COSC 31093 / BECS 31213

Enterprise Software Design and Architecture

Group Project – Group 04

Netflix Case study

Source Code

Figure 1: Netflix logo, (Nisha, 2023)

Acknowledgement

As the students of group 4, we would like to express my sincere gratitude to Dr. Mrs. Toshini Kumarika for her invaluable guidance and support throughout this module. Her expertise and dedication have been instrumental in enhancing my understanding and skills. I would also like to thank all group members for their collaboration and teamwork. Their contributions and commitment have been essential to the successful completion of this assignment.

Thank you all for your encouragement and assistance.

Controller Classes

UserController

```
package com.netflix.netflix_backend.controller;
import com.netflix.netflix_backend.exception.NotFoundException;
import com.netflix.netflix_backend.model.User;
import com.netflix.netflix_backend.repository.UserRepository;
import jakarta.servlet.http.HttpSession;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import java.util.List;

@RestController
@CrossOrigin("http://localhost:3000")
@RequestMapping("/user")
public class UserController {
    @Autowired
```

```
private UserRepository userRepository;

//Get All User Data
@GetMapping("/searchAll")
List<User> getAllUser(){
   return userRepository.findAll();
}

//Add new User to the system
@PostMapping("/add")
User newUser(@RequestBody User newUser){
   return userRepository.save(newUser);
}

//Search User by id
@GetMapping("/searchById/{id}")
User getUserById(@PathVariable Long id){
   return userRepository.findById(id).orElseThrow(()->new NotFoundException(id));
}

//search by category
@GetMapping("/searchByEmail/{userEmail}")
User getUserByCategory(@PathVariable String userEmail) {
   return userRepository.findByUserEmail(userEmail);
}

//Update User data
@PutMapping("/update/{id}")
User updateUser(@RequestBody User updateUser, @PathVariable Long id){
   return userRepository.findById(id).map(user -> {
      user.setUserEmail(updateUser.getUserEmail());
```

```
            user.setUserName(updateUser.getUserName());
            user.setPassword(updateUser.getPassword());
            return userRepository.save(user);
        }).orElseThrow(()->new NotFoundException(id));
    }

    //delete User
    @DeleteMapping("/delete/{id}")
    String deleteUser(@PathVariable Long id) {
        if (!userRepository.existsById(id)) {
            throw new NotFoundException(id);
        }
        userRepository.deleteById(id);
        return "User with id " + id + " has been deleted successfully.";
    }

    @PostMapping("/register")
    public String register(@RequestBody User user) {
        if (userRepository.findByUserEmail(user.getUserEmail()) != null) {
            return "User already registered as a user";
        }
        userRepository.save(user);
        return "User registered successfully";
    }

    @PostMapping("/login")
    public String login(@RequestBody User user, HttpSession session) {
        User existingUser = userRepository.findByUserEmail(user.getUserEmail());
        if (existingUser != null && existingUser.getPassword().equals(user.getPassword())) {
            session.setAttribute("user", user);
            return "Login successful";
```

```
        } else {
            return "Invalid username or password";
        }
    }
    @PostMapping("/logout")
    public String logout(HttpSession session) {
        session.invalidate();
        return "Logged out successfully";
    }
}
```

VideoController

```
package com.netflix.netflix_backend.controller;
import com.netflix.netflix_backend.exception.NotFoundException;
import com.netflix.netflix_backend.model.Video;
import com.netflix.netflix_backend.repository.VideoRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@CrossOrigin("http://localhost:3000")
@RequestMapping("/video")
public class VideoController {
    @Autowired
    private VideoRepository videoRepository;

    //Get All Video Data
    @GetMapping("/searchAll")
    List<Video> getAllVideo(){
```

```
    return videoRepository.findAll();
}

//Add new video to the system
@PostMapping("/add")
Video newVideo(@RequestBody Video newVideo){
    return videoRepository.save(newVideo);
}

//Search video by id
@GetMapping("/searchById/{id}")
Video getVideoById(@PathVariable Long id){
    return videoRepository.findById(id).orElseThrow(()->new NotFoundException(id));
}

//Search video by video title
@GetMapping("/searchByTitle/{videoTitle}")
Video getVideoByTitle(@PathVariable String videoTitle){
    return videoRepository.findByVideoTitle(videoTitle);
}

//search by category
@GetMapping("/searchByCategory/{category}")
public List<Video> getVideosByCategory(@PathVariable String category) {
    return videoRepository.findByCategory(category);
}

//search by video type
@GetMapping("/searchByVideoType/{videoType}")
Video getVideoByVideoType(@PathVariable String videoType){
    return videoRepository.findByVideoType(videoType);
```

```
    }

    //Update video data
    @PutMapping("/update/{id}")
    Video updatePatients(@RequestBody Video updateVideo, @PathVariable Long id){
        return videoRepository.findById(id).map(video -> {
            video.setVideoTitle(updateVideo.getVideoTitle());
            video.setYoutubeId(updateVideo.getYoutubeId());
            video.setCategory(updateVideo.getCategory());
            video.setVideoType(updateVideo.getVideoType());
            return videoRepository.save(video);
        }).orElseThrow(()->new NotFoundException(id));
    }

    //delete video
    @DeleteMapping("/delete/{id}")
    String deleteVideo(@PathVariable Long id) {
        if (!videoRepository.existsById(id)) {
            throw new NotFoundException(id);
        }
        videoRepository.deleteById(id);
        return "Video with id " + id + " has been deleted successfully.";
    }
}
```

ViewController

```
package com.netflix.netflix_backend.controller;
import com.netflix.netflix_backend.exception.NotFoundException;
import com.netflix.netflix_backend.model.User;
import com.netflix.netflix_backend.model.Video;
import com.netflix.netflix_backend.model.View;
```

```
import com.netflix.netflix_backend.model.ViewDetails;
import com.netflix.netflix_backend.repository.UserRepository;
import com.netflix.netflix_backend.repository.VideoRepository;
import com.netflix.netflix_backend.repository.ViewRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import java.util.List;
import java.util.stream.Collectors;

@RestController
@CrossOrigin("http://localhost:3000")
@RequestMapping("/view")
public class ViewController {
    @Autowired
    private ViewRepository viewRepository;
    @Autowired
    private VideoRepository videoRepository;
    @Autowired
    private UserRepository userRepository;

    //Get All View Data
    @GetMapping("/searchAll")
    List<View> getAllViews(){
        return viewRepository.findAll();
    }

    //Add new View to the system
    @PostMapping("/add")
    View newView(@RequestBody View newView){
        return viewRepository.save(newView);
    }
```

```
//Search View by id
@GetMapping("/searchById/{id}")
View getViewById(@PathVariable Long id){
   return viewRepository.findById(id).orElseThrow(()->new NotFoundException(id));
}

//Search View by video ID
@GetMapping("/searchByVideoId/{videoId}")
public List<View> getViewByVideoId(@PathVariable Long videoId){
   return viewRepository.findByVideoId(videoId);
}

//search View by user ID
@GetMapping("/searchByUserId/{userId}")
public List<View> getViewByUserId(@PathVariable Long userId){
   return viewRepository.findByUserId(userId);
}

//delete view
@DeleteMapping("/delete/{id}")
String deleteView(@PathVariable Long id) {
   if (!viewRepository.existsById(id)) {
      throw new NotFoundException(id);
   }
   viewRepository.deleteById(id);
   return "View with id " + id + " has been deleted successfully.";
}

//return all data
@GetMapping("/searchViewData")
```

```
    public List<ViewDetails> getAllViewDetails() {
        List<View> views = viewRepository.findAll();
        return views.stream().map(view -> {
            Video video = videoRepository.findById(view.getVideoId()).orElse(null);
            User user = userRepository.findById(view.getUserId()).orElse(null);
            return new ViewDetails(view, video, user);
        }).collect(Collectors.toList());
    }
}
```

Exception_Classes

NotFoundAdvice

```
package com.netflix.netflix_backend.exception;
import org.springframework.data.crossstore.ChangeSetPersister;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotation.ResponseStatus;
import java.util.HashMap;
import java.util.Map;

public class NotFoundAdvice {
    @ResponseBody
    @ExceptionHandler(ChangeSetPersister.NotFoundException.class)
    @ResponseStatus(HttpStatus.NOT_FOUND)
    public Map<String, String> exceptionHandler(ChangeSetPersister.NotFoundException exception){
        Map<String, String> errorMap = new HashMap<>();
        errorMap.put("errorMessage", exception.getMessage());
        return errorMap;
```

```
    }
}
```

NotFoundException

```
package com.netflix.netflix_backend.exception;

public class NotFoundException extends RuntimeException{
    public NotFoundException(Long id){
        super("Could not found the entity with id "+id);
    }
    public NotFoundException(String character){
        super("Could not found the entity with id "+character);
    }

}
```

Model Classes

User

```
package com.netflix.netflix_backend.model;
import jakarta.persistence.*;
import java.time.LocalDate;
import java.time.LocalTime;

@Entity
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long userId;
    private String userName;
    private String userEmail;
```

```
    private String password;
    @Column(name = "userRole")
    private String userRole = "User";
    @Column(name = "dateViewed")
    private LocalDate dateViewed = LocalDate.now();
    @Column(name = "timeViewed")
    private LocalTime timeViewed = LocalTime.now();

    public Long getUserId() {
        return userId;
    }

    public String getUserRole() {
        return userRole;
    }

    public void setUserRole(String userRole) {
        this.userRole = userRole;
    }

    public void setUserId(Long userId) {
        this.userId = userId;
    }

    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }
```

```
public String getUserEmail() {
    return userEmail;
}

public void setUserEmail(String userEmail) {
    this.userEmail = userEmail;
}

public String getPassword() {
    return password;
}

public void setPassword(String password) {
    this.password = password;
}

public LocalDate getDateViewed() {
    return dateViewed;
}

public void setDateViewed(LocalDate dateViewed) {
    this.dateViewed = dateViewed;
}

public LocalTime getTimeViewed() {
    return timeViewed;
}

public void setTimeViewed(LocalTime timeViewed) {
    this.timeViewed = timeViewed;
```

```
    }
}
```

Video

```
package com.netflix.netflix_backend.model;

import jakarta.persistence.*;

import java.time.LocalDate;
import java.time.LocalTime;

@Entity
public class Video {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long videoId;
    private String youtubeId;
    private String videoTitle;
    private String category;
    private String videoType;
    @Column(name = "dateAdded")
    private LocalDate dateAdded = LocalDate.now();
    @Column(name = "timeAdded")
    private LocalTime timeAdded = LocalTime.now();

    public Long getVideoId() {
        return videoId;
    }

    public void setVideoId(Long videoId) {
        this.videoId = videoId;
```

```
}

public String getYoutubeId() {
    return youtubeId;
}

public void setYoutubeId(String youtubeId) {
    this.youtubeId = youtubeId;
}

public String getVideoTitle() {
    return videoTitle;
}

public void setVideoTitle(String videoTitle) {
    this.videoTitle = videoTitle;
}

public String getCategory() {
    return category;
}

public void setCategory(String category) {
    this.category = category;
}

public String getVideoType() {
    return videoType;
}

public void setVideoType(String videoType) {
```

```
        this.videoType = videoType;
    }

    public LocalDate getDateAdded() {
        return dateAdded;
    }

    public void setDateAdded(LocalDate dateAdded) {
        this.dateAdded = dateAdded;
    }

    public LocalTime getTimeAdded() {
        return timeAdded;
    }

    public void setTimeAdded(LocalTime timeAdded) {
        this.timeAdded = timeAdded;
    }
}
```

View

```
package com.netflix.netflix_backend.model;
import jakarta.persistence.*;
import java.time.LocalDate;
import java.time.LocalTime;

@Entity
public class View {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long viewId;
```

```
private Long videoId;
private Long userId;
@Column(name = "dateViewed")
private LocalDate dateViewed = LocalDate.now();
@Column(name = "timeViewed")
private LocalTime timeViewed = LocalTime.now();

public Long getViewId() {
   return viewId;
}

public void setViewId(Long viewId) {
   this.viewId = viewId;
}

public Long getVideoId() {
   return videoId;
}

public void setVideoId(Long videoId) {
   this.videoId = videoId;
}

public Long getUserId() {
   return userId;
}

public void setUserId(Long userId) {
   this.userId = userId;
}
```

```
    public LocalDate getDateViewed() {
        return dateViewed;
    }

    public void setDateViewed(LocalDate dateViewed) {
        this.dateViewed = dateViewed;
    }

    public LocalTime getTimeViewed() {
        return timeViewed;
    }

    public void setTimeViewed(LocalTime timeViewed) {
        this.timeViewed = timeViewed;
    }
}
```

ViewDetails

```
package com.netflix.netflix_backend.model;
public class ViewDetails {
    private Long viewId;
    private String videoName;
    private String videoTitle;
    private String userName;
    private String userEmail;
    private String dateViewed;
    private String timeViewed;

    public ViewDetails(View view, Video video, User user) {
        if (video != null) {
```

```
        this.videoName = video.getYoutubeId();
        this.videoTitle = video.getVideoTitle();
    }
    if (user != null) {
        this.userName = user.getUserName();
        this.userEmail = user.getUserEmail();
    }
    this.dateViewed = String.valueOf(view.getDateViewed());
    this.timeViewed = String.valueOf(view.getTimeViewed());
    this.viewId = view.getViewId();
}

// Getters and setters
public String getVideoName() {
    return videoName;
}

public void setVideoName(String videoName) {
    this.videoName = videoName;
}

public String getVideoTitle() {
    return videoTitle;
}

public void setVideoTitle(String videoTitle) {
    this.videoTitle = videoTitle;
}

public String getUserName() {
    return userName;
```

```
}

public void setUserName(String userName) {
    this.userName = userName;
}

public String getUserEmail() {
    return userEmail;
}

public void setUserEmail(String userEmail) {
    this.userEmail = userEmail;
}

public String getDateViewed() {
    return dateViewed;
}

public void setDateViewed(String dateViewed) {
    this.dateViewed = dateViewed;
}

public String getTimeViewed() {
    return timeViewed;
}

public void setTimeViewed(String timeViewed) {
    this.timeViewed = timeViewed;
}
public Long getViewId() {
    return viewId;
```

```
    }

    public void setViewId(Long viewId) {
        this.viewId = viewId;
    }
}
```

Repository Interfaces

UserRepository

```
package com.netflix.netflix_backend.repository;
import com.netflix.netflix_backend.model.User;
import org.springframework.data.jpa.repository.JpaRepository;
import java.util.List;

public interface UserRepository extends JpaRepository<User, Long> {
    User findByUserEmail(String username);
}
```

VideoRepository

```
package com.netflix.netflix_backend.repository;
import com.netflix.netflix_backend.model.Video;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;

public interface VideoRepository extends JpaRepository<Video, Long> {
    Video findByVideoTitle(String videoTitle);
    List<Video> findByCategory(String category);
    Video findByVideoType(String videoType);
```

```
}
```

ViewRepository

```
package com.netflix.netflix_backend.repository;
import com.netflix.netflix_backend.model.Video;
import com.netflix.netflix_backend.model.View;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;

public interface ViewRepository extends JpaRepository<View, Long> {
    List<View> findByVideoId(Long videoId);
    List<View> findByUserId(Long userId);
}
```

Resources

application.properties

```
spring.application.name=netflix-backend
spring.datasource.url=jdbc:mysql://localhost:3306/netflix
spring.datasource.username=root
spring.datasource.password=
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
```

server.error.whitelabel.enabled=false

Dependencies

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">

        <modelVersion>4.0.0</modelVersion>

        <parent>

                <groupId>org.springframework.boot</groupId>

                <artifactId>spring-boot-starter-parent</artifactId>

                <version>3.3.1</version>

                <relativePath/> <!-- lookup parent from repository -->

        </parent>

        <groupId>com.netflix</groupId>

        <artifactId>netflix-backend</artifactId>

        <version>0.0.1-SNAPSHOT</version>

        <name>netflix-backend</name>

        <description>Netflix backend API development for ESD course module</description>

        <url/>

        <licenses>

                <license/>

        </licenses>

        <developers>

                <developer/>
```

```
</developers>
<scm>
	<connection/>
	<developerConnection/>
	<tag/>
	<url/>
</scm>
<properties>
	<java.version>17</java.version>
</properties>
<dependencies>
	<dependency>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-data-jpa</artifactId>
	</dependency>
	<dependency>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-web</artifactId>
	</dependency>

	<dependency>
		<groupId>com.mysql</groupId>
		<artifactId>mysql-connector-j</artifactId>
		<scope>runtime</scope>
	</dependency>
	<dependency>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-test</artifactId>
```

```
            <scope>test</scope>
        </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-web</artifactId>
        <version>6.1.10</version>
    </dependency>
  </dependencies>
        <build>
            <plugins>
                <plugin>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-maven-plugin</artifactId>
                </plugin>
            </plugins>
        </build>
</project>
```

Student number	Group Member
PS/2020/010	A.E.P.P. JAYASEKARA
PS/2020/012	A.M.L. ATHUKORALA
PS/2020/021	M.A.B. KAVEESH
PS/2020/024	K.D.C.D. FERNANDO

Table 1: Group members