LITHAN

Design, Develop, Implement & Document Community Portal Website

Module Project

Module: Web Development Foundations

Course: NICF Advanced Certificate in Software & Applications (Development & Deployment)

Start Date	**: 12/09/2022**
End Date	**:23/09/2022**
Submission Date	**:02/11/2022**

Learner Name	**: Chathushi Jayarathna**
Enrollment ID	**:**
Presentation Date	**:**

Document History

LITHAN

Version Number	Effective Date of release	Summary of Included Changes	Author
1	4th March 2016	First Edition	Satya CVS
2	08th Sep 2017	Changed for Module 5	Shrinivas K R
3	20th Jul 2018	Changed for Module 5 RQF	Shrinivas K R

Contents

LITHAN

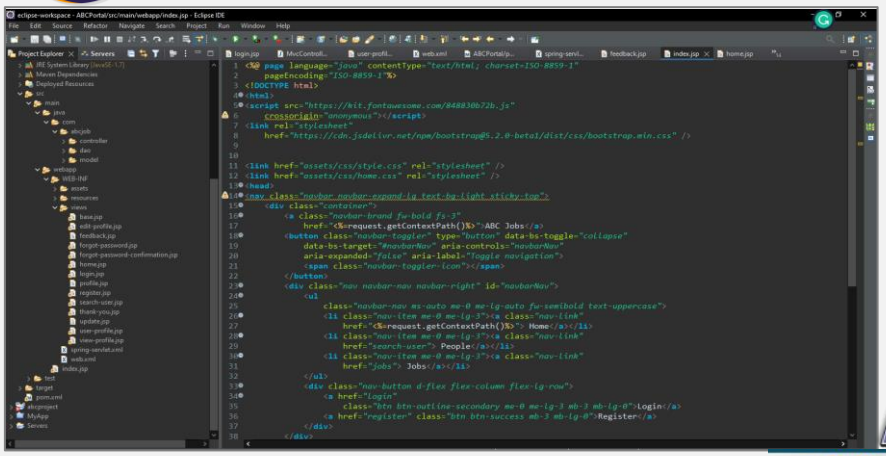
S. No.	Description
01	Development Tools
02	Business Process
03	Application Block Diagram
04	Applications
05	Models, Controllers & Classes Developed
06	UI Design
07	Includes
08	Software Development Methodology
09	Project Plan
10	Application Screen Shots
11	Project Milestones & Tasks
12	Milestone Feedback & Action Taken
13	Modifications Made On Feedback
14	Project Results
15	Proposed Improvements

1. Development Tools

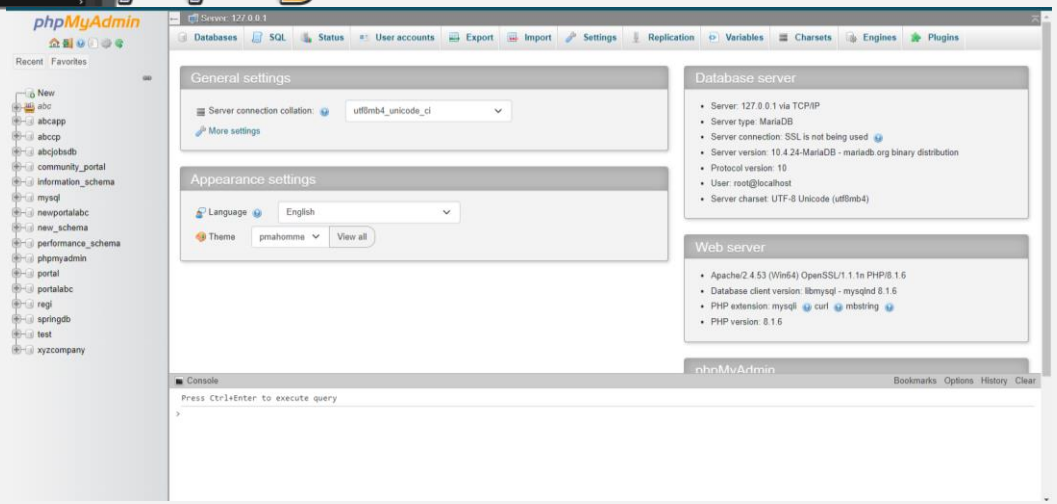
LITHAN

❑ Development Tools Screen captures

Eclipse IDE



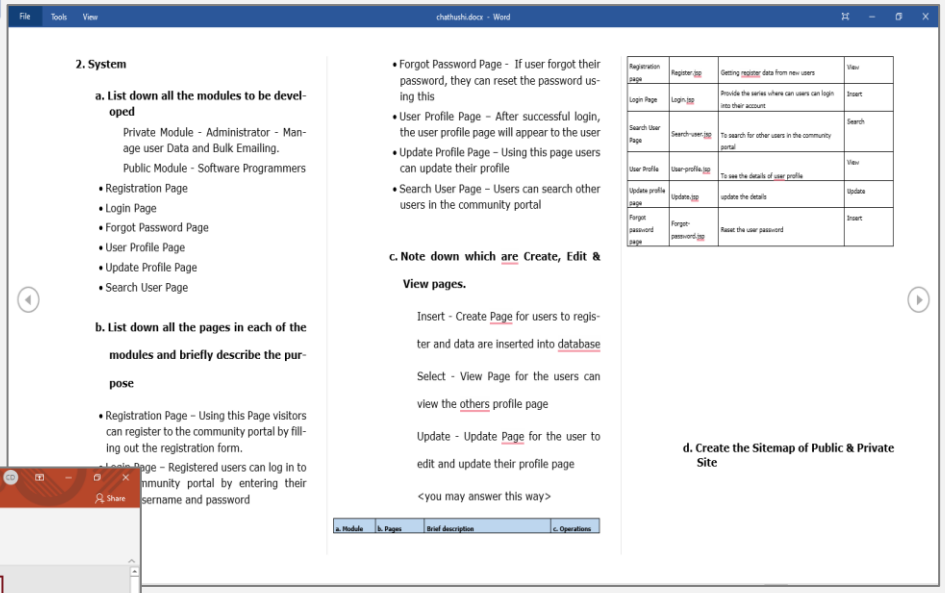
phpMyAdmin



1. Development Tools

LITHAN

Microsoft Word



2. System

a. List down all the modules to be developed

Private Module - Administrator - Manage user Data and Bulk Emailing.

Public Module - Software Programmers

- Registration Page
- Login Page
- Forgot Password Page
- User Profile Page
- Update Profile Page
- Search User Page

b. List down all the pages in each of the modules and briefly describe the purpose

- Registration Page – Using this Page visitors can register to the community portal by filling out the registration form.
- Login Page – Registered users can log in to community portal by entering their username and password
- Forgot Password Page - If user forgot their password, they can reset the password using this
- User Profile Page – After successful login, the user profile page will appear to the user
- Update Profile Page – Using this page users can update their profile
- Search User Page – Users can search other users in the community portal

Registration page	Register.jsp	Getting register data from new users	View
Login Page	Login.jsp	Provide the series where can users can login into their account	Insert
Search User Page	Search-user.jsp	To search for other users in the community portal	Search
User Profile	User-profile.jsp	To see the details of user profile	View
Update profile page	Update.jsp	update the details	Update
Forgot password page	Forgot-password.jsp	Reset the user password	Insert

c. Note down which are Create, Edit & View pages.

Insert - Create Page for users to register and data are inserted into database

Select - View Page for the users can view the others profile page

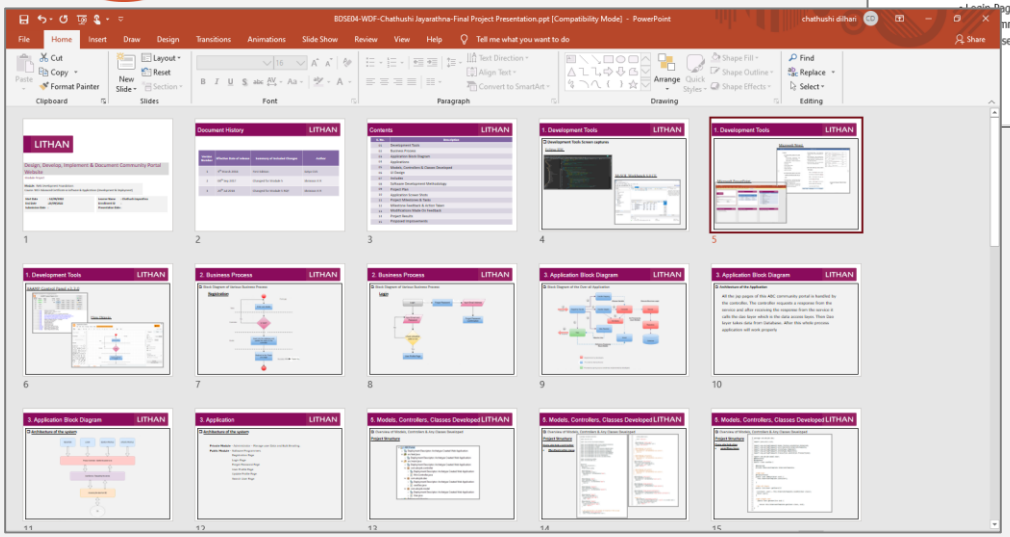
Update - Update Page for the user to edit and update their profile page

<you may answer this way>

a. Module	b. Pages	Brief description	c. Operations

d. Create the Sitemap of Public & Private Site

Microsoft PowerPoint



1. Development Tools

LITHAN

XAAMP Control Panel v3.3.0

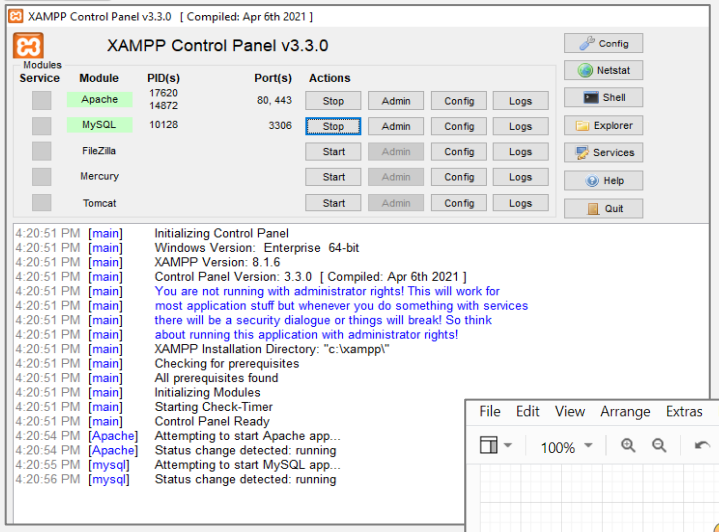
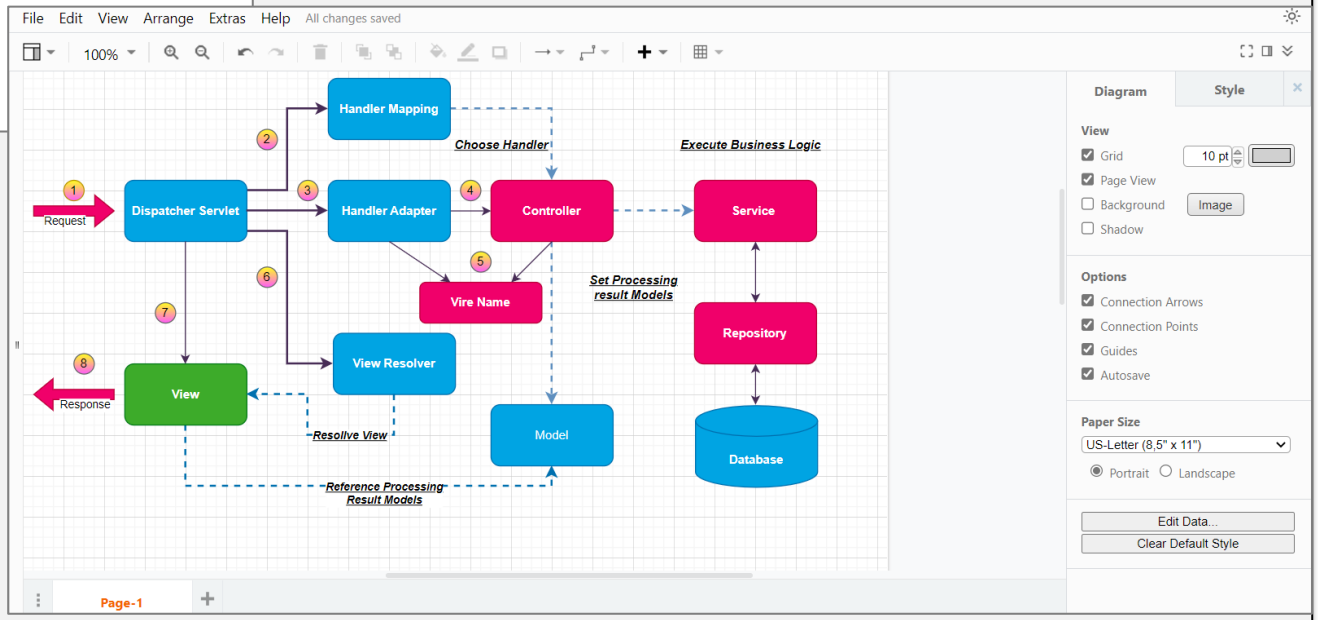


Diagram.Net

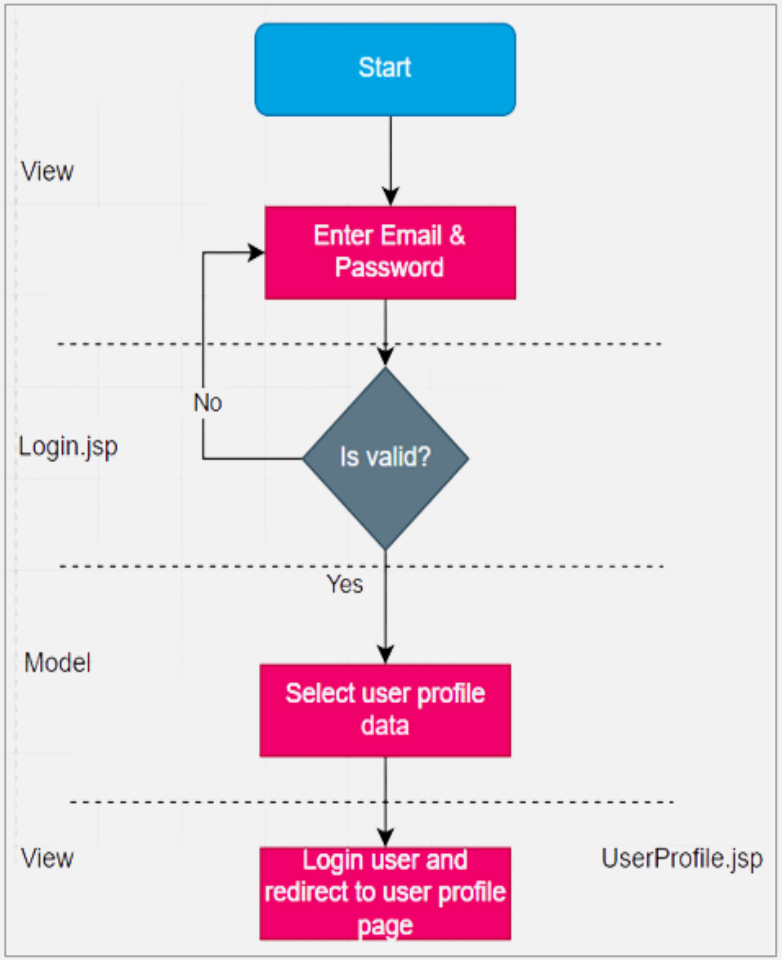


2. Business Process

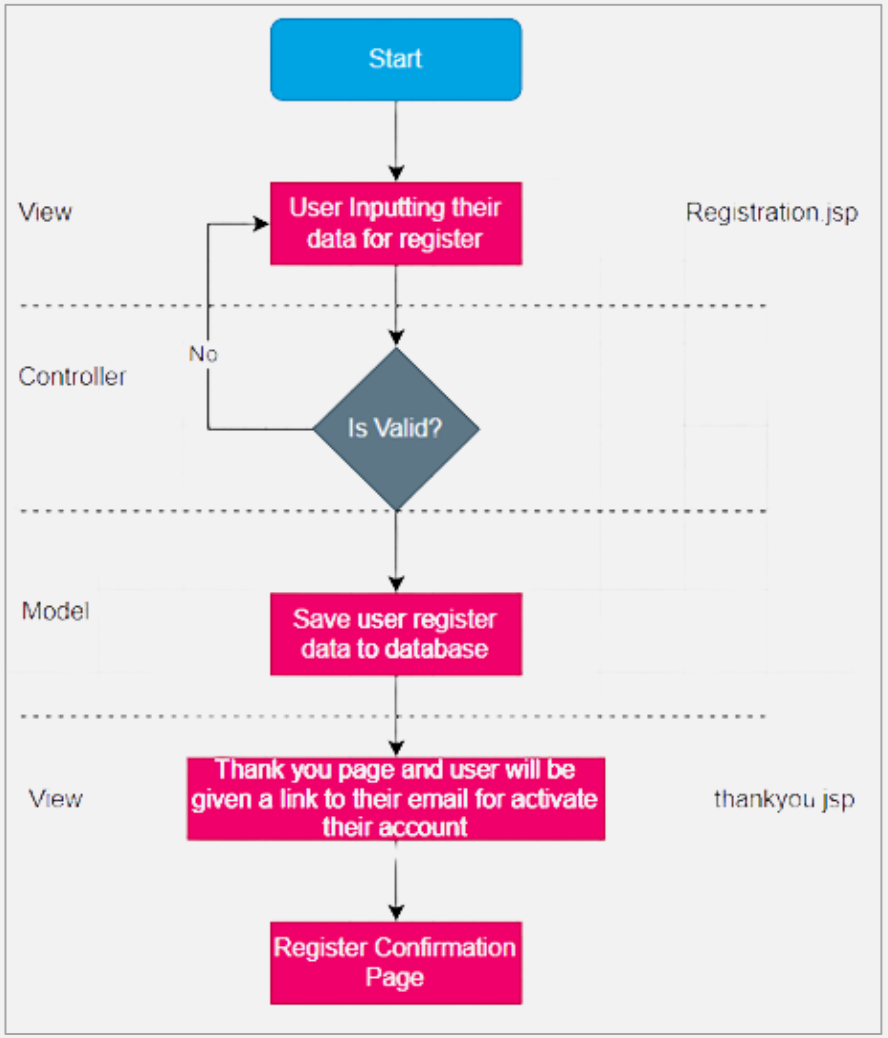
LITHAN

- Block Diagram of Various Business Process

Login



Registration

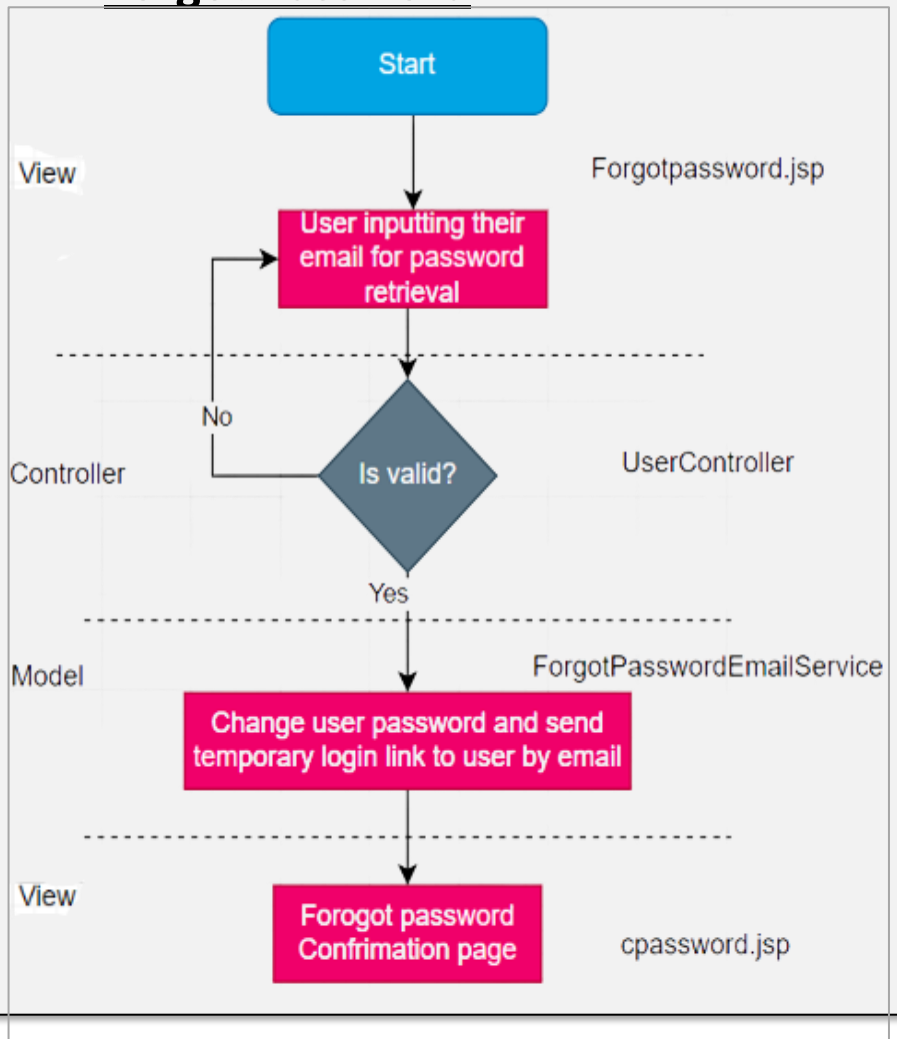


2. Business Process

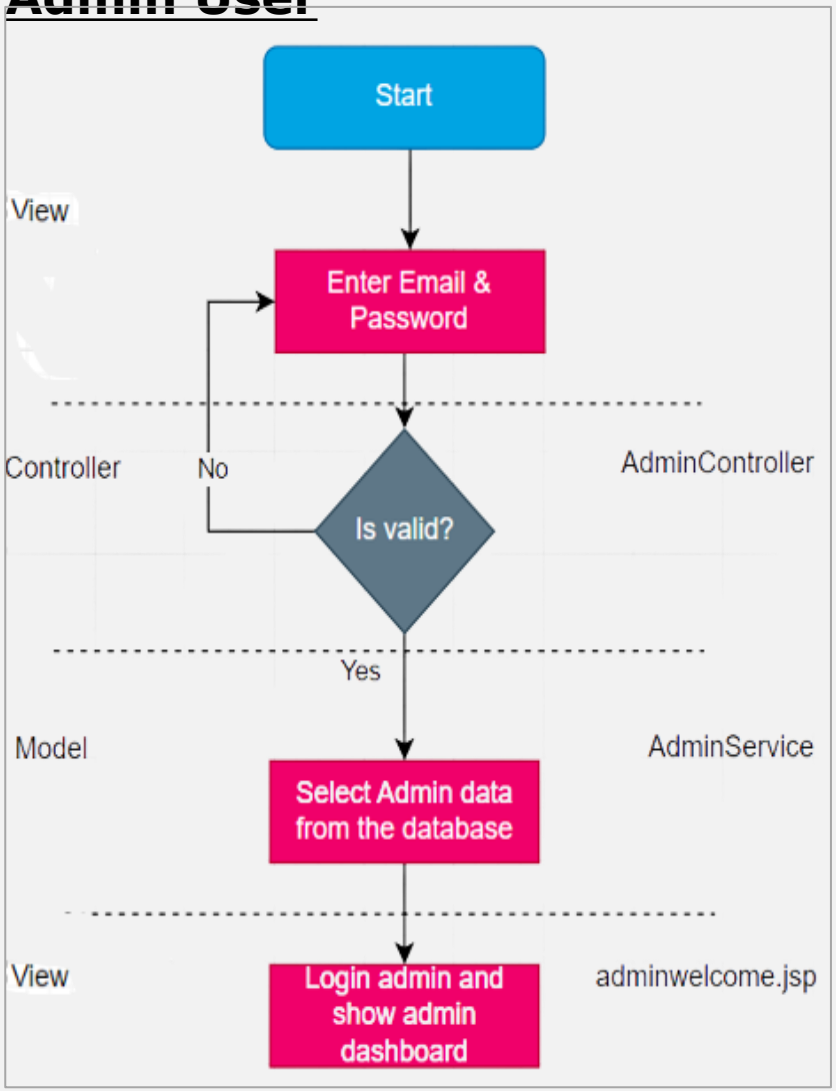
LITHAN

- Block Diagram of Various Business Process

Forgot Password



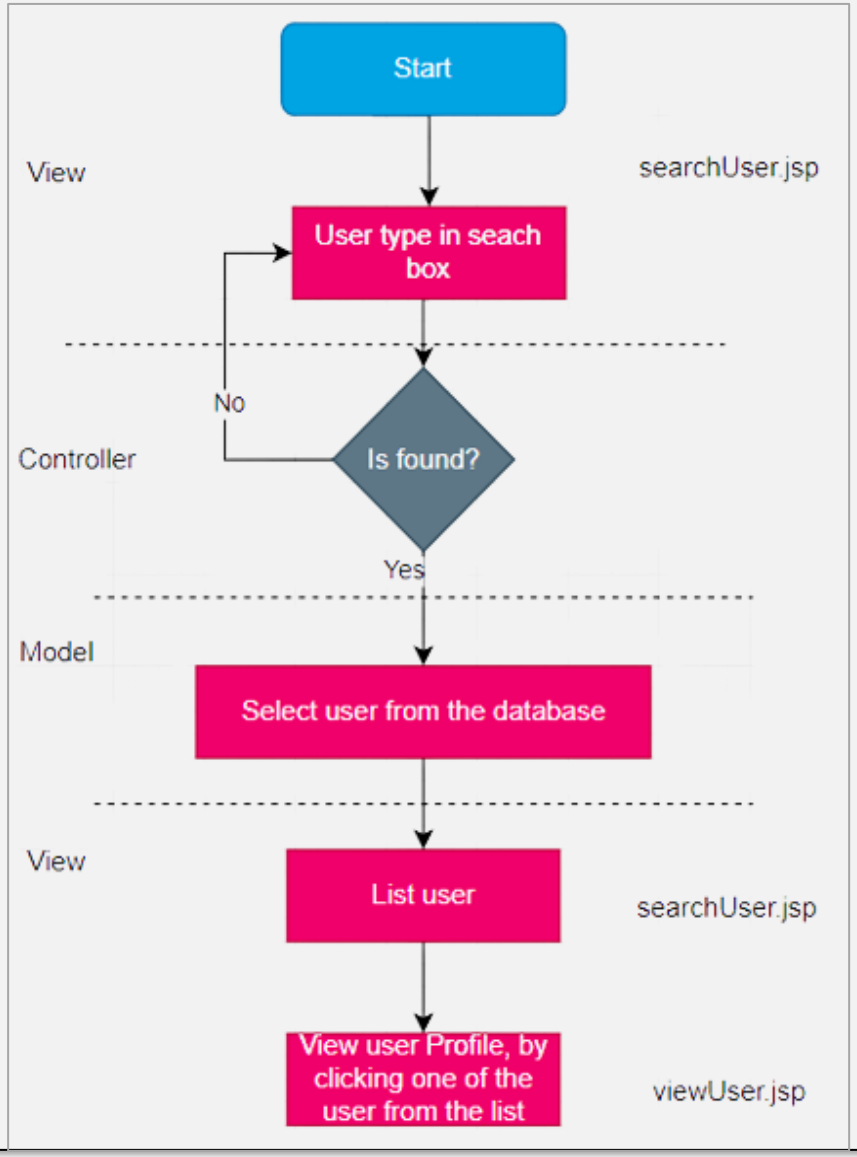
Admin User



2. Business Process

LITHAN

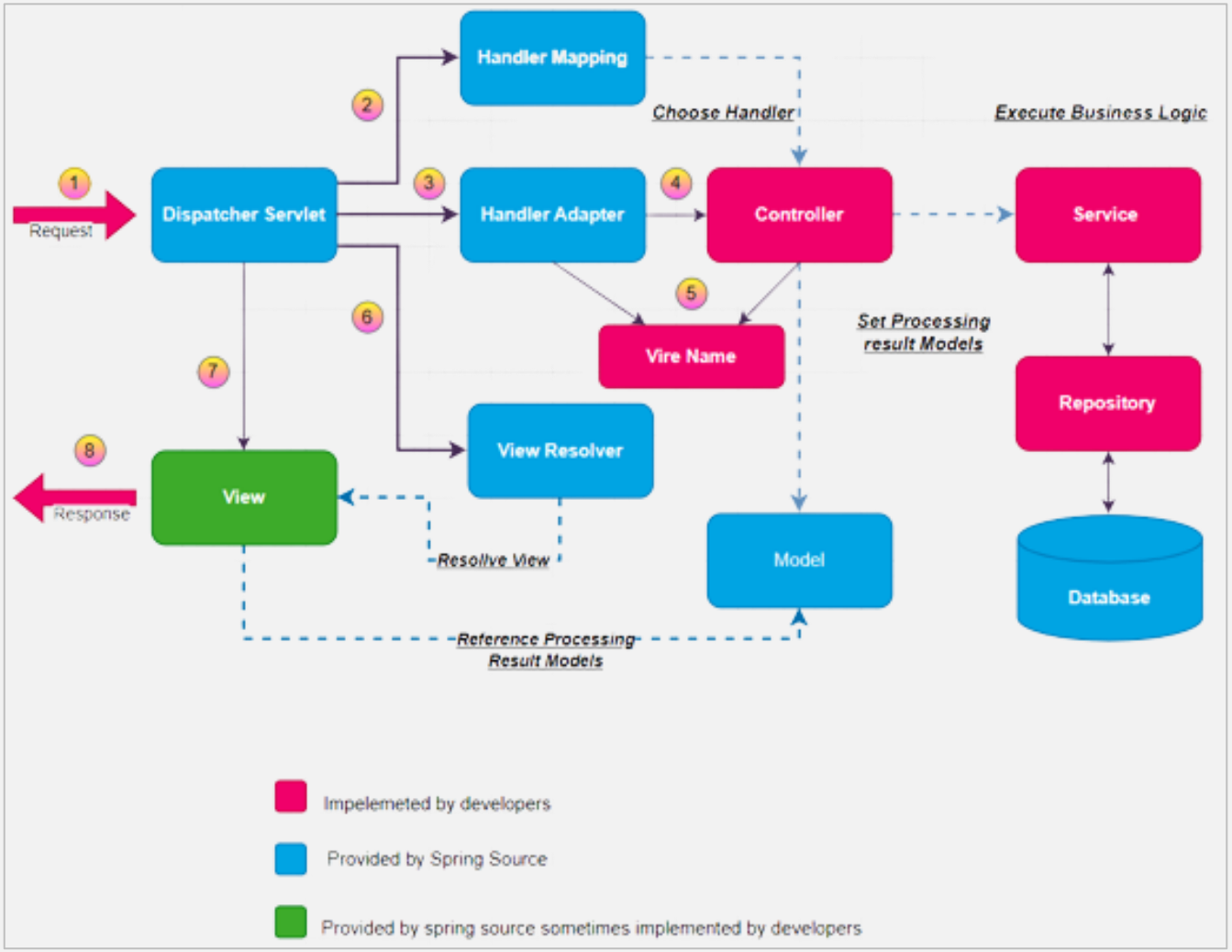
Search User



3. Application Block Diagram

LITHAN

- Block Diagram of the Over all Application



3. Application Block Diagram

LITHAN

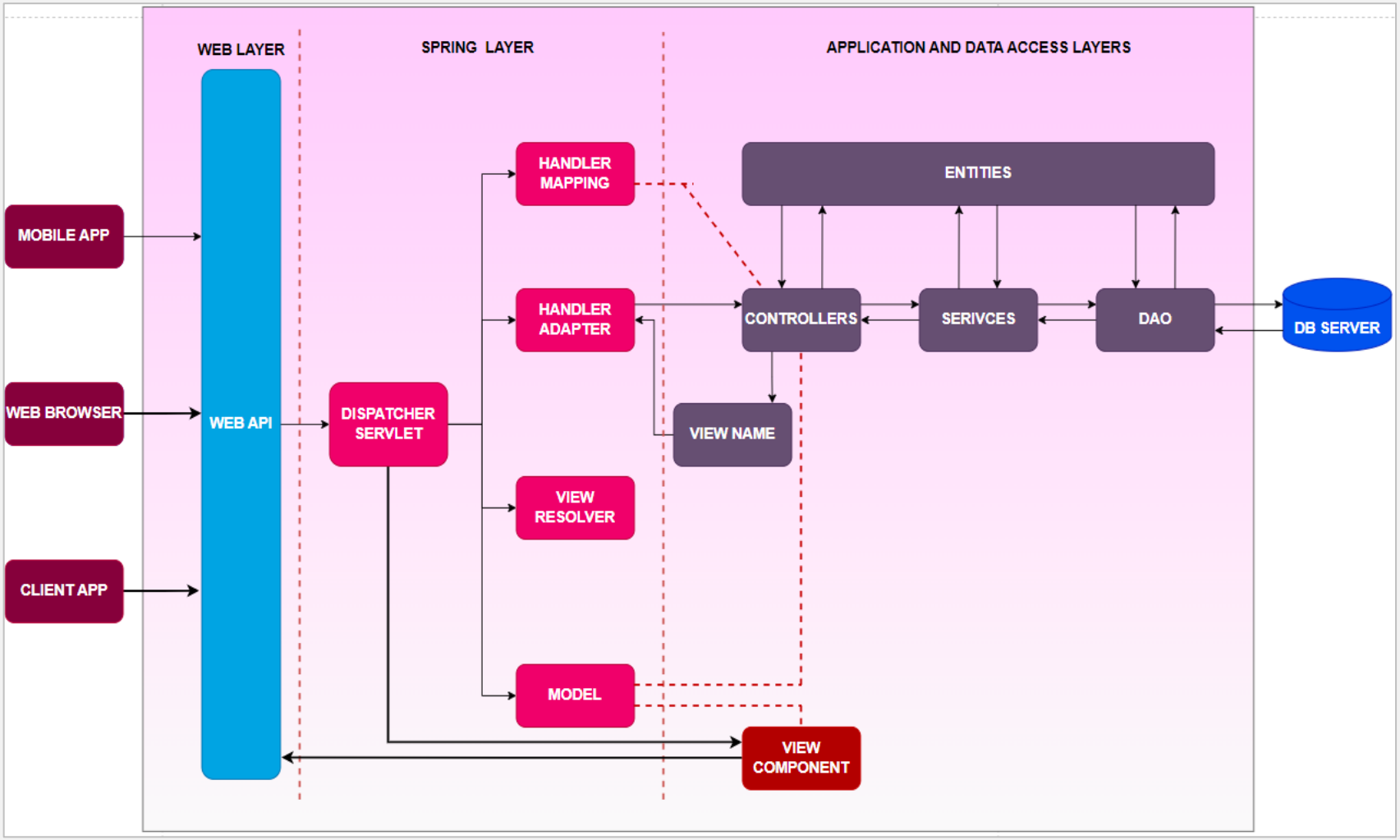
- **Architecture of the Application**

All the jsp pages of this ABC community portal is handled by the controller. The controller requests a response from the service and after receiving the response from the service it calls the dao layer which is the data access layer. Then Dao layer takes data from Database. After this whole process application will work properly. After successfully importing the ABC JOB portal on the System the running server will Go to the dispatcher servlet for view resolver. The dispatcher servlet will locate server to view resolver Where Web-Inf called the JSP pages and css pages and also the Annotation class called all the annotated classes and behaving/presenting the multipart view of JSP pages according to the classes.

3. Application Block Diagram

LITHAN

❑ Architecture of the system



3. Application

LITHAN

❑ Architecture of the system

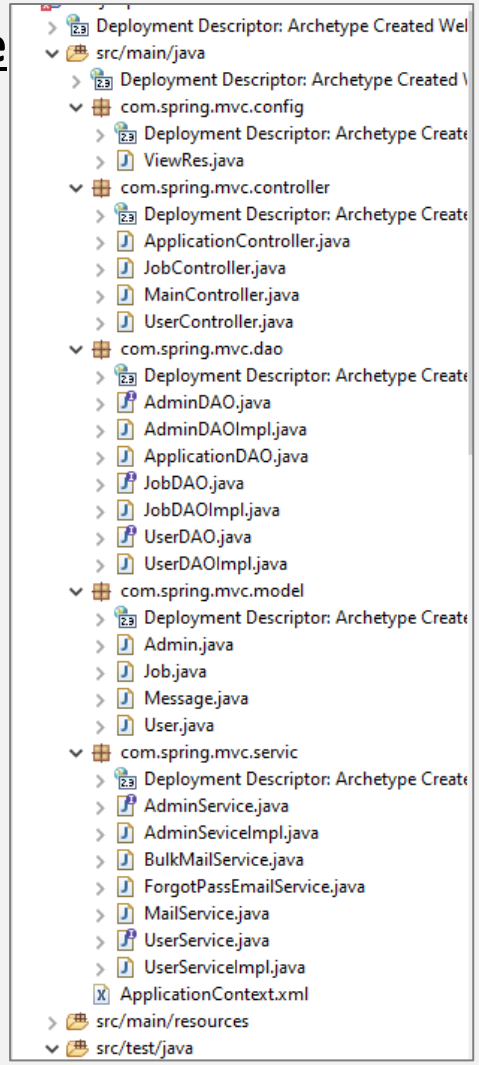
1. **Private Module** - Administrator - Manage user Data and Bulk Emailing.
2. **Public Module** - Software Programmers
    - Registration Page
    - Login Page
    - Forgot Password Page
    - Forgot Password Reset
    - User Profile Page
    - User Home Page
    - Edit User Profile
    - Search User Page

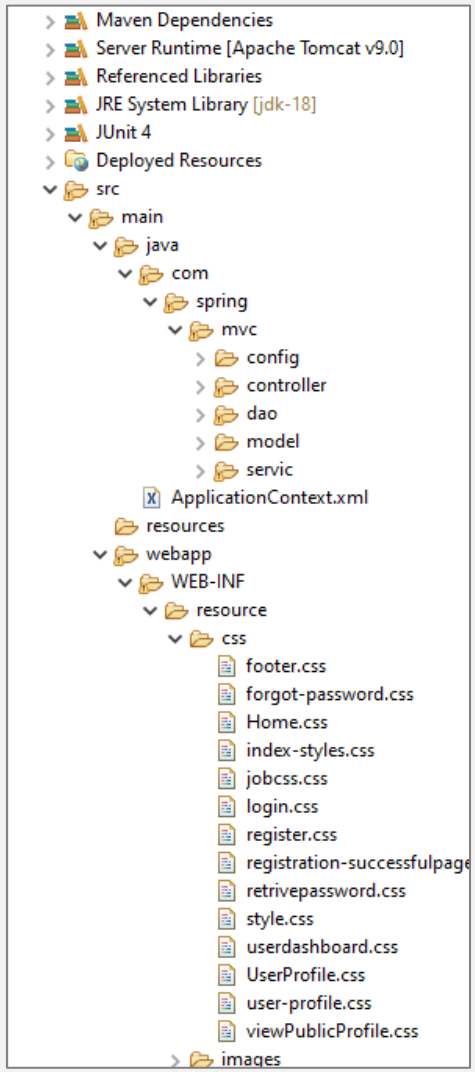
5. Models, Controllers, Classes Developed

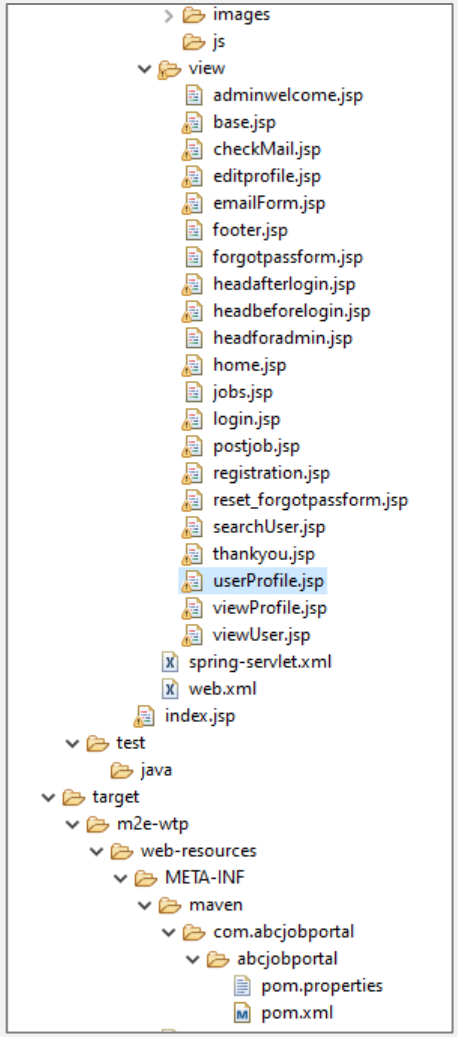
LITHAN

- Overview of Models, Controllers & Any Classes Developed

Project Structure







5. Models, Controllers, Classes Developed

LITHAN

❑ Overview of Models, Controllers & Any Classes Developed

Project Structure

1. com.spring.mvc.config

- **ViewRes (To connect resources files "images, css,js")**

```
1  package com.spring.mvc.config;
2
3  import org.springframework.context.annotation.Bean;
4  import org.springframework.context.annotation.ComponentScan;
5  import org.springframework.context.annotation.Configuration;
6  import org.springframework.web.servlet.config.annotation.EnableWebMvc;
7  import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;
8  import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
9  import org.springframework.web.servlet.view.InternalResourceViewResolver;
10
11 @EnableWebMvc
12 @Configuration
13 @ComponentScan("abcjobportal")
14 public class ViewRes implements WebMvcConfigurer {
15
16   @Bean(name = "viewResolver")
17   public InternalResourceViewResolver getViewResolver() {
18       InternalResourceViewResolver viewResolver = new InternalResourceViewResolver();
19       viewResolver.setPrefix("/WEB-INF/views/");
20       viewResolver.setSuffix(".jsp");
21       return viewResolver;
22
23 }
24
25   @Override
26   public void addResourceHandlers(ResourceHandlerRegistry registry)
27   {
28        registry.addResourceHandler("/resource/**")
29          .addResourceLocations("/resource/css")
30          .addResourceLocations("/resource/images")
31          .addResourceLocations("/resource/js")
32         ;
33
34   }
35 }
36
```

Used for displaying the information to the user in a specific format.

5. Models, Controllers, Classes Developed

LITHAN

2. com.spring.mvc.controller

Used to mark a class as a web request handler.

- ✓ **ApplicationContoller.java**

```java
package com.spring.mvc.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;

import com.spring.mvc.dao.ApplicationDAO;

import com.spring.mvc.model.User;

@Controller
public class ApplicationController {

  private ApplicationContext context;
  @Autowired
  private ApplicationDAO applicationDAO;

  @RequestMapping(value = "/resetPass", method = RequestMethod.POST)
  public String edit(@ModelAttribute("bean") User user, Model model) {
    context = new ClassPathXmlApplicationContext("ApplicationContext.xml");
    ApplicationDAO obj = context.getBean("dao", ApplicationDAO.class);
    List list = obj.resetPassword(user);
    model.addAttribute("data", list);
    model.addAttribute("reset", " Reset Password Successful");
    return "login";

  }

  @RequestMapping("/edit/{userId}")
  public String updaeUserForm(@PathVariable("userId") Long uId, Model model) {
    User user = this.applicationDAO.getUser(uId);
    model.addAttribute("user", user);
    return "editprofile";
  }

  // for update profile
  @RequestMapping(value = "edit/update", method = RequestMethod.POST)
  public String updateUser(@RequestParam("id") Long id, @RequestParam("name") String name,
      @RequestParam("email") String email, @RequestParam("contact") String contact,
      @RequestParam("city") String city) {
    User user = new User();
    user.setId(id);
    user.setName(name);
    user.setEmail(email);
    user.setCity(city);
    user.setContact(contact);

    applicationDAO.updateprofile(user);
    return "editprofile";// will come edit profile page

  }

  @RequestMapping("/find")
  public String search(Model model) {
    context = new ClassPathXmlApplicationContext("ApplicationContext.xml");
    User user = context.getBean("argon", User.class);
    model.addAttribute("bean", user);
    return "search";
  }

  @RequestMapping(value = "/searchview", method = RequestMethod.GET)
  public String searchView(@ModelAttribute("bean") User user, Model model) {
    return "search";
  }

  @RequestMapping(value = "/searchUser", method = RequestMethod.POST)
  public String searchUsrs(@ModelAttribute("bean") User user, Model model) {
    context = new ClassPathXmlApplicationContext("ApplicationContext.xml");
    ApplicationDAO obj = context.getBean("dao", ApplicationDAO.class);

    List list = obj.search(user);
    if (!list.isEmpty()) {
      model.addAttribute("data", list);
    } else {

      model.addAttribute("msg", "User Not found");
    }
    return "searchUser";
  }

  @RequestMapping("searchUser")
  public String searchUser(Model model, @ModelAttribute("bean") User user) {
    context = new ClassPathXmlApplicationContext("ApplicationContext.xml");

    ApplicationDAO obj = context.getBean("dao", ApplicationDAO.class);
    List<User> list = obj.display();

    model.addAttribute("data", list);

    return "searchUser";

  }
  @RequestMapping("viewUser")
  public String viewUser(Model model, @ModelAttribute("bean") User user) {
    context = new ClassPathXmlApplicationContext("ApplicationContext.xml");

    ApplicationDAO obj = context.getBean("dao", ApplicationDAO.class);
    List<User> list = obj.display();

    model.addAttribute("data", list);

    return "viewUser";

  }

  @RequestMapping("/viewProfile/{userId}")
  public String userProfilesearch(@PathVariable("userId") Long userId, Model model) {

    User user = this.applicationDAO.getUser(userId);
    model.addAttribute("user", user);
    return "viewProfile";

  }

  @RequestMapping("admindash")
  public String admindasdisplay(Model model, @ModelAttribute("bean") User user) {
    context = new ClassPathXmlApplicationContext("ApplicationContext.xml");

    ApplicationDAO obj = context.getBean("dao", ApplicationDAO.class);
    List<User> list = obj.display();

    model.addAttribute("data", list);

    return "admindash";

  }

  @RequestMapping("/findadmin")
  public String adminsearch(Model model) {
    context = new ClassPathXmlApplicationContext("ApplicationContext.xml");
    User user = context.getBean("argon", User.class);
    model.addAttribute("bean", user);
    return "admindash";
  }

  @RequestMapping(value = "/adminsearchview", method = RequestMethod.GET)
  public String adminsearchView(@ModelAttribute("bean") User user, Model model) {
    return "admindash";
  }
  @RequestMapping(value = "/adminsearchUsrs", method = RequestMethod.POST)
  public String adminsearchUsrs(@ModelAttribute("bean") User user, Model model) {
    context = new ClassPathXmlApplicationContext("ApplicationContext.xml");
    ApplicationDAO obj = context.getBean("dao", ApplicationDAO.class);

    List list = obj.search(user);
    if (!list.isEmpty()) {
      model.addAttribute("data", list);
    } else {

      model.addAttribute("msg", "User Not found");
    }
    return "admindash";
  }

  @RequestMapping(value = "/editor", method = RequestMethod.POST)
  public String adminedit(@ModelAttribute("bean") User user, Model model) {
    context = new ClassPathXmlApplicationContext("ApplicationContext.xml");
    ApplicationDAO obj = context.getBean("dao",ApplicationDAO.class);
    List list = obj.update(user);
    model.addAttribute("data", list);
    model.addAttribute("msg", " Record update succesfullysuccessfully");
    return "admindash";

  }
  @RequestMapping(value = "/delete", method = RequestMethod.GET)
  public String adminedelete(@ModelAttribute("bean") User user, Model model) {
    context = new ClassPathXmlApplicationContext("ApplicationContext.xml");
    ApplicationDAO obj = context.getBean("dao",ApplicationDAO.class);
    List list = obj.delete(user);
    model.addAttribute("data", list);
    model.addAttribute("msg", " Record deleted succesfullysuccessfully");
    return "viewUser";
```

5. Models, Controllers, Classes Developed

LITHAN

MainController.java

The Controller in MVC architecture **handles any incoming URL request.** The Controller is a class, derived from the base class System

```java
package com.spring.mvc.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class MainController {

  @RequestMapping("/registration")
  public String reistration() {
    return "registration";

  }
  @RequestMapping("/thankyou")
  public String thankyou() {
    return "thankyou";

  }

  @RequestMapping("/login")
  public String login() {
    return "login";

  }
  @RequestMapping("/jobs")
  public String jobs() {
    return "jobs";

  }

  @RequestMapping("/adminwelcome")
  public String adminwelcome() {
    return "adminwelcome";

  }
  @RequestMapping("/postjob")
  public String postjob() {
    return "postjob";

  }
  @RequestMapping("/forgotpassform")
  public String forgotpassform() {
    return "forgotpassform";

  }
  @RequestMapping("/forgot")
  public String forgot() {
    return "forgot";

  }
  @RequestMapping("/home")
  public String home() {
    return "home";

  }
  @RequestMapping("/userProfile")
  public String userProfile() {
    return "userProfile";

  }
  @RequestMapping("/editprofile")
  public String editprofile() {
    return "editprofile";

  }
@RequestMapping("/jobDetails")
public String jobDetails() {
  return "jobDetails";

}
@RequestMapping("/viewProfile")
public String viewProfile() {
  return "viewProfile";

}

@RequestMapping("/emailForm")
public String emailForm() {
  return "emailForm";

}
  @RequestMapping("error")
  public String error() {
    return "error";

  }
  @RequestMapping("success")
  public String success() {
    return "success";

  }
}
```

5. Models, Controllers, Classes Developed

UserController.java

```java
package com.spring.mvc.controller;

import java.util.Map;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.servlet.ModelAndView;

import com.spring.mvc.model.Admin;
import com.spring.mvc.model.User;
import com.spring.mvc.servic.AdminService;
import com.spring.mvc.servic.ForgotPassEmailService;
import com.spring.mvc.servic.MailService;
import com.spring.mvc.servic.UserService;

@Controller
public class UserController {

  @Autowired//To call the userservice
  private UserService userService;
  @Autowired//To call the mail servic

  private MailService mailService;
  @Autowired
  private AdminService adminService;
  @Autowired
  private ForgotPassEmailService forgotPassEmailService;


  //to get value from registration form
  @GetMapping(value="/registration")
  public ModelAndView registration(Model model) {
    User user=new User();
    model.addAttribute("user",user);
    System.out.println(user);
    ModelAndView modelAndView=new ModelAndView("registration");

    return modelAndView;

  }
  @PostMapping("/registration")
  public String register(@ModelAttribute("user") User user, Model model,
HttpServletRequest request,
      String emailToRecipient, String userName,String EmailID) {
    emailToRecipient = request.getParameter("email");
    userName = request.getParameter("name");

    userService.registerUser(user);
    mailService.sendMail(emailToRecipient, userName); /// sending mail to the user
thathis registration is completed
    request.getSession().setAttribute("name", userName);
    request.getSession().setAttribute("email", emailToRecipient);

    model.addAttribute("success", "Registration is successful");

    return "thankyou";
  }
  @RequestMapping(value = "/login", method = RequestMethod.POST)
  public String login(@ModelAttribute("user") User user, Model model, HttpSession
session,
      @ModelAttribute("admin") Admin admin) {
    User user2 = userService.loginUser(user);// For user
    Admin user3 = adminService.adminlogin(admin);// For admin
    System.out.println("user2" + user2);
    System.out.println("user3" + user3);

    if (user2 != null) { // If user not null it will come welcome page for him
      System.out.println("hello");
      model.addAttribute("user", user2);
      session.setAttribute("user", user2);
      return "home";
    }

    if (user3 != null) { // If admin not null it will come adminwelcome page for him
      System.out.println("hello");
      model.addAttribute("admin", user3);
      session.setAttribute("admin", user3);
      return "adminwelcome";
    }

//If user could not get it will show error
    if (user2 == null) {
      System.out.println("on");
      model.addAttribute("error", "Invalid Credential");

    }

//If user information can not get it will show error
    if (user3 == null) {
      System.out.println("on");
      model.addAttribute("error", "Invalid Credential");

    }

    return "login";
  }
  //logout
  public String loginDisplay(Model m,HttpSession session) {
    User user=new User();
    if(session.getAttribute("user") !=null){
      session.invalidate();
      System.out.println("here");
      m.addAttribute("success","You have logout successfully");

    }
    m.addAttribute("user",user);
    return "login";
}



  //For checking email is it in database or not and send mail to the definite email
  //start
    @RequestMapping(value = "/emailConfirm", method = RequestMethod.POST)
    public String checkMail(@ModelAttribute("user") User user, Model model,
HttpSession session,
        @RequestParam(value = "email") String email, HttpServletRequest request) {
      User user4 = userService.checkMail(user);
      System.out.println(user4);
      if (user4 != null) {
        System.out.println("hello");
        model.addAttribute("user", user4);
        session.setAttribute("user", user4);
        forgotPassEmailService.resetMail(email); // sending mail to the user to reset
his/her pasword
        request.getSession().setAttribute("email", email);

        //return "checkMail";
        return "checkMail";

      }
      if (user4 == null) {
        System.out.println("on");
        model.addAttribute("error", "Invalid Credential");
      }

      return "forgotpassform";



  }

    @RequestMapping(value="/resetPassword/{email}")
    public String resetPassword(@PathVariable String email,Map<String,String> model)
    {
      //check if the email id is valid and registered with us.
      model.put("emailid", email);
      return "reset_forgotpassform";
    }


}
```

5. Models, Controllers, Classes Developed

LITHAN

3. com.spring.mvc.dao

✓ AdminDao.java

```
1  package com.spring.mvc.dao;
2
3  import com.spring.mvc.model.Admin;
4
5  public interface AdminDAO {
6
7      public Admin adminlogin(Admin admin);
8  }
```

DAO means data access object. It support in Spring is aimed at making it easy yo work with data access technologies like JDBS, Hibernate, JPA or JDO in a consistent way

✓ AdminDaoImpl.java

```
1  package com.spring.mvc.dao;
2
3  import javax.persistence.NoResultException;
12
13 @Repository
14 @Transactional
15 public class AdminDAOImpl implements AdminDAO{
16     @Autowired
17     private SessionFactory factory;
18
19     public Admin adminlogin(Admin admin) {
20
21     Session session=factory.getCurrentSession();
22     try {
23         Query<Admin> query=session.createQuery("FROM admin_table where email=:email and password=:password",Admin
24
25     query.setParameter("email",admin.getEmail());
26         query.setParameter("password",admin.getPassword());
27     admin=(Admin) query.getSingleResult();
28 //      query.setMaxResults(1).uniqueResult();
29
30         return admin;
31 }catch(NoResultException e) {
32     return null;
33     }
34
35
36 }}
```

5. Models, Controllers, Classes Developed

LITHAN

✓ ApplicationDao.java

```
package com.spring.mvc.dao;

import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

import javax.transaction.Transactional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.dao.DataAccessException;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.PreparedStatementCallback;
import org.springframework.jdbc.core.ResultSetExtractor;
import org.springframework.orm.hibernate5.HibernateTemplate;
import org.springframework.stereotype.Repository;

import com.spring.mvc.model.Job;
import com.spring.mvc.model.User;

@Repository
public class ApplicationDAO {
  @Autowired
  private JdbcTemplate jdbcTemplate;

  @Autowired
  private HibernateTemplate hibernateTemplate;

//getting the data from jdbc template
  public JdbcTemplate getJdbcTemplate() {
    return jdbcTemplate;
  }

//setting the data from jdbc
  public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
    this.jdbcTemplate = jdbcTemplate;
  }

  public List search(User user) {
    final List list = new ArrayList();
    String sql = "select * from user_table where name = '" + user.getName() + "'or
email ='" + user.getEmail()
    + "'or contact='" + user.getContact() + "' or password='" + user.getPassword()  +
"' or id='" + user.getId() + "'";
    jdbcTemplate.query(sql, new ResultSetExtractor<User>() {

      public User extractData(ResultSet rs) throws SQLException, DataAccessException
{
        while (rs.next()) {
          User user = new User();
          user.setId(rs.getLong(1));
          user.setName(rs.getString(2));
          user.setEmail(rs.getString(3));
          user.setContact(rs.getString(4));
          user.setCity(rs.getString(5));
          user.setPassword(rs.getString(6));

          list.add(user);
        }
        return null;
      }
    });
    return list;
  }

  public List display() {
    final List list = new ArrayList();
    String sql = "select * from user_table";
    jdbcTemplate.query(sql, new ResultSetExtractor<User>() {
      public User extractData(ResultSet rs) throws SQLException, DataAccessException
{
        while (rs.next()) {
          User user = new User();
          user.setId(rs.getLong(1));
          user.setName(rs.getString(2));
          user.setEmail(rs.getString(3));
          user.setContact(rs.getString(4));
          user.setCity(rs.getString(5));
          user.setPassword(rs.getString(6));

          list.add(user);
        }
        return null;
      }
    });
    return list;
  }

  public List resetPassword(final User user) {
    String sql = "update user_table set password=? where email=?";
    jdbcTemplate.execute(sql, new PreparedStatementCallback<Boolean>() {

      public Boolean doInPreparedStatement(PreparedStatement ps) throws SQLException,
DataAccessException {
        ps.setString(1, user.getPassword());
        ps.setString(2, user.getEmail());

        return ps.execute();
      }

    });

    List list = display();
    return list;

  }

  public List updateprofile(final User user) {
    String sql = "update user_table set name=?,email=? ,contact=? where id=?";
    jdbcTemplate.execute(sql, new PreparedStatementCallback<Boolean>() {

      public Boolean doInPreparedStatement(PreparedStatement ps) throws SQLException,
DataAccessException {
        ps.setLong(1, 1);
        ps.setString(2, user.getName());
        ps.setString(3, user.getContact());
        ps.setString(4, user.getEmail());

        return ps.execute();
      }

    });
    List list = display();
    return list;
  }

  public List<User> findUserByID(Long id) {
    final List list = new ArrayList();
    String sql = "select * from user_table where id=?";
    jdbcTemplate.query(sql, new ResultSetExtractor<User>() {

      public User extractData(ResultSet rs) throws SQLException, DataAccessException
{
        while (rs.next()) {
          User user = new User();
          user.setId(rs.getLong(1));
          user.setName(rs.getString(2));
          user.setEmail(rs.getString(3));
          user.setContact(rs.getString(4));
          user.setCity(rs.getString(5));
          user.setPassword(rs.getString(6));

          list.add(user);
          return null;
        }
        return null;
      }

    });
    return list;

  }

  @Transactional
  public User getUser(Long userId) {
    return this.hibernateTemplate.get(User.class, userId);
  }

  @Transactional
  public List<Job> getjobDetails() {
    List<Job> jobs = this.hibernateTemplate.loadAll(Job.class);

    return jobs;

  }

  public List update(final User user) {
    String sql = "update user_table  set name=?,email=?,contact=?,city=? where id=?
";
    jdbcTemplate.execute(sql, new PreparedStatementCallback<Boolean>() {

      public Boolean doInPreparedStatement(PreparedStatement ps) throws SQLException
DataAccessException {
        ps.setString(4, user.getCity());

        ps.setString(3, user.getContact());
        ps.setString(2, user.getEmail());
        ps.setString(1, user.getName());

        ps.setLong(5, user.getId());
        return ps.execute();
      }

    });

    // TODO Auto-generated method stub
    List list = display();
    return list;
  }

  public List delete(User user) {
    String str = "delete from user_table where id='" + user.getId() + "' ";
    jdbcTemplate.update(str);
    List list = display();
    return list;
  }

}
```

5. Models, Controllers, Classes Developed

LITHAN

- ✓ **UserDao.java**

```java
package com.spring.mvc.dao;

import com.spring.mvc.model.User;

public interface UserDAO {
    public void registerUser(User user);
    public User loginUser(User user);
    public User checkMail(User user);
}
```

- ✓ **UserDaoImplementation.java**

```java
package com.spring.mvc.dao;


import javax.persistence.NoResultException;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.query.Query;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;
import com.spring.mvc.model.User;
@Repository
@Transactional
public class UserDAOImpl implements UserDAO {
//To call the session factory
  @Autowired
  private SessionFactory factory;

public void registerUser(User user) {
  Session session=factory.getCurrentSession();
  session.save(user);

}
```

5. Models, Controllers, Classes Developed

LITHAN

4.com.spring.mvc.model

✓ Admin.java

Model means bean. JavaBeans are classes that encapsulate many objects into a single object(the bean). It is a java class that should follow conventions and also model can be an object or collection of objects which basically contains the data of the application

```java
package com.spring.mvc.model;

import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.validation.constraints.NotNull;

@javax.persistence.Entity(name = "admin_table")
@javax.persistence.Table(name = "admin_table")
public class Admin {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private Long id;
  @NotNull
  private String name;
  @NotNull
  private String email;
  @NotNull
  private String password;
  public Long getId() {
    return id;
  }
  public void setId(Long id) {
    this.id = id;
  }
  public String getName() {
    return name;
  }
  public void setName(String name) {
    this.name = name;
  }
  public String getEmail() {
    return email;
  }
  public void setEmail(String email) {
    this.email = email;
  }
  public String getPassword() {
    return password;
  }
  public void setPassword(String password) {
    this.password = password;
  }
  public Admin(Long id, String name, String email, String password) {
    super();
    this.id = id;
    this.name = name;
    this.email = email;
    this.password = password;
  }
  @Override
  public String toString() {
    return "Admin [id=" + id + ", name=" + name + ", email=" + email + ", password="
+ password + "]";
  }
  public Admin() {
    super();
    // TODO Auto-generated constructor stub
  }

}
```

5. Models, Controllers, Classes Developed

LITHAN

✓ User.java

```
}
public void setCity(String city) {
  this.city = city;
}
public String getPassword() {
  return password;
}
public void setPassword(String password) {
  this.password = password;
}
public String getContact() {
  return contact;
}
public void setContact(String contact) {
  this.contact = contact;
}
public String getEmail() {
  return email;
}
public void setEmail(String email) {
  this.email = email;
}
public User(Long id, String name, String city, String password, String contact,
String email) {
  super();
  this.id = id;
  this.name = name;
  this.city = city;
  this.password = password;
  this.contact = contact;
  this.email = email;
}
@Override
public String toString() {
  return "User [id=" + id + ", name=" + name + ", city=" + city + ", password=" +
password + ", contact=" + contact
      + ", email=" + email + "]";
}
public User() {
  super();
  // TODO Auto-generated constructor stub
}

}
```

```
package com.spring.mvc.model;

import javax.persistence.Column;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@javax.persistence.Entity(name = "User_table")
@javax.persistence.Table(name = "User_table")
public class User {

  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)

  @Column(name="id")
  private Long id;
@Column(name="name")
  private String name;
@Column(name="city")
  private String city;
@Column(name="password")
  private String password;
@Column(name="contact")
  private String contact;
@Column(name="email")
  private String email;
public Long getId() {
  return id;
}
public void setId(Long id) {
  this.id = id;
}
public String getName() {
  return name;
}
public void setName(String name) {
  this.name = name;
}
public String getCity() {
  return city;
```

5. Models, Controllers, Classes Developed

LITHAN

5. com.spring.mvc.service

- ✓ **AdminService.java**

```
1  package com.spring.mvc.servic;
2
3  import com.spring.mvc.model.Admin;
4
5  public interface AdminService {
6      public Admin adminlogin(Admin admin);
7  }
8
```

A service class is used by a client to interact with some functionality in the application. Usually, it is public and has some business meaning.

- ✓ **AdminServiceImpl.java**

```
1  package com.spring.mvc.servic;
2
3  import javax.transaction.Transactional;
10 @Service
11 @Transactional
12 public class AdminSeviceImpl implements AdminService {
13
14     @Autowired
15     private AdminDAO adminDAO;
16     public Admin adminlogin(Admin admin) {
17         // TODO Auto-generated method stub
18         return adminDAO.adminlogin(admin);
19     }
20
21 }
```

5. Models, Controllers, Classes Developed

LITHAN

- ✓ **BulkMailService.java**

```java
import org.springframework.stereotype.Service;


@Service
public class BulkMailService {


    @Autowired
    private JavaMailSender mailSender;

    public  void sendMails(String receiver , String subject, String body) {

        MimeMessage message = this.mailSender.createMimeMessage();
        MimeMessageHelper mimeHelper;

        try {
            mimeHelper = new MimeMessageHelper(message, true);
            mimeHelper.setTo(receiver);
            mimeHelper.setFrom("abcjob36@gmail.com");
            mimeHelper.setSubject(subject);
            mimeHelper.setText(body, true);
            mailSender.send(message);

        } catch (MessagingException e) {
            System.out.println("Error Sending message" + e.getMessage());
```

- ✓ **ForgotPasswordMail.java**

```java
package com.spring.mvc.servic;

import javax.mail.MessagingException;

@Service
public class ForgotPassEmailService {

    @Autowired

    private JavaMailSender mailSender;

    public void resetMail(String emailID) {
        MimeMessage message = this.mailSender.createMimeMessage();
        MimeMessageHelper messageHelper;
        try {
            messageHelper = new MimeMessageHelper(message, true);
            messageHelper.setTo(emailID);
            messageHelper.setFrom("abcjobs36@gmail.com");
            messageHelper.setSubject("Forgot password reset");
            messageHelper.setText(
                    "<html><body style=\"font-size: 1rem;\"><h2>Good day! This is ABC JOBS</h2><h4>Use this link t
                            + "<button style=\""
                            + " border-radius: 1.5rem; text-decoration:none;background:white;padding:15px; color:
                            + "<a href='http://localhost:8082/portalAbc/reset_forgotpassform/" + emailID
                            + "/'> Reset your Password</a></button> </body></html>",
                    true);
            mailSender.send(message);
        } catch (MessagingException e) {
            System.out.println("Error sendind mail" + e.getMessage());
        }
    }
}
```

5. Models, Controllers, Classes Developed

LITHAN

- ✓ **MailService.java**

```java
package com.spring.mvc.servic;

import java.util.Date;

@Service
public class MailService {

    @Autowired
    private JavaMailSender mailSender;

    public void sendMail(String emailId, String userName) {

        MimeMessage message = this.mailSender.createMimeMessage();
        MimeMessageHelper mimeHelper;

        try {
            mimeHelper = new MimeMessageHelper(message, true);
            mimeHelper.setTo(emailId);
            mimeHelper.setFrom("abcjob36@gmail.com");
            mimeHelper.setSubject("Registration Confirmation");
            mimeHelper.setText("<html><body><h1>ABC Ptd Ltd</h1>" + "\n" + "<h3>Hi " + userName + "</h3>" + "\n" +
                    + " <h3 style=\"color: rgb(114, 114, 202);\">You have registered successfully</h3>\r\n"

                    + "" + "\n" + new Date() + "</body></html>", true);
            mailSender.send(message);

        } catch (MessagingException e) {
            System.out.println("Error Sending message" + e.getMessage());
        }

    }
}
```

- ✓ **UserService.java**

```java
package com.spring.mvc.servic;

import com.spring.mvc.model.User;


public interface UserService {
    public void registerUser(User user);
    public User loginUser(User user);
    public User checkMail(User user);
}

```

5. Models, Controllers, Classes Developed

LITHAN

- ✓ **UserServiceImplementation.java**

```
1  package com.spring.mvc.servic;
2
3  import javax.transaction.Transactional;
11 @Service
12 @Transactional
13 @Repository
14 public class UserServiceImpl implements UserService    {
15     @Autowired
16     private UserDAO userDAO;
17
18     public void registerUser(User user) {
19
20         userDAO.registerUser(user);
21     }
22
23     public User loginUser(User user) {
24
25         return userDAO.loginUser(user);
26     }
27
28     public User checkMail(User user) {
29         // TODO Auto-generated method stub
30         return userDAO.checkMail(user);
31     }
32
33 }
34
```

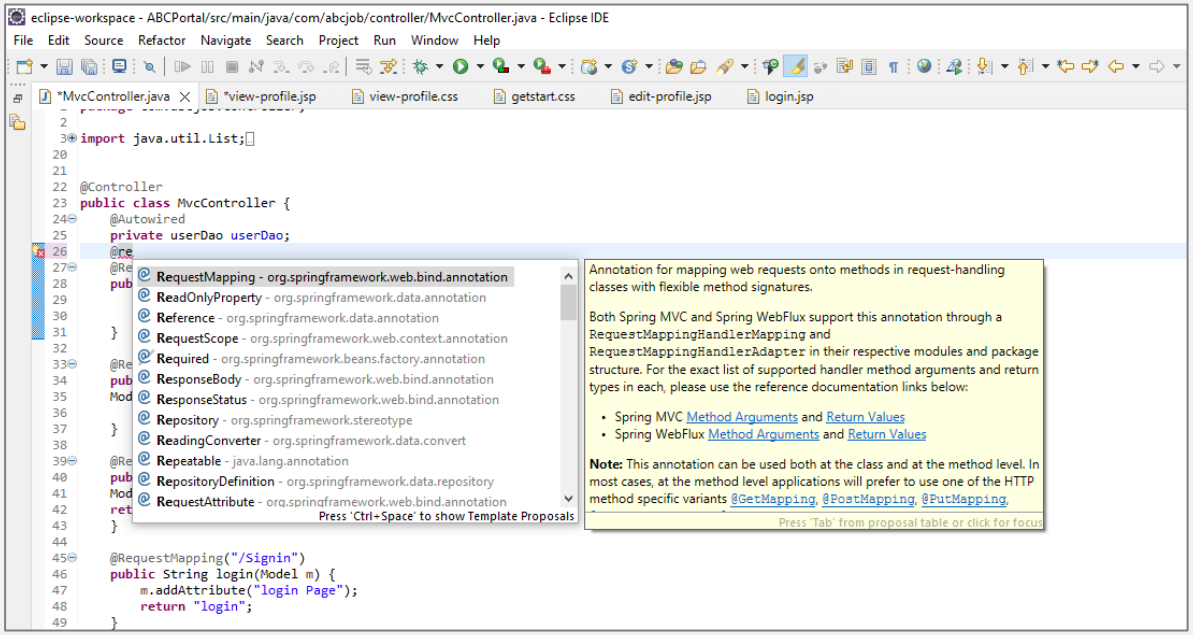
6. Tool Screens

LITHAN

- **Tool Screens features**

Smart Code Completion

We get code of completion as quickly as we start typing in eclipse, we don’t always need to apply a keyboard shortcut to deliver the suggestions. suggestions are also context sensitive so that they must make feel for the current place we’re typing

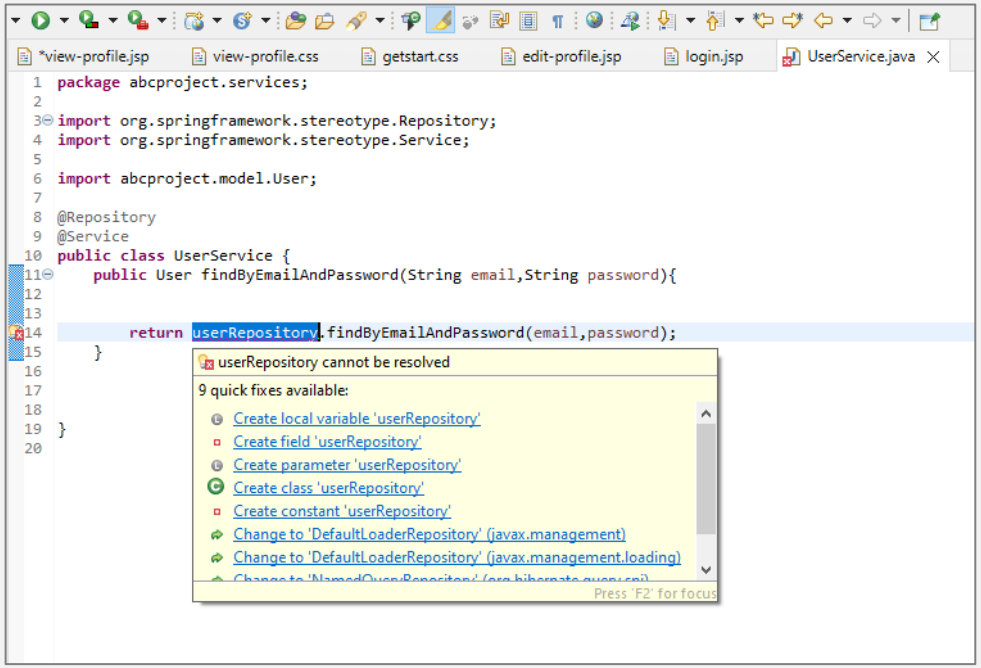


6. Tool Screens

LITHAN

Quick Fixes

Each time Ide detects an error, the eclipse highlights the error the use of a wavy red line underneath the offending code and a marker in the left editor margin. moving the mouse over the underlined code or the marker will indicate the error. The marker may be selected with the left mouse to set off the quick restoration pop-up, indicating moves that may be undertaken to repair the error. rather, pressing Ctrl+1 will activate brief restore from the keyboard.



Getter and Setter

Getters and setters are used to shield your data, in particular when developing classes. For each example variable, a getter method returns its value while a setter method sets or updates its value. Given this, getters and setters are also called accessors and mutators, respectively

```
public String getFristname() {
    return fristname;
}
public void setFristname(String fristname) {
    this.fristname = fristname;
}
public String getLastname() {
    return lastname;
}
public void setLastname(String lastname) {
    this.lastname = lastname;
}
public String getEmail() {
    return email;
}
public void setEmail(String email) {
    this.email = email;
}
```

7. Classes - Entities

LITHAN

- Entities

S. No.	Class Name	Description
1	User	Entity for connecting to tb_users table, used for saving user account
2	UserProfile	Entity for connecting to tb_user_profile table, used for saving user profile
3	Admin	Entity for connecting to tb_admin table, used for fetch admin table
4	BulkEmail	Entity for sending bulk email from admin side

8. Classes Developed

LITHAN

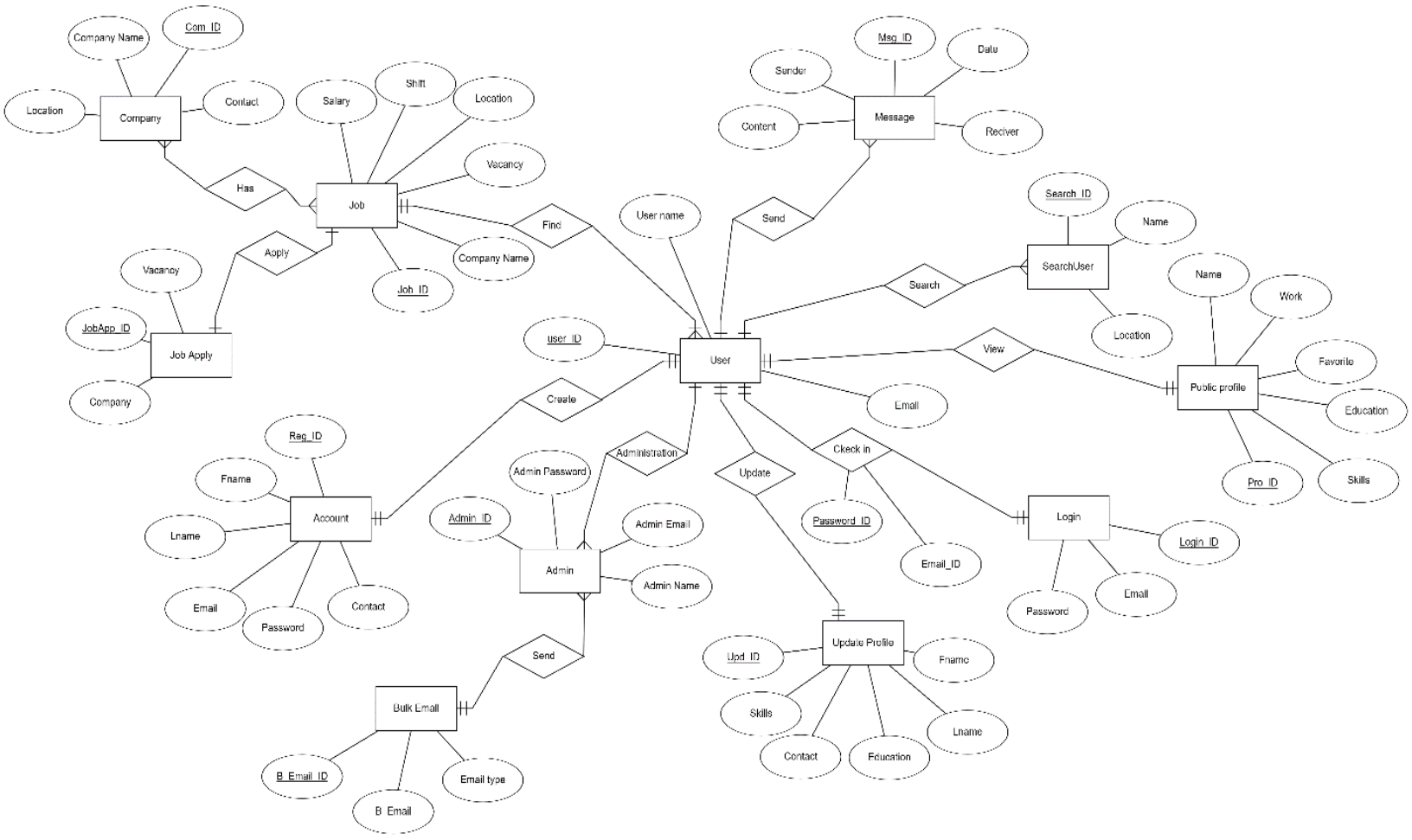
- Business Classes

S. No.	Class Name	Description
1	LoginService	Service class to handle business logic for user login
2	RegisterService	Service class to handle business logic for user registering to the website
3	AdminService	Service class to handle business logic for administration system of the website
4	ForgotPasswordService	Service class to handle business logic for user password retrieval functionality
6	UpdateProfileService	Service class to handle business logic for user who want edit their profile page
7	EmailSenderService	Service class to handle business logic for sending email
8	SearchService	Service class to handle business logic for search user and view user profile page

9. Database Design

LITHAN

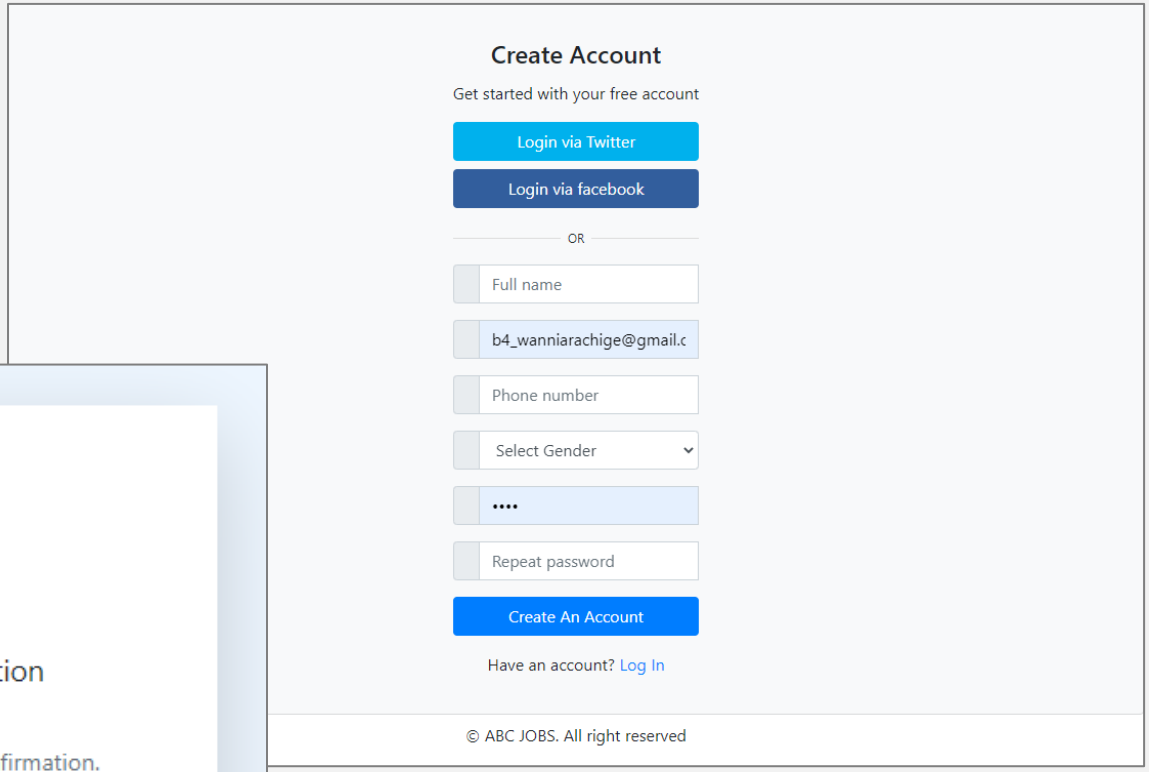
- Paste the ER Diagram Developed in Module 4



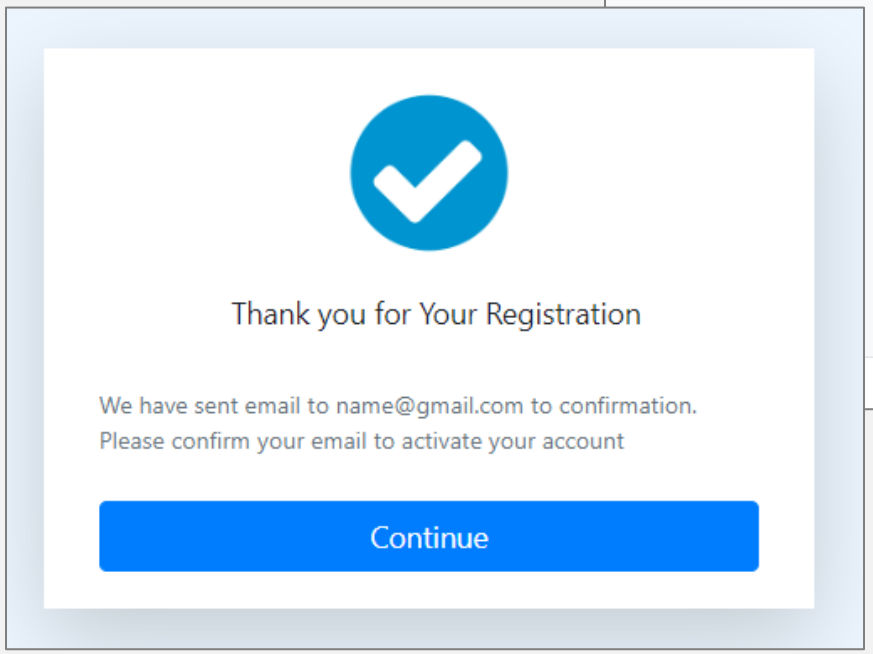
10. UI Design

LITHAN

Registration



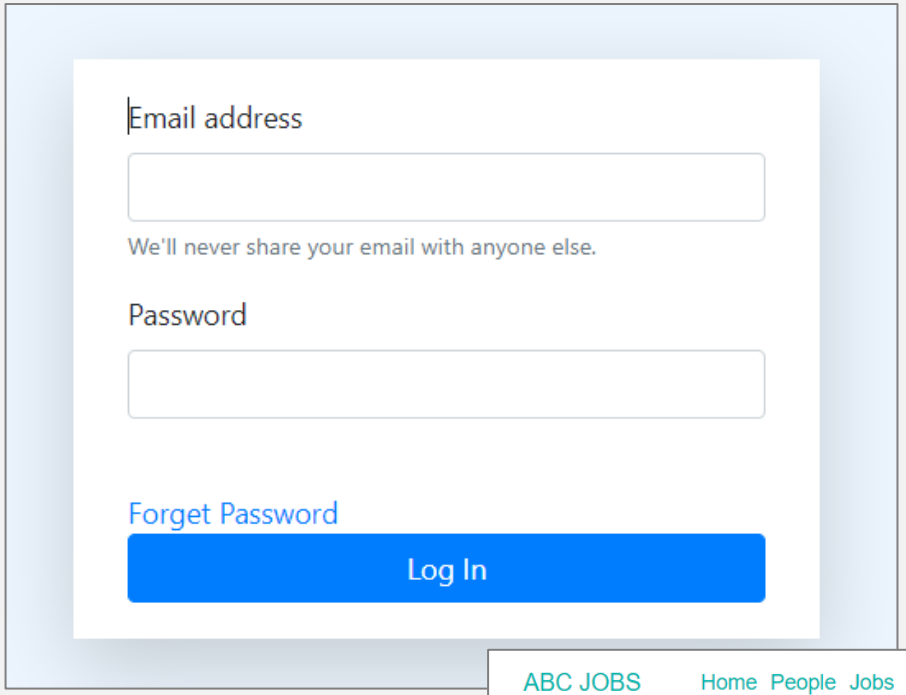
Thank You



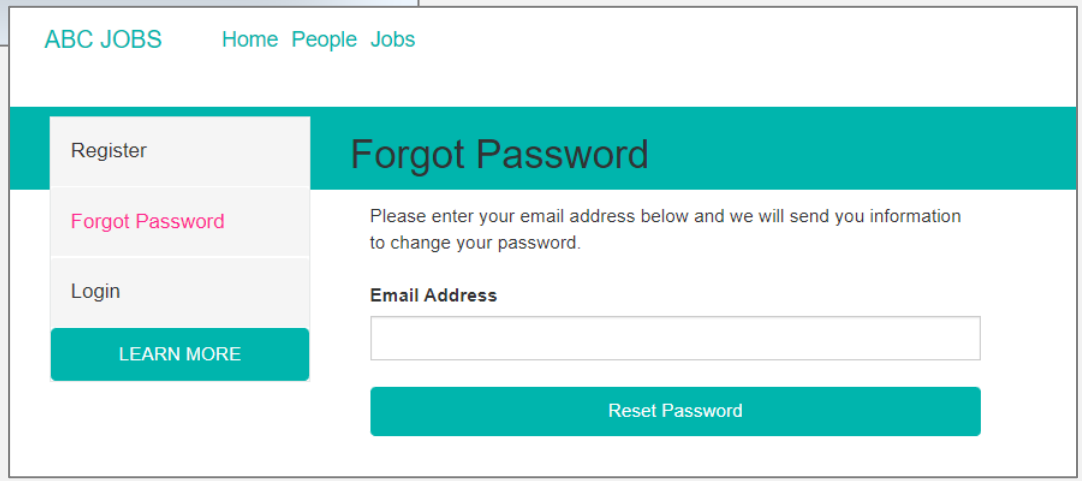
10. UI Design

LITHAN

Login



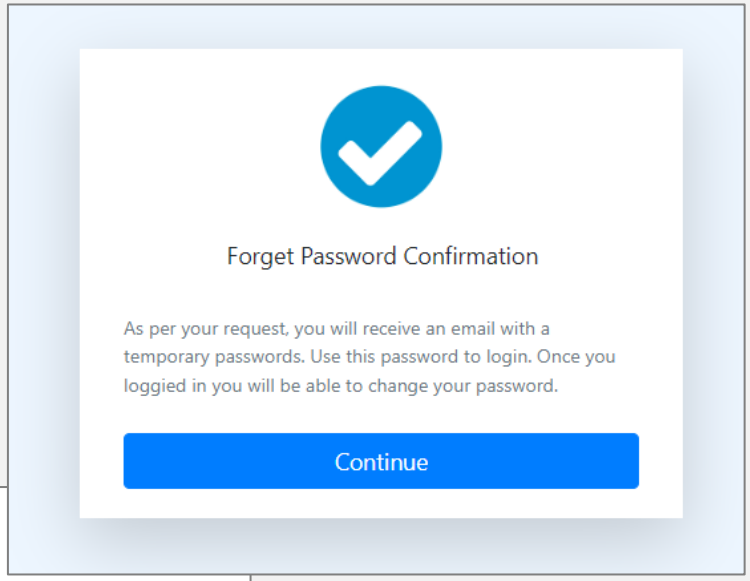
Forgot Password



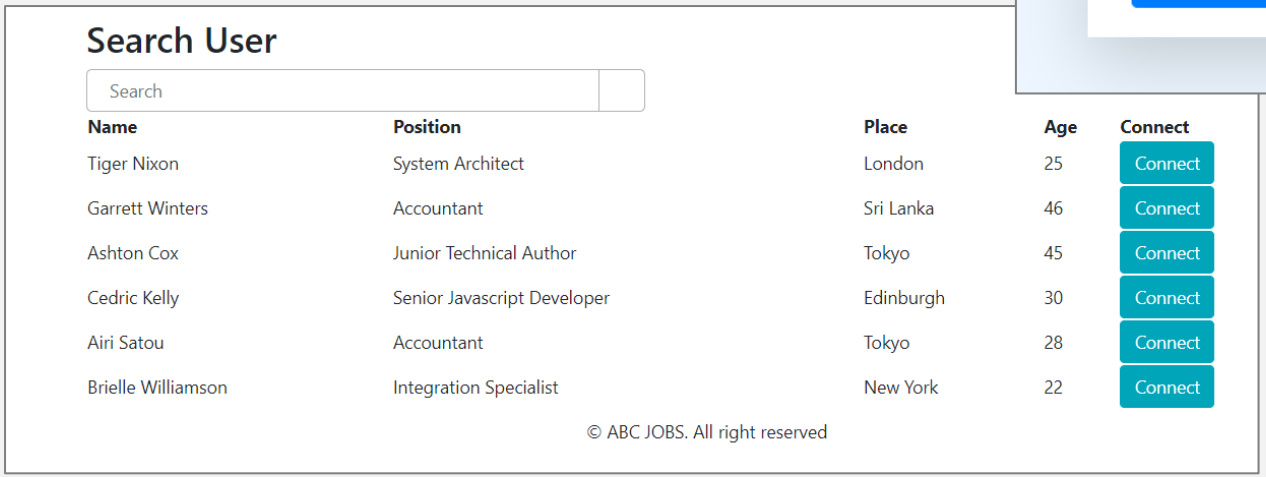
10. UI Design

LITHAN

Forgot Password Confirmation



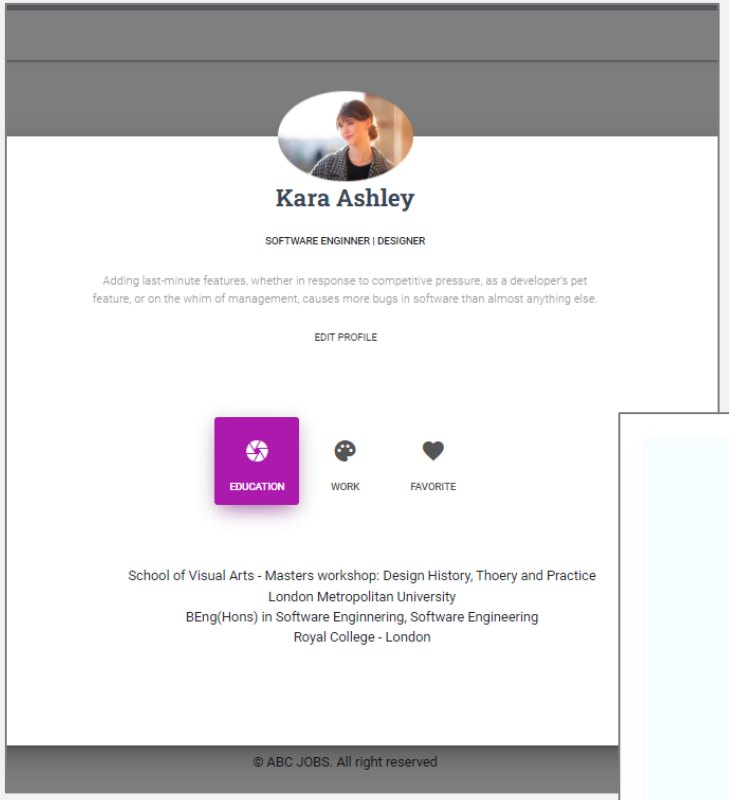
Search User



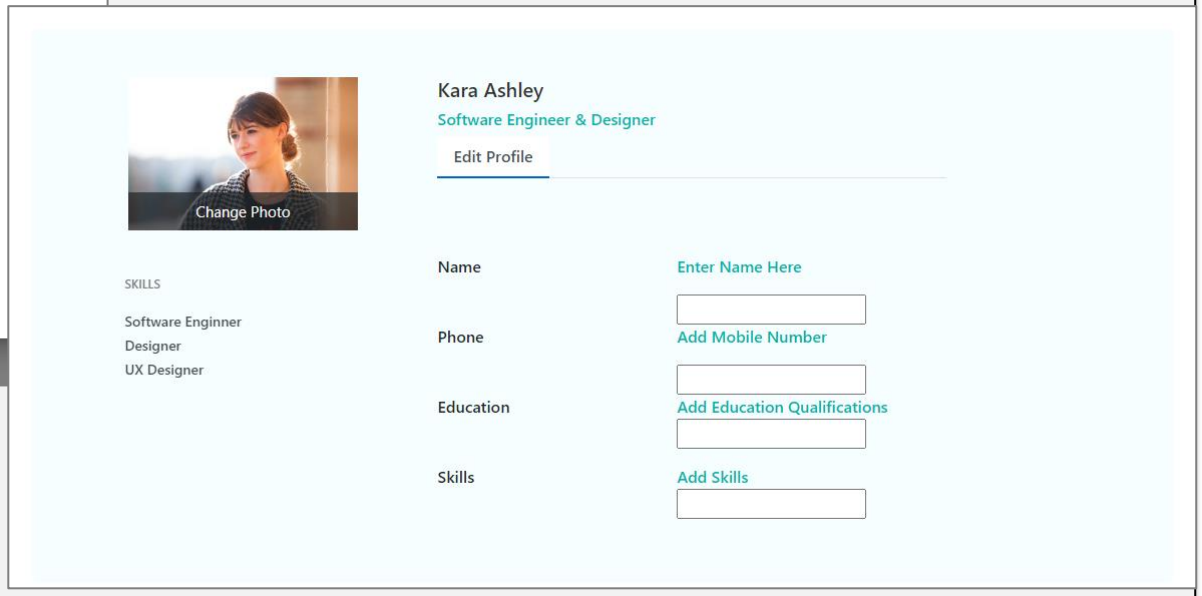
10. UI Design

LITHAN

User Profile



Edit Profile



11.Spring Frameworks

LITHAN

❑ Java spring includes

S. No.	Include	Purpose of Inclue
01	Spring-Core	Spring allows you to create apps out of "plain old java objects" and non-invasively deploy corporate services to POJOs
02	Spring-context	Responsible for instantiating, configuring, and assembling beans by reading configuration metadata from XML, Java annotations, and/or Java code in the configuration files
03	Spring-beans	Beans make up the applications structure. It's a just object that IoC containers instantiates
04	Spring-jdbc	This helps to connect the database and run queries
05	Spring-webmvc	Model view controller architecture and ready components for developing flexible applications proviedes

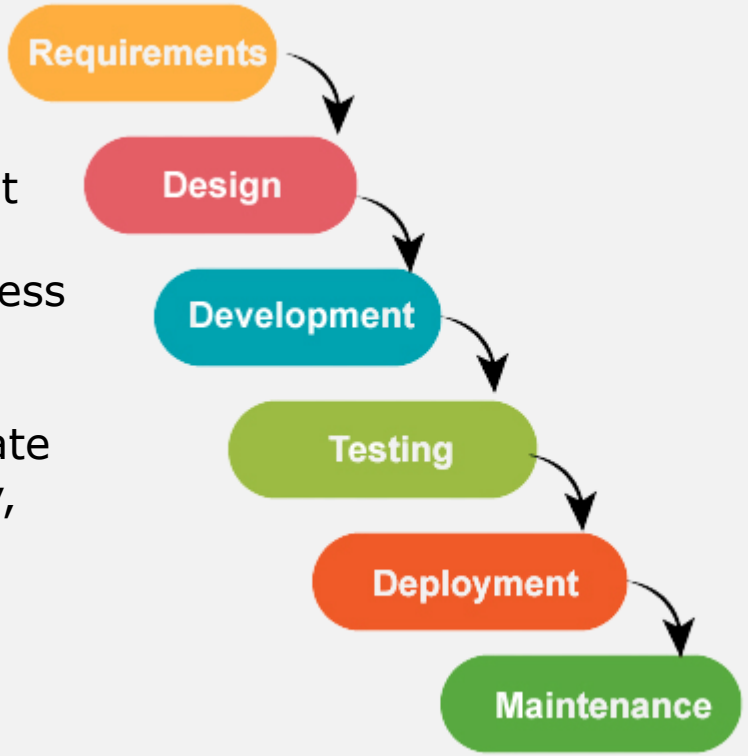
12. Software Development Methodology

LITHAN

- Software Development Methodology being used

Waterfall Model

The waterfall method turned into the first SDLC version to be used broadly in Software Engineering to ensure the success of the assignment. In "The Waterfall" technique, the entire system of software program development is split into separate stages. in this Waterfall model, generally, the outcome of one segment acts as the input for the subsequent section sequentially



13. Project Plan

LITHAN

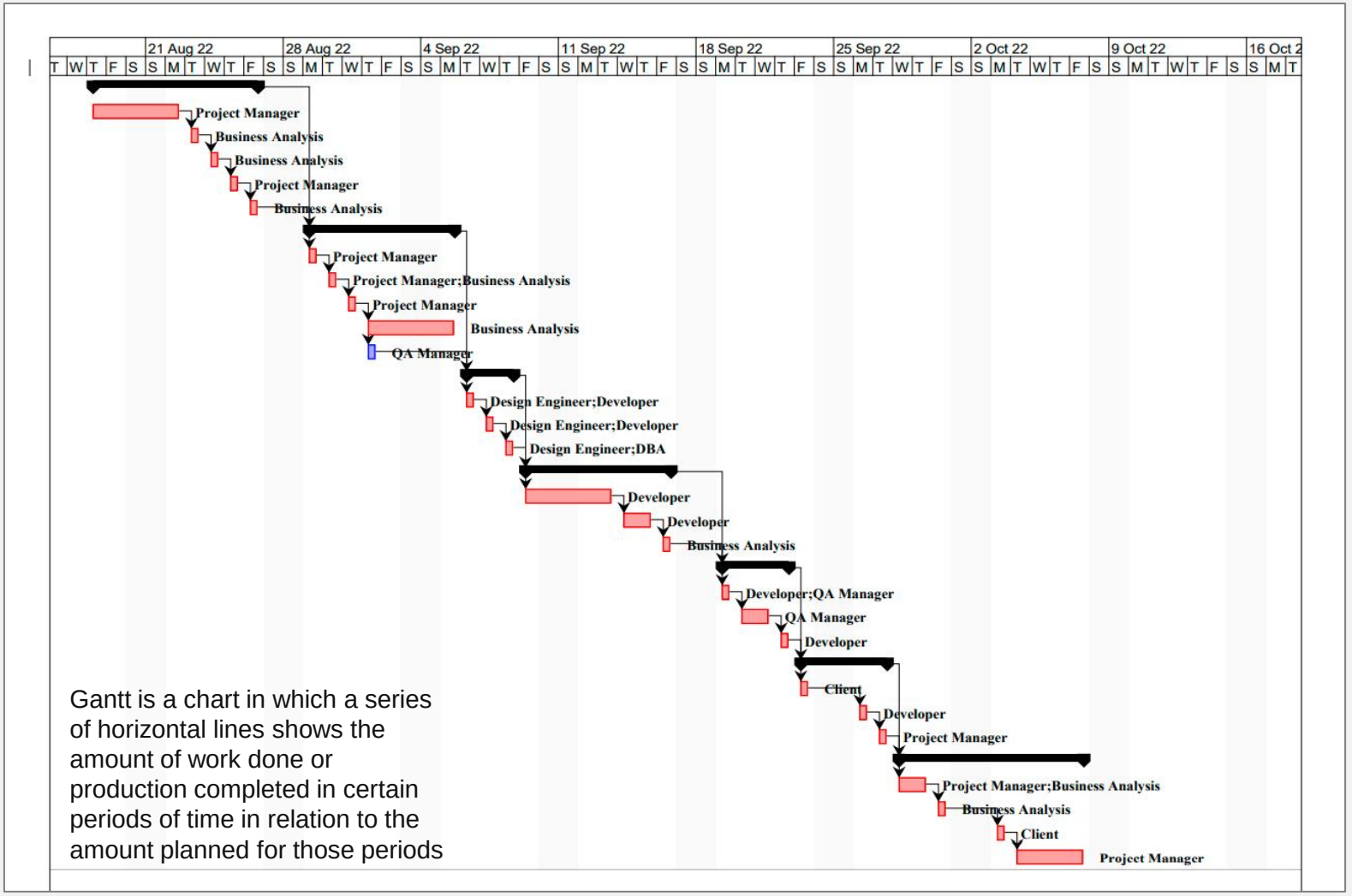
- Project Plan

		Name	Duration	Start	Finish	Predecessors	Resource Names	M
1		**Requirement Gathering**	**7 days**	**8/18/22 8:00 AM**	**8/26/22 5:00 PM**			
2		Kickoff Meetings	3 days	8/18/22 8:00 AM	8/22/22 5:00 PM		Project Manager	
3		Identify Stackholders	1 day	8/23/22 8:00 AM	8/23/22 5:00 PM	2	Business Analysis	
4		Gather and Documentation	1 day	8/24/22 8:00 AM	8/24/22 5:00 PM	3	Business Analysis	
5		Set the Objective	1 day	8/25/22 8:00 AM	8/25/22 5:00 PM	4	Project Manager	
6		Prepare SRS Document	1 day	8/26/22 8:00 AM	8/26/22 5:00 PM	5	Business Analysis	
7		**Planning**	**6 days**	**8/29/22 8:00 AM**	**9/5/22 5:00 PM**	**1**		
8		Document the Scope	1 day	8/29/22 8:00 AM	8/29/22 5:00 PM	6	Project Manager	
9		Cost Estimation	1 day	8/30/22 8:00 AM	8/30/22 5:00 PM	8	Project Manager;Business Analysis	
10		Allocating Resources	1 day	8/31/22 8:00 AM	8/31/22 5:00 PM	9	Project Manager	
11		Risk Planning	3 days	9/1/22 8:00 AM	9/5/22 5:00 PM	10	Business Analysis	
12		Test Planning	1 day	9/1/22 8:00 AM	9/1/22 5:00 PM	10	QA Manager	
13		**Design**	**3 days**	**9/6/22 8:00 AM**	**9/8/22 5:00 PM**	**7**		
14		Prepare Technical Design	1 day	9/6/22 8:00 AM	9/6/22 5:00 PM	12	Design Engineer;Developer	
15		Prepare Prototype	1 day	9/7/22 8:00 AM	9/7/22 5:00 PM	14	Design Engineer;Developer	
16		Design Database	1 day	9/8/22 8:00 AM	9/8/22 5:00 PM	15	Design Engineer;DBA	
17		**Implementation**	**6 days**	**9/9/22 8:00 AM**	**9/16/22 5:00 PM**	**13**		
18		Coding the Source Code	3 days	9/9/22 8:00 AM	9/13/22 5:00 PM	16	Developer	
19		Implement the database	2 days	9/14/22 8:00 AM	9/15/22 5:00 PM	18	Developer	
20		Prepare Business Logics	1 day	9/16/22 8:00 AM	9/16/22 5:00 PM	19	Business Analysis	
21		**Testing**	**4 days**	**9/19/22 8:00 AM**	**9/22/22 5:00 PM**	**17**		
22		Conduct Unit Testing	1 day	9/19/22 8:00 AM	9/19/22 5:00 PM	20	Developer;QA Manager	
23		Conduct Integration Test	2 days	9/20/22 8:00 AM	9/21/22 5:00 PM	22	QA Manager	
24		Fix the erros	1 day	9/22/22 8:00 AM	9/22/22 5:00 PM	23	Developer	
25		**Deployment**	**3 days**	**9/23/22 8:00 AM**	**9/27/22 5:00 PM**	**21**		
26		UAT Testing	1 day	9/23/22 8:00 AM	9/23/22 5:00 PM	24	Client	
27		Fix errors	1 day	9/26/22 8:00 AM	9/26/22 5:00 PM	26	Developer	
28		Deploy the website	1 day	9/27/22 8:00 AM	9/27/22 5:00 PM	27	Project Manager	
29		**Maintenance**	**8 days**	**9/28/22 8:00 AM**	**10/7/22 5:00 PM**	**25**		
30		Release the user guide Document	2 days	9/28/22 8:00 AM	9/29/22 5:00 PM	28	Project Manager;Business Analysis	
31		Proveide post deployment	1 day	9/30/22 8:00 AM	9/30/22 5:00 PM	30	Business Analysis	
32		Service level agreement	1 day	10/3/22 8:00 AM	10/3/22 5:00 PM	31	Client	
33		Complete Project	4 days	10/4/22 8:00 AM	10/7/22 5:00 PM	32	Project Manager	

13. Project Plan

LITHAN

- Gantt Chart



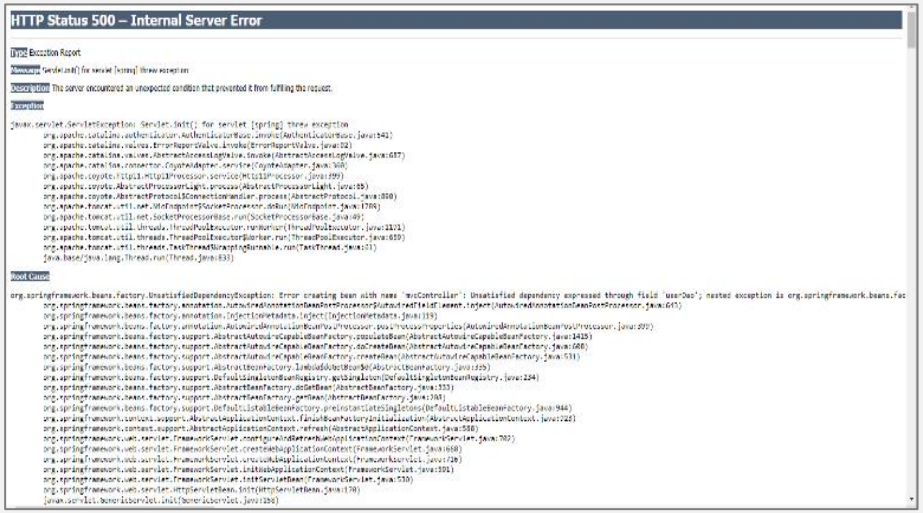
Gantt is a chart in which a series of horizontal lines shows the amount of work done or production completed in certain periods of time in relation to the amount planned for those periods

14. Application Screen Shots

LITHAN

- Screenshots demonstrates the error

Error 1



Solution

- Look at HTTP Status
- Check out the Root Cause
- Look at the last sentence
- Unable to create requested service [org.hibernate.engine.jdbc.env.spi.JdbcEnvironment]
- Likely caused by JDBC or repository interfaces
- Check persistence.xml
- Fixed jdbc:mysql://localhost:5476 to jdbc:mysql://localhost:3306

14. Application Screen Shots

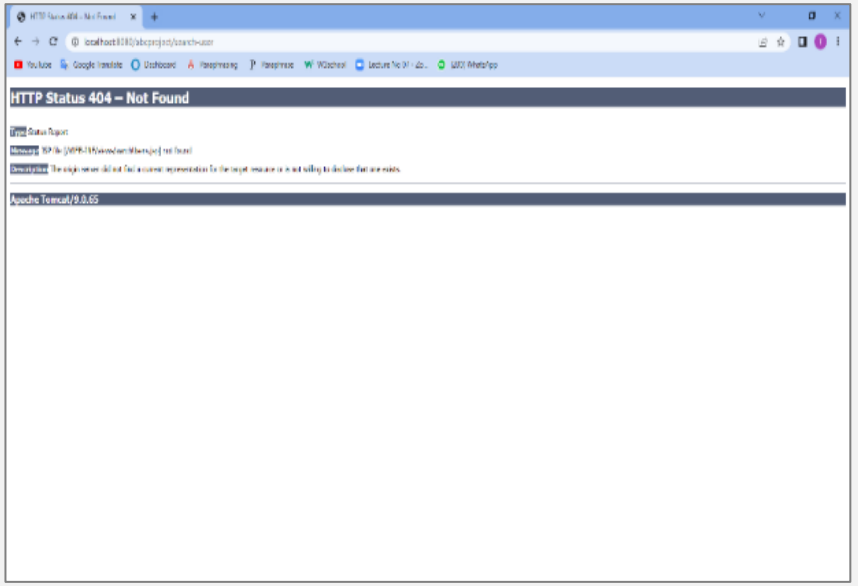
LITHAN

- Screenshots demonstrates the error

Error 2

Solution

- Look at HTTP Status
- Check out the Root Cause
- Look at the message
- JSP file [/WEB-INF/views/earchUsers.jsp] not found
- Check the MainController.java class
- Add return page to search method

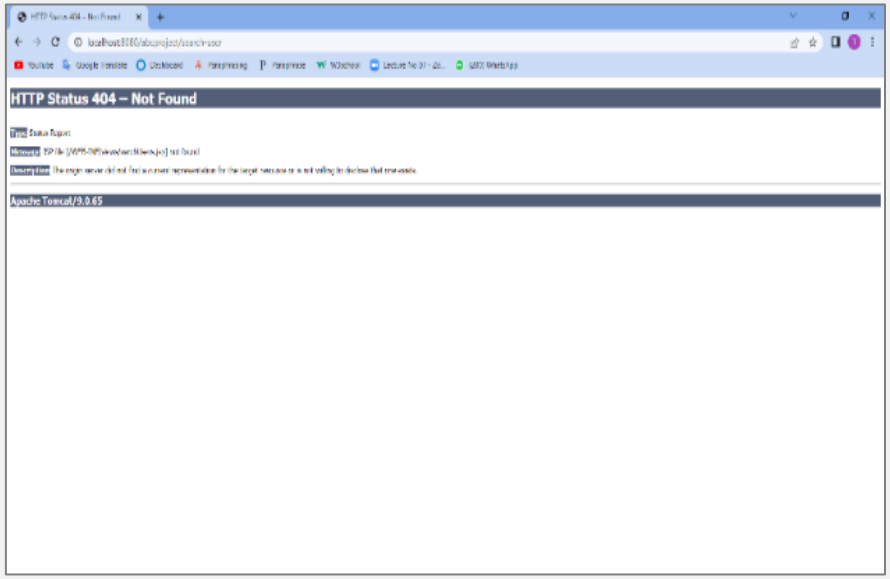


14. Application Screen Shots

LITHAN

- Screenshots demonstrates the error

Error



Solution

- Look at HTTP Status
- Check out the Root Cause
- Look at the description

-The origin server did not find a current representation for the target resource or is not willing to disclose that one exists.

- Check the MainController.java class
- Add request mapping for home.jsp

15. Project Milestones & Tasks

LITHAN

Project Task ID	Project Task Description	Project Milestone ID
1	Setup the Development Environment based on the Technical Development Environment	1
2	Power point Detailing the Business Process in the Proposed Application	1
3	Project Report detailing the Technical Design	1
4	Power Point Detailing the Project Plan	1
5	Develop the UI HTML standard pages (Based on Module 3 work)	2
6	Create common Struts Includes, Models, Views & Controllers	2
7	Develop Classes	2
8	Develop Struts Application	2

16. Project Milestones & Tasks

LITHAN

Project Task ID	Project Task Description	Project Milestone ID
9	Develop Test Cases	3
10	Develop Project Documentation	3
11	Develop User Manual	3

17. Milestone Feedback & Action taken

LITHAN

Project Milestone ID	Milestone Feedback received from Tutor / Learning Facilitator	Action Taken (Yes / No)
1	Add dependencies	Yes
	Download spring frameworks and add	Yes
2	Connect registration with data base	Yes
	Add user dashboard	Yes
3	If u can add bulk email system	yes

18. Modifications Made based On Feedback

LITHAN

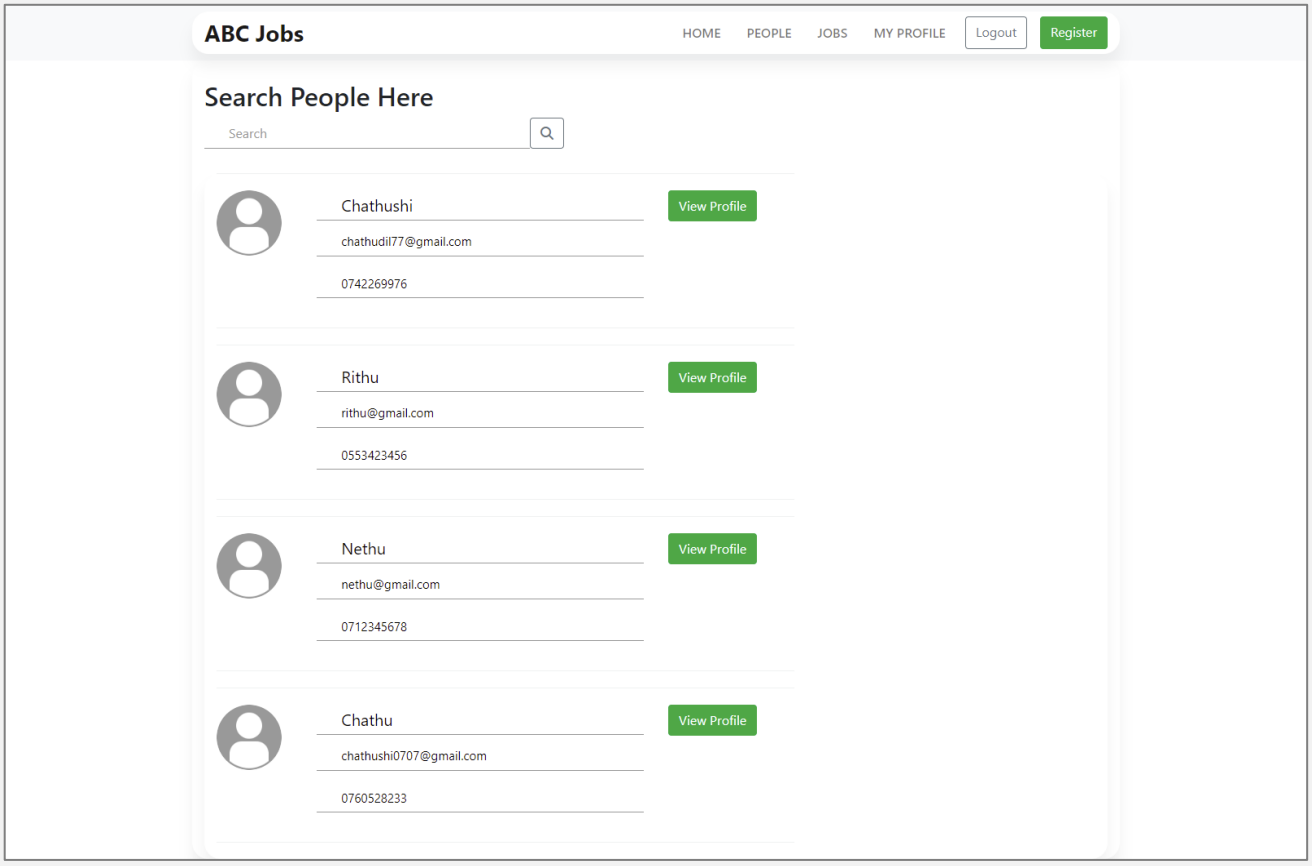
I received feedback to add suggestions to the search user page. So I added account suggestions to the search user page

Before

18. Modifications Made based On Feedback

LITHAN

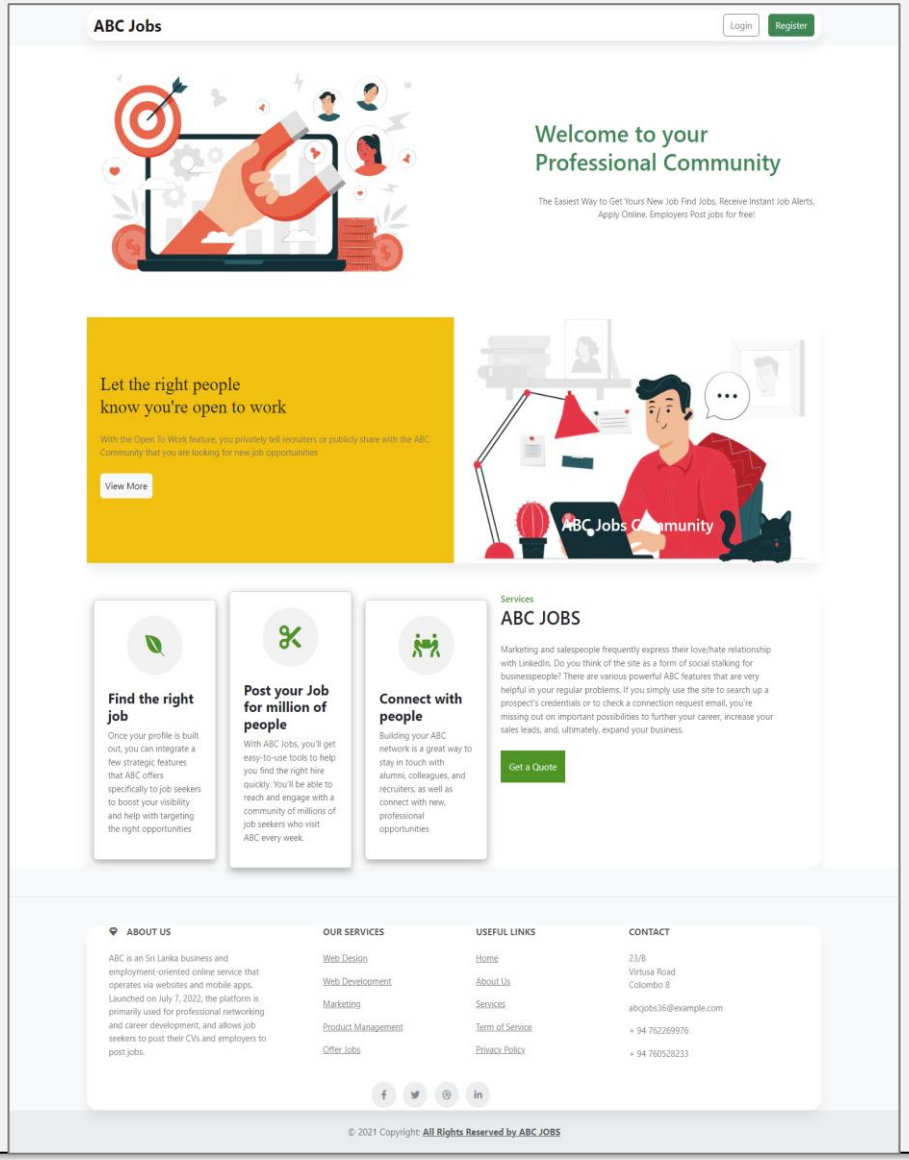
After



19. Project Results

LITHAN

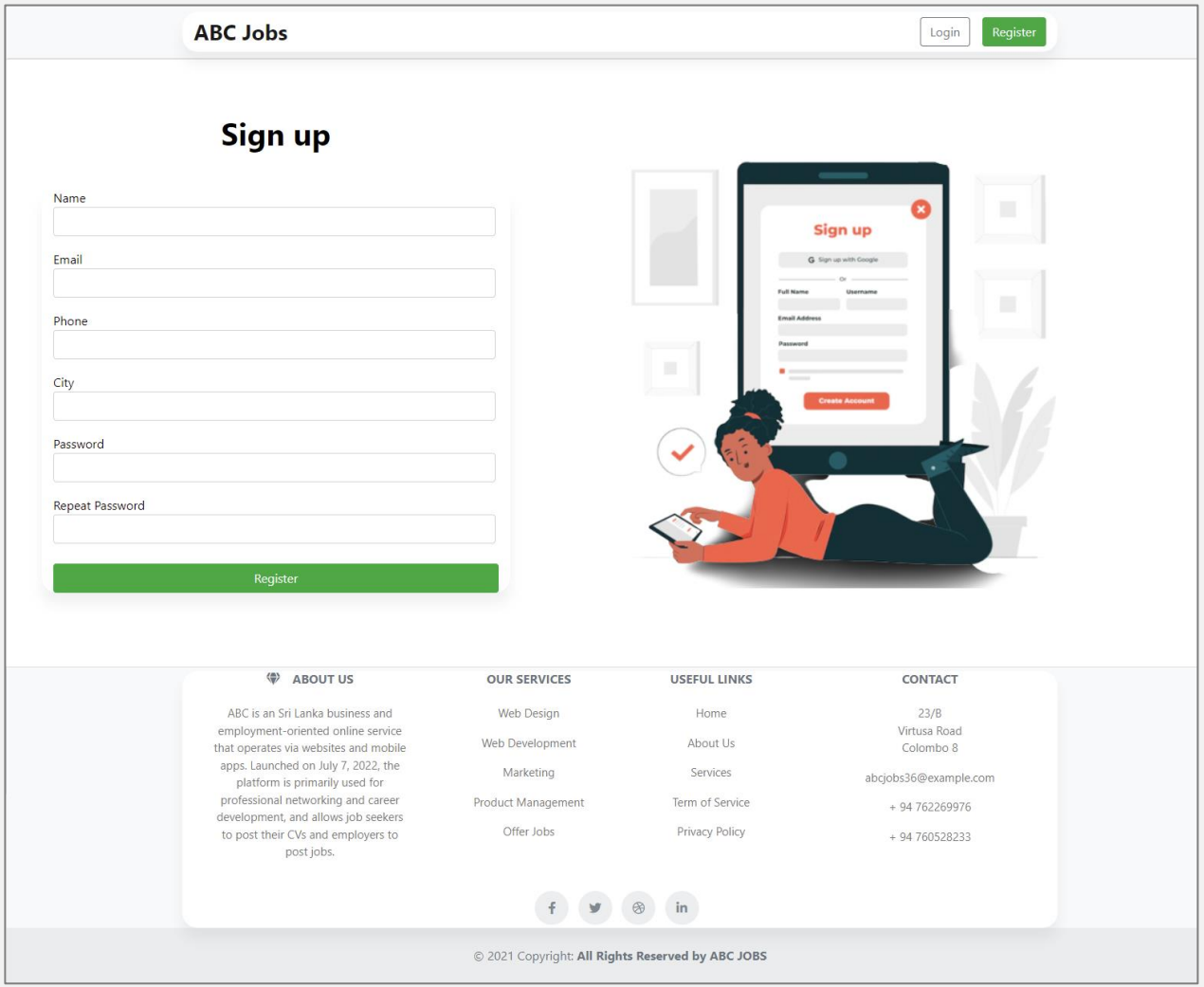
Index Page



19. Project Results

LITHAN

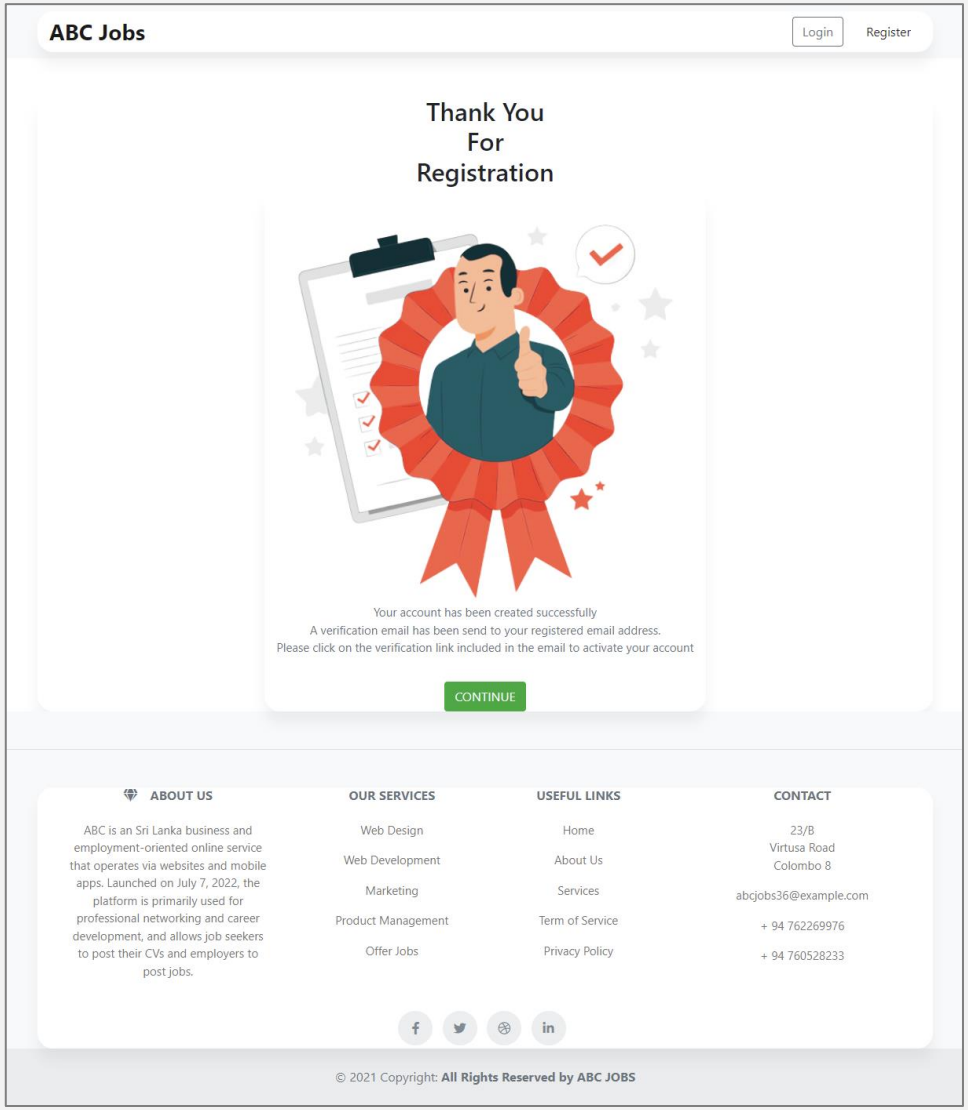
Registration



19. Project Results

LITHAN

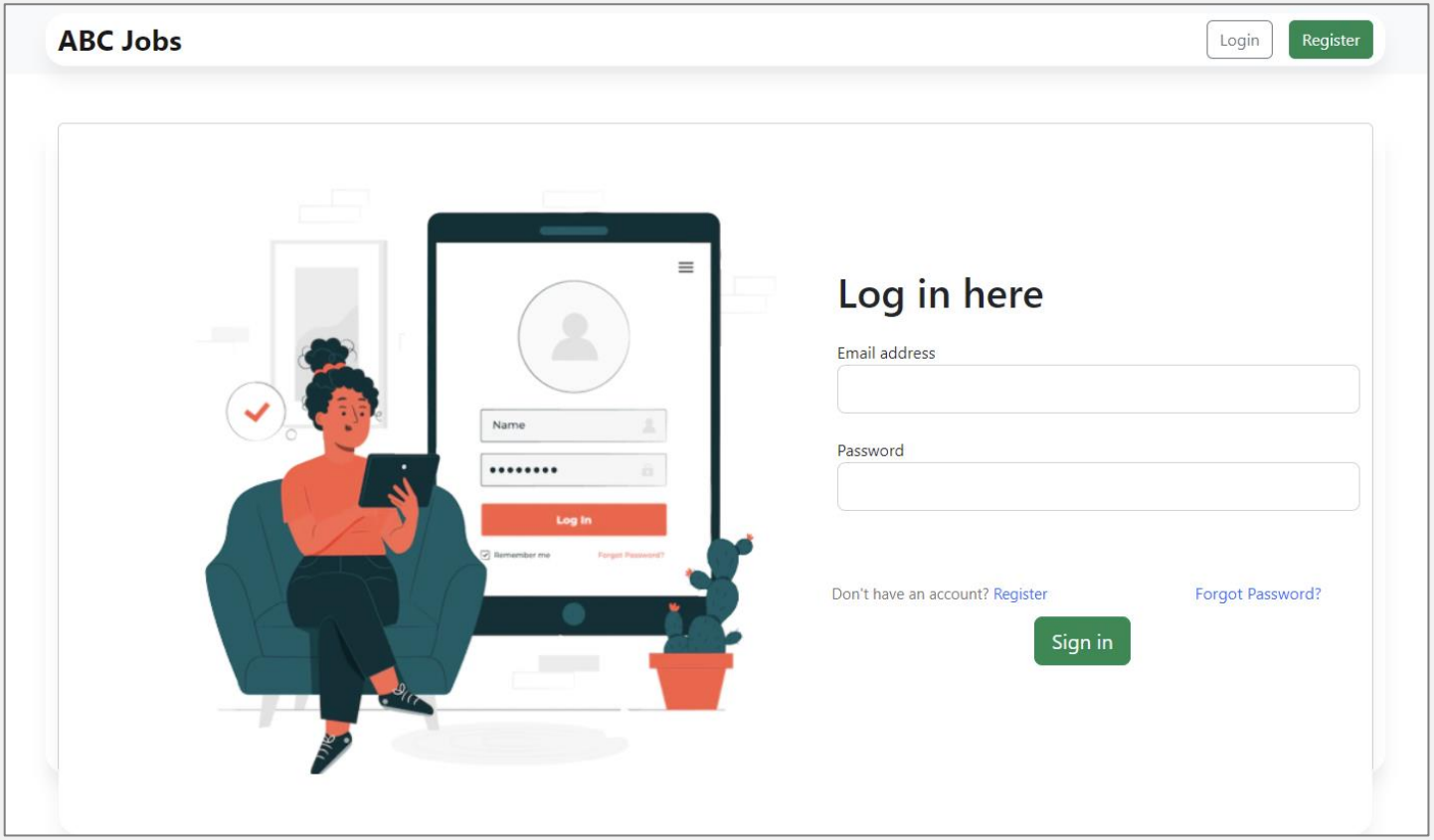
Thank you



19. Project Results

LITHAN

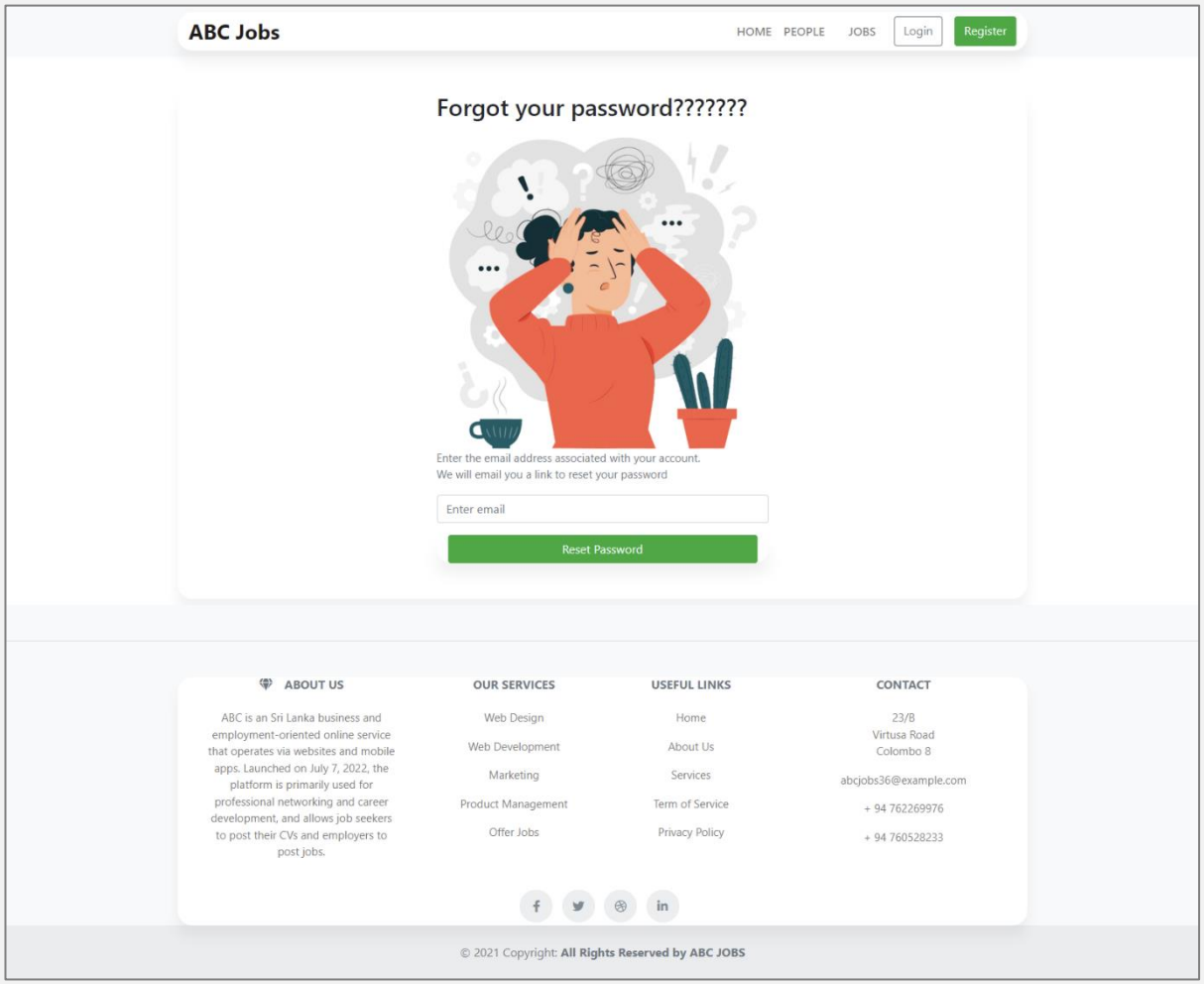
Login



19. Project Results

LITHAN

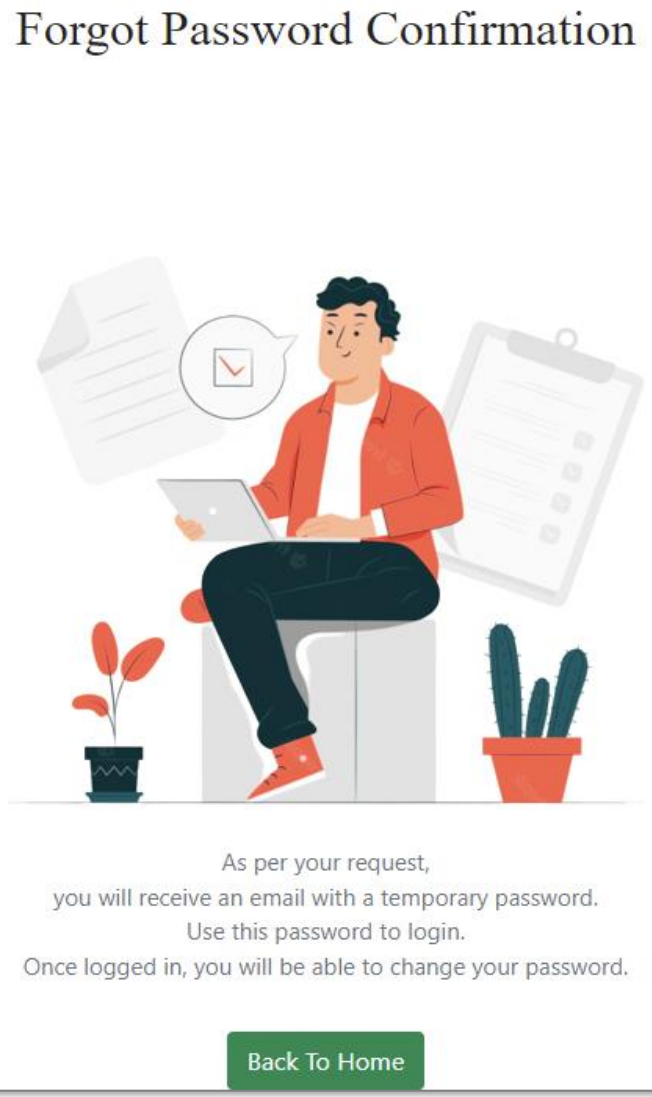
Forgot Password



19. Project Results

LITHAN

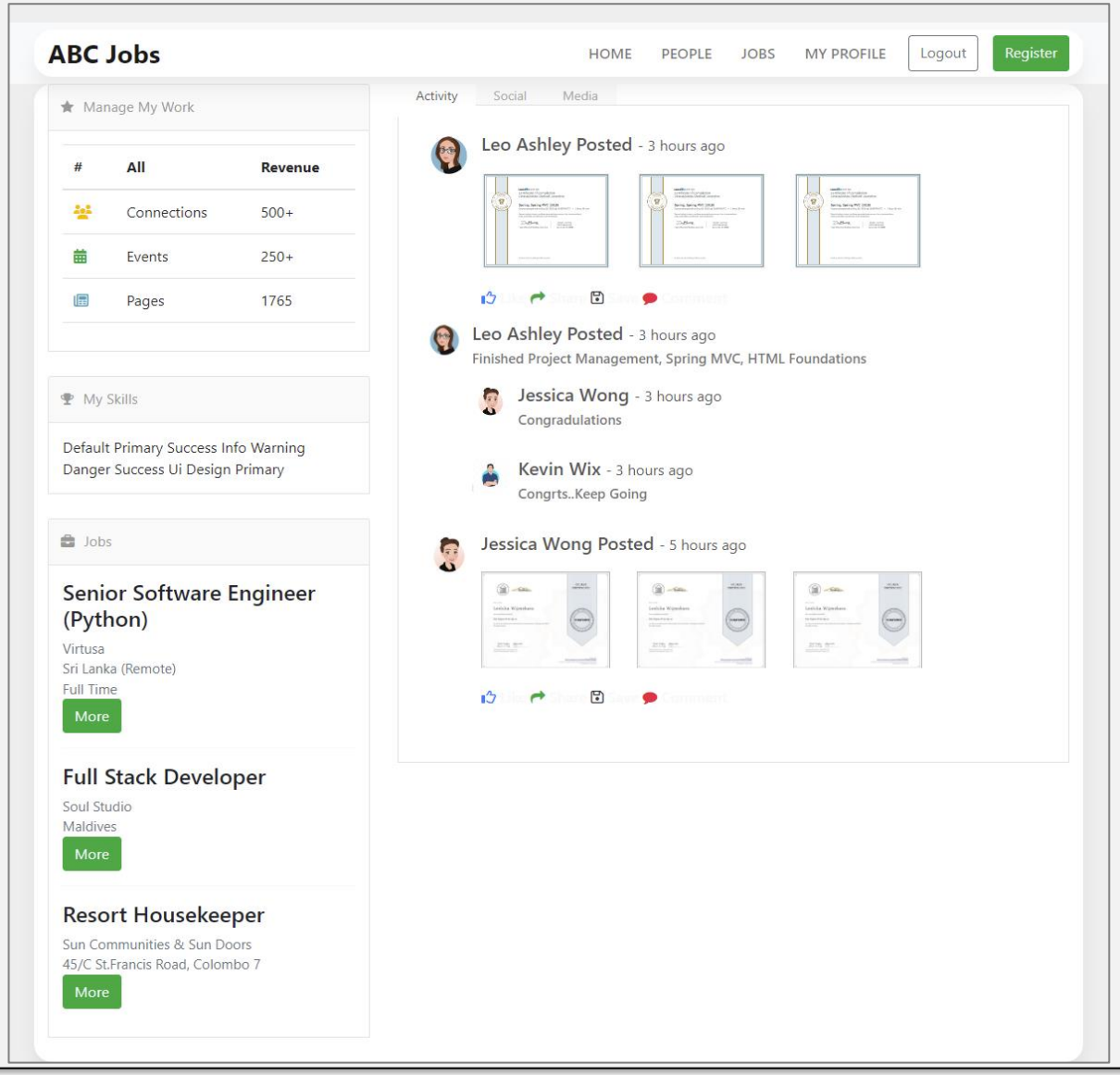
Forgot Password Confirmation



19. Project Results

LITHAN

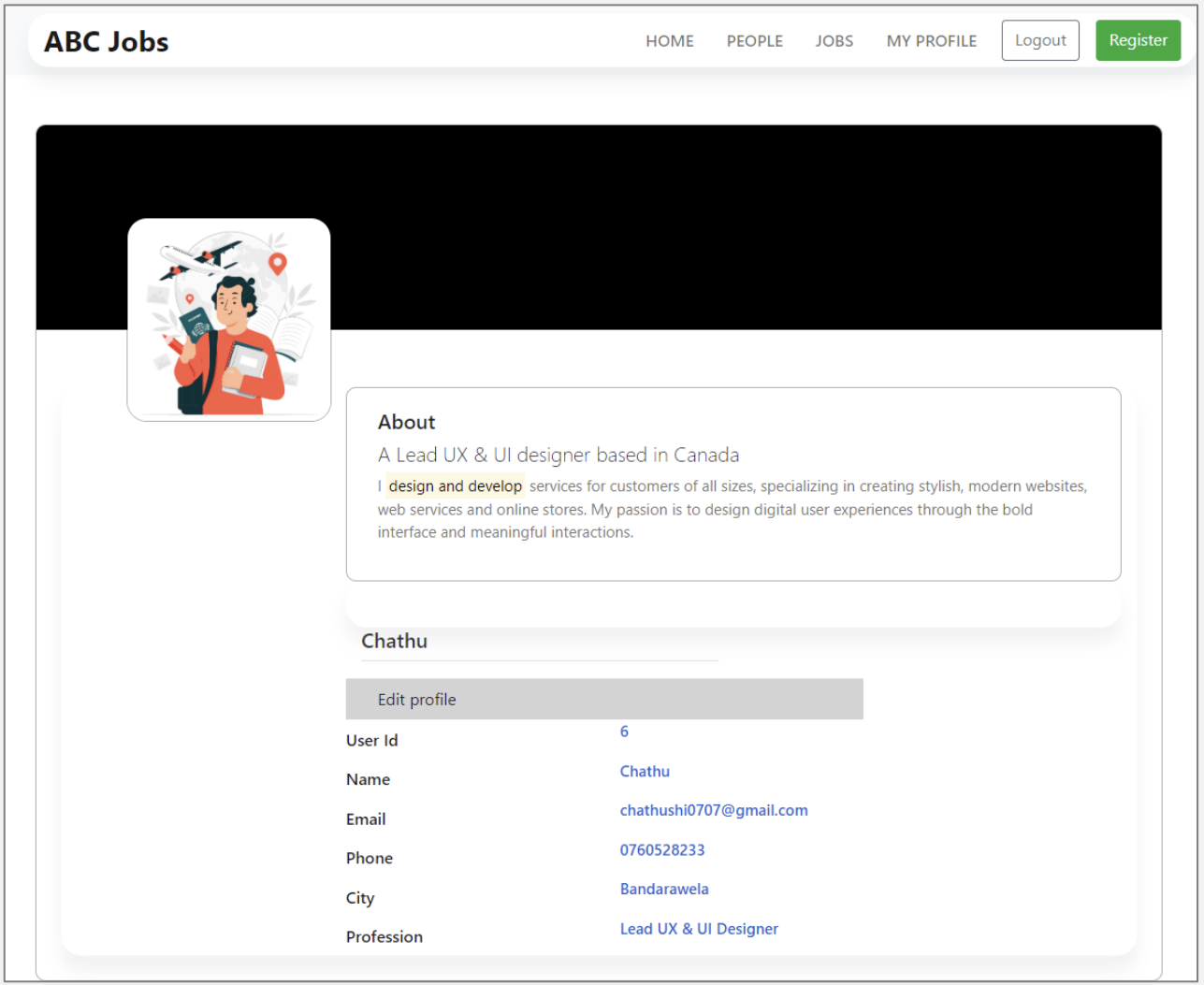
User Dashboard



19. Project Results

LITHAN

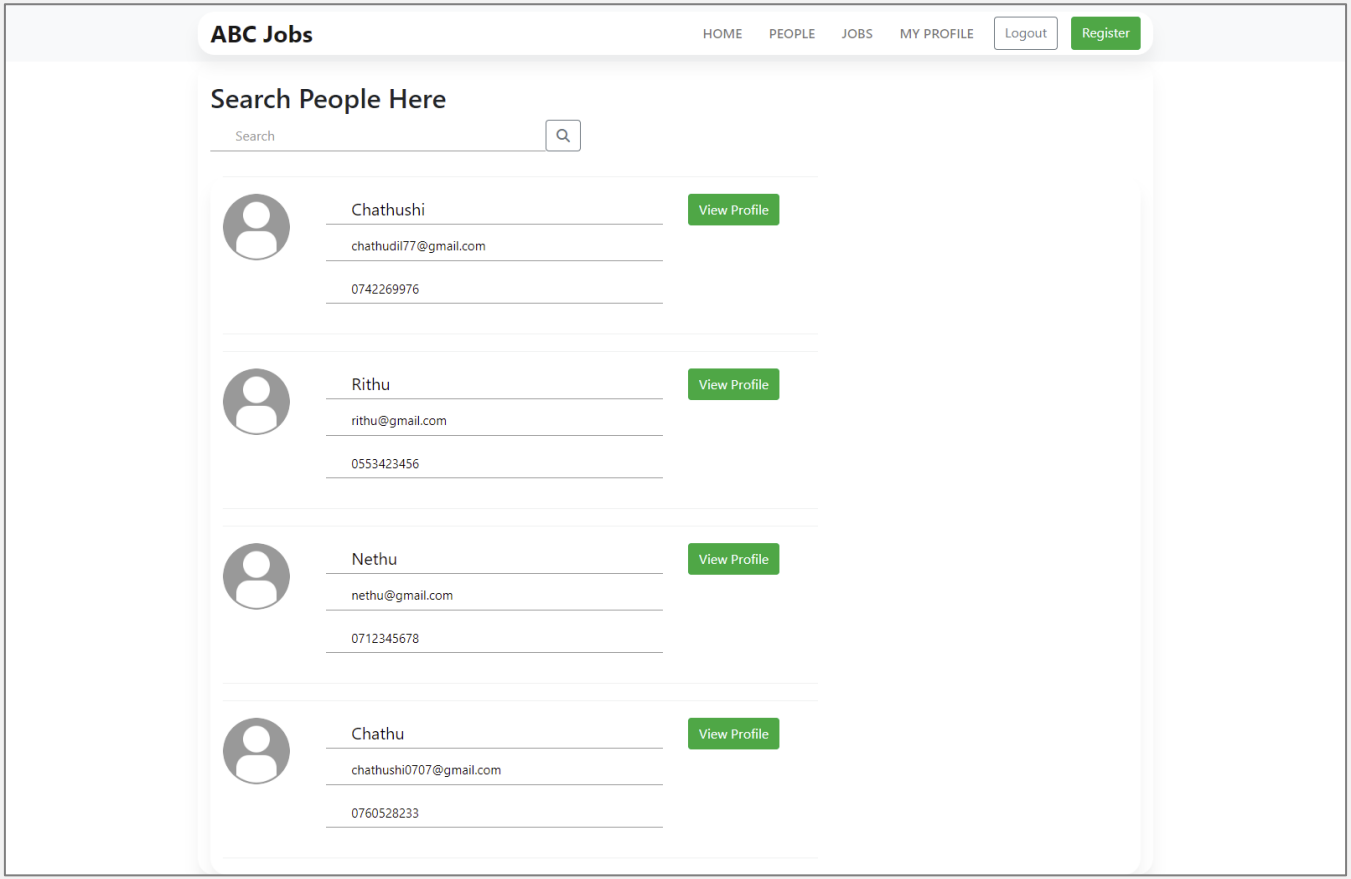
User Profile Page



19. Project Results

LITHAN

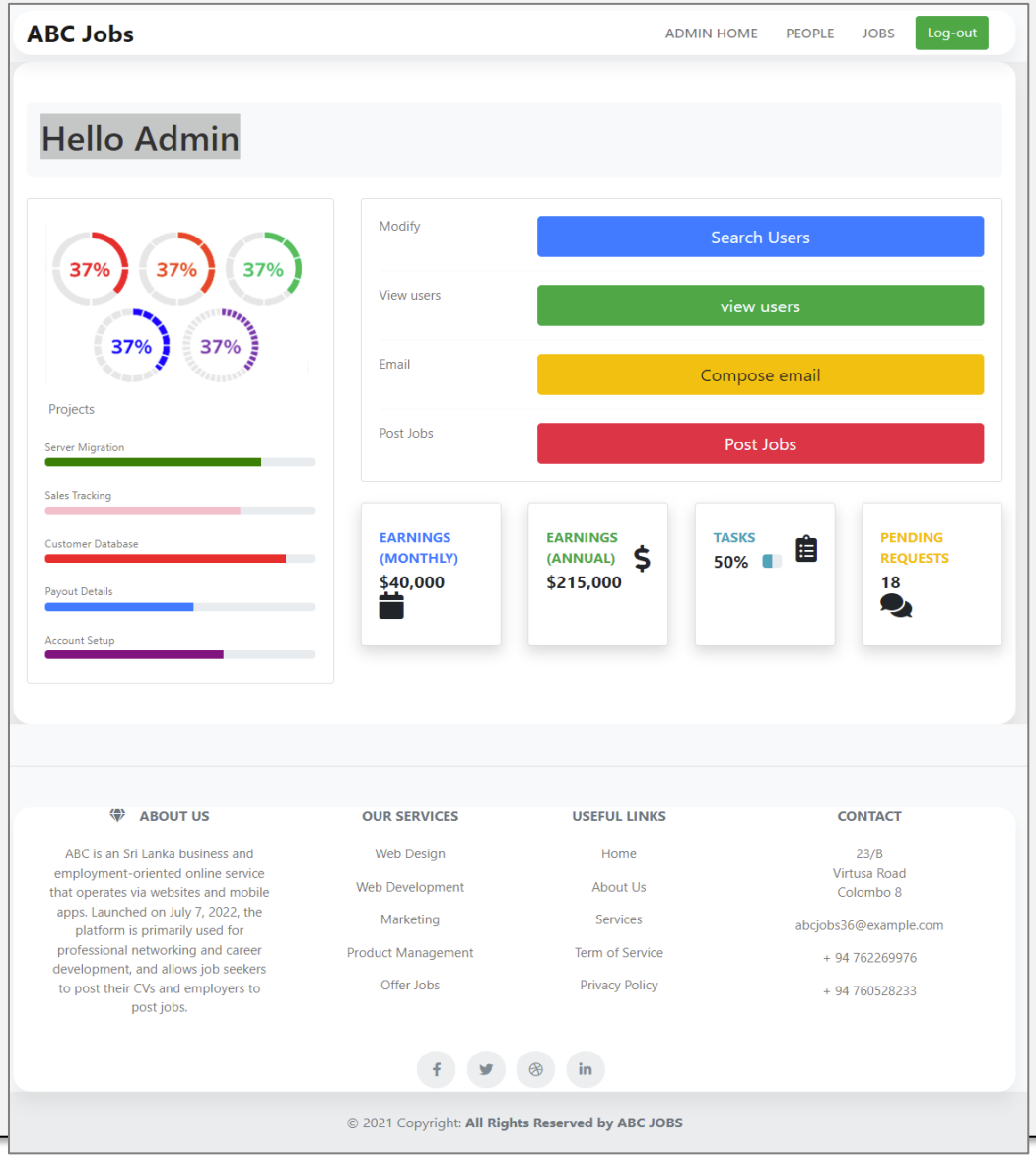
Search User Page



19. Project Results

LITHAN

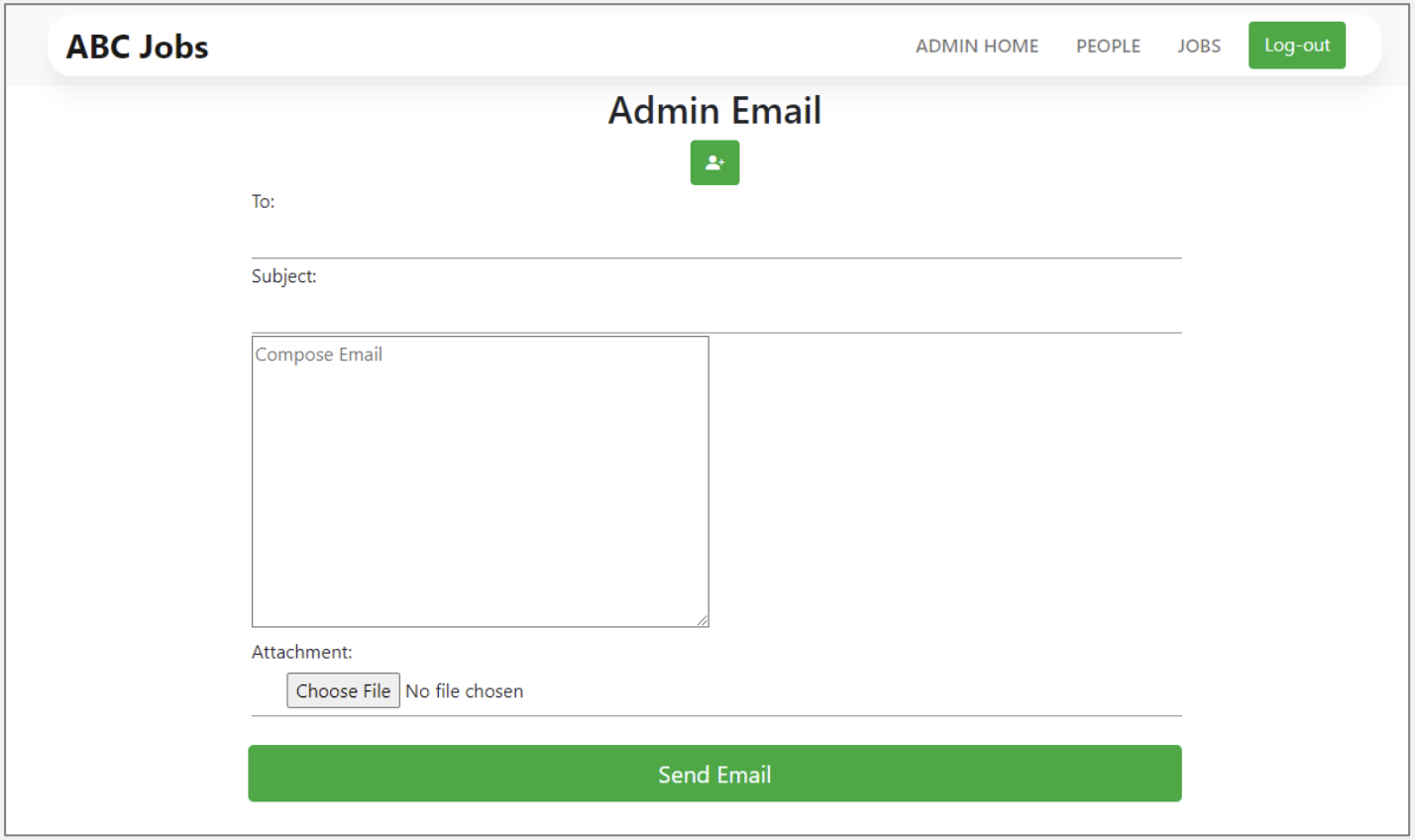
Admin Profile Page



19. Project Results

LITHAN

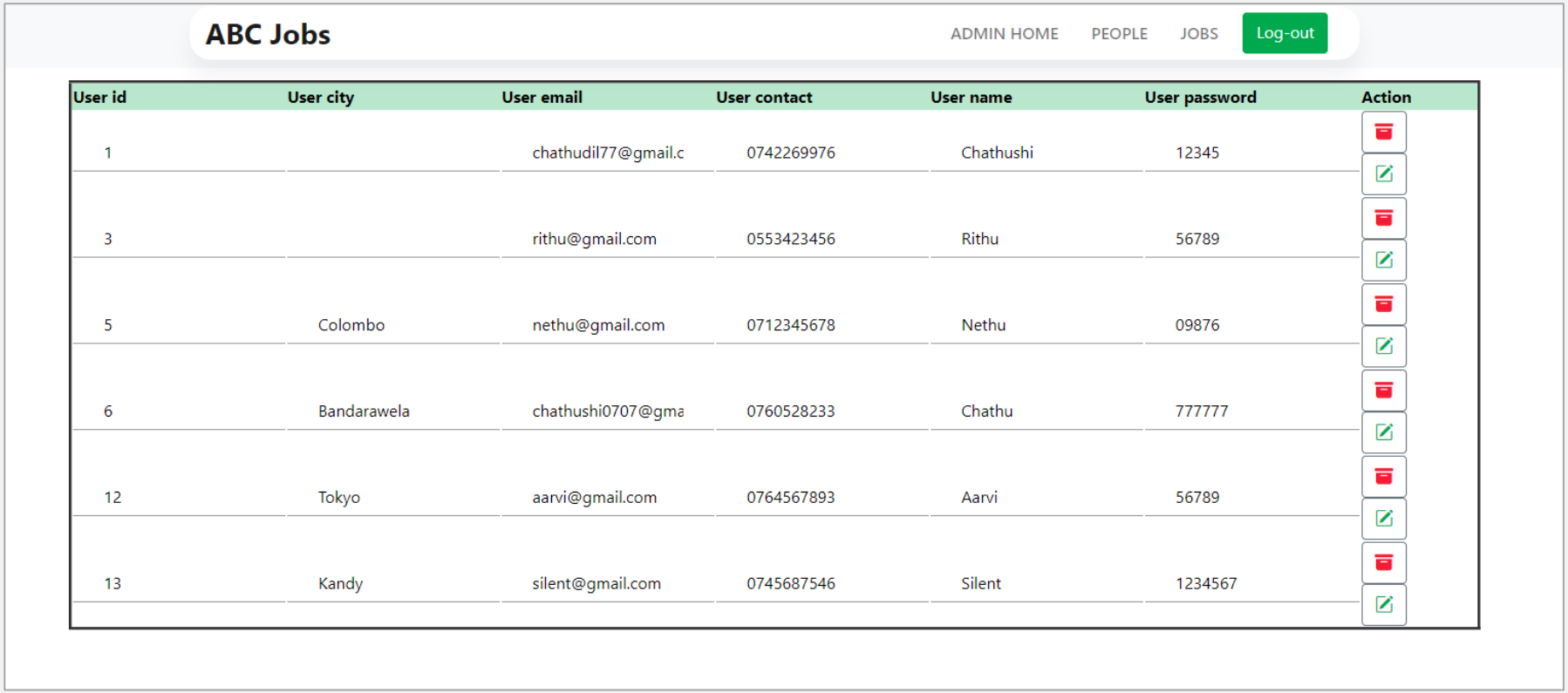
Bulk Email



19. Project Results

LITHAN

View User (Admin)



User id	User city	User email	User contact	User name	User password	Action
1		chathudil77@gmail.c	0742269976	Chathushi	12345	
3		rithu@gmail.com	0553423456	Rithu	56789	
5	Colombo	nethu@gmail.com	0712345678	Nethu	09876	
6	Bandarawela	chathushi0707@gma	0760528233	Chathu	777777	
12	Tokyo	aarvi@gmail.com	0764567893	Aarvi	56789	
13	Kandy	silent@gmail.com	0745687546	Silent	1234567	

20. Proposed Improvements

LITHAN

❑ List of Improvements

- Creating more better code and not repeating that can be reused
- Consistent with the naming convention
- Implement other functionalities