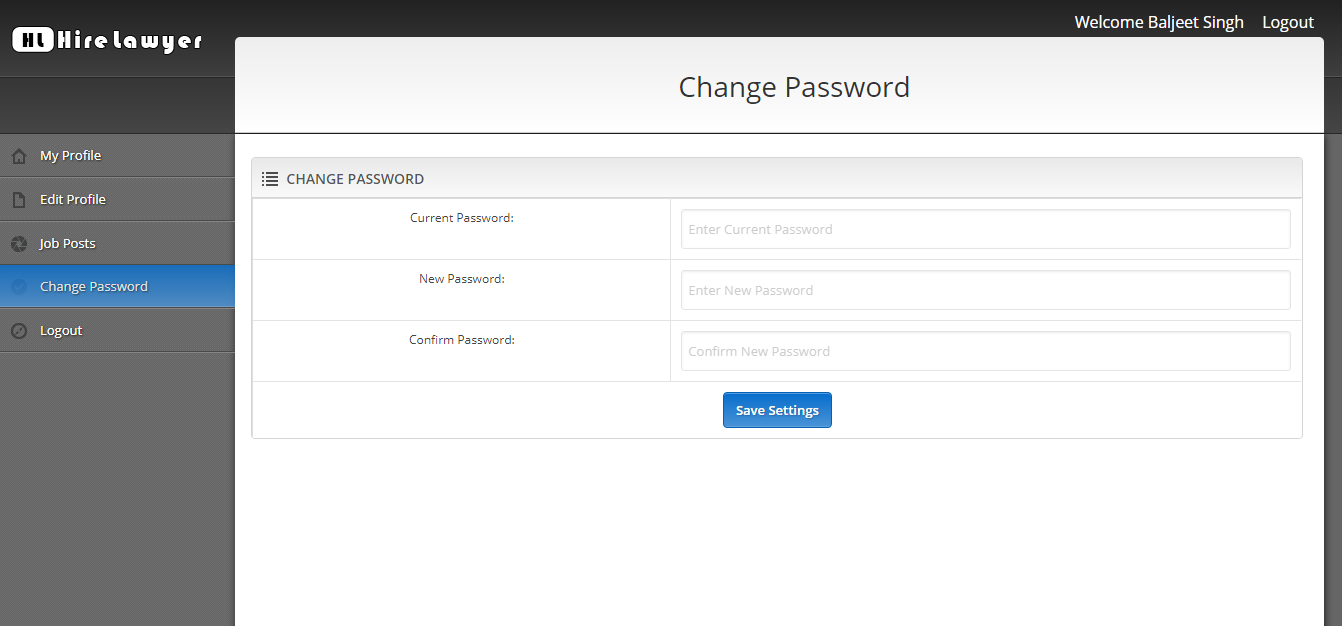
LAWYER VIEWED CLIENT CASES LIST SCREEN



LAWYER QUOTE ON CASE SCREEN



LAWYER CHANGE PASSWORD SCREEN



**Project review document**

**Ques: What was the App that your group created?**

**Ans: HireLawyer** app created by our group. It was just like online hiring of the lawyer for the cases that was posted by the client according to the lawyer specialties.

**Ques: What did you learn about how the Agile and Scrum methodologies operate?**

**Ans:**

1. Individual Reflection.  For 10 to 15 minutes, each individual works silently to write down the things that they have learned in the appropriate category on the handout.  Everyone should try to get at least a couple things into each of the four categories, but more is welcome.
2. Sharing with the Group.  Systematically going around the group and getting people to read from what they have written.  This is another 10 to 15 minutes.  This stage should not get bogged down in discussion, but brief clarifying questions should be welcome.
3. Identifying Important Learning.  The group now has open discussion to decide on a small number of things it considers the most important that it has learned.  This could be based on popularity, but impact, depth, or uniqueness might also be factors in considering importance.  These are the items that get written down on the flip-chart.  This is usually the longest part of the retrospective and can take up to 30 minutes.

**Ques: What are the other Software development methodologies?**

**Ans:** The Agile Software Development Method Uses an Iterative and Team-Based Approach. With the traditional method, the details of the entire project have been visualized and defined before the project starts. In contrast, the agile methodology allows for more flexibility in that changes can more easily be made even after the project starts. It is best employed if the scope of the project cannot be clearly defined in advance. This also means that making unplanned software development changes with the traditional method is costlier than with agile.

**In practice how effective did you find this methodology?**

Make it a continuous process to identify and eliminate waste, encouraging lean processes and practices. Automate as many manual processes as you can. Create incentives wisely to boost morale and reward positive behavior recognizing in a balanced approach both teams and individuals for on-time results, high quality, creativity, collaboration, leadership, and process improvements. Create opportunities for growth with rotating team roles.

**Ques : What did you find was the strength of it?**

**Ans:** One of the best rules of Scrum is also one of the simplest: a clear separation of scope (Product Backlog) from work (Sprint Backlog).

The Product Backlog is scope and is owned by the product owner

To clarify, the Product Backlog is the domain of the Product Owner. The Product Owner oversees and rules the Product Backlog, completely. That’s the job of the product owner: to own the vision, roadmap and scope of the product. To choose what goes in it, what won’t, and what the priorities of scope are. The development team have no say about what goes in or out of the product backlog. Their job is to not to define scope, but to turn scope into work.

The Sprint Backlog is work and is owned by the developers

The Sprint Backlog is the collection of work for the current sprint. It is looked after by the development team (that’s the mini team within the scrum team, made up of “developers” only). The Product Owner has no say over what goes into the Sprint Backlog. The Scrum Master has no say over what goes into the Sprint Backlog. It is up to the developers to choose a certain amount of scope. They decide how much scope and which scope, read the Scrum Guide if you don’t believe me. They can then put it into a sprint and work on it. Once it goes into the sprint, it has turned from being “scope” to being “work” and is now in the domain of the “developers”.

**Ques: What were its weaknesses ?**

**Ans:** Agile Scrum is one of the leading causes of scope creep because unless there is a definite end date, the project management stakeholders will be tempted to keep demanding that new functionality be delivered.

If a task is not well defined, estimating project costs and time will not be accurate. In such a case, the task can be spread over several sprints.

If the team members are not committed, the project will either never complete or fail.

It is good for small, fast moving projects as it works well only with small team.

This methodology needs experienced team members only. If the team consists of people who are novices, the project cannot be completed in time.

Scrum works well for project management when the Scrum Master trusts the team they are managing. If they practice too strict control over the team members, it can be extremely frustrating for them, leading to demoralization and the failure of the project.

If any of the team members leave during a development it can have a huge inverse effect on the project development

Project quality manager is hard to implement and quantify unless the test team are able to conduct regression testing after each sprint.

**Ques: What were the best features of the process ?**

**Ans:** Due to sprints and constant feedback, it becomes easier to cope with the changes.

Daily meetings make it possible to measure individual productivity. This leads to the improvement in the productivity of each of the team members.

Issues are identified well in advance through the daily meetings and hence can be resolved in speedily

It is easier to deliver a quality product in a scheduled time.

**Ques: How did they appeal to you ?**

**Ans:** Daily Standup (Stand-Up Meetings)

Daily stand-up meetings are also called Daily Scrum meetings. The Scrum sessions are held daily by the team so they can share pertinent information with each other. These meetings are designed to keep all team members equally informed and updated on the status of the project. The key to each meeting is brevity.

During the daily scrum meetings, each member should answer these three questions:

Q: What did I do yesterday?

Q: What will I do today?

Q: What problems are hindering my progress?

User Stories

A user story is a brief description of the function wanted by the end user. There are three elements to the user story. They are:

A written description, planning the story (usually written on a card)

A conversation about the story to gain better understanding

A series of tests confirming the story.

The stories are written from the end user’s perspective and uses language that they understand. Stories act as currency between developers and customers; both parties clearly understand them.

**Ques: What was the worst feature of the process ?**

**Ans:** The worst thing about estimates is that they push a company in the direction of doing work that’s estimable. This causes programmers to favor the low-yield, easy stuff that the business doesn’t actually need (even if bumbling middle managers think otherwise) but that is “safe”. Anything that’s actually worth doing has a non-zero chance of failure and too many unknown unknowns for estimates to be useful. Agile’s focus on short-term business value ends up pushing people away from the kinds of work that the company actually needs, in favor of each programmer’s real-time reputation management.

**Ques: What did you enjoy about this style of programming ?**

**Ans:** Group discussion for the problem we faced on the project sprint and get the idea from other to accomplish the task.

**Ques: What are the downsides for you of group work ?**

**Ans:** As we follow Agile / Scrum methodology, sometimes we need to wait for the other team member to complete his task, so that the next one will be discussed and distributed to make it complete.

**Ques: How did this session change your opinion of working in industry?**

**Ans:** My opinion is that a team work is the best strategy to complete any of the task provided.

**Ques: What form of Source Control did you use?**

**Ans:** Github

**Ques: What were its strengths ?**

**Ans:** Always we have backup of the project work done by team members. And anyone can show log to view their team member changes and move to the next step of the action that is required to perform.

**Ques: What were its weaknesses ?**

**Ans:** We are not asking the actually weakness of the github, but the major concern of it is the conflicts case that we need to understand how to resolve it properly.

**Ques: How effective as a source control did you find it ?**

**Ans:** It always be the good source control for the specially Agile scrum methodology.

**Ques: If you had to use it again what would you change ?**

**Ans:** Next time we use the git with multiple branches. So every team member push their code in their branch after complete his sprint and the other team member can easily pull from that branch to get his changes.

**Ques: Do you feel that this has been a worthwhile experiment?**

**Ans:** Yes, it worthwhile.

**Why** : we came to know how to work with team and understand the actually project workflow scenario. Be active in the group meeting and discussion. Point raises for the hurdles that you faced on the work progress.