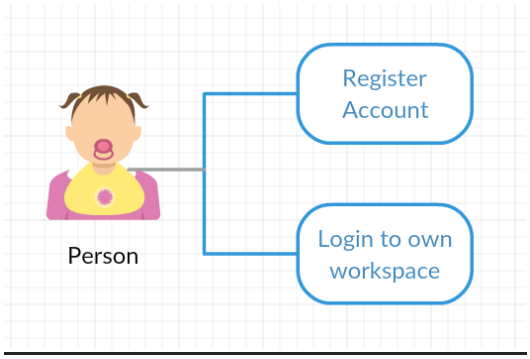
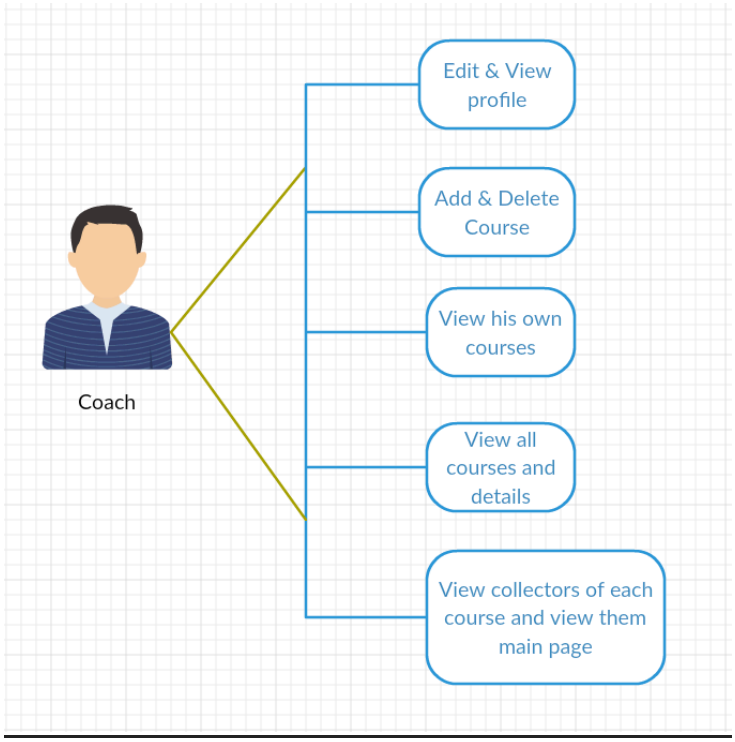
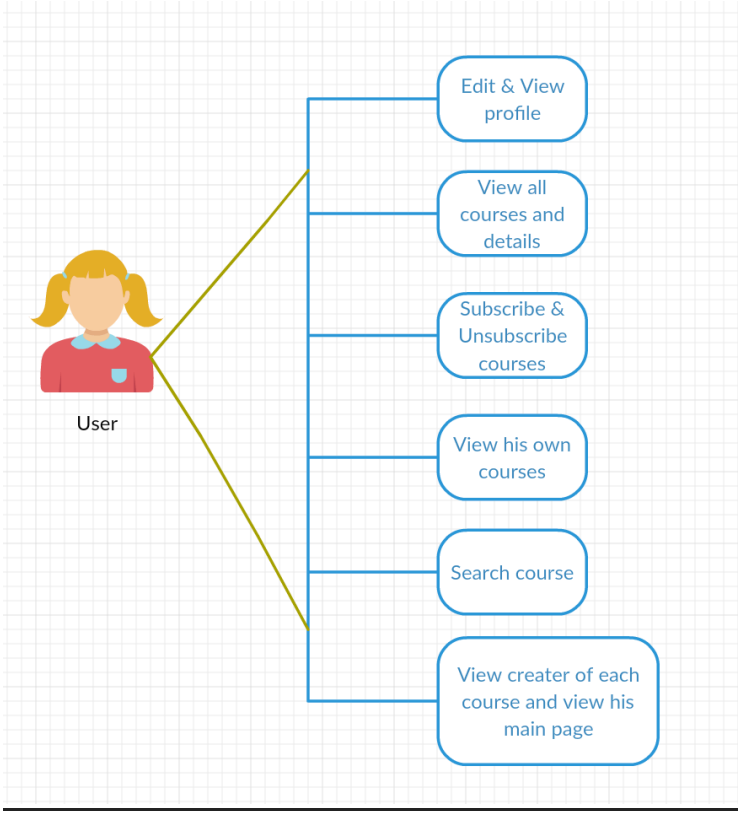
Summary of the project:

My project is a fitness application. I provide a platform for workout coach who can post free workout video courses. In addition, users can access all the video posted by the coaches and subscribe these courses. The course is a communication bridge between users and coaches.

Summary of the Functionality Performed:

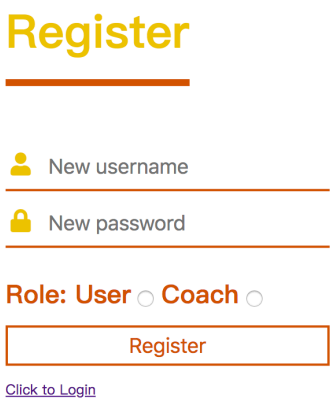


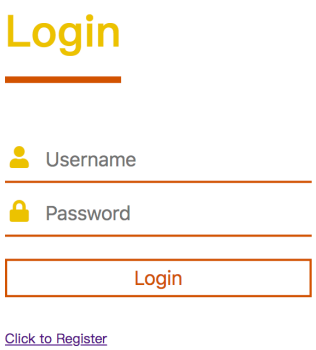




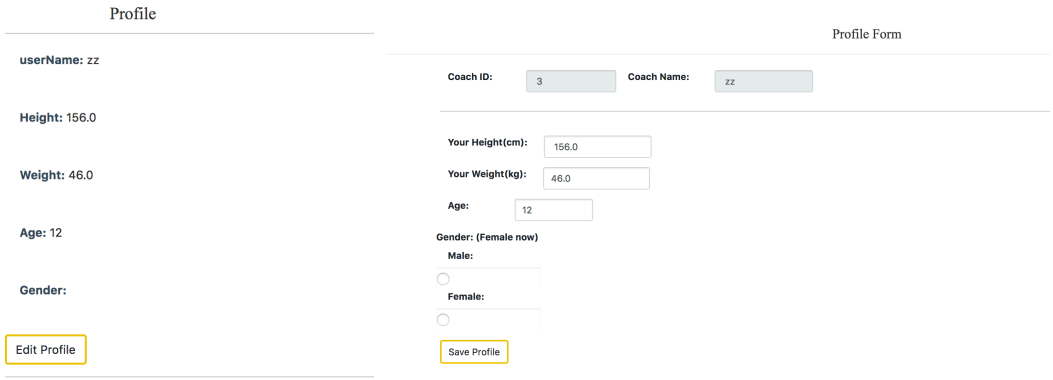
Description of roles and tasks + Screenshots:

1. Register (Select role) + Login



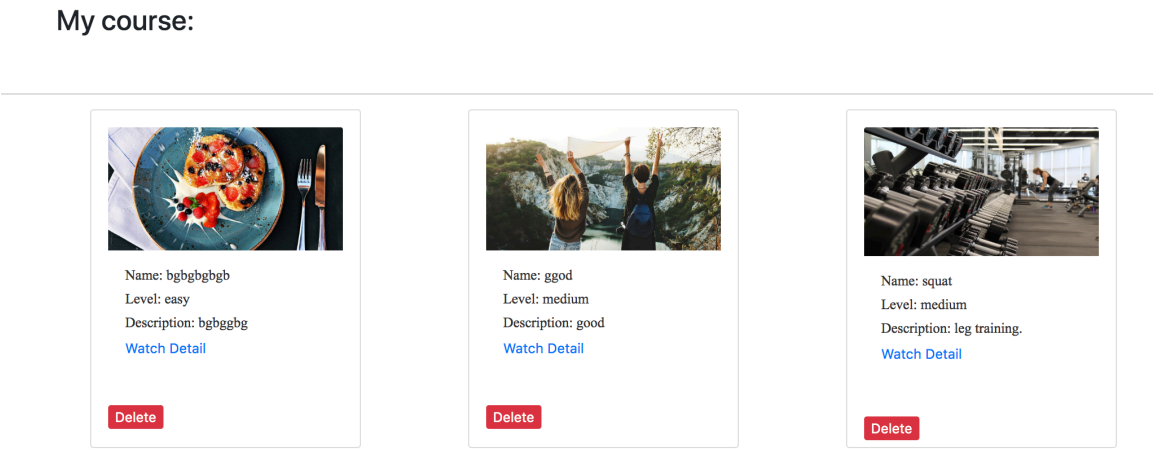


2. Coach role +User role (View profile + Edit profile)

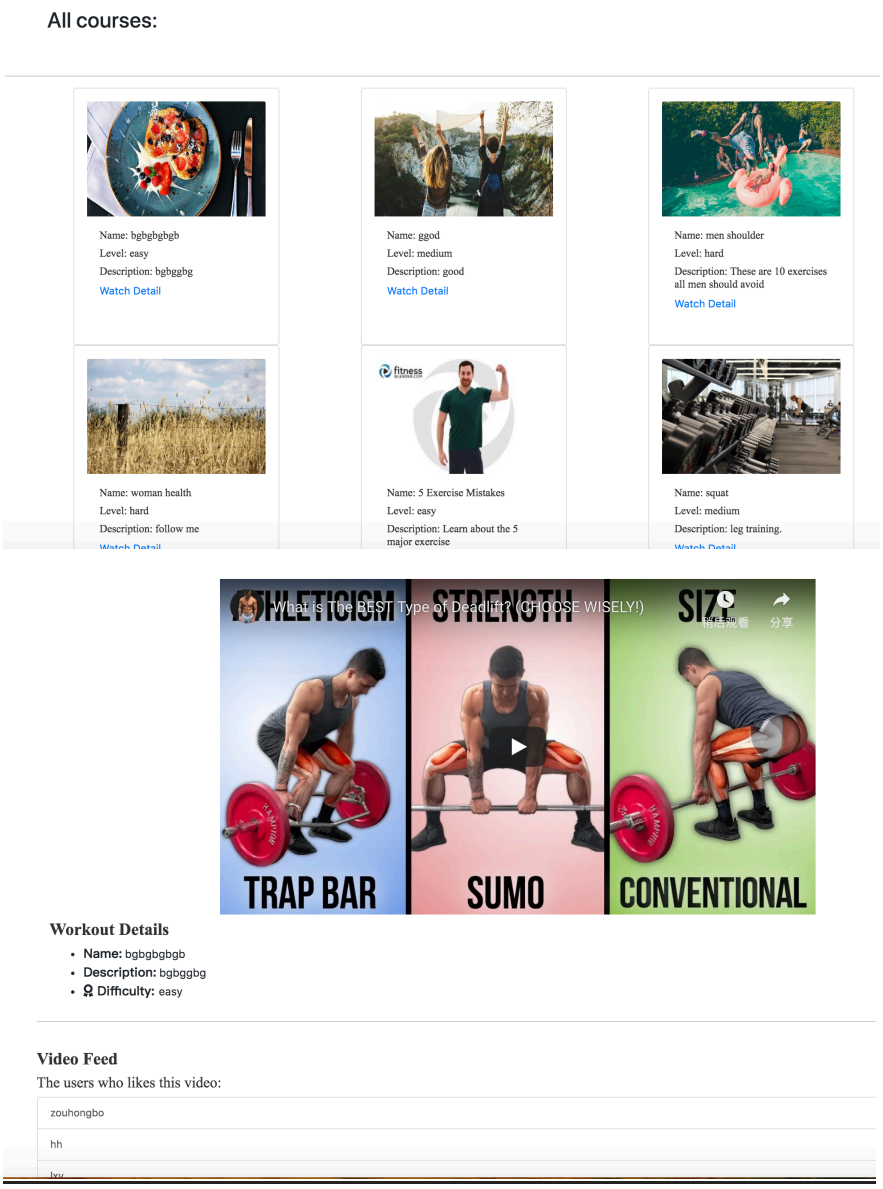


3. Coach role: add course

4. Coach role: after adding a new course, see his courses, can delete course on the page



5. Coach Role: see all course on the application, click to see details



6. Coach Role: click user's name to see user's main page (favorite courses + basic information)

User zouhongbo's Main page:

Favorite Course:

Name: bgbgbgbgb
Level: easy
Description: bgbggbg
Watch Detail

Name: men shoulder
Level: hard
Description: These are 10 exercises all men should avoid
Watch Detail

Basic information:

- **Name:** zouhongbo
- **Height:** 1.0
- **Weight:** 2.0
- **Age:** 34
- **Gender:** Male

7. User role: Search courses by search content of difficulty level of courses

All courses:

Search | Search

Easy Level | Medium Level | Hard Level

Name: bgbgbgbgb
Level: easy
Description: bgbggbg
Watch Detail

Name: ggod
Level: medium
Description: good
Watch Detail

Name: men shoulder
Level: hard
Description: These are 10 exercises all men should avoid
Watch Detail

8. User Role: click see course detail to see creator's main page (posted courses + basic information) + Subscribe this course

Workout Details

- **Name:** bgbgbgbgb
- **Description:** bgbggbg
- **Difficulty:** easy

Subscirbe this course!

Want to see this creater's Main page?

Coach zz's Main page:

Favorite Course:

Name: bgbgbgbgb
Level: easy
Description: bgbggbg
Watch Detail

Name: ggod
Level: medium
Description: good
Watch Detail

Name: squat
Level: medium
Description: leg training.
Watch Detail

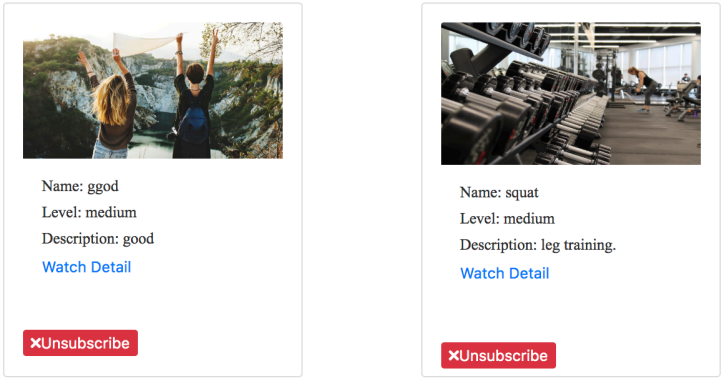
Basic information:

- **Name:** zz
- **Height:** 156.0
- **Weight:** 46.0
- **Age:** 12
- **Gender:** Female

---

9. User Role: after subscribing the course, can see all his own courses and can unsubscribe the course on this page.

My course:



Summary of the technologies used:

IDE：NetBeans

Structure: Spring MVC

Mapping: Hibernate (XML)

Stylesheet: Bootstrap

Language: Java Html CSS

APPENDIX

Controller:

```
public class CancelController extends AbstractController {

	UserDAO userDAO;
	CourseDAO courseDAO;

	@Override
	protected ModelAndView handleRequestInternal(HttpServletRequest hsr, HttpServletResponse hsr1) throws Exception {
		ModelAndView mv = null;
		String id =hsr.getParameter("courseID");

		long courseid =Long.valueOf(id);
	  courseDAO=(CourseDAO)getApplicationContext().getBean("coursedao");
	Course course=courseDAO.getCourseDetail(courseid);

	HttpSession session=hsr.getSession();
	User user=(User)session.getAttribute("User");

	userDAO=(UserDAO)getApplicationContext().getBean("userdao");

	  long userid=user.getUserID();
	    userDAO=(UserDAO)getApplicationContext().getBean("userdao");
	      userDAO.cancelSubscribe(user,course);

		List<Course> myCourse = userDAO.getMyCourse(userid);

	hsr.setAttribute("myCourse", myCourse);
	  mv= new ModelAndView("userCourse","myCourse",myCourse);

	  return mv;
	}
}

public class CoachMyCourseController extends AbstractController{

	CoachDAO coachDAO;
```

```
    protected ModelAndView handleRequestInternal(HttpServletRequest hsr, HttpServletResponse
hsr1) throws Exception {
          ModelAndView mv=null;
          System.out.println("hello");
          HttpSession session=hsr.getSession();
            coachDAO=(CoachDAO)getApplicationContext().getBean("coachdao");
         Coach coach=(Coach)session.getAttribute("Coach");
         System.out.println(coach);
         long coachid=coach.getCoachID();
         System.out.println(coachid);

//            String id=hsr.getParameter("coachID");
//            if(coachid!=null){
//              long coachID=Long.valueOf(id);
          List<Course> myCourse = coachDAO.getMyCourse(coachid);

               System.out.println(myCourse);
            hsr.setAttribute("myCourse", myCourse);
            mv=new ModelAndView("coachCourse","myCourse",myCourse);
//
//            }else{
//                  throw new Exception("id is null");
//            }

              return mv;
    }
}
public class CoachSeeUserController extends AbstractController{

     UserDAO userDAO;

    @Override
    protected ModelAndView handleRequestInternal(HttpServletRequest hsr, HttpServletResponse
hsr1) throws Exception {
            ModelAndView mv = null;
          String id =hsr.getParameter("userID");

          long userID =Long.valueOf(id);

               userDAO=(UserDAO)getApplicationContext().getBean("userdao");
```

```
            User user = userDAO.getUser(userID);
               List<Course> myCourse = userDAO.getMyCourse(userID);

                  hsr.setAttribute("User",user);
                  hsr.setAttribute("myCourse", myCourse);
            mv= new ModelAndView("CoachSeeUser","myCourse",myCourse);

            return mv;
      }


}public class CourseController extends SimpleFormController{

      CourseDAO courseDAO;
      CoachDAO coachDAO;

      public CourseController() {
            setCommandClass(Course.class);
            setCommandName("course");
            setSuccessView("coachCourse");
            setFormView("addCourse");
      }

      @Override
      protected ModelAndView onSubmit(HttpServletRequest request, HttpServletResponse response,
Object command, BindException errors) throws Exception {
            HttpSession session=request.getSession();
            ModelAndView mv;


            courseDAO =(CourseDAO)getApplicationContext().getBean("coursedao");
            coachDAO =(CoachDAO)getApplicationContext().getBean("coachdao");
            Course course =(Course) command;

           Coach coach=(Coach)session.getAttribute("Coach");
//            assign id
          long coachid=coach.getCoachID();
           course.setCoachID(coachid);
//            assign correct coverurl
            String url=course.getCoverurl();
            String correct="./css/images/"+url;
```

```
            System.out.println(correct);
            course.setCoverurl(correct);

            String videourl =course.getUrl();

            String correctvideo =videourl.replace("watch?v=", "embed/");
              System.out.println(correctvideo);
              course.setUrl(correctvideo);

            int result =courseDAO.add(course);


            if(result==1){
                  System.out.println("save success");
                  Course cc= new Course(course.getCoachID(),course.getUrl(), course.getName(),
course.getDescription(), course.getLevel(),course.getCoverurl());

                    List<Course> myCourse = coachDAO.getMyCourse(coachid);
                    request.setAttribute("myCourse", myCourse);
                      session.setAttribute("myCourse", myCourse);
                   mv= new ModelAndView(getSuccessView(),"myCourse", myCourse);

            }else{
                  System.out.println("save fail");
                  mv= new ModelAndView(new RedirectView("course.htm"));

      }
            return mv;

      }


}public class DetailController extends AbstractController{

        UserDAO userDAO;
      CourseDAO courseDAO;

      @Override
      protected ModelAndView handleRequestInternal(HttpServletRequest hsr, HttpServletResponse
hsr1) throws Exception {
            ModelAndView mv = null;
```

```
            String id =hsr.getParameter("courseID");
            String role=hsr.getParameter("role");

//              find the real courseid of this click --courseid
            long courseid =Long.valueOf(id);
            courseDAO=(CourseDAO)getApplicationContext().getBean("coursedao");
            Course course=courseDAO.getCourseDetail(courseid);
            hsr.setAttribute("course", course);

//          get the detail
            if(role.equals("user")){
            mv = new ModelAndView("userCourseDetail","course",course);
            }else if(role.equals("coach")){
                mv = new ModelAndView("coachCourseDetail","course",course);
            }


            return mv;
        }


}public class LoginController extends SimpleFormController{


         PersonDAO personDAO;
             public LoginController(){

            setCommandClass(Person.class);
              setCommandName("login");
//              setSuccessView("login-success");
              setFormView("login");
}



        @Override
        protected ModelAndView onSubmit(HttpServletRequest request, HttpServletResponse response,
Object command, org.springframework.validation.BindException errors) throws Exception {
              HttpSession session = request.getSession();

              ModelAndView mv;
```

```
            personDAO = (PersonDAO) getApplicationContext().getBean("persondao");
            Person p = (Person) command;
//              String type=p.getPersonType();

//          if(type.equals("User")){
            User loginu=personDAO.authenticateLogin(p.getUserName(), p.getPassword());
            if (loginu== null) {
                    mv = new ModelAndView(new RedirectView("login.htm"));
//                      return mv;
            }else{
                    session.setAttribute("User", loginu);
                            setSuccessView("UserPage");
                              mv = new ModelAndView(getSuccessView(),"User",loginu);
                              return mv;
            }
//      }else if(type.equals("Coach")){
                   Coach loginc=personDAO.authenticateLogin2(p.getUserName(), p.getPassword());
                    if (loginc== null) {
                    mv = new ModelAndView(new RedirectView("login.htm"));
//                      return mv;
            }else{
                    session.setAttribute("Coach", loginc);
                     setSuccessView("CoachPage");
                     mv = new ModelAndView(getSuccessView(),"Coach",loginc);
                      return mv;
            }
//      }

//              mv = new ModelAndView(getSuccessView(),"Person",p);

          return mv;

}

}
public class MoreVideosController extends AbstractController{

        UserDAO userDAO;

        @Override
```

```
    protected ModelAndView handleRequestInternal(HttpServletRequest hsr, HttpServletResponse
hsr1) throws Exception {

            ModelAndView mv=null;
            userDAO =(UserDAO)getApplicationContext().getBean("userdao");

             List<Course> allCourse = userDAO.getAllCourse();

             String role = hsr.getParameter("role");
              hsr.setAttribute("allCourse", allCourse);
             if(role.equals("user")){
                    mv=new ModelAndView("userMoreVideos","allCourse",allCourse);
             }else if(role.equals("coach")){
                    mv=new ModelAndView("coachMoreVideos","allCourse",allCourse);
             }
            return mv;
    }



}public class Profile2Controller extends SimpleFormController{

    CoachDAO coachDAO;

    public Profile2Controller() {

        setCommandClass(Coach.class);
        setCommandName("coachprofile");
           setSuccessView("coachProfile");
        setFormView("edit2Profile");
    }

    @Override
    protected ModelAndView onSubmit(HttpServletRequest request, HttpServletResponse response,
Object command, BindException errors) throws Exception {
          HttpSession session = request.getSession();
        ModelAndView mv = null;

       coachDAO=(CoachDAO) getApplicationContext().getBean("coachdao");
       Coach u=(Coach) command;
   int                                                                                    update
```

```
=coachDAO.updateCoach(u.getCoachID(),u.getHeight(),u.getWeight(),u.getAge(),u.getGender());

            if(update==0){
                    mv = new ModelAndView(new RedirectView("profile2.htm"));
            }else{
                  session.setAttribute("Coach", u);
                    mv = new ModelAndView(getSuccessView(),"Coach",u);

            }

          return mv;
      }


}public class RegisterController extends    SimpleFormController{

      PersonDAO personDAO;

            public RegisterController(){
            setCommandClass(Person.class);
            setCommandName("register");
    //          setSuccessView("login");
            setFormView("register");

}



      @Override
      protected ModelAndView onSubmit(HttpServletRequest request, HttpServletResponse response,
Object command, org.springframework.validation.BindException errors) throws Exception{
            HttpSession session = request.getSession();
            ModelAndView mv;

            personDAO = (PersonDAO) getApplicationContext().getBean("persondao");
            Person p = (Person) command;
        int duplication=personDAO.duplicateVerify(p.getUserName());


          if(duplication==1){
                //setSuccessView("login");
```

```
            session.setAttribute("message", "This account name has been used, choose a new one");
        //现在会抛出很丑陋的异常。
            mv = new ModelAndView(new RedirectView("register.htm"));
            return mv;
    }else{
                int result= personDAO.add(p);

        if (result== 1) {
                        String role=p.getPersonType();
                    if(role.equals("User")){
                        User u = new User(p.getUserName(),p.getPassword(),p.getPersonType());
                        personDAO.addUser(u);
                      session.setAttribute("User", u);
                      setSuccessView("UserPage");
                        mv = new ModelAndView(getSuccessView(),"User",u);
                        return mv;
                    }else if(role.equals("Coach")){
                        Coach c= new Coach(p.getUserName(),p.getPassword(),p.getPersonType());
                        personDAO.addCoach(c);
                         session.setAttribute("Coach", c);
                      setSuccessView("CoachPage");
                      mv = new ModelAndView(getSuccessView(),"Coach",c);
                       return mv;
                    }
                        mv = new ModelAndView(getSuccessView(),"Person",p);
                } else {
                        mv = new ModelAndView(new RedirectView("register.htm"));
                }
        }

return mv;

}
    }
        /**
 *
 * @author zhouzhou
 */
public class SubscribeController extends AbstractController{
    UserDAO userDAO;
```

```
        CourseDAO courseDAO;
        long courseid;

        @Override
        protected ModelAndView handleRequestInternal(HttpServletRequest hsr, HttpServletResponse
hsr1) throws Exception {
            ModelAndView mv = null;
            String id =hsr.getParameter("courseID");
              System.out.println(id);
                if(id!=null && id.length()!=0){
                  //从 subscirbe 进来
              courseid =Long.valueOf(id);
              courseDAO=(CourseDAO)getApplicationContext().getBean("coursedao");
            Course course=courseDAO.getCourseDetail(courseid);
                    System.out.println("find the course by id:"+course);
            HttpSession session=hsr.getSession();
            User user=(User)session.getAttribute("User");


           userDAO=(UserDAO)getApplicationContext().getBean("userdao");
          //set 不代表添加 到数据库了 ， 需要在 userDAO 里再更新一次

              long userid=user.getUserID();
                userDAO=(UserDAO)getApplicationContext().getBean("userdao");
            userDAO.updateSubscribe(user,course);


                    //subscirbe 成功
                      List<Course> myCourse = userDAO.getMyCourse(userid);
//                    System.out.println("users's course:"+ myCourse);

            hsr.setAttribute("myCourse", myCourse);
              mv= new ModelAndView("userCourse","myCourse",myCourse);

                }else{

                    //从导航栏进来   看 user my course
                      HttpSession session=hsr.getSession();
                      User user=(User)session.getAttribute("User");
                      long userid=user.getUserID();
                      userDAO=(UserDAO)getApplicationContext().getBean("userdao");
```

```
        List<Course> myCourse = userDAO.getMyCourse(userid);
                System.out.println("user course is "+ myCourse);
             hsr.setAttribute("myCourse", myCourse);
          mv= new ModelAndView("userCourse","myCourse",myCourse);
            }



        return mv;
    }


}public class UserSeeCoachController extends AbstractController{

    CoachDAO coachDAO;

    @Override
    protected ModelAndView handleRequestInternal(HttpServletRequest hsr, HttpServletResponse
hsr1) throws Exception {

          ModelAndView mv = null;
          String id =hsr.getParameter("coachID");

          long coachID =Long.valueOf(id);
          coachDAO=(CoachDAO)getApplicationContext().getBean("coachdao");

          Coach coach=coachDAO.getCoach(coachID);
            List<Course> myCourse = coachDAO.getMyCourse(coachID);

              hsr.setAttribute("Coach",coach);
                hsr.setAttribute("myCourse", myCourse);
        mv= new ModelAndView("UserSeeCoach","myCourse",myCourse);

        return mv;
    }


}public class deleteController extends AbstractController{

       CoachDAO coachDAO;
```

```
        CourseDAO courseDAO;

    @Override
    protected ModelAndView handleRequestInternal(HttpServletRequest hsr, HttpServletResponse hsr1) throws Exception {
            ModelAndView mv=null;

//              get coach id
              HttpSession session=hsr.getSession();
              coachDAO=(CoachDAO)getApplicationContext().getBean("coachdao");
              Coach coach=(Coach)session.getAttribute("Coach");
              long coachid=coach.getCoachID();


              //delete the course

                String id =hsr.getParameter("courseID");
                 long courseid =Long.valueOf(id);
                  courseDAO=(CourseDAO)getApplicationContext().getBean("coursedao");
                    Course course=courseDAO.getCourseDetail(courseid);
                 long createrID =course.getCoachID();
                 //check the ability first
                 if(createrID!=coachid){
                          JOptionPane.showMessageDialog(null, "You don't have permission to delete!");
                          List<Course> myCourse = coachDAO.getMyCourse(coachid);
                          hsr.setAttribute("myCourse", myCourse);
                          mv= new ModelAndView("coachCourse","myCourse", myCourse);
                 }else{
                         int result= coachDAO.deleteCourse(courseid);
                         if (result==1){
                               List<Course> myCourse = coachDAO.getMyCourse(coachid);
                               hsr.setAttribute("myCourse", myCourse);
                               mv= new ModelAndView("coachCourse","myCourse", myCourse);
                         }else{
                               JOptionPane.showMessageDialog(null, "Error happens when delete!");
                               List<Course> myCourse = coachDAO.getMyCourse(coachid);
                               hsr.setAttribute("myCourse", myCourse);
                               mv= new ModelAndView("coachCourse","myCourse", myCourse);
                         }
                 }
```

```
            return mv;
    }


}public class searchController extends AbstractController{

    CourseDAO courseDAO;

    @Override
    protected ModelAndView handleRequestInternal(HttpServletRequest hsr, HttpServletResponse
hsr1) throws Exception {

        ModelAndView mv=null;

        courseDAO= (CourseDAO)getApplicationContext().getBean("coursedao");

          String level=hsr.getParameter("level");
          String search =hsr.getParameter("search");
          if(level!=null && level.length()!=0){
          List<Course> searchLevel =courseDAO.searchLevel(level);

          hsr.setAttribute("searchCourse", searchLevel);
        mv=new ModelAndView("searchResult","searchCourse",searchLevel);
          }else{
         List<Course> searchCourse =courseDAO.searchCourse(search);
        hsr.setAttribute("searchCourse", searchCourse);
        mv=new ModelAndView("searchResult","searchCourse",searchCourse);
          }
        return mv;
    }



}
```

DAO:

```
public class CoachDAO extends DAO{

    public CoachDAO() {
    }
```

```
        public int updateCoach(long coachID, Double height, Double weight,int age,String gender)throws CoachException{
            int result;
            try{
                super.begin();

                String hql="update Coach u set u.height=:height,u.weight=:weight,u.age=:age,u.gender=:gender where u.coachID=:coachID";
                Query query =getSession().createQuery(hql);
                query.setLong("coachID", coachID);

                query.setDouble("height", height);
                query.setDouble("weight", weight);
                query.setInteger("age", age);
                query.setString("gender", gender);
//                query.setMaxResults(1);
                result=query.executeUpdate();
                super.commit();

    }catch(Exception e){
        super.rollback();
        throw new CoachException("Could not update coach",e);
    }
      return result;
}

        public Course addCourse(Course course) throws CoachException{

            try{
                super.begin();
                getSession().save(course);
               super.commit();
            }catch(Exception e){
                super.rollback();
                throw new CoachException("coach add course fail",e);
        }
            return course;
}

        public Coach getCoach(long coachID)throws CoachException{
            Coach c;
```

```
            try{
                super.begin();
                String hql="from Coach where coachID=:coachID";
                Query query = getSession().createQuery(hql);
                query.setLong("coachID", coachID);
                query.setMaxResults(1);
                c=(Coach)query.uniqueResult();
                super.commit();
                return c;
            }catch(Exception e){
                 super.rollback();
                throw new CoachException("cannot find this coach",e);
            }

        }

//          get my course
        public List<Course> getMyCourse(long coachID)throws CoachException{
            List list=null;
            try{
                super.begin();
                String hql ="from Course where coachID=:coachID";
                Query query=getSession().createQuery(hql);
                query.setLong("coachID", coachID);
                list=query.list();
                super.commit();
            }catch(Exception e){
                super.rollback();
                throw new CoachException("get my course fail",e);
        }
            return list;

}

//          get all course
            public List getAllCourse()throws CoachException{
            List list;
            try{
                super.begin();
                String hql ="from Course";
                Query query=getSession().createQuery(hql);
```

```
                    list=query.list();
                    super.commit();
                }catch(Exception e){
                    super.rollback();
                    throw new CoachException("get all course fail",e);

            }
                return list;

}

              //delete the course
            public int deleteCourse(long courseID)throws CoachException{
               int result;
              try{
                  super.begin();

                String hql="DELETE FROM Course WHERE courseID =:courseID";
                Query query =getSession().createQuery(hql);

                //check the reference..
                query.setLong("courseID", courseID);
                result=query.executeUpdate();
                super.commit();

      }catch(Exception e){
            super.rollback();
            throw new CoachException("Could not delete course",e);
      }
         return result;
}
}public class CourseDAO extends DAO{

      public CourseDAO() {
      }

//       add new course to database
      public int add(Course p) throws CourseException{
             int result=0;
            try{
                super.begin();
```

```
                getSession().save(p);
                super.commit();
                result=1;
        }catch(Exception e){
            super.rollback();
            throw new CourseException("Add course fail",e);
        }
            return result;
       }

//       get detail through the course id
        public Course getCourseDetail(long courseID) throws CourseException{
            Course course;
            try{
                super.begin();
                String hql="from Course where courseID=:courseID";
                Query query = getSession().createQuery(hql);
                query.setLong("courseID", courseID);
                course=(Course)query.uniqueResult();
                super.commit();
            }catch(Exception e){
                        super.rollback();
                        throw new CourseException("get course detail fail",e);
                 }
            return course;
        }

        //return the course that match the search content
        public List<Course> searchCourse(String search) throws CourseException{
              List list=null;
                  try{
                        super.begin();
                        Criteria crit =getSession().createCriteria(Course.class);
                        Criterion                nameMatch                =Restrictions.ilike("name",
search,MatchMode.ANYWHERE);
                        Criterion          descriptionMatch          =Restrictions.ilike("description",
search,MatchMode.ANYWHERE);
                        Criterion                levelMatch                =Restrictions.ilike("level",
search,MatchMode.ANYWHERE);
                        Disjunction disjunction =Restrictions.disjunction();
                        disjunction.add(nameMatch);
```

```
                    disjunction.add(descriptionMatch);
                    disjunction.add(levelMatch);
                    crit.add(disjunction);
                    list =crit.list();
                    super.commit();
                }catch(Exception e){
                    super.rollback();
                    throw new CourseException("get the    search course fail",e);
            }
                return list;


    }

    //search by level
      public List<Course> searchLevel(String level) throws CourseException{
        List list=null;
                try{
                    super.begin();
                    Criteria crit =getSession().createCriteria(Course.class);
                    Criterion levelMatch =Restrictions.ilike("level", level,MatchMode.EXACT);
                    crit.add(levelMatch);
                    list =crit.list();
                    super.commit();
                }catch(Exception e){
                    super.rollback();
                    throw new CourseException("get the search level course fail",e);
            }
                return list;

       }
}
public class PersonDAO extends DAO {

    public PersonDAO() {
    }

    public int add(Person p)throws PersonException{
        int result=0;
          try{
              super.begin();
```

```
                getSession().save(p);

                super.commit();
                result=1;
        }catch(Exception e){
            super.rollback();
            throw new PersonException("Could not get person"+p.getUserName(),e);
        }
            return result;
}

     public User addUser(User u)throws PersonException{

          try{
                super.begin();
                getSession().save(u);

                super.commit();

        }catch(Exception e){
            super.rollback();
            throw new PersonException("Could not get user"+u.getUserName(),e);
        }
            return u;
}

     public Coach addCoach(Coach c)throws PersonException{

          try{
                super.begin();
                getSession().save(c);

                super.commit();

        }catch(Exception e){
            super.rollback();
            throw new PersonException("Could not get coach"+c.getUserName(),e);
        }
            return c;
}
```

```
    public User authenticateLogin(String userName, String password) throws    PersonException {
          User p;
        try {
            super.begin();
            //hql 的 from 后面是对象 Person，而不是数据库的表名  person
            String hql="from User where userName=:userName AND password=:password";
            Query query =getSession().createQuery(hql);
            query.setString("userName", userName);
            query.setString("password", password);
            query.setMaxResults(1);
            p=(User)query.uniqueResult();
            super.commit();
            return p;
        } catch (Exception e) {
               super.rollback();
            throw new PersonException("Cant match the username and password",e);
        }
    }

          public Coach authenticateLogin2(String userName, String password) throws
PersonException {
              Coach p;
        try {
            super.begin();
            //hql 的 from 后面是对象 Person，而不是数据库的表名  person
            String hql="from Coach where userName=:userName AND password=:password";
            Query query =getSession().createQuery(hql);
            query.setString("userName", userName);
            query.setString("password", password);
            query.setMaxResults(1);
            p=(Coach)query.uniqueResult();
            super.commit();
            return p;
        } catch (Exception e) {
               super.rollback();
            throw new PersonException("Cant match the username and password",e);
        }
    }

    public int duplicateVerify(String userName)throws PersonException{
```

```
            int duplication=0;
            try{
                    super.begin();
                     String hql="from Person where userName=:userName";
                       Query query =getSession().createQuery(hql);
                         query.setString("userName", userName);
                         query.setMaxResults(1);
                          List l=query.list();
//                          verify the duplication,if yes,duplication changes to 1
                         if(l==null    || l.isEmpty()){
                                duplication=0;
                         }else{
                                duplication=1;
                         }
                          super.commit();
            }catch (Exception e) {
                  super.rollback();
              throw new PersonException("This username has been used.",e);
             }

            return duplication;
        }


      }public class UserDAO    extends DAO{

      public UserDAO() {
      }

          public int updateUser(long userID, Double height, Double weight,int age,String gender)throws
UserException{
              int result;
            try{
                 super.begin();

                String                hql="update                User                u                set
u.height=:height,u.weight=:weight,u.age=:age,u.gender=:gender where u.userID=:userID";
                Query query =getSession().createQuery(hql);
                query.setLong("userID", userID);

                query.setDouble("height", height);
```

```
                query.setDouble("weight", weight);
                query.setInteger("age", age);
                query.setString("gender", gender);
//                query.setMaxResults(1);
                result=query.executeUpdate();
                super.commit();

        }catch(Exception e){
            super.rollback();
            throw new UserException("Could not update user",e);
        }
           return result;
}

        //              get all course
            public List getAllCourse()throws UserException{
                List list;
                try{
                    super.begin();
                    String hql ="from Course";
                    Query query=getSession().createQuery(hql);
                    list=query.list();
                    super.commit();
                }catch(Exception e){
                    super.rollback();
                    throw new UserException("get all course fail",e);

            }
                return list;

}
              //get user object by id
                public User getUser(long userID)throws UserException{
                    User c;
                    try{
                        super.begin();
                        String hql="from User where userID=:userID";
                        Query query = getSession().createQuery(hql);
                        query.setLong("userID", userID);
                        query.setMaxResults(1);
                        c=(User)query.uniqueResult();
```

```
                super.commit();
                return c;
            }catch(Exception e){
                  super.rollback();
                throw new UserException("cannot find this user",e);
            }

        }


       //              get my course
        public List<Course> getMyCourse(long userID)throws UserException{
            List<Course> list=null;
            try{
                super.begin();
                  Criteria crit =getSession().createCriteria(Course.class);
                  Criteria userCrit =crit.createCriteria("users");
                  userCrit.add(Restrictions.eq("userID", userID));
                  list=crit.list();
                super.commit();
            }catch(Exception e){
                super.rollback();
                throw new UserException("get my course fail",e);
        }
            return list;

}


//              user subscribe the course
    public void updateSubscribe(User user, Course course)throws UserException{
        try{
            super.begin();
            //add course for user
            Set<Course> userCourses=user.getCourses();
            userCourses.add(course);
            user.setCourses(userCourses);
               System.out.println("Get course:"+user.getCourses());
            //add user for course
            Set<User> courseUsers =course.getUsers();
            courseUsers.add(user);
```

```
                course.setUsers(courseUsers);
                System.out.println("Get user:"+course.getUsers());
                getSession().merge(user);
                getSession().merge(course);
                getSession().save(user);
                getSession().save(course);
                super.commit();
        }catch(Exception e){
            super.rollback();
            throw new UserException("Could not subscribe",e);
        }
                }
          //                user cancel subscribe the course
            public void cancelSubscribe (User user, Course course)throws UserException{
                  try{
                          super.begin();
//                          user
                          Set<Course> userCourses=user.getCourses();
                          userCourses.remove(course);
                          user.setCourses(userCourses);
//                             course
                          Set<User> courseUsers =course.getUsers();
                          courseUsers.remove(user);
                          course.setUsers(courseUsers);

                getSession().merge(user);
                getSession().merge(course);
                getSession().save(user);
                getSession().save(course);
                super.commit();

                  }catch(Exception e){
            super.rollback();
            throw new UserException("Could not cancel subscribe",e);
        }
            }
}
```

POJO:

```
public class Coach extends Person{
```

```
private long coachID;
private double height;
private double weight;
private int age;
private String gender;
private Set<Course> course;

public Coach() {
}


public Coach(String userName, String password, String type) {
	super(userName, password, type);
}

public long getCoachID() {
	return coachID;
}

public void setCoachID(long coachID) {
	this.coachID = coachID;
}

public double getHeight() {
	return height;
}

public void setHeight(double height) {
	this.height = height;
}

public double getWeight() {
	return weight;
}

public void setWeight(double weight) {
	this.weight = weight;
}
```

```
    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public String getGender() {
        return gender;
    }

    public void setGender(String gender) {
        this.gender = gender;
    }

    public Set<Course> getCourse() {
        return course;
    }

    public void setCourse(Set<Course> course) {
        this.course = course;
    }



    @Override
    public String toString() {
        return "Coach{" + "coachID=" + coachID + '}';
    }

}public class Course {
    private long courseID;
    private long coachID;
    private String url;
    private String name;
    private String description;
    private String    level;
    private String coverurl;
    private Set<User> users =new HashSet();
```

```
    public Course() {
    }


    public Course(long coachID,String url, String name, String description, String level, String coverurl) {
        this.coachID=coachID;
        this.url = url;
        this.name = name;
        this.description = description;
        this.level = level;
    }



    public long getCourseID() {
        return courseID;
    }

    public void setCourseID(long courseID) {
        this.courseID = courseID;
    }


    public long getCoachID() {
        return coachID;
    }

    public void setCoachID(long coachID) {
        this.coachID = coachID;
    }


    public String getUrl() {
        return url;
    }

    public void setUrl(String url) {
        this.url = url;
    }
```

```
public String getName() {
	return name;
}

public void setName(String name) {
	this.name = name;
}

public String getDescription() {
	return description;
}

public void setDescription(String description) {
	this.description = description;
}

public String getLevel() {
	return level;
}

public void setLevel(String level) {
	this.level = level;
}

public Set<User> getUsers() {
	return users;
}

public void setUsers(Set<User> users) {
	this.users = users;
}


public String getCoverurl() {
	return coverurl;
}

public void setCoverurl(String coverurl) {
	this.coverurl = coverurl;
}
```

```
@Override
public String toString() {
       return "Course{" + "courseID=" + courseID + ", name=" + name + '}';
}
 }

 public class Person {

          private long personID;
          private String personType;
        private String userName;
        private String password;

        public Person() {
        }


           public Person(String userName,String password,String type) {
              this.userName=userName;
              this.password=password;
              this.personType=type;
        }

             public Person(String userName,String type){
              this.userName=userName;
              this.personType=type;
        }



        public long getPersonID() {
              return personID;
        }

        public void setPersonID(long personID) {
              this.personID = personID;
        }

        public String getPersonType() {
              return personType;
```

```
        }

        public void setPersonType(String personType) {
            this.personType = personType;
        }


        public String getUserName() {
            return userName;
        }

        public void setUserName(String userName) {
            this.userName = userName;
        }

        public String getPassword() {
            return password;
        }

        public void setPassword(String password) {
            this.password = password;
        }
}
public class User extends Person{

      private long userID;
        private double height;
        private double weight;
        private int age;
        private String gender;
        private Set<Course> courses =new HashSet();

        public User() {
        }

        public User(String userName, String password, String type) {
            super(userName, password, type);
        }


        public long getUserID() {
```

```
        return userID;
}

public void setUserID(long userID) {
        this.userID = userID;
}

public double getHeight() {
        return height;
}

public void setHeight(double height) {
        this.height = height;
}

public double getWeight() {
        return weight;
}

public void setWeight(double weight) {
        this.weight = weight;
}




public int getAge() {
        return age;
}

public void setAge(int age) {
        this.age = age;
}

public String getGender() {
        return gender;
}

public void setGender(String gender) {
        this.gender = gender;
}
```

```
    public Set<Course> getCourses() {
        return courses;
    }

    public void setCourses(Set<Course> courses) {
        this.courses = courses;
    }
}
```

Validator:

```
public class CourseValidator implements Validator{

    @Override
    public boolean supports(Class aClass) {
        return Course.class.isAssignableFrom(aClass);
    }

    @Override
    public void validate(Object o, Errors errors) {
        ValidationUtils.rejectIfEmptyOrWhitespace(errors, "url", "error.url.required", "url required");
        ValidationUtils.rejectIfEmptyOrWhitespace(errors, "name", "error.name.required", "name required");
        ValidationUtils.rejectIfEmptyOrWhitespace(errors, "description", "error.description.required", "description required");
        ValidationUtils.rejectIfEmptyOrWhitespace(errors, "coverurl", "error.coverurl.required", "picture required");

        if(errors.hasErrors())
            return;
    }
```

```
}
public class LoginValidator    implements Validator{

    @Override
    public boolean supports(Class aClass) {
        return Person.class.isAssignableFrom(aClass);
    }

    @Override
    public void validate(Object o, Errors errors) {
        ValidationUtils.rejectIfEmptyOrWhitespace(errors,                    "userName",
"error.userName.required", "userName required");
        ValidationUtils.rejectIfEmptyOrWhitespace(errors,                     "password",
"error.password.required", "password required");

          if(errors.hasErrors())
                return;
    }



}
public class RegisterValidator implements Validator{

    @Override
    public boolean supports(Class aClass) {
        return Person.class.isAssignableFrom(aClass);
    }

    @Override
    public void validate(Object o, Errors errors) {
        ValidationUtils.rejectIfEmptyOrWhitespace(errors,                    "userName",
"error.userName.required", "userName required");
        ValidationUtils.rejectIfEmptyOrWhitespace(errors,                     "password",
"error.password.required", "password required");

ValidationUtils.rejectIfEmptyOrWhitespace(errors,"personType","error.personType.required","please
select a role");
          if(errors.hasErrors())
                return;
    }
```

```
    }
```

View:

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Coach</title>
        <link rel="stylesheet" href="./css/bootstrap.min.css" type="text/css">
          <link rel="stylesheet" href="./css/bootstrap.css" type="text/css">
          <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.8.1/css/all.css"
                integrity="sha384-
50oBUHEmvpQ+1lW4y57PTFmhCaXp0ML5d60M1M7uH2+nqUivzIebhndOJK28anvf"
crossorigin="anonymous">
        <link rel="stylesheet" href="./css/form.css" type="text/css">
        </head>
        <body>
            <!--nav bar-->
            <nav class="navbar navbar-expand-lg navbar-light bg-light">
      <a class="navbar-brand">Casual Fitness</a>
      <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false"
aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
      </button>

      <div class="collapse navbar-collapse" id="navbarSupportedContent">
        <ul class="navbar-nav mr-auto">

          <!-- current page:coach's main page -->
          <li class="nav-item active">
            <a class="nav-link" href="CoachPage.jsp" ><i class="fas fa-home"></i> Home<span
class="sr-only">(current)</span></a>
          </li>

          <!-- profile page -->
          <li class="nav-item">
            <a class="nav-link" href="coachProfile.jsp"><i class="far fa-id-card"></i> Profile</a>
          </li>
```

```
            <!-- update my new video -->
           <li class="nav-item">
        <a class="nav-link" href="mycourse.htm" > <i class="fab fa-discourse"></i> My courses</a>
            </li>


                <!-- update my new video -->
                <li class="nav-item">
                    <a class="nav-link"    href="course.htm" > <i class="fas fa-cloud-upload-alt"></i>
Upload course video</a>

                </li>
          <!--show all information-->
              <div class="container">

                 <h1>User ${requestScope.User.getUserName()}'s Main page:</h1>

                <hr/>
                <h3>Favorite Course:</h3>
                      <div class="row align-items-start" >
                          <c:forEach var="result" items="${requestScope.myCourse}" >
                              <div class="card col-sm-offset-1" style="width:20rem;height:25rem">


                                    <div class="card-body">
                                        <img class="card-img-top" src="${result.getCoverurl()}"
alt="Card image" style="width:100%; height:50%; left:0px;top:125px;"
                                    </div>

                                      <div class="card-body"    >
                                          <h5      class="card-text"><b>Name:</b>      <c:out
value="${result.getName()}" /></h5>
                                     <h5       class="card-text"><b>Level:</b>       <c:out
value="${result.getLevel()}" /></h5>
                                     <h5    class="card-text"><b>Description:</b>   <c:out
value="${result.getDescription()}" /></h5>
                                     <a
href="detail.htm?courseID=${result.getCourseID()}&role=coach" class="card-link">Watch Detail</a>
                                  </div>
                              </div>
```

```
                        </div>
                        </c:forEach>

    </div>
            <h3>Basic information:</h3>
              <ul class="details-list">

                    <li>
                      <span class="detail-header">Name:</span>
                      <span  class="detail-value  demi">  ${requestScope.User.getUserName()}
</span>
                    </li>
                    <li>
                      <span class="detail-header">Height:</span>
                      <span    class="detail-value    demi">    ${requestScope.User.getHeight()}
</span>
                    </li>
                    <li>
                      <span    class="detail-header"><i    class="fas    fa-award"></i>    Weight:
</span>
                      <span    class="detail-value    demi">    ${requestScope.User.getWeight()}
</span>
                    </li>
                     <li>
                      <span class="detail-header"><i class="fas fa-award"></i> Age:    </span>
                      <span class="detail-value demi"> ${requestScope.User.getAge()} </span>
                    </li>
                    <li>
                      <span    class="detail-header"><i    class="fas    fa-award"></i>    Gender:
</span>
                      <span    class="detail-value    demi">    ${requestScope.User.getGender()}
</span>
                    </li>

          </div>

        </body>
    </html>

              <!-- view all videos -->
            <li class="nav-item">
```

```
                    <a class="nav-link" href="moreVideos.htm?role=coach" > <i class="fab fa-youtube"></i> More videos</a>
                </li>

            </ul>

            <form class="form-inline my-2 my-lg-0" action='login.htm'>
                Hello Coach ${sessionScope.Coach.getUserName()}  
                <button class="btn btn-outline-success my-2 my-sm-0" ><i class="fas fa-sign-out-alt" ></i>Logout</button>
            </form>

        </div>
    </nav>
        <!--nav bar end-->

            <div>
              <img src="./css/images/coachmain.jpg" alt="main picture"/>
            </div>

            <div class="joinus" >Find Your Fitness.<br/>
                    Something for Everyone.
                <br/>
                <a href="moreVideos.htm?role=coach">
                    <span class="menu-icon"><i class="icon -fitness" aria-hidden="true"></i></span>
                    <span class="menu-label">  <i class="fab fa-youtube"></i> More Videos</span></a></div>


        </body>
    </html>
    <%@page contentType="text/html" pageEncoding="UTF-8"%>
    <!DOCTYPE html>
    <html>
        <head>
            <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
            <title>User</title>
              <link rel="stylesheet" href="./css/bootstrap.min.css" type="text/css">
             <link rel="stylesheet" href="./css/bootstrap.css" type="text/css">
             <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.8.1/css/all.css"
```

```
                integrity="sha384-
50oBUHEmvpQ+1lW4y57PTFmhCaXp0ML5d60M1M7uH2+nqUivzIebhndOJK28anvf"
crossorigin="anonymous">
                <link rel="stylesheet" href="./css/form.css" type="text/css">

        </head>

        <body>
            <!--nav bar-->
            <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <a class="navbar-brand">Casual Fitness</a>

        <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false"
aria-label="Toggle navigation">
          <span class="navbar-toggler-icon"></span>
        </button>

        <div class="collapse navbar-collapse" id="navbarSupportedContent">
          <ul class="navbar-nav mr-auto">


            <li class="nav-item active">
              <a class="nav-link" href="UserPage.jsp"><i class="fas fa-home"></i>Home<span
class="sr-only">(current)</span></a>
            </li>

              <li class="nav-item">
                <a class="nav-link" href="userProfile.jsp"><i class="far fa-id-
card"></i>Profile</a>
            </li>


            <li class="nav-item">
                    <a class="nav-link" href="usermycourse.htm"> <i class="fab fa-
discourse"></i>My Courses</a>
            </li>


            <li class="nav-item">
              <a class="nav-link" href="moreVideos.htm?role=user"><i class="fab fa-
```

```
youtube"></i>More Videos</a>
                </li>
              </ul>
                <form class="form-inline my-2 my-lg-0" action='login.htm'>Hello User ${sessionScope.User.getUserName()}  
                <button class="btn btn-outline-success my-2 my-sm-0"><i class="fas fa-sign-out-alt" ></i>Logout</button>
            </form>
                </div>
        </nav>
                <!--nav bar end-->

                <div >
                    <img src="./css/images/usermain.jpg" alt="main picture"/>
                </div>

                <div class="joinus" >Find Your Fitness.<br/>
                            Something for Everyone.
                        <br/>
                        <a href="moreVideos.htm?role=user">
                            <span class="menu-icon"><i class="icon -fitness" aria-hidden="true"></i></span>
                            <span class="menu-label">  <i class="fab fa-youtube"></i> More Videos</span></a></div>

            </body>
        </html>


          <div id="container2">
            <!-- title -->
            <div class="page-header">
                    <h3 class="text-center">
                        Add Course
                    </h3>
            </div>

            <form:form commandName="course"    method="post" action="course.htm">


                Your coach    id is ${sessionScope.Coach.coachID}
```

```
<hr>


<!--URL-->
 <div class="form-group">
        <label class="col-sm-6 control-label">URL: </label>
         <form:input class="col-lg-6 form-control" path="url" type="text"
pattern="^(https|ftp|file):\/\/+[\w+&@#/%?~|!:,.;=]+"/>
         <font style="color:red"><form:errors path="url"/></font>
 </div>

 <!--Name-->
 <div class="form-group">
        <label class="col-sm-6 control-label">Name: </label>
         <form:input class="col-lg-6 form-control" path="name" type="text" />
         <font style="color:red"><form:errors path="name"/></font>
 </div>

  <!--Description-->
 <div class="form-group">
        <label class="col-sm-6 control-label">Description: </label>
         <form:input class="col-lg-6 form-control" path="description" type="text"
/>
         <font style="color:red"><form:errors path="description"/></font>
 </div>


 <!--Level-->
 <div class="form-group">
        <label class="col-sm-6 control-label">Difficulty Level: </label>
        <form:select path="level">
            <form:option value="easy" label="easy" />
            <form:option value="medium" label="medium" />
            <form:option value="hard" label="hard" />
        </form:select>
 </div>

<!--coverurl-->
   <div class="form-group">
        <label class="col-sm-6 control-label">Choose a picture for your course:
```

```
</label>
                        <form:input class="col-lg-6 form-control" path="coverurl" type="file" />
                        <font style="color:red"><form:errors path="coverurl"/></font>
                    </div>


                            <input type="submit" class="btn" value="Upload Course" />
                </form:form>



                <!--Coach-->



            </div>

        </body>
    </html>
    <!--show all information-->
        <div class="container">

            <h1>My course:</h1>

        <hr/>
            <div class="row align-items-start" >
                <c:forEach var="result" items="${requestScope.myCourse}" >
                    <div class="card col-sm-offset-1" style="width:20rem;height:25rem">


                        <div class="card-body">
                            <img class="card-img-top" src="${result.getCoverurl()}"
alt="Card image" style="width:100%; height:50%; left:0px;top:125px;"
                        </div>

                        <div class="card-body"    >
                            <h5      class="card-text"><b>Name:</b>      <c:out
value="${result.getName()}" /></h5>
                        <h5        class="card-text"><b>Level:</b>        <c:out
value="${result.getLevel()}" /></h5>
                        <h5    class="card-text"><b>Description:</b>   <c:out
```

```
value="${result.getDescription()}" /></h5>
                                                    <a
href="detail.htm?courseID=${result.getCourseID()}&role=coach" class="card-link">Watch Detail</a>
                                                </div>
                                            </div>

                                            <div class="card-body">

                                                <!-- deleteButton -->
                                                <a
href="delete.htm?courseID=${result.getCourseID()}" class=" btn-sm btn-danger">Delete</a>
                                            </div>
                                    </div>

                                    </c:forEach>

        </div>
                </div>

            </body>
        </html>

                    <!--show the details-->
                     <div class ="container">
        <div class="show-header">
            <div class="container -flush">
                <div class="media">
                    <div class="responsive-video">
                        <iframe width="700" height="394" src="${requestScope.course.getUrl()}"
frameborder="0" allowfullscreen="1"></iframe>
                    </div>
                </div>


                <div class="info has-actions">

                    <div class="video-details">
                        <h2 class="heading caps-half demi ">Workout Details</h2>

                            <ul class="details-list">
```

```
                        <li>
                            <span class="detail-header">Name:</span>
                            <span     class="detail-value     demi">${requestScope.course.getName()}
</span>
                        </li>
                        <li>
                            <span class="detail-header">Description:</span>
                            <span class="detail-value demi">${requestScope.course.getDescription()}
</span>
                        </li>
                        <li>
                            <span   class="detail-header"><i   class="fas   fa-award"></i>   Difficulty:
</span>
                            <span     class="detail-value     demi">${requestScope.course.getLevel()}
</span>
                        </li>

                    </ul>

                </div></div></div></div>
                    <hr>
                    <div>
        <div class="feed">
          <h2 class="heading -x-small caps-half demi details">Video Feed</h2>
        </div>

                    <h3>    The users who likes this video:</h3>

                    <div class="list-group">
                            <c:forEach var="result" items="${requestScope.course.getUsers()}" >
                                <a        href="coachseeuser.htm?userID=${result.getUserID()}"
class="list-group-item list-group-item-action" >
                                    ${result.getUserName()}
                                </a>
                            </c:forEach>
                    </div>

                <!--show all information-->
            <div class="container">

                    <h1>All courses:</h1>
```

```
                <hr/>
                        <div class="row align-items-start" >
                            <c:forEach var="result" items="${requestScope.allCourse}" >
                                <div class="card col-sm-offset-1" style="width:20rem;height:25rem">


                                        <div class="card-body">
                                            <img class="card-img-top" src="${result.getCoverurl()}"
alt="Card image" style="width:100%; height:50%; left:0px;top:125px;"
                                        </div>

                                            <div class="card-body"    >
                                                <h5      class="card-text"><b>Name:</b>      <c:out
value="${result.getName()}" /></h5>
                                                <h5       class="card-text"><b>Level:</b>       <c:out
value="${result.getLevel()}" /></h5>
                                                <h5    class="card-text"><b>Description:</b>   <c:out
value="${result.getDescription()}" /></h5>
                                                <a
href="detail.htm?courseID=${result.getCourseID()}&role=coach" class="card-link">Watch Detail</a>
                                        </div>
                                </div>

                                </div>
                                </c:forEach>

        </div>
                </div>
            </body>
        </html>
           <section id="profile" name="profile">

                <div class="container desc">

                        <div class="col-lg-2 col-lg-offset-1">
                          <h3>Profile</h3>
                        </div>
                    <br/>
                    <hr>
                    <form action="profile2.htm">
```

```
            <!-- userName -->
            <div class="col-lg-12 ">
              <p>
                <label> userName:</label>    ${Coach.getUserName()}
              </p>
            </div>
          <!-- height -->
            <div class="col-lg-12 ">
              <p>
                <label>    Height:    </label>    ${sessionScope.Coach.getHeight()}
              </p>
            </div>
              <!-- Weight -->
              <div class="col-lg-12 ">
                <p>
                            <label> Weight: </label>      ${sessionScope.Coach.getWeight()}
                </p>
              </div>
                <!-- Age -->
            <div class="col-lg-12 ">
              <p>
                <label>    Age:      </label>      ${sessionScope.Coach.getAge()}
              </p>
            </div>
                <!-- Gender -->
            <div class="col-lg-12 ">
              <p>
                <label>    Gender:      </label>    ${Coach.getGender()}
              </p>
            </div>


              <input type="submit" class="btn" value='Edit Profile'/>
          </form>
          <hr>
        </div>
        <!--/.container -->
    </section>
<!--form to save coach's information-->
              <section id="profile" name="profile">
```

```
<!-- title -->
<div class="page-header">
        <h3 class="text-center">
                Profile Form
            </h3>
</div>

<div class='container'>
<!-- row -->
<form:form commandName="userprofile"    method="post" action="profile.htm">

    <div class="form-row">
        <!--userID->
-->          <div class="form-group">
            <label for="userID" class="col-lg-6 control-label">User ID: </label>
             <form:input class="form-control col-lg-6" path="userID" type="text"
value=" ${sessionScope.User.getUserID()}" readonly="true"/>
        </div>

         <!--userName-->
        <div class="form-group">
            <label for="userName" class="col-lg-6 control-label">User Name: </label>
             <form:input class="form-control col-lg-6" path="userName" type="text"
value=" ${sessionScope.User.getUserName()}" readonly="true"/>
        </div>
    </div>

        <hr><br/>

       <div class="form-row">
         <!--Height-->
        <div class="form-group">
            <label    class="col-lg-6 control-label">Your Height(cm): </label>
                <form:input class="form-control col-lg-6" path="height" type="text"
value=" ${sessionScope.User.getHeight()}" pattern="$*[0-9]+([.]{1}[0-9]+){0,1}$|^$" />
        </div>
        </div>

         <div class="form-row">
         <!--Weight-->
        <div class="form-group">
```

```
                        <label    class="col-lg-6 control-label">Your Weight(kg): </label>
                         <form:input class="form-control col-lg-6" path="weight" type="text" value="${sessionScope.User.getWeight()}" pattern="$*[0-9]+([.]{1}[0-9]+){0,1}$|^$"/>
                    </div>
                     </div>

                     <div class="form-row">
                     <!--Age-->
                    <div class="form-group">
                        <label for=" Age" class="col-lg-6 control-label">Age: </label>
                         <form:input class="col-lg-6 form-control" path="age" type="text" value="${sessionScope.User.getAge()}" pattern="[0-9]+" />
                    </div>

                     </div>

                     <div class="form-row">
                     <!--Gender-->
                    <div class="form-group">
                        <label for=" Gender" class="col-lg-15 control-label">Gender: (${sessionScope.User.getGender()} now)</label>
                          <br/>
                        <label class="col-lg-7 control-label"> Male:</label>
                        <form:radiobutton class="form-control" path="gender" value="Male" readonly="true" name="gender" />


                          <label class="col-lg-7 control-label"> Female:</label>
                          <form:radiobutton class=" form-control" path="gender" value="Female" readonly="true" name="gender" />


                    </div>
                     </div>

                    <input type='submit' value='Save Profile' class='btn'/>
            </form:form>
            </div>   </section>

            </body>
        </html>
```

```
  <body>
        <div class="login-box">
            <h1>Login</h1>
         <form:form commandName="login"    method="post" action="login.htm">
            <!--username-->
            <div class="textbox">
                  <i class="fas fa-user"></i>
<!--              <label>username:</label>       -->
            <form:input path="userName" type="text" placeholder="Username"/>
            <form:errors path="userName"/>
            </div>
            <!--password-->
            <div class="textbox">
              <i class="fas fa-lock"></i>
<!--              <label>password:</label>              -->
            <form:input path="password" type="password" placeholder="Password"/>
            <form:errors path="password"/>
            </div>
            <input type="submit"    value="Login" class="btn"/>
         </form:form>
           <br/>
            <div><a href="register.htm">Click to Register</a></div>
        </div>
         </body>
</html>
<body>
         <div class="login-box">
            <h1>Register</h1>
         <form:form commandName="register"    method="post" action="register.htm">
              <div class="textbox">
                  <i class="fas fa-user"></i>
            <form:input path="userName" type="text" placeholder="New username"/>
            <form:errors path="userName" />
              </div>
               <div class="textbox">
            <i class="fas fa-lock"></i>
            <form:input path="password" type="password" placeholder="New password"/>
            <form:errors path="password" />
             </div>
            <br/>
```

```
                    <label> Role: </label>
                     <label>User</label>  <form:radiobutton path="personType" value="User"
name="personType" />
                      <label>Coach</label>  <form:radiobutton path="personType" value="Coach"
name="personType" />
                      <br/>
                      <span class="role"><form:errors path="personType" /></span>

                     <input type="submit"   value="Register" class="btn"/>
                </form:form>
                      <div><a href="login.htm">Click to Login</a></div>
                      <h3>${sessionScope.message}</h3>
                     </div>

          </body>
      </html>
        <div class="container">

                      <h1>Result:</h1>
        <form class="form-inline my-2 my-lg-0" action="search.htm" >
                <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-
label="Search" name="search">
                <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>
              </form><br/>
                       <a href="search.htm?level=easy" class=" btn-sm btn-danger">Easy Level</a>
                 <a href="search.htm?level=medium" class=" btn-sm btn-secondary">Medium
Level</a>
                   <a href="search.htm?level=hard" class=" btn-sm btn-warning">Hard Level</a>
                <hr/>
                      <div class="row align-items-start" >
                          <c:forEach var="result" items="${requestScope.searchCourse}" >
                              <div class="card col-sm-offset-1" style="width:20rem;height:25rem">


                                      <div class="card-body">
                                          <img class="card-img-top" src="${result.getCoverurl()}"
alt="Card image" style="width:100%; height:50%; left:0px;top:125px;"
                                      </div>

                                      <div class="card-body"   >
                                          <h5     class="card-text"><b>Name:</b>     <c:out
```

```
value="${result.getName()}" /></h5>
                                                <h5       class="card-text"><b>Level:</b>       <c:out
value="${result.getLevel()}" /></h5>
                                                <h5   class="card-text"><b>Description:</b>   <c:out
value="${result.getDescription()}" /></h5>
                                                <a
href="detail.htm?courseID=${result.getCourseID()}&role=user" class="card-link">Watch Detail</a>
                                            </div>
                                        </div>

                                        </div>
                                        </c:forEach>

        </div>
                </div>
            </body>
        </html>


        </div>
                    </div>


            </body>
        </html>
```