**Doctor Appointment Java Application**

This application provides a registration and login for both doctors and patients. When a doctor is register, he can log in by giving username and password. The doctor can view the booking request by patients and if he accepts the patient requests the status will be shown as booking confirmed to the patient.

The application helps the patients to manage their appointments. It has dashboard so that patient can create their appointments. It also helps the doctors to manage their appointments too.

The application helps the patient to communicate with the doctor/s to create an appointment. Doctors can accept the appointment and patients can view the prescriptions. The patients must be registered and log in to book a doctor. When a patient chose a doctor then he can send a request. The request will be forwarded to admin and admin forward to doctor and if he is available he will send the confirmation request back to admin. The admin update the booking request and says confirmed to the patient. The patient can view the status in the status tab and also he will get the mail saying the booking is Confirmed.

The application has 3 modules (users):

* Admin
* Doctor
* Patient

Admin makes the connection between doctor and patient. Admin needs to login with username and password and in the admin home screen, he can see the basic functionalities of admin. Admin can view the registered doctors and patients. He can also view the patient’s request and doctors requests and he will confirm the patients and doctors requests.

The doctor need to be registered by giving the necessary details like firstname, lastname, specialization, address, email, phonenumber, gender and type of user - doctor. After registering he need to log in and in the home screen he can view the basic functionalities. He can view the patient request forwarded from admin and he can accept and he can also view the feedback given by patients. The doctor is notified via email with his username and password. The doctor will login, enters his username and password in the login page. After successful login, the doctor is directed to the doctor dashboard.

The patient needs to be registered and log in after logging on he can search for the doctor. He needs to fill the patient information (firstname, lastname, age, address, email, phonenumber, gender and type of user - patient). A patient can request an appointment with a specific doctor. Basing on the doctor availability the admin will confirm the booking request and will send to mail that the booking is confirmed he can also view in the status and he can also give feedback basing the performance of the doctor.

Doctor appointment: the doctor will login and proceeds to the dashboard where a list of appointments (sorted by date), with the status (***NEW***, ***ACCEPTED***, ***REJECTED***, ***COMPLETED***, ***CANCELED***).

The domain contains entities: *Address, Admin, Appointment, Authority, Doctor, Email, Patient, Phone number, User.*

Authority – is contained by User. The spring security module uses authority role for managing access to the pages and services.

Admin – is a subclass of Person. It is the specialization to contain administration related staff. As it has is arelationship with *Person*, the super class contains all the address and identification fiels. It creates *Doctor* instance.

User – contains the login information. It contains the *Authority.*

Patient - [keeps the patient information. Again it has is a relationship with Person. It contains all the appointments. It creates appointments.](https://www.google.com/search?client=firefox-b-d&sxsrf=ALeKk00iMLuXfHufnUIZjKzDdeKqQaW5tA:1589120676152&q=It%09keeps%09the%09patient%09information.%09Again%09it%09has%09is%09a%09relationship%09with%09Person+.%09It+contains%09all%09the%09appointments.%09It%09creates%09appointments.%09It%09can%09also%09pay%09for+appointments.&spell=1&sa=X&ved=2ahUKEwjEs6Drv6npAhVNAxAIHSMWCT8QBSgAegQIDBAq)

Doctor - keeps the doctor information. [And it has is a relationship with Person. It contains related appointments.](https://www.google.com/search?client=firefox-b-d&sxsrf=ALeKk02y0i3CSikAiuplf-5CmYdPhI8fiQ:1589120860938&q=And%09it%09has%09is%09a%09relationship%09with%09Person+.%09It+contains%09related%09appointments.%09It%09approved%09the%09invoice.&spell=1&sa=X&ved=2ahUKEwjC1q7DwKnpAhXJlosKHZuWCSMQBSgAegQIDBAq)

Appointment – is contained by both *Patient* and *Doctor.*

Person – is the generalization of *Doctor* and *Patient.* It contains identification and communication information. It contains *Address.*

I made some changes to the code. I added Person as composition over inheritance to maintain a non-hierarchical database. But inheritance is good because of the is-a relationship to the Doctor and Patient. We could do the validations based on pattern in the domain model rather than the extra validation classes.

Issues: - validation is not fully implemented.

Instructions:

1. REST:

Controller: AppointmentController

Method: findDoctorBySpecialization

Function: This method accepts a Specialization from a Path Variable (spec ID) and returns a Map containing Doctor objects found after querying the DB of doctors with the supplied specialization.

2. Validation:

Controller: AppointmentController

Method: showRegistrationAppointment ­ RequestMethod.POST

Function: Save new appointment from patient with validations declared in the Appointment.

Controller: AdminController

Method: saveDoctor

Function: Save new doctor information from admin with validations declared in the Doctor.

3. JPA – Hibernate:

Controller: AppointmentController

Method: showRegistrationAppointment ­ RequestMethod.POST

Function: Save new appointment from patient with validations declared in the Appointment domain class.

Controller: PatientController

Method: showPatientHome

Function: Retrieve and shows the patient appointments

Controller: AdminController

Method: saveDoctor

Function: Save new doctor information from admin with validations declared in the Doctor domain class.

4. Security:

Controller: LoginController , Methods: - showLogin ­ loads the login form

- loginSuccess ­> redirects to corresponding user home page by Role after successful login

- loginFailed ­> loads the login form with fail message

- logout ­> loads home page after successful logout.