Hackerank – 7

1. **Spring MusicTrack (sort by title)**
   Track.service.impl

```
service:package com.music.track.service.impl;

import com.music.track.dto.TrackRequest;
import com.music.track.model.Track;
import com.music.track.repository.TrackRepository;
import com.music.track.service.TrackService;
import java.util.List
@Service
public class TrackServiceImpl implements TrackService {
        @Autowired
        private TrackRepository trackRepository;
  @Override
  public Track createTrack(TrackRequest trackRequest) {
    Track t=new Track();
    t.setTitle(trackRequest.getTitle());
    t.setAlbumName(trackRequest.getAlbumName());
    t.setReleaseDate(trackRequest.getReleaseDate());
    t.setPlayCount(trackRequest.getPlayCount());
    return trackRepository.save(t);
  }
  @Override
  public List<Track> getAllTracks() {
    return trackRepository.findAll();
  }
   @Override
  public void deleteTrack(Long trackId) {
        trackRepository.deleteById(trackId);
  }
  @Override
  public List<Track> sortedTracks() {
    return trackRepository.findAll(Sort.by("title"));
  }

}
```

Track.dto

```
import java.util.Date;

public class TrackRequest {       //change the record to class ann add getter setter

private String title;
private String albumName;
private Date releaseDate;
private Integer playCount;
```

```
 public String getTitle() {
            return title;
}
public void setTitle(String title) {
            this.title = title;
}
public String getAlbumName() {
            return albumName;
}
public void setAlbumName(String albumName) {
            this.albumName = albumName;
}
public Date getReleaseDate() {
            return releaseDate;
}
public void setReleaseDate(Date releaseDate) {
            this.releaseDate = releaseDate;
}
public Integer getPlayCount() {
            return playCount;
}
public void setPlayCount(Integer playCount) {
            this.playCount = playCount;
}
public  TrackRequest(String title,
                      String albumName,
                      Date releaseDate,
                      Integer playCount) {
            this.title=title;
            this.releaseDate=releaseDate;
            this.albumName=albumName;
            this.playCount=playCount;
}
}
```

Track.controller

```
controller:package com.music.track.controller;
import com.music.track.dto.TrackRequest;
import com.music.track.model.Track;
import java.util.List;
@RestController
@RequestMapping("music/platform/v1/tracks")
public class TrackController
   private final TrackService trackService;
```

```
    @Autowired
    public TrackController(TrackService trackService) {
        this.trackService = trackService;
    }
    @PostMapping()
    public ResponseEntity<Track> createTrack(@RequestBody TrackRequest
trackRequest){
            Track createdTrack =trackService.createTrack(trackRequest);
        return ResponseEntity.status(201).body(createdTrack);
    }
    @GetMapping()
    public ResponseEntity<List<Track>> getAllTracks(){
        return ResponseEntity.ok(trackService.getAllTracks());
    }
    @DeleteMapping("/{trackId}")
    public ResponseEntity<Void> deleteTrack(@PathVariable Long trackId){
            trackService.deleteTrack(trackId);
        return ResponseEntity.noContent().build();
    }
    @GetMapping("/sorted")
    public List<Track> getTracksSorted() {
        return trackService.sortedTracks();

    }
}
```

2. **TrackInfo (filter by title)**

HackerRank 6

Spring-2 (stereotypes, just adding annotations and mapping)
Controller

```
package com.hackerrank.stereotypes.controller;
import com.hackerrank.stereotypes.model.Person;
@RestController
@RequestMapping("/contact")
public class ContactController {
            @Autowired
    ContactService contactService;
    @PostMapping("/save")
    public ResponseEntity<Person> save(@RequestBody Person person){
        Person saved = contactService.save(person);
        return new ResponseEntity(saved, HttpStatus.CREATED);
    }
    @GetMapping("/retrieve/{id}")
    public ResponseEntity<Person> retrieve(@PathVariable Integer id){
        Person person = contactService.retrieve(id);
        return new ResponseEntity(person, HttpStatus.OK);
    }
}
```

Add @Entity in person.java, @Repository in ContactRepository.java, @Service and @Auowired(above contactRepository declaration) in ContactService.java

Spring-1(3 types of Bean creation)

Xmlbased config (an xml file)

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
            http://www.springframework.org/schema/beans/spring-
beans.xsd">

  <bean id="thirdPartyNotificationService"

class="com.hackerrank.configstyles.service.ThirdPartyNotificationService">
     <constructor-arg value="THIRD_PARTY_SERVICE"/>
  </bean>

</beans>
```

Xmlbased java config

```
package com.hackerrank.configstyles.xmlbased;

import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.ImportResource;

@Configuration
@ImportResource({"classpath*:xml_based_configuration.xml"})
public class XmlBasedConfiguration {
}
```

Java based config

```
package com.hackerrank.configstyles.javabased;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import com.hackerrank.configstyles.service.CallNotificationService;
import com.hackerrank.configstyles.service.SmsNotificationService;

@Configuration
public class JavaBasedConfiguration {

@Bean(name = "smsNotificationService")
  public SmsNotificationService smsNotificationService() {
     return new SmsNotificationService("SMS_SERVICE");
  }

  @Bean(name = "callNotificationService")
  public CallNotificationService callNotificationService() {
        return new CallNotificationService("CALL_SERVICE");
    }}
```

SpringBoot file (app.java)

```
package com.hackerrank.configstyles;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Import;
import com.hackerrank.configstyles.javabased.JavaBasedConfiguration;
import com.hackerrank.configstyles.xmlbased.XmlBasedConfiguration;


@SpringBootApplication
@ComponentScan(basePackages = "com.hackerrank.configstyles.service")
@Import({JavaBasedConfiguration.class, XmlBasedConfiguration.class})
public class App {
    public static void main(String[] args) {
        SpringApplication.run(App.class, args);
    }
}
```

EmailService.java

```
package com.hackerrank.configstyles.service;

import org.springframework.stereotype.Component;

@Component("emailNotificationService")
public class EmailNotificationService implements NotificationService {
    private String serviceName;

    public EmailNotificationService() {
        this.serviceName = "EMAIL_SERVICE";
    }

    public ServiceResponse sendNotification(String notification) {
        return new ServiceResponse(serviceName, notification);
    }
}
```

Hackerank 5

Springboot-trial-product

Sorted_Products

```
package com.hackerrank.sample.dto;
import com.fasterxml.jackson.annotation.JsonProperty;
public class SortedProducts {
    @JsonProperty("barcode")
    private String barcode;
    @JsonProperty("price")
    private int price;
```

```
          public SortedProducts(String barcode , int price) {
             this.barcode = barcode;
             this.price = price;
          }
          public String getBarcode() {
             return barcode;
          }
          public void setBarcode(String barcode) {
             this.barcode = barcode;
          }

          public double getPrice() {
             return price;
          }

          public void setPrice(int price) {
             this.price = price;
          }
       }
```

SampleController.java

```
       import org.springframework.http.HttpStatus;
       import org.springframework.http.ResponseEntity;
       import org.springframework.web.bind.annotation.CrossOrigin;
       i@RestController
       public class SampleController {
                  final String uri = "https://jsonmock.hackerrank.com/api/inventory";
                  RestTemplate restTemplate = new RestTemplate();
                  String result = restTemplate.getForObject(uri, String.class);
                  JSONObject root = new JSONObject(result);
                  JSONArray data = root.getJSONArray("data");
                          @CrossOrigin
                          @GetMapping("/filter/price/{initial_price}/{final_price}")
                          private ResponseEntity< ? >
       filtered_books(@PathVariable("initial_price") int init_price ,
       @PathVariable("final_price") int final_price)
                          {
         List<FilteredProducts> list=new ArrayList<>();

                 for(int i=0;i<data.length();i++){
                          JSONObject json=data.getJSONObject(i);
                          int price=json.getInt("price");


                          if(price>init_price && price<final_price){
                                           String barcode=json.getString("barcode");
                                   list.add(new FilteredProducts(barcode));
                          }
                 }
```

```
            return
list.size()>0?ResponseEntity.status(200).body(list):ResponseEntity.notFound().build();

@CrossOrigin
 @GetMapping("/sort/price")
private ResponseEntity<?> sorted_books() {
   List<SortedProducts> list = new ArrayList<>();

   // Build product objects
   for (int i = 0; i < data.length(); i++) {
      JSONObject j = data.getJSONObject(i);
      String barcode = j.getString("barcode");
      int price = j.getInt("price");
      list.add(new SortedProducts(barcode, price));
   }
   List<FilteredProducts> filteredProductses =
      list.stream()
         .sorted(Comparator.comparing(SortedProducts::getPrice))
         .map(p -> new FilteredProducts(p.getBarcode()))
                                                   .collect(Collectors.toList());

   return new ResponseEntity<>(filteredProductses, HttpStatus.OK);
}
}
```

Springboot-uploader
RequestController.java

```
import org.springframework.http.HttpHeaders;
@RestController
public class RequestController {
   public static final String UPLOAD_DIR = "uploads/";

   @PostMapping("/uploader")
   public ResponseEntity uploader(@RequestParam("fileName") String fileName,
@RequestParam("file") MultipartFile file) {
      try {
          long maxSize = 100 * 1024;
         if (file.getSize() > maxSize) {
            return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
                         .body("Internal Server Error: File too large");
         }
          String cleanFileName=StringUtils.cleanPath(fileName);
          Path path=Paths.get(UPLOAD_DIR,cleanFileName);
          Files.write(path, file.getBytes());
          return ResponseEntity.status(HttpStatus.CREATED).build();
      }catch(Exception e) {
          return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
```

```
            }
        }
        @GetMapping("/downloader")
        public ResponseEntity downloader(@RequestParam String fileName) {
            try {
                String cleanFileName=StringUtils.cleanPath(fileName);
                Path path=Paths.get(UPLOAD_DIR,cleanFileName);
                if(!Files.exists(path)) {
                        return ResponseEntity.status(HttpStatus.NOT_FOUND).build();
                }
                byte[] fileBytes=Files.readAllBytes(path);
                HttpHeaders headers=new HttpHeaders();
                headers.add(HttpHeaders.CONTENT_DISPOSITION,"attachment:
    filename="+ cleanFileName);
                return ResponseEntity.ok().headers(headers).body(fileBytes);
            }catch(Exception e) {
                return
    ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
            }
        }
    }
```

Hackerrank 1-custom request employee validator
EmployyeValidator.java

```
    package com.hackerrank.validator.validation;
    import com.hackerrank.validator.model.Employee;
    import org.springframework.validation.Errors;
    import org.springframework.validation.Validator;
    public class EmployeeValidator implements Validator {
        @Override
        public boolean supports(Class<?> aClass) {
            return Employee.class.isAssignableFrom(aClass);
        }
        @Override
        public void validate(Object target, Errors errors) {
            Employee emp = (Employee) target;
            if (emp.getFullName() == null || emp.getFullName().trim().isEmpty()) {
                errors.rejectValue("fullName", "fullName.empty", "The fullName is a
    mandatory field");
            }
            if (emp.getMobileNumber() == null) {
                errors.rejectValue("mobileNumber", "mobileNumber.empty", "The
    mobileNumber is a mandatory field");
            } else {
                int length = emp.getMobileNumber().toString().length();
                if (length != 10) {
                    errors.rejectValue("mobileNumber", "mobileNumber.invalid", "The
    mobileNumber is a mandatory field");
                }
            }
            if (emp.getEmailId() == null || emp.getEmailId().trim().isEmpty()) {
              errors.rejectValue("emailId", "emailId.empty", "The emailId is a
    mandatory field");
            } else if (!emp.getEmailId().contains("@")) {
              errors.rejectValue("emailId", "emailId.invalid", "The emailId should be in
    a valid email format");
            }
            if (emp.getDateOfBirth() == null || emp.getDateOfBirth().trim().isEmpty()) {
              errors.rejectValue("dateOfBirth", "dateOfBirth.empty", "The dateOfBirth
    is a mandatory field");
            } else if (!emp.getDateOfBirth().matches("\\d{4}-\\d{2}-\\d{2}")) {
              errors.rejectValue("dateOfBirth", "dateOfBirth.invalid", "The dateOfBirth
    should be in YYYY-MM-DD format");
            }
          }
        }
```

Other CRUD questions

1. **Playlist Management Api**
   PlayList Controller

```
package com.example.playlist.controller;
import com.example.playlist.model.PlayList;
import com.example.playlist.repository.PlayListRepository;
import java.util.List;
@RestController
@RequestMapping("/v1/playlists")
public class PlayListController {
  @Autowired
  private PlayListRepository repository;
  @PostMapping
  public ResponseEntity<PlayList> createPlayList(@RequestBody PlayList playList) {
    PlayList saved = repository.save(playList);
    return ResponseEntity.status(201).body(saved);
  }
@GetMapping
public ResponseEntity<List<PlayList>> getAllPlayLists() {
  List<PlayList> playlists = repository.findAll(Sort.by(Sort.Direction.DESC, "id"));
  return ResponseEntity.ok(playlists);
  @GetMapping("/{id}")
  public ResponseEntity<PlayList> getPlayListById(@PathVariable Long id) {
    return repository.findById(id)
        .map(ResponseEntity::ok)
        .orElse(ResponseEntity.status(404).body(null));
  @DeleteMapping("/{id}")
  public ResponseEntity<Void> deletePlayList(@PathVariable Long id) {
    if (repository.existsById(id)) {
      repository.deleteById(id);
      return ResponseEntity.noContent().build();
```

```
        } else {
            return ResponseEntity.status(404).build();
        }
    }
}
```

2. **Trading Platform**
   TradeService.java

```
package com.example.trading.service;
import com.example.trading.model.Trader;
import java.util.List;
@Service
public class TraderService {

    @Autowired
    private TraderRepository repository;

    public Trader registerTrader(Trader trader) {
        if (repository.existsByEmail(trader.getEmail())) {
            return null;
        }
        trader.setCreatedAt(LocalDateTime.now());
        trader.setUpdatedAt(LocalDateTime.now());
        return repository.save(trader);
    }
    public List<Trader> getAllTraders() {
        return repository.findAllByOrderByIdAsc();
    }
    public Trader getTraderByEmail(String email) {
        return repository.findByEmail(email);
    }
    public Trader updateTraderName(String email, String name) {
        Trader trader = repository.findByEmail(email);
        if (trader != null) {
            trader.setName(name);
            trader.setUpdatedAt(LocalDateTime.now());
            return repository.save(trader);
        }
        return null;
    }
    public Trader addMoney(String email, Integer amount) {
        Trader trader = repository.findByEmail(email);
        if (trader != null) {
            trader.setAmount(trader.getAmount() + amount);
            trader.setUpdatedAt(LocalDateTime.now());
            return repository.save(trader);
        }
        return null;
    }
}
```

TradeController

```
package com.example.trading.controller;
import org.springframework.web.bind.annotation.*;
import java.util.List;
@RestController
@RequestMapping("/trading/traders")
public class TraderController {
  @Autowired
  private TraderService service;
  @PostMapping("/register")
  public ResponseEntity<Trader> registerTrader(@RequestBody Trader trader) {
    Trader saved = service.registerTrader(trader);
    if (saved == null) {
      return ResponseEntity.badRequest().build(); // 400
    }
    return ResponseEntity.status(201).body(saved); // 201
  }
  @GetMapping("/all")
  public ResponseEntity<List<Trader>> getAllTraders() {
    return ResponseEntity.ok(service.getAllTraders()); // 200
  }
  @GetMapping
  public ResponseEntity<Trader> getTraderByEmail(@RequestParam String email) {
    Trader trader = service.getTraderByEmail(email);
    if (trader == null) {
      return ResponseEntity.status(404).build(); // 404
    }
    return ResponseEntity.ok(trader); // 200
  }
  @PutMapping
  public ResponseEntity<Trader> updateTraderName(@RequestBody Trader request) {
    Trader updated = service.updateTraderName(request.getEmail(), request.getName());
    if (updated == null) {
      return ResponseEntity.status(404).build(); // 404
    }
    return ResponseEntity.ok(updated); // 200
  }
  @PutMapping("/add")
  public ResponseEntity<Trader> addMoney(@RequestBody Trader request) {
    Trader updated = service.addMoney(request.getEmail(), request.getAmount());
    if (updated == null) {
      return ResponseEntity.status(404).build(); // 404
    }
    return ResponseEntity.ok(updated); // 200
  }
}
```

3. **FIZZBUZZ Exception (REST advice)**
   FizzBuzzController

```
package com.hackerrank.restcontrolleradvice.controller;

import com.hackerrank.restcontrolleradvice.dto.FizzBuzzResponse;
@RestController
public class FizzBuzzController {
   @GetMapping(value = "/controller_advice/{code}", produces =
MediaType.APPLICATION_JSON_VALUE)
   public ResponseEntity<FizzBuzzResponse> getFizzBuzzResponse(@PathVariable("code") String
code)
          throws FizzException, BuzzException, FizzBuzzException {
      if (FizzBuzzEnum.FIZZ.getValue().equals(code)) {
         throw new FizzException("Fizz Exception has been thrown", "Internal Server Error");
      } else if (FizzBuzzEnum.BUZZ.getValue().equals(code)) {
         throw new BuzzException("Buzz Exception has been thrown", "Bad Request");
      } else if (FizzBuzzEnum.FIZZBUZZ.getValue().equals(code)) {
         throw new FizzBuzzException("FizzBuzz Exception has been thrown", "Insufficient
Storage");
      }
      return ResponseEntity.ok(new FizzBuzzResponse("Successfully completed fizzbuzz test",
200));
   }}
```

FizzBuzzExceptionHandler

```
package com.hackerrank.restcontrolleradvice.exception;
@RestControllerAdvice
public class FizzBuzzExceptionHandler {
   @ExceptionHandler(FizzException.class)
   public ResponseEntity<GlobalError> handleFizzException(FizzException ex) {
      GlobalError error = new GlobalError(ex.getMessage(), ex.getErrorReason());
      return new ResponseEntity<>(error, HttpStatus.INTERNAL_SERVER_ERROR); // 500
   }
   @ExceptionHandler(BuzzException.class)
   public ResponseEntity<GlobalError> handleBuzzException(BuzzException ex) {
      GlobalError error = new GlobalError(ex.getMessage(), ex.getErrorReason());
      return new ResponseEntity<>(error, HttpStatus.BAD_REQUEST); // 400
   }
   @ExceptionHandler(FizzBuzzException.class)
   public ResponseEntity<GlobalError> handleFizzBuzzException(FizzBuzzException ex) {
      GlobalError error = new GlobalError(ex.getMessage(), ex.getErrorReason());
      return new ResponseEntity<>(error, HttpStatus.INSUFFICIENT_STORAGE); // 507
   }
}

9. Artist Management Apis
@RestController
@RequestMapping("/v1/artists")
public class ArtistController{
@Autowired
private ArtistRepository repository;
```

```
@GetMapping("/")
public String root(){
return "API is active";
}
@PostMapping
public ResponseEntity<Artist> create(@RequestBody Artist artist){
  return ResponseEntity.status(201).body(repository.save(artist)); }
@GetMapping
public ResponseEntity<List<Artist>> getall(){
    return ResponseEntity.status(200).body(repository.findAll(Sort.by(Sort.Direction.ASC,"id")));
}
@GetMapping("/{id}")
public ResponseEntity<Artist> getbyid(@PathVariable Long id){
return repository.findById(id).map(ResponseEntity::ok).orElse(ResponseEntity.notFound().build());
}
@DeleteMapping("/{artistid}")
public ResponseEntity<void> delbyid(@PathVariable Long artistid){
repository.deleteById(artistid);
return ResponseEntity.status(204).build();
}
 Add this in repository
Optional<Artist>findById(Long Id);
```