|  |  |
| --- | --- |
| **Full Name** | MEGHA MANOHAR DODKE |
| **Batch** | FS JAVA DEVELOPER-DEC-2021 |
| **Project Title** | Learner's Academy Administrator |
| **Project Submission Date** | 26 March , 2022 |
| **Duration of Sprints** | 7 days |
| **GIT HUB DETAILS** |  |

**FLOW  CHART**

**SERVLET**

**CONTROLLER**

**WEB**

**BROWSER**

**DATABASE**

**JSPPAGE**

**VIEW**

**NO**

**CORRECT**

**DATA**

**LOGIN**

**PAGE**

**LOGIN**

**BEFORE**

**NO**

**YES**

**GET**

**TEACHERS**

**DATA**

**GET**

**STUDENTs**

**DATA**

**GET**

**SUBJECT**

**DATA**

**GET**

**CLASS**

**DATA**

**GET**

**CLASS**

**INFORMATION**

|  |
| --- |
| **SOURCE\_CODE**  **Learnersacademy**  **Dao**  ClassReportDetailsDao  package com.learnersacademy.dao;  import java.util.ArrayList;  import java.util.Iterator;  import java.util.List;  import org.hibernate.Criteria;  import org.hibernate.Session;  import org.hibernate.Transaction;  import org.hibernate.criterion.Projections;  import org.hibernate.criterion.Restrictions;  import com.learnersacademy.entity.Student;  import com.learnersacademy.entity.TeacherClassSubjectMapping;  import com.learnersacademy.util.HibernateUtil;  public class ClassReportDetailsDao {  public List<TeacherClassSubjectMapping> getTeacherClassSubjectMappingsDetails(int classId) {  Transaction transaction = null;  List<TeacherClassSubjectMapping> getTeachersHandlingSubjectsList = new ArrayList<TeacherClassSubjectMapping>();  try (Session session = HibernateUtil.getSessionFactory().openSession()) {  transaction = session.beginTransaction();  Criteria teachersHandlingSubjectsCriteria = session.createCriteria(TeacherClassSubjectMapping.class);  teachersHandlingSubjectsCriteria.add(Restrictions.eq("classId", classId));  teachersHandlingSubjectsCriteria.setProjection(Projections.distinct(Projections.projectionList()  .add(Projections.property("teacherId"))  .add(Projections.property("teacherName"))  .add(Projections.property("subjectId"))  .add(Projections.property("subjectName"))  ));  List results = teachersHandlingSubjectsCriteria.list();  if(results != null && results.size() > 0) {  Iterator<Object> it = results.iterator();  while(it.hasNext()) {  Object[] row = (Object[]) it.next();  TeacherClassSubjectMapping teachersClassesSubjectsMappingObj = new TeacherClassSubjectMapping();  teachersClassesSubjectsMappingObj.setTeacherId((Integer) row[0]);  teachersClassesSubjectsMappingObj.setTeacherName((String) row[1]);  teachersClassesSubjectsMappingObj.setSubjectId((Integer) row[2]);  teachersClassesSubjectsMappingObj.setSubjectName((String) row[3]);  getTeachersHandlingSubjectsList.add(teachersClassesSubjectsMappingObj);  }  }  transaction.commit();  } catch (Exception e) {  if (transaction != null) {  transaction.rollback();  }  e.printStackTrace();  }  return getTeachersHandlingSubjectsList;    }    public List<Student> getStudentDetails(int classId) {  Transaction transaction = null;  List<Student> getStudentInTheClassList = new ArrayList<Student>();  try (Session session = HibernateUtil.getSessionFactory().openSession()) {  transaction = session.beginTransaction();  Criteria studentsCriteria = session.createCriteria(Student.class);  studentsCriteria.add(Restrictions.eq("classId", classId));  studentsCriteria.setProjection(Projections.distinct(Projections.projectionList()  .add(Projections.property("id"))  .add(Projections.property("name"))  .add(Projections.property("emergencyContactNumber"))  .add(Projections.property("bloodGroup"))  .add(Projections.property("gender"))  ));  List results = studentsCriteria.list();  if(results != null && results.size() > 0) {  Iterator<Object> it = results.iterator();  while(it.hasNext()) {  Object[] row = (Object[]) it.next();  Student student = new Student();  student.setId((Integer) row[0]);  student.setName((String) row[1]);  student.setEmergencyContactNumber((String) row[2]);  student.setBloodGroup((String) row[3]);  student.setGender((String) row[4]);  getStudentInTheClassList.add(student);  }  }  transaction.commit();  } catch (Exception e) {  if (transaction != null) {  transaction.rollback();  }  e.printStackTrace();  }  return getStudentInTheClassList;    }  }  ClassRoomDao  package com.learnersacademy.dao;  import java.util.List;  import org.hibernate.Session;  import org.hibernate.Transaction;  import com.learnersacademy.entity.ClassRoom;  import com.learnersacademy.util.HibernateUtil;  public class ClassRoomDao {  public ClassRoom getClassRoom(int id) {  Transaction transaction = null;  ClassRoom classRoom = null;  try (Session session = HibernateUtil.getSessionFactory().openSession()) {  transaction = session.beginTransaction();  classRoom = session.get(ClassRoom.class, id);  transaction.commit();  } catch (Exception e) {  if (transaction != null) {  transaction.rollback();  }  e.printStackTrace();  }  return classRoom;  }  public ClassRoom saveClassRoom(ClassRoom classRoom) {  Transaction transaction = null;  ClassRoom createdClassRoom = null;  Session session = null;  try {  session = HibernateUtil.getSessionFactory().openSession();  transaction = session.beginTransaction();  session.save(classRoom);  transaction.commit();  } catch (Exception e) {  if (transaction != null) {  transaction.rollback();  }  e.printStackTrace();  } finally {  session.close();  }  return createdClassRoom;  }  @SuppressWarnings("unchecked")  public List<ClassRoom> getAllClassRooms() {  Transaction transaction = null;  List<ClassRoom> listOfClassRooms = null;  try (Session session = HibernateUtil.getSessionFactory().openSession()) {  transaction = session.beginTransaction();  listOfClassRooms = session.createQuery("from ClassRoom").getResultList();  transaction.commit();  } catch (Exception e) {  if (transaction != null) {  transaction.rollback();  }  e.printStackTrace();  }  return listOfClassRooms;  }    public void deleteClass(int id) {  Transaction transaction = null;  Session session = null;  try {  session = HibernateUtil.getSessionFactory().openSession();  transaction = session.beginTransaction();  ClassRoom classRoomObj = session.get(ClassRoom.class, id);  session.delete(classRoomObj);  transaction.commit();  } catch (Exception e) {  if (transaction != null) {  transaction.rollback();  }  e.printStackTrace();  } finally {  session.close();  }  }  }  ClassSubjectMappingDao  package com.learnersacademy.dao;  import java.util.List;  import org.hibernate.Session;  import org.hibernate.Transaction;  import com.learnersacademy.entity.ClassSubjectMapping;  import com.learnersacademy.util.HibernateUtil;  public class ClassSubjectMappingDao {  public ClassSubjectMapping saveClassSubjectMapping(ClassSubjectMapping classSubjectMapping) {  Transaction transaction = null;  ClassSubjectMapping createdClassSubjectMapping = null;  Session session = null;  try {  session = HibernateUtil.getSessionFactory().openSession();  transaction = session.beginTransaction();  session.save(classSubjectMapping);  transaction.commit();  } catch (Exception e) {  if (transaction != null) {  transaction.rollback();  }  e.printStackTrace();  } finally {  session.close();  }  return createdClassSubjectMapping;  }  @SuppressWarnings("unchecked")  public List<ClassSubjectMapping> getAllClassSubjectMapping() {  Transaction transaction = null;  List<ClassSubjectMapping> listOfClassSubjectMappings = null;  try (Session session = HibernateUtil.getSessionFactory().openSession()) {  transaction = session.beginTransaction();  listOfClassSubjectMappings = session.createQuery("from ClassSubjectMapping").getResultList();  transaction.commit();  } catch (Exception e) {  if (transaction != null) {  transaction.rollback();  }  e.printStackTrace();  }  return listOfClassSubjectMappings;  }    public void deleteClassSubjectMapping(int id) {  Transaction transaction = null;  Session session = null;  try {  session = HibernateUtil.getSessionFactory().openSession();  transaction = session.beginTransaction();  ClassSubjectMapping mappingObj = session.get(ClassSubjectMapping.class, id);  session.delete(mappingObj);  transaction.commit();  } catch (Exception e) {  if (transaction != null) {  transaction.rollback();  }  e.printStackTrace();  } finally {  session.close();  }  }    }  StudentDao  package com.learnersacademy.dao;  import java.util.List;  import org.hibernate.Session;  import org.hibernate.Transaction;  import com.learnersacademy.entity.Student;  import com.learnersacademy.util.HibernateUtil;  public class StudentDao {  public Student getStudent(int id) {  Transaction transaction = null;  Student student = null;  try (Session session = HibernateUtil.getSessionFactory().openSession()) {  transaction = session.beginTransaction();  student = session.get(Student.class, id);  transaction.commit();  } catch (Exception e) {  if (transaction != null) {  transaction.rollback();  }  e.printStackTrace();  }  return student;  }  public Student saveStudent(Student student) {  Transaction transaction = null;  Student createdStudent = null;  Session session = null;  try {  session = HibernateUtil.getSessionFactory().openSession();  transaction = session.beginTransaction();  session.save(student);  transaction.commit();  } catch (Exception e) {  if (transaction != null) {  transaction.rollback();  }  e.printStackTrace();  } finally {  session.close();  }  return createdStudent;  }  @SuppressWarnings("unchecked")  public List<Student> getAllStudents() {  Transaction transaction = null;  List<Student> listOfStudents = null;  try (Session session = HibernateUtil.getSessionFactory().openSession()) {  transaction = session.beginTransaction();  listOfStudents = session.createQuery("from Student").getResultList();  transaction.commit();  } catch (Exception e) {  if (transaction != null) {  transaction.rollback();  }  e.printStackTrace();  }  return listOfStudents;  }    public void deleteStudents(int id) {  Transaction transaction = null;  Session session = null;  try {  session = HibernateUtil.getSessionFactory().openSession();  transaction = session.beginTransaction();  Student studentsObj = session.get(Student.class, id);  session.delete(studentsObj);  transaction.commit();  } catch (Exception e) {  if (transaction != null) {  transaction.rollback();  }  e.printStackTrace();  } finally {  session.close();  }  }    }  SubjectDao  package com.learnersacademy.dao;  import java.util.List;  import org.hibernate.Session;  import org.hibernate.Transaction;  import com.learnersacademy.entity.Subject;  import com.learnersacademy.util.HibernateUtil;  public class SubjectDao {  public Subject getSubject(int id) {  Transaction transaction = null;  Subject subject = null;  try (Session session = HibernateUtil.getSessionFactory().openSession()) {  transaction = session.beginTransaction();  subject = session.get(Subject.class, id);  transaction.commit();  } catch (Exception e) {  if (transaction != null) {  transaction.rollback();  }  e.printStackTrace();  }  return subject;  }  public Subject saveSubject(Subject subject) {  Transaction transaction = null;  Subject createdSubject = null;  Session session = null;  try {  session = HibernateUtil.getSessionFactory().openSession();  transaction = session.beginTransaction();  session.save(subject);  transaction.commit();  } catch (Exception e) {  if (transaction != null) {  transaction.rollback();  }  e.printStackTrace();  } finally {  session.close();  }  return createdSubject;  }  @SuppressWarnings("unchecked")  public List<Subject> getAllSubjects() {  Transaction transaction = null;  List<Subject> listOfSubjects = null;  try (Session session = HibernateUtil.getSessionFactory().openSession()) {  transaction = session.beginTransaction();  listOfSubjects = session.createQuery("from Subject").getResultList();  transaction.commit();  } catch (Exception e) {  if (transaction != null) {  transaction.rollback();  }  e.printStackTrace();  }  return listOfSubjects;  }    public Subject updateStudent(Subject subject) {  Transaction transaction = null;  Subject createdSubject = null;  Session session = null;  try {  session = HibernateUtil.getSessionFactory().openSession();  transaction = session.beginTransaction();  session.update(subject);  transaction.commit();  } catch (Exception e) {  if (transaction != null) {  transaction.rollback();  }  e.printStackTrace();  } finally {  session.close();  }  return createdSubject;  }  public void deleteSubjects(int id) {  Transaction transaction = null;  Session session = null;  try {  session = HibernateUtil.getSessionFactory().openSession();  transaction = session.beginTransaction();  Subject subjectsObj = session.get(Subject.class, id);  session.delete(subjectsObj);  transaction.commit();  } catch (Exception e) {  if (transaction != null) {  transaction.rollback();  }  e.printStackTrace();  } finally {  session.close();  }  }    }  TeacherClassSubjectMappingDao  package com.learnersacademy.dao;  import java.util.List;  import org.hibernate.Session;  import org.hibernate.Transaction;  import com.learnersacademy.entity.TeacherClassSubjectMapping;  import com.learnersacademy.util.HibernateUtil;  public class TeacherClassSubjectMappingDao {  public TeacherClassSubjectMapping saveTeacherClassSubjectMapping(TeacherClassSubjectMapping teacherClassSubjectMapping) {  Transaction transaction = null;  TeacherClassSubjectMapping createdTeacherClassSubjectMapping = null;  Session session = null;  try {  session = HibernateUtil.getSessionFactory().openSession();  transaction = session.beginTransaction();  session.save(teacherClassSubjectMapping);  transaction.commit();  } catch (Exception e) {  if (transaction != null) {  transaction.rollback();  }  e.printStackTrace();  } finally {  session.close();  }  return createdTeacherClassSubjectMapping;  }  @SuppressWarnings("unchecked")  public List<TeacherClassSubjectMapping> getAllTeacherClassSubjectMapping() {  Transaction transaction = null;  List<TeacherClassSubjectMapping> listOfTeacherClassSubjectMappings = null;  try (Session session = HibernateUtil.getSessionFactory().openSession()) {  transaction = session.beginTransaction();  listOfTeacherClassSubjectMappings = session.createQuery("from TeacherClassSubjectMapping").getResultList();  transaction.commit();  } catch (Exception e) {  if (transaction != null) {  transaction.rollback();  }  e.printStackTrace();  }  return listOfTeacherClassSubjectMappings;  }    public void deleteTeacherClassSubjectMapping(int id) {  Transaction transaction = null;  Session session = null;  try {  session = HibernateUtil.getSessionFactory().openSession();  transaction = session.beginTransaction();  TeacherClassSubjectMapping mappingObj = session.get(TeacherClassSubjectMapping.class, id);  session.delete(mappingObj);  transaction.commit();  } catch (Exception e) {  if (transaction != null) {  transaction.rollback();  }  e.printStackTrace();  } finally {  session.close();  }  }  }  TeacherDao  package com.learnersacademy.dao;  import java.util.List;  import org.hibernate.Session;  import org.hibernate.Transaction;  import com.learnersacademy.entity.Teacher;  import com.learnersacademy.util.HibernateUtil;  public class TeacherDao {  public Teacher getTeacher(int id) {  Transaction transaction = null;  Teacher teacher = null;  try (Session session = HibernateUtil.getSessionFactory().openSession()) {  transaction = session.beginTransaction();  teacher = session.get(Teacher.class, id);  transaction.commit();  } catch (Exception e) {  if (transaction != null) {  transaction.rollback();  }  e.printStackTrace();  }  return teacher;  }  public Teacher saveTeacher(Teacher teacher) {  Transaction transaction = null;  Teacher createdTeacher = null;  Session session = null;  try {  session = HibernateUtil.getSessionFactory().openSession();  transaction = session.beginTransaction();  session.save(teacher);  transaction.commit();  } catch (Exception e) {  if (transaction != null) {  transaction.rollback();  }  e.printStackTrace();  } finally {  session.close();  }  return createdTeacher;  }  @SuppressWarnings("unchecked")  public List<Teacher> getAllTeachers() {  Transaction transaction = null;  List<Teacher> listOfTeachers = null;  try (Session session = HibernateUtil.getSessionFactory().openSession()) {  transaction = session.beginTransaction();  listOfTeachers = session.createQuery("from Teacher").getResultList();  transaction.commit();  } catch (Exception e) {  if (transaction != null) {  transaction.rollback();  }  e.printStackTrace();  }  return listOfTeachers;  }    public void deleteTeacher(int id) {  Transaction transaction = null;  Session session = null;  try {  session = HibernateUtil.getSessionFactory().openSession();  transaction = session.beginTransaction();  Teacher teacherObj = session.get(Teacher.class, id);  session.delete(teacherObj);  transaction.commit();  } catch (Exception e) {  if (transaction != null) {  transaction.rollback();  }  e.printStackTrace();  } finally {  session.close();  }  }    }  **ENTITY**  **ClassRoom**  package com.learnersacademy.entity;  import java.util.Date;  import java.util.List;  import javax.persistence.CascadeType;  import javax.persistence.Entity;  import javax.persistence.GeneratedValue;  import javax.persistence.GenerationType;  import javax.persistence.Id;  import javax.persistence.JoinColumn;  import javax.persistence.JoinTable;  import javax.persistence.ManyToMany;  import javax.persistence.OneToMany;  import javax.persistence.Table;  import org.hibernate.annotations.CreationTimestamp;  @Entity  @Table(name = "class")  public class ClassRoom {    @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  private int id;  private String className;  private String sectionName;  private int totalNumberOfStudents;  private String roomNo;  private String classTeacherName;  @CreationTimestamp  private Date createdDt;    public int getId() {  return id;  }  public String getClassName() {  return className;  }  public void setClassName(String className) {  this.className = className;  }    public String getSectionName() {  return sectionName;  }  public void setSectionName(String sectionName) {  this.sectionName = sectionName;  }  public int getTotalNumberOfStudents() {  return totalNumberOfStudents;  }  public void setTotalNumberOfStudents(int totalNumberOfStudents) {  this.totalNumberOfStudents = totalNumberOfStudents;  }  public String getRoomNo() {  return roomNo;  }  public void setRoomNo(String roomNo) {  this.roomNo = roomNo;  }  public String getClassTeacherName() {  return classTeacherName;  }  public void setClassTeacherName(String classTeacherName) {  this.classTeacherName = classTeacherName;  }  public Date getCreatedDt() {  return createdDt;  }  public void setCreatedDt(Date createdDt) {  this.createdDt = createdDt;  }  public ClassRoom() {  super();  }  public ClassRoom(String className, String sectionName, int totalNumberOfStudents, String roomNo,  String classTeacherName) {  super();  this.className = className;  this.sectionName = sectionName;  this.totalNumberOfStudents = totalNumberOfStudents;  this.roomNo = roomNo;  this.classTeacherName = classTeacherName;  }  }  **ClassSubjectMapping**  package com.learnersacademy.entity;  import java.util.Date;  import javax.persistence.Entity;  import javax.persistence.GeneratedValue;  import javax.persistence.GenerationType;  import javax.persistence.Id;  import javax.persistence.Table;  import org.hibernate.annotations.CreationTimestamp;  @Entity  @Table(name = "class\_subject\_mapping")  public class ClassSubjectMapping {  @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  private int id;  private int classId;  private String className;  private int subjectId;  private String subjectName;  @CreationTimestamp  private Date createdDt;    public int getId() {  return id;  }    public int getClassId() {  return classId;  }    public void setClassId(int classId) {  this.classId = classId;  }    public String getClassName() {  return className;  }    public void setClassName(String className) {  this.className = className;  }    public int getSubjectId() {  return subjectId;  }    public void setSubjectId(int subjectId) {  this.subjectId = subjectId;  }    public String getSubjectName() {  return subjectName;  }    public void setSubjectName(String subjectName) {  this.subjectName = subjectName;  }    public Date getCreatedDt() {  return createdDt;  }    public void setCreatedDt(Date createdDt) {  this.createdDt = createdDt;  }    public ClassSubjectMapping() {  super();  }    public ClassSubjectMapping(int classId, String className, int subjectId, String subjectName) {  super();  this.classId = classId;  this.className = className;  this.subjectId = subjectId;  this.subjectName = subjectName;  }    }  **Student**  package com.learnersacademy.entity;  import java.util.Date;  import javax.persistence.Entity;  import javax.persistence.GeneratedValue;  import javax.persistence.GenerationType;  import javax.persistence.Id;  import javax.persistence.Table;  import org.hibernate.annotations.CreationTimestamp;  import org.hibernate.annotations.UpdateTimestamp;  @Entity  @Table(name = "student")  public class Student {  @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  private int id;  private String name;  private String email;  private String emergencyContactNumber;  private String bloodGroup;  private String gender;  private int age;  private int classId;  private String className;  private String address;  @CreationTimestamp  private Date createdDt;  @UpdateTimestamp  private Date updateDt;    public int getId() {  return 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 getEmergencyContactNumber() {  return emergencyContactNumber;  }  public void setEmergencyContactNumber(String emergencyContactNumber) {  this.emergencyContactNumber = emergencyContactNumber;  }  public String getBloodGroup() {  return bloodGroup;  }  public void setBloodGroup(String bloodGroup) {  this.bloodGroup = bloodGroup;  }  public String getGender() {  return gender;  }  public void setGender(String gender) {  this.gender = gender;  }  public int getAge() {  return age;  }  public void setAge(int age) {  this.age = age;  }  public int getClassId() {  return classId;  }  public void setClassId(int classId) {  this.classId = classId;  }  public String getClassName() {  return className;  }  public void setClassName(String className) {  this.className = className;  }  public String getAddress() {  return address;  }  public void setAddress(String address) {  this.address = address;  }  public Date getCreatedDt() {  return createdDt;  }  public void setCreatedDt(Date createdDt) {  this.createdDt = createdDt;  }  public void setId(int id) {  this.id = id;  }  public Date getUpdateDt() {  return updateDt;  }  public void setUpdateDt(Date updateDt) {  this.updateDt = updateDt;  }  public Student() {  super();  }    public Student(String name, String email, String emergencyContactNumber, String bloodGroup, String gender, int age,  int classId, String className, String address) {  super();  this.name = name;  this.email = email;  this.emergencyContactNumber = emergencyContactNumber;  this.bloodGroup = bloodGroup;  this.gender = gender;  this.age = age;  this.classId = classId;  this.className = className;  this.address = address;  }  }  **Subject**  package com.learnersacademy.entity;  import java.util.Date;  import javax.persistence.Entity;  import javax.persistence.GeneratedValue;  import javax.persistence.GenerationType;  import javax.persistence.Id;  import javax.persistence.Table;  import org.hibernate.annotations.CreationTimestamp;  @Entity  @Table(name = "subject")  public class Subject {    @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  private int id;  private String subjectName;  private String subjectDescription;  @CreationTimestamp  private Date createdDt;    public int getId() {  return id;  }    public String getSubjectName() {  return subjectName;  }  public void setSubjectName(String subjectName) {  this.subjectName = subjectName;  }  public String getSubjectDescription() {  return subjectDescription;  }  public void setSubjectDescription(String subjectDescription) {  this.subjectDescription = subjectDescription;  }  public Date getCreatedDt() {  return createdDt;  }  public void setCreatedDt(Date createdDt) {  this.createdDt = createdDt;  }    public Subject() {  super();  }    public Subject(String subjectName, String subjectDescription) {  super();  this.subjectName = subjectName;  this.subjectDescription = subjectDescription;  }  @Override  public String toString() {  return "Subject [id=" + id + ", subjectName=" + subjectName + ", createdDt=" + createdDt + "]";  }        }  **Teacher**  package com.learnersacademy.entity;  import java.util.Date;  import javax.persistence.Entity;  import javax.persistence.GeneratedValue;  import javax.persistence.GenerationType;  import javax.persistence.Id;  import javax.persistence.Table;  import org.hibernate.annotations.CreationTimestamp;  @Entity  @Table(name = "teacher")  public class Teacher {  @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  private int id;  private String firstName;  private String lastName;  private String contactNumber;  private String emailId;  private String qualification;  private String gender;  private int age;  private String martialStatus;  @CreationTimestamp  private Date createdDt;    public int getId() {  return id;  }  public String getFirstName() {  return firstName;  }  public void setFirstName(String firstName) {  this.firstName = firstName;  }  public String getLastName() {  return lastName;  }  public void setLastName(String lastName) {  this.lastName = lastName;  }  public String getContactNumber() {  return contactNumber;  }  public void setContactNumber(String contactNumber) {  this.contactNumber = contactNumber;  }  public String getEmailId() {  return emailId;  }  public void setEmailId(String emailId) {  this.emailId = emailId;  }  public String getQualification() {  return qualification;  }  public void setQualification(String qualification) {  this.qualification = qualification;  }  public String getGender() {  return gender;  }  public void setGender(String gender) {  this.gender = gender;  }  public int getAge() {  return age;  }  public void setAge(int age) {  this.age = age;  }  public String getMartialStatus() {  return martialStatus;  }  public void setMartialStatus(String martialStatus) {  this.martialStatus = martialStatus;  }  public Date getCreatedDt() {  return createdDt;  }  public void setCreatedDt(Date createdDt) {  this.createdDt = createdDt;  }  public Teacher() {  super();  }  public Teacher(String firstName, String lastName, String contactNumber, String emailId, String qualification,  String gender, int age, String martialStatus) {  super();  this.firstName = firstName;  this.lastName = lastName;  this.contactNumber = contactNumber;  this.emailId = emailId;  this.qualification = qualification;  this.gender = gender;  this.age = age;  this.martialStatus = martialStatus;  }    }  **TeacherClassSubjectMapping**  package com.learnersacademy.entity;  import java.util.Date;  import javax.persistence.Entity;  import javax.persistence.GeneratedValue;  import javax.persistence.GenerationType;  import javax.persistence.Id;  import javax.persistence.Table;  import org.hibernate.annotations.CreationTimestamp;  @Entity  @Table(name = "teacher\_class\_subject\_mapping")  public class TeacherClassSubjectMapping {  @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  private int id;  private int classId;  private String className;  private int subjectId;  private String subjectName;  private int teacherId;  private String teacherName;  @CreationTimestamp  private Date createdDt;    public int getId() {  return id;  }  public void setId(int id) {  this.id = id;  }  public int getClassId() {  return classId;  }  public void setClassId(int classId) {  this.classId = classId;  }  public String getClassName() {  return className;  }  public void setClassName(String className) {  this.className = className;  }  public int getSubjectId() {  return subjectId;  }  public void setSubjectId(int subjectId) {  this.subjectId = subjectId;  }  public String getSubjectName() {  return subjectName;  }  public void setSubjectName(String subjectName) {  this.subjectName = subjectName;  }  public int getTeacherId() {  return teacherId;  }  public void setTeacherId(int teacherId) {  this.teacherId = teacherId;  }  public String getTeacherName() {  return teacherName;  }  public void setTeacherName(String teacherName) {  this.teacherName = teacherName;  }  public Date getCreatedDt() {  return createdDt;  }  public void setCreatedDt(Date createdDt) {  this.createdDt = createdDt;  }    public TeacherClassSubjectMapping() {  super();  }    public TeacherClassSubjectMapping(int classId, String className, int subjectId, String subjectName, int teacherId,  String teacherName) {  super();  this.classId = classId;  this.className = className;  this.subjectId = subjectId;  this.subjectName = subjectName;  this.teacherId = teacherId;  this.teacherName = teacherName;  }    @Override  public String toString() {  return "TeacherClassSubjectMapping [teacherId=" + teacherId + ", teacherName=" + teacherName + ", subjectId=" + subjectId+ ", subjectName=" + subjectName + "]";  }    }  **SERVLETS**  **ClassReportDetailsServlet**  package com.learnersacademy.servlets;  import java.io.IOException;  import java.util.List;  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 com.learnersacademy.dao.ClassReportDetailsDao;  import com.learnersacademy.entity.Student;  import com.learnersacademy.entity.TeacherClassSubjectMapping;  /\*\*  \* Servlet implementation class ClassReportDetailsServlet  \*/  public class ClassReportDetailsServlet extends HttpServlet {  private static final long serialVersionUID = 1L;  private ClassReportDetailsDao classReportDetailsDao;  public ClassReportDetailsServlet() {  super();  }  public void init() {  classReportDetailsDao = new ClassReportDetailsDao();  }  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  String action = request.getParameter("action");  try {  switch (action) {  case "classReportDetails":  generateClassReportDetails(request, response);  break;  }  } catch (Exception e) {  e.printStackTrace();  }  }  private void generateClassReportDetails(HttpServletRequest request, HttpServletResponse response) {  String classId = request.getParameter("classId");  List<TeacherClassSubjectMapping> teacherClassSubjectMappings = classReportDetailsDao.getTeacherClassSubjectMappingsDetails(Integer.parseInt(classId));  List<Student> studentDetails = classReportDetailsDao.getStudentDetails(Integer.parseInt(classId));  try {  HttpSession session = request.getSession();  session.setAttribute("teacherClassSubjectMappings", teacherClassSubjectMappings);  session.setAttribute("studentDetails", studentDetails);  RequestDispatcher dispatcher = request.getRequestDispatcher("pages/listClassDetailsReport.jsp");  dispatcher.forward(request, response);  } catch (Exception e) {  e.printStackTrace();  }  }  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  doGet(request, response);  }  }  **ClassRoomServlet**  package com.learnersacademy.servlets;  import java.io.IOException;  import java.util.List;  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 com.learnersacademy.dao.ClassRoomDao;  import com.learnersacademy.entity.ClassRoom;  import net.sf.json.JSONObject;  /\*\*  \* Servlet implementation class ClassRoomServlet  \*/  public class ClassRoomServlet extends HttpServlet {  private static final long serialVersionUID = 1L;  private ClassRoomDao classRoomDao;  public ClassRoomServlet() {  super();  }  public void init() {  classRoomDao = new ClassRoomDao();  }  private ClassRoom getClassRoom(HttpServletRequest request, HttpServletResponse response) {  String classRoomId = request.getParameter("id");  ClassRoom classRoom = classRoomDao.getClassRoom(Integer.parseInt(classRoomId));  HttpSession session = request.getSession();  session.setAttribute("classRoom", classRoom);  return classRoom;  }  private List<ClassRoom> getClassRooms(HttpServletRequest request, HttpServletResponse response, boolean delFlag) {  List<ClassRoom> classRooms = classRoomDao.getAllClassRooms();  String reportFlag = request.getParameter("reportFlag");  try {  HttpSession session = request.getSession();  session.setAttribute("classRooms", classRooms);  session.setAttribute("delFlag", delFlag);  if("Y".equals(reportFlag)) {  RequestDispatcher dispatcher = request.getRequestDispatcher("pages/listClassReport.jsp");  dispatcher.forward(request, response);  } else {  RequestDispatcher dispatcher = request.getRequestDispatcher("pages/list-classRooms.jsp");  dispatcher.forward(request, response);  }  } catch (Exception e) {  e.printStackTrace();  }  return classRooms;  }  public List<ClassRoom> getAllClassDetails(HttpServletRequest request, HttpServletResponse response) throws IOException {  List<ClassRoom> classRooms = classRoomDao.getAllClassRooms();  String flag = request.getParameter("servletName");  try {  HttpSession session = request.getSession();  session.setAttribute("classRooms", classRooms);  if(!"mappingServlet".equals(flag)) {  RequestDispatcher dispatcher = request.getRequestDispatcher("pages/addStudent.jsp");  dispatcher.forward(request, response);  }  } catch (Exception e) {  e.printStackTrace();  }  return classRooms;  }  private ClassRoom createClassRoom(HttpServletRequest request, HttpServletResponse response) {  String className = request.getParameter("className");  String sectionName = request.getParameter("sectionName");  int totalNumberOfStudents = Integer.parseInt(request.getParameter("totalNumberOfStudents"));  String roomNo = request.getParameter("roomNo");  String classTeacherName = request.getParameter("classTeacherName");  ClassRoom classRoomModel = new ClassRoom(className, sectionName, totalNumberOfStudents, roomNo, classTeacherName);  ClassRoom newClassRoom = classRoomDao.saveClassRoom(classRoomModel);  getClassRooms(request, response, false);  return newClassRoom;  }  private void deleteClass(HttpServletRequest request, HttpServletResponse response) {  int classId = Integer.parseInt(request.getParameter("id"));  classRoomDao.deleteClass(classId);  getClassRooms(request, response, true);  }  private void redirectToAddJsp(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  RequestDispatcher dispatcher = request.getRequestDispatcher("pages/addClass.jsp");  dispatcher.forward(request, response);  }    protected void doGet(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  String action = request.getParameter("action");  try {  switch (action) {  case "new":  createClassRoom(request, response);  break;  case "list":  getClassRooms(request, response, false);  break;  case "delete":  deleteClass(request, response);  break;  case "listClassName":  getAllClassDetails(request, response);  break;  case "classById":  getClassRoom(request, response);  break;    case "addJsp":  redirectToAddJsp(request, response);  break;  }  } catch (Exception e) {  e.printStackTrace();  }  }  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  doGet(request, response);  }  }  **ClassSubjectMappingServlet**  package com.learnersacademy.servlets;  import java.io.IOException;  import java.util.List;  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 com.learnersacademy.dao.ClassSubjectMappingDao;  import com.learnersacademy.entity.ClassRoom;  import com.learnersacademy.entity.ClassSubjectMapping;  import com.learnersacademy.entity.Student;  import com.learnersacademy.entity.Subject;  /\*\*  \* Servlet implementation class ClassSubjectMappingServlet  \*/  public class ClassSubjectMappingServlet extends HttpServlet {  private static final long serialVersionUID = 1L;  private ClassSubjectMappingDao classSubjectMappingDao;  public ClassSubjectMappingServlet() {  super();  }  public void init() {  classSubjectMappingDao = new ClassSubjectMappingDao();  }  private List<ClassSubjectMapping> getClassSubjectMappings(HttpServletRequest request, HttpServletResponse response, boolean delFlag) {  List<ClassSubjectMapping> classSubjectMappings = classSubjectMappingDao.getAllClassSubjectMapping();  try {  HttpSession session = request.getSession();  session.setAttribute("classSubjectMappings", classSubjectMappings);  session.setAttribute("delFlag", delFlag);  RequestDispatcher dispatcher = request.getRequestDispatcher("pages/listClassSubjectMapping.jsp");  dispatcher.forward(request, response);  } catch (Exception e) {  e.printStackTrace();  }  return classSubjectMappings;  }  private ClassSubjectMapping createClassSubjectMapping(HttpServletRequest request, HttpServletResponse response) {  int classId = Integer.parseInt(request.getParameter("classesNameCombo"));  int subjectId = Integer.parseInt(request.getParameter("subjectsNameCombo"));  String className = request.getParameter("className");  String subjectName = request.getParameter("subjectName");  ClassSubjectMapping mappingModel = new ClassSubjectMapping(classId, className, subjectId, subjectName);  ClassSubjectMapping newMapping = classSubjectMappingDao.saveClassSubjectMapping(mappingModel);  getClassSubjectMappings(request, response, false);  return newMapping;  }  private void deleteClassSubjectMapping(HttpServletRequest request, HttpServletResponse response) {  int mappingId = Integer.parseInt(request.getParameter("id"));  classSubjectMappingDao.deleteClassSubjectMapping(mappingId);  getClassSubjectMappings(request, response, true);  }  private void getSubjectClassDetails(HttpServletRequest request, HttpServletResponse response) {  RequestDispatcher dispatcher;  try {  dispatcher = request.getRequestDispatcher("classRoom?action=listClassName&servletName=mappingServlet");  dispatcher.include(request, response);  HttpSession session = request.getSession(false);  List<ClassRoom> classRooms= (List<ClassRoom>) session.getAttribute("classRooms");  dispatcher = request.getRequestDispatcher("subject?action=list&servletName=mappingServlet");  dispatcher.include(request, response);  List<Subject> subjects= (List<Subject>) session.getAttribute("subjects");    dispatcher = request.getRequestDispatcher("pages/addClassSubjectMapping.jsp");  dispatcher.forward(request, response);  } catch (ServletException e) {  e.printStackTrace();  } catch (IOException e) {  e.printStackTrace();  }  }  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  String action = request.getParameter("action");  try {  switch (action) {  case "new":  createClassSubjectMapping(request, response);  break;  case "list":  getClassSubjectMappings(request, response, false);  break;  case "delete":  deleteClassSubjectMapping(request, response);  break;  case "mappingDetails":  getSubjectClassDetails(request, response);  break;  }  } catch (Exception e) {  e.printStackTrace();  }  }  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  doGet(request, response);  }  }  **LoginServlet**  package com.learnersacademy.servlets;  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;  public class LoginServlet extends HttpServlet {  private static final long serialVersionUID = 1L;  public LoginServlet() {  super();  }  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  String username = request.getParameter("username");  String password = request.getParameter("password");  if(username.equals("admin") && password.equals("admin@123")) {  request.getSession().setAttribute("loggedInUser", "admin");  RequestDispatcher dispatcher = request.getRequestDispatcher("pages/dashboard.jsp");  dispatcher.forward(request, response);  } else {  RequestDispatcher dispatcher = request.getRequestDispatcher("pages/error.jsp");  dispatcher.forward(request, response);  }  }  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  doGet(request, response);  }  }  **LogoutServlet**  package com.learnersacademy.servlets;  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;  public class LogoutServlet extends HttpServlet {  private static final long serialVersionUID = 1L;    public LogoutServlet() {  super();  }  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  RequestDispatcher dispatcher = request.getRequestDispatcher("pages/login.jsp");  dispatcher.forward(request, response);  }  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  doGet(request, response);  }  }  **StudentServlet**  package com.learnersacademy.servlets;  import java.io.IOException;  import java.util.Date;  import java.util.List;  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 com.learnersacademy.dao.StudentDao;  import com.learnersacademy.entity.ClassRoom;  import com.learnersacademy.entity.Student;  /\*\*  \* Servlet implementation class StudentServlet  \*/  public class StudentServlet extends HttpServlet {  private static final long serialVersionUID = 1L;  private StudentDao studentDao;  public StudentServlet() {  super();  }  public void init() {  studentDao = new StudentDao();  }  private Student getStudent(HttpServletRequest request, HttpServletResponse response) {  String studentId = request.getParameter("id");  Student student = studentDao.getStudent(Integer.parseInt(studentId));  return student;  }  private List<Student> getStudents(HttpServletRequest request, HttpServletResponse response, boolean delFlag) {  List<Student> students = studentDao.getAllStudents();  try {  HttpSession session = request.getSession();  session.setAttribute("students", students);  session.setAttribute("delFlag", delFlag);  RequestDispatcher dispatcher = request.getRequestDispatcher("pages/list-students.jsp");  dispatcher.forward(request, response);  } catch (Exception e) {  e.printStackTrace();  }  return students;  }  private Student createStudent(HttpServletRequest request, HttpServletResponse response) {  String name = request.getParameter("studentName");  String email = request.getParameter("email");  String emergencyContactNumber = request.getParameter("emergencyContactNumber");  String bloodGroup = request.getParameter("bloodGroup");  String gender = request.getParameter("gender");  int age = Integer.parseInt(request.getParameter("age"));  int classId = Integer.parseInt(request.getParameter("classesNameCombo"));  String address = request.getParameter("address");  Student newStudent = null;  try {  RequestDispatcher dispatcher = request.getRequestDispatcher("classRoom?action=classById&id="+classId);  dispatcher.include(request, response);  HttpSession session = request.getSession(false);  ClassRoom classRoom = (ClassRoom) session.getAttribute("classRoom");  String className = classRoom.getClassName() != null && !classRoom.getClassName().isEmpty()?classRoom.getClassName():"";  Student studentModel = new Student(name, email, emergencyContactNumber, bloodGroup, gender, age, classId, className, address);  newStudent = studentDao.saveStudent(studentModel);  getStudents(request, response, false);  } catch (ServletException e) {  e.printStackTrace();  } catch (IOException e) {  e.printStackTrace();  }  return newStudent;  }    private void deleteStudent(HttpServletRequest request, HttpServletResponse response) {  int studentId = Integer.parseInt(request.getParameter("id"));  studentDao.deleteStudents(studentId);  getStudents(request, response, true);  }      protected void doGet(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  String action = request.getParameter("action");  try {  switch (action) {  case "new":  createStudent(request, response);  break;  case "list":  getStudents(request, response, false);  break;    case "delete":  deleteStudent(request, response);  break;  }  } catch (Exception e) {  e.printStackTrace();  }  }  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  doGet(request, response);  }  }  **SubjectServlet**  package com.learnersacademy.servlets;  import java.io.IOException;  import java.util.List;  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 com.learnersacademy.dao.SubjectDao;  import com.learnersacademy.entity.Subject;  /\*\*  \* Servlet implementation class SubjectServlet  \*/  public class SubjectServlet extends HttpServlet {  private static final long serialVersionUID = 1L;  private SubjectDao subjectDao;  public SubjectServlet() {  super();  }  public void init() {  subjectDao = new SubjectDao();  }  private Subject getSubject(HttpServletRequest request, HttpServletResponse response) {  String subjectId = request.getParameter("id");  Subject subject = subjectDao.getSubject(Integer.parseInt(subjectId));  return subject;  }  private List<Subject> getSubjects(HttpServletRequest request, HttpServletResponse response, boolean delFlag) {  List<Subject> subjects = subjectDao.getAllSubjects();  String flag = request.getParameter("servletName");  try {  HttpSession session = request.getSession();  session.setAttribute("subjects", subjects);  session.setAttribute("delFlag", delFlag);  if(!"mappingServlet".equals(flag)) {  RequestDispatcher dispatcher = request.getRequestDispatcher("pages/listSubjects.jsp");  dispatcher.forward(request, response);  }  } catch (Exception e) {  e.printStackTrace();  }  return subjects;  }  private Subject createSubject(HttpServletRequest request, HttpServletResponse response) {  String subjectName = request.getParameter("subjectName");  String subjectDescription = request.getParameter("subjectDescription");  Subject subjectModel = new Subject(subjectName, subjectDescription);  Subject newSubject = subjectDao.saveSubject(subjectModel);  getSubjects(request, response, false);  return newSubject;  }  private void deleteSubject(HttpServletRequest request, HttpServletResponse response) {  int subjectId = Integer.parseInt(request.getParameter("id"));  subjectDao.deleteSubjects(subjectId);  getSubjects(request, response, true);  }  private void redirectToAddJsp(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  RequestDispatcher dispatcher = request.getRequestDispatcher("pages/addSubject.jsp");  dispatcher.forward(request, response);  }  protected void doGet(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  String action = request.getParameter("action");  try {  switch (action) {  case "new":  createSubject(request, response);  break;  case "list":  getSubjects(request, response, false);  break;  case "delete":  deleteSubject(request, response);  break;    case "addJsp":  redirectToAddJsp(request, response);  break;  }  } catch (Exception e) {  e.printStackTrace();  }  }  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  doGet(request, response);  }  }  **TeacherClassSubjectMappingServlet**  package com.learnersacademy.servlets;  import java.io.IOException;  import java.util.List;  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 com.learnersacademy.dao.TeacherClassSubjectMappingDao;  import com.learnersacademy.entity.ClassRoom;  import com.learnersacademy.entity.Subject;  import com.learnersacademy.entity.Teacher;  import com.learnersacademy.entity.TeacherClassSubjectMapping;  /\*\*  \* Servlet implementation class TeacherClassSubjectMappingServlet  \*/  public class TeacherClassSubjectMappingServlet extends HttpServlet {  private static final long serialVersionUID = 1L;    private TeacherClassSubjectMappingDao teacherClassSubjectMappingDao;  public TeacherClassSubjectMappingServlet() {  super();  }    public void init() {  teacherClassSubjectMappingDao = new TeacherClassSubjectMappingDao();  }    private List<TeacherClassSubjectMapping> getTeacherClassSubjectMappings(HttpServletRequest request, HttpServletResponse response, boolean delFlag) {  List<TeacherClassSubjectMapping> teacherClassSubjectMappings = teacherClassSubjectMappingDao.getAllTeacherClassSubjectMapping();  try {  HttpSession session = request.getSession();  session.setAttribute("teacherClassSubjectMappings", teacherClassSubjectMappings);  session.setAttribute("delFlag", delFlag);  RequestDispatcher dispatcher = request.getRequestDispatcher("pages/listTeacherClassSubjectMapping.jsp");  dispatcher.forward(request, response);  } catch (Exception e) {  e.printStackTrace();  }  return teacherClassSubjectMappings;  }    private TeacherClassSubjectMapping createTeacherClassSubjectMapping(HttpServletRequest request, HttpServletResponse response) {  int classId = Integer.parseInt(request.getParameter("classesNameCombo"));  String className = request.getParameter("className");  int subjectId = Integer.parseInt(request.getParameter("subjectsNameCombo"));  String subjectName = request.getParameter("subjectName");  int teacherId = Integer.parseInt(request.getParameter("teachersNameCombo"));  String teacherName = request.getParameter("teacherName");    TeacherClassSubjectMapping mappingModel = new TeacherClassSubjectMapping(classId, className, subjectId, subjectName, teacherId, teacherName);  TeacherClassSubjectMapping newMapping = teacherClassSubjectMappingDao.saveTeacherClassSubjectMapping(mappingModel);  getTeacherClassSubjectMappings(request, response, false);  return newMapping;  }    private void deleteTeacherClassSubjectMapping(HttpServletRequest request, HttpServletResponse response) {  int mappingId = Integer.parseInt(request.getParameter("id"));  teacherClassSubjectMappingDao.deleteTeacherClassSubjectMapping(mappingId);  getTeacherClassSubjectMappings(request, response, true);  }  private void getTeacherSubjectClassDetails(HttpServletRequest request, HttpServletResponse response) {  RequestDispatcher dispatcher;  try {  dispatcher = request.getRequestDispatcher("classRoom?action=listClassName&servletName=mappingServlet");  dispatcher.include(request, response);  HttpSession session = request.getSession(false);  dispatcher = request.getRequestDispatcher("subject?action=list&servletName=mappingServlet");  dispatcher.include(request, response);    dispatcher = request.getRequestDispatcher("teacher?action=list&servletName=mappingServlet");  dispatcher.include(request, response);    dispatcher = request.getRequestDispatcher("pages/addTeacherClassSubjectMapping.jsp");  dispatcher.forward(request, response);  } catch (ServletException e) {  e.printStackTrace();  } catch (IOException e) {  e.printStackTrace();  }  }    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  String action = request.getParameter("action");  try {  switch (action) {  case "new":  createTeacherClassSubjectMapping(request, response);  break;  case "list":  getTeacherClassSubjectMappings(request, response, false);  break;    case "delete":  deleteTeacherClassSubjectMapping(request, response);  break;    case "mappingDetails":  getTeacherSubjectClassDetails(request, response);  break;  }  } catch (Exception e) {  e.printStackTrace();  }  }  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  doGet(request, response);  }  }  **TeacherServlet**  package com.learnersacademy.servlets;  import java.io.IOException;  import java.util.List;  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 com.learnersacademy.dao.TeacherDao;  import com.learnersacademy.entity.Teacher;  public class TeacherServlet extends HttpServlet {  private static final long serialVersionUID = 1L;    private TeacherDao teacherDao;    public TeacherServlet() {  super();  }  public void init() {  teacherDao = new TeacherDao();  }    private Teacher getTeacher(HttpServletRequest request, HttpServletResponse response) {  String teacherId = request.getParameter("id");  Teacher teacher = teacherDao.getTeacher(Integer.parseInt(teacherId));  return teacher;  }  private List<Teacher> getTeachers(HttpServletRequest request, HttpServletResponse response, boolean delFlag) {  List<Teacher> teachers = teacherDao.getAllTeachers();  String flag = request.getParameter("servletName");  try {  HttpSession session = request.getSession();  session.setAttribute("teachers", teachers);  session.setAttribute("delFlag", delFlag);  if(!"mappingServlet".equals(flag)) {  RequestDispatcher dispatcher = request.getRequestDispatcher("pages/listTeacher.jsp");  dispatcher.forward(request, response);  }  } catch (Exception e) {  e.printStackTrace();  }  return teachers;  }  private Teacher createTeacher(HttpServletRequest request, HttpServletResponse response) {  String firstName = request.getParameter("firstName");  String lastName = request.getParameter("lastName");  String contactNumber = request.getParameter("contactNumber");  String emailId = request.getParameter("emailAddress");  String qualification = request.getParameter("qualification");  int age = Integer.parseInt(request.getParameter("age"));  String martialStatus = request.getParameter("martialStatus");  String gender = request.getParameter("gender");  Teacher teacherModel = new Teacher(firstName, lastName, contactNumber, emailId, qualification, gender, age, martialStatus);  Teacher newTeacher = teacherDao.saveTeacher(teacherModel);  getTeachers(request, response, false);  return newTeacher;  }  private void deleteTeacher(HttpServletRequest request, HttpServletResponse response) {  int teacherId = Integer.parseInt(request.getParameter("id"));  teacherDao.deleteTeacher(teacherId);  getTeachers(request, response, true);  }    private void redirectToAddJsp(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  RequestDispatcher dispatcher = request.getRequestDispatcher("pages/addTeacher.jsp");  dispatcher.forward(request, response);  }    protected void doGet(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  String action = request.getParameter("action");  try {  switch (action) {  case "new":  createTeacher(request, response);  break;  case "list":  getTeachers(request, response, false);  break;    case "delete":  deleteTeacher(request, response);  break;    case "addJsp":  redirectToAddJsp(request, response);  break;  }  } catch (Exception e) {  e.printStackTrace();  }  }  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  doGet(request, response);  }  }  **UTIL**  **HibernateUtil**  package com.learnersacademy.util;  import java.util.Properties;  import org.hibernate.SessionFactory;  import org.hibernate.boot.registry.StandardServiceRegistryBuilder;  import org.hibernate.cfg.Configuration;  import org.hibernate.cfg.Environment;  import org.hibernate.service.ServiceRegistry;  import com.learnersacademy.entity.ClassRoom;  import com.learnersacademy.entity.ClassSubjectMapping;  import com.learnersacademy.entity.Student;  import com.learnersacademy.entity.Subject;  import com.learnersacademy.entity.Teacher;  import com.learnersacademy.entity.TeacherClassSubjectMapping;  public class HibernateUtil {    private static SessionFactory sessionFactory;    public static SessionFactory getSessionFactory() {    if(sessionFactory == null) {  try {  Configuration configuration = new Configuration();    Properties hibernateProperties = new Properties();  hibernateProperties.put(Environment.DRIVER, "com.mysql.cj.jdbc.Driver");  hibernateProperties.put(Environment.URL, "jdbc:mysql://127.0.0.1:3306/learnersacademy");  hibernateProperties.put(Environment.USER, "root");  hibernateProperties.put(Environment.PASS, "Megha@123");  hibernateProperties.put(Environment.DIALECT, "org.hibernate.dialect.MySQL5InnoDBDialect");  hibernateProperties.put(Environment.SHOW\_SQL, "true");  hibernateProperties.put(Environment.FORMAT\_SQL, "true");  hibernateProperties.put(Environment.HBM2DDL\_AUTO, "update");  configuration.setProperties(hibernateProperties);    configuration.addAnnotatedClass(Student.class);  configuration.addAnnotatedClass(ClassRoom.class);  configuration.addAnnotatedClass(Teacher.class);  configuration.addAnnotatedClass(Subject.class);  configuration.addAnnotatedClass(ClassSubjectMapping.class);  configuration.addAnnotatedClass(TeacherClassSubjectMapping.class);    ServiceRegistry serviceRegistry = new StandardServiceRegistryBuilder()  .applySettings(configuration.getProperties()).build();    sessionFactory = configuration.buildSessionFactory(serviceRegistry);  }  catch (Exception e) {  e.printStackTrace();  }  }  return sessionFactory;  }    }  **JSP  P AGES**  **addClass.jsp**  <%@ page language="java" contentType="text/html; charset=ISO-8859-1"  pageEncoding="ISO-8859-1"%>  <!DOCTYPE html>  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" />  <div class="container-fluid">  <div class="card-header py-3">  <h6 class="m-0 font-weight-bold text-primary"><a href="pages/dashboard.jsp">Home</a>  &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;  <a href="/LearnersAcademyAdministrativePortal/logout" style='float:right'>Logout</a></h6>  </div>  <div class="d-sm-flex align-items-center justify-content-between mb-4">  <h1 class="h3 mb-0 text-gray-800">Add Classes</h1> </div>  <div class="row">  <div class="col-lg-12">  <form class="addClasses" action="/LearnersAcademyAdministrativePortal/classRoom" method="post">  <div class="form-row">  <div class="form-group col-md-5">  <label for="className">Class Name</label>  <input type="text" class="form-control" id="className" name="className" placeholder="Enter Class Name" required="required"> </div>  <div class="form-group col-md-5">  <label for="sectionName">Section Name</label>  <input type="text" class="form-control" id="sectionName" name="sectionName" placeholder="Enter Section Name"> </div>  <div class="form-group col-md-5">  <label for="totalNumberOfStudents">Total No of Students</label>  <input type="number" class="form-control" id="totalNumberOfStudents" name="totalNumberOfStudents" placeholder="Enter Total Number of Students" required="required"> </div>  <div class="form-group col-md-5">  <label for="roomNo">Room Name</label>  <input type="text" class="form-control" id="roomNo" name="roomNo" placeholder="Enter Room Name"> </div>  <div class="form-group col-md-5">  <label for="classTeacherName">Class Teacher Name</label>  <input type="text" class="form-control" id="classTeacherName" name="classTeacherName" placeholder="Enter Class Teacher Name"> </div>  </div>  <div class="form-group">  <button type="reset" id="resetFormAddClasses" name="resetFormAddClasses" class="btn btn-danger">clear</button>  <button type="submit" id="submitFormAddClasses" name="submitFormAddClasses" class="btn btn-primary">Submit</button>  </div>  <input type="hidden" name="action" value="new"/>  </form>  </div>  </div>  </div>  **addClassSubjectMapping.jsp**  <%@page import="com.learnersacademy.entity.ClassRoom"%>  <%@page import="com.learnersacademy.entity.Subject"%>  <%@page import="java.util.List"%>  <%@ page language="java" contentType="text/html; charset=ISO-8859-1"  pageEncoding="ISO-8859-1"%>  <!DOCTYPE html>  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.0/jquery.min.js"></script>  <script src="http://ajax.aspnetcdn.com/ajax/jquery.validate/1.9/jquery.validate.js"></script>  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" />  <%  List<ClassRoom> classRooms = (List<ClassRoom>) session.getAttribute("classRooms");  List<Subject> subjects = (List<Subject>) session.getAttribute("subjects");  %>  <div class="container-fluid">  <div class="card-header py-3">  <h6 class="m-0 font-weight-bold text-primary"><a href="pages/dashboard.jsp">Home</a>  &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;  <a href="/LearnersAcademyAdministrativePortal/logout" style='float:right'>Logout</a></h6>  </div>  <div class="d-sm-flex align-items-center justify-content-between mb-4">  <h1 class="h3 mb-0 text-gray-800">Add Class Subject Mapping</h1> </div>  <div class="row">  <div class="col-lg-12">  <form class="addClassesSubjectsMapping" action="/LearnersAcademyAdministrativePortal/classSubjectMapping" method="post">  <div class="form-row">  <div class="form-group col-md-5">  <label for="subjectsNameCombo">Subject Name</label>  <select class="form-control" id="subjectsNameCombo" name="subjectsNameCombo">  <option value= 0 selected>Select Subject Name </option>  <% if(subjects != null && subjects.size()>0){%>  <% for(Subject sub : subjects){  %>  <option value="<%= sub.getId()%>"><%= sub.getSubjectName()%></option>  <%}  }%>  </select>  </div>  <div class="form-group col-md-5" style="display : none;">  <label for="subjectId">Subject Id</label>  <input type="text" class="form-control" id="subjectId" name="subjectId" placeholder="Enter Subject Id"> </div>  <div class="form-group col-md-5" style="display : none;">  <label for="subjectName">Subject Name</label>  <input type="text" class="form-control" id="subjectName" name="subjectName" placeholder="Enter Subject Name"> </div>  </div>  <div style="display: none" id="errorSub">  <label><span style="color: red">Please Select Valid Subject Name</span></label>  </div>  <div class="form-row">  <div class="form-group col-md-5">  <label for="classesNameCombo">Class Name</label>  <select class="form-control" id="classesNameCombo" name="classesNameCombo">  <option value= 0 selected>Select Class Name </option>  <% if(classRooms != null && classRooms.size()>0){%>  <% for(ClassRoom cr : classRooms){  %>  <option value="<%= cr.getId()%>"><%= cr.getClassName()%></option>  <%}  }%>  </select>  </div>  <div class="form-group col-md-5" style="display : none;">  <label for="classId">Class Id</label>  <input type="text" class="form-control" id="classId" name="classId" placeholder="Enter Class Id"> </div>  <div class="form-group col-md-5" style="display : none;">  <label for="className">Class Name</label>  <input type="text" class="form-control" id="className" name="className" placeholder="Enter Class Name"> </div>    </div>  <div style="display: none" id="errorClass">  <label><span style="color: red">Please Select Valid Class Name</span></label>  </div>  <div style="display: none" id="errorSubmit">  <label><span style="color: red">Please Select Valid Option</span></label>  </div>  <div class="form-group">  <button type="reset" id="resetFormAddClassesSubjectsMapping" name="resetFormAddClassesSubjectsMapping" class="btn btn-danger">clear</button>  <button type="submit" id="submitFormAddClassesSubjectsMapping" name="submitFormAddClassesSubjectsMapping" class="btn btn-primary">Submit</button>  </div>  <input type="hidden" name="action" value="new"/>  </form>  </div>  </div>  </div>  <script type="text/javascript">  $(document).ready(function() {  $("#subjectsNameCombo").click(function() {  var subjectId = $("#subjectsNameCombo option:selected").val();  if(subjectId == 0){  document.getElementById("errorSub").style.display="inline";  document.getElementById("errorClass").style.display="none";  document.getElementById("errorSubmit").style.display="none";  return false;  } else {  document.getElementById("errorSub").style.display="none";  }  });  });  $(document).ready(function() {  $("#classesNameCombo").click(function() {  var classId = $("#classesNameCombo option:selected").val();  if(classId == 0){  document.getElementById("errorClass").style.display="inline";  document.getElementById("errorSub").style.display="none";  document.getElementById("errorSubmit").style.display="none";  return false;  } else {  document.getElementById("errorClass").style.display="none";  }  });  });  $(document).ready(function() {  $("#submitFormAddClassesSubjectsMapping").click(function() {  var subjectId = $("#subjectsNameCombo option:selected").val();  var classId = $("#classesNameCombo option:selected").val();  document.forms[0].elements["subjectName"].value = $("#subjectsNameCombo option:selected").text();  document.forms[0].elements["className"].value = $("#classesNameCombo option:selected").text();  if(subjectId == 0 || classId == 0){  document.getElementById("errorSubmit").style.display="inline";  document.getElementById("errorClass").style.display="none";  document.getElementById("errorSub").style.display="none";  return false;  } else {  document.getElementById("errorSubmit").style.display="none";  }  });  });  </script>  **addStudent.jsp**  <%@page import="com.learnersacademy.entity.ClassRoom"%>  <%@page import="java.util.List"%>  <%@ page language="java" contentType="text/html; charset=ISO-8859-1"  pageEncoding="ISO-8859-1"%>  <!DOCTYPE html>  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.0/jquery.min.js"></script>  <script src="http://ajax.aspnetcdn.com/ajax/jquery.validate/1.9/jquery.validate.js"></script>  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" />  <%  List<ClassRoom> classRooms = (List<ClassRoom>) session.getAttribute("classRooms");  %>  <div class="container-fluid">  <div class="card-header py-3">  <h6 class="m-0 font-weight-bold text-primary"><a href="pages/dashboard.jsp">Home</a>  &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;  <a href="/LearnersAcademyAdministrativePortal/logout" style='float:right'>Logout</a></h6>  </div>  <div class="d-sm-flex align-items-center justify-content-between mb-4">  <h1 class="h3 mb-0 text-gray-800">Add Students</h1> </div>  <div class="row">  <div class="col-lg-12">  <form class="addStudents" id="addStudents" action="/LearnersAcademyAdministrativePortal/student" method="POST">  <div class="form-row">  <div class="form-group col-md-5">  <label for="studentName">Student Name</label>  <input type="text" class="form-control" id="studentName" name="studentName" placeholder="Enter Student Name" required="required"> </div>  <div class="form-group col-md-5">  <label for="fathersName">Email Address</label>  <input type="email" class="form-control" id="email" name="email" placeholder="Enter Email Address"> </div>  <div class="form-group col-md-5">  <label for="emergencyContactNumber">Emergency Contact Number</label>  <input type="text" class="form-control" id="emergencyContactNumber" name="emergencyContactNumber" placeholder="Enter Emergency Contact Number" required="required" maxlength="10"> </div>  <div class="form-group col-md-5">  <label for="bloodGroup">Blood Group</label>  <input type="text" class="form-control" id="bloodGroup" name="bloodGroup" placeholder="Enter Blood Group" required="required"> </div>  <div class="form-group col-md-5">  <label for="age">Age</label>  <input type="number" class="form-control" id="age" name="age" placeholder="Enter Age"> </div>  <div class="form-group col-md-5">  <label for="gender">Gender</label>  <select class="form-control" id="gender" name="gender">  <option selected>Enter Gender </option>  <option value="male">Male</option>  <option value="female">Female</option>  <option value="others">Others</option>  </select>  </div>  <div class="form-group col-md-5">  <label for="classesNameCombo">Class Name</label>  <select class="form-control" id="classesNameCombo" name="classesNameCombo">  <option value= 0 selected>Select Class Name </option>  <% if(classRooms != null && classRooms.size()>0){  for(ClassRoom cr : classRooms){  %>  <option value="<%= cr.getId()%>"><%= cr.getClassName()%></option>  <%}  }%>  </select>  </div>  <div style="display: none" id="errorClass">  <label><span style="color: red">Please Select Valid Class Name</span></label>  </div>  <div class="form-group col-md-5" style="display : none;">  <label for="classId">Class Id</label>  <input type="text" class="form-control" id="classId" name="classId" placeholder="Enter Class Id"> </div>  <div class="form-group col-md-5" style="display : none;">  <label for="className">Class Name</label>  <input type="text" class="form-control" id="className" name="className" placeholder="Enter Class Name"> </div>  <div class="form-group col-md-10">  <label for="address">Address</label>  <textarea class="form-control" id="address" name="address" rows="3"></textarea>  </div>  </div>  <div class="form-group">  <button type="reset" id="resetFormAddStudents" name="resetFormAddStudents" class="btn btn-danger">clear</button>  <button type="submit" id="submitFormAddStudents" name="submitFormAddStudents" class="btn btn-primary">Submit</button>  </div>  <input type="hidden" name="action" value="new">  </form>  </div>  </div>  </div>  <script type="text/javascript">  $(document).ready(function() {  $("#classesNameCombo").click(function() {  var classId = $("#classesNameCombo option:selected").val();  if(classId == 0){  document.getElementById("errorClass").style.display="inline";  return false;  } else {  document.getElementById("errorClass").style.display="none";  }  });  });  $(document).ready(function() {  $("#submitFormAddStudents").click(function() {  var classId = $("#classesNameCombo option:selected").val();  if(classId == 0){  document.getElementById("errorClass").style.display="inline";  return false;  } else {  document.getElementById("errorClass").style.display="none";  }  });  });  </script>  **addSubject.jsp**  <%@ page language="java" contentType="text/html; charset=ISO-8859-1"  pageEncoding="ISO-8859-1"%>  <!DOCTYPE html>  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" />  <div class="container-fluid">  <div class="card-header py-3">  <h6 class="m-0 font-weight-bold text-primary"><a href="pages/dashboard.jsp">Home</a>  &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;  <a href="/LearnersAcademyAdministrativePortal/logout" style='float:right'>Logout</a></h6>  </div>  <div class="d-sm-flex align-items-center justify-content-between mb-4">  <h1 class="h3 mb-0 text-gray-800">Add Subjects</h1>  </div>  <div class="row">  <div class="col-lg-12">  <form class="addSubjects "action="/LearnersAcademyAdministrativePortal/subject" method="POST">  <div class="form-row">  <div class="form-group col-md-5">  <label for="subjectName">Subject Name</label> <input type="text"  class="form-control" id="subjectName" name="subjectName"  placeholder="Enter Subject Name" required="required">  </div>  <div class="form-group col-md-10">  <label for="subjectDescription">Subject Description</label>  <textarea class="form-control" id="subjectDescription"  name="subjectDescription" rows="3" maxlength="500"></textarea>  </div>  </div>  <div class="form-group">  <button type="reset" id="resetFormAddSubjects"  name="resetFormAddSubjects" class="btn btn-danger">clear</button>  <button type="submit" id="submitFormAddSubjects"  name="submitFormAddSubjects" class="btn btn-primary">Submit</button>  </div>  <input type="hidden" name="action" value="new"/>  </form>  </div>  </div>  </div>  **addTeacher.jsp**  <%@ page language="java" contentType="text/html; charset=ISO-8859-1"  pageEncoding="ISO-8859-1"%>  <!DOCTYPE html>  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" />  <div class="container-fluid">  <div class="card-header py-3">  <h6 class="m-0 font-weight-bold text-primary"><a href="pages/dashboard.jsp">Home</a>  &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;  <a href="/LearnersAcademyAdministrativePortal/logout" style='float:right'>Logout</a></h6>  </div>  <div class="d-sm-flex align-items-center justify-content-between mb-4">  <h1 class="h3 mb-0 text-gray-800">Add Teachers</h1>  </div>  <div class="row">  <div class="col-lg-12">  <form class="addTeachers" action="/LearnersAcademyAdministrativePortal/teacher" method="POST">  <div class="form-row">  <div class="form-group col-md-5">  <label for="firstName">First Name</label> <input type="text"  class="form-control" id="firstName" name="firstName"  placeholder="Enter First Name" required="required">  </div>  <div class="form-group col-md-5">  <label for="lastName">Last Name</label> <input type="text"  class="form-control" id="lastName" name="lastName"  placeholder="Enter Last Name">  </div>  <div class="form-group col-md-5">  <label for="Contact Number">Contact Number</label> <input  type="text" class="form-control" id="contactNumber"  name="contactNumber" placeholder="Enter Contact Number" required="required" maxlength="10">  </div>  <div class="form-group col-md-5">  <label for="emailAddress">Email Address</label> <input  type="email" class="form-control" id="emailAddress"  name="emailAddress" aria-describedby="emailHelp"  placeholder="Enter Email Address" required="required">  </div>  <div class="form-group col-md-5">  <label for="qualification">Qualification</label> <input  type="text" class="form-control" id="qualification"  name="qualification" placeholder="Enter Qualification" required="required">  </div>  <div class="form-group col-md-5">  <label for="age">Age</label> <input type="number"  class="form-control" id="age" name="age" placeholder="Enter Age">  </div>  <div class="form-group col-md-5">  <label for="martialStatus">Martial Status</label> <select  class="form-control" id="martialStatus" name="martialStatus">  <option selected>Enter Martial Status </option>  <option value="Single">Single</option>  <option value="Married">Married</option>  </select>  </div>  <div class="form-group col-md-5">  <label for="gender">Gender</label> <select class="form-control"  id="gender" name="gender">  <option selected>Enter Gender </option>  <option value="male">Male</option>  <option value="female">Female</option>  <option value="others">Others</option>  </select>  </div>  </div>  <div class="form-group">  <button type="reset" id="resetFormAddTeachers"  name="resetFormAddTeachers" class="btn btn-danger">clear</button>  <button type="submit" id="submitFormAddTeachers"  name="submitFormAddTeachers" class="btn btn-primary">Submit</button>  </div>  <input type="hidden" name="action" value="new"/>  </form>  </div>  </div>  </div>  **addTeacherClassSubjectMapping.jsp**  <%@page import="com.learnersacademy.entity.ClassRoom"%>  <%@page import="com.learnersacademy.entity.Subject"%>  <%@page import="com.learnersacademy.entity.Teacher"%>  <%@page import="java.util.List"%>  <%@ page language="java" contentType="text/html; charset=ISO-8859-1"  pageEncoding="ISO-8859-1"%>  <!DOCTYPE html>  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.0/jquery.min.js"></script>  <script src="http://ajax.aspnetcdn.com/ajax/jquery.validate/1.9/jquery.validate.js"></script>  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" />  <%  List<ClassRoom> classRooms = (List<ClassRoom>) session.getAttribute("classRooms");  List<Subject> subjects = (List<Subject>) session.getAttribute("subjects");  List<Teacher> teachers = (List<Teacher>) session.getAttribute("teachers");  %>  <div class="container-fluid">  <div class="card-header py-3">  <h6 class="m-0 font-weight-bold text-primary"><a href="pages/dashboard.jsp">Home</a>  &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;  <a href="/LearnersAcademyAdministrativePortal/logout" style='float:right'>Logout</a></h6>  </div>  <div class="d-sm-flex align-items-center justify-content-between mb-4">  <h1 class="h3 mb-0 text-gray-800">Add Teachers Classes Subjects Mapping</h1> </div>  <div class="row">  <div class="col-lg-12">  <form class="addTeachersClassesSubjectsMapping" action="/LearnersAcademyAdministrativePortal/teacherClassSubjectMapping" method="post">  <div class="form-row">  <div class="form-group col-md-5">  <label for="teachersNameCombo">Teacher Name</label>  <select class="form-control" id="teachersNameCombo" name="teachersNameCombo">  <option value= 0 selected>Select Teacher Name </option>  <% if(teachers != null && teachers.size()>0){%>  <% for(Teacher teacher : teachers){  %>  <option value="<%= teacher.getId()%>"><%= teacher.getFirstName()%></option>  <%}  }%>  </select>  </div>  <div class="form-group col-md-5" style="display : none;">  <label for="teacherId">Teacher Id</label>  <input type="text" class="form-control" id="teacherId" name="teacherId" placeholder="Enter Teacher Id"> </div>  <div class="form-group col-md-5" style="display : none;">  <label for="teacherName">Teacher Name</label>  <input type="text" class="form-control" id="teacherName" name="teacherName" placeholder="Enter Teacher Name"> </div>  </div>  <div style="display: none" id="errorTea">  <label><span style="color: red">Please Select Valid Teacher Name</span></label>  </div>  <div class="form-row">  <div class="form-group col-md-5">  <label for="classesNameCombo">Class Name</label>  <select class="form-control" id="classesNameCombo" name="classesNameCombo">  <option value= 0 selected>Select Class Name </option>  <% if(classRooms != null && classRooms.size()>0){%>  <% for(ClassRoom cr : classRooms){  %>  <option value="<%= cr.getId()%>"><%= cr.getClassName()%></option>  <%}  }%>  </select>  </div>  <div class="form-group col-md-5" style="display : none;">  <label for="classId">Class Id</label>  <input type="text" class="form-control" id="classId" name="classId" placeholder="Enter Class Id"> </div>  <div class="form-group col-md-5" style="display : none;">  <label for="className">Class Name</label>  <input type="text" class="form-control" id="className" name="className" placeholder="Enter Class Name"> </div>    </div>  <div style="display: none" id="errorClass">  <label><span style="color: red">Please Select Valid Class Name</span></label>  </div>  <div class="form-row">  <div class="form-group col-md-5">  <label for="subjectsNameCombo">Subject Name</label>  <select class="form-control" id="subjectsNameCombo" name="subjectsNameCombo">  <option value= 0 selected>Select Subject Name </option>  <% if(subjects != null && subjects.size()>0){%>  <% for(Subject sub : subjects){  %>  <option value="<%= sub.getId()%>"><%= sub.getSubjectName()%></option>  <%}  }%>  </select>  </div>  <div class="form-group col-md-5" style="display : none;">  <label for="subjectId">Subject Id</label>  <input type="text" class="form-control" id="subjectId" name="subjectId" placeholder="Enter Subject Id"> </div>  <div class="form-group col-md-5" style="display : none;">  <label for="subjectName">Subject Name</label>  <input type="text" class="form-control" id="subjectName" name="subjectName" placeholder="Enter Subject Name"> </div>  </div>  <div style="display: none" id="errorSub">  <label><span style="color: red">Please Select Valid Subject Name</span></label>  </div>  <div style="display: none" id="errorSubmit">  <label><span style="color: red">Please Select Valid Option</span></label>  </div>  <div class="form-group">  <button type="reset" id="resetFormAddTeachersClassesSubjectsMapping" name="resetFormAddTeachersClassesSubjectsMapping" class="btn btn-danger">clear</button>  <button type="submit" id="submitFormAddTeachersClassesSubjectsMapping" name="submitFormAddTeachersClassesSubjectsMapping" class="btn btn-primary">Submit</button>  </div>  <input type="hidden" name="action" value="new"/>  </form>  </div>  </div>  </div>  <script type="text/javascript">  $(document).ready(function() {  $("#teachersNameCombo").click(function() {  var teacherId = $("#teachersNameCombo option:selected").val();  if (teacherId == 0) {  document.getElementById("errorTea").style.display = "inline";  document.getElementById("errorSub").style.display = "none";  document.getElementById("errorClass").style.display = "none";  document.getElementById("errorSubmit").style.display = "none";  return false;  } else {  document.getElementById("errorTea").style.display = "none";  }  });  });  $(document).ready(function() {  $("#subjectsNameCombo").click(function() {  var subjectId = $("#subjectsNameCombo option:selected").val();  if (subjectId == 0) {  document.getElementById("errorSub").style.display = "inline";  document.getElementById("errorTea").style.display = "none";  document.getElementById("errorClass").style.display = "none";  document.getElementById("errorSubmit").style.display = "none";  return false;  } else {  document.getElementById("errorSub").style.display = "none";  }  });  });  $(document).ready(function() {  $("#classesNameCombo").click(function() {  var classId = $("#classesNameCombo option:selected").val();  if (classId == 0) {  document.getElementById("errorClass").style.display = "inline";  document.getElementById("errorTea").style.display = "none";  document.getElementById("errorSub").style.display = "none";  document.getElementById("errorSubmit").style.display = "none";  return false;  } else {  document.getElementById("errorClass").style.display = "none";  }  });  });  $(document).ready(function() {  $("#submitFormAddTeachersClassesSubjectsMapping").click(function() {  var subjectId = $("#subjectsNameCombo option:selected").val();  var classId = $("#classesNameCombo option:selected").val();  var teacherId = $("#teachersNameCombo option:selected").val();  document.forms[0].elements["subjectName"].value = $("#subjectsNameCombo option:selected").text();  document.forms[0].elements["className"].value = $("#classesNameCombo option:selected").text();  document.forms[0].elements["teacherName"].value = $("#teachersNameCombo option:selected").text();  if (subjectId == 0 || classId == 0 || teacherId == 0) {  document.getElementById("errorSubmit").style.display = "inline";  document.getElementById("errorTea").style.display = "none";  document.getElementById("errorSub").style.display = "none";  document.getElementById("errorClass").style.display = "none";  return false;  } else {  document.getElementById("errorSubmit").style.display = "none";  }  });  });  </script>  **dashboard.jsp**  <%@ page language="java" contentType="text/html; charset=ISO-8859-1"  pageEncoding="ISO-8859-1"%>  <!DOCTYPE html>  <jsp:include page="menu.jsp" />  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">  <title>Welcome To Learner's Academy</title>  <style type="text/css">  html {  background: url(/LearnersAcademyAdministrativePortal/images/img1.jpg) no-repeat center center fixed;  -webkit-background-size: cover;  -moz-background-size: cover;  -o-background-size: cover;  background-size: cover;  }  </style>  </head>  <body>  </body>  </html>  **error.jsp**  <%@ page language="java" contentType="text/html; charset=ISO-8859-1"  pageEncoding="ISO-8859-1"%>  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">  <title>Learner's Academy</title>  </head>  <body>  <center>  <h1>Learner's Academy</h1>  <h2>  <a href="pages/login.jsp">Login</a>  <p>Looks like your credentials are incorrect!, Please try again! </p>  </h2>  </center>  </body>  </html>  **listClassDetailsReport.jsp**  <%@page import="com.learnersacademy.entity.TeacherClassSubjectMapping"%>  <%@page import="com.learnersacademy.entity.Student"%>  <%@page import="java.util.List"%>  <%@ page language="java" contentType="text/html; charset=ISO-8859-1"  pageEncoding="ISO-8859-1"%>  <%@taglib prefix="c" uri="http://java.sun.com/jstl/core"%>  <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>  <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions"%>  <!DOCTYPE html>  <link rel="stylesheet"  href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" />  <%  List<TeacherClassSubjectMapping> teacherClassSubjectMappings = (List<TeacherClassSubjectMapping>) session.getAttribute("teacherClassSubjectMappings");  List<Student> studentDetails = (List<Student>) session.getAttribute("studentDetails");  %>  <div class="container-fluid">  <div class="card-header py-3">  <h6 class="m-0 font-weight-bold text-primary"><a href="pages/dashboard.jsp">Home</a>  &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;  <a href="/LearnersAcademyAdministrativePortal/logout" style='float:right'>Logout</a></h6>  </div>  <div class="d-sm-flex align-items-center justify-content-between mb-4">  <h1 class="h3 mb-0 text-gray-800">List Class Details Report</h1> </div>  <div class="row">  <div class="col-lg-12">  <div class="card shadow mb-4">  <div class="card-header py-3">  <h6 class="m-0 font-weight-bold text-primary">Teachers and Handling Subjects</h6>  </div>  <div class="card-body">  <div class="table-responsive">  <table class="table table-bordered teachersHandlingSubjectsList" id="dataTable teachersHandlingSubjectsList" width="100%" cellspacing="0">  <thead>  <tr>  <th>Teacher Name</th>  <th>Subject Name</th>  </tr>  </thead>  <tfoot>  <tr>  <c:forEach var="teacherClassSubjectMapping" items="${teacherClassSubjectMappings}">  <tr>  <td><c:out value="${teacherClassSubjectMapping.teacherName}" /></td>  <td><c:out value="${teacherClassSubjectMapping.subjectName}" /></td>  </tr>  </c:forEach>  </tr>  </tfoot>  </table>  </div>  </div>  </div>    <!-- DataTales Example -->  <div class="card shadow mb-4">  <div class="card-header py-3">  <h6 class="m-0 font-weight-bold text-primary">Students in the Class</h6>  </div>  <div class="card-body">  <div class="table-responsive">  <table class="table table-bordered studentsInTheClassList" id="dataTable studentsInTheClassList" width="100%" cellspacing="0">  <thead>  <tr>  <th>Student Name</th>  <th>Gender</th>  <th>Contact Number</th>  <th>Blood Group</th>  </tr>  </thead>  <tfoot>  <tr>  <c:forEach var="student" items="${studentDetails}">  <tr>  <td><c:out value="${student.name}" /></td>  <td><c:out value="${student.gender}" /></td>  <td><c:out value="${student.emergencyContactNumber}" /></td>  <td><c:out value="${student.bloodGroup}" /></td>  </tr>  </c:forEach>  </tr>  </tfoot>  </table>  </div>  </div>  </div>  </div>  </div>  </div>      **listClassReport.jsp**  <%@page import="com.learnersacademy.entity.ClassRoom"%>  <%@page import="java.util.List"%>  <%@ page language="java" contentType="text/html; charset=ISO-8859-1"  pageEncoding="ISO-8859-1"%>  <%@taglib prefix="c" uri="http://java.sun.com/jstl/core"%>  <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>  <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions"%>  <!DOCTYPE html>  <link rel="stylesheet"  href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" />  <%  List<ClassRoom> classRooms = (List<ClassRoom>) session.getAttribute("classRooms");  %>  <div class="container-fluid">  <div class="card-header py-3">  <h6 class="m-0 font-weight-bold text-primary"><a href="pages/dashboard.jsp">Home</a>  &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;  <a href="/LearnersAcademyAdministrativePortal/logout" style='float:right'>Logout</a></h6>  </div>  <div class="d-sm-flex align-items-center justify-content-between mb-4">  <h1 class="h3 mb-0 text-gray-800">List Classes Report</h1>  </div>  <div class="row">  <div class="col-lg-12">  <div class="card shadow mb-4">  <div class="card-header py-3">  <h6 class="m-0 font-weight-bold text-primary">Classes Report</h6>  </div>  <div class="card-body">  <div class="table-responsive">  <table class="table table-bordered classesListReport"  id="dataTable" width="100%" cellspacing="0">  <thead>  <tr>  <th>Class Name</th>  <th>Action</th>  </tr>  </thead>  <tfoot>  <tr>  <c:forEach var="classroom" items="${classRooms}">  <tr>  <td><c:out value="${classroom.className}" /></td>  <td><a  href="/LearnersAcademyAdministrativePortal/classReport?action=classReportDetails&classId=<c:out value='${classroom.id}' />">Generate Report</a></td>  </tr>  </c:forEach>  </tr>  </tfoot>  </table>  </div>  </div>  </div>  </div>  </div>  </div>  **list-classRooms.jsp**  <%@page import="com.learnersacademy.entity.ClassRoom"%>  <%@page import="com.learnersacademy.entity.Student"%>  <%@page import="java.util.List"%>  <%@ page language="java" contentType="text/html; charset=ISO-8859-1"  pageEncoding="ISO-8859-1"%>  <%@taglib prefix="c" uri="http://java.sun.com/jstl/core"%>  <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>  <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions"%>  <!DOCTYPE html>  <link rel="stylesheet"  href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" />  <%  List<ClassRoom> classRooms = (List<ClassRoom>) session.getAttribute("classRooms");  boolean delFlag = (boolean)session.getAttribute("delFlag");  %>  <div class="container-fluid">  <div class="card-header py-3">  <h6 class="m-0 font-weight-bold text-primary"><a href="pages/dashboard.jsp">Home</a>  &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;  <a href="/LearnersAcademyAdministrativePortal/logout" style='float:right'>Logout</a></h6>  </div>  <div class="d-sm-flex align-items-center justify-content-between mb-4">  <h1 class="h3 mb-0 text-gray-800">List Classes</h1>  </div>  <div class="row">  <div class="col-lg-12">  <!-- DataTales Example -->  <div class="card shadow mb-4">  <div class="card-header py-3">  <h6 class="m-0 font-weight-bold text-primary">Classes</h6>  </div>  <div class="card-body">  <div class="table-responsive">  <table class="table table-bordered classesList" id="dataTable"  width="100%" cellspacing="0">  <thead>  <tr>  <th>Class Name</th>  <th>Total Number of Students</th>  <th>Created Dt</th>  <th>Action</th>  </tr>  </thead>  <tfoot>  <tr>  <c:forEach var="classroom" items="${classRooms}">  <tr>  <td><c:out value="${classroom.className}" /></td>  <td><c:out value="${classroom.totalNumberOfStudents}" /></td>  <td><c:out value="${classroom.createdDt}" /></td>  <td><a  href="/LearnersAcademyAdministrativePortal/classRoom?action=delete&id=<c:out value='${classroom.id}' />">Delete</a></td>  </tr>  </c:forEach>  </tr>  </tfoot>  </table>  </div>  </div>  <% if(delFlag){%>  <div>  <p>  <span style="color: red;">Record deleted successfully!</span>  </p>  </div>  <%}%>  </div>  </div>  </div>  </div>  **listClassSubjectMapping.jsp**  <%@page import="com.learnersacademy.entity.ClassSubjectMapping"%>  <%@page import="java.util.List"%>  <%@ page language="java" contentType="text/html; charset=ISO-8859-1"  pageEncoding="ISO-8859-1"%>  <%@taglib prefix="c" uri="http://java.sun.com/jstl/core"%>  <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>  <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions"%>  <!DOCTYPE html>  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" />  <%  List<ClassSubjectMapping> classSubjectMappings = (List<ClassSubjectMapping>) session.getAttribute("classSubjectMappings");  boolean delFlag = (boolean)session.getAttribute("delFlag");  %>  <div class="container-fluid">  <div class="card-header py-3">  <h6 class="m-0 font-weight-bold text-primary"><a href="pages/dashboard.jsp">Home</a>  &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;  <a href="/LearnersAcademyAdministrativePortal/logout" style='float:right'>Logout</a></h6>  </div>  <div class="d-sm-flex align-items-center justify-content-between mb-4">  <h1 class="h3 mb-0 text-gray-800">List Classes Subjects Mapping</h1>  </div>  <div class="row">  <div class="col-lg-12">  <div class="card shadow mb-4">  <div class="card-header py-3">  <h6 class="m-0 font-weight-bold text-primary">Classes Subjects  Mapping</h6>  </div>  <div class="card-body">  <div class="table-responsive">  <table class="table table-bordered classesSubjectsMappingList"  id="dataTable" width="100%" cellspacing="0">  <thead>  <tr>  <th>Class Name</th>  <th>Subject Name</th>  <th>Created Dt</th>  <th>Action</th>  </tr>  </thead>  <tfoot>  <tr>  <c:forEach var="classSubjectMapping" items="${classSubjectMappings}">  <tr>  <td><c:out value="${classSubjectMapping.className}" /></td>  <td><c:out value="${classSubjectMapping.subjectName}" /></td>  <td><c:out value="${classSubjectMapping.createdDt}" /></td>  <td><a href="/LearnersAcademyAdministrativePortal/classSubjectMapping?action=delete&id=<c:out value='${classSubjectMapping.id}' />">Delete</a></td>  </tr>  </c:forEach>    </tr>  </tfoot>  </table>  </div>  </div>  <% if(delFlag){%>  <div><p><span style="color: red;">Record deleted successfully!</span></p></div>  <%}%>  </div>  </div>  </div>  </div>  **list-students.jsp**  <%@page import="com.learnersacademy.entity.Student"%>  <%@page import="java.util.List"%>  <%@ page language="java" contentType="text/html; charset=ISO-8859-1"  pageEncoding="ISO-8859-1"%>  <%@taglib prefix="c" uri="http://java.sun.com/jstl/core"%>  <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>  <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions"%>  <!DOCTYPE html>  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" />  <%  List<Student> students = (List<Student>) session.getAttribute("students");  boolean delFlag = (boolean)session.getAttribute("delFlag");  %>  <div class="container-fluid">  <div class="card-header py-3">  <h6 class="m-0 font-weight-bold text-primary"><a href="pages/dashboard.jsp">Home</a>  &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;  <a href="/LearnersAcademyAdministrativePortal/logout" style='float:right'>Logout</a></h6>  </div>  <div class="d-sm-flex align-items-center justify-content-between mb-4">  <h1 class="h3 mb-0 text-gray-800">List Students</h1> </div>  <div class="row">  <div class="col-lg-12">  <div class="card shadow mb-4">  <div class="card-header py-3">  <h6 class="m-0 font-weight-bold text-primary">Students</h6>  </div>  <div class="card-body">  <div class="table-responsive">  <table class="table table-bordered studentsList" id="dataTable" width="100%" cellspacing="0">  <thead>  <tr>  <th>Student Name</th>  <th>Class Name</th>  <th>Contact Number</th>  <th>Blood Group</th>  <th>Created Dt</th>  <th>Action</th>  </tr>  </thead>  <tfoot>  <tr>  <c:forEach var="student" items="${students}">  <tr>  <td><c:out value="${student.name}" /></td>  <td><c:out value="${student.className}" /></td>  <td><c:out value="${student.emergencyContactNumber}" /></td>  <td><c:out value="${student.bloodGroup}" /></td>  <td><c:out value="${student.createdDt}" /></td>  <td><a href="/LearnersAcademyAdministrativePortal/student?action=delete&id=<c:out value='${student.id}' />">Delete</a></td>  </tr>  </c:forEach>    </tr>  </tfoot>  </table>  </div>  </div>  <% if(delFlag){%>  <div><p><span style="color: red;">Record deleted successfully!</span></p></div>  <%}%>  </div>  </div>  </div>  </div>  **listSubjects.jsp**  <%@page import="com.learnersacademy.entity.ClassRoom"%>  <%@page import="com.learnersacademy.entity.Student"%>  <%@page import="com.learnersacademy.entity.Subject"%>  <%@page import="java.util.List"%>  <%@ page language="java" contentType="text/html; charset=ISO-8859-1"  pageEncoding="ISO-8859-1"%>  <%@taglib prefix="c" uri="http://java.sun.com/jstl/core"%>  <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>  <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions"%>  <!DOCTYPE html>  <link rel="stylesheet"  href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" />  <%  List<Subject> subjects = (List<Subject>) session.getAttribute("subjects");  boolean delFlag = (boolean)session.getAttribute("delFlag");  %>  <div class="container-fluid">  <div class="card-header py-3">  <h6 class="m-0 font-weight-bold text-primary"><a href="pages/dashboard.jsp">Home</a>  &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;  <a href="/LearnersAcademyAdministrativePortal/logout" style='float:right'>Logout</a></h6>  </div>  <div class="d-sm-flex align-items-center justify-content-between mb-4">  <h1 class="h3 mb-0 text-gray-800">List Subjects</h1>  </div>  <div class="row">  <div class="col-lg-12">  <div class="card shadow mb-4">  <div class="card-header py-3">  <h6 class="m-0 font-weight-bold text-primary">Subjects</h6>  </div>  <div class="card-body">  <div class="table-responsive">  <table class="table table-bordered subjectsList" id="dataTable"  width="100%" cellspacing="0">  <thead>  <tr>  <th>Subject Name</th>  <th>Created Dt</th>  <th>Action</th>  </tr>  </thead>  <tfoot>  <tr>  <c:forEach var="subject" items="${subjects}">  <tr>  <td><c:out value="${subject.subjectName}" /></td>  <td><c:out value="${subject.createdDt}" /></td>  <td><a href="/LearnersAcademyAdministrativePortal/subject?action=delete&id=<c:out value='${subject.id}' />">Delete</a></td>  </tr>  </c:forEach>  </tr>  </tfoot>  </table>  </div>  </div>  <% if(delFlag){%>  <div>  <p>  <span style="color: red;">Record deleted successfully!</span>  </p>  </div>  <%}%>  </div>  </div>  </div>  </div>  **listTeacher.jsp**  <%@page import="com.learnersacademy.entity.Teacher"%>  <%@page import="java.util.List"%>  <%@ page language="java" contentType="text/html; charset=ISO-8859-1"  pageEncoding="ISO-8859-1"%>  <%@taglib prefix="c" uri="http://java.sun.com/jstl/core"%>  <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>  <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions"%>  <!DOCTYPE html>  <link rel="stylesheet"  href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" />  <%  List<Teacher> teachers = (List<Teacher>) session.getAttribute("teachers");  boolean delFlag = (boolean)session.getAttribute("delFlag");  %>  <div class="container-fluid">  <div class="card-header py-3">  <h6 class="m-0 font-weight-bold text-primary"><a href="pages/dashboard.jsp">Home</a>  &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;  <a href="/LearnersAcademyAdministrativePortal/logout" style='float:right'>Logout</a></h6>  </div>  <div class="d-sm-flex align-items-center justify-content-between mb-4">  <h1 class="h3 mb-0 text-gray-800">List Teachers</h1>  </div>  <div class="row">  <div class="col-lg-12">  <div class="card shadow mb-4">  <div class="card-header py-3">  <h6 class="m-0 font-weight-bold text-primary">Teachers</h6>  </div>  <div class="card-body">  <div class="table-responsive">  <table class="table table-bordered teachersList" id="dataTable"  width="100%" cellspacing="0">  <thead>  <tr>  <th>Teacher Name</th>  <th>Contact Number</th>  <th>Qualification</th>  <th>Gender</th>  <th>Created Dt</th>  <th>Action</th>  </tr>  </thead>  <tfoot>  <tr>  <c:forEach var="teacher" items="${teachers}">  <tr>  <td><c:out value="${teacher.firstName}, ${teacher.lastName}" /></td>  <td><c:out value="${teacher.contactNumber}" /></td>  <td><c:out value="${teacher.qualification}" /></td>  <td><c:out value="${teacher.gender}" /></td>  <td><c:out value="${teacher.createdDt}" /></td>  <td><a  href="/LearnersAcademyAdministrativePortal/teacher?action=delete&id=<c:out value='${teacher.id}' />">Delete</a></td>  </tr>  </c:forEach>  </tr>  </tfoot>  </table>  </div>  </div>  <% if(delFlag){%>  <div>  <p>  <span style="color: red;">Record deleted successfully!</span>  </p>  </div>  <%}%>  </div>  </div>  </div>  </div>  **listTeacherClassSubjectMapping.jsp**  <%@page import="com.learnersacademy.entity.TeacherClassSubjectMapping"%>  <%@page import="java.util.List"%>  <%@ page language="java" contentType="text/html; charset=ISO-8859-1"  pageEncoding="ISO-8859-1"%>  <%@taglib prefix="c" uri="http://java.sun.com/jstl/core"%>  <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>  <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions"%>  <!DOCTYPE html>  <link rel="stylesheet"  href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" />  <%  List<TeacherClassSubjectMapping> teacherClassSubjectMappings = (List<TeacherClassSubjectMapping>) session.getAttribute("teacherClassSubjectMappings");  boolean delFlag = (boolean)session.getAttribute("delFlag");  %>  <div class="container-fluid">  <div class="card-header py-3">  <h6 class="m-0 font-weight-bold text-primary"><a href="pages/dashboard.jsp">Home</a>  &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;  <a href="/LearnersAcademyAdministrativePortal/logout" style='float:right'>Logout</a></h6>  </div>  <div class="d-sm-flex align-items-center justify-content-between mb-4">  <h1 class="h3 mb-0 text-gray-800">List Teachers Classes Subjects Mapping</h1>  </div>  <div class="row">  <div class="col-lg-12">  <div class="card shadow mb-4">  <div class="card-header py-3">  <h6 class="m-0 font-weight-bold text-primary">Teachers Classes Subjects Mapping</h6>  </div>  <div class="card-body">  <div class="table-responsive">  <table  class="table table-bordered teachersClassesSubjectsMappingList"  id="dataTable" width="100%" cellspacing="0">  <thead>  <tr>  <th>Teacher Name</th>  <th>Class Name</th>  <th>Subject Name</th>  <th>Created Dt</th>  <th>Action</th>  </tr>  </thead>  <tfoot>  <tr>  <c:forEach var="teacherClassSubjectMapping" items="${teacherClassSubjectMappings}">  <tr>  <td><c:out value="${teacherClassSubjectMapping.teacherName}" /></td>  <td><c:out value="${teacherClassSubjectMapping.className}" /></td>  <td><c:out value="${teacherClassSubjectMapping.subjectName}" /></td>  <td><c:out value="${teacherClassSubjectMapping.createdDt}" /></td>  <td><a href="/LearnersAcademyAdministrativePortal/teacherClassSubjectMapping?action=delete&id=<c:out value='${teacherClassSubjectMapping.id}' />">Delete</a></td>  </tr>  </c:forEach>  </tr>  </tfoot>  </table>  </div>  </div>  <% if(delFlag){%>  <div>  <p>  <span style="color: red;">Record deleted successfully!</span>  </p>  </div>  <%}%>  </div>  </div>  </div>  </div>  **login.jsp**  <%@ page language="java" contentType="text/html; charset=ISO-8859-1"  pageEncoding="ISO-8859-1"%>  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">  <title>Login Page</title>  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" />  </head>  <body>  <center>  <h1>Learner's Academy</h1>  <div class="container fluid">  <div>  <h4>Login Form</h4>  <form action="/LearnersAcademyAdministrativePortal/login" class="was-validated">  <div class="form-group">  <label for="email">Username</label> <input type="text"  class="form-control" id="email" name="username"  placeholder="Enter your username" required />  <div class="valid-feedback">Valid.</div>  <div class="invalid-feedback">Please fill out this field</div>  </div>  <div class="form-group">  <label for="password">Password</label> <input type="password"  class="form-control" id="password" name="password"  placeholder="Enter your password" required minlength="3" />  <div class="valid-feedback">Valid.</div>  <div class="invalid-feedback">Please fill out this field</div>  </div>  <div class="form-group form-check">  <label class="form-check-label"></label><input type="checkbox"  class="form-check-input" name="remember" required /> I agree  terms and conditions.  <div class="valid-feedback">Valid.</div>  <div class="invalid-feedback">Check this checkbox to  continue.</div>  </div>  <button type="submit" class="btn btn-primary">Login</button>  </form>  <hr />  </div>  </div>  </center>  </body>  </html>  **menu.jsp**  <%@ page language="java" contentType="text/html; charset=ISO-8859-1"  pageEncoding="ISO-8859-1"%>  <!DOCTYPE html>  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">  <link href="/LearnersAcademyAdministrativePortal/css/menuNavigation.css" rel="stylesheet" type="text/css">  <header>  <div class="navbar">  <div class="dropdown">  <button class="dropbtn">Student<i class="fa fa-caret-down"></i></button>  <div class="dropdown-content">  <a href="/LearnersAcademyAdministrativePortal/classRoom?action=listClassName">Add Student</a>  <a href="/LearnersAcademyAdministrativePortal/student?action=list">List Students</a>  </div>  </div>  <div class="dropdown">  <button class="dropbtn">ClassRoom<i class="fa fa-caret-down"></i></button>  <div class="dropdown-content">  <a href="/LearnersAcademyAdministrativePortal/classRoom?action=addJsp">Add Class</a>  <a href="/LearnersAcademyAdministrativePortal/classRoom?action=list">List Classes</a>  </div>  </div>  <div class="dropdown">  <button class="dropbtn">Teacher<i class="fa fa-caret-down"></i></button>  <div class="dropdown-content">  <a href="/LearnersAcademyAdministrativePortal/teacher?action=addJsp">Add Teachers</a>  <a href="/LearnersAcademyAdministrativePortal/teacher?action=list">List Teachers</a>  </div>  </div>  <div class="dropdown">  <button class="dropbtn">Subject<i class="fa fa-caret-down"></i></button>  <div class="dropdown-content">  <a href="/LearnersAcademyAdministrativePortal/subject?action=addJsp">Add Subjects</a>  <a href="/LearnersAcademyAdministrativePortal/subject?action=list">List Subjects</a>  </div>  </div>  <div class="dropdown">  <button class="dropbtn">Classes Mapping<i class="fa fa-caret-down"></i></button>  <div class="dropdown-content">  <a href="/LearnersAcademyAdministrativePortal/classSubjectMapping?action=mappingDetails">Class Subject Mapping</a>  <a href="/LearnersAcademyAdministrativePortal/classSubjectMapping?action=list">List Mapped Class-Subject</a>  </div>  </div>  <div class="dropdown">  <button class="dropbtn">Teachers Mapping<i class="fa fa-caret-down"></i></button>  <div class="dropdown-content">  <a href="/LearnersAcademyAdministrativePortal/teacherClassSubjectMapping?action=mappingDetails">Teacher Class Mapping</a>  <a href="/LearnersAcademyAdministrativePortal/teacherClassSubjectMapping?action=list">List Mapped Teacher-Class</a>  </div>  </div>  <div class="dropdown">  <button class="dropbtn">Class Reports<i class="fa fa-caret-down"></i></button>  <div class="dropdown-content">  <a href="/LearnersAcademyAdministrativePortal/classRoom?action=list&reportFlag=Y">Class Report Details</a>  </div>  </div>  <a href="/LearnersAcademyAdministrativePortal/logout">Log out</a>  </div>  </header>  **SCREENSHOTS**  **1.LOGIN**  **Screenshot (158).png**  **2.HOME Page**  **Screenshot (159).png**  **3.LIST CLASSES**  **Screenshot (160).png**  **4.LIST SUBJECTS**  **Screenshot (161).png**  **5.LIST TEACHERS**  **Screenshot (163).png**  **6.LIST SUBJECTS**  **Screenshot (161).png**  **7.LIST CLASSREPORT**  **Screenshot (162).png**  **8.LIST CLASSDETAILSREPORT**  **Screenshot (166).png**  **9.LIST TEACHERSCLASSESSUBJECTMAING**  **Screenshot (168).png**  **10.LOGOUT**  **Screenshot (172).png** |