**java\_springboot\_restful\_29\_AlbumController – Build Files Upload for Album**

**In application. properties; add below code**

# File Settings

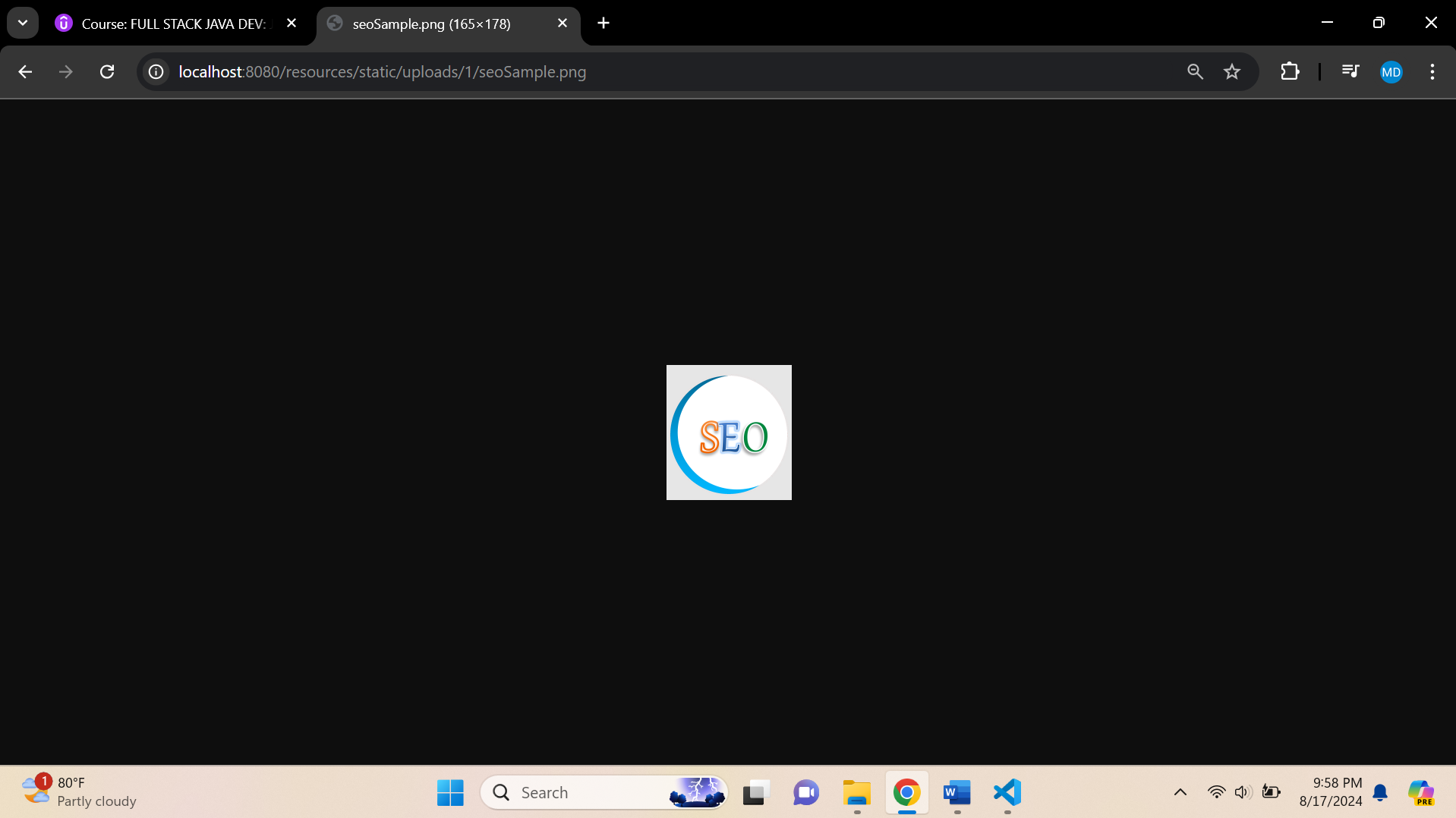
spring.servlet.multipart.max-file-size=10MB

spring.servlet.multipart.max-request-size=10MB

spring.mvc.static-path-pattern=/resources/static/\*\*

**Access to the static file**

URL: http://localhost:8080/resources/static/uploads/1/seoSample.png

****

**Create a model -> Photo.java -> in folder -> models**

package org.studyeasy.SpringRestdemo.model;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

import jakarta.persistence.JoinColumn;

import jakarta.persistence.ManyToOne;

import lombok.AllArgsConstructor;

import lombok.Getter;

import lombok.NoArgsConstructor;

import lombok.Setter;

import lombok.ToString;

@Entity

@Getter

@Setter

@ToString

@AllArgsConstructor

@NoArgsConstructor

public class Photo {

    @Id

    @GeneratedValue(strategy = GenerationType.SEQUENCE)

    private long id;

    private String name;

    private String description;

    private String originalFileName;

    private String fileName;

    @ManyToOne

    @JoinColumn(name = "album\_id", referencedColumnName = "id", nullable = false)

    private Album album;

}

**Create a PhotoRepository.java in repository folder**

package org.studyeasy.SpringRestdemo.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import org.studyeasy.SpringRestdemo.model.Photo;

@Repository

public interface PhotoRepository extends JpaRepository<Photo, Long>{

}

**Create a PhotoService.java in service folder**

package org.studyeasy.SpringRestdemo.service;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import org.studyeasy.SpringRestdemo.model.Photo;

import org.studyeasy.SpringRestdemo.repository.PhotoRepository;

@Service

public class PhotoService {

    @Autowired

    private PhotoRepository photoRepository;

    public Photo save(Photo photo){

        return photoRepository.save(photo);

    }

}

**Updated photos method in AlbumController**

@PostMapping(value = "{album\_id}/upload\_photos", consumes = { "multipart/form-data" })

    @Operation(summary = "Upload Photo into album")

    @SecurityRequirement(name = "studyeasy-demo-api")

    @ApiResponse(responseCode = "400", description = "Please check the payload of token")

    public ResponseEntity<List<String>> photos(@RequestPart(required = true) MultipartFile[] files,

            @PathVariable long album\_id, Authentication authentication) {

        String email = authentication.getName();

        Optional<Account> optionalAccount = accountService.findByEmail(email);

        Account account = optionalAccount.get();

        Optional<Album> optionalAlbum = albumService.findById(album\_id);

        Album album;

        if (optionalAlbum.isPresent()) {

            album = optionalAlbum.get();

            if (account.getId() != album.getAccount().getId()) {

                return ResponseEntity.status(HttpStatus.BAD\_GATEWAY).body(null);

            }

        } else {

            return ResponseEntity.status(HttpStatus.BAD\_GATEWAY).body(null);

        }

        List<String> fileNamesWithSuccess = new ArrayList<>();

        List<String> fileNamesWithError = new ArrayList<>();

        Arrays.asList(files).stream().forEach(file -> {

            // Checking the type of the file is correct or not

            String contentType = file.getContentType();

            if (contentType.equals("image/png")

                    || contentType.equals("image/jpg")

                    || contentType.equals("image/jpeg")) {

                fileNamesWithSuccess.add(file.getOriginalFilename());

    // When we are storing the file in the database, there is a possibility that

    // file name from the user

    // is repeating and if that happens, then file from the user will get repeated and in server will get replaced.

    // In order to prevent that, we need to create a random string.;

                int length = 10;

                boolean useLetters = true;

                boolean useNumbers = true;

                try {

                    String fileName = file.getOriginalFilename();

                    String generatedString = RandomStringUtils.random(length, useLetters, useNumbers);

                    String final\_photo\_name = generatedString + fileName;

                    String absolute\_fileLocation = AppUtil.get\_photo\_upload\_path(final\_photo\_name, album\_id);

                    Path path = Paths.get(absolute\_fileLocation);

                    Files.copy(file.getInputStream(), path, StandardCopyOption.REPLACE\_EXISTING);

                    Photo photo = new Photo();

                    photo.setName(fileName);

                    photo.setFileName(final\_photo\_name);

                    photo.setOriginalFileName(fileName);

                    photo.setAlbum(album);

                    photoService.save(photo);

                } catch (Exception e) {

                }

            }else{

                fileNamesWithError.add(file.getOriginalFilename());

            }

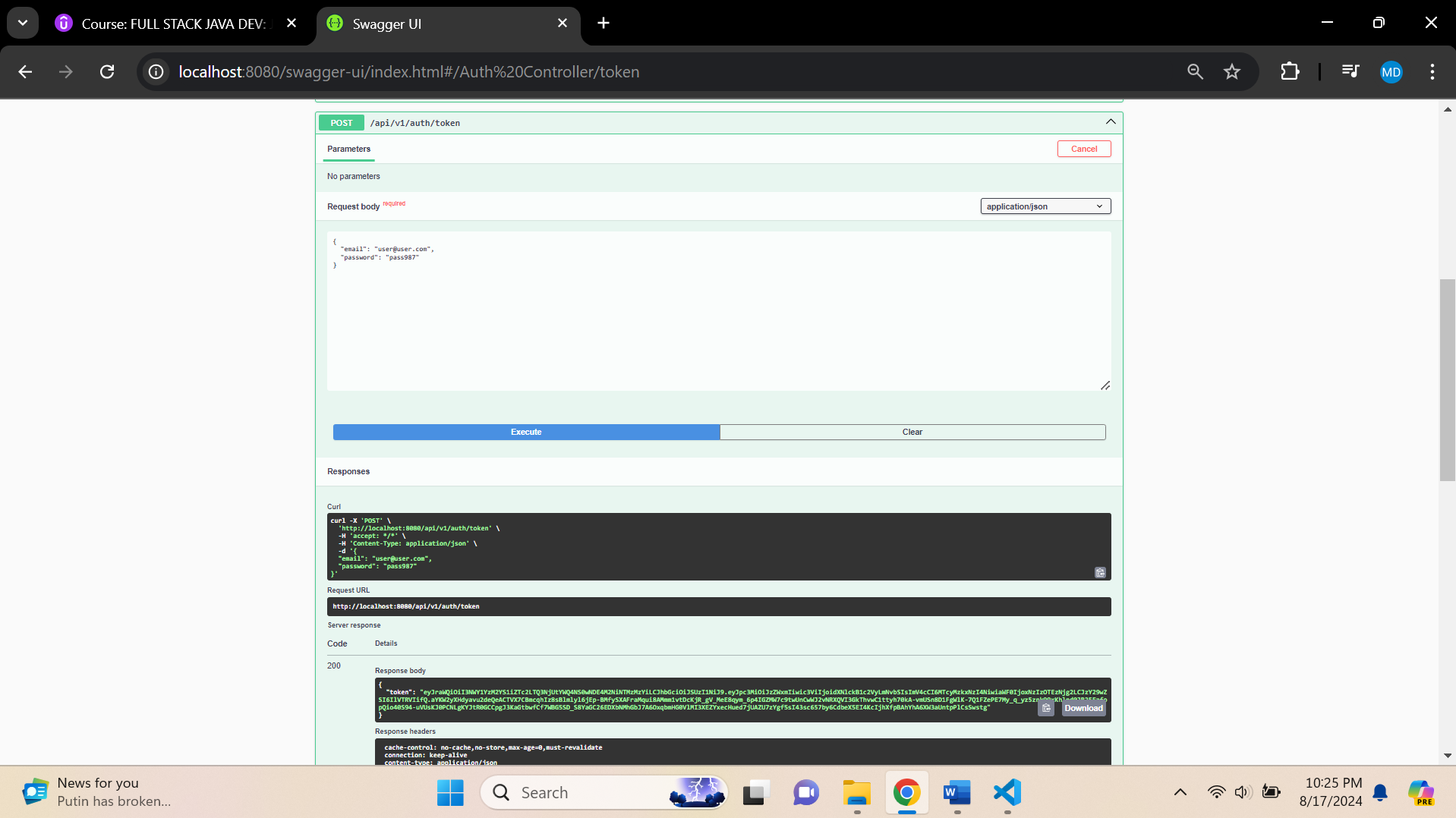
        });

        return ResponseEntity.ok(fileNamesWithSuccess);

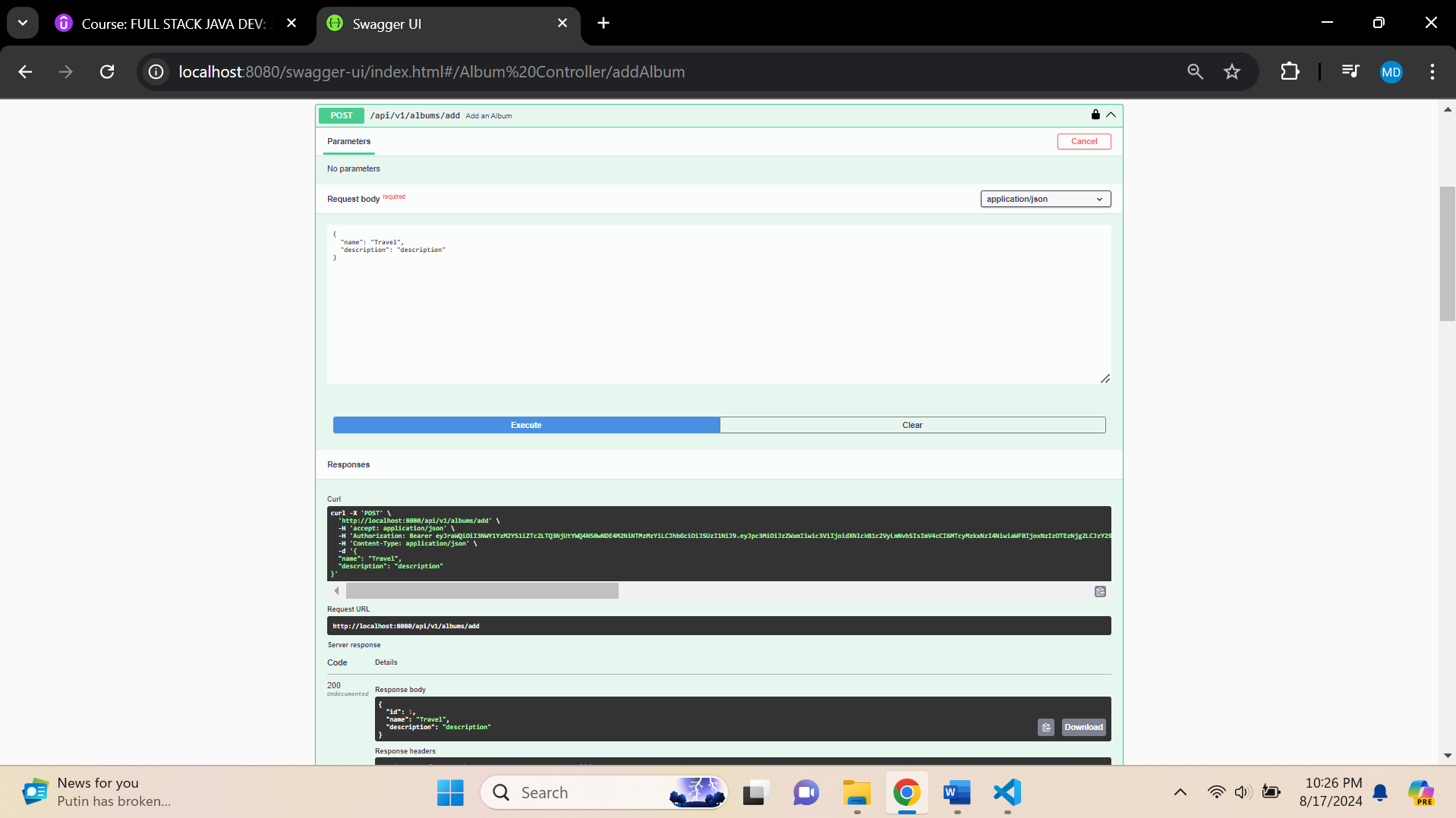
}

**Output**

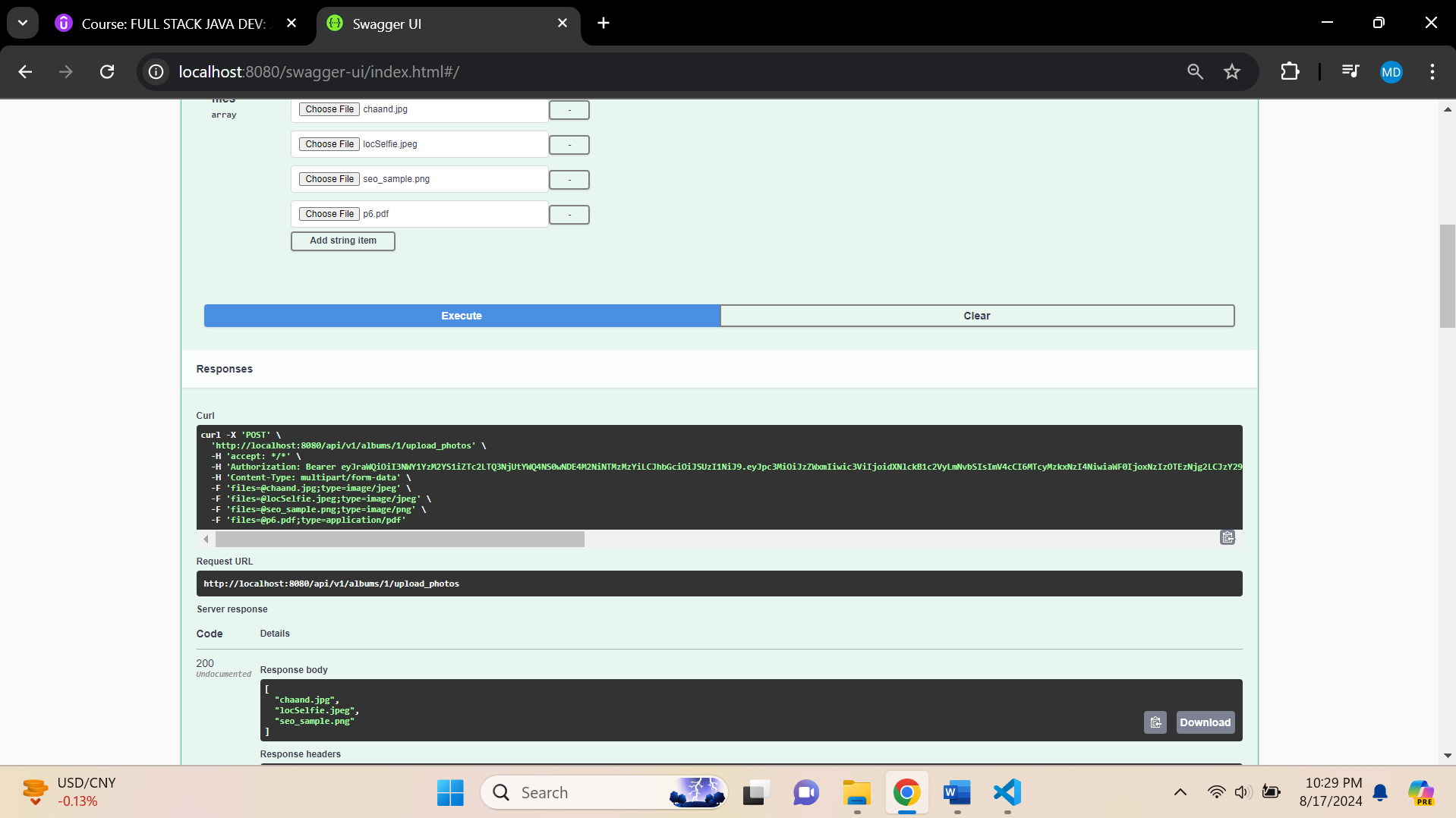
**Generate a token for the user and authorize**

****

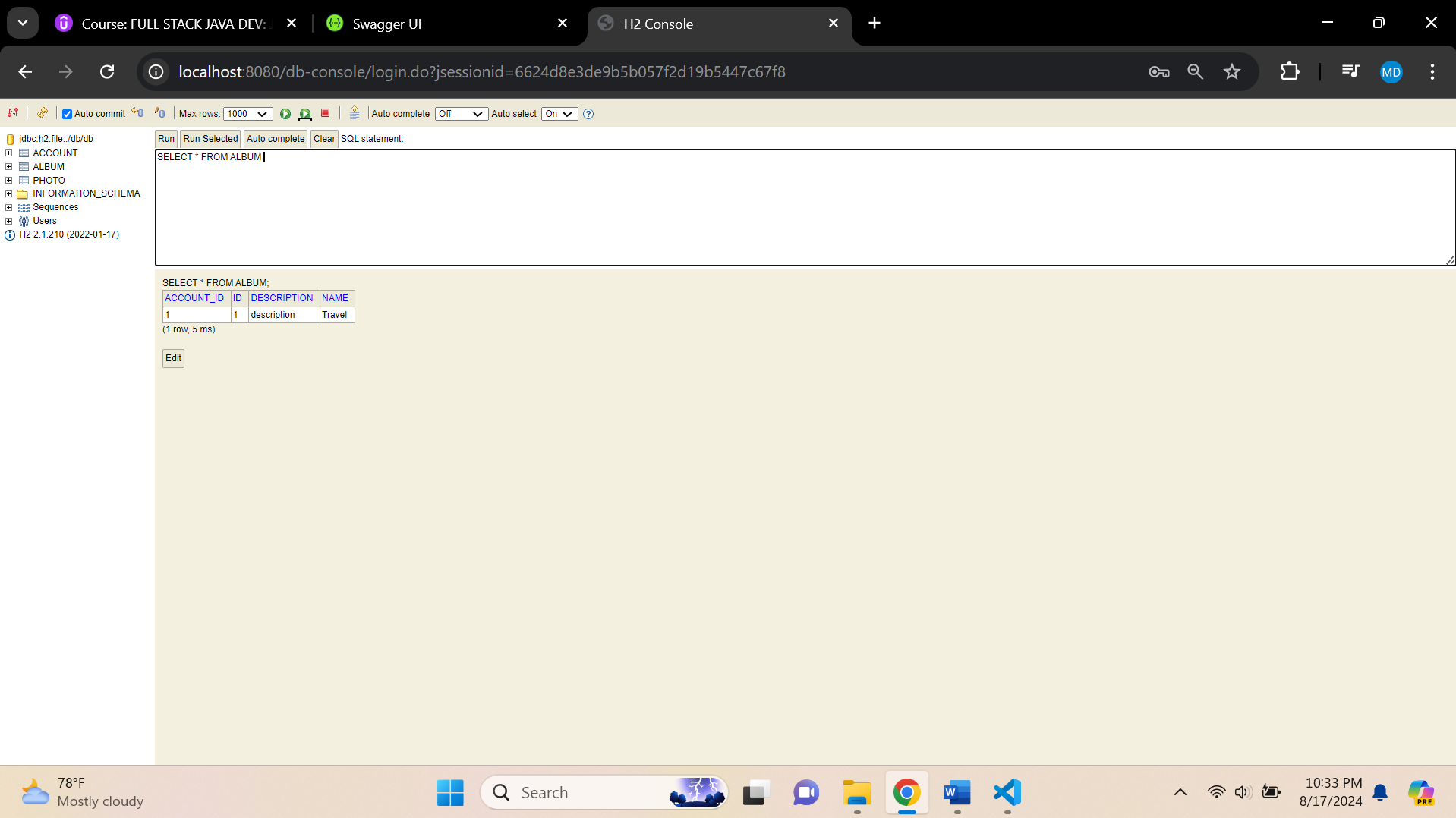
**Add album**

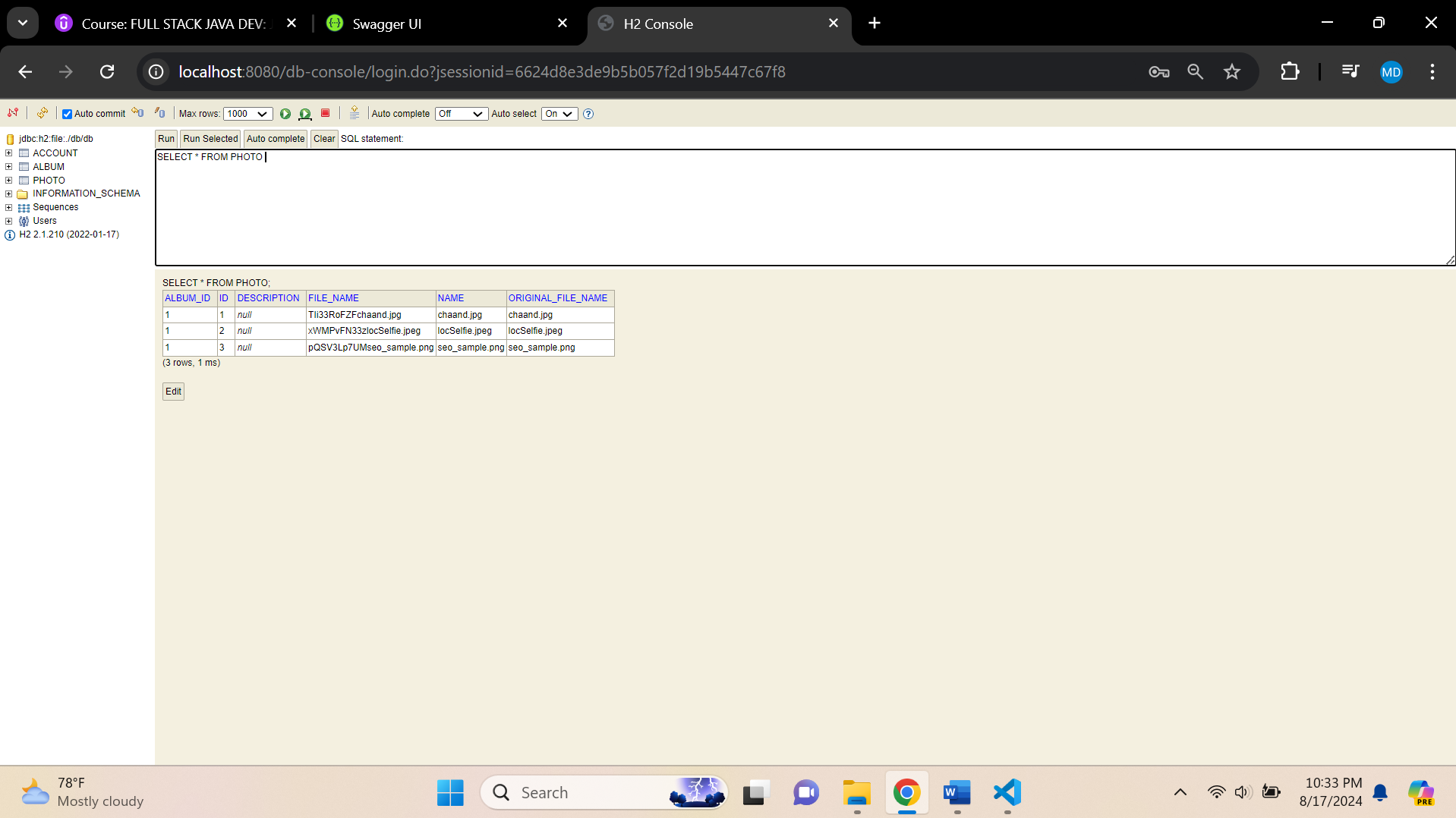
****

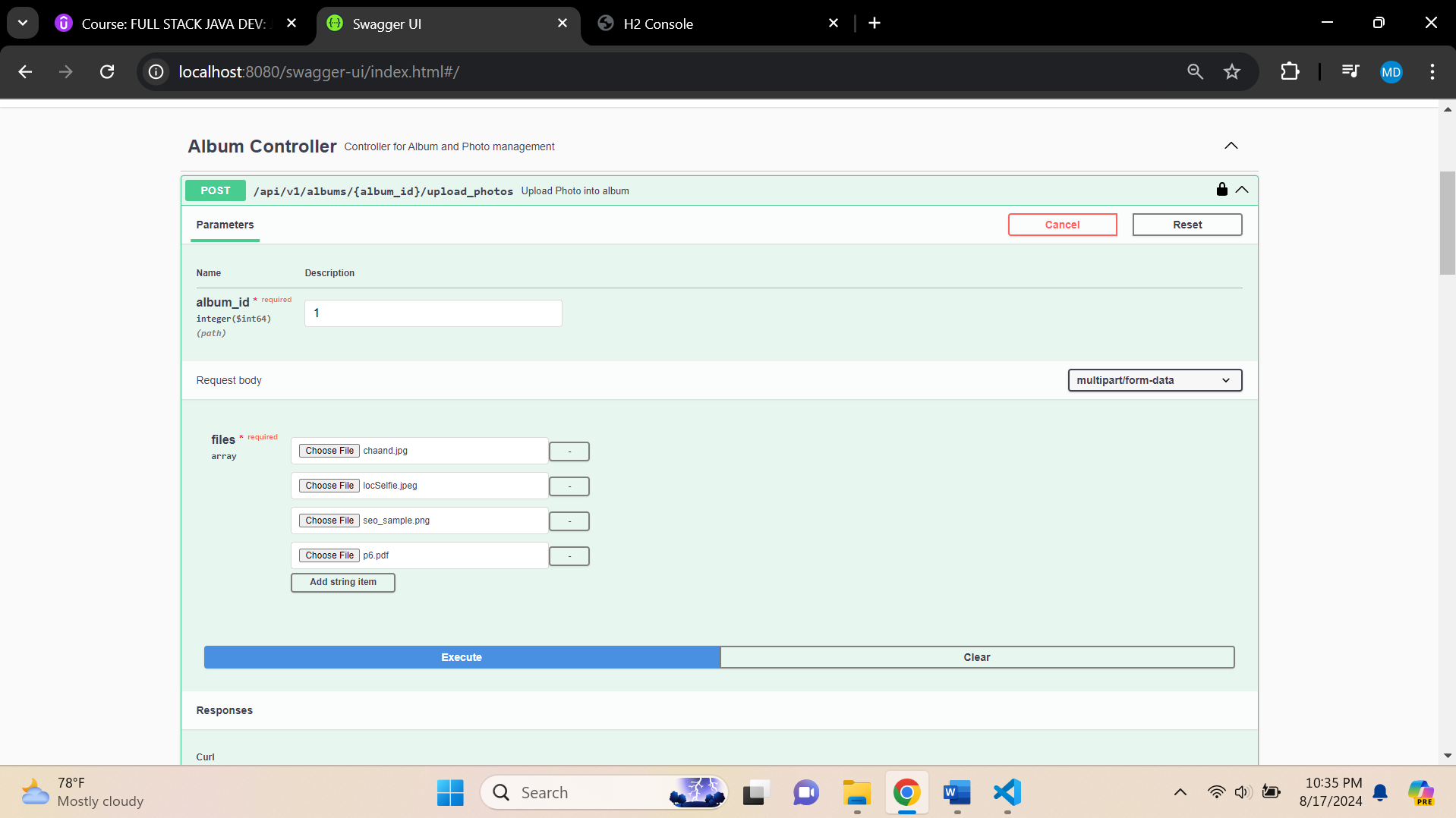
**After adding images to Api**

****

**Db Console Output**

****

****

****

**Small Little changes in the API**

**Updated AlbumController.java**

package org.studyeasy.SpringRestdemo.controller;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.nio.file.StandardCopyOption;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

import java.util.Optional;

import org.apache.commons.lang3.RandomStringUtils;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.security.core.Authentication;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestPart;

import org.springframework.web.bind.annotation.ResponseStatus;

import org.springframework.web.bind.annotation.RestController;

import org.springframework.web.multipart.MultipartFile;

import org.studyeasy.SpringRestdemo.model.Account;

import org.studyeasy.SpringRestdemo.model.Album;

import org.studyeasy.SpringRestdemo.model.Photo;

import org.studyeasy.SpringRestdemo.payload.auth.album.AlbumPayloadDTO;

import org.studyeasy.SpringRestdemo.payload.auth.album.AlbumViewDTO;

import org.studyeasy.SpringRestdemo.service.AccountService;

import org.studyeasy.SpringRestdemo.service.AlbumService;

import org.studyeasy.SpringRestdemo.service.PhotoService;

import org.studyeasy.SpringRestdemo.util.AppUtils.AppUtil;

import org.studyeasy.SpringRestdemo.util.constants.AlbumError;

import io.swagger.v3.oas.annotations.Operation;

import io.swagger.v3.oas.annotations.responses.ApiResponse;

import io.swagger.v3.oas.annotations.security.SecurityRequirement;

import io.swagger.v3.oas.annotations.tags.Tag;

import jakarta.validation.Valid;

import lombok.extern.slf4j.Slf4j;

@RestController

@RequestMapping("api/v1")

@Tag(name = "Album Controller", description = "Controller for Album and Photo management")

@Slf4j

public class AlbumController {

    @Autowired

    private AccountService accountService;

    @Autowired

    private AlbumService albumService;

    @Autowired

    private PhotoService photoService;

    @PostMapping(value = "/albums/add", consumes = "application/json", produces = "application/json")

    @ResponseStatus(HttpStatus.CREATED)

    @ApiResponse(responseCode = "400", description = "Please add valid name a description")

    @ApiResponse(responseCode = "201", description = "Account added")

    @Operation(summary = "Add an Album")

    @SecurityRequirement(name = "studyeasy-demo-api")

    public ResponseEntity<AlbumViewDTO> addAlbum(@Valid @RequestBody AlbumPayloadDTO albumPayloadDTO,

            Authentication authentication) {

        try {

            Album album = new Album();

            album.setName(albumPayloadDTO.getName());

            album.setDescription(albumPayloadDTO.getDescription());

            // Extract Account

            String email = authentication.getName();

            Optional<Account> optionalAccount = accountService.findByEmail(email);

            Account account = optionalAccount.get();

            album.setAccount(account);

            album = albumService.save(album);

            AlbumViewDTO albumViewDTO = new AlbumViewDTO(album.getId(), album.getName(), album.getDescription());

            return ResponseEntity.ok(albumViewDTO);

        } catch (Exception e) {

            log.debug(AlbumError.ADD\_ALBUM\_ERROR.toString() + ": " + e.getMessage());

            return ResponseEntity.status(HttpStatus.BAD\_REQUEST).body(null);

        }

    }

    // List all the albums based on the logged in users

    @GetMapping(value = "/albums", produces = "application/json")

    @ApiResponse(responseCode = "200", description = "List of albums")

    @ApiResponse(responseCode = "401", description = "Token missing")

    @ApiResponse(responseCode = "403", description = "Token Error")

    @Operation(summary = "List album api")

    @SecurityRequirement(name = "studyeasy-demo-api")

    public List<AlbumViewDTO> albums(Authentication authentication) {

        String email = authentication.getName();

        Optional<Account> optionalAccount = accountService.findByEmail(email);

        Account account = optionalAccount.get();

        List<AlbumViewDTO> albums = new ArrayList<>();

        for (Album album : albumService.findByAccount\_id(account.getId())) {

            albums.add(new AlbumViewDTO(album.getId(), album.getName(), album.getDescription()));

        }

        return albums;

    }

    @PostMapping(value = "/albums/{album\_id}/upload\_photos", consumes = { "multipart/form-data" })

    @Operation(summary = "Upload Photo into album")

    @SecurityRequirement(name = "studyeasy-demo-api")

    @ApiResponse(responseCode = "400", description = "Please check the payload of token")

    public ResponseEntity<List<String>> photos(@RequestPart(required = true) MultipartFile[] files,

            @PathVariable long album\_id, Authentication authentication) {

        String email = authentication.getName();

        Optional<Account> optionalAccount = accountService.findByEmail(email);

        Account account = optionalAccount.get();

        Optional<Album> optionalAlbum = albumService.findById(album\_id);

        Album album;

        if (optionalAlbum.isPresent()) {

            album = optionalAlbum.get();

            if (account.getId() != album.getAccount().getId()) {

                return ResponseEntity.status(HttpStatus.BAD\_GATEWAY).body(null);

            }

        } else {

            return ResponseEntity.status(HttpStatus.BAD\_GATEWAY).body(null);

        }

        List<String> fileNamesWithSuccess = new ArrayList<>();

        List<String> fileNamesWithError = new ArrayList<>();

        Arrays.asList(files).stream().forEach(file -> {

            // Checking the type of the file is correct or not

            String contentType = file.getContentType();

            if (contentType.equals("image/png")

                    || contentType.equals("image/jpg")

                    || contentType.equals("image/jpeg")) {

                fileNamesWithSuccess.add(file.getOriginalFilename());

                // When we are storing the file in the database, there is a possibility that

                // file name from the user

                // is repeating and if that happens, then file from the user will get repeated

                // and in server will

                // get replaced.

                // In order to prevent that, we need to create a random string.;

                int length = 10;

                boolean useLetters = true;

                boolean useNumbers = true;

                try {

                    String fileName = file.getOriginalFilename();

                    String generatedString = RandomStringUtils.random(length, useLetters, useNumbers);

                    String final\_photo\_name = generatedString + fileName;

                    String absolute\_fileLocation = AppUtil.get\_photo\_upload\_path(final\_photo\_name, album\_id);

                    Path path = Paths.get(absolute\_fileLocation);

                    Files.copy(file.getInputStream(), path, StandardCopyOption.REPLACE\_EXISTING);

                    Photo photo = new Photo();

                    photo.setName(fileName);

                    photo.setFileName(final\_photo\_name);

                    photo.setOriginalFileName(fileName);

                    photo.setAlbum(album);

                    photoService.save(photo);

                } catch (Exception e) {

                }

            }else{

                fileNamesWithError.add(file.getOriginalFilename());

            }

        });

        return ResponseEntity.ok(fileNamesWithSuccess);

    }

}

**Updated Account.java**

package org.studyeasy.SpringRestdemo.model;

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

import lombok.Getter;

import lombok.Setter;

import lombok.ToString;

@Entity

@Setter

@Getter

@ToString

public class Account {

    @Id

    @GeneratedValue(strategy = GenerationType.SEQUENCE)

    private long id;

    @Column(unique = true)

    private String email;

    private String password;

    private String Authorities;

}