

School Data Management

Computer Science and Engineering

By

Aryan Datta 181219

Under the supervision of

Dr. Monika Bharti



Department of Computer Science & Engineering and Information Technology

**Jaypee University of Information Technology, Wahnaghat, 173234,
Himachal Pradesh, INDIA**

DECLARATION

I on my behalf declare that the project that I have done has been made by us under the direction of (Dr Monica Bharti) the Jaypee University of Information Technology. I also announce that neither this project nor any part of this project has been submitted elsewhere for any purpose.

Supervised to:

Dr. Monika Bharti

Department of Computer Science & Engineering and Information Technology
Jaypee University of Information Technology

Submitted by:

Aryan Datta 181219

Computer Science & Engineering
Department Jaypee University of Information
Technology

CERTIFICATE

This is to certify that the project which is being represented in the project report named “School data management” in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science And Engineering and represented to the Department of Computer Science and Engineering (CSE), Jaypee University of Information Technology, Waknaghat is an authentic record of work done by Aryan Datta (181219) during the period from January 2022 to May 2022 under the supervision of Dr Monica Bharti, Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat.

Aryan Datta 181219

The above statement made is correct to the best of my knowledge.

Dr. Monika Bharti
Assistant Professor (SG)
Computer Science & Engineering and Information Technology
Jaypee University of Information Technology, Waknaghat

ACKNOWLEDGEMENT

Firstly, I am so gratefulness to Almighty God and thankful for his divine grace that made it possible to successfully complete the project work in a smooth way.

I am grateful and wish my profound indebtedness to my supervisor Dr. Monika Bharti, who is in department of computer science Jaypee University of Information Technology, Wazirpur, where she was very helpful and guided us throughout the thesis to carry out this project and also provide us with the necessary materials that we need. Her endless scholarly guidance, patience, continual encouragement, constant scholarly guidance and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts, and correcting them at all stages have made it possible to complete this project.

I would like to express my heartiest gratitude and thankful to Dr. Monika Bharti, Department of computer science, for her considerate nature and kind help to finish my project on time.

I would also generously thank each of the individuals who have helped me continuously and straightforwardly or in a roundabout way to make this project a success. In this odd and strange situation, I also want to thank the various staff individuals, both educating and non-educating, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patients of my parents.

Aryan Datta 181219

TABLE OF CONTENT

<i>Content</i>	<i>Page Number</i>
1.Chapter 1 (Introduction)	1-9
2.Chapter 2 (Literature Survey)	10
3.Chapter 3 (System Development)	11-22
4.Chapter 4 (Performance Analysis)	23-25
5. Chapter 5 (Conclusions)	26-32
References	33
Appendix	34

LIST OF FIGURES

Fig No.	Figure Name	Page No.
1.1	Introduction	1
1.2	Problem Statement	2
1.3	Objective	3
1.4	Methodology	4-8
1.5	Organizations	8-9
2.0	Literature Survey	10-11
3.1	Method Used	12-14
3.2	Computational Development	14-22
4	Performance Analysis	23-25
5	Conclusion	26-32

6	Output	34

ABSTRACT

My parents both are teacher .so I have seen many cases in which they would find it difficult to collect the data of different student , teachers and sometime even different schools because in order to save the data they are still using typicall tables with rows and column but this take too much space and a single table for whole school would not be possible. So one day I came across a very great data base which can easily solve this problem which is there for so many years and the data base that I am talking about is mongoDB .

In this project I have used spring boot and mongo DB in order to save and utilize the data in such a way which is fast and effective . This project contains basic CRUD functions and even some complex function which will be helpful in order to find the data of a particular student or even a teacher from the same set of data .

CHAPTER 01

INTRODUCTION

- 1.1 Introduction**
- 1.2 Problem Statement**
- 1.3 Objective**
- 1.4 Methodology**
- 1.5 Organization**

1.1 INTRODUCTION

This project is based on MongoDB along with Spring Boot .MongoDB is a type of database which instead of using typical table use it uses fields and document which help the data to make it more scalable and useful .Spring Boot act as a back door entry to data in ease we can do all the CRUD operations on data in microseconds due to the features provided by the spring boot.

1.2 PROBLEM STATEMENT

Inspired from the hustle that I have seen my dad was doing in order to find the records from school registers I decided that I will use the latest database and latest technology in order to make data easily available in the places where is abundance of data.

Spring Boot and MongoDB are free to use programs so practically speaking all the registers and the tables can be easily replaced with this program running in the administration pc this project require no money and is ready to deploy in any pc one can just run the program.

Creating two java services and perform communication between each other. The functionality should be as follows:

- Service 1 manages details of the school and it's db stores the school name, school's unique code. Class details like class code, class name, student strength, instructor's code
- Service 2 manages user details and stores user details like name, userId, role [STUDENT, INSTRUCTOR], class code(Check class existence using rest call before entering record in db)

The aim is to provide APIs to fetch details of all related entities for any fetched details like if i search based on student id, it should return me his school name, teacher name, class name. Below APIs need to be created:

- /schools :- to fetch details for all schools includes class, student as well as teacher details, add query param on school name\
- /school/{schoolId} : fetch details for a particular school
- /school/{schoolId}/class/{classId}: fetch details for a class in a school
- /users :- fetch all user details including school, class and teacher details
- /user/{userId} : fetch details of a specific user
- /user/{schoolId}/teachers :- to fetch teachers' details for a school including school Details
- /user/{schoolId}/students :- to fetch students' details for a school including school Details

1.3 OBJECTIVE

All in all the main aim to make this application is to make the best use of technology in order to do CRUD operation on the data and make it free and easy to use I have used different technology like maven, swagger to make the app more interactive .This project can be used where the data is in abundance like schools, institutes etc.

1.4 METHODOLOGY

1.4.1 MongoDB

MongoDB, the most popular NoSQL website, is a site based on open source content. The term 'NoSQL' means 'unrelated'. It means that MongoDB is not based on a related table-like website structure but provides a completely different way of storing and retrieving data. This storage format is called BSON (similar to the JSON format).

SQL websites store data in table format. This data is stored in a predefined data model that is not very flexible in today's increasingly real-world systems. Modern applications are more networked, more social and more interactive than ever before. Apps store additional data and access it at higher rates.

The Relational Database Management System (RDBMS) is not a good option when it comes to managing big data due to its structure as it does not measure horizontally. If the site is running on a single server, it will reach the rating limit. NoSQL database is very sensitive and provides high performance. MongoDB is such a NoSQL website that measures by adding multiple servers and increasing productivity with its flexible document model.

```
1  _id: 1546
2  empName: "Aryan "
3  empSalary: "900000 "
4  _class: "com.example.emp.Emp "
```

Have you ever wondered, "How my brain work when I am trying to make application data schema?" It is one of the most common questions developers have about MongoDB. And the answer is, to. This is because a text website is rich in vocabulary that is able to express data relationships in different ways than SQL. There are many things to consider when choosing a schema. Is your app legible or difficult to write? What data is usually accessed together? What are your performance characteristics? How will your data set grow and measure?

In this post, we will discuss the basics of data modeling using real-world examples. You will learn the basic techniques and vocabulary you can use when designing the schema of your website for your application.

Well, first of all, did you know that the design of the appropriate MongoDB schema is the most important part of deploying such a fast, fast, affordable website? It is true, and the schema structure is often one of the most overlooked aspects of MongoDB management. Why is MongoDB Schema Design so important? However, there are a few good reasons. In my experience, most people who come to MongoDB tend to think of the construction of a MongoDB schema as similar to the construction of a legacy related schema, which does not allow you to make full use of all the information provided by MongoDB information. First, let's take a look at how the design of an asset-related website compares to the construction of the MongoDB schema.



1.4.1 Spring Boot

Prior to the introduction of Enterprise Java Beans (EJB), Java developers had to use JavaBeans to build web applications. Although JavaBeans contributed to the development of user interface (UI) components, they were unable to provide the services, such as performance management and security, which were required to develop robust and secure business applications. The arrival of EJB has been identified as a solution to this problem. EJB expands Java components, such as the Web and business components, and provides services that assist in the development of business applications. However, building a business plan with EJB was not easy, as the developer needed to perform a variety of tasks, such as doing Home and Remote interfaces and using the life-cycle repetition methods that lead to EJB code delivery problems. Looking for an easy way to improve business applications.

The Spring Framework has emerged as a solution to all of these problems. This framework uses various new strategies such as Aspect-Oriented Programming (AOP), Plain Old Java Object (POJO), and dependency injection (DI), to improve business plans, thus eliminating them. The complexities involved in building business applications using EJB, Spring is an open source lightweight solution that allows Java EE 7 developers to build simple, reliable, and efficient business applications. This framework focuses on providing a variety of ways to help you manage your business assets. It has made the development of Web applications much easier compared to older Java frameworks and Application Programs (APIs), such as Java website integration (JDBC), JavaServer Pages (JSP), and Java Servlet.

The Spring Framework can be considered as a subgroup of structures, also called layers, such as Spring AOP, Spring Object-Relational Mapping (Spring ORM), Spring Web Flow, and Spring Web MVC. You can use any of these modules separately while building a Web application. Modules can also be grouped together to provide the best performance on a Web application.

Features of Spring Framework

The features of the Spring framework such as IoC, AOP, and performance management, make it unique among the framework frameworks. Some of the most important features of the Spring Framework are the following:

IoC Container:

Refers to the main container that uses the DI or IoC pattern to explicitly provide object reference in class during operation. This pattern serves as another pattern for the service area. The IoC container contains an integral code that handles application configuration management.

The Spring Framework offers two packages, namely `org.springframework.beans` and `org.springframework.context` which assists in providing the functionality of the IoC container.

Data access framework:

Allows developers to use continuous APIs, such as JDBC and Hibernate, to store persistent data on a website. It helps solve various engineer problems, such as how to communicate with a website connection, how to make sure communications are closed, how to deal with differences, and how to implement transaction management. It also enables developers to write easily access code for persistence data throughout the application.

Spring MVC Framework:

Allows you to create web applications based on MVC architecture. All user-made requests first go through the controller and are sent to a different view, that is, to different JSP pages or Servlets. Form management and verified form features of the Spring MVC framework can be easily integrated with all popular viewing technologies such as JSP, Jasper Report, FreeMarker, and Velocity.

Performance Management:

Helps in managing application performance without touching its code. This framework provides the Java Transaction API (JTA) for global applications hosted by application server and local operations managed using JDBC Hibernate, Java Data Objects (JDO), or other data access APIs. It allows the developer to model multiple tasks on the basis of advertising and programmatic management.

Spring Web Service:

Produces final web service points and definitions based on Java classes, but is difficult to manage in the application. To solve this problem, the Spring Web Service provides horizontally controlled alternatives to Extensible Markup Language (XML) (XML learning and deception) strategy. Spring provides an effective map to transfer XML message request to an object and the developer can easily distribute an XML (object) message between two machines.

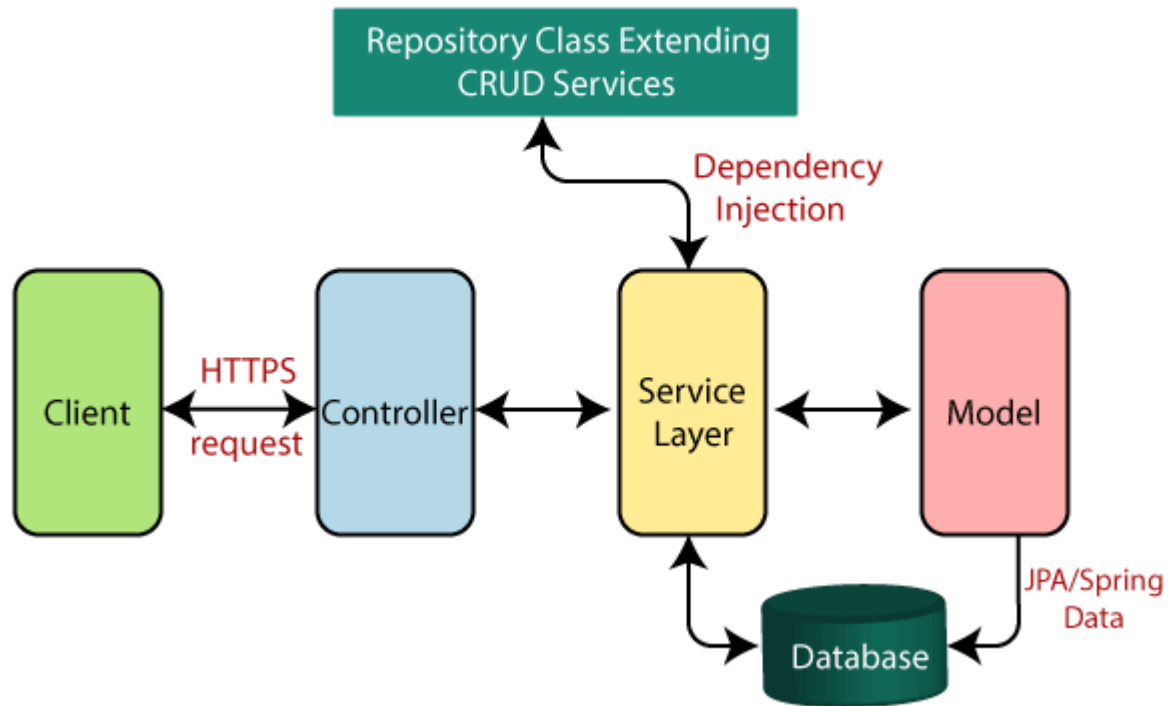
JDBC Abbreviation Layout:

Helps users manage errors easily and effectively. JDBC editing code can be minimized if this removal layer is applied to a web application. This layer handles differently as `DriverNotFoundException`. All `SQLExceptions` are translated into the `DataAccessException` class. Spring data access exceptions are not specific to JDBC and that is why Data Access Equipment (DAO) is not limited to JDBC.

Spring TestContext Frame:

Provides unit resources and integration testing for Spring applications. In addition, the Spring TestContext framework provides direct integration test functions such as context management and DI caching for test fixtures, as well as practical test management with automatic recurring semantics.

Spring Boot flow architecture



1.5 ORGANIZATION

Chapter 1 contains the Introduction, Problem Statement, Objective, Methodology of the Project or System.

Chapter 2 contains the literature survey in which some of the previous works is studied and compised in order to make this model.

Chapter 3 discusses the system developmnt as it contains all the computational, mathematical, analytecal, theoretical data.

Chapter 4 is all about the performnce anelysis in which diffrent models' results is compisd , with algorithm improviment and outputs

Chapter 5 discusses the conclusions, future scope of the model, its applications in real world, advantages and limitations.

Then i have the References section.

Last is the Appendices section which includes the source code of model.

CHAPTER 02

LITERATURE SURVEY

Spring Boot makes it easy to create standalone applications, with a production level that you can "just use".

We take a visual view of the spring stadium and the libraries of foreign companies to get started with a little controversy. Most Spring Boot apps require minimal Spring configuration.

If you are looking for information on a particular version, or instructions on how to improve from a previous release, check out the project release notes section in our wiki.

Features

Create independent spring apps

Embed Tomcat, Jetty or Undertow directly (no need to send WAR files)

Provide a 'first' reliance on ideas to make your construction preparation easier

Automatically customize Spring and foreign libraries whenever possible

Provide ready-to-produce features such as metrics, health tests, and external configurations

There is no code production at all and there is no need for XML configuration

CHAPTER 03

SYSTEM DEVELOPMENT

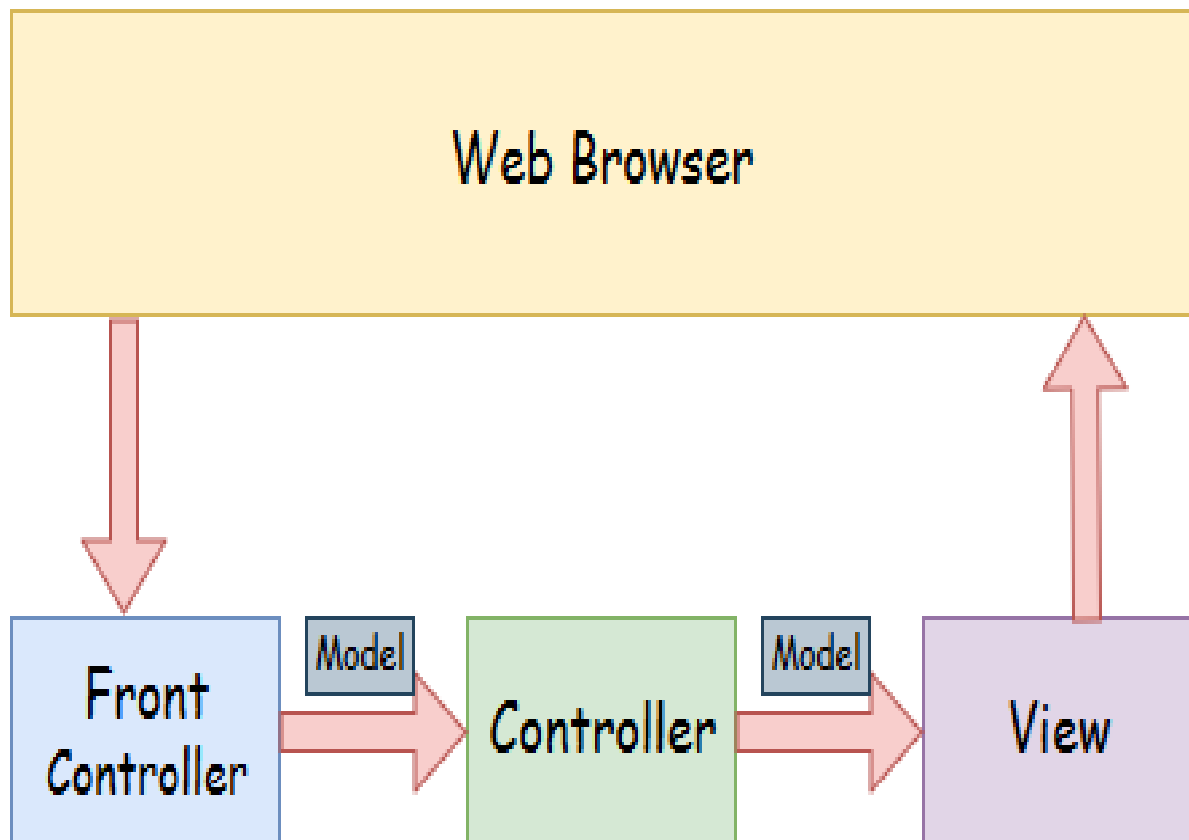
- 3.1 Methods used
- 3.2 Theoretical and Mathematical development
- 3.3 Computational development

3.1 Functions/Methods used

Spring MVC:

Spring MVC is a Java framework used to build web applications. It follows the Model-View-Controller design pattern. It uses all the basic features of the main spring frame such as control modification, dependence injection.

Spring MVC provides a great solution to use MVC in the spring framework with the help of the DispatcherServlet. Here, DispatcherServlet is a class that receives an incoming application and places it on a map that uses the appropriate as controls, models, and views.

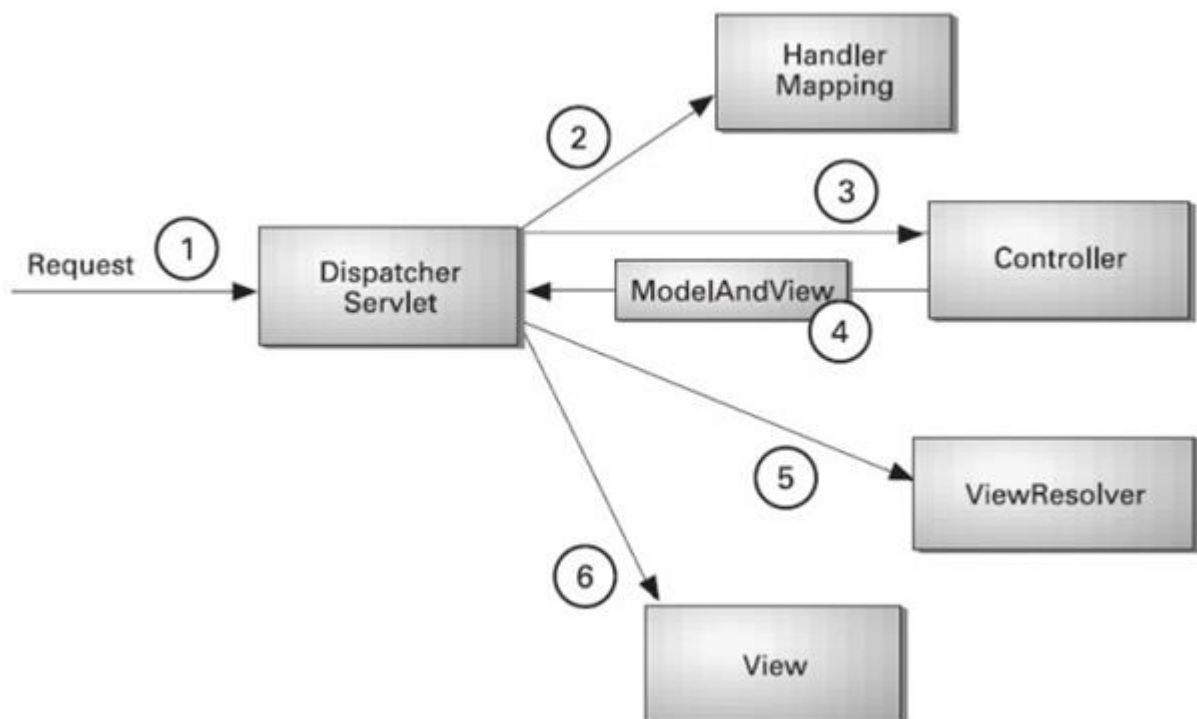


Model - Model contains application data. Data can be a single object or a set of objects.

Controller - The controller contains the business intelligence of the application. Here, the @ Controller annotation is used to mark a category as a controller.

View - View represents information provided in a specific format. Typically, JSP + JSTL is used to create a web page. Although spring also supports other viewing technologies such as Apache Velocity, Thymeleaf and FreeMarker.

Front Control - In Spring Web MVC, the DispatcherServlet section acts as the front controller. There is a responsibility to manage the flow of the Spring MVC application.



As shown in the picture, every incoming request is received by the DispatcherServlet which serves as the front controller.

DispatcherServlet retrieves the map holder from the XML file and transfers the application to the controller.

The controller returns the ModelAndView object.

DispatcherServlet checks the installation of the viewing solution in the XML file and asks for the specified viewing component.

Swaggers:

Swagger lets you define the structure of your APIs for machines to read. The ability of APIs to define their structure is the root of all that is amazing in Swagger. Why is it so big? However, by learning the structure of your API, we can automatically create beautiful and interactive APIs. And we can automatically generate clients for your API clients in multiple languages and explore other possibilities such as automated testing. Swagger does this by asking your API to reinstall YAML or JSON which contains a detailed description of your entire API. This file is actually a list of your API application that adheres to OpenAPI Specification. Details are asked to enter information such as:

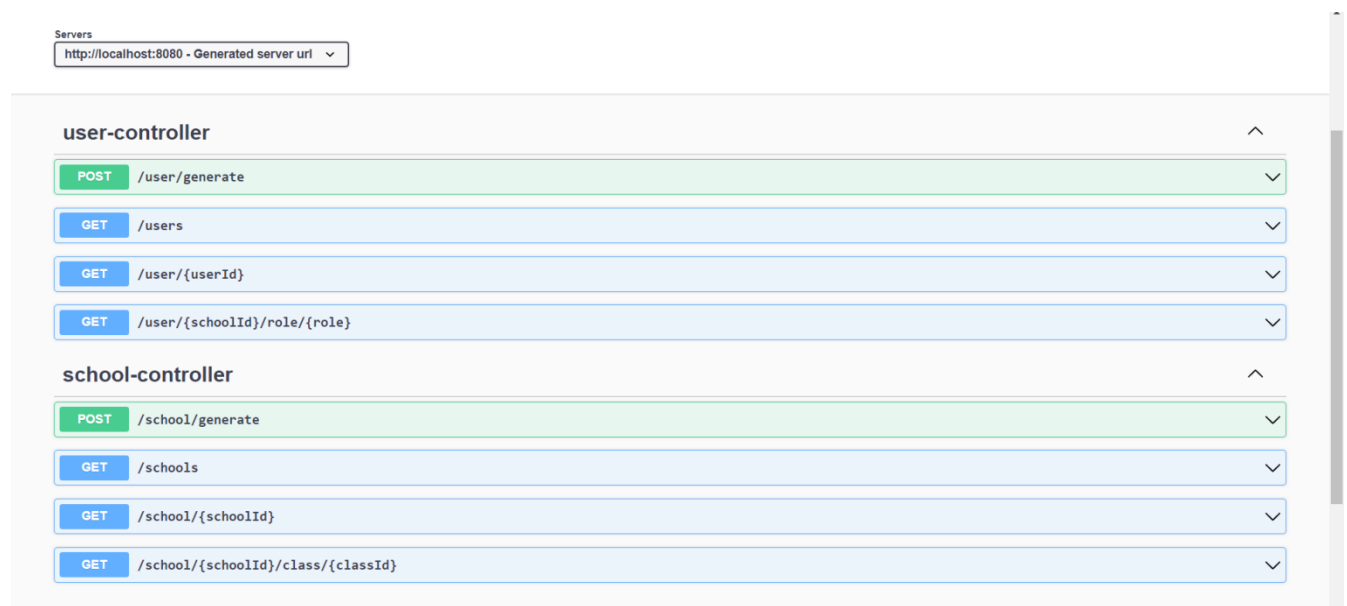
What are all the functions supported by your API?

What are the components of your API and what it returns?

Does your API need some authorization?

And interesting features like terms, contact details and license to use the API.

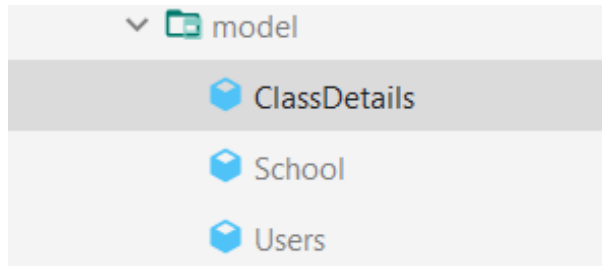
You can write a Swagger spec for your own API, or have it generated automatically from annotations in your source code. Check out swagger.io/open-source-integrations for a list of tools that allow you to generate Swagger code



3.2 Computational Development

3.2.1 Code Breakage

- Model



We have three different POJO classes(in spring boot POJO classes means plain old java object which means its simple a class of entity) in my project which is

1. ClassDetails POJO::

```
ClassDetails.java x
1  package com.student.model;
2
3  import lombok.Data;
4
5  @Data
6  public class ClassDetails {
7      private String classCode;
8      private String className;
9      private Integer studentStrength;
10     private String instructorCode;
11 }
12
```

It contains the basic features of a class like

2. School POJO::

```
School.java x
1  package com.student.model;
2  import org.springframework.data.mongodb.core.mapping.Document;
3  import lombok.Data;
4  import lombok.Getter;
5  import lombok.Setter;
6  @Document("school")
7  @Data
8  @Getter
9  @Setter
10 public class School {
11     private String schoolId;
12     private String schoolName;
13     private ClassDetails classDetails;
14 }
15
```

3. Users POJO::



```

1  package com.student.model;
2
3  import org.springframework.data.mongodb.core.mapping.Document;
4
5  import lombok.Data;
6
7  @Data
8  @Document("users")
9  public class Users {
10     private String userId;
11     private String userName;
12     private String role;
13     private String classCode;
14     private String schoolId;
15 }
16

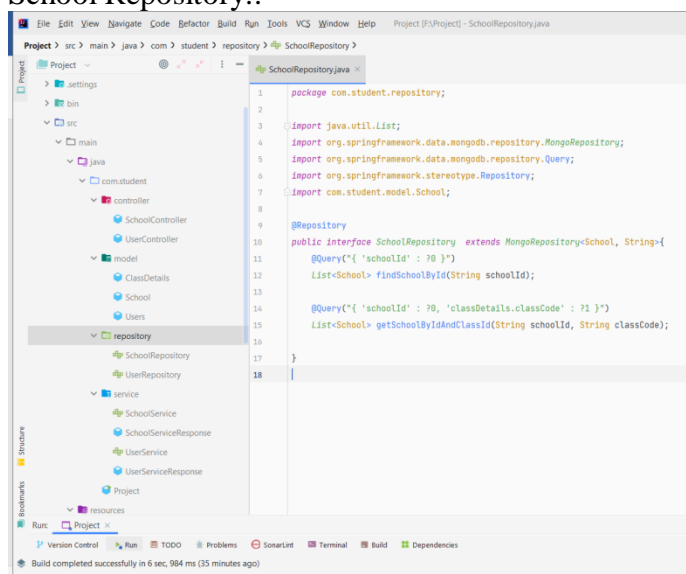
```

- Repository

@Repository is an Spring annotation that shows that a decorated classroom is a repository. A repository is a method of combining storage, retrieval, and search behavior that mimics a collection of items. An @Component annotation technology that allows startup classes to be automatically detected by class-based scanning.

@ComponentScan ensures that classes adorned with @Component and its outlets including @Repository are available and registered as Spring Beans. @ComponentScan is automatically installed with @SpringBootApplication.

School Repository::

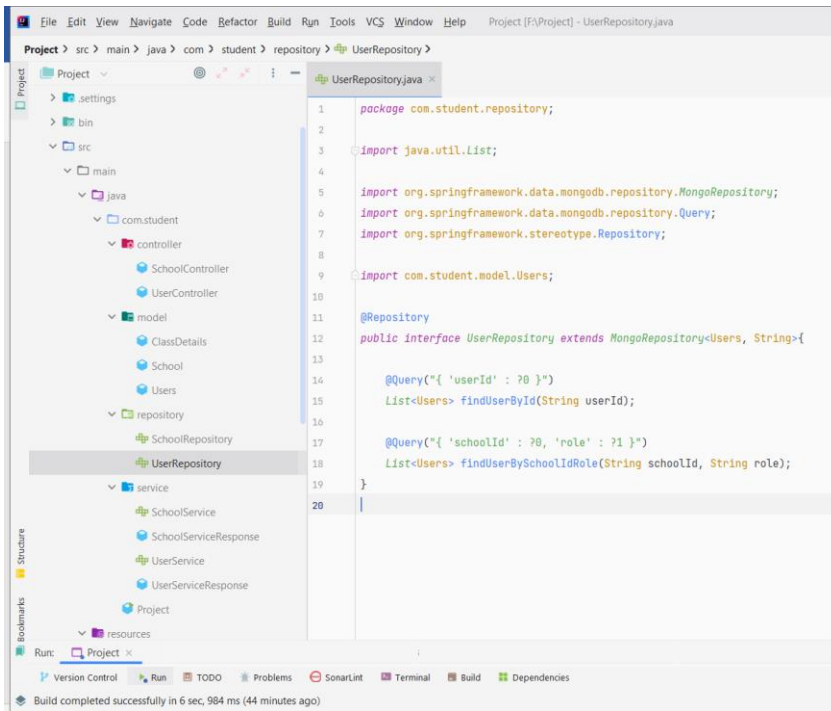


```

1  package com.student.repository;
2
3  import java.util.List;
4  import org.springframework.data.mongodb.repository.MongoRepository;
5  import org.springframework.data.mongodb.repository.Query;
6  import org.springframework.stereotype.Repository;
7  import com.student.model.School;
8
9  @Repository
10 public interface SchoolRepository extends MongoRepository<School, String>{
11     @Query("{ 'schoolId' : ?0 }")
12     List<School> findSchoolById(String schoolId);
13
14     @Query("{ 'schoolId' : ?0, 'classDetails.classCode' : ?1 }")
15     List<School> getSchoolByIdAndClassId(String schoolId, String classCode);
16 }
17
18

```

User Repository::



Services::

The Spring boot service component is defined as a class file that includes an @ Service annotation and allows developers to add business functions. Annotation is also used for classes that provide these business functions. These sections are automatically detected in the spring context where annotation-based setting is used as well as a class-based scan as it provides a special @Component annotation. The function performed as part of the annotations is provided as a visual connector and is independent of the model without the integration mode. This adjective is a general purposeful belief, and engineers may choose to limit the semantics and apply the same as it seems appropriate.

How does it work?

Here we will see the performance of the service component in the Spring Boot, but before that, we need to know not only the value of the service component but also the value of the component of the @Service. It is very important for us to know that standard applications contain different layers, for example, data access, data presentation, services and business, and in each of these layers, there are different beans that allow automatic discovery of layers. @Service annotation helps define classes in the service layer. As mentioned earlier, the @Service component provides the @Component feature. Now, this makes it even more interesting for us to understand that the @Component annotation is used in all applications so that the bean is marked as part of the Spring, and the bean labeled @Service occurs in the business concept component and is therefore only used in the service layer due to additional specialization.

The service component ensures that the class file that includes business ideas is in a separate layer and keeps it separate from the @RestController class file. With the application of the @ Service annotation, we need to add spring core dependencies. During the early spring release, all the beans were announced in the XML file, but considering the stiffness indicator, this becomes a herculean function, so the @Component annotation provides an annotation-based injection with a Java-based configuration. Thus, having an injection based on the definition reduces the declaration of a bean as a 'bean marker'. In addition to @Service, which is mostly used in the service layer, @ Controller is mostly used in the presentation layer and the @ Repository Data Access or DAO or Persistence layer.

We will now look at the performance of a service component using an example. We will announce an interest rate calculator service in the event of a bank deposit. Initially, we will build a service layer, which will contain a service section that will contain the required function. These activities can be simple interests and / or lucrative activities combined. The class will be defined as @Service so that the spring context can detect it automatically, and the pattern can be strengthened from the context. We will now describe a new class that will contain an example of a service class. Application context confirmed. The context.scan () function will now look at all the annotations, and the required annotation will be registered. Now, according to an @ Service annotation, we're building a bean for a custom-made service category. This bean will be a model that will be used to summon any activity within the classroom. Once the required tasks are performed, the application context is closed. Bean suspension can be achieved by transparent distribution.

In my project I have used two different services one for the User and the second one for the school.

```
Project > src > main > java > com > student > service > SchoolServiceResponse >
Project
  model
    ClassDetails
    School
    Users
  repository
    SchoolRepository
    UserRepository
  service
    SchoolService
    SchoolServiceResponse
    UserService
    UserServiceResponse
    Project
  resources
    static
    templates
    application.yml
  test
  target
  .classpath
  .factorypath
  .gitignore
  .project
Run: Project
Version Control Run TODO Problems SonarLint Terminal Build Dependencies
Build completed successfully in 6 sec, 984 ms (55 minutes ago) Terminal Alt+F12 35:1 CRLF UTF-8 Tab* Atom One Light (Material)
```

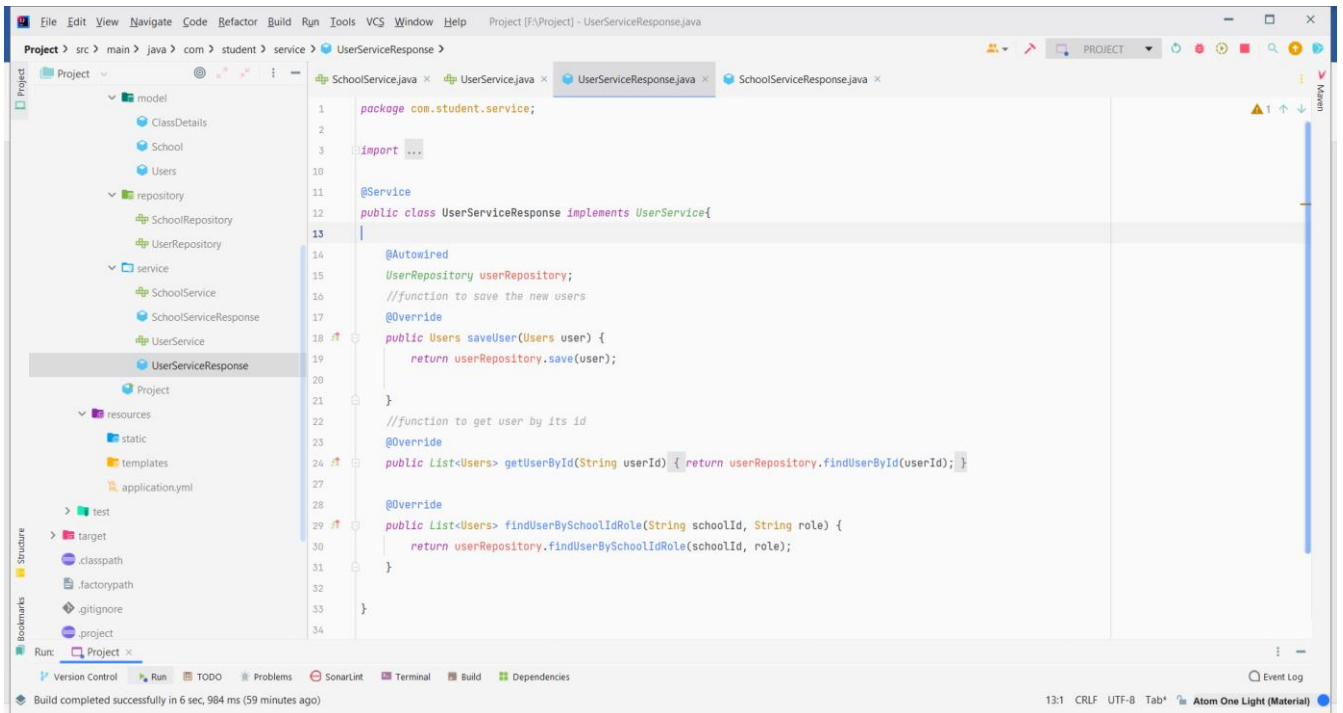
```
7 import com.student.model.School;
8 import com.student.repository.SchoolRepository;
9
10 @Service
11 public class SchoolServiceResponse implements SchoolService {
12
13     @Autowired
14     SchoolRepository schoolRepository;
15
16     //function to save the school information
17     @Override
18     public School save(School school) { return schoolRepository.save(school); }
19
20     //function to find the school by its id
21     @Override
22     public List<School> getSchoolById(String schoolId) { return schoolRepository.findSchoolById(schoolId); }
23
24     //function to find the schoolId and classId
25     @Override
26     public List<School> getSchoolByIdAndClassId(String schoolId, String classId) {
27         return schoolRepository.getSchoolByIdAndClassId(schoolId, classId);
28     }
29 }
30
31
32
33
34
35
```

This is school service response which is a response for interface of the school

```
Project > src > main > java > com > student > service > SchoolService >
Project
  model
    ClassDetails
    School
    Users
  repository
    SchoolRepository
    UserRepository
  service
    SchoolService
    SchoolServiceResponse
    UserService
    UserServiceResponse
    Project
  resources
    static
    templates
    application.yml
  test
  target
  .classpath
  .factorypath
  .gitignore
  .project
Run: Project
Version Control Run TODO Problems SonarLint Terminal Build Dependencies
Build completed successfully in 6 sec, 984 ms (55 minutes ago) Terminal Alt+F12 35:1 CRLF UTF-8 Tab* Atom One Light (Material)
```

```
1 package com.student.service;
2
3 import java.util.List;
4
5 import com.student.model.School;
6
7
8 public interface SchoolService {
9
10     public School save(School school);
11
12     public List<School> getSchoolById(String schoolId);
13
14     public List<School> getSchoolByIdAndClassId(String schoolId, String classId);
15
16 }
17
```

And now for the user we have another service in place which is also a response of a interface why I made a interface? It is because as a developer if someone want to go through my code he/she will be easily able to find the functions and all.



Controller::

In the Spring Boot, the control section is responsible for processing incoming REST API requests, modifying the model, and retrieving feedback to be provided in response.

The spring categories are explained to `@Controller` or `@RestController` annotation. These control sections mark the application holder to allow Spring to be seen as a RESTful service during operation.

In this tutorial, we will include a description of `@Controller` and annotations of `@RestController`, their usage conditions, and the differences between the two annotations.

Spring `@Controller` annotation is also an `@Component` annotation. The `@Controller` annotation indicates that a particular category uses the control role. The Spring Control Annotation is often used in conjunction with the capture methods with annotations based on the `@RequestMapping` annotation. It can only be used in classrooms. Used to mark a class as a web application host. It is widely used with Spring MVC applications. This annotation serves as an example of an annotation section, which illustrates its role. The dispatcher scans such classes with annotations in the ways they are mapped and finds annotations `@RequestMapping`. Let's understand all of this by example.

The process

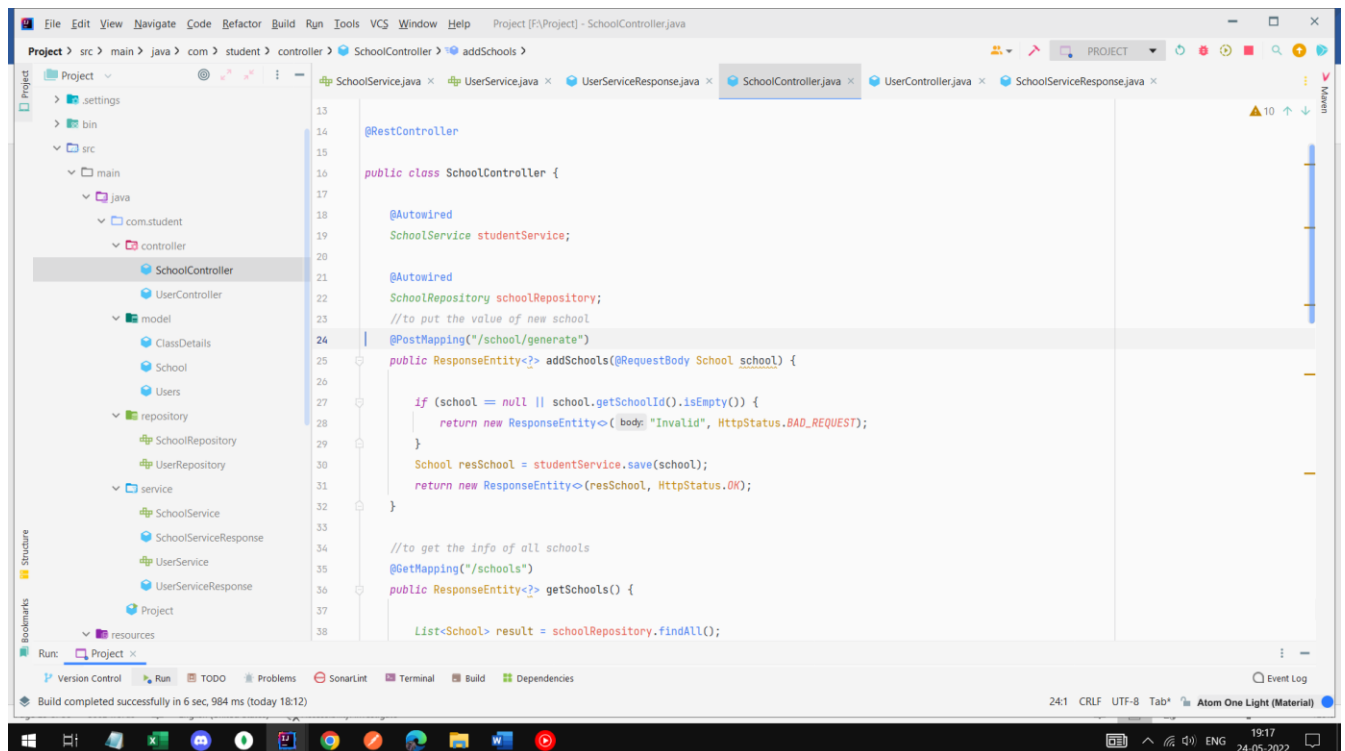
Create a simple Spring Boot project

Add spring-web dependence to your pom.xml file

Create one package and name it “controller”

Create a class within the package

Use our application within the DemoApplication.java file



- **Features provided by my controllers of School::**

1. @PostMapping("/school/generate")

```
public ResponseEntity<?> addSchools(@RequestBody School school) {
    if (school == null || school.getSchoolId().isEmpty()) {
        return new ResponseEntity<>("Invalid", HttpStatus.BAD_REQUEST);
    }
    School resSchool = studentService.save(school);
    return new ResponseEntity<>(resSchool, HttpStatus.OK);
}
```

This is one of the major function this will help to enter the collection of entities to the data that we are using in our project.

2. @GetMapping("/schools")

```
public ResponseEntity<?> getSchools() {
    List<School> result = schoolRepository.findAll();
    if (result.isEmpty()) {
        return new ResponseEntity<>("No Data Found", HttpStatus.OK);
    }
    return new ResponseEntity<>(result, HttpStatus.OK);
}
```

This feature will help to abstract all the information of the schools from the database that we have used.

3. @GetMapping("/school/{schoolId}")

```
public ResponseEntity<?> getSchoolById(@PathVariable("schoolId") String
schoolId) {
    if (schoolId.isEmpty()) {
        return new ResponseEntity<>("School Id Can not be empty",
HttpStatus.BAD_REQUEST);
    }

    List<School> result = studentService.getSchoolById(schoolId);
    if (result == null || result.isEmpty()) {
        return new ResponseEntity<>("No data found", HttpStatus.OK);
    }
    return new ResponseEntity<>(result, HttpStatus.OK);
}
```

This feature will help in order to abstract the exact information of a school that we want.

4. @GetMapping("/school/{schoolId}/class/{classId}")

```
public ResponseEntity<?>
getClassByIdAndSchoolId(@PathVariable("schoolId") String schoolId,
@PathVariable("classId") String classId)
{
    if (schoolId.isEmpty()) {
        return new ResponseEntity<>("School Id Can not be empty",
HttpStatus.BAD_REQUEST);
    }
}
```

```

        if (classId.isEmpty()) {
            return new ResponseEntity<>("Class Id can not be empty",
                HttpStatus.BAD_REQUEST);
        }

        List<School> result = studentService.getSchoolByIdAndClassId(schoolId,
            classId);
        if (result == null || result.isEmpty()) {
            return new ResponseEntity<>("No data found", HttpStatus.OK);
        }
        return new ResponseEntity<>(result, HttpStatus.OK);
    }
}

```

This feature will help in abstracting the value of a particular class from a particular school

- **Features provided by my controllers of User::**

1. **@PostMapping("/user/generate")**

```

public ResponseEntity<?> addUser(@RequestBody Users user) {

    if (user==null || user.getUserId().isEmpty()) {
        return new ResponseEntity<>("Invalid Input",HttpStatus.BAD_REQUEST);
    }

    Users resSchool =userService.saveUser(user);
    return new ResponseEntity<>(resSchool, HttpStatus.OK);
}

```

This function of the project will help us in order to generate a new user in the data we can say that this feature will work as a gateway to enter the data

2. **@GetMapping("/users")**

```

public ResponseEntity<?> getUser() {

    List<Users> result =userRepository.findAll();
    if(result.isEmpty()) {
        return new ResponseEntity<>("No data found", HttpStatus.OK);
    }
    return new ResponseEntity<>(result, HttpStatus.OK);
}

```

This function will help us in order to get the data of all the users that are present in the database of a particular institute or an organistaion.

3. **@GetMapping("/user/{userId}")**

```

public ResponseEntity<?> getUserById(@PathVariable("userId") String userId) {

    if (userId.isEmpty()) {
        return new ResponseEntity<>("userId Can not be null",
            HttpStatus.BAD_REQUEST);
    }

    List<Users> result= userService.getUserById(userId);
    if(result==null || result.isEmpty()) {
        return new ResponseEntity<>("No data found", HttpStatus.OK);
    }
    return new ResponseEntity<>(result, HttpStatus.OK);
}

```

This feature of the user controller will help us in order find a particular user from the database by only using the userid that the one need to find.

4. @GetMapping("/user/{schoolId}/role/{role}")

```
public ResponseEntity<?> getUserByRole(@PathVariable("schoolId") String schoolId,
@PathVariable("role") String role) {

    if (schoolId.isEmpty()) {
        return new ResponseEntity<>("School Id Can not be null",
HttpStatus.BAD_REQUEST);
    }

    List<Users> result= userService.findUserBySchoolIdRole(schoolId, role);
    if(result==null || result.isEmpty()) {
        return new ResponseEntity<>("No record found", HttpStatus.OK);
    }
    return new ResponseEntity<>(result, HttpStatus.OK);
}
```

This feature of the user controller will help the one in order to find the user according to school and the role that he/she is working on.

The Main JAVA class that will help to run the program

```
package com.student;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication
@ComponentScan(basePackages = { "com.student" })
public class Project {

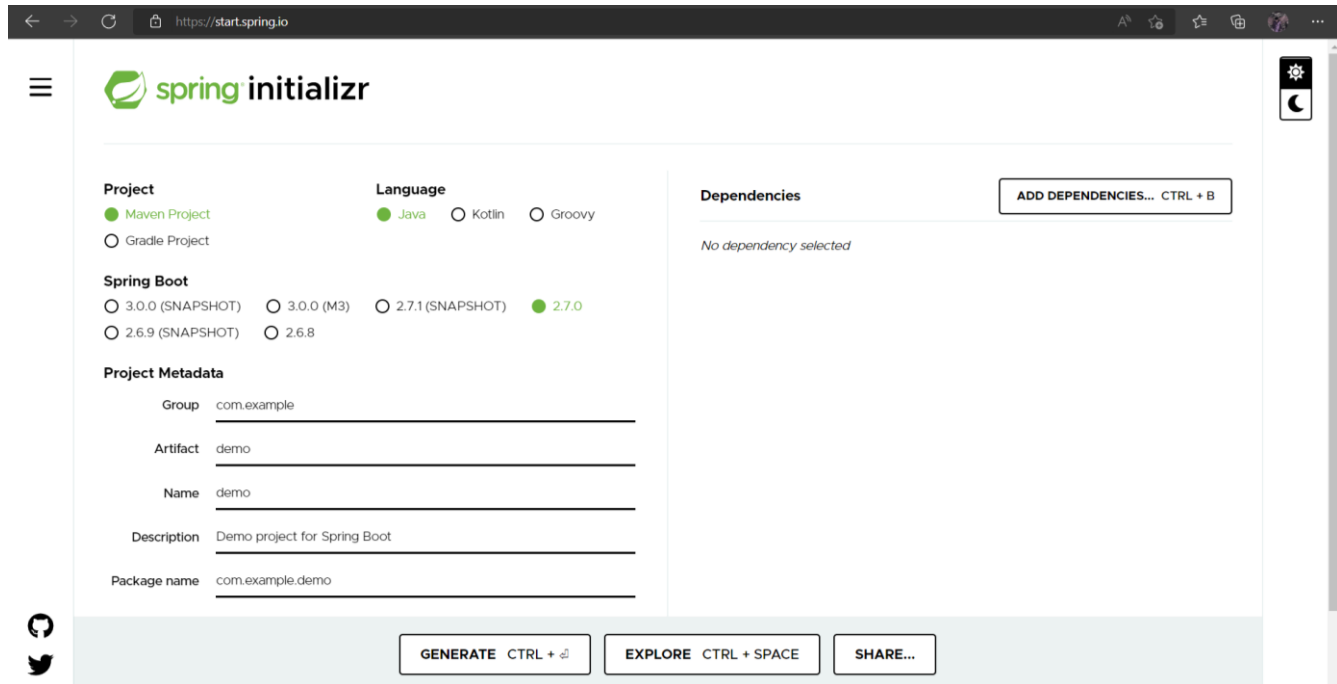
    public static void main(String[] args) {
        SpringApplication.run(Project.class, args);
    }

}
```

CHAPTER 04

PERFORMANCE ANALYSIS

Spring Initializr ::



The screenshot shows the Spring Initializr web application interface. The browser address bar displays <https://start.spring.io>. The page features the Spring Initializr logo and a hamburger menu icon on the left. The main content area is divided into several sections:

- Project:** Includes radio buttons for **Maven Project** (selected) and **Gradle Project**.
- Language:** Includes radio buttons for **Java** (selected), **Kotlin**, and **Groovy**.
- Spring Boot:** Includes radio buttons for versions: **3.0.0 (SNAPSHOT)**, **3.0.0 (M3)**, **2.7.1 (SNAPSHOT)**, **2.7.0** (selected), **2.6.9 (SNAPSHOT)**, and **2.6.8**.
- Project Metadata:** Includes input fields for **Group** (com.example), **Artifact** (demo), **Name** (demo), **Description** (Demo project for Spring Boot), and **Package name** (com.example.demo).
- Dependencies:** Includes a button **ADD DEPENDENCIES... CTRL + B** and the text *No dependency selected*.

At the bottom, there are three buttons: **GENERATE CTRL + G**, **EXPLORE CTRL + SPACE**, and **SHARE...**. Social media icons for GitHub and Twitter are also present on the left side of the bottom bar.

The project started out as a small HTML form, which allows you to generate a Spring Boot app with a few options. Welcome to the Spring community, so the team has added new ways to contact you, with your command line or your favorite IDE.

After a while, we decided to work on updating the original UI. We simplified the form, omitted key sections in the front and middle and pushed some into the automatically hidden section. In view of the dependency value, we created a search box to check for available startups.

The project is now a popular service even with a library that you can use and expand according to your needs. These days, the Web UI is not even the most popular client, but we are still committed to working and improving the service.

The Spring Initializr team has been tempted many times to add new features and options, or solve new problems. Do we have to build complex structures to work on project limitations? Should I add more ideas on how to build, distribute or use your Spring Boot app in production? Should we convert it to the original Spring Boot market place?

With this new update, the design often had a (much needed) update and we chose not to show the full list of available dependencies. One can see that list as a function in terms of bad project metadata (we need to improve that!) Or the lack of accessibility on the spring.io website (we need to fix that!).

While recognizing your current limitations, we feel that “promptly updating the newly installed Spring Boot system” is still the main promise of our service and developers should know what kind of application they want to build when creating a project.

However, we think the Spring Boot team can do more to help developers upgrade and update their applications, so we are working to expand what Spring Initializr can do (see previous blog posts).

This new update is a new step towards additional changes on start.spring.io

MongoDB::

While the MongoDB text model provides the flexibility and precise API that developers love, MongoDB information self-control is difficult, time-consuming, and expensive, especially as the application scale. AWS has created Amazon DocumentDB (compatible with MongoDB) as a fully-fledged and compliant MongoDB text messaging service that allows you to use your existing MongoDB drivers, MongoDB clients, and tools with Amazon DocumentDB.

As a fully-fledged AWS data service, Amazon DocumentDB allows you to set up, protect, and measure MongoDB-related information in the cloud without having to worry about storing and pasting web software setting up and protecting web collections, using collection management software, configuration, backups, and monitoring of production workloads.

You can transfer MongoDB workloads to Amazon DocumentDB using the AWS Database Migration Service (AWS DMS) and command line services like `mongodump` and `mongorestore`.

Benefits of using MongoDB Tasks

Amazon DocumentDB supports the MongoDB API, and there are some benefits and advantages of using your MongoDB load on Amazon DocumentDB.

Scalability

Amazon DocumentDB separates storage by counting, allowing each one to measure independently so you can easily measure reading volume up to millions of requests per second. You can increase your reading capacity to millions of requests per second by adding up to 15 readings per minute, regardless of the size of your data.

High Availability and Stability

Amazon DocumentDB is built for 99.99% availability and duplicates six copies of your data in all three AWS Availability Zones (AZs). In Amazon DocumentDB, continuous backup is automatically enabled, providing 1 day point-in-time recovery (PITR).

Safety and Compliance

Amazon DocumentDB runs on Amazon VPC, allowing you to separate your collection from your virtual network. Amazon DocumentDB supports decryption encryption via AWS KMS, transaction encryption, partial access control, and compliance certificates including PCI DSS, ISO 9001, 27001, 27017, and 27018, SOC 1, 2 and 3, and HITRUST, moreover, HIPAA eligibility.

AWS Service Integration

Amazon DocumentDB easily integrates with other AWS services to expand performance, including Amazon CloudWatch monitoring and alarms, AWS CloudTrail search logs, AWS Glue ETL, and AWS.

Performance Of Spring Boot::

It has sometimes been suggested that the Spring and Spring Boot are "heavy", perhaps because they allow apps to hit more than their own weight, providing many features of the user code that are not too heavy. In this article we focus on memory usage and ask if we can measure the effect of using Spring? In particular we would like to know more about the actual overhead of using Spring compared to other JVM applications. We start by building a basic app with Spring Boot, and look at a few different ways to measure when it works. Then we look at some comparison points: transparent Java applications, applications that use Spring but not Spring Boot, applications that use Spring Boot but do not have automatic configuration, and other sample Ratpack applications.

Vanilla Spring Boot App

As a base we build a consistent application with a few webjars and `spring.resources.enabled = true`. This is great for rendering good looking content that probably has a REST endpoint or two. The source code for the app we used to test is on github. You can build it with mvnw wrapper script if you have JDK 1.8 available also on your way (mvnw package). It can be presented as follows:

```
$ java -Xmx32m -Xss256k -jar target / demo-0.0.1-SNAPSHOT.jarCOPY
```

If we add a load, just to warm up the series pools and force all code methods to be used:

```
$ ab -n 2000 -c 4 http: // localhost: 8080 / COPY
```

We can try and limit the series a bit in `application.properties`:

```
server.tomcat.max-threads: 4COPY
```

but in the end it does not make much difference to the numbers. We conclude with the analysis below that it will save at least MB in the size of the stack we use. All the Spring Boot webapps we review have this same configuration.

We may need to worry about the size of the class, in order to estimate what is happening in memory. Despite some online claims that JVM memory makes a map of all the pots in the classpath, we actually do not find any evidence that the size of the classpath has any effect on the operating system. For reference, the size of the dependable jars (excluding JDK) for a vanilla sample is 18MB:

```
$ jar -tvf target / demo-0.0.1-SNAPSHOT.jar | grep lib /.*. jar | awk '{tot += $ 1;} QEDA {print tot}'  
18893563 COPY
```

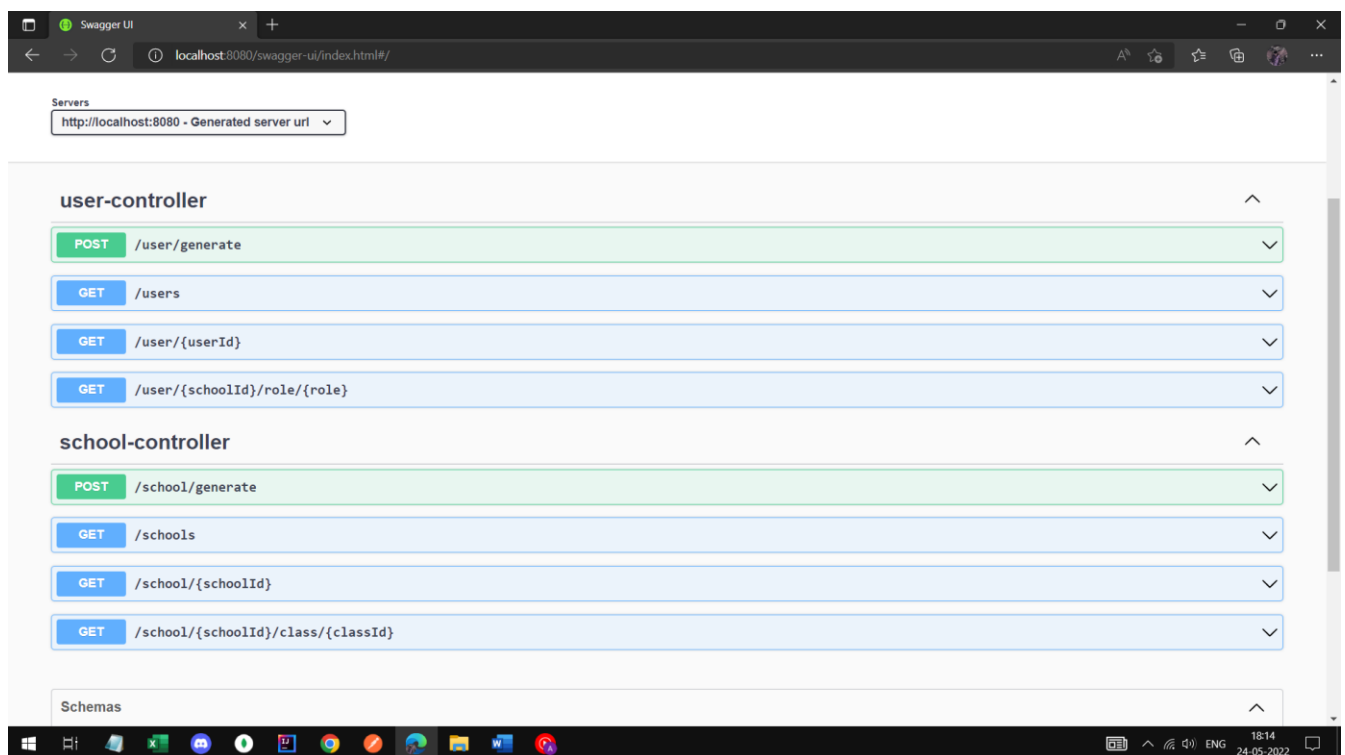
This includes Spring Boot and Actuator launchers, as well as 3 or 4 webjars for static services and a web finder. A small Spring Boot app that combines Spring and some cutting but no web server with about 5MB of pots.

Swagger::

Swagger is a powerful yet easy-to-use API developer tools for teams and individuals, allowing for the development of the entire API life cycle, from design and writing, to testing and implementation.

Swagger contains a mix of open source, free and commercially available tools that allow anyone, from technology engineers to smart street product managers to create amazing APIs that everyone loves.

Swagger is made up of SmartBear Software, which is one of the leading software tools for teams. SmartBear is behind the big names in the software space, including Swagger, SoapUI and QACComplete.



This will help any person to use CRUD operations on the data without using the coding part.

CHAPTER 05

CONCLUSIONS

5.1 Conclusion

5.2 Applications

5.3 Future Scope

5.4 Advantages

5.5 Limitations

5.1 Conclusion

This project presented in this thesis not only provide the better way to store the data but it also help us in making the project more accesable for the people who are using this outside the computer science domain . This project will help us in doing basic CRUD functions and also there are many complex functions that can be done on the mongoDB data.

MongoDB is a collection .So its not like a table cause we don't have to make relation database in this so that we can save the time in it.

5.2 Applications

This project can have numerous applications in real world.

As can be seen from the example above, instead of storing Students and Subjects in separate databases such as in SQL, here at MongoDB, we are storing them together in a single "text". This speeds up the process of data acquisition. Most large apps like Facebook require this small increase in performance because billions of users use the app and therefore, a small increase in performance can have a huge impact on the whole. This is one of the most important benefits of using MongoDB.

Other benefits include:

Integration: MongoDB allows data sharing across all nodes in the collection to ensure that there is no single point failure on the stored server.

Support for Second Indicators: MongoDB allows not only the primary index but also the second important index for most operating systems.

Caching: MongoDB saves a lot of data to allow faster detection of query results.

A nice feature set: There are various features in MongoDB (Ad-hoc queries, Indicator, Multiplication, Rate Launch, File Storage, Merge, JavaScript extrusion server, snow clusters, etc.) that make it an easy-to-use website.

1. API testing. Also, it's really comforting for devs to look at their codes, which work in the UI. It was easy to understand things, especially when I returned to the project after a short break. Activating

commands from the Swagger UI has always been easier or more understandable.

2. The Swagger UI helped business people, such as my product owner, gain insight into the operation of the system, as it was developed over time. Obviously, you can't always expect a business person to set up a code on their system or check the API response in a browser by using a local code. I always added a url to the swagger UI and he could easily feed the parameters to the UI and check the response system came back with. Very simple and easy I can say.

Having a Common API Design for Different Teams

Swagger goes to APIs for what writing software framework is.

As a framework it helps to establish a software development process. Swagger provides global standards with OpenAPI, which allows for the creation of a universe in APIs.

Different people have different styles of writing software and if it were not for frameworks and standards. Keeping software written by any other engineer can be worse than a nightmare.

Swagger specifies standard behavior levels, patterns and RESTful interface for APIs. Different microservices teams in the organization do not have to break their heads to understand, feed and modify external APIs.

Swagger provides a Swagger Hub tool with a built-in API configuration tool that integrates your code with organizational design guidelines.

With this, it does not matter if you have a group of 3 or 300. Things are always simple.

What is SwaggerHub?

SwaggerHub is a platform for designing and documenting design APIs with Open API.

Helps manage within different groups and projects by creating folders with different APIs and approval levels for better planning. Items can be shared with authorized and responsible business executives. It helps developers to work with stakeholders, to implement new changes, to get changes updated, to integrate things. Have an ongoing conversation and follow the news.

SwaggerHub provides custom design models that can be stored in dedicated stores called domains and can be referenced and reused across code. When you write backend code our APIs interact with a few other background services. Instead of shooting the events of those, SwaggerHub enables us to mock those APIs that make rapid progress.

API Enhancing Features

When writing APIs there are many things you can fix such as error management, code modularity, compliance agreements and things. Swagger provides us with quick coding tools for our APIs that take care of all these things. Using swagger takes the fastest route to create an API and go to removable code.

Swagger open source tool CodeGen generates boilerplate code when compiling API. Empowering an engineer to focus on a business genius instead of investing in writing materials.

CodeGen takes care of transparent boilerplate pipes and codes. Just like the Spring Boot project does in the Spring-based app.

Creating API Documents

After doing with the writing code. Another major, tedious task for engineers is to duplicate their codes. My head hurts when I think about it.

Swagger allows us to separate by producing and storing API documents. It saves a lot of time !! Also, we do not have to go back and change the documents every time the code changes.

All documents are automatically updated by Swagger. We can also produce different versions of the API according to our needs and desires. We may also configure Swagger to generate existing API scripts.

API Testing with Swagger

So, Men !! Now our API is ready. Let's move on to testing.

OpenAPI Spec enables us to perform effective, efficient and secure testing of our API. ReadyAPI forum helps to run API tests continuously.

With Swagger Checker, developers can test the application and response API to make sure they are working as expected. The tool also helps in retrospective testing after a major code change.

Swagger has features that allow for automatic API testing, funny APIs, that generate load testing. API stress test. Simulating corners such as high traffic conditions, communication and external data.

Monitoring API in Production

Once the code has been used in production it should be noted for consistent behavior. With Swagger, we can monitor our production APIs for availability, speed and performance.

It helps us track API behavior by ensuring that it works properly and generates a warning if we see anything wrong. With the Swagger API, we can systematically manage the API monitoring process.

Write from the beginning of this article. I have been developing the term OpenAPI. What's going on? Let's sing.

2. What is OpenAPI?

OpenAPI is the international standard for writing RESTful APIs. It is similar to a specification that allows developers around the world to rate the structure of their APIs. Also, adhere to all security, editing, debugging and other advanced processes when writing REST APIs from the ground up. And not just the bottom line, even existing APIs can be customized to suit global standards.

Besides, does this not sound right, why does compliance with international standards for product development help?

Originally, OpenAPI was known as the Swagger specification. Swagger came up with the best practices for building APIs and those leading practices became OpenAPI specifications.

Tools like SwaggerHub help developers develop APIs in a browser-based editor that complies with standards and has complete control over the design process.

With tools like Swagger Inspector, we can also generate our own API specifications and transfer them to other teams in the organization.

5.3 FUTURE SCOPE

- Integrate A server
It can be used in the govt schools so that the website can be fast and storing the data will be easy .
- As I am using the swaggers it is easy for me or to any other dedveloper or any person who want to use the api to perform CRUD operation easily on the data that is connected to the project.
- In future we can also design a front end of the website so that it can be used by everyone.
- We can also use these projects in surveys.

5.4 ADVANTAGES

First, it is based on Java, one of the world's most popular programming languages.

Alternatively, Spring Boot can help you quickly build any apps without having to worry about their safe and secure configuration.

Spring Boot has a large user community which means you can get free reading materials and tutorials. Spring Boot has many strings. This is useful if you are doing long or repetitive tasks. When the main thread is eaten, the others are used simultaneously.

Other additional benefits include:

Reduce time spent on development and increase the overall efficiency of the development team.

Helps automatically adjust all parts of the Spring-grade production grade application.

It prepares the creation and testing of Java-based applications by providing automated unit setting and integration testing.

It helps to avoid all manual work boilerplate coding, annotations, and complex XML configuration.

Comes with HTTP embedded servers like Jetty and Tomcat to test web applications.

Integrating the Spring Boot with the Spring ecosystem that includes Spring Data, Spring Security, Spring ORM, and Spring JDBC is easy.

It provides many plugins that developers can use to work with embedded and memory websites smoothly and easily.

Allows easy connection to online sites and services like Oracle, PostgreSQL, MySQL, MongoDB, Redis, Solr, ElasticSearch, Rabbit MQ, ActiveMQ, and many more.

Provides administrator support - you can manage remotely with the app.

It makes it easy to lean and comes with the Embedded Servlet Container.

Provides flexibility in setting up XML setting, Java beans, and Databases.

Provides easy access to the Command Line Interface which makes the development and testing of Java Boot built with Java or Groovy faster.

5.5 Limitations:

A major challenge that many engineers face when using the Spring Boot is lack of control. Visual style incorporates a number of additional (often unused) dependencies that increase the file size of the feed.

The Spring Boot artifact can be applied directly to Docker containers. This helps to determine when you need to quickly create microservices. However, some engineers argue that since the Spring Boot was designed to be lightweight and fast, it should not be used for monolithic applications.

While Spring Boot comes with basic logging and health monitoring tools for your app, this is not enough. Tools like Retrace help teams easily monitor Java applications. This tool helps detect slow SQL queries, provides performance reports and CPUs and identifies the most common errors in translating logs.

Additionally, it can be very challenging to update your Legacy Spring code. You can overcome this problem by using tools like Spring Boot CLI (Command Line Interface) that will help you change your asset code.

Some of the disadvantages are:

If you have never worked with Spring before and want to learn about proxies, dependence injections, and the AOP program, it is not recommended to start with Spring Boot as it does not include much of this information.

You really need to understand the many basic spring programs (and a little spring history), as well as some advanced topics to fix and solve them.

Spring Boot works well with microservices. Spring Boot materials can also be distributed directly to Docker containers. However, some developers do not recommend a framework for building large and monolithic applications.

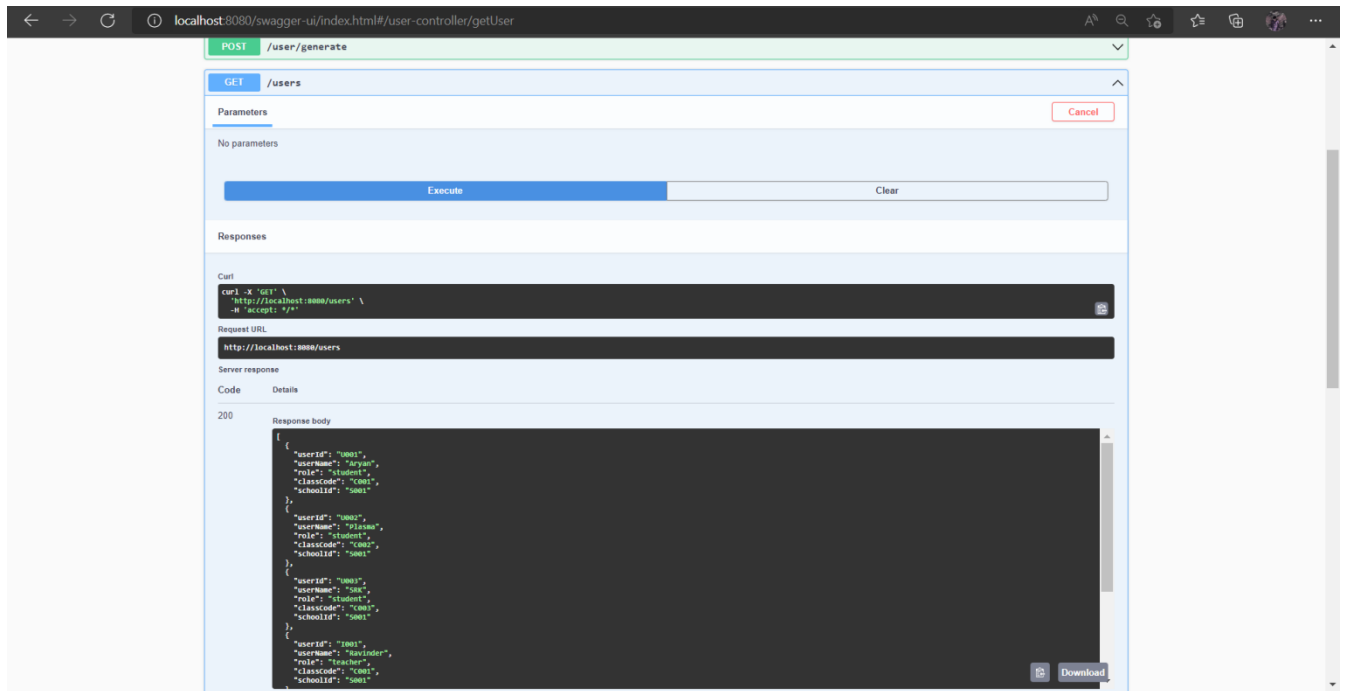
If you are unfamiliar with other Spring Ecosystem projects (Spring Security, Spring AMQP, Spring Integration, etc.), using them with Spring Boot will cause you to miss out on many points that you would understand if you started using them independently.

REFERENCE

Dealt with bugs and errors with the help of YouTube and Stack Overflow.

Studied about the required libraries/modules from GeeksforGeeks

SOME OUTPUT OF THE PROJECTS ::



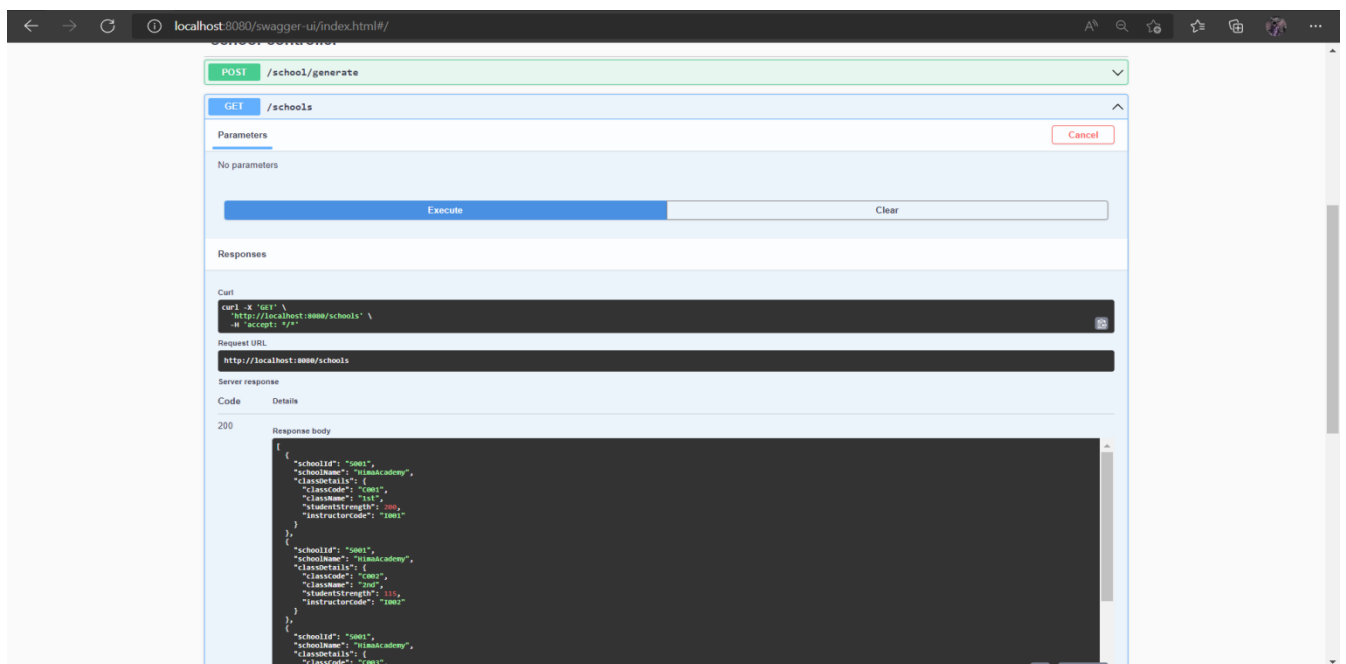
This screenshot shows the Swagger UI for the `/user/generate` endpoint. The interface includes a `POST` method, a `/users` endpoint, and a `Cancel` button. The `Parameters` section indicates no parameters are present. The `Responses` section shows a `200` status code with a `Response body` containing a JSON array of user objects. The `Code` and `Details` tabs are visible, with the `Code` tab selected. The `Response body` is a JSON array of user objects, each containing `userId`, `username`, `role`, `classcode`, and `schoolid`.

```
curl -X 'GET' \
  'http://localhost:8080/users' \
  -H 'accept: */*'

Request URL
http://localhost:8080/users

Server response

Code    Details
200     Response body
{
  "userId": "10001",
  "username": "Aryan",
  "role": "student",
  "classcode": "C001",
  "schoolid": "S001"
},
{
  "userId": "10002",
  "username": "Piyush",
  "role": "student",
  "classcode": "C002",
  "schoolid": "S001"
},
{
  "userId": "10003",
  "username": "Rohit",
  "role": "student",
  "classcode": "C003",
  "schoolid": "S001"
},
{
  "userId": "10004",
  "username": "Ravinder",
  "role": "teacher",
  "classcode": "C001",
  "schoolid": "S001"
}
```



This screenshot shows the Swagger UI for the `/school/generate` endpoint. The interface includes a `POST` method, a `/schools` endpoint, and a `Cancel` button. The `Parameters` section indicates no parameters are present. The `Responses` section shows a `200` status code with a `Response body` containing a JSON array of school objects. The `Code` and `Details` tabs are visible, with the `Code` tab selected. The `Response body` is a JSON array of school objects, each containing `schoolid`, `schoolname`, `classcode`, `classdetails`, `classname`, `studentlength`, and `instructorcode`.

```
curl -X 'GET' \
  'http://localhost:8080/schools' \
  -H 'accept: */*'

Request URL
http://localhost:8080/schools

Server response

Code    Details
200     Response body
{
  "schoolid": "S001",
  "schoolname": "Himalacademy",
  "classcode": "C001",
  "classdetails": {
    "classcode": "C001",
    "classname": "101",
    "studentlength": 200,
    "instructorcode": "1001"
  }
},
{
  "schoolid": "S002",
  "schoolname": "Himalacademy",
  "classcode": "C002",
  "classdetails": {
    "classcode": "C002",
    "classname": "102",
    "studentlength": 111,
    "instructorcode": "1002"
  }
},
{
  "schoolid": "S003",
  "schoolname": "Himalacademy",
  "classcode": "C003",
  "classdetails": {
    "classcode": "C003",
    "classname": "103",
    "studentlength": 111,
    "instructorcode": "1003"
  }
}
```