NAME:	**SNEHITHA VANKAYALA**
BATCH:	**MS FSD DEC 2021 COHORT1 (Phase2)**
PROJECT TITLE:	**LEARNER'S ACADEMY**
SUBMISSION DATE:	**27th MARCH 2022**

Project'Details

This project aims to design and develop a backend administrative portal for the Learner's Academy using Java EE technologies. I developed it as a project of phase 2 for the Become a back-end expert course. The goal of this project is to apply servlet, jsp and JDBC concepts.

Product Backlog:

1. Create database and tables.
2. Connect the database to the project.
3. Create models classes.
4. Create a database utility class to retrieve data.
5. Create login page.
6. Create JSP files for all pages of the project.
7. Create a servlet to get requests and send responses to the JSP files.
8. Add cookies.
9. Create a CSS file to format the contents.
10. Debug and Test the project.

Technologies and tools Used

• Servlet: to do the business logic and works a controller for the project.

• JSP: to handle the presentation view.

• SQL: to create and manage the database.

• JDBC: to make operations on the database for the project.

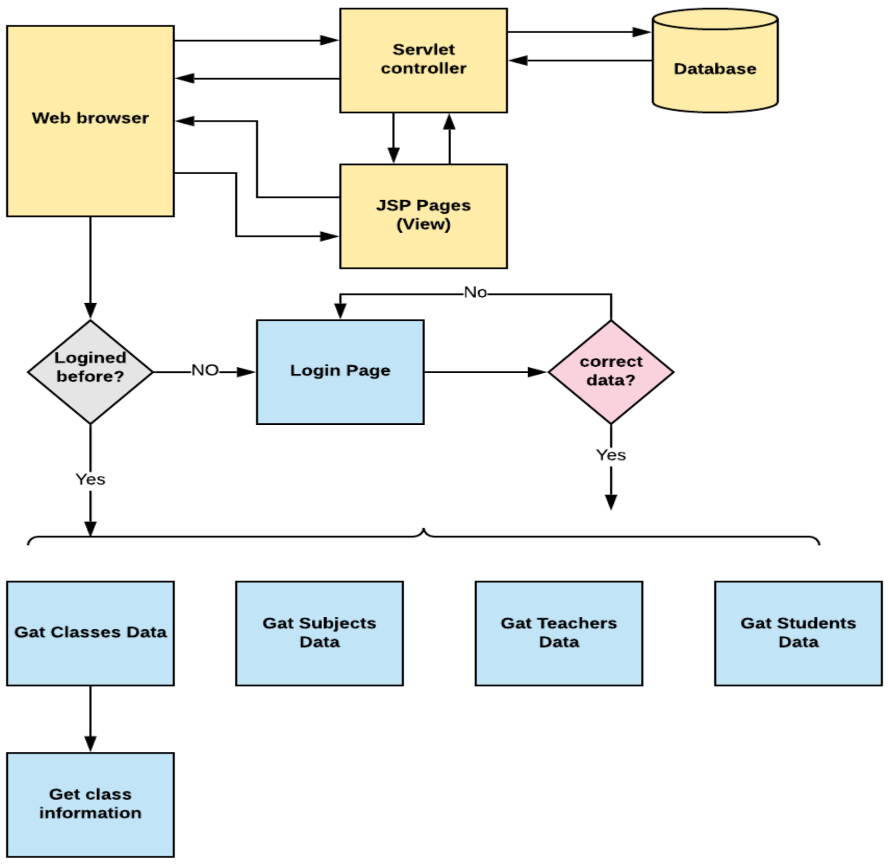
• CSS: to format the contents.

• phpMyAdmin: to administrate and manage the database manually.

• Eclipse: to write and run the code.

• Tomcat: to run and deploy servlet application.

FLOW CHART



SOURCE_CODE

<u>Learnersacademy</u>

<u>Dao</u>

<u>ClassReportDetailsDao</u>

```
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.projectio
nList()
					.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();

		}

	}


}
```

<u>SubjectDao</u>

```
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;
	}
}
```

<u>Subject</u>

```
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;

        }


}
```

<u>TeacherClassSubjectMapping</u>

```
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 +
"]";
	}
```

```
}
```

<u>SERVLETS</u>

<u>ClassReportDetailsServlet</u>

```
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);

	}


}
```

<u>LoginServlet</u>

```
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

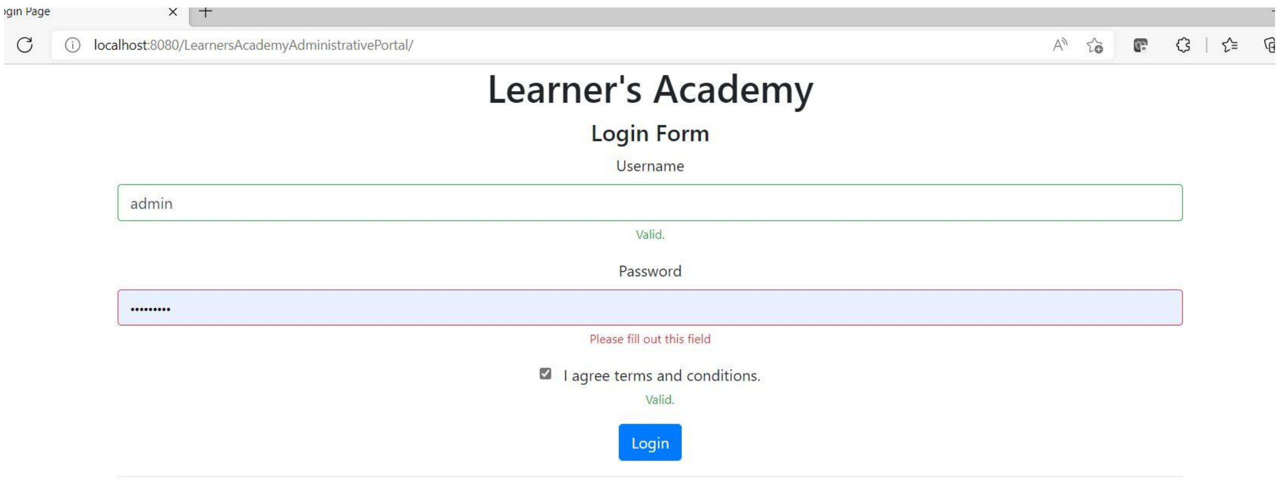
HibernateUtil

```
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;
          }

}
```

SCREENSHOTS

1.



Learner's Academy

Login Form

Username

admin

Valid.

Password

•••••••••

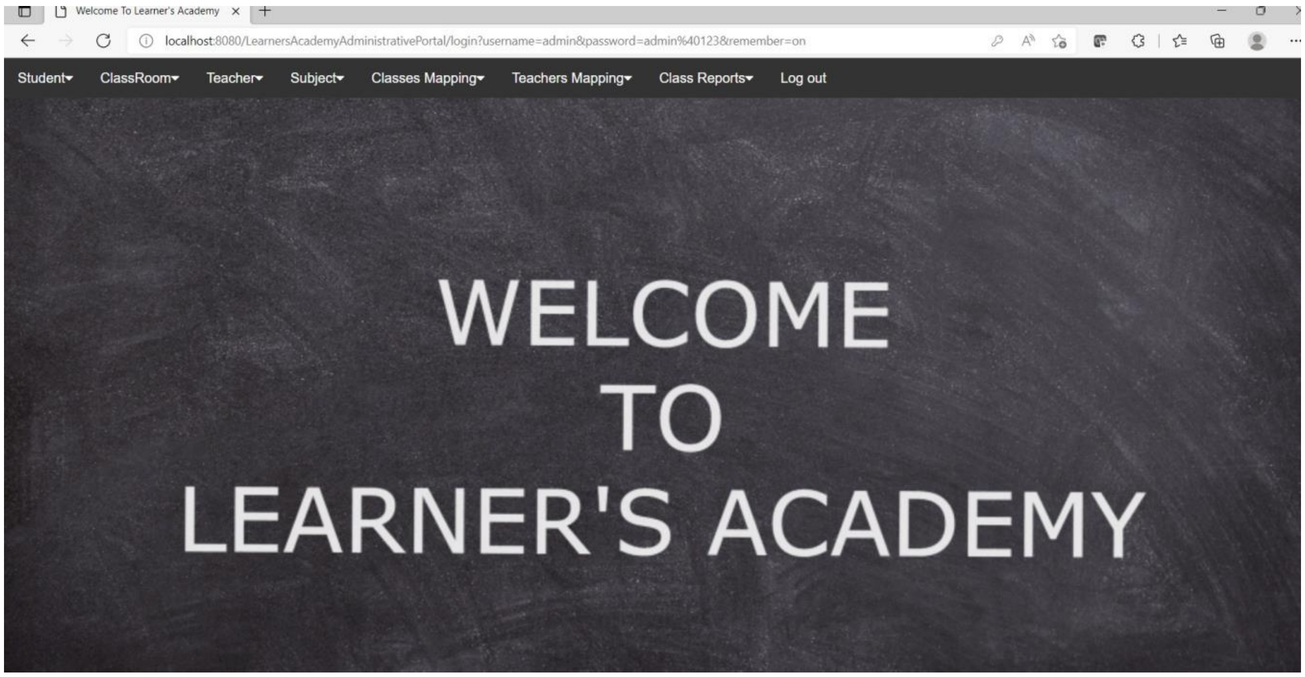
Please fill out this field

I agree terms and conditions.

Valid.

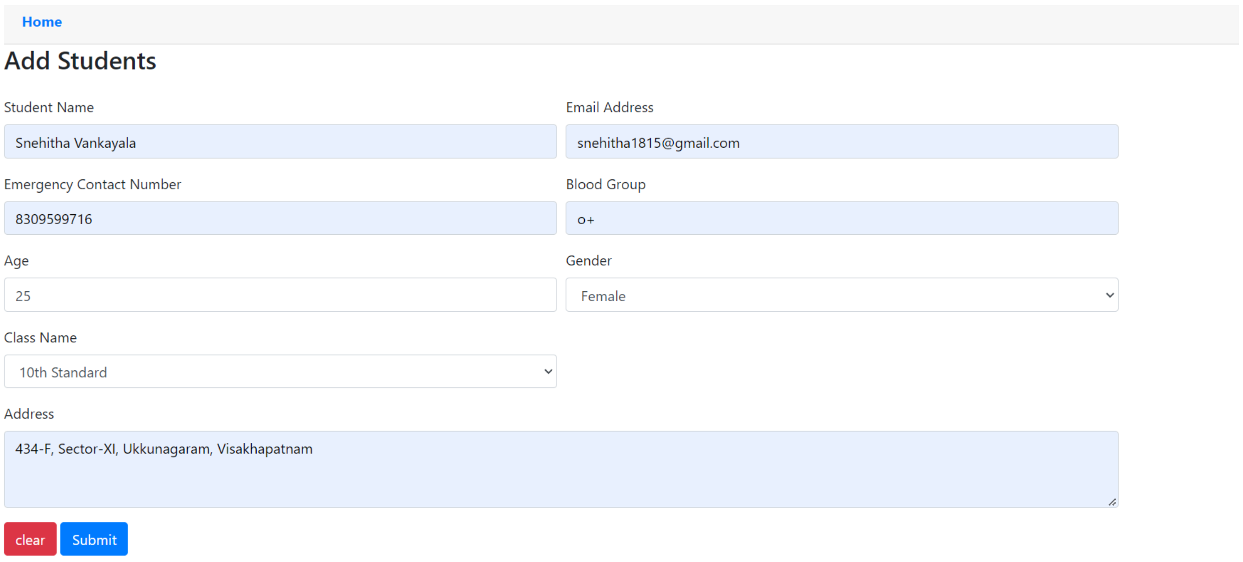
Login

2.



Welcome To Learner's Academy
localhost:8080/LearnersAcademyAdministrativePortal/login?username=admin&password=admin%40123&remember=on
Student
ClassRoom
Teacher
Subject
Classes Mapping
Teachers Mapping
Class Reports
Log out
WELCOME
TO
LEARNER'S ACADEMY

3.



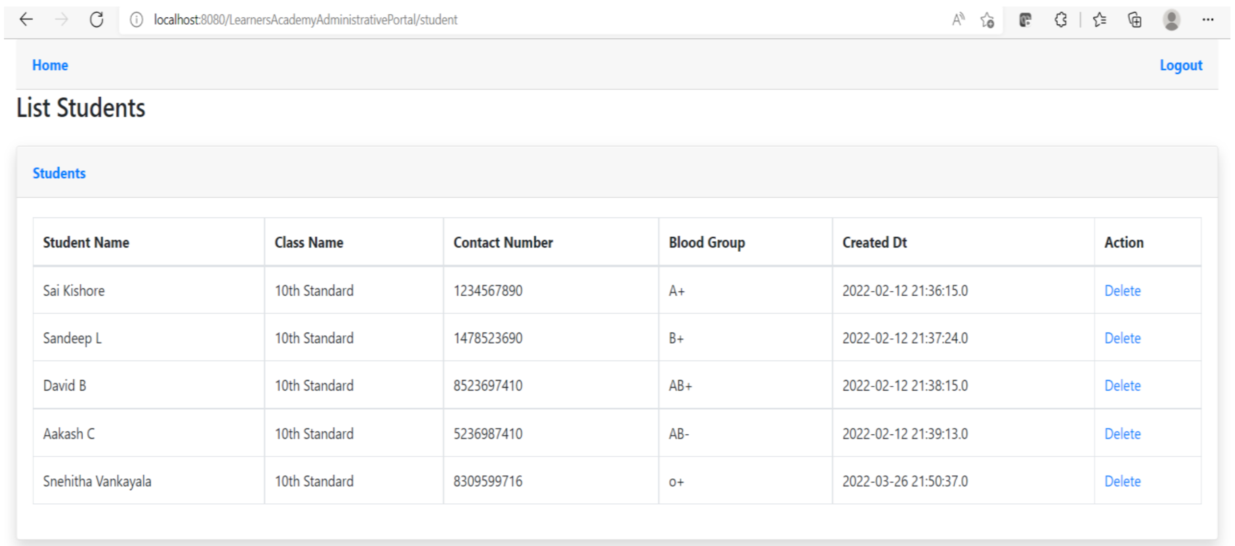
Home

Add Students

Student Name: Snehitha Vankayala	Email Address: snehitha1815@gmail.com
Emergency Contact Number: 8309599716	Blood Group: o+
Age: 25	Gender: Female
Class Name: 10th Standard	

Address: 434-F, Sector-XI, Ukkunagaram, Visakhapatnam

4.



localhost:8080/LearnersAcademyAdministrativePortal/student

Home

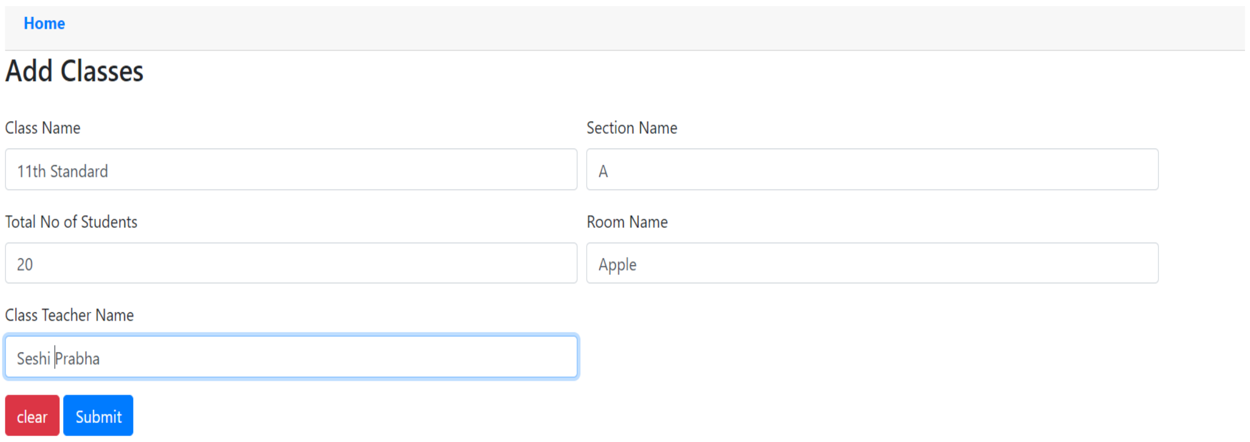
Logout

List Students

Students

Student Name	Class Name	Contact Number	Blood Group	Created Dt	Action
Sai Kishore	10th Standard	1234567890	A+	2022-02-12 21:36:15.0	Delete
Sandeep L	10th Standard	1478523690	B+	2022-02-12 21:37:24.0	Delete
David B	10th Standard	8523697410	AB+	2022-02-12 21:38:15.0	Delete
Aakash C	10th Standard	5236987410	AB-	2022-02-12 21:39:13.0	Delete
Snehitha Vankayala	10th Standard	8309599716	o+	2022-03-26 21:50:37.0	Delete

5.

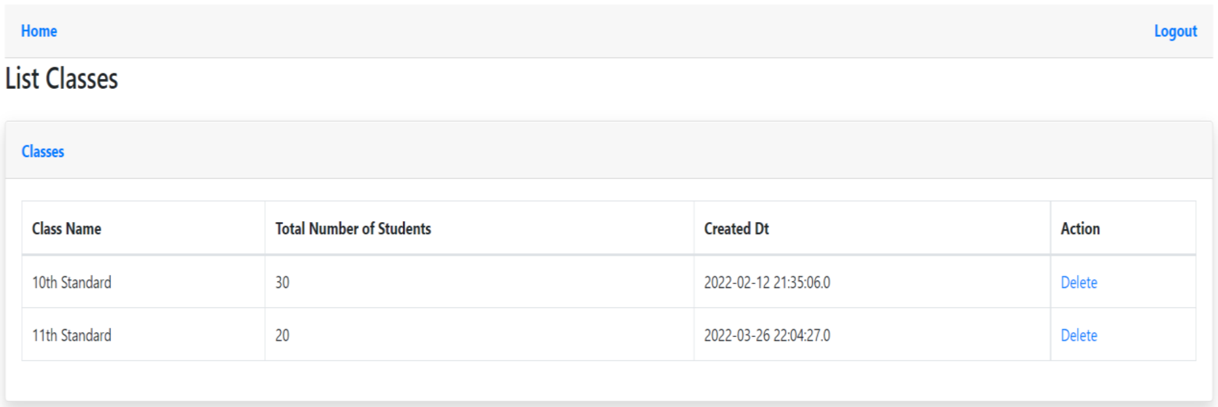


Home

Add Classes

Class Name	Section Name
11th Standard	A
Total No of Students	Room Name
20	Apple
Class Teacher Name	
Seshi Prabha	

6.



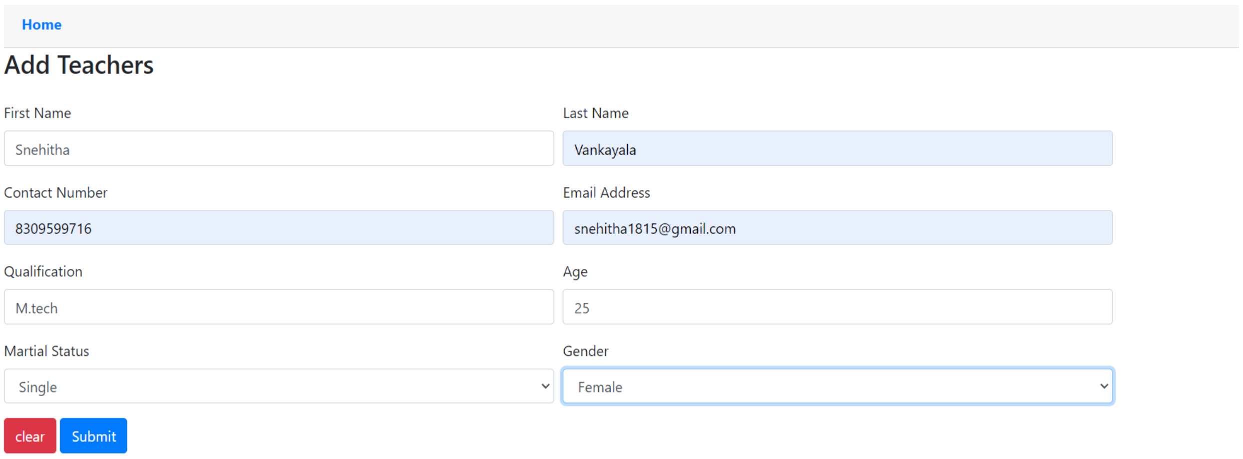
Home Logout

List Classes

Classes

Class Name	Total Number of Students	Created Dt	Action
10th Standard	30	2022-02-12 21:35:06.0	Delete
11th Standard	20	2022-03-26 22:04:27.0	Delete

7.

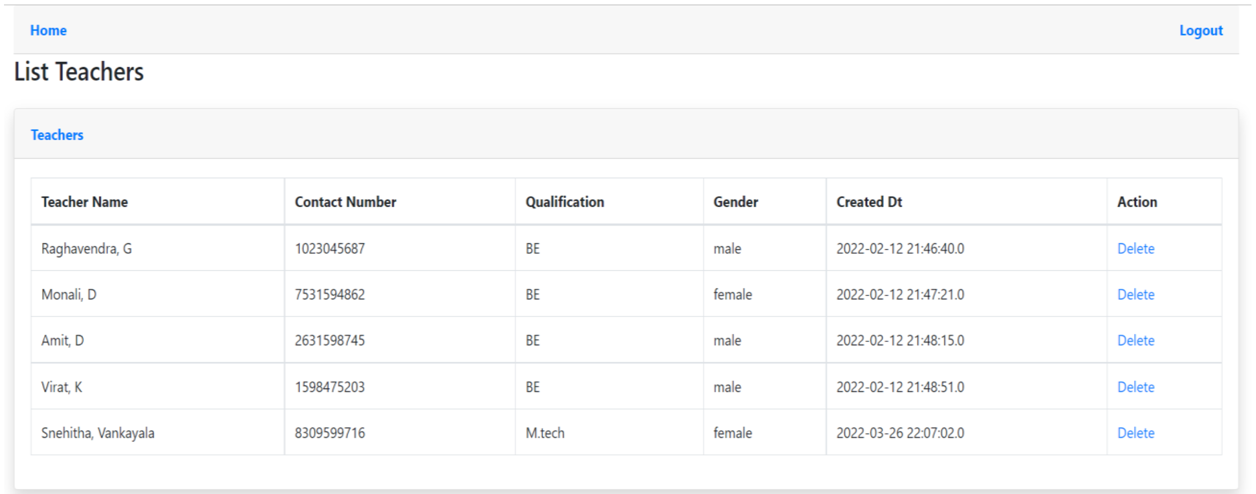


Home

Add Teachers

First Name: Snehitha	Last Name: Vankayala
Contact Number: 8309599716	Email Address: snehitha1815@gmail.com
Qualification: M.tech	Age: 25
Martial Status: Single	Gender: Female

8.

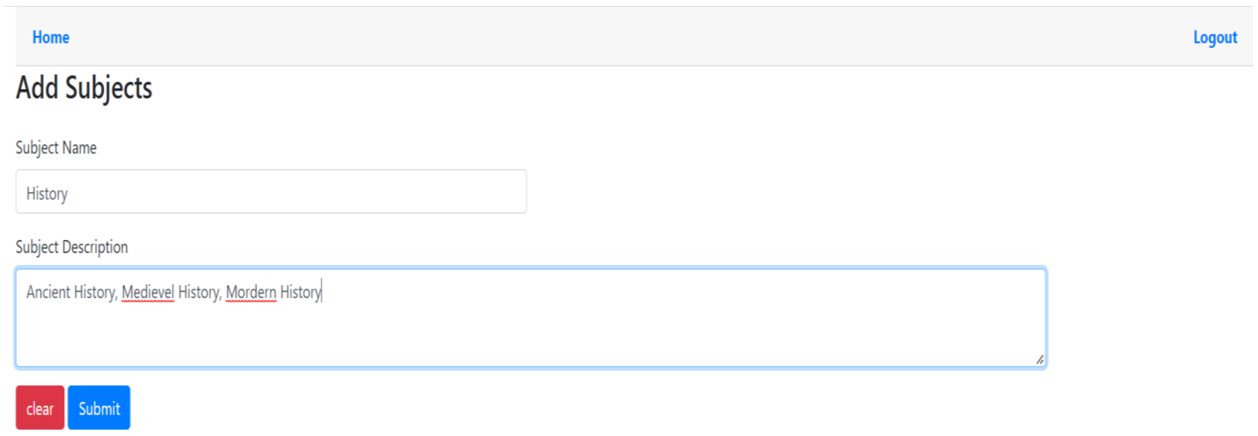


List Teachers

Teachers

Teacher Name	Contact Number	Qualification	Gender	Created Dt	Action
Raghavendra, G	1023045687	BE	male	2022-02-12 21:46:40.0	Delete
Monali, D	7531594862	BE	female	2022-02-12 21:47:21.0	Delete
Amit, D	2631598745	BE	male	2022-02-12 21:48:15.0	Delete
Virat, K	1598475203	BE	male	2022-02-12 21:48:51.0	Delete
Snehitha, Vankayala	8309599716	M.tech	female	2022-03-26 22:07:02.0	Delete

9.



Add Subjects

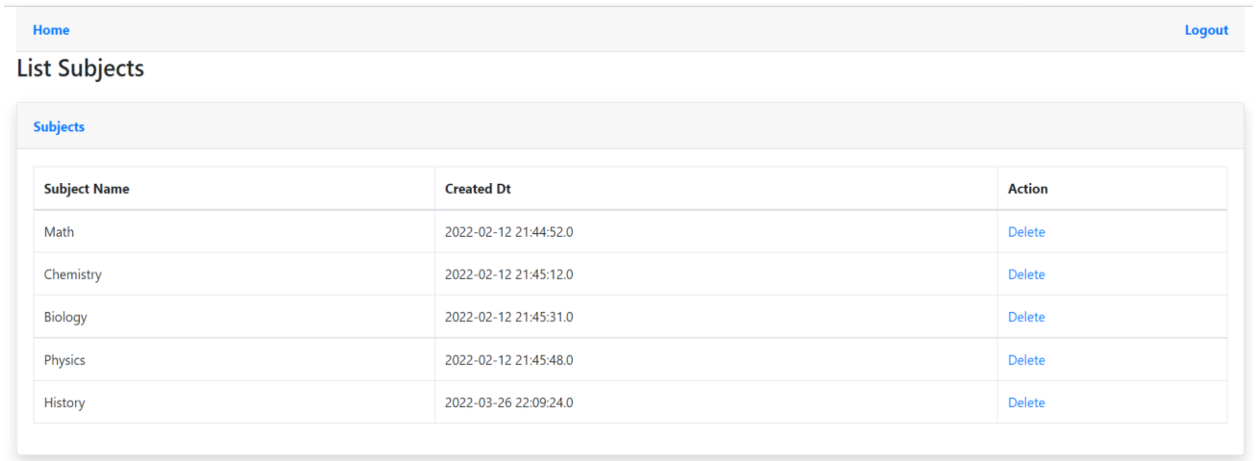
Subject Name

History

Subject Description

Ancient History, Medievel History, Mordern History

10.

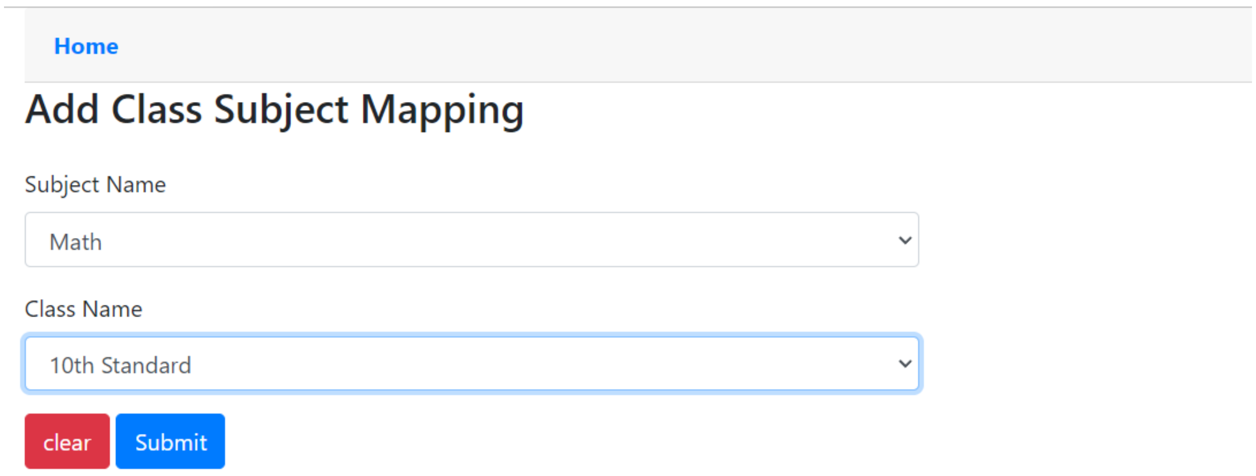


List Subjects

Subjects

Subject Name	Created Dt	Action
Math	2022-02-12 21:44:52.0	Delete
Chemistry	2022-02-12 21:45:12.0	Delete
Biology	2022-02-12 21:45:31.0	Delete
Physics	2022-02-12 21:45:48.0	Delete
History	2022-03-26 22:09:24.0	Delete

11.

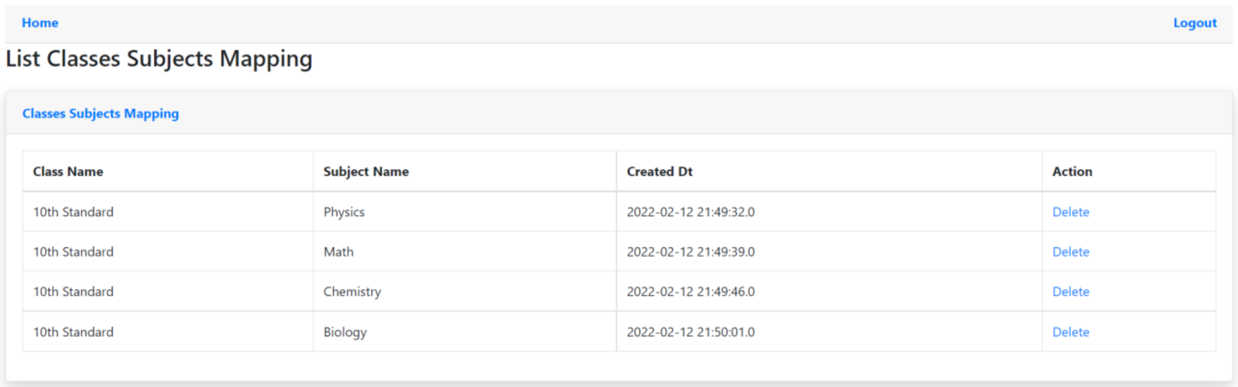


Add Class Subject Mapping

Subject Name: Math

Class Name: 10th Standard

12.

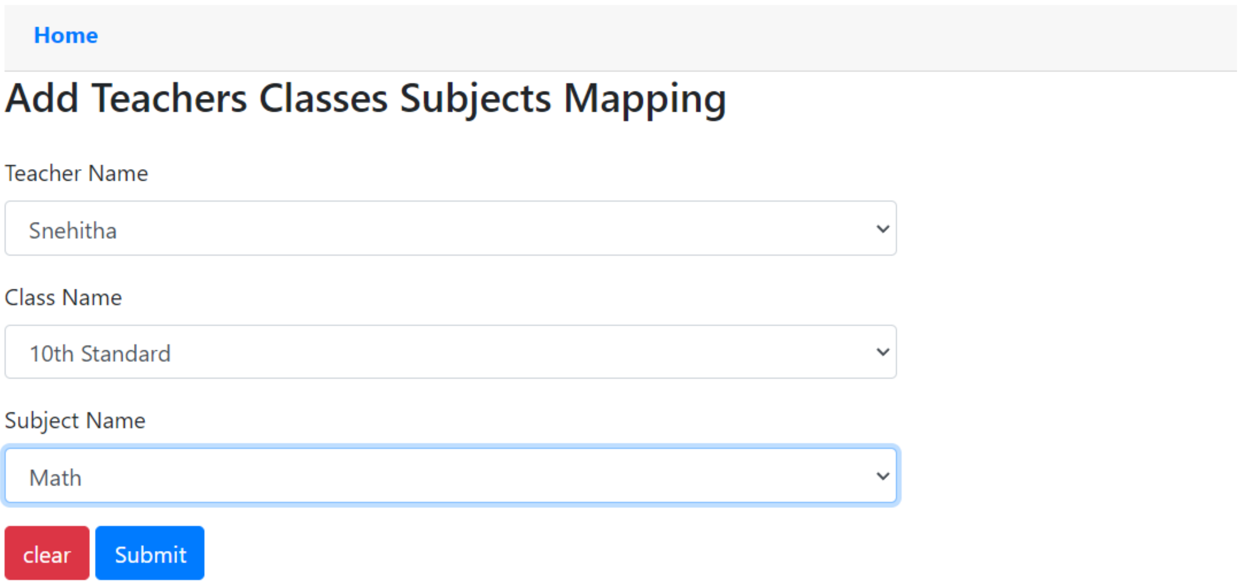


List Classes Subjects Mapping

Classes Subjects Mapping

Class Name	Subject Name	Created Dt	Action
10th Standard	Physics	2022-02-12 21:49:32.0	Delete
10th Standard	Math	2022-02-12 21:49:39.0	Delete
10th Standard	Chemistry	2022-02-12 21:49:46.0	Delete
10th Standard	Biology	2022-02-12 21:50:01.0	Delete

13.



Home

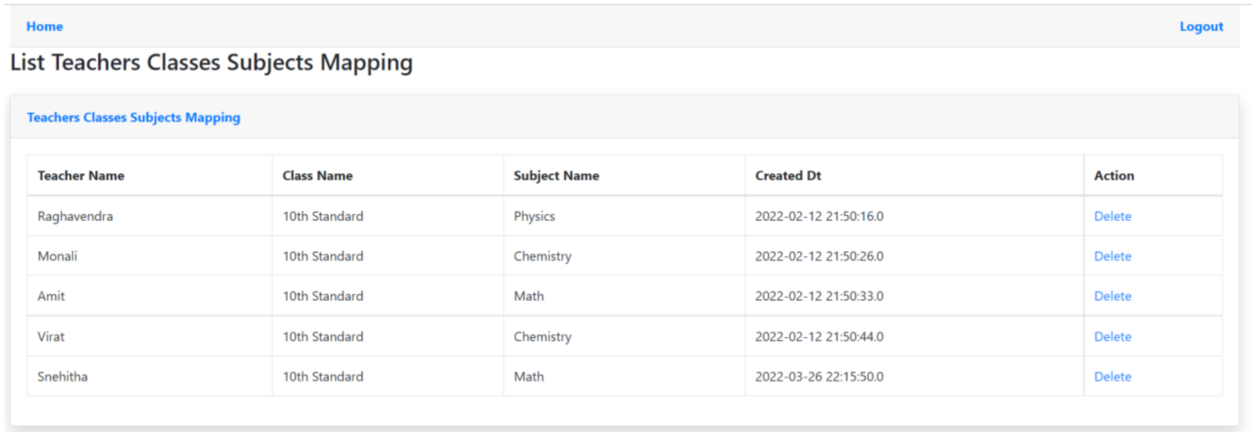
Add Teachers Classes Subjects Mapping

Teacher Name: Snehitha

Class Name: 10th Standard

Subject Name: Math

14.



Home

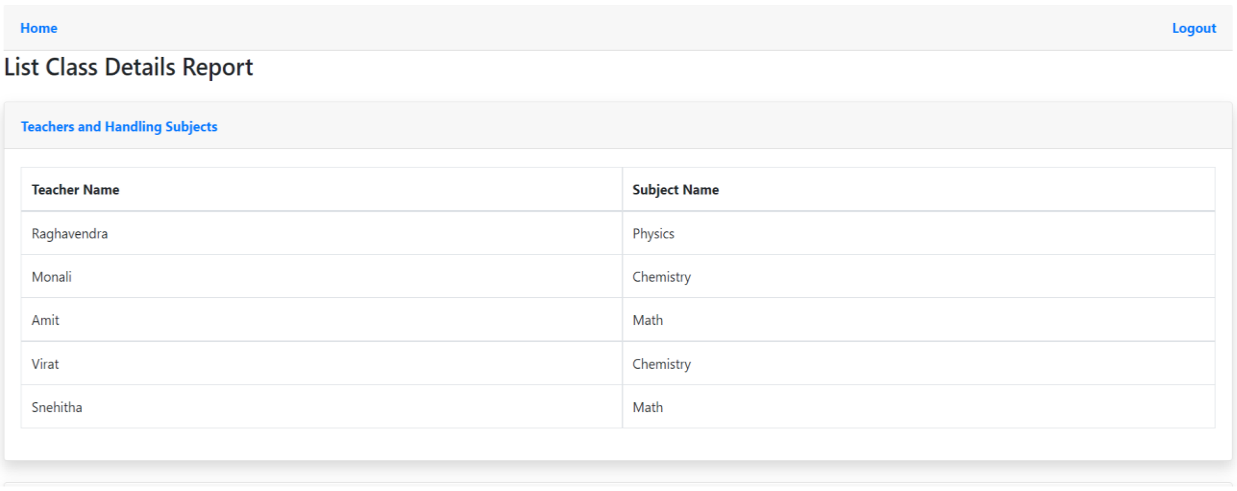
Logout

List Teachers Classes Subjects Mapping

Teachers Classes Subjects Mapping

Teacher Name	Class Name	Subject Name	Created Dt	Action
Raghavendra	10th Standard	Physics	2022-02-12 21:50:16.0	Delete
Monali	10th Standard	Chemistry	2022-02-12 21:50:26.0	Delete
Amit	10th Standard	Math	2022-02-12 21:50:33.0	Delete
Virat	10th Standard	Chemistry	2022-02-12 21:50:44.0	Delete
Snehitha	10th Standard	Math	2022-03-26 22:15:50.0	Delete

15.



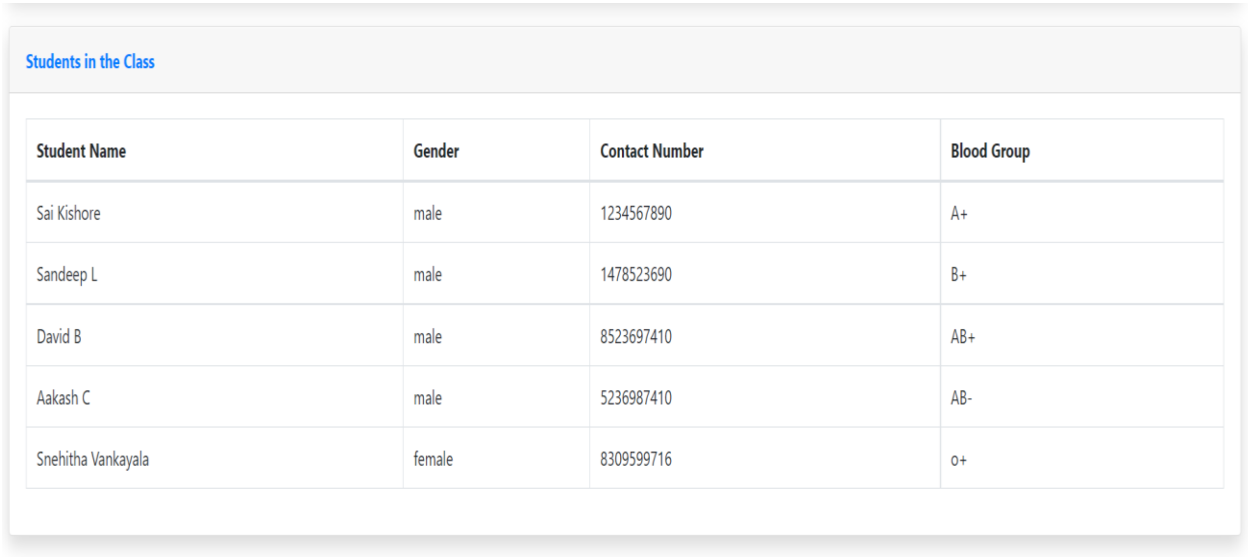
Home Logout

List Class Details Report

Teachers and Handling Subjects

Teacher Name	Subject Name
Raghavendra	Physics
Monali	Chemistry
Amit	Math
Virat	Chemistry
Snehitha	Math

16.



Students in the Class

Student Name	Gender	Contact Number	Blood Group
Sai Kishore	male	1234567890	A+
Sandeep L	male	1478523690	B+
David B	male	8523697410	AB+
Aakash C	male	5236987410	AB-
Snehitha Vankayala	female	8309599716	o+