**INFO6250** **Web Development Tools & Methds SEC 06**

**——Mini Slack**

1. **Summary**

In the past few days, I developed a software application by using SpringMVC, Hibernate, Annotation, JavaScript, CSS. There are two roles, teachers and students that can register and activate new user, retrieve password by using e-mail, update basic information and password, manage and send messages with each other and upload files to whoever you want.

1. **Key Functionalities**

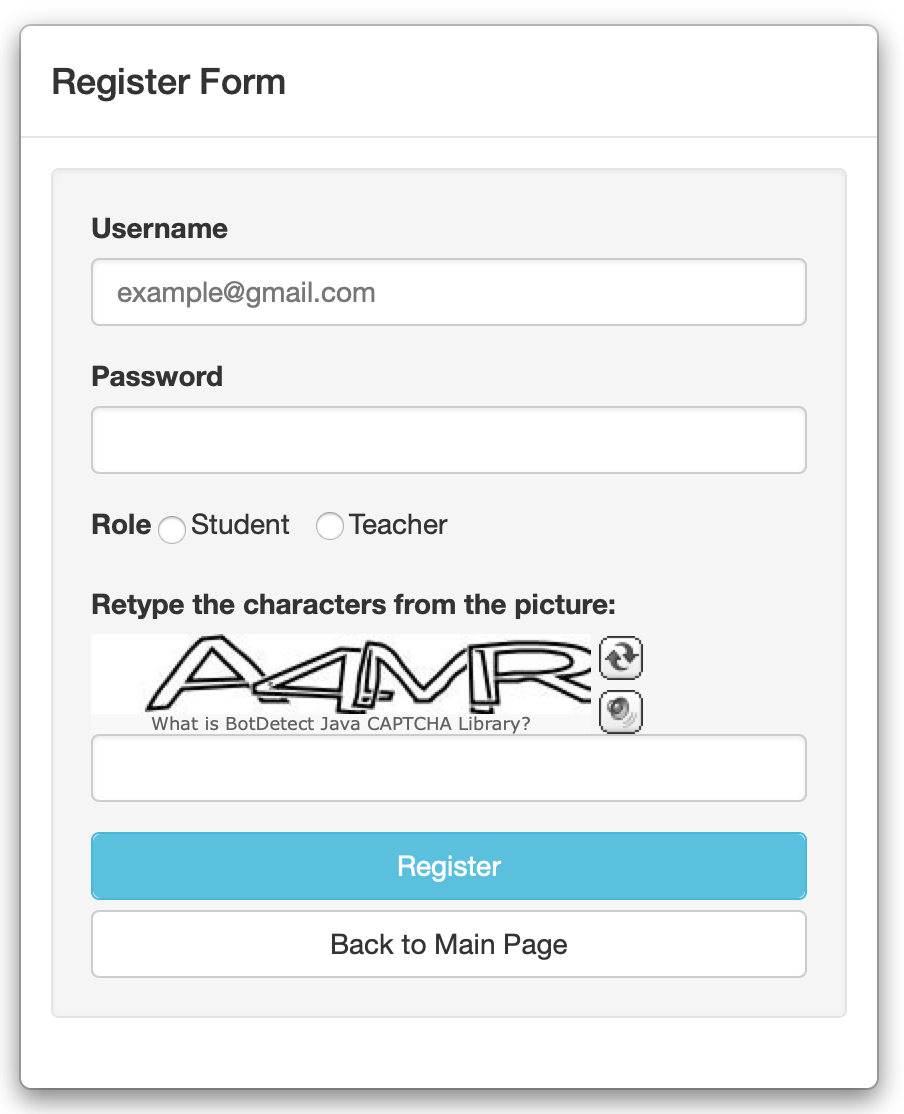
* Register new user by E-mail address and activate it using links attached with e-mail
* Update basic information based on different roles after login
* Show updated information just
* Update password
* Retrieve password by using registered e-mail
* Send messages and file(less than 5M) to specified user with different topics
* Delete history record

1. **Key Technologies**

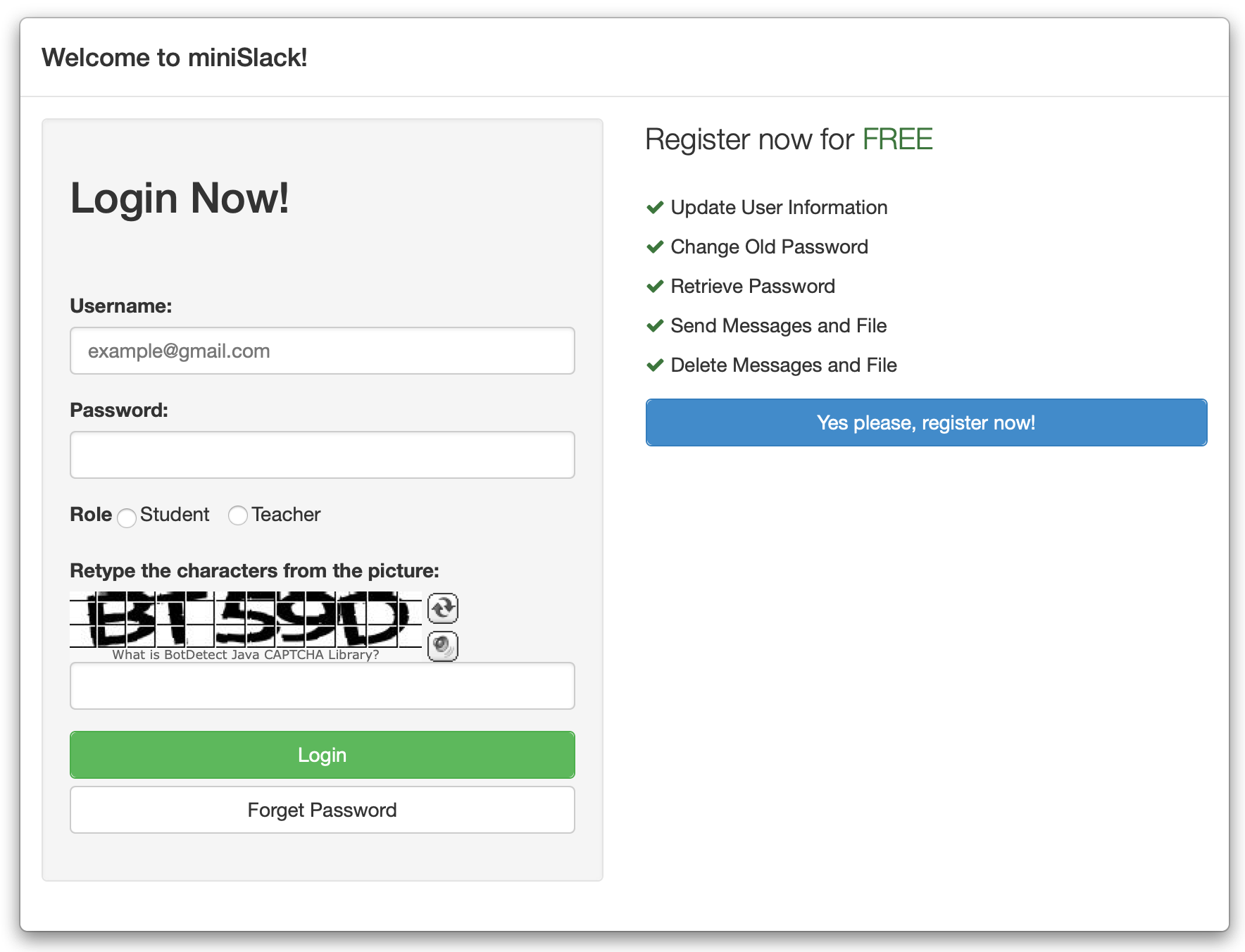
* SpringMVC
* Hibernate
* Annotation Mapping
* JavaScript
* CSS

1. **Screenshots**

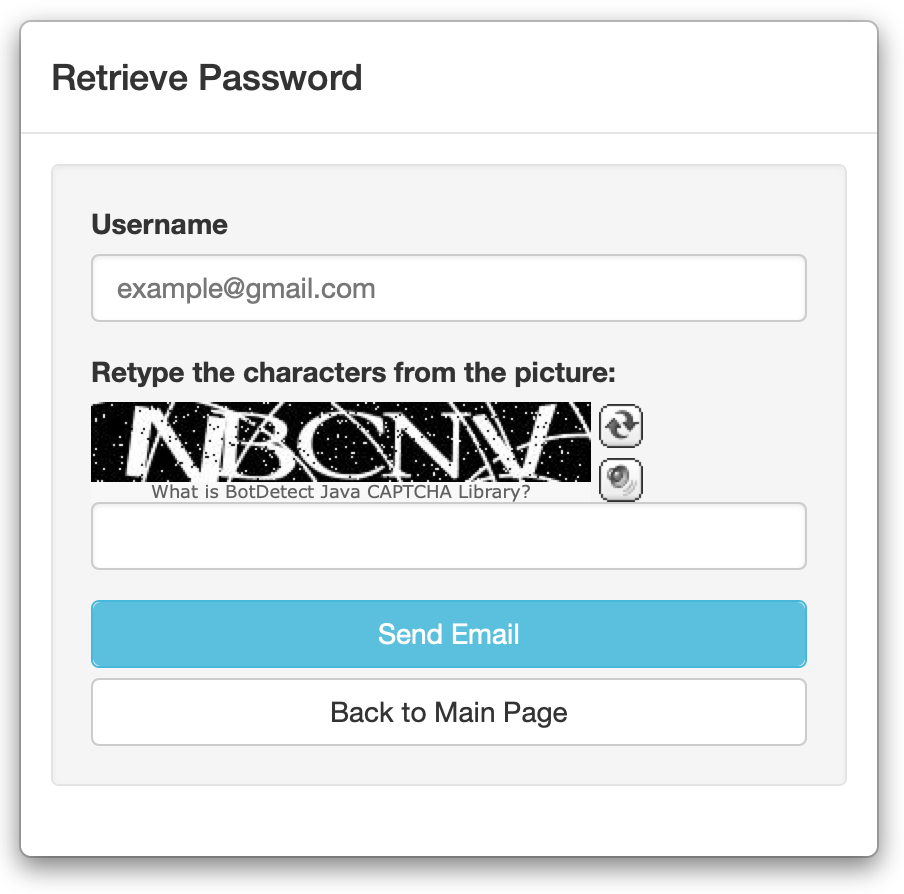
* Signup Page

****

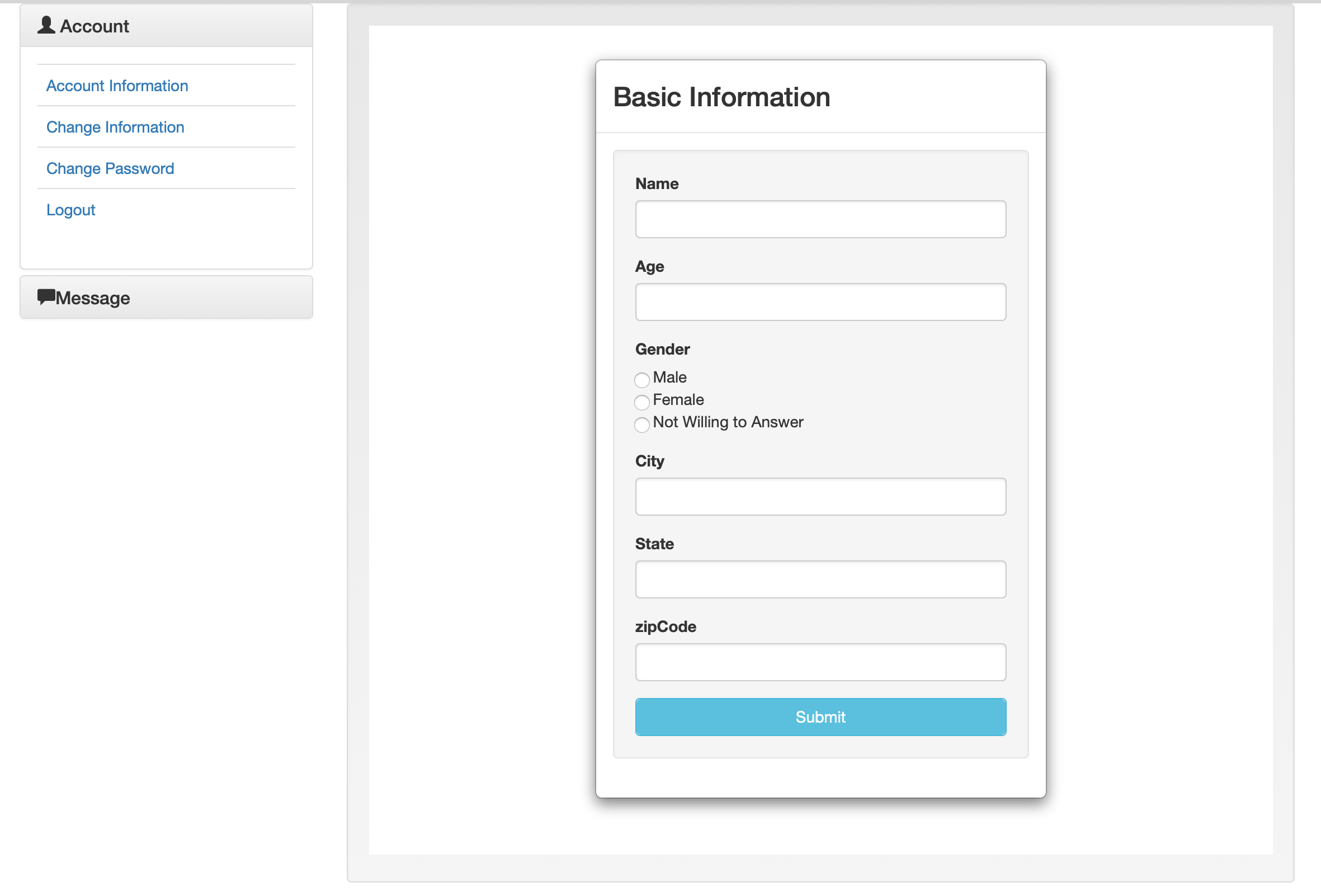
* Login Page



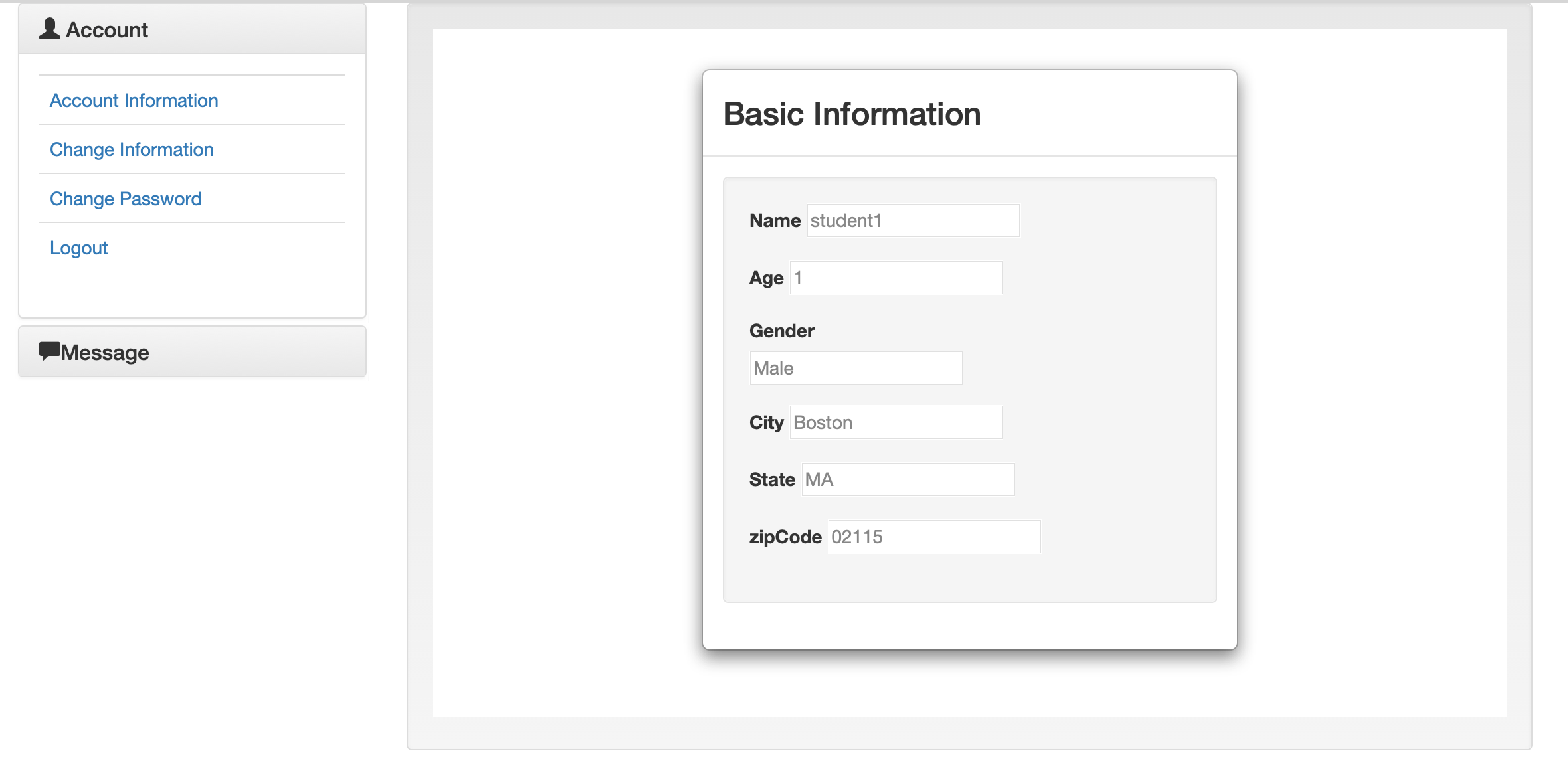
* Retrieve Password



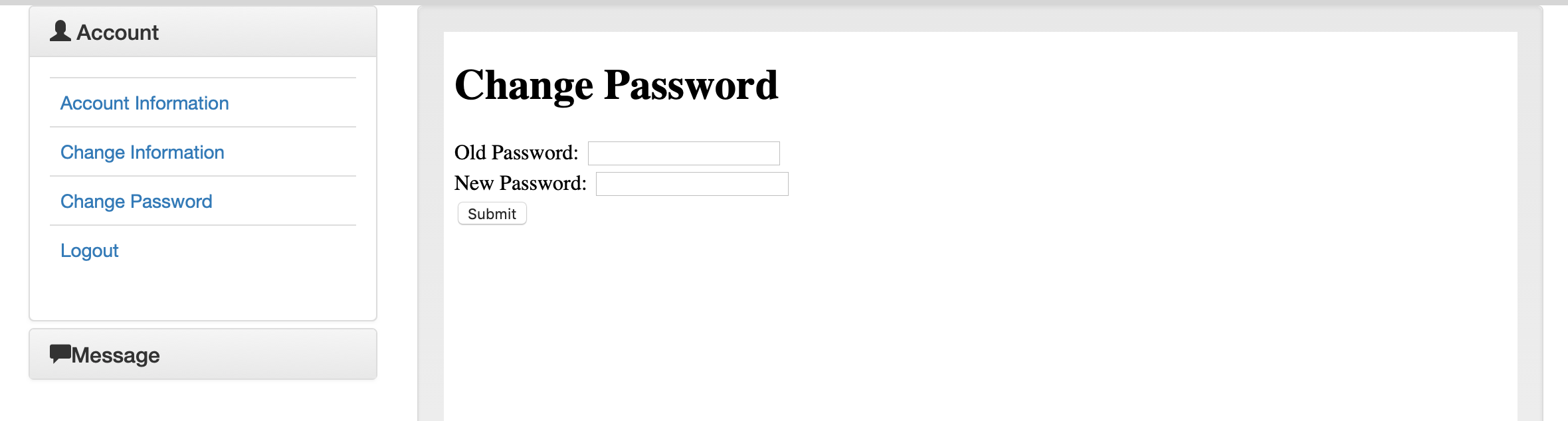
* Update new Information



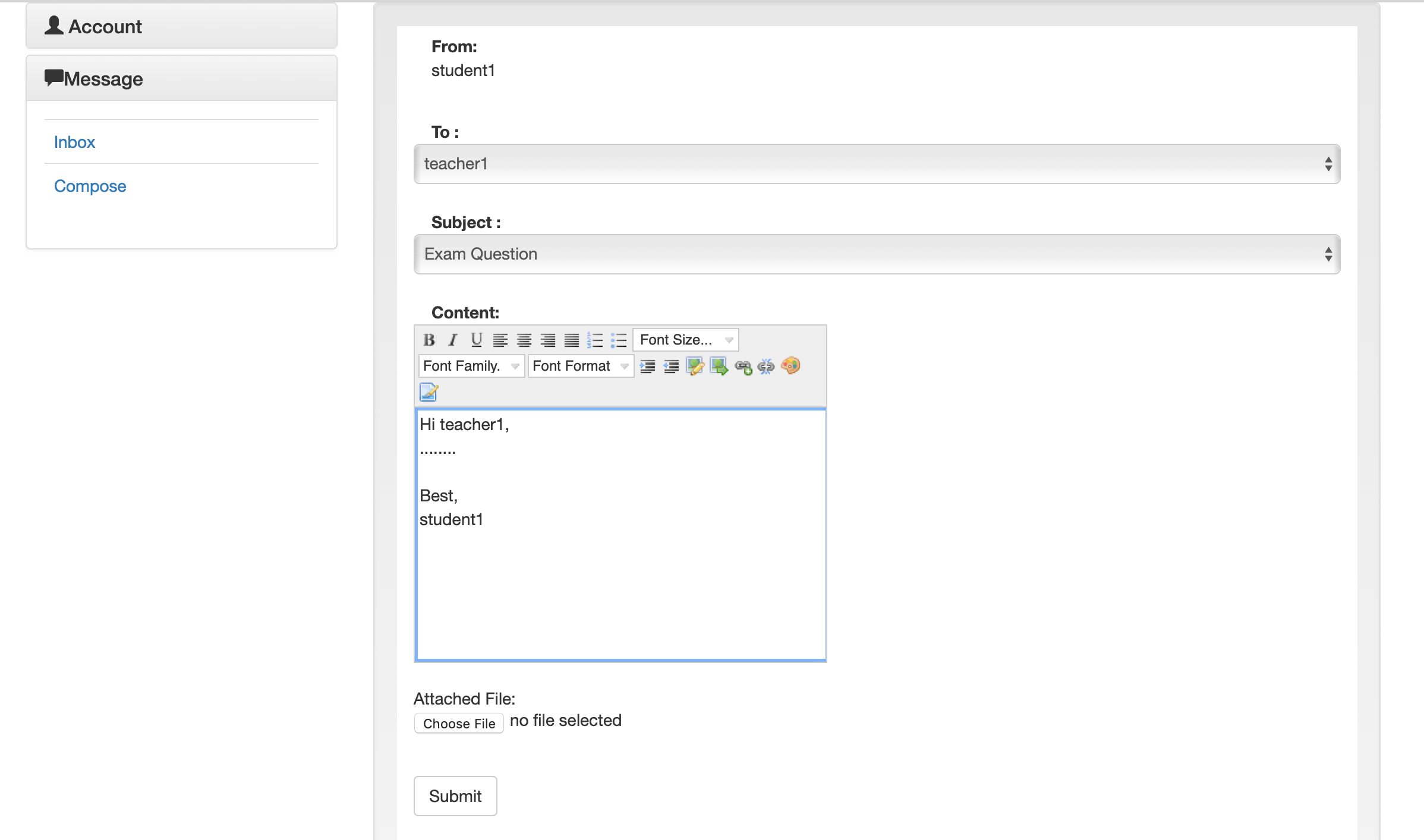
* Show basic information



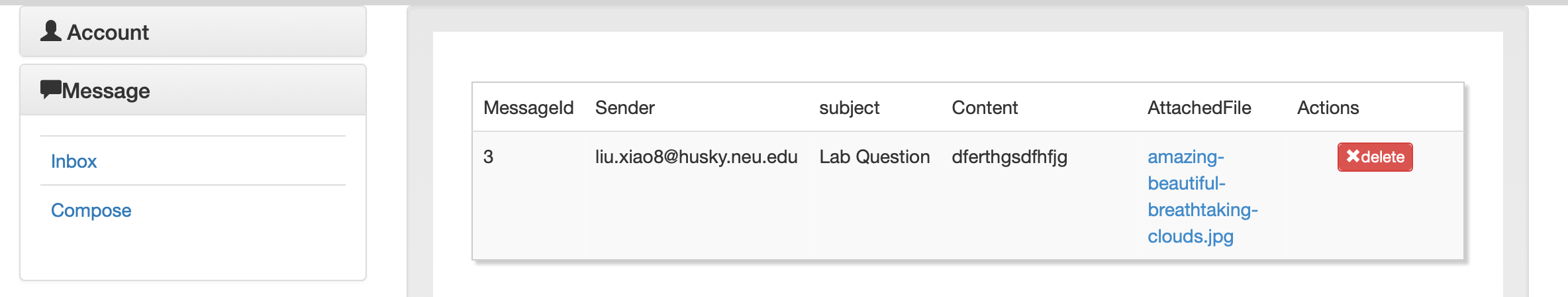
* Change old password



* Send messages and file



* Manage messages



1. **Appendix**

* **HomeController.java**

package com.mywork.finalproject.controller;

import com.captcha.botdetect.web.servlet.Captcha;

import java.util.ArrayList;

import java.util.Locale;

import java.util.Random;

import java.util.logging.Level;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpSession;

import org.apache.commons.mail.DefaultAuthenticator;

import org.apache.commons.mail.Email;

import org.apache.commons.mail.EmailException;

import org.apache.commons.mail.SimpleEmail;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

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

import org.springframework.web.bind.annotation.RequestMethod;

import com.mywork.finalproject.dao.UserDAO;

import com.mywork.finalproject.pojo.Student;

import com.mywork.finalproject.pojo.Teacher;

import com.mywork.finalproject.pojo.User;

/\*\*

\* Handles requests for the application home page.

\*/

@Controller

public class HomeController {

private static final Logger logger = LoggerFactory.getLogger(HomeController.class);

/\*\*

\* Simply selects the home view to render by returning its name.

\*/

@RequestMapping(value = "/user/register.htm", method = RequestMethod.GET)

public String home() {

return "userRegisterForm";

}

@RequestMapping(value = "/user/register.htm", method = RequestMethod.POST)

public String handleRegisterForm(HttpServletRequest request, UserDAO userDao, ModelMap map){

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

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

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

Captcha captcha = Captcha.load(request, "CaptchaObject");

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

if(userDao.get(username)!=null){

map.addAttribute("errorMessage", "This Email has been registered!");

return "error";

}

if(captcha.validate(captchaCode))

{

HttpSession session = request.getSession();

User user = new User();

if (role.equals("student"))

{

user = new Student();

}

else if (role.equals("teacher"))

{

user = new Teacher();

}

user.setUsername(username);

user.setPassword(password);

user.setStatus(0);//0代表未激活，1代表已激活

try{

User u = userDao.register(user);

Random rand = new Random();

int randomNum1 = rand.nextInt(5000000);

int randomNum2 = rand.nextInt(5000000);

try{

String str = "http://localhost:8080/finalproject/user/validateemail.htm?username=" + username + "&key1="

+ randomNum1 + "&key2=" + randomNum2 + "&role=" + role;

session.setAttribute("newUser", u);

session.setAttribute("key1", randomNum1);

session.setAttribute("key2", randomNum2);

sendEmail(username, "Click on this link to activate your account : " + str);

}catch(Exception e){

System.out.println("Email cannot be sent");

}

}catch(Exception e){

e.printStackTrace();

}

}else{

map.addAttribute("errorMessage", "Invalid Captcha!");

return "userRegisterForm";

}

return "userCreated";

}

public void sendEmail(String useremail, String message) {

try {

Email email = new SimpleEmail();

email.setHostName("smtp.googlemail.com");

email.setSmtpPort(465);

email.setAuthenticator(new DefaultAuthenticator("kuku.xiao1026@gmail.com", "sb4827590"));

email.setSSLOnConnect(true);

email.setFrom("kuku.xiao1026@gmail.com"); // This user email does not exist.

email.setSubject("INFO6250 FinalProjrct");

email.setMsg(message); // Retrieve email from the DAO and send this

email.addTo(useremail);

email.send();

} catch (EmailException ex) {

java.util.logging.Logger.getLogger(HomeController.class.getName()).log(Level.SEVERE, null, ex);

}

}

@RequestMapping(value = "/user/login.htm", method = RequestMethod.GET)

public String showLoginForm() {

return "home";

}

@RequestMapping(value = "/user/login.htm", method = RequestMethod.POST)

public String handleLoginForm(HttpServletRequest request, UserDAO userDao, ModelMap map){

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

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

try{

User user = userDao.get(username, password);

if(user != null && user.getStatus() == 0)

{

map.addAttribute("errorMessage", "Please activate your account to login!");

return "error";

}

else if(user != null && user.getStatus() == 1)

{

HttpSession session = request.getSession();

session.setAttribute("user", user);

if (user instanceof Student)

{

session.removeAttribute("list");

ArrayList<User> list = userDao.getAll();

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

if(list.get(i).getId() == user.getId()) {

list.remove(i);

}

}

session.setAttribute("list",list);

return "studentDashboard";

} else if (user instanceof Teacher)

{

session.removeAttribute("list");

ArrayList<User> list = userDao.getAll();

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

if(list.get(i).getId() == user.getId()) {

list.remove(i);

}

}

session.setAttribute("list",list);

return "teacherDashboard";

}

}else

{

map.addAttribute("errorMessage", "Invalid username/password!");

return "error";

}

}catch(Exception e)

{

e.printStackTrace();

}

return null;

}

@RequestMapping(value = "/user/validateemail.htm", method = RequestMethod.GET)

public String validateemail(HttpServletRequest request, UserDAO userDao, ModelMap map){

HttpSession session = request.getSession();

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

int key1 = Integer.parseInt(request.getParameter("key1"));

int key2 = Integer.parseInt(request.getParameter("key2"));

if((Integer)(session.getAttribute("key1")) == key1 && (Integer)(session.getAttribute("key2")) == key2)

{

try{

boolean updateStatus = userDao.updateUser(username);

if (updateStatus) {

return "home";

} else {

return "error";

}

}catch(Exception e){

e.printStackTrace();

}

}else{

map.addAttribute("errorMessage", "Link expired , generate new link");

map.addAttribute("resendLink", true);

return "error";

}

return "home";

}

@RequestMapping(value = "/user/forgotpassword.htm", method = RequestMethod.POST)

public String handleForgotPasswordForm(HttpServletRequest request, UserDAO userDao){

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

Captcha captcha = Captcha.load(request, "CaptchaObject");

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

User user = userDao.get(username);

if (captcha.validate(captchaCode) && user != null) {

sendEmail(username, "Your password is : " + user.getPassword());

return "forgotPasswordSuccess";

} else {

request.setAttribute("captchamsg", "Captcha is not valid");

return "forgotPassword";

}

}

@RequestMapping(value = "/user/forgotpassword.htm", method = RequestMethod.GET)

public String getForgotPasswordForm() {

return "forgotPassword";

}

}

* **AccountController.java**

package com.mywork.finalproject.controller;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpSession;

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

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

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.ResponseBody;

import com.mywork.finalproject.dao.UserDAO;

import com.mywork.finalproject.pojo.Student;

import com.mywork.finalproject.pojo.Teacher;

import com.mywork.finalproject.pojo.User;

@Controller

public class AccountController {

@RequestMapping(value = "/accountInfo/show.htm", method = RequestMethod.GET)

public String showAccountInfo(HttpServletRequest request, ModelMap map){

HttpSession session = request.getSession();

User currentUser = (User)session.getAttribute("user");

// prepare for showing different info in jsp page

if(currentUser instanceof Student) {

Student student = (Student) currentUser;

map.addAttribute("student", student);

return "studentInfo";

}

else if(currentUser instanceof Teacher) {

Teacher teacher = (Teacher) currentUser;

map.addAttribute("teacher", teacher);

return "teacherInfo";

}

return null;

}

@RequestMapping(value = "/accountInfo/changeInfo.htm", method = RequestMethod.GET)

public String changeAccount(HttpServletRequest request, ModelMap map){

HttpSession session = request.getSession();

User currentUser = (User)session.getAttribute("user");

// prepare for showing different info in jsp page

if(currentUser instanceof Student) {

Student student = (Student) currentUser;

map.addAttribute("student", student);

}

else if(currentUser instanceof Teacher) {

Teacher teacher = (Teacher) currentUser;

map.addAttribute("teacher", teacher);

}

return "changeAccountInfo";

}

@RequestMapping(value = "/accountInfo/changeInfo.htm", method = RequestMethod.POST)

public String changeResult(HttpServletRequest request, UserDAO userDao, ModelMap map)throws Exception{

HttpSession session = request.getSession();

User currentUser = (User)session.getAttribute("user");

int accountId = currentUser.getId();

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

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

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

if(currentUser instanceof Student) {

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

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

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

userDao.updateStudent(accountId+"", name, age, gender, city, state, zipCode);

}

else if(currentUser instanceof Teacher) {

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

userDao.updateTeacher(accountId+"", name, age, gender, subject);

}

return "changeInfoSuccess";

}

@RequestMapping(value = "/accountInfo/changePw.htm", method = RequestMethod.GET)

public String changePw(HttpServletRequest request, UserDAO userDao, ModelMap map){

return "changePassword";

}

@RequestMapping(value = "/accountInfo/changePw.htm", method = RequestMethod.POST)

public String showNewPw(HttpServletRequest request, UserDAO userDao, ModelMap map)throws Exception{

HttpSession session = request.getSession();

User currentUser = (User)session.getAttribute("user");

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

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

if(oldPw.equals(newPw)){

map.addAttribute("errorMessage", "The two passwords must be not same");

return "error";

}

else{

if(currentUser.getPassword().equals(oldPw)){

userDao.updateUserPassword(currentUser.getId()+"", newPw);

return "changeInfoSuccess";

}

else {

map.addAttribute("errorMessage", "The old password is wrong!");

return "error";

}

}

}

}

* **MessageController.java**

package com.mywork.finalproject.controller;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpSession;

import org.apache.commons.io.FileUtils;

import org.springframework.http.HttpHeaders;

import org.springframework.http.HttpStatus;

import org.springframework.http.MediaType;

import org.springframework.http.ResponseEntity;

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

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

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.multipart.MultipartFile;

import com.mywork.finalproject.dao.MessageDAO;

import com.mywork.finalproject.dao.UserDAO;

import com.mywork.finalproject.pojo.Message;

import com.mywork.finalproject.pojo.Student;

import com.mywork.finalproject.pojo.Teacher;

import com.mywork.finalproject.pojo.User;

import java.io.File;

import java.util.ArrayList;

import java.util.List;

@Controller

public class MessageController {

@RequestMapping(value = "/messages.htm", method = RequestMethod.GET)

public String showMessagePage(HttpServletRequest request, MessageDAO messageDao, UserDAO userDao, ModelMap map) {

HttpSession session = request.getSession();

User currentUser = (User) session.getAttribute("user");

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

if (action.equals("inbox")) {

String receiverUsername = currentUser.getUsername();

List<Message> messageList = messageDao.getByReceiver(receiverUsername);

map.addAttribute("messageList", messageList);

return "messagesInbox";

} else if (action.equals("compose")) {

map.addAttribute("sender", currentUser);

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

if (reply == null) {

return "messagesCompose";

}else if (reply.equals("yes")) {

map.addAttribute("replySender", currentUser);

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

map.addAttribute("receiver", request.getParameter("replyReceiver"));

map.addAttribute("reply", "yes");

map.addAttribute("subject", subject);

return "messagesCompose";

}

} else if (action.equals("reply")) {

return "messagesInbox";

} else if(action.equals("delete")) {

int messageid = Integer.parseInt(request.getParameter("messageid"));

messageDao.delete(messageid);

return "redirect:/messages.htm?action=inbox";

}

return null;

}

@RequestMapping(value = "/messages.htm", method = RequestMethod.POST)

public String handleMessagesRequests(@RequestParam("attachedfile") MultipartFile file, HttpServletRequest request, UserDAO userDao, MessageDAO messageDao, ModelMap map) throws Exception {

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

if (action.equals("compose")) {

HttpSession session = request.getSession();

String receivername = request.getParameter("receiver");

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

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

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

Message message = new Message();

message.setSubject(subject);

String username = "";

if(userDao.getStudent(receivername) == null) {

username = userDao.getTeacher(receivername).getUsername();

}else {

username = userDao.getStudent(receivername).getUsername();

}

User receiver = userDao.get(username);

message.setReceiver(receiver.getUsername());

User sender = (User) session.getAttribute("user");

message.setSender(sender.getUsername());

message.setContent(content);

// 判断这个文件不为空

if (!file.isEmpty()) {

// 服务端的images目录需要手动创建好,上传到服务器目录下

// String path = session.getServletContext().getRealPath("/images");

String path = "/Users/lx/Sites/INFO6250FinalProject";

// 获取原始文件名

String fileName = file.getOriginalFilename();

// 截取文件的扩展名

String extName = fileName.substring(fileName.lastIndexOf("."));

File myFile = new File(path, fileName);

// 完成文件上传

file.transferTo(myFile);

message.setAttachedfile(fileName);

}

messageDao.create(message);

if (replyFlag == null) {

return "redirect:/messages.htm?action=inbox";

}

if (replyFlag.equals("finished")) {

String text = "You have successfully replied to " + receiver.getUsername();

map.addAttribute("content", text);

return "success";

}

}

return null;

}

@RequestMapping(value = "/message/downloadFile.htm", method = RequestMethod.GET)

public ResponseEntity<byte[]> downloadAssignment(HttpServletRequest request, @RequestParam("filename") String filename, ModelMap model) throws Exception {

String path = "/Users/lx/Sites/INFO6250FinalProject";

File file = new File(path + File.separator + filename);

HttpHeaders headers = new HttpHeaders();

// 下载显示的文件名，解决中文名称乱码问题

String downloadFileName = new String(filename.getBytes("UTF-8"), "iso-8859-1");

// 通知浏览器以attachment（下载方式）打开图片

headers.setContentDispositionFormData("attachment", downloadFileName);

// application/octet-stream ： 二进制流数据（最常见的文件下载）。

headers.setContentType(MediaType.APPLICATION\_OCTET\_STREAM);

return new ResponseEntity<byte[]>(FileUtils.readFileToByteArray(file), headers, HttpStatus.CREATED);

}

}

* **DAO.java**

package com.mywork.finalproject.dao;

import java.util.logging.Level;

import java.util.logging.Logger;

import org.hibernate.HibernateException;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.cfg.Configuration;

public class DAO {

public DAO() {

}

static final Logger log = Logger.getAnonymousLogger();

private final SessionFactory sf = new Configuration().configure("hibernate.cfg.xml").buildSessionFactory();

private Session session;

public Session getSession() throws Exception

{

if(session == null || !session.isOpen())

{

session = sf.openSession();

}

return session;

}

public void begin() throws Exception

{

getSession().beginTransaction();

}

public void commit() throws Exception{

getSession().getTransaction().commit();

}

public void rollback() throws Exception

{

try{

getSession().getTransaction().rollback();

}

catch(HibernateException e){

log.log(Level.WARNING, "Cannot rollback", e);

}

}

public void close() throws Exception

{

getSession().close();

}

}

* **UserDAO**

package com.mywork.finalproject.dao;

import java.util.ArrayList;

import java.util.List;

import org.hibernate.HibernateException;

import org.hibernate.Query;

import com.mywork.finalproject.pojo.Student;

import com.mywork.finalproject.pojo.Teacher;

import com.mywork.finalproject.pojo.User;

public class UserDAO extends DAO {

public User register(User u) throws Exception

{

try{

begin();

getSession().save(u);

commit();

return u;

}catch(HibernateException e){

rollback();

throw new Exception("Exception while creating user: " + e.getMessage());

}

}

public User get(String username){

try{

begin();

Query q = getSession().createQuery("from User where username = :username");

q.setString("username", username);

User user = (User) q.uniqueResult();

close();

return user;

}catch(Exception e){

e.printStackTrace();

}

return null;

}

public Student getStudent(String name){

try{

begin();

Query q = getSession().createQuery("from Student where name = :name");

q.setString("name", name);

Student student = (Student) q.uniqueResult();

close();

return student;

}catch(Exception e){

e.printStackTrace();

}

return null;

}

public Teacher getTeacher(String name){

try{

begin();

Query q = getSession().createQuery("from User where name = :name");

q.setString("name", name);

Teacher teacher = (Teacher) q.uniqueResult();

close();

return teacher;

}catch(Exception e){

e.printStackTrace();

}

return null;

}

public ArrayList<User> getAll(){

try{

begin();

Query q = getSession().createQuery("from User");

ArrayList<User> list = (ArrayList)q.list();

close();

return list;

}catch(Exception e){

e.printStackTrace();

}

return null;

}

public ArrayList<Teacher> getAllTeacher(){

try{

begin();

Query q = getSession().createQuery("from Teacher");

ArrayList<Teacher> list = (ArrayList)q.list();

close();

return list;

}catch(Exception e){

e.printStackTrace();

}

return null;

}

public ArrayList<Student> getAllStudent(){

try{

begin();

Query q = getSession().createQuery("from Student");

ArrayList<Student> list = (ArrayList)q.list();

close();

return list;

}catch(Exception e){

e.printStackTrace();

}

return null;

}

public User get(String username,String password)

{

try {

begin();

Query q = getSession().createQuery("from User where username = :username and password = :password");

q.setString("username", username);

q.setString("password", password);

User user = (User) q.uniqueResult();

if(user == null){

}else{

close();

return user;

}

}catch(Exception e){

System.out.println(e.getMessage());

}

return null;

}

public boolean updateUser(String username) throws Exception {

try{

begin();

Query q = getSession().createQuery("from User where username = :username");

q.setString("username", username);

User user = (User)q.uniqueResult();

if(user != null){

user.setStatus(1);

getSession().update(user);

commit();

return true;

}else{

return false;

}

}catch(HibernateException e){

rollback();

throw new Exception("Exception while creating user: " + e.getMessage());

}

}

public boolean updateStudent(String id, String name, String age, String gender,

String city, String state, String zipCode) throws Exception {

try {

begin();

Query q = getSession().createQuery("from Student where id = :id");

q.setString("id", id);

Student student = (Student)q.uniqueResult();

if(student != null){

student.setName(name);

student.setAge(Integer.parseInt(age));

student.setGender(gender);

student.setCity(city);

student.setState(state);

student.setZipCode(zipCode);

getSession().update(student);

commit();

close();

return true;

}else{

return false;

}

}catch(HibernateException e){

rollback();

throw new Exception("Exception while creating user: " + e.getMessage());

}

}

public boolean updateTeacher(String id, String name, String age, String gender, String subject)throws Exception{

try{

begin();

Query q = getSession().createQuery("from Teacher where id = :id");

q.setString("id", id);

Teacher teacher = (Teacher)q.uniqueResult();

if(teacher != null){

teacher.setName(name);

teacher.setAge(Integer.parseInt(age));

teacher.setGender(gender);

teacher.setSubject(subject);

getSession().update(teacher);

commit();

close();

return true;

}else{

return false;

}

}catch(HibernateException e){

rollback();

throw new Exception("Exception while creating user: " + e.getMessage());

}

}

public boolean updateUserPassword(String id, String password) throws Exception{

try{

begin();

Query q = getSession().createQuery("from User where id = :id");

q.setString("id", id);

User user = (User) q.uniqueResult();

if(user != null){

user.setPassword(password);

getSession().update(user);

commit();

return true;

}else{

return false;

}

}catch(HibernateException e){

rollback();

throw new Exception("Exception while creating user: " + e.getMessage());

}

}

public Student addInfo(Student student) throws Exception{

try {

begin();

getSession().save(student);

commit();

return student;

} catch (HibernateException e) {

rollback();

throw new Exception("Exception : " + e.getMessage());

}

}

}

* **MessageDAO.java**

package com.mywork.finalproject.dao;

import java.util.List;

import org.hibernate.HibernateException;

import org.hibernate.query.Query;

import com.mywork.finalproject.pojo.Message;

public class MessageDAO extends DAO{

public List<Message> getByReceiver(String receiverUsername) {

try {

begin();

Query q = getSession().createQuery("from Message where receiver = :receiver");

q.setString("receiver", receiverUsername);

List<Message> messages = q.list();

close();

return messages;

} catch (Exception e) {

System.out.println(e.getMessage());

}

return null;

}

public Message create(Message message) throws Exception{

try {

begin();

System.out.println("inside DAO");

getSession().save(message);

commit();

return message;

}catch(HibernateException e) {

rollback();

throw new Exception("Exception while creating message: " + e.getMessage());

}

}

public boolean update(String id,String subject,String attachedfile,String content) throws Exception{

try {

begin();

System.out.println("inside DAO");

Query q = getSession().createQuery("from Message where messageid = : id");

q.setString("id", id);

Message message = (Message)q.uniqueResult();

if(message != null) {

message.setSubject(subject);

message.setAttachedfile(attachedfile);

message.setContent(content);

getSession().update(message);

commit();

return true;

}else {

return false;

}

}catch(HibernateException e) {

rollback();

throw new Exception("Exception while editing appointment :" + e.getMessage());

}

}

public boolean delete(int messageid) {

try {

begin();

Query q = getSession().createQuery("delete from Message where messageid = :messageid");

q.setInteger("messageid", messageid);

q.executeUpdate();

close();

return true;

} catch (Exception e) {

System.out.println(e.getMessage());

}

return false;

}

}

* **User.java**

package com.mywork.finalproject.pojo;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Inheritance;

import javax.persistence.InheritanceType;

import javax.persistence.Table;

@Entity

@Table(name = "user\_table")

@Inheritance(strategy = InheritanceType.JOINED)

public class User {

public User() {

}

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "id", unique = true, nullable = false)

private int id;

@Column(name = "username")

private String username;

@Column(name = "password")

private String password;

@Column(name = "status")

private int status;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

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 int getStatus() {

return status;

}

public void setStatus(int status) {

this.status = status;

}

}

* **Student.java**

package com.mywork.finalproject.pojo;

import javax.persistence.Column;

import javax.persistence.Entity;

@Entity

public class Student extends User{

public Student() {

super();

}

@Column(name="name")

private String name;

@Column(name="age")

private int age;

@Column(name="gender")

private String gender;

@Column(name="city")

private String city;

@Column(name="state")

private String state;

@Column(name="zipCode")

private String zipCode;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

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 String getCity() {

return city;

}

public void setCity(String city) {

this.city = city;

}

public String getState() {

return state;

}

public void setState(String state) {

this.state = state;

}

public String getZipCode() {

return zipCode;

}

public void setZipCode(String zipCode) {

this.zipCode = zipCode;

}

}

* **Teacher.java**

package com.mywork.finalproject.pojo;

import javax.persistence.Column;

import javax.persistence.Entity;

@Entity

public class Teacher extends User{

public Teacher() {

super();

}

@Column(name="name")

private String name;

@Column(name="age")

private int age;

@Column(name="gender")

private String gender;

@Column(name="subject")

private String subject;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

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 String getSubject() {

return subject;

}

public void setSubject(String subject) {

this.subject = subject;

}

}

* **Message.java**

package com.mywork.finalproject.pojo;

import javax.persistence.CascadeType;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToOne;

import javax.persistence.Table;

import javax.persistence.FetchType;

@Entity

@Table(name = "Message")

public class Message {

public Message() {

}

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name="messageid", unique = true, nullable = false)

private int messageid;

@Column(name = "sender")

private String sender;

@Column(name = "receiver")

private String receiver;

@Column(name="subject")

private String subject;

@Column(name="content")

private String content;

@Column(name="attachedfile")

private String attachedfile;

public int getMessageid() {

return messageid;

}

public void setMessageid(int messageid) {

this.messageid = messageid;

}

public String getSender() {

return sender;

}

public void setSender(String sender) {

this.sender = sender;

}

public String getReceiver() {

return receiver;

}

public void setReceiver(String receiver) {

this.receiver = receiver;

}

public String getSubject() {

return subject;

}

public void setSubject(String subject) {

this.subject = subject;

}

public String getContent() {

return content;

}

public void setContent(String content) {

this.content = content;

}

public String getAttachedfile() {

return attachedfile;

}

public void setAttachedfile(String attachedfile) {

this.attachedfile = attachedfile;

}

}