Build a REST Web Service Using Spring Boot and MongoDB

This Lab provides a step-by-step guide for building and configuring a REST web service in Java using the Spring Boot and MongoDB.

Prerequisites:

- Spring Tool Suite 3/4
- Java 1.8
- MongoDB Compass Community
- Postman

The Spring Tool Suite is an open-source, Eclipse-based IDE distribution that provides a superset of the Java EE distribution of Eclipse. It includes features that making working with Spring applications even easier.

Business Scenario:

An application has to be developed for management of interns hired by Yash technologies. The application will provide following services:

1. Based on technical evaluation, intern will be hired and details will be entered in system. Interns technical level will be decided based on college semester marks. Interns will have to go through training based on levels and clear certifications.
2. Intern's details can be retrieved by HR to assign projects to intern.
3. Intern's personal details can be updated.
4. Intern's level can be updated based on semester scores.
5. Intern's details will be removed from system if intern has completed internship.

There will be two applications to suffice need of above system.

InternsUIApp will be developed using Angular/ReactJS and InternsBusinessApp will be business application.

To create a prototype of an InternsBusinessApp, We will use Spring Boot REST

Spring Boot - Introduction

Spring Boot is an open source Java-based framework used to create a micro Service. It is developed by Pivotal Team and is used to build stand-alone and production ready spring

applications. This chapter will give you an introduction to Spring Boot and familiarizes you with its basic concepts.

What is Spring Boot?

Spring Boot provides a good platform for Java developers to develop a stand-alone and production-grade spring application that you can **just run**. You can get started with minimum configurations without the need for an entire Spring configuration setup.

Advantages

Spring Boot offers the following advantages to its developers −

- Easy to understand and develop spring applications
- Increases productivity
- Reduces the development time

Goals

Spring Boot is designed with the following goals −

- To avoid complex XML configuration in Spring
- To develop a production ready Spring applications in an easier way
- To reduce the development time and run the application independently
- Offer an easier way of getting started with the application

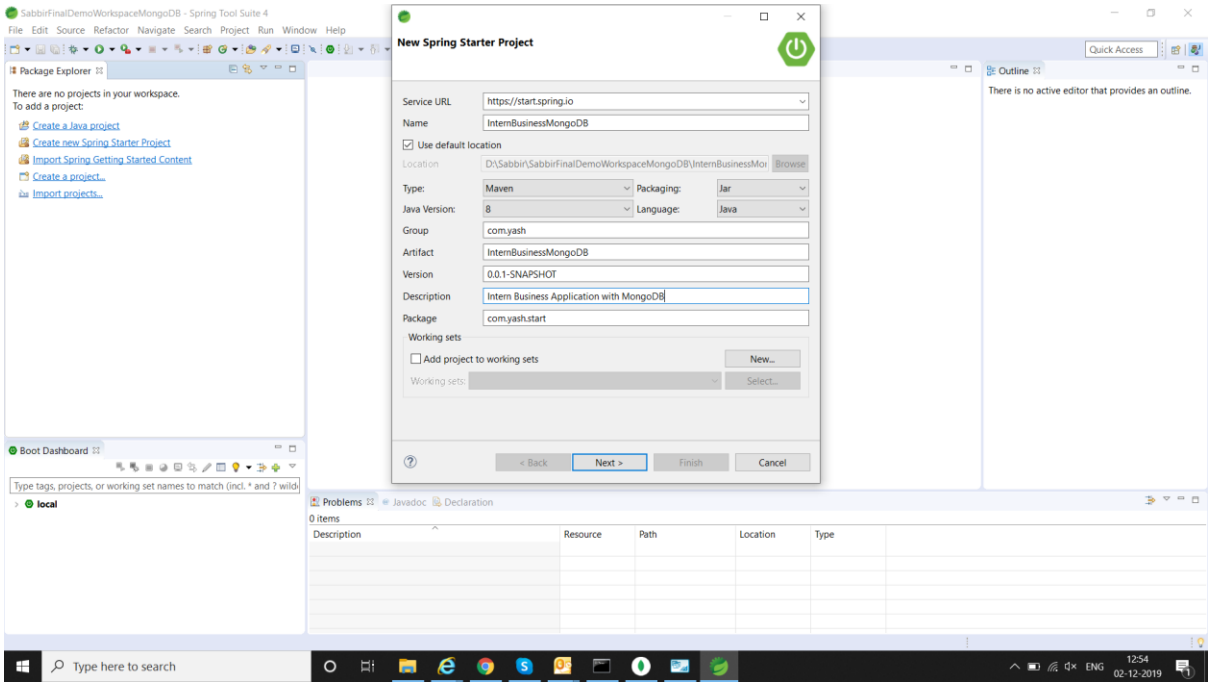
Why Spring Boot?

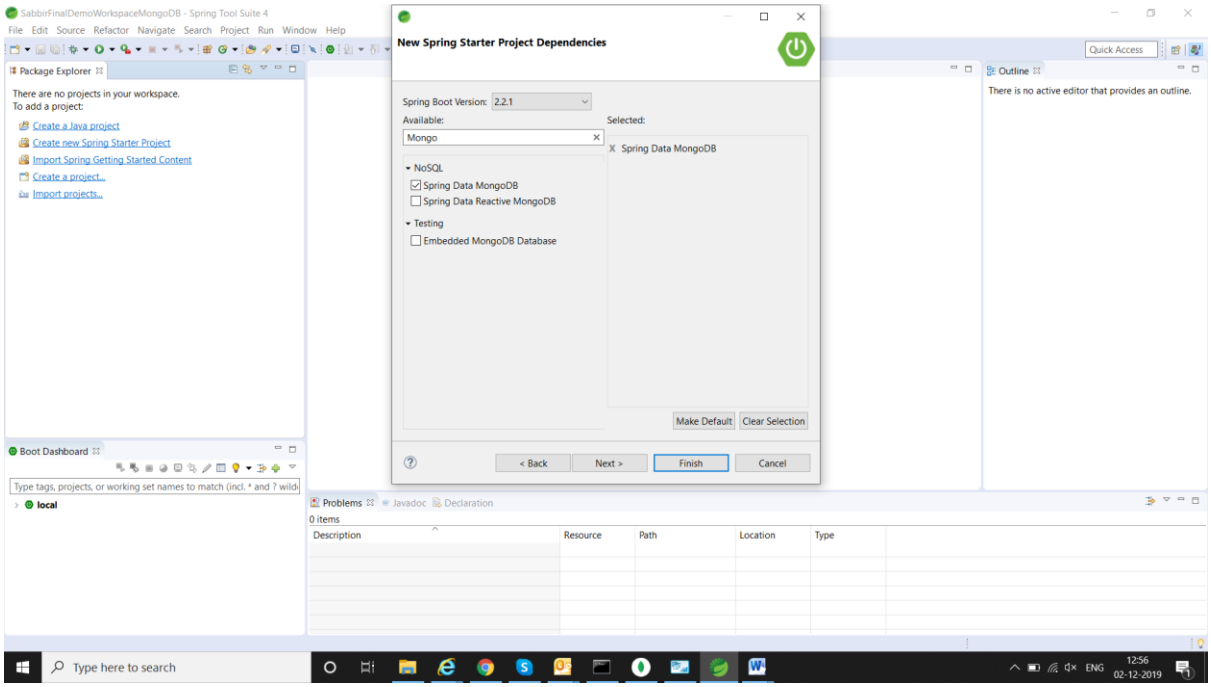
You can choose Spring Boot because of the features and benefits it offers as given here −

- It provides a flexible way to configure Java Beans, XML configurations, and Database Transactions.
- It provides a powerful batch processing and manages REST endpoints.
- In Spring Boot, everything is auto configured; no manual configurations are needed.
- It offers annotation-based spring application
- Eases dependency management
- It includes Embedded Servlet Container

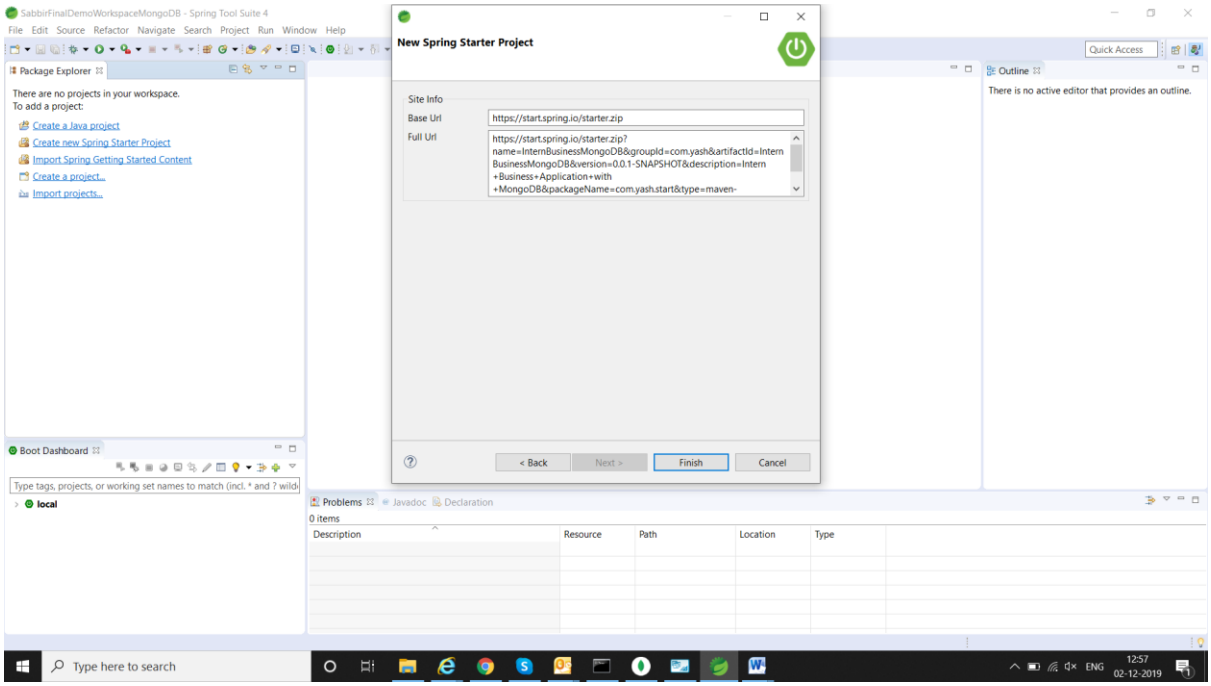
Spring Boot REST Project

Create new Spring Starter Project as shown in below image. Eclipse should have Spring support for this, all the latest Eclipse release has built-in Spring support.





Select Spring Web, No SQL- >Spring MongoDB



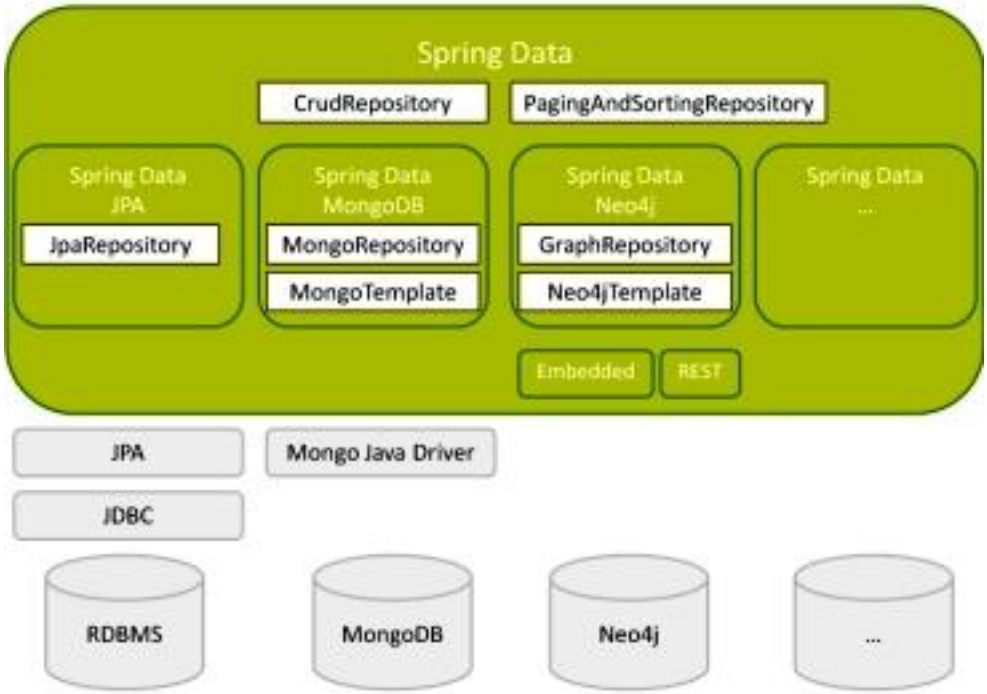
In com.yash.start package, apply below annotation on InternBusinessMongoDBApplication class

```
package com.yash.start;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.autoconfigure.domain.EntityScan;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.data.mongodb.repository.config.EnableMongoRepositories;
@SpringBootApplication
@EnableAutoConfiguration
@ComponentScan(basePackages="com.yash.*")
@EntityScan(basePackages="com.yash.entity")
@EnableMongoRepositories("com.yash.dao")
public class InternBusinessMongoDBApplication {
	public static void main(String[] args) {
		SpringApplication.run(InternBusinessMongoDBApplication.class, args);
	}

}
```

@EnableMongoRepositories scans the **current package** or **packages** mentioned in the **basePackages** attribute for any interface that extends **Spring Data** interface

In this application we will utilise Spring Data.

Spring Data is a high level SpringSource project whose purpose is to unify and ease the access to different kinds of persistence stores, both relational database systems and NoSQL data stores

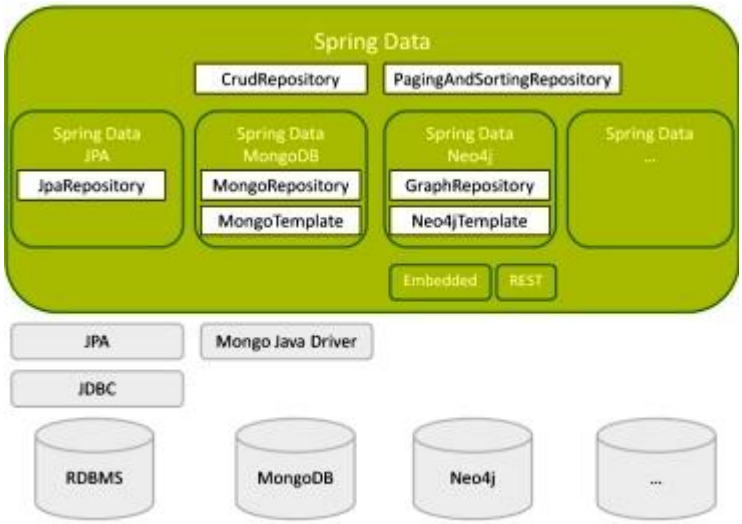


With every kind of persistence store, your repositories (a.k.a. DAOs, or Data Access Objects) typically offer CRUD (Create-Read-Update-Delete) operations on single domain objects, finder methods, sorting and pagination. Spring Data provides generic interfaces for these aspects (CrudRepository, PagingAndSortingRepository) as well as persistence store specific implementations.
Spring Data projects support the followings aspects:

- Templating
- Object/Datastore mapping
- Repository support

MongoDB is a document-based NoSQL database, providing high performance and high availability. Spring provides seamless integration with the **Mongo** database through **Spring Data MongoDB** which is a part of **Spring Data** project.

Spring Data is a high level SpringSource project whose purpose is to unify and ease the access to different kinds of persistence stores, both relational database systems and NoSQL data stores.



With every kind of persistence store, your repositories (a.k.a. DAOs, or Data Access Objects) typically offer CRUD (Create-Read-Update-Delete) operations on single domain objects, finder methods, sorting and pagination. Spring Data provides generic interfaces for these aspects (CrudRepository, PagingAndSortingRepository) as well as persistence store specific implementations.

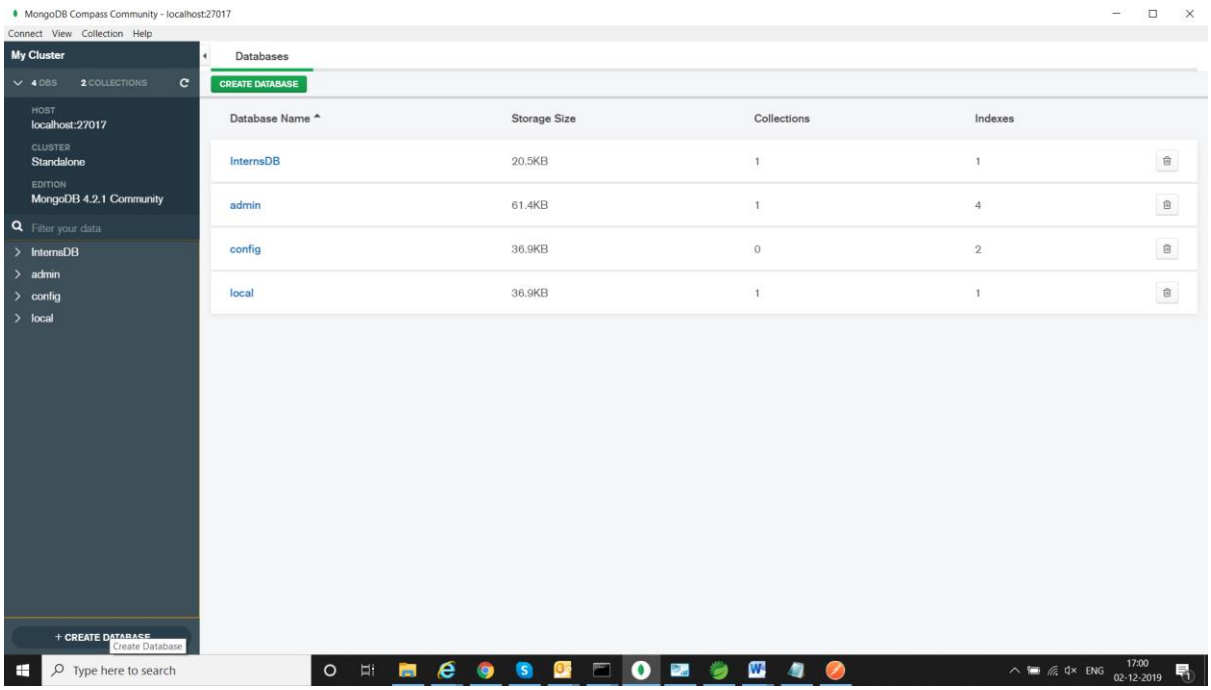
- Templating
- Object/Datastore mapping
- Repository support
  Other Spring Data projects like Spring Data Redis or Spring Data Riak essentially provide only templates, because the corresponding datastores persist unstructured data that cannot be mapped or queried.

Object/Datastore Mapping

As we have discussed in previous section,JPA introduced a standard for O/R mapping (i.e. mapping object graphs to relational database tables). Hibernate is probably the most common O/R mapper that implements the JPA specification.

With Spring Data, this support is extended to NoSQL datastores with object-like data structures. But these data structures can be quite different from each other, so it would be difficult to make up a common API for object/datastore mapping. Each type of datastore comes with its own set of annotations to provide the needed meta information for the mapping. Let’s see how a simple domain object User may be mapped to our data stores.

Start Mongo DB Compass Community and create a new database InternsDB



Create a collection in InternsDB with name “Interns”

Within collection Interns add an document(similar to database row) with following details,

Let Object Id be as it is.

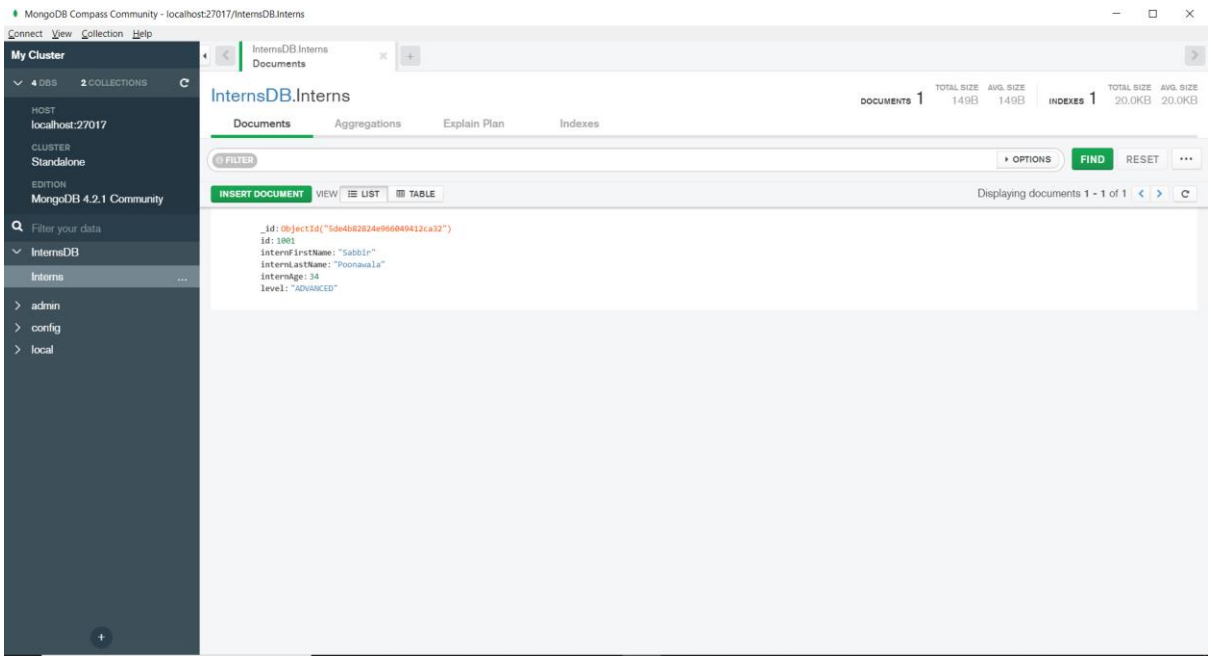
id:1001

internFirstName:"Sabbir"

internLastName:"Poonawala"

internAge:34

level:"ADVANCED"

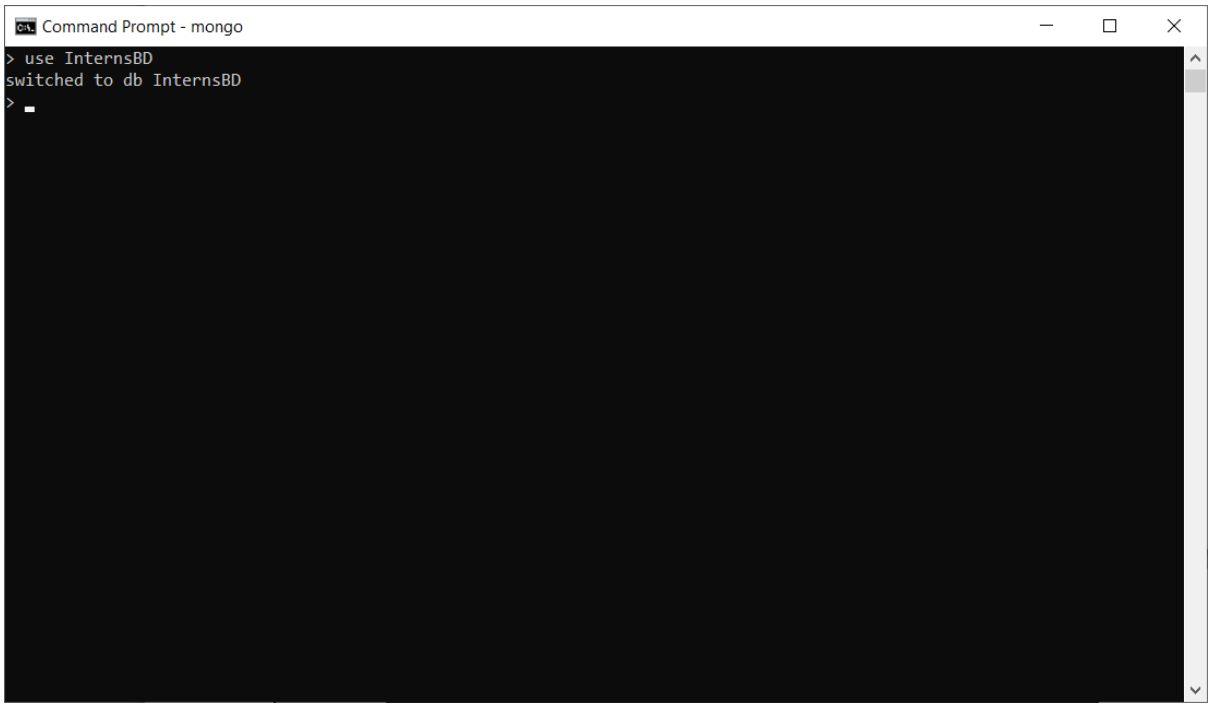


To create a user for access to InternsDB database,

Open CMD and type below path (installation directory of MongoDB)

cd C:\Program Files\MongoDB\Server\4.2\bin

Type : use InternsDB



Copy below command,

```
db.createUser(
  {
    user: "sabbir",
    pwd: passwordPrompt(), // or cleartext password
    roles: []
  }
)
```

On prompting for password, enter password as “sabbir”

In InternBusinessMongoDB, application.properties file mention below properties.

```
spring.data.mongodb.uri=mongodb://sabbir:sabbir@localhost:27017/InternsDB
server.port=8089
```

Create an Business Entity Interns in com.yash.entity package.

In **MongoDB**, we can relate the **Collections** to a **Table** in the relational database.

We have already created collection Interns in InternsDB database using MongoDB compass community.

Collection can also be created using below commands on command prompt.

```
db.createCollection('Interns');
```

Once we have created the **Interns** Collection, we need to add some intern data into the collection, each of them is called as **Document** and contains the **id, internFirstName, internLastName, internAge** and **level** of the intern. Just execute the below command to insert 1 intern document.

```
db.Interns.insert(
                {id: 1001,internFirstName:”amit", internLastName:
"desai",”internAge”:21,”level”:”ADVANCED”});
```

```java
package com.yash.entity;
import java.io.Serializable;
import org.springframework.data.annotation.Id;
import org.springframework.data.annotation.Transient;
import org.springframework.data.mongodb.core.mapping.Document;
import com.yash.helper.Levels;
/*
 * Business Entity represents Collection Interns
 */
@Document("Interns")
public class Interns implements Serializable {

	@Id
	private String objectId;

	private int id;
	private String internFirstName;
	private String internLastName;
	private int internAge;
	private Levels level;
	@Transient
	private int semester1Marks;
	@Transient
	private int semester2Marks;
	@Transient
	private int semester3Marks;

	public String getObjectId() {
		return objectId;
	}
	public void setObjectId(String objectId) {
		this.objectId = objectId;
	}
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getInternFirstName() {
		return internFirstName;
	}
	public void setInternFirstName(String internFirstName) {
		this.internFirstName = internFirstName;
	}
	public String getInternLastName() {
		return internLastName;
	}
	public void setInternLastName(String internLastName) {
		this.internLastName = internLastName;
	}
	public int getInternAge() {
		return internAge;
	}
	public void setInternAge(int internAge) {
		this.internAge = internAge;
	}
	public Levels getLevel() {
		return level;
```

```
	}
	public void setLevel(Levels level) {
		this.level = level;
	}

	public int getSemester1Marks() {
		return semester1Marks;
	}
	public void setSemester1Marks(int semester1Marks) {
		this.semester1Marks = semester1Marks;
	}
	public int getSemester2Marks() {
		return semester2Marks;
	}
	public void setSemester2Marks(int semester2Marks) {
		this.semester2Marks = semester2Marks;
	}
	public int getSemester3Marks() {
		return semester3Marks;
	}
	public void setSemester3Marks(int semester3Marks) {
		this.semester3Marks = semester3Marks;
	}
	@Override
	public int hashCode() {
		final int prime = 31;
		int result = 1;
		result = prime * result + id;
		result = prime * result + internAge;
		result = prime * result + ((internFirstName == null) ? 0 :
internFirstName.hashCode());
		result = prime * result + ((internLastName == null) ? 0 :
internLastName.hashCode());
		result = prime * result + ((level == null) ? 0 : level.hashCode());
		result = prime * result + semester1Marks;
		result = prime * result + semester2Marks;
		result = prime * result + semester3Marks;
		return result;
	}
	@Override
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		if (obj == null)
			return false;
		if (getClass() != obj.getClass())
			return false;
		Interns other = (Interns) obj;
		if (id != other.id)
			return false;
		if (internAge != other.internAge)
			return false;
		if (internFirstName == null) {
			if (other.internFirstName != null)
				return false;
		} else if (!internFirstName.equals(other.internFirstName))
			return false;
		if (internLastName == null) {
```

```
                    if (other.internLastName != null)
                          return false;
             } else if (!internLastName.equals(other.internLastName))
                    return false;
             if (level != other.level)
                    return false;
             if (semester1Marks != other.semester1Marks)
                    return false;
             if (semester2Marks != other.semester2Marks)
                    return false;
             if (semester3Marks != other.semester3Marks)
                    return false;
             return true;
      }
      @Override
      public String toString() {
             return "Interns [id=" + id + ", internFirstName=" + internFirstName
+ ", internLastName=" + internLastName
                          + ", internAge=" + internAge + ", level=" + level + ",
semester1Marks=" + semester1Marks
                          + ", semester2Marks=" + semester2Marks + ",
semester3Marks=" + semester3Marks + "]";
      }


}
```

The **id** attribute is for the internal use of the **MongoDB**, the **@Id** annotation on top of it informs Spring that this field will be used as the primary identifier. Since we are not persisting marks we have marked fields for marks as transient.

MongoDB stores the data in Collection, **Spring Data MongoDB** automatically maps the **Entity (Interns class)** with the **Collection (Intern)** only when the **name of the Entity and collection are same**, however, if it is different then we need to use **@Document** annotation to point the right Collection.

Create a package com.yash.dao and create an interface InternsRepository.

```
package com.yash.dao;

import java.util.Optional;

import org.springframework.data.annotation.Id;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Slice;
import org.springframework.data.mongodb.repository.MongoRepository;
import org.springframework.stereotype.Repository;

import com.yash.entity.Interns;
@Repository
public interface InternsRepository extends MongoRepository<Interns,Id>{

public Optional<Interns>findById(int id);
```

```
public Slice<Interns> findByInternFirstName(String internFirstName, Pageable
pageable);
}
```

MongoRepository will by default provide you with the generic methods like **save(), findAll(), insert(),** etc.. Out-of-the-box we can also add our custom methods and **Spring Data** has the query builder mechanism built in which strips the prefixes **find…By, read…By,** and **get…By,** We have used the same mechanism and built our custom method **findById().** Spring Data will automatically create an implementation in the runtime.

Create service interface InternsService in package com.yash.service

```
package com.yash.service;
import java.util.List;
import java.util.Optional;
import org.springframework.stereotype.Service;
import com.yash.entity.Interns;
@Service
public interface InternsService {
	List<Interns> retrieveInternsService();
	Optional<Interns> retrieveInternsByIdService(int internId);
	boolean registerInternService(Interns interns);
	boolean updateInternService(Interns interns);
	boolean updateInternLevelService(Interns interns);
	boolean removeInternService(int internId);
	List<Interns> getAllInternPage(Integer pageNo, Integer pageSize, String
sortBy);
	List<Interns> getAllInternSlice(Integer pageNo, Integer pageSize, String
sortBy,String internFirstName);
}
```

And implementation class, InternsServiceImpl

```
package com.yash.service;
import java.util.ArrayList;
import java.util.List;
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Slice;
import org.springframework.data.domain.Sort;
import org.springframework.data.mongodb.core.MongoTemplate;
import org.springframework.data.mongodb.core.query.Criteria;
import org.springframework.data.mongodb.core.query.Query;
import org.springframework.data.mongodb.core.query.Update;
import org.springframework.stereotype.Service;
import com.yash.dao.InternsRepository;
import com.yash.entity.Interns;
import com.yash.exception.InternsException;
import com.yash.helper.Levels;
```

```
@Service
public class InternsServiceImpl implements InternsService {
	@Autowired
	private InternsRepository internsRepository;

	@Autowired
	private MongoTemplate template;

	public List<Interns> retrieveInternsService() {
		// TODO Auto-generated method stub
		List<Interns> interns= internsRepository.findAll();
		System.out.println(interns);
		if(interns.isEmpty()){
			throw new InternsException("No interns records found");
		}else{
			return interns;
		}
	}
	@Override
	public Optional<Interns> retrieveInternsByIdService(int internId) {
		// TODO Auto-generated method stub
		Optional<Interns> intern=internsRepository.findById(internId);
		return intern;
	}
	public Levels determineLevelBySemesterMarks(Interns interns){
		int sem1Marks=interns.getSemester1Marks();
		int sem2Marks=interns.getSemester2Marks();
		int sem3Marks=interns.getSemester3Marks();
		int semAverage=(sem1Marks+sem2Marks+sem3Marks)/3;
		if(semAverage<=50){
			throw new InternsException("Intern did not match
eligibility");
		}
		if(semAverage>50 && semAverage<=60){
			return Levels.BEGINNER;
		}else if(semAverage>60 && semAverage<70){
			return Levels.INTERMEDIATE;
		}else{
			return Levels.ADVANCED;
		}
	}
	@Override
	public boolean registerInternService(Interns interns) {
		// TODO Auto-generated method stub
		Levels level=determineLevelBySemesterMarks(interns);
		interns.setLevel(level);
		Interns internsDB=internsRepository.insert(interns);
		if(interns.getId()==internsDB.getId())
		return true;
		else
	    return false;
	}
	@Override
	public boolean updateInternService(Interns interns) {
		// TODO Auto-generated method stub
		Levels level=determineLevelBySemesterMarks(interns);
		Query query = new Query();
		query.addCriteria(Criteria.where("id").is(interns.getId()));
		Update update = new Update();
```

```
            update.set("internFirstName", interns.getInternFirstName());
            update.set("internLastName", interns.getInternLastName());
            update.set("internAge", interns.getInternAge());
            update.set("level", level);
            template.updateFirst(query, update, Interns.class);
            return true;
      }
      @Override
      public boolean updateInternLevelService(Interns interns) {
            // TODO Auto-generated method stub
            Levels level=determineLevelBySemesterMarks(interns);

            Query query = new Query();
            query.addCriteria(Criteria.where("id").is(interns.getId()));
            Update update = new Update();
            update.set("level", level);
            template.updateFirst(query, update, Interns.class);
            return true;
      }
      @Override
      public boolean removeInternService(int internId) {
            // TODO Auto-generated method stub

            Optional<Interns>
internsOptional=internsRepository.findById(internId);
            Interns interns=internsOptional.get();
            internsRepository.delete(interns);
            return true;
      }
      @Override
      public List<Interns> getAllInternPage(Integer pageNo, Integer pageSize,
String sortBy) {
            // TODO Auto-generated method stub
            Pageable paging = PageRequest.of(pageNo, pageSize, Sort.by(sortBy));
        Page<Interns> pagedResult = internsRepository.findAll(paging);
        if(pagedResult.hasContent()) {
            return pagedResult.getContent();
        } else {
            return new ArrayList<Interns>();
        }
      }

      @Override
      public List<Interns> getAllInternSlice(Integer pageNo, Integer pageSize,
String sortBy,String internFirstName) {
            // TODO Auto-generated method stub
            Pageable paging = PageRequest.of(pageNo, pageSize,
Sort.by(sortBy).descending());
            Slice<Interns> slicedResult =
internsRepository.findByInternFirstName(internFirstName, paging);
            System.out.println("Slice number:"+slicedResult.getNumber());
        int numberOfElements = slicedResult.getNumberOfElements();
        System.out.println("Number of elements:"+numberOfElements);

            List<Interns> internList = slicedResult.getContent();
            return internList;
      }

}
```

We are using out-of-box methods , findAll(),save(),delete() of MongoRepository.

The *MongoTemplate* follows the standard template pattern in Spring and provides a ready to go, basic API to the underlying persistence engine.

One of the more common ways to query MongoDB with Spring Data is by making use of the *Query* and *Criteria* classes – which very closely mirror native operators.

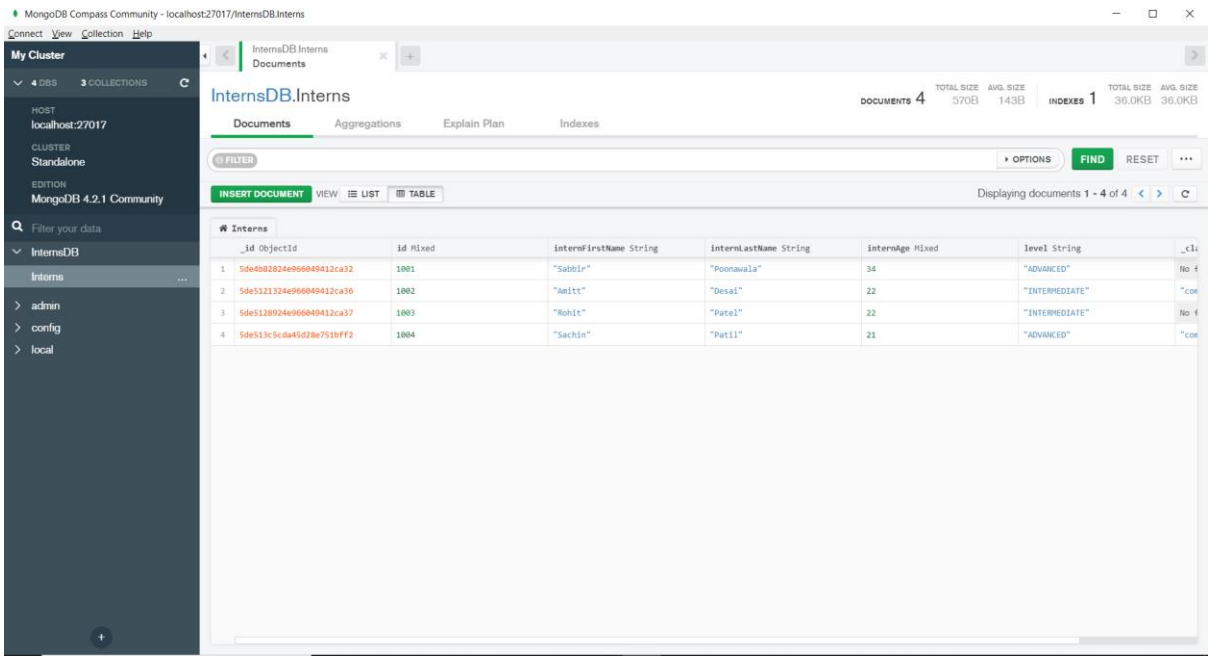
Create a RestController InternsController in com.yash.controller package. Controller remains all most the same as previous section.

```
package com.yash.controller;
import java.util.List;
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PatchMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import com.yash.entity.Interns;
import com.yash.service.InternsService;

@RestController
@RequestMapping("/interns-management")
public class InternsController {
	@Autowired
	private InternsService internService;
	@GetMapping("/yash-interns")
	public ResponseEntity<List<Interns>> retrieveAllInterns(){
		List<Interns> internsList=internService.retrieveInternsService();
		ResponseEntity<List<Interns>> response=new
ResponseEntity<List<Interns>>(internsList,HttpStatus.OK);
		return response;
	}
	@GetMapping("/yash-interns/{internId}")
	public ResponseEntity<Optional<Interns>>
retrieveInternById(@PathVariable("internId") int internId){
		Optional<Interns>
interns=internService.retrieveInternsByIdService(internId);
		ResponseEntity<Optional<Interns> > response=null;
	   Interns internsObj=interns.get();
		if(internsObj.getId()!=0){
		response=new
ResponseEntity<Optional<Interns>>(interns,HttpStatus.FOUND);
		}else{
		response=new
ResponseEntity<Optional<Interns>>(interns,HttpStatus.NOT_FOUND);
		}
```

```
            return response;
      }
      @PostMapping("yash-interns")
      public ResponseEntity<Void> registerIntern(@RequestBody Interns interns){
            boolean
registrationResult=internService.registerInternService(interns);
            ResponseEntity<Void> response=null;
            if(registrationResult){
                  response=new ResponseEntity<Void>(HttpStatus.CREATED);
            }else{
                  response=new ResponseEntity<Void>(HttpStatus.CONFLICT);
            }
            return response;
      }
      @PutMapping("yash-interns-manage")
      public ResponseEntity<Void> updateIntern(@RequestBody Interns interns)
      {
            boolean updateIntern=internService.updateInternService(interns);
            ResponseEntity<Void> response=null;
            if(updateIntern==true)
                  response=new ResponseEntity<Void>(HttpStatus.ACCEPTED);
            else
                  response=new ResponseEntity<Void>(HttpStatus.NOT_MODIFIED);
            return response;
      }
      @PatchMapping("yash-interns-level")
      public ResponseEntity<Void> updateInternLevel(@RequestBody Interns interns)
      {
            boolean updateLevel=internService.updateInternLevelService(interns);
            ResponseEntity<Void> response=null;
            if(updateLevel==true)
                  response=new ResponseEntity<Void>(HttpStatus.ACCEPTED);
            else
                  response=new ResponseEntity<Void>(HttpStatus.NOT_MODIFIED);
            return response;
      }

      @DeleteMapping("yash-interns-manage/{internId}")
      public ResponseEntity<Void> deleteStudent(@PathVariable("internId")int
internId){
      ResponseEntity<Void> response=null;
    boolean internRemoved=internService.removeInternService(internId);
    if(internRemoved){
                  response=new ResponseEntity<Void>(HttpStatus.OK);
            }else{
                  response=new ResponseEntity<Void>(HttpStatus.NOT_FOUND);

            }
            return response;
      }

    @GetMapping("yash-interns-paging")
    public ResponseEntity<List<Interns>> getAllInternsPage(
                        @RequestParam(defaultValue = "0") Integer pageNo,
                        @RequestParam(defaultValue = "10") Integer pageSize,
                        @RequestParam(defaultValue = "id") String sortBy)
    {
        List<Interns> list = internService.getAllInternPage(pageNo, pageSize,
sortBy);
```

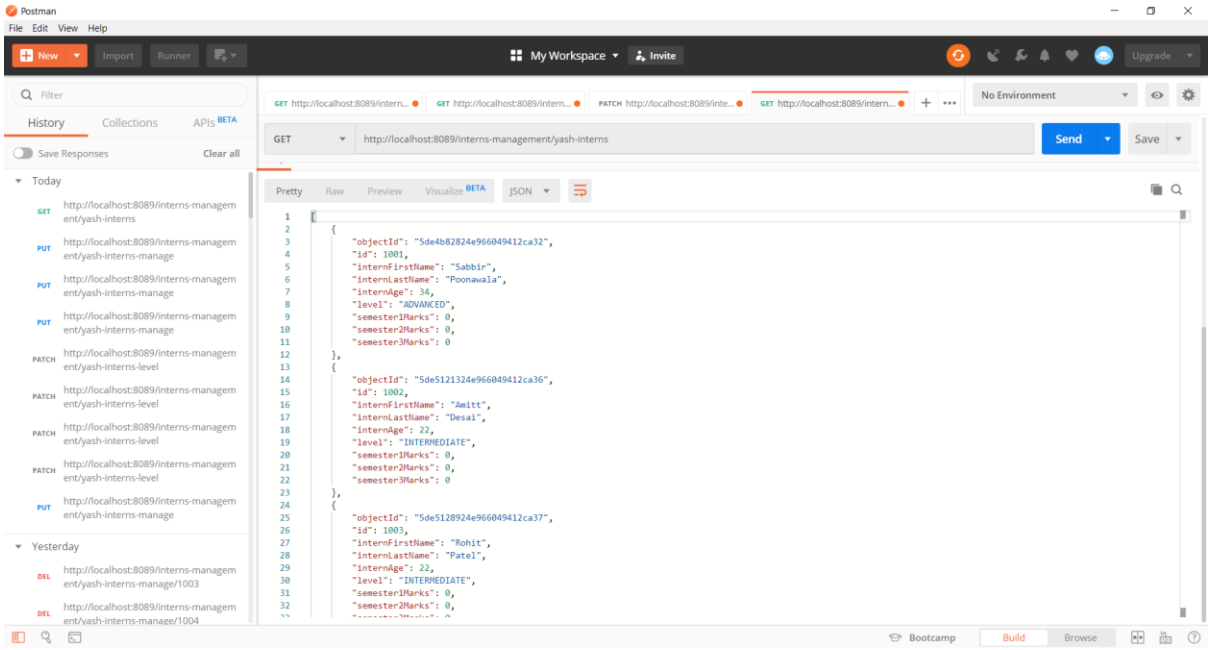
```
        return new ResponseEntity<List<Interns>>(list, new HttpHeaders(),
HttpStatus.OK);
    }

    @GetMapping("yash-interns-slice")
    public ResponseEntity<List<Interns>> getAllInternsSlice(
                        @RequestParam(defaultValue = "0") Integer pageNo,
                        @RequestParam(defaultValue = "10") Integer pageSize,
                        @RequestParam(defaultValue = "id") String sortBy,
                        @RequestParam String internFirstName)
    {
        List<Interns> list = internService.getAllInternSlice(pageNo, pageSize,
sortBy,internFirstName);
        return new ResponseEntity<List<Interns>>(list, new HttpHeaders(),
HttpStatus.OK);
    }


}
```

Documents in MongoDB,

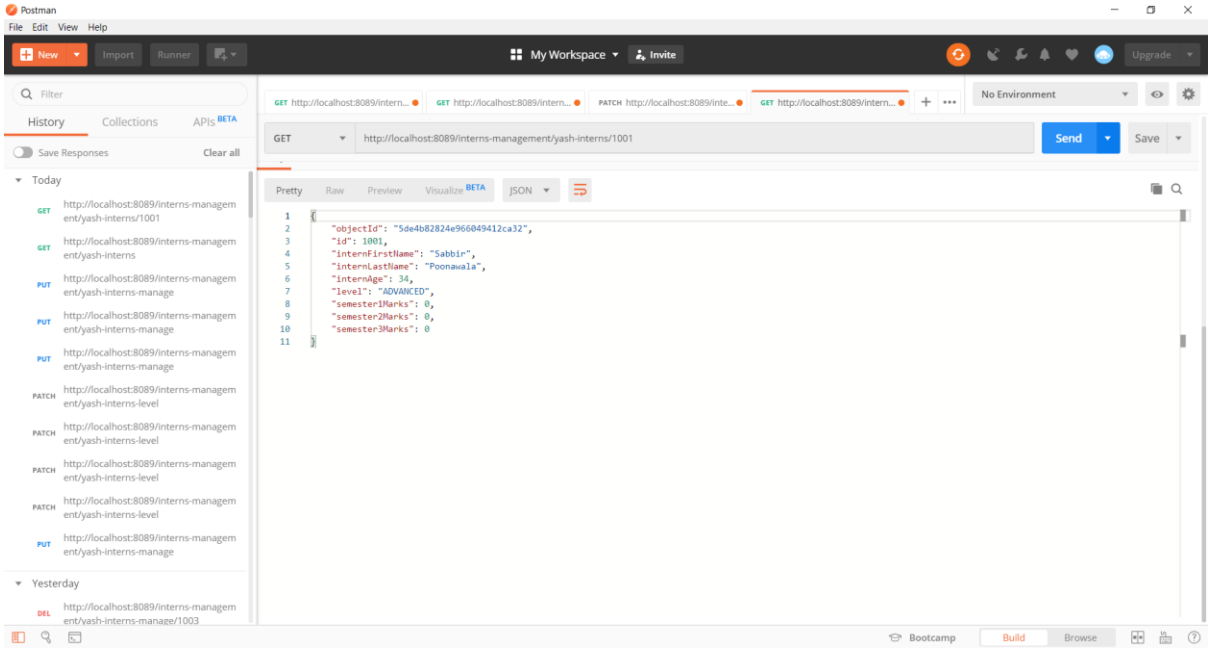


To test above services using postman,

GET : http://localhost:8089/interns-management/yash-interns



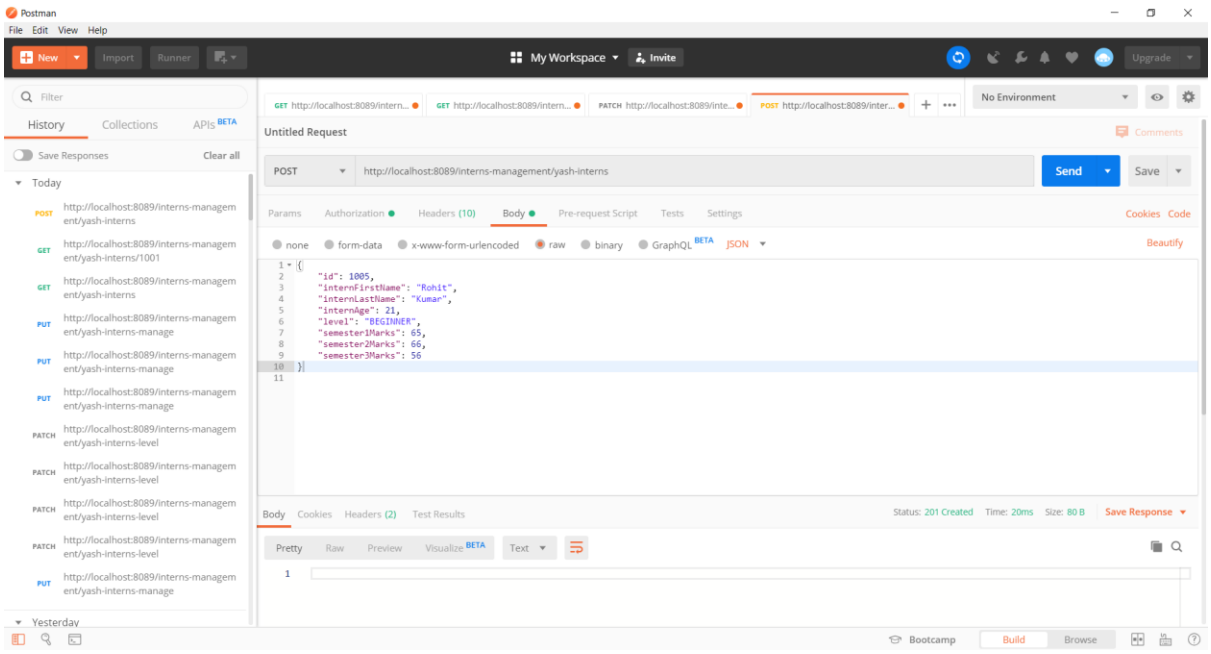
GET: http://localhost:8089/interns-management/yash-interns/1001

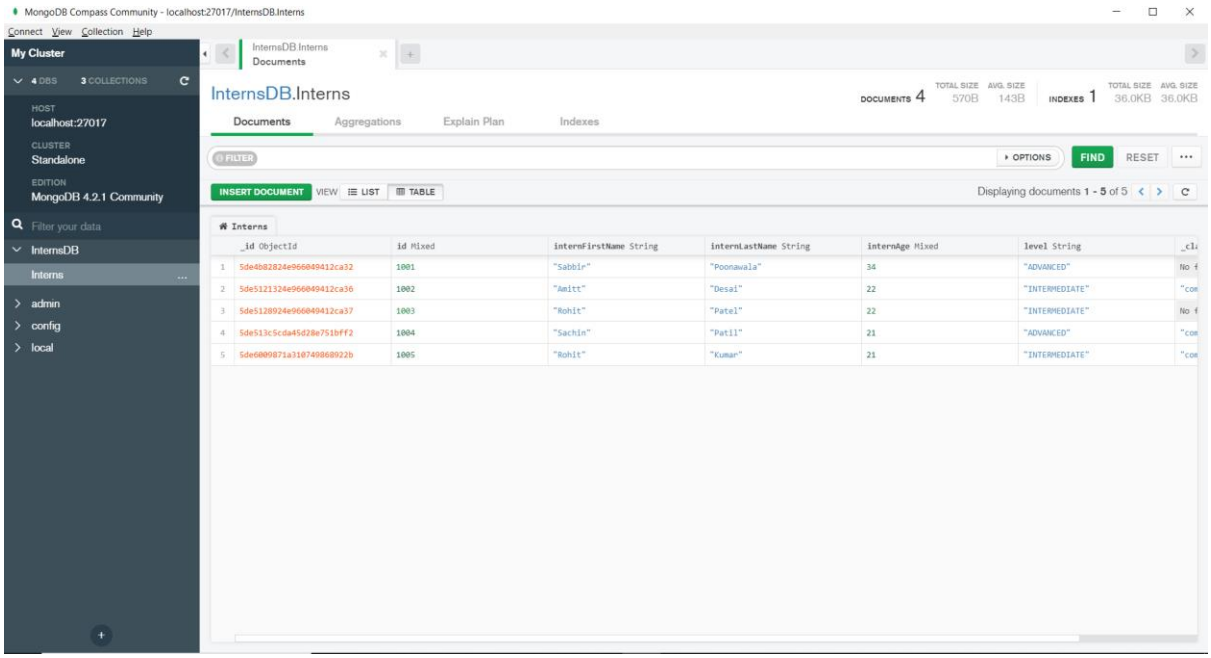


POST: http://localhost:8089/interns-management/yash-interns

JSON data:


```
{

   "id": 1005,
   "internFirstName": "Rohit",
   "internLastName": "Kumar",
   "internAge": 21,
   "level": "BEGINNER",
   "semester1Marks": 65,
   "semester2Marks": 66,
   "semester3Marks": 56
}
```

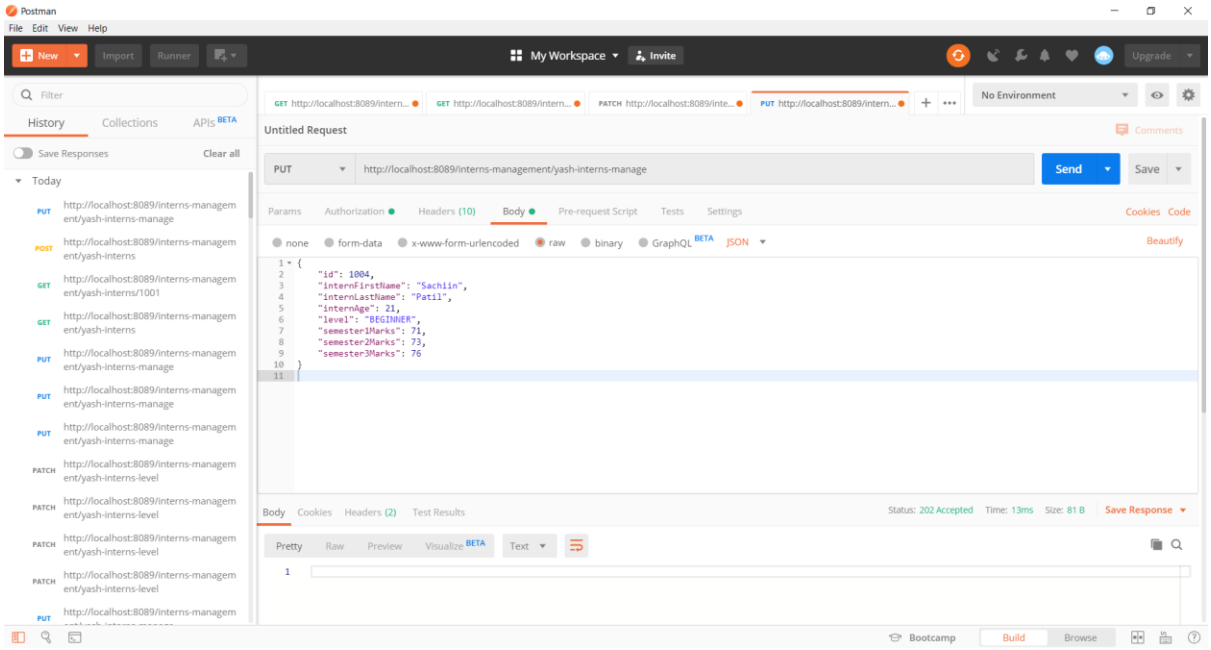



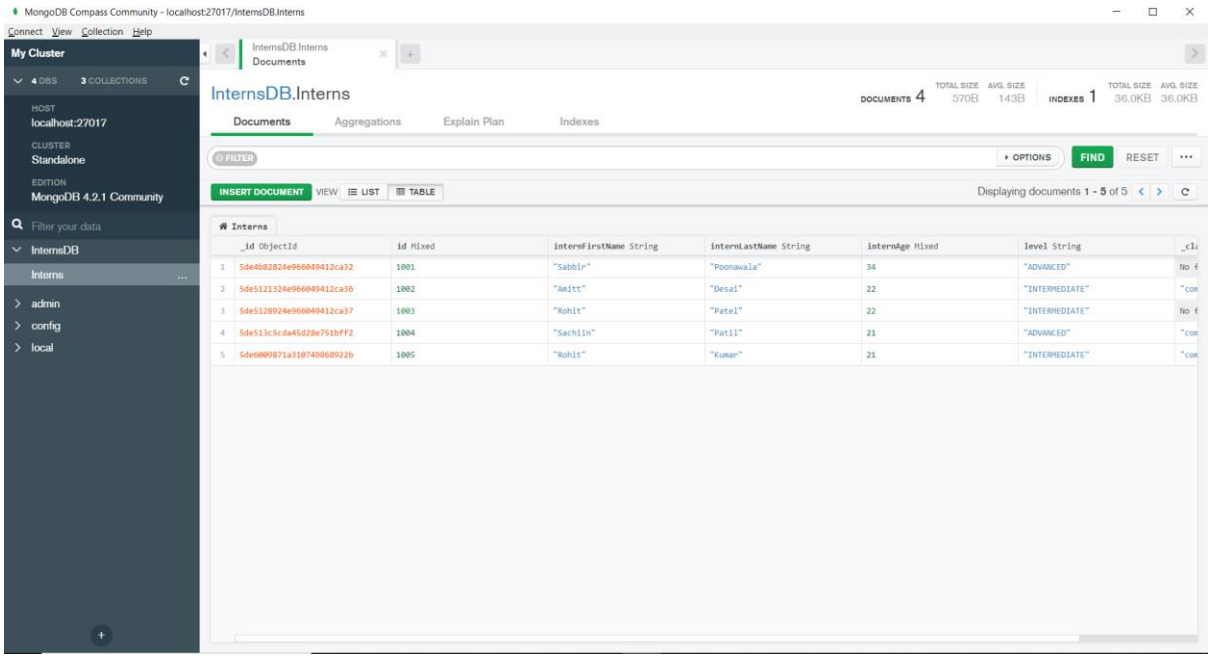


PUT: http://localhost:8089/interns-management/yash-interns-manage

JSON data:


```
{

   "id": 1004,
   "internFirstName": "Sachiin",
   "internLastName": "Patil",
   "internAge": 21,
   "level": "BEGINNER",
   "semester1Marks": 71,
   "semester2Marks": 73,
   "semester3Marks": 76
}
```

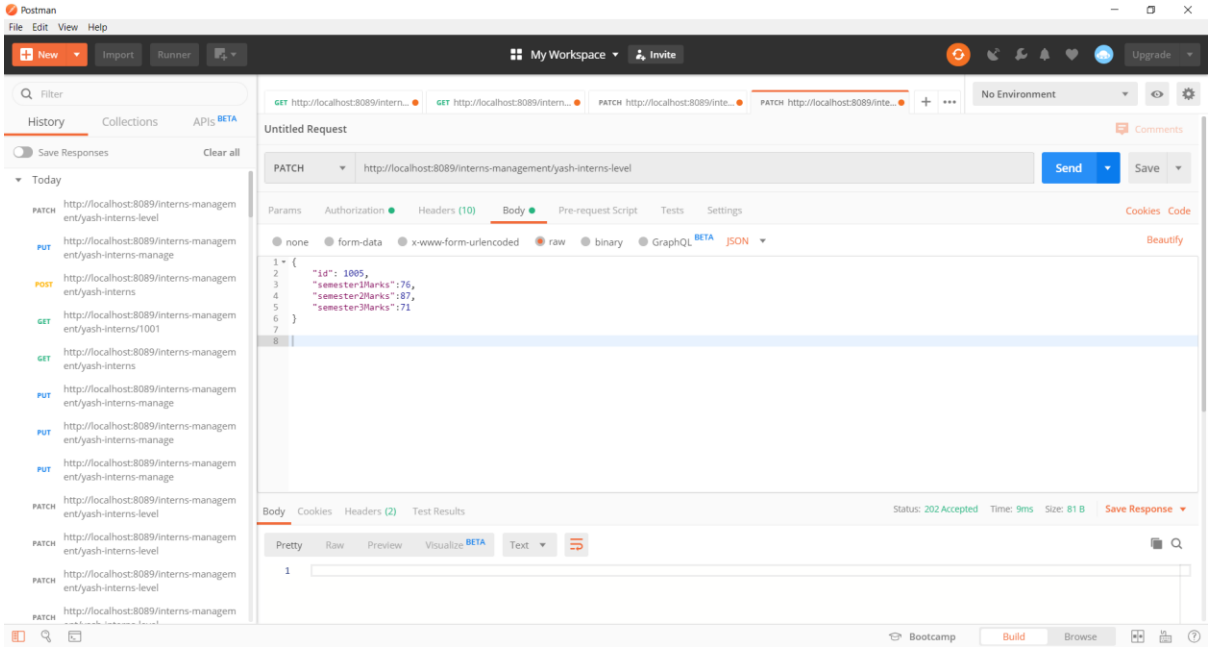

Postman
File Edit View Help
New Import Runner
My Workspace Invite
Upgrade
History Collections APIs BETA
Save Responses Clear all
Today
Untitled Request
PUT http://localhost:8089/interns-management/yash-interns-manage
Send Save
Params Authorization Headers (10) Body Pre-request Script Tests Settings
Cookies Code
none form-data x-www-form-urlencoded raw binary GraphQL BETA JSON
Beautify
{
"id": 1004,
"internFirstName": "Sachiin",
"internLastName": "Patil",
"internAge": 21,
"level": "BEGINNER",
"semester1Marks": 71,
"semester2Marks": 73,
"semester3Marks": 76
}
Body Cookies Headers (2) Test Results
Status: 202 Accepted Time: 13ms Size: 81 B
Save Response
Pretty Raw Preview Visualize BETA Text
Bootcamp Build Browse

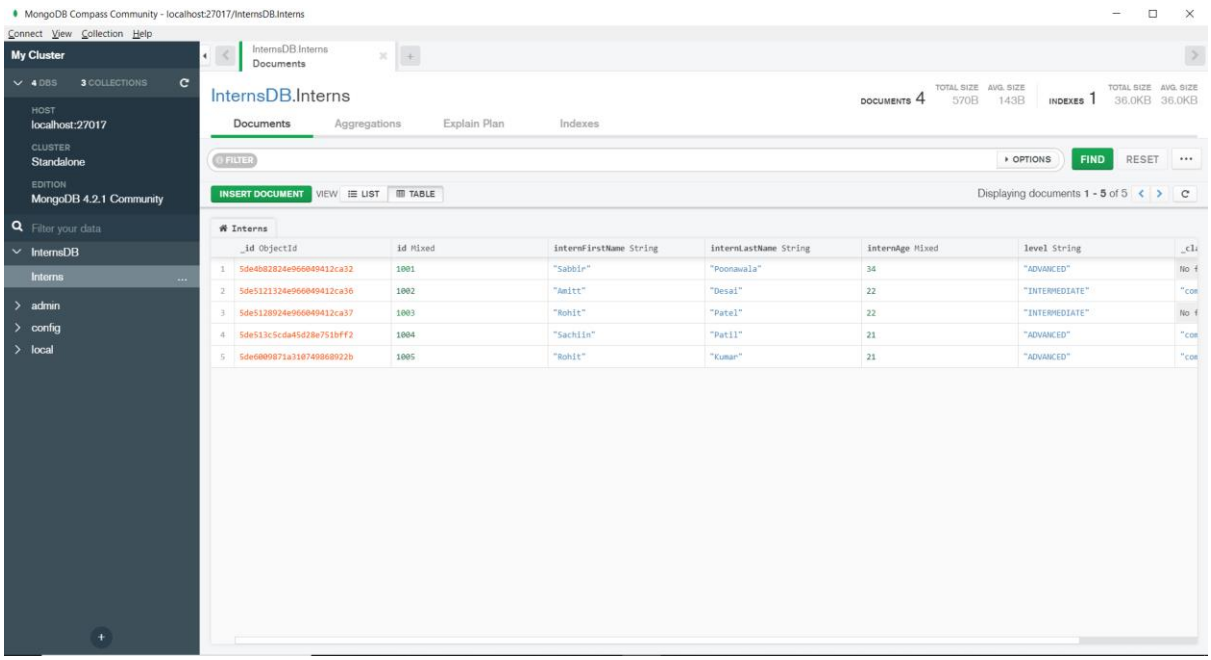

MongoDB Compass Community - localhost:27017/InternsDB.Interns
Connect View Collection Help
My Cluster
4 DBS 3 COLLECTIONS
HOST
localhost:27017
CLUSTER
Standalone
EDITION
MongoDB 4.2.1 Community
Filter your data
InternsDB
Interns
admin
config
local
InternsDB.Interns
DOCUMENTS 4 TOTAL SIZE 570B AVG. SIZE 143B
INDEXES 1 TOTAL SIZE 36.0KB AVG. SIZE 36.0KB
Documents Aggregations Explain Plan Indexes
FILTER OPTIONS FIND RESET
INSERT DOCUMENT VIEW LIST TABLE
Displaying documents 1 - 5 of 5
Interns
_id ObjectId
id Mixed
internFirstName String
internLastName String
internAge Mixed
level String
5de4b82824e966049412ca32 1001 "Sabbir" "Poonawala" 34 "ADVANCED"
5de5121324e966049412ca36 1002 "Amitt" "Desai" 22 "INTERMEDIATE"
5de5128924e966049412ca37 1003 "Rohit" "Patel" 22 "INTERMEDIATE"
5de513c5cda45d28e751bff2 1004 "Sachiin" "Patil" 21 "ADVANCED"
5de6009871a310749868922b 1005 "Rohit" "Kumar" 21 "INTERMEDIATE"

PATCH: http://localhost:8089/interns-management/yash-interns-level

JSON Data:

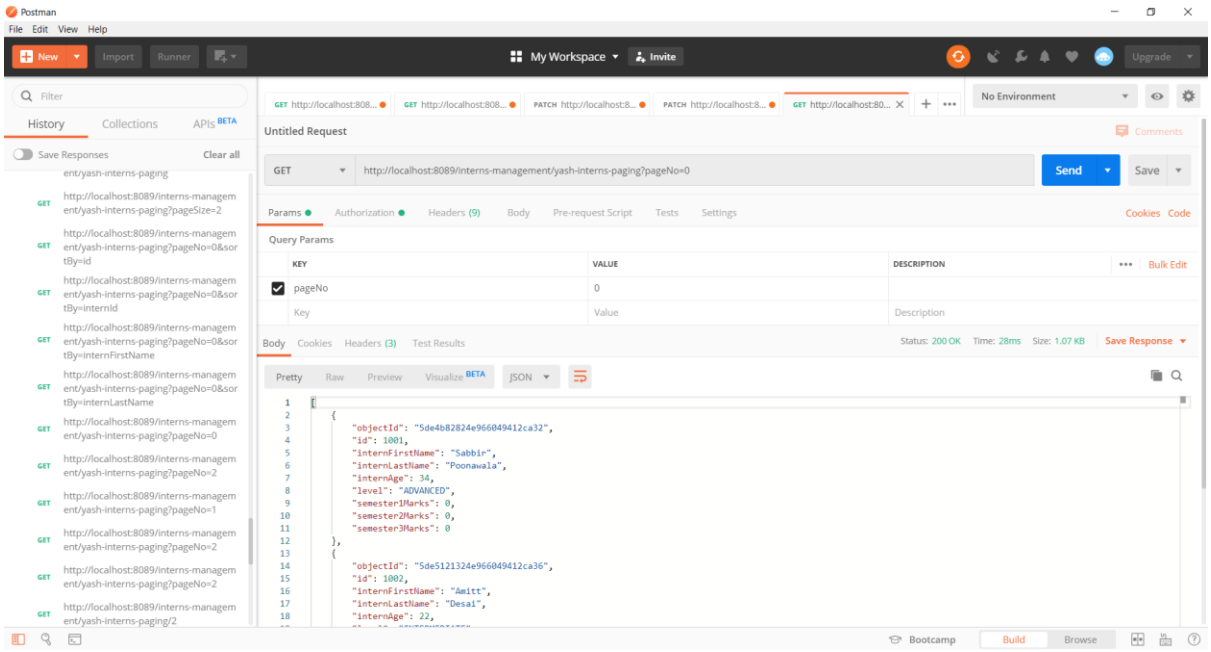
```
{

   "id": 1004,
   "semester1Marks":76,
   "semester2Marks":87,
   "semester3Marks":71
}
```



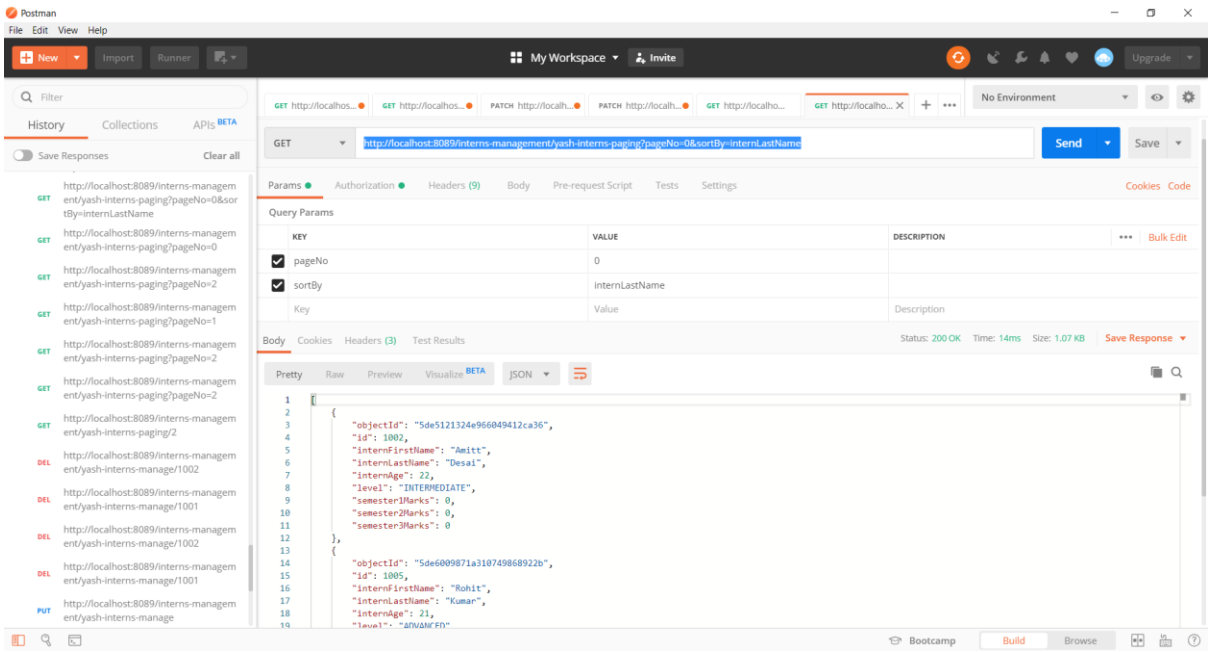


Pagination,

http://localhost:8089/interns-management/yash-interns-paging?pageNo=0

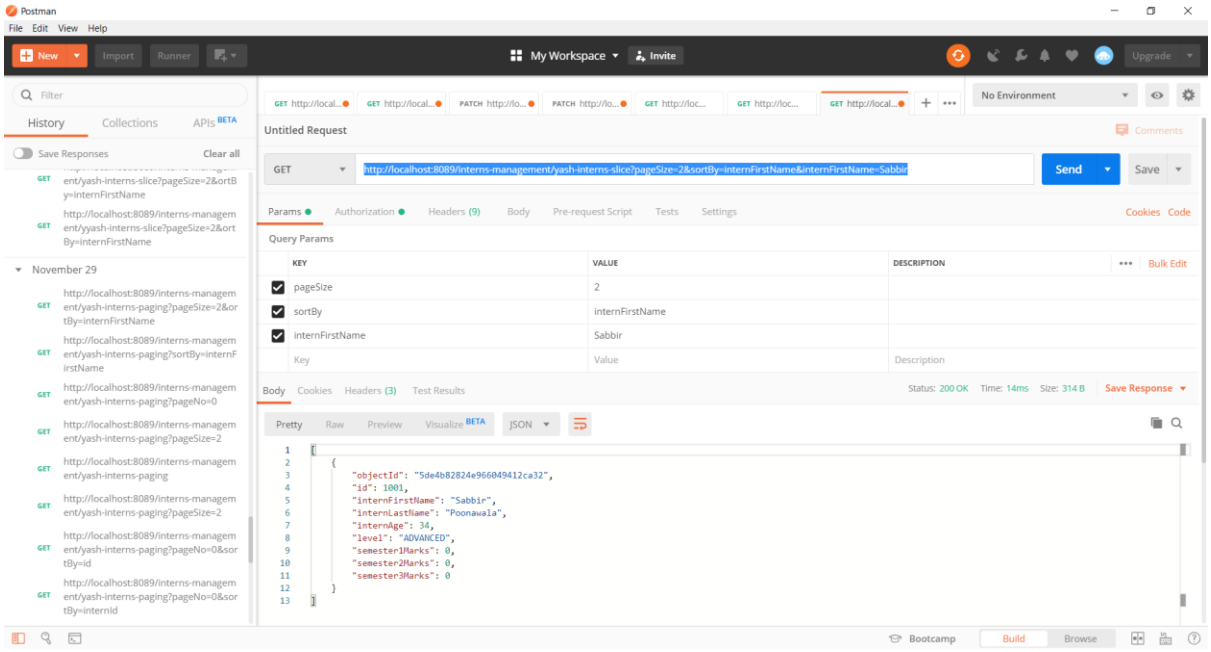


http://localhost:8089/interns-management/yash-interns-paging?pageNo=0&sortBy=internLastName



Slice

http://localhost:8089/interns-management/yash-interns-slice?pageSize=2&sortBy=internFirstName&internFirstName=Sabbir



Note: Testing using RestTemplate remain same as discussed in previous section.