Master’s Project

Summer 2023

REST API Banking back-end development in Java Spring Boot framework utilizing CRUD functionality.

**By Sumit Mistry**

CWID 50067462

CSIT-697 Master's Project

Term: Summer-2023

**Advisor: Dr. George Antoniou**

Department of Computer Science

Montclair State University

Montclair, NJ 07043

Table of Contents

[1.Introduction 4](#_Toc134571054)

[2.Purpose 4](#_Toc134571055)

[3.Abstract 4](#_Toc134571056)

[4.Proposed Work 5](#_Toc134571057)

[4.1 CSIT 696 - Research Methods in Computing 5](#_Toc134571058)

[4.2 CSIT 697- Master's Project 5](#_Toc134571059)

[5.REST-Architecture 5](#_Toc134571060)

[6.REST - Characteristics 6](#_Toc134571061)

[6.1 Client-Server Architecture: 6](#_Toc134571062)

[6.2 Statelessness: 6](#_Toc134571063)

[6.3 Cacheability: 6](#_Toc134571064)

[6.4 Layered System: 6](#_Toc134571065)

[6.5 Code-On-Demand: 6](#_Toc134571066)

[6.6 Uniform Interface: 7](#_Toc134571067)

[What contains within RESTful API client request? 7](#_Toc134571068)

[7.Project Prerequisites 7](#_Toc134571069)

[8.Spring Boot and Spring Initializer: 7](#_Toc134571070)

[8.1 Advantage of using Spring Boot 7](#_Toc134571071)

[8.2 Spring Initializer: https://start.spring.io/ 8](#_Toc134571072)

[9.@SpringBootApplication and Spring Boot annotations 8](#_Toc134571073)

[9.1@EnableAutoConfiguration 8](#_Toc134571074)

[9.2@Configuration 8](#_Toc134571075)

[9.3@ComponentScan 8](#_Toc134571076)

[10.MVC Application architecture: 8](#_Toc134571077)

[10.1 DAO 8](#_Toc134571078)

[10.2 Repository 8](#_Toc134571079)

[10.3 Model 8](#_Toc134571080)

[10.4 Controller 8](#_Toc134571081)

[10.5 @Entity 9](#_Toc134571082)

[10.6 @Table 9](#_Toc134571083)

[10.7 @Column 9](#_Toc134571084)

[10.8 @Id 9](#_Toc134571085)

[10.9 @GeneratedValue 9](#_Toc134571086)

[10.10 GenerationType.AUTO 9](#_Toc134571087)

[10.11 GenerationType.IDENTITY - 9](#_Toc134571088)

[10.12 GenerationType.SEQUENCE 9](#_Toc134571089)

[10.13 GenerationType.TABLE 9](#_Toc134571090)

[10.14 @Repository 9](#_Toc134571091)

[10.15 @RequestMapping 9](#_Toc134571092)

[10.16 @RestController 9](#_Toc134571093)

[@Controller 9](#_Toc134571094)

[@ResponseBody 9](#_Toc134571095)

[@RequestBody 9](#_Toc134571096)

[11.GitHub (Source code) 10](#_Toc134571097)

[12.Proposed work CSIT 691 : Independent Computer Science Study 10](#_Toc134571098)

[13.Assumption 10](#_Toc134571099)

[14.Planned Tasks breakdown 11](#_Toc134571100)

[15.References: 13](#_Toc134571101)

[16.App Deployment/ App in action / Demo: 14](#_Toc134571102)

[16.1 Spring Init with “Hello World” 14](#_Toc134571103)

[16.2 MySQL server setup: 15](#_Toc134571104)

[16.3 MySQL workbench 19](#_Toc134571105)

[16.4 MySQL: Create Table: 20](#_Toc134571106)

[16.5 Testing End-Points: 21](#_Toc134571107)

[16.6 Postman: POST(200 OK response) 22](#_Toc134571108)

[16.7 Postman: GET (200 OK response) 25](#_Toc134571109)

[16.8 Postman: PUT/UPDATE (200 OK response) 26](#_Toc134571110)

[16.9 Postman: DELETE (200 OK response) 28](#_Toc134571111)

[17.Best Practices and lesson learned 29](#_Toc134571112)

[17.1 Unknow error 29](#_Toc134571113)

[17.2 VM 30](#_Toc134571114)

[17.3 Environmental Variable 30](#_Toc134571115)

[17.4 Student\_ID autoincrement 30](#_Toc134571116)

[17.5 DELETE operation 31](#_Toc134571117)

1.Introduction

|  |  |
| --- | --- |
| **Acronyms** | **Explanation** |
| REST | Representational(XML, JSON) State(data) Transfer(between client and server) |
| API | Application Programming Interface |
| REST API | API enables two applications to communicate with each other over HTTP |

The purpose of this project is to illustrate how new technology can help establishing communication between client and server over HTTP. Building REST API is the target for the project where user can person certain operation to perform the business logic. User triggers an API, which further calls the server, server performs whole bunch of business logic like CRUD operations and returns the result.

2.Purpose

The goal of presenting a research method is to create a prototype of an Application Programming Interface(API) in the Java language using Spring boot framework and implementing a REST architectural capability to manage the CRUD operation that supports the user request to do some basic POST GET PUT DELETE requests. The prototype will be serving like a Minimum Viable Product (MVP) that will fulfill user’s requirement to make a backend REST API web-application.

Another objective for this project is to demonstrate the new technologies implementation. Spring Boot, a Java framework which can manage all the classes of Java application and doesn’t need Embedded Tomcat server to run Spring Boot applications. Spring Boot helps configuring application where the certain dependencies are automatically taken care of.

The implementation of the prototype shows how the development will be implemented as well as some samples of its functionality. After the implementation, the results will be presented and assessed. In conclusion, the proposal to further develop the prototype will be presented to continue building complex banking application platform so that the clients or users can use it instantly.

The solution will be posted on git repository and will be an open-source code. If version controlling required to enhance the API functionality, it will be implemented at the later stage of the project during Master Project.

3.Abstract

The most widely recognized form of web REST API has ability to manage and transfer data between client and server. This reasech will help progressively evolving Businesses and Financial services to deploy their web-services to expand the business and competitive presence in the market. This research is focusing on designing higher quality REST API web-app services by identifying the key research areas that contribute straightforward development and deployment with more reliability by incorporating Spring Boot framework. The project aims higher quality standard web-app designing with better performance promise.

4.Proposed Work

## 4.1 CSIT 696 - Research Methods in Computing: Completed

Researched the usage of REST API Spring boot framework in building towards backend web-app using Java.

## 4.2 CSIT 697- Master's Project

Building the backend Banking platform using REST API Spring Boot Java, this will be a next step towards creating a useful REST API that will be readily usable by consumers.

5.REST-Architecture

REST API architecture is designed in such a manner that it follows certain principles that help them maintain performance, highly available, extensible, and easily scalable.

Diagram

Description automatically generated

Figure:1 REST Architecture <https://www.linkedin.com/pulse/understanding-rest-architecture-gabriel-gitonga>

Gitonga, G. (n.d.). Understanding rest architecture. LinkedIn. Retrieved April 12, 2023, from <https://www.linkedin.com/pulse/understanding-rest-architecture-gabriel-gitonga>

6.REST - Characteristics

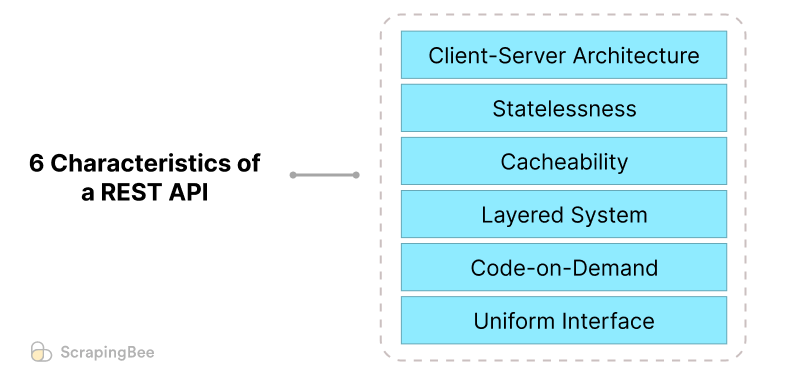


Figure:2 REST Characteristics <https://www.scrapingbee.com/blog/six-characteristics-of-rest-api/>

Khalid, Y. (2023c). What are the 6 characteristics of a rest api? Retrieved from <https://www.scrapingbee.com/blog/six-characteristics-of-rest-api/>

## 6.1 Client-Server Architecture:

RESTful APIs has client-server architecture as its core architecture. Client can be any device or application that can make HTTP requests. Server is application in a device that allows API to responds to client requests.

## 6.2 Statelessness:

each request from client to server or vice versa are self-contained. Each request contains auth token and so does not depend on other request as they are independent from others. REST request has to carry an authentication token in the request headers. Each REST call has to carry authentication token and so state will be independent of the client or server.

## 6.3 Cacheability:

RESTful APIs utilize cache to reduce the load on the server. Server’s API response can be caches by the client and that is the reason when client calls the same api call later, the same requested cache data can be useful for the faster response.

## 6.4 Layered System:

REST API architecture includes client and server, the client and server are connected through single end-point, but the server are connected to multiple server or back end systems and so when the server needs the replacement/maintenance for upgradation, the API calls/communication will not be impacted by this multi layered system.

## 6.5 Code-On-Demand:

For client to understand and to be able to execute the code that the server sends back, client needs to get some portion of the code from server. Only because of this functionality, client can customize the user experience on client end. This characteristic more exploits the server than the meaningful application.

## 6.6 Uniform Interface:

API uses and accepts only common sets method of like GET, POST, PUT and DELETE for data communication, XML and JSON are commonly used for API response. All such standard format and communication protocol helps client and server to understand and communicate easily in a consistent manner.

## What contains within RESTful API client request?

1. Method (GET, POST, PUT, DELETE)
2. HTTP headers in a JSON form (Data)

7.Project Prerequisites

1. Java Development Kit (JDK) version 8 or newer.
2. Maven 2.8 or newer
3. Spring Boot Initializer
4. MySQL server / database
5. MySQL Workbench/GUI schema lookup tool
6. IntelliJ IDE for code development.
7. Postman to test the APIs
8. Linux
9. Git
10. Dev-Ops tools(if required)
11. Docker (if required)

8.Spring Boot and Spring Initializer:

. \_\_\_\_ \_ \_\_ \_ \_

/\\ / \_\_\_'\_ \_\_ \_ \_(\_)\_ \_\_ \_\_ \_ \ \ \ \

( ( )\\_\_\_ | '\_ | '\_| | '\_ \/ \_` | \ \ \ \

\\/ \_\_\_)| |\_)| | | | | || (\_| | ) ) ) )

' |\_\_\_\_| .\_\_|\_| |\_|\_| |\_\\_\_, | / / / /

=========|\_|==============|\_\_\_/=/\_/\_/\_/

:: Spring Boot :: (v2.5.6)

This is a Java framework built on top of the [Spring](https://spring.io/projects/spring-framework/). It helps in developing web applications. It helps in creating REST APIs with minimal configurations and minimal efforts.

## 8.1 Advantage of using Spring Boot

1. An auto-configuration feature by Spring Boot that configures your application automatically for certain dependencies
2. Embedded Tomcat server to run Spring Boot applications

## 8.2 Spring Initializer: <https://start.spring.io/>

9.@SpringBootApplication and Spring Boot annotations

9.1@EnableAutoConfiguration

Auto-configuration feature of Spring Boot. Auto-configuration feature by Spring Boot that configures your application automatically for certain dependencies. If the dependency is available in your classpath, Spring Boot will auto-create the beans for it. Beans in Spring are objects that are instantiated and managed by Spring

9.2@Configuration

It specifies a configuration class, where you’ll provide all the bean definitions that your application is using. Spring Boot will use the bean definitions provided in the configuration class to instantiate them at runtime.

9.3@ComponentScan

Allows Spring Boot to scan the package for components like Service, Controller, Repository, etc. and register beans for each of those classes.

10.MVC Application architecture:

10.1 DAO

The DAO (data access layer) provides an interface to connect with the database and access the data stored in the database. A single DAO class can deal with queries retrieving different types of entities.

10.2 Repository

This layer is similar to the DAO layer which connects to the database and accesses the data. However the repository layer provides a greater abstraction compared to the DAO layer. Every class is responsible for accessing and manipulating one entity. This tutorial will use the repository layer.

Service - This layer calls the DAO layer to get the data and perform business logic on it. The business logic in the service layer could be - performing calculations on the data received, filtering data based on some logic, etc.

10.3 Model

The model contains all the Java objects that will be mapped to the database table using. The DAO will fetch the data from the database and populate the respective model with that data and return it to the service layer and vice versa.

10.4 Controller

This is the topmost layer, called when a request comes for a particular REST API. The controller will process the REST API request, calls one or more services and returns an HTTP response to the client.

10.5 @Entity annotation specifies that this Java class is mapped to the database table.

10.6 @Table with the help of the property name specifies which particular table this class is mapped to.

10.7 @Column on each Java instance variable allows defining a set of properties like name, length etc. The name property will be the name of the field in the database table that this instance variable will map to.

10.8 @Id on a field tells Spring Boot that this particular field is the primary key in the table

10.9 @GeneratedValue specifies the strategy that will be used for generating primary key

10.10 GenerationType.AUTO - This is the default strategy used by Spring Boot. If you use this strategy, the JPA provider will decide on an appropriate strategy to generate the primary key depending on the dialect given in the application.properties file.

10.11 GenerationType.IDENTITY - this strategy uses the database identity column to determine the primary key strategy. For example, you defined the emp\_id column as auto-increment in the database while creating the employee table. Now when you use this strategy then a unique primary key is generated by starting from 1 and incrementing every time a new row is inserted in the table.

10.12 GenerationType.SEQUENCE - this strategy uses database sequence to generate the primary keys.

10.13 GenerationType.TABLE - this strategy uses a database table to generate primary keys.

10.14 @Repository on the class indicates that the class is a data repository that will contain CRUD operations. CRUD is an acronym that stands for the four basic operations of the database - Create, Read, Update, Delete.

10.15 @RequestMapping annotation on the class defines a base URL for all the REST APIs created in this controller. This base URL is followed by individual REST endpoints given to each of the controller methods.

10.16 @RestController on the class is a combination of:

@Controller - tells Spring Boot that this class is a controller.

@ResponseBody - indicates that the return value of the methods inside the controller will be returned as the response body for the REST API.

@RequestBody annotation is used to map the request body of the endpoint to the method parameter. emp will contain the request JSON passed to this endpoint.

11.GitHub (Source code)

CSIT 696: Research Methods in Computing

https://github.com/SumitMistry/CSIT696\_Research\_Java\_REST\_API

12.Proposed work CSIT 691 : Independent Computer Science Study

The study is performed to understand the Java API JDBC, its connection strategies, and configurable parameters related study that can help a Java app to successfully connect the PostgreSQL; one of the popular relational databases.

* Setup DBMS: MySQL server
* Configure and initiate database schema and creating a table
* Setup Java configuration and test “Hello World”
* Code the Java application as per the architecture defined in the ***Figure 3***:

Developer will code per below 4 major layers:

1. Method
2. Controller
3. Repository
4. DAO
5. Spring-Java Application Runner

* Create a JDBC connection string for MySQl using Spring default Application.Properties.
* If required; Implement SQLException handling using class: import java.sql.SQLException
* If needed, docker will be implemented during deployment of dB and Java App.
* Research REST API and Spring Boot for:
  1. Constructing complex web-app
  2. Simplicity to use for dependency injection.
  3. Spring Beans
* Summarize the research work.

13.Assumption

If needed, Docker(containerization) will be implemented, and the documentation will be provided. This project will establish the basic of the Spring Boot and REST API building, API consumption and it’s configuration. Furthermore, the web-app will be developed from the existing package CSIT-696 submission. The final Maven JAR package of the snapshot to be used for further development of the web-application for financial institution integrating stable and scalable RDBMS.

14.Planned Tasks breakdown

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Task | Description | Deliverable | Due (approx.) |
| 1 | Project Proposal draft | Prototyping drafting REST API web-application development and research strategy. | Refined draft document of the proposed project | 04-Nov-2022 |
| 2 | **Prerequisites:**  IDE, Java JDK, Maven (mvn build), Cloud services, SQL equivalent dB git clone | Installation and deploying the required IDE, environment, software and services. | Report with progress of the task | 10-Nov-2022  to  14-Nov-2022 |
| 3 | Linux and Git repository: | Linux and Git repository setup and Linux or cloud | Report with progress of the task | 20-Nov-2022  to  24-Nov-2022 |
| 4 | Java project structure setup Eclipse IDE | Setup Java with ‘HelloBolo’ test project and IDE setup | Report with progress of the task | 26-Nov-2022 |
| 5 | Spring Initializer and Spring Boot framework setup | Initialize spring boot framework for the project | Submit report with progress of the task | 12-Dec-2022  to  24-Dec-2022 |
| 6 | Configure Database and dB Implementation | Implement JDBC/hibernate/ PostgreSQL/SQL database scripts and server-side scripts. | Submit report with progress of the task | 24-Dec-2022  to  31-Dec-2022 |
| 7 | Integration (If required):  dB DAO controller, JDBC | Optional integration if require an additional Database connectors or framework. | Ad-Hoc | Ad-Hoc |
| 8 | RESTful API setup.  Using HTTP Methods for RESTful Services that has POST GET PUT DELETE functions | Project will have CRUD functionality that user can leverage its Create read, update delete operations. | Provide the GitHub repository | 17-Jan-2023 |
| 9 | Markdown documentation, screenshots, git repository, inclusively HTTP response codes. | Provide markdown readme.md and detailed documentation about the project design and instruction to install or run the project | Provide the GitHub repository and readme.md | 17-Jan-2023 |
| 10 | Testing the URLs With Postman | Testing the functionality of the REST API end point. | Submit a report with test data and screenshot if needed | 31-Jan-2023  to  05-Feb-2023 |
| 11 | Swagger(if required) | Swagger back-end web-API testing end point | Swagger documentation with git link. | 31-Jan-2023  to  05-Feb-2023 |
| 12 | Dockerize The API(if needed) | Dockerizring the application if paid Linux subscription were used. | Ad-Hoc | Ad-Hoc |
| 13 | Best Practices and lesson learned | Best Practices and lesson learned during the project. | Submit a report on the current state of the art approaches. | 10-Feb-2023  to  21-Feb-2023 |
| 14 | Research method conclusion (The report) | The report will be generated. | Submit a report on the current state of the art approaches. | 10-Feb-2023  to  21-Feb-2023 |
| 15 | Presentation | Live session of the REST API project with CRUD operation demo.  All supported documents should be done by the presentation. | Powerpoint presentation and demo should be completed. | 10-Feb-2023  to  21-Feb-2023 |
| 16 | CSIT 697  Master's Project proposal | Finalize project scope and objectives and prepare draft data science project report.  Use the existing structure and code to develop and enhance the CRUD operation of the REST API application to develop the backend banking platform using web-app. | Submit Masters project proposal to continue developing further complex banking application platform requirements so that the project will be deliverable to the client/user which can be readily usable. | 15-Feb-2023  to  04-Mar-2023 |

15.References:

1. Java <https://docs.oracle.com/javase/7/docs/api/>
2. Maven <https://maven.apache.org/plugins/maven-javadoc-plugin/usage.html>
3. Spring Initializer <https://start.spring.io/>, <https://www.springboottutorial.com/spring-boot-crud-rest-service-with-jpa-hibernate>, <https://www.springboottutorial.com/spring-boot-rest-api-projects-with-code-examples>
4. Swagger documentation <https://swagger.io/docs/>
5. Bachina, B. (2020, December 17). How to develop and build java REST API. Medium. Retrieved October 25, 2022, from <https://medium.com/bb-tutorials-and-thoughts/how-to-develop-and-build-java-rest-api-65f708c22fb3>
6. REST API repository class Khan, N. (2022, September 22). How to create rest apis with Java and spring boot. Twilio Blog. Retrieved October 25, 2022, from <https://www.twilio.com/blog/create-rest-apis-java-spring-boot>
7. Jersey.java.net. (n.d.). Jersey 2.25-1 user guide. Retrieved from https://jersey.java.net/documentation/latest/user-guide.html
8. Dinadasa, Chinthaka and Java Dev. “Microservices - Core Banking Service Implementation.” Java to Dev, 27 May 2021, javatodev.com/microservices-core-banking-service-implementation/. Accessed 25 Oct. 2022.

‌

1. Minh, Nam Ha. “Spring Boot REST API CRUD with HATEOAS Tutorial.” Www.codejava.net, 6 Sept. 2022, www.codejava.net/frameworks/spring-boot/rest-api-crud-with-hateoas-tutorial. Accessed 25 Oct. 2022.

‌

1. GitHub.com, <https://github.com/>

16.App Deployment/ App in action / Demo:

## 16.1 Spring Init with “Hello World”

The summarized steps for the app deployment by following the below steps:

Hello World. This is CSIT-696 Springboot REST API project.

  .   \_\_\_\_          \_            \_\_ \_ \_

 /\\ / \_\_\_'\_ \_\_ \_ \_(\_)\_ \_\_  \_\_ \_ \ \ \ \

( ( )\\_\_\_ | '\_ | '\_| | '\_ \/ \_` | \ \ \ \

 \\/  \_\_\_)| |\_)| | | | | || (\_| |  ) ) ) )

  '  |\_\_\_\_| .\_\_|\_| |\_|\_| |\_\\_\_, | / / / /

 =========|\_|==============|\_\_\_/=/\_/\_/\_/

 :: Spring Boot ::               (v2.7.11)

2023-05-07 21:22:57.160  INFO 13404 --- [           main] 696ResearchMethodsInComputingApplication : Starting Csit696ResearchMethodsInComputingApplication using Java 1.8.0\_211 on Sumit-i7 with PID 13404 (D:\github\CSIT696\_Research\_Java\_REST\_API\target\classes started by sumit in D:\github\CSIT696\_Research\_Java\_REST\_API)

2023-05-07 21:22:57.164  INFO 13404 --- [           main] 696ResearchMethodsInComputingApplication : No active profile set, falling back to 1 default profile: "default"

2023-05-07 21:22:58.172  INFO 13404 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.

2023-05-07 21:22:58.200  INFO 13404 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 10 ms. Found 0 JPA repository interfaces.

2023-05-07 21:23:00.012  INFO 13404 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)

2023-05-07 21:23:00.032  INFO 13404 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]

2023-05-07 21:23:00.033  INFO 13404 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.74]

2023-05-07 21:23:00.277  INFO 13404 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext

2023-05-07 21:23:00.278  INFO 13404 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 3036 ms

2023-05-07 21:23:00.391  WARN 13404 --- [           main] ConfigServletWebServerApplicationContext : Exception encountered during context initialization - cancelling refresh attempt: org.springframework.beans.factory.UnsatisfiedDependencyException: Error creating bean with name 'dataSourceScriptDatabaseInitializer' defined in class path resource [org/springframework/boot/autoconfigure/sql/init/DataSourceInitializationConfiguration.class]: Unsatisfied dependency expressed through method 'dataSourceScriptDatabaseInitializer' parameter 0; nested exception is org.springframework.beans.factory.BeanCreationException: Error creating bean with name 'dataSource' defined in class path resource [org/springframework/boot/autoconfigure/jdbc/DataSourceConfiguration$Hikari.class]: Bean instantiation via factory method failed; nested exception is org.springframework.beans.BeanInstantiationException: Failed to instantiate [com.zaxxer.hikari.HikariDataSource]: Factory method 'dataSource' threw exception; nested exception is org.springframework.boot.autoconfigure.jdbc.DataSourceProperties$DataSourceBeanCreationException: Failed to determine a suitable driver class

2023-05-07 21:23:00.396  INFO 13404 --- [           main] o.apache.catalina.core.StandardService   : Stopping service [Tomcat]

2023-05-07 21:23:00.427  INFO 13404 --- [           main] ConditionEvaluationReportLoggingListener :

Error starting ApplicationContext. To display the conditions report re-run your application with 'debug' enabled.

2023-05-07 21:23:00.457 ERROR 13404 --- [           main] o.s.b.d.LoggingFailureAnalysisReporter   :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

APPLICATION FAILED TO START

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Description:

Failed to configure a DataSource: 'url' attribute is not specified and no embedded datasource could be configured.

Reason: Failed to determine a suitable driver class

Action:

Consider the following:

  If you want an embedded database (H2, HSQL or Derby), please put it on the classpath.

  If you have database settings to be loaded from a particular profile you may need to activate it (no profiles are currently active).

Process finished with exit code 1

## 16.2 MySQL server setup:

sudo apt install mysql-server

root@ubuntu-22:/etc# sudo apt install mysql-server

Reading package lists... Done

Building dependency tree... Done

Reading state information... Done

The following packages were automatically installed and are no longer required:

  libflashrom1 libftdi1-2 libllvm13

Use 'sudo apt autoremove' to remove them.

The following additional packages will be installed:

  libaio1 libcgi-fast-perl libcgi-pm-perl libevent-core-2.1-7

  libevent-pthreads-2.1-7 libfcgi-bin libfcgi-perl libfcgi0ldbl

  libhtml-template-perl libmecab2 libprotobuf-lite23 mecab-ipadic

  mecab-ipadic-utf8 mecab-utils mysql-client-8.0 mysql-client-core-8.0

  mysql-server-8.0 mysql-server-core-8.0

Suggested packages:

  libipc-sharedcache-perl mailx tinyca

The following NEW packages will be installed:

  libaio1 libcgi-fast-perl libcgi-pm-perl libevent-core-2.1-7

  libevent-pthreads-2.1-7 libfcgi-bin libfcgi-perl libfcgi0ldbl

  libhtml-template-perl libmecab2 libprotobuf-lite23 mecab-ipadic

  mecab-ipadic-utf8 mecab-utils mysql-client-8.0 mysql-client-core-8.0

  mysql-server mysql-server-8.0 mysql-server-core-8.0

0 upgraded, 19 newly installed, 0 to remove and 1 not upgraded.

Need to get 29.2 MB of archives.

After this operation, 242 MB of additional disk space will be used.

Do you want to continue? [Y/n] Y

Get:1 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64 mysql-client-core-8.0 amd64 8.0.32-0ubuntu0.22.04.2 [2,677 kB]

Get:2 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64 mysql-client-8.0 amd64 8.0.32-0ubuntu0.22.04.2 [22.7 kB]

Get:3 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 libaio1 amd64 0.3.112-13build1 [7,176 B]

Get:4 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 libevent-core-2.1-7 amd64 2.1.12-stable-1build3 [93.9 kB]

Get:5 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 libevent-pthreads-2.1-7 amd64 2.1.12-stable-1build3 [7,642 B]

Get:6 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 libmecab2 amd64 0.996-14build9 [199 kB]

Get:7 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libprotobuf-lite23 amd64 3.12.4-1ubuntu7.22.04.1 [209 kB]

Get:8 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64 mysql-server-core-8.0 amd64 8.0.32-0ubuntu0.22.04.2 [17.5 MB]

Get:9 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64 mysql-server-8.0 amd64 8.0.32-0ubuntu0.22.04.2 [1,427 kB]

Get:10 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 libcgi-pm-perl all 4.54-1 [188 kB]

Get:11 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 libfcgi0ldbl amd64 2.4.2-2build2 [28.0 kB]

Get:12 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 libfcgi-perl amd64 0.82+ds-1build1 [22.8 kB]

done!

update-alternatives: using /var/lib/mecab/dic/ipadic to provide /var/lib/mecab/d

ic/debian (mecab-dictionary) in auto mode

Setting up libcgi-fast-perl (1:2.15-1) ...

Setting up mysql-server-core-8.0 (8.0.32-0ubuntu0.22.04.2) ...

Setting up mecab-ipadic-utf8 (2.7.0-20070801+main-3) ...

Compiling IPA dictionary for Mecab.  This takes long time...

reading /usr/share/mecab/dic/ipadic/unk.def ... 40

emitting double-array: 100% |###########################################|

/usr/share/mecab/dic/ipadic/model.def is not found. skipped.

reading /usr/share/mecab/dic/ipadic/Suffix.csv ... 1393

reading /usr/share/mecab/dic/ipadic/Postp.csv ... 146

reading /usr/share/mecab/dic/ipadic/Noun.name.csv ... 34202

reading /usr/share/mecab/dic/ipadic/Noun.verbal.csv ... 12146

reading /usr/share/mecab/dic/ipadic/Noun.org.csv ... 16668

reading /usr/share/mecab/dic/ipadic/Noun.others.csv ... 151

reading /usr/share/mecab/dic/ipadic/Filler.csv ... 19

reading /usr/share/mecab/dic/ipadic/Noun.demonst.csv ... 120

reading /usr/share/mecab/dic/ipadic/Auxil.csv ... 199

reading /usr/share/mecab/dic/ipadic/Adnominal.csv ... 135

reading /usr/share/mecab/dic/ipadic/Adverb.csv ... 3032

reading /usr/share/mecab/dic/ipadic/Conjunction.csv ... 171

reading /usr/share/mecab/dic/ipadic/Noun.csv ... 60477

reading /usr/share/mecab/dic/ipadic/Verb.csv ... 130750

reading /usr/share/mecab/dic/ipadic/Others.csv ... 2

reading /usr/share/mecab/dic/ipadic/Noun.nai.csv ... 42

reading /usr/share/mecab/dic/ipadic/Noun.adjv.csv ... 3328

reading /usr/share/mecab/dic/ipadic/Noun.proper.csv ... 27328

reading /usr/share/mecab/dic/ipadic/Noun.place.csv ... 72999

reading /usr/share/mecab/dic/ipadic/Symbol.csv ... 208

reading /usr/share/mecab/dic/ipadic/Noun.number.csv ... 42

reading /usr/share/mecab/dic/ipadic/Noun.adverbal.csv ... 795

reading /usr/share/mecab/dic/ipadic/Postp-col.csv ... 91

reading /usr/share/mecab/dic/ipadic/Prefix.csv ... 221

reading /usr/share/mecab/dic/ipadic/Adj.csv ... 27210

reading /usr/share/mecab/dic/ipadic/Interjection.csv ... 252

emitting double-array: 100% |###########################################|

reading /usr/share/mecab/dic/ipadic/matrix.def ... 1316x1316

emitting matrix      : 100% |###########################################|

done!

update-alternatives: using /var/lib/mecab/dic/ipadic-utf8 to provide /var/lib/me

cab/dic/debian (mecab-dictionary) in auto mode

Setting up mysql-server-8.0 (8.0.32-0ubuntu0.22.04.2) ...

update-alternatives: using /etc/mysql/mysql.cnf to provide /etc/mysql/my.cnf (my

.cnf) in auto mode

Renaming removed key\_buffer and myisam-recover options (if present)

mysqld will log errors to /var/log/mysql/error.log

mysqld is running as pid 10658

Created symlink /etc/systemd/system/multi-user.target.wants/mysql.service → /lib

/systemd/system/mysql.service.

Setting up mysql-server (8.0.32-0ubuntu0.22.04.2) ...

Processing triggers for man-db (2.10.2-1) ...

Processing triggers for libc-bin (2.35-0ubuntu3.1) ...

root@ubuntu-22:/etc#

Commands used:

sudo mysql -u root

ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql\_native\_password BY

'password';

sumit@ubuntu-22:~$ sudo mysql -u root

Welcome to the MySQL monitor.  Commands end with ; or \g.

Your MySQL connection id is 23

Server version: 8.0.32-0ubuntu0.22.04.2 (Ubuntu)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its

affiliates. Other names may be trademarks of their respective

owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql\_native\_password BY 'password';

Query OK, 0 rows affected (0.09 sec)

mysql> exit

Bye

sumit@ubuntu-22:~$

sumit@ubuntu-22:~$ sudo mysql -u root -p

Enter password: password

mysql> show databases;

mysql> CREATE DATABASE student-schema;

Query OK, 1 row affected (0.09 sec)

## 16.3 MySQL workbench

<https://dev.mysql.com/downloads/workbench/>

Ubuntu Linux 22.04 (x86, 64-bit), DEB Package 8.0.33 24.5M (mysql-workbench-community\_8.0.33-1ubuntu22.04\_amd64.deb) MD5: 559f0e6d26e6a71e8359b854657e0c60

<https://dev.mysql.com/downloads/file/?id=517978>

Configuration:

Hostname= localhost

Port= 3306

Username= root

Password = password (force set)

A screenshot of a computer

Description automatically generated

## 16.4 MySQL: Create Table:

Create database student\_schema;

CREATE TABLE `student\_schema`.`student` (

`student\_id` INT NOT NULL AUTO\_INCREMENT,

`first\_name` VARCHAR(45) NULL DEFAULT NULL,

`last\_name` VARCHAR(45) NULL DEFAULT NULL,

`email\_id` VARCHAR(45) NULL DEFAULT NULL,

PRIMARY KEY (`student\_id`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_bin;

DROP DATABASE student\_schema;

CREATE database student\_schema;

CREATE TABLE `student\_schema`.`student` (

`student\_id` INT NOT NULL AUTO\_INCREMENT,

`first\_name` VARCHAR(45) NULL DEFAULT NULL,

`last\_name` VARCHAR(45) NULL DEFAULT NULL,

`email\_id` VARCHAR(45) NULL DEFAULT NULL,

PRIMARY KEY (`student\_id`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_bin;

A screenshot of a computer

Description automatically generated

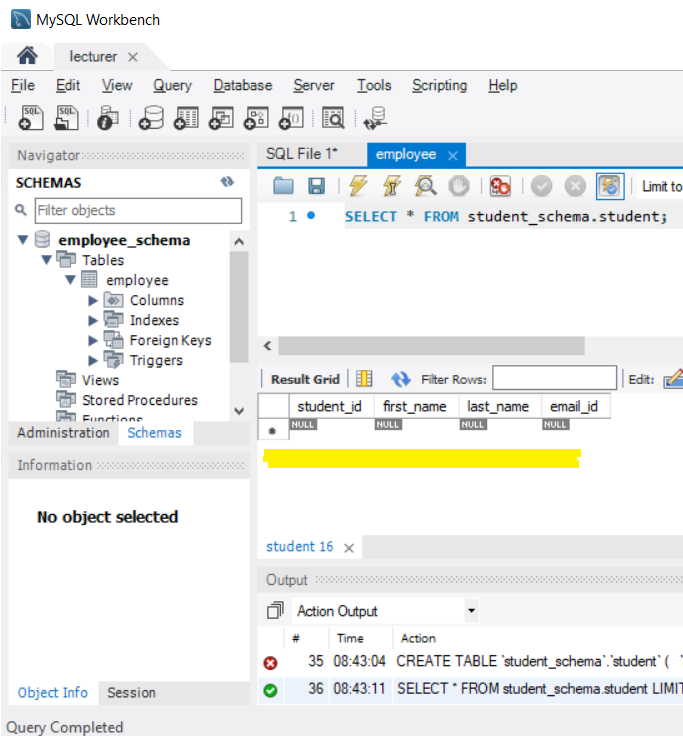
## 16.5 Testing End-Points:

localhost:8080/api/studentservices

## 16.6 Postman: POST(200 OK response)

A screenshot of a computer

Description automatically generated with medium confidence



Graphical user interface, text, application, email

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, text, application

Description automatically generated

## 16.7 Postman: GET (200 OK response)

Alternative locally check in browser: <http://localhost:8080/api/studentservices>

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, text, application, email

Description automatically generated

## 16.8 Postman: PUT/UPDATE (200 OK response)

Postman URL to correct 3rd student record = localhost:8080/api/studentservices/3

Before updating:

Graphical user interface, text

Description automatically generated

After update:

Graphical user interface, text, email, website

Description automatically generated

Graphical user interface, text

Description automatically generated

## 16.9 Postman: DELETE (200 OK response)

Before:

Graphical user interface, text

Description automatically generated

After:

Graphical user interface, text, application, email

Description automatically generated

"status": 405: Not allowed.

Graphical user interface, text, application, email

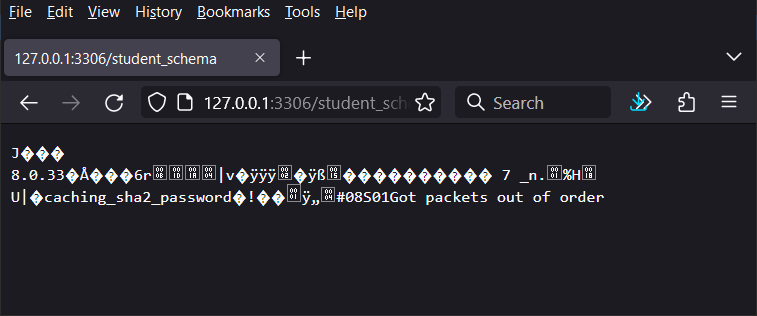
Description automatically generated

Improvement and patch to be delivered in CSIT-697 Master’s REST API project(Banking App).

17.Best Practices and lesson learned

## 17.1 Unknow error

Causing the Springboot server connection problem:



## 17.2 VM

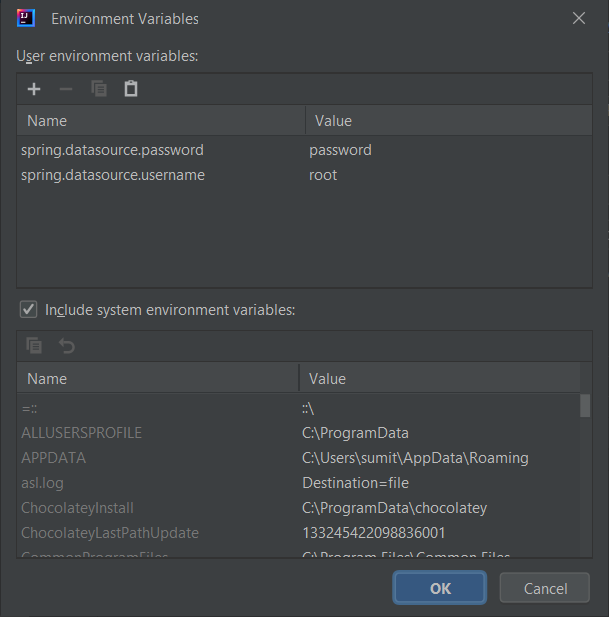
Frequent configuration to support multiple project and multiple instances had damaged the Virtual environment setup. Rebuild was necessary to

## 17.3 Environmental Variable

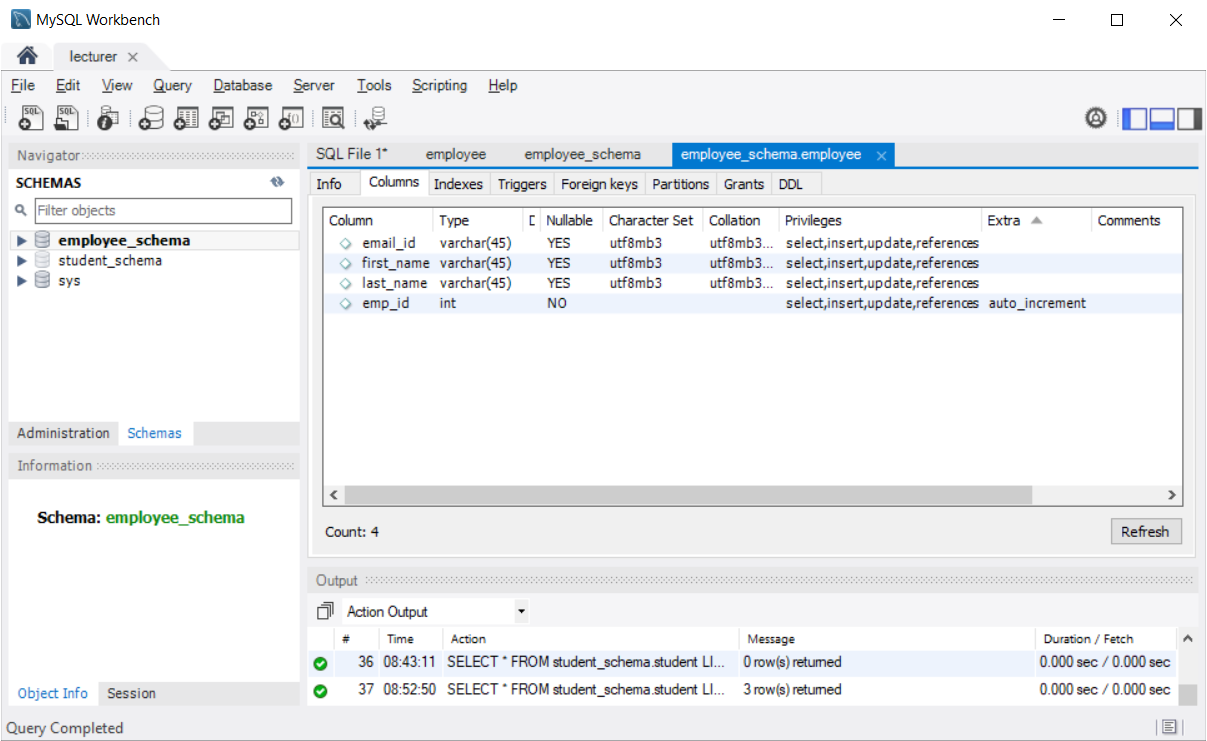
MySQL username and password were now passed through the Environmental variable within IDE rather than providing in src/main/resources/application.properties

Before: src/main/resources/application.properties

Now: Environmental variable



## 17.4 Student\_ID autoincrement



## 17.5 DELETE operation

"status": 405: Not allowed for the MySQL driver: planning to move to PostgreSQL in complete version of REST API within CSIT-697

Graphical user interface, text, application, email

Description automatically generated