EA Project Description – June 2024

Goal

The goal of this project is to give you hands-on, practical experience with building a RESTful system of microservices using Spring Boot, Spring MVC, Spring Security, Spring Data JPA and Spring Cloud Services.

Course Attendance System

We are trying to create a **collection** of RESTful web services to automate the process of course attendance taking for Compro students. Imagine we have a badge scanner at every classroom and every day that you go to class, you scan you badge on your way in. The scanner should create an AttendanceRecord for each badge scan that has the student barcode number (could be alphanumeric QR code also), location ID as we all as date and time of the scan.

Built into this system, we also know which courses a student are registered for. Each course has many Course Offerings. Each "Course Offering" is an instance of a course offered at a particular time, by a particular faculty (for example CS544-2024-06-V017-PaymanSalek is the June 2024 offering of CS544 by Payman Salek in room V017). Each student can register for many course offerings (one at a time, but many over time) and each course offering has many students on the roster (class list). Each course also has a start and end date and list of "Sessions". Each Session has a date and also start time and end time. List of sessions for a course offering could be an in-memory computation or be stored in database (I will leave it to you to decide). Either way standard courses meet every morning (between the start date and end date of the course), Monday through Saturday from 10:00am – 12:30pm and every afternoon from 1:30pm – 3:30pm (you can compute this list based on the start and end date of the course). The last day of the course is always half a day (only morning session).

From the list of sessions for a course and the attendance records, you should be able to create an attendance list. Imagine a GET end point with the following format:

/course-offerings/CS544-2024-06-01/attendance

This should return a Microsoft Excel spreadsheet (you can use the Apache POI to convert data into Excel) with columns that represent course sessions (e.g. 2024-06-10-AM) and rows that represent each student in that course offering and then a 1 or zero in each column in from of student representing present or absent.

There are multiple roles in the system: student, faculty, staff, sys-admin, etc.

Step 1 – Requirements (Domain Driven Design)

Create a UML class diagram of the business domain based on the requirements given above

EA Project Description – June 2024

Use-cases

1. Students should be able to **GET /student-view/course-offerings** and receive the list of courses they are registered in and their grade in that course if the grade is already published.

2. Students should be able to **GET /student-view/course-offerings/{offeringId}/attendance** and receive the list of sessions for that course and their attendance (if the course has started or is in the past)

3. Students should be able to **GET /student-view/attendance-records** and receive all of **their** attendance records (date and time, location, location type, etc.).

4. Students should be able to **GET /student-view/course-offerings/{offeringId}** and see the course information such as who is teaching it, when it starts, classroom, etc. (minus registration list or course roster). Students should be able to do this for any course; whether they are registered for it or not.

5. Admins (staff, faculty and sys-admins) should be able to **GET /admin-view/course-offerings/{offeringId}/attendance** and download an Excel sheet of the attendance records for the course (This is the most sophisticated use-case in the whole project. It is harder than you think).

6. Admins (staff, faculty and sys-admins) should be able to **GET /admin-view/course-offerings?date=YYYY-MM-DD** and see all the courses that are in session during that date (startDate <= date and endDate >= date).

7. Admins (staff, faculty and sys-admins) should be able to **GET /admin-view/course-offerings/{offeringId}** and see all the course offering with all of its details: start date, end date, faculty who teaches it, list of registrations (class roster), etc.

8. Admins (staff, faculty and sys-admins) should be able to **GET /admin-view/students/{studentID}** and see all the details of one student as well as all the courses the student is registered for (no need to bring back the attendance records list).

9. Sys-Admins should be able to **CRUD /sys-admin/students**

10. Sys-Admins should be able to **CRUD /sys-admin/course-offerings**

11. Sys-Admins should be able to **CRUD /sys-admin/courses**

12. Sys-Admins should be able to **CRUD /sys-admin/locations**

EA Project Description – June 2024

Working as a team

You are required to work with your team to create a project. But you should be aware that everyone still must give their own presentation. There are no “team” presentations, although I do ask that members of a team present consecutively (no gaps / other people in between). You need to work together and share responsibilities and workload.

Must Have

Your project must have the following features:

- Send emails asynchronously. Use a JMS/Kafka message queue and a DB table
- Send an email reminder 8 and 4 hours prior to the end of the registration period
- Use Git for code repo
- Junit testing for all of your classes (controller, service, repository, domain entities)
- Have more than one REST service (at a minimum, your email service should be a separate service, but you may have other services as well)

Extra Credit

- Spring Security using token based authentication (OAUTH2)
- Web layer of your API using WebFlux and non-blocking IO
- Deploy your application to the cloud (AWS, Google, Azure, Cloud Foundry, …)
- Create an automated pipeline for CI/CD (continuous integration/continuous delivery)
- Use Spring Cloud Feign for your service calls

Step 2 - Architectural Analysis and Design

Sit with your team and agree on a high-level architecture for your application. Your decision should include your choice of technologies. How do you secure the application?

Step 3 – Proof of Concept (POC)

Choose one simple use-case and make sure it works end to end. For example, login for one type of user.

Step 4 – Divide and Conquer

Carefully read the requirements and divide the tasks (use-cases) among team members. Each team member is responsible for designing, developing and testing his/her use-case.

Step 5 – Integration Testing

At least once a day sit together and integrate your code and test together and iterate (correct mistakes and refactor your design and development).

Presentation Delivery

I will schedule your presentations from 9:30-12:30 and 1:30 – 4:30 on Thursday and Friday morning. You will have to present individually. Each group will have one hour, which means each member of the group gets to present for about 5 minutes and I get to ask questions for a couple of more minutes. I will evaluate you based on the following factors:

1. Clarity of Speech – Your presentation should be coherent and understandable. It is ok if you have an accent. It is ok if your English is not as fluent as a native speaker. However, it is ***not*** ok if you talk too fast! Enunciate and speak clearly.

2. Knowledge of Your Application – You are expected to be knowledgeable about the overall design of your app. All layers and all use-cases, not just yours.

3. Ability to Answer Questions – You need to be able to answer questions about the design and coding of your app. It always shows when your team members have done all the work and you have been mostly observing. Try to be an active participant and you will get full grade for this category.

4. Working Demo – Your app needs to work (obviously!). So, if you succeed to show me working features, you will get maximum grade.

5. Code quality – You need to write readable, good quality code. Hint: use “Sonar Lint” or similar plug-ins for your IDE to check the quality of your code.

6. Extra credit – you can get up to 10% of extra credit based on how well you design your application and how well you integrate it.