**IBM**

SUMMER INTERNSHIP REPORT

ON STREAM ADVISOR

**Submitted by**

Paras Sharma (R114216017)

**Under the guidance of**

Mr. Manjit S Sodhi Mr. Gagan Deep Singh

Technical Lead Assistant Professor

IBM UPES



**Department of Informatics**

**School of Computer Science**

**University of Petroleum & Energy Studies**

**Bidholi, Via Prem Nagar, Dehradun, UK**

**January – 2019**

**Table of Contents**

[DECLARATION 3](#_Toc20260680)

[ACKNOWLEDGEMENT 4](#_Toc20260681)

1. PROJECT OVERVIEW
   1. PROBLEM STATEMENT
   2. DEFINING THE PROBLEM

1. USE CASE DIAGRAM
2. ACTIVITY DIAGRAM
3. SET UP INSTRUCTIONS
4. INTERACTING WITH PORTAL
5. METHODOLOGY
6. CODE & DEMO
7. OUTPUT SCREEN
8. CONCLUSION

# DECLARATION

I hereby declare that this submission is my own and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for award of any other Degree or Diploma of the University or other Institute of Higher learning, except where due acknowledgement has been made in the text.

PARAS SHARMA

(Enroll No. R114216017)

# ACKNOWLEDGEMENT

I would like to express my deepest appreciation to all those who provided me the possibility to complete this report. I am very thankful to IBM for giving me the opportunity to undertake the summer internship. I wish to express my sincere gratitude to Mr. Manjit S Sodhi, Technical Lead, IBM for providing me an opportunity to do my project work on “Stream Advisor”. This project bears an imprint of many people. I sincerely thank my project guide Mr. Gagan Deep Singh, Assistant Professor, UPES Dehradun for guidance and encouragement for carrying out this project work.

I have to appreciate the guidance given by other supervisors as well as other seniors in my team.

## PROJECT OVERVIEW

Stream Advisor is basically a helpful tool for students who have just completed their schooling and want to pursue engineering in the field of their choice. It basically consists of psychometric test which is used for finding the interest of a student in a particular branch. In this project, we have created an online portal on which the user has to create his account in the Sign Up page by entering the details required by page. After creating the account, the user Login to the application with his/her username and password. On successful login, an instructions page appears with the guidelines for the test. The user clicks on “Start Test” button and takes the test. The test contains questions related to different branches of engineering and the user has three options to answer the question.

The questions actually judge the interest and skills of the candidate in different engineering branches based on the option he selects. Once the user answers all the questions and submit the test, a graph is displayed which depicts which engineering branch is most suitable for the user.

Hence, Stream Advisor which we can also call as Engineering Branch Selector helps to uncover following benefits:

* The best suited engineering branch for us
* Our key personality traits
* Strengths and Weaknesses
* Our Learning Patterns, Study Habits and Professional Interests
* Natural Inclination for Further Studies and Future Jobs

This Application is suitable for following which includes as follows:

* 11th-12th grade students aspiring for career in Engineering
* Engineering College Entrants
* Fresh Graduates/Under-Graduates in Engineering Courses

## PROBLEM STATEMENT

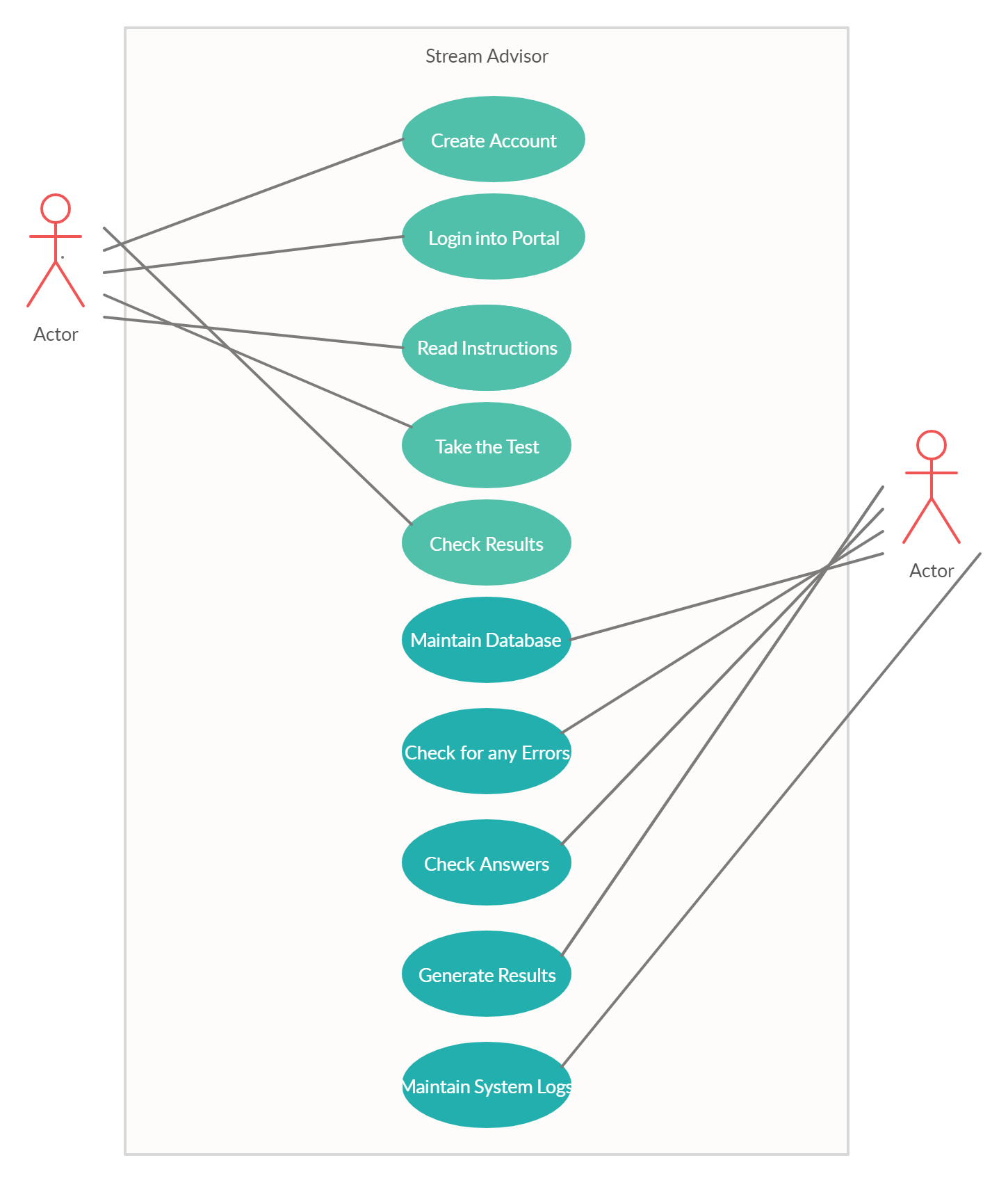
To develop an online portal using Web Technology based on Decision Support System in which a student chooses the appropriate engineering branch based upon his/her skillset of interests.

## 1.2. DEFINING THE PROBLEM

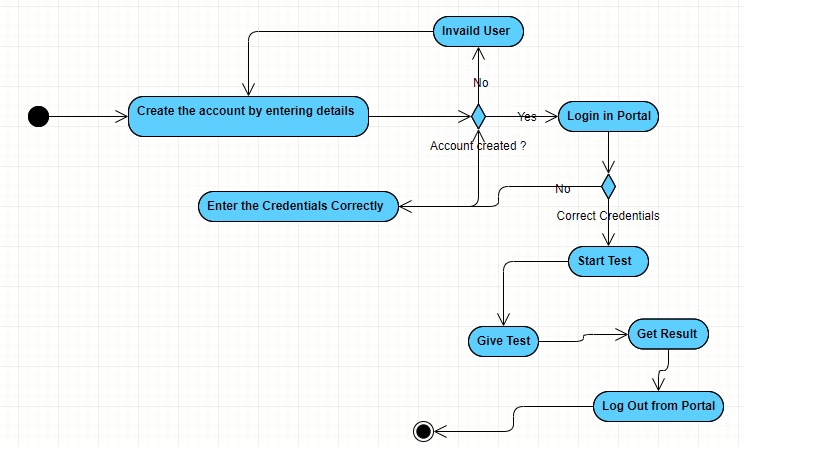
For building up the online portal, we need following tools and technologies which are given as follows:

* PostgreSQL v11.0 ( For creating Database tables for following – Options, Questions, and User Details )
* TomCat Server v9.0 ( For connecting our project to web application)
* Hibernate ( For Database Connectivity)
* Eclipse Oxygen ( For Java EE)

## USE CASE DIAGRAM



## ACTIVITY DIAGRAM

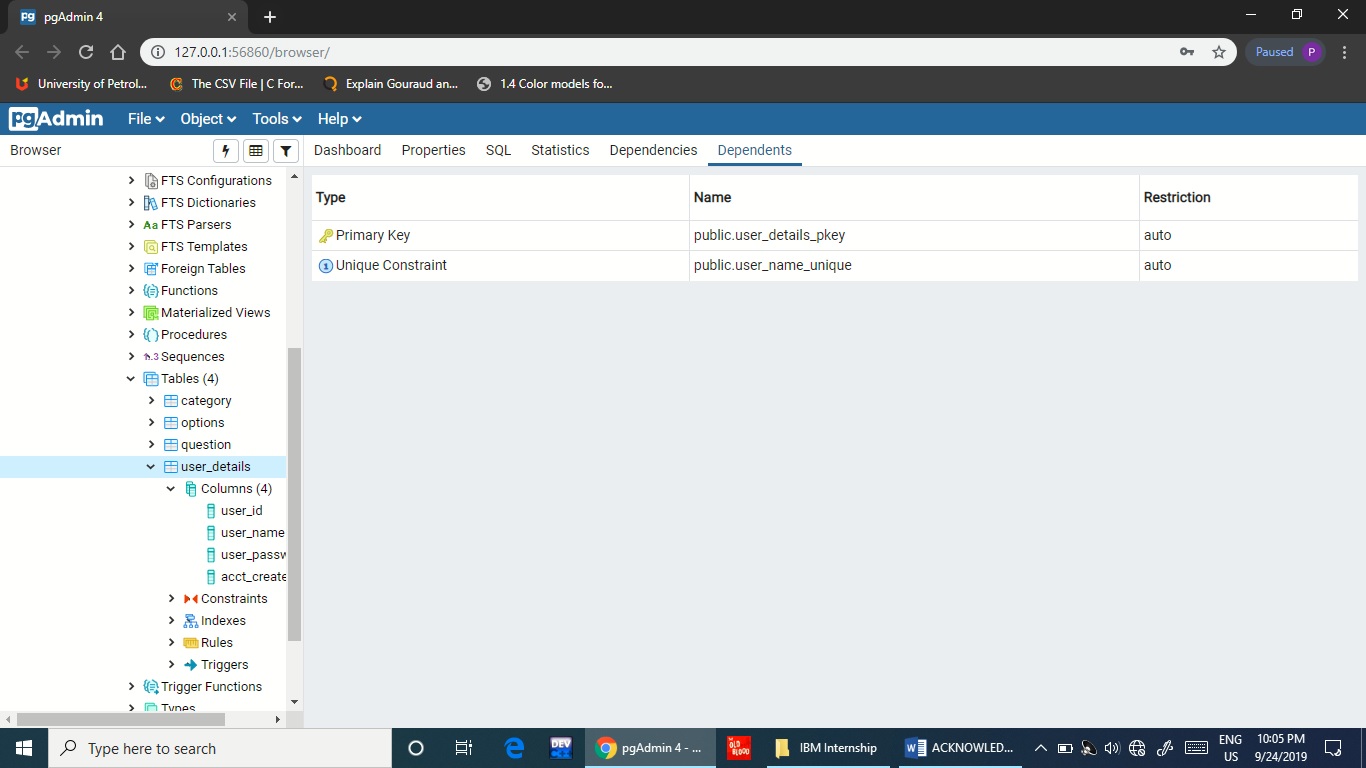


## SET UP INSTRUCTIONS

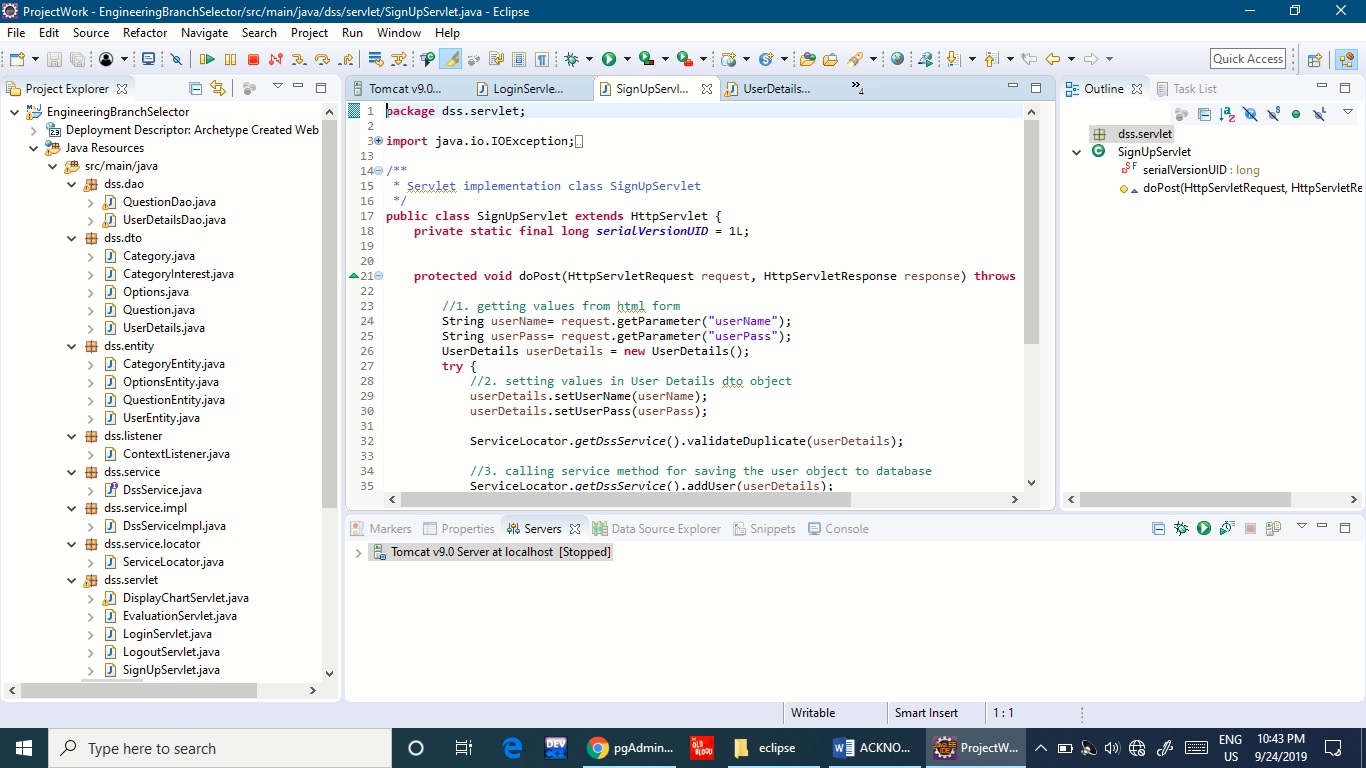
**STEP 1:** Install the PostgreSQL and define all the tables required for creating the details for portal.

## SETUP FIREBASE

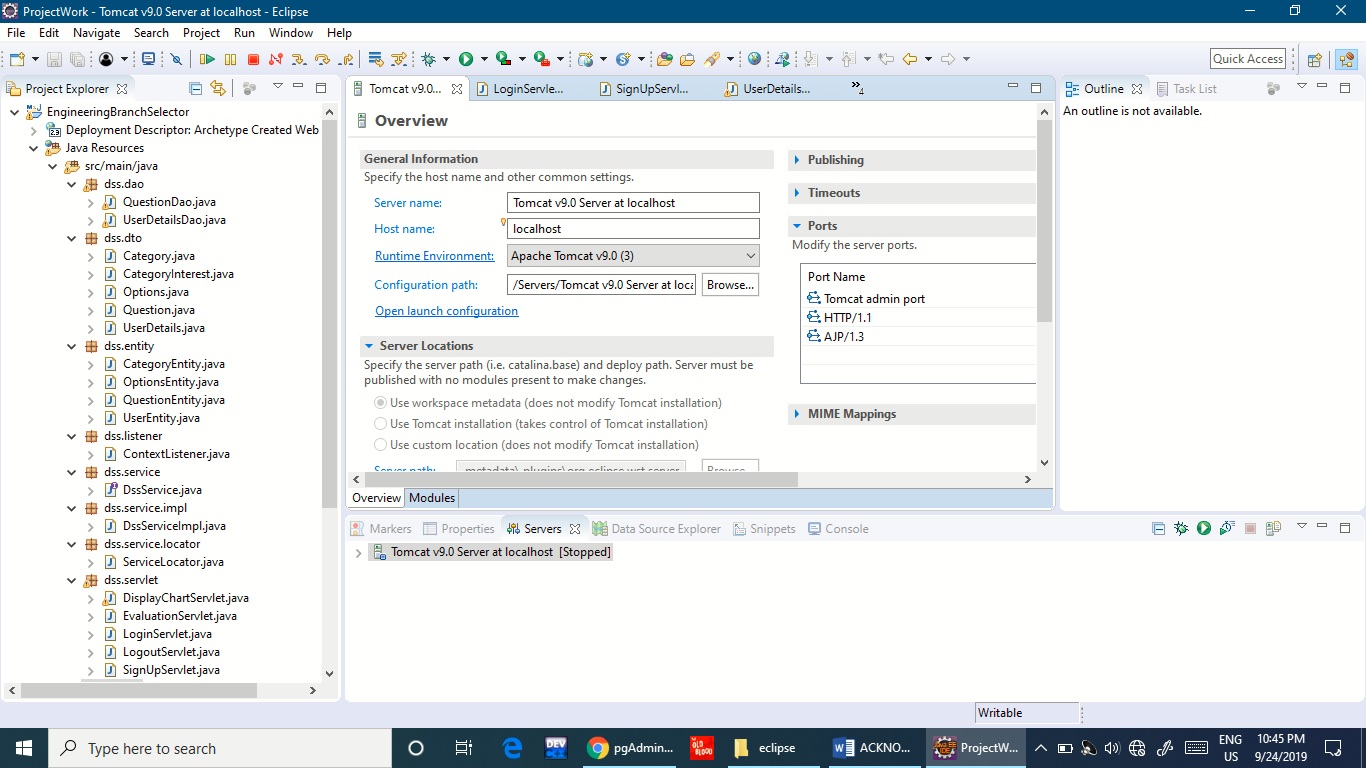
**STEP 2:** Open pgAdmin 4 application and enter the password for connecting to PostgreSQL server.



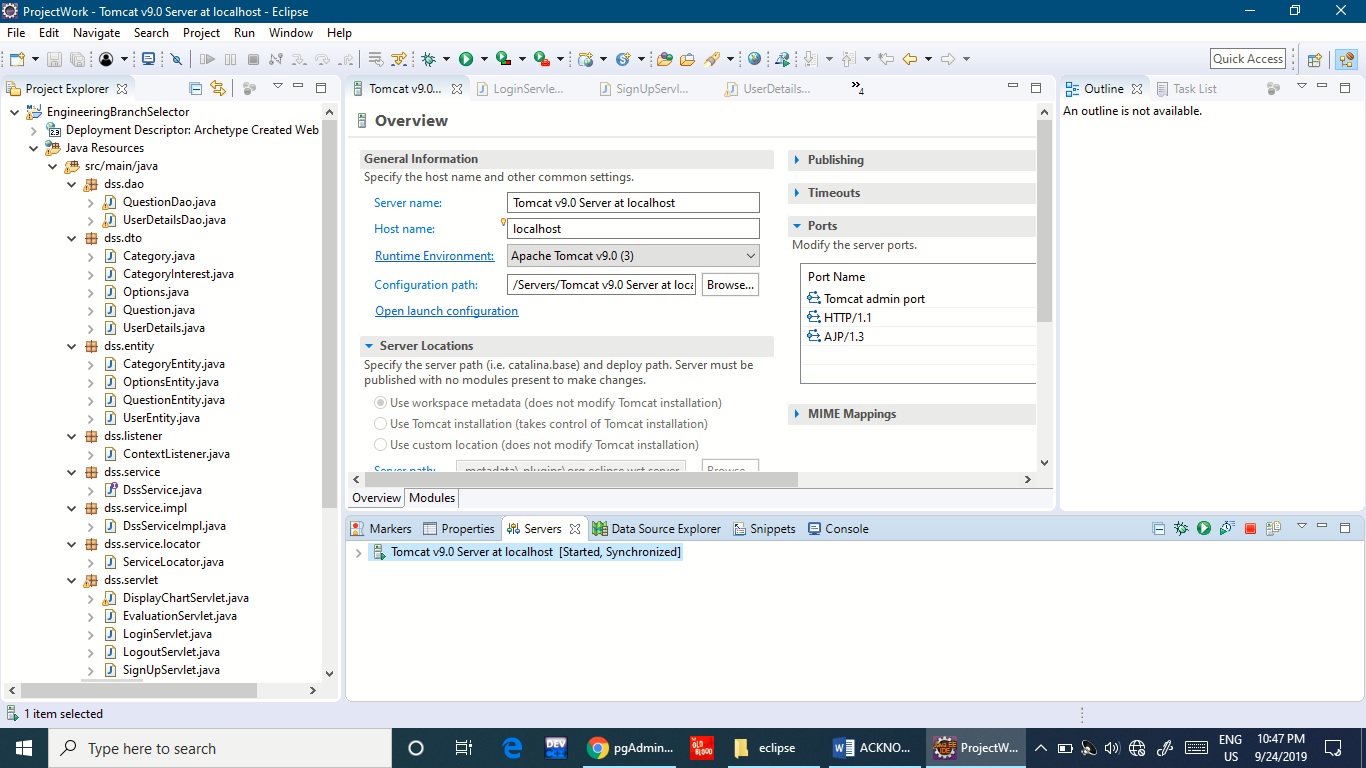
**STEP 3:** Open the Eclipse Oxygen IDE for Java EE where your project is present Import the settings from your existing project and open it.



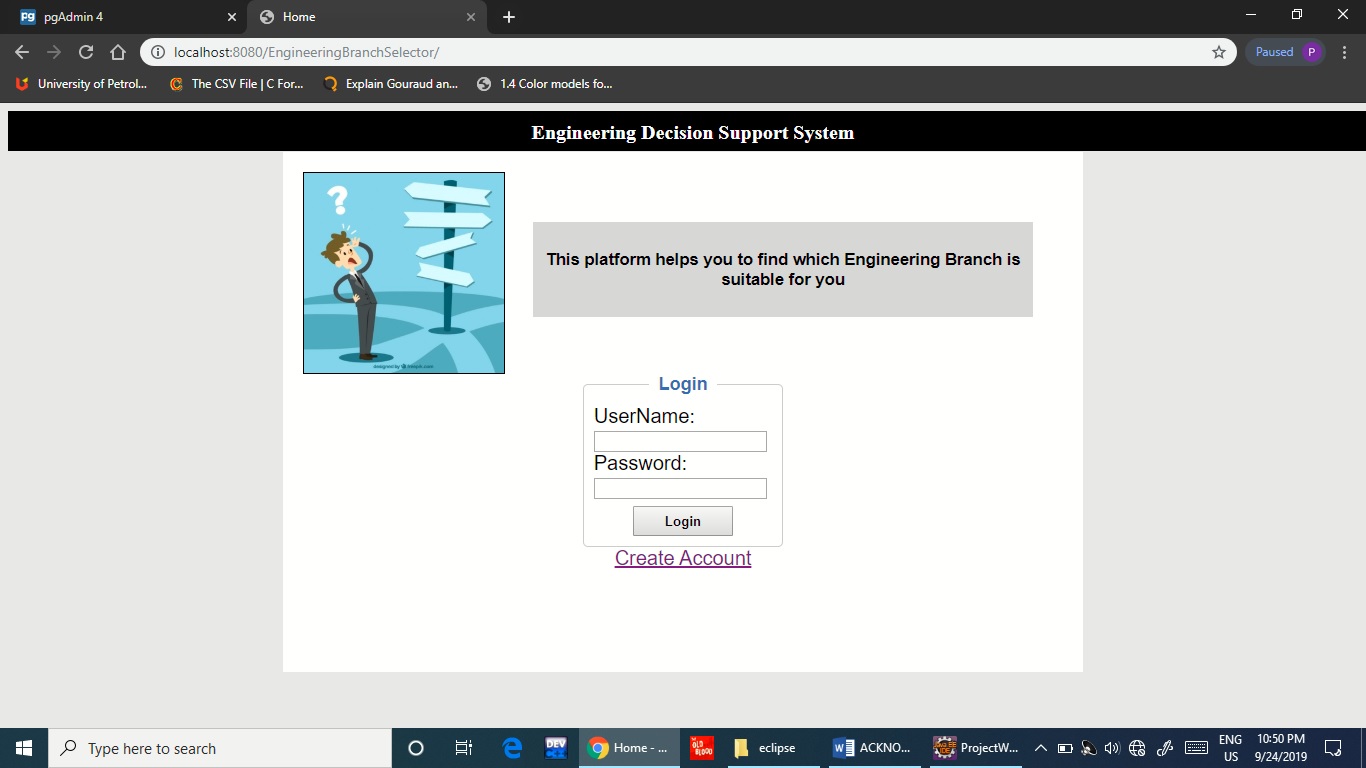
**STEP 4:** Install the TomCat Server v9.0 for connecting the project through web page to manage the portal we have created for our project.



**STEP 5:** After installing the server, configure your project with the server and start the server.

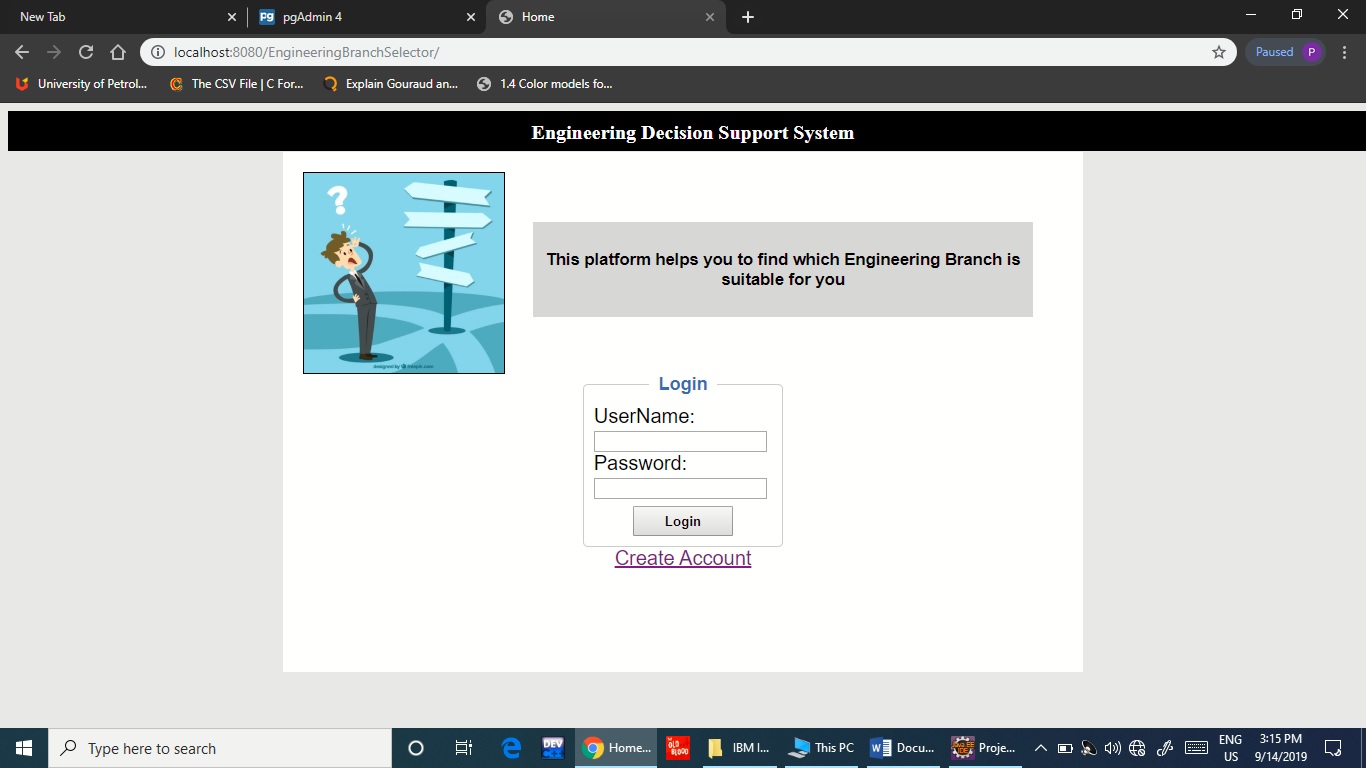


**STEP 6:** Open up any web browser and type the following URL – “localhost: 8080/ EngineeringBranchSelector/

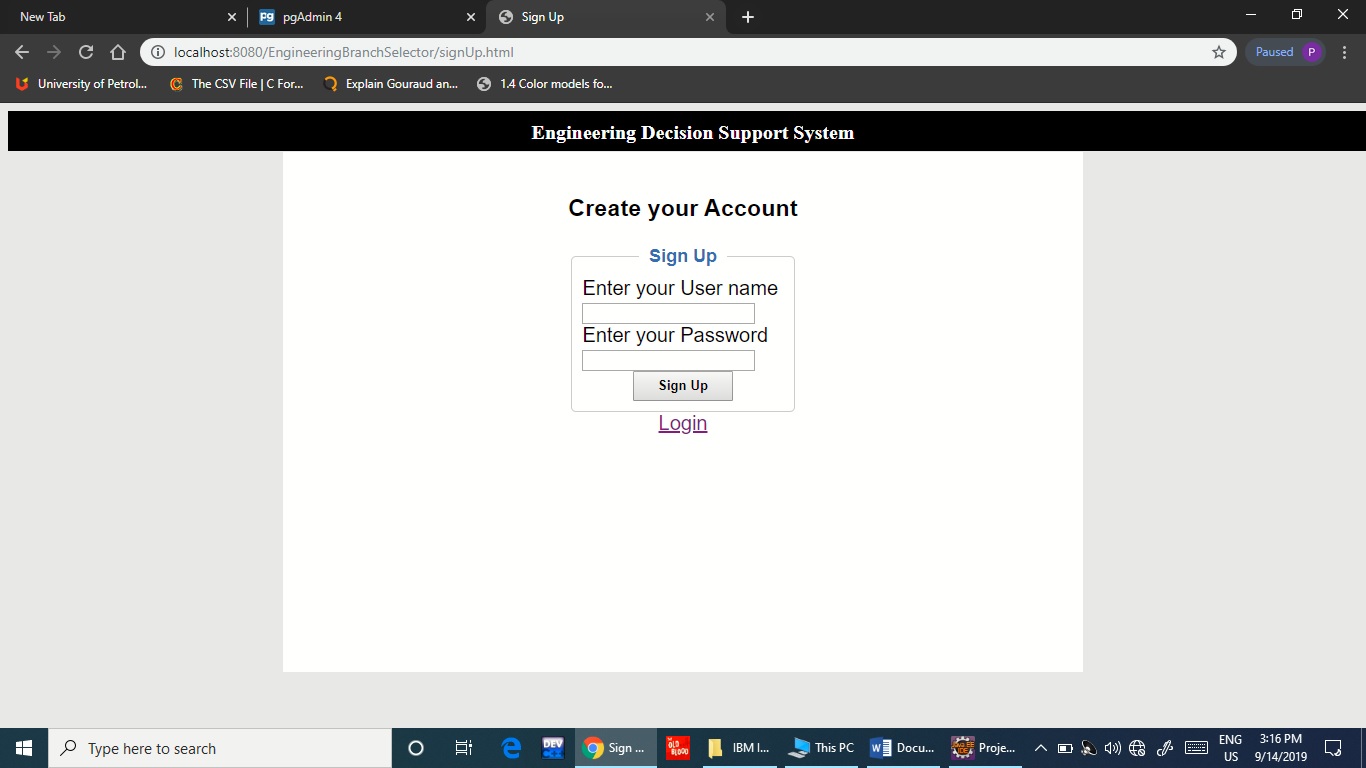


**INTERACTING WITH APPLICATION**

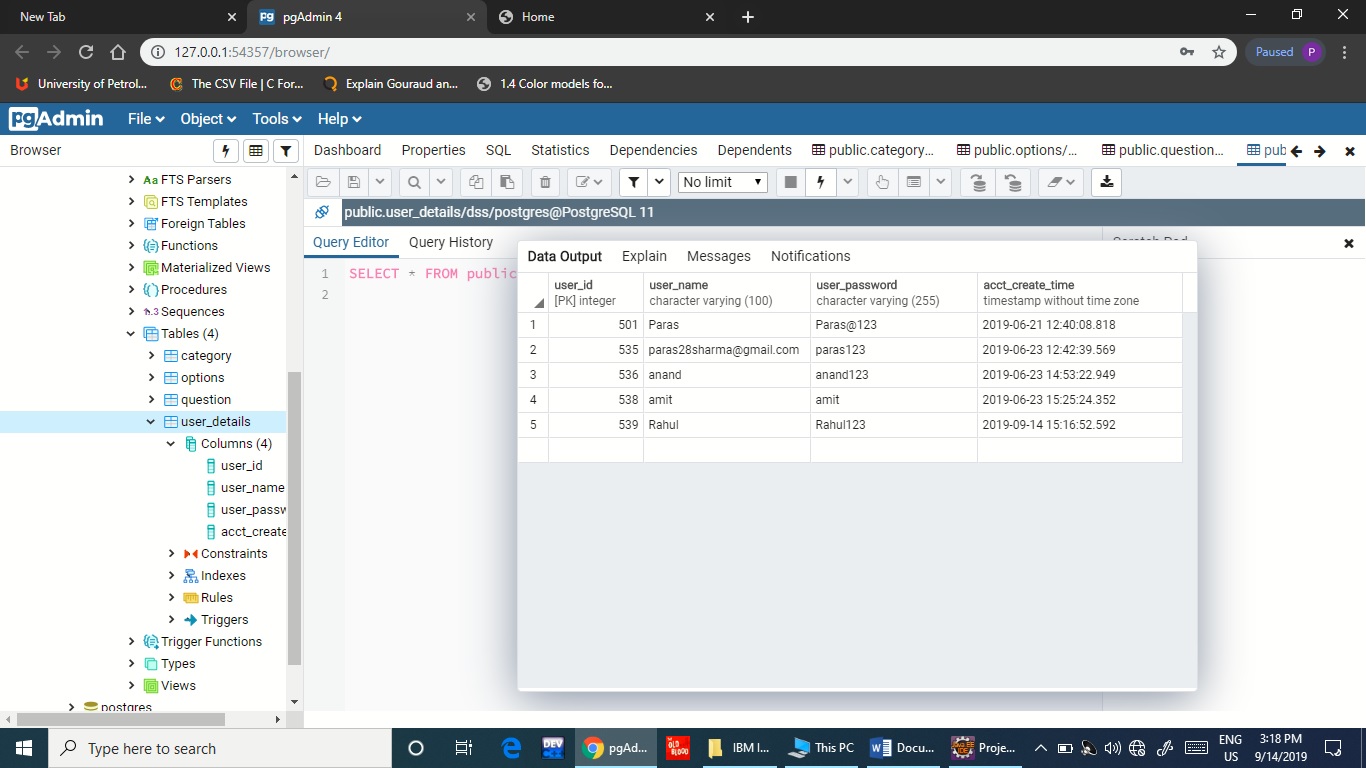
1. Suppose a user has to access the Engineering Branch Selector to know about his interest in various engineering fields. First he would access the home page.



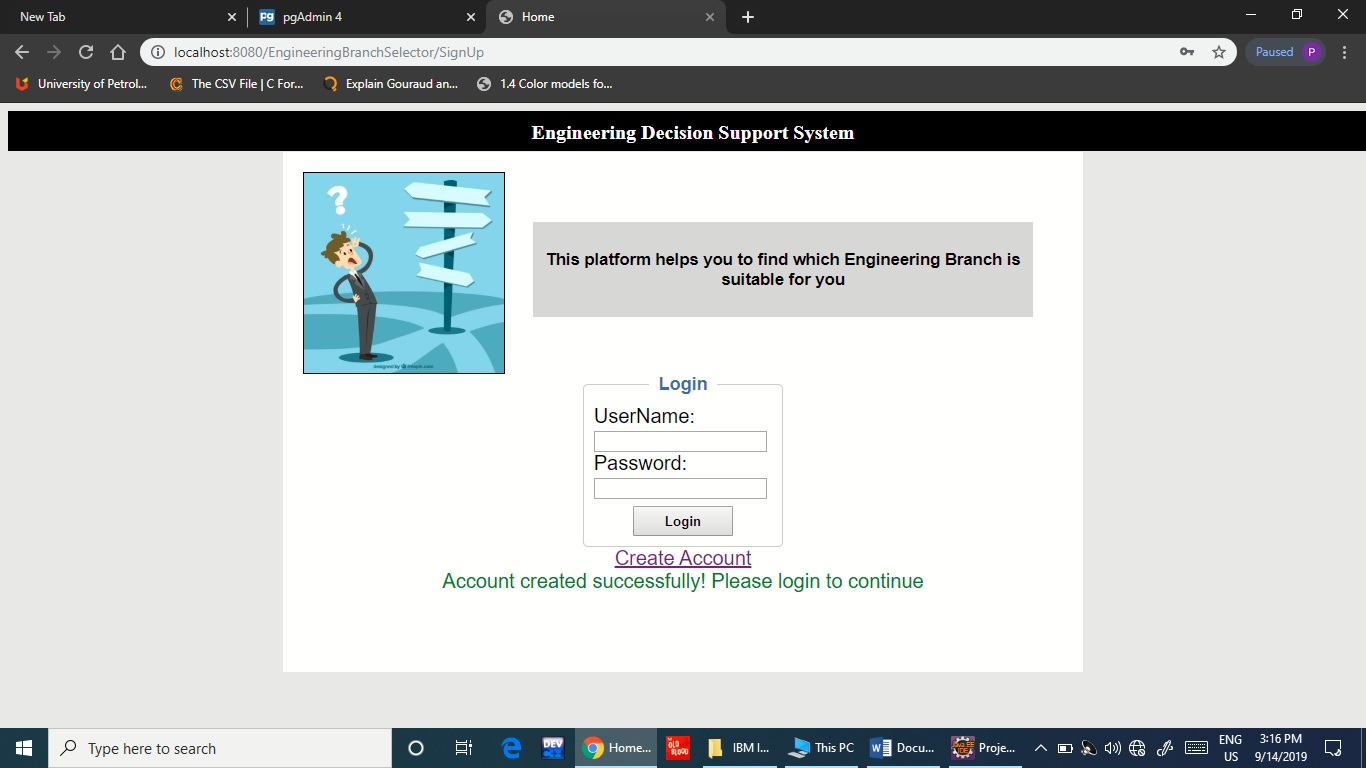
1. If the user is new to the system, he/she has to register with details to create account.



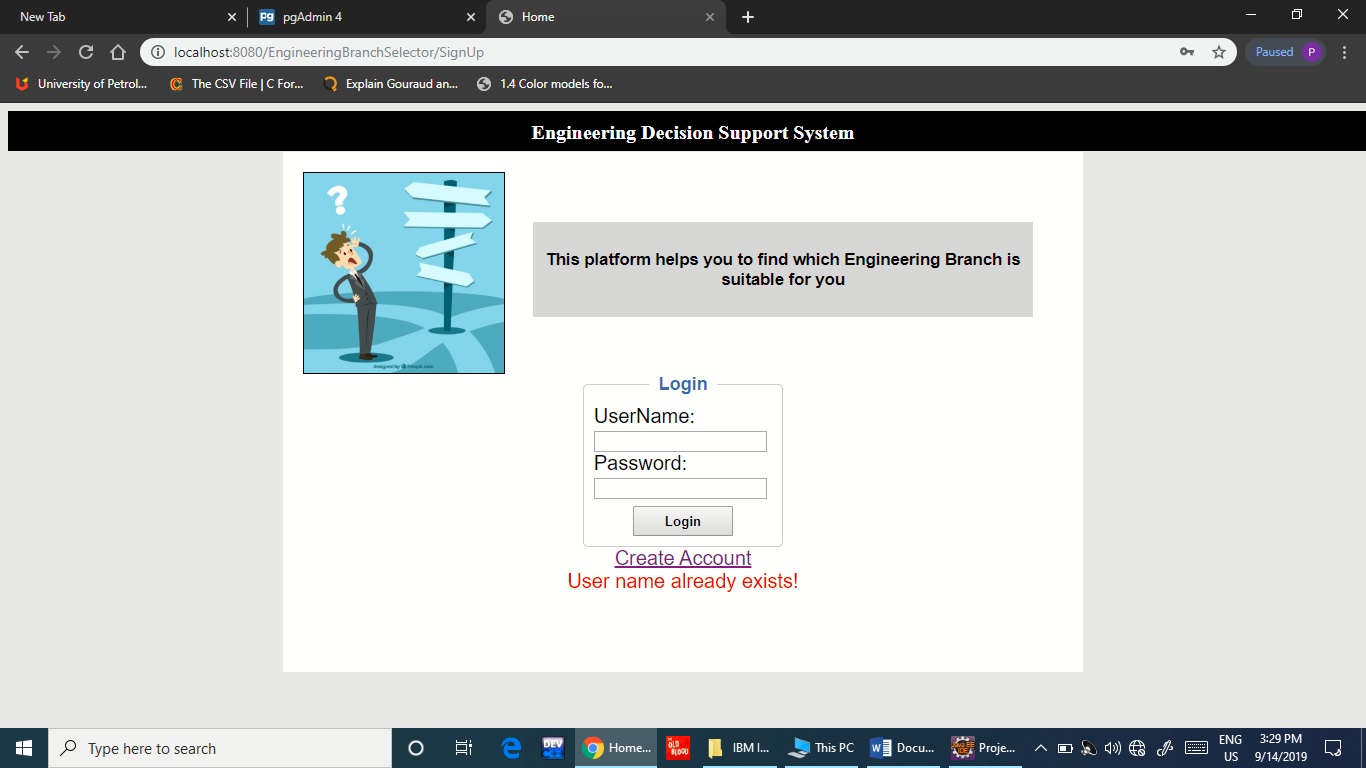
1. After entering the details, the user details will be entered in the PostgreSQL Database.



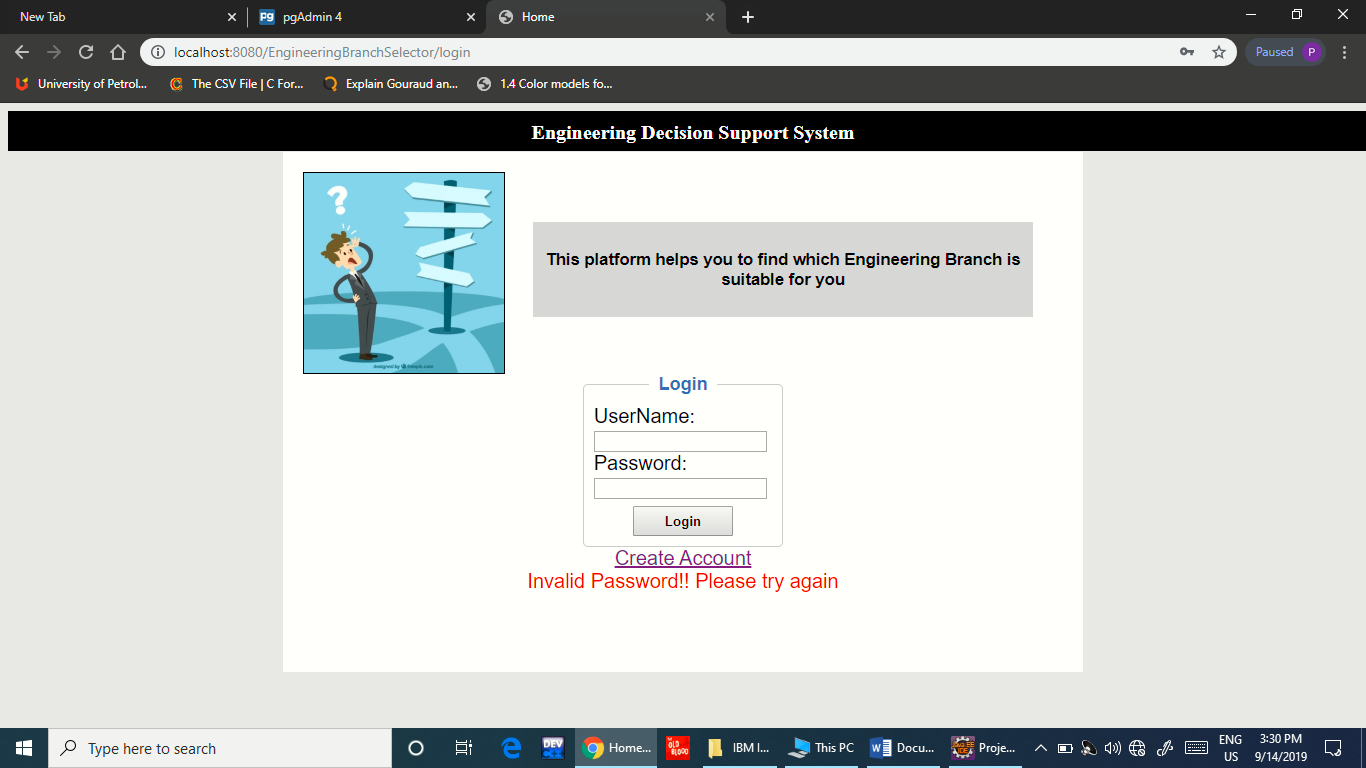
1. After the user has provided all the details in the form page, a message will be displayed “Account Created Successfully!!!”



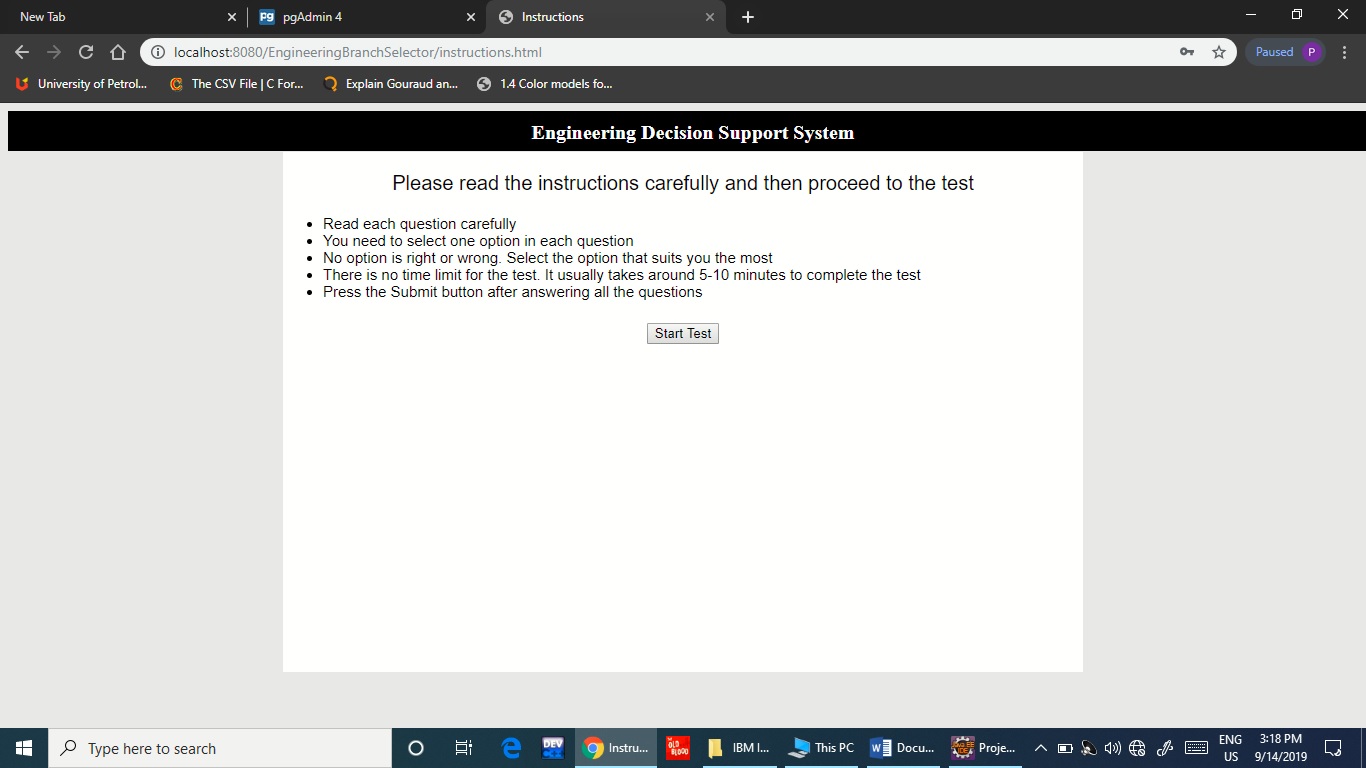
1. If an already registered user tries to create account, then the following message is displayed – “User Already Exists”



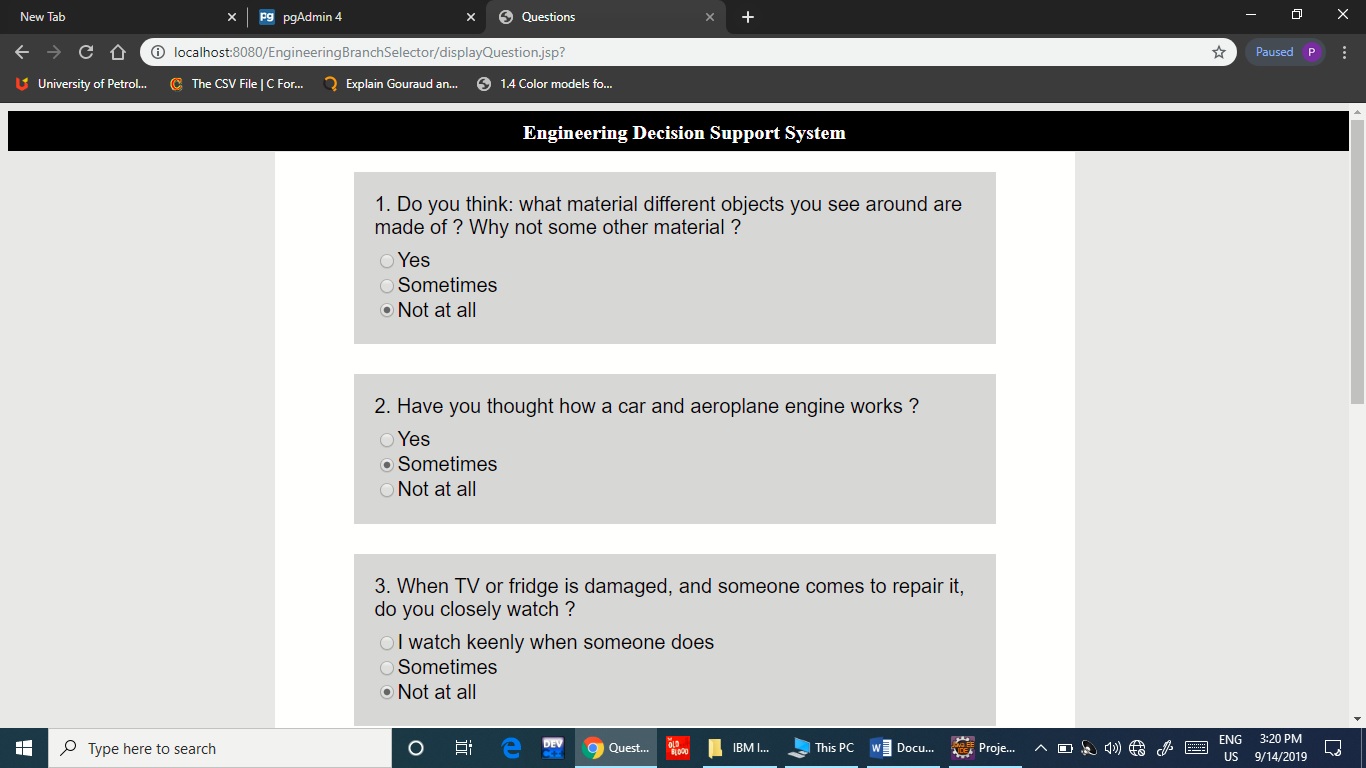
1. If the user enters the incorrect password, then the following message is displayed – “Invalid Password”



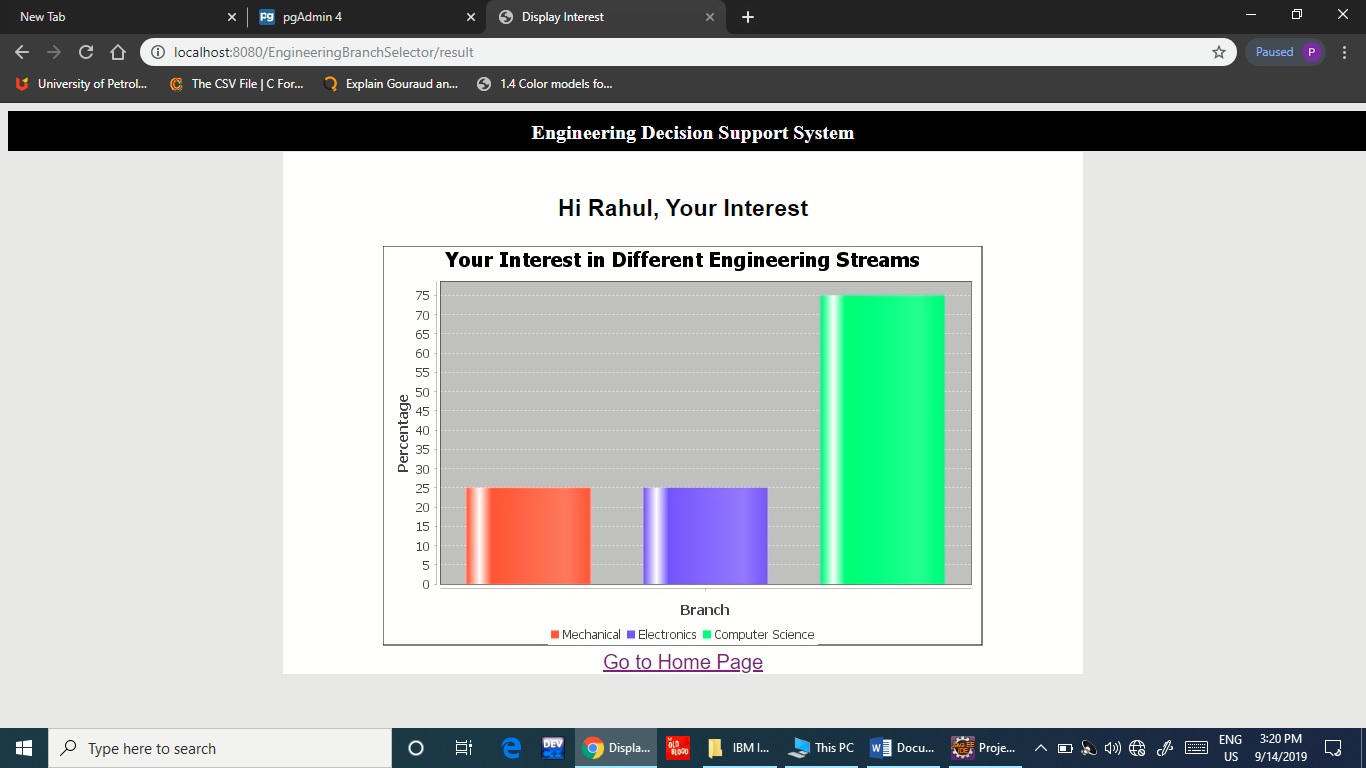
1. After the user enters the correct credentials, the link to instructions of test is opened.



1. After clicking the button for test, the test opens in which the user has to answer MCQ questions based upon his/her interest.



1. After completing the test, the results are obtained and displayed according to the options chosen by user.



**METHODOLOGY**

* The project that have completed during my internship period is a psychometric test used for finding the interest of a student in a particular engineering branch.
* First of all, the user has to create his account in the Sign up Page.
* Then the user Login to the application with his username and password.
* On Successful Login, an instructions page appears with the guidelines for the test.
* On clicking the “Start Test” button, the test starts for the user.
* Test contains different questions where in each question, user has to select one option.
* The Questions actually help to identify interest and relevant skills of the candidate in different engineering branches based on the option he selects.
* Once the candidate answers all the questions and submits the test, Result graph displaying the interest in particular engineering branches is shown to the user.

## CODE & DEMO

* First of all, I have installed all the necessary software that were required to build the system.
* On pgAdmin (PostgreSQL), I created three tables namely – Category, Options, Question, User\_Details. These tables are used in for connecting the server with Java using Hibernate.
* This project is a type of Maven Project. In this project, I have made around 8 packages whose details are as follows:
  + - dss.dao
    - dss.dto
    - dss.entity
    - dss.listner
    - dss.service.impl
    - dss.service.locator
    - dss.servlet
    - dss.util

1. dss.dao

* QuestionDao.java

package dss.dao;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

import org.hibernate.Query;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import dss.dto.Category;

import dss.dto.Options;

import dss.dto.Question;

import dss.entity.CategoryEntity;

import dss.entity.OptionsEntity;

import dss.entity.QuestionEntity;

import dss.service.DssService;

import dss.service.locator.ServiceLocator;

import dss.util.HibernateUtil;

public class QuestionDao {

public Category getCategorybyId(Integer categoryId) {

SessionFactory sessionFactory = null;

Session session = null;

try

{

sessionFactory = HibernateUtil.getSessionFactory();

session = sessionFactory.openSession();

session.beginTransaction();

Query catQuery = session.createQuery(" from CategoryEntity where categoryId = ?1");

catQuery.setParameter(1,categoryId);

CategoryEntity categoryEntity = (CategoryEntity) catQuery.getSingleResult();

Category category = new Category();

category.setCategoryId(categoryEntity.getCategoryId());

category.setCategoryName(categoryEntity.getCategoryName());

return category;

}

catch(Exception e){

e.printStackTrace();

throw e;

}

finally

{

if(session!=null)

{

session.close();

}

}

}

/\*public static void main(String[] args) throws Exception {

new QuestionDao().getQuestionList();

}\*/

private static final int QUESTIONS\_TO\_DISPLAY\_PER\_CATEGORY = 2;

public List<Integer>loadCategoryIds(){

SessionFactory sessionFactory = null;

Session session = null;

try

{

sessionFactory = HibernateUtil.getSessionFactory();

session = sessionFactory.openSession();

session.beginTransaction();

Query catQuery = session.createQuery("select categoryId from CategoryEntity");

List<Integer>catList = catQuery.getResultList();

return catList;

}

catch(Exception e){

e.printStackTrace();

throw e;

}

finally

{

if(session!=null)

{

session.close();

}

}

}

public List<Question> getQuestionList()

{

SessionFactory sessionFactory = null;

Session session = null;

try

{

List<Question>questionList = new ArrayList<>();

sessionFactory = HibernateUtil.getSessionFactory();

session = sessionFactory.openSession();

session.beginTransaction();

List<Integer>questionIdList = new ArrayList<>();

DssService dssService = ServiceLocator.getDssService();

List<Integer>catList = dssService.getCategoryIds();

for (Integer category : catList) {

Query questQuery = session.createQuery("select questionId from QuestionEntity where category\_Id=?1");

questQuery.setParameter(1, category);

List<Integer>list = questQuery.getResultList();

Collections.shuffle(list);

for (int i=0;i<QUESTIONS\_TO\_DISPLAY\_PER\_CATEGORY;i++) {

questionIdList.add(list.get(i));

}

}

Collections.shuffle(questionIdList);

System.out.println("question list: "+questionIdList);

System.out.println("question list size: "+questionIdList.size());

Query questionListQuery = session.createQuery("from QuestionEntity where questionId in ?1");

questionListQuery.setParameter(1, questionIdList);

List<QuestionEntity>quesList = questionListQuery.getResultList();

for (QuestionEntity questionEntity : quesList) {

List<Options> optionsList = new ArrayList<Options>();

Query optionListQuery = session.createQuery("from OptionsEntity where question\_Id in ?1");

optionListQuery.setParameter(1, questionEntity.getQuestionId());

List<OptionsEntity>optionList = optionListQuery.getResultList();

for (OptionsEntity optionEntity : optionList) {

Options options = new Options();

options.setOptionId(optionEntity.getOptionId());

options.setOptionName(optionEntity.getOptionName());

options.setQuestionId(optionEntity.getQuestionEntity().getQuestionId());

optionsList.add(options);

}

System.out.println(questionEntity.getCategoryEntity().getCategoryName() + " - " + questionEntity.getQuestion());

Question question = new Question();

question.setQuestionId(questionEntity.getQuestionId());

question.setQuestionName(questionEntity.getQuestion());

question.setOptionsList(optionsList);

question.setCategoryId(questionEntity.getCategoryEntity().getCategoryId());

questionList.add(question);

}

return questionList;

}

catch(Exception e){

e.printStackTrace();

return null;

//throw new Exception(e);

}

finally{

if(session!=null)

{

session.close();

}

}

}

}

* UserDetailsDao.java

package dss.dao;

import java.sql.Timestamp;

import java.util.Calendar;

import java.util.List;

import org.hibernate.Query;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import dss.dto.UserDetails;

import dss.entity.UserEntity;

import dss.util.HibernateUtil;

public class UserDetailsDao {

/\*public static void main(String[] args) {

UserDetailsDao dao = new UserDetailsDao();

UserDetails userDetails = new UserDetails();

userDetails.setUserName("Paras");

userDetails.setUserPass("Paras@123");

dao.saveUser(userDetails);

}\*/

public UserDetails authenticateUser(String userName) throws Exception{

SessionFactory sessionFactory = null;

Session session = null;

UserDetails userDetails = null;

try

{

sessionFactory = HibernateUtil.getSessionFactory();

session = sessionFactory.openSession();

session.beginTransaction();

Query catQuery = session.createQuery("from UserEntity where userName = ?1");

catQuery.setParameter(1, userName);

List<UserEntity> userEntityList = catQuery.getResultList();

if(userEntityList != null && userEntityList.size() > 0) {

UserEntity userEntity = userEntityList.get(0);

userDetails = new UserDetails();

userDetails.setUserName(userEntity.getUserName());

userDetails.setUserPass(userEntity.getUserPass());

}

return userDetails;

}

catch(Exception e){

e.printStackTrace();

throw e;

}

finally

{

if(session!=null)

{

session.close();

}

}

}

public void saveUser(UserDetails userDetails){

SessionFactory sessionFactory = null;

Session session = null;

try

{

sessionFactory = HibernateUtil.getSessionFactory();

session = sessionFactory.openSession();

session.beginTransaction();

UserEntity userentity = new UserEntity();

userentity.setUserName(userDetails.getUserName());

userentity.setUserPass(userDetails.getUserPass());

Calendar calendar = Calendar.getInstance();

Timestamp now = new Timestamp(calendar.getTime().getTime());

userentity.setAccCreateTime(now);

session.save(userentity);

session.getTransaction().commit();

}

catch(Exception e){

e.printStackTrace();

throw e;

}

finally

{

if(session!=null)

{

session.close();

}

}

}

}

1. dss.dto

* Category.java

package dss.dto;

public class Category {

private Integer categoryId;

private String categoryName;

public Integer getCategoryId() {

return categoryId;

}

public void setCategoryId(Integer categoryId) {

this.categoryId = categoryId;

}

public String getCategoryName() {

return categoryName;

}

public void setCategoryName(String categoryName) {

this.categoryName = categoryName;

}

}

* CategoryInterest.java

package dss.dto;

public class CategoryInterest {

private Integer categoryId;

private String categoryName;

private float interest;

public Integer getCategoryId() {

return categoryId;

}

public void setCategoryId(Integer categoryId) {

this.categoryId = categoryId;

}

public float getInterest() {

return interest;

}

public void setInterest(float interest) {

this.interest = interest;

}

public String getCategoryName() {

return categoryName;

}

public void setCategoryName(String categoryName) {

this.categoryName = categoryName;

}

}

* Options.java

package dss.dto;

public class Options {

private Integer optionId;

private String optionName;

private Integer questionId;

public Integer getOptionId() {

return optionId;

}

public void setOptionId(Integer optionId) {

this.optionId = optionId;

}

public String getOptionName() {

return optionName;

}

public void setOptionName(String optionName) {

this.optionName = optionName;

}

public Integer getQuestionId() {

return questionId;

}

public void setQuestionId(Integer questionId) {

this.questionId = questionId;

}

}

* Question.java

package dss.dto;

import java.util.List;

public class Question {

private Integer questionId;

private String questionName;

private List<Options> optionsList;

private Integer categoryId;

public Integer getQuestionId() {

return questionId;

}

public void setQuestionId(Integer questionId) {

this.questionId = questionId;

}

public String getQuestionName() {

return questionName;

}

public void setQuestionName(String questionName) {

this.questionName = questionName;

}

public List<Options> getOptionsList() {

return optionsList;

}

public void setOptionsList(List<Options> optionsList) {

this.optionsList = optionsList;

}

public Integer getCategoryId() {

return categoryId;

}

public void setCategoryId(Integer categoryId) {

this.categoryId = categoryId;

}

}

* UserDetails.java

package dss.dto;

public class UserDetails {

private String userName;

private String userPass;

public String getUserName() {

return userName;

}

public void setUserName(String userName) {

this.userName = userName;

}

public String getUserPass() {

return userPass;

}

public void setUserPass(String userPass) {

this.userPass = userPass;

}

}

1. dss.entity

* CategoryEntity.java

package dss.entity;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity

@Table(name = "category")

public class CategoryEntity {

@Id

@Column(name="category\_Id")

private Integer categoryId;

@Column(name="category\_Name")

private String categoryName;

public Integer getCategoryId() {

return categoryId;

}

public void setCategoryId(Integer categoryId) {

this.categoryId = categoryId;

}

public String getCategoryName() {

return categoryName;

}

public void setCategoryName(String categoryName) {

this.categoryName = categoryName;

}

}

* OptionsEntity.java

package dss.entity;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.OneToOne;

import javax.persistence.Table;

@Entity

@Table(name="options")

public class OptionsEntity {

@Id

@Column(name="option\_id")

private Integer optionId;

@Column(name="option\_name")

private String optionName;

@OneToOne

@JoinColumn(name="question\_id")

private QuestionEntity questionEntity;

public Integer getOptionId() {

return optionId;

}

public void setOptionId(Integer optionId) {

this.optionId = optionId;

}

public String getOptionName() {

return optionName;

}

public void setOptionName(String optionName) {

this.optionName = optionName;

}

public QuestionEntity getQuestionEntity() {

return questionEntity;

}

public void setQuestionEntity(QuestionEntity questionEntity) {

this.questionEntity = questionEntity;

}

}

* QuestionEntity.java

package dss.entity;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.OneToOne;

import javax.persistence.Table;

@Entity

@Table(name="question")

public class QuestionEntity {

@Id

@Column(name="question\_id")

private Integer questionId;

@Column(name="question\_name")

private String question;

@OneToOne

@JoinColumn(name = "category\_id")

private CategoryEntity categoryEntity;

public Integer getQuestionId() {

return questionId;

}

public void setQuestionId(Integer questionId) {

this.questionId = questionId;

}

public String getQuestion() {

return question;

}

public void setQuestion(String question) {

this.question = question;

}

public CategoryEntity getCategoryEntity() {

return categoryEntity;

}

public void setCategoryEntity(CategoryEntity categoryEntity) {

this.categoryEntity = categoryEntity;

}

}

* UserEntity.java

package dss.entity;

import java.sql.Timestamp;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.SequenceGenerator;

import javax.persistence.Table;

@Entity

@Table(name="user\_details")

public class UserEntity {

@Id

@Column(name="user\_id")

@SequenceGenerator(name="user\_id\_key",allocationSize=1,sequenceName="user\_id\_seq")

@GeneratedValue(generator="user\_id\_key",strategy=GenerationType.SEQUENCE)

private Integer userId;

@Column(name="user\_name")

private String userName;

@Column(name="user\_password")

private String userPass;

@Column(name="acct\_create\_time")

private Timestamp accCreateTime;

public Integer getUserId() {

return userId;

}

public void setUserId(Integer userId) {

this.userId = userId;

}

public String getUserName() {

return userName;

}

public void setUserName(String userName) {

this.userName = userName;

}

public String getUserPass() {

return userPass;

}

public void setUserPass(String userPass) {

this.userPass = userPass;

}

public Timestamp getAccCreateTime() {

return accCreateTime;

}

public void setAccCreateTime(Timestamp accCreateTime) {

this.accCreateTime = accCreateTime;

}

}

1. dss.listener

* ContextListener.java

package dss.listener;

import javax.servlet.ServletContextEvent;

import javax.servlet.ServletContextListener;

import javax.servlet.annotation.WebListener;

import dss.util.HibernateUtil;

@WebListener

public class ContextListener implements ServletContextListener

{

@Override

public void contextInitialized(ServletContextEvent sce) {

HibernateUtil.getSessionFactory();

}

@Override

public void contextDestroyed(ServletContextEvent sce) {

HibernateUtil.destroySessionFactory();

}

}

1. dss.service

* DssService.java

package dss.service;

import java.util.List;

import java.util.Map;

import dss.dto.CategoryInterest;

import dss.dto.Question;

import dss.dto.UserDetails;

public interface DssService {

public List<Question>loadAllQuestions() throws Exception;

public List<Integer> getCategoryIds() throws Exception;

public List<CategoryInterest> calculateCategoryInterest(Map<Integer,List<String>>catIdQuestionIdsMap) throws Exception;

public void addUser(UserDetails userDetails) throws Exception;

public void validateLoginUser(UserDetails userDetails) throws Exception;

public void validateDuplicate(UserDetails userDetails) throws Exception;

}

1. dss.service.impl

* DssServiceImpl.java

package dss.service.impl;

import java.util.ArrayList;

import java.util.List;

import java.util.Map;

import dss.dao.QuestionDao;

import dss.dao.UserDetailsDao;

import dss.dto.Category;

import dss.dto.CategoryInterest;

import dss.dto.Question;

import dss.dto.UserDetails;

import dss.service.DssService;

import dss.service.locator.ServiceLocator;

public class DssServiceImpl implements DssService{

private static final int NO\_OF\_QUES\_PER\_CATEGORY = 2;

private static final int NO\_OF\_OPTIONS\_PER\_QUESTION = 3;

private static final int MAXIMUM\_INTEREST = 100;

private static final int SOME\_INTEREST = 50;

private static final int NO\_INTEREST = 0;

public List<Question>loadAllQuestions() throws Exception{

try {

QuestionDao questionDao = new QuestionDao();

List<Question>questionList = questionDao.getQuestionList();

if(questionList == null || questionList.size() == 0) {

throw new Exception("Question List is empty");

}

return questionList;

}

catch (Exception e) {

throw e;

}

}

public List<Integer> getCategoryIds() throws Exception{

try {

QuestionDao questionDao = new QuestionDao();

List<Integer> idsList = questionDao.loadCategoryIds();

if(idsList == null || idsList.size() == 0) {

throw new Exception("Category Id List is empty");

}

return idsList;

}

catch (Exception e) {

throw e;

}

}

public List<CategoryInterest> calculateCategoryInterest(Map<Integer,List<String>>catIdQuestionIdsMap) throws Exception{

List<Integer>catIds = ServiceLocator.getDssService().getCategoryIds();

List<CategoryInterest>categoryInterestList = new ArrayList<CategoryInterest>();

for (int i = 0;i < catIds.size(); i++) {

Integer categoryId = catIds.get(i);

List<String>questionAndOptionList = catIdQuestionIdsMap.get(categoryId);

int totalInterestPerCategory = 0;

for (String questionAndOption : questionAndOptionList) {

String option = questionAndOption.split("\_")[1];

Integer optionId = Integer.parseInt(option);

switch(optionId % NO\_OF\_OPTIONS\_PER\_QUESTION) {

case 2:{

totalInterestPerCategory += MAXIMUM\_INTEREST;

break;

}

case 0:{

totalInterestPerCategory += SOME\_INTEREST;

break;

}

case 1:{

totalInterestPerCategory += NO\_INTEREST;

break;

}

}

}

float percent = totalInterestPerCategory/NO\_OF\_QUES\_PER\_CATEGORY;

CategoryInterest categoryInterest = new CategoryInterest();

categoryInterest.setCategoryId(categoryId);

QuestionDao dao = new QuestionDao();

Category category = dao.getCategorybyId(categoryId);

categoryInterest.setCategoryName(category.getCategoryName());

categoryInterest.setInterest(percent);

categoryInterestList.add(categoryInterest);

}

return categoryInterestList;

}

public void addUser(UserDetails userDetails) throws Exception {

try {

UserDetailsDao dao = new UserDetailsDao();

dao.saveUser(userDetails);

}

catch(Exception e){

throw e;

}

}

public void validateDuplicate(UserDetails userDetails) throws Exception{

UserDetailsDao dao = new UserDetailsDao();

UserDetails dbDetails = dao.authenticateUser(userDetails.getUserName());

if(dbDetails != null) {

throw new Exception("User name already exists!");

}

}

public void validateLoginUser(UserDetails inputDetails ) throws Exception{

try {

UserDetailsDao dao = new UserDetailsDao();

UserDetails dbDetails = dao.authenticateUser(inputDetails.getUserName());

//validate username

if(dbDetails == null) {

throw new Exception("User is not registered");

}

//validating user password

if(!(inputDetails.getUserPass().equals(dbDetails.getUserPass()))) {

throw new Exception("Invalid Password!! Please try again");

}

}

catch(Exception e) {

throw e;

}

}

}

1. dss.service.locator

* ServiceLocator.java

package dss.service.locator;

import dss.service.DssService;

import dss.service.impl.DssServiceImpl;

public class ServiceLocator {

private static DssService dssService;

public static DssService getDssService() {

if(dssService == null) {

dssService = new DssServiceImpl();

}

return dssService;

}

}

1. dss.servlet

* DisplayChartServlet.java

package dss.servlet;

import java.io.IOException;

import java.io.OutputStream;

import java.util.List;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import org.jfree.chart.ChartFactory;

import org.jfree.chart.ChartUtils;

import org.jfree.chart.JFreeChart;

import org.jfree.chart.plot.PlotOrientation;

import org.jfree.data.category.DefaultCategoryDataset;

import dss.dto.CategoryInterest;

/\*\*

\*

\*

\* Servlet implementation class DisplayChartServlet

\*/

public class DisplayChartServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

HttpSession session = request.getSession();

List<CategoryInterest> catInterstList = (List<CategoryInterest>) session.getAttribute("categoryInterestList");

DefaultCategoryDataset dataset = new DefaultCategoryDataset();

for(CategoryInterest categoryInterest : catInterstList)

{

String categoryName = categoryInterest.getCategoryName();

float interestPercent = categoryInterest.getInterest();

dataset.addValue(interestPercent, categoryName, "");

}

JFreeChart chart = ChartFactory.createBarChart("Your Interest in Different Engineering Streams", "Branch", "Percentage", dataset, PlotOrientation.VERTICAL, true, true, false);

chart.setBorderVisible(true);

if (chart != null) {

int width = 600;

int height = 400;

response.setContentType("image/jpeg");

OutputStream out = response.getOutputStream();

ChartUtils.writeChartAsJPEG(out, chart, width, height);

}

}

}

* EvaluationServlet.java

package dss.servlet;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Collection;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import dss.dto.CategoryInterest;

import dss.service.locator.ServiceLocator;

/\*\*

\* Servlet implementation class EvaluationServlet

\*/

public class EvaluationServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

/\*\*

\* @see HttpServlet#HttpServlet()

\*/

public EvaluationServlet() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

Map<Integer,List<String>>catIdQuestionIdsMap = new HashMap<>();

Map<String,String[]>requestMap = request.getParameterMap();

Collection<String[]>valueList = requestMap.values();

for (String[] strings : valueList) {

String value = strings[0];

String[] valArr = value.split("\_");

String optionId = valArr[0];

String quesId = valArr[1];

String catId = valArr[2];

Integer catIdInt = Integer.parseInt(catId);

List<String>quesIdsList = catIdQuestionIdsMap.get(catIdInt);

if(quesIdsList==null) {

quesIdsList = new ArrayList<>();

quesIdsList.add(quesId+"\_"+optionId);

}

else {

quesIdsList.add(quesId+"\_"+optionId);

}

catIdQuestionIdsMap.put(catIdInt, quesIdsList);

}

List<CategoryInterest>listCatInterest = null;

try {

listCatInterest = ServiceLocator.getDssService().calculateCategoryInterest(catIdQuestionIdsMap);

} catch (Exception e) {

e.printStackTrace();

}

HttpSession httpSession = request.getSession();

httpSession.setAttribute("categoryInterestList", listCatInterest);

RequestDispatcher rd = request.getRequestDispatcher("showInterest.jsp");

rd.forward(request, response);

}

}

* LoginServlet.java

package dss.servlet;

import java.io.IOException;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import dss.dto.UserDetails;

import dss.service.locator.ServiceLocator;

/\*\*

\* Servlet implementation class LoginServlet

\*/

public class LoginServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

String userName = request.getParameter("userName");

String userPass = request.getParameter("userPass");

UserDetails userDetails = new UserDetails();

userDetails.setUserName(userName);

userDetails.setUserPass(userPass);

try {

//validate user

ServiceLocator.getDssService().validateLoginUser(userDetails);

//save usename in session

HttpSession httpSession = request.getSession();

httpSession.setAttribute("username", userName);

//redirect to instructions page

response.sendRedirect("instructions.html");

} catch (Exception e) {

// in case of error forward it back to login page with the error message

request.setAttribute("error", e.getMessage());

RequestDispatcher requestDispatcher = request.getRequestDispatcher("login.jsp");

requestDispatcher.forward(request, response);

}

}

}

* LogOutServlet.java

package dss.servlet;

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

/\*\*

\* Servlet implementation class LogoutServlet

\*/

public class LogoutServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

HttpSession session = request.getSession();

session.invalidate();

response.sendRedirect("login.jsp");

}

}

* SignUpServlet.java

package dss.servlet;

import java.io.IOException;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import dss.dto.UserDetails;

import dss.service.locator.ServiceLocator;

/\*\*

\* Servlet implementation class SignUpServlet

\*/

public class SignUpServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

//1. getting values from html form

String userName= request.getParameter("userName");

String userPass= request.getParameter("userPass");

UserDetails userDetails = new UserDetails();

try {

//2. setting values in User Details dto object

userDetails.setUserName(userName);

userDetails.setUserPass(userPass);

ServiceLocator.getDssService().validateDuplicate(userDetails);

//3. calling service method for saving the user object to database

ServiceLocator.getDssService().addUser(userDetails);

request.setAttribute("success", "Account created successfully! Please login to continue");

} catch (Exception e) {

request.setAttribute("error",e.getMessage());

}

RequestDispatcher requestDispatcher = request.getRequestDispatcher("login.jsp");

requestDispatcher.forward(request, response);

}

}

1. dss.util

* HibernateUtil.java

package dss.util;

import org.hibernate.SessionFactory;

import org.hibernate.cfg.Configuration;

public class HibernateUtil {

private static SessionFactory sessionFactory;

public static SessionFactory getSessionFactory()

{

try{

if(sessionFactory==null)

{

Configuration configuration = new Configuration();

configuration.configure("hibernate.cfg.xml");

sessionFactory = configuration.buildSessionFactory();

}

}

catch(Exception e)

{

e.printStackTrace();

}

return sessionFactory;

}

public static void destroySessionFactory() {

if(sessionFactory != null) {

sessionFactory.close();

sessionFactory = null;

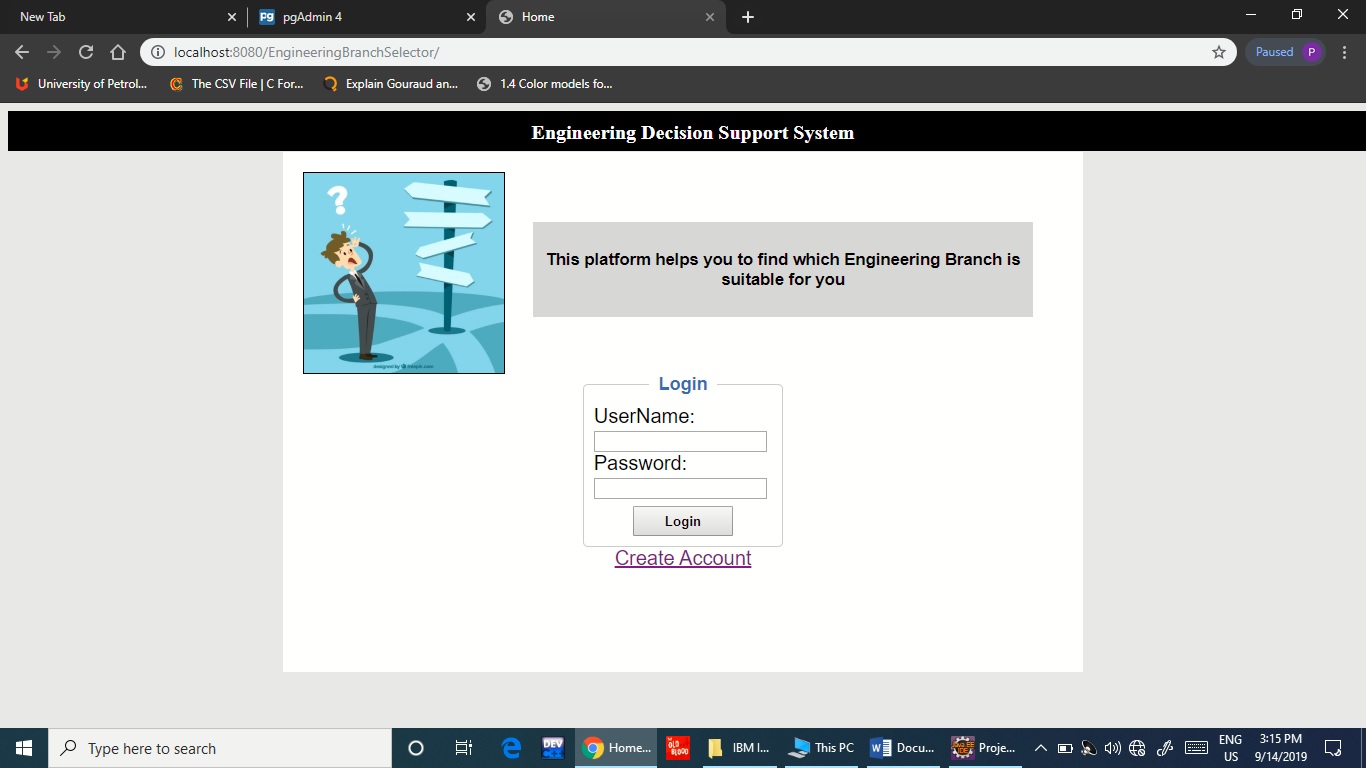
}

}

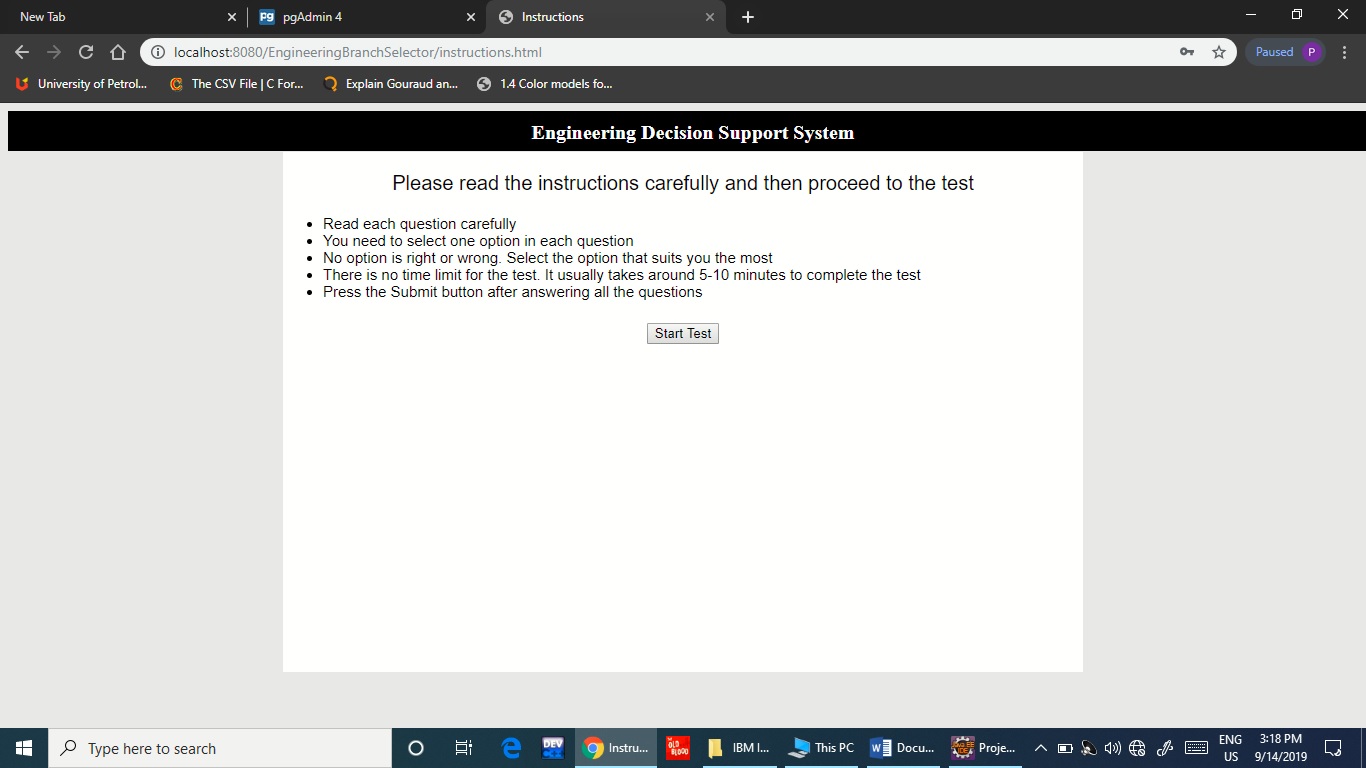
}

**OUPUT PAGE**

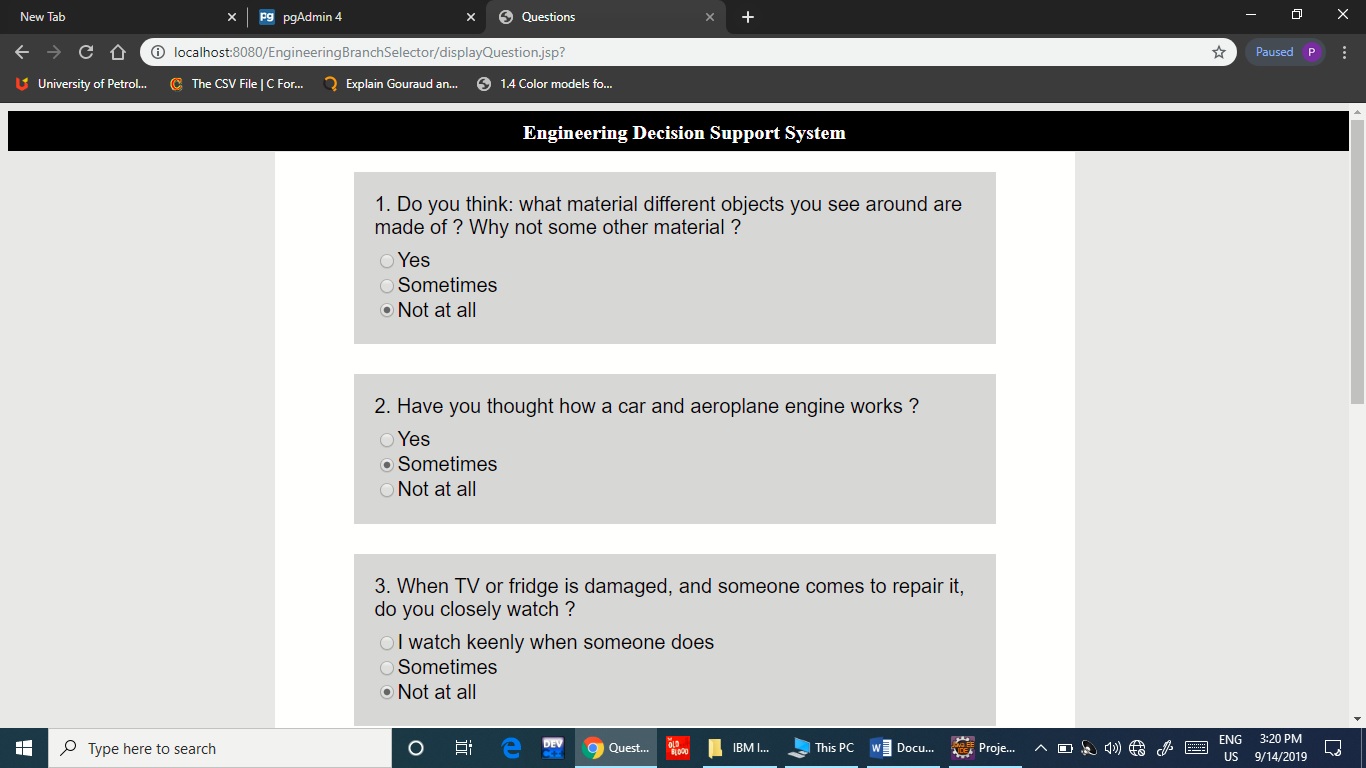
1. Sign Up Page



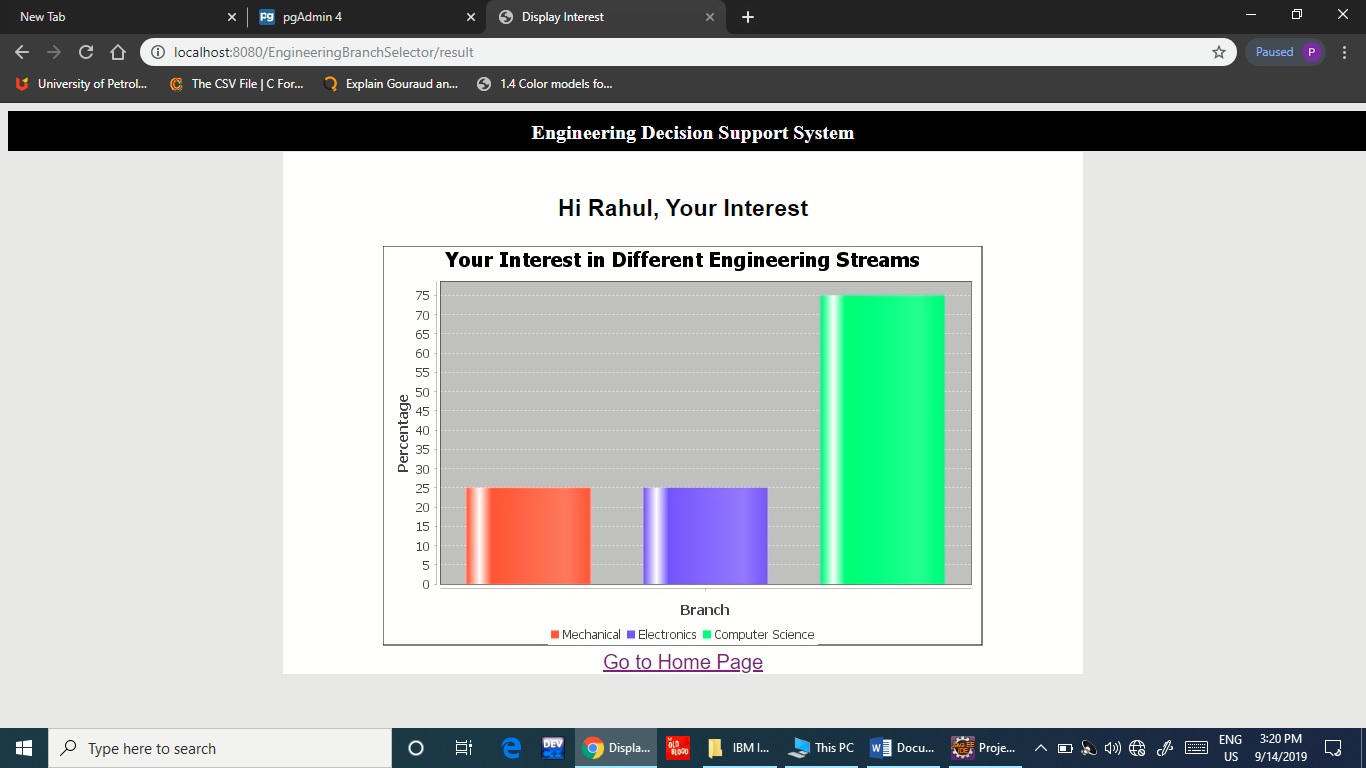
1. Instruction Page



1. Test Page



1. Result Page



**CONCLUSION**

Through the help of this project, we were able to provide assistance to those students who really wanted to identify their skills and interest in the particular engineering branches. With the help of this system, the students can easily make up their decisions and plan out their overall career effectively.

With the help of PostgreSQL, we were able to maintain database of the users who have registered in our system. With the help of JavaScript and JSP, we were able to make different modules related to basic operations performed on websites such as Sign up page, Login page etc. to make our system fully functional.

Hence, we would conclude that this project proves to be a helping mentor for student who wants to pursue engineering on the basis of their skills and interest.