1. **<https://github.com/mirna-hatemm/university-timetabling>**
2. [**https://swe-s7.atlassian.net/browse/UN-1?atlOrigin=eyJpIjoiYjQ4ZWM0MDI3NTA4NDkwMmE0ZjNhZTNkZTNmMjQ3YzIiLCJwIjoiaiJ9**](https://swe-s7.atlassian.net/browse/UN-1?atlOrigin=eyJpIjoiYjQ4ZWM0MDI3NTA4NDkwMmE0ZjNhZTNkZTNmMjQ3YzIiLCJwIjoiaiJ9)
3. **Description of what the project does:**

University-timetabling is a comprehensive educational scheduling system that supports developing course and exam timetables, managing changes to these timetables, sharing rooms with other events, and scheduling students to individual classes. It is a distributed system that allows multiple university and departmental schedule managers to coordinate efforts to build and modify a schedule that meets their diverse organizational needs while allowing for minimization of student course conflicts. It can be used alone to create and maintain a school's schedule of classes and/or exams, or interfaced with an existing student information system.

We used the **opportunistic approach technique** to obtain this description as we begin with top-down approach to gain an overview of the University-timetabling program functions, then apply bottom-up strategies by looking for the recognizable idioms within the code and then combine these recognized units to understand even larger sections of the code and verify hypotheses resulting from top-down reading was correct.

1. **A description for the feature that you will start working on it:**

In uni-timetabling project when a user (student or instructor) forgets his password and tries to login he uses the option of “forgot password” which ask him to enter his email address to receive a link to restore his password or a temporarily password to use to login then change it but what if he can’t access his email address right now?

so we will add a feature of accepting this link or temporarily password on the mobile number as a “SMS”

This will require some changes in the tables of data base (user, instructor) to add the phone number of each user and changes in the class named passwordchangebackend.java we will add the implementations of receiving on phone number

/\*

\* Licensed to The Apereo Foundation under one or more contributor license

\* agreements. See the NOTICE file distributed with this work for

\* additional information regarding copyright ownership.

\*

\* The Apereo Foundation licenses this file to you under the Apache License,

\* Version 2.0 (the "License"); you may not use this file except in

\* compliance with the License. You may obtain a copy of the License at:

\*

\* http://www.apache.org/licenses/LICENSE-2.0

\*

\* Unless required by applicable law or agreed to in writing, software

\* distributed under the License is distributed on an "AS IS" BASIS,

\* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

\*

\* See the License for the specific language governing permissions and

\* limitations under the License.

\*

\*/

package org.unitime.timetable.server;

import java.io.PrintWriter;

import java.io.StringWriter;

import java.util.Date;

import java.util.HashMap;

import java.util.HashSet;

import java.util.List;

import java.util.Map;

import java.util.Random;

import java.util.Set;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;

import org.unitime.commons.Email;

import org.unitime.localization.impl.Localization;

import org.unitime.timetable.defaults.ApplicationProperty;

import org.unitime.timetable.gwt.client.admin.PasswordPage.PasswordChangeRequest;

import org.unitime.timetable.gwt.client.admin.PasswordPage.PasswordChangeResponse;

import org.unitime.timetable.gwt.command.client.GwtRpcException;

import org.unitime.timetable.gwt.command.server.GwtRpcImplementation;

import org.unitime.timetable.gwt.command.server.GwtRpcImplements;

import org.unitime.timetable.gwt.resources.GwtConstants;

import org.unitime.timetable.gwt.resources.GwtMessages;

import org.unitime.timetable.model.User;

import org.unitime.timetable.model.UserData;

import org.unitime.timetable.model.dao.UserDAO;

import org.unitime.timetable.security.SessionContext;

import org.unitime.timetable.security.rights.Right;

import org.unitime.timetable.spring.security.MD5PasswordEncoder;

import org.unitime.timetable.util.Constants;

import freemarker.template.Configuration;

import freemarker.template.Template;

/\*\*

\* @author Tomas Muller

\*/

@GwtRpcImplements(PasswordChangeRequest.class)

public class PasswordChangeBackend implements GwtRpcImplementation<PasswordChangeRequest, PasswordChangeResponse>{

protected static GwtMessages MESSAGES = Localization.create(GwtMessages.class);

protected static GwtConstants CONSTANTS = Localization.create(GwtConstants.class);

private static final String sGenCharset = "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";

@Autowired HttpServletRequest iRequest;

@Override

public PasswordChangeResponse execute(PasswordChangeRequest request, SessionContext context) {

org.hibernate.Session hibSession = UserDAO.getInstance().getSession();

if (request.isReset()) {

try {

Set<String> userIds = new HashSet<String>();

userIds.addAll((List<String>)hibSession.createQuery(

"select distinct externalUniqueId from TimetableManager where lower(emailAddress) = :email and externalUniqueId is not null")

.setString("email", request.getEmail().toLowerCase()).list());

userIds.addAll((List<String>)hibSession.createQuery(

"select distinct externalUniqueId from Staff where lower(email) = :email and externalUniqueId is not null")

.setString("email", request.getEmail().toLowerCase()).list());

userIds.addAll((List<String>)hibSession.createQuery(

"select distinct externalUniqueId from DepartmentalInstructor where lower(email) = :email and externalUniqueId is not null")

.setString("email", request.getEmail().toLowerCase()).list());

userIds.addAll((List<String>)hibSession.createQuery(

"select distinct externalUniqueId from Student where lower(email) = :email and externalUniqueId is not null")

.setString("email", request.getEmail().toLowerCase()).list());

if (userIds.isEmpty())

throw new GwtRpcException(MESSAGES.errorEmailNotValid());

boolean matched = false;

for (String userId: userIds) {

User user = User.findByExternalId(userId);

if (user == null) continue;

matched = true;

String key = "";

Random rnd = new Random();

for (int i = 0; i < 32; i++)

key += sGenCharset.charAt(rnd.nextInt(sGenCharset.length()));

UserData.setProperty(userId, "Password.TempKey", encode(key));

UserData.setProperty(userId, "Password.TempKeyStamp", Long.toString(System.currentTimeMillis() + 48 \* 60 \* 60 \* 1000, 16));

Email email = Email.createEmail();

email.setSubject(MESSAGES.emailPasswordChange());

Configuration cfg = new Configuration(Configuration.VERSION\_2\_3\_0);

cfg.setClassForTemplateLoading(PasswordChangeBackend.class, "");

cfg.setLocale(Localization.getJavaLocale());

cfg.setOutputEncoding("utf-8");

cfg.setEncoding(Localization.getJavaLocale(), "utf-8");

Template template = cfg.getTemplate("PasswordResetEmail.ftl");

Map<String, Object> input = new HashMap<String, Object>();

input.put("msg", MESSAGES);

input.put("const", CONSTANTS);

String url = iRequest.getScheme()+"://"+iRequest.getServerName()+":"+iRequest.getServerPort()+iRequest.getContextPath();

if (!url.endsWith("/")) url += "/";

url += "gwt.jsp?page=password&user=" + user.getUsername() + "&key=" + key;

input.put("username", user.getUsername());

input.put("url", url);

input.put("version", MESSAGES.pageVersion(Constants.getVersion(), Constants.getReleaseDate()));

input.put("ts", new Date());

input.put("sender", ApplicationProperty.EmailSenderName.value());

StringWriter s = new StringWriter();

template.process(input, new PrintWriter(s));

s.flush(); s.close();

email.setHTML(s.toString());

email.addRecipient(request.getEmail(), null);

email.send();

}

if (!matched)

throw new GwtRpcException(MESSAGES.errorNoMatchingUser());

} catch (GwtRpcException e) {

throw e;

} catch (Exception e) {

throw new GwtRpcException(MESSAGES.failedToResetPassword(e.getMessage()), e);

}

} else {

try {

if (request.hasUsername()) {

User user = User.findByUserName(request.getUsername());

if (user == null)

throw new GwtRpcException(MESSAGES.errorBadCredentials());

if (encode(request.getOldPassword()).equals(UserData.getProperty(user.getExternalUniqueId(), "Password.TempKey"))) {

long ts = Long.valueOf(UserData.getProperty(user.getExternalUniqueId(), "Password.TempKeyStamp", "0"), 16);

if (ts < System.currentTimeMillis())

throw new GwtRpcException(MESSAGES.errorPasswordResetExpired());

user.setPassword(encode(request.getNewPassword()));

hibSession.update(user);

hibSession.flush();

UserData.removeProperty(user.getExternalUniqueId(), "Password.TempKey");

UserData.removeProperty(user.getExternalUniqueId(), "Password.TempKeyStamp");

return new PasswordChangeResponse();

}

} else {

context.checkPermissionAnyAuthority(Right.ChangePassword);

}

String username = (request.hasUsername() ? request.getUsername() : context.getUser().getUsername());

User user = (User)hibSession.createQuery(

"from User where username = :username and password = :password")

.setString("username", username)

.setString("password", encode(request.getOldPassword()))

.setMaxResults(1).uniqueResult();

if (user == null) {

if (request.hasUsername())

throw new GwtRpcException(MESSAGES.errorBadCredentials());

else

throw new GwtRpcException(MESSAGES.errorOldPasswordNotValid());

}

if (request.getNewPassword() == null || request.getNewPassword().isEmpty())

throw new GwtRpcException(MESSAGES.errorEnterNewPassword());

user.setPassword(encode(request.getNewPassword()));

hibSession.update(user);

hibSession.flush();

} catch (IllegalArgumentException e) {

throw new GwtRpcException(MESSAGES.failedToChangePassword(e.getMessage()), e);

}

}

return new PasswordChangeResponse();

}

private static String encode(String password) {

return MD5PasswordEncoder.getEncodedPassword(password);

}

}

this is the class which will we add the implementation.

The expected results from this feature is to have the ability to recover forgoten password throw receiving of SMS on the phone you provided when craeting account on the site .

this task now is "To do "

this task is assigned to the members of this team