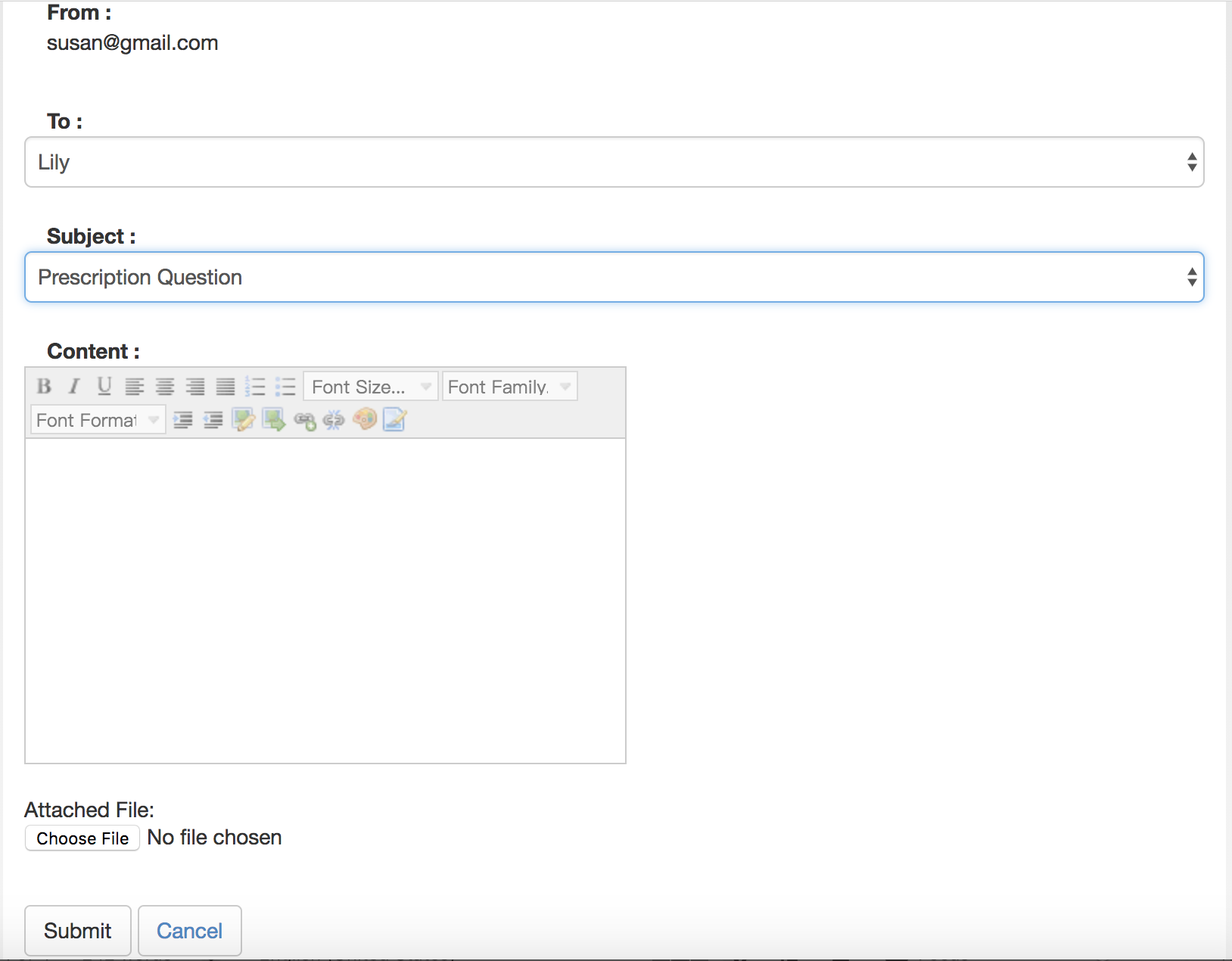
Report of Doctor – Patient Communication Platform

1. Introduction

I built a communication platform between doctors and patients to help them send and reply messages, request and confirm appointments, write recipes with test conducted and prescriptions assigned, manage patients, retrieve medical histories and update basic contact information and password.

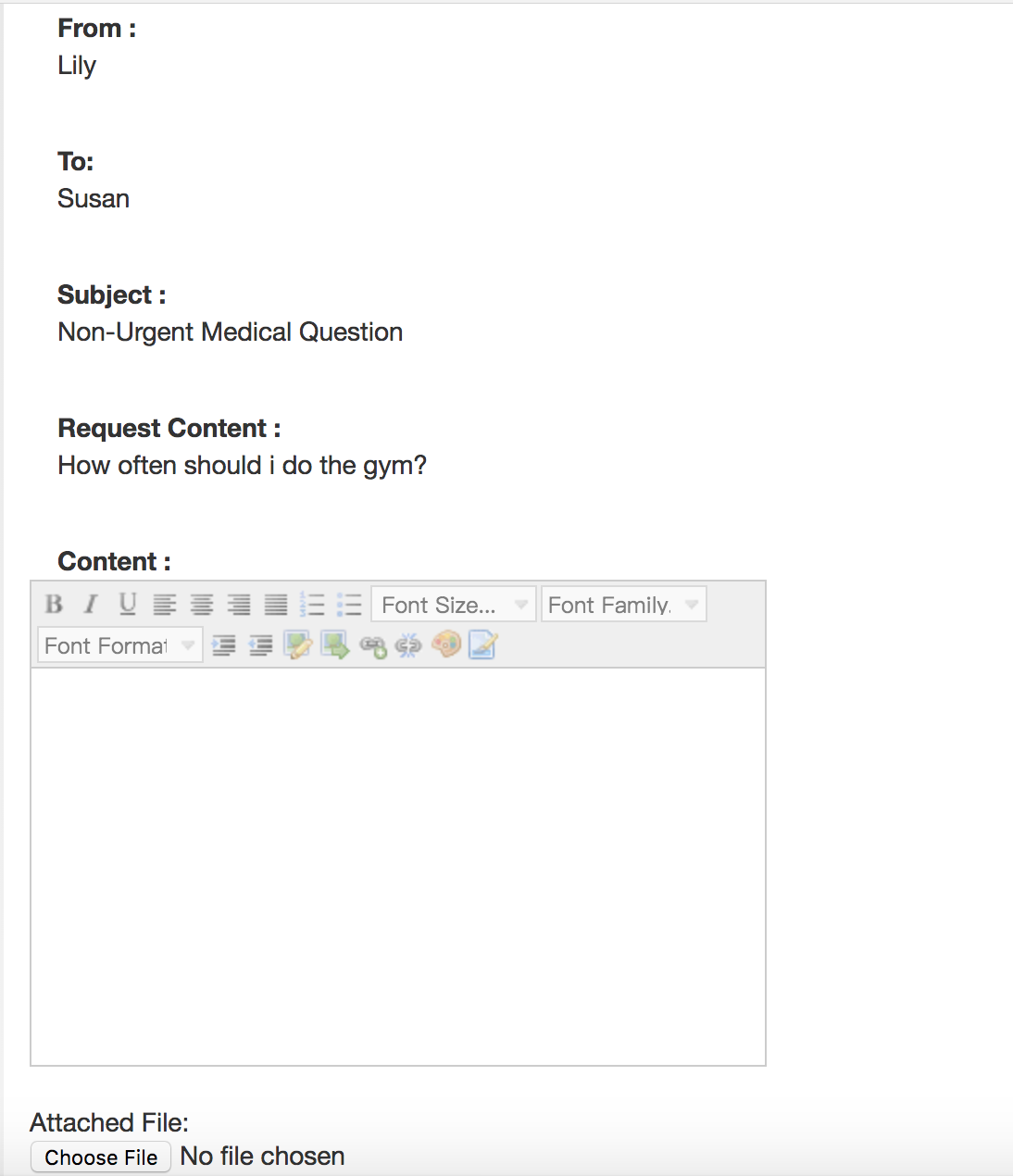
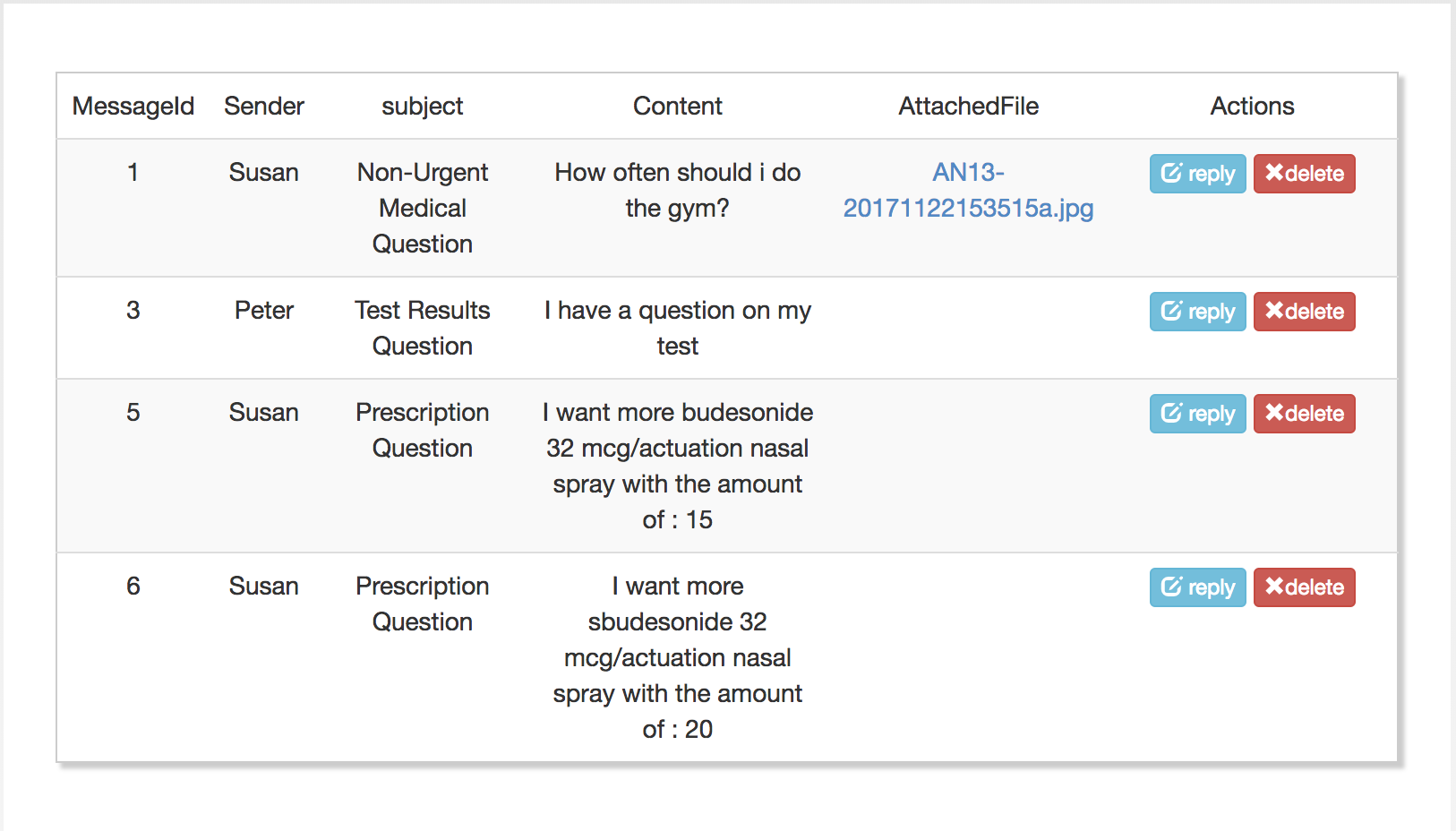
1. Functionality
2. Send Message

“From” part can automatically insert the current user’s username, and “To” part is patients lists or doctor lists corresponding to the role of current user. You can add sentences and change their styles in “Content” part, you also can upload attached files.



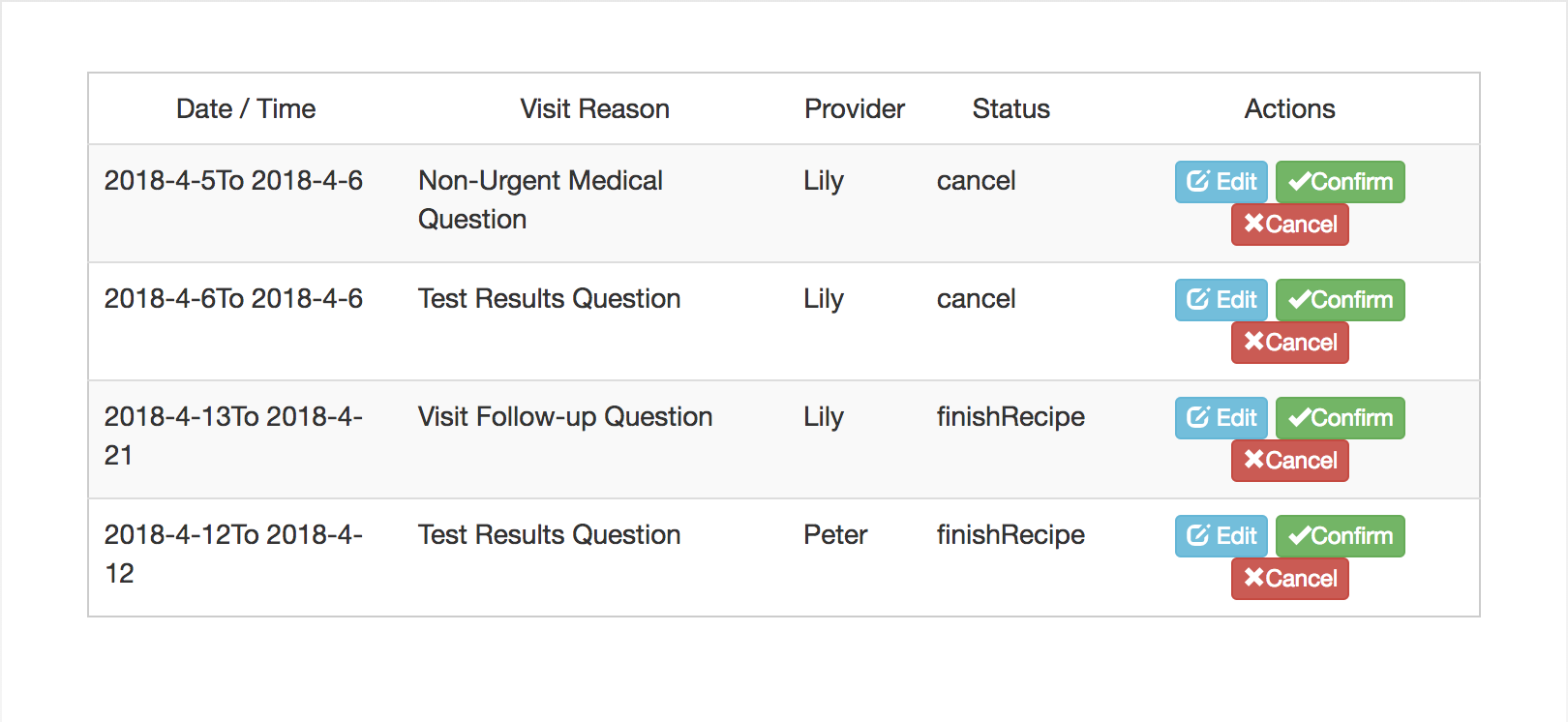
1. Reply and delete in message inbox

The reply page has “From”, “To”, “Subject” settled, “Request Content” hint and editable “Content” part



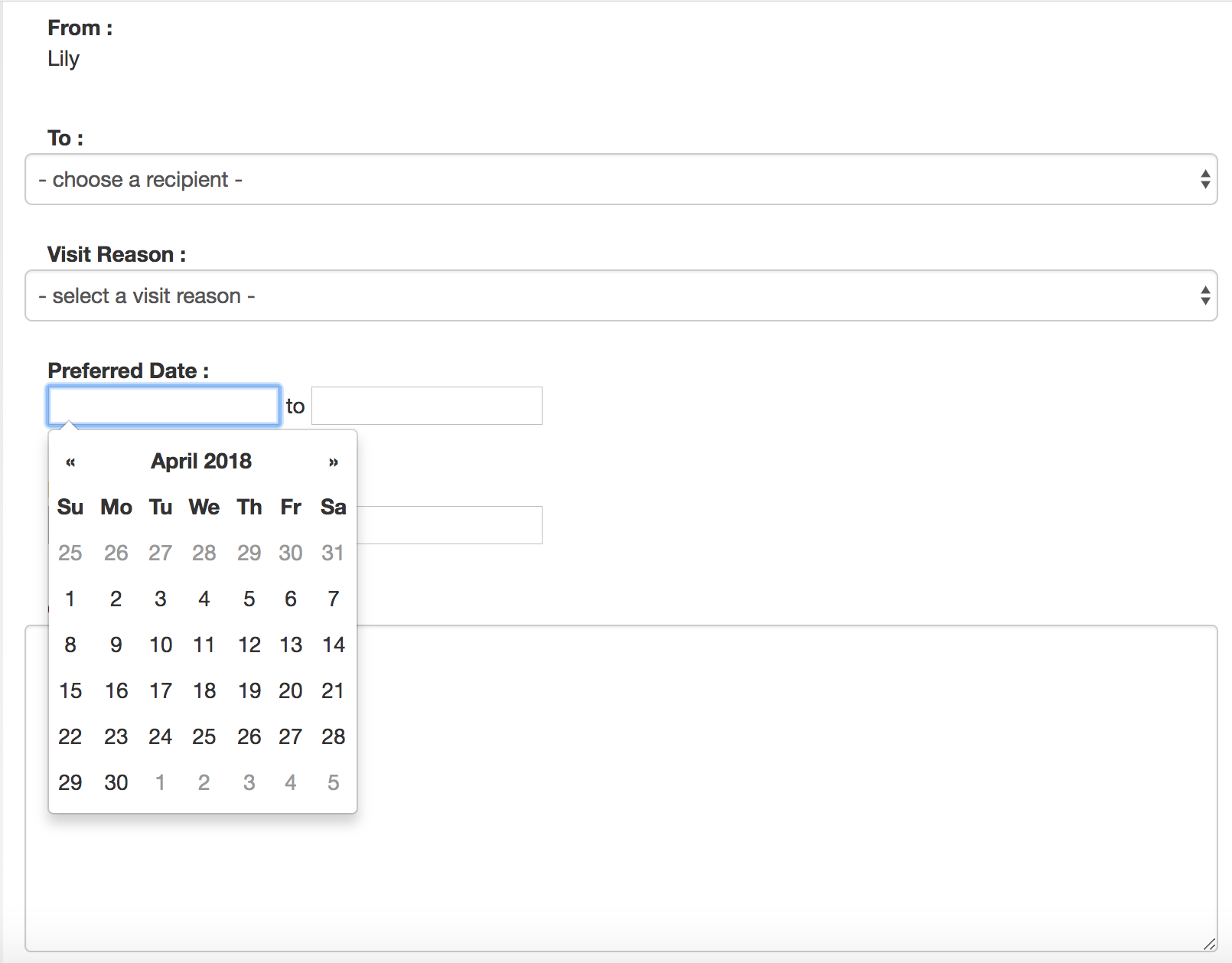
1. Manage Appointments

You can confirm or cancel this appointment, or change the date or time of this appointment



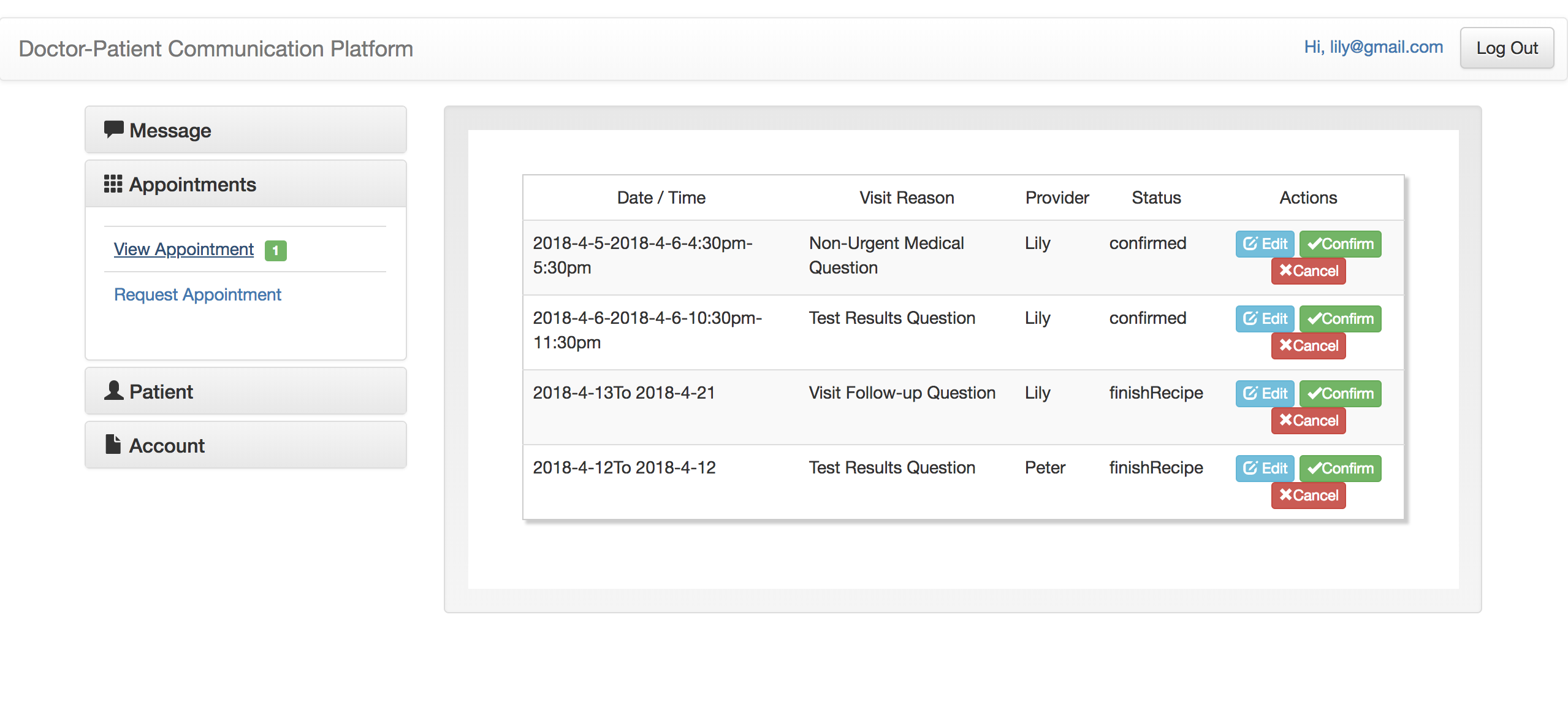
1. Request Appointment

“To” part will show patient lists or doctor lists corresponding to the role of current user, you can select preferred data and time range.



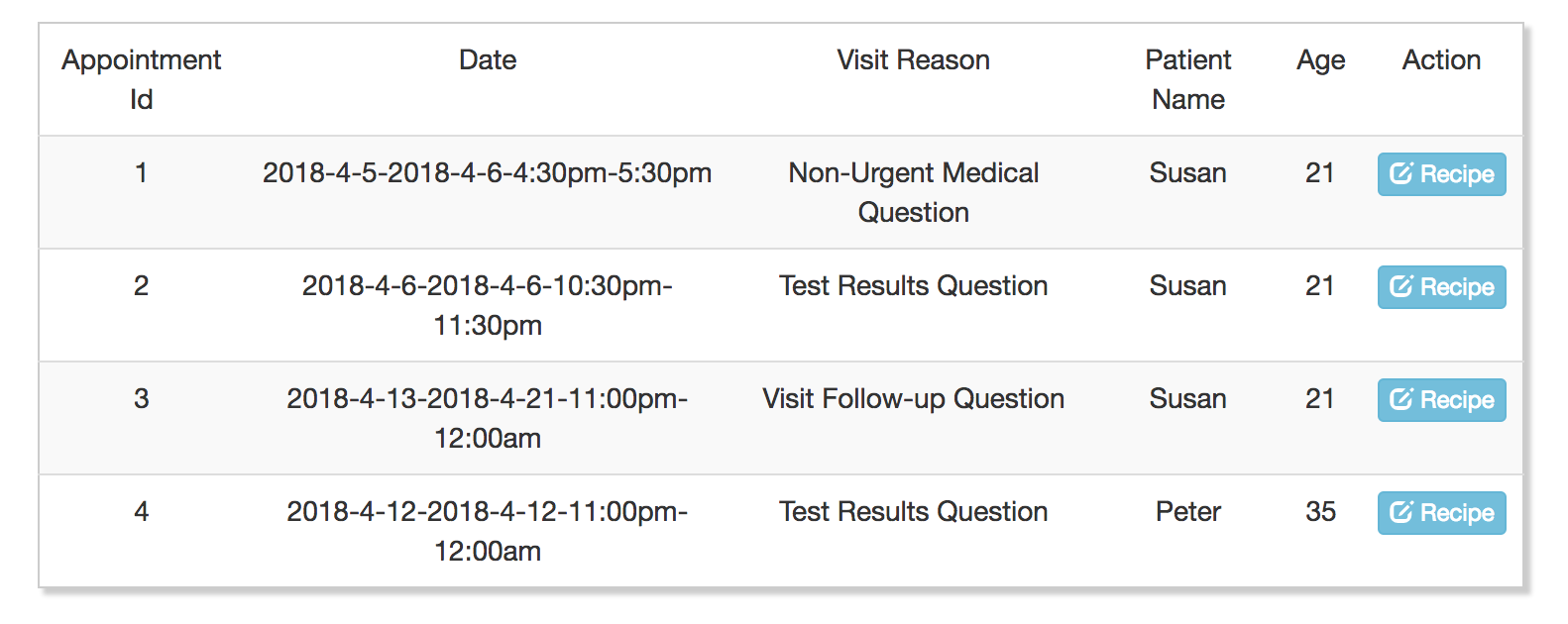
1. Manage Appointments

There is a remind symbol on nav bar which shoes how many appointments you haven’t processed, and the right part shows all the appointments, you can confirm or cancel or edit the appointment time before this appointment is confirmed

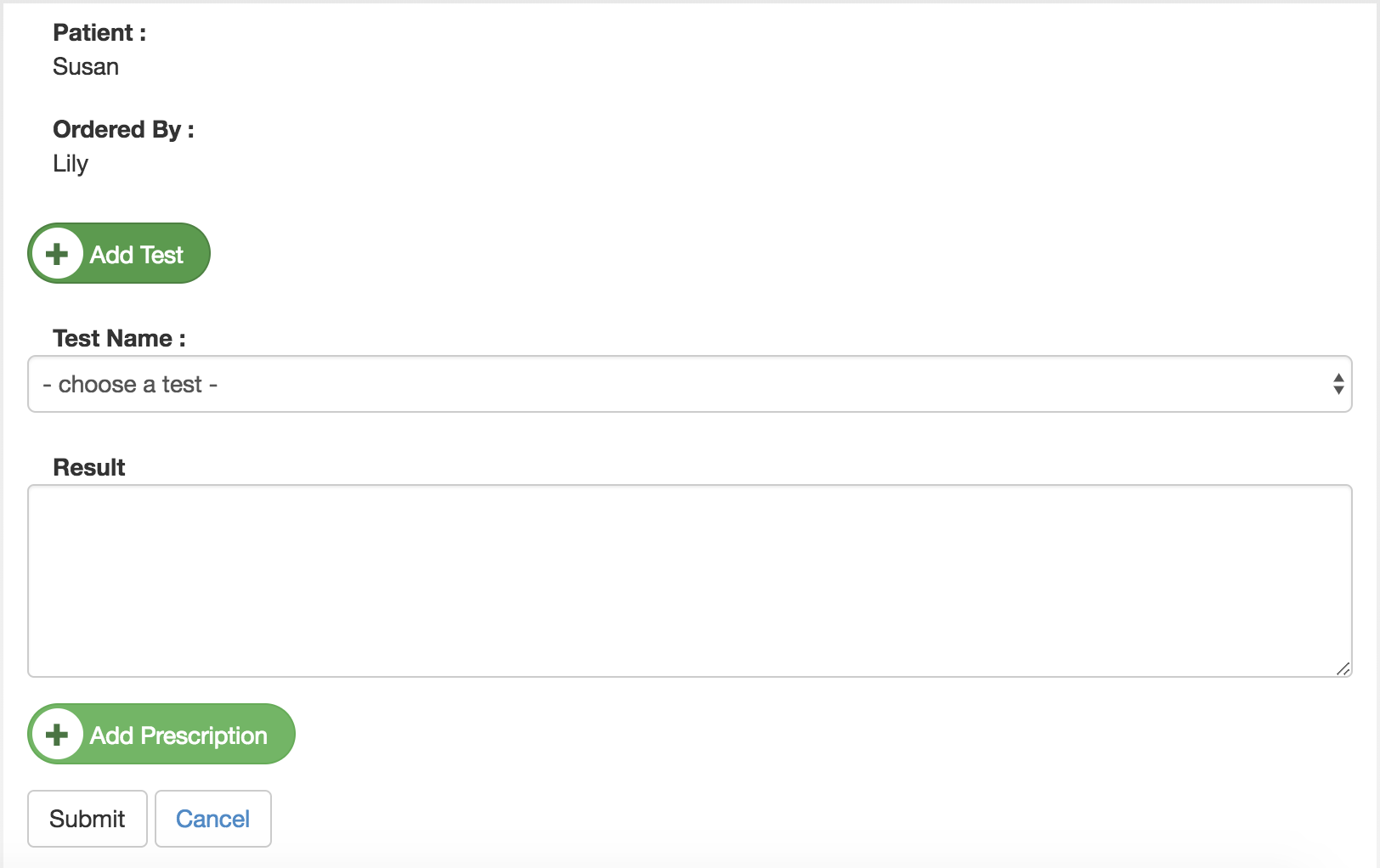


1. Write Recipes

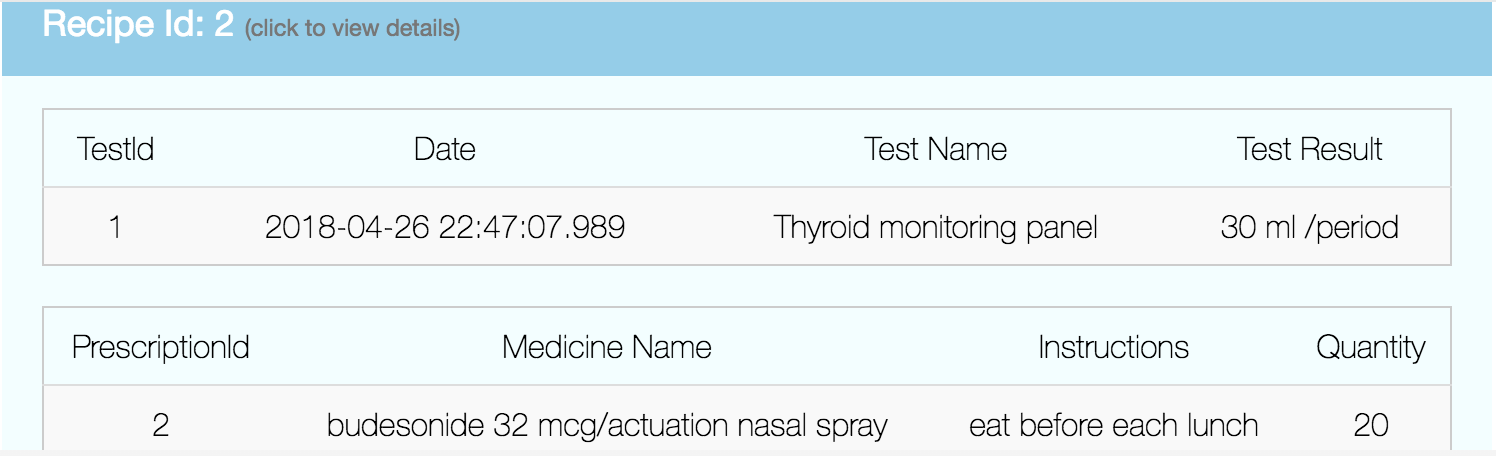
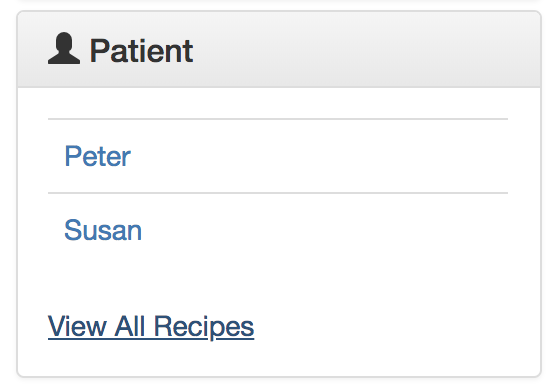
Write recipe on each confirmed appointment



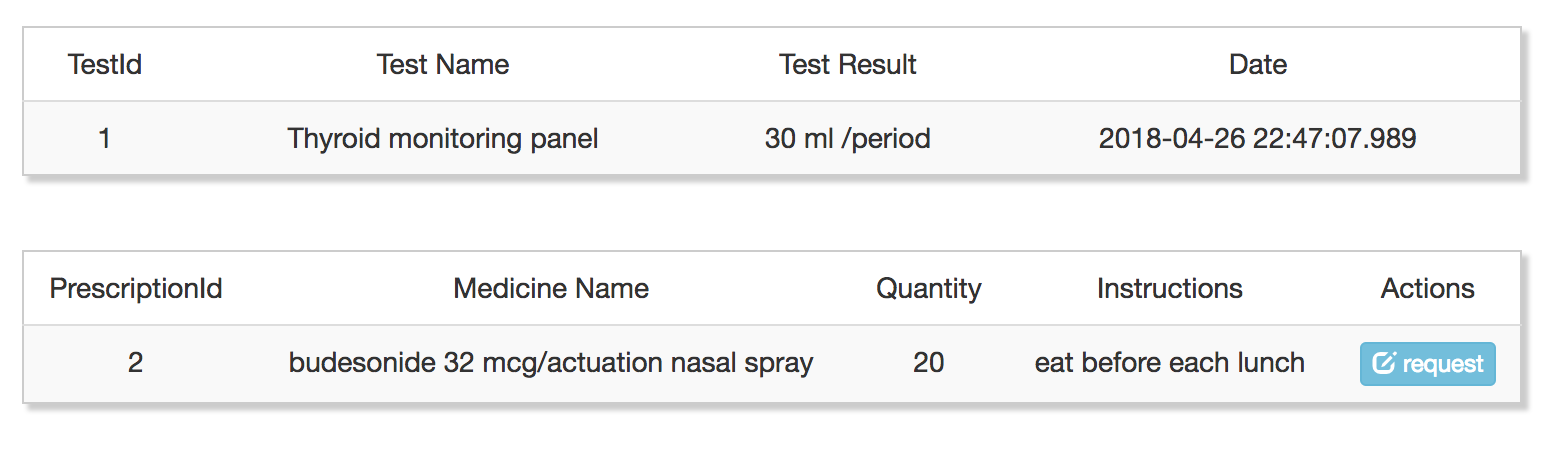
You can add many test and prescription in the recipe



1. Manage Patients and retrieve their medical records

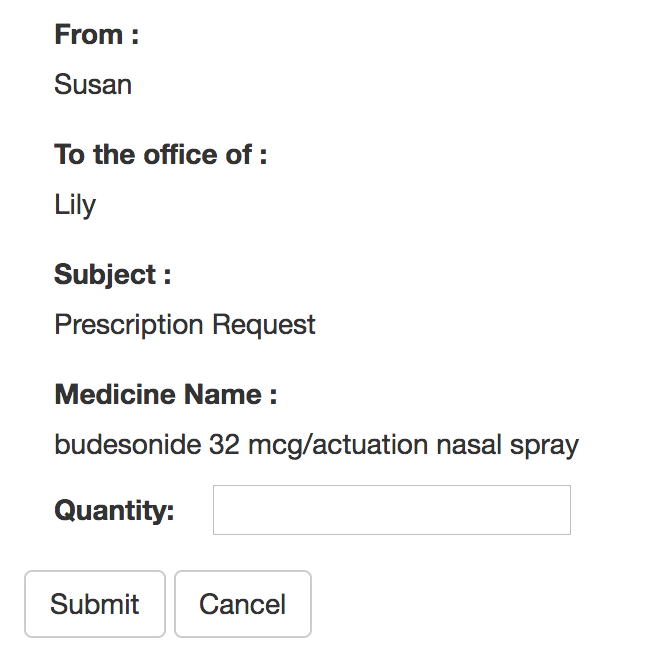


1. Manage Recipes and retrieve each recipe’s details



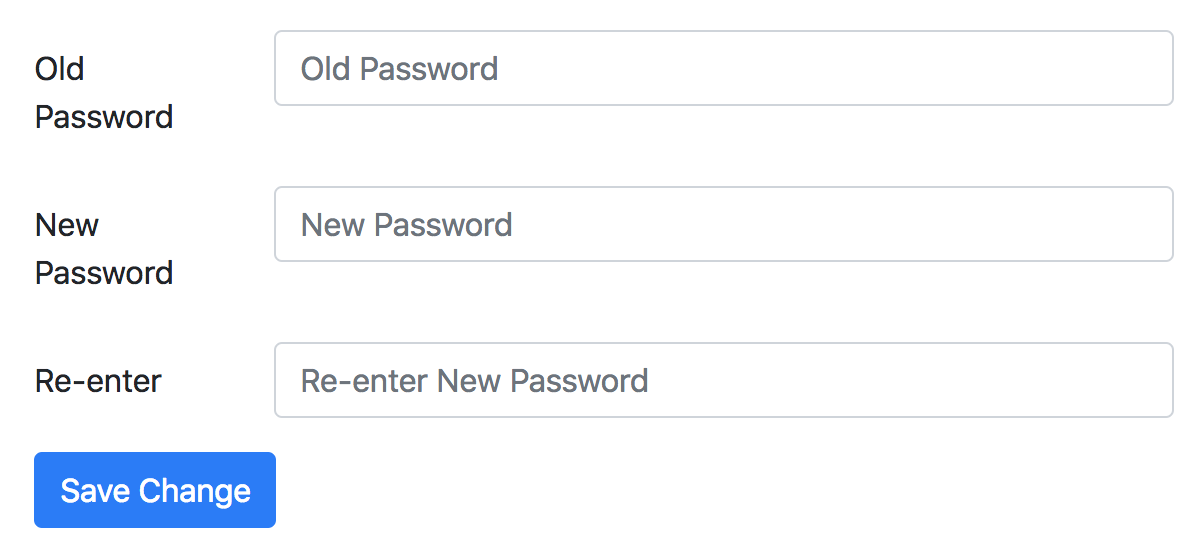
1. Request to add prescription

If you don’t have prescription and you want more, you can send a request to doctor to get more prescription

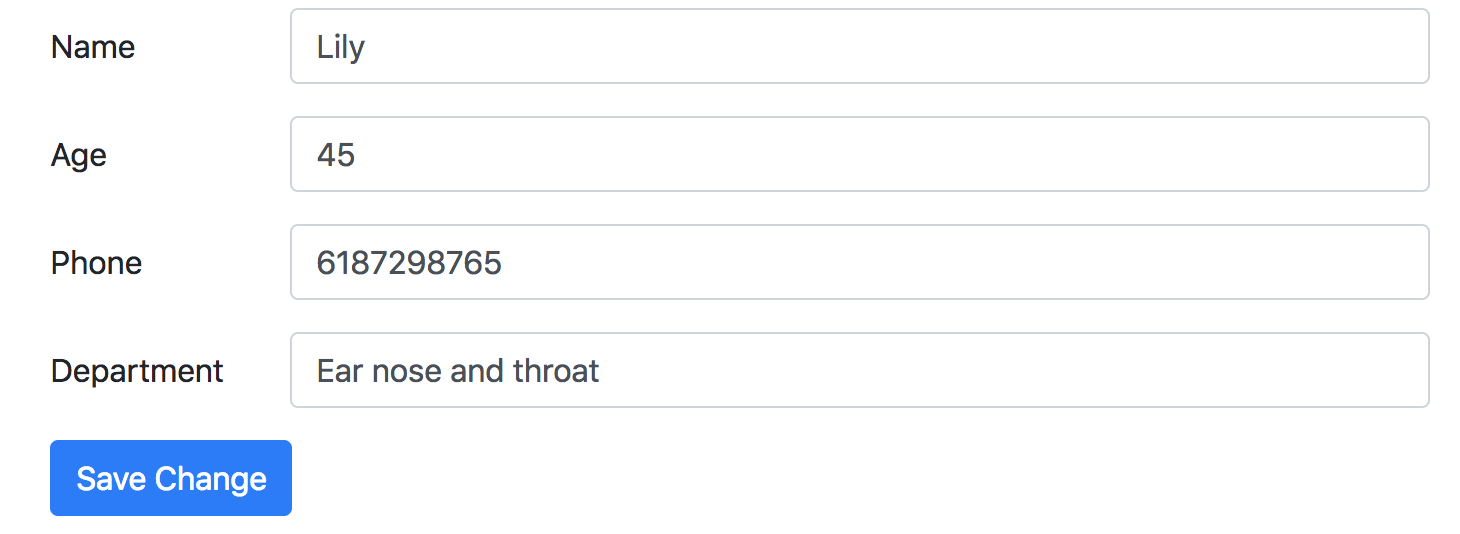
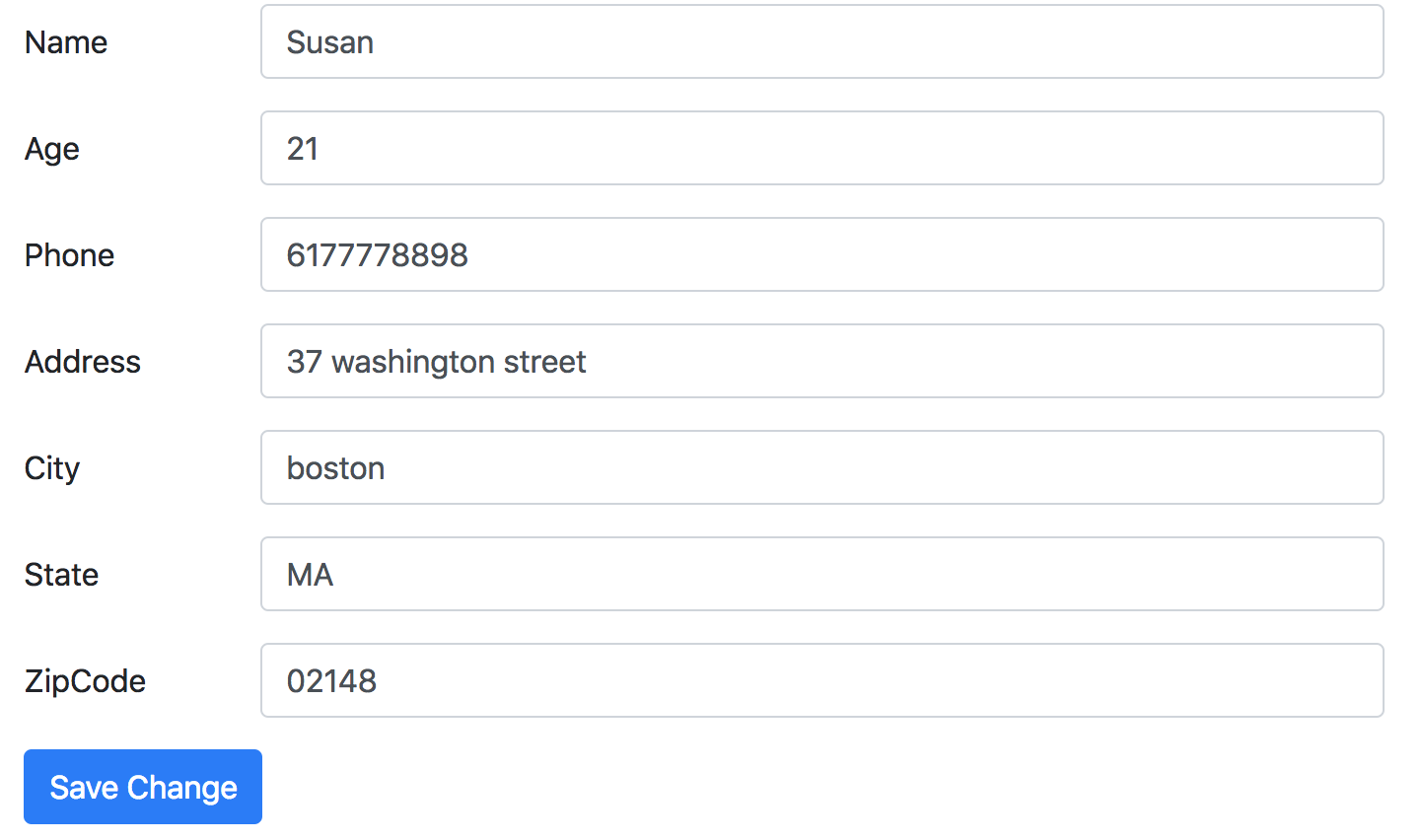


1. Change password

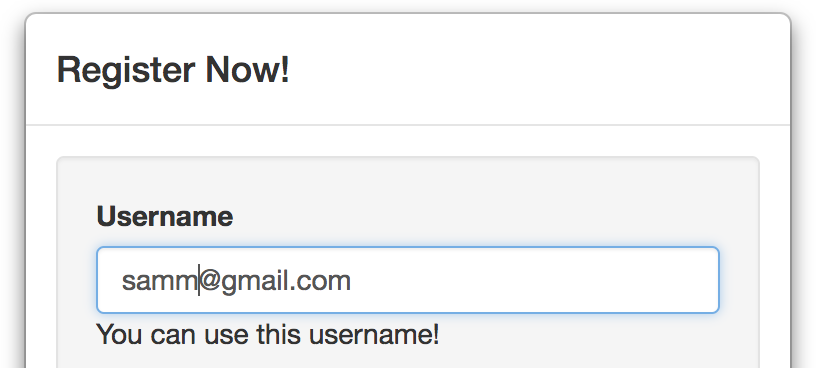
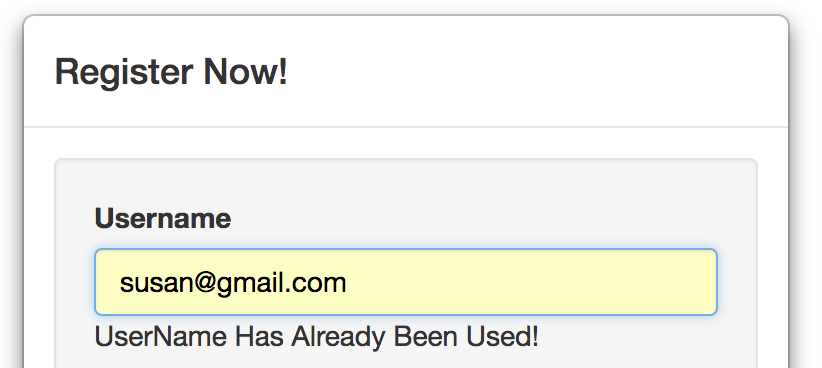
You need to input the right old password and make the new password and re-enter match



1. Update basic contact information



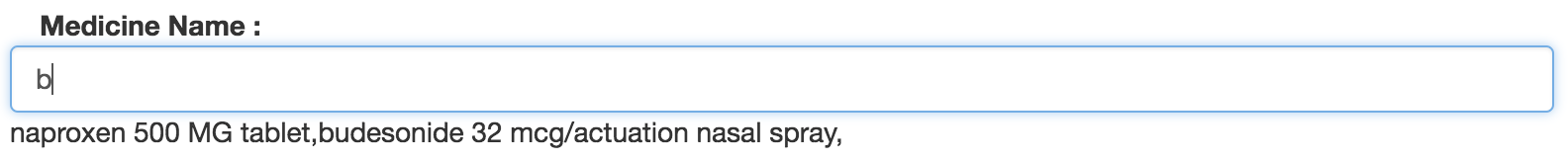
1. Technologies
2. Ajax
3. Check the usability of username when register, and give the feedback

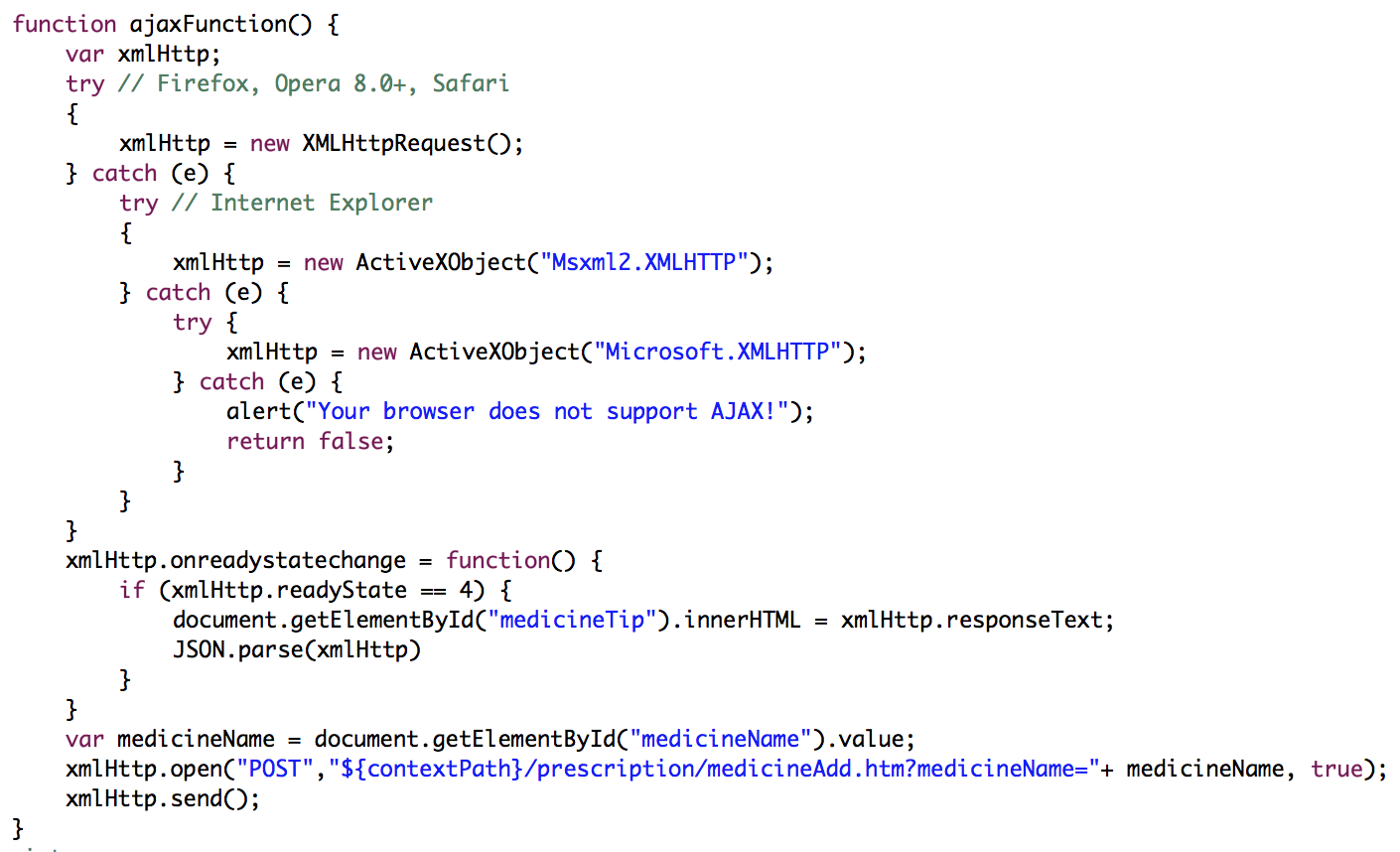


1. Update patients list of one doctor using ajax, to get the lasted recipe records of the patients and easy to look up for the doctor

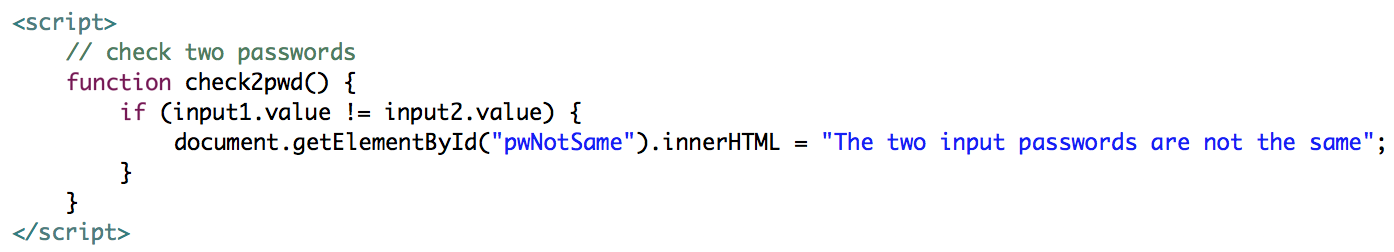
 

1. Use Ajax while the doctor inputting medicine name, there will be a hint for getting the correct medicine name in an easier and faster way

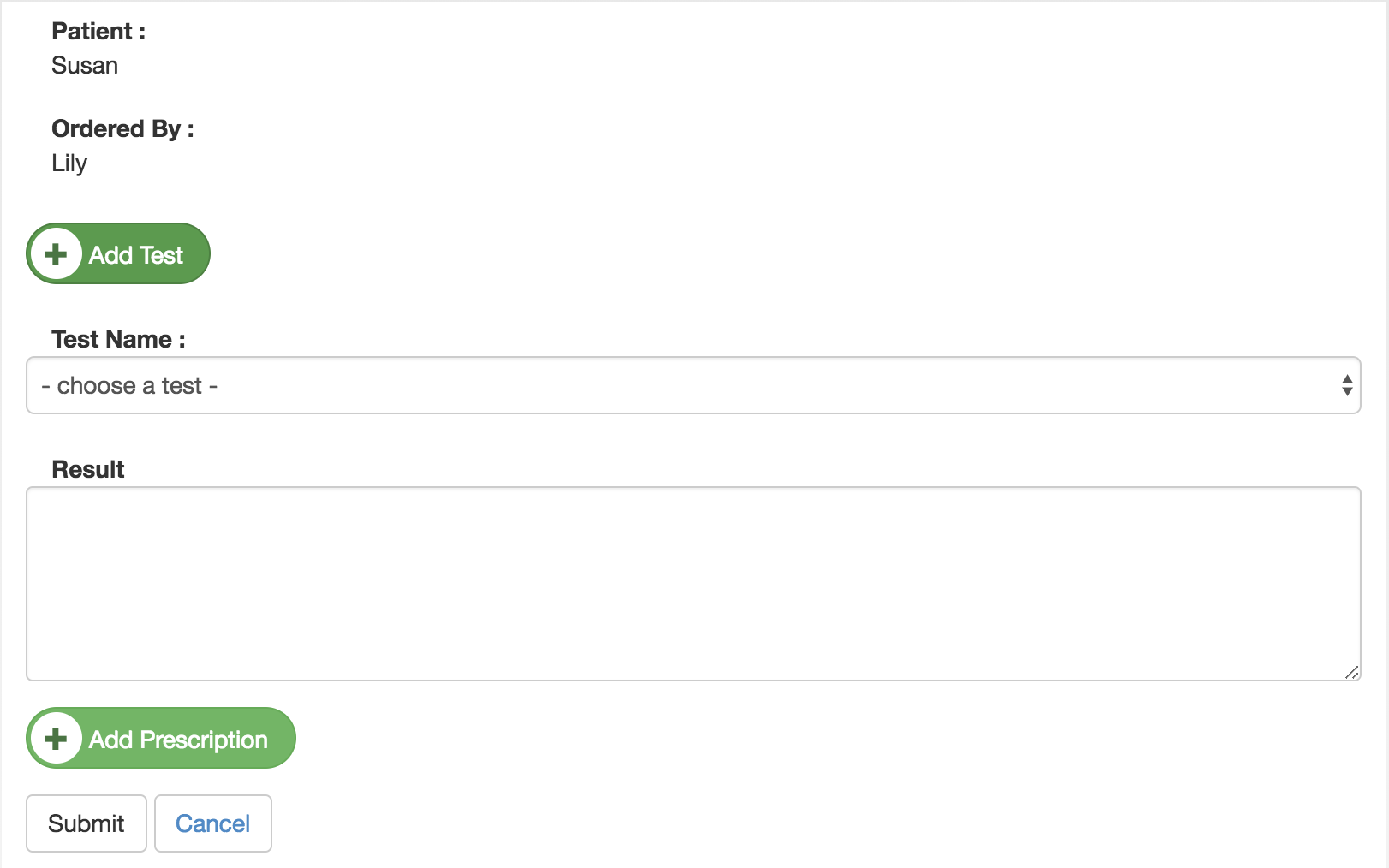


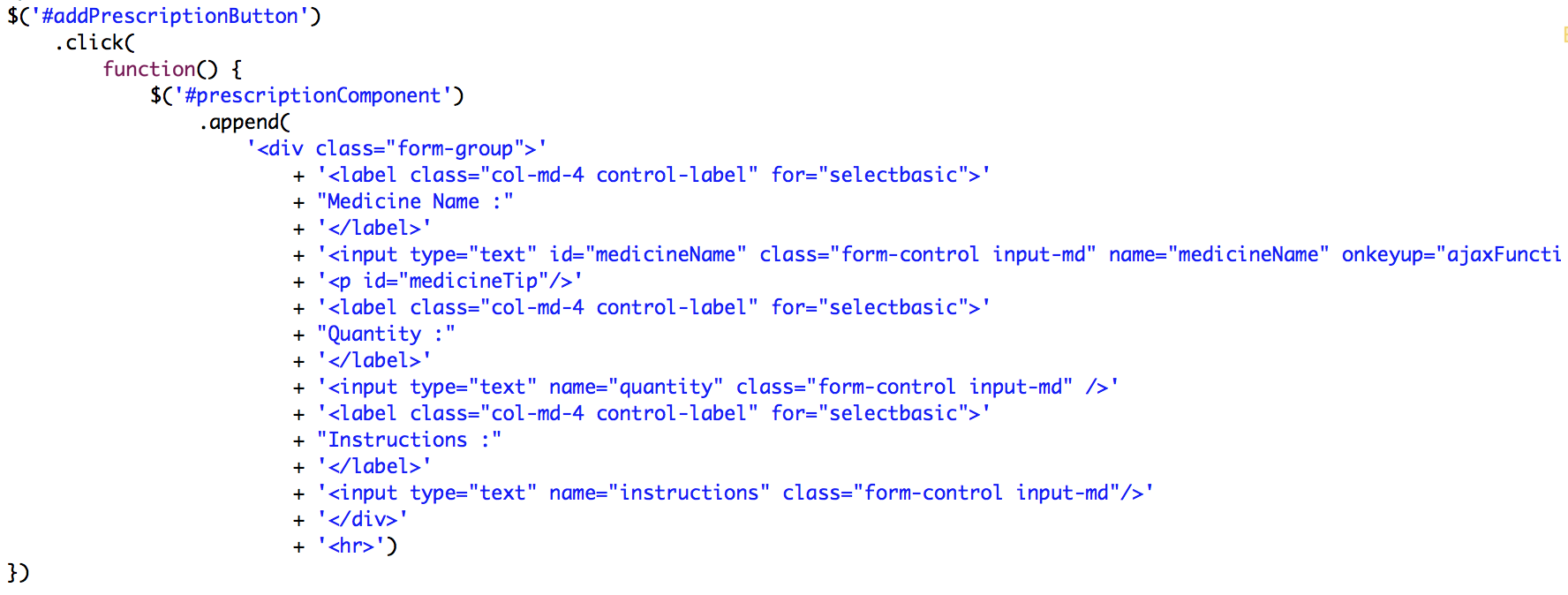


1. JavaScript & JQuery
2. In the page of change password, use javascript to check whether the re-enter new password is same as new password

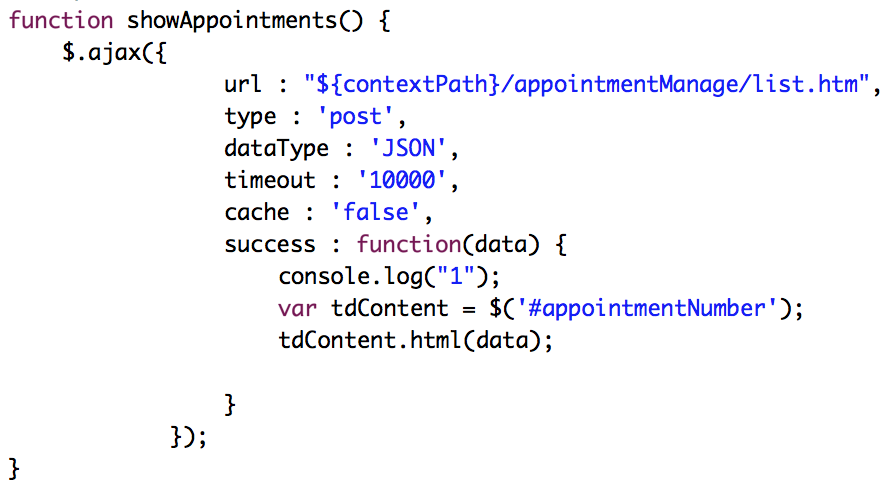
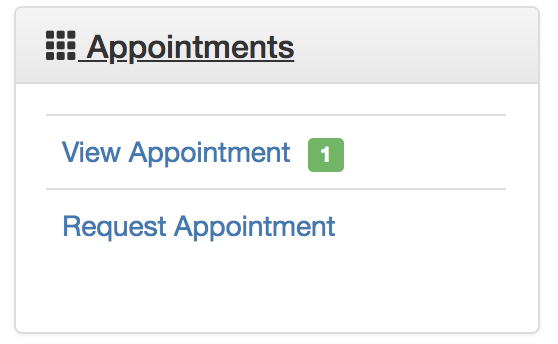


1. In the page of “recipe-edit”, use JQuery to realize that one more test / prescription part will be added after clicking the green add button





1. In the nav bar, use ajax to get the number of unprocessed appointments and show in the green square to remind user



1. Interceptor

Prevent unlogged user to get into the business logics pages, check whether if the user has been logged and saved in session, if not, redirect to login page



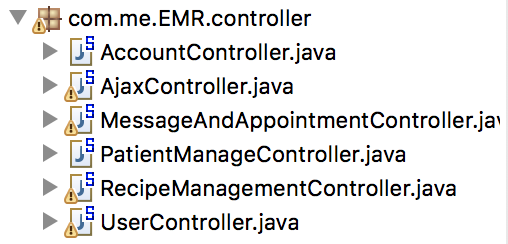


1. Filter

Use filter to delete the characters which you want to discard, try to protect the security of data in database in case of Injection Problem



1. Roles Defination
2. Patient
3. Send or reply message
4. View or delete message
5. Request appointment
6. View, confirm or cancel appointment
7. Change appointment date and time
8. Retrieve recipe records and details
9. Ask for more prescription
10. Change password
11. Update basic contact information
12. Doctor
13. Send or reply message
14. View or delete message
15. Request appointment
16. View, confirm or cancel appointment
17. Change appointment date and time
18. Manage Patients who has confirmed appointments
19. Write recipes (tests and prescriptions) on each appointment
20. Change password
21. Update basic contact information
22. Controller Source Codes



1. User Controller

@Controller

**public** **class** UserController {

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

/\*\*

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

\*/

@RequestMapping(value = "/\*", method = RequestMethod.***GET***)

**public** String showLogin() {

**return** "user-login";

}

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

**public** String showLoginForm() {

**return** "user-login";

}

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

**public** String reLoginForm() {

**return** "user-login";

}

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

**public** String handleLoginForm(HttpServletRequest request, HttpServletResponse response, UserDAO userDao,

ModelMap map) {

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

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

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

**try** {

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

**if** (u != **null** && u.getStatus() == 1) {

HttpSession session = request.getSession();

session.setAttribute("user", u);

session.setMaxInactiveInterval(120 \* 60);

**try** {

**if** (remember.equals("yes")) {

Cookie usernameCookie = **new** Cookie("userName", username);

Cookie passwordCookie = **new** Cookie("password", password);

usernameCookie.setMaxAge(365 \* 24 \* 60 \* 60);

passwordCookie.setMaxAge(365 \* 24 \* 60 \* 60);

response.addCookie(usernameCookie);

response.addCookie(passwordCookie);

}

}**catch**(Exception e) {

}

**if** (u **instanceof** Patient) {

**return** "redirect:/patient/dashboard.htm";

} **else** **if** (u **instanceof** Doctor) {

**return** "redirect:/doctor/dashboard.htm";

}

} **else** **if** (u != **null** && u.getStatus() == 0) {

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

**return** "error";

} **else** {

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

**return** "error";

}

} **catch** (Exception e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**return** **null**;

}

@RequestMapping(value = "/patient/dashboard.htm", method = RequestMethod.***GET***)

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

HttpSession session = request.getSession();

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

map.addAttribute("user", u);

**return** "patient-dashboard";

}

@RequestMapping(value = "/doctor/dashboard.htm", method = RequestMethod.***GET***)

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

HttpSession session = request.getSession();

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

map.addAttribute("user", u);

**return** "doctor-dashboard";

}

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

**public** String showCreateForm() {

**return** "user-register-form";

}

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

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

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

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

**if** (captcha.validate(captchaCode)) {

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

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

HttpSession session = request.getSession();

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

User user = **new** User();

**if** (role.equals("patient")) {

user = **new** Patient();

} **else** **if** (role.equals("doctor")) {

user = **new** Doctor();

}

user.setUsername(username);

user.setPassword(password);

user.setStatus(0);

**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/EMR/user/validateemail.htm?username=" + username + "&key1="

+ randomNum1 + "&key2=" + randomNum2;

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** "user-create-form";

}

**return** "user-created";

}

**public** **void** sendEmail(String useremail, String message) {

**try** {

Email email = **new** SimpleEmail();

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

email.setSmtpPort(465);

email.setAuthenticator(**new** DefaultAuthenticator("contactapplication2018@gmail.com", "springmvc"));

email.setSSLOnConnect(**true**);

email.setFrom("no-reply@msis.neu.edu"); // This user email does not

// exist

email.setSubject("Web tools lab ");

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

email.addTo(useremail);

email.send();

} **catch** (EmailException e) {

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

}

}

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

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

// The user will be sent the following link when the use registers

// This is the format of the email

// http://hostname:8080/lab10/user/validateemail.htm?email=useremail&key1=<random\_number>&key2=<body

// of the email that when user registers>

HttpSession session = request.getSession();

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

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

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

System.***out***.println(session.getAttribute("key1"));

System.***out***.println(session.getAttribute("key2"));

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

**try** {

System.***out***.println("HI\_\_\_\_\_\_\_\_");

**boolean** updateStatus = userDao.updateUser(username);

**if** (updateStatus) {

**return** "user-login";

} **else** {

**return** "error";

}

} **catch** (Exception e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

} **else** {

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

map.addAttribute("resendLink", **true**);

**return** "error";

}

**return** "user-login";

}

@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");

**if** (captcha.validate(captchaCode)) {

User user = userDao.get(username);

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

**return** "forgot-password-success";

} **else** {

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

**return** "forgot-password";

}

}

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

**public** String getForgotPasswordForm(HttpServletRequest request) {

**return** "forgot-password";

}

}

1. Message and Appointment Controller

@Controller

**public** **class** MessageAndAppointmentController {

@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")) {

**int** receiverid = currentUser.getAccountid();

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

map.addAttribute("messageList", messageList);

**return** "messages-inbox";

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

map.addAttribute("sender", currentUser);

**if**(currentUser **instanceof** Doctor) {

List<Patient> patientList = userDao.getAllPatients();

map.addAttribute("patientList", patientList);

map.addAttribute("role","doctor");

}**else** **if**(currentUser **instanceof** Patient) {

List<Doctor> doctorList = userDao.getAllDoctors();

map.addAttribute("role","patient");

map.addAttribute("doctorList", doctorList);

}

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

**if** (reply == **null**) {

**return** "messages-compose";

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

map.addAttribute("replySender", currentUser);

**int** receiverid = Integer.*parseInt*(request.getParameter("replyReceiver"));

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

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

User receiver = userDao.getById(receiverid);

map.addAttribute("requestContent", requestContent);

map.addAttribute("replyReceiver", receiver);

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

map.addAttribute("subject", subject);

**return** "messages-compose";

}

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

**return** "messages-inbox";

} **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 {

HttpSession session = request.getSession();

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

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

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

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);

**if**(currentUser **instanceof** Doctor) {

**int** replyReceiverid = Integer.*parseInt*(request.getParameter("replyReceiver"));

Patient receiver = userDao.getByPatientId(replyReceiverid);

message.setReceiver(receiver);

}**else** **if**(currentUser **instanceof** Patient) {

Doctor receiver = userDao.getByDoctorName(receivername);

message.setReceiver(receiver);

}

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

message.setSender(sender);

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/chenxinyun/Google Drive/NEU/6250/final-project-6250";

// 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 !";

map.addAttribute("content", text);

**return** "success";

}

}

**return** **null**;

}

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

**public** String requestMedicine( HttpServletRequest request,

UserDAO userDao, MessageDAO messageDao, ModelMap map) **throws** Exception {

**int** receiverid = Integer.*parseInt*(request.getParameter("receiver"));

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

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

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

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

Message message = **new** Message();

HttpSession session = request.getSession();

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

message.setSender(sender);

User receiver = userDao.getById(receiverid);

message.setReceiver(receiver);

message.setSubject(subject);

message.setContent("I want more " + medicineName + " with the amount of : " + quantity);

messageDao.create(message);

**if**(requestFlag.equals("yes")) {

String text = "You have successfully requested for " + medicineName;

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/chenxinyun/Google Drive/NEU/6250/final-project-6250";

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***);

}

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

**public** String showAppointmentPage(HttpServletRequest request, UserDAO userDao, AppointmentDAO appointmentDao,

ModelMap map) **throws** Exception {

HttpSession session = request.getSession();

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

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

**if** (action.equals("view")) {

**int** receiverid = currentUser.getAccountid();

List<Appointment> appointmentList = appointmentDao.getByUserId(receiverid);

map.addAttribute("appointmentList", appointmentList);

**return** "appointments-view";

} **else** **if** (action.equals("request")) {

**if**(currentUser **instanceof** Doctor) {

List<Patient> patientList = userDao.getAllPatients();

map.addAttribute("patientList", patientList);

map.addAttribute("role","doctor");

}**else** **if**(currentUser **instanceof** Patient) {

List<Doctor> doctorList = userDao.getAllDoctors();

map.addAttribute("role","patient");

map.addAttribute("doctorList", doctorList);

}

map.addAttribute("sender", currentUser);

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

**if** (edit == **null**) {

**return** "appointments-request";

} **else** **if** (edit.equals("yes")) {

map.addAttribute("replySender", currentUser);

**int** receiverid = Integer.*parseInt*(request.getParameter("replyReceiver"));

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

User receiver = userDao.getById(receiverid);

map.addAttribute("replyReceiver", receiver);

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

map.addAttribute("visitReason", visitReason);

**return** "appointments-request";

}

}**else** **if**(action.equals("confirmed")) {

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

Appointment appointment = appointmentDao.getById(appointmentid);

String confirmedTimes = request.getParameter("startDate") + "-" +request.getParameter("endDate")+ "-" +request.getParameter("startTime")+ "-" +request.getParameter("endTime");

appointment.setConfirmedTimes(confirmedTimes);

appointmentDao.confirmTime(appointmentid, confirmedTimes, "confirmed");

**return** "redirect:/appointments.htm?action=view";

}**if**(action.equals("cancel")) {

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

appointmentDao.cancel(appointmentid);

**return** "redirect:/appointments.htm?action=view";

}

**return** **null**;

}

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

**public** String handleAppointmentsRequests(HttpServletRequest request, UserDAO userDao, AppointmentDAO appointmentDao,

ModelMap map) **throws** Exception {

HttpSession session = request.getSession();

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

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

**if** (action.equals("request")) {

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

System.***out***.println("receivername: "+receivername);

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

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

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

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

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

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

Appointment appointment = **new** Appointment();

**if**(currentUser **instanceof** Patient) {

appointment.setStatus("send to doctor");

Doctor receiver = userDao.getByDoctorName(receivername);

appointment.setReceiver(receiver);

}**else** **if**(currentUser **instanceof** Doctor) {

appointment.setStatus("send to patient");

Patient receiver = userDao.getByName(receivername);

System.***out***.println("receiver: "+receiver);

appointment.setReceiver(receiver);

}

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

appointment.setSender(sender);

appointment.setVisitReason(visitReason);

appointment.setPreferredDateBegin(startDate);

appointment.setPreferredDateEnd(endDate);

appointment.setPreferredTimeBegin(startTime);

appointment.setPreferredTimeEnd(endTime);

appointmentDao.create(appointment);

**if** (editFlag == **null**) {

String text = "You have successfully requested the appointment !";

map.addAttribute("content", text);

**return** "success";

//return "redirect:/appointments.htm?action=view";

} **else** **if** (editFlag.equals("finished")) {

String text = "You have successfully adjusted the time of appointment !";

map.addAttribute("content", text);

**return** "success";

}

}

**return** **null**;

}

}

1. Patient Manage Controller

@Controller

**public** **class** PatientManageController {

@RequestMapping(value = "/patientManage/detail.htm", method = RequestMethod.***GET***)

**public** String handlePatientList(HttpServletRequest request, RecipeDAO recipeDao, AppointmentDAO appointmentDao, MessageDAO messageDao,ModelMap map,

UserDAO userDao) {

HttpSession session = request.getSession();

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

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

**if** (action.equals("view")) {

**int** receiverid = currentUser.getAccountid();

List<Appointment> appointmentList = appointmentDao.getByUserId(receiverid);

map.addAttribute("appointmentList", appointmentList);

map.addAttribute("doctorName", currentUser.getName());

**return** "allPatients-list";

} **else** **if** (action.equals("recipe")) {

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

Appointment appointment = appointmentDao.getById(appointmentid);

**if** (appointment.getSender() **instanceof** Doctor) {

map.addAttribute("patientName", ((Patient) appointment.getReceiver()).getName());

} **else** {

map.addAttribute("patientName", ((Patient) appointment.getSender()).getName());

}

map.addAttribute("appointmentid", appointmentid);

map.addAttribute("doctorName", currentUser.getName());

**return** "recipe-edit";

} **else** **if** (action.equals("history")) {

**int** patientid = Integer.*parseInt*(request.getParameter("patientid"));

List<Recipe> recipeList = recipeDao.getByPatientId(patientid);

map.addAttribute("recipeList", recipeList);

**return** "recipe-history";

} **else** **if** (action.equals("addPrescription")) {

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

Message message = messageDao.getById(messageid);

**if** (message.getSender() **instanceof** Doctor) {

map.addAttribute("patientName", ((Patient) message.getReceiver()).getName());

} **else** {

map.addAttribute("patientName", ((Patient) message.getSender()).getName());

}

map.addAttribute("doctorName", currentUser.getName());

**return** "recipe-edit";

}

**return** **null**;

}

@RequestMapping(value = "/patientManage/detail.htm", method = RequestMethod.***POST***)

**public** String handlePatientOperation(HttpServletRequest request, AppointmentDAO appointmentDao, ModelMap map,

UserDAO userdao, RecipeDAO recipeDao) **throws** Exception {

HttpSession session = request.getSession();

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

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

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

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

Patient patient = userdao.getByName(patientName);

**if** (action.equals("recipe")) {

String[] medicineNameList = request.getParameterValues("medicineName");

String[] quantityList = request.getParameterValues("quantity");

String[] instructionList = request.getParameterValues("instructions");

String[] testNameList = request.getParameterValues("testName");

String[] testResultList = request.getParameterValues("testResult");

List<Prescription> prescriptionList = **new** ArrayList<Prescription>();

Recipe recipe = **new** Recipe();

recipe.setDoctor(currentUser);

recipe.setPatient(patient);

recipeDao.create(recipe);

**if**(testNameList != **null**) {

**for** (**int** i = 0; i < testNameList.length; i++) {

Test test = **new** Test();

test.setTestName(testNameList[i]);

test.setTestResult(testResultList[i]);

test.setDate(**new** Date());

test.setDoctor(currentUser);

test.setRecipe(recipe);

recipeDao.createTest(test);

}

}

**if**(medicineNameList != **null**) {

**for** (**int** i = 0; i < medicineNameList.length; i++) {

Prescription prescription = **new** Prescription();

prescription.setMedicineName(medicineNameList[i]);

prescription.setQuantity(quantityList[i]);

prescription.setInstructions(instructionList[i]);

prescription.setDoctor(currentUser);

prescription.setDate(**new** Date());

**if** (prescription.getRecipe() == **null**) {

prescription.setRecipe(recipe);

}

recipeDao.createPrescription(prescription);

System.***out***.println("prescription created successfully!" +prescription);

prescriptionList.add(prescription);

}

}

recipe.setPrescriptionList(prescriptionList);

recipeDao.update(recipe);

**if**(appointmentid != **null**) {

appointmentDao.finishRecipe(appointmentid, "finishRecipe");

}

String text = "You have successfully finished " + patient.getName() + "'s recipe";

map.addAttribute("content", text);

**return** "success";

}

**return** **null**;

}

}

1. Recipe Manage Controller

@Controller

**public** **class** RecipeManagementController {

@RequestMapping(value = "/patientView/detail.htm", method = RequestMethod.***GET***)

**public** String handlePatientList(HttpServletRequest request, RecipeDAO recipeDao, ModelMap map,

UserDAO userDao) {

HttpSession session = request.getSession();

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

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

**if**(action.equals("view")) {

**int** recipeid = Integer.*parseInt*(request.getParameter("recipeid"));

System.***out***.println("recipe is : "+recipeid );

Recipe recipe = recipeDao.getByRecipeId(recipeid);

List<Test> testList = recipe.getTestList();

List<Prescription> prescriptionList = recipe.getPrescriptionList();

map.addAttribute("testList",testList);

map.addAttribute("prescriptionList",prescriptionList);

**return** "patient-history-view";

}**else** **if**(action.equals("request")) {

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

**int** doctorid = Integer.*parseInt*(request.getParameter("doctor"));

Doctor doctor = (Doctor)userDao.getById(doctorid);

map.addAttribute("sender", currentUser.getName());

map.addAttribute("doctor", doctor);

map.addAttribute("medicineName", medicineName);

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

**return** "messages-compose";

}

**return** **null**;

}

}

1. Account Controller

@Controller

**public** **class** AccountController {

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

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

HttpSession session = request.getSession();

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

currentUser = userDao.getById(currentUser.getAccountid());

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

**if**(action.equals("password")) {

**return** "account-password";

}**else** **if**(action.equals("contact")) {

**if**(currentUser **instanceof** Patient) {

map.addAttribute("role", "patient");

map.addAttribute("currentUser", currentUser);

}

**else** **if**(currentUser **instanceof** Doctor) {

map.addAttribute("role", "doctor");

map.addAttribute("currentUser", currentUser);

}

**return** "account-contactInfo";

}

// prepare for showing different info in jsp page

**return** "account-contactInfo";

}

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

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

HttpSession session = request.getSession();

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

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

**int** accountid = currentUser.getAccountid();

**if**(action.equals("modify-basic")) {

// modify basic info

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

**int** age = Integer.*parseInt*(request.getParameter("age"));

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

**if**(currentUser **instanceof** Patient) {

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

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

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

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

userDao.updatePatient(accountid+"", name, age, phone, streetAddress, city, state, zipCode);

}**else** **if**(currentUser **instanceof** Doctor) {

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

userDao.updateDoctor(accountid, name, age, phone, department);

}

// tell jsp page to display basic info changing successfully.

**return** "redirect:/accountInfo/detail.htm?action=contact";

} **else** **if**(action.equals("modify-password")) {

// modify password

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

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

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

userDao.updateUserPw(accountid, newPw);

String text = "You have successfully edited the password";

map.addAttribute("content", text);

}

**else** {

String text = "Your Old Password is not correct!";

map.addAttribute("content", text);

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

}

}

**return** "success";

}

}

1. Ajax Controller

@Controller

**public** **class** AjaxController {

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

@ResponseBody

**public** String checkUserNameExisted(HttpServletRequest request, UserDAO userDao) {

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

User existUser = userDao.get(username);

**if** (existUser == **null**) {

**return** "You can use this username!";

} **else** {

**return** "UserName Has Already Been Used!";

}

}

@RequestMapping(value = "/patientManage/list.htm", method = RequestMethod.***POST***, produces = "application/json")

@ResponseBody

**public** String handlePatientList(HttpServletRequest request, AppointmentDAO appointmentDao)

**throws** JsonGenerationException, JsonMappingException, IOException {

HttpSession session = request.getSession();

System.***out***.println("here");

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

**int** receiverid = currentUser.getAccountid();

Set<Patient> patientList = appointmentDao.getPatients(receiverid);

System.***out***.println("222: " + patientList);

List<Map<String,Object>> list = **new** ArrayList<Map<String,Object>>();

**for**(Patient p : patientList) {

**if**(p != **null**) {

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

map.put("accountid", p.getAccountid());

map.put("name", p.getName());

list.add(map);

}

}

System.***out***.println("2929: " + list);

ObjectMapper mapper = **new** ObjectMapper();

String jsonPatientList = mapper.writeValueAsString(list);

**return** jsonPatientList;

}

@RequestMapping(value = "/patientView/list.htm", method = RequestMethod.***POST***, produces = "application/json")

@ResponseBody

**public** String handlePatientViewHistory(HttpServletRequest request, RecipeDAO recipeDao)

**throws** JsonGenerationException, JsonMappingException, IOException {

HttpSession session = request.getSession();

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

**int** patientid = currentUser.getAccountid();

List<Recipe> recipeList = recipeDao.getByPatientId(patientid);

List<String> list = **new** ArrayList<String>();

**for**(Recipe r : recipeList) {

String recipeid = Integer.*toString*(r.getRecipeid());

list.add(recipeid);

}

ObjectMapper mapper = **new** ObjectMapper();

String jsonRecipeList = mapper.writeValueAsString(list);

**return** jsonRecipeList;

}

@RequestMapping(value = "/appointmentManage/list.htm", method = RequestMethod.***POST***)

@ResponseBody

**public** String handleAppointmentNumber(HttpServletRequest request, AppointmentDAO appointmentDao)

**throws** JsonGenerationException, JsonMappingException, IOException {

HttpSession session = request.getSession();

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

**int** receiverid = currentUser.getAccountid();

List<Appointment> appointmentList = appointmentDao.getByReceiverId(receiverid);

List<Appointment> notConfirmedAppointment = **new** ArrayList<Appointment>();

**for**(Appointment a : appointmentList) {

**if**(!a.getStatus().equals("confirmed") && !a.getStatus().equals("cancel")) {

notConfirmedAppointment.add(a);

}

}

String notNumber = Integer.*toString*(notConfirmedAppointment.size());

System.***out***.println("notNumber: "+notNumber);

**return** notNumber;

}

ArrayList<String> medicineList;

**public** AjaxController() {

medicineList = **new** ArrayList<String>();

medicineList.add("naproxen 500 MG tablet");

medicineList.add("budesonide 32 mcg/actuation nasal spray");

medicineList.add("flunisolide 25 mcg (0.025 %) Spry");

medicineList.add("Cloud computing");

medicineList.add("Data Science");

}

@RequestMapping(value = "/prescription/medicineAdd.htm", method = RequestMethod.***POST***)

@ResponseBody

**public** String ajaxService(HttpServletRequest request) {

String queryString = request.getParameter("medicineName");

String result = "";

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

**if** (medicineList.get(i).toLowerCase().contains(queryString.toLowerCase())) {

result += medicineList.get(i) + ",";

}

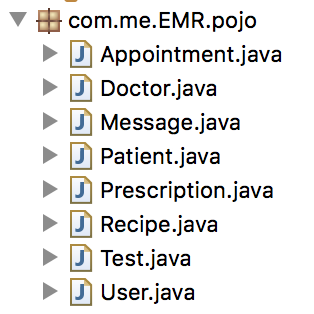
}

**return** result;

}

}

1. Pojo Source Codes



1. User

@Entity

@Table(name = "user\_table")

@Inheritance(strategy = InheritanceType.***JOINED***)

**public** **class** User {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

//@GeneratedValue(generator="assigned")

//@GenericGenerator(name = "assigned", strategy = "assigned")

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

**private** **int** accountid;

@Column(name = "username")

**private** String username;

@Column(name = "password")

**private** String password;

@Column(name = "status")

**private** **int** status;

**public** User() {

}

**public** **int** getAccountid() {

**return** accountid;

}

**public** **void** setAccountid(**int** accountid) {

**this**.accountid = accountid;

}

**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;

}

}

1. Patient

@Entity

**public** **class** Patient **extends** User{

@Column(name="name")

**private** String name;

@Column(name="age")

**private** **int** age;

@Column(name="phone")

**private** String phone;

@Column(name="streetAddress")

**private** String streetAddress;

@Column(name="city")

**private** String city;

@Column(name="state")

**private** String state;

@Column(name="zipCode")

**private** String zipCode;

@OneToMany(mappedBy="patient",fetch=FetchType.***EAGER***,cascade=CascadeType.***ALL***)

**public** List<Recipe> recipes;

**public** Patient() {

**super**();

}

**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 getPhone() {

**return** phone;

}

**public** **void** setPhone(String phone) {

**this**.phone = phone;

}

**public** String getStreetAddress() {

**return** streetAddress;

}

**public** **void** setStreetAddress(String streetAddress) {

**this**.streetAddress = streetAddress;

}

**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;

}

**public** List<Recipe> getRecipes() {

**return** recipes;

}

**public** **void** setRecipes(List<Recipe> recipes) {

**this**.recipes = recipes;

}

}

1. Doctor

@Entity

**public** **class** Doctor **extends** User{

@Column(name="name")

**private** String name;

@Column(name="age")

**private** **int** age;

@Column(name="phone")

**private** String phone;

@Column(name="department")

**private** String department;

@OneToMany

@JoinColumn(name="recipeid")

**public** List<Recipe> recipes;

**public** Doctor() {

**super**();

}

**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 getPhone() {

**return** phone;

}

**public** **void** setPhone(String phone) {

**this**.phone = phone;

}

**public** String getDepartment() {

**return** department;

}

**public** **void** setDepartment(String department) {

**this**.department = department;

}

**public** List<Recipe> getRecipes() {

**return** recipes;

}

**public** **void** setRecipes(List<Recipe> recipes) {

**this**.recipes = recipes;

}

}

1. Message

@Entity

**public** **class** Message {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

**private** **int** messageid;

@ManyToOne(cascade = CascadeType.***ALL***)

@JoinColumn(name="sender")

**private** User sender;

@ManyToOne(cascade = CascadeType.***ALL***)

@JoinColumn(name="receiver")

**private** User receiver;

@Column(name="subject")

**private** String subject;

@Column(name="content")

**private** String content;

@Column(name="attachedfile")

**private** String attachedfile;

**public** Message() {

**super**();

}

**public** **int** getMessageid() {

**return** messageid;

}

**public** **void** setMessageid(**int** messageid) {

**this**.messageid = messageid;

}

**public** User getSender() {

**return** sender;

}

**public** **void** setSender(User sender) {

**this**.sender = sender;

}

**public** User getReceiver() {

**return** receiver;

}

**public** **void** setReceiver(User 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;

}

}

1. Appointment

@Entity

**public** **class** Appointment {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

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

**private** String appointmentid;

@ManyToOne(cascade = CascadeType.***ALL***)

@JoinColumn(name="sender")

**private** User sender;

@ManyToOne(cascade = CascadeType.***ALL***)

@JoinColumn(name="receiver")

**private** User receiver;

@Column(name="visitReason")

**private** String visitReason;

@Column(name="preferredDateBegin")

**private** String preferredDateBegin;

@Column(name="preferredDateEnd")

**private** String preferredDateEnd;

@Column(name="preferredTimeBegin")

**private** String preferredTimeBegin;

@Column(name="preferredTimeEnd")

**private** String preferredTimeEnd;

@Column(name="confirmedTimes")

**private** String confirmedTimes;

@Column(name="status")

**private** String status;

**public** Appointment() {

**super**();

}

**public** String getAppointmentid() {

**return** appointmentid;

}

**public** **void** setAppointmentid(String appointmentid) {

**this**.appointmentid = appointmentid;

}

**public** User getSender() {

**return** sender;

}

**public** **void** setSender(User sender) {

**this**.sender = sender;

}

**public** User getReceiver() {

**return** receiver;

}

**public** **void** setReceiver(User receiver) {

**this**.receiver = receiver;

}

**public** String getVisitReason() {

**return** visitReason;

}

**public** **void** setVisitReason(String visitReason) {

**this**.visitReason = visitReason;

}

**public** String getPreferredDateBegin() {

**return** preferredDateBegin;

}

**public** **void** setPreferredDateBegin(String preferredDateBegin) {

**this**.preferredDateBegin = preferredDateBegin;

}

**public** String getPreferredDateEnd() {

**return** preferredDateEnd;

}

**public** **void** setPreferredDateEnd(String preferredDateEnd) {

**this**.preferredDateEnd = preferredDateEnd;

}

**public** String getPreferredTimeBegin() {

**return** preferredTimeBegin;

}

**public** **void** setPreferredTimeBegin(String preferredTimeBegin) {

**this**.preferredTimeBegin = preferredTimeBegin;

}

**public** String getPreferredTimeEnd() {

**return** preferredTimeEnd;

}

**public** **void** setPreferredTimeEnd(String preferredTimeEnd) {

**this**.preferredTimeEnd = preferredTimeEnd;

}

**public** String getConfirmedTimes() {

**return** confirmedTimes;

}

**public** **void** setConfirmedTimes(String confirmedTimes) {

**this**.confirmedTimes = confirmedTimes;

}

**public** String getStatus() {

**return** status;

}

**public** **void** setStatus(String status) {

**this**.status = status;

}

}

1. Recipe

@Entity

**public** **class** Recipe {

@Id

@GeneratedValue(strategy=GenerationType.***IDENTITY***)

@Column(name="recipeid")

**private** **int** recipeid;

@ManyToOne(fetch=FetchType.***LAZY***)

@JoinColumn(name="patient")

**private** Patient patient;

@OneToOne

@JoinColumn(name="doctor")

**private** Doctor doctor;

@OneToMany(mappedBy="recipe",fetch=FetchType.***LAZY***)

**private** List<Prescription> prescriptionList;

@OneToMany(mappedBy="recipe",fetch=FetchType.***LAZY***)

**private** List<Test> testList;

**public** Recipe() {

**super**();

}

**public** **int** getRecipeid() {

**return** recipeid;

}

**public** **void** setRecipeid(**int** recipeid) {

**this**.recipeid = recipeid;

}

**public** Patient getPatient() {

**return** patient;

}

**public** **void** setPatient(Patient patient) {

**this**.patient = patient;

}

**public** Doctor getDoctor() {

**return** doctor;

}

**public** **void** setDoctor(Doctor doctor) {

**this**.doctor = doctor;

}

**public** List<Prescription> getPrescriptionList() {

**return** prescriptionList;

}

**public** **void** setPrescriptionList(List<Prescription> prescriptionList) {

**this**.prescriptionList = prescriptionList;

}

**public** List<Test> getTestList() {

**return** testList;

}

**public** **void** setTestList(List<Test> testList) {

**this**.testList = testList;

}

}

1. Test

@Entity

**public** **class** Test {

@Id

@GeneratedValue(strategy=GenerationType.***IDENTITY***)

@Column(name="testid")

**private** String testid;

@ManyToOne(fetch=FetchType.***LAZY***)

@JoinColumn(name="recipe")

**private** Recipe recipe;

@Column(name="testName")

**private** String testName;

@Column(name="testResult")

**private** String testResult;

@OneToOne

@JoinColumn(name="approvedBy")

**private** User doctor;

@Column(name="date")

**private** Date date;

**public** Test() {

**super**();

}

**public** String getTestid() {

**return** testid;

}

**public** **void** setTestid(String testid) {

**this**.testid = testid;

}

**public** Recipe getRecipe() {

**return** recipe;

}

**public** **void** setRecipe(Recipe recipe) {

**this**.recipe = recipe;

}

**public** String getTestName() {

**return** testName;

}

**public** **void** setTestName(String testName) {

**this**.testName = testName;

}

**public** String getTestResult() {

**return** testResult;

}

**public** **void** setTestResult(String testResult) {

**this**.testResult = testResult;

}

**public** User getDoctor() {

**return** doctor;

}

**public** **void** setDoctor(User doctor) {

**this**.doctor = doctor;

}

**public** Date getDate() {

**return** date;

}

**public** **void** setDate(Date date) {

**this**.date = date;

}

}

1. Prescription

@Entity

**public** **class** Prescription {

@Id

@GeneratedValue(strategy=GenerationType.***IDENTITY***)

@Column(name="prescriptionid")

**private** String prescriptionid;

@ManyToOne(fetch=FetchType.***LAZY***)

@JoinColumn(name="recipe")

**private** Recipe recipe;

@Column(name="medicineName")

**private** String medicineName;

@Column(name="instructions")

**private** String instructions;

@OneToOne

@JoinColumn(name="approvedBy")

**private** User doctor;

@Column(name="date")

**private** Date date;

@Column(name="quantity")

**private** String quantity;

**public** Prescription() {

**super**();

}

**public** String getPrescriptionid() {

**return** prescriptionid;

}

**public** **void** setPrescriptionid(String prescriptionid) {

**this**.prescriptionid = prescriptionid;

}

**public** String getMedicineName() {

**return** medicineName;

}

**public** **void** setMedicineName(String medicineName) {

**this**.medicineName = medicineName;

}

**public** String getInstructions() {

**return** instructions;

}

**public** Recipe getRecipe() {

**return** recipe;

}

**public** **void** setRecipe(Recipe recipe) {

**this**.recipe = recipe;

}

**public** User getDoctor() {

**return** doctor;

}

**public** **void** setDoctor(User doctor) {

**this**.doctor = doctor;

}

**public** **void** setInstructions(String instructions) {

**this**.instructions = instructions;

}

**public** Date getDate() {

**return** date;

}

**public** **void** setDate(Date date) {

**this**.date = date;

}

**public** String getQuantity() {

**return** quantity;

}

**public** **void** setQuantity(String quantity) {

**this**.quantity = quantity;

}

}