package com.polstatstis.digiexam.entity;  
  
import jakarta.persistence.\*;  
import lombok.\*;  
  
import java.time.LocalDateTime;  
import java.util.List;  
  
@Data  
@NoArgsConstructor  
@AllArgsConstructor  
@Builder  
@Entity  
@Table(name = "exams")  
public class Exam {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long id;  
  
 @Column(nullable = false)  
 private String title;  
  
 @Column(nullable = false)  
 private String description;  
  
 @Column(nullable = false)  
 private LocalDateTime date;  
  
 @ElementCollection  
 @CollectionTable(name = "exam\_questions", joinColumns = @JoinColumn(name = "exam\_id"))  
 @Column(name = "question\_id")  
 private List<Long> questions;  
  
 @ManyToOne  
 @JoinColumn(name = "created\_by", nullable = false)  
 private User createdBy;  
}

package com.polstatstis.digiexam.entity;  
  
import jakarta.persistence.\*;  
import lombok.\*;  
  
@Data  
@NoArgsConstructor  
@AllArgsConstructor  
@Builder  
@Entity  
@Table(name = "exam\_results")  
public class ExamResult {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long id;  
  
 @ManyToOne  
 @JoinColumn(name = "user\_id", nullable = false)  
 private User user;  
  
 @ManyToOne  
 @JoinColumn(name = "exam\_id", nullable = false)  
 private Exam exam;  
  
 @Column(nullable = false)  
 private Integer score;  
}

package com.polstatstis.digiexam.entity;  
  
import jakarta.persistence.\*;  
import lombok.\*;  
  
import java.util.List;  
  
@Data  
@NoArgsConstructor  
@AllArgsConstructor  
@Builder  
@Entity  
@Table(name = "questions")  
public class Question {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long id;  
  
 @Column(nullable = false)  
 private String text;  
  
 @ElementCollection  
 @CollectionTable(  
 name = "question\_options",  
 joinColumns = @JoinColumn(name = "question\_id")  
 )  
 @Column(name = "option\_text") // Diubah dari 'option' menjadi 'option\_text'  
 private List<String> options;  
  
 @Column(nullable = false)  
 private String answer;  
  
 @ManyToOne  
 @JoinColumn(name = "created\_by", nullable = false)  
 private User createdBy;  
}

package com.polstatstis.digiexam.entity;  
  
import jakarta.persistence.\*;  
import lombok.\*;  
  
@Data  
@NoArgsConstructor  
@AllArgsConstructor  
@Builder  
@Entity  
@Table(name = "users")  
public class User {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long id;  
  
 @Column(nullable = false, unique = true)  
 private String email;  
  
 @Column(nullable = false)  
 private String password;  
  
 @Column(nullable = false)  
 private String name;  
  
 @Enumerated(EnumType.STRING)  
 @Column(nullable = false)  
 private Role role;  
  
 public enum Role {  
 ADMIN, USER  
 }  
}

package com.polstatstis.digiexam.repository;  
  
import com.polstatstis.digiexam.entity.Exam;  
import org.springframework.data.jpa.repository.JpaRepository;  
import io.swagger.v3.oas.annotations.Operation;  
import io.swagger.v3.oas.annotations.media.Content;  
import io.swagger.v3.oas.annotations.media.Schema;  
import io.swagger.v3.oas.annotations.responses.ApiResponse;  
import io.swagger.v3.oas.annotations.responses.ApiResponses;  
  
import java.util.List;  
  
public interface ExamRepository extends JpaRepository<Exam, Long> {  
 @Operation(summary = "Find exams created by a user")  
 @ApiResponses(value = {  
 @ApiResponse(responseCode = "200", description = "Found exams created by the user",  
 content = {@Content(mediaType = "application/json",  
 schema = @Schema(implementation = Exam.class))}),  
 @ApiResponse(responseCode = "404", description = "No exams found",  
 content = @Content)})  
  
 List<Exam> findByCreatedById(Long userId);  
}

package com.polstatstis.digiexam.repository;  
  
import com.polstatstis.digiexam.entity.ExamResult;  
import org.springframework.data.jpa.repository.JpaRepository;  
import io.swagger.v3.oas.annotations.Operation;  
import io.swagger.v3.oas.annotations.media.Content;  
import io.swagger.v3.oas.annotations.media.Schema;  
import io.swagger.v3.oas.annotations.responses.ApiResponse;  
import io.swagger.v3.oas.annotations.responses.ApiResponses;  
  
import java.util.List;  
  
public interface ExamResultRepository extends JpaRepository<ExamResult, Long> {  
 @Operation(summary = "Find exam results by user ID")  
 @ApiResponses(value = {  
 @ApiResponse(responseCode = "200", description = "Found exam results",  
 content = { @Content(mediaType = "application/json",  
 schema = @Schema(implementation = ExamResult.class)) }),  
 @ApiResponse(responseCode = "404", description = "Exam results not found",  
 content = @Content) })  
  
 List<ExamResult> findByUserId(Long userId);  
  
 @Operation(summary = "Find exam results by exam ID")  
 @ApiResponses(value = {  
 @ApiResponse(responseCode = "200", description = "Found exam results",  
 content = { @Content(mediaType = "application/json",  
 schema = @Schema(implementation = ExamResult.class)) }),  
 @ApiResponse(responseCode = "404", description = "Exam results not found",  
 content = @Content) })  
  
 List<ExamResult> findByExamId(Long examId);  
}

package com.polstatstis.digiexam.repository;  
  
import com.polstatstis.digiexam.entity.Question;  
import org.springframework.data.jpa.repository.JpaRepository;  
import io.swagger.v3.oas.annotations.Operation;  
import io.swagger.v3.oas.annotations.media.Content;  
import io.swagger.v3.oas.annotations.media.Schema;  
import io.swagger.v3.oas.annotations.responses.ApiResponse;  
import io.swagger.v3.oas.annotations.responses.ApiResponses;  
  
import java.util.List;  
  
public interface QuestionRepository extends JpaRepository<Question, Long> {  
 @Operation(summary = "Find questions by user ID")  
 @ApiResponses(value = {  
 @ApiResponse(responseCode = "200", description = "Found questions",  
 content = {@Content(mediaType = "application/json",  
 schema = @Schema(implementation = Question.class))}),  
 @ApiResponse(responseCode = "404", description = "Questions not found",  
 content = @Content)})  
  
 List<Question> findByCreatedById(Long userId);  
}

package com.polstatstis.digiexam.repository;  
  
import com.polstatstis.digiexam.entity.User;  
import org.springframework.data.jpa.repository.JpaRepository;  
import io.swagger.v3.oas.annotations.Operation;  
import io.swagger.v3.oas.annotations.media.Content;  
import io.swagger.v3.oas.annotations.media.Schema;  
import io.swagger.v3.oas.annotations.responses.ApiResponse;  
import io.swagger.v3.oas.annotations.responses.ApiResponses;  
  
import java.util.Optional;  
  
public interface UserRepository extends JpaRepository<User, Long> {  
  
 @Operation(summary = "Find user by email", description = "Returns a single user by email", tags = { "user" })  
 @ApiResponses(value = {  
 @ApiResponse(responseCode = "200", description = "successful operation", content = @Content(schema = @Schema(implementation = User.class))),  
 @ApiResponse(responseCode = "404", description = "User not found") })  
  
 Optional<User> findByEmail(String email);  
}

package com.polstatstis.digiexam.dto;  
  
import lombok.\*;  
  
@Data  
@NoArgsConstructor  
@AllArgsConstructor  
@Builder  
public class UserDTO {  
  
 private Long id;  
 private String email;  
 private String name;  
 private String role;  
}

package com.polstatstis.digiexam.dto;  
  
import lombok.\*;  
  
import java.time.LocalDateTime;  
import java.util.List;  
  
@Data  
@NoArgsConstructor  
@AllArgsConstructor  
@Builder  
public class ExamDTO {  
  
 private Long id;  
 private String title;  
 private String description;  
 private LocalDateTime date;  
 private List<Long> questionIds;  
 private Long createdById;  
}

package com.polstatstis.digiexam.mapper;  
  
import com.polstatstis.digiexam.dto.UserDTO;  
import com.polstatstis.digiexam.entity.User;  
  
public class UserMapper {  
  
 public static UserDTO toDTO(User user) {  
 return UserDTO.builder()  
 .id(user.getId())  
 .email(user.getEmail())  
 .name(user.getName())  
 .role(user.getRole().name())  
 .build();  
 }  
  
 public static User toEntity(UserDTO userDTO) {  
 return User.builder()  
 .id(userDTO.getId())  
 .email(userDTO.getEmail())  
 .name(userDTO.getName())  
 .role(User.Role.valueOf(userDTO.getRole()))  
 .build();  
 }  
}

package com.polstatstis.digiexam.mapper;  
  
import com.polstatstis.digiexam.dto.ExamDTO;  
import com.polstatstis.digiexam.entity.Exam;  
  
public class ExamMapper {  
  
 public static ExamDTO toDTO(Exam exam) {  
 return ExamDTO.builder()  
 .id(exam.getId())  
 .title(exam.getTitle())  
 .description(exam.getDescription())  
 .date(exam.getDate())  
 .questionIds(exam.getQuestions())  
 .createdById(exam.getCreatedBy().getId())  
 .build();  
 }  
  
 public static Exam toEntity(ExamDTO examDTO) {  
 return Exam.builder()  
 .id(examDTO.getId())  
 .title(examDTO.getTitle())  
 .description(examDTO.getDescription())  
 .date(examDTO.getDate())  
 .questions(examDTO.getQuestionIds())  
 .build();  
 }  
}

package com.polstatstis.digiexam.service;  
  
import com.polstatstis.digiexam.dto.UserLoginDTO;  
import com.polstatstis.digiexam.dto.UserRegistrationDTO;  
import com.polstatstis.digiexam.dto.UserDTO;  
import com.polstatstis.digiexam.entity.User;  
import com.polstatstis.digiexam.exception.UserNotFoundException;  
import com.polstatstis.digiexam.mapper.UserMapper;  
import com.polstatstis.digiexam.repository.UserRepository;  
import com.polstatstis.digiexam.security.JwtUtil;  
import lombok.RequiredArgsConstructor;  
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
import org.springframework.security.crypto.password.PasswordEncoder;  
import org.springframework.stereotype.Service;  
import com.polstatstis.digiexam.config.SecurityConfig;  
  
  
@Service  
@RequiredArgsConstructor  
public class AuthService {  
  
 private final UserRepository userRepository;  
 private final PasswordEncoder passwordEncoder;  
 private final JwtUtil jwtUtil;  
  
 public UserDTO register(UserRegistrationDTO registrationDTO) {  
 User user = User.builder()  
 .email(registrationDTO.getEmail())  
 .password(passwordEncoder.encode(registrationDTO.getPassword()))  
 .name(registrationDTO.getName())  
 .role(User.Role.USER)  
 .build();  
 userRepository.save(user);  
 return UserMapper.toDTO(user);  
 }  
  
 public String login(UserLoginDTO loginDTO) {  
 User user = userRepository.findByEmail(loginDTO.getEmail())  
 .orElseThrow(() -> new UserNotFoundException("User not found"));  
 if (!passwordEncoder.matches(loginDTO.getPassword(), user.getPassword())) {  
 throw new IllegalArgumentException("Invalid password");  
 }  
 return jwtUtil.generateToken(user);  
 }  
}

package com.polstatstis.digiexam.service;  
  
import com.polstatstis.digiexam.dto.UserDTO;  
import com.polstatstis.digiexam.entity.User;  
import com.polstatstis.digiexam.mapper.UserMapper;  
import com.polstatstis.digiexam.repository.UserRepository;  
import com.polstatstis.digiexam.exception.UserNotFoundException;  
import lombok.RequiredArgsConstructor;  
import org.springframework.stereotype.Service;  
  
@Service  
@RequiredArgsConstructor  
public class UserService {  
  
 private final UserRepository userRepository;  
  
 public UserDTO getUserProfile(Long userId) {  
 User user = userRepository.findById(userId)  
 .orElseThrow(() -> new UserNotFoundException("User not found"));  
 return UserMapper.toDTO(user);  
 }  
  
 public UserDTO updateUserProfile(Long userId, UserDTO userDTO) {  
 User user = userRepository.findById(userId)  
 .orElseThrow(() -> new UserNotFoundException("User not found"));  
 user.setName(userDTO.getName());  
 user.setEmail(userDTO.getEmail());  
 userRepository.save(user);  
 return UserMapper.toDTO(user);  
 }  
  
 public void deleteUser(Long userId) {  
 if (!userRepository.existsById(userId)) {  
 throw new UserNotFoundException("User not found");  
 }  
 userRepository.deleteById(userId);  
 }  
}

package com.polstatstis.digiexam.service;  
  
import com.polstatstis.digiexam.dto.ExamDTO;  
import com.polstatstis.digiexam.entity.Exam;  
import com.polstatstis.digiexam.mapper.ExamMapper;  
import com.polstatstis.digiexam.repository.ExamRepository;  
import com.polstatstis.digiexam.exception.ExamNotFoundException;  
import lombok.RequiredArgsConstructor;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
import java.util.stream.Collectors;  
  
@Service  
@RequiredArgsConstructor  
public class ExamService {  
  
 private final ExamRepository examRepository;  
  
 public ExamDTO createExam(ExamDTO examDTO) {  
 Exam exam = ExamMapper.toEntity(examDTO);  
 examRepository.save(exam);  
 return ExamMapper.toDTO(exam);  
 }  
  
 public List<ExamDTO> getAllExams() {  
 return examRepository.findAll().stream()  
 .map(ExamMapper::toDTO)  
 .collect(Collectors.toList());  
 }  
  
 public ExamDTO getExamById(Long examId) {  
 Exam exam = examRepository.findById(examId)  
 .orElseThrow(() -> new ExamNotFoundException("Exam not found"));  
 return ExamMapper.toDTO(exam);  
 }  
  
 public void deleteExam(Long examId) {  
 if (!examRepository.existsById(examId)) {  
 throw new ExamNotFoundException("Exam not found");  
 }  
 examRepository.deleteById(examId);  
 }  
}

package com.polstatstis.digiexam.config;  
  
import com.polstatstis.digiexam.security.JwtAuthenticationFilter;  
import lombok.RequiredArgsConstructor;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.authentication.AuthenticationManager;  
import org.springframework.security.authentication.AuthenticationProvider;  
import org.springframework.security.authentication.dao.DaoAuthenticationProvider;  
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;  
import org.springframework.security.config.http.SessionCreationPolicy;  
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
import org.springframework.security.crypto.password.PasswordEncoder;  
import com.polstatstis.digiexam.service.UserDetailsServiceImpl;  
import org.springframework.security.web.SecurityFilterChain;  
import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;  
  
@Configuration  
@EnableWebSecurity  
@RequiredArgsConstructor  
public class SecurityConfig {  
  
 private final JwtAuthenticationFilter jwtAuthenticationFilter;  
 private final UserDetailsServiceImpl userDetailsService;  
  
 @Bean  
 public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {  
 http  
 .csrf().disable()  
 .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS)  
 .and()  
 .authorizeHttpRequests()  
 .requestMatchers("/api/auth/\*\*", "/docs/\*\*").permitAll()  
 .anyRequest().authenticated()  
 .and()  
 .authenticationProvider(authenticationProvider())  
 .addFilterBefore(jwtAuthenticationFilter, UsernamePasswordAuthenticationFilter.class);  
  
 return http.build();  
 }  
  
 @Bean  
 public AuthenticationProvider authenticationProvider() {  
 DaoAuthenticationProvider provider = new DaoAuthenticationProvider();  
 provider.setUserDetailsService(userDetailsService);  
 provider.setPasswordEncoder(passwordEncoder());  
 return provider;  
 }  
  
 @Bean  
 public PasswordEncoder passwordEncoder() {  
 return new BCryptPasswordEncoder();  
 }  
  
 @Bean  
 public AuthenticationManager authenticationManager(HttpSecurity http) throws Exception {  
 AuthenticationManagerBuilder builder = http.getSharedObject(AuthenticationManagerBuilder.class);  
 builder.userDetailsService(userDetailsService).passwordEncoder(passwordEncoder());  
 return builder.build();  
 }  
}

package com.polstatstis.digiexam.security;  
  
import com.polstatstis.digiexam.entity.User;  
import io.jsonwebtoken.Jwts;  
import io.jsonwebtoken.SignatureAlgorithm;  
import lombok.RequiredArgsConstructor;  
import org.springframework.stereotype.Component;  
  
import java.util.Date;  
  
@Component  
@RequiredArgsConstructor  
public class JwtUtil {  
  
 private final String secretKey = "yourSecretKey"; // use a secure secret key  
  
 public String generateToken(User user) {  
 return Jwts.builder()  
 .setSubject(user.getEmail())  
 .claim("role", user.getRole())  
 .setIssuedAt(new Date())  
 .setExpiration(new Date(System.currentTimeMillis() + 1000 \* 60 \* 60 \* 10)) // 10 hours  
 .signWith(SignatureAlgorithm.HS256, secretKey)  
 .compact();  
 }  
  
 public boolean validateToken(String token, User user) {  
 String username = extractUsername(token);  
 return (username.equals(user.getEmail()) && !isTokenExpired(token));  
 }  
  
 public String extractUsername(String token) {  
 return Jwts.parser()  
 .setSigningKey(secretKey)  
 .parseClaimsJws(token)  
 .getBody()  
 .getSubject();  
 }  
  
 private boolean isTokenExpired(String token) {  
 return extractExpiration(token).before(new Date());  
 }  
  
 private Date extractExpiration(String token) {  
 return Jwts.parser()  
 .setSigningKey(secretKey)  
 .parseClaimsJws(token)  
 .getBody()  
 .getExpiration();  
 }  
}

package com.polstatstis.digiexam.controller;  
  
import com.polstatstis.digiexam.dto.UserLoginDTO;  
import com.polstatstis.digiexam.dto.UserRegistrationDTO;  
import com.polstatstis.digiexam.dto.UserDTO;  
import com.polstatstis.digiexam.service.AuthService;  
import io.swagger.v3.oas.annotations.Operation;  
import io.swagger.v3.oas.annotations.media.Content;  
import io.swagger.v3.oas.annotations.media.Schema;  
import io.swagger.v3.oas.annotations.responses.ApiResponse;  
import io.swagger.v3.oas.annotations.responses.ApiResponses;  
import lombok.RequiredArgsConstructor;  
import org.springframework.data.domain.Page;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
  
@RestController  
@RequestMapping("/api/auth")  
@RequiredArgsConstructor  
public class AuthController {  
  
 private final AuthService authService;  
  
 @Operation(summary = "User registration.")  
 @ApiResponses(value = {  
 @ApiResponse(responseCode = "200", description = "User registered successfully", content = {  
 @Content(mediaType = "application/json", schema = @Schema(implementation = Page.class))}), @ApiResponse(responseCode = "400", description = "Invalid input", content = @Content)})  
  
 @PostMapping("/register")  
 public ResponseEntity<UserDTO> register(@RequestBody UserRegistrationDTO registrationDTO) {  
 UserDTO user = authService.register(registrationDTO);  
 return ResponseEntity.ok(user);  
 }  
  
 @Operation(summary = "User login to get access token.")  
 @ApiResponses(value = {  
 @ApiResponse(responseCode = "200", description = "Email and access token", content = {  
 @Content(mediaType = "application/json", schema = @Schema(implementation = Page.class))}), @ApiResponse(responseCode = "401", description = "Invalid credentials", content = @Content)})  
  
 @PostMapping("/login")  
 public ResponseEntity<String> login(@RequestBody UserLoginDTO loginDTO) {  
 String token = authService.login(loginDTO);  
 return ResponseEntity.ok(token);  
 }  
}

package com.polstatstis.digiexam.controller;  
  
import com.polstatstis.digiexam.dto.UserLoginDTO;  
import com.polstatstis.digiexam.dto.UserRegistrationDTO;  
import com.polstatstis.digiexam.dto.UserDTO;  
import com.polstatstis.digiexam.service.AuthService;  
import io.swagger.v3.oas.annotations.Operation;  
import io.swagger.v3.oas.annotations.media.Content;  
import io.swagger.v3.oas.annotations.media.Schema;  
import io.swagger.v3.oas.annotations.responses.ApiResponse;  
import io.swagger.v3.oas.annotations.responses.ApiResponses;  
import lombok.RequiredArgsConstructor;  
import org.springframework.data.domain.Page;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
  
@RestController  
@RequestMapping("/api/auth")  
@RequiredArgsConstructor  
public class AuthController {  
  
 private final AuthService authService;  
  
 @Operation(summary = "User registration.")  
 @ApiResponses(value = {  
 @ApiResponse(responseCode = "200", description = "User registered successfully", content = {  
 @Content(mediaType = "application/json", schema = @Schema(implementation = Page.class))}), @ApiResponse(responseCode = "400", description = "Invalid input", content = @Content)})  
  
 @PostMapping("/register")  
 public ResponseEntity<UserDTO> register(@RequestBody UserRegistrationDTO registrationDTO) {  
 UserDTO user = authService.register(registrationDTO);  
 return ResponseEntity.ok(user);  
 }  
  
 @Operation(summary = "User login to get access token.")  
 @ApiResponses(value = {  
 @ApiResponse(responseCode = "200", description = "Email and access token", content = {  
 @Content(mediaType = "application/json", schema = @Schema(implementation = Page.class))}), @ApiResponse(responseCode = "401", description = "Invalid credentials", content = @Content)})  
  
 @PostMapping("/login")  
 public ResponseEntity<String> login(@RequestBody UserLoginDTO loginDTO) {  
 String token = authService.login(loginDTO);  
 return ResponseEntity.ok(token);  
 }  
}

package com.polstatstis.digiexam.controller;  
  
import com.polstatstis.digiexam.dto.UserDTO;  
import com.polstatstis.digiexam.service.UserService;  
import lombok.RequiredArgsConstructor;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
import io.swagger.v3.oas.annotations.Operation;  
import io.swagger.v3.oas.annotations.media.Content;  
import io.swagger.v3.oas.annotations.media.Schema;  
import io.swagger.v3.oas.annotations.responses.ApiResponse;  
import io.swagger.v3.oas.annotations.responses.ApiResponses;  
  
@RestController  
@RequestMapping("/api/users")  
@RequiredArgsConstructor  
public class UserController {  
  
 private final UserService userService;  
  
 @Operation(summary = "Get user by id")  
 @ApiResponses(value = {  
 @ApiResponse(responseCode = "200", description = "Found the user",  
 content = { @Content(mediaType = "application/json",  
 schema = @Schema(implementation = UserDTO.class)) }),  
 @ApiResponse(responseCode = "404", description = "User not found",  
 content = @Content) })  
  
 @GetMapping("/{id}")  
 public ResponseEntity<UserDTO> getUserProfile(@PathVariable Long id) {  
 UserDTO user = userService.getUserProfile(id);  
 return ResponseEntity.ok(user);  
 }  
  
 @Operation(summary = "Update user profile")  
 @ApiResponses(value = {  
 @ApiResponse(responseCode = "200", description = "User profile updated",  
 content = { @Content(mediaType = "application/json",  
 schema = @Schema(implementation = UserDTO.class)) }),  
 @ApiResponse(responseCode = "404", description = "User not found",  
 content = @Content) })  
  
 @PutMapping("/{id}")  
 public ResponseEntity<UserDTO> updateUserProfile(@PathVariable Long id, @RequestBody UserDTO userDTO) {  
 UserDTO updatedUser = userService.updateUserProfile(id, userDTO);  
 return ResponseEntity.ok(updatedUser);  
 }  
  
 @Operation(summary = "Delete user")  
 @ApiResponses(value = {  
 @ApiResponse(responseCode = "204", description = "User deleted",  
 content = @Content),  
 @ApiResponse(responseCode = "404", description = "User not found",  
 content = @Content) })  
  
 @DeleteMapping("/{id}")  
 public ResponseEntity<Void> deleteUser(@PathVariable Long id) {  
 userService.deleteUser(id);  
 return ResponseEntity.noContent().build();  
 }  
}

package com.polstatstis.digiexam.controller;  
  
import com.polstatstis.digiexam.dto.ExamDTO;  
import com.polstatstis.digiexam.service.ExamService;  
import io.swagger.v3.oas.annotations.Operation;  
import lombok.RequiredArgsConstructor;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
import io.swagger.v3.oas.annotations.Operation;  
import io.swagger.v3.oas.annotations.media.Content;  
import io.swagger.v3.oas.annotations.media.Schema;  
import io.swagger.v3.oas.annotations.responses.ApiResponse;  
import io.swagger.v3.oas.annotations.responses.ApiResponses;  
  
import java.util.List;  
  
@RestController  
@RequestMapping("/api/exams")  
@RequiredArgsConstructor  
public class ExamController {  
  
 private final ExamService examService;  
  
 @Operation(summary = "Create an exam")  
 @ApiResponses(value = {  
 @ApiResponse(responseCode = "200", description = "Exam created successfully"),  
 @ApiResponse(responseCode = "400", description = "Invalid input"),  
 @ApiResponse(responseCode = "500", description = "Server error")  
 })  
  
 @PostMapping  
 public ResponseEntity<ExamDTO> createExam(@RequestBody ExamDTO examDTO) {  
 ExamDTO createdExam = examService.createExam(examDTO);  
 return ResponseEntity.ok(createdExam);  
 }  
  
 @Operation(summary = "Get all exams")  
 @ApiResponses(value = {  
 @ApiResponse(responseCode = "200", description = "Exams retrieved successfully"),  
 @ApiResponse(responseCode = "500", description = "Server error")  
 })  
  
 @GetMapping  
 public ResponseEntity<List<ExamDTO>> getAllExams() {  
 List<ExamDTO> exams = examService.getAllExams();  
 return ResponseEntity.ok(exams);  
 }  
  
 @Operation(summary = "Get exam by ID")  
 @ApiResponses(value = {  
 @ApiResponse(responseCode = "200", description = "Exam retrieved successfully"),  
 @ApiResponse(responseCode = "404", description = "Exam not found"),  
 @ApiResponse(responseCode = "500", description = "Server error")  
 })  
  
 @GetMapping("/{id}")  
 public ResponseEntity<ExamDTO> getExamById(@PathVariable Long id) {  
 ExamDTO exam = examService.getExamById(id);  
 return ResponseEntity.ok(exam);  
 }  
  
 @Operation(summary = "Update an exam")  
 @ApiResponses(value = {  
 @ApiResponse(responseCode = "200", description = "Exam updated successfully"),  
 @ApiResponse(responseCode = "404", description = "Exam not found"),  
 @ApiResponse(responseCode = "500", description = "Server error")  
 })  
  
 @DeleteMapping("/{id}")  
 public ResponseEntity<Void> deleteExam(@PathVariable Long id) {  
 examService.deleteExam(id);  
 return ResponseEntity.noContent().build();  
 }  
}

package com.polstatstis.digiexam.exception;  
  
public class JwtAuthenticationException extends RuntimeException {  
 public JwtAuthenticationException(String message) {  
 super(message);  
 }  
}

package com.polstatstis.digiexam;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class DigiexamApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.run(DigiexamApplication.class, args);  
 }  
  
}