

# Central Connecticut State University

**University Student Portal using the *Semantic Web Approach***

CS-595 Capstone Final Report

**Team Members:**

*Nagaharika Ogirala*

*Swetaa Ganesan*

*Sherin Cheriyan*

**Advisor:**

*Dr. Neli Zlatareva*

**Spring 2021**

**Table of contents:**

**Acknowledgement**

**List of Tables and Figures**

**Abstract**

1. **Introduction**
   1. **Domain**

**1.1.1** Existing System

**1.1.2** Proposed System

1. **Knowledge Gathering**
   1. **Domain – Department of CS**

**2.1.1** Courses Datasets

**2.1.2** Faculty Datasets

* 1. **Skillset**

**2.2.1** Semantic Web

**2.2.2** Spring Boot Micro-services

**2.2.3** Angular

1. **Design and Analysis**
   1. **Technical Stack Decisions**

**3.1.1** Semantic Web Technology

**3.1.2** Java Technologies

* 1. **Semantic Web Schema Models**

**3.2.1** Course Schema

**3.2.2** Faculty Schema

**3.2.3** Integrated Schema

* 1. **Course Op Service**

**3.3.1** Jena API Service

**3.3.2** Front-end JSP Design

* 1. **Login and Registration Services**

**3.4.1** Backend Design

**3.4.2** Front-end Angular Design

**3.4.3** Additional Services

* 1. **It all fits (Integration)**

**3.5.1** Role of Microservices

**3.5.2** Microservice Integration

**3.5.3** Wiring with Spring boot

1. **Planning and Development Strategies**
   1. **Agile Development**

**4.1.1** Use Cases Diagram

**4.1.2** User Stories

**4.1.3** Sprint Planning and Implementation

* 1. **Version Control**

**4.2.1** GitHub

**4.1.2** Multiple Repositories

1. **Implementation**
   1. **Control Flow**

**5.1.1** Application Flow

**5.1.2** Sequence Models

* 1. **Services and Endpoints Used**

**5.2.1** SPARQL Endpoint

**5.2.2**

1. **Application**
   1. **User Interface**

**6.1.1** Course Op Pages

**6.1.2** Login Page

**6.1.3** Registration Page

1. **Possible Extensions**
2. **Conclusion**
3. **References**

**1.Introduction**

The university student portal is developed using the Angular User Interface and back-end is developed with Java using Spring Boot. Here the data is created and updated through the Ontology model developed using Protégé. The ontology creates a relationship between various entities and classes, it also describes how these entities can be grouped according to their similarities and differences. In our ontology model, we are creating two different ontologies: Faculty Ontology and Courses Ontology. The Faculty Ontology contains details of the faculty and staff members. The Course Ontology contains the course details, time, days and the semester, in which the courses are provided by the department for both graduate and under-graduate students. These ontologies are then combined, thus providing a relationship between the faculty and the course, where we can then obtain the details of the professors teaching the different courses.

Jena API is used to maintain the schema of the ontology using various Inference and Ontology models. This application is built in Java SE 10 using the Eclipse IDE. The Angular framework is used to build the User Interface for the data available in the Ontology models. Spring Boot works at the middle layer, wiring the Angular User Interface with the ontology models and/or database for student login.

The intent of the web application is to help students log into their University account, as well as register for courses and view their registration status. We are using two possible data-sources, one being Protégé to populate domain information(turtle files) and Database to store the login information and student registration status. Jena API is used to integrate the models in runtime.

**1.1 Domain**

**1.1.1 Existing System**

**1.1.2 Proposed System**

This application is used for course registration for students, where students can login to the portal using their University username and password. This application mainly focuses on the Computer Science Department for both under-graduate and graduate students. The user can register for the courses available for current semester and browse courses for any other semesters. They can also add and drop courses depending on their registration status. The user will be displayed with the courses, faculty who teaches the course, semester, and time when the course is offered.

A Microservice is created for integrating the services from Course Op Model, Registration and login services. This framework maintains a database for all the student login and course details, where data can be added and removed based on the changes made by the user. Spring Boot is a Java-based framework used to create microservice. We use the student login and registration data, which is maintained in the relational DB and the RDF data maintained in the Course Op services.

**2.Knowledge Gathering**

**2.1 Domain – Department of CS**

This system is designed to courses and the faculty who teaches the course. This helps the students to register to courses according to their requirements in that particular semester. In this application we are providing the registration service only for Computer Science department students. This datasets is represented in RDF triple format using Protégé editor.

**2.1.1 Course Datasets**

The Course data is collected from the CCSU websites for all the courses under CS department . This data is collected based on different Semester i.e., Spring/Summer/Fall. It contains details like when the course is provided , timings/days for different courses, semester when the course is offered, Description and its prerequisites.

**2.1.2 Faculty Datasets**

The faculty datasets contain the data of different faculty and staff from the CS department. The data is obtained from the CCSU official website. This contains the Name, email, Phone and website for all the faculty and staff. This data is combined with the courses to define who teaches what so that the students can choose courses based on the different faculty available.

**2.2. Skillset**

**2.2.1 Semantic Web**

The Semantic Web is a vision about an extending the existing World Wide Web. The data in www is unstructured if this data is made a part of the new Semantic web, can be referenced directly by their unique identifier, called Universal Resource Identifier (URI). URIs are interconnected in a graph infrastructure thus comprising a huge library of information that can be easily and uniformly accessed by Semantic web applications.

The benefit of using Semantic web over the existing web is, the Semantic web facilitates the machines to understand the information allowing the users to perform task effortlessly. To enable the encoding of semantic with the data, technologies such as Resource description framework (RDF) and Web ontology Language(OWL) are used. These technologies are used to formally represent metadata.

**2.2.2 Spring Boot Micro-services**

Microservices are increasingly used to create larger, more complex applications that are better developed and managed as a combination of smaller services that work cohesively together for larger, application-wide functionality. Spring Boot is a java-based framework which builds a standalone environment which communicates with many microservices available in our application.

The benefits of using Microservice architecture is it would allow the developers to scale up or down the size of the project independently. This feature is also useful in decreasing the expenses for the overall application also the failure of one module would not affect the whole operation of the application.

**2.2.3 Angular Framework**

Angular is a component-based framework for building scalable web application. The basic building blocks of the Angular framework are Angular components that are organized into NgModules. NgModules collect related code into functional sets, an Angular application is defined by a set of NgModules. An application always has at least a root module that enables bootstrapping, and typically has many more feature modules.

Angular applications are built using TypeScript language, a superscript for JavaScript, which ensures higher security as it supports types (primitives, interfaces, etc.). It helps catch and eliminate errors early when writing the code or performing maintenance tasks. With Angular, we do not need any additional getter and setter functions. Since, every object it uses is POJO, which enables object manipulation by providing all the conventional JavaScript functionalities.

**3. Design and Analysis**

**3.1 Technical Stack Design**

In this section we will discuss on the different technologies used in this application. They are mainly Semantic web Technology and Java Technology along with Angular UI for Front End services.

**3.1.1 Semantic web Technologies:**

The data in www is unstructured if this data is made a part of the new Semantic web, can be referenced directly by their unique identifier, called Universal Resource Identifier (URI). URIs are interconnected in a graph infrastructure thus comprising a huge library of information that can be easily and uniformly accessed by Semantic web applications.

1. **Turtle:**

A Turtle file allows writing down an RDF graph in a compact textual form. An RDF graph represents information using semantic triples consisting of a subject, predicate and object. Each item in the triple is expressed as a Web URL. Turtle provides a way to group three URIs to make a triple, and ways to abbreviate such information. Subjects are referenced by a number of predicates.

1. **Protégé:**

Protégé is an editor and a knowledge-based framework which helps us to create ontology models. The Protégé version we are using is Protégé 5.5.0. It is an ontology editing environment with full support for the OWL 2 web ontology language and direct in-memory connection to description logic reasoner like Pellet.

1. **Apache Jena:**

Apache Jena API is a Semantic web framework for java. It provides extensive java libraries for helping developers to develop code that handles RDF, RDFa, OWL and SPARQL. The API version used here is API 3.15.0. The graphs are represented as an abstract model. Jena includes a rule-based inference engine to perform reasoning based on OWL and RDFS ontologies, and a variety of storage strategies to store RDF triples in memory or on disk. These models are queried using SPARQL.

1. **Openllet:**

Openllet is a plugin to implement Pellet reasoner in Jena API. The version of Openllet is 2.6.5 and is currently only supported by Jena-API version of 3.15. For further API support to work with Openllet, Java SE 10 is required. Dependencies required for Jena are:

* + Jena-core: It is a jena framework for building semantic web. It provides a pragmatic environment for RDF, RDFS and QWL.
  + Jena-arq: ARQ is a sparqul 1.1 query engine for apache jena.
  + Jena-tdb: It is a storage system for jena and arq.
  + Jena-cmds: It is a command line tool
  + Jena-iri: Provides implementation for IRI and URI specifications.

**3.1.2 Java Technologies:**

Java is an object-oriented programming language. This implies that the execution of a Java program creates objects, modifies them, and updates object references. The program behavior during the execution is therefore described by the evolution of its object list. The semantic specification of our subset of Java is composed of about 850 inference rules. Some modules are expressed in a Natural Semantics style. The Java version used here is Java SE 10. This version is only compatible with both semantic and openllet reasoner.

1. **Apache Maven 3.6.3:**

Maven is a build automation tool used primarily for Java projects. Maven addresses two aspects of building software: how software is built, and its dependencies. An XML file describes the software project being built, its dependencies on other external modules and components, the build order, directories, and required plugins. It comes with pre-defined targets for performing certain well-defined tasks such as compilation of code and its packaging. The maven version used here is Maven 3.6.3.

1. **Spring Boot:**

Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications thatyou can "just run". We take an opinionated view of the Spring platform and third-party libraries so you can get started with minimum fuss. Most Spring Boot applications need minimal Spring configuration. Provide opinionated 'starter' dependencies to simplify your build configuration automatically configure Spring and 3rd party libraries whenever possible.

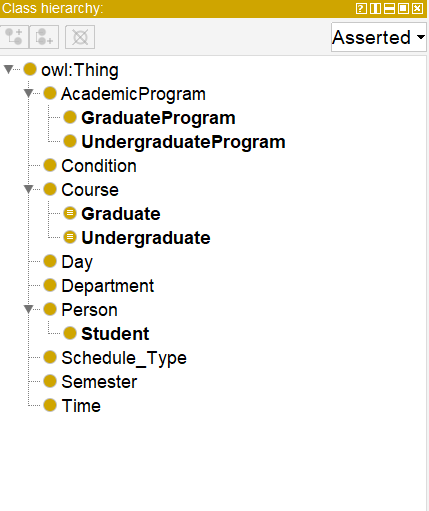
* 1. **Semantic Web Schema Models**

This Ontology model contains two different ontology Faculty Ontology and Courses Ontology. The faculty ontology contains details of Faculty and staff members . The Course Ontology contains the course details, time, days and the semester in which the courses are provided by the department for both graduate and under-graduate students. The two ontology are combined to provide a relationship between the faculty and the course where we obtain the details of professors teaching different courses.

**3.2.1Course Schema**

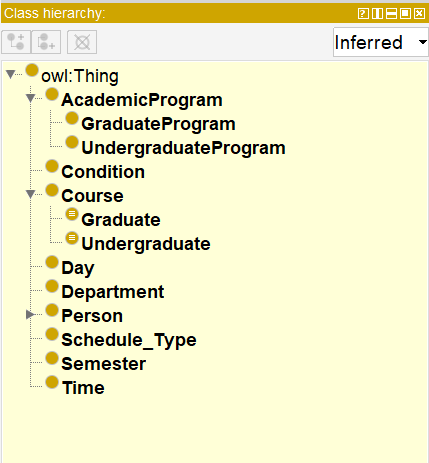
**1. Class Hirarchy: Asserted class hierarchy:**

The asserted class hierarchy view is one of the primary navigation devices for named OWL classes. In the asserted class there are various domains like Academic Program, Courses, department, Schedule-Type, Semester, Time in Course Ontology.



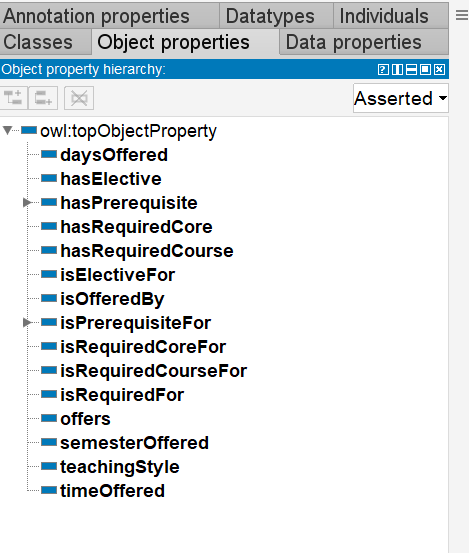
**2. Inferred class hierarchy:**

The inferred class is obtained after running the reasoned. The inferred hierarchy helps to classify entities depending on the values that are been inferred. The inferred hierarchy can be obtained with protégé application using pellet reasoner. The following image shows us how reasoner is done in protégé .



**3. Object Property:**

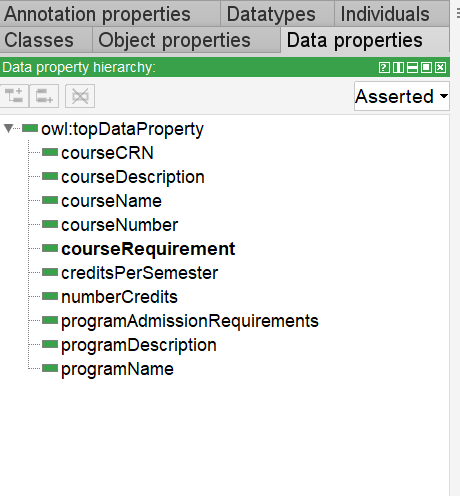
The object property helps to create relationship between various individuals in a class. The object properties used here are hasElective which is inverse of isElectiveof similarly we can other object properties from the below given diagram.



**4. Data Property:**

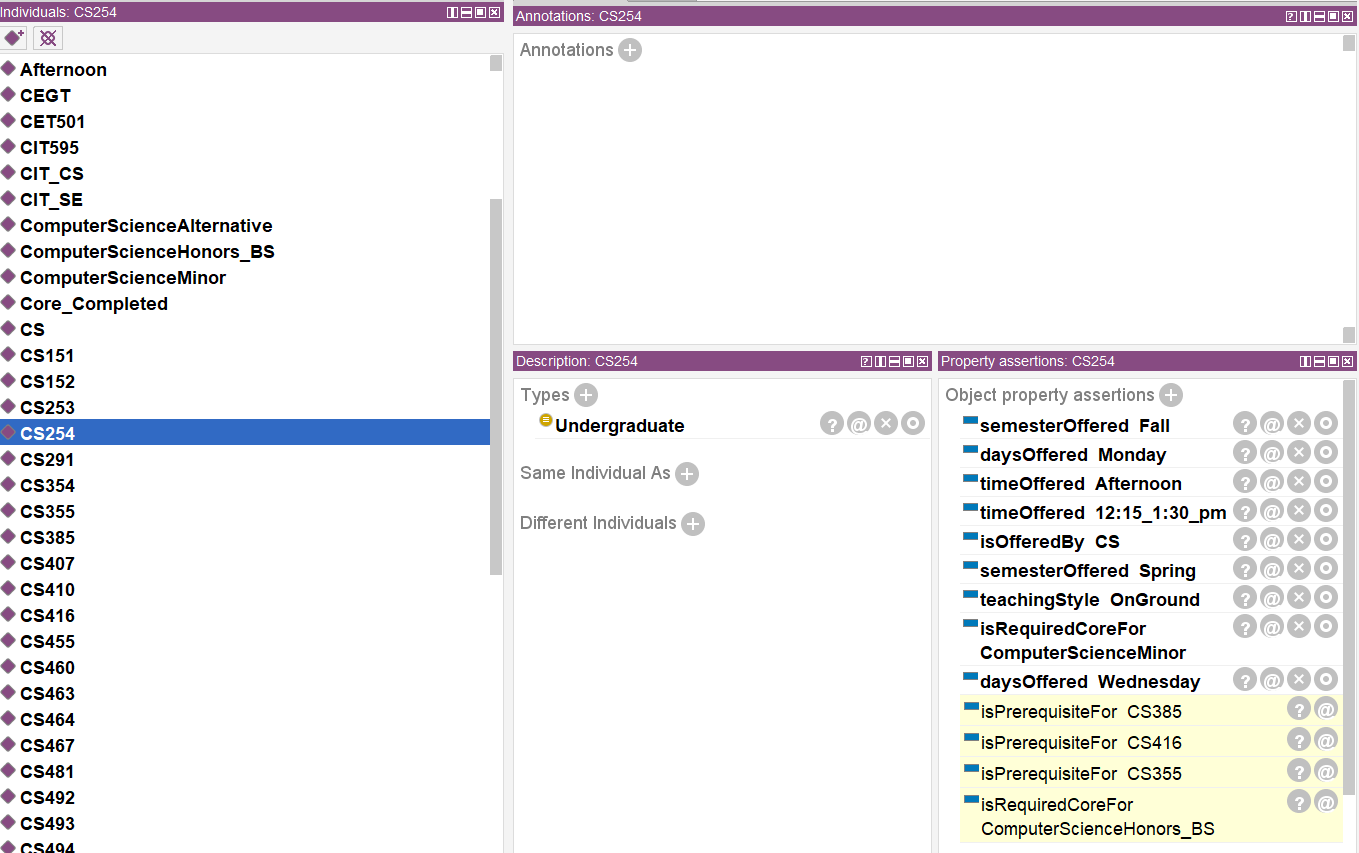
Data Property describes what kind of *values* a triple with the property should have.Data property relates individuals to literal data (e.g., strings, numbers, date, times, etc.)

* Few Data properties are defined for literal and integer values
* The data property hierarchy view displays the asserted and inferred data property hierarchies.



**5. Individuals and Properties:**

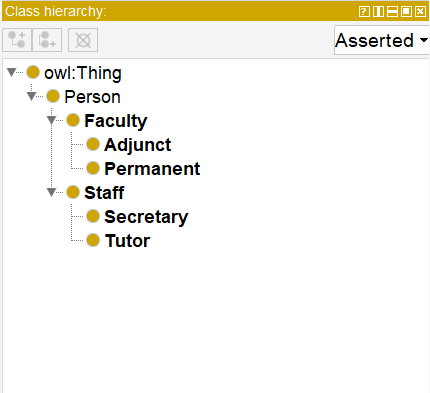
The named individual helps to create a set of individuals and it also defines object or data property for each entity. The image shows the list of individuals defined in the Faculty and course schema.



**3.2.2Faculty Schema**

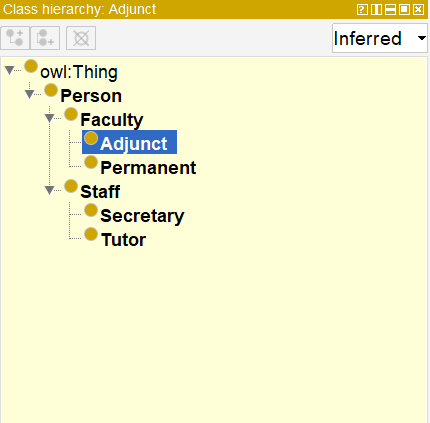
**1. Class Hierarchy: Asserted class hierarchy:**

The asserted class hierarchy view is one of the primary navigation devices for named OWL classes. In the asserted class there are various domains like name, email, phone and website for faculty and staff for Faculty Ontology.



**2. Inferred class hierarchy:**

The inferred class is obtained after running the reasoned. The inferred hierarchy helps to classify entities depending on the values that are been inferred. The inferred hierarchy can be obtained with protégé application using pellet reasoner. The following image shows us how reasoner is done in protégé .



**3.2.3 Integrated Schema**

The integrated Schema is where both the Faculty and Course models are integrated together to provide a coherent relationship between them. The relationship we generate is to provide the users with the data which faculty teaches what course. The data is updated and integrated using a pellet reasoner using jena.

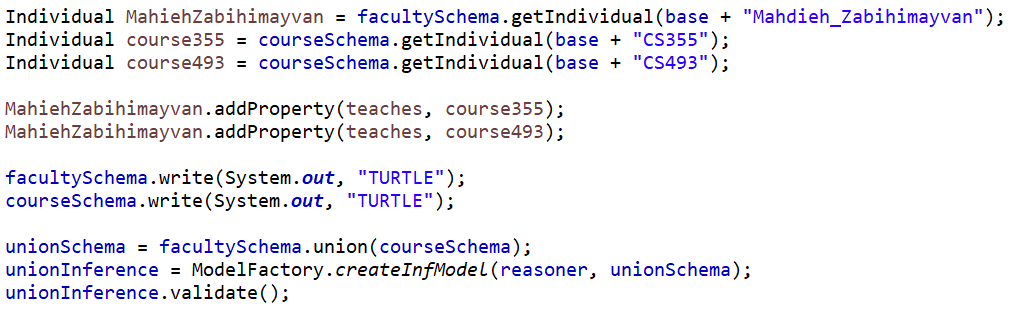
1. **Usage of pellet reasoner in Integrated schema:**

unionSchema = facultySchema.union(courseSchema);

unionInference = ModelFactory.createInfModel(reasoner, unionSchema);

The pellet reasoner is used to validate and obtain results for course schema, faculty and read Integrity Schema. The execution and the output of the reasoner running using the pellet reasoner is obtained.

* The two schemas StudentProgarm and Faculty are merged after adding a property teaches in Jena API and reasoner is run to validate the new integrated schema.
* unionShcema is now considered to contain the merged data from both the Model Schemas
* The Faculty and course ontology are integrated together where classes, resources, Individuals and property are created.
* A relationship can be developed by adding triples to these properties.



* 1. **Course Op Service**

This service provides the list of courses and faculty based in CS department. The data is obtained from RDF triple store and the data is represented through SPARQL queries in Jena model. The data retrieved is then displayed in users through JSP pages.

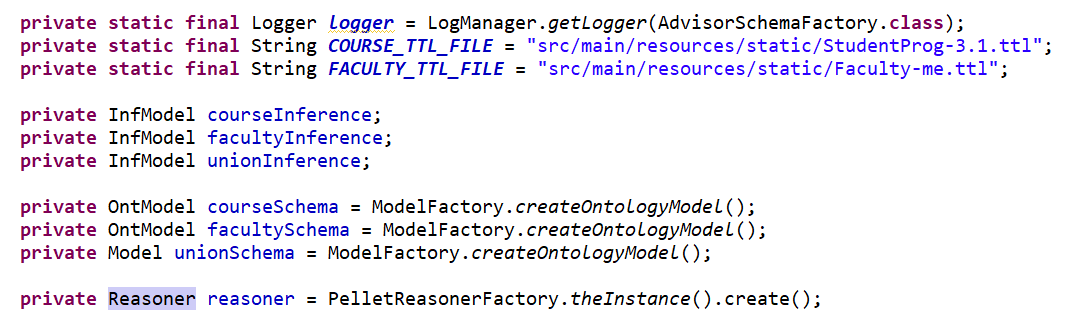
**3.3.1 Jena API Service**

Jena API is used to Capture the validated Schema from the reasoner and can be considered as Inference Model after reasoning . Jena API uses various Models to maintain the Schema in a relevant way to User. Some of them are: InfModel, OntModel.

Likewise, Reasoner also have different flavors.

•RuleReasoner is available with API and ReasonerRegistry class is used to obtain the same

•Pellet reasoner needs an external plugin and PelletReasonerFactory is the class used foto get this reasoner object.



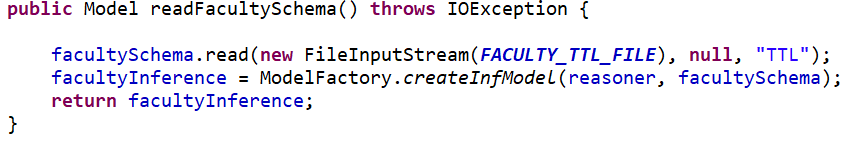
**1. Reasoner:**

Jena allows of three types of reasoners:

* Internal reasoner integrated into Jena framework. We will work only with these.
* External reasoners offered as external Java (.jar) files. I was able to find “old” references on how Pellet reasoner can be integrated in Jena.
* External reasoners remotely offered via the Description Logics Information Group (DIG) interface. We will not discuss those.

We are using pellet reasoner for Jena integration of different schema which is course and faculty schema. This integrates course and faculty schema to create Merged schema.

The image shown below is the faculty schema:



**3.3.2 Front-end JSP Design**

**3.4 Login and Registration Services**

**3.4.1 Backend Design**

**3.4.2 Front-end Angular Design**

**3.4.3 Additional Services**

**1. Forgot Password**

**2. Registration Override**

**3.5 It all fits (Integration)**

**3.5.1 Role of Microservices**

**3.5.2 Microservice Integration**

**3.5.3 Wiring with Spring boot**