Assignment 3
System Extensibility
and
Web APIs

Sachdeva, Ruchir
rs222ww@student.lnu.se
Linnaeus University, Växjö

Fatima, Benish
bf222cy@student.lnu.se
Linnaeus University, Växjö

November 4, 2018

Abstract

TThis assignment aims at designing and coding the system as per specifications in Assignment 3. We have used We will use angular 6 [2], springframework [12] and hibernate [5] to code the application keeping in mind the productivity and portability these frameworks provide by keeping the design loosely coupled making it easily maintainable and extensible. Application is built using jdk1.8 and jre1.8 on IntelliJ IDE. Database used is mysql [7] Database dump [3] is migrated as sql script using liquibase [6]. Patient test data files is also included as provided in specification. We have uses maven for building the application and spring-boot application [13] is deployed on Heroku [4].

1 Introduction

The aim of this project is to develop a web-based application with the use of different web APIs and data formats. We created 2 applications, one for frontend and other for backend . The frontend code is written in angular 6 [2] and is run on node server [8]. Choice for using angular 6 is because it uses typescript, which can be easily compiled to both javascript and nativescript. So a lot of frontend code can be reused across web and mobile application. Choice to have a separate backend application is so that we can reuse the backend code entirely for web and mobile applications. Front end Angular component makes rest calls to backend component and acordingly display views to user. Front end uses d3 [16] fro data visualisation, and AGM [1] for loading google maps. The backend component is a springboot[13] microservice, exposing different operations as REST APIs. The springboot application and angular application is deployed on Heroku. [4] The authentiction and authorisation will be handled using the abstraction provided by spring security [14], The social authentication is done in angular using oAuth [9]

2 Requirement Analysis

Below are certain key requirements of this project that should be fulfilled for the web application.

2.1 Part A

1. User has to login using third-party service providers according to their assigned roles, multiple external authentication services, at least 3 (Facebook, Twitter and Google).
2. YouTube API to used load the video list related to Parkinson's Disease.
3. Visualization of user specific data as per defined roles using different visualization tool.
4. Patients should see the exercises assigned to them.
5. Doctors should be able to see only his patient's data. Data for spiral and tapping exercises should be represented using visualization tool.
6. Researchers like doctors should see only his/her patient's data with an addition of also be able to view the patient's geographical location. Researchers can also add annotation for the session data.

2.2 Part B

As a design and documented feature only (not a part of implementation), offers mechanism for:

1. specific semantic web features.
2. smart/complex data search.

3 System workflow

Below is the system workflow which is also describes the use case diagram in 1.

3.1 Login

1. The user clicks on the login page.
2. The menu on login page displays the social media logins (Google, Facebook and Linked).
3. Login can also happen using User form with password = 'pass' for all users in system (patient1, patient2, doc, researcher).
4. Upon successfull authentication, backend server will return an encrypted JWT with authenticated user information.
5. Upon authentication, the user is redirected to the homepage where the user can have access to the features assigned to their roles.
6. for any consequent request to backend server, frontend ill have to send back the JWT token in header which has user information.
7. backend will receive JWT token in request from frontend and authenticate according to user information inside.

3.2 Patient

1. The patient views test sessions assigned to him/her.
2. The patient views you tube videos for PD exercises in widget.

3.3 Doctor

1. The doctor views test sessions assigned to his/her patients and visualize the data and see any data annotations.

3.4 Researcher

1. The Researcher views test sessions assigned to his/her patients and visualize the data and data annotations.
2. The Researcher can add data annotations to visualized data.
3. The Researcher also can see the patient's location of Google Map.

4 Backend architecture

The springboot backend application has been structured into 5 different layers.

1. Model layer 4.1
2. Controller layer 4.2
3. Service layer 4.5
4. Repository layer **??**
5. Config layer **??**

4.1 Model layer

This layer identifies all the domain entities involved in the system. Data, Medicine, Note, Organization, Role, Test, TestSession, Therapy, TherapyList, User and UserBean are the java classes which are modelled as per specifications. We will use hibernate [5] for Object relational modeling of these domain entities.

4.2 Controller layer

Since our backend application is a springboot application, all our resource handling would be done using secured REST APIs. REST APIs are secured using spring security. These resources are modelled as controllers. The users will have special roles and different REST APIs will be secured based on these roles such that a user of a specific role only can only perform the operation that the user is authorised to perform. Different user roles are Patient, Physician, Researcher. We have a Controller layer. We define a LoginController and a UserController. LoginController exposes APIs to login a user. UserController is used to expose API to return test sessions, therapies, add notes, get users for all patients of a researcher or a physician, whosoever has logged in as per their role. If no user has logged in or use has no role to see intended data, then no data will be returned. Researcher's RSS Feed is fetched from [10]

4.3 Service layer

The Rest resource controllers will call the services to perform the intended operations. Services will have a transactional scope when needed such that an operation as a whole is successfully executed or all is aborted. We have a UserService, NoteService, SecurityContextHandler And TokenHandler. UserService is used to call Repositories so as to answer questions related to user, such as what are the therapies assigned by a medical user, or what are the test sessions for patients of a medical user or a researcher. NoteService is used to add note to a test session by logged in user. SecurityContextHandler is used to authenticate a user using spring security or to fetch the current logged in user. TokenHandler is a service to manage JWT tokens, which are used for securing communication between backend and frontend.

4.4 Repository layer

The Services will need to fetch data from the database. For this purpose we have created a ¡repository layer which interacts with the database using hibernate/JPA. We use Spring-data for making database calls. NoteRepository, TestSessionRepository, TherapyRepository, UserRepository are java classes which are jpa repositories to fetch data from database for respective entity. DataRepository is a java class to load data from data files based on dataUrl.

4.5 Config layer

We use spring security for securing our application. SecurityConfig is used to secure our REST APIs and WebConfig for addinf cors info in header.

5 Front end architecture

Our front end aplication is written in angular6 and is running on node.js. It has following responsibilities:

1. Authenticate user with social and fetch the access token.
2. Authenticate user with backend by sending in user credentials from form or by sending the access token fetched form federated identity provider.
3. Call backend server with appropriate authorization header to fetch resources using REST API ina secured way
4. Use AGM [1] to load google maps of patient's location.
5. Use D3.js for data visualization.
6. Manage flow between user views.

6 Design diagrams

We have designed the architecture using 3 types of UML diagrams:

6.1 Usecase diagram

usecase.jpg 1 file is attached with the submission. It is used to represent different actors of the system and different use cases that they can perform, while establishing associations between them

6.2 Class diagram

classdiagram.jpg file is attached with the submission. This file shows different domain entities and their relationships along with the Controller classes which will expose the REST APIs consumed by the client, the service layer classes which which will ahve transactional scope and encapsulate the single unit of work to be performed related to any resource requested, posted, updated or deleted. The dao layers are the persistence layers that service interacts to. For external web services we will have a client to call them.

6.3 Sequence diagram

sequence.jpg file is attached with the submission. This file presents the entire sequence of events with respect from actors and how the application caters the flow.

7 Scenario for a web semantic feature

In our application's current scope a doctor assigns exercises/tests to the patient. We collect the patient's test responses. This patient's test data is also shown in visualisations. We can extend this funtionality to generate progress reports. So I propose **Patient's progress report** as a new semantic feature.

In this feature we will record patient's all tests' data over a period of time. We will expose a ReportService which will fetch all this data from database. We will create two datasets : *expected data*, and *actual data*. "Expected data" is the data that a healthy user would generate after performing the exercises and "Actual data" is the patient's responses that we have stored with us over a period of time.

We will process these datasets, test the normality of data sets using Shapiro-Wilk Normality test [11]. Further with the tested and processed data, we will perform T-Test [15] to see how significantly different both datasets are. We will report the result in form of progress to the user

Please refer to files semantic-feature-usecase.jpg and semantic-feature-sequence.jpg in submission zip file.

8 Smart data search

Our application already has a well defined architecture for data search. We have exposed all our model classes as Hibernate entities to facilitate Object-Relational Modelling. We have further exposed services which interact with our controllers to return the intended data. The controllers expose the REST APIs which are restricted as per different users' roles.

Now adding more flavor to this, we have very smartly exposed all crud operations related to all entities in our system. So we do not need any repository, service or controller to query our database. These REST APIs are again secured as per user role. At the moment these APIs are retricted and not in use by our client application.

We can assign a researcher's role to these APIs and then a researcher would be able to perform any search on any entity without writing a single peice of code more. All we will need to do is create views in the client application and call these APIs.
We can fetch all entities, add restrictions to request path, get sorted data as per any column, ascending or descending, get paginated data, get limited size of result set etc merely by an API call.
We can search by specifying predefined attributes like "nameStartsWith" and specifying starting character etc.

Refer below to some APIs for Note. There are may more features which our backend is ready to cater. We can perform any nestd complex query search with ease. Just making our client call these APIs and securing them in backend will make them avaliable for us end to end. Please refer to search-usecase.jpg and search-sequence.jpg in submission zip file.

```
GET https://assignment3-lnu.herokuapp.com/notes
GET https://assignment3-lnu.herokuapp.com/notes{&sort,page,size}
GET https://assignment3-lnu.herokuapp.com/notes/1
GET https://assignment3-lnu.herokuapp.com/notes/search/nameStartsWith?name=K&sort=name,desc
GET https://assignment3-lnu.herokuapp.com/notes/med-user/1
GET https://assignment3-lnu.herokuapp.com/notes/med-user/role/1
GET https://assignment3-lnu.herokuapp.com/notes/user/search/byNotes/nameStartsWith?name=K&sc
GET https://assignment3-lnu.herokuapp.com/notes/test-session/search/byNotes/nameStartsWith?n
```

9 Application urls

- **Backend server github url:** https://github.com/ruchirsachdeva/assignment3
- **Frontend client github url:** https://dashboard.heroku.com/apps/assignment3-lnu
- **Application main page:** https://lit-beach-29911.herokuapp.com/
- **Backend server:** This won't be accessible directly because of security. https://pd-social-server.herokuapp.com

References

[1] Angular google maps documentation. https://angular-maps.com/api-docs/agm-core/changelog.html.

[2] Angular 6 documentation. https://angular.io/docs.

[3] Database dump. https://mymoodle.lnu.se/pluginfile.php/3460998/mod_resource/content/1/pd.sql.

[4] Heroku deployment. https://devcenter.heroku.com/articles/deploying-spring-boot-apps-to-heroku.

[5] Hibernate. http://hibernate.org/orm/documentation/5.3/.

[6] Liquibase. https://www.liquibase.org/documentation/index.html.

[7] My sql. https://dev.mysql.com/doc/.

[8] Node.js documentation. https://nodejs.org/en/docs/.

[9] O auth 2 documentation. https://oauth.net/2/.

[10] Rss feed url. https://www.news-medical.net/tag/feed/parkinsons-disease.aspxl.

[11] Shapiro-wik normality test website. http://sdittami.altervista.org/shapirotest/ShapiroTest.html.

[12] Spring framework documentation. https://docs.spring.io/spring/docs/current/spring-framework-reference/, .

[13] Springboot documentation. https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/, .

[14] Spring security documentation. https://docs.spring.io/spring-security/site/docs/current/reference/htmlsingle/, .

[15] T-test calculator website. https://www.socscistatistics.com/tests/studentttest/Default2.aspx.

[16] D3 documentation. https://github.com/d3/d3/wiki.

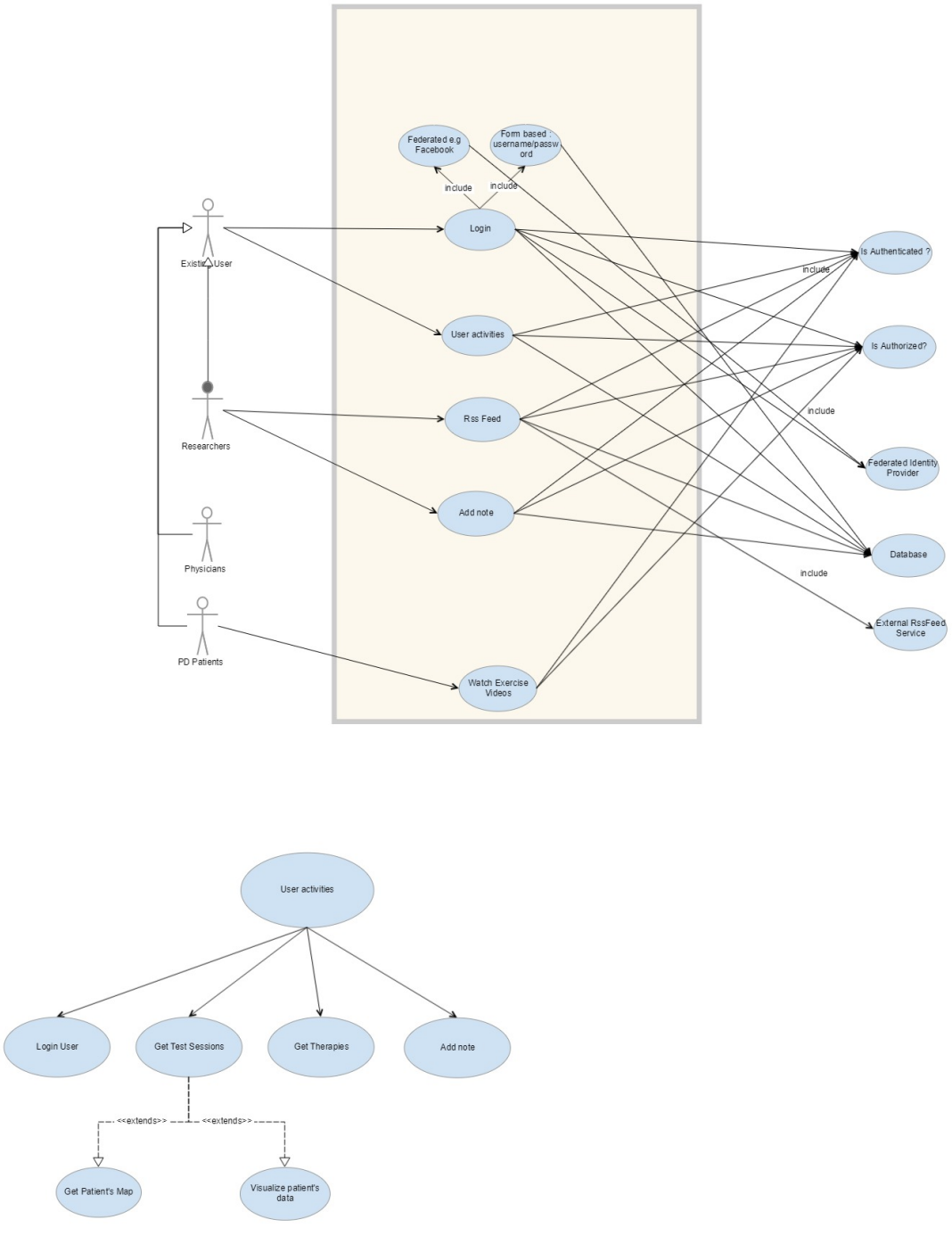

Figure 1: Use Case diagram